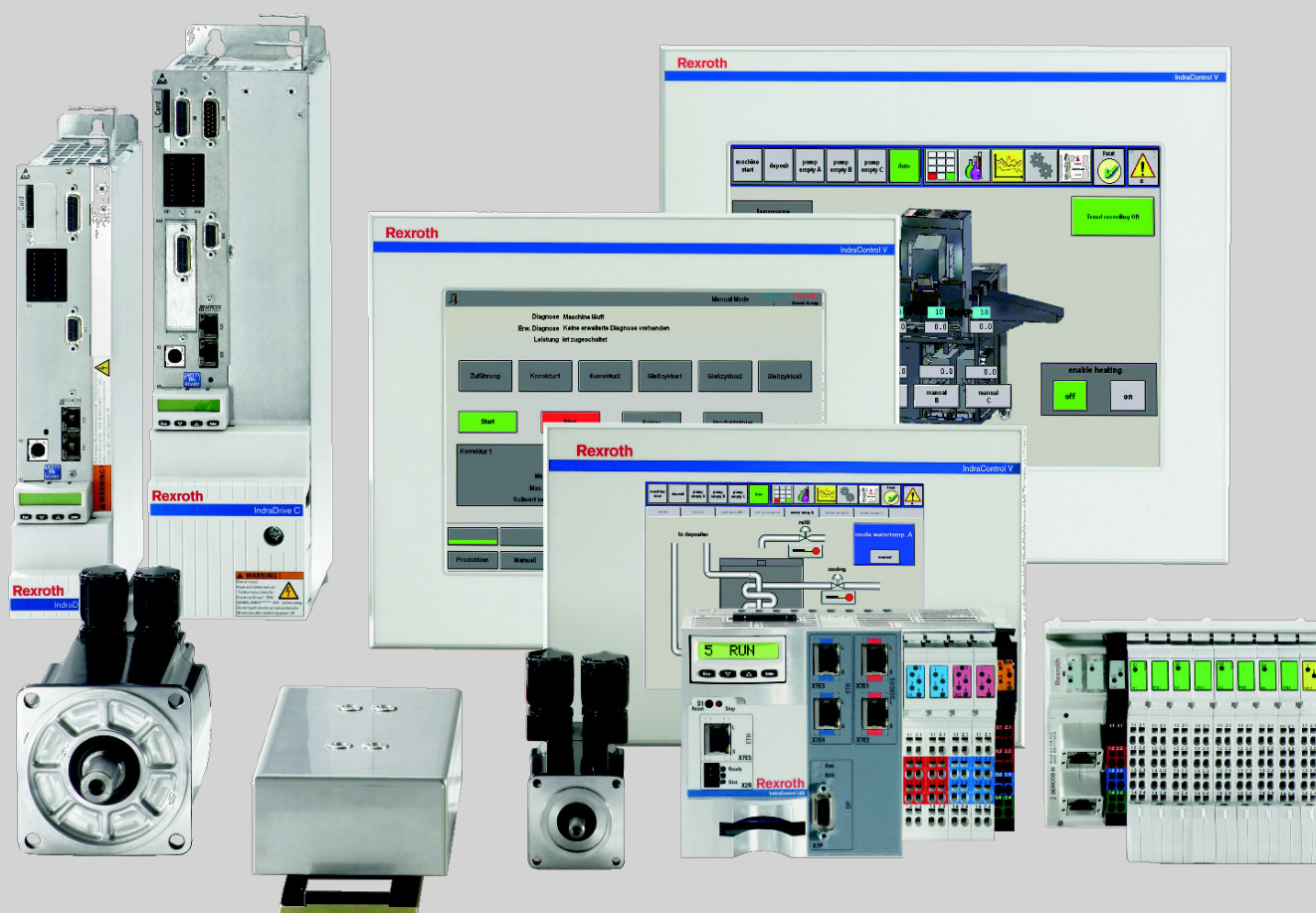


IndraWorks 15VRS

SPS-Programmiersystem IndraLogic 2G

Anwendungsbeschreibung
R911396134

Ausgabe 01



Titel	IndraWorks 15VRS SPS-Programmiersystem IndraLogic 2G
Art der Dokumentation	Anwendungsbeschreibung
Dokumentations-Type	DOK-IWORKS-IL2GPRO*V15-AP01-DE-P
Interner Ablagevermerk	RS-23254ab1d57f74b60a347e862c73003a-1-de-DE-8
Änderungsverlauf	Ausgabe 01, 2020-11 Siehe Tab. 1-1 "Änderungsverlauf" auf Seite 1
Schutzvermerk	© Bosch Rexroth AG 2020 Alle Rechte vorbehalten, auch bzgl. jeder Verfügung, Verwertung, Reproduktion, Bearbeitung, Weitergabe sowie für den Fall von Schutzrechtsanmeldungen.
Verbindlichkeit	Die angegebenen Daten dienen allein der Produktbeschreibung und sind nicht als zugesicherte Eigenschaften im Rechtssinne zu verstehen. Änderungen im Inhalt der Dokumentation und Liefermöglichkeiten der Produkte sind vorbehalten.
Redaktion	Entwicklung Automationssysteme SPS-Engineering, GeKü (MiNi)

Inhaltsverzeichnis

	Seite
1 Zu dieser Dokumentation.....	1
1.1 Gültigkeit der Dokumentation.....	1
1.2 Gebrauch der Sicherheitshinweise.....	2
1.2.1 Aufbau der Sicherheitshinweise.....	2
1.2.2 Erläuterung der Signalwörter und der Signalgrafik.....	2
1.2.3 Verwendete Symbole.....	4
1.2.4 Erläuterung der Signalgrafik auf dem Gerät.....	4
1.3 Erforderliche und ergänzende Dokumentationen.....	4
1.4 Bezeichnungen und Abkürzungen.....	5
2 Projekt anlegen und konfigurieren.....	7
2.1 'Leeres Projekt' anlegen.....	8
2.2 Objekte hinzufügen.....	8
2.3 SPS-Compilerversion ändern.....	10
2.4 IndraLogic 2G-Projekt öffnen.....	10
2.5 Projekt konfigurieren.....	11
2.5.1 Bibliotheksinfo abfragen und bearbeiten.....	12
2.5.2 Projekteinstellungen vornehmen.....	14
3 Projekt exportieren und transferieren.....	15
3.1 Projekt transferieren.....	15
4 Vergleich von SPS-Objekten.....	17
4.1 Vergleich: Allgemeiner Baustein.....	18
4.2 Vergleich: Logic.....	19
4.3 Vergleich: Application.....	20
4.4 Vergleich: Taskkonfiguration.....	23
4.5 Vergleich: Task.....	24
4.6 Vergleich: Ordner (Unterschied zu IndraWorks Ordner).....	24
4.7 Vergleich: Datentypen	25
4.8 Vergleich: Globale Variablenliste, PersistentVariablen, Netzwerkvariablen.....	26
4.9 Vergleich: Netzwerkvariablenlisten empfängerseitig.....	28
4.10 Vergleich: Programme, Funktionsbausteine, Funktionen.....	29
4.11 Vergleich: Schnittstellen, Schnittstellen-Methoden, Schnittstellen-Eigenschaften.....	36
4.12 Vergleich: Bibliotheksverwalter.....	37
4.13 Vergleich: Bibliotheks Info / Library Info.....	38
4.14 Vergleich: Datenserver.....	39
4.15 Vergleich: Datenquelle (Datenserver).....	41
4.16 Vergleich: Symbolkonfiguration.....	42
4.17 Vergleich: Visualisierungsmanager.....	42
4.18 Vergleich: Visualisierung.....	43
4.19 Vergleich: Textliste.....	43
4.20 Vergleich: Bildersammlung.....	44

	Seite
4.21 Vergleich: Rezepturverwalter.....	44
4.22 Vergleich: Rezepturdefinition.....	46
4.23 Vergleich: Geräte (Onboard E/A, Inline E/A, Feldbusse, ...).....	46
5 Online-Vergleich von SPS-Objekten.....	51
5.1 Übersicht über den Vergleich.....	51
5.2 Detailvergleich.....	52
5.3 Optionen.....	54
5.4 Fehlereinträge im Meldungsfenster.....	55
6 Projekt schützen und speichern.....	57
6.1 Schreibschutz einrichten.....	60
6.2 Benutzerverwaltung einrichten.....	61
6.3 Objekte im Projekt über Zugriffsrechte schützen.....	62
6.4 Anmeldung über Benutzerkonto und Passwort-Manager.....	62
6.5 Projekt speichern.....	65
7 Mehrsprachige SPS-Kommentare.....	67
8 E/A-Anbindung konfigurieren.....	71
8.1 Projektextplorer und Geräteeditor.....	71
8.2 Geräte im Projektextplorer.....	72
8.3 Geräte konfigurieren und E/A Abbild konfigurieren.....	75
9 Applikation programmieren.....	85
9.1 Bezeichner vergeben.....	85
9.2 Variablen deklarieren	85
9.2.1 Deklarationseditor verwenden.....	87
9.2.2 Dialog 'Variable deklarieren' verwenden.....	88
9.2.3 Array deklarieren.....	89
9.2.4 Globale Variablen deklarieren.....	90
9.3 Quellcode in IEC erstellen.....	91
9.3.1 FUP/KOP/AWL.....	92
Programmieren in Funktionsplan (FUP).....	95
Programmieren in Kontaktplan (KOP).....	97
Programmieren in Anweisungsliste (AWL).....	98
9.3.2 Continuous Function Chart (CFC).....	99
Im CFC-Editor programmieren.....	100
9.3.3 Strukturierter Text (ST), Erweiterter Strukturierter Text (ExST).....	104
Programmieren in Strukturierter Text (ST).....	105
9.3.4 Ablaufsprache (AS) / Sequential Function Chart (SFC).....	105
Programmieren in Ablaufsprache (AS / SFC).....	106
9.4 Funktionsbaustein, Funktion oder Methode mit externer Implementierung aufrufen.....	111
9.5 Eingabehilfen nutzen.....	112

	Seite
9.6	Pragmas verwenden..... 114
9.7	Bibliotheksbausteine verwenden..... 116
9.8	Text in einer Textliste verwalten..... 117
9.8.1	Statischen Text in einer globalen Textliste verwalten..... 122
9.8.2	Dynamischen Text in einer Textliste verwalten..... 125
9.9	Bildersammlungen verwenden..... 126
9.10	Programmatischer Zugriff auf E/As..... 128
9.10.1	Variablenkonfiguration - VAR_CONFIG..... 128
9.10.2	AT-Deklaration..... 130
9.11	Syntax prüfen und Code analysieren..... 132
9.11.1	Syntax prüfen..... 132
9.11.2	Code statisch analysieren..... 132
9.12	Orientieren und Navigieren..... 135
9.12.1	Verwendungsstellen mit der Querverweisliste finden..... 135
9.12.2	Deklaration finden - 'Gehe zu Definition'..... 136
9.12.3	Lesezeichen setzen und verwenden..... 137
9.13	Projektweites Suchen und Ersetzen..... 138
9.14	Refactoring..... 139
9.15	Taskkonfiguration..... 143
9.15.1	Erstellen einer Taskkonfiguration..... 144
9.16	Datenpersistenz..... 144
9.16.1	Daten mit persistenten Variablen erhalten..... 148
9.16.2	Daten mit Retain-Variablen erhalten..... 150
9.16.3	Daten mit Rezepturen erhalten..... 151
9.16.4	VAR PERSISTENT-Variablen deklarieren..... 151
9.16.5	Speichern der Werte einer persistenten Variablenliste in einer Rezeptur..... 152
9.17	Bausteine für implizite Prüfung verwenden..... 153
9.18	Objektorientiert programmieren..... 154
9.18.1	Erweitern eines Funktionsbausteins..... 154
9.18.2	Schnittstellen implementieren (IMPLEMENTS)..... 156
9.18.3	Erweitern einer Schnittstelle..... 158
9.18.4	Methodenaufruf..... 158
10	Mit Steuerungsnetzwerken arbeiten..... 163
10.1	Netzwerk und Adressierung..... 163
10.1.1	Netzwerktopologie..... 163
10.1.2	Kommunikation im Steuerungsverbund über Gateway..... 164
10.2	Symbolkonfiguration..... 167
10.3	Netzwerkvariablen..... 168
10.3.1	Konfigurieren eines Netzwerkvariablen-Austauschs..... 171
10.4	Datenquellenverwalter..... 173
10.4.1	Datenquellenverwalter Grundlagen..... 174
10.4.2	Datenquelle initial hinzufügen..... 175
10.4.3	Datenquellenvariablen bearbeiten..... 179
10.4.4	Kommunikation bearbeiten..... 182
10.4.5	Datenschnittstelle aktualisieren..... 182

	Seite
10.4.6 Entfernte Daten verwenden.....	183
10.5 Untergeordnete Sicherheitssteuerung.....	184
11 Applikation auf die SPS übertragen.....	187
11.1 Verbindung zur SPS konfigurieren.....	187
11.2 Applikationscode erzeugen.....	187
11.3 Applikationscode laden, Einloggen und SPS starten.....	188
11.4 Erzeugen einer Bootapplikation.....	189
12 Testen und Fehler beheben.....	193
12.1 Testen im Simulationsbetrieb.....	193
12.2 Verwenden von Haltepunkten.....	193
12.3 Schrittweises Abarbeiten eines Programms (Stepping).....	196
12.4 Forcen und Schreiben von Variablen.....	198
12.5 Reset der Applikation durchführen.....	200
12.6 Ablaufkontrolle.....	201
12.7 Aktuelle Abarbeitungsposition mit der Aufrufliste bestimmen.....	204
12.8 Taskaufstellung überprüfen.....	204
13 Applikation zur Laufzeit.....	205
13.1 Monitoring von Werten.....	205
13.1.1 Monitoring in Programmierobjekten aufrufen.....	205
13.1.2 Überwachungslisten verwenden.....	211
13.2 Werte ändern mit Rezepturen.....	212
13.3 Datenerfassung mit 'Trace'.....	216
13.3.1 Erste Schritte mit Trace.....	218
13.3.2 Tracekonfiguration erstellen.....	219
13.3.3 Datenerfassung steuern.....	223
13.3.4 Zugriff auf alle Traces der Steuerung.....	223
13.3.5 Navigieren in den Tracedaten.....	223
13.3.6 Trace verwalten.....	223
13.3.7 Statistik anzeigen.....	223
13.4 Tasks überwachen.....	224
13.5 Logbuch der SPS lesen.....	224
13.6 Fehleranalyse mit Core Dump.....	224
13.7 SPS-Shell zur Abfrage von Informationen verwenden.....	226
13.8 SPS-Operationssteuerung über Systemvariablen.....	227
14 Applikation auf der SPS aktualisieren.....	229
14.1 Ausführen eines Online Changes.....	229
14.2 Ausführen eines Downloads.....	230
15 Dateien von/zur SPS kopieren.....	233

	Seite
16 Bibliotheken erzeugen und verwalten.....	235
16.1 Informationen für Bibliotheksentwickler.....	236
16.2 Hinzufügen einer Bibliothek zur Applikation.....	237
16.3 Hinzufügen einer Bibliothek zum Repository.....	237
16.4 Bibliotheksdatei exportieren.....	238
17 Referenz Programmierung.....	239
17.1 Programmiersprachen und ihre Editoren.....	239
17.1.1 Deklarationseditor.....	239
17.1.2 Generelle Funktionalitäten in allen grafischen Editoren.....	240
17.1.3 Strukturierter Text und Erweiterter Strukturierter Text (ExST).....	241
ST-Editor.....	241
ST-Editor im Online-Betrieb.....	242
ST-Ausdrücke.....	242
Zuweisungen	244
ST-Zuweisungsoperator.....	244
ST-Zuweisungsoperator für Ausgänge.....	244
ExST-Zuweisung 'S='.....	244
ExST-Zuweisung 'R='.....	245
ExST-Zuweisung ':=' als Ausdruck.....	246
Zuweisungsoperator 'REF'	246
Anweisungen.....	246
ST-Anweisung 'IF'.....	246
ST-Anweisung 'FOR'.....	247
ST-Anweisung 'CASE'.....	248
ST-Anweisung 'WHILE'.....	249
ST-Anweisung 'REPEAT'.....	250
ST-Anweisung 'RETURN'.....	251
ST-Anweisung 'JMP'.....	251
ST-Anweisung 'EXIT'.....	251
EXST-Anweisung 'CONTINUE'.....	252
ST - Aufruf Funktionsbaustein.....	252
ST - Kommentare.....	252
17.1.4 Ablaufsprache (AS).....	253
AS-Editor.....	253
AS-Editor im Online-Betrieb.....	253
Abarbeitungsreihenfolge in AS.....	254
Qualifizierer für Aktionen in AS.....	256
Implizite Variablen.....	257
AS-Flags.....	258
Elemente.....	262
AS-Elemente 'Schritt' und 'Transition'.....	262
AS-Element 'Aktion'.....	264
AS-Element 'Verzweigung'.....	267
AS-Element 'Sprung'.....	269
AS-Element 'Makro'.....	269

	Seite
AS-Elementeigenschaften.....	270
17.1.5 Funktionsplan/Kontaktplan/Anweisungsliste (FUP/KOP/AWL).....	272
FUP/KOP/AWL-Editor.....	272
FUP/KOP/AWL-Editor im Online-Betrieb.....	277
Modifikatoren und Operatoren in AWL.....	278
Elemente.....	282
FUP/KOP/AWL-Element 'Netzwerk'.....	282
FUP/KOP/AWL-Element 'Baustein'.....	282
FUP/KOP/AWL-Element 'Zuweisung'.....	282
FUP/KOP/AWL-Element 'Baustein mit EN/ENO'.....	283
FUP/KOP/AWL-Element 'Eingang'.....	283
FUP/KOP/AWL-Element 'Sprungmarke'.....	283
FUP/KOP/AWL-Element 'Sprung'.....	283
FUP/KOP/AWL-Element 'Return'.....	284
FUP/KOP/AWL-Element 'Leitungsverzweigung'.....	284
FUP/KOP/AWL-Element 'Execute'.....	285
KOP-Element 'Kontakt'.....	285
KOP-Element 'Spule'.....	286
KOP-Element 'Leitungsverzweigung Start/Ende'.....	286
Geschlossene Leitungsverzweigung.....	286
17.1.6 Continuous Function Chart (CFC) und Seitenorientierter CFC.....	288
CFC-Editor.....	288
CFC-Editor seitenorientiert.....	291
CFC-Editor im Onlinebetrieb.....	293
Elemente.....	297
CFC-Element 'Seite'.....	297
CFC-Element 'Kontrollpunkt'.....	297
CFC-Element 'Eingang'.....	297
CFC-Element 'Ausgang'.....	297
CFC-Element 'Baustein'.....	297
CFC-Element 'Sprung'.....	298
CFC-Element 'Marke'.....	298
CFC-Element 'Return'.....	298
CFC-Element 'Kompositor'.....	298
CFC-Element 'Selektor'.....	299
CFC-Element 'Kommentar'.....	299
CFC-Element 'Verbindungsmarke-Quelle/Ziel'.....	299
CFC-Element 'Bausteineingang'.....	300
CFC-Element 'Bausteinausgang'.....	300
17.2 Variablen.....	300
17.2.1 Lokale Variablen - VAR.....	300
17.2.2 Eingabevariablen - VAR_INPUT.....	300
17.2.3 Ausgabevariablen - VAR_OUTPUT.....	301
17.2.4 Ein- und Ausgabevariable - VAR_IN_OUT.....	301
17.2.5 Globale Variablen - VAR_GLOBAL.....	306
17.2.6 Temporäre Variablen - VAR_TEMP.....	306

	Seite
17.2.7 Statische Variablen - VAR_STAT.....	306
17.2.8 Externe Variablen - VAR_EXTERNAL.....	307
17.2.9 Instanzvariablen - VAR_INST.....	307
17.2.10 Konstante Variablen - VAR CONSTANT.....	308
17.2.11 Persistente Variable - PERSISTENT.....	308
17.2.12 Retain-Variable - RETAIN.....	310
17.2.13 Konfigurationsvariablen - VAR_CONFIG.....	311
17.2.14 SUPER.....	312
17.2.15 THIS.....	313
17.3 Operatoren.....	315
17.3.1 Operator 'ADD'.....	319
17.3.2 Operator 'MUL'.....	319
17.3.3 Operator 'SUB'.....	320
17.3.4 Operator 'DIV'.....	320
17.3.5 Operator 'MOD'.....	321
17.3.6 Operator 'MOVE'.....	322
17.3.7 Operator 'INDEXOF'.....	322
17.3.8 Operator 'SIZEOF'.....	322
17.3.9 Operator 'NOT'.....	323
17.3.10 Operator 'AND'.....	323
17.3.11 Operator 'OR'.....	323
17.3.12 Operator 'XOR'.....	324
17.3.13 Operator 'AND_THEN'.....	324
17.3.14 Operator 'OR_ELSE'.....	325
17.3.15 Operator 'SHL'.....	325
17.3.16 Operator 'SHR'.....	326
17.3.17 Operator 'ROL'.....	326
17.3.18 Operator 'ROR'.....	327
17.3.19 Operator 'SEL'.....	328
17.3.20 Operator 'MAX'.....	329
17.3.21 Operator 'MIN'.....	329
17.3.22 Operator 'LIMIT'.....	329
17.3.23 Operator 'MUX'.....	330
17.3.24 Operator 'GT'.....	330
17.3.25 Operator 'LT'.....	331
17.3.26 Operator 'LE'.....	331
17.3.27 Operator 'GE'.....	331
17.3.28 Operator 'EQ'.....	332
17.3.29 Operator 'NE'.....	332
17.3.30 Operator 'ADR'.....	333
17.3.31 Operator 'Inhaltsoperator'.....	333
17.3.32 Operator 'BITADR'.....	334
17.3.33 Operator 'CAL'.....	334
17.3.34 Operator 'BOOL_TO'.....	335
17.3.35 Operator 'TO_BOOL'.....	336
17.3.36 Operator 'TO_<xxx>'.....	337

Inhaltsverzeichnis

	Seite
17.3.37 Operator '<INT-Typ>_TO_<INT-Typ>'.....	337
17.3.38 Operator 'REAL_TO-/ LREAL-TO'.....	338
17.3.39 Operator 'TIME_TO/TIME_OF_DAY_TO'.....	339
17.3.40 Operator 'DATE_TO/DT_TO'.....	340
17.3.41 Operator 'STRING_TO'.....	341
17.3.42 Operator 'TRUNC'.....	342
17.3.43 Operator 'TRUNC_INT'.....	342
17.3.44 Operator 'ABS'.....	343
17.3.45 Operator 'SQRT'.....	343
17.3.46 Operator 'LN'.....	343
17.3.47 Operator 'LOG'.....	344
17.3.48 Operator 'EXP'.....	344
17.3.49 Operator 'EXPT'.....	344
17.3.50 Operator 'SIN'.....	345
17.3.51 Operator 'COS'.....	346
17.3.52 Operator 'TAN'.....	346
17.3.53 Operator 'ASIN'.....	346
17.3.54 Operator 'ACOS'.....	347
17.3.55 Operator 'ATAN'.....	347
17.3.56 Operator '___DELETE'.....	347
17.3.57 Operator '___ISVALIDREF'.....	349
17.3.58 Operator '___NEW'.....	349
17.3.59 Operator '___QUERYINTERFACE'.....	351
17.3.60 Operator '___QUERYPOINTER'.....	352
17.3.61 Operator '___TRY', '___CATCH', '___FINALLY', '___ENDTRY'.....	353
17.3.62 Operator '___VARINFO'.....	354
17.3.63 Operator '___CURRENTTASK'.....	357
17.3.64 Operator '___COMPARE_AND_SWAP'.....	357
17.3.65 Operator '___XADD'.....	358
17.3.66 Operator 'TEST_AND_SET'.....	359
17.3.67 Operator - Globaler Namensraum.....	359
17.3.68 Operator - Namensraum für Globale Variablenlisten.....	360
17.3.69 Operator - Bibliotheks-Namensraum.....	360
17.3.70 Operator - Enumerations-Namensraum.....	361
17.3.71 Operator 'INI'.....	361
17.4 Operanden.....	361
17.4.1 BOOL-Konstanten.....	362
17.4.2 TIME-Konstanten.....	362
17.4.3 DATE-Konstante.....	363
17.4.4 DATE_AND_TIME-Konstante.....	364
17.4.5 TIME_OF_DAY-Konstante.....	365
17.4.6 Zahlenkonstanten.....	365
17.4.7 REAL/LREAL-Konstante.....	366
17.4.8 Stringkonstante.....	367
17.4.9 WSTRING-Konstanten.....	368
17.4.10 Getypte Konstanten / Typed Literals.....	369

	Seite
17.4.11 Zugriff auf Variablen von Arrays, Strukturen und Bausteinen.....	369
17.4.12 Bitzugriff auf Variablen.....	369
17.4.13 Adressen.....	371
17.4.14 Funktionen.....	373
17.5 Datentypen.....	374
17.5.1 Datentyp 'BOOL'.....	375
17.5.2 Ganzzahlige Datentypen.....	375
17.5.3 Datentyp 'REAL' / 'LREAL'.....	375
17.5.4 Datentyp 'STRING'.....	376
17.5.5 Zeitdatentypen.....	376
17.5.6 Datentyp 'UNION'.....	377
17.5.7 Datentyp 'ANY' und 'ANY <type>'.....	378
17.5.8 Datentyp 'LTIME'.....	381
17.5.9 Datentyp 'WSTRING'.....	382
17.5.10 Datentyp 'BIT'.....	382
17.5.11 Spezialdatentypen '__UXINT' und '__XWORD'.....	382
17.5.12 Datentyp 'POINTER'.....	383
17.5.13 Datentyp 'REFERENCE'.....	384
17.5.14 Datentyp 'ARRAY'.....	386
17.5.15 Datentyp '__VECTOR'.....	391
17.5.16 Datentyp 'STRUCT'.....	396
17.5.17 Enumerationen.....	398
17.5.18 Unterbereichstypen.....	401
17.6 Pragmas.....	402
17.6.1 Meldungspragmas.....	402
17.6.2 Attribut-Pragmas.....	403
Benutzerdefinierte Attribute.....	403
Attribut 'call_after_global_init_slot'.....	404
Attribut 'call_after_init'.....	404
Attribut 'call_after_online_change_slot'.....	405
Attribut 'call_before_global_exit_slot'.....	406
Attribut 'call_on_type_change'.....	406
Attribut 'const_replaced', Attribut 'const_non_replaced'.....	407
Attribut 'dataflow'.....	407
Attribut 'displaymode'.....	408
Attribut 'estimated-stack-usage'.....	409
Attribut 'ExpandFully'.....	412
Attribut 'global_init_slot'.....	413
Attribut 'hide'.....	414
Attribut 'hide_all_locals'.....	415
Attribut 'initialize_on_call'.....	415
Attribut 'init_namespace'.....	416
Attribut 'init_on_onlchange'.....	416
Attribut 'instance-path'.....	417
Attribut 'io_function_block', 'io_function_block_mapping'.....	418
Attribut 'linkalways'.....	419

Inhaltsverzeichnis

	Seite
Attribut 'monitoring'.....	419
Attribut 'namespace'.....	423
Attribut 'no_assign', Attribut 'no_assign_warning'.....	424
Attribut 'no_check'.....	424
Attribut 'no_copy'.....	425
Attribut 'no-exit'.....	425
Attribut 'noinit'.....	426
Attribut 'no_instance_in_retain'.....	426
Attribut 'no_virtual_actions'.....	426
Attribut 'pingroup'.....	429
Attribut 'pin_presentation_order_inputs/outputs'.....	429
Attribut 'obsolete'.....	430
Attribut 'pack_mode'.....	431
Attribut 'qualified_only'.....	434
Attribut 'reflection'.....	435
Attribut 'to_string'.....	435
Attribut 'strict'.....	436
Attribut 'subsequent'.....	436
Attribut 'symbol'.....	436
Attribut 'warning disable', Attribut 'warning restore'.....	438
Attribut 'enable_dynamic_creation'.....	438
Attribut 'ProcessValue'.....	438
17.6.3 Bedingte Pragmas.....	439
17.6.4 Region-Pragma.....	445
17.7 Bezeichner.....	445
17.8 Verschattungsregeln.....	451
17.9 Schlüsselwörter	454
17.10 Methode 'FB_Init', 'FB_Reinit' und 'FB_Exit'.....	455
17.11 Compiler Fehlermeldungen.....	459
17.11.1 Compilerfehler C0001.....	460
17.11.2 Compilerfehler C0002.....	460
17.11.3 Compilerfehler C0003.....	460
17.11.4 Compilerfehler C0004.....	461
17.11.5 Compilerfehler C0005.....	461
17.11.6 Compilerfehler C0006.....	461
17.11.7 Compilerfehler C0007.....	462
17.11.8 Compilerfehler C0008.....	462
17.11.9 Compilerfehler C0009.....	462
17.11.10 Compilerfehler C0010.....	462
17.11.11 Compilerfehler C0011.....	463
17.11.12 Compilerfehler C0013.....	463
17.11.13 Compilerfehler C0015.....	463
17.11.14 Compilerfehler C0018.....	464
17.11.15 Compilerfehler C0020.....	464
17.11.16 Compilerfehler C0022.....	465
17.11.17 Compilerfehler C0023.....	465

	Seite
17.11.18 Compilerfehler C0026.....	465
17.11.19 Compilerfehler C0027.....	466
17.11.20 Compilerfehler C0030.....	466
17.11.21 Compilerfehler C0031.....	466
17.11.22 Compilerfehler C0032.....	467
17.11.23 Compilerfehler C0035.....	467
17.11.24 Compilerfehler C0037.....	467
17.11.25 Compilerfehler C0038.....	468
17.11.26 Compilerfehler C0039.....	468
17.11.27 Compilerfehler C0040.....	469
17.11.28 Compilerfehler C0041.....	469
17.11.29 Compilerfehler C0043.....	470
17.11.30 Compilerfehler C0044.....	470
17.11.31 Compilerfehler C0045.....	471
17.11.32 Compilerfehler C0046.....	471
17.11.33 Compilerfehler C0047.....	471
17.11.34 Compilerfehler C0048.....	472
17.11.35 Compilerfehler C0049.....	472
17.11.36 Compilerfehler C0050.....	472
17.11.37 Compilerfehler C0061.....	473
17.11.38 Compilerfehler C0062.....	473
17.11.39 Compilerfehler C0064.....	474
17.11.40 Compilerfehler C0066.....	474
17.11.41 Compilerfehler C0068.....	474
17.11.42 Compilerfehler C0069.....	475
17.11.43 Compilerfehler C0070.....	475
17.11.44 Compilerfehler C0072.....	475
17.11.45 Compilerfehler C0074.....	476
17.11.46 Compilerfehler C0075.....	476
17.11.47 Compilerfehler C0076.....	476
17.11.48 Compilerfehler C0077.....	477
17.11.49 Compilerfehler C0080.....	477
17.11.50 Compilerfehler C0081.....	478
17.11.51 Compilerfehler C0082.....	478
17.11.52 Compilerfehler C0084.....	478
17.11.53 Compilerfehler C0086.....	479
17.11.54 Compilerfehler C0087.....	479
17.11.55 Compilerfehler C0089.....	480
17.11.56 Compilerfehler C0090.....	480
17.11.57 Compilerfehler C0091.....	480
17.11.58 Compilerfehler C0094.....	481
17.11.59 Compilerfehler C0096.....	481
17.11.60 Compilerfehler C0097.....	482
17.11.61 Compilerfehler C0098.....	482
17.11.62 Compilerfehler C0101.....	482
17.11.63 Compilerfehler C0102.....	483

Inhaltsverzeichnis

	Seite
17.11.64 Compilerfehler C0104.....	483
17.11.65 Compilerfehler C0114.....	483
17.11.66 Compilerfehler C0116.....	483
17.11.67 Compilerfehler C0117.....	484
17.11.68 Compilerfehler C0118.....	484
17.11.69 Compilerfehler C0119.....	484
17.11.70 Compilerfehler C0120.....	485
17.11.71 Compilerfehler C0122.....	485
17.11.72 Compilerfehler C0124.....	485
17.11.73 Compilerfehler C0125.....	486
17.11.74 Compilerfehler C0126.....	486
17.11.75 Compilerfehler C0130.....	486
17.11.76 Compilerfehler C0131.....	487
17.11.77 Compilerfehler C0132.....	487
17.11.78 Compilerfehler C0136.....	487
17.11.79 Compilerfehler C0139.....	488
17.11.80 Compilerfehler C0140.....	488
17.11.81 Compilerfehler C0141.....	489
17.11.82 Compilerfehler C0142.....	489
17.11.83 Compilerfehler C0143.....	489
17.11.84 Compilerfehler C0144.....	490
17.11.85 Compilerfehler C0145.....	490
17.11.86 Compilerfehler C0149.....	490
17.11.87 Compilerfehler C0161.....	491
17.11.88 Compilerfehler C0162.....	491
17.11.89 Compilerfehler C0168.....	491
17.11.90 Compilerfehler C0169.....	492
17.11.91 Compilerfehler C0174.....	492
17.11.92 Compilerfehler C0175.....	492
17.11.93 Compilerfehler C0177.....	492
17.11.94 Compilerfehler C0178.....	493
17.11.95 Compilerfehler C0182.....	493
17.11.96 Compilerfehler C0185.....	493
17.11.97 Compilerfehler C0189.....	494
17.11.98 Compilerfehler C0190.....	494
17.11.99 Compilerfehler C0191.....	494
17.11.100 Compilerfehler C0195.....	494
17.11.101 Compilerfehler C0196.....	495
17.11.102 Compilerfehler C0197.....	495
17.11.103 Compilerfehler C0198.....	495
17.11.104 Compilerfehler C0199.....	496

	Seite
17.11.105 Compilerfehler C0201.....	496
17.11.106 Compilerfehler C0203.....	496
17.11.107 Compilerfehler C0204.....	497
17.11.108 Compilerfehler C0205.....	497
17.11.109 Compilerfehler C0206.....	497
17.11.110 Compilerfehler C0208.....	498
17.11.111 Compilerfehler C0211.....	498
17.11.112 Compilerfehler C0212.....	498
17.11.113 Compilerfehler C0216.....	498
17.11.114 Compilerfehler C0217.....	499
17.11.115 Compilerfehler C0218.....	499
17.11.116 Compilerfehler C0219.....	499
17.11.117 Compilerfehler C0221.....	500
17.11.118 Compilerfehler C0222.....	500
17.11.119 Compilerfehler C0224.....	500
17.11.120 Compilerfehler C0227.....	501
17.11.121 Compilerfehler C0228.....	501
17.11.122 Compilerfehler C0234.....	501
17.11.123 Compilerfehler C0235.....	502
17.11.124 Compilerfehler C0236.....	502
17.11.125 Compilerfehler C0237.....	502
17.11.126 Compilerfehler C0238.....	503
17.11.127 Compilerfehler C0239.....	503
17.11.128 Compilerfehler C0240.....	503
17.11.129 Compilerfehler C0241.....	504

	Seite
17.11.130 Compilerfehler C0242.....	504
17.11.131 Compilerfehler C0243.....	505
17.11.132 Compilerfehler C0380.....	505
17.11.133 Compilerfehler C0509.....	505
17.11.134 Compilerfehler C0511.....	506
 18 Referenz Benutzeroberfläche.....	 507
18.1 Objekte.....	507
18.1.1 Objekt 'Allgemeiner Baustein'.....	507
18.1.2 Objekt 'Applikation'.....	508
18.1.3 Objekt 'Datenquellenverwalter'.....	510
18.1.4 Objekt 'Datenquelle'- Registerkarte 'Variablen'.....	511
18.1.5 Objekt 'Datenquelle' - Registerkarte 'Typabbildungen'.....	513
18.1.6 Registerkarte 'Kommunikation'.....	514
18.1.7 Objekt 'Datenquelle' - Registerkarte 'Allgemein und Diagnose'.....	517
18.1.8 Objekt 'DUT'.....	517
18.1.9 Objekt 'Gerät' und Generischer Geräteeditor.....	519
Generischer Geräteeditor.....	519
Registerkarte 'Parameter'.....	520
Registerkarte 'Applikationen'.....	521
Registerkarte 'Sichern und Wiederherstellen'.....	522
Registerkarte 'Synchronisierte Dateien'.....	524
Registerkarte 'Dateien'.....	525
Registerkarte 'Log'.....	525
Registerkarte 'SPS-Einstellungen'.....	527
Registerkarte 'Parameter'.....	529
Registerkarte 'SPS-Shell'.....	532
Registerkarte '<Gerätename> E/A-Abbild'.....	534
Registerkarte '<Gerätename> IEC-Objekte'.....	537
Registerkarte 'Benutzer und Gruppen'.....	539
Registerkarte 'Zugriffsrechte'.....	540
Registerkarte 'Task-Aufstellung'.....	542
Registerkarte 'Status'.....	543
Registerkarte 'Information'.....	543
18.1.10 Objekt 'GlobalTextList'.....	543
18.1.11 Objekt 'GVL'- Globale Variablenliste.....	544
18.1.12 Objekt 'Persistente Variablenliste'.....	546
18.1.13 Objekt 'Bildersammlung'.....	547
18.1.14 Objekt 'Bibliotheksverwalter'.....	548
18.1.15 Objekt 'Globale Netzwerkvariablen-Liste (Sender)'.....	551
18.1.16 Objekt 'Globale Netzwerkvariablen-Liste (Empfänger)'.....	552
18.1.17 Objekt 'POU'.....	552

	Seite
Objekt 'Programm' (PROGRAM).....	554
Objekt 'Funktionsbaustein' (FUNCTION_BLOCK).....	556
Objekt 'Funktion' (FUNCTION).....	558
Objekt 'Schnittstelle' (INTERFACE).....	562
Objekt 'Methode'	564
Objekt 'Schnittstellenmethode'.....	567
Objekt 'Schnittstelleneigenschaft'.....	568
Objekt 'Eigenschaft'.....	571
Objekt 'Aktion'.....	574
Objekt 'Transition'.....	576
18.1.18 Überwachung des SPS-Programms der Applikation während der Laufzeit.....	576
POU-Vorlagen.....	578
18.1.19 Objekt 'Projekteinstellungen'.....	579
18.1.20 Objekt 'Bibliotheksinfo'.....	579
18.1.21 Objekt 'Rezepturverwalter'.....	583
18.1.22 Objekt 'Rezepturdefinition'.....	587
18.1.23 Objekt 'Textliste'.....	588
18.1.24 Objekt 'Symbolkonfiguration'.....	588
18.1.25 Objekt 'Taskkonfiguration'.....	598
Registerkarte 'Eigenschaften'.....	598
Registerkarte 'Systemereignisse'.....	599
Registerkarte 'Überwachung'.....	601
Registerkarte 'Variablenverwendung'.....	601
Registerkarte 'Taskgruppen'.....	602
18.1.26 Objekt 'Task'.....	602
Registerkarte 'Konfiguration'.....	602
18.1.27 Objekt 'Trace'.....	605
18.1.28 Objekt 'Zugriffskontrolle'.....	608
18.2 Menübefehle.....	610
18.2.1 Menü 'Bearbeiten'.....	610
Standardbefehle.....	610
Suchen und Ersetzen.....	610
Befehl 'Suchen'.....	610
Befehl 'Ersetzen'.....	612
Befehl 'Abwärts Suchen'.....	613
Befehl 'Aufwärts suchen'.....	613
Befehl 'Abwärts suchen (Auswahl)'.....	613
Befehl 'Aufwärts suchen (Auswahl)'.....	613
Lesezeichen.....	614
Befehl 'Lesezeichen umschalten'.....	614
Befehl 'Nächstes Lesezeichen'.....	614
Befehl 'Vorheriges Lesezeichen'.....	614
Befehl 'Lesezeichen löschen'.....	614
Datei als Text einfügen.....	615
Befehl 'Datei als Text einfügen'.....	615
Durchsuchen.....	615

	Seite
Befehl 'Gehe zur Definition'.....	615
Befehl 'Gehe zu Referenz'.....	615
Befehl 'Gehe zu Instanz'.....	616
Befehl 'Querverweise ausgeben'.....	616
Befehl 'Aufrufbaum ausgeben'.....	616
Refactoring.....	617
Befehl 'Refactoring Umbenennen'.....	617
Befehl 'Variable hinzufügen'.....	618
Befehl 'Variable entfernen...'.....	619
Befehl 'Variablen neu ordnen...'.....	620
Befehl 'Betreffende Anschlüsse aktualisieren'.....	620
Erweitert.....	621
Befehl 'Leerzeichen anzeigen'.....	621
Befehl 'Einrückungshilfslinien anzeigen'.....	621
Befehl 'Alles reduzieren'.....	621
Befehl 'Alles erweitern'.....	622
Befehl 'Selektierte Zeilen auskommentieren'.....	622
Befehl 'Selektierte Zeilen Auskommentierung entfernen'.....	623
Befehl 'Gehe zu'.....	623
Befehl 'In Großbuchstaben umwandeln'.....	623
Befehl 'In Kleinbuchstaben umwandeln'.....	624
Befehl 'Korrespondierende Klammer'.....	624
Befehl 'Bis korrespondierende Klammer markieren'.....	624
Eingabehilfe.....	624
Befehl 'Eingabehilfe'.....	624
Variable deklarieren.....	625
Befehl 'Variable deklarieren'.....	625
18.2.2 Menü 'Ansicht'.....	628
Projektexplorer.....	628
Bibliothek.....	628
Andere Fenster.....	628
Befehl 'Werkzeuge'.....	628
Befehl 'Querverweisliste'.....	628
Befehl 'Aufrufbaum'.....	631
Befehl 'Querverweise in der klassischen Ansicht ausgeben'.....	632
Befehl 'Elementeigenschaften'.....	632
Befehl 'Speicher'.....	633
Befehl 'Ganzer Bildschirm'.....	634
Befehl 'Eigenschaften'.....	634
Befehl 'Startbild'.....	634
18.2.3 Menü 'Projekt'.....	634
Befehl 'Projekteinstellungen'.....	634
Befehl 'Benutzerverwaltung'.....	634
Konfiguration der Benutzerverwaltung.....	635
Befehl 'SPS-Compilerversion'.....	641
Befehl 'SPS-Projektvergleich'.....	641

	Seite
Befehl 'Hinzufügen'.....	643
Befehl 'Ordner hinzufügen'.....	644
Befehl 'SPS-Objekte eines externen IndraWorks-Projektes anzeigen'.....	644
Befehl 'Objekt (offline) bearbeiten'.....	646
Befehl 'Aktive Applikation setzen'.....	646
Befehl 'E/A-Abbild bearbeiten'.....	647
Befehl 'E/A-Abbild aus CSV importieren'.....	648
Befehl 'E/A-Abbild in CSV exportieren'.....	648
Befehl 'Globale Variablenliste (GVL) in XML-Format austauschen'.....	648
18.2.4 Menü 'Erstellen'.....	649
Befehl 'Übersetzen'.....	649
Befehl 'Code erzeugen'.....	649
Befehl 'Bereinigen'.....	650
Befehl 'Neu übersetzen'.....	650
Befehl 'Laufzeitsystemdateien erzeugen'.....	650
Befehl 'Alles bereinigen'.....	651
18.2.5 Menü 'Debuggen'.....	652
Befehl 'Einloggen'.....	652
Befehl 'Ausloggen'.....	655
Befehl 'Bootapplikation erzeugen'.....	656
Befehl 'Aktuellen Gerätebenutzer ausloggen'.....	656
Befehl 'Laden'.....	656
Befehl 'Online-Change'.....	657
Befehl 'Start'.....	659
Befehl 'Stop'.....	659
Befehl 'Einzelzyklus'.....	659
Befehl 'Mehrfacher Download'.....	659
Befehl 'Reset Warm'.....	661
Befehl 'Reset Kalt'.....	662
Befehl 'Reset Ursprung'.....	663
Befehl 'Haltepunkte'.....	664
Befehl 'Neuer Haltepunkt'.....	666
Befehl 'Haltepunkt bearbeiten'.....	666
Befehl 'Haltepunkt umschalten'.....	666
Befehl 'Haltepunkt aktivieren'.....	666
Befehl 'Haltepunkt deaktivieren'.....	666
Befehl 'Aufrufliste'.....	667
Befehl 'Überwachung' - 'Überwachungsliste <n>'.....	667
Befehl 'Überwachung' - 'Alle Forces anzeigen'.....	668
Befehl 'Alle Forces zur Überwachungsliste hinzufügen'.....	668
Befehl 'Prozedurschritt'.....	669
Befehl 'Einzelschritt'.....	669
Befehl 'Ausführen bis Rücksprung'.....	669
Befehl 'Ausführen bis Cursor'.....	670
Befehl 'Nächste Anweisung festlegen'.....	670
Befehl 'Nächste Anweisung anzeigen'.....	670

	Seite
Befehl 'Werte forcen'.....	671
Befehl 'Werte schreiben'.....	672
Befehl 'Forcen für alle Werte aufheben'.....	672
Befehl 'Ablaufkontrollmodus umschalten'.....	673
Befehl 'Betriebsmodus'.....	673
Befehl 'Core Dump'.....	674
Befehl 'Core Dump erzeugen'.....	674
Befehl 'Core Dump laden'.....	675
Befehl 'Geräte-Logbuch vom Core Dump laden'.....	675
Befehl 'Core Dump schließen'.....	675
Befehl 'Anzeigemodus'.....	676
18.2.6 Menü 'Extras'.....	676
Bibliotheken.....	676
Befehl 'Bibliotheks-Repository'.....	676
Erzeugen einer Bibliothek im IndraWorks.....	678
Befehl 'Prüfe Bibliothekskompatibilität...'.....	681
Befehl 'Anpassen'.....	682
Befehl 'Optionen'.....	682
18.2.7 Menü 'AS'.....	682
Befehl 'Initialschritt'.....	682
Befehl 'Schritt-Transition einfügen'.....	683
Befehl 'Schritt-Transition danach einfügen'.....	683
Befehl 'Parallel'.....	684
Befehl 'Alternativ'.....	684
Befehl 'Verzweigung einfügen'.....	684
Befehl 'Verzweigung rechts einfügen'.....	684
Befehl 'Aktionsassoziation einfügen'.....	686
Befehl 'Aktionsassoziation danach einfügen'.....	687
Befehl 'Sprung einfügen'.....	687
Befehl 'Sprung danach einfügen'.....	687
Befehl 'Makro einfügen'.....	687
Befehl 'Makro danach einfügen'.....	688
Befehl 'Makro anzeigen'.....	688
Befehl 'Makro verlassen'.....	688
Befehl 'Einfügen danach'.....	688
18.2.8 Menü 'CFC'.....	689
Befehl 'Arbeitsblatt bearbeiten'.....	689
Befehl 'Seitengröße bearbeiten'.....	689
Befehl 'Negieren'.....	689
Befehl 'Befehl 'EN/ENO'.....	689
Befehl 'Kein'.....	690
Befehl 'R-Reset'.....	690
Befehl 'S-SET'.....	690
Befehl 'Ref= (Referenzzuweisung)'.....	691
Befehl 'An den Anfang'.....	691
Befehl 'Ans Ende'.....	692

	Seite
Befehl 'Eins vor'.....	692
Befehl 'Eins zurück'.....	693
Befehl 'Nach Datenfluss anordnen'.....	693
Befehl 'Topologisch anordnen'.....	694
Befehl 'Ausführungsreihenfolge setzen'.....	694
Befehl 'Parameter bearbeiten'.....	694
Befehl 'Vorbereitete Parameter im Projekt speichern'.....	695
Befehl 'Selektierte Anschlüsse verbinden'.....	696
Befehl 'Verbindung lösen'.....	696
Befehl 'Nächste Kollision zeigen'.....	696
Befehl 'Verbundene Anschlüsse selektieren'.....	697
Befehl 'Anschlüsse zurücksetzen'.....	697
Befehl 'Nicht verbundene Anschlüsse entfernen'.....	697
Befehl 'Eingangsanschluss hinzufügen'.....	697
Befehl 'Ausgangsanschluss hinzufügen'.....	697
Befehl 'Alle Verbindungen routen'.....	698
Befehl 'Kontrollpunkt entfernen'.....	698
Befehl 'Kontrollpunkt erzeugen'.....	698
Befehl 'Verbindungsmarke'.....	699
Befehl 'Gruppe erzeugen'.....	699
Befehl 'Gruppierung aufheben'.....	699
Befehl 'Funktionsbausteinelement für Forcen vorbereiten'.....	699
Befehl 'FB-Eingang forcen'.....	700
Befehl 'Attributierte Komponente als Eingang verwenden'.....	701
18.2.9 Menü 'FUP/KOP/AWL'.....	702
Befehl 'Netzwerk einfügen'.....	702
Befehl 'Netzwerk einfügen unterhalb'.....	703
Befehl 'Kommentierung ein/aus'.....	703
Befehl 'Zuweisung einfügen'.....	703
Befehl 'Bausteinaufruf einfügen'.....	703
Befehl 'Baustein mit EN/ENO einfügen'.....	704
Befehl 'Leeren Baustein einfügen'.....	704
Befehl 'Leeren Baustein mit EN/ENO einfügen'.....	704
Befehl 'Sprung einfügen'.....	705
Befehl 'Sprungmarke einfügen'.....	705
Befehl 'Return einfügen'.....	705
Befehl 'Bausteineingang einfügen'.....	705
Befehl 'Spule einfügen'.....	706
Befehl 'Set-Spule einfügen'.....	706
Befehl 'Reset-Spule einfügen'.....	706
Befehl 'Kontakt einfügen'.....	706
Befehl 'Kontakt einfügen (rechts)'.....	706
Befehl 'Kontakt parallel einfügen (unterhalb)'.....	707
Befehl 'Kontakt parallel einfügen (oberhalb)'.....	707
Befehl 'Parallelen Modus wechseln'.....	708
Befehl 'Negierten Kontakt einfügen'.....	708

	Seite
Befehl 'Negierten Kontakt parallel einfügen (unterhalb)'.....	708
Befehl 'Kontakte einfügen: Darunter einfügen'.....	708
Befehl 'Kontakte einfügen: Darüber einfügen'.....	709
Befehl 'Kontakte einfügen: Rechts einfügen (danach)'.....	709
Befehl 'AWL-Zeile danach einfügen'.....	709
Befehl 'AWL-Zeile löschen'.....	709
Befehl 'Negation'.....	709
Befehl 'Flankenerkennung'.....	710
Befehl 'Set/Reset'.....	710
Befehl 'Weiterverschaltung festlegen'.....	710
Befehl 'Leitungsverzweigung einfügen'.....	710
Befehl 'Leitungsverzweigung oberhalb einfügen'.....	711
Befehl 'Leitungsverzweigung unterhalb einfügen'.....	711
Befehl 'Leitungsverzweigung Startpunkt setzen'.....	711
Befehl 'Leitungsverzweigung Endpunkt setzen'.....	711
Befehl 'Parameter aktualisieren'.....	712
Befehl 'Nicht verwendete FB-Aufruf-Parameter entfernen'.....	712
Befehl 'Baustein reparieren'.....	712
Befehl 'Als Funktionsbausteinsprache anzeigen'.....	713
Befehl 'Als Kontaktplan anzeigen'.....	713
Befehl 'Als Anweisungsliste anzeigen'.....	713
Befehl 'Gehe zu'.....	714
18.2.10 Menü 'Bibliothek'.....	714
Befehl 'Bibliothek hinzufügen'.....	714
Befehl 'Versuchen Bibliothek neu zu laden'.....	715
Befehl 'Eigenschaften'.....	716
Befehl 'Platzhalter'.....	718
Befehl 'Bibliotheksinformationen'.....	719
18.2.11 Menü 'Bildersammlung'.....	719
Befehl 'Bilddatei einfügen'.....	719
18.2.12 Menü 'Deklarationen'.....	720
Befehl 'Deklarationskopf editieren'.....	720
Befehl 'Nach unten verschieben'.....	720
Befehl 'Nach oben verschieben'.....	720
18.2.13 Menü 'Deklarationen' (Persistenz).....	721
Befehl 'Liste neu ordnen und Lücken bereinigen'.....	721
Befehl 'Aktuelle Werte in Rezeptur speichern'.....	721
Befehl 'Werte aus Rezeptur wiederherstellen'.....	722
Befehl 'Alle Instanzpfade hinzufügen'.....	722
18.2.14 Menü 'Rezepturen'.....	723
Befehl 'Variable einfügen'.....	723
Befehl 'Rezeptur einfügen'.....	723
Befehl 'Rezeptur entfernen'.....	723
Befehl 'Rezeptur laden'.....	724
Befehl 'Rezeptur speichern'.....	724
Befehl 'Rezeptur lesen'.....	725

	Seite
Befehl 'Rezeptur schreiben'.....	725
Befehl 'Rezeptur laden und schreiben'.....	725
Befehl 'Rezeptur lesen und speichern'.....	726
Befehl 'Variable entfernen'.....	726
Befehl 'Strukturierte Variablen aktualisieren'.....	727
18.2.15 Menü 'Textliste'.....	727
Befehl 'Sprache einfügen'.....	727
Befehl 'Globale Textliste anlegen'.....	727
Befehl 'Alles als Text exportieren'.....	728
Befehl 'Alles als Unicode-Text exportieren'.....	728
Befehl 'Text einfügen'.....	729
Befehl 'Import/Export Textlisten'.....	729
Befehl 'Sprache entfernen'.....	730
Befehl 'Nicht verwendete Textlisteneinträge entfernen'.....	731
Befehl 'Visualisierungs-Text-IDs prüfen'.....	731
Befehl 'Visualisierungs-Text-IDs aktualisieren'.....	731
Befehl 'Textlistenunterstützung hinzufügen'.....	731
Befehl 'Textlistenunterstützung entfernen'.....	732
18.2.16 Menü 'Trace'.....	732
Befehl 'Variable hinzufügen'.....	732
Befehl 'Automatisch skalieren'.....	732
Befehl 'Komprimieren'.....	732
Befehl 'Konfiguration'.....	733
Befehl 'Cursor'.....	733
Befehl 'Download Trace'.....	734
Befehl 'Symbolische Tracekonfiguration exportieren'.....	734
Befehl 'Trace laden'.....	735
Befehl 'Zoomen mit Maus'.....	736
Befehl 'in Mehrkanal konvertieren'.....	736
Befehl 'in Einkanal konvertieren'.....	736
Befehl 'Onlineliste'.....	737
Befehl 'Trigger zurücksetzen'.....	738
Befehl 'Ansicht zurücksetzen'.....	738
Befehl 'Trace speichern'.....	738
Befehl 'Trace starten'.....	739
Befehl 'Trace stoppen'.....	739
Befehl 'Strecken'.....	739
Befehl 'Trace hochladen'.....	740
Befehl 'Statistik'.....	740
18.2.17 Sonstige.....	741
Befehl 'Simulation'.....	741
Befehl 'Zu Überwachungsliste hinzufügen'.....	742
Befehl 'Schnittstellen implementieren'.....	742
Befehl 'Ergebnisse auf aktuelle Deklaration beschränken'.....	743
Befehl 'Ausführung an behandelten Exceptions anhalten'.....	743
18.3 Dialoge.....	744

Inhaltsverzeichnis

	Seite
18.3.1	Dialog 'Funktionsbaustein auswählen'..... 744
18.3.2	Dialog 'Eigenschaften Haltepunkt'..... 745
18.3.3	Dialog 'Wert vorbereiten'..... 745
18.3.4	Dialog 'Neuer Haltepunkt'..... 745
18.3.5	Dialog 'Monitoringbereich'..... 748
18.3.6	Dialog 'Eigenschaften'..... 748
	Dialog 'Eigenschaften' - 'Allgemein'..... 748
	Dialog 'Eigenschaften' - 'Boot-Applikation'..... 748
	Dialog 'Eigenschaften' - 'Übersetzen'..... 749
	Dialog 'Eigenschaften' - 'Zugriffskontrolle'..... 750
	Dialog 'Eigenschaften' - 'Bitmap'..... 751
	Dialog 'Eigenschaften' - 'Optionen Applikationserzeugung'..... 751
	Dialog 'Eigenschaften' - 'Netzwerkvariablen'..... 751
	Dialog 'Eigenschaften' - 'Netzwerkeinstellungen'..... 753
	Dialog 'Eigenschaften' - Registerkarte 'CFC-Ausführungsreihenfolge'..... 753
	Dialog 'Eigenschaften' - 'AS-Einstellungen'..... 756
	Dialog 'Eigenschaften' - 'Verknüpfung mit Datei'..... 756
	Dialog 'Eigenschaften' - 'Bildersammlung'..... 757
	Dialog 'Eigenschaften' - 'Optionen'..... 757
18.3.7	Dialog 'Projekteinstellungen'..... 758
	Dialog 'Projekteinstellungen' - 'AS'..... 758
	Dialog 'Projekteinstellungen' - 'Compileroptionen'..... 759
	Dialog 'Projekteinstellungen' - 'Compilerwarnungen'..... 760
	Dialog 'Projekteinstellungen' - 'Static Analysis Light'..... 761
	Dialog 'Projekteinstellungen' - 'Visualisierung'..... 763
18.3.8	Dialog 'Eigenschaften' - 'Textliste'..... 765
18.3.9	Dialog 'Optionen'..... 765
	Dialog 'Optionen' - 'Allgemeine Einstellungen'..... 765
	Dialog 'Optionen' - 'CFC-Editor'..... 765
	Dialog 'Optionen' - 'Deklarationseditor'..... 766
	Dialog 'Optionen' - 'Geräteeditor'..... 767
	Dialog 'Optionen' - 'FUP, KOP und AWL'..... 768
	Dialog 'Optionen' - 'IndraLogic 1x-Konverter'..... 769
	Dialog 'Optionen' - 'Bibliotheken'..... 771
	Dialog 'Optionen' - 'Refactoring'..... 771
	Dialog 'Optionen' - 'Sichere FUP-Optionen'..... 772
	Dialog 'Optionen' - 'AS-Editor'..... 773
	Dialog 'Optionen' - 'Intelligentes Codieren'..... 775
	Dialog 'Optionen' - 'Syntaxhervorhebung'..... 776
	Dialog 'Optionen' - 'Texteditor'..... 776
	Dialog 'Optionen' - 'Visualisierung'..... 779
	Dialog 'Optionen' - 'Visualisierungsstile'..... 781
18.3.10	Dialog 'Tracekonfiguration'..... 782
	Dialog 'Erweiterte Traceeinstellungen'..... 782
	Dialog 'Tracekonfiguration'..... 783

	Seite
19 Bibliotheken.....	789
19.1 Richtlinien für das Erstellen von Bibliotheken.....	789
19.2 Bibliothek 'Recipe Management'.....	789
20 IndraLogic 2G-Visualisierung.....	803
20.1 Visualisierung im Überblick.....	803
20.1.1 Anzeigen der Vorschau auf eine Bibliotheksvisualisierung.....	805
20.1.2 Verfügbare Visualisierungselemente anzeigen und kennzeichnen.....	806
20.1.3 Symbolbibliothek in der Visualisierung verwenden.....	809
20.2 Befehle.....	812
20.2.1 Befehl 'Visualisierungselement einfügen'.....	813
20.2.2 Befehl 'Schnittstelleneditor'.....	813
20.2.3 Befehl 'Tastaturbedienung'.....	813
20.2.4 Befehl 'Elementliste'.....	813
20.2.5 Befehl 'Tastaturbedienung aktivieren'.....	813
20.2.6 Ausrichtung.....	814
Befehl 'Links ausrichten'.....	814
Befehl 'Oben ausrichten'.....	814
Befehl 'Rechts ausrichten'.....	814
Befehl 'Unten ausrichten'.....	814
Befehl 'Vertikal zentrieren'.....	814
Befehl 'Horizontal zentrieren'.....	814
Befehl 'Horizontalen Abstand gleichmachen'.....	814
Befehl 'Horizontalen Abstand vergrößern'.....	815
Befehl 'Horizontalen Abstand verkleinern'.....	815
Befehl 'Horizontalen Abstand entfernen'.....	815
Befehl 'Vertikalen Abstand gleichmachen'.....	816
Befehl 'Vertikalen Abstand vergrößern'.....	816
Befehl 'Vertikalen Abstand verkleinern'.....	816
Befehl 'Vertikalen Abstand entfernen'.....	817
Befehl 'Breite gleichmachen'.....	817
Befehl 'Höhe gleichmachen'.....	817
Befehl 'Größe gleichmachen'.....	818
Befehl 'Größe an Raster anpassen'.....	818
20.2.7 Reihenfolge.....	818
Befehl 'Um Eins nach vorn legen'.....	818
Befehl 'Nach vorn legen'.....	818
Befehl 'Um Eins nach hinten legen'.....	818
Befehl 'Nach hinten legen'.....	819
20.2.8 Befehl 'Gruppieren'.....	819
20.2.9 Befehl 'Gruppierung aufheben'.....	819
20.2.10 Befehl 'Frame-Auswahl'.....	819
20.2.11 Befehl 'Hintergrund'.....	821
20.2.12 Befehl 'Alles selektieren'.....	821
20.2.13 Befehl 'Alles deselektieren'.....	822
20.2.14 Befehl 'Trace konfigurieren...'.....	822

	Seite
20.2.15	Befehl 'Elemente für Trace-Kontrolle einfügen'..... 822
20.2.16	Befehl 'Frame-Parameter in <geöffnete Visualisierung> aktualisieren'..... 822
20.2.17	Befehl 'Visualisierungselement vervielfachen'..... 823
20.3	Editor..... 825
20.3.1	Ansicht 'Werkzeuge'..... 826
20.3.2	Ansicht 'Eigenschaften'..... 829
20.3.3	Registerkarte 'Schnittstelleneditor'..... 831
	Grundlegendes zum Schnittstelleneditor..... 831
	Aktualisierung der Frameparameter..... 833
	Pragmas..... 836
	Attribut 'VAR_IN_OUT_AS_POINTER'..... 836
	Attribut 'parameterstringof'..... 838
20.3.4	Registerkarte 'Tastaturkonfiguration'..... 839
20.3.5	Registerkarte 'Elementliste'..... 841
20.3.6	Dialog 'Visualisierung hinzufügen'..... 842
20.3.7	Dialog 'Kennzeichen und Einträge konfigurieren'..... 842
20.4	Visualisierungselemente..... 843
20.4.1	Elemente konfigurieren - allgemein..... 843
	Benutzereingabe beim Editieren eines Elements..... 843
	Text und Sprache verwenden..... 845
	Eingabekonfiguration..... 849
	Farbverlaufeditor..... 862
	Semitransparente Farbe verwenden..... 863
	Vervielfachen eines Elements..... 865
	IEC-Variable auf ein Element ziehen..... 868
20.4.2	'Basis'..... 869
	Element 'Rechteck', 'Abgerundetes Rechteck' und 'Ellipse'..... 869
	Element 'Linie'..... 878
	Element 'Polygon', 'Linienzug', 'Bézierkurve'..... 887
	Element 'Kreissektor'..... 896
	Bild und dessen Dialog..... 906
	Element 'Bild'..... 906
	Dialog 'Eingabehilfe'..... 915
	Element 'Frame'..... 917
20.4.3	'Allgemeine Steuerelemente'..... 929
	Element 'Text'..... 929
	Element 'Combobox Integer'..... 931
	Element 'Combobox Array'..... 934
	Element 'Registersteuerelement'..... 937
	Element 'Schaltfläche'..... 940
	Element 'Groupbox'..... 949
	Element 'Tabelle'..... 952
	Element 'Textfeld'..... 963
	Element 'Scrollbar'..... 970
	Element 'Schieberegler'..... 975
	Element 'SpinControl'..... 978

	Seite
Element 'Unsichtbare Eingabe'.....	982
Element 'Fortschrittsbalken'.....	986
Element 'Radiobutton'.....	988
Element 'Checkbox'.....	990
20.4.4 'Messgeräte'.....	992
Element 'Balkenanzeige', 'Balkenanzeige frei konfigurierbar'.....	992
Element 'Zeigerinstrument 90°'.....	996
Element 'Zeigerinstrument 180°'.....	1001
Element 'Zeigerinstrument 360°'.....	1005
Element 'Potentiometer'.....	1009
Element 'Histogramm'.....	1013
20.4.5 'Lampen/Schalter/Bilder'.....	1019
Element 'Bildwechsler'.....	1019
Element 'Lampe'.....	1023
Element 'Hebelschalter', 'Energieschalter', 'Drückschalter', 'Drückschalter LED', 'Kippschalter'.....	1026
Element 'Drehschalter'.....	1029
20.4.6 'Spezielle Steuerelemente'.....	1032
Element 'Trace' und seine Dialoge.....	1032
Element 'Trace'.....	1032
Dialog 'Tracekonfiguration, Aufzeichnungseinstellungen'.....	1041
Dialog 'Tracekonfiguration, Variableneinstellung'.....	1043
Dialog 'Erweiterte Trace-Einstellungen'.....	1046
Dialog 'Anzeigeeinstellungen bearbeiten'.....	1046
Dialog 'Kopiere Einstellungen von Traceinstanz'.....	1048
Dialog 'Traceassistent'.....	1048
Befehl 'Tracekonfiguration exportieren'.....	1049
Element 'ActiveX-Element'.....	1050
Element 'Wartesymbol Blume'.....	1052
20.4.7 Visualisierungselementeigenschaften.....	1054
Eigenschaft 'Diagramm'.....	1054
Eigenschaft 'Elementname'.....	1054
Eigenschaft 'Zahlenformat'.....	1054
Eigenschaft 'Ausrichtung'.....	1054
Eigenschaft 'Position'.....	1054
Eigenschaft 'Rahmen anzeigen'.....	1055
Eigenschaft 'Tooltip anzeigen'.....	1055
Eigenschaft 'Texteigenschaften'.....	1055
Eigenschaft 'Texteigenschaften'.....	1056
Eigenschaft 'Texte'.....	1056
Eigenschaft 'Beschriftung der Skalenstriche'.....	1056
Eigenschaft 'Zeiten'.....	1057
20.5 Visualisierungsmanager mit Clients.....	1058
20.5.1 Visualisierungsmanager.....	1058
'Visualisierungsmanager' - 'Einstellungen'.....	1058
'Visualisierungsmanager' - 'Standardtastaturkonfiguration'.....	1060
'Visualisierungsmanager' - 'Visualisierungen'.....	1060

	Seite
Registerkarte 'Schriftarteneinstellung'.....	1060
20.5.2 Die Clients und ihre Editoren.....	1061
Diagnosevisualisierung.....	1061
20.6 Dialog 'Eigenschaften' - 'Visualisierungsobjekt'.....	1063
20.6.1 Anwendungsfälle.....	1064
Eingabedialoge.....	1064
Globaler Dialog.....	1069
Programmatischer Zugriff auf Eingabedialoge.....	1070
Ereignisse und Eingabeaktionen bemerken.....	1074
20.7 Visualisierungsbibliotheken.....	1078
20.8 Optionen.....	1079
20.8.1 Dialog 'Optionen' - 'Visualisierung'.....	1079
20.8.2 Dialog 'Optionen' - 'Visualisierungsstile'.....	1079
20.9 Visualisierungsprofil.....	1080
20.10 Visualisierungsstil.....	1080
20.11 Tastaturbedienung im Online-Modus.....	1081
 21 Service und Support.....	 1083
 Index.....	 1085

1 Zu dieser Dokumentation

Ausgaben dieser Dokumentation

Ausgabe	Stand	Bemerkung
01	2020-11	Erstausgabe zur Version 15VRS

Tab. 1-1: Änderungsverlauf

1.1 Gültigkeit der Dokumentation

Zielgruppe Diese Dokumentation richtet sich an Anwender, die die SPS-Komponente einer Steuerung des Systems MLC, MLD oder MTX programmieren und in Betrieb nehmen, oder durch eine intern genutzte Visualisierung unterstützen wollen.

Erforderliche Qualifikationen: Person, die aufgrund ihrer fachlichen Ausbildung, Kenntnisse und Erfahrungen sowie Kenntnis der einschlägigen Normen und Bestimmungen die ihr übertragenen Arbeiten beurteilen und mögliche Gefahren erkennen kann.

Die Dokumentation beschreibt das SPS-Programmierwerkzeug IndraLogic 2G und seine Verwendung.

Das kostenfreie IndraLogic ist eine IEC 61131-3 Programmierplattform für Automatisierungsgeräte mit Steuerungsaufgaben. Sie stellt Ihnen vielfältige und komfortable Engineering-Funktionen für Ihre Entwicklungsarbeit zur Verfügung:

	Siehe hierzu
Projektkonfiguration durch Assistenten (Wizards)	Kap. 2 "Projekt anlegen und konfigurieren" auf Seite 7
Erzeugung von professionellen Steuerungsapplikationen nach IEC 61131-3 mit vielen Standardfunktionen	Kap. 9 "Applikation programmieren" auf Seite 85
Komfortable Programmierung mit Maus und Tastatur in allen Sprachen der IEC 61131-3 Entsprechende Editoren für FUP, KOP, AWL, ST, AS, dazu die Varianten CFC und Erweiterter CFC	Kap. 17.1 "Programmiersprachen und ihre Editoren" auf Seite 239 Kap. 17 "Referenz Programmierung" auf Seite 239 Kap. 18 "Referenz Benutzeroberfläche" auf Seite 507
Eingabeunterstützung für die Eingabe und Konfiguration unterschiedlichster Daten	Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112
Unterstützung objektorientierter Programmierung Echte objektorientierte Programmierung nach 61131-3 3rd Edition in allen Sprachen der IEC 61131-3 ohne zusätzliche Tools möglich Vererbung von Programmbausteinen auf ähnliche Applikationsteile zur Reduktion von Entwicklungszeit und Fehlern Objektorientierte Programmierung ist kein Muss: Funktionale oder objektorientierte Programmierung sind beliebig einsetzbar und mischbar	Kap. 9.18 "Objektorientiert programmieren" auf Seite 154
Umfassender Vergleich von SPS-Objekten, auch für grafische Editoren Umfassender Online-Vergleich von SPS-Objekten	Kap. 4 "Vergleich von SPS-Objekten" auf Seite 17 Kap. 5 "Online-Vergleich von SPS-Objekten" auf Seite 51

	Siehe hierzu
Debugging- und Online-Eigenschaften zur Optimierung des Applikationscodes und zur Beschleunigung von Test und Inbetriebnahme	Kap. 12 "Testen und Fehler beheben" auf Seite 193 Befehlsüberblick Kap. 18.2.5 "Menü 'Debuggen'" auf Seite 652
Integrierte Compiler für viele CPU-Plattformen zur Optimierung der Performance der Steuerung	Kap. "Dialog 'Projekteinstellungen- 'Compileroptionen'" auf Seite 759
Security-Eigenschaften zur Absicherung des Quellcodes und des Betriebs der Steuerung	Kap. 6 "Projekt schützen und speichern" auf Seite 57 Kap. 18.1.28 "Objekt 'Zugriffskontrolle'" auf Seite 608
Feldbusunterstützung und Programmierung von Geräten	Kap. 8 "E/A-Anbindung konfigurieren" auf Seite 71
Bibliothekskonzept zur einfachen Wiederverwendung von Applikationscode	Kap. 16 "Bibliotheken erzeugen und verwalten" auf Seite 235
Diagnosevisualisierung	Kap. 20.1 "Visualisierung im Überblick" auf Seite 803

Kap. 21 "Service und Support" auf Seite 1083 enthält Informationen zu unserem Kundendienst-Helpdesk.

Diese Dokumentation unterstützt den Anwender in den Phasen

- Programmierung des SPS-Programms,
- Debugging und Inbetriebnahme,
- Visualisierung.

1.2 Gebrauch der Sicherheitshinweise

1.2.1 Aufbau der Sicherheitshinweise

Die Sicherheitshinweise sind wie folgt aufgebaut:

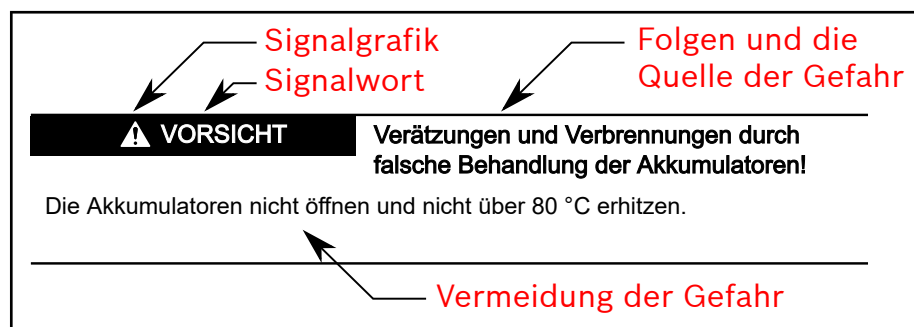


Abb. 1-1: Aufbau der Sicherheitshinweise

1.2.2 Erläuterung der Signalwörter und der Signalgrafik

Die Sicherheitshinweise in der vorliegenden Dokumentation beinhalten bestimmte Signalwörter (Gefahr, Warnung, Vorsicht, Hinweis) und gegebenenfalls eine Signalgrafik (nach ANSI Z535.6-2006).

Das Signalwort soll die Aufmerksamkeit auf den Sicherheitshinweis lenken und bezeichnet die Schwere der Gefährdung.

Die Signalgrafik (Warndreieck mit Ausrufezeichen), welche den Signalwörtern Gefahr, Warnung und Vorsicht vorangestellt wird, weist auf Gefährdungen für Personen hin.

⚠ GEFAHR

Bei Nichtbeachtung dieses Sicherheitshinweises **werden** Tod oder schwere Körpverletzung eintreten.

⚠ WARNUNG

Bei Nichtbeachtung dieses Sicherheitshinweises **können** Tod oder schwere Körpverletzung eintreten.

⚠ VORSICHT

Bei Nichtbeachtung dieses Sicherheitshinweises können mittelschwere oder leichte Körpverletzung eintreten.

HINWEIS

Bei Nichtbeachtung dieses Sicherheitshinweises können Sachschäden eintreten.

1.2.3 Verwendete Symbole

Fingerzeige werden wie folgt dargestellt:



Dies ist ein Hinweis.

Tipps werden wie folgt dargestellt:



Dies ist ein Tipp.

1.2.4 Erläuterung der Signalgrafik auf dem Gerät



Wenn dieses Symbol am Gerät angebracht ist, beachten Sie unbedingt die Dokumentation zu dem Gerät. In der jeweiligen Dokumentation finden Sie die Art der Gefährdung sowie die notwendigen Schritte zur Vermeidung der Gefährdung.

1.3 Erforderliche und ergänzende Dokumentationen

Dokumentationstitel mit Typenschlüssel und Materialnummer

IndraWorks 15VRS Software-Installation DOK-IWORKS-SOFTINS*V15-CORS-DE-P, R911393449 Die Dokumentation beschreibt den Installationsprozess von IndraWorks.
IndraWorks 15VRS Engineering DOK-IWORKS-ENGINEE*V15-APRS-DE-P, R911393302 Diese Dokumentation beschreibt die Verwendung von IndraWorks, in das sich die Rexroth Engineering-Tools integrieren. Beschrieben werden das Arbeiten mit IndraWorks und die Bedienung der Oszilloskop-Funktion.
IndraWorks 15VRS Feldbusse DOK-IWORKS-FB*****V15-APRS-DE-P, R911393282 Die Dokumentation dient der Beschreibung der von den Systemen MLC und MTX unterstützten Feldbus- und lokalen Peripherieanschlüssen. Der Schwerpunkt dieser Dokumentation liegt in der Konfiguration, Parametrierung, Inbetriebnahme und Diagnose der verschiedenen Peripherieanschlüssen. Sie ist die Grundlage für die Online-Hilfe.
IndraWorks 15VRS Basis-Bibliotheken IndraLogic 2G DOK-IL*2G*-BASLIB**V15-LIRS-DE-P, R911398632 Das vorliegende Handbuch beschreibt die systemübergreifenden SPS-Bibliotheken.
IndraWorks 15VRS Feldbus-Bibliotheken DOK-IWORKS-FB*LIB**V15-LIRS-DE-P, R911393274 Das vorliegende Handbuch beschreibt die Bibliotheken der Feldbusse: RIL_ProfibusDP_02, RIL_ProfibusDP Slave, RIL_ProfinetIO, RIL_ProfinetIODevice, RIL_EtherNetIPAdapter, RIL_MappingList, RIL_SERCOSIII, RIL_Inline, einschließlich deren Diagnosen, sowie die Fehlerreaktion der Bausteine.
IndraMotion MLC 15VRS PLCopen-Bibliotheken DOK-MLC***-FUNLIB**V15-LIRS-DE-P, R911393306 Diese Dokumentation beschreibt die Funktionsbausteine, Funktionen und Datentypen der Bibliotheken RIL_CommonTypes, ML_Base und ML_PLCopen für IndraMotion MLC. Die Dokumentation enthält ferner die Fehlerreaktion der Bausteine.

IndraMotion MLC 15VRS HMI-Anbindung DOK-MLC***-HMI*****V15-APRS-DE-P, R911399269 Dieses Dokument beschreibt die, vom System MLC unterstützten Visualisierungssysteme sowie deren Anbindung.
Rexroth IndraMotion MLC 15VRS Erste Schritte DOK-MLC***-F*STEP**V15-CORS-DE-P, R911398630 Diese Dokumentation beschreibt die "ersten Schritte" mit der IndraMotion MLC und RobotControl. Beschrieben werden die Hard- und Softwarevoraussetzungen und das Erstellen eines Projekts.

Tab. 1-2: Dokumentationsübersicht

1.4 Bezeichnungen und Abkürzungen

Begriff	Erklärung
AS (SFC)	Strukturierungsmittel "Ablaufsprache" = Sequential Function Chart
AWL (IL)	SPS-Programmiersprache "Anweisungsliste" = Instruction List
CFC	SPS-Programmiersprache "Continuous Function Chart"
DUT	Datentyp
EN/ENO	boolescher Eingang (ENable) EN / boolescher Ausgang ENO (ENable Out)
FUP (FBD)	SPS-Programmiersprache "Funktionsplan" = Function Block Diagram
GNVL	Globale Netzwerkvariablenliste
GVL	Globale Variablenliste
IEC	International Electrotechnical Commission (Normungsgremium für Elektrotechnik)
IndraWorks Engineering Framework	Projektierungs- und Inbetriebnahmewerkzeug von Bosch Rexroth
KOP (LD)	SPS-Programmiersprache "Kontaktplan" = Ladder Diagram
POE (POU)	"Programmorganisationseinheit" = Program Organization Unit

Tab. 1-3: Verwendete Bezeichnungen und Abkürzungen

2 Projekt anlegen und konfigurieren

Was ist ein Projekt?

- Ein Projekt enthält die Objekte, die zur Erstellung eines Steuerungsprogramms ("Applikation") nötig sind:
 - Reine Programmierbausteine, beispielsweise Programme, Funktionsbausteine, Funktionen, GVLs.
 - Objekte, die zusätzlich benötigt werden, um die Applikation auf einer SPS ausführen zu können, beispielsweise Task-Konfiguration, Bibliotheksverwalter, Symbolkonfiguration, Gerätekonfiguration, Visualisierungen, externe Dateien.
- In einem Projekt können Sie mehrere Steuerungsgeräte mit je einer Applikation programmieren und anbinden.
- Applikationsspezifische Bausteine verwaltet IndraLogic im Projektexplorer unterhalb der "Applikation".
Projektweit verfügbare Bausteine verwaltet IndraLogic im Projektexplorer im Ordner "Allgemeiner Bausteinteil".
- Für das Erstellen von Projekten gibt es Vorlagen, die bereits bestimmte Objekte enthalten.
- In den **Projekteinstellungen** und **Bibliotheksinformationen** sind Grundkonfigurationen und Informationen zum Projekt definiert. Beispielsweise:
 - Compiler-Einstellungen
 - Benutzerverwaltung
 - Autor
 - Daten zur Projektdatei.

In den Konfigurationsdialogen zur **Projektumgebung** gibt es Einstellungen zur Versionskompatibilität des Projekts.

- Sie legen ein Projekt als Datei im Dateisystem ab. Optional können Sie es zusammen mit projektrelevanten Dateien und Informationen in einem Projektarchiv verpacken. Möglich ist auch die Ablage in einem Quellcodeverwaltungssystem wie SVN.
- Jedes Projekt enthält die Information, mit welcher IndraLogic-Version es erstellt wurde. Wenn Sie es in einer anderen Version öffnen, weist IndraLogic auf mögliche oder nötige Aktualisierungen bezüglich Speicherformat, Bibliotheksversionen etc. hin.
- Sie können Projekte vergleichen, exportieren, importieren und eine Dokumentation dazu erzeugen lassen.
- Sie können ein Projekt gegen Veränderung und komplett, auch gegen Lesen schützen. Durch das Verwenden einer Benutzerverwaltung können Sie den Zugriff auf das Projekt und sogar auf einzelne Objekte im Projekt gezielt steuern.

Siehe auch

- [Kap. 8.1 "Projektexplorer und Geräteeditor" auf Seite 71](#)
- [Kap. 18.1.2 "Objekt 'Applikation'" auf Seite 508](#)
- [Kap. 18.1.17 "Objekt 'POU'" auf Seite 552](#)
- [Kap. 18.3.7 "Dialog 'Projekteinstellungen'" auf Seite 758](#)
- [Kap. "Befehl 'Bibliotheksinformationen'" auf Seite 719](#)
- [Kap. 6 "Projekt schützen und speichern" auf Seite 57](#)

2.1 'Leeres Projekt' anlegen

1. Wählen Sie Befehl **Datei ► Neues Projekt** und dann Vorlage **Allgemeine Vorlagen ► Leeres Projekt**. Geben Sie einen Namen, hier als Beispiel `myProject`, und einen Ablageort im Dateisystem ein.

Das Projekt `myProject` erscheint im Projektextplorer.

2. Wählen Sie z. B. im Kontextmenü des Projektes **Hinzufügen ► IndraMotion MLC ► IndraControl L65**. Weitere erforderliche Informationen werden durch die Dialoge des Wizards "<steuerung> einfügen" abgefragt.

In der Titelleiste des Programmiersystem-Hauptfensters erscheint der Projektname **MyProject**.

Im Projektextplorer ist folgende Struktur angelegt:

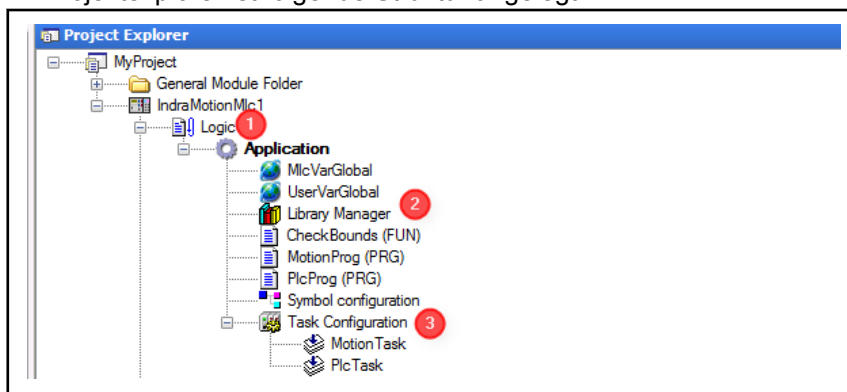


Abb. 2-1:

Je nach ausgewählter Steuerung können auch noch zusätzliche Geräte-knoten erscheinen.

Die SPS erscheint immer automatisch mit einem Unterknoten (**Logic** (1)).

Die Basisobjekte darunter sind: eine Applikation mit einem **Bibliotheks-verwalter** (2) und einer **Taskkonfiguration** (3).

Die Task-Konfiguration definiert die Haupt-Task (**PlcTask**) für die Abarbeitung des Standard-Programmbausteins **PlcProg**.

Der Bibliotheksverwalter enthält bereits einen Satz Bibliotheken (abhängig von der ausgewählten Steuerung), die nachher zur Programmierung verwendet werden können.

Wenn Sie nun **PlcProg** mit fehlerfreiem Code füllen, können Sie diesen ohne weitere Programmierobjekte zu benötigen auf die Steuerung laden und zur Ausführung bringen.

Siehe auch

- [Kap. 8.1 "Projektextplorer und Geräteeditor" auf Seite 71](#)
- [Kap. 2.2 "Objekte hinzufügen" auf Seite 8](#)
- [Kap. 9 "Applikation programmieren" auf Seite 85](#)
- [Kap. 8 "E/A-Anbindung konfigurieren" auf Seite 71](#)
- [Kap. 9.7 "Bibliotheksbausteine verwenden" auf Seite 116](#)

2.2 Objekte hinzufügen

Voraussetzung: Ein Projekt ist geöffnet.

Siehe

- [Kap. 2.1 "Leeres Projektanlegen" auf Seite 8](#)

Die folgende Anleitung zeigt einige Möglichkeiten beim Anlegen von Objekten im Projekt.

1. Selektieren Sie einen Eintrag im Projektextplorer oder im Ordner "Allgemeiner Baustein", beispielsweise das Objekt **Application** im Projektextplorer.
2. Wählen Sie den Kontextmenü-Befehl **Hinzufügen**.
Abhängig vom ausgewählten Eintrag im Explorer bietet IndraLogic die passenden Objekte zur Auswahl an.
3. Wählen Sie beispielsweise das Objekt **POU** und im daraufhin erscheinenden Dialog **Objekt Hinzufügen-Assistent** den Typ **Programm** mit Implementierungssprache **Strukturierter Text (ST)** und Namen "prog".
Nach Klicken auf **Fertigstellen** fügt IndraLogic ein Programm-Objekt **prog** im Projektextplorer unterhalb von **Application** ein. Das Objekt ist damit applikationsweit bekannt.
4. Selektieren Sie ein Objekt im Explorer und wählen im Kontextmenü den Befehl **Eigenschaften**.
Der Dialog **Eigenschaften** erscheint mit den objektrelevanten Kategorien. Wenn Sie eine Benutzerverwaltung verwenden, könnten Sie hier beispielsweise den Zugriff auf das Objekt einschränken.
5. Selektieren Sie einen Eintrag im Explorer, unterhalb dessen Sie einen Ordner anlegen möchten, um in ihm bestimmte Objekte zu sammeln.



Die Anordnung von Geräteknoten und Geräteobjekten können Sie nicht durch selbst angelegte Ordner strukturieren.

6. Wählen Sie im Kontextmenü des Objekts **Application** den Befehl **Ordner hinzufügen** und definieren im erscheinenden Dialog einen Namen für den neuen Ordner. Klicken Sie **OK**.
Der Ordner erscheint im Explorer.
7. Selektieren Sie ein Objekt im Baum und verschieben Sie es durch Ziehen mit der Maus innerhalb des Baums an eine andere Position, beispielsweise in den Ordner.
8. Selektieren Sie den Ordner "Allgemeiner Baustein" im Projektextplorer. Wählen Sie Befehl **Hinzufügen** und dann beispielsweise das Objekt **Textliste**.
Das Objekt **Textliste** wird in den Ordner "Allgemeiner Baustein" eingefügt! Das Objekt ist damit projektglobal bekannt.

Siehe auch

- [Kap. 2.1 "Leeres Projektanlegen" auf Seite 8](#)
- [Kap. "Befehl 'Hinzufügen'" auf Seite 643](#)
- [Kap. "Befehl 'Ordner hinzufügen'" auf Seite 644](#)
- [Kap. "Befehl 'Eigenschaften'" auf Seite 634](#)

2.3 SPS-Compilerversion ändern

Arbeiten mit IndraWorks,
Kompatibilitätsmodus

Projekt ► Kompatibilitätsmodus ändern...



Ab IndraWorks Version 15 werden im Kompatibilitätsmodus keine Vorgängerversionen <15VRS unterstützt!

Projekte aus Vorgängerversionen müssen konvertiert werden, um in Version 15 genutzt werden zu können.

Bei der Konvertierung kann es eventuell erforderlich sein, dass der Steuerungstyp geändert und die Projektierung angepasst werden muss.

Beachten Sie hierzu bitte die weiterführenden Dokumentationen:

Dokumentationen im Rexroth Medienverzeichnis:

- System MLC Projektkonvertierung: DOK-MLC***-PRO-CONV*V15-APRS-DE-P
- System MTX Projektkonvertierung: DOK-MTX***-PRO-CONV****-PRRS-DE-P

SPS-Compilerversion

Ausgehend vom eingestellten Kompatibilitätsmodus kann ein SPS-Compiler zur Erzeugung des Projektcodes gewählt werden.

Die Version des Compilers, mit welcher der aktuelle Projektcode für die Verwendung auf dem Zielgerät erzeugt wird, ist unter **Projekt ► SPS-Compilerversion** definiert.

Im Dialogfenster wird die Compilerversion in Zuordnung zur IndraWorks-Version angezeigt.



Wenn Sie ein Projekt öffnen, in dem nicht die neueste Compilerversion eingestellt ist, erscheint der Dialog **Projektumgebung** mit einem entsprechenden Hinweis und der Möglichkeit, direkt zu aktualisieren.

Voraussetzung: Ein Projekt ist geöffnet.

Im Ordner "Allgemeiner Bausteinteil", unter **Projekteinstellungen ► Compileroptionen**, wird die Compilerversion (nicht änderbar) angezeigt.

Siehe auch

- [Kap. 11.2 "Applikationscode erzeugen" auf Seite 187](#)
- [Kap. "Dialog 'Projekteinstellungen- 'Compileroptionen'" auf Seite 759](#)

2.4 IndraLogic 2G-Projekt öffnen

Voraussetzung: IndraLogic 2G ist gestartet, es kann auch bereits ein Projekt geöffnet sein.

1. Wählen Sie Befehl **Datei ► Projekt öffnen**.
2. Wählen Sie im Dialog **Projekt öffnen** das gewünschte IndraLogic 2G-Projekt oder -Projektarchiv aus dem Dateisystem. In der rechten unteren Ecke des Dialogs gibt es dazu einen Dateifilter.



Bei Bedarf können Sie das Projekt "Schreibgeschützt öffnen", um zu verhindern, dass die bestehende Projektdatei nachher versehentlich mit Änderungen überschrieben wird.

Wenn kein anderes Projekt bereits geöffnet ist, sind folgende Fälle möglich:

- Sie haben ein Projekt ausgewählt, das mit einer neueren IndraLogic-Version gespeichert wurde:

Ein solches Projekt enthält eventuell Daten, welche nicht geladen werden können. Sie können das Projekt dennoch öffnen, müssen jedoch für diesen Fall Folgendes beachten: Das Projekt kann unerwartetes Verhalten zeigen, weil es nicht vollständig interpretiert werden kann.

Objekte, welche IndraLogic nicht oder nur unvollständig laden konnte, werden im Projektextplorer rot und mit dem Text "[unbekannt]" oder "[unvollständig]" gekennzeichnet.

Objekte, die der aktuellen Version unbekannt sind, kann IndraLogic nicht im Editor anzeigen.

Unvollständige Objekte zeigt IndraLogic im Editor mit einer Warnung an, dass der angezeigte Inhalt möglicherweise nicht dem Original entspricht.

Unvollständige Projekte kann IndraLogic nicht unter ihrem Originalnamen abspeichern. Dies wird durch einen Schreibschutz-Hinweis in der oberen rechten Ecke angezeigt. Sie können die Datei unter einem anderen Namen abspeichern.

- Sie haben ein Projekt ausgewählt, nach dessen letzter Änderung sich IndraLogic nicht regulär beendet hatte, aber die Projektoption **Automatisch Speichern** aktiviert war: Sie erhalten nun den Dialog **Auto Save Backup** zur Handhabung der Sicherungskopie.
- Sie haben ein Projekt ausgewählt, das schreibgeschützt ist: Sie können nun das Projekt in schreibgeschütztem Modus öffnen oder den Schreibschutz entfernen.
- Sie haben ein Projekt ausgewählt, das mit einem Passwort geschützt ist. Sie werden nun zur Eingabe des Passworts aufgefordert.

Der Dialog **Projektumgebung** erscheint.

3. Prüfen Sie die im Dialog **Projektumgebung** angebotenen Aktualisierungen. Bei Projekten, die mit sehr alten IndraLogic 2G-Versionen erstellt wurden, erhalten Sie eventuell zusätzlich Hinweise, wie Sie bestimmte Objekte ersetzen müssen, um das Projekt weiter verwenden zu können. Je nachdem, ob Sie das Projekt belassen wollen, wie es ist, oder ob sie es unter möglichst aktuellen Bedingungen weiterentwickeln wollen, führen Sie die Aktualisierungen durch oder nicht.

Jede Aktualisierung bewirkt, dass ein Einloggen ohne Download/Online-Change nicht mehr möglich ist.

Das Projekt wird geöffnet.

Siehe auch

- [Kap. 6.1 "Schreibschutz einrichten" auf Seite 60](#)

2.5 Projekt konfigurieren

Sie können Ihr IndraLogic-Projekt in den folgenden Dialogen konfigurieren:

- **Projekteinstellungen:** Allgemeine Einstellungen zum Verhalten von Editoren, zum Compiler, zur Benutzerverwaltung etc.

- **Bibliotheksinformationen:** Ausstatten des Projekts mit individuellen kennzeichnenden Informationen

Siehe auch

- [Kap. 18.3.7 "Dialog 'Projekteinstellungen'" auf Seite 758](#)
- [Kap. 18.1.20 "Objekt 'Bibliotheksinfo'" auf Seite 579](#)

2.5.1 Bibliotheksinfo abfragen und bearbeiten

Sie können Informationen, die Ihr Projekt und die dazugehörige Datei betreffen, mit Hilfe des Objekts **Bibliotheksinfo** abfragen und bestimmte Informationen bearbeiten.

Das Objekt enthält Informationen über

- Dateiattribute
- Metainformationen wie Herstellerfirma, Titel oder Autor
- Eigenschaften mit Schlüssel
- Statistik
- signierte Bibliotheken

IndraLogic speichert die Bibliotheksinfo als Objekt direkt im Projekt. Wenn Sie dann ein Projekt auf ein anderes System übertragen, wird auch das Objekt **Bibliotheksinfo** mit übertragen, ohne dass dafür ein Projektarchiv benötigt wird.

Sie können mit Hilfe von Eigenschaftenschlüsseln über Funktionen von extern auf die Bibliotheksinfo zugreifen. Wenn es sich um ein Bibliotheksprojekt handelt, können Sie zusätzlich Informationen zur Lizenzierung abfragen und die Datei signieren.

Siehe auch

- [Kap. 18.1.20 "Objekt 'Bibliotheksinfo'" auf Seite 579](#)

Metainformation bearbeiten

1. Voraussetzung für die hier beschriebenen Schritte ist, dass das Objekt "Bibliotheksinfo" zuvor im Projekt hinzugefügt wurde!
Öffnen Sie im Projektextplorer den Ordner "Allgemeiner Baustein" und öffnen Sie darin das Objekt "Bibliotheksinfo".
Der Dialog **Bibliotheksinfo** öffnet sich.

2. Wählen Sie Registerkarte **Zusammenfassung**.
3. Geben Sie in den Eingabefelder Ihre Daten ein, zum Beispiel 0.0.0.1 im Feld **Version**.

IndraLogic erstellt für jeden eingegebenen Wert eine Eigenschaft mit Schlüssel und verwaltet sie unter der Registerkarte **Eigenschaft**. Wenn es sich um ein Bibliotheksprojekt handelt, verwendet IndraLogic die Eigenschaften weiter und sortiert im Bibliotheks-Repository danach.

Wenn Sie aktivieren, dass IndraLogic für diese Eigenschaften jeweils eine Funktion erstellt, können Sie auf die Eigenschaften programmatisch zugreifen.

Funktionen für Eigenschaftenzugriff erstellen

1. Voraussetzung für die hier beschriebenen Schritte ist, dass das Objekt "Bibliotheksinfo" zuvor im Projekt hinzugefügt wurde!
Öffnen Sie im Projektextplorer den Ordner "Allgemeiner Baustein" und öffnen Sie darin das Objekt "Bibliotheksinfo".

Der Dialog **Bibliotheksinfo** öffnet sich.

2. Aktivieren Sie die Option **Automatisch POU's für den Eigenschaftenzugriff erzeugen**.

Voraussetzung: Die folgende Eigenschaft ist definiert.

Beispiel

```
Schlüssel = nProp1
Typ= Zahl
Wert= 333
```

1. Aktivieren Sie die Option **Automatisch POU's für den Eigenschaftenzugriff erzeugen**.
2. Deklarieren Sie eine Variable des Typs `DINT`, zum Beispiel `showprop : DINT;`
3. Rufen Sie die Funktion `GetNumberProperty` auf: `showprop := GetNumberProperty("nProp1");`

Sie erhalten Zugriff auf den Wert in der Applikation.

Bibliotheksprojekt signieren

Voraussetzung: Das Projekt ist ein Bibliotheksprojekt. Es steht Ihnen eine private Schlüsseldatei mit einem zugehörigen Token zur Verfügung.

1. Voraussetzung für die hier beschriebenen Schritte ist, dass das Objekt "Bibliotheksinfo" zuvor im Projekt hinzugefügt wurde!

Öffnen Sie im Projektextplorer den Ordner "Allgemeiner Baustein" und öffnen Sie darin das Objekt "Bibliotheksinfo".

Der Dialog **Bibliotheksinfo** öffnet sich.

2. Wählen Sie Registerkarte **Signieren**.
3. Aktivieren Sie die Option **Signieren aktivieren**.
4. Geben Sie den Speicherort Ihrer privaten Schlüsseldatei an.
IndraLogic trägt den Token automatisch in **Token des öffentlichen Schlüssels** ein, zum Beispiel `427A5701DA3CF3CF`.
5. Speichern Sie das Projekt als **Übersetzte Bibliothek**.

Sie sehen das Token im Bibliotheks-Repository oder Bibliotheksverwalter in den Details zur Bibliothek und können es mit dem veröffentlichten Token des Bibliotheksherstellers vergleichen. Sie überprüfen somit, ob die Bibliothek tatsächlich vom gewünschten Hersteller stammt. Außerdem passt eine auf Binärebene gefälschte Bibliothek nicht mehr zur Signatur und wird deshalb nicht geladen. Sie können sich somit auf die Integrität der Bibliothek verlassen.

Private Schlüsseldatei erstellen

1. Voraussetzung für die hier beschriebenen Schritte ist, dass das Objekt "Bibliotheksinfo" zuvor im Projekt hinzugefügt wurde!

Öffnen Sie im Projektextplorer den Ordner "Allgemeiner Baustein" und öffnen Sie darin das Objekt "Bibliotheksinfo".

Der Dialog **Bibliotheksinfo** öffnet sich.

2. Wählen Sie die Registerkarte **Signieren**.
3. Wählen Sie die Schaltfläche **Private Schlüsseldatei erzeugen**.

Der Dialog **Private Schlüsseldatei erstellen** erscheint.

4. Wählen Sie einen sichern Ablageort, zum Beispiel `D:\for lib developers only\mycomp_libkey.libpk` und beenden Sie den Dialog mit **Speichern**.

Siehe auch

- [Kap. 18.1.20 "Objekt 'Bibliotheksinfo'" auf Seite 579](#)

2.5.2 Projekteinstellungen vornehmen

Sie können Einstellungen, die das Verhalten von IndraLogic und das bestimmter Editoren betreffen, im Objekt **Projekteinstellungen** konfigurieren.

Die Einstellungen gelten projektweit und werden für aktive Editoren sofort angewendet.

Sie erreichen die Dialoge des Objekts über **Ordner "Allgemeiner Baustein-
teil" ► Projekteinstellungen**.

IndraLogic speichert die Projekteinstellungen als Objekt direkt im Projekt. Wenn Sie dann ein Projekt auf ein anderes System übertragen, wird auch das Objekt **Projekteinstellungen** mit übertragen, ohne dass dafür ein Projektarchiv benötigt wird.

Siehe auch

- [Kap. "Befehl 'Projekteinstellungen'" auf Seite 634](#)
- [Kap. "Dialog 'Projekteinstellungen- 'AS'" auf Seite 758](#)
- [Kap. "Dialog 'Projekteinstellungen- 'Compileroptionen'" auf Seite 759](#)
- [Kap. "Dialog 'Projekteinstellungen- 'Compilerwarnungen'" auf Seite 760](#)
- [Kap. "Dialog 'Projekteinstellungen- 'Static Analysis Light'" auf Seite 761](#)
- [Kap. "Dialog 'Projekteinstellungen- 'Visualisierung'" auf Seite 763](#)

3 Projekt exportieren und transferieren

Für den Austausch der Daten von IndraLogic-Projekten mit anderen Programmen stehen Ihnen Export- und Importfunktionen zur Verfügung.

Ein Austausch von IndraLogic-Projekten zwischen IndraLogic-Entwicklungssystemen erfolgt durch eine Kopie der Exportdatei (*.iwx) oder einer gezippten Exportdatei (*.iwxz).

3.1 Projekt transferieren

Wenn Sie ein Projekt auf einen anderen Rechner übertragen und von dort zur selben SPS verbinden wollen, ohne dass ein Online Change oder Download erforderlich ist, beachten Sie folgende Punkte.

- Stellen Sie sicher, dass das Projekt nur feste Versionen von Bibliotheken (Ausnahme Interface-Bibliotheken), Visualisierungsprofil und Compiler verlangt
- Stellen Sie sicher, dass die Bootapplikation aktuell ist.

Erzeugen Sie dann ein Projektarchiv, das Sie auf dem anderen Rechner entpacken.

Übertragen eines Projekts auf anderes System

Voraussetzung: Auf Rechner "PC1" ist ein Projekt geöffnet, das Sie auf einen anderen Rechner "PC2" transferieren und von dort wieder mit der selben Steuerung verbinden.

1. Stellen Sie sicher, dass nur Bibliotheken mit festen Versionen im Projekt eingebunden sind, mit Ausnahme reiner Schnittstellenbibliotheken. Öffnen Sie dazu den **Bibliotheksverwalter** und prüfen Sie alle Einträge, die ein "*" anstelle einer festen Versionsangabe haben.
2. Stellen Sie sicher, dass eine feste Compiler-Version in den Projekteinstellungen gesetzt ist. Zur Prüfung wählen Sie **Ordner "Allgemeiner Bausteinteil" ► Projekteinstellungen** und Kategorie **Compiler-Optionen**.
3. Stellen Sie sicher, dass die Applikation, die gerade geöffnet ist, die gleiche ist wie die, die gerade auf der SPS verwendet wird. Das heißt, die "Bootapplikation" muss identisch sein mit dem Projekt im Programmiersystem. Prüfen Sie dazu den Projektnamen in der Titelleiste des Programmiersystemfensters: Wenn ein Sternchen hinter dem Namen angezeigt wird, wurde das Projekt geändert, aber noch nicht gespeichert. Dann stimmen möglicherweise Applikation und Bootapplikation nicht überein!

Erzeugen Sie in diesem Fall zuerst eine (neue) Bootapplikation. Es hängt von der SPS und den Applikationseigenschaften ab, ob dies automatisch beim Laden (Download) der Applikation geschieht. Zum expliziten Erzeugen wählen Sie Befehl **Debuggen ► Bootapplikation erzeugen**. Führen Sie dann einen Download mit Hilfe der Befehle **Debuggen ► Einloggen** und **Debuggen ► Laden** durch.

Starten Sie danach mit Befehl **Debuggen ► Start** die Applikation auf der Steuerung.

Nun läuft die gewünschte Applikation auf der SPS, zu der Sie sich später auf "PC2" vom selben Projekt aus wieder verbinden möchten.

4. Erzeugen Sie ein Projektarchiv: Wählen Sie **Projekt ► Archivieren**.

Im Dialog **Projekt archivieren** wählen Sie auch die folgenden Informationen mit aus:

- **Programmcode ins Archive aufnehmen**
- **Standardbibliotheken ins Archive aufnehmen**

Speichern Sie das Projektarchiv an einem von PC2 zugänglichen Ort.

5. Loggen Sie aus der Steuerung aus: Wählen Sie dazu **Debuggen ► Ausloggen**. Sie können die SPS ohne Bedenken stoppen und neu starten, bevor Sie von "PC2" erneut verbinden.
6. Extrahieren Sie das Projektarchiv auf "PC2": Wählen Sie **Projekt ► Wiederherstellen**, und öffnen Sie das oben gespeicherte Archiv. Aktivieren Sie im Dialog **Projektarchiv extrahieren** die gleichen Informationen wie oben beim Erzeugen des Archivs beschrieben.
7. Öffnen Sie das Projekt und loggen Sie sich wieder auf SPS "xy" ein.
IndraLogic fordert keinen Online Change oder Download, das Projekt läuft.

Siehe auch

- [Kap. "Dialog 'Projekteinstellungen- 'Compileroptionen'" auf Seite 759](#)
- [Kap. "Befehl 'Bootapplikation erzeugen'" auf Seite 656](#)

4 Vergleich von SPS-Objekten

Der Vergleich von SPS-Objekten erlaubt:

1. SPS-Objekte des **aktuellen** Projektes miteinander zu vergleichen,
2. SPS-Objekte des **aktuellen** Projektes mit denen einer Export-Datei (*.iwx,*.iwx) zu vergleichen,
3. SPS-Objekte zweier Export-Dateien untereinander zu vergleichen.

Im ersten Fall sind die gewünschten Objekte im Projekt-Explorer zu markieren.

Im zweiten Fall sind das gewünschte Objekt im Projekt-Explorer zu markieren und im Kontext-Menü des Objektes "Vergleichen..." zu wählen. Im dann erscheinende Fenster ist die gewünschte Exportdatei zu übernehmen.

Ist der Vergleich zweier Export-Dateien beabsichtigt, so der Menüpunkt über **Extras ► Vergleichen...** zu erreichen.

Beim SPS-Projektvergleich -wie auch beim Vergleich von SPS-Objekten- können Sie:

- Leerzeichen ignorieren, berücksichtigt werden dann nur einzelne semantisch relevante Leerzeichen
- Kommentare ignorieren, nur programmrelevanter Text wird verglichen
- Eigenschaften von SPS-Objekte ignorieren

Siehe auch: [Kap. 5.3 "Optionen" auf Seite 54.](#)

Die Beschreibung erfolgt anhand der Kombination Projekt/Export-Datei, sie gilt sinngemäß für Export-Datei/Export-Datei.

Ist der Vergleich infolge unterschiedlicher Objekttypen unmöglich, erscheint eine Fehlermeldung.

Grundlagen des Vergleiches im IndraWorks Engineering:

- Rexroth IndraWorks 15VRS Engineering,
DOK-IWORKS-ENGINEE*V15-APRS-DE-P, R911393302

Vergleich von SPS-Objekten

- [Allgemeiner Bausteinteil, Seite 18,](#)
- [Logic, Seite 19,](#)
- [Application, Seite 20,](#)
- [Taskkonfiguration, Seite 23,](#)
- [Task, Seite 18,](#)
- [Ordner \(Unterschied zu IndraWorks Ordner\), Seite 24,](#)
- [Datentypen, Seite 25,](#)
- [Globale Variablenliste, PersistentVariablen, Netzwerkvariablen, Seite 26,](#)
- [Netzwerkvariablenlisten empfängerseitig, Seite 28,](#)
- [Programme, Funktionsbausteine, Funktionen..., Seite 29,](#)
- [Schnittstellen, Schnittstellenmethoden, Schnittstelleneigenschaften, Seite 36,](#)
- [Bibliotheksverwalter, Seite 37,](#)
- [Bibliotheks Info / Library Info, Seite 38,](#)
- [Datenserver, Seite 39,](#)
- [Datenquelle \(Datenserver\), Seite 41,](#)

Vergleich von SPS-Objekten

- Symbolkonfiguration, Seite 42,
- Visualisierungsmanager, Seite 42,
- Visualisierung, Seite 43,
- Textliste, Seite 43,
- Bildersammlung, Seite 44,
- Rezepturverwalter, Seite 44,
- Rezepturdefinition, Seite 46,
- Geräte (Onboard E/A, Inline E/A, Feldbusse, ...), Seite 46.

4.1 Vergleich: Allgemeiner Baustein

Hier wird nur der Name im Fenster angezeigt. Ein „Zusammenführen“ ist nicht vorgesehen, da das Element fix ist.

Unterschiede können sich aus dem Namen oder aus der Sprachabhängigkeit ergeben.

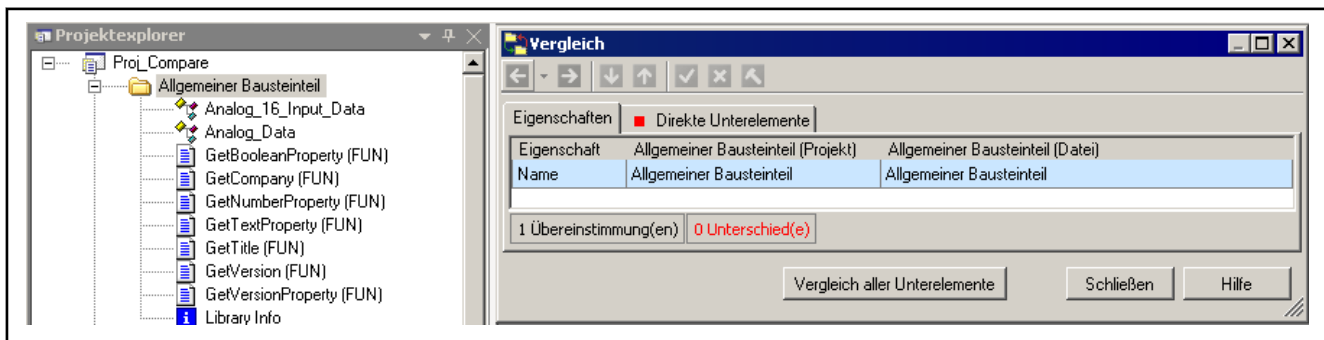


Abb. 4-1: Vergleich: Allgemeiner Baustein

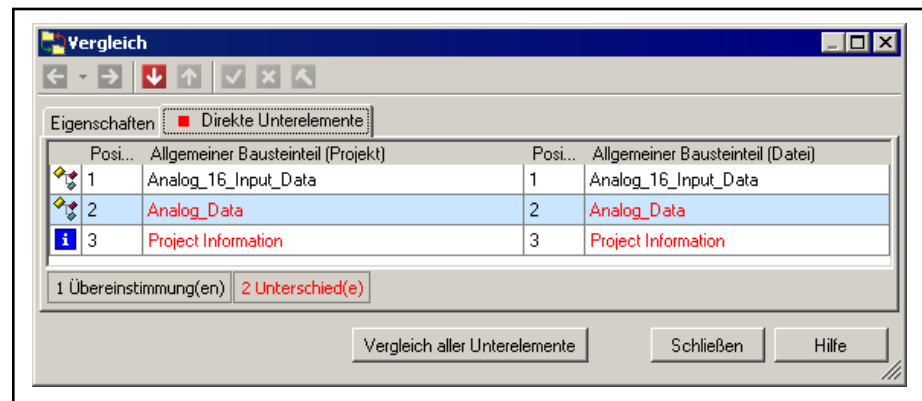


Abb. 4-2: Vergleich: Auflistung der "Direkten Unterelemente", zwei Elemente mit Differenzen (rot)

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).



Die Begriffe "Project Information" / "Projektinformation" bzw. "Library Info" / "Bibliotheks Info" bezeichnen das gleiche Element.

Vergleich: Alle Unterelemente

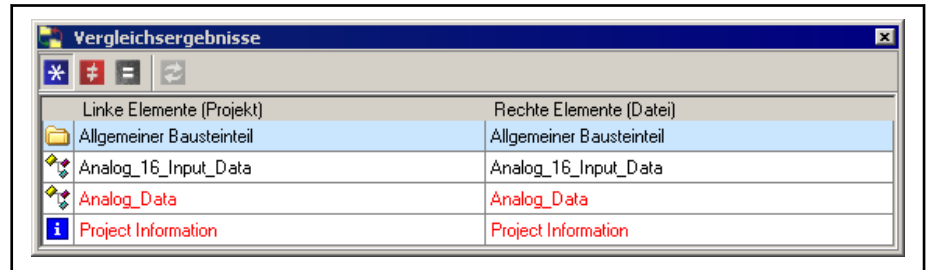


Abb. 4-3: Vergleich: "Alle Unterelemente", zwei Elemente mit Differenzen (rot) mit Filtermöglichkeit "alle Elemente", "ungleiche Elemente", "gleiche Elemente"

4.2 Vergleich: Logic

Hier wird nur der Name im Fenster angezeigt. Ein „Zusammenführen“ ist nicht vorgesehen, da das Element fix ist.

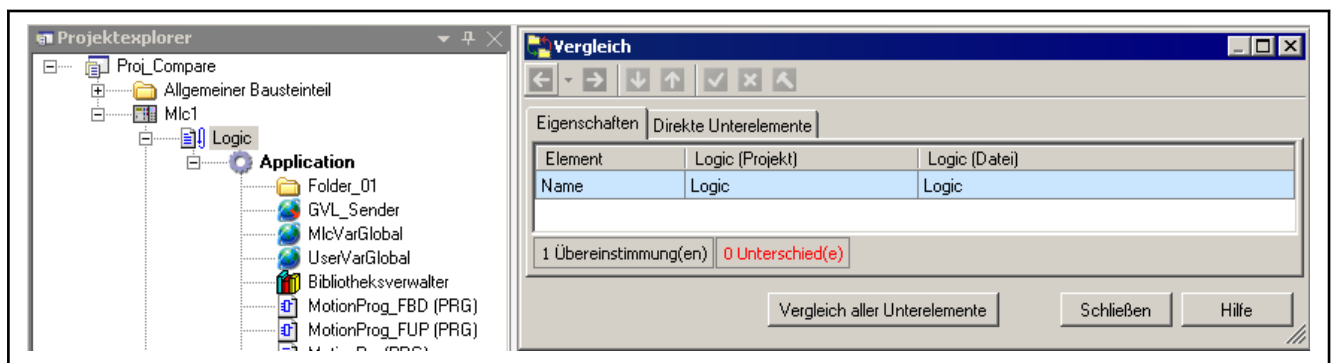


Abb. 4-4: Vergleich: Logic

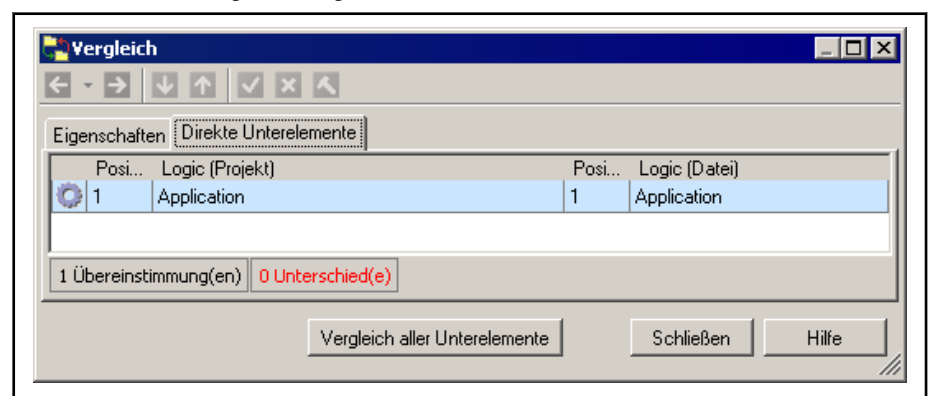


Abb. 4-5: Vergleich: Logic mit einzigem Unterelement "Applikation"

Vergleich von SPS-Objekten

Vergleich: Alle Unterelemente

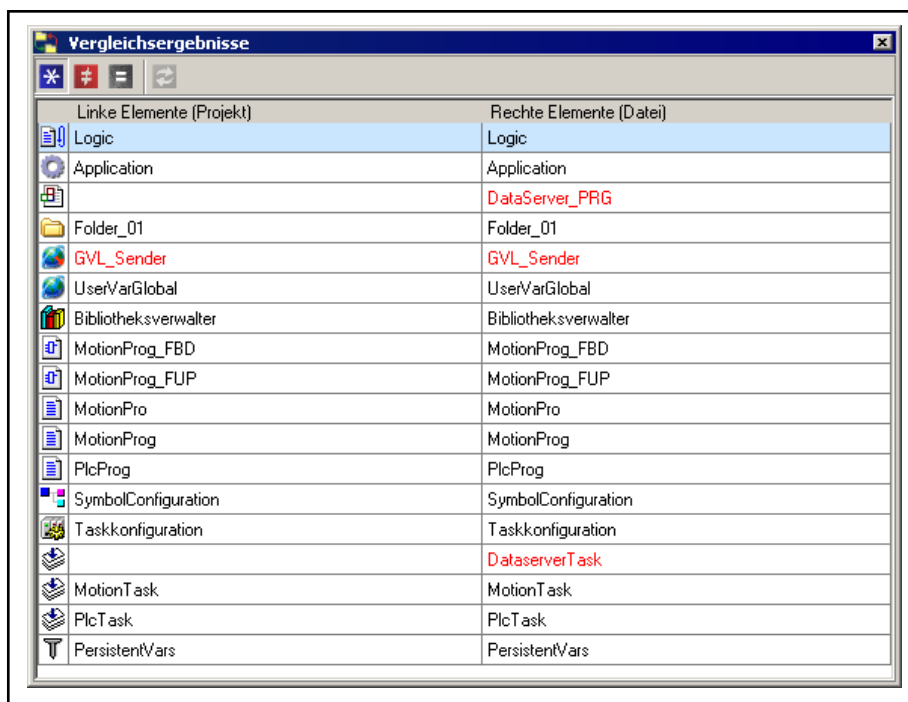


Abb. 4-6: Vergleich: Logic mit allen Unterelementen mit Filtermöglichkeit "alle Elemente", "ungleiche Elemente", "gleiche Elemente"

4.3 Vergleich: Application

Hier wird der Name angezeigt, die Optionen zum Bootprojekt-Handling und die übrigen Application-Eigenschaften. Diese finden sich unter Eigenschaften im Kontextmenü des Applikationsknoten (Kap. 18.1.2 "Objekt 'Applikation'" auf Seite 508).

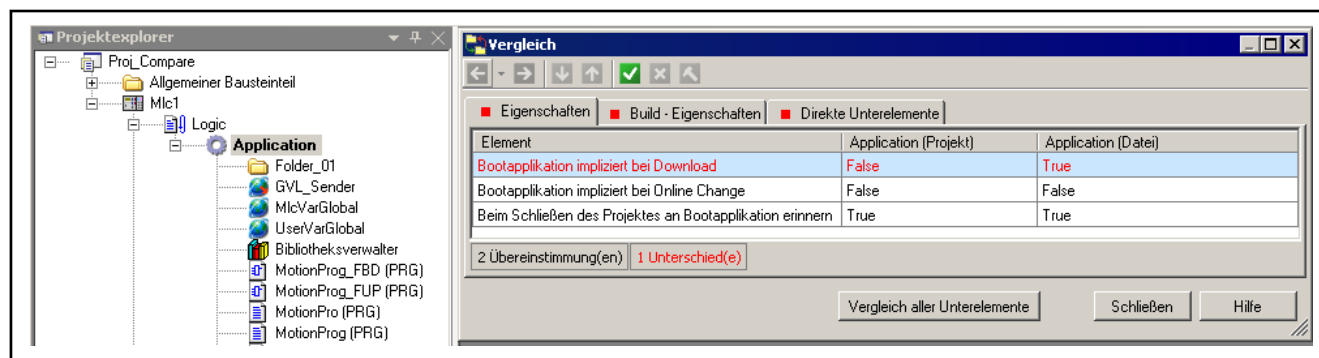


Abb. 4-7: Vergleich: Applikation

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- Übersetzen'" auf Seite 749).

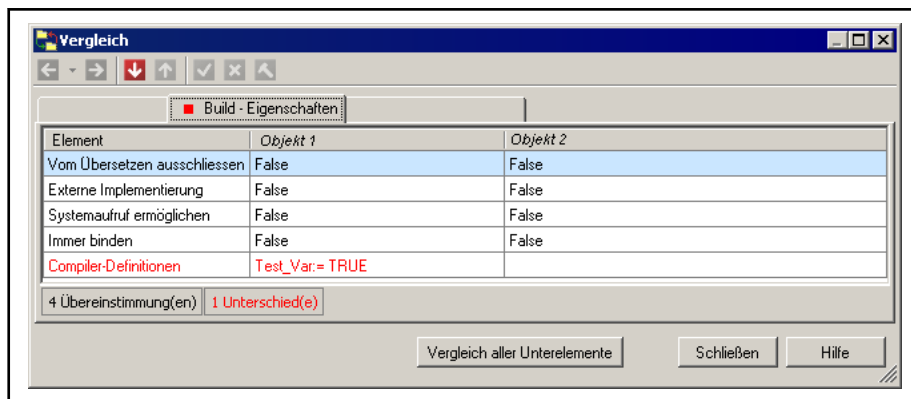


Abb. 4-8: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch der nächste Unterschied angewählt werden (bei der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes wird das Zusammenführen verhindert (rückgängig gemacht).

Das Programm "DataServer_PRG" (rechts Position 7) fehlt im Projekt und soll aus der Datei übernommen werden.

"GVL_Sender" unterscheidet sich inhaltlich und in der Reihenfolge der Objekte gibt es ab Position 7 Unterschiede.

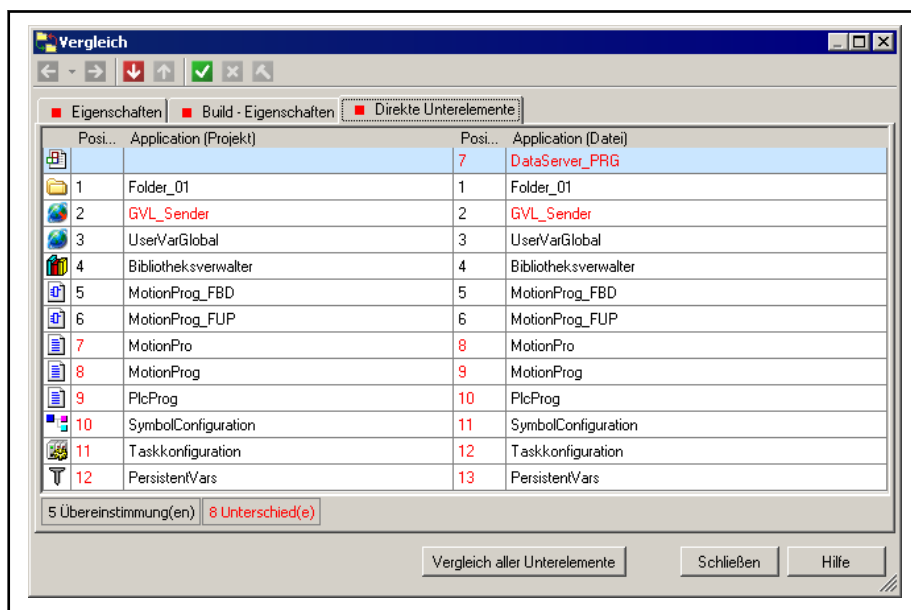


Abb. 4-9: Vergleich: "Direkte Unterelemente"

Vergleich von SPS-Objekten

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

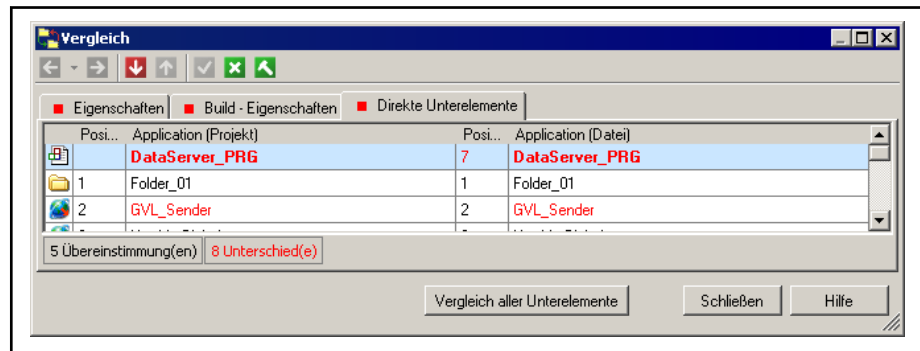


Abb. 4-10: Vergleich: Übernahme von Datei nach Projekt durch "Zusammenführung starten"

Vergleich: Alle Unterelemente

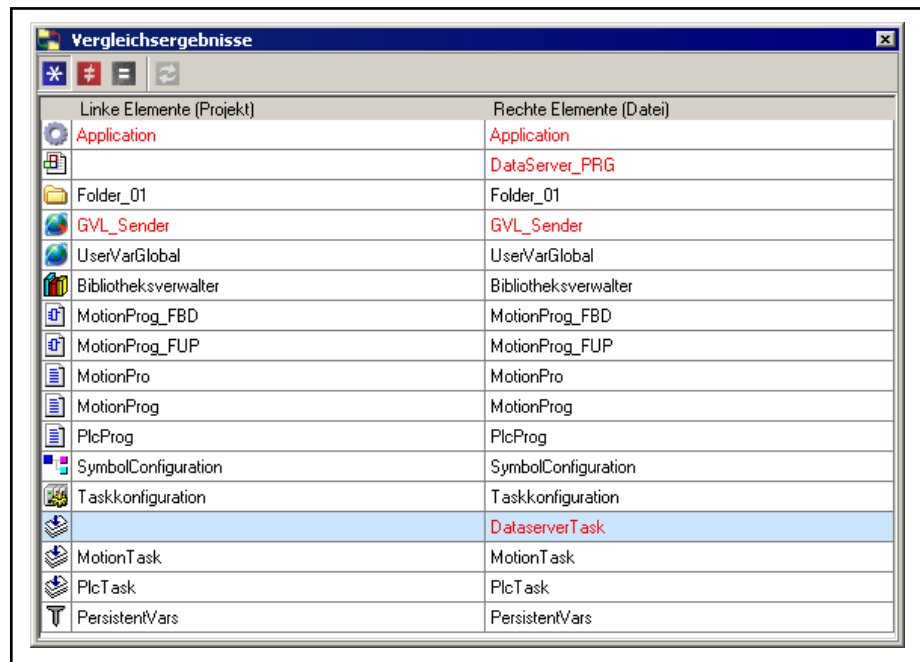


Abb. 4-11: Vergleich: Application mit allen Unterelementen mit Filtermöglichkeit "alle Elemente", "ungleiche Elemente", "gleiche Elemente"

Zusätzlich zu den direkten Unterelementen werden auch deren Unterelemente angezeigt. Hier die "Unter-Unterelemente" von Taskkonfiguration (DataServerTask, nur rechts, MotionTask und PlcTask).

4.4 Vergleich: Taskkonfiguration

Hier wird nur der Name Taskkonfiguration im Fenster angezeigt (Kap. 18.1.25 "Objekt 'Taskkonfiguration'" auf Seite 598). Ein „Zusammenführen“ ist nicht vorgesehen, da das Element fix ist.

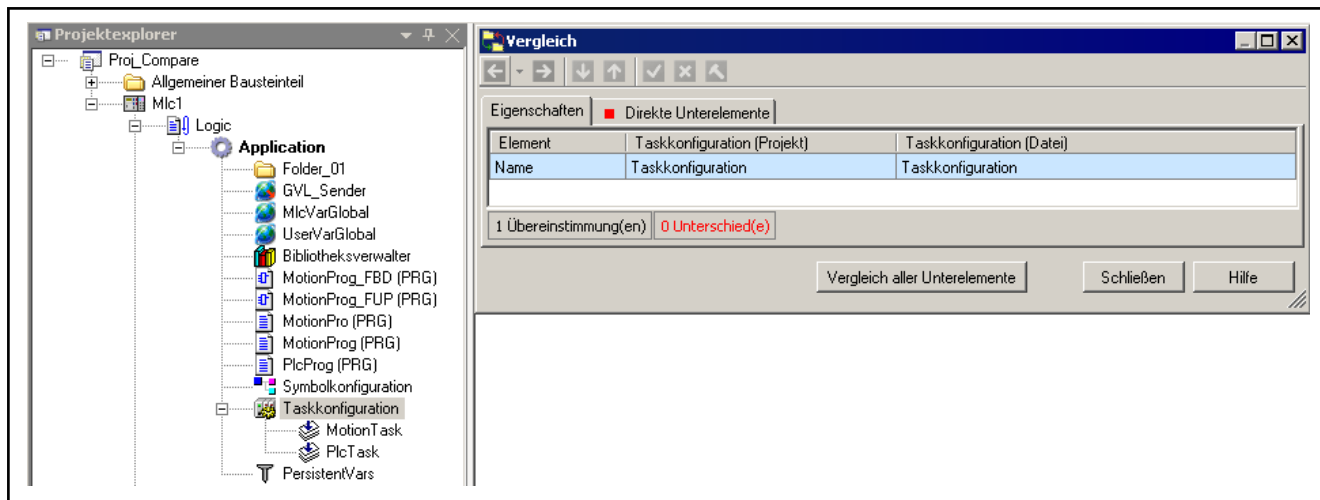


Abb. 4-12: Vergleich: Taskkonfiguration

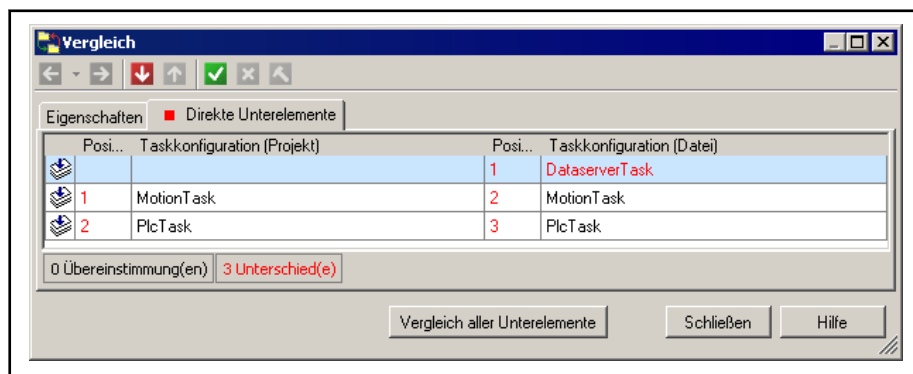


Abb. 4-13: Vergleich: Taskkonfiguration, Tasks als "Direkte Unterelemente"

Die Task "DataloaderTask" ist nur in der Dateiablage enthalten und kann in das Projekt übernommen werden.

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch der nächste Unterschied angewählt werden (bei der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes wird das Zusammenführen verhindert (rückgängig gemacht).

Vergleich von SPS-Objekten

Vergleich: Alle Unterelemente

Linke Elemente (Projekt)	Rechte Elemente (Datei)
Taskkonfiguration	Taskkonfiguration
	DataserverTask
MotionTask	MotionTask
PlcTask	PlcTask

Abb. 4-14: Vergleich: Application mit allen Unterelementen mit Filtermöglichkeit "alle Elemente", "ungleiche Elemente", "gleiche Elemente"

4.5 Vergleich: Task

Der Vergleich von Task-Objekten erfolgt in tabellarischer Form. Hier Vergleich zwischen den Tasks "PlcTask" und "MotionTask" (Kap. 18.1.26 "Objekt 'Task'" auf Seite 602).



Ein Mergen wird nicht empfohlen...

Element	PlcTask (Projekt)	MotionTask (Projekt)
KindOfTask	Cyclic	ExternalEvent
Priority	5	2
Interval	t#20ms	
IntervalUnit		
Enabled	False	True
Time		t#2ms
TimeUnit	ms	ms
Sensitivity		1
POU	PlcProg	MotionProg

2 Übereinstimmung(en) 7 Unterschied(e)

Schließen Hilfe

Abb. 4-15: Vergleich: Task, hier Vergleich zwischen den Tasks "PlcTask" und "MotionTask"

4.6 Vergleich: Ordner (Unterschied zu IndraWorks Ordner)

Hier wird lediglich der Name angezeigt. Dieser kann umbenannt werden, falls der Ordner in Datei (rechts) einen anderen Namen trägt.

Element	Folder_01 (Projekt)	Folder_01 (Datei)
Name	Folder_01	Folder_01

1 Übereinstimmung(en) 0 Unterschied(e)

Schließen Hilfe

Abb. 4-16: Vergleich: Ordner

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

4.7 Vergleich: Datentypen

Der Vergleich betrifft alle Datentypen, die als selbständige Objekte mit dem DUT-Editor angelegt werden können.

Unterstützt werden u.a. Strukturen (STRUCT), Felder (ARRAY), Unions (UNION), Aufzählungsdattentypen (ENUM) und Teilbereichsdattentypen (SUBRANGE).

Beim Vergleich von Datentypen wird der Inhalt textbasiert dargestellt und kann modifiziert werden.

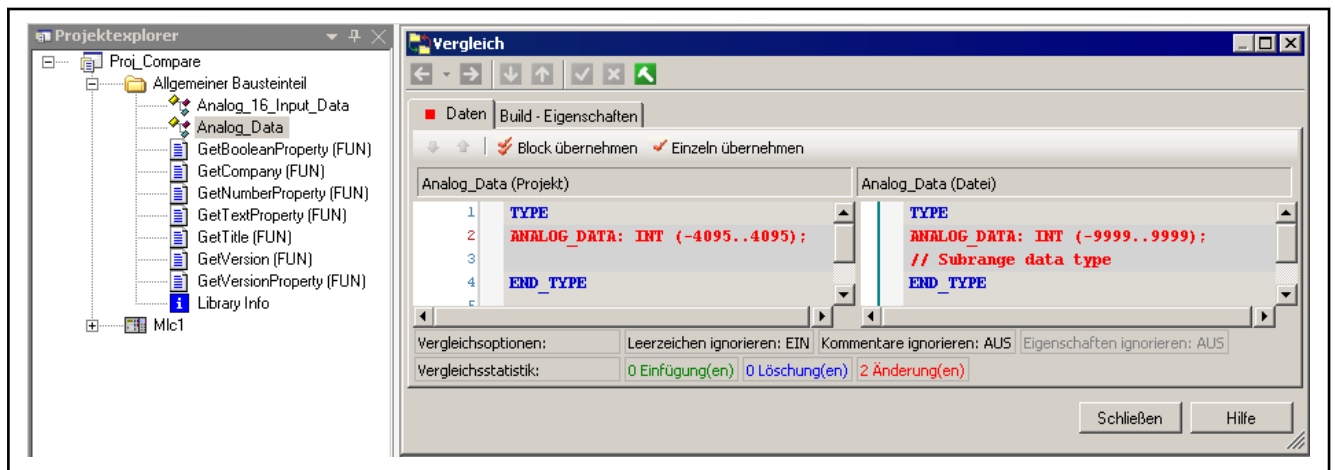


Abb. 4-17: Vergleich: Datentypen

Siehe auch

- [Kap. 18.1.8 "Objekt 'DUT'" auf Seite 517](#)
- [Kap. 17.5.16 "Datentyp 'STRUCT'" auf Seite 396](#)
- [Kap. 17.5.14 "Datentyp 'ARRAY'" auf Seite 386](#)
- [Kap. 17.5.6 "Datentyp 'UNION'" auf Seite 377](#)
- [Kap. 17.5.17 "Enumerationen" auf Seite 398](#)
- [Kap. 17.5.18 "Unterbereichstypen" auf Seite 401](#)

Mit  bzw.  wird der nächste/vorherige Unterschied angewählt.

Mit  **Block übernehmen** kann der markierte Text-Block als Ganzes übernommen werden, mit  **Einzeln übernehmen** kann der Text zeilenweise übernommen werden. Dabei wird zunächst das mögliche Ergebnis der Zu-

sammenführung nur angezeigt. Durch erneutes Betätigen der Taste wird die Anzeige zurückgesetzt.

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749).

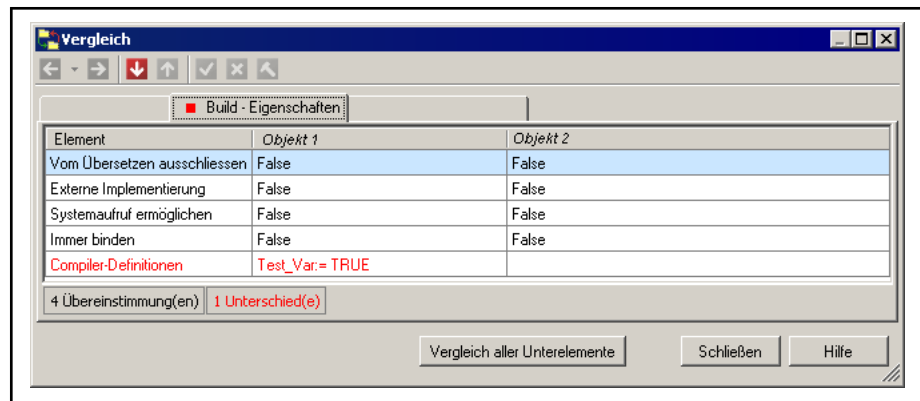


Abb. 4-18: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

4.8 Vergleich: Globale Variablenliste, PersistentVariablen, Netzwerkvariablen

Der Vergleich betrifft alle Globale Variablenlisten ①, Globale Variablenlisten genutzt als Netzwerkvariablen senderseitig ② und Persistenten Variablenlisten ③, die als selbständige Objekte einer Applikation (oder im Allgemeinen Bausteinteil) hinzugefügt werden können.

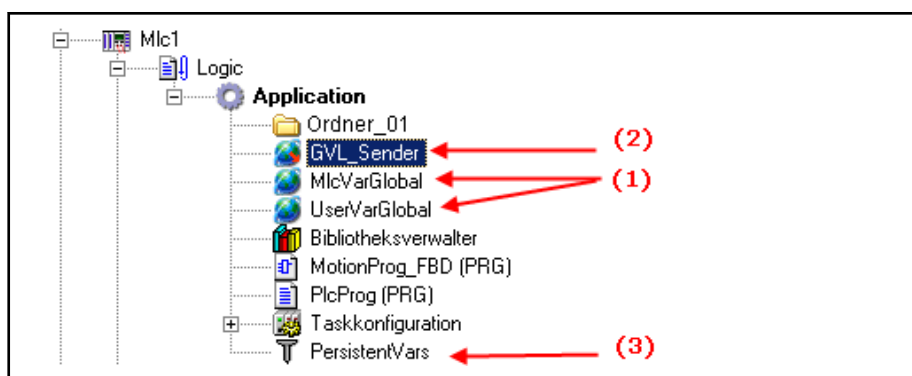


Abb. 4-19: Vergleich: Mögliche Globale Variablenlisten

Siehe auch

- [Kap. 18.1.11 "Objekt 'GVL'- Globale Variablenliste" auf Seite 544](#)
- [Kap. 18.1.12 "Objekt 'Persistente Variablenliste'" auf Seite 546](#)

Beim Vergleich von Datentypen wird der Inhalt textbasiert dargestellt und kann modifiziert werden.

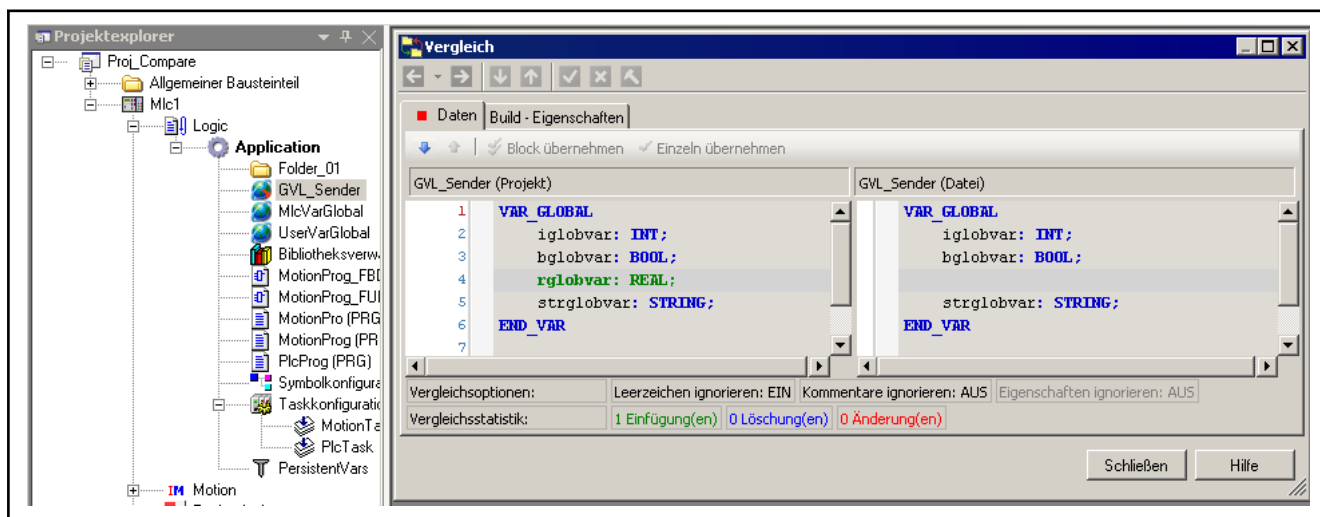


Abb. 4-20: Vergleich: Globale Variablenlisten

Mit bzw. wird der nächste/vorherige Unterschied angewählt.

Mit **Block übernehmen** kann der markierte Text-Block als Ganzes übernommen werden, mit **Einzel übernehmen** kann der Text zeilenweise übernommen werden. Dabei wird zunächst das mögliche Ergebnis der Zusammenführung nur angezeigt. Durch erneutes Betätigen der Taste wird die Anzeige zurückgesetzt.

Durch Anwahl des Hammers wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes wird das Zusammenführen verhindert (rückgängig gemacht).

Das nächste Register zeigt die Übersetzungseigenschaften ([Kap. "Dialog 'Eigenschaften- Übersetzen'" auf Seite 749](#)).

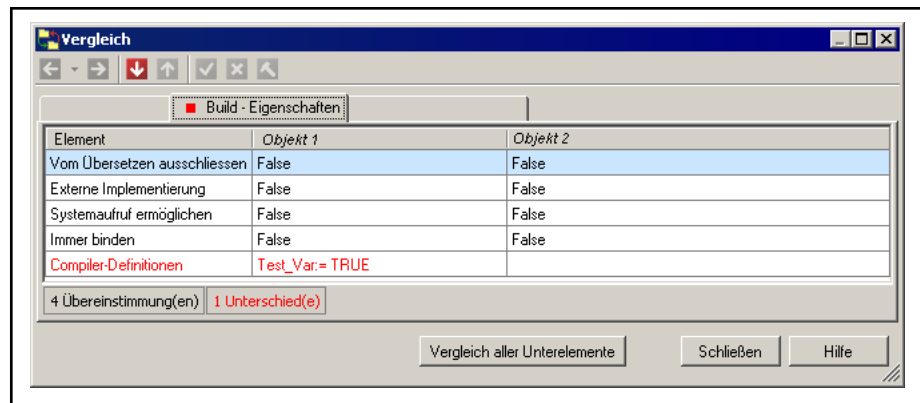


Abb. 4-21: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

4.9 Vergleich: Netzwerkvariablenlisten empfängerseitig

Der Vergleich betrifft Globale Variablenlisten genutzt als 'Netzwerkvariablen empfängerseitig', die als selbständige Objekte einer Applikation erzeugt wurden (Kap. 18.1.16 "Objekt 'Globale Netzwerkvariablen-Liste (Empfänger)'" auf Seite 552).



Diese Listen werden automatisch erzeugt und innerhalb des Projektes aktuell gehalten.

Der Vergleich wird also nur zur Kontrolle auf Unterschiede durchgeführt.

Beim Vergleich von Netzwerkvariablenlisten wird der Inhalt XML-basiert dargestellt. Eine Modifikation (Zusammenführung) ist nicht möglich.

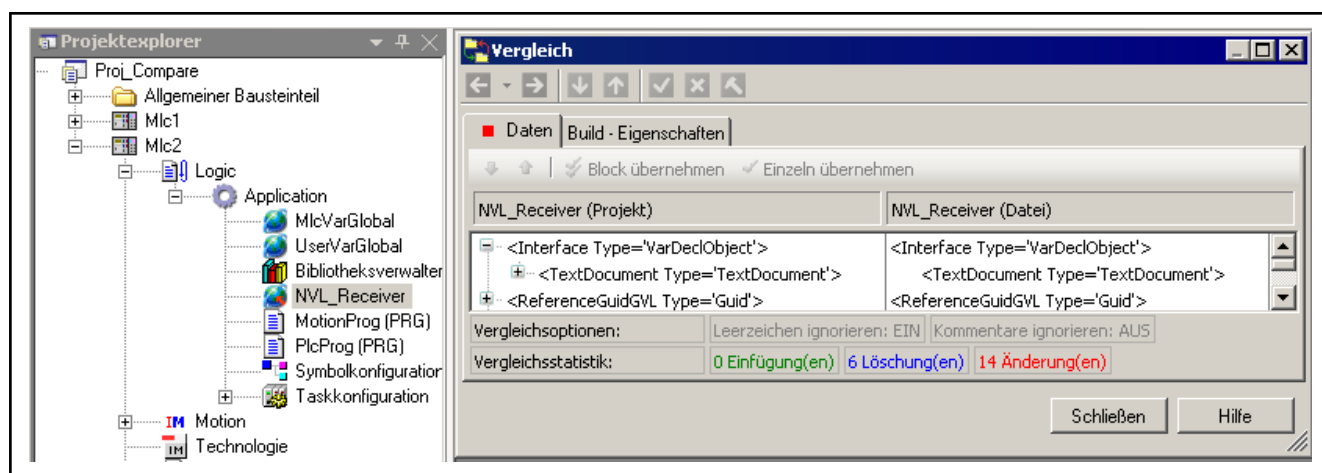


Abb. 4-22: Vergleich: Netzwerkvariablenlisten empfängerseitig

Mit bzw. wird der nächste/vorherige Unterschied angewählt.

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749).

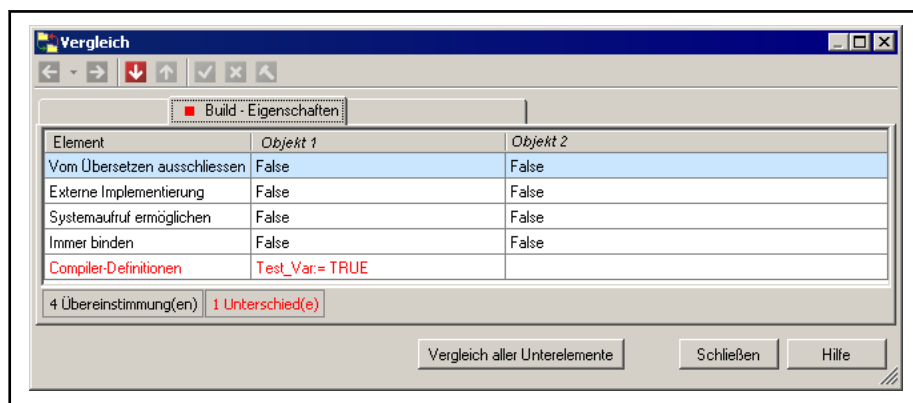


Abb. 4-23: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch der nächste Unterschied angewählt werden (bei der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes wird das Zusammenführen verhindert (rückgängig gemacht).

4.10 Vergleich: Programme, Funktionsbausteine, Funktionen...

Der Vergleich betrifft die folgenden Objekte:

- Kap. "Objekt 'Programm(PROGRAM)' auf Seite 554
- Kap. "Objekt 'Funktionsbaustein' (FUNCTION_BLOCK)" auf Seite 556
- Kap. "Objekt 'Funktion(FUNCTION)' auf Seite 558
- Kap. "Objekt 'Aktion'" auf Seite 574

Vergleich von SPS-Objekten

- Kap. "AS-Elemente 'Schrittund 'Transition'" auf Seite 262
- Kap. "Objekt 'Methode' " auf Seite 564
- Kap. "Objekt 'Eigenschaft'" auf Seite 571, PROPERTY mit GET und SET

Die genannten Objekte können in den im System verfügbaren IEC-Programmiersprachen-Editoren deklariert und implementiert werden.

Vergleich: ST

Der Deklarationsteil und Objekte, die in Strukturiertem Text implementiert wurden, werden textbasiert zum Vergleich gebracht (Kap. 17.1.3 "Strukturierter Text und Erweiterter Strukturierter Text (ExST)" auf Seite 241).

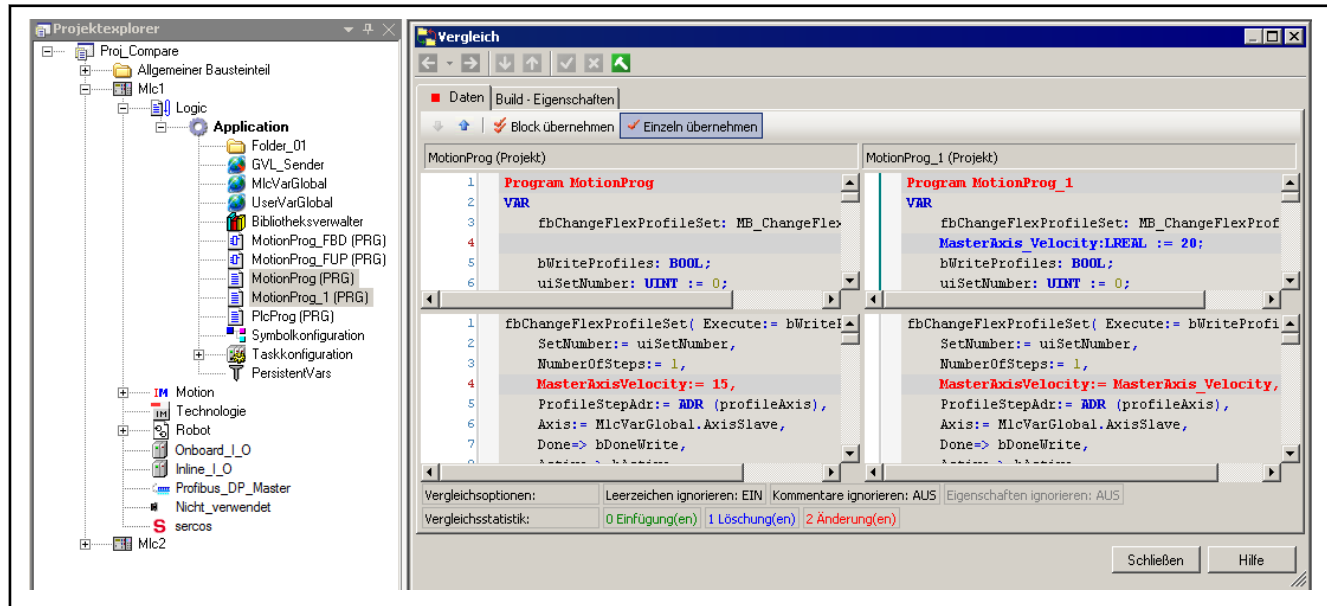


Abb. 4-24: Vergleich: PROGRAM in ST, Deklaration und Implementation

Eine Zusammenführung ist sowohl im Deklarationsteil als auch in der Implementation möglich.

Mit bzw. wird der nächste/vorherige Unterschied angewählt.

Mit **Block übernehmen** kann der markierte Text-Block als Ganzes übernommen werden, mit **Einzel übernehmen** kann der Text zeilenweise übernommen werden. Dabei wird zunächst das mögliche Ergebnis der Zusammenführung nur angezeigt. Durch erneutes Betätigen der Taste wird die Anzeige zurückgesetzt.

Durch Anwahl des Hammers wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes wird das Zusammenführen verhindert (rückgängig gemacht).

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749).

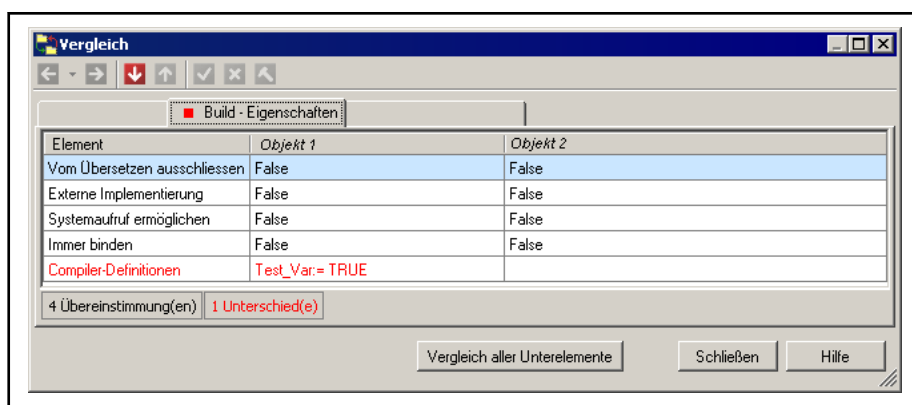


Abb. 4-25: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

Vergleich: FBD (FUP)

Objekte, die als Funktionsplan implementiert wurden, werden grafisch zum Vergleich gebracht.

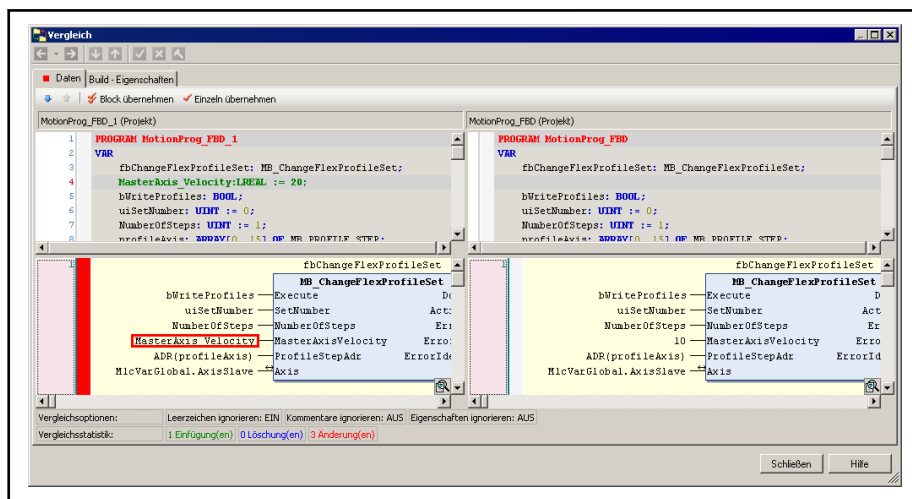


Abb. 4-26: Vergleich: PROGRAM in FBD, Deklaration und Implementation

Eine Zusammenführung ist sowohl im Deklarationsteil als auch in der Implementation möglich.

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749).

Vergleich von SPS-Objekten

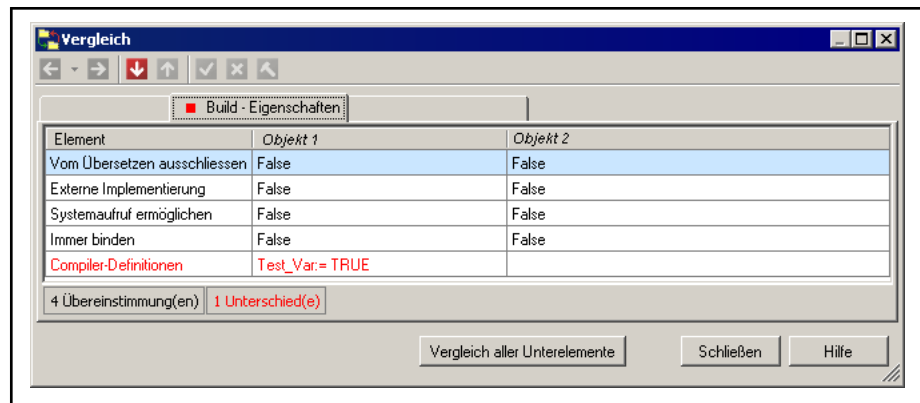


Abb. 4-27: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch der nächste Unterschied angewählt werden (bei der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes wird das Zusammenführen verhindert (rückgängig gemacht).

Vergleich: IL (AWL)

In Anweisungsliste implementierte POU's werden als XML verglichen. Ein Zusammenführen ist hier nicht möglich.

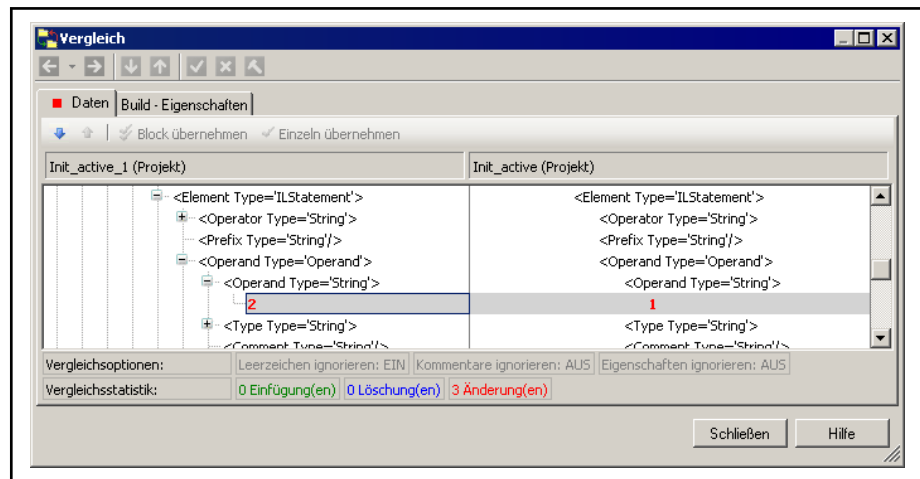


Abb. 4-28: Vergleich: Aktion in Anweisungsliste, Implementation

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- Übersetzen'" auf Seite 749).

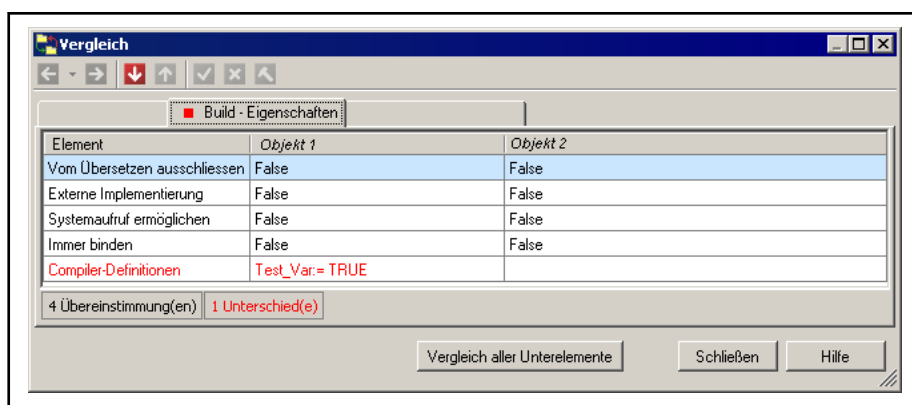


Abb. 4-29: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

Vergleich: SFC (AS)

Objekte, die mittels SFC (Ablaufsprache) strukturiert wurden, werden grafisch zum Vergleich gebracht. Ein Zusammenführen ist nur im Deklarationsteil möglich.

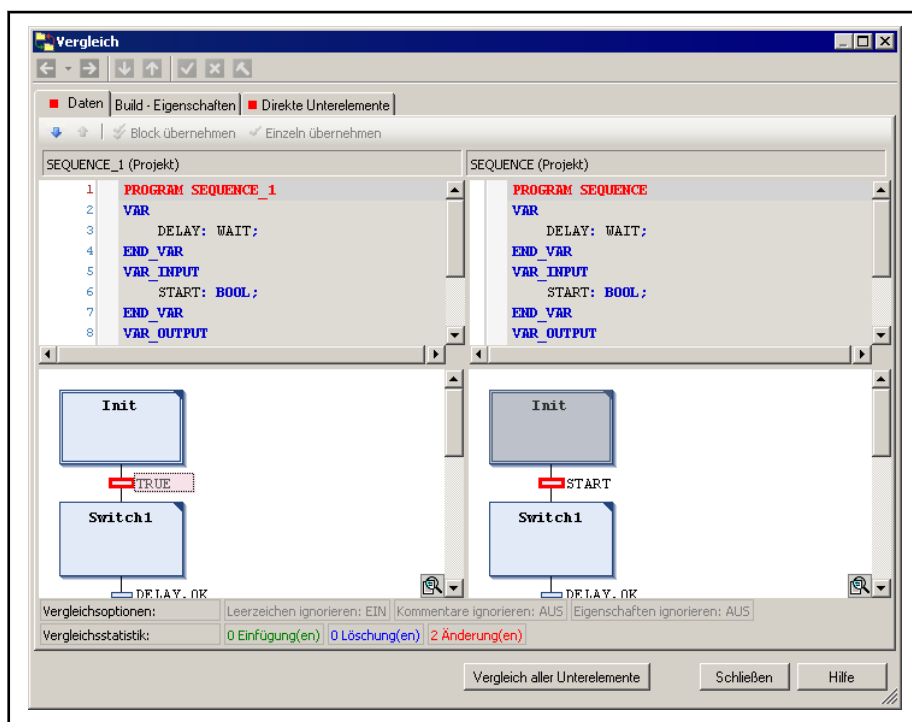


Abb. 4-30: Vergleich: FUNCTION_BLOCK mit SFC-Struktur, Deklaration und Implementation

Vergleich von SPS-Objekten

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749).

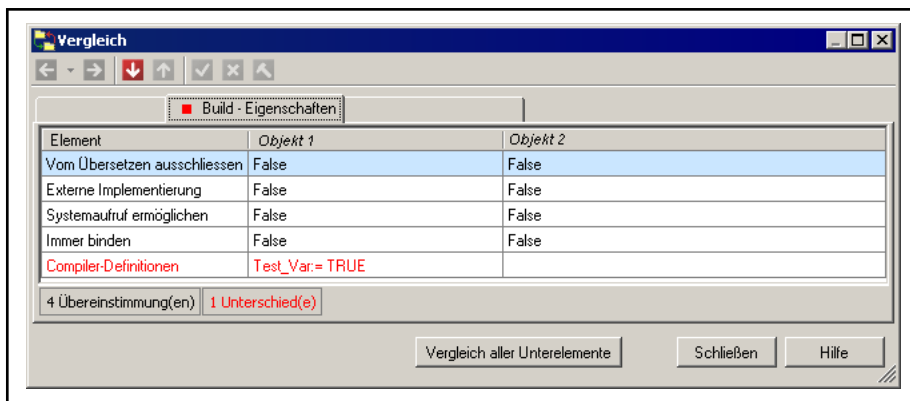


Abb. 4-31: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch der nächste Unterschied angewählt werden (bei der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes wird das Zusammenführen verhindert (rückgängig gemacht).

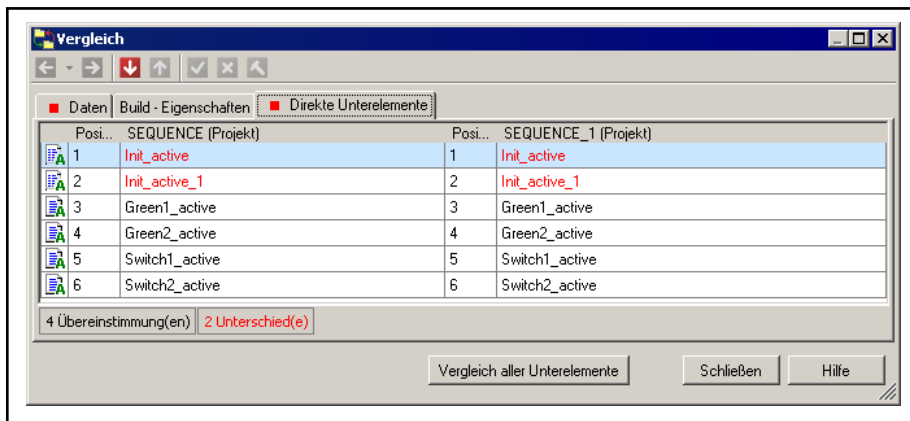


Abb. 4-32: Vergleich: POU "Direkte Unterelemente"

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch der nächste Unterschied angewählt werden (bei der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes wird das Zusammenführen verhindert (rückgängig gemacht).

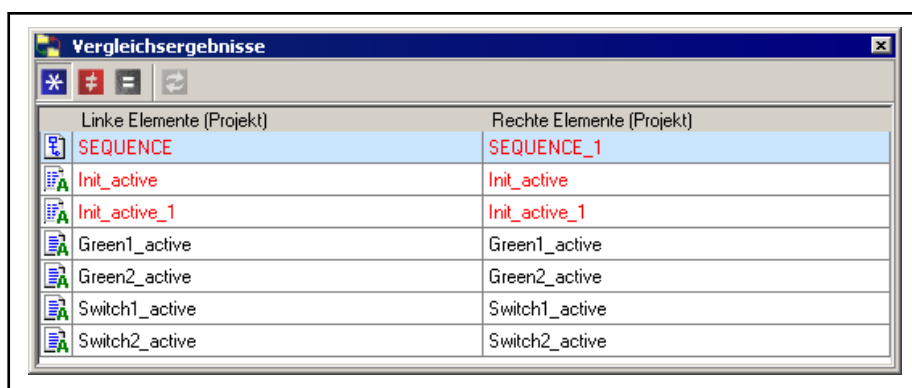


Abb. 4-33: Vergleich: POU mit allen Unterelementen mit Filtermöglichkeit "alle Elemente", "ungleiche Elemente", "gleiche Elemente"

Vergleich: CFC

In Anweisungsliste implementierte POUs werden als XML verglichen. Ein Zusammenführen ist hier nicht möglich.

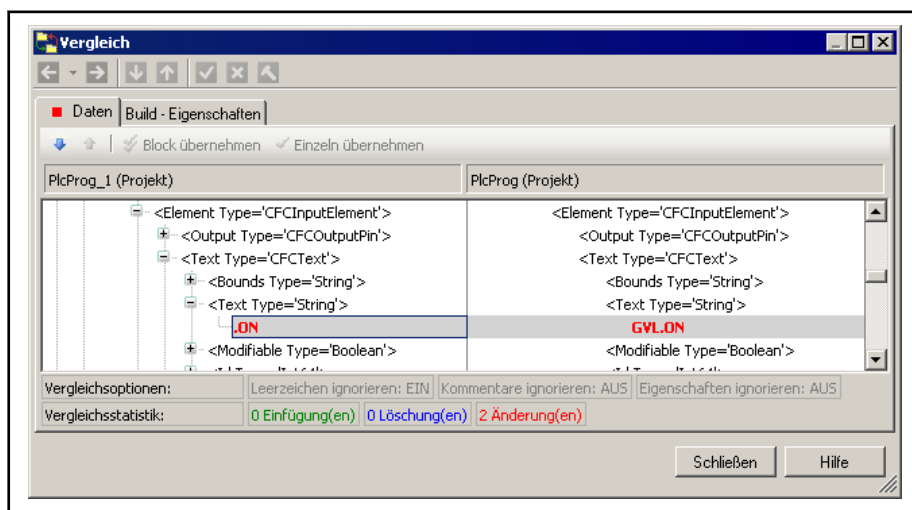


Abb. 4-34: Vergleich: PROGRAM in CFC, Deklaration und Implementation

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- Übersetzen'" auf Seite 749).

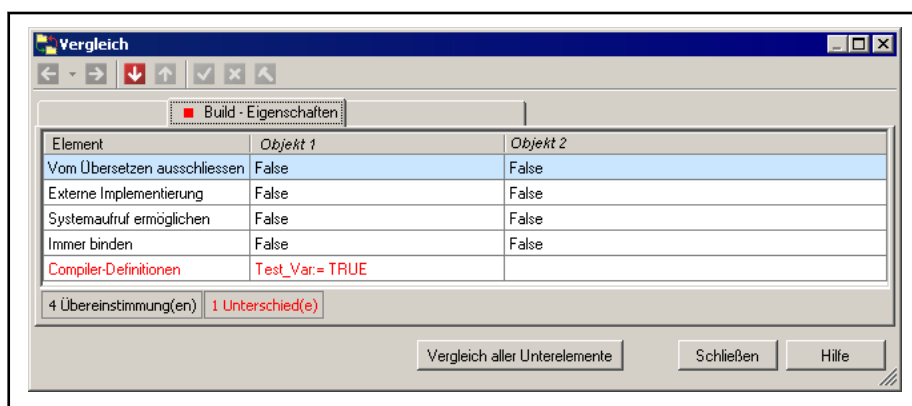


Abb. 4-35: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Vergleich von SPS-Objekten

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

4.11 Vergleich: Schnittstellen, Schnittstellen-Methoden, Schnittstellen-Eigenschaften

Beim Vergleich von Schnittstellen, Schnittstellenmethoden, Schnittstelleneigenschaften wird der Inhalt XML-basiert dargestellt (Kap. "Objekt 'Schnittstelle(INTERFACE)' auf Seite 562).

Eine Modifikation (Zusammenführung) ist nicht möglich.

Mit  bzw.  wird der nächste/vorherige Unterschied angewählt.

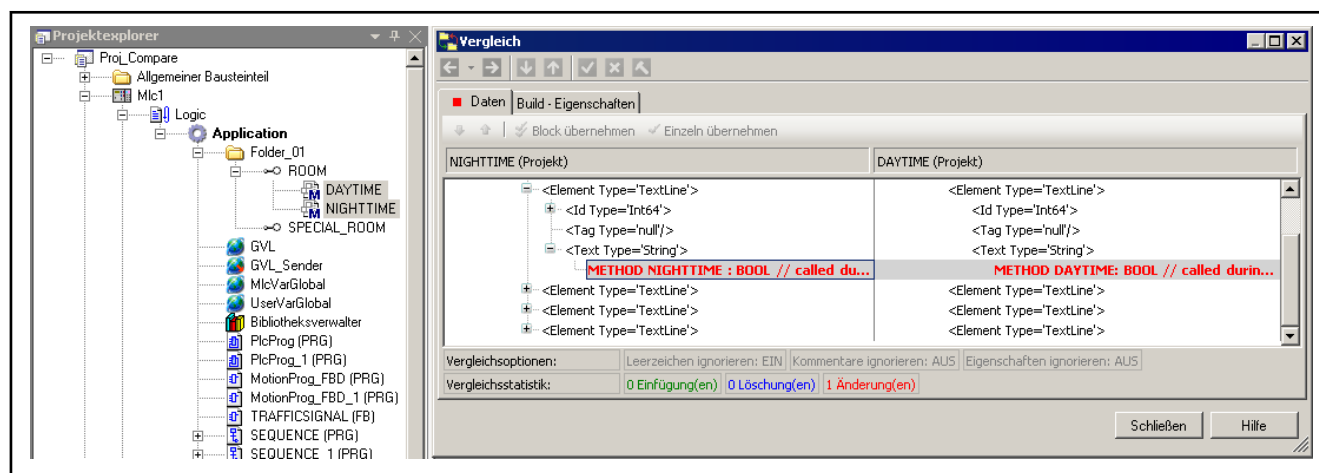


Abb. 4-36: Vergleich: Schnittstellen, Schnittstellenmethoden, Schnittstelleneigenschaften

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften-Übersetzen'" auf Seite 749).

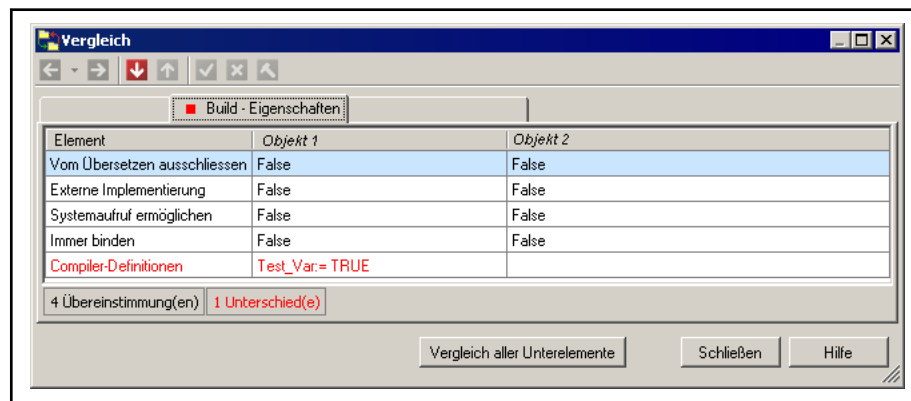


Abb. 4-37: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

4.12 Vergleich: Bibliotheksverwalter

Beim Vergleich von Bibliotheksverwaltern wird der Inhalt tabellarisch aufgelistet. Eine Übernahme ist zeilenweise möglich.

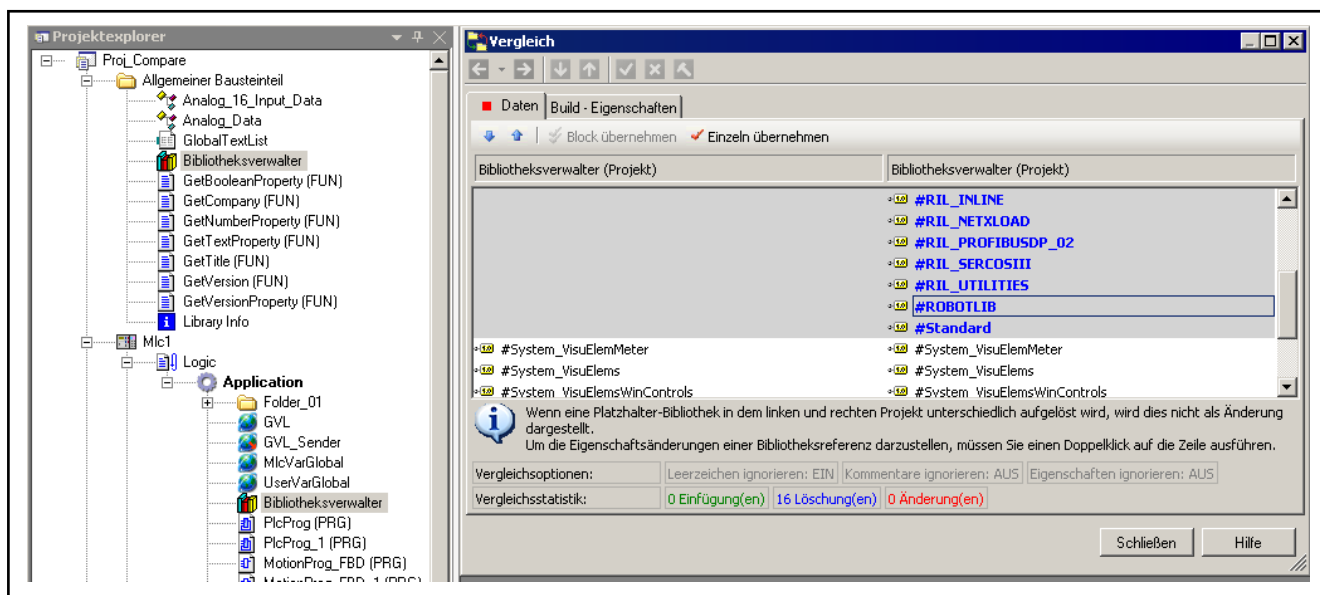


Abb. 4-38: Vergleich: Bibliotheksverwalter, hier ML_Robot im Projekt hinzu genommen

Mit  bzw.  wird der nächste/vorherige Unterschied angewählt.

Mit  **Block übernehmen** kann der markierte Text-Block als Ganzes übernommen werden, mit  **Einzel übernehmen** kann der Text zeilenweise übernommen werden. Dabei wird zunächst das mögliche Ergebnis der Zusammenführung nur angezeigt. Durch erneutes Betätigen der Taste wird die Anzeige zurückgesetzt.

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749).

Vergleich von SPS-Objekten

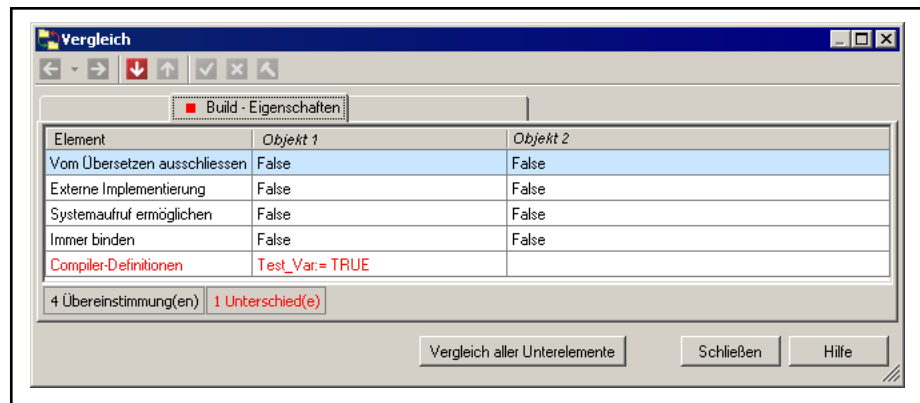


Abb. 4-39: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

4.13 Vergleich: Bibliotheks Info / Library Info

Beim Vergleich von "Bibliotheks Info" wird der Inhalt XML-basiert dargestellt. Eine Modifikation (Zusammenführung) ist nicht möglich.

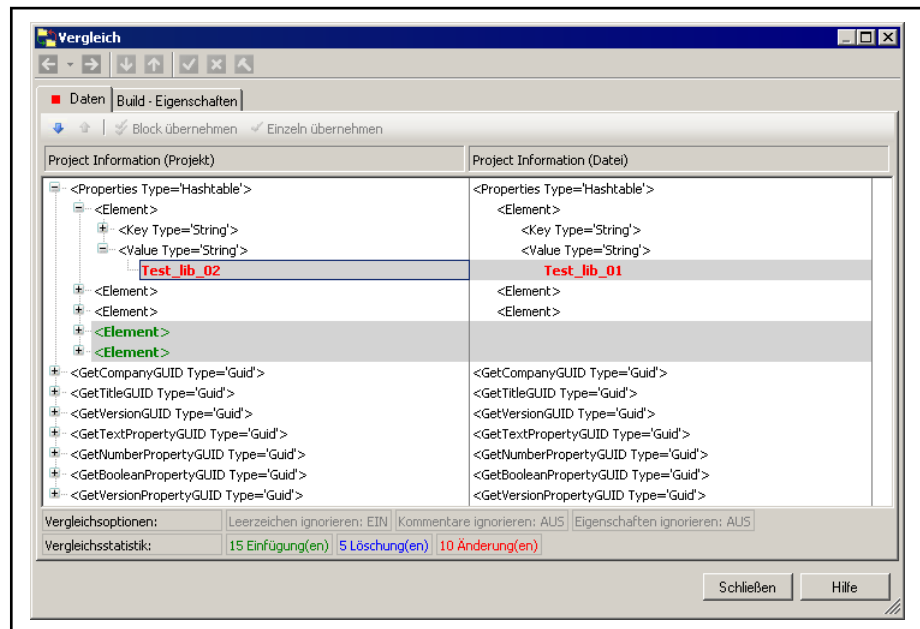


Abb. 4-40: Vergleich: Bibliotheks Info / Library Info, Bibliothek freigegeben mit Versionsnummer 10.06.0.1



Die Begriffe "Project Information" / "Projektinformation" bzw. "Library Info" / "Bibliotheks Info" bezeichnen das gleiche Element.

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749).

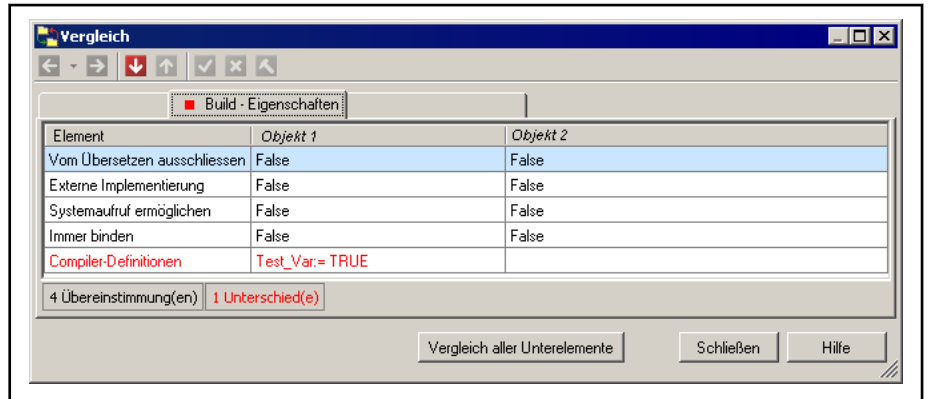


Abb. 4-41: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

4.14 Vergleich: Datenserver

Beim Vergleich von Datenservern wird der Inhalt XML-basiert dargestellt. Eine Modifikation (Zusammenführung) ist nicht möglich.

Vergleich von SPS-Objekten

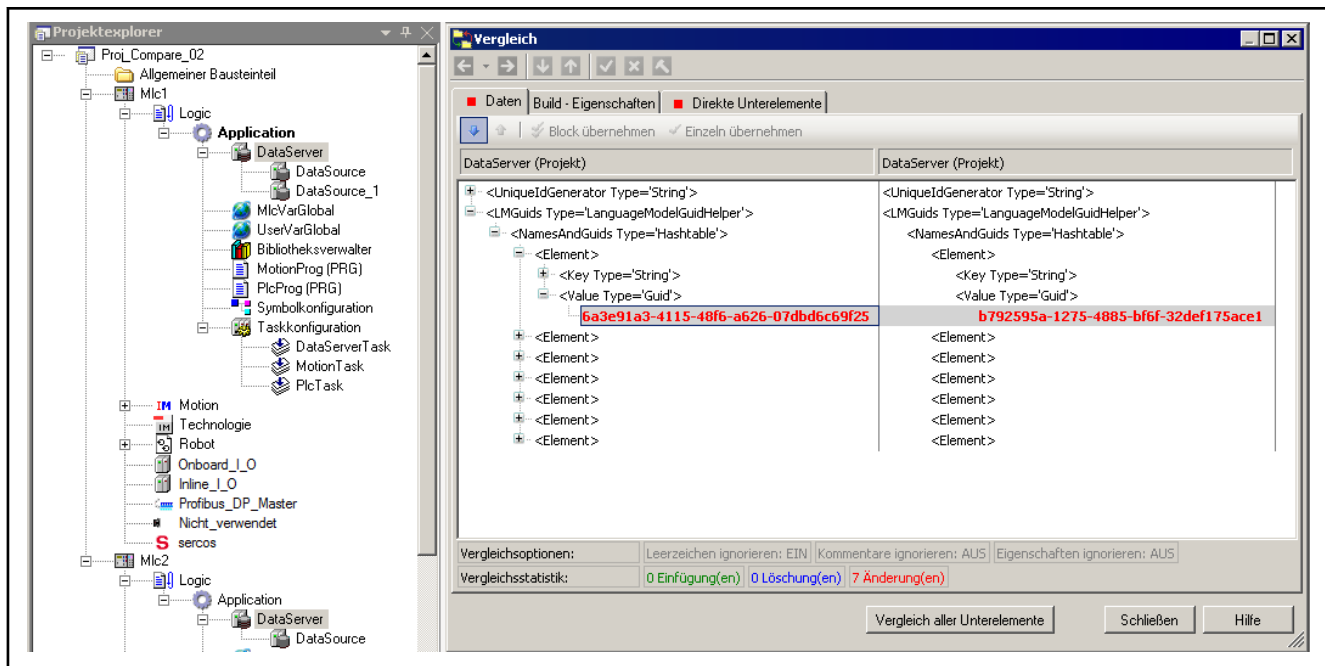


Abb. 4-42: Vergleich: Datenserver

Mit bzw. wird der nächste/vorherige Unterschied angewählt.

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften-Übersetzen'" auf Seite 749).

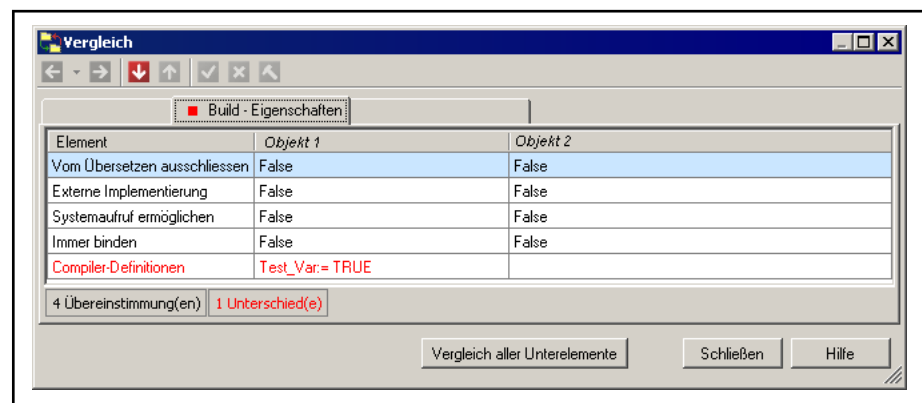


Abb. 4-43: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch der nächste Unterschied angewählt werden (bei der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes wird das Zusammenführen verhindert (rückgängig gemacht).

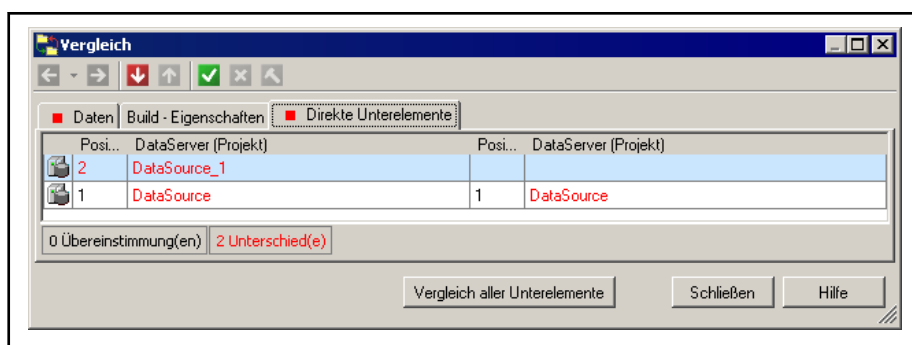


Abb. 4-44: Vergleich: Datenserver, Direkte Unterelemente

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch der nächste Unterschied angewählt werden (bei der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes wird das Zusammenführen verhindert (rückgängig gemacht).

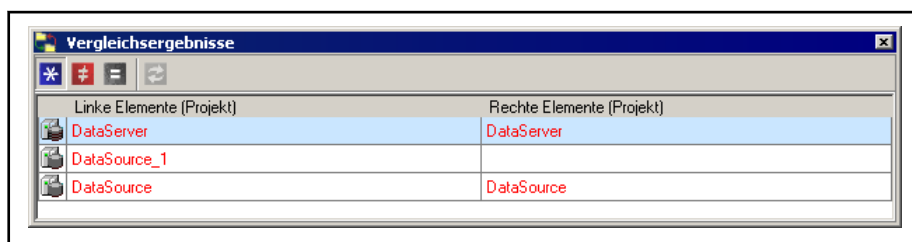


Abb. 4-45: Vergleichsergebnisse: Datenserver

4.15 Vergleich: Datenquelle (Datenserver)

Beim Vergleich von Datenquellen wird der Inhalt XML-basiert dargestellt. Eine Modifikation (Zusammenführung) ist nicht möglich.

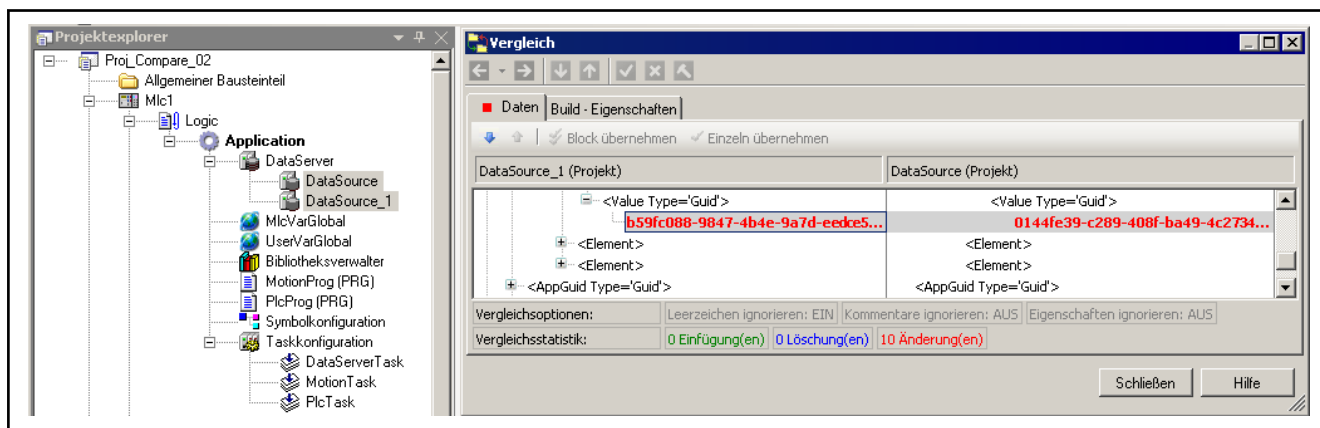


Abb. 4-46: Vergleich: DataSources

Mit bzw. wird der nächste/vorherige Unterschied angewählt.

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749).

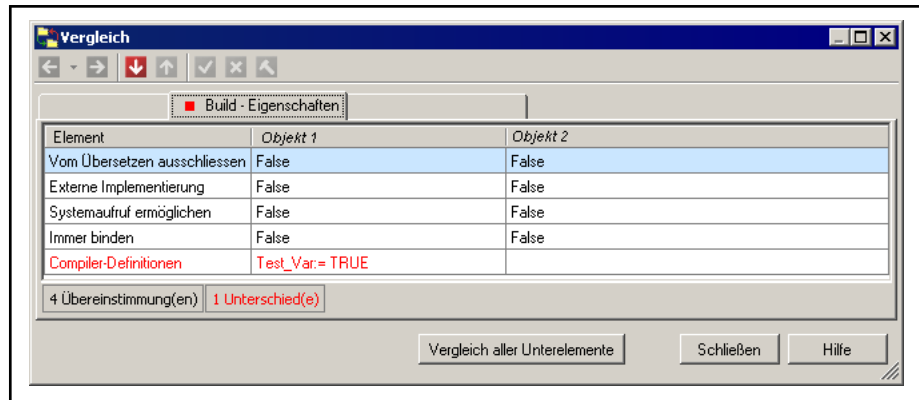


Abb. 4-47: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch der nächste Unterschied angewählt werden (bei der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes wird das Zusammenführen verhindert (rückgängig gemacht).

4.16 Vergleich: Symbolkonfiguration

Beim Vergleich von Symbolkonfigurationen wird der Inhalt XML-basiert dargestellt. Eine Modifikation (Zusammenführung) ist nicht möglich.

4.17 Vergleich: Visualisierungsmanager

Beim Vergleich von Visualisierungsmanager wird der Inhalt XML-basiert dargestellt. Eine Modifikation (Zusammenführung) ist nicht möglich.

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749).

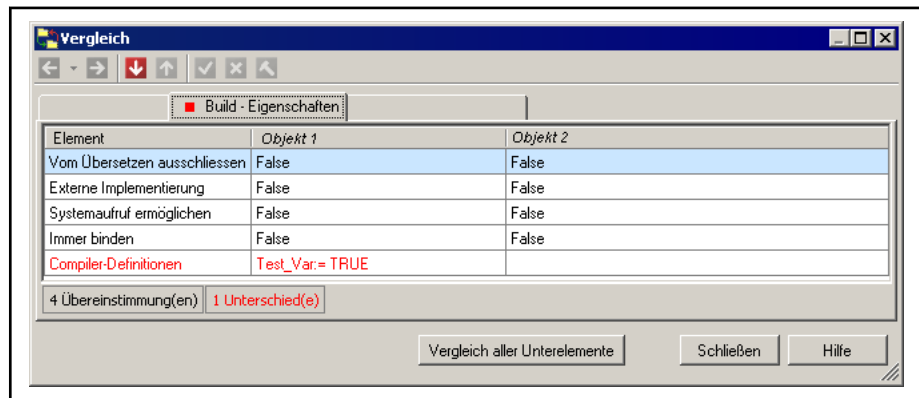


Abb. 4-48: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

4.18 Vergleich: Visualisierung

Beim Vergleich von Visualisierungen wird der Inhalt XML-basiert dargestellt. Eine Modifikation (Zusammenführung) ist nicht möglich.

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749).

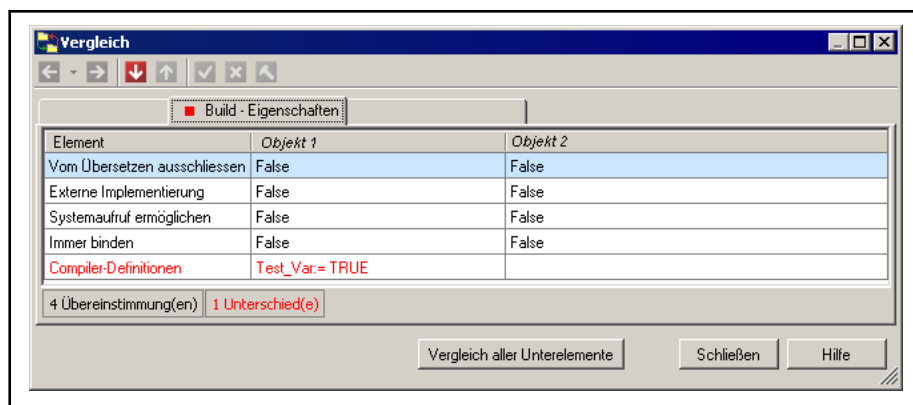


Abb. 4-49: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

4.19 Vergleich: Textliste

Beim Vergleich von Textlisten wird der Inhalt XML-basiert dargestellt. Eine Modifikation (Zusammenführung) ist nicht möglich.

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749).

Vergleich von SPS-Objekten

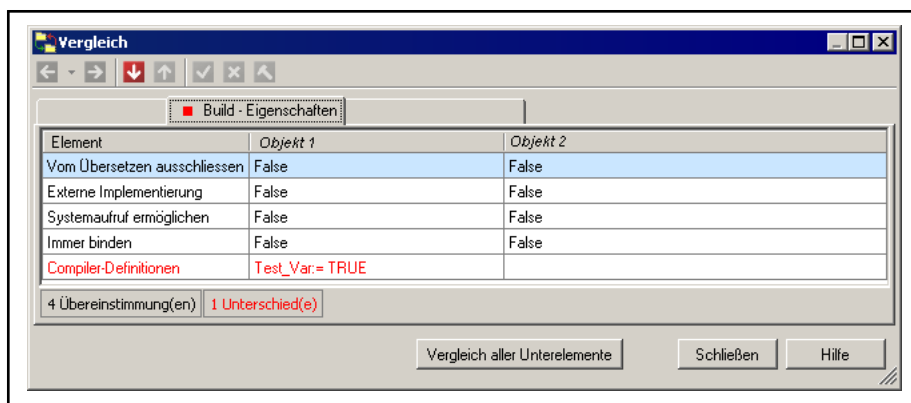


Abb. 4-50: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch der nächste Unterschied angewählt werden (bei der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes wird das Zusammenführen verhindert (rückgängig gemacht).

4.20 Vergleich: Bildersammlung

Beim Vergleich von Bildersammlungen wird der Inhalt XML-basiert dargestellt. Eine Modifikation (Zusammenführung) ist nicht möglich.

4.21 Vergleich: Rezepturverwalter

Beim Vergleich von Rezepturverwaltern wird der Inhalt XML-basiert dargestellt. Eine Modifikation (Zusammenführung) ist nicht möglich.

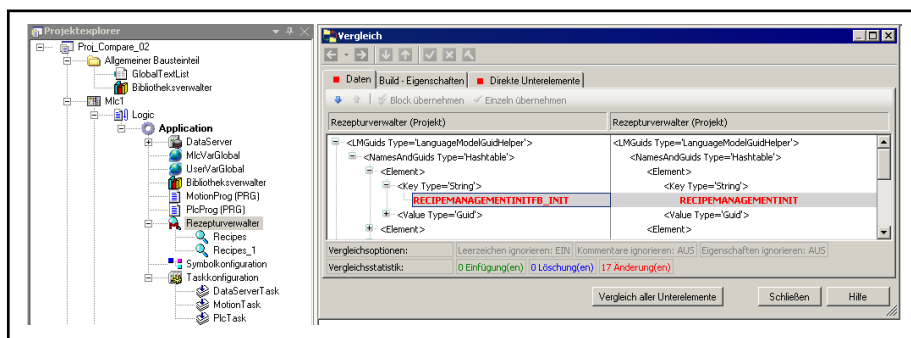


Abb. 4-51: Vergleich: Rezepturverwalter

Mit bzw. wird der nächste/vorherige Unterschied angewählt.

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749).

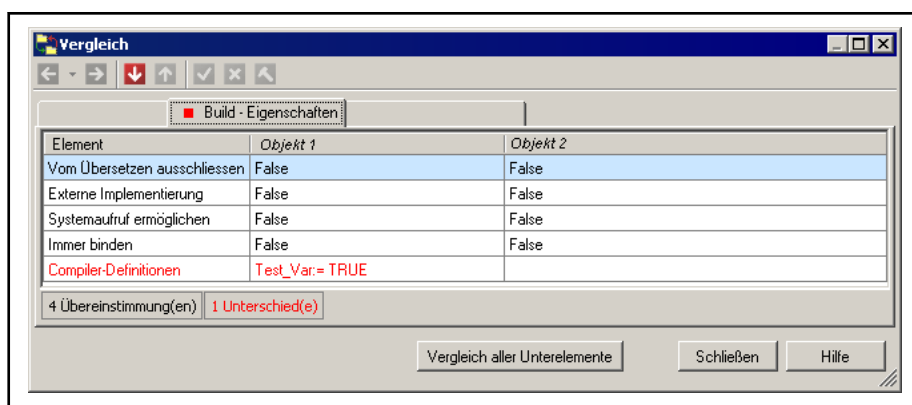


Abb. 4-52: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

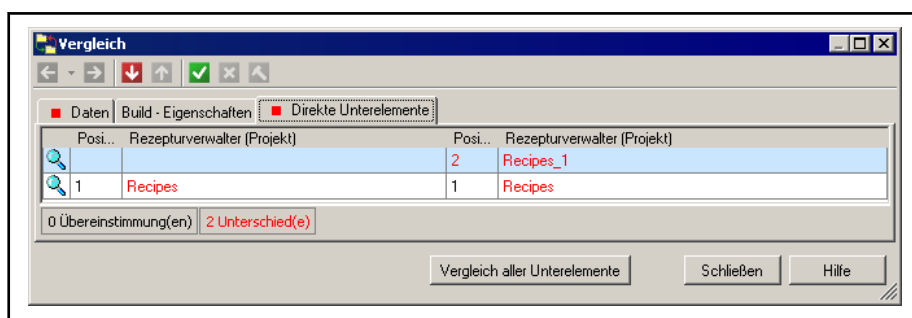


Abb. 4-53: Vergleich: Rezepturverwalter, Direkte Unterelemente

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

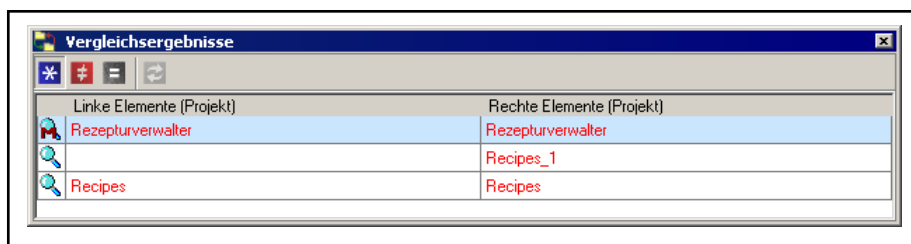


Abb. 4-54: Vergleichsergebnisse: Rezepturverwalter

4.22 Vergleich: Rezepturdefinition

Beim Vergleich von Rezepturverwaltern wird der Inhalt XML-basiert dargestellt. Eine Modifikation (Zusammenführung) ist nicht möglich.

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749).

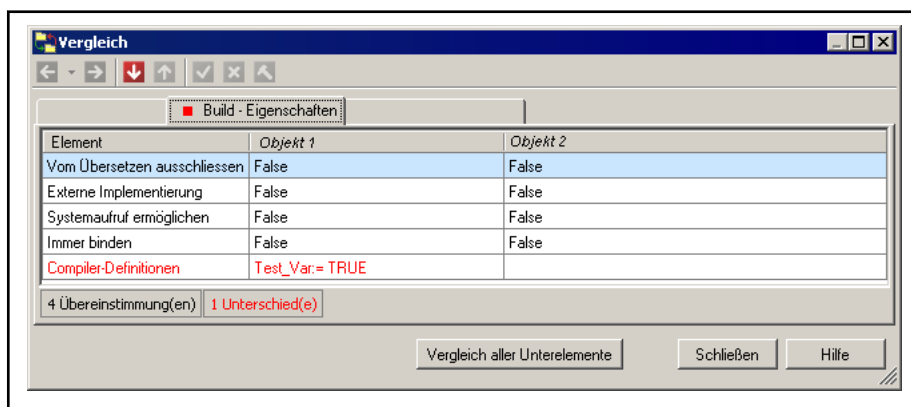


Abb. 4-55: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

4.23 Vergleich: Geräte (Onboard E/A, Inline E/A, Feldbusse, ...)

Der Vergleich betrifft die zur jeweiligen Steuerung gehörenden Geräte (Onboard E/As, Inline E/As, Feldbusse). Da sich diese von Steuerungstyp zu Steuerungstyp unterscheiden, sind die nachfolgenden Vergleichsergebnisse als Beispiele zu sehen.

Prinzipiell wird versucht, dem Anwender die von der Eingabe her bekannte Umgebung beim Vergleich wieder anzubieten und auf der Basis des Vergleiches ein Zusammenführen zu ermöglichen.

Onboard E/A:

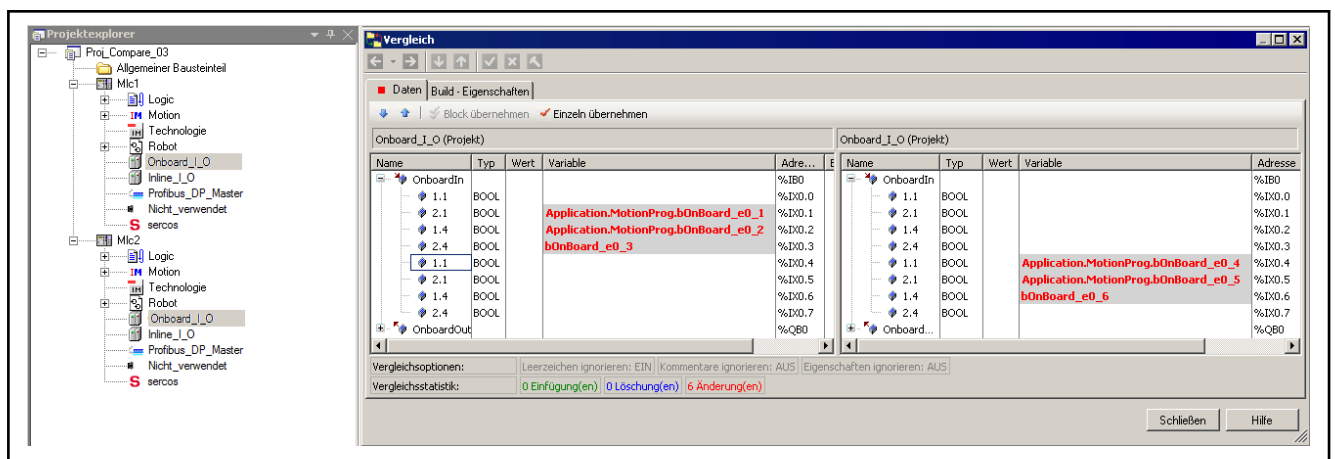


Abb. 4-56: Vergleich: Onboard E/A:

Mit bzw. wird der nächste/vorherige Unterschied angewählt.

Mit **Block übernehmen** kann der markierte Text-Block als Ganzes übernommen werden, mit **Einzeln übernehmen** kann der Text zeilenweise übernommen werden. Dabei wird zunächst das mögliche Ergebnis der Zusammenführung nur angezeigt. Durch erneutes Betätigen der Taste wird die Anzeige zurückgesetzt.

Durch Anwahl des Hammers wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes wird das Zusammenführen verhindert (rückgängig gemacht).

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- Übersetzen'" auf Seite 749).

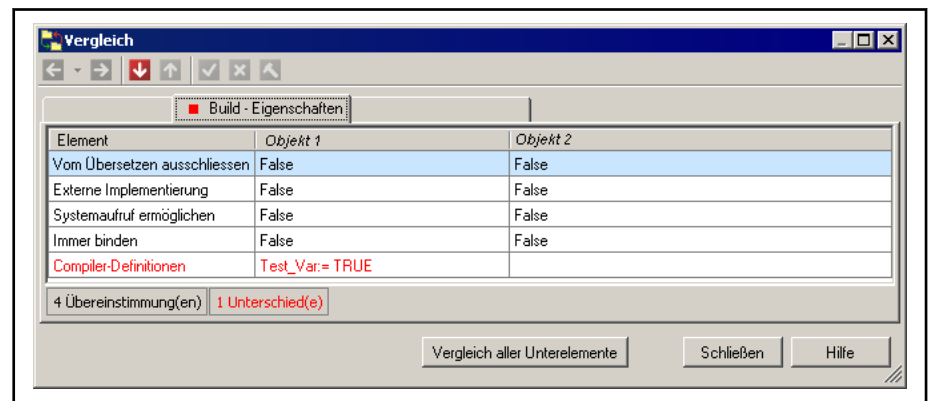


Abb. 4-57: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch der nächste Unterschied angewählt werden (bei der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers wird das Zusammenführen ausgeführt.

Vergleich von SPS-Objekten

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

Inline E/A-Objekte: Der Vergleich zeigt die zu den Inline E/A-Objekten gehörigen Inline-Zyklenzähler und Diagnosedaten-Register. Ein Mergen ist nicht vorgesehen.

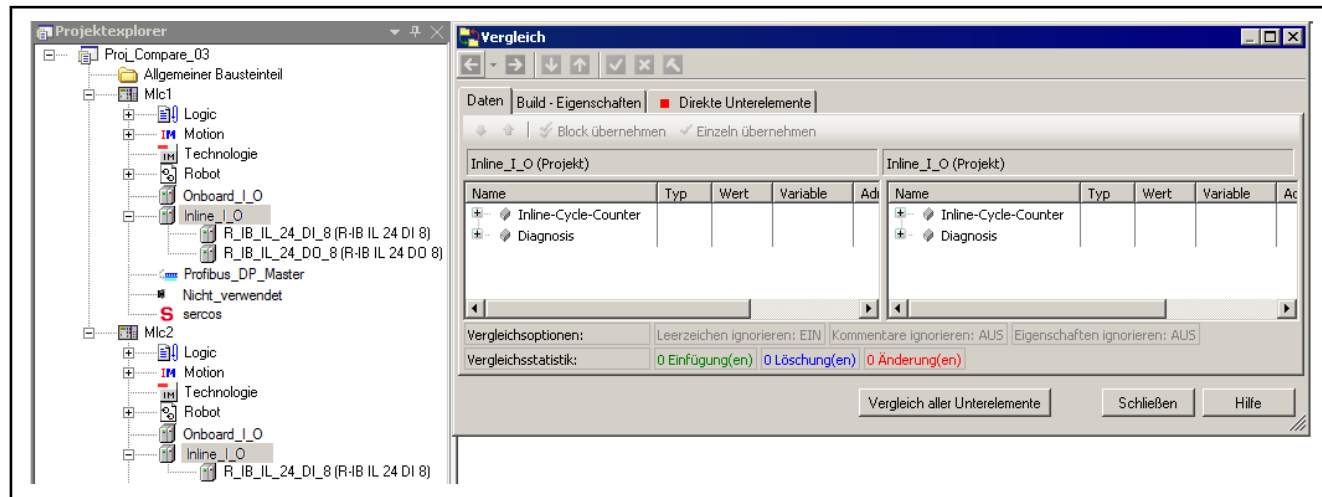


Abb. 4-58: Vergleich: Inline E/A: Inline-Zyklenzähler und Diagnosedaten-Register

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften-Übersetzen'" auf Seite 749).

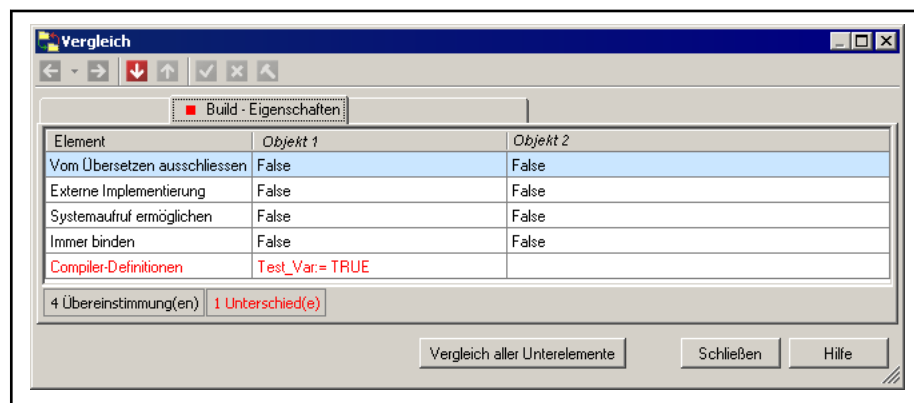


Abb. 4-59: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

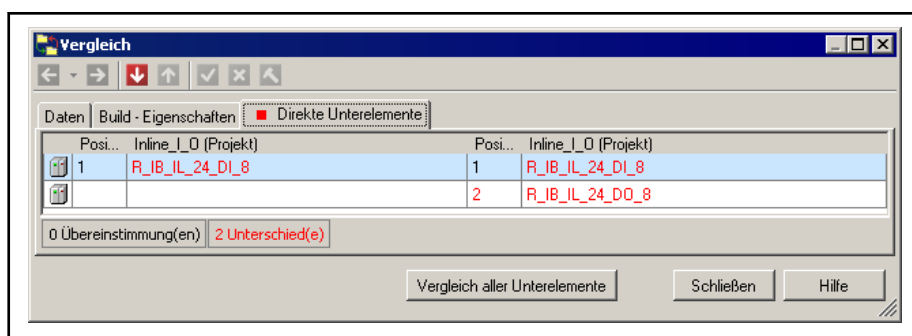


Abb. 4-60: Vergleich: Inline E/A: Direkte Unterelemente

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

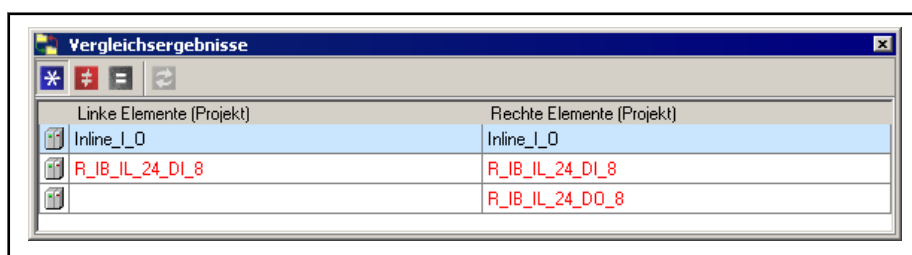


Abb. 4-61: Vergleichsergebnisse: Inline E/A: alle Unterelemente

Inline E/A-Module

Die rote Färbung der Namen der Inline E/A-Module bei "Direkten Unterelementen" und "Vergleichsergebnissen" weist bei gleichem Namen auf Unterschiede in der Belegung hin:

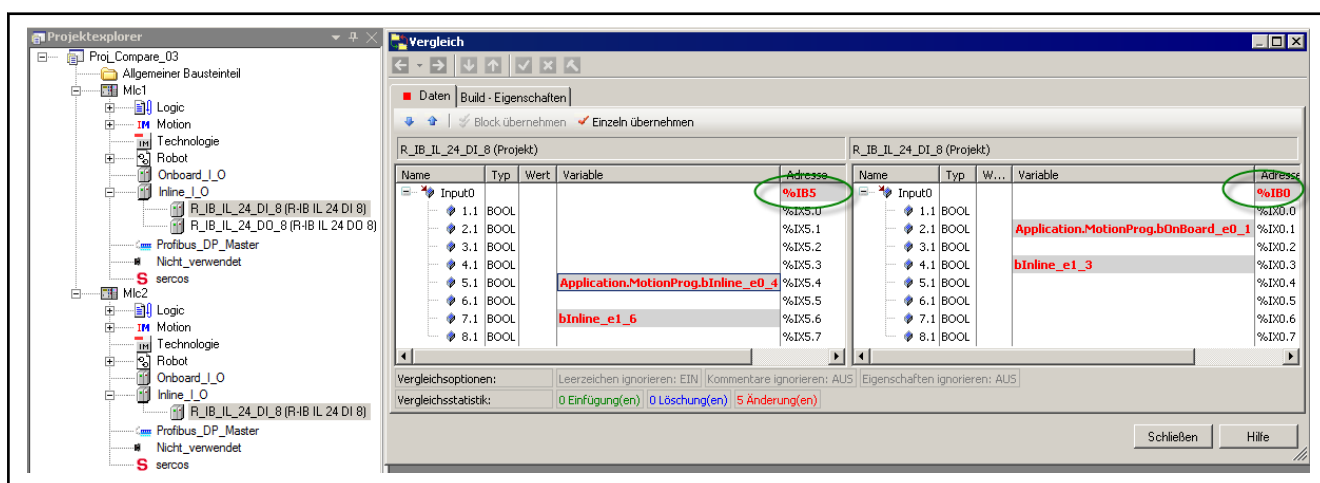


Abb. 4-62: Vergleich: Inline E/A-Module

Die unterschiedlichen Adressen bei gleicher Lage der Module im Projektor zeigen auf eine Adressverschiebung hin.

Vergleich von SPS-Objekten

Mit  bzw.  wird der nächste/vorherige Unterschied angewählt.

Mit  **Block übernehmen** kann der markierte Text-Block als Ganzes übernommen werden, mit  **Einzeln übernehmen** kann der Text zeilenweise übernommen werden. Dabei wird zunächst das mögliche Ergebnis der Zusammenführung nur angezeigt. Durch erneutes Betätigen der Taste wird die Anzeige zurückgesetzt.

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

Das nächste Register zeigt die Übersetzungseigenschaften (Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749).

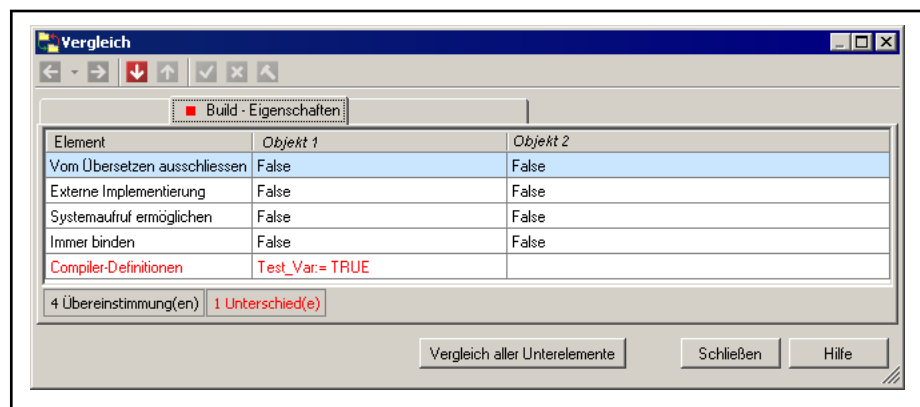


Abb. 4-63: Vergleich: Übersetzungseigenschaften Objekt 1/ Objekt 2, ein Unterschied erkannt und rot markiert

Unterscheidet sich die Belegung in den Spalten der beiden Objekte, kann durch  der nächste Unterschied angewählt werden (bei  der vorherige Unterschied).

Erscheint das Häkchen , kann durch dessen Betätigung die Übernahme (das Zusammenführen) vorbereitet werden:

Durch Anwahl des Hammers  wird das Zusammenführen ausgeführt.

Durch Anwahl des Kreuzes  wird das Zusammenführen verhindert (rückgängig gemacht).

5 Online-Vergleich von SPS-Objekten

Um den Vergleich der SPS-Objekte zwischen Engineering PC und Steuerung unterstützen zu können, müssen entsprechende Daten zu einem definierten Zeitpunkt auf der Steuerung abgelegt werden. Für den Strukturvergleich werden für die jeweiligen Objekte Checksummen übertragen, so dass diese schnell erzeugt und übertragen werden können. Für den Detailvergleich werden gezippte Exportdateien auf die Steuerung transferiert. Bei "großen" Projekten kann deren Erzeugen und Generieren einige Zeit in Anspruch nehmen. Diese Aktionen werden nach dem eigentlichen Download der Applikation gestartet und nebenläufig ausgeführt. Damit sind die Auswirkungen auf den OnlineChange-, Download- und Monitoring-Betrieb minimal.

Damit die Daten für die "Übersicht über den Vergleich" der SPS-Objekte auf die Steuerung übertragen werden, muss die Option "Online-Vergleich aktivieren" gesetzt sein ([Optionen auf Seite 54](#)).

Bei deaktivierter Option werden keine Daten auf die Steuerung übertragen. Wird der Online-Vergleich mit dieser Einstellung gestartet, werden die SPS-Objekte als "nicht vergleichbar" dargestellt. Diese Option ist standardmäßig eingeschaltet.

Sollen zusätzlich die Quellen für den "Detaillierten Vergleich" übertragen werden, muss die Option "Quelldateien übertragen" aktiviert sein. Diese Option ist standardmäßig ausgeschaltet.

Sind die Optionen entsprechend gesetzt, werden die Daten für den Online-Vergleich nach dem Übertragen des geänderten Codes auf die Steuerung erzeugt (ausgelöst durch **Debuggen ▶ Einloggen**) und auf der Steuerung transferiert. Ist das Bootprojekt aktuell (also z.B. nach einem OnlineChange mit Download des Bootprojektes), werden auch die Quellen übertragen. Ist das Bootprojekt nicht aktuell (z.B. nach einem OnlineChange ohne Download des Bootprojektes) kann der Anwender das Kommando **Debuggen ▶ Bootapplikation erzeugen** ausführen und damit das Übertragen der Quellen auf die Steuerung auslösen.

WARNUNG

Schutz der Daten auf der Steuerung

Die übertragenen Daten sind derzeit lediglich mit einem Passwort geschützt. Jeder Anwender kann mit der entsprechenden IndraWorks-Version über die Online-Vergleichsfunktionalität auf die Quelldaten zugreifen.

Anwender, die dies verhindern wollen, müssen das Übertragen der Quellen für jede Steuerung separat im Eigenschaftendialog der Steuerung im Registerblatt "SPS" mit Hilfe der **SPS-Quellen zusammen mit Boot-Applikation übertragen** deaktivieren

Wird der Online-Vergleich gestartet oder aktualisiert, werden die auf der Steuerung abgelegten Daten auf den Engineering PC übertragen und für die "Übersicht über den Vergleich" bzw. für den "Detaillierter Vergleich" ausgewertet.

5.1 Übersicht über den Vergleich

Die SPS - Objekte werden anhand ihres Namens zugeordnet.

Status gleich: Zwei Objekte erhalten des Status "gleich", wenn die auf der Steuerung für das Objekt abgelegte Checksumme mit der im Projekt ermittelten Checksumme übereinstimmt.

Status ungleich: Sind die Checksummen der Objekte unterschiedlich, wird geprüft ob die Änderung zu einem OnlineChange oder Download führen wird. Ist dies der Fall, wird "ungleich" angezeigt.

Status ungefähr gleich: Wurde z.B. lediglich ein Kommentar geändert oder eine Eigenschaft modifiziert wird der Status "ungefähr gleich" angezeigt.

Über den Tooltip können zusätzliche Informationen eingesehen werden:

1. Am "Logic"-Knoten wird eine Warnung angezeigt, wenn die Daten für den Online-Vergleich nicht oder nicht vollständig auf die Steuerung übertragen wurden.
2. Am "Applikation" Knoten wird eine Warnung angezeigt, wenn das Bootprojekt nicht aktuell ist (dann kann keine Detailvergleich durchgeführt werden, weil immer nur die Quellen für das Bootprojekt auf die Steuerung übertragen werden.)
3. Im Tooltip des Vergleichs am Applikationsobjekt wird angezeigt, ob die Applikation per OnlineChange eingeloggt werden kann.
4. Hat sich z.B. lediglich ein Kommentar in einer POU geändert, wird im Tooltip angezeigt, dass diese Änderung keine Auswirkung auf das Kompilat hat.
5. Stammen die beiden Objekte aus unterschiedlichen Projekten (Projekt1 mit POU "Name1" wurde auf die Steuerung geladen und das Projekt2 ist geöffnet, in dem ebenfalls eine POU mit dem Namen "Name1" enthalten ist) werden die Objekte auch dann als unterschiedlich angezeigt, wenn sie identischen Inhalt haben. Dies ist deshalb der Fall, weil die Objekte aufgrund unterschiedlicher Guids eine unterschiedliche Checksummen haben. Im Tooltip wird angezeigt, dass die Objekte unterschiedliche IDs haben.
6. Ist ein Objekt jeweils nur auf einer Seite vorhanden, wird der Status "ungleich" angezeigt.
7. Safetyobjekte erhalten den Status "nicht vergleichbar"

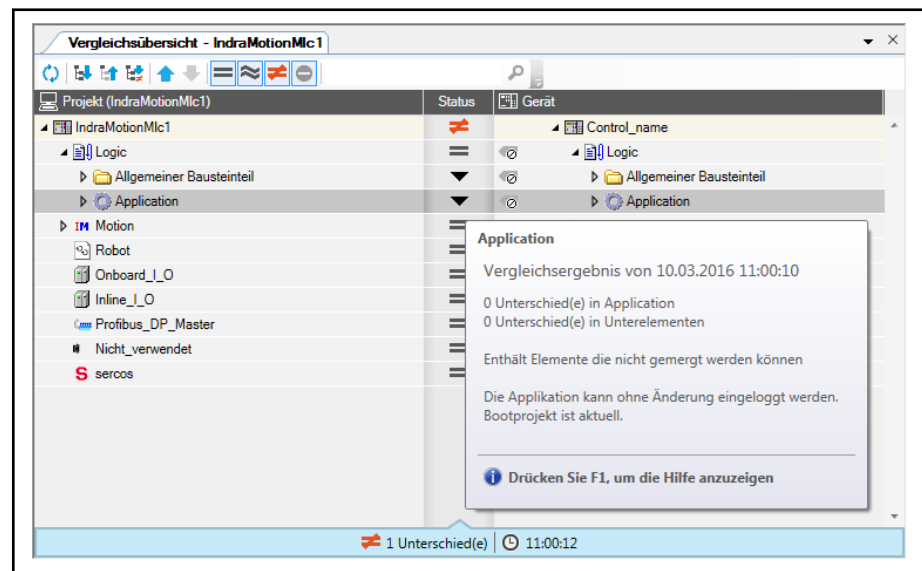


Abb. 5-1: Beispiel für eine "Vergleichsübersicht"

5.2 Detailvergleich

Der "Detaillierte Vergleich" wird durch doppelklicken auf einer Zeile der Vergleichsübersicht geöffnet bzw. aktualisiert, wenn bereits ein geöffneter Detail-

vergleich vorhanden ist. Er wird im unteren Bereich des Fensters der Vergleichsübersicht angezeigt.

Liegen keine gültigen Quellen auf der Steuerung ab, wird dies im geöffneten Fenster angezeigt:

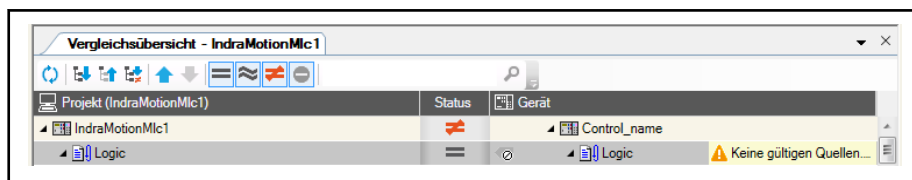


Abb. 5-2: Dateivergleich ohne gültigen Quellen auf der Steuerung

Dies kann dann der Fall sein, wenn die entsprechende Option nicht aktiviert ist, die Quellen auf der Steuerung gelöscht wurden oder das Bootprojekt nicht aktuell ist (wird als Warnung am Applikationsknoten visualisiert).

Liegen Quelldaten auf der Steuerung ab, wird ein Detailvergleich ausgeführt, der analog zum Vergleich mit Exportdateien funktioniert.

Stammen die zu vergleichenden Objekte aus unterschiedlichen Projekten, d.h. das Projekt auf Steuerung wurde mit einem Projekt geladen, das aktuell nicht im Engineering PC geöffnet ist, werden diese im Strukturvergleich, wie unter [Übersicht über den Vergleich auf Seite 51](#) beschrieben als ungleich dargestellt. Beim Detailvergleich wird oberhalb des eigentlichen Vergleichs angezeigt, dass dies der Fall ist.

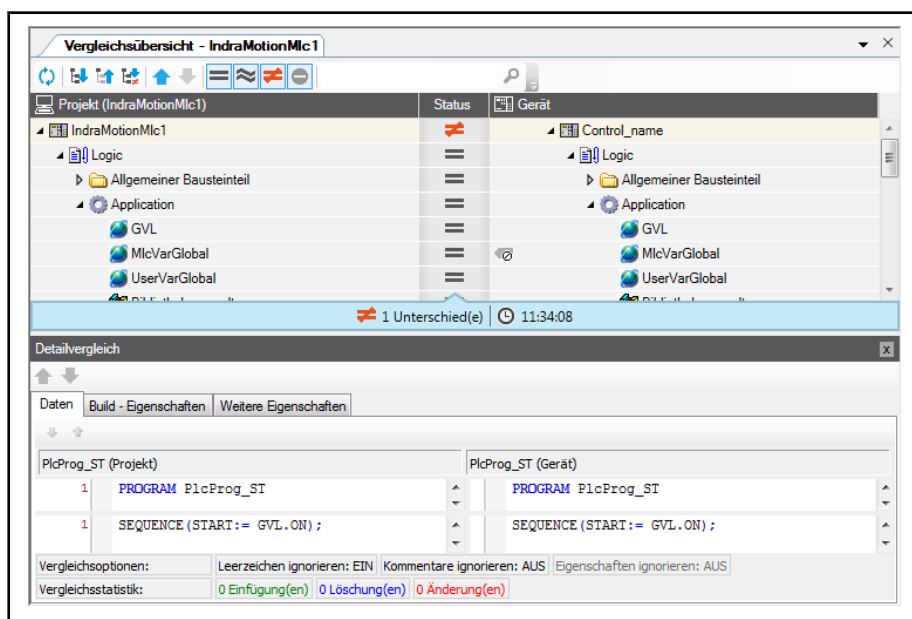


Abb. 5-3: Zu vergleichenden Objekte aus unterschiedlichen Projekten

Ist ein Objekt lediglich auf einer Seite des Strukturvergleichs vorhanden, wird im Detailvergleich dort die verfügbare Quelle angezeigt und auf der anderen Seite der Hinweis "Keine Quelldaten verfügbar".

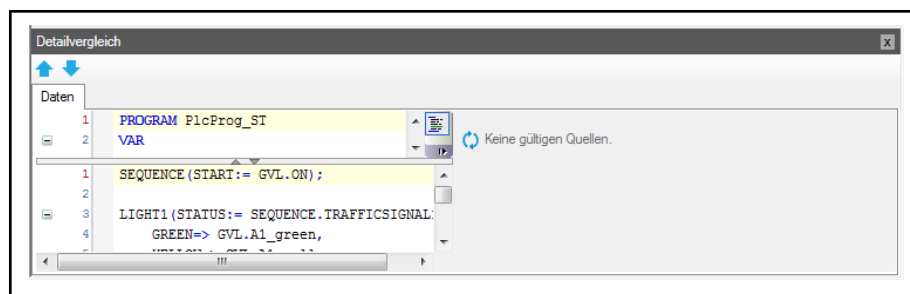


Abb. 5-4: Objekt nur im geöffneten Projekt des Engineering PC vorhanden:

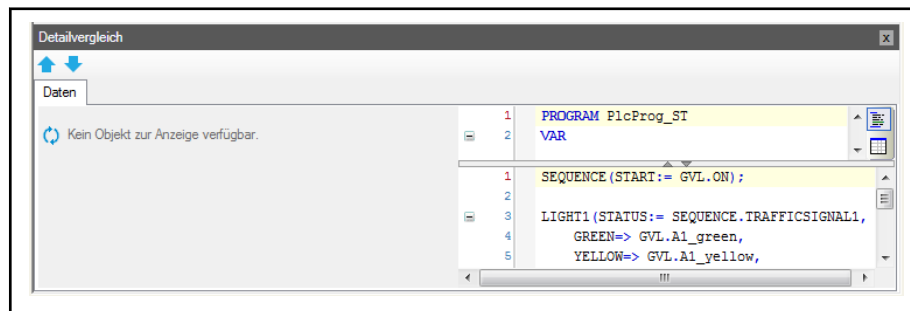


Abb. 5-5: Objekt nur auf der Steuerung vorhanden

**Hinweise für den Abgleich:**

Wird am Applikationsknoten die Warnung "Applikationsnamen unterschiedlich" angezeigt, kann IndraWorks nicht feststellen, ob das Bootprojekt auf der Steuerung aktuell ist.

Der Anwender muss dann die Applikation im geöffneten Projekt umbenennen (analog zum Namen der Applikation auf der Steuerung). Damit kann dann festgestellt werden, ob die Quellen auf der Steuerung vorhanden sind und zur laufenden Applikation passen. Ist das der Fall kann der Anwender einen Abgleich durchführen.

5.3 Optionen

In den Optionendialogen stehen Ihnen Einstellmöglichkeiten zur Verfügung, mit denen Sie die Vergleichsmöglichkeiten für den **Projekt-Vergleich** individuell anpassen können. Um in den Optionendialog zu gelangen, wählen Sie das Menü **Extras ► Optionen....**

Knoten Vergleich

Texteinstellungen	Hier können Sie die Farbe und die Schriftart für den Vergleich anpassen
Externe Anwendung	Hier können Sie eine externe Vergleichsanwendung angeben, um Unterschiede in Dateien anzuzeigen. Voraussetzung ist, dass die Vergleichsanwendung auf dem PC installiert ist. Klicken Sie auf die Schaltfläche "..." und wählen Sie im "Öffnen-Dialog" die externe Anwendung aus.
SPS-Optionen	Hier können Sie festlegen, wie sich der Vergleich bei SPS-Elementen verhält. Sie können angeben, ob Leerzeichen, Kommentare oder Eigenschaften berücksichtigt werden sollen.

Für den **Online-Vergleich** finden Sie im Kontextmenü des Eigenschaftsdialogs die Einstellung, um die SPS-Programmquellen auf die Steuerung zu übertragen.

Steuerung ► Eigenschaften ► SPS

SPS-Quellen zusammen mit Boot-Applikation übertragen	Damit die Programmquellen verglichen werden können, muss zusätzlich diese Option aktiv sein.

5.4 Fehlereinträge im Meldungsfenster

Die Erzeugung und Übertragung der Daten für den Onlinevergleich auf eine Steuerung wird durchgeführt, während der Anwender bereits weitere Aktionen im Projekt ausführen kann. Dadurch kann es vorkommen, dass für den Onlinevergleich notwendige Vorgänge noch nicht abgeschlossen wurden und bereits neue Aktivitäten anstehen.

Der Anwender wird mit Fehlermeldungen in der Kategorie "Online-Vergleich" im Meldungsfenster darauf hingewiesen. Diese Fehlermeldungen beziehen sich **ausschließlich** auf den Onlinevergleich, d.h. wird z.B. die Meldung angezeigt:

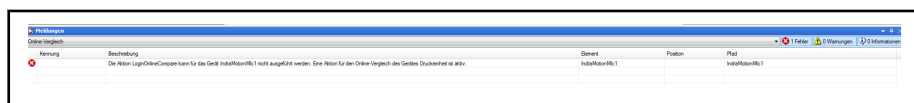


Abb. 5-6: Beispiel für eine Meldung in der Kategorie "Online-Vergleich"

"Die Aktion LoginOnlineCompare kann für das Gerät IndraMotionMlc1 nicht ausgeführt werden. Eine Aktion für den Online-Vergleich des Gerätes IndraMotionMlc1 ist aktiv.", wurde das Kommando "Login" für die Applikation korrekt ausgeführt (d.h. der Code wurde korrekt auf die Steuerung übertragen!). Jedoch konnten die für den Online-Vergleich notwendigen Aktionen (Erzeugung und Übertragung der Struktur- und nicht durchgeführt werden. Ein anschließender Online-Vergleich wird nicht korrekt funktionieren, weil die aktuellen Daten für den Onlinevergleich nicht auf der Steuerung vorliegen.

Folgende Meldungen können angezeigt werden:

- Die Aktion LoginOnlineCompare kann nicht...
- Die Aktion ClearOnlineCompare kann nicht...
- Die Aktion ClearAllOnlineCompare kann nicht...

- Die Aktion CreateBootApplicationOnlineCompare kann nicht...
- Die Aktion ResetOriginOnlineCompare kann nicht...

Anwender, die den Online-Vergleich nicht verwenden, können diese Meldung ignorieren. Anwender die den Online-Vergleich verwenden wollen, müssen den Vorgang wiederholen. Tritt der Fehler weiterhin auf, sollte das Projekt geschlossen und erneut geöffnet werden, bevor der Vorgang wiederholt wird.

6 Projekt schützen und speichern

Allgemein zu Schreib- und Zugriffsschutz

Sie können ein Projekt über Zugriffs- und Schreibschutz vor versehentlicher Änderung schützen. Sie können es zusätzlich mit einem Leseschutz (Know-How-Schutz) versehen.

Schreibschutz

Um das gesamte Projekt mit einem einfachen Schreibschutz zu versehen, gibt es folgende Möglichkeiten:

- Sie setzen in den **Bibliotheksinformationen** (Ordner "Allgemeiner Bausteinteil") den Status **Freigegeben**.
- Sie aktivieren in den Eigenschaften der Projektdatei im lokalen Dateisystem die Option "Schreibgeschützt".

Um nur bestimmte Objekte in einem Projekt vor Änderungen zu schützen, oder nur bestimmten Anwendern Zugriff zu gewähren, können Sie eine Benutzer- und Zugriffsrechteverwaltung (siehe unten) einsetzen.

Schreibschutz und Zugriffsschutz dienen jedoch nicht als Know-How-Schutz der Bausteine!

Sowohl IndraLogic selbst als auch Personen mit Kenntnissen des Projektdatei-Formats können mit IndraLogic erzeugte Bausteine ansehen oder modifizieren.

Know-how-Schutz

Know-How-Schutz eines Projekts erreichen Sie durch Verschlüsselung der Projektdatei. Die Verschlüsselung kann entweder mit einem Projekt-Passwort oder mit Hilfe eines Zertifikats realisiert werden.

Der Schutz über das Zertifikat wird empfohlen, da in diesem Fall kein Geheimnis zwischen den berechtigten Anwendern geteilt werden muss. Der gewünschte Projektverschlüsselungstyp wird in den Projekteinstellungen aktiviert.

Know-How-Schutz einer Bibliothek erreichen Sie durch Bereitstellung als ziel-systemunabhängige "Geschützte Bibliothek" (*.compiled-library). Die Bibliotheksdatei beinhaltet in diesem Format keinen Quellcode mehr, sondern nur noch verschlüsselten Precompile-Kontext. Der Compiler ist weiterhin in der Lage, diese Daten zu interpretieren. Ob noch Zugriff durch andere IndraLogic-Komponenten möglich ist, hängt von deren Funktionalität ab und ist im Einzelfall zu beachten.

Know-How-Schutz und Kopierschutz einer Boot-Applikation erreichen Sie mit Hilfe eines Zertifikats. Die Aktivierung erfolgt in den Objekt-Eigenschaften der Applikation.

Siehe auch

- [Kap. 6 "Projekt schützen und speichern" auf Seite 57](#)
- [Kap. 6.3 "Objekte im Projekt über Zugriffsrechte schützen" auf Seite 62](#)
- [Kap. 16.1 "Informationen für Bibliotheksentwickler" auf Seite 236](#)

Benutzerverwaltung und Passwort-Manager

In IndraLogic können Benutzerkonten mit unterschiedlichen Rechten verwaltet werden. Für jedes Konto können Sie festlegen, mit welchen Aktionen der Benutzer auf ein Projektobjekt zugreifen darf.

Sie konfigurieren die Benutzerverwaltung in den **Projekteinstellungen** in der Kategorie **Benutzer und Gruppen**.

Vor dem Anlegen von Benutzern und Gruppen beachten Sie bitte Folgendes:

- Rechte können nur an Benutzergruppen vergeben werden. Sie müssen also jeden Benutzer einer Gruppe zuordnen.

- Automatisch gibt es immer eine Gruppe "Everyone" und standardmäßig ist zunächst jeder Benutzer und jede andere Gruppe Mitglied dieser Gruppe. Somit ist jedes Benutzerkonto automatisch wenigstens mit den definierten Standardrechten ausgestattet.

Sie können die Gruppe "Everyone" nicht löschen, nur umbenennen, und Sie können keine Mitglieder aus dieser Gruppe entfernen.

Vorsicht: Standardmäßig hat "Everyone" nicht das Recht, die aktuelle Benutzer-, Gruppen- und Rechtekonfiguration zu ändern!

- Automatisch gibt es immer eine Gruppe "Owner", die einen Benutzer "Owner" enthält.

Ab V3.5 hat in einem neuen Projekt zunächst nur der "Owner" das Recht, die aktuelle Benutzer-, Gruppen- und Rechtekonfiguration zu ändern!

Somit kann nur "Owner" dieses Recht an eine andere Gruppe vergeben.

Zunächst kann der "Owner" mit Benutzernamen "Owner" und leerem Passwort einloggen. Sie können weitere Benutzer der Gruppe "Owner" hinzufügen oder aus dieser entfernen, aber mindestens ein Mitglied muss enthalten bleiben. Die Gruppe "Owner" können Sie - wie "Everyone" - nicht löschen, und sie besitzt immer alle Zugriffsrechte. So wird verhindert, dass ein Projekt durch das Verneinen aller Zugriffsrechte für alle Gruppen unbrauchbar gemacht wird.

Sie können sowohl die Gruppe "Owner" als auch den Benutzer "Owner" umbenennen.

- Wenn das Programmiersystem oder ein Projekt neu gestartet wird, ist zunächst kein Benutzer im Projekt eingeloggt. Dann aber kann sich der Anwender über ein bestimmtes Benutzerkonto mit Benutzernamen und Passwort einloggen um die für das Konto definierten Zugriffsrechte zu bekommen.
- Jedes Projekt hat seine eigene Benutzerverwaltung! Um also beispielsweise bestimmte Zugriffsrechte auf eine im Projekt eingebundene Bibliothek zu bekommen, muss sich der Anwender explizit in das Bibliotheksprojekt einloggen.

Benutzer und Gruppen, die in verschiedenen Projekten definiert sind, sind nicht gleich, auch wenn sie gleiche Namen haben.

- Eine Benutzerverwaltung in einem Projekt ist nur sinnvoll, wenn sie mit entsprechender Rechtevergabe für den Zugriff auf Projekt und Objekte verbunden wird. Die allgemeine Rechteverwaltung für ein Projekt erfolgt im Dialog **Rechte** der **Benutzerverwaltung**. Sie können die Zugriffsrechte auf ein einzelnes Projektobjekt außerdem auf der Registerkarte **Zugriffskontrolle** der **Eigenschaften** des Objekts verändern.
- Für das Anmelden und Abmelden in/von einem Projekt als definierter Benutzer gibt es standardmäßig Menübefehle unter **Projekt ▶ Benutzerverwaltung**. Ein Passwort-Manager erlaubt das Verwalten der Zugangsdaten auf Ihrem Computer.



Ab V3.5 hat in einem neuen Projekt zunächst nur der "Owner" das Recht, die aktuelle Benutzer-, Gruppen- und Rechtekonfiguration zu ändern! Somit kann nur "Owner" dieses Recht an eine andere Gruppe vergeben.



IndraLogic speichert die Benutzer-Passwörter unzugänglich. Wenn Sie ein Passwort vergessen, wird das Benutzerkonto unbrauchbar. Wenn Sie das "Owner"-Passwort vergessen, wird möglicherweise das gesamte Projekt unbrauchbar!

Passwort-Manager

Der Passwort-Manager ermöglicht es, die Zugangsdatensätze zu speichern, die Sie während der Anmelde-Prozeduren für Projekte eingeben. Er ist über eine Schaltfläche im Anmeldedialog zugänglich und bietet schnellen Zugriff auf die gerade benötigten Zugangsdaten. Dies kann beispielsweise hilfreich sein, wenn Sie parallel an einigen Bibliotheksprojekten arbeiten, die durch verschiedene Passwörter geschützt sind.

Der Passwort-Manager selbst ist durch ein individuelles Master-Passwort geschützt. Wenn Sie den Passwort-Manager erstmalig benutzen möchten, fordert IndraLogic Sie auf, dieses Passwort im Passwort-Manager-Konfigurationsdialog zu definieren. IndraLogic merkt sich das Master-Passwort, bis Sie die aktuelle IndraLogic-Sitzung beenden. Sie müssen das Passwort immer eingeben, wenn Sie das erste Mal während einer neuen Sitzung in den Passwort-Manager einloggen möchten oder nachdem Sie es geändert haben.

Siehe auch

- [Kap. 6.3 "Objekte im Projekt über Zugriffsrechte schützen" auf Seite 62](#)
- [Kap. 6.4 "Anmeldung über Benutzerkonto und Passwort-Manager" auf Seite 62](#)

Rechteverwaltung

Eine Rechteverwaltung für den Zugriff auf Projekt und Objekte in einem Projekt ist nötig, um eine Benutzerverwaltung sinnvoll zu machen.

Die allgemeine Rechteverwaltung für ein Projekt konfigurieren Sie im Editor **Rechte** der **Benutzerverwaltung**. Die Zugriffsrechte auf ein einzelnes Projektobjekt können Sie außerdem auf der Registerkarte **Zugriffskontrolle** des Dialogs **Eigenschaften** des Objekts verändern.

Vor der Rechtevergabe beachten Sie bitte Folgendes:

- In einem neuen Projekt setzt IndraLogic grundsätzlich alle Rechte zur Ausführung von Aktionen auf Objekte mit dem **Standardwert** "erlaubt" (Standardrecht).
Einzige Ausnahme ist das Recht auf Änderung der aktuellen Benutzer-, Gruppen- und Rechtekonfiguration. Dieses hat zunächst nur die "Owner"-Gruppe.
- Wenn Sie Mitglied einer Gruppe sind, die Rechte ändern darf, können Sie das beim weiteren Arbeiten auf einem Projekt jederzeit für jedes Recht tun.
Sie ändern ein Recht durch Umsetzen zwischen "erlaubt" und "verboten" bzw. durch Zurücksetzen auf den Standard.

Siehe auch

- [Kap. 6.2 "Benutzerverwaltung einrichten" auf Seite 61](#)
- [Kap. 6.3 "Objekte im Projekt über Zugriffsrechte schützen" auf Seite 62](#)

Ablage, Speichern

Versehen Sie die Projektdatei vor dem Speichern im Dateisystem mit dem gewünschten Schutz, siehe oben. Für eine schreibgeschützte Projektdatei erhalten Sie je nach Art des Schreibschutzes verschiedene Optionen, die Datei dennoch zu speichern.

Wenn das Projekt später in einer älteren IndraLogic-Version geöffnet werden soll, ist es sinnvoll, das Projekt genau für diese Version (Dateityp) zu speichern, da IndraLogic Sie im Zuge dieser Speicherung auch sofort über eventuelle Datenverluste informiert.

Falls Sie Bibliotheksprojekte speichern möchten, beachten Sie die Regeln zur Erstellung von Bibliotheken. Erwägen Sie auch die Möglichkeit, eine Bibliothek direkt in ein Bibliotheks-Repository zu installieren.

Wenn Sie ein Projekt auf einem anderen Computer weiter verwenden möchten, ist es sinnvoll, nicht nur die Projektdatei zu speichern, sondern ein Projektarchiv aus allen relevanten Zusatzdateien zu erstellen.

Sie können einstellen, dass bei jedem Speichern des Projekts zusätzlich eine Sicherungskopie dieses Projekts angelegt wird. Außerdem können Sie konfigurieren, dass Projekte generell in bestimmten Zeitintervallen automatisch gespeichert werden.

Siehe auch

- [Kap. 3 "Projekt exportieren und transferieren" auf Seite 15](#)
- [Kap. 6.5 "Projekt speichern" auf Seite 65](#)
- [Kap. 16.1 "Informationen für Bibliotheksentwickler" auf Seite 236](#)

6.1 Schreibschutz einrichten

Ein Projekt kann über Zugriffs- und Schreibschutz vor versehentlicher Änderung geschützt werden. Es kann aber auch zusätzlich einen Leseschutz (Know-How-Schutz) erhalten. Sie haben folgende Möglichkeiten

Projekt mit Attribut "Freigegeben" versehen

Voraussetzung: Projekt ist geöffnet.

1. Wählen Sie **<Projektname> ► Ordner "Allgemeiner Bausteinteil" ► Bibliotheksinformationen**, dann Registerkarte **Zusammenfassung**.
2. Aktivieren Sie die Option **Freigegeben**, bestätigen Sie mit **OK**.
3. Speichern Sie das Projekt, beispielsweise mit **<Strg>+<S>**.



Das Metadatum **Freigegeben** wirkt sowohl bei Bibliotheksprojekten als auch in Standardprojekten!

So wird Ihnen beispielsweise in einem freigegebenen Standardprojekt verwehrt eine neue Liste mit **Persistenten Variablen** abzufragen. Dies gelingt Ihnen erst, wenn das Metadatum gelöscht wurde.

Siehe auch: [Kap. 9.16.5 "Speichern der Werte einer persistenten Variablenliste in einer Rezeptur" auf Seite 152](#).

Siehe auch

- [Kap. 18.1.20 "Objekt 'Bibliotheksinfo'" auf Seite 579](#)

Projekt im Dateisystem mit der Eigenschaft "Schreibgeschützt" versehen

1. Versehen Sie die Projektdatei in ihrem lokalen Dateisystem mit dem Eigenschaftsattribut "Schreibgeschützt".

Wenn Sie das Projekt bereits geöffnet hatten, erscheint nun beim Versuch, es unter gleichem Namen zu speichern, ein Dialog mit Hinweis auf den bestehenden Schreibschutz. In diesem Dialog erhalten Sie folgende Möglichkeiten:

Über die Schaltfläche **Speichern unter...** können Sie das Projekt unter einem anderen Namen bzw. anderen Pfad speichern.

Brechen Sie den Speichervorgang ab, eventuell um zuerst den Schreibschutz auf der Platte zu entfernen.

Wenn Sie das Projekt neu öffnen, erscheint rechts oben im Hauptfenster eine Zeile **Das Projekt kann nicht gespeichert werden...** Sie müssen nun eine der angebotenen Optionen wählen, falls Sie die Projektdatei speichern möchten.

6.2 Benutzerverwaltung einrichten



Es geht hier um die Benutzerverwaltung für eine IndraLogic-Projektdatei.

Die folgende Anleitung beschreibt, wie Sie in einem Projekt erstmalig die Benutzerverwaltung anpassen können. Es geht um das Definieren eines Benutzers und einer Gruppe, der er angehört.

Voraussetzung: Das Projekt, für das die Benutzerverwaltung eingerichtet werden soll, ist geöffnet. Es liegt noch keine angepasste Benutzerkonfiguration vor.

1. Wählen Sie **Projekt ▶ Benutzerverwaltung ▶ Benutzer und Gruppen** und dann Registerkarte **Benutzer**. Benutzer **Owner** ist standardmäßig bereits angelegt.
2. Klicken Sie auf **Hinzufügen**.
Der Dialog **Benutzer hinzufügen** erscheint.
3. Tragen Sie einen Anmeldenamen, beispielsweise "Dev1", und ein Passwort ein.
Lassen Sie die Option **Aktiviert** aktiviert. Klicken Sie **OK**.
Beim erstmaligen Anlegen einer Gruppe fordert IndraLogic Sie nun auf, sich als für diese Aktion berechtigt zu authentifizieren.
Geben Sie in diesem Fall "Owner" als **Benutzername** ein. Geben Sie kein **Passwort** ein und klicken Sie **OK**.
Benutzer **Dev1** erscheint in der Liste und ist automatisch "Mitglied der Gruppe 'Everyone'".
4. Wechseln Sie zur Registerkarte **Gruppen**, um den Benutzer einer neuen Gruppe hinzuzufügen.
Die Gruppen **Everyone** und **Owner** sind bereits angelegt.
5. Klicken Sie auf **Hinzufügen**, um den Dialog **Gruppe hinzufügen** zu erhalten.
6. Geben Sie zumindest einen Namen für die neue Gruppe an, beispielsweise "Developers". Haken Sie im Feld **Mitglieder** den Eintrag **Benutzer 'Dev1'** an. Klicken Sie **OK**.
Gruppe **Developers** erscheint nun mit "hat Benutzermitglied 'Dev1'".
7. Wechseln Sie zur Registerkarte **Benutzer**.
Benutzer **Dev1** erscheint nun als Mitglied von Gruppe 'Everyone' und 'Developers'.



Mit Hilfe der **Export/Import**-Funktionalität im Dialog **Projekteinstellungen** in Kategorie **Benutzer und Gruppen** können Sie die Benutzerverwaltung aus einem anderen Projekt übernehmen.

Siehe auch

- [Kap. 6 "Projekt schützen und speichern" auf Seite 57](#)
- [Kap. "Befehl 'Rechte'" auf Seite 639](#)

6.3 Objekte im Projekt über Zugriffsrechte schützen

Schützen einzelner Objekte durch Setzen von Zugriffsrechten im Editor 'Rechte'

1. Wählen Sie **Projekt ► Benutzerverwaltung ► Rechte**
Das Fenster des Editors **Rechte** erscheint. Links sehen Sie die Aktionen-Kategorien. Rechts die aktuell vorhandenen Benutzergruppen.
2. Klappen Sie die betreffende Aktionen-Kategorie auf und darunter die Aktion, für die Sie ein Recht ändern möchten.
3. Selektieren Sie im Fenster **Aktionen** das Ziel der Aktion. Selektieren Sie im Fenster **Rechte** die Gruppe, für die Sie das Recht ändern möchten. Mehrfachauswahl ist möglich.
Die Schaltflächen der Symbolleiste werden bedienbar.
4. Klicken Sie auf die entsprechende Schaltfläche, um das Recht der Gruppe für die Aktion auf das Zielobjekt zu ändern.
IndraLogic aktualisiert das Symbol vor der Gruppe, dem neuen Recht entsprechend. Das Recht ist unmittelbar wirksam.

Siehe auch

- [Kap. "Befehl 'Rechte'" auf Seite 639](#)

Schützen einzelner Objekte durch Setzen von Zugriffsrechten in den Objekteigenschaften

Hier können Sie konfigurieren, ob die Mitglieder einer Gruppe das Recht haben, das Objekt anzusehen, zu bearbeiten, zu entfernen oder ihm Kindobjekte hinzuzufügen oder zu entfernen.

1. Wählen Sie das Objekt im Projektextplorer aus.
2. Wählen Sie im Kontextmenü den Befehl **Eigenschaften** und im Dialog die Kategorie **Zugriffskontrolle**.
3. In der Tabelle unter **Gruppen, Aktionen und Rechte** doppelklicken Sie auf das Symbol des Rechts, das Sie ändern möchten.
Es erscheint eine Auswahlliste der möglichen Rechte: **Erlauben, Verbie-**
ten, Zurücksetzen.
4. Wählen Sie das gewünschte Recht und klicken **Übernehmen** oder **OK**
.
Das Recht ist unmittelbar für die Aktion und Gruppe wirksam. Das Symbol ändert sich entsprechend.

Siehe auch

- [Kap. "Dialog 'Eigenschaften- 'Zugriffskontrolle'" auf Seite 750](#)

6.4 Anmeldung über Benutzerkonto und Passwort-Manager

Anmelden in Projekt ohne Verwenden der Passwort-Manager-Funktionen

Voraussetzung: Ein Projekt ist geöffnet.

Sie möchten sich für dieses Projekt oder für eine darin eingebundene Bibliothek als definierter Benutzer anmelden, um es oder sie mit bestimmten Rechten zu bearbeiten. Sie haben für das jeweilige Projekt oder die Bibliothek die erforderlichen Zugangsdaten.

1. Wählen Sie **Projekt ► Benutzerverwaltung ► Benutzer anmelden....**
Der Dialog **Anmelden** erscheint.
2. Wählen Sie bei **Projekt/Bibliothek** die Projektdatei aus und geben die dafür erforderlichen Zugangsdaten **Benutzername** und **Passwort** ein.
3. Melden Sie sich mit **OK** an.
Wenn bereits ein anderer Benutzer angemeldet ist, wird dieser durch die neue Anmeldung automatisch abgemeldet.

Master-Passwort für Passwort-Manager setzen

*Voraussetzung: Ein Projekt ist geöffnet. Sie haben den Dialog **Anmelden** vor sich, um sich für ein Projekt oder eine darin eingebundene Bibliothek als definierter Benutzer anzumelden. Sie möchten den Passwort-Manager verwenden, um Zugangsdaten darin zu speichern.*

1. Wählen Sie **Projekt ► Benutzerverwaltung ► Benutzer anmelden....**
2. Klicken Sie im Dialog **Anmelden** auf die Schaltfläche .
Wenn Sie das erste Mal mit dem Passwort-Manager arbeiten, erscheint der Dialog **Passwort-Manager Konfiguration**.
3. Geben Sie eine Zeichenfolge als künftiges Master-Passwort ein. Bestätigen Sie es in der zweiten Zeile und klicken **OK**.
IndraLogic merkt sich das Master-Passwort, bis Sie die aktuelle IndraLogic Sitzung beenden. Sie müssen dieses Passwort immer eingeben, wenn Sie das erste Mal während einer neuen Sitzung in den Passwort-Manager einloggen möchten, oder nachdem Sie es geändert haben.



Wenn Sie Ihr Master-Passwort vergessen haben, haben Sie keine Möglichkeit mehr, auf die bereits gespeicherten Zugangsdaten zuzugreifen! In diesem Fall können Sie den Passwort-Manager nur noch zurücksetzen. Danach müssen Sie von Neuem beginnen, Passwörter im Manager zu speichern!

Anmeldedaten im Passwort-Manager speichern

Voraussetzung: Ein Projekt ist geöffnet. Sie möchten sich für dieses Projekt oder für eine darin eingebundene Bibliothek als definierter Benutzer anmelden, um es oder sie mit bestimmten Rechten zu bearbeiten. Sie haben für das jeweilige Projekt oder die Bibliothek die erforderlichen Zugangsdaten. Diese Zugangsdaten sind noch nicht im Passwort-Manager hinterlegt.

1. Wählen Sie **Projekt ► Benutzerverwaltung ► Benutzer anmelden....**, um den Dialog **Anmelden** zu erhalten.
2. Wählen Sie bei **Projekt/Bibliothek** die Projektdatei aus.
3. Geben Sie Benutzernamen und Passwort für das Projekt oder die Bibliothek ein.
4. Klicken Sie auf die Schaltfläche .
Wenn Sie das erste Mal mit dem Passwort-Manager arbeiten, werden Sie aufgefordert ein Master-Passwort zu definieren. Sehen Sie hierzu oben die Anleitung "Master-Passwort für Passwort-Manager setzen".
Wenn Sie das erste Mal in dieser IndraLogic Session den Passwort-Manager aufrufen, werden Sie aufgefordert das Master-Passwort einzugeben.
5. Wenn gefordert, geben Sie das Master-Passwort ein.

Das Menü des Passwort-Managers erscheint.

6. Wählen Sie die Option **Zugangsdaten lokal auf diesem Computer speichern**.

Die Anmeldung erfolgt. Die Daten sind im Passwort-Manager gespeichert.

Anmeldedaten aus dem Passwort-Manager holen

Voraussetzung: Ein Projekt ist geöffnet. Sie möchten sich für dieses Projekt oder für eine darin eingebundene Bibliothek als definierter Benutzer anmelden, um es oder sie mit bestimmten Rechten zu bearbeiten. Die dazu erforderlichen Zugangsdaten sind bereits im Passwort-Manager gespeichert.

1. Wählen Sie den Befehl **Projekt ► Benutzerverwaltung ► Benutzer anmelden....**, um den Dialog **Anmelden** zu erhalten.

2. Klicken Sie auf die Schaltfläche .

Wenn Sie das erste Mal mit dem Passwort-Manager arbeiten, werden Sie aufgefordert ein Master-Passwort zu definieren. Sehen Sie hierzu oben die Anleitung "Master-Passwort für Passwort-Manager setzen".

Wenn Sie das erste Mal in dieser IndraLogic Session den Passwort-Manager aufrufen, werden Sie aufgefordert das Master-Passwort einzugeben.

3. Wenn gefordert, geben Sie das Master-Passwort ein.

Das Menü des Passwort-Managers erscheint.

4. Wählen Sie den passenden Eintrag **Gespeicherte Zugangsdaten für <Benutzername> verwenden**.

5. Die Anmeldung erfolgt automatisch mit den aus dem Passwort-Manager gelesenen Daten.

Passwort-Manager öffnen, Master-Passwort ändern

Voraussetzung: Ein Projekt ist geöffnet. Sie möchten den Passwort-Manager öffnen, um die Einträge zu sehen, zu bearbeiten oder das Master-Passwort zu ändern. Sie haben sich bereits einmal mit dem Master-Passwort eingeloggt.

1. Wählen Sie **Projekt ► Benutzerverwaltung ► Benutzer anmelden....**, um den Dialog **Anmelden** zu erhalten.

2. Klicken Sie die Schaltfläche .

Wählen sie **Passwort-Manager öffnen**.

Das Fenster des Passwort-Managers erscheint.

3. Klicken Sie auf **Master-Passwort ändern** und nehmen die Änderung vor.

Abmelden aus Projekt

*Voraussetzung: Ein Projekt ist geöffnet. Ein Benutzer ist angemeldet, was an einem Namenseintrag im Feld **Aktueller Benutzer** in der Statusleiste erkennbar ist.*

1. Wählen Sie **Projekt ► Benutzerverwaltung ► Benutzer abmelden....** Alternativ führen Sie einen Doppelklick auf das Feld **Aktueller Benutzer** in der Statusleiste aus.

Wenn der Benutzer nur in einem einzigen Projekt angemeldet ist, wird er nun ohne weitere Interaktion abgemeldet. Im Feld **Aktueller Benutzer** in der Statusleiste erscheint wieder -

Wenn der Benutzer in mehreren Projekten angemeldet ist, erscheint der Dialog **Abmelden**. Wählen Sie dort konkret das Projekt oder Bibliotheksprojekt aus, für das der Benutzer abgemeldet werden soll.

Siehe auch

- [Kap. 6 "Projekt schützen und speichern" auf Seite 57](#)
- [Kap. "Befehl 'Benutzer anmelden'" auf Seite 635](#)
- [Kap. 6.2 "Benutzerverwaltung einrichten" auf Seite 61](#)

6.5 Projekt speichern

Projekt unter gleichem Namen speichern

Voraussetzung: Das Projekt ist geöffnet. Es besteht kein Schreibschutz für die Projektdatei.

1. Wählen Sie **Projekt ► Speichern**.

IndraLogic speichert die Projektdatei mit dem aktuellen Projektnamen, der in der Titelleiste des Hauptfensters steht. Falls das Projekt seit dem letzten Speichern geändert wurde, ist der Projektname mit einem Sternchen versehen.

Projekt unter anderem Namen oder Format speichern

Voraussetzung: Das Projekt ist geöffnet.

1. Wählen Sie **Projekt ► Speichern unter....**

Der Dialog **Projekt speichern** erscheint.

2. Wählen Sie einen Ablageort im Dateisystem und den gewünschten **Dateityp** (Projektdatei (*.xiwp) oder Bibliotheksdatei (*.xiwp)) und die gewünschte Speicherversion. Wenn Sie das Projekt später in einer älteren Version öffnen möchten, ist es sinnvoll genau für diese Version zu speichern, da dann im Meldungsfenster sofort über eventuelle Datenverluste informiert wird.

Wenn die Projektdatei keinem Schreibschutz unterliegt, speichert IndraLogic sie im ausgewählten Pfad. Andernfalls erhalten Sie Hinweise, wie weiter zu verfahren ist.



Ein Schreibschutz, der in IndraLogic vergeben wurde (**Ordner "Allgemeiner Bausteinteil" ► Bibliotheksinfo ► Zusammenfassung "Freigegeben"**) verändert das Verhalten nicht.

Projekt speichern nach Erinnerung

Voraussetzung: Das Projekt ist geöffnet.

1. Wählen Sie **Extras ► Optionen ► Allgemein ► Projekte**.

Projekt speichern erscheint.

2. Aktivieren Sie die Option **Sicherungskopie erzeugen**.
3. Aktivieren die Option **Bei Änderungen in diesem Minutenintervall an das Speichern erinnern** und wählen Sie ein Zeitintervall.
4. Schließen Sie den Dialog **Optionen** mit **OK**.

Siehe auch

- [Kap. 6 "Projekt schützen und speichern" auf Seite 57](#)

7 Mehrsprachige SPS-Kommentare

SPS-Kommentare können in unterschiedlichen Sprachen angezeigt werden. Für diese Funktionalität werden die SPS-Lokalisierungsdaten genutzt.



SPS-Kommentare können in verschiedenen Landessprachen dargestellt werden. Ein Editieren ist jedoch (z.Zt.) nur in der unter Optionen eingestellten Sprache der Benutzeroberfläche möglich, die beim Erzeugen des Projektes eingestellt war.

Nach Hinzufügen einer zusätzlichen Sprache über das Menü **Projekt ► Sprache ► Projektsprachen verwalten** erscheint der Knoten „SPS-Lokalisierungsdaten“ unter dem „Allgemeiner Baustein“. Über diesen Knoten kann die SPS-Lokalisierungstabelle geöffnet werden. Für jede Landessprache existiert eine separate Spalte.

Das Hinzufügen einer neuen Landessprache geschieht folgendermaßen:

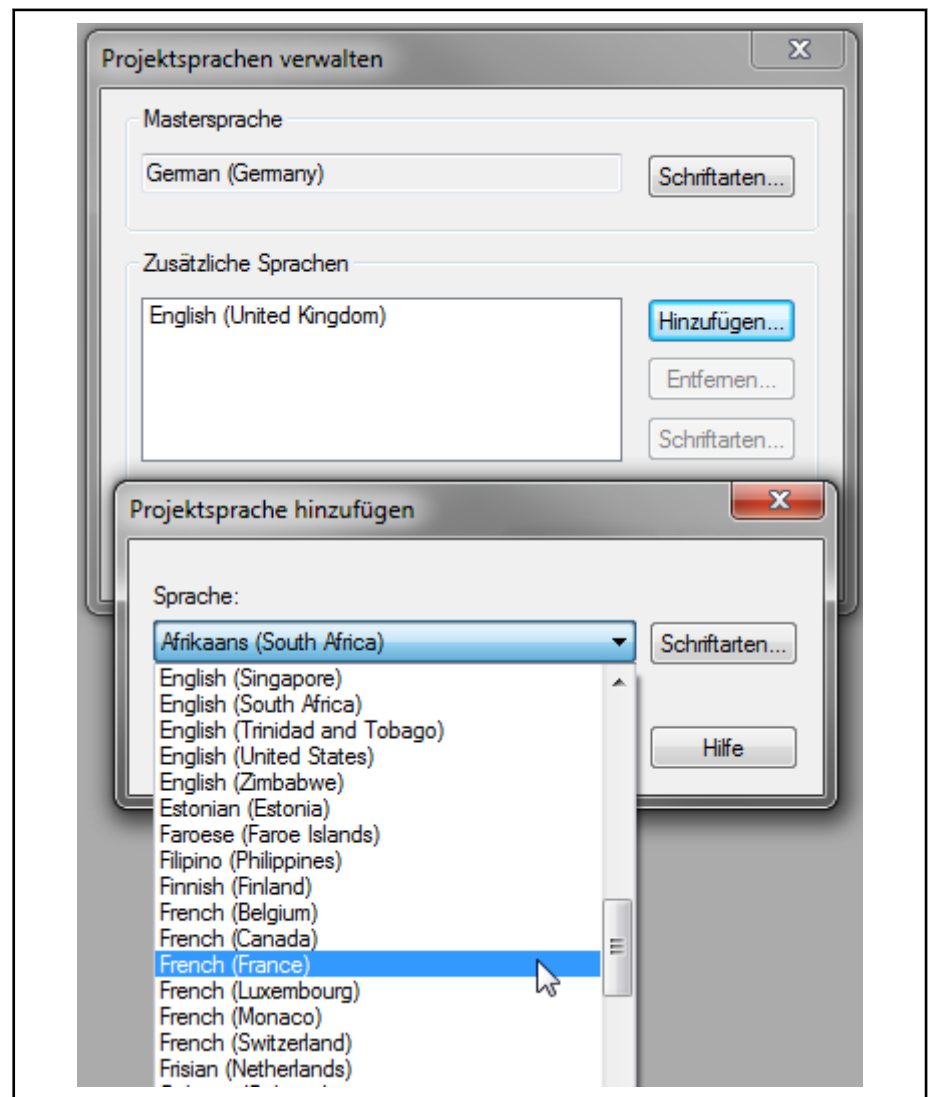


Abb. 7-1: Projektsprache hinzufügen

Mehrsprachige SPS-Kommentare

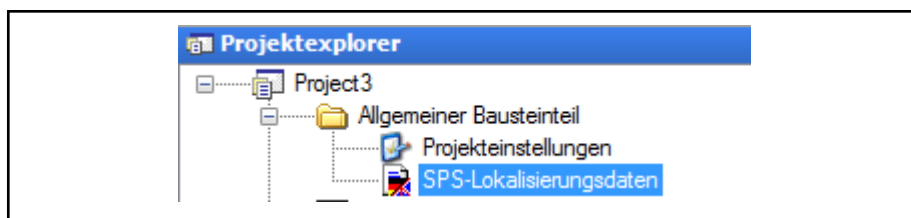


Abb. 7-2: SPS-Lokalisierungsdaten Knoten aktiviert

Eine Übersetzung der SPS-Kommentare in eine andere Landessprache erfolgt über die Exportdatei (Excel-File). Für jede Landessprache kann über den Menüpunkt **Projekt** ► **Sprache** ► **Übersetzungsdatei exportieren** eine eigene Übersetzungsdatei generiert werden. Diese enthält sowohl die zur Übersetzung notwendige, frei wählbare Ausgangssprache als auch die o.g. Landessprache als Zielsprache. Mit einem entsprechenden Editor (z.B. Excel) kann die so erzeugte csv-Datei bearbeitet und anschließend importiert werden. Der Import erfolgt über **Projekt** ► **Sprache** ► **Übersetzungsdatei importieren**.

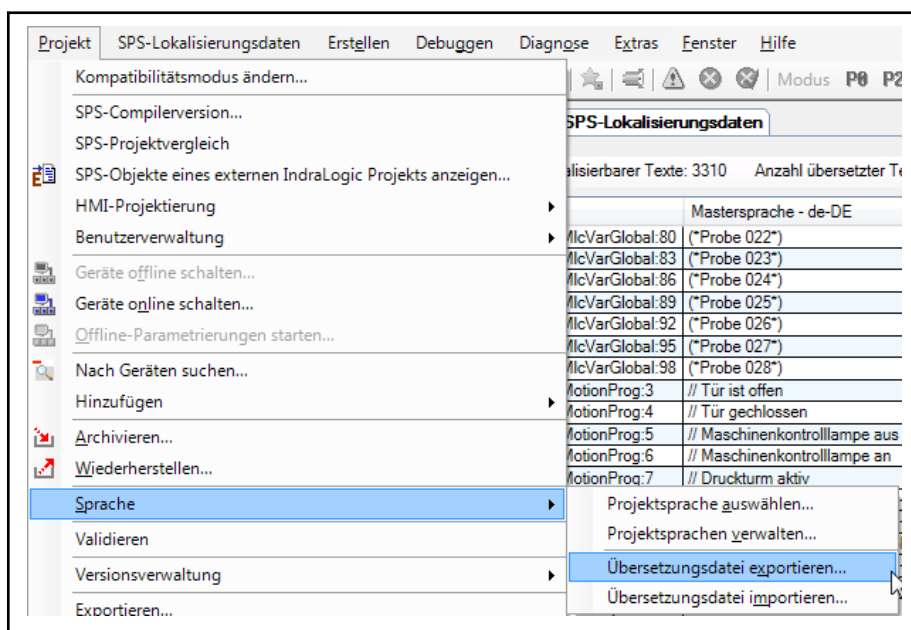


Abb. 7-3: Übersetzungsdatei exportieren

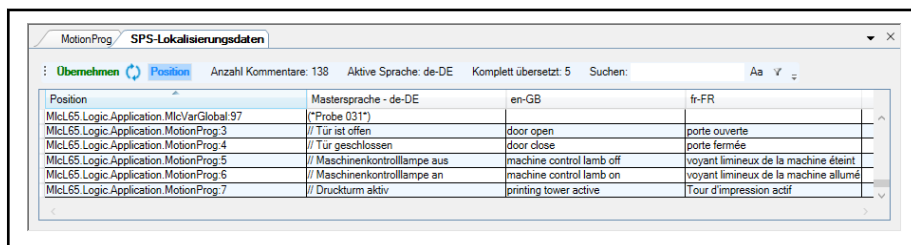


Abb. 7-4: SPS-Lokalisierungsdaten Tabelle in verschiedenen Sprachen

Durch einen Doppelklick in eine Zelle kann hier die Übersetzung der Kommentare direkt eingegeben werden. Das Betätigen des Menüpunktes **Übernehmen** bewirkt, dass die übersetzten Kommentare auch in den geöffneten Editoren aktualisiert werden.

Mit dem Menüpunkt **Position** in der SPS-Lokalisierungsdaten-Tabelle kann der Programmpfad auf die Kommentare aus- bzw. eingeblendet werden. Sind die Programmpfade ausgeblendet, erscheint der jeweilige Programmpfad im Tooltip.

Ein Doppelklick auf ein Originalkommentar öffnet die Position und markiert die Fundstelle in der POE.

Durch das Selektieren in der obersten Zeile lassen sich die Kommentare sortieren.

Mit dem Menüpunkt **Suchen**: lassen sich ganze Kommentare oder einzelne Zeichenfolgen suchen. Die Fundstellen werden farbig markiert und durch das Trichtersymbol können die restlichen Zeilen eingeblendet und ausgeblendet werden. Der Menüpunkt **Aa** erlaubt eine Suche, bei der die Groß- Kleinschreibung berücksichtigt wird.

Über **Projekt ► Sprache ► Projektsprache auswählen** wird die Sprache ausgewählt, in der die Kommentare erscheinen sollen. Die Statuszeile zeigt zusätzlich die gewählte Sprache an.

Ist eine andere Sprache als die Mastersprache ausgewählt, dann wird die Sprache automatisch in die Mastersprache zurück geschaltet, wenn Kommentare editiert werden.

8 E/A-Anbindung konfigurieren

Sie können in Ihrem IndraLogic-Projekt die zu steuernde Hardware mit Hilfe von Geräteobjekten in einer Baumstruktur (Projektnavigator) abbilden. Dadurch lässt sich die Verknüpfung von Hardware und Applikation leicht handhaben:

In den Konfigurationseditoren der Geräteobjekte können Sie Einstellungen für die Kommunikation zwischen IndraLogic und der Steuerung, und vor allem die E/A-Zuordnung vornehmen. Die E/A-Zuordnung, auch E/A-Abbild genannt, ist die Verknüpfung der Eingänge und Ausgänge der Steuerung mit den Variablen Ihrer Applikation.

Siehe auch

- [Kap. 8.1 "Projektexplorer und Geräteeditor" auf Seite 71](#)
- [Kap. 8.2 "Geräte im Projektexplorer" auf Seite 72](#)
- [Kap. 8.3 "Geräte konfigurieren und E/A Abbild konfigurieren" auf Seite 75](#)

8.1 Projektexplorer und Geräteeditor

Projektexplorer In der Ansicht **Geräte**, auch "**Projektexplorer**" genannt, stellen Sie Applikationen zielgerätbezogen zusammen. Sie können hier die Hardware eines oder mehrerer Steuerungsgeräte oder Feldbussysteme abbilden, die Kommunikation mit ihr konfigurieren und ihr jeweils mehrere Applikationen zuweisen.

Der Wurzelknoten des Projektexplorers ist ein symbolischer Knoteneintrag: **<projektname>**.

Darunter hängen Sie die Geräte-Objekte für ein oder mehrere Steuerungsgeräte, auch "Zielsysteme" genannt, ein. Jedes Geräte-Objekt repräsentiert eine bestimmte Hardware; Beispiele Steuerungsgerät, Feldbusknoten, Buskoppler, Antrieb, E/A-Modul, Monitor.

Beim Einfügen des Objekts unterstützt Sie ein Einfüge-Assistent, der immer die passenden Geräte aus Ihrem lokalen Geräte-Repository anbietet.

Wenn Sie bereits mit einem Steuerungsnetzwerk verbunden sind, können Sie die Hardware auf vorhandene Geräte durchsuchen und in die aktuelle Konfiguration in den Projektexplorer übernehmen.

Für das Anlegen der Geräteobjekte im Projektexplorer, also das Abbilden der zu steuernden Hardware-Umgebung, gelten bestimmte Regeln (siehe unten). Die hierarchische Anordnung von Applikationsobjekten und Geräteobjekten definiert die Gültigkeitsbereiche anderer Objekte wie beispielsweise Bibliotheken und GVLs.

Es gibt programmierbare Geräte und rein parametrierbare Geräte. Der Gerätetyp definiert die mögliche Einfügeposition im Projektexplorer und die Auswahl der Objekte, die unterhalb des Geräts eingefügt werden können. Programmierbare Geräte erhalten automatisch einen zusätzlichen, rein organisatorischen Knoten **SPS-Logik** unterhalb des Geräte-Objekts. Unter diesem Knoten können Sie die Objekte einfügen, die für das Programmieren des Geräts benötigt werden; also die Applikation(en) und beispielsweise GVLs oder Textlisten.

Jedes Gerät ist durch seine Gerätebeschreibung definiert und muss auf dem lokalen System installiert sein, um für das Einfügen im Projektexplorer verfügbar zu sein. Die Gerätebeschreibungsdatei definiert die Eigenschaften eines Geräts hinsichtlich Konfigurierbarkeit, Programmierbarkeit und möglichen Verbindungen zu anderen Geräten.



Der Ordner "Allgemeine Bausteinteile" enthält Objekte, die Sie projektweit verwenden können.

Programmierobjekte, die nur für eine bestimmte Applikation bestimmt sind, müssen Sie unterhalb des Applikationsobjekts in der Ansicht "Projektextplorer" einfügen.

Geräteeditor

Beachten Sie die Möglichkeit, die aktive Applikation auf einem "Simulationsgerät" laufen zu lassen, das standardmäßig innerhalb des Programmiersystems automatisch zur Verfügung steht. Im Simulationsbetrieb können Sie auch ohne Hardware das Online-Verhalten der Applikation testen. Sie aktivieren die Simulation über das Kontextmenü der Steuerung.

In den Registerkarten des Geräteeditors können Sie Einstellungen zur Kommunikation zwischen IndraLogic und dem Zielgerät vornehmen. Sie öffnen den Editor mit einem Doppelklick auf das Geräteobjekt im Projektextplorer. Der Editor enthält generische und gerätespezifische Registerkarten. Er ist betitelt mit dem Namen des Geräts.

Siehe auch

- [Kap. 18.1.9 "Objekt 'Gerät und Generischer Geräteeditor' auf Seite 519](#)
- [Kap. 8.3 "Geräte konfigurieren und E/A Abbild konfigurieren" auf Seite 75](#)

8.2 Geräte im Projektextplorer

Alle Objekte, die benötigt werden, um eine Applikation (also ein Steuerungsprogramm) auf einem Gerät (Steuerung, SPS) auszuführen, werden im Projektextplorer in einer Baumstruktur verwaltet ([Kap. 18.1.2 "Objekt 'Applikation'" auf Seite 508](#)).

Man kann diese Objekte auch als "Ressourcen"-Objekte bezeichnen. Geräteobjekte, Applikationsobjekte, Taskkonfiguration und Task sind "Ressourcen"-Objekte.

Programmierobjekte, wie einzelne POU's, Globale Variablenlisten und Bibliotheksverwalter können unterhalb einer Steuerung im Projektextplorer verwaltet werden und sind dann nur für Ihre Applikation verwendbar.



Programmierobjekte, die global gültig sein sollen, werden im Ordner "Allgemeiner Baustein" im Projektextplorer verwaltet.

Zur Konvertierung von Geräte-Referenzen beim Öffnen von Projekten, die in einem anderen Format erstellt wurden, sehen Sie

IndraMotion MLC - Projektkonvertierung.

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" ist im Projektextplorer unterhalb des Geräte-Knoten 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 unterhalb des Geräte-Knoten 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 jeweiligen Feldbusses, der die Dialoge zur Konfiguration eines Gerätes bereitstellt.

Wird eine Steuerung neu in den Projektextplorer eingefügt, wird voreingestellt gleich die automatisch mit eingefügte Applikation als "Mapping-Applikation" eingetragen.

Siehe auch

- [Kap. 9.10.2 "AT-Deklaration" auf Seite 130](#)

Details siehe auch

Generelle Informationen zum Projektextplorer

- Mapping-Dialog der einzelnen Feldbusse (Rexroth IndraWorks 12VRS/13VRS/14VRS , Feldbusse, DOK-IWORKS-FB*****V1x-AP02-DE-P)
- Der Wurzelknoten im Projektextplorer ist der Name des Projektes, der beim Anlegen eines neuen Projektes vergeben wird.  <project name>.
- Die Konfigurationsbäume der "Steuerungskonfiguration" und "Taskkonfiguration", die in IndraLogic 1.x in separaten Fenstern gehandhabt wurden, sind in IndraLogic 2G im Objektbaum des Gerätes integriert. Die Konfiguration der Geräte- und Task-Parameter wird in separaten Editor-Dialogen durchgeführt.

Siehe auch

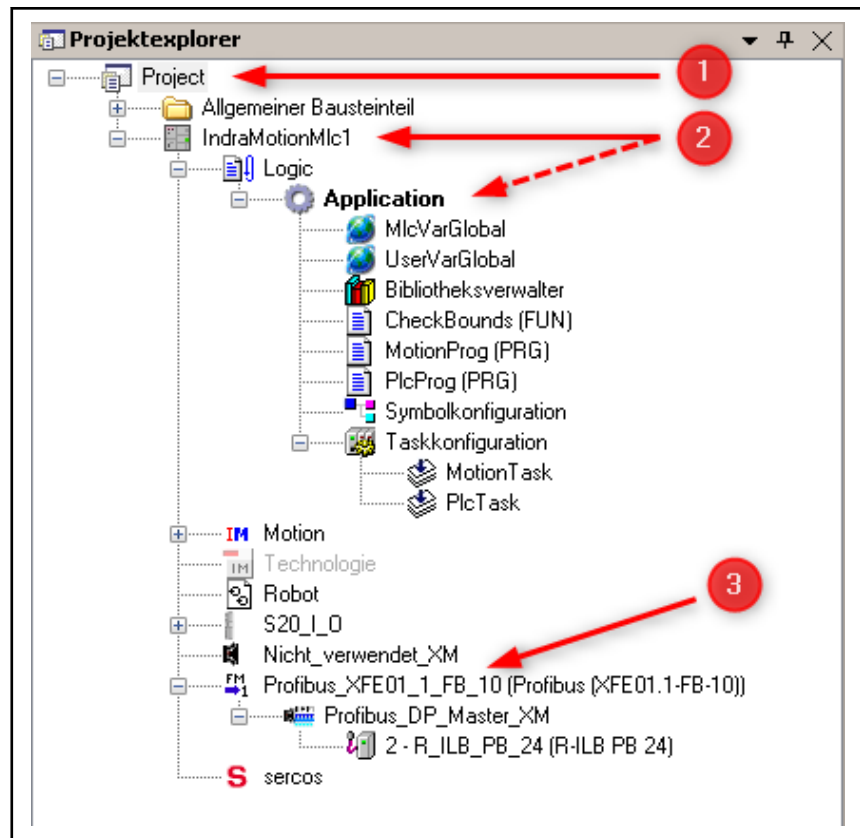
- [Kap. 9.15 "Taskkonfiguration" auf Seite 143](#)
- ["Steuerungskonfiguration" auf Seite 72](#)

So wird die Struktur der zu steuernden Hardware-Umgebung im Objektbaum des Gerätes durch die entsprechende Anordnung von Objekten abgebildet und es ist möglich, ein heterogenes System von mehrfach vernetzten Steuerungen und zugrundeliegenden Feldbussen aufzusetzen.

Sehen Sie unten die Regeln für das Anordnen von Objekten unterhalb des Geräteknotens.

- Ein **Geräte**-Objekt repräsentiert eine bestimmte Hardware (Zielsystem). Beispiele: Steuerungsgerät, Antrieb, E/A-Modul, Monitor.
- Ein Eintrag im Projektextplorer zeigt jeweils das Symbol, den symbolischen (im Explorer editierbaren) Gerätenamen und dahinter den Gerätetyp (=Gerätename, wie in der Gerätebeschreibung definiert).
- Es gibt **programmierbare** Geräte und **parametrierbare** Geräte. Der Gerätetyp bestimmt die mögliche Einfügeposition unterhalb des Geräteknotens und die Auswahl der Objekte, die unterhalb des Gerätes eingefügt werden können.

Programmierbare Geräte erhalten automatisch einen zusätzlichen  **Logic**-Knoten unterhalb des Geräte-Objekts. Unter diesem Knoten können dann die Objekte, die für das Programmieren des Gerätes benötigt werden, eingefügt werden (Visualisierungen, GVL, Textlisten usw.):



- (1) Projektname
 (2) Gerät, programmierbar, mit Applikation
 (3) Gerät, parametrierbar

Abb. 8-1: Geräte im Projektexplorer

- Innerhalb eines Projektes können ein oder mehrere programmierbare Geräte - unabhängig von Hersteller oder Typ - konfiguriert werden ("Anordnen und Konfigurieren von Objekten im Projektexplorer" auf Seite 74).
- Die Konfiguration eines Geräts bezüglich Kommunikation, Parameter oder I/O-Abbild wird im "Geräte-Editor-Dialog" durchgeführt. Der Dialog wird mit einem Doppelklick auf den Geräte-Eintrag geöffnet (Kap. "Generischer Geräteeditor" auf Seite 519).
- Im **Online-Modus** enthalten Symbole am Geräte-Eintrag Zusatzinformationen (siehe Tool-Tipps).

Zusätzliche Diagnose-Informationen zur SPS-Komponente des Gerätes gibt das Logbuch (Doppelklick auf das Geräteobjekt) (Kap. "Registerkarte 'Log'" auf Seite 525).

Anordnen und Konfigurieren von Objekten im Projektexplorer

- Welche Objekttypen eingefügt werden können, hängt von der gegenwärtig 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;

Weiterhin stehen immer nur solche Geräte zum Einfügen zur Auswahl, die korrekt auf dem lokalen System installiert wurden und zur gegenwärtig ausgewählten Position im Explorer passen.

Um ein Objekt einzufügen, markieren Sie die Position im Baum, wo das Objekt eingefügt werden soll und verwenden Sie im Kontextmenü **Hinzufügen** ► **<Objekttyp>**. Alternativ können Sie Objekte aus der Bibliothek durch Ziehen mit der Maus an die entsprechende Stelle einfügen.



Programmierbare Geräte lassen sich nur aus der Bibliothek "Drive and Control" einfügen. Markieren Sie das gewünschte Gerät (z.B. IndraMotion MLC L65) und ziehen es mit der Maus auf den Wurzelknoten.

- Die Anordnung von Objekten unterhalb einer Applikation sind nach Objekttyp und alphabetisch sortiert. Eine beliebige Veränderung Ihrer Position ist nicht möglich. Andererseits lassen sich die darunter befindlichen Objekte (z.B. Aktionen, Transitionen) durch Ziehen mit der Maus an die entsprechende Stelle beliebig positionieren.
- Ein bereits im Projektextplorer eingefügtes Gerät kann durch eine andere Version desselben Gerätes oder durch ein Gerät anderen Typs ersetzt werden. Der eingerückt unterhalb des Gerätes vorhandene Konfigurationsbaum kann dabei soweit als möglich erhalten bleiben. Markieren Sie dazu den Geräteknoten und wählen im Kontextmenü **Gerät aktualisieren**.
- Installation und Deinstallation von Geräten kann im Gerätedatenbank...-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 bus-spezifische Beschreibungsdateien, wie z.B. *.gsd-Dateien (Profibus) über den Geräte-Repository-Dialog installiert werden.
- Ein Gerät wird als Knoten im Projektextplorer eingefügt. Wenn solche in der Gerätebeschreibung definiert sind, werden automatisch die Unterknoten mit eingefügt. Ein Unterknoten kann wiederum auch ein programmierbares Gerät sein.
- Unterhalb eines "Geräte"-Objekts können weitere Geräte eingefügt werden, die auf dem lokalen System installiert sind und in der Bibliothek angeboten werden. Für jede Ebene werden immer zuerst die programmierbaren Geräte (SPS-Logik) angeordnet, dann die restlichen Typen.
- Eine Applikation wird automatisch unterhalb des Logic-Knotens (symbolischer Knoten für programmierbare Geräte) eingefügt. Für jedes Gerät ist nur eine Applikation zulässig. Unterhalb einer Applikation können dann die weiteren, für die Programmierung benötigten Objekte eingehängt werden, wie z.B. Datentypen (DUT), Globale Variablenliste (GVL), Visualisierung usw. Eine Taskkonfiguration wird automatisch unterhalb jeder Applikation eingefügt. Darin sind die entsprechenden Programmaufrufe zu definieren.

8.3 Geräte konfigurieren und E/A Abbild konfigurieren

Geräte konfigurieren

Die im Projektextplorer eingehängten Geräteobjekte können Sie im zugehörigen Geräteeditor konfigurieren. Die Möglichkeiten hängen von der Gerätebeschreibung ab. Der "generische Geräteeditor" stellt Registerkarten bereit, die gegebenenfalls durch gerätespezifische Registerkarten ergänzt sind.

Voraussetzung: Sie haben ein Standardprojekt geöffnet, in dessen Projektextplorer eine Standard-SPS und darunter ein Feldbus-Geräteobjekt eingefügt sind.

1. Führen Sie einen Doppelklick auf das Geräteobjekt der Standard-SPS im Projektextplorer Ihres Projekts aus.

Der Editor **<Gerätename>** erscheint im IndraLogic-Hauptfenster. Die Registerkarte **Applikationen** ist im Vordergrund.

Wechseln Sie auf die anderen Registerkarten, um Konfigurationseinstellungen für die Steuerung vorzunehmen.

2. Führen Sie einen Doppelklick auf das Feldbus-Geräteobjekt im Projektextplorer Ihres Projekts aus.

Der Editor **<Feldbus-Gerätename>** erscheint im IndraLogic-Hauptfenster. Je nach Gerät sind spezifische Registerkarten verfügbar. Sehen Sie zu den Konfigurationsmöglichkeiten bitte die Hilfeseiten zum jeweiligen Geräteeditor. Wenn die Option **Generische Konfigurationseditoren anzeigen** in **Extras ► Optionen**, Kategorie **Geräteeditor** aktiviert ist, sehen Sie auch die vom generischen Geräteeditor beige gesteuerten Registerkarten.

Siehe auch

- [Kap. "Generischer Geräteeditor" auf Seite 519](#)

Allgemeines zum E/A-Abbild ("Mapping")

Es hängt vom Gerätetyp ab, ob Sie ein E/A-Abbild auf Projektvariablen, oder auf ganze Funktionsbausteine konfigurieren können oder nicht.

Ein E/A-Abbild zu konfigurieren heißt Eingangs- und Ausgangskanäle des Geräts mit Variablen des Projekts zu verknüpfen. Wir verwenden dafür auch den Begriff **"Mapping"**.

Beachten Sie generell das Folgende für das Abbilden von Eingängen und Ausgängen eines Geräts auf Variablen in IndraLogic:

- Sie können auf Variablen, die auf einen Eingang abgebildet sind, nicht schreibend zugreifen.
- Sie können eine bereits bestehende Variable nur auf einen Eingang abbilden.
- Sie können im E/A-Abbild direkt neue globale implizite Variablen erzeugen und auf einen Gerätekanal abbilden.
- Das Speicher-Layout von Strukturen wird vom Gerät vorgegeben.
- Sie können im E/A-Abbild Adressen ändern und Werte fixieren.
- Für jede Variable, die einem E/A-Kanal im Dialog **E/A-Abbild** zugewiesen ist, können Sie während des Übersetzens der Applikation "Force-Variablen" erzeugen lassen (siehe weiter unten). Über diese Variablen können Sie beispielsweise bei der Inbetriebnahme einer Anlage über eine Visualisierung/HMI einen Wert auf dem Eingang oder Ausgang forcieren.
- Änderungen im E/A-Abbild können Sie mit einem Online-Change zur Steuerung übertragen.
- Wenn ein Pointer auf einen Geräteeingang verwendet wird, gilt der Zugriff als schreibender Zugriff, beispielsweise `pTest := ADR(input);`. Dies führt bei der Codeerzeugung zu einer Compilerwarnung "...kein gültiges Zuordnungsziel". Wenn Sie ein Konstrukt dieser Art benötigen, müssen Sie den Eingangswert `input` zuerst auf eine Variable mit Schreibzugriff kopieren.
- Eine E/A-Adresse kann auch über die "AT-Deklaration" im IEC-Code mit einer Variablen verknüpft werden.

Da sich eine Gerätekonfiguration oftmals noch ändert, empfehlen wir allerdings, die Zuweisungen nur im Geräteeditor vorzunehmen.

Falls Sie die AT-Deklaration verwenden, beachten Sie Folgendes:

- Eine AT-Deklaration ist nur bei lokalen oder globalen Variablen zulässig, nicht bei Eingangs- oder Ausgangsvariablen von Bausteinen.
- Implizite "Force-Variablen" für E/As (siehe unten) können für AT-Deklarationen nicht erzeugt werden.
- Wenn Sie eine AT-Deklaration bei Struktur- oder Funktionsbausteinvvariablen verwenden, werden alle Instanzen auf die gleiche Speicherstelle zugreifen. Dies entspricht dann der Verwendung von "statischen Variablen" in klassischen Programmiersprachen wie beispielsweise "C".



Wenn ein Pointer auf einen Geräteeingang verwendet wird, gilt der Zugriff (beispielsweise `pTest := ADR(input);`) als schreibender Zugriff. Dies führt bei der Codeerzeugung zu einer Compilerwarnung "...kein gültiges Zuordnungsziel".

Wenn Sie ein Konstrukt dieser Art benötigen, müssen Sie den Eingangswert (`input`) zuerst auf eine Variable mit Schreibzugriff kopieren.



Alternativ können Sie auch im Programmiercode über die AT-Deklaration eine Variable einer Adresse zuweisen.

In Anbetracht möglicher Änderungen der Gerätekonfiguration empfehlen wir jedoch, die Zuweisungen nur im Geräteeditor vorzunehmen.



Sie können die E/A-Abbildkonfiguration eines Geräts in eine "csv"-Datei exportieren von einer solchen importieren.

Siehe auch

- [Kap. "Registerkarte '<Gerätename> E/A-Abbild'" auf Seite 534](#)
- [Kap. "Befehl 'E/A-Abbild in CSV exportieren'" auf Seite 648](#)

Verknüpfen eines Geräteeingangs mit einer bestehenden Projektvariable ("Mapping")

Voraussetzung: Im Projektextplorer Ihres Projekts ist ein Gerät eingehängt, das eine E/A-Abbild-Konfiguration in IndraLogic unterstützt. Somit erhalten Sie im Geräteeditor auf Registerkarte **E/A-Abbild** eine tabellenförmige Darstellung der Eingangskanäle und Ausgangskanäle des Geräts mit Angabe der Adressen und Datentypen.



Abbilden "zu großer" Datentypen

Wenn Sie eine Variable eines Datentyps, der größer ist als ein Byte, auf eine Byte-Adresse abbilden, wird dort der Wert der Variablen auf Byte-Größe abgeschnitten!

Für das Monitoring des Variablenwerts im Dialog **E/A-Abbild** 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.



Wenn eine UNION durch E/A-Kanäle im Abbild-Dialog vertreten ist, hängt es vom Gerät ab, ob auch ein Abbilden auf das "Root"-Element möglich ist.

1. Deklarieren Sie in einem Programmbaustein beispielsweise eine Variable `xBool14` vom Typ `BOOL`, mit der Sie aus der Applikation auf einen Eingang des Zielgeräts zugreifen wollen.
2. Öffnen Sie den Geräteeditor mit einem Doppelklick auf das Geräteobjekt im Projektextplorer und wechseln Sie zur Registerkarte **E/A-Abbild**.
3. Betrachten Sie die Spalte **Variable** mit der Anzeige der Geräteeingangskanäle  und Geräteausgangskanäle , die je nach Gerät noch durch organisatorische Knoten  sortiert sein können. Wir nehmen an, es gibt einen Geräteeingang vom Typ `BYTE`. Er wird mit seinen einzelnen Bit-Adressen (Bit-Kanälen) unterhalb des `BYTE`-Knotens dargestellt.
- 4.



Beim Abbilden strukturierter Variablen verhindert der Editor, dass Sie sowohl die Strukturvariable (beispielsweise auf `%QB0`) als auch einzelne Strukturelemente (beispielsweise auf `%QB0.1` und `QB0.2`) eintragen.

Das bedeutet: Wenn es einen Haupteintrag mit einem Unterbaum von Bit-Kanal-Einträgen in der Abbild-Tabelle gibt, dann können Sie entweder in der Zeile des Haupteintrags eine Variable eintragen, oder in den Zeilen der Unterelemente (Bit-Kanäle), aber nicht in beiden.

Sie können nun entweder den gesamten Kanal mit einer vom Typ her passenden Variable belegen, **ODER** einzelne seiner Bit-Kanaladressen mit passenden Variablen vom Typ `BOOL` oder `BIT`. Doppelklicken Sie zunächst auf einen Bit-Eingangskanal in Spalte **Variablen**.

Ein Eingabefeld öffnet.

5. Um eine bestehende Variable auf den Kanal zu legen, müssen Sie die gewünschte Projektvariable mit komplettem Pfad eintragen. Öffnen Sie über  die Eingabehilfe. Wählen Sie beispielsweise die in `PlcProg` deklarierte Variable `Application.PlcProg.xBool14`.

Die Variable wird eingefügt. In der Spalte **Mapping** erscheint das Symbol .

Die Adresse ist nun durchgestrichen. Das bedeutet nicht, dass die Adresse nicht mehr verfügbar ist, denn Werte von bestehenden Variablen werden auf einem anderen Speicherplatz verwaltet.

Aber:

Sie sollten die Adresse dennoch - speziell bei Ausgängen - nicht mit einer weiteren Variablen belegen, um Uneindeutigkeiten beim Schreiben der Werte zu vermeiden.



Ab IndraWorks 14V22 wird für ein Mapping auf eine bestehende Variable automatisch der Initialisierungswert der Variablen als Standardwert verwendet. Das Feld "Standardwert" können Sie nur noch bearbeiten, wenn Sie ein Mapping auf eine neu erzeugte Variable vornehmen, oder wenn kein Mapping eingetragen ist. In älteren Versionen musste der Anwender explizit dafür sorgen, dass Standardwert und Initialisierungswert identisch sind.

Variable	Mapping	Channel	Address	Type	Default Value	Unit
		Output0	%QB1			
		Byte0	%QB1	BYTE		
		1.1	%QX1.0	BOOL		
		2.1	%QX1.1	BOOL		
Application.PlcProg.xBool4		1.4	%QX1.2	BOOL		

Abb. 8-2:

- Löschen Sie die Variablenzuweisung und die Deklaration von `xBool4` wieder.
- Deklarieren Sie in "PlcProg" beispielsweise eine Variable `byte_goto-device` vom Typ `BYTE`, mit der Sie aus der Applikation auf einen Eingang des Zielgeräts zugreifen wollen.
- Klicken Sie auf die Wurzel des Kanals, den `BYTE`-Knoten. Wählen Sie wieder über die Eingabehilfe die Variable `Application.PlcProg.byte_goto-device`.

Die Variable wird eingefügt, alle Bit-Adressen des Hauptkanals werden durchgestrichen, und Sie sollten sie nicht zusätzlich belegen.

Variable	Mapping	Channel	Address	Type	Default Value	Unit
		Output0	%QB1			
Application.PlcProg.byte_goto-device		Byte0	~~%QB1~~	BYTE		
		1.1	~~%QX1.0~~	BOOL		
		2.1	~~%QX1.1~~	BOOL		
		1.4	~~%QX1.2~~	BOOL		
		2.4	~~%QX1.3~~	BOOL		
		3.1	~~%QX1.4~~	BOOL		
		4.1	~~%QX1.5~~	BOOL		
		3.4	~~%QX1.6~~	BOOL		
		4.4	~~%QX1.7~~	BOOL		
		Byte1	%QB2	BYTE		

Abb. 8-3:

- Löschen Sie die Variablenzuweisung und die Deklaration von `byte_goto-device` wieder.

Abbilden eines Geräteeingangs auf eine neu erzeugte Projektvariable

Siehe auch

- [Kap. "Registerkarte '<Gerätename> E/A-Abbild'" auf Seite 534](#)

Im Folgenden bilden Sie einen Geräteausgang auf eine globale implizite Variable ab, die Sie zu diesem Zweck direkt im Dialog **E/A-Abbild** neu erzeugen.



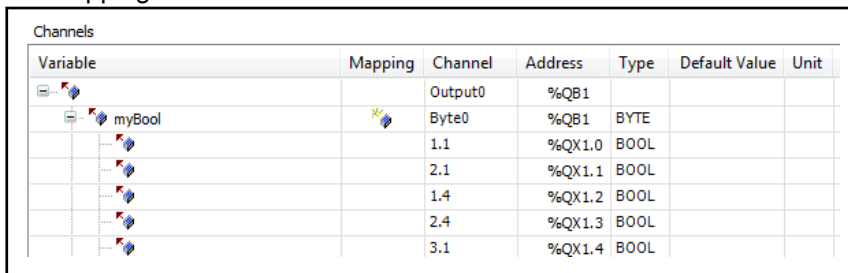
Somit ist der Dialog **E/A-Abbild** ein weiterer Ort für das Deklarieren einer globalen Variable.

Voraussetzung: Im Projektextplorer Ihres Projekts ist ein Gerät eingehängt, das eine E/A-Abbild-Konfiguration in IndraLogic unterstützt. Somit sehen Sie im Geräteeditor auf Registerkarte **E/A-Abbild** eine tabellenförmige Darstellung der Eingangs- und Ausgangskanäle des Geräts mit Angabe der Adressen und Datentypen.

1. Öffnen Sie den Geräteeditor mit einem Doppelklick auf das Geräteobjekt im Projektextplorer und wechseln Sie zur Registerkarte **E/A-Abbild**.
2. Klicken Sie in der Mapping-Tabelle auf einen Kanaleintrag in Spalte **Variable**, um ein Eingabefeld zu öffnen.
3. Geben Sie einen einfachen Namen (ohne ".") für eine neue Variable ein, Beispiel: `myBool`.

IndraLogic legt die Variable als implizite globale Variable im Projekt an und weist sie der Kanaladresse direkt zu.

Deshalb erscheint in diesem Fall die Adresse nicht durchgestrichen wie bei Mappings auf bestehende Variablen. 



Variable	Mapping	Channel	Address	Type	Default Value	Unit
myBool		Output0	%QB1	BYTE		
		Byte0	%QB1	BYTE		
		1.1	%QX1.0	BOOL		
		2.1	%QX1.1	BOOL		
		1.4	%QX1.2	BOOL		
		2.4	%QX1.3	BOOL		
		3.1	%QX1.4	BOOL		

Abb. 8-4:

Verknüpfen eines Gerätekanals mit einer Funktionsbaustein-Instanz

4. Löschen Sie die Variablenzuweisung von `myBool` wieder.

Wenn es vom Gerät unterstützt wird, können Sie ganze Funktionsbausteine auf einen Eingangs- oder Ausgangskanal abbilden. Damit können Sie zu Wartungszwecken beispielsweise die Häufigkeit der Signaländerungen zählen oder einen Kanalwert skalieren.

Im Folgenden Beispiel bilden Sie einen Geräteausgangskanal auf einen Funktionsbaustein ab, in dem der Baustein den Kanalausgangswert skaliert.

Voraussetzungen:

- Im Projekt ist ein Gerät mit digitalem Ausgang angebunden, das FB-Mapping unterstützt
- Es gibt einen Funktionsbaustein `Scale_Output_Int` mit der unten dargestellten Implementierung

Wichtig sind die Attribute am Baustein selbst und vor dem Ausgangsparameter, mit dem der Kanalausgang bearbeitet werden soll.

```
{attribute 'io_function_block'}
FUNCTION_BLOCK Scale_Output_Int
```



```

VAR_INPUT
  iInput : INT;
  iNumerator : INT;
  iDenominator : INT := 1;
  iOffset : INT := 0;
END_VAR
VAR_OUTPUT
  {attribute 'io_function_block_mapping'}
  iOutput : INT;
END_VAR
VAR
  iDenominator <> 0 THEN
    iOutput := TO_INT(TO_DINT(iInput) * TO_DINT(iNumerator) / TO_DINT(iDenominator)) + iOffset;

```

Vorgehensweise zum Verknüpfen eines Gerätekanals

1. Öffnen Sie den generischen Geräteeditor des betreffenden Gerätemoduls mittels Doppelklick im Projektexplorer
2. Öffnen Sie im Geräteeditor die Registerkarte E/A-Abbild
3. Selektieren Sie den Ausgang, der mit dem Funktionsbaustein verbunden werden soll per Doppelklick
4. Klicken Sie auf die Schaltfläche  "FB für E/A-Kanal hinzufügen"
 Der Dialog "Funktionsbaustein auswählen" öffnet sich.
 Im linken Teil sehen Sie unter dem Knoten "Applikation" den Baustein `Scale_Output_int`.
 Wenn Bibliotheken im Projekt eingebunden sind, die entsprechende Bausteine enthalten, erscheinen auch diese zur Auswahl.
5. Selektieren Sie den Baustein `myScaleOutputInt` und bestätigen Sie den Dialog mit "OK"
 Nach dem Bestätigen wird der Pfad des Funktionsbausteinparameters `iOutput` in der Spalte "Variable" im Mapping-Dialog eingetragen. Der Pfad setzt sich aus dem Applikationsnamen, dem Gerätekanalnamen und dem gewählten FB-Ausgang zusammen.
Beispiel: `App1.Out_4_Int_myScale_Output_Int_1.iOutput`
6. Selektieren Sie den Kanaleintrag und wählen Sie die Schaltfläche  "Gehe zu Instanz"
 Der Fokus wechselt zur Registerkarte "IEC-Objekte" zum dort angelegten Eintrag für das neue IEC-Objekt `Out_4_Int_myScale_Output_Int_1`.
 Im Onlinebetrieb sehen Sie in dieser Ansicht den aktuellen, durch den FB skalierten Wert des Parameters `iOutput` für den Kanal `Out_4_Int`.
 Sie können den Wert wie in anderen Monitoring-Ansichten auch schreiben und forcen.

Siehe auch

- [Kap. 18.3.1 "Dialog 'Funktionsbaustein auswählen'" auf Seite 744](#)
- [Kap. "Registerkarte '<Gerätename> E/A-Abbild'" auf Seite 534](#)
- [Kap. "Registerkarte '<Gerätename> IEC-Objekte'" auf Seite 537](#)
- [Kap. "Attribut 'io_function_block', 'io_function_block_mapping'" auf Seite 418](#)

Ändern und Fixieren eines Adresswerts im E/A-Abbild

Voraussetzung: Sie haben ein Projekt mit E/A-Abbild, sehen Sie hierzu die oben stehenden Abschnitte dieses Kapitels.

E/A-Anbindung konfigurieren

Der Adresswert eines gesamten Kanals (nicht den eines einzelnen Unterelements des Kanals!) kann in der Mapping-Tabelle geändert und fixiert werden. Öffnen Sie hierzu den Generischen Geräteeditor des E/A-Objekts mittels Doppelklick im Projektextplorer und wechseln Sie zur Registerkarte **E/A-Abbild**.

Durch die Fixierung kann die Adressierung an eine vorgegebene Maschinenkonfiguration angepasst werden, wodurch der Adresswert auch dann beibehalten wird, wenn sich die Anordnung der Module ändert.

Ohne Fixierung führt eine Änderung der Modul-Anordnung zu einer automatischen Anpassung der Adresswerte.

1. Öffnen Sie den Generischen Geräteeditor des E/A-Objekts mittels Doppelklick im Projektextplorer und wechseln Sie zur Registerkarte **E/A-Abbild**.
2. Deklarieren Sie in "PlcProg" beispielsweise eine Variable `xBool4` vom Typ `BOOL`, mit der Sie aus der Applikation auf einen Eingang des Zielgeräts zugreifen wollen. Weisen Sie die Variable wieder wie nachstehend zu.

Variable	Mapping	Channel	Address	Type	Default Value	Unit
		Output0	%QB1			
		Byte0	%QB1	BYTE		
		1.1	%QX1.0	BOOL		
		2.1	%QX1.1	BOOL		
		1.4	%QX1.2	BOOL		

Abb. 8-5:

3. Klicken Sie in der Mapping-Tabelle auf einen Kanaleintrag in Spalte **Adresse**, um ein Eingabefeld zu öffnen. Dies ist nur für die "Root"-Adresse eines Kanals möglich, nicht für eine einzelne seiner Unterelemente!

Ändern Sie deshalb den obersten Adresseintrag eines Kanals in der Tabelle, beispielsweise von `QB1` auf `QB5`. Verlassen Sie das Eingabefeld.

Der Adresswert ist verändert. Vor der Adresse erscheint das Symbol . Es zeigt an, dass die Adresse fixiert ist. Auch die Adressen der Unterelemente des Kanals werden entsprechend geändert. Wenn Sie nun im Projektextplorer die Position des Geräteobjekts innerhalb anderer Geräteobjekte mit Eingangs-/Ausgangskanälen ändern, passt IndraLogic diese Adressen nicht der neuen Reihenfolge an, wie es ohne Fixierung der Fall wäre.

Variable	Mapping	Channel	Address	Type	Default Value	Unit
		Output0	 %QB5			
		Byte0	%QB5	BYTE		
		1.1	%QX5.0	BOOL		
		2.1	%QX5.1	BOOL		
		1.4	%QX5.2	BOOL		

Abb. 8-6:

4. Um die manuelle Änderung bzw. Fixierung rückgängig zu machen, öffnen Sie erneut das Eingabefeld des Adresswerts, löschen den Adresseintrag und drücken die Eingabetaste.

IndraLogic setzt die Adresse und die betroffenen nachfolgenden Adressen auf die Werte zurück, die sie vor der Änderung hatten und entfernt das Symbol .

Siehe auch

- [Kap. "Registerkarte '<Gerätename> E/A-Abbild'" auf Seite 534](#)
- [Kap. 17.4.13 "Adressen" auf Seite 371](#)

Konfigurieren der E/A-Variable-naktualisierung

Abhängig vom Gerät, das Sie im Projekt anbinden, aktualisiert in IndraLogic die an dessen Eingängen und Ausgängen anliegenden Variablen in unterschiedlicher Art. Sie können die Einstellungen dazu im Dialog **E/A-Abbild** explizit umstellen.

Siehe auch

- [Kap. "Registerkarte '<Gerätename> E/A-Abbild'" auf Seite 534](#)

Monitoring von Variablen im E/A-Abbild im Online-Betrieb

Voraussetzung: Sie haben eine Applikation mit einer Gerätekonfiguration, die E/A-Abbilder enthält, fehlerfrei übersetzt. Die zugehörige Hardware und das Bussystem laufen. Sie haben sich mit dem Befehl **Debuggen ► Einloggen** auf die Steuerung verbunden, die Applikation geladen und gestartet.

1. Öffnen Sie die Registerkarte **E/A-Abbild** des Steuerungsgeräts im Geräteeditor. Zum Öffnen des Editors doppelklicken Sie auf das Geräteobjekt im Projektextplorer.

Die Mapping-Tabelle enthält nun zusätzlich die Spalten **Aktueller Wert** und **Neuer Wert**.

Wenn eine Strukturvariable auf das "Root"-Element einer Adresse¹ abgebildet ist, zeigt IndraLogic im Online-Betrieb in dieser Zeile keinen Wert an. Wenn jedoch beispielsweise eine `DWORD`-Variable auf die Adresse abgebildet ist, werden sowohl in der "Root"-Zeile als auch in den darunter eingerückten Bit-Kanal-Zeilen die jeweiligen Werte gemonitort. Grundsätzlich bleibt das Feld in der "Root"-Zeile immer dann leer, wenn sich der Wert aus mehreren Unterelementen zusammensetzen würde.

¹ "Root" = oberstes Element dieser Adresse im Mapping-Dialog

2. Tragen Sie für einen Eintrag in der Spalte **Neuer Wert** einen bestimmten Variablenwert ein und drücken <F7> für ein Forcen oder <Strg>+<F7> für ein Schreiben des Werts.

In der Spalte **Aktueller Wert** wird - wie beim Monitoring im Deklarationseditor oder in Überwachen-Listen - der geforcte Variablenwert mit vorangestelltem roten F-Symbol oder der geschriebene Wert angezeigt.



Im Online-Betrieb werden Ein- und Ausgänge, welche der SPS-Code **NICHT** verwendet, von der SPS nicht gelesen, wodurch der angezeigte Wert falsch sein könnte.

Der **Aktuelle Wert** der betroffenen Variablen erscheint mit einem grauen Hintergrund.

E/A-Anbindung konfigurieren

Variable	Mapping	Channel	Address	Type	Default Value	Current Value	Prepared Value
Application.PlcProg.invar1		OnboardIn	%I0.0	BYTE		F 3	
		1.1	%IX0.0	BOOL		F TRUE	
		2.1	%IX0.1	BOOL		F TRUE	
		1.4	%IX0.2	BOOL		F FALSE	
		2.4	%IX0.3	BOOL		F FALSE	
		1.1	%IX0.4	BOOL		F FALSE	
		2.1	%IX0.5	BOOL		F FALSE	
		1.4	%IX0.6	BOOL		F FALSE	
		2.4	%IX0.7	BOOL		F FALSE	
		OnboardOut	%Q0.0	BYTE		0	
		1.1	%QX0.0	BOOL		FALSE	
		2.1	%QX0.1	BOOL		FALSE	
		1.4	%QX0.2	BOOL		FALSE	

- (1) geforderte Werte auf der Steuerung
 (2) auf der Steuerung nicht verwendete Werte, Wert grau dargestellt

Abb. 8-7:

E/A-Abbild in einem Dialog für mehrere Geräte

Es gibt eine Tabelle, die das E/A-Abbild eines Geräts plus die E/A-Abbilder aller unter ihm im Projektextplorer eingehängten Subelemente darstellt. Sie können dort die E/A-Abbilder genauso bearbeiten wie in den einzelnen Mapping-Tabellen der jeweiligen Geräteeditoren.

Voraussetzung: Im Projektextplorer Ihres Projekts sind mehrere Steuerungsgeräte eingehängt, die jeweils eine E/A-Abbild-Konfiguration ermöglichen.

1. Selektieren Sie die gewünschte Steuerung im Projektextplorer und wählen Sie im Kontextmenü den Befehl **E/A-Abbild bearbeiten**.

Der Dialog **E/A-Abbild bearbeiten** erscheint, in dem die E/A-Abbild-Konfigurationen aller im Projekt eingehängten Geräte in einer Tabelle erscheinen. Sie können die Einträge bearbeiten wie im Dialog **E/A-Abbild** des zugehörigen Geräteeditors.

2. Selektieren Sie nun im Projektextplorer eines der Steuerungsobjekte und wählen Sie erneut im Kontextmenü den Befehl **E/A-Abbild bearbeiten**.

Der Dialog **E/A-Abbild bearbeiten** zeigt nun nur noch die E/A-Tabelle für die im und unter dem selektierten Objekt gefundenen E/A-Abbild-Konfigurationen.

3. Setzen Sie in der Leiste oberhalb der Tabelle einen gewünschten **Filter** oder geben Sie im Feld **Suche Variable** einen Variablennamen ein, um die Verwendung dieser Variable im Mapping zu sehen.

Siehe auch

- [Kap. "Befehl 'E/A-Abbild bearbeiten'" auf Seite 647](#)

9 Applikation programmieren

Um ein auf der Steuerung lauffähiges Applikationsprogramm, zu erstellen, füllen Sie POU's mit Deklarationen und Implementierungscode (Quellcode), stellen die Verknüpfung der E/As der Steuerung mit Applikationsvariablen her und konfigurieren eine Taskzuordnung.

Nach Prüfung und Fehlerbeseitigung erstellt der IndraLogic-Compiler dann den Applikationscode, der auf die Steuerung geladen werden kann.

Die Programmierung der Applikationsbausteine (POUs) wird unterstützt durch die Programmierspracheneditoren und einigen weiteren Funktionalitäten wie beispielsweise Textlisten, Bildersammlungen, Pragmas und Verwendung von fertigen Bausteinen aus dem SPS-Programmiersystem oder aus Bibliotheken.

Es gibt Mittel zur Syntaxprüfung und Codeanalyse, zum Herstellen von Datenpersistenz und zur Verschlüsselung des Applikationscodes, der auf die Steuerung geladen wird.

9.1 Bezeichner vergeben

Bezeichner sind die Namen von Variablen und Programmierobjekten, zum Beispiel Programme, Funktionsbausteine, Methode etc. und Namen anderer Objekte der Applikation und des Projekts.

Es gibt Regeln, die Sie bei der Vergabe von Bezeichner einhalten müssen. Darüber hinaus gibt es Empfehlungen, die dazu dienen, die Bezeichner einheitlich und aussagekräftig zu gestalten.

Die Bezeichner von Variablen vergeben Sie bei der Variablendeklaration. Sie können diese Bezeichner im Deklarationsteil des Programmierobjekts ändern.

Die Bezeichner für Programmierobjekte und andere Objekte vergeben Sie im Dialog beim Hinzufügen des jeweiligen Objekts. Sie können den Bezeichner eines bestehenden Objekts der Applikation oder des Projekts im Eigenschaftendialog des Objekts ändern.

Allerdings können Sie die Bezeichner von Objekten, die nur einmal pro Applikation oder nur einmal pro Projekt vorhanden sein können, nicht ändern, zum Beispiel die Bezeichner **Bibliotheksverwalter** und **ImagePool**.

Siehe auch

- [Kap. 17.7 "Bezeichner" auf Seite 445](#)

9.2 Variablen deklarieren

Variablendeklaration wo und wie

In IndraLogic-Projekten können Sie Variablen an folgenden Stellen deklarieren:

- Deklarationsteil einer POU
- Dialog **Variable deklarieren**, wenn der Fokus in einer POU liegt
- DUT-, GVL- oder NVL-Editor
- E/A-Abbild-Konfiguration eines E/A-Geräteobjekts

Die Variablendeklarationen in globalen Variablenlisten (GVL, NVL) und im Deklarationsteil von POU's führen Sie im Deklareditor durch.

Für alle anderen Varianten stehen spezielle Editoren oder Dialoge zur Verfügung.



Durch das Hinzufügen von Pragmas können Sie das Verhalten und die Eigenschaften einer oder mehrerer Variablen beeinflussen ([Kap. 17.6 "Pragmas" auf Seite 402](#)).

Syntax für die Variablendeklaration in POU's oder globalen Variablenlisten, wenn für den Deklarationseditor die textuelle Anzeige ausgewählt ist:

```
<Bezeichner> {AT <Adresse>}:<Datentyp> {:=<Initialisierung>;}
```

- **<Bezeichner>**: Name der Variablen. Die aufgeführten Regeln müssen Sie bei der Vergabe eines Bezeichners zwingend beachten ([Kap. 17.7 "Bezeichner" auf Seite 445](#)). Zusätzlich finden Sie Empfehlungen, die der Vereinheitlichung bei der Namensvergabe dienen.
- **{AT <Adresse>}** (optional): Mithilfe des Schlüsselworts **AT** können Sie eine Variable direkt an eine bestimmte Adresse binden.
- **<Datentyp>**
- **{:=<Initialisierung>}** (optional)



In Funktionsbausteinen können Sie auch Variablen mit unvollständigen Adressangaben deklarieren.

Damit Sie eine solche Variable in einer lokalen Instanz verwenden können, muss allerdings ein entsprechender Eintrag für diese Variable in der "Variablenkonfiguration" vorliegen.

Wenn Sie eine Variable im tabellarischen Deklarationseditor deklarieren, wird automatisch die korrekte Syntax hergestellt.

Siehe auch

- [Kap. 9.2.1 "Deklarationseditor verwenden" auf Seite 87](#)
- [Kap. 17.5 "Datentypen" auf Seite 374](#)
- [Kap. 17.2 "Variablen" auf Seite 300](#)
- [Kap. "Befehl 'Variable deklarieren'" auf Seite 625](#)
- [Kap. 17.7 "Bezeichner" auf Seite 445](#)
- [Kap. 18.1.11 "Objekt 'GVL'- Globale Variablenliste" auf Seite 544](#)
- [Kap. 9.6 "Pragmas verwenden" auf Seite 114](#)
- [Kap. 9.2.2 "Dialog 'Variable deklarieren' verwenden" auf Seite 88](#)

Variableninitialisierung

Der Standard-Initialwert für alle Deklarationen ist 0.

Im Deklarationsteil können Sie für jede Variable und jeden Datentyp auch benutzerdefinierte Initialisierungswerte angeben.

Die benutzerdefinierte Initialisierung beginnt mit dem Zuweisungsoperator **:=** und besteht aus einem beliebigen, gültigen Ausdruck der Programmiersprache ST (strukturierter Text).

Somit definieren Sie den Initialwert mit Hilfe von Konstanten, anderen Variablen oder Funktionen.

Wenn Sie eine Variable verwenden, müssen Sie diese ebenfalls (vorher!) initialisieren.

Beispiele

```
VAR
var1:INT := 12;           // initialization value 12
```

```

x: INT := 13 + 8;           // initialization value defined by an expression of constants*)

y: INT := x + fun(4);       (* initialization value defined by an expression,
                             that contains a function call;
                             notice the order! *)

z: POINTER TO INT := ADR(y); (* not described in the standard IEC61131-3:
                             initialization value defined by an adress function;
                             Notice: the pointer will not be initialized
                             during an Online Change *)

END_VAR

```

Hinweise zur Initialisierungsreihenfolge



Ab Compiler-Version 3.5.3.40 werden Variablen in einem Funktionsbaustein in folgender Reihenfolge initialisiert:

Zuerst alle Konstanten gemäß der Reihenfolge ihrer Deklarationen, danach alle anderen Variablen gemäß der Reihenfolge ihrer Deklarationen.



Ab Compiler-Version 3.3.2.0 werden Variablen aus globalen Variablenlisten immer vor den lokalen Variablen einer POU initialisiert.

Siehe auch

- [Kap. 17.5.14 "Datentyp 'ARRAY'" auf Seite 386](#)
- [Kap. 17.5.16 "Datentyp 'STRUCT'" auf Seite 396](#)
- [Kap. 17.5 "Datentypen" auf Seite 374](#)

9.2.1 Deklarationseditor verwenden

Der Deklarationseditor dient der Deklaration von Variablen in Variablenlisten und POU's.

Der Deklarationseditor bietet zwei mögliche Ansichten:  textuell,  tabellarisch.

Im Dialog **Extras ▶ Optionen ▶ IndraLogic 2G ▶ Deklarationseditor** definieren Sie, ob entweder nur die textuelle oder nur die tabellarische Ansicht verfügbar ist, oder ob Sie über die Schaltflächen am rechten Rand des Editorfensters zwischen den beiden Ansichten wechseln können.

Wenn der Deklarationseditor in Verbindung mit einem Programmiersprachen-Editor verwendet wird, erscheint er als Deklarationsteil im oberen Teil des Fensters einer POU.

Siehe auch

- [Kap. "Befehl 'Variable deklarieren'" auf Seite 625](#)
- [Kap. 9.2.2 "Dialog 'Variable deklarierenverwenden'" auf Seite 88](#)
- [Kap. "Befehl 'Deklarationskopf editieren'" auf Seite 720](#)
- ["Dialog 'Eingabehilfe- Registerkarte 'Kategorien'" auf Seite 625](#)

Deklarieren im textuellen Deklarationseditor

Das Verhalten und das Aussehen des textuellen Editors konfigurieren Sie mit den Einstellungen im Dialog **Tools ▶ Optionen ▶ Texteditor**. Die Einstellungen betreffen Farben, Zeilennummern, Tab-Breiten, Einrückungen etc.. Es stehen die üblichen Windows-Funktionen zur Verfügung, gegebenenfalls auch die IntelliMouse-Funktionen.

Voraussetzung: Sie haben ein Programmierobjekt (POU, GVL oder NVL) eines Projekts geöffnet. Der Fokus ist im textuellen Deklarationseditor.

Deklarieren im tabellarischen Deklarationseditor

1. Geben Sie die Variablendeklarationen in korrekter Syntax ein. Über <F2> erhalten Sie den Dialog **Eingabehilfe** zur Auswahl des Datentyps oder eines Schlüsselworts.

Im tabellarischen Deklarationseditor fügen Sie Variablendeklarationen in einer Tabelle mit den folgenden Spalten ein: **Gültigkeitsbereich**, **Name**, **Adresse**, **Datentyp**, **Initialisierung**, **Kommentar** und **Attribute** (Pragmas).

Voraussetzung: Sie haben ein Programmierobjekt (POU oder GVL) eines Projekts geöffnet. Der Fokus liegt im tabellarischen Deklarationseditor.

1. Klicken Sie auf Schaltfläche  im Deklarationskopf oder wählen Sie den Befehl **Einfügen** im Kontextmenü.
IndraLogic fügt eine neue Zeile für eine Variablendeklaration ein und das Eingabefeld für den Variablennamen öffnet sich.
2. Geben Sie einen gültigen Variablenbezeichner ein.
3. Öffnen Sie nach Bedarf die anderen Felder der Deklarationszeile mit einem Doppelklick und wählen Sie die gewünschten Angaben aus den Auswahllisten oder mit Hilfe der erscheinenden Dialoge.

Siehe auch

- [Kap. 18.3.3 "Dialog 'Wert vorbereiten'" auf Seite 745](#)
- ["Deklarationseditor im Onlinebetrieb" auf Seite 239](#)
- ["Forcen im Deklarationsteil" auf Seite 198](#)
- [Kap. "Befehl 'Variable deklarieren'" auf Seite 625](#)
- [Kap. "Befehl 'Deklarationskopf editieren'" auf Seite 720](#)
- ["Dialog 'Eingabehilfe- Registerkarte 'Kategorien'" auf Seite 625](#)

9.2.2 Dialog 'Variable deklarieren' verwenden

Voraussetzung: Ein Programmierobjekt (POU oder GVL) eines Projekts ist geöffnet.

1. Wählen Sie den Befehl **Bearbeiten ► Variable deklarieren**.
Der Dialog **Variable deklarieren** öffnet sich.
2. Wählen Sie aus der Auswahlliste **Gültigkeitsbereich** den gewünschten Gültigkeitsbereich für die Variable aus.
3. Geben Sie einen Variablennamen in das Eingabefeld **Name** ein.
4. Wählen Sie aus der Auswahlliste **Datentyp** den gewünschten Datentyp aus.
5. Wenn der Initialisierungswert vom Standard-Initialisierungswert abweichen soll, geben Sie einen Initialisierungswert für die Variable ein.
6. Schließen Sie Ihre Eingaben mit einem Klick auf **OK** ab.
IndraLogic listet die neu deklarierte Variable im Deklarationsteil Ihres Programmierobjekts.



Mit Hilfe von Pragmas im Deklarationsteil können Sie die Verarbeitung der Deklaration durch den Compiler beeinflussen.

Siehe auch

- [Kap. "Befehl 'Variable deklarieren'" auf Seite 625](#)
- [Kap. 17.5 "Datentypen" auf Seite 374](#)

- [Kap. 9.6 "Pragmas verwenden" auf Seite 114](#)

9.2.3 Array deklarieren

Voraussetzung: Ein Programmierobjekt (POU oder GVL) eines Projekts ist geöffnet.

1. Wählen Sie den Befehl **Bearbeiten ► Variable deklarieren**

Der Dialog **Variable deklarieren** öffnet sich.

2. Wählen Sie aus der Auswahlliste **Gültigkeitsbereich** den gewünschten Gültigkeitsbereich für das Array aus.

3. Geben Sie einen Bezeichner für das Array in das Eingabefeld **Name** ein.

4. Klicken Sie auf die Schaltfläche  neben dem Eingabefeld **Datentyp** und wählen Sie im Auswahlménü den Eintrag **Array-Assistent** aus.

5. Geben Sie in die Eingabefelder **Dimension 1**: die untere und die obere Indexgrenze der 1. Dimension des Arrays ein, zum Beispiel: 1 und 3.

Das Feld **Ergebnis** zeigt die 1. Dimension des Arrays an, zum Beispiel: `ARRAY [1..3] OF ?`.

6. Geben Sie im Eingabefeld **Basistyp**: den Datentyp des Arrays direkt

oder mit Hilfe der  **Eingabehilfe** oder des **Array-Assistent** ein, zum Beispiel: `DINT`.

Das Feld **Ergebnis** zeigt jetzt auch den Datentyp des Arrays an, zum Beispiel: `ARRAY [1..3] OF DINT`.

7. Definieren Sie entsprechend der Schritte 5 und 6 die Dimensionen 2 und 3 des Arrays, zum Beispiel: Dimension 2: 1 und 4, Dimension 3: 1 und 2.

Das Feld **Ergebnis** zeigt das Array mit den definierten Dimensionen an: `Array [1..3, 1..4, 1..2] OF DINT`. Das Array besteht aus $3 * 4 * 2 = 24$ Elementen.



In einem Array variabler Länge deklarieren Sie die Dimensionsgrenzen mit dem Sternchen-Platzhalter *. Arrays variabler Länge sind nur in VAR_IN_OUT-Deklarationen von Funktionsbausteinen, Methoden oder Funktionen erlaubt.

Beispiel für ein zweidimensionales Array variabler Länge: `aiUnknownLengthData : ARRAY [*,*] OF INT`

8. Klicken Sie auf **OK**.

Im Dialog **Variable deklarieren** zeigt das Feld **Datentyp** das Array an.

9. Wenn Sie die Initialisierungswerte des Arrays ändern wollen, klicken Sie

auf die Schaltfläche  neben dem Eingabefeld **Initialisierungswert**.

Der Dialog **Initialisierungswert** öffnet sich.

10. Selektieren Sie die Zeile des Array-Elements, dessen Initialisierungswert Sie ändern wollen.
Beispiel:
Array-Element [1, 1, 1] auswählen.
11. Geben Sie im Eingabefeld unterhalb der Auflistung den gewünschten Initialisierungswert ein und klicken Sie auf die Schaltfläche **Wert auf ausgewählte Zeilen anwenden**, zum Beispiel: Wert 4.
IndraLogic zeigt den geänderten Initialisierungswert der selektierten Zeile an.
12. Klicken Sie auf **OK**.
Im Feld **Initialisierungswert** des Dialogs **Variable deklarieren** zeigt IndraLogic die Initialisierungswerte des Arrays an, zum Beispiel:
[4, 23 (0)].
13. Optional geben Sie einen **Kommentar** in das Eingabefeld ein.
14. Klicken Sie auf **OK**, um die Deklaration des Arrays abzuschließen.
IndraLogic fügt die Deklaration des Arrays im Deklarationsteil des Programmierobjekts hinzu.

Siehe auch

- [Kap. "Befehl 'Variable deklarieren'" auf Seite 625](#)
- [Kap. 17.5.14 "Datentyp 'ARRAY'" auf Seite 386](#)

9.2.4 Globale Variablen deklarieren

Globale Variablen definieren, die innerhalb der Applikation verfügbar sind.

Voraussetzung: Ein Projekt ist geöffnet.

1. Selektieren Sie im Projektextplorer Ihres Projekts die Applikation, in der die globalen Variablen gültig sein sollen.
2. Wählen Sie den Kontextmenübefehl **Hinzufügen ► Globale Variablenliste**.
IndraLogic fügt die **GVL** im Projektextplorer unter der Applikation ein und öffnet sie im Editor.
3. Wählen Sie den Menübefehl **Bearbeiten ► Variable deklarieren**.
Der Dialog **Variable deklarieren** öffnet sich.
4. Wählen Sie in der Auswahlliste **Gültigkeitsbereich** den Eintrag **VAR_GLOBAL**.
5. Geben Sie im Feld **Name** einen Namen für die globale Variable ein.
6. Wählen Sie in der Auswahlliste **Datentyp** einen Datentyp aus.
7. Wenn Ihre Variable einen anderen Initialisierungswert als den Standard-Initialisierungswert haben soll, klicken Sie auf  neben dem Feld **Initialisierungswert**.
Der Dialog **Initialisierungswert** öffnet sich.
8. Doppelklicken Sie auf die Zelle **Initialisierungswert** Ihrer Variable und geben Sie den gewünschten gültigen Wert ein.
9. Klicken Sie auf **OK**.

Der Initialisierungswert wird im Dialog **Variable deklarieren** angezeigt.

10. Aktivieren Sie bei Bedarf eines der **Flags**.
11. Bestätigen Sie Ihre Eingaben mit einem Klick auf die Schaltfläche **OK**.
IndraLogic fügt die deklarierte Variable in der GVL ein.
Die globale Variable ist in der gesamten Applikation Ihres Projekts verfügbar.

Globale Variablen definieren, die im ganzen Projekt verfügbar sind.

1. Gehen Sie zum Ordner "Allgemeiner Baustein".
2. Selektieren Sie den Ordner und wählen Sie den Kontextmenübefehl **Hinzufügen ► Globale Variablenliste**.
IndraLogic fügt die **GVL** in den Ordner ein.
3. Öffnen Sie den Editor durch einen Doppelklick auf das Objekt.
4. Wählen Sie den Menübefehl **Bearbeiten ► Variable deklarieren**.
Der Dialog **Variable deklarieren** öffnet sich.
5. Wählen Sie in der Auswahlliste **Gültigkeitsbereich** den Eintrag **VAR_GLOBAL**.
6. Geben Sie im Feld **Name** einen Namen für die globale Variable ein.
7. Wählen Sie in der Auswahlliste **Datentyp** einen Datentyp aus.
8. Wenn Ihre Variable einen anderen Initialisierungswert als der Standard-Initialisierungswert haben soll, geben Sie diesen in der Spalte **Initialisierung** ein.
9. Aktivieren Sie bei Bedarf eines der **Flags**.
10. Bestätigen Sie Ihre Eingaben mit einem Klick auf die Schaltfläche **OK**.
IndraLogic fügt die deklarierte Variable in der GVL ein.
Die globale Variable ist nun im gesamten Projekt verfügbar.

Siehe auch

- [Kap. "Befehl 'Variable deklarieren'" auf Seite 625](#)

9.3 Quellcode in IEC erstellen

Quellcode:

Mit "Quellcode" bezeichnen wir den Implementierungscode, den Sie mit Hilfe der entsprechenden Programmiersprachen-Editoren in den Programmierbausteinen einfügen.

Folgende POU-Bausteintypen stehen dazu zur Verfügung:

- Programm
- Funktionsbaustein
- Funktion
- Methode
- Eigenschaft
- Aktion
- Schnittstelle

Programmiersprache:

Beim Anlegen eines Programmierbausteins entscheiden Sie, in welcher Implementierungssprache Sie programmieren wollen. Neben den IEC-Sprachen steht auch CFC zur Verfügung.

Programmiersprachen-Editoren:

Ein Programmierbaustein wird zur Bearbeitung im entsprechenden Programmiersprachen-Editor geöffnet, wenn Sie einen Doppelklick auf das Objekt im Projektextplorer ausführen.

Der Baustein erscheint also entweder im textuellen ST-Editor oder in einem der grafischen Editoren für FUP/KOP/AWL, AS oder CFC.

Jeder Editor besteht aus zwei Fenstern:

Im oberen Fenster nehmen Sie die Deklarationen vor, je nach Einstellung in textueller oder tabellarischer Form.

Im unteren Fenster fügen Sie den Implementierungscode ein. Die Darstellung und das Verhalten jedes Editors können Sie projektweit in der zugehörigen Registerkarte der IndraLogic-Optionen konfigurieren.

Beachten Sie die Möglichkeit, einen Programmierbaustein auch während des Onlinebetriebs der Applikation im Offlinezustand des Editors zu öffnen.

Siehe auch

- [Kap. 17.1 "Programmiersprachen und ihre Editoren" auf Seite 239](#)
- [Kap. "Befehl 'Objekt \(offline\) bearbeiten'" auf Seite 646](#)

9.3.1 FUP/KOP/AWL

Ein kombinierter Editor ermöglicht das Programmieren in den Sprachen FUP (Funktionsplan), KOP (Kontaktplan) und AWL (Anweisungsliste).

Die Basiseinheit der FUP- und KOP-Programmierung ist ein Netzwerk.

Jedes Netzwerk enthält eine Struktur, die Folgendes darstellen kann: Einen logischen oder arithmetischen Ausdruck, den Aufruf eines Programmierbausteins (Funktion, Funktionsbaustein, Programm usw.), einen Sprung, oder eine Return-Anweisung.

AWL benötigt eigentlich keine Netzwerke. In IndraLogic besteht jedoch auch ein AWL-Programm aus mindestens einem Netzwerk, um das Konvertieren in FUP oder KOP zu unterstützen.

Im Hinblick darauf sollten Sie auch ein AWL-Programm sinnvoll in Netzwerke unterteilen.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Funktionsplan (FUP)

Der Funktionsplan ist eine graphisch orientierte IEC 61131-Programmiersprache. Er arbeitet mit einer Liste von Netzwerken. Dabei enthält jedes Netzwerk eine Struktur, die logische und arithmetische Ausdrücke, Aufrufe von Funktionsbausteinen, einen Sprung oder eine Return-Anweisung enthalten kann.

Hierbei werden Bausteine benutzt, die aus der booleschen Algebra bekannt sind. Bausteine und Variablen werden mit Verbindungslinien verknüpft. Der Signalfluss verläuft im Netzwerk von links nach rechts. Der Signalfluss im Editor verläuft von oben nach unten, beginnend mit Netzwerk 1.

Beispiel:

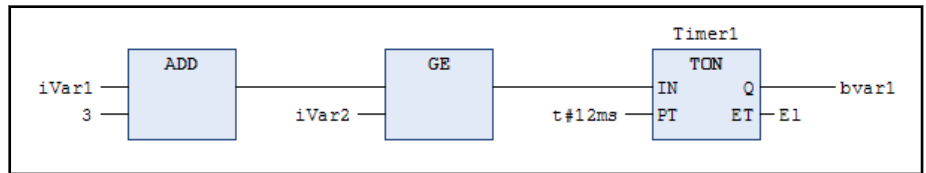


Abb. 9-1:



CFC ist ebenfalls eine Programmiersprache nach dem Prinzip von FUP, jedoch mit folgenden Unterschieden:

- Der CFC-Editor ist nicht netzwerkorientiert.
- Sie können die Elemente im CFC-Editor frei platzieren.
- Direktes Einfügen von Rückkopplungen ist möglich.
- Die Abarbeitungsreihenfolge wird durch eine Liste der aktuell eingefügten Elemente bestimmt, die Sie verändern können.

Siehe auch

- [Kap. "Programmieren in Funktionsplan \(FUP\)" auf Seite 95](#)
- [Kap. 18.2.9 "Menü 'FUP/KOP/AWL'" auf Seite 702 \(Befehle\)](#)
- [Kap. 9.3.2 "Continuous Function Chart \(CFC\)" auf Seite 99](#)

Kontaktplan (KOP)

Der Kontaktplan (KOP) ist eine grafisch orientierte Programmiersprache, die dem Prinzip eines elektrischen Schaltplans angenähert ist. Einerseits eignet sich der Kontaktplan dazu, logische Schaltwerke zu konstruieren, andererseits kann man aber auch Netzwerke wie im FUP erstellen.

Daher können Sie den KOP sehr gut zum Steuern von Aufrufen anderer Programmbausteine verwenden.

Der Kontaktplan besteht aus einer Folge von Netzwerken.

Ein Netzwerk wird auf der linken Seite von einer vertikalen Linie (Stromschiene) begrenzt.

Ein Netzwerk enthält einen Schaltplan aus Kontakten, Spulen, optional Bausteinen (POUs) und Verbindungslinien.

Auf der linken Seite eines Netzwerks gibt es einen Kontakt oder eine Folge von Kontakten, die von links nach rechts den Zustand ON oder OFF weitergeben, was den booleschen Werten `TRUE` und `FALSE` entspricht. Zu jedem Kontakt gehört eine boolesche Variable. Wenn diese Variable `TRUE` ist, dann wird der Zustand über die Verbindungslinie von links nach rechts weitergegeben. Ansonsten wird `OFF` weitergegeben. Somit erhält/erhalten die Spule(n) im rechten Teil des Netzwerks den von links kommenden Wert `ON` oder `OFF` und dementsprechend wird der Wert `TRUE` oder `FALSE` in die ihnen zugewiesene boolesche Variable geschrieben.

Wenn die Elemente in Reihe geschaltet werden, dann bedeutet dies eine UND-Verknüpfung. Wenn sie parallel geschaltet werden, dann bedeutet dies eine ODER-Verknüpfung.

Ein Strich durch ein Element bedeutet eine Negierung des Elements. Die Negation eines Eingangs oder Ausganges wird durch ein Kreissymbol angezeigt.

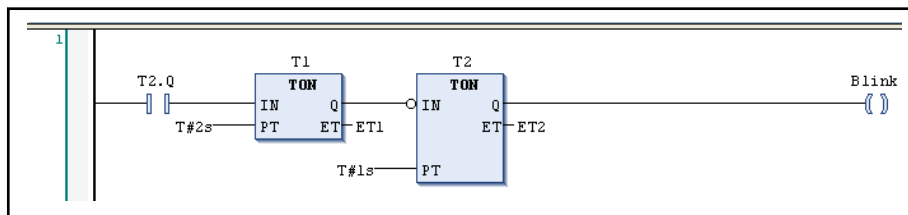
Beispiel:

Abb. 9-2:

Die IEC 61131-3 definiert einen vollständigen KOP-Befehlssatz, bestehend aus verschiedenen Typen von Kontakten und Spulen.

Kontakte leiten den Strom (entsprechend ihres Typs) von links nach rechts.

Spulen speichern den eingehenden Wert.

Kontakte und Spulen sind booleschen Variablen zugeordnet.

Sie können ein KOP-Netzwerk durch Sprünge, Rücksprünge, Marken und Kommentare ergänzen.

Siehe auch

- [Kap. "Programmieren in Kontaktplan \(KOP\)" auf Seite 97](#)
- [Kap. 18.2 "Menübefehle" auf Seite 610](#)

Anweisungsliste (AWL)

Die Anweisungsliste ist eine Assembler-ähnliche IEC 61131-konforme Programmiersprache. Sie unterstützt akkumulator-basiertes Programmieren.

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 eine Marke stehen, gefolgt von einem Doppelpunkt. Sie dient der Kennzeichnung der Anweisung und Sie können die Marke als Sprungziel verwenden. Ein Kommentar muss das letzte Element in einer Zeile sein. Leere Zeilen können zwischen Anweisungen eingefügt werden.

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 Parameter aus dem Akkumulator ausgeführt. Das Ergebnis der Operation wird wieder in den Akkumulator geschrieben, von wo Sie es mit Hilfe einer **ST**-Anweisung gezielt speichern sollten.

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.

Beispiel:

1	LD	BVar1		
	ST	inst.IN	starts timer with rising e...	
	JMPC	mark1		
	CAL	inst (		
		PT:=t1,		is True
		ET:=>t2)		
	LD	inst.Q	gets TRUE delay time after...	
	ST	inst2.IN	starts timer with rising e...	
2				
		mark1:		
	LD	iVar2		
	ADD	230		

Abb. 9-3:

Siehe auch

- [Kap. "Programmieren in Anweisungsliste \(AWL\)" auf Seite 98](#)
- [Kap. "Modifikatoren und Operatoren in AWL" auf Seite 278](#)
- [Kap. 18.2.9 "Menü 'FUP/KOP/AWL'" auf Seite 702](#)

Programmieren in Funktionsplan (FUP)

Anlegen einer POU in der Implementierungssprache Funktionsplan (FUP)

1. Selektieren Sie eine Applikation im Projektexplorer.
2. Wählen Sie den Befehl **Hinzufügen ► POU**.
Der Dialog **POU hinzufügen** öffnet sich.
3. Geben Sie einen Namen ein und wählen Sie die Implementierungssprache **Funktionsplan (FUP)**.
Klicken Sie auf **Fertigstellen**.
Die POU wird zum Projektexplorer hinzugefügt und im Editor geöffnet. Der Editor enthält den Deklarationseditor im oberen Teil und den Implementierungsteil mit einem leeren Netzwerk im unteren Teil. Automatisch wird auch die Ansicht **Werkzeug** geöffnet, in der die passenden Elemente, Operatoren und Funktionsbausteine für die FUP-Programmierung bereitstehen.

Siehe auch

- [Kap. 18.1.17 "Objekt 'POU'" auf Seite 552](#)
- [Kap. 9.3.1 "FUP/KOP/AWL" auf Seite 92](#)

Netzwerk programmieren

1. Klicken Sie in das automatisch eingefügte leere Netzwerk im Implementierungsteil.
Das Netzwerk wird gelb hinterlegt, der Bereich mit der Netzwerknummer am linken Rand wird rot hinterlegt.
2. Öffnen Sie das Kontextmenü mit der rechten Maustaste.
Sie erhalten unter anderem die Einfügebefehle für die an dieser Stelle einfügbaren Elemente.

3. Fügen Sie die für Ihre Programmierung benötigten Elemente über die Menübefehle oder durch Ziehen der Elemente aus der Werkzeugbox ein.
4. Wählen Sie beispielsweise den Befehl **Zuweisung einfügen**.
Eine Zuweisungslinie wird eingefügt. Jeweils drei Fragezeichen stehen für Zuweisungsquelle und Zuweisungsziel.
5. Selektieren Sie die Fragezeichen und ersetzen Sie diese mit der gewünschten Variable. Die Eingabehilfe steht zur Verfügung.
6. Bewegen Sie den Cursor über die Zuweisungslinie.
Die möglichen Einfügepositionen für weitere Elemente werden als graue Rauten angezeigt. Ein Klick auf eine Raute selektiert die Position und es stehen wiederum die passenden Einfügebefehle zur Verfügung.
7. Alternativ können Sie ein Element aus der Werkzeugbox mit der Maus ins Netzwerk ziehen. Klicken Sie beispielsweise in der Werkzeugbox auf das Baustein-Element, halten die Maustaste gedrückt und bewegen den Cursor über das Netzwerk.
Jede mögliche Einfügeposition leuchtet grün auf.
8. Lassen Sie die Maustaste los, um den Baustein einzufügen.
Die Bausteinbox wird im Netzwerk dargestellt. Der Bausteintyp im Inneren und der im Falle eines Funktionsbausteins benötigte Instanzname oberhalb der Box sind noch mit drei Fragezeichen freigehalten.
9. Selektieren Sie die Zeichenfolge ??? im Inneren der Box und ersetzen Sie sie mit dem Bausteinnamen. Die Eingabehilfe steht zur Verfügung.
Die Eingänge und Ausgänge des gewählten Bausteins werden dargestellt. Sie sind noch mit Fragezeichen freigehalten, bei Funktionsbausteinen ebenso der Instanzname.

Siehe auch

- [Kap. 18.2.9 "Menü 'FUP/KOP/AWL'" auf Seite 702](#)
- [Kap. 9.3.1 "FUP/KOP/AWL" auf Seite 92](#)

Leitungsverzweigungen (Subnetzwerke) programmieren

1. Fügen Sie im Implementierungsteil Ihrer POU ein neues Netzwerk über den Befehl **FUP/KOP/AWL ► Netzwerk einfügen** oder aus der Werkzeugbox ein.
2. Ziehen Sie beispielsweise einen Operator **ADD** in das leere Netzwerk und ersetzen Sie die Zeichen ??? mit 2 Variablen vom Datentyp `INT`.
3. Ziehen Sie das Element **Leitungsverzweigung** aus der Werkzeugbox in Ihre Implementierung und lassen Sie die Maustaste an der grünen Einfügeposition direkt am Ausgang des Operator los.
Die Leitungsverzweigung spaltet die Abarbeitungslinie am Ausgang der Operator-Box in 2 Subnetzwerke.
4. Jedem der beiden Subnetzwerke können Sie nun weitere FUP-Elemente und auch weitere Leitungsverzweigungen hinzufügen.

Siehe auch

- [Kap. "FUP/KOP/AWL-Element 'Leitungsverzweigung'" auf Seite 284](#)
- [Kap. "Befehl 'Leitungsverzweigung einfügen'" auf Seite 710](#)

Programmieren in Kontaktplan (KOP)

Anlegen einer POU in der Implementierungssprache Kontaktplan (KOP)

1. Selektieren Sie die Applikation im Projektexplorer.
2. Wählen Sie den Befehl **Hinzufügen ► POU**.
Es öffnet sich der Dialog **POU hinzufügen**.
3. Geben Sie einen Namen ein und wählen Sie die Implementierungssprache **Kontaktplan (KOP)**.
Klicken Sie auf **Fertigstellen**.
IndraLogic fügt die POU zum Projektexplorer hinzu und öffnet sie im Editor. Im Implementierungsteil ist ein leeres Netzwerk eingefügt. Das leere Netzwerk ist links durch eine vertikale Linie, die eine Stromschiene darstellt, begrenzt. Automatisch wird auch die Ansicht **Werkzeuge** geöffnet, in der die passenden Elemente, Operatoren und Funktionsbausteine für die KOP-Programmierung bereitgestellt werden.

Hinzufügen eines Kontakts und eines Funktionsbausteins (TON)

Voraussetzung: Eine POU mit der Implementierungssprache KOP ist im Editor geöffnet und ein leeres Netzwerk ist eingefügt.

1. Klicken Sie in der Ansicht **Werkzeuge** auf die Kategorie **Kontaktplan Elemente**.
2. Klicken Sie auf das Element **Kontakt**, ziehen Sie es in Ihr Netzwerk und lassen Sie die Maus auf der Einfügeposition **Hier starten** los.
Der Kontakt wird links im Netzwerk direkt an der vertikalen Linie hinzugefügt.
3. Klicken Sie auf ??? und geben Sie den Bezeichner einer booleschen Variablen ein. Hierfür steht Ihnen auch die Eingabehilfe zur Verfügung.
4. Klicken Sie in der Ansicht **Werkzeuge** auf die Kategorie **Funktionsbausteine** und ziehen Sie den Funktionsbaustein **TON** auf eine Einfügeposition auf der Verbindungslinie rechts von dem eingefügten Kontakt.
IndraLogic fügt den Baustein **TON** rechts von dem Kontakt ein. Der Kontakt ist mit dem Eingang **IN** des TON-Bausteins verbunden.
5. Geben Sie am Eingang **PT** eine Zeitkonstante, beispielsweise **T#3s** ein.
Wenn die Variable Ihres Kontakts **TRUE** wird, wird auch der Eingang **IN** des TON-Bausteins **TRUE**. Mit einer Einschaltverzögerung von beispielsweise **T#3s** leitet der TON-Baustein den Wert **TRUE** an den Ausgang **Q** weiter.

Einfügen einer geschlossenen Leitungsverzweigung

Voraussetzung: Eine POU mit der Implementierungssprache KOP ist im Editor geöffnet und ein leeres Netzwerk ist eingefügt.

1. Klicken Sie in das leere Netzwerk und wählen Sie den Befehl **FUP/KOP/AWL ► Kontakt einfügen**.
2. Selektieren Sie die Verbindungslinie links des Kontakts und wählen Sie den Befehl **FUP/KOP/AWL ► Verzweigung Startpunkt setzen**.
Der Startpunkt auf der Verbindungslinie wird mit einem roten Rechteck gekennzeichnet. Alle möglichen Endpunkte der Verzweigung kennzeichnet IndraLogic mit einem blauen Rechteck.
3. Klicken Sie auf ein blaues Rechteck, um den Endpunkt Ihrer geschlossenen Leitungsverzweigung festzulegen.

IndraLogic fügt die Leitungsverzweigung zwischen dem Start- und dem Endpunkt ein. Der Programmfluss wird beide Zweige bis zum Endpunkt durchlaufen.

Wenn Sie die Leitungsverzweigung nicht an einem Kontakt, sondern an einem Baustein einfügen, wird der Baustein nur aufgerufen, wenn keiner der anderen Zweige `TRUE` ist.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "Geschlossene Leitungsverzweigung" auf Seite 286](#)
- [Kap. "KOP-Element 'Kontakt'" auf Seite 285](#)

Programmieren in Anweisungsliste (AWL)



AWL kann bei Bedarf über die IndraWorks-Optionen aktiviert werden.

Extras ▶ Optionen ▶ IndraLogic 2G ▶ FUP, KOP und AWL ▶ AWL ▶ AWL aktivieren

Anlegen einer POU in der Implementierungssprache Anweisungsliste (AWL)

1. Selektieren Sie die Applikation im Projektextplorer.
2. Wählen Sie den Befehl **Hinzufügen ▶ POU**.
Es öffnet sich der Dialog **POU hinzufügen**.
3. Geben Sie einen Namen ein und wählen Sie die Implementierungssprache **Anweisungsliste (AWL)**.
Klicken Sie auf **Fertigstellen**.
IndraLogic fügt die POU zum Projektextplorer hinzu und öffnet sie im Editor. Im Implementierungsteil ist bereits ein Netzwerk eingefügt.

Programmieren eines Netzwerks (beispielsweise einer ADD-Operation)

Voraussetzung: Eine POU (AWL) ist im Editor geöffnet und besitzt ein leeres Netzwerk.

1. Klicken Sie in die 1. Spalte der farblich markierten Zeile und geben Sie den Operator `LD` ein.
2. Drücken Sie die Taste `<Tab>`
Der Cursor springt in die 2. Spalte
3. Geben Sie den ersten Summanden Ihrer ADD-Operation ein, zum Beispiel `6`
4. Drücken Sie `<Strg>+<Eingabe>` oder wählen Sie den Befehl **FUP/KOP/AWL ▶ AWL-Zeile danach einfügen**
IndraLogic fügt eine neue Anweisungszeile unterhalb ein. Der Fokus liegt in der ersten Spalte dieser Zeile.
5. Geben Sie `ADD` ein und drücken Sie `<Tab>`.
6. Geben den 2. Summanden Ihrer ADD-Operation ein, zum Beispiel `12`.
7. Drücken Sie `<Strg>+<Eingabe>`
8. Geben den Operator `ST` ein und drücken Sie `<Tab>`.
9. Geben Sie eine Variable vom Datentyp `INT` an, zum Beispiel `iVar`.
Das Ergebnis, im Beispiel `16`, wird in der `iVar` gespeichert.

Aufrufen eines Funktionsbausteins

Voraussetzung: Eine POU (AWL) ist im Editor geöffnet und besitzt ein leeres Netzwerk. Im Deklarationsteil ist eine Variable mit Datentyp <Funktionsbaustein> deklariert, beispielsweise C1:CTU;

1. Klicken Sie in die erste Spalte der farblich markierten Zeile und wählen Sie den Befehl **FUP/KOP/AWL ► Bausteinaufruf einfügen**.

Die Eingabehilfe öffnet sich.

2. Wählen Sie in der Kategorie **Funktionsbausteine** oder in der Kategorie **Bausteinaufrufe** den gewünschten Funktionsbaustein aus, beispielsweise den Zähler **CTU** aus der Bibliothek **Standard**, und klicken Sie auf **OK**.

→ IndraLogic fügt den ausgewählten Funktionsbaustein **CTU** wie folgt ein:

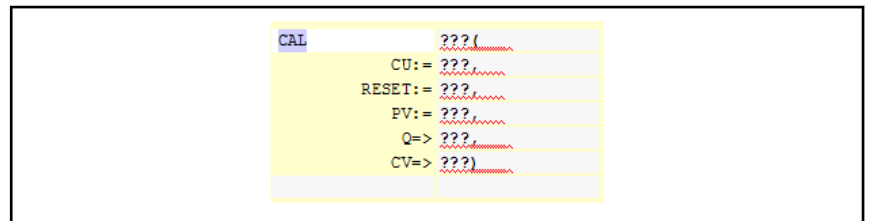


Abb. 9-4:

3. Ersetzen Sie die Zeichenfolgen ??? mit dem Variablennamen und den Werten oder Variablen für die Ein-/Ausgänge des Funktionsbausteins.
4. Alternativ zum Einfügen des Funktionsbausteins über die Eingabehilfe können Sie den Aufruf direkt im Editor eingeben.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "Modifikatoren und Operatoren in AWL" auf Seite 278](#)

9.3.2 Continuous Function Chart (CFC)

Die Implementierungssprache "Continuous Function Chart (CFC)" ist eine grafische Programmiersprache und erweitert die Standardsprachen der IEC 61131-3.

Die grafische Programmierung in "CFC" ähnelt der Programmierung in "FUP", bietet aber wesentliche Erweiterungen:

- Eingefügte Elemente können frei positioniert werden (z. B. Funktionsbausteine)
- Elemente können mit grafischen Verbindungen zu einem Netzwerk verschaltet werden
- Zwischen den Elementen können Rückkopplungen eingefügt werden
- Möglichkeit zur Seitenanordnung / komplexe Programme können damit auf mehrere Seiten verteilt angeordnet werden.

Ein Vorteil von CFC ist, dass der Funktionsplan (Programm) wie ein Schaltplan, oder ein Blockschaltbild gelesen werden kann.

Die Ausführungsreihenfolge eines CFC-Funktionsplans ist datenflussbasierend, wobei ein Programmierbaustein jeweils mehrere Datenflüsse verarbeiten kann. In diesem Fall haben die Datenflüsse keine gemeinsamen Daten und im Editor können sich mehrere Netzwerke befinden, die untereinander keine Verbindungen haben.

Programmierbausteine in FUP, KOP oder AWL haben dagegen eine netzwerkbasierende Ausführungsreihenfolge.

Die Implementierungssprache "Continuous Function Chart (CFC) - seitenorientiert" ist ebenfalls eine grafische Programmiersprache und erweitert die Standardsprachen der IEC 61131-3. "CFC - seitenorientiert" bietet Vorteile bei der grafischen Programmierung von komplexen und raumeinnehmenden Funktionsplänen, in dem die Netzwerke auf beliebig vielen Seiten verteilt angeordnet werden können.

Auf den jeweiligen Seiten stehen Ihnen die selben Elemente und Befehle zur Verfügung wie in Continuous Function Chart (CFC). Zusätzlich besitzt jede Seite zwei Randbereiche (links / rechts):

- Im linken Randbereich können Eingänge und Zielverbindungsmarken angeordnet werden
- Im rechten Randbereich können Ausgänge und Quellverbindungsmarken angeordnet werden

Die Randbereiche bieten Unterstützung beim Einfügen von Verbindungslinien und verhelfen zu einer besseren Übersichtlichkeit.



Es ist nicht möglich einen Programmierbaustein zwischen den Implementierungssprache Continuous Function Chart (CFC) - seitenorientiert und Continuous Function Chart (CFC) umzuschalten.

Siehe auch

- [Kap. "CFC-Editor" auf Seite 288](#)
- [Kap. "CFC-Editor im Onlinebetrieb" auf Seite 293](#)
- [Kap. "CFC-Editor seitenorientiert" auf Seite 291](#)

Im CFC-Editor programmieren

Sie können im CFC-Editor Programmierbausteine miteinander verdrahten und anschauliche Blockschaltbilder erstellen.

Der Editor unterstützt Sie mit folgenden Funktionen:

- Programmieren mit Elementen und Verbindungslinien
- Ziehen von Instanzen und Variablen in den Editierbereich
- Autorouting der Verbindungslinien
- Automatisches Verknüpfen
- Fixieren von Verbindungslinien durch Kontrollpunkte
- Kollisionserkennung
- Eingabeunterstützung bei Verbindungsmarken
- Forcen und Schreiben von Werten im Onlinebetrieb
- Verschieben der Selektion mittels Pfeiltasten
- Reduzierte Darstellung eines Bausteins ohne unverbundene Anschlüsse

Elemente einfügen und mit Verbindungslinien verschalten

1. Platzieren Sie ein Element **Baustein** und ein Element **Ausgang** im Editor.
2. Klicken Sie auf den Ausgang des Elements **Baustein**.
Der Ausgang wird mit einem roten Quadrat gekennzeichnet.
3. Ziehen Sie mit gedrückter Maustaste eine Verbindungslinie vom Baustein Ausgang des Elements **Baustein** zum Bausteineingang des Elements **Ausgang**.

Beim Erreichen des Bausteineingangs ändert der Cursor sein Symbol.

4. Lassen Sie die Maustaste los.

Der Ausgangspin des Bausteins ist mit dem Eingangspin des Ausgangs verdrahtet.



Alternativ können Sie die beiden Pins mit gedrückter <Strg>-Taste selektieren und den Befehl **Selektierte Anschlüsse verbinden** auswählen.

Verbindungsmarken erzeugen

Voraussetzung: Sie haben einen CFC-Programmierbaustein mit verbundenen Elementen.

1. Selektieren Sie eine Verbindungslinie zwischen zwei Elementen.
→ Die Verbindungslinie wird selektiert dargestellt. Die Enden der Verbindungslinie sind mit roten Quadraten markiert ■.
2. Wählen Sie den Befehl **CFC ► Verbindungsmarke**.
Die Verbindung wird aufgetrennt und durch eine "Quellverbindungsmarke" und eine "Zielverbindungsmarke" ersetzt.
Der Name der beiden Marken wird automatisch erzeugt, kann danach aber beliebig geändert werden.



Abb. 9-5:

3. Zum Ändern des Namens klicken Sie in die Quellverbindungsmarke.
→ Der Name ist jetzt selektiert und kann editiert werden.
4. Geben Sie als Name für die Quellverbindungsmarke z. B. SimpleMark ein.
→ Quellverbindungsmarke und Zielverbindungsmarke erhalten automatisch den geänderten Namen.



Abb. 9-6:

Siehe auch

- [Kap. "Befehl 'Verbindungsmarke'" auf Seite 699](#)

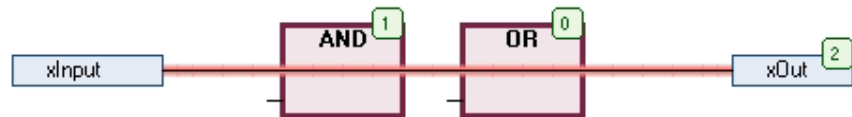
Kollisionen lösen und
Verbindungslinien durch
Kontrollpunkte fixieren

Das nachfolgende Beispiel zeigt Ihnen die Verwendung des Befehls **"Alle Verbindungen routen"** sowie die Verwendung von Kontrollpunkten.

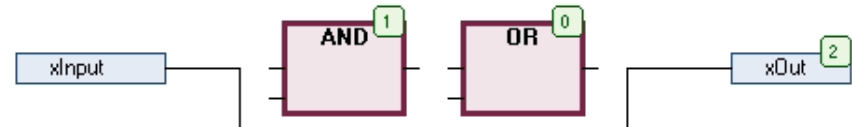
1. Platzieren Sie die Elemente **Eingang** und **Ausgang** gemäß folgendem Beispiel und verbinden Sie die Elemente



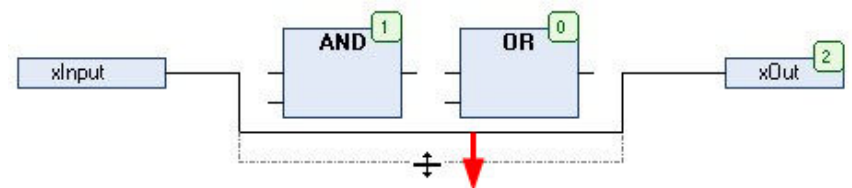
2. Platzieren Sie zwei Elemente des Typs "**Baustein**" auf der Linie.
 → Die Verbindungslinie und die Bausteine werden wegen der Kollision rot gezeichnet.



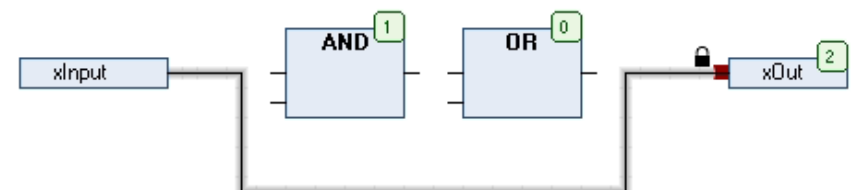
3. Führen Sie den Befehl **CFC ► Routing ► Alle Verbindungen routen** aus.
 → Die Kollision wird gelöst



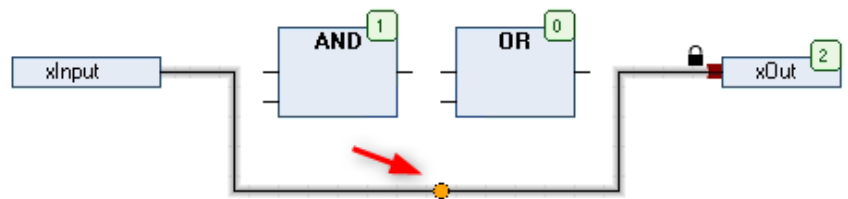
4. Verschieben Sie die Position der Verbindungslinie.



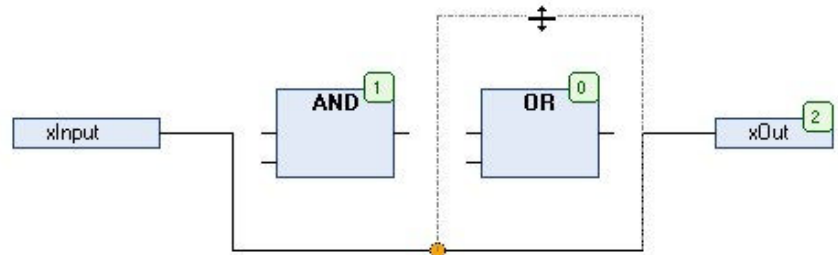
→ Die Verbindungslinie wurde manuell verändert, so dass sie IndraLogic für das automatische Routing sperrt. Dies wird durch ein Schloss am Verbindungsende angezeigt.



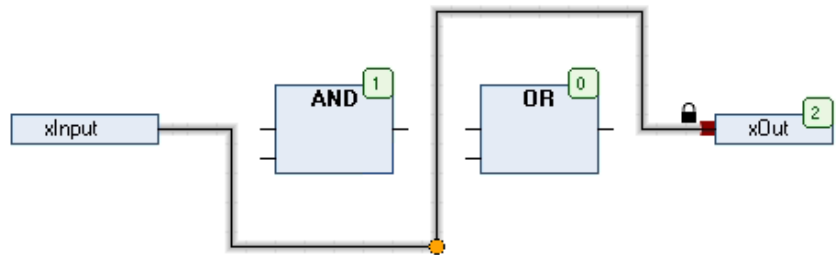
5. Platzieren Sie jetzt einen Kontrollpunkt auf der Verbindungslinie.
 Selektieren Sie hierzu die Verbindungslinie an der Stelle, an der Sie einen Kontrollpunkt setzen wollen und wählen Sie den Befehl **CFC ► Routing ► Kontrollpunkt erzeugen**
 → Auf der Verbindungslinie wird an der selektierten Stelle ein Kontrollpunkt erzeugt. Die Verbindungslinie ist an der Position des Kontrollpunkts im Editor fixiert.



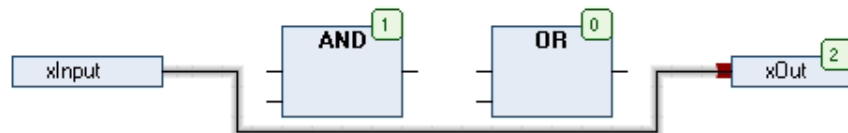
6. Verändern Sie die Verbindung gemäß nachfolgendem Beispiel



Durch den Kontrollpunkt können Sie die Verbindungslinie individuell verändern.
Eine Verbindungslinie kann jeweils mit mehreren Kontrollpunkten fixiert werden.



7. Entfernen Sie den Kontrollpunkt mit dem Befehl **CFC ▶ Routing ▶ Kontrollpunkt entfernen**.
Kontrollpunkte können nur einzeln gelöscht werden.
8. Lösen die Verbindung mit dem Befehl **CFC ▶ Routing ▶ Verbindung lösen**, oder alternativ durch einen Mausklick auf das Schloss-Symbol der selektierten Verbindung.
9. Selektieren Sie die Verbindungslinie und wählen Sie den Befehl **CFC ▶ Routing ▶ Alle Verbindungen routen**.
→ Die Verbindungslinie wird automatisch geroutet.



Verbindungen innerhalb einer Gruppe werden nicht automatisch geroutet!

Siehe auch

- [Kap. "Befehl 'Kontrollpunkt entfernen'" auf Seite 698](#)

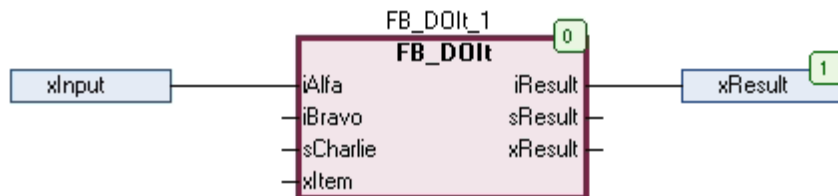
Darstellung eines Bausteins reduzieren

- [Kap. "Befehl 'Verbindung lösen'" auf Seite 696](#)
- [Kap. "Befehl 'Alle Verbindungen routen'" auf Seite 698](#)

Voraussetzung: Ein CFC-Programmierbaustein ist geöffnet. Im Editor werden dessen Bausteine mit allen deklarierten Anschlüssen angezeigt.

1. Selektieren Sie einen Baustein, dessen Anschlüsse teilweise unverbunden sind.

Beispiel:



→ Der Baustein beansprucht unnötigen Darstellungsraum für die unverbundenen Anschlüsse.

2. Zum Ausblenden der unverbundenen Anschlüsse wählen Sie den Befehl **CFC ► Pins ► Nicht verbundene Anschlüsse entfernen**

→ Der Baustein benötigt jetzt weniger Platz und wird nur mit den verbundenen Anschlüssen dargestellt:



Siehe auch

- [Kap. 17.1.2 "Generelle Funktionalitäten in allen grafischen Editoren" auf Seite 240](#)
- [Kap. "CFC-Editor" auf Seite 288](#)
- [Kap. "CFC-Editor seitenorientiert" auf Seite 291](#)
- [Kap. "Elemente" auf Seite 297](#)

9.3.3 Strukturierter Text (ST), Erweiterter Strukturierter Text (ExST)

Der ST-Editor dient dem Programmieren von POU's in der IEC-61131-3-Programmiersprache Strukturierter Text (ST) und Erweiterter Strukturierter Text. Der Erweiterte Strukturierte Text bietet bezüglich der Norm IEC 61131-3 einige zusätzliche Funktionen.

Der Strukturierte Text ist eine Programmiersprache, vergleichbar mit anderen Hochsprachen wie C oder PASCAL, die die Entwicklung komplexer Algorithmen erlaubt.

Der Programmcode besteht aus einer Kombination von **Ausdrücken** und **Anweisungen**, die auch bedingt (IF...THEN...ELSE) oder in Schleifen (WHILE...DO) ausgeführt werden können.

Ein Ausdruck ist ein Konstrukt, das nach seiner Auswertung einen Wert zurück liefert. Ausdrücke sind auch Operatoren und Operanden zusammen. Auch Zuweisungen können Sie als Ausdruck verwenden. Ein Operand kann eine Konstante, eine Variable, ein Funktionsaufruf oder ein weiterer Ausdruck sein.

Anweisungen steuern, wie die Ausdrücke verarbeitet werden sollen.

Für diesen Texteditor können Sie in den Dialogen **Optionen** und **Anpassen** des Menüs **Extras** verschiedene Einstellungen bezüglich Verhalten, Aussehen und Menüs vornehmen. Für diesen Editor stehen auch die bekannten Windows-Funktionen zur Verfügung (zum Beispiel IntelliMouse).

Siehe auch

- [Kap. "Programmieren in Strukturiertem Text \(ST\)" auf Seite 105](#)
- [Kap. "Dialog 'Optionen- 'Texteditor'" auf Seite 776](#)
- Dialog 'Anpassen' - 'Menü'

ExST - Erweiterter Strukturiertes Text

Erweiterter Strukturiertes Text (ExST) ist eine IndraLogic-spezifische Erweiterung bezüglich des IEC 61131-3 Standards für Strukturiertes Text (ST).

Siehe auch

- [Kap. "ExST-Zuweisung 'R='" auf Seite 245](#)
- [Kap. "ExST-Zuweisung 'S='" auf Seite 244](#)
- [Kap. "ExST-Zuweisung ':=als Ausdruck" auf Seite 246](#)

Programmieren in Strukturiertem Text (ST)

Prinzip Sie programmieren die Programmiersprachen "Strukturierter Text" und "Erweiterter Strukturierter Text" im ST-Editor.

Der Programmcode besteht aus einer Kombination von Ausdrücken und Anweisungen, die auch bedingt oder in Schleifen ausgeführt werden können.

Jede Anweisung müssen Sie mit einem ; abschließen.

Die Deklaration der Variablen erfolgt im Deklarationseditor.

Anlegen einer POU in der Implementierungssprache Strukturiertes Text (ST)

1. Selektieren Sie eine Applikation im Projektexplorer.
2. Wählen Sie den Befehl **Hinzufügen ► POU**.
Der Dialog **Objekt Hinzufügen Assistent** öffnet sich.
3. Geben Sie einen Namen ein und wählen Sie die **Implementierungssprache Strukturiertes Text (ST)**. Klicken Sie auf **Fertigstellen**.
Die POU wird im Projektexplorer hinzugefügt und im Editor geöffnet.
Im oberen Teil der POU fügen Sie nun die Variablendeklarationen ein, im unteren Teil der POU geben Sie den ST-Programmcode ein.

Siehe auch

- [Kap. "ST-Ausdrücke" auf Seite 242](#)
- [Kap. 17.1.3 "Strukturierter Text und Erweiterter Strukturierter Text \(ExST\)" auf Seite 241](#)
- [Kap. "ST - Aufruf Funktionsbaustein" auf Seite 252](#)
- [Kap. "ST - Kommentare" auf Seite 252](#)
- [Kap. 9.2.1 "Deklarationseditor verwenden" auf Seite 87](#)

9.3.4 Ablaufsprache (AS) / Sequential Function Chart (SFC)

Mit dem AS-Editor programmieren Sie POU's in der IEC 61131-3 Programmiersprache "Ablaufsprache" (Sequential Function Chart, SFC).

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 Programmierobjekte sind, den "Schritt"-Elementen zugeordnet. Der Ablauf der Abarbeitung der Schritte wird durch "Transitions"-Elemente gesteuert.

Siehe auch

- [Kap. "AS-Editor" auf Seite 253](#)

Programmieren in Ablaufsprache (AS / SFC)

Anlegen einer POU in der Implementierungssprache SFC

1. Selektieren Sie eine Applikation im Projektextplorer.
2. Wählen Sie den Befehl **Hinzufügen ► POU**.
Der Dialog "POU hinzufügen" öffnet sich
3. Geben Sie einen Namen ein und wählen Sie die Implementierungssprache **Ablaufsprache (AS)**.
Klicken Sie auf **Fertigstellen**.
IndraLogic fügt die POU zum Projektextplorer hinzu und öffnet sie im Editor.

Hinzufügen einer Schritt-Transition

1. Selektieren Sie die Transition nach dem Init-Schritt.
Die Transition ist rot markiert.
2. Wählen Sie den Befehl **SFC ► Schritt-Transition danach einfügen**.
IndraLogic fügt Schritt **Step0** und Transition **Trans0** ein.
3. Selektieren Sie die Transition **Trans0** und wählen Sie den Befehl **SFC ► Schritt-Transition einfügen**.
IndraLogic fügt Transition **Trans1** und Schritt **Step1** vor **Trans0** ein.

Sie können die Elemente **Schritt** und **Transition** auch mit Drag&Drop von der Ansicht **Werkzeuge** in das Diagramm ziehen.

Siehe auch

- [Kap. "AS-Elemente 'Schritt' und 'Transition'" auf Seite 262](#)
- [Kap. "Befehl 'Schritt-Transition einfügen'" auf Seite 683](#)
- [Kap. "Befehl 'Schritt-Transition danach einfügen'" auf Seite 683](#)

Hinzufügen einer Eingangsaktion

1. Selektieren Sie den Schritt **Step0**.
2. Wählen Sie den Befehl **SFC ► Eingangsaktion hinzufügen**.
Im Standardfall erhalten Sie eine Eingabeaufforderung zur Festlegung des Duplizierungsmodus für die Schrittaktionen. Sie entscheiden damit, ob beim künftigen Kopieren des Schritts die Referenz zu den bestehenden Schrittaktionsobjekten kopiert wird, oder ob die Objekte "eingebettet" werden sollen. Das Einbetten hat zur Folge, dass beim Kopieren des Schritts neue Schrittaktionsobjekte angelegt werden. Der Duplizierungsmodus ist in der Schritteigenschaft "Beim Kopieren duplizieren" definiert. Solange sie deaktiviert ist, rufen die kopierten Schritte dieselben Aktionen auf wie der aktuelle Schritt.
Das Erscheinen der Eingabeaufforderung können Sie im Dialog, oder den Indraworks-Optionen generell abschalten.
[Siehe: "Abfragedialog zum Wählen des Duplizierungsmodus" auf Seite 110](#)
3. Belassen Sie für dieses Beispiel die Standardeinstellung **"Referenz kopieren..."** und bestätigen Sie mit **"OK"**.
Der Dialog "Eingangsaktion hinzufügen" öffnet sich.

4. Geben Sie für den Namen "Step0_entry" ein und wählen Sie die Implementierungssprache "ST". Klicken Sie auf **Fertigstellen**.

IndraLogic fügt die Aktion "Step0_entry" unterhalb der POU im Projekexplorer ein und öffnet die Aktion im Editor.

In der Eingangsaktion "Step0_entry" programmieren Sie nun Anweisungen, die einmalig bei Aktivierung des Schritts **Step0** ausgeführt werden sollen.

5. Schließen Sie den Editor von "Step0_entry".

Der Schritt **Step0** ist nun mit einem **E** in der linken unteren Ecke gekennzeichnet. Mit einem Doppelklick auf diese Markierung öffnen Sie den Editor.

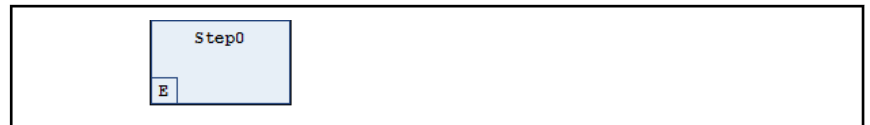


Abb. 9-7:

Die Eingangsaktion Step0_entry steht nun in den Eigenschaften des Schritts unter "Eingangsaktion". Dort können Sie bei Bedarf auch eine andere Aktion auswählen.

6. Selektieren und kopieren sie Step0. Drücken Sie [Strg]+[V].

In der eingefügten Kopie des Schritts findet sich diesselbe, oben eingefügte Eingangsaktion wieder. Der neue Schritt ruft also genau diesselbe Aktion auf.

Siehe auch

- ["Schrittaktionen" auf Seite 265](#)

Hinzufügen einer Ausgangsaktion

1. Selektieren Sie den Schritt **Step0**.
2. Wählen Sie den Befehl **SFC ► Ausgangsaktion hinzufügen**.

Im Standardfall erhalten Sie eine Eingabeaufforderung zur Festlegung den Duplizierungsmodus für die Schrittaktionen des Schritts, wie unter "Hinzufügen einer Eingangsaktion" beschrieben. Danach öffnet sich der Dialog "Ausgangsaktion hinzufügen".

Siehe: ["Abfragedialog zum Wählen des Duplizierungsmodus" auf Seite 110](#)

3. Geben Sie für den Namen "Step0_exit" ein und wählen Sie die Implementierungssprache "ST". Klicken Sie auf **Fertigstellen**.

IndraLogic fügt die Aktion "Step0_exit" unterhalb der POU im Projekexplorer ein und öffnet die Aktion im Editor.

In der Ausgangsaktion "Step0_exit" programmieren Sie nun Anweisungen, die einmalig vor dem Deaktivieren des Schritts **Step0** ausgeführt werden sollen.

4. Schließen Sie den Editor von "Step0_exit".

Der Schritt **Step0** ist nun mit einem **X** in der rechten unteren Ecke gekennzeichnet. Mit einem Doppelklick auf diese Markierung öffnen Sie den Editor.



Abb. 9-8:

Sie können die Ausgangsaktion in den Eigenschaften des Schritts unter **Ausgangsaktion** definieren. Dort können Sie bei Bedarf auch eine andere Aktion auswählen.

Siehe auch

- ["Schrittaktionen" auf Seite 265](#)

Hinzufügen einer Aktion

1. Führen Sie einen Doppelklick auf den Schritt **Step0** aus.

Im Standardfall erhalten Sie eine Eingabeaufforderung zur Festlegung des Duplizierungsmodus für die Schrittaktionen des Schritts, wie unter "Hinzufügen einer Eingangsaktion" beschrieben. Es öffnet sich der Dialog **Aktion hinzufügen**.

Siehe: ["Abfragedialog zum Wählen des Duplizierungsmodus" auf Seite 110](#)

2. Geben Sie "Step0_active" als Namen ein und wählen Sie die Implementierungssprache "ST". Klicken Sie auf **Fertigstellen**.

IndraLogic fügt die Aktion "Step0_active" unterhalb der POU im Projekexplorer eingefügt und öffnet die Aktion im Editor.

In der Schrittaktion Step0_active programmieren Sie nun Anweisungen, die ausgeführt werden sollen, solange der Schritt aktiv ist.

3. Schließen Sie den Editor von "Step0_active".

Der Schritt **Step0** ist nun mit einem schwarzen Dreieck in der rechten oberen Ecke gekennzeichnet.

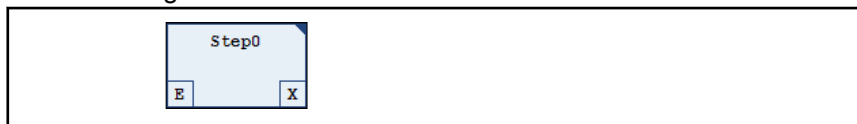


Abb. 9-9:

Sie können die Aktion in den Eigenschaften des Schritts unter **Schrittaktion** definieren. Dort können Sie bei Bedarf auch eine andere Aktion auswählen.

Siehe auch

- ["Schrittaktionen" auf Seite 265](#)

Hinzufügen einer alternativen Verzweigung

1. Selektieren Sie den Schritt **Step1**.
2. Wählen Sie den Befehl **SFC ► Verzweigung rechts einfügen**.

IndraLogic fügt den Schritt **Step2** rechts von **Step1** ein. Die Schritte sind als parallele Verzweigung mit einer Doppellinie verbunden.

3. Selektieren Sie eine der beiden Doppellinien.
Die Doppellinie wird rot gekennzeichnet.

4. Wählen Sie den Befehl **SFC ► Alternativ**

IndraLogic wandelt die Verzweigung in eine alternative Verzweigung um. Die Doppellinie ändert sich zu einer einfachen Linie.

Sie können eine alternative Verzweigung durch den Befehl **SFC ► Parallel** in eine parallele Verzweigung umwandeln.

Siehe auch

- [Kap. "AS-Element 'Verzweigung'" auf Seite 267](#)
- [Kap. "Befehl 'Parallel'" auf Seite 684](#)
- [Kap. "Befehl 'Verzweigung einfügen'" auf Seite 684](#)
- [Kap. "Befehl 'Verzweigung rechts einfügen'" auf Seite 684](#)

Hinzufügen eines Sprungs

1. Selektieren Sie den Schritt **Step2**.
2. Wählen Sie den Befehl **SFC ► Sprung danach einfügen**.
IndraLogic fügt nach dem Schritt **Step2** den Sprung **Step** ein.
3. Klicken Sie auf das Sprungziel "Step" des Sprungs
Sie können nun das Sprungziel manuell eingeben oder über die Eingabehilfe auswählen. Wählen Sie "Step0".

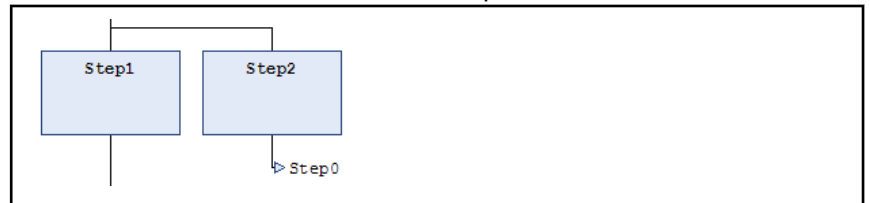


Abb. 9-10:

Siehe auch

- [Kap. "AS-Element 'Sprung'" auf Seite 269](#)
- [Kap. "Befehl 'Sprung einfügen'" auf Seite 687](#)
- [Kap. "Befehl 'Sprung danach einfügen'" auf Seite 687](#)

Hinzufügen eines Makros

1. Selektieren Sie den Schritt **Step1**.
2. Wählen Sie den Befehl **SFC ► Makro danach einfügen**.
IndraLogic fügt nach dem Schritt **Step1** das Makro **Macro0** ein.
3. Doppelklicken Sie auf das Element **Macro0**.
Das Makro öffnet sich im Implementierungsteil des Editors. In der Kopfzeile steht der Name "Macro0".
4. Wählen Sie den Befehl **SFC ► Schritt-Transition einfügen**.
IndraLogic fügt eine Schritt-Transition-Kombination ein.
5. Wählen Sie den Befehl **SFC ► Makro verlassen**.
Der Implementierungsteil zeigt wieder das Hauptdiagramm.

Siehe auch

- [Kap. "AS-Element 'Makro'" auf Seite 269](#)
- [Kap. "Befehl 'Makro einfügen'" auf Seite 687](#)
- [Kap. "Befehl 'Makro danach einfügen'" auf Seite 688](#)

Hinzufügen einer Assoziation

1. Selektieren Sie den Schritt **Step2**.
2. Wählen Sie den Befehl **SFC ► Assoziation einfügen**.
IndraLogic fügt rechts neben dem Schritt **Step2** eine Assoziation hinzu.

3. Klicken Sie in das linke Feld der Assoziation zur Auswahl des Qualifizierers.

Sie können den Qualifizierer manuell eingeben oder über die Eingabehilfe [...] auswählen. Wählen Sie "P".

4. Klicken Sie in das rechte Feld der Assoziation zur Auswahl des Aktion.

Sie können die Aktion manuell eingeben oder über die Eingabehilfe [...] auswählen.

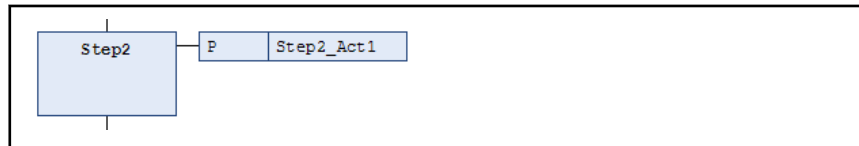


Abb. 9-11:

Siehe auch

- ["IEC-Aktionen" auf Seite 265](#)
- [Kap. "Qualifizierer für Aktionen in AS" auf Seite 256](#)
- [Kap. "Befehl 'Aktionsassoziation einfügen'" auf Seite 686](#)
- [Kap. "Befehl 'Aktionsassoziation danach einfügen'" auf Seite 687](#)

Siehe auch

- [Kap. "AS-Editor" auf Seite 253](#)

Abfragedialog zum Wählen des Duplizierungsmodus

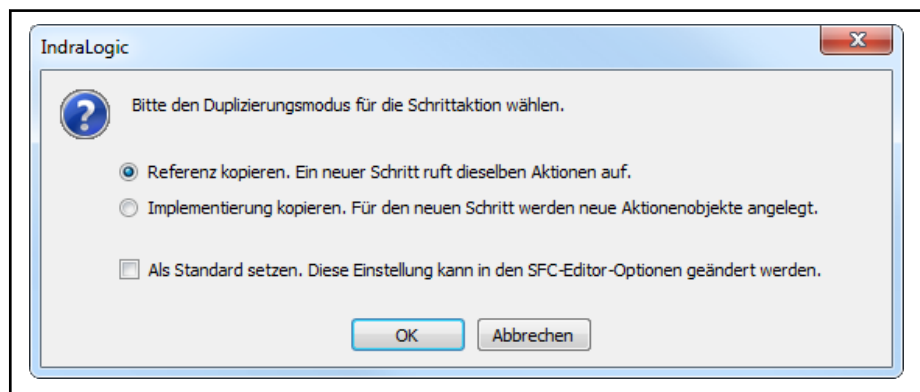


Abb. 9-12:

Optionen:

- *"Referenz kopieren. Ein neuer Schritt ruft dieselben Aktionen auf":* Wenn der Schritt im AS kopiert wird, wird die Verknüpfung zu der/den Schrittktionen mit kopiert. Die voneinander kopierten Schritte werden alle dieselben Aktionen aufrufen.
- *"Implementierung kopieren. Für den neuen Schritt werden neue Aktionsobjekte angelegt":* Dies bedeutet eine "Einbettung" der Schrittktionen für die kopierten Schritte. Die neu erzeugten Aktionsobjekte erscheinen standardmäßig unter dem AS-Baustein im Knoten "Application" oder im Ordner "Allgemeiner Bausteinteil". Diese Objekte enthalten zunächst eine Kopie des ursprünglichen Implementierungscodes der jeweiligen Aktion.

"Als Standard setzen": Die selektierte Option (Standardeinfügemethode) wird beim Hinzufügen einer neuen Eingangs-/Ausgangsaktion standardmäßig gesetzt, der Abfragedialog erscheint nicht mehr.

Die Standardeinfügemethode kann in den IndraWorks Optionen geändert werden: **Extras ▶ Optionen ▶ IndraLogic 2G ▶ AS-Editor**

9.4 Funktionsbaustein, Funktion oder Methode mit externer Implementierung aufrufen

Ein Laufzeitsystem kann die Implementierung eines Funktionsbausteins, einer Funktion oder einer Methode enthalten, zum Beispiel aus einer Bibliothek. Wenn Sie dafür in Ihrer Applikation eine gleichnamige POU mit der Eigenschaft **Externe Implementierung** ohne Implementierung erstellen, können Sie die bereits bestehende Implementierung ausführen.



Dabei sollten Sie beachten, dass Sie lokale Variablen nur in einem externen Funktionsbaustein deklarieren.

Eine externe Funktion oder Methode darf keine lokale Variable enthalten.

Beim Applikations-Download sucht IndraLogic für jede externe POU die zugehörige Implementierung im Laufzeitsystem und verlinkt sie.

POU mit externer Implementierung erstellen

1. Wählen Sie für Ihre Applikation (oder den Ordner "Allgemeiner Baustein") den Kontextmenüpunkt **Hinzufügen ▶ POU**.
2. Aktivieren Sie **Funktionsbaustein**, oder **Funktion** und geben Sie als Name den Namen der zugehörigen Implementierung des Laufzeitsystems. Beenden Sie den Dialog mit **Fertigstellen**.

Die POU mit dem Namen der Laufzeitsystem-POU ist im Projektexplorer erstellt.

3. Selektieren Sie die POU und aktivieren Sie im Kontextmenü **Eigenschaften**.

Der Dialog öffnet sich.

4. Wählen Sie die Registerkarte **Übersetzen**.
5. Aktivieren Sie die Option **Externe Implementierung (Spätes Binden im Laufzeitsystem)**.

Die POU ist deklariert und Sie können einen Aufruf der POU implementieren.

Methode mit externer Implementierung erstellen

1. Selektieren Sie einen Funktionsbaustein im Projektexplorer.
2. Wählen Sie **Hinzufügen ▶ POU-Methode** und geben Sie als Name den Namen der zugehörigen Implementierung des Laufzeitsystems an. Beenden Sie den Dialog mit **Fertigstellen**.

Die Methode ist erstellt.

3. Selektieren Sie die Methode und aktivieren Sie im Kontextmenü **Eigenschaften**.

Der Dialog öffnet sich.

4. Wählen Sie die Registerkarte **Übersetzen**.
5. Aktivieren Sie die Option **Externe Implementierung (Spätes Verlinken im Laufzeitsystem)**.

Die Methode ist deklariert und Sie können einen Aufruf der Methode implementieren.

9.5 Eingabehilfen nutzen

IndraLogic bietet Funktionalitäten und Assistenten, die Ihnen beim Programmieren die Codeeingabe erleichtern.

Eingabehilfe Der Dialog bietet Ihnen alle Programmierelemente an, die Sie an der aktuellen Cursorposition einfügen können. Sie öffnen den Dialog **Eingabehilfe** durch Auswählen des Befehls **Bearbeiten ► Eingabehilfe** oder durch das Tastaturkürzel <F2>.

Siehe auch

- ["Dialog 'Eingabehilfe- Registerkarte 'Kategorien'" auf Seite 625](#)

Dialog 'Variable deklarieren' Ein Dialog unterstützt Sie bei der Deklaration von Variablen.

Siehe auch

- [Kap. "Befehl 'Variable deklarieren'" auf Seite 625](#)

"Komponenten auflisten" Die Funktion "Komponenten auflisten" ist eine Eingabeunterstützung im Texteditor und erleichtert die Eingabe von gültigen Bezeichnern.

Sie aktivieren die Funktion folgendermaßen:

Wählen Sie den Befehl **Extras ► Optionen ► IndraLogic 2G**, und anschließend die Kategorie **Intelligentes Kodieren**. Aktivieren Sie die Option **Komponenten auflisten, nachdem ein Punkt (.) eingegeben wurde**.

- Wenn Sie anstelle einer globalen Variablen nur einen Punkt . eingeben, erscheint eine Auswahlliste mit allen verfügbaren globalen Variablen. Durch Doppelklicken auf eine Variable der Auswahlliste oder Drücken der Taste <Eingabe> fügen Sie die ausgewählte Variable hinter dem Punkt ein.
- Wenn Sie anstelle einer globalen Variablen oder hinter einer Funktionsbaustein-Instanzvariablen oder einer Strukturvariablen einen Punkt eingeben, bietet IndraLogic in einer Auswahlliste entsprechend alle globalen Variablen, alle Ein- und Ausgabevariablen des Funktionsbausteins oder alle Strukturkomponenten an.

Durch Doppelklicken auf eine Variable der Auswahlliste oder Drücken der Taste <Eingabe> fügen Sie die ausgewählte Variable hinter dem Punkt ein.

Hinweis: Wenn Sie zusätzlich die lokalen Variablen von Funktionsbaustein-Instanzen erhalten wollen, aktivieren Sie in den IndraLogic-Optionen für Intelligentes Kodieren auch die Option **Alle Variablen einer Instanz in der Eingabehilfe anzeigen**.

- Wenn bereits ein Komponentenzugriff (mit Punkt) für eine Auswahlliste erfolgt ist, wird beim nächsten Komponentenzugriff der zuletzt selektierte Eintrag vorausgewählt.
- Wenn Sie eine beliebige Zeichenfolge eingeben und anschließend <Strg> + <Leertaste> drücken, erscheint eine Auswahlliste mit allen verfügbaren POUs und globalen Variablen. Das erste Element dieser Liste, das mit der vorher eingegebenen Zeichenfolge beginnt, wird automatisch ausgewählt und Sie können es über einen Doppelklick oder über die Taste <Eingabe> im Editor einfügen.

Übereinstimmungen mit der eingegeben Zeichenfolge werden in der Auswahlliste gelb hervorgehoben. Wenn die eingegebene Zeichenfolge geändert wird, wird die angezeigte Auswahlliste aktualisiert.

- Im ST-Editor können Sie die angezeigte Auswahlliste nach Gültigkeitsbereiche filtern:
Abhängig von der jeweils angezeigten Auswahlliste können Sie mit Hilfe der Tasten [Pfeil nach rechts] und [Pfeil nach links] zwischen folgenden Auswahllisten wechseln:
 - "Alle Einträge"
 - "Schlüsselwörter"
 - "Globale Deklarationen"
 - "Lokale Deklarationen"
- Wenn Sie einen Funktionsbaustein, eine Methode oder eine Funktion aufrufen und dazu die öffnende runde Klammer für das Eintragen der POU-Parameter eingeben, zeigt IndraLogic einen Tooltip an. Dieser Tooltip enthält Informationen zu den Parametern, so wie sie innerhalb der POU deklariert sind. Der Tooltip bleibt solange sichtbar, bis Sie ihn mit einem Mausklick oder durch Setzen des Fokus außerhalb der aktuellen Ansicht schließen. Wenn Sie den Tooltip versehentlich schließen, können Sie ihn mit <Strg> + <Umschalttaste> + <Leertaste> wieder öffnen.



Mit dem Pragmaattribut 'hide' können Sie Variablen aus der Funktion "Komponenten auflisten" ausschließen.

Beispiel:

Eingabe einer Strukturvariablen:

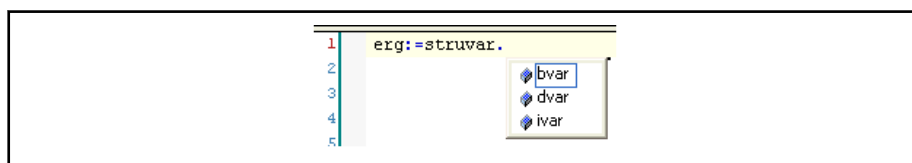


Abb. 9-13:

Aufruf eines Funktionsbausteins:

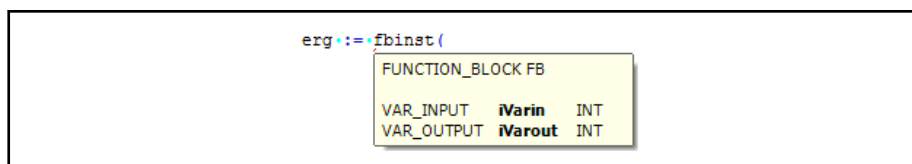


Abb. 9-14:

Siehe auch

- [Kap. "Dialog 'Optionen- 'Intelligentes Codieren'" auf Seite 775](#)
- [Kap. "Attribut 'hide'" auf Seite 414](#)

Kurzformmodus

Der Kurzformmodus ermöglicht die Eingabe von Kurzformen für die Variablendeklaration im Deklarationseditor und in den Texteditoren, in denen Variablendeklarationen möglich sind. Sie aktivieren diesen Modus, indem Sie eine Deklarationszeile mit der Tastenkombination <Strg> + <Eingabe> beenden.

IndraLogic unterstützt folgende Kurzformen:

- 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 oder BOOL ergibt BOOL
 - I oder INT ergibt INT
 - R oder REAL ergibt REAL
 - S oder STRING ergibt STRING
- Wenn durch diese Regeln kein Datentyp festgelegt wird, ist der Datentyp automatisch `BOOL` und der letzte Bezeichner wird nicht als Datentyp benutzt (siehe Beispiel 1).
- Jede eingegebene Konstante wird, je nach Typ der Deklaration, zu einer Initialisierung oder einer Stringlängenangabe (siehe Beispiele 2 und 3)
- Eine Adresse (wie in `%MD12`) wird automatisch um das AT-Attribut erweitert (siehe Beispiel 4)
- Ein Text nach einem Strichpunkt `;` wird zu einem Kommentar (siehe Beispiel 3)
- Alle anderen Zeichen in der Zeile werden ignoriert (siehe Ausrufezeichen in Beispiel 5)

Beispiel:

Beispiel	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;

Siehe auch

- [Kap. 9 "Applikation programmieren" auf Seite 85](#)

Smart-Tag-Funktionen

Smart Tags erleichtern das Erstellen von Programmcode, in dem sie direkt beim Programmierelement geeignete Befehle zur Auswahl bereitstellen. Wenn Sie den Cursor auf ein Programmierelement setzen, für das eine

Smart-Tag-Funktion zur Verfügung steht, erscheint das Symbol . Bei ei-

nem Klick auf  werden die Befehle angezeigt, die Sie auswählen können.

Verfügbare Smart Tags:

- Bei nicht deklarierten Variablen im Implementierungsteil im ST-Editor stellt die Smart-Tag-Funktion den Befehl "Variable deklarieren" zur Verfügung.

Siehe auch [Kap. "Befehl 'Variable deklarieren'" auf Seite 625](#)

9.6 Pragmas verwenden

Was ist ein Pragma in IndraLogic?

Ein Pragma ist eine spezielle Anweisung im Quellcode einer Applikation. Diese Anweisung beeinflusst die Eigenschaften einer oder mehrerer Variablen bezüglich der Vorkompilierung oder der Kompilierung (Codegenerierung).

Es gibt Pragmas für unterschiedliche Effekte:

Initialisierung einer Variablen, Monitoring einer Variablen, Hinzufügen einer Variablen zu einer Parameterliste oder zur Symbolkonfiguration, Erzwingen von Meldungsoutputs während des Übersetzungsvorgangs, Verhalten einer Variablen unter bestimmten Bedingungen etc.



Die Groß-/Kleinschreibung muss eingehalten werden!

Wie und wo fügen Sie ein Pragma ein

Eine Pragma-Anweisung wird in geschweiften Klammern angegeben.

Innerhalb der Anweisung verwendete Attribute und Texte werden in Hochkommas eingeschlossen.

Die öffnende Klammer darf unmittelbar auf einen Variablennamen folgen. Öffnende und schließende Klammer müssen sich in derselben Zeile befinden.

Beispiele

```
{warning 'This is not allowed'}
{attribute 'obsolete' := 'datatype fbl not valid!'}
```

Mögliche Einfügepositionen:

- Im Deklarationsteil einer POU
Im textuellen Deklarationseditor geben Sie Pragmas direkt als Zeile ein.
Im tabellarischen Editor geben Sie Pragmas, die oberhalb der ersten Deklarationszeile stehen müssen, im Dialog **Deklarationskopf editieren / Attribute** ein.
- In einer globalen Variablenliste
- Im Implementierungsteil einer POU in einer separaten Zeile oder in einer Zeile zusammen mit Codetext
FBD/LD/AWL: Pragmas in Netzwerken des FUP/KOP/AWL-Editors werden wie eine Sprungmarke eingegeben:
Wählen Sie dazu den Befehl **FUP/KOP/AWL ► Sprungmarke einfügen** und ersetzen dann den Standardtext **Label:** im Textfeld der Marke durch die entsprechende Pragma-Anweisung.
Wenn Sie ein Pragma zusätzlich zu einer Sprungmarke verwenden wollen, tragen Sie zunächst das Pragma und dann die Sprungmarke ein.



Pragmas in IndraLogic sind keine Eins-zu-Eins-Implementierungen der C-Präprozessor-Direktiven. Sie müssen ein Pragma wie eine normale Anweisung positionieren. Sie dürfen es nicht innerhalb eines Ausdrucks verwenden.

Falsche und richtige Positionierung eines bedingten Pragmas

Falsch:

```
{IF defined(abc)}
IF x = abc THEN
{ELSE}
IF x = 12 THEN
{END_IF}
y := {IF defined(cde)} 12; {ELSE} 13; {END_IF}
END_IF
```

Richtig:

```
{IF defined(abc)}
IF x = abc THEN
```

```

{ IF defined(cde) }
  Y := 12;
{ ELSE }
  Y := 13;
{ END_IF }
END_IF
{ ELSE }
IF x = 12 THEN
{ IF defined(cde) }
  Y := 12;
{ ELSE }
  Y := 13;
{ END_IF }
END_IF
{ END_IF }

```



Im Dialog **Eigenschaften**, Kategorie **Übersetzen** können Sie Defines angeben, die in Pragmas abgefragt werden können.

Wirkungsbereich:

Abhängig vom Typ und Inhalt eines Pragmas wirkt ein Pragma auf Folgendes:

- die nachfolgenden Deklarationen
- genau auf die nachfolgende Anweisung
- auf alle nachfolgenden Anweisungen bis es mit einem entsprechenden Pragma wieder aufgehoben wird
- auf alle nachfolgenden Anweisungen, bis dasselbe Pragma mit anderen Parametern ausgeführt, oder das Ende des Codes erreicht wird.

"Code" in diesem Kontext heißt: Deklarationsteil, Implementierungsteil, Globale Variablenliste, Typdeklaration.

Somit wirkt ein Pragma, das alleine in der ersten Zeile des Deklarationsteils steht und nicht durch ein weiteres abgelöst oder aufgehoben wird, auf das gesamte Objekt.

Welche Arten von Pragmas gibt es?

Die IndraLogic-Pragmas sind in folgende Kategorien aufgeteilt:

- **Attributpragmas:** Beeinflussung der Kompilierung und der Vorkompilierung
- **Meldungspragmas:** Ausgabe von benutzerdefinierten Meldungen während des Übersetzungsvorgangs
- **Bedingte Pragmas:** Beeinflussung der Codegenerierung
- **Benutzerdefinierte Pragmas**

Siehe auch

- [Kap. 9.2.2 "Dialog "Variable deklarierenverwenden" auf Seite 88](#)
- [Kap. 17.6.2 "Attribut-Pragmas" auf Seite 403](#)
- [Kap. 17.6.3 "Bedingte Pragmas" auf Seite 439](#)
- [Kap. 17.6.1 "Meldungspragmas" auf Seite 402](#)

9.7 Bibliotheksbausteine verwenden

Bibliotheken sind Sammlungen von Objekten, die Sie in Ihre Applikation einbinden können. Die in Bibliotheken enthaltenen Objekte können Sie im Projekt genau so verwenden wie Objekte, die Sie direkt im Projekt definiert haben.

Bibliotheken können enthalten:

- POUs wie Funktionsbausteine oder Funktionen

- Schnittstellen (Interfaces) und ihre Methoden und Eigenschaften (Properties)
- Datentypen wie Enumerationen, Strukturen, Aliases, Unions, Felder
- Globale Variablen, Konstanten, Parameterlisten
- Textlisten, Bildersammlungen, Visualisierungen, Visualisierungselemente
- Externe Dateien (beispielsweise Dokumentationen)

Die Verwaltung der Bibliotheken in einem Projekt erfolgt im Bibliotheksverwalter. Die vorherige Installation der Bibliothek auf dem System führen Sie über den Dialog des Bibliotheks-Repositories durch.

Siehe auch

- [Kap. 16 "Bibliotheken erzeugen und verwalten" auf Seite 235](#)

Verwenden eines Bibliotheksbausteins

Die nachfolgende Anweisung beschreibt, wie Sie den Zählerbaustein `CTUD` aus der Bibliothek `Standard` in Ihr Programm einfügen.

1. Öffnen Sie eine POU im Editor und setzen Sie den Cursor in den Deklarationsteil.
2. Geben Sie einen Namen für die Funktionsbausteininstanz gefolgt von einem Doppelpunkt ein (zum Beispiel `iCounter1:`).
3. Öffnen Sie mit <F2> die Eingabehilfe.
4. Selektieren Sie in der Kategorie **Strukturierte Typen** den Funktionsbaustein `CTUD` aus der Bibliothek `Standard` (Unterordner Counter).

Aktivieren Sie die Option **Mit Namensraum-Präfix einfügen**.

5. Schließen Sie den Dialog mit **OK**.

Der Funktionsbaustein wird mit Namensraum-Präfix in den Deklarationsteil eingefügt: `iCounter1:Standard.CTUD`.

6. Zeile durch ; abschließen.

Siehe auch

- [Kap. 16 "Bibliotheken erzeugen und verwalten" auf Seite 235](#)

9.8 Text in einer Textliste verwalten

Textlisten dienen dazu, Texte für eine Visualisierung in mehreren Sprachen bereitzustellen.

Sie können die Texte im Unicode-Format eingeben, so dass Ihnen sämtliche Sprachen und Schriftzeichen zur Verfügung stehen.

Sie können Textlisten exportieren und importieren und damit die Texte außerhalb des aktuellen Projekts übersetzen.

IndraLogic unterscheidet zwischen statischem Text, der im Objekt **GlobalTextList** verwaltet wird, und dynamischem Text, der in Objekten des Typs **Textliste** verwaltet wird.

Das Objekt **GlobalTextList** wird automatisch (beim Hinzufügen eines Visualisierungsobjektes im Projekt) im Ordner "Allgemeiner Bausteinteil" angelegt.

Die übrigen Textlistenobjekte können entweder in den Ordner "Allgemeiner Bausteinteil" projektglobal oder unterhalb der jeweiligen Applikation für eine Steuerung **Hinzugefügt** werden.

Statische Texte sind Texte innerhalb der Visualisierung, die zur Laufzeit nur ihre Sprache ändern können. Die Text-ID bleibt dabei konstant.



Die Liste der **Statischen Texte** füllt sich automatisch in den Spalten "ID" und "Standard/Default" wenn Texte im Visualisierungseditor eingegeben werden.

Die sprachabhängigen Spalten (z. B. "en_US") können manuell in der "GlobalTextList" übersetzt werden, oder extern, nach Export, und danach wieder importiert werden.

Dynamische Texte können Sie über eine IEC-Variable steuern, die die Text-ID enthält. Damit können Sie in einem Visualisierungselement zur Laufzeit variierenden Text anzeigen lassen. Zum Beispiel können Sie ein Textfeld so konfigurieren, dass es einen Fehlertext zu einer Fehlernummer ausgibt.



Die Liste der **Dynamischen Texte** muss durch den Anwender komplett erstellt werden (ID, Standard/Default).

Die sprachabhängigen Spalten (z. B. "en_US") können manuell in der Textliste übersetzt werden, oder extern, nach Export, und danach wieder importiert werden.

Beide Textlistentypen enthalten eine Tabelle mit Texteinträgen.

Ein Eintrag besteht aus einer ID zur Identifizierung, dem Ausgangstext und dessen Übersetzungen.

In einer Textliste oder einer globalen Textliste können Sie einen Ausgangstext in beliebig vielen Sprachen übersetzen.

Die Übersetzungen sind die Basis für die Sprachauswahl und die Sprachumschaltung in Visualisierungen.



Die Namen der sprachabhängigen Spalten sollten in allen Tabellen gleich gewählt werden.

GlobalTextList füllen

1. Öffnen Sie Ihr Visualisierungsobjekt und tragen z. B. in Textfelder die gewünschten Texte ein.
2. Öffnen Sie die **GlobalTextList** im Ordner "Allgemeiner Baustein".

Die **GlobalTextList** enthält die automatisch gespeicherten Texte.

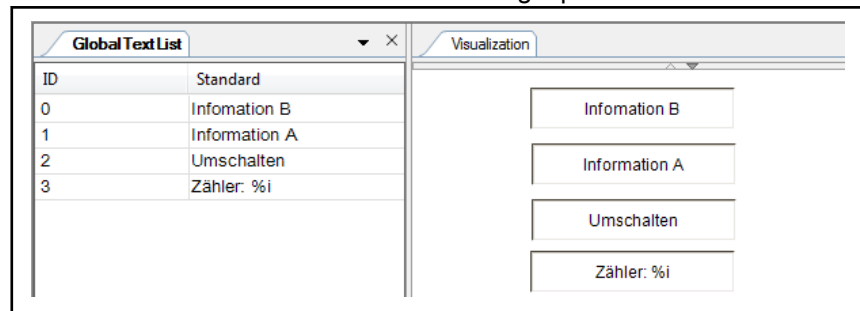


Abb. 9-15: Automatische Erzeugung der GlobalTextList (Beispiel in deutscher Sprache)

Textliste "Text_list_A" anlegen und füllen

1. **Hinzufügen** einer Textliste im Ordner "Allgemeiner Baustein" oder unterhalb einer Applikation. Nennen Sie die Liste z. B. "Text_list_A".
2. Tragen Sie die IDs und die gewünschten Texte ein.

Die Abbildung zeigt die manuell eingegebenen IDs und Texte der Spalte "Standard/Default"

Text_list_A	
ID	Standard
A	Information A
B	Information B: OK
C	Information C
D	Informaiton D
E	Information E
F	Information F

Abb. 9-16: Manuelle Erzeugung von "Text_list_A" (Beispiel in deutscher Sprache)

Sprache hinzufügen und übersetzen

Voraussetzung: Ein Projekt mit Textliste oder globaler Textliste ist geöffnet.

1. Doppelklicken Sie im Projektexplorer auf ein Objekt des Typs **Textliste** oder **GlobalTextList**.

Das Menü **Textliste** erscheint in der Menüleiste und die Textliste öffnet im Editor.

2. Wählen Sie **VI-Logic Textliste ► Sprache einfügen**
3. Geben Sie einen Namen für die Sprache ein, zum Beispiel `en-US`. Beenden Sie den Dialog mit **OK**.
Eine Spalte mit der Überschrift `en-US` erscheint.
4. Geben Sie in der Spalte die Übersetzung des Ausgangstextes ein.



Mit dem Befehl "Sprache umbenennen" aus dem Kontextmenü der Textliste können Sie den Namen einer Sprache in der Tabelle korrigieren.

Textlisten exportieren

Voraussetzung: Eine Textliste oder die globale Textliste eines Projekt ist geöffnet.

1. Doppelklicken Sie auf das Objekt **GlobalTextList** oder auf ein Objekt des Typs **Textliste**.

Das Objekt öffnet sich.

2. Wählen Sie **VI-Logic Textliste ► Import/Export Textlisten**

Der Dialog **Import/Export** öffnet sich.

3. Klicken Sie bei **Exportdatei auswählen** auf und wählen Sie dort Verzeichnis und Dateiname, zum Beispiel `Text_lists_exported`
4. Aktivieren Sie die Option **Exportieren**.
5. Beenden Sie den Dialog **Import/Export** mit **OK**.

IndraLogic exportiert in die Datei die Textlisteneinträge aller Textlisten des Projekts. Die Tabelle enthält eine Spalte mit dem Textlistenamen.

	A	B	C	D
1	TextList	Id	Default	en_US
2	GlobalTextList		Information B	Information B_en
3	GlobalTextList		Information A	Information A_en
4	GlobalTextList		Umschalten	Switch
5	GlobalTextList		Zähler: %i	Counter : %i
6	Text_list_A	A	Information A	Information A_en
7	Text_list_A	B	Information B: OK	Information B_en: OK
8	Text_list_A	C	Information C	Information C_en
9	Text_list_A	D	Information D	Information D_en
10	Text_list_A	E	Information E	Information E_en
11	Text_list_A	F	Information F	Information F_en

Abb. 9-17: Exportdatei "Text_lists_exported.csv"

Exportdatei "Text_lists_exported.csv"

```

TextList;Id;Default;en_US
GlobalTextList;;Information B;Information B_en
GlobalTextList;;Information A;Information A_en
GlobalTextList;;Umschalten;Switch
GlobalTextList;;Zähler: %i;Counter : %i
Text_list_A;A;Information A;Information A_en
Text_list_A;B;Information B: OK;Information B_en: OK
Text_list_A;C;Information C;Information C_en
Text_list_A;D;Information D;Information D_en
Text_list_A;E;Information E;Information E_en
Text_list_A;F;Information F;Information F_en

```

Datei mit Textlisteneinträgen importieren

Eine importierbare Datei ist eine Datei mit Format `.csv`. Die erste Zeile ist eine Kopfzeile, wie zum Beispiel `TextList Id Default en_US`.

Die weiteren Zeilen enthalten Textlisteneinträge. Sie erhalten so eine Datei, wenn Sie die Textlisten des Projekts in eine Datei exportierten. Sie können dort, außerhalb von IndraLogic, die Textlisteneinträge bearbeiten und die Datei anschließend importieren.

IndraLogic behandelt beim Import die Textlisteneinträge für die `GlobalTextList` und für dynamische Textlisten aber unterschiedlich.

GlobalTextList

- Bei einer unbekannten ID legt IndraLogic keine neuen Textlisteneinträge an.
- IndraLogic ignoriert Änderungen, die die ID oder den Ausgangstext betreffen.
- IndraLogic übernimmt Änderungen der Übersetzungen

Textliste

- Bei einer neuen ID ergänzt IndraLogic die zugehörige Textliste um einen Textlisteneintrag
- Bei einer bestehenden ID, die im Ausgangstext nicht übereinstimmen, wird der Ausgangstext der Textliste mit dem Ausgangstext der Datei überschrieben.
- IndraLogic übernimmt Änderungen der Übersetzungen.

Datei importieren

Voraussetzung: Ein Projekt mit Textliste oder globaler Textliste ist geöffnet.

1. Doppelklicken Sie auf das Objekt **GlobalTextList** oder auf ein Objekt des Typs **Textliste**.
Das Objekt öffnet sich.
2. Wählen Sie **VI-Logic Textliste ► Import/Export Textliste**

Der Dialog **Import/Export** öffnet sich.

3. Klicken Sie in **Datei für Vergleich oder Import auswählen** auf  und wählen Sie dort Verzeichnis und Dateiname, zum Beispiel `Text_lists_corrected.csv`.
 4. Aktivieren Sie die Option **Importieren**.
 5. Beenden Sie den Dialog mit **OK**.
- IndraLogic importiert die Textlisteneinträge der Datei in die zugehörigen Textlisten.

Beispiel

Inhalt der Datei `Text_lists_corrected.csv`

```
TextList;Id;Default;en_US
GlobalTextList;;Information B;Information B_en
GlobalTextList;;Information A;Information A_en
GlobalTextList;;Umschalten;Switch
GlobalTextList;;Zähler: %i;Counter : %i
Text_list_A;A;Information A;Information A2_en
Text_list_A;B;Information B: OK;Information B_en: OK
Text_list_A;C;Information C;Information C_en
Text_list_A;D;Information D;Information D_en
Text_list_A;E;Information E;Information E_en
Text_list_A;F;Information F;Information F_en
Text_list_A;G;Information G;Information G_en
```

Dieser Inhalt wird so im Projekt in den gleichnamigen Textlisten übernommen.

Siehe auch

- [Kap. "Befehl 'Import/Export Textlisten'" auf Seite 729](#)

Textlisten mit Datei vergleichen und Unterschied exportieren

Voraussetzung: Ein Projekt mit Textliste oder globaler Textliste ist geöffnet.

1. Doppelklicken Sie auf das Objekt **GlobalTextList** oder auf ein Objekt des Typs **Textliste**.
Das Objekt öffnet sich.
2. Wählen Sie im Kontextmenü **VI-Logic Textliste ► Import/Export Textliste**.
Der Dialog **Import/Export** öffnet sich.
3. Klicken Sie in **Datei für Vergleich oder Import auswählen** auf  und wählen Sie dort Verzeichnis und Dateiname der Vergleichsdatei aus, zum Beispiel `Text_lists_corrected.csv`.
4. Klicken Sie bei **Exportdatei auswählen** auf  und wählen Sie dort Verzeichnis und Dateiname der Datei aus, die das Vergleichsergebnis enthalten wird.
5. Aktivieren Sie die Option **Nur Textunterschiede exportieren**.
6. Beenden Sie den Dialog mit **OK**.

IndraLogic liest die Importdatei und vergleicht die Textlisteneinträge, die die gleiche ID haben. Wenn sie nicht übereinstimmen, schreibt IndraLogic den Textlisteneinträge der Textliste in die Exportdatei. Bei der globalen Textliste vergleicht IndraLogic bei gleichen Ausgangstexten die Übersetzungen. Wenn sie nicht übereinstimmen, schreibt IndraLogic den Textlisteneinträge in die Exportdatei.

Siehe auch

- [Kap. "Befehl 'Sprache einfügen'" auf Seite 727](#)
- [Kap. "Befehl 'Import/Export Textlisten'" auf Seite 729](#)
- [Kap. "Befehl 'Sprache entfernen'" auf Seite 730](#)
- [Kap. 18.1.23 "Objekt 'Textliste'" auf Seite 588](#)

9.8.1 Statischen Text in einer globalen Textliste verwalten

Die globale Textliste ist die zentrale Stelle im Projekt für Texte, die in der Visualisierung ausgegeben werden (Ordner "Allgemeiner Baustein").

Wenn Sie in der Visualisierung in einem Element einen Text erstmals konfigurieren, erstellt IndraLogic die globale Textliste.

Wenn Sie einen weiteren Text verfassen, ergänzt IndraLogic die Tabelle. Sie enthält somit automatisch alle Texte, die Sie in den Visualisierungen des Projekts verfasst haben.

Die IDs vergibt IndraLogic als fortlaufende ganzen Zahlen beginnend mit "0".

Sie können die globale Textliste gegen die statischen Texte der Visualisierungen prüfen, aktualisieren und abgleichen.

Es ist nicht möglich, den Ausgangstext oder die ID direkt in der Tabelle zu bearbeiten. Sie können einen Ausgangstext durch einen anderen ersetzen, indem Sie eine Ersetzungsdatei erstellen und diese importieren. Dafür stehen Ihnen Menübefehle zur Verfügung.

Visualisierungselement mit statischem Text konfigurieren

Ein Text in einer **GlobalTextList** kann eine Formatierungsangabe enthalten.

*Voraussetzung: Ein Projekt mit Visualisierung ist geöffnet. Das Objekt **GlobalTextList** enthält die Texte, die in den Visualisierungen des Projekts definiert sind.*

1. Doppelklicken Sie auf die Visualisierung.
Der Editor öffnet sich.
2. Selektieren Sie ein Element, das über die Eigenschaft **Text** verfügt, zum Beispiel ein **Textfeld**.
3. Geben Sie bei der Eigenschaft **Text** einen Text ein, zum Beispiel `Statische Information A`.
IndraLogic erweitert die globale Textliste in der POUs-Ansicht um den neuen Text.

Globale Textliste prüfen

*Voraussetzung: Ein Projekt mit Visualisierung ist geöffnet. Das Objekt **GlobalTextList** enthält die Texte, die in den Visualisierungen des Projekts definiert sind.*

1. Doppelklicken Sie im Projektexplorer auf Objekt **GlobalTextList**.
Die Tabelle mit den statischen Texten öffnet sich.
2. Wählen Sie **VI-Logic Textliste ► Visualisierungstext-IDs prüfen**
IndraLogic meldet, wenn ein Ausgangstext der Textliste nicht übereinstimmt mit dem statischen Text, der über die ID identifiziert ist. Der Ausgangstext in der Globalen Textliste und der Text in den Visualisierungen, die die gleiche ID haben, stimmen nicht überein.

IDs der Globale Textliste aktualisieren

*Voraussetzung: Ein Projekt mit Visualisierung ist geöffnet. Das Objekt **GlobalTextList** enthält die Texte, die in den Visualisierungen des Projekts definiert sind.*

1. Doppelklicken Sie im Projektextplorer auf das Objekt **GlobalTextList**.
Liste mit den Texteinträgen öffnet sich.
2. Wählen Sie **VI-Logic Textliste ► Visualisierungstext-IDs aktualisieren**
IndraLogic ergänzt die globale Textliste, wenn in den Visualisierungen des Projekts ein Text in der Eigenschaft **Statischer Text** nicht mit einem Ausgangstext übereinstimmt.

Globale Textliste entfernen und mit aktuellen IDs neu erstellen

Voraussetzung: Ein Projekt mit Visualisierung ist geöffnet. Das Objekt "GlobalTextList" enthält die Texte, die in den Visualisierungen des Projekts definiert sind.

1. Selektieren Sie im Ordner "Allgemeiner Baustein" das Objekt "GlobalTextList" und wählen Sie im Kontextmenü den Befehl "Löschen".
⇒ Das Objekt ist entfernt.
2. Öffnen Sie eine Visualisierung.
3. Wählen Sie den Befehl **Visualisierung ► Globale Textliste anlegen**.
⇒ Im Ordner "Allgemeiner Baustein" ist das Objekt "GlobalTextList" neu erstellt. Die globale Textliste enthält die statischen Texte aus den bereits vorhandenen Visualisierungen des Projekts.

IDs aus der Globalen Textliste entfernen

Voraussetzung: Ein Projekt mit Visualisierung ist geöffnet. Das Objekt **GlobalTextList** enthält die Texte, die in den Visualisierungen des Projekts definiert waren.

1. Doppelklicken Sie im Projektextplorer auf Objekt **GlobalTextList**.
Tabelle mit den Texten öffnet sich.
2. Wählen Sie **VI-Logic Textliste ► Nicht verwendete Textlisteneinträge entfernen**
IndraLogic entfernt die Textlisteneinträge, deren IDs in den Visualisierungen des Projekts nicht referenziert sind.

Globale Textliste mit Ersetzungsdatei bearbeiten

Eine Ersetzungsdatei ist eine Datei im Format `.csv`. Die erste Zeile ist eine Kopfzeile: `defaultalt defaultneu REPLACE`. Die weiteren Zeilen enthalten den alten Ausgangstext, den neuen Ausgangstext und dann den Befehl `REPLACE`. Als Trennzeichen sind Tabulator, Komma oder Strichpunkt erlaubt. Eine Mischung der Trennzeichen innerhalb einer Datei ist nicht erlaubt.

Beispiel (Tabulator als Trennzeichen)

defaultalt	defaultneu	REPLACE
Information A	Information A1	REPLACE

Wenn Sie eine Ersetzungsdatei importieren, arbeitet IndraLogic die Ersetzungsdatei zeilenweise ab und führt die spezifizierten Ersetzungen in der **GlobalTextList** durch. Außerdem ersetzt IndraLogic in den Visualisierungen den bisherigen Text durch den Ersatztext. Wenn der Ersatztext bereits als statischer Text vorhanden war, erkennt IndraLogic das, harmonisiert die statischen Texte und belässt nur einen Textlisteneintrag.

Voraussetzung: Ein Projekt mit Textliste oder globaler Textliste ist geöffnet.

1. Doppelklicken Sie auf das Objekt **GlobalTextList**.
Das Objekt öffnet sich.

2. Wählen Sie **VI-Logic Textliste** ► **Import/Export Textliste**
Der Dialog **Import/Export** öffnet sich.
3. Klicken Sie bei **Datei für Vergleich oder Import auswählen** auf  und wählen Sie dort Verzeichnis und Dateiname, zum Beispiel `ReplaceGlobalTextList.csv`.
4. Aktivieren Sie die Option **Ersetzungsdatei importieren**.
5. Beenden Sie den Dialog mit **OK**.
Die Texte in den Textlisten und den Visualisierungen sind ersetzt.

Beispiel

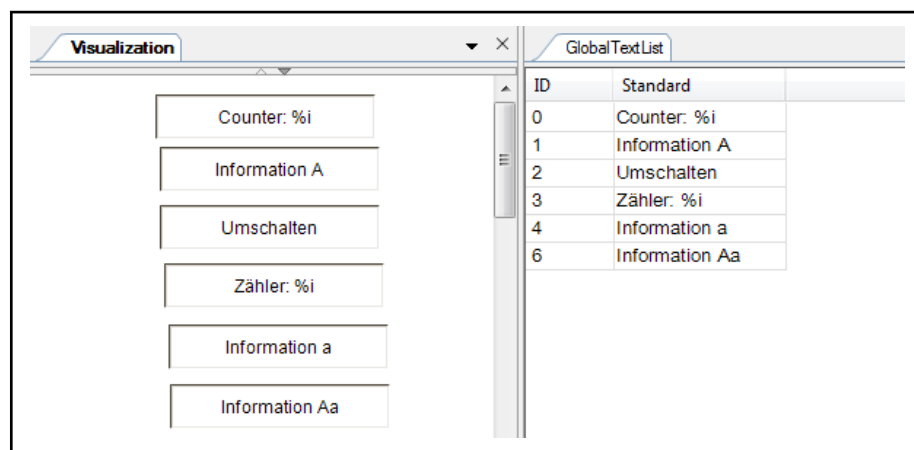


Abb. 9-18: Die globale Textliste enthält die folgenden Ausgangstexte:
Die Ersetzungsdatei enthält folgende Ersetzungen:

```
defaultalt;defaultneu;REPLACE
Counter: %i;Counter1: %i;REPLACE
Zähler: %i;Counter2: %i;REPLACE
Information A;Information A2;REPLACE
Information a;Information A2;REPLACE
Information Aa;Information A2;REPLACE
Umschalten;Switch2;REPLACE
```

IndraLogic erkennt doppelte Textlisteneinträge und entfernt sie.
Die Texte in der Visualisierung sind ausgetauscht.
Die globale Textliste enthält danach folgende Einträge:

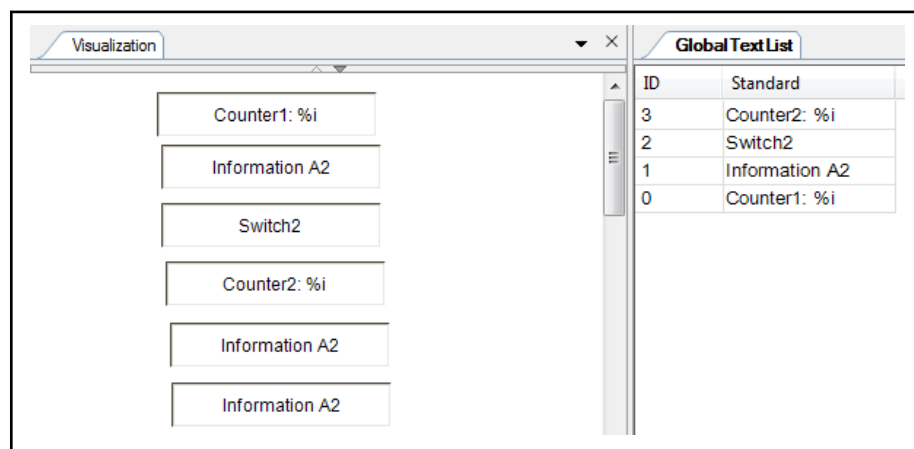


Abb. 9-19: Visualisierung und GlobalTextList nach dem Austausch

Siehe auch

- [Kap. "Befehl 'Globale Textliste anlegen'" auf Seite 727](#)
- [Kap. "Befehl 'Visualisierungs-Text-IDs prüfen'" auf Seite 731](#)
- [Kap. "Befehl 'Visualisierungs-Text-IDs aktualisieren'" auf Seite 731](#)
- [Kap. 18.1.10 "Objekt 'GlobalTextList'" auf Seite 543](#)

9.8.2 Dynamischen Text in einer Textliste verwalten

Sie können in einer "Textliste für dynamischen Text" Texte erstellen und übersetzen, um sie dann in einer Visualisierung dynamisch auszugeben.

Das Objekt des Typs **Textliste** kann global im Ordner "Allgemeiner Baustein" oder unter einer Applikation im Projektextplorer liegen und enthält eine Tabelle mit Textlisteneinträgen, die Sie bearbeiten und erweitern können.

Ein Textlisteneintrag besteht aus einer ID zur Identifizierung, dem Ausgangstext und dessen Übersetzungen. Sie können einer Textliste neue Textlisteneinträge hinzufügen. Dafür stehen Ihnen Menübefehle zur Verfügung.

Textliste für dynamische Textausgabe erstellen

Voraussetzung: Ein Projekt mit Visualisierung ist geöffnet.

1. Selektieren Sie den Ordner "Allgemeiner Baustein" oder im Projektextplorer eine Applikation und wählen Sie den Befehl **Hinzufügen**.
2. Wählen Sie **Textliste**.
3. Geben Sie einen Namen ein, zum Beispiel `Text_list_A`. Beenden Sie den Dialog mit **Fertigstellen**.

Ein Objekt des Typs **Textliste** ist erstellt.

4. Klicken Sie unter Spalte **Standard** und öffnen Sie das Eingabefeld. Geben Sie einen Text ein, zum Beispiel `Information A`.

Der Ausgangstext ist erstellt. Er dient als Schlüssel in der Tabelle und als Ausgangstext für Übersetzungen.

5. Geben Sie in Spalte **ID** eine beliebige Zeichenkette ein, zum Beispiel `A`.

Ein Textlisteneintrag mit Ausgangstext und ID ist definiert. Wenn Sie in einer Visualisierung die Eigenschaft **Dynamische Texte** eines Elements konfigurieren, können Sie nun die Textliste `Text_list_A` auswählen und die ID `A` zuweisen.

6. Doppelklicken Sie in die leere Zeile am Ende der Tabelle unter **Standard/Default** und geben Sie weitere Textlisteneinträge ein.

Text_list_A		
ID	Standard	
A	Information A	
B	Information B	
C	Information: OK	

Abb. 9-20:

Text dynamisch ausgeben

Sie können in einer Visualisierung die dynamische Ausgabe von Texten konfigurieren, die in einer Textliste verfasst wurden, indem Sie die Eigenschaft **Dynamische Texte** eines Elements konfigurieren. Sie können direkt eine Textliste und eine ID zuweisen, aber auch IEC-Variablen, in der Sie programmatisch die Werte setzen.

Voraussetzung: Ein Projekt mit Visualisierung ist geöffnet und eine Textliste ist im Projektextplorer.

1. Öffnen Sie die Textliste, zum Beispiel `Text_list_A`.
2. Doppelklicken Sie auf die Visualisierung.
Der Editor öffnet sich.
3. Ziehen Sie ein Element, zum Beispiel des Typs **Textfeld**, in die Visualisierung.
4. Konfigurieren Sie dessen Eigenschaft **Dynamische Texte**, indem Sie in der Eigenschaft **Textliste** eine auswählen, zum Beispiel `'Text_list_A'` und in **Textindex** eine ID aus der Textliste einfügen, zum Beispiel `'A'`.

Achten Sie auf die Hochkommata.

Sie können auch jeweils eine IEC-Variable des Typs `STRING` für den Textlistenamen und der ID zuweisen.

Die IEC-Variablen ermöglichen programmatisch den Zugriff auf die Texte der Textlisten.

5. Übersetzen Sie die Applikation, laden Sie sie auf die Steuerung und starten Sie sie.

Die Visualisierung gibt im Textfeld den Text aus der Textliste aus: `Information A`.

Siehe auch

- [Kap. 18.1.23 "Objekt 'Textliste'" auf Seite 588](#)

9.9 Bildersammlungen verwenden

Eine Bildersammlung ist eine Tabelle mit Bilddateien. Durch Angabe der ID und zusätzlich des Namens der Bildersammlung kann IndraLogic eine Bilddatei eindeutig referenzieren, wenn Sie sie im Projekt, beispielsweise in einer Visualisierung, verwenden. Ein Projekt kann mehrere Bildersammlungen enthalten. Sie können Bildersammlungen im Projektextplorer unter der Applikation oder im Ordner "Allgemeiner Bausteinteil" anlegen.

In einem Bibliotheksprojekt können Sie eine Bildersammlung über ihre Objekteigenschaften zu einer Symbolbibliothek für die Visualisierung machen.



Wir empfehlen, die Größe einer Bilddatei vor ihrer Einbindung so weit wie möglich zu reduzieren. Dies optimiert die Ladezeit der Visualisierung.

Wenn Sie ein Bildelement per Namen in eine Visualisierung einfügen (Eigenschaft des Elementes 'Bild' oder 'Bildwechsler') legt IndraLogic automatisch eine globale Bildersammlung an. Dabei verwendet IndraLogic den Standardnamen **GlobalImagePool**. Die ID (**Statische ID**) wird dabei von dem Dateinamen des Bildes abgeleitet.

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

- Suchreihenfolge: Wenn Sie ein Bild wählen, das im **GlobalImagePool** verwaltet wird, müssen Sie den Name der Bildersammlung nicht angeben.

Die Suchreihenfolge für Bilddateien entspricht folgender Reihenfolge:

- 1. globaler GlobalImagePool
- 2. Bildersammlungen, die der gerade aktiven Applikation zugewiesen sind
- 3. Bildersammlungen, die neben dem GlobalImagePool im Ordner "Allgemeiner Baustein" liegen
- 4. Bildersammlungen in Bibliotheken
- Eindeutiger Zugriff: Sie können das gewünschte Bild direkt und eindeutig ansprechen, indem Sie der ID den Namen der Bildersammlung voranstellen, gemäß Syntax

<Name der Sammlung>.<Bild ID>.

Siehe auch

- [Kap. 18.1.13 "Objekt 'Bildersammlung'" auf Seite 547](#)

Anlegen einer Bildersammlung

1. Selektieren Sie im Projektextplorer das Objekt **Applikation**. Wählen Sie den Befehl **Hinzufügen ► Bildersammlung**. Es öffnet sich der Dialog **Bildersammlung hinzufügen**.
2. Geben Sie einen Namen für die Bildersammlung ein (zum Beispiel "Images1") und bestätigen Sie den Dialog mit der Schaltfläche **Fertigstellen**. Die Bildersammlung wird zum Projektextplorer hinzugefügt.
3. Öffnen Sie das Objekt Bildersammlung durch Doppelklick.
4. Führen Sie einen Doppelklick auf das Feld **ID** aus und vergeben Sie eine geeignete ID (zum Beispiel "Icon1"). Alternativ können Sie neue Bilder über den Kontextmenübefehl **Bilddatei einfügen** zur Liste hinzufügen.
5. Führen Sie einen Doppelklick auf die Spalte **Dateiname** aus. Klicken Sie auf die Schaltfläche . Es öffnet sich der Dialog **Bildauswahl**.
6. Klicken Sie auf die Schaltfläche  und wählen Sie die Bilddatei aus. Die Bilddatei wird in verkleinerter Darstellung in der Spalte **Bild** angezeigt. Der Name der Datei wird im Feld **Dateiname** angezeigt.

Die Bilddatei kann nun über den Namen `Images1.Icon1` referenziert werden.

Siehe auch

- [Kap. "Befehl 'Bilddatei einfügen'" auf Seite 719](#)

Verwenden einer Bilddatei im Visuelelement 'Bild'

Wenn Sie ein Bildelement in eine Visualisierung einfügen, können Sie den Typ des Bilds festlegen:

- **Statisches Bild:** Geben Sie in der Konfiguration des Elements (Eigenschaft **Statische ID**) die ID der Bilddatei oder den Namen der Bildersammlung + ID ein. Beachten Sie dabei die Bemerkungen zu Suchreihenfolge und Zugriff.
- **Dynamisches Bild:** Geben Sie in der Konfiguration des Elements (Eigenschaft **Bitmap ID variable**) die Variable ein, die die ID der Bilddatei definiert, zum Beispiel `PlcProg.imagevar`. Sie können ein dynamisches

Element im Online-Betrieb in Abhängigkeit von einer Variable austauschen

Siehe auch

- [Kap. "Element 'Bild'" auf Seite 906](#)

Verwenden einer Bilddatei für den Visualisierungshintergrund

In der Hintergrunddefinition einer Visualisierung können Sie ein Bild angeben. Sie können das Bild, wie oben für ein Visualisierungselement beschrieben, durch den Namen der Bildersammlung plus den Dateinamen definieren.

Siehe auch

- [Kap. 20.2.11 "Befehl 'Hintergrund'" auf Seite 821](#)

9.10 Programmatischer Zugriff auf E/As

IndraLogic stellt zur Abbildung von Projektvariablen auf Eingangs-, Ausgangs- und Speicheradressen folgende Funktionalitäten zur Verfügung:

- Zuweisung von Projektvariablen auf Eingangs-, Ausgangs- und Speicheradressen in der Registerkarte **E/A-Abbild** des Editors der Geräte
- Programmatischer Zugriff auf E/As
 - Variablenkonfiguration
 - AT-Deklaration



Wir empfehlen Ihnen, die Abbildung von Projektvariablen auf Eingangs-, Ausgangs- und Speicheradressen in der Registerkarte **E/A-Abbild** des Editors des jeweiligen Geräts vorzunehmen.

Siehe auch

- [Kap. 8.3 "Geräte konfigurieren und E/A Abbild konfigurieren" auf Seite 75](#)

9.10.1 Variablenkonfiguration - VAR_CONFIG

Sie verwenden die "Variablenkonfiguration", um Variablen von Funktionsbausteinen auf das Prozessabbild zu mappen.

Bei der Deklaration im Funktionsbaustein weisen Sie die Variablen den Ein-/Ausgängen von Geräten zu, ohne die vollständige Adresse anzugeben.

Die Angabe der exakten Adressen erfolgt später zentral für alle Funktionsbaustein-Instanzen der Applikation in einer globalen Variablenliste mit **VAR_CONFIG**-Deklarationen. Diese globale Variablenliste mit den **VAR_CONFIG**-Deklarationen wird Variablenkonfiguration genannt.



Änderungen von Variablen, die direkt den E/A-Adressen zugewiesen sind, zeigt IndraLogic unmittelbar im Prozessabbild an. Änderungen von Variablen, die über eine Variablenkonfiguration abgebildet sind, zeigt IndraLogic erst am Ende der zuständigen Task an.

Deklaration der Variablen im Funktionsbaustein:

Bei der Deklaration der Variablen im Funktionsbaustein deklarieren Sie die Variablen zwischen den Schlüsselwörtern **VAR** und **END_VAR**. Dabei ordnen Sie den Variablen unvollständige Adressen zu. Diese unvollständigen Adressen kennzeichnen Sie mit einem *****.

Syntax:

<Bezeichner> AT %<I|Q>*: <Datentyp>;

Beispiel

Es werden 2 lokale E/A-Variablen definiert: die Eingangsvariable `xLocIn` und die Ausgangsvariable `xLocOut`

```
FUNCTION_BLOCK locio
VAR
    xLocIn AT %I*: BOOL := TRUE;
    xLocOut AT %Q*: BOOL;
END_VAR
```

Die endgültige Definition der Adressen erfolgt in der Variablenkonfiguration in einer globalen Variablenliste:

In der globalen Variablenliste, die Sie als Variablenkonfiguration verwenden, geben Sie die Deklarationen der Variablen mit den exakten Adressen zwischen den Schlüsselwörtern `VAR_CONFIG` und `END_VAR` ein.

Die `VAR_CONFIG`-Variablen müssen Sie mit dem vollständigen Instanzpfad angeben, wobei die einzelnen POU- und Instanznamen durch jeweils einen Punkt getrennt sind.

Die Deklaration muss eine Adresse enthalten, deren Klasse (Eingang/Ausgang) mit der Klasse der unvollständig angegebenen Adresse (%I*, %Q*) im Funktionsbaustein übereinstimmt. Der Datentyp muss ebenfalls übereinstimmen.

Syntax:

<instance variable path> AT %<I|Q><location>: <data type>;

Wenn die Instanz des Pfads nicht existiert, erfolgt eine Fehlermeldung. IndraLogic gibt auch einen Fehler aus, wenn keine Adresskonfiguration für eine Variable vorliegt, die Sie mit einer unvollständigen Adresse deklariert haben.

Beispiel

Folgende Verwendung des Funktionsbausteins `locio` des obigen Beispiels liegt in einem Programm vor:

```
PROGRAM PlcProg
VAR
    locioVar1: locio;
    locioVar2: locio;
END_VAR
```

Dann würde eine korrekte Variablenkonfiguration in einer globalen Variablenliste beispielsweise folgendermaßen aussehen:

```
VAR_CONFIG
    PlcProg.locioVar1.xLocIn AT %IX1.0 : BOOL;
    PlcProg.locioVar1.xLocOut AT %QX0.0 : BOOL;
    PlcProg.locioVar2.xLocIn AT %IX1.0 : BOOL;
    PlcProg.locioVar2.xLocOut AT %QX0.3 : BOOL;
END_VAR
```

Siehe auch

- [Kap. 17.2.13 "Konfigurationsvariablen - VAR_CONFIG" auf Seite 311](#)
- [Kap. 9.10.2 "AT-Deklaration" auf Seite 130](#)
- [Kap. 17.4.13 "Adressen" auf Seite 371](#)

Anlegen einer Variablenkonfiguration

Voraussetzungen: Sie haben ein Projekt geöffnet, das eine Steuerungskonfiguration mit einem Feldgerät besitzt. Das Projekt enthält ein Programm, zum Beispiel `PlcProg` und einen Funktionsbaustein, zum Beispiel `func1`. Das

Feldgerät besitzt Ein- und Ausgänge. In den Optionen für den Deklarationsektor ist die textuelle Ansicht ausgewählt.

Im Funktionsbaustein Variablen auf Geräte-E/As mit unvollständigen Adressen zuweisen und anschließend eine Variablenkonfiguration erstellen.

1. Doppelklicken Sie im Projektexplorer auf den Funktionsbaustein, zum Beispiel `func1`.

Der Editor des Bausteins öffnet sich.

2. Geben Sie im Deklarationsteil zwischen den Schlüsselwörtern `VAR` und `END_VAR` ein:

`xLocIn AT %I*: BOOL := TRUE;` und in die nächste Zeile

`XLocOut AT %Q*:BOOL;`

Sie haben eine Eingangsvariable `xLocIn` deklariert und diese der unvollständigen Eingangsadresse `%I*` eines Feldgeräts zugewiesen. Die deklarierte Ausgangsvariable haben Sie der unvollständigen Ausgangsadresse `%Q*` zugewiesen.

3. Klicken Sie auf das Objekt `PlcProg` im Projektexplorer und fügen Sie im Deklarationsteil des Programms zwischen `VAR` und `END_VAR` zum Beispiel folgende Deklarationen ein:

`locioVar1: func;`

`locioVar2: func;`

4. Selektieren Sie das Objekt **Application** im Projektexplorer, wählen Sie den Kontextmenü-Befehl **Hinzufügen ► Globale Variablenliste** und klicken Sie im Dialog **Globale Variablenliste hinzufügen** auf die Schaltfläche **Fertigstellen**.

Die globale Variablenliste wird im Projektexplorer hinzugefügt und ihr Editor öffnet sich.

5. Ändern Sie das Schlüsselwort `VAR_GLOBAL` in `VAR_CONFIG` um.

6. Wählen Sie den Befehl **Alle Instanzpfade hinzufügen**

Folgende Instanzpfade werden hinzugefügt:

`PlcProg.logioVar1.xLocIn AT %I*;`

`PlcProg.logioVar2.xLocIN AT %I*;`

`PlcProg.logioVar1.xLocOut AT %Q*;`

`PlcProg.logioVar2.xLocOut AT %Q*;`

7. Jetzt ersetzen Sie die unvollständigen Adressangaben `%I*` und `%Q*` mit den exakten vollständigen Adressen.

Siehe auch

- [Kap. "Befehl 'Alle Instanzpfade hinzufügen'" auf Seite 722](#)

9.10.2 AT-Deklaration

Das Schlüsselwort `AT` weist in der Variablendeklaration eine Projektvariable einer bestimmten Eingabe-, Ausgabe- oder Speicheradresse der im Projektexplorer konfigurierten Steuerung zu. Ebenso können Sie Zuweisung von Variablen auf eine Adresse im Dialog **E/A-Abbild** der Geräts in der Steuerungskonfiguration festlegen.

Syntax:

```
<variable name> AT <address> : <data type>;
```

```
<address> : %<memory area prefix> (<size prefix> ? <memory position>
```

Die AT-Deklaration ermöglicht, der Adresse einen aussagekräftigen Namen zu geben. Eventuell nötige Änderungen bezüglich des ein- oder ausgehenden Signals dürfen Sie nur an einer Stelle durchführen, zum Beispiel in der Deklaration.

Beispiele:

VAR wInput AT %IW0 : WORD; END_VAR	Variablendeklaration mit Adressangabe eines Eingangsworts
VAR xActuator AT %QW0 : BOOL; END_VAR	Boolesche Variablendeklaration Hinweis: Für boolesche Variable wird intern ein Byte alloziert, wenn keine Einzelbitadresse angegeben ist. Eine Wertänderung von xActuator betrifft folglich den Bereich von QX0.0 bis QX0.7.
VAR xSensor AT IX7.5 : BOOL; END_VAR	Boolesche Variablendeklaration mit expliziter Angabe einer Einzelbitadresse. Beim Zugriff wird nur das Eingangsbit 7.5 gelesen.
VAR xSensor AT IX* : BOOL; END_VAR	Bei der Adressangabe ist statt der Speicherposition der Platzhalter * angegeben. Die endgültige Adressangabe wird in der Variablenkonfiguration vorgenommen. Hinweis: Das ist in Funktionsbausteinen möglich.

Wenn Sie eine Variable einer Adresse zuweisen, müssen Sie Folgendes beachten:

- Auf Variablen, die auf einen Eingang gelegt sind, können Sie nicht schreibend zugreifen. Dies führt zu einem Compiler-Fehler.
- AT-Deklarationen können Sie nur für lokale und globale Variablen durchführen, nicht aber für Ein- und Ausgabevariablen von Bausteinen.
- AT-Deklarationen dürfen Sie nicht in persistenten Variablenlisten verwenden!
- Wenn Sie AT-Deklarationen bei Strukturkomponenten oder Funktionsbausteinvariablen anwenden, verwenden alle Instanzen denselben Speicher. Dies entspricht der Verwendung von "statischen Variablen" in klassischen Programmiersprachen wie zum Beispiel "C".
- Das Speicher-Layout von Strukturen ist ebenfalls abhängig vom Zielsystem.



Wenn Sie nicht explizit eine Einzelbitadresse angeben, werden boolesche Variablen byteweise alloziert.

Beispiel:

```
PROGRAM PLC_PRG
VAR
```

```
xVar AT %QW0 : BOOL;
END_VAR
```

```
xVar := TRUE;
```

Siehe auch

- [Kap. 9.10.1 "Variablenkonfiguration - VAR_CONFIG" auf Seite 128](#)
- [Kap. 17.2.13 "Konfigurationsvariablen - VAR_CONFIG" auf Seite 311](#)
- [Kap. 17.4.13 "Adressen" auf Seite 371](#)

9.11 Syntax prüfen und Code analysieren

IndraLogic bietet Ihnen nützliche Funktionen, um Sie bei der Programmerstellung zu unterstützen und Fehler festzustellen. Durch die Syntax-Prüfung werden bereits während der Programmeingabe Fehler markiert und entsprechende Meldungen ausgegeben.

Die statische Codeanalyse in IndraLogic hilft Ihnen zusätzlich festgelegte Kodierrichtlinien einzuhalten sowie fehleranfällige Konstrukte zu erkennen.

Siehe auch

- [Kap. 9.11.1 "Syntax prüfen" auf Seite 132](#)
- [Kap. 9.11.2 "Code statisch analysieren" auf Seite 132](#)

9.11.1 Syntax prüfen

Während Sie den Code eingeben, führt die Vorkompilierung in IndraLogic bereits einige grundlegenden Prüfungen durch. Dabei werden fehlerhafte Stellen im Editor rot unterkringelt und eine Fehlermeldung in der Meldungsansicht ausgegeben.

Nach der Programmierung müssen Sie die Applikation übersetzen. Dies erfolgt mit dem Befehl **Erstellen** ► **Übersetzen** oder mit der Taste <Umsch> +<F11>. Dabei prüft der Compiler das Programm und gibt gegebenenfalls weitere Fehler aus.

IndraLogic erzeugt den Applikationscode aus dem im Entwicklungssystem geschriebenen Quellcode automatisch vor dem Download der Applikation auf die Steuerung. Dabei wird vor dem Erzeugen des Applikationscodes eine Prüfung der Zuweisungen, der Datentypen und der Verfügbarkeit von Bibliotheken durchgeführt. Weiterhin werden beim Erzeugen des Applikationscodes die Speicheradressen vergeben. Sie können diesen Vorgang auch explizit über den Befehl **Erstellen** ► **Code erzeugen** ausführen. Damit können Sie mögliche Fehler in Ihrem Quellcode finden, auch wenn die Steuerung noch nicht verbunden ist.

IndraLogic gibt alle Fehler und Warnungen in der Meldungsansicht in der Kategorie "Übersetzen" aus. Durch einen Doppelklick auf die Fehlermeldung öffnen Sie die betroffene POU im Editor; dabei wird die fehlerhafte Stelle markiert. Alternativ können Sie auch über das Kontextmenü der Fehlermeldung zu den fehlerhaften Stellen springen.

Beachten Sie hierzu auch die Einstellungen in den IndraLogic-Optionen.

Siehe auch

- [Kap. "Dialog 'Optionen- 'Intelligentes Codieren'" auf Seite 775](#)

9.11.2 Code statisch analysieren

IndraLogic bietet die Option den Quellcode während einer Codegenerierung zusätzlich statisch analysieren lassen (auch als "Linten" bezeichnet).

Beim "Linten" wird ermittelt, ob der Quellcode den von Ihnen festgelegten Codierrichtlinien folgt - gemäß der Idee des "Lint"-Analysewerkzeugs.

Die Definition der Coderichtlinien bzw. Regeln können Sie im Allgemeinen Bausteinteil des Projektextplorers in den **"Projekteinstellungen"** → **"Static Analysis Light"** vornehmen.

Die Prüfung erfolgt dann automatisch bei jeder Codegenerierung, beispielsweise wenn Sie den Befehl **"Erstellen"** → **"Code erzeugen"** ausführen.

Wenn Regelabweichungen vorliegen, werden diese als Fehlermeldungen der Kategorie **"Übersetzen"** im Meldungsfenster ausgegeben.

Die dabei gemeldeten Fehler haben den Präfix: SA<Zahl>.



Bei der statischen Codeanalyse mit „Static Analysis Light“ wird nur der Applikationscode des Projekts geprüft.

Bibliotheken sind von der Prüfung ausgeschlossen.

GVL-Variablen im Ordner "Allgemeiner Bausteinteil" werden nicht zwingend geprüft.

Bei Projekten mit mehreren Applikationen werden nur die Objekte geprüft, die sich in der gerade aktiven Applikation befinden.

Wenn das Projekt nur eine Applikation enthält, werden auch die Objekte im Ordner "Allgemeiner Bausteinteil" geprüft.

Codezeilen in Implementierungen mit Pragma von der statischen Analyse ausschließen

Mit Hilfe des Pragmas `{analysis ...}` kann Programm-Code so gekennzeichnet werden, dass die dabei angegebenen Regeln nicht geprüft werden. Folglich werden die so gekennzeichneten Codezeilen nicht statisch analysiert. Während der Prüfung wird der gekennzeichnete Code ignoriert.

Syntax:

```
{analysis <Vorzeichen><Regelnummer>|,<weitere Kombinationen von Vorzeichen und Regelnummer, kommasepariert>}
```

-<Regelnummer> : Prüfung der Regel SA<Regelnummer> ausschalten.

+<Regelnummer> : Prüfung der Regel SA<Regelnummer> einschalten.

Implementierungscode ausschließen

Voraussetzung: In den "Projekteinstellungen" sind Regeln zur statischen Analyse aktiviert.

Vorgehensweise um einen Code-Bereich von der Analyse auszuschließen:

1. Fügen Sie das Pragma `{analysis -<number>}` oberhalb der Codezeile ein, die den Beginn des nicht zu prüfenden Code enthält. Beispielsweise für die Regel SA0024
→ Die Codezeile ist die erste Zeile des Code-Fragments, der nicht gegen die Regel SA0024 geprüft wird.
2. Fügen Sie das Pragma `{analysis +<number>}` unterhalb der Codezeile ein, die das Ende des nicht zu prüfenden Code enthält. Beispielsweise für die Regel SA0024
→ Die Codezeile oberhalb ist die letzte Zeile des Code-Fragments, der nicht gegen die Regel SA0024 geprüft wird.

Beispiel: Nicht getyptes Literal ignorieren

Das folgende Codebeispiel zeigt, wie eine Prüfungsregel für 2 Zeilen deaktiviert werden kann, im Beispiel "SA0024: Nicht getypte Literale".

Durch die Deaktivierung wird kein Fehler ausgegeben, obwohl die Code-Zeile `nTest := #99; fehlerhaft ist (korrekter Code: nTest := DINT#99;).`

```
{analysis -24}
nTest := 99;
iVar := INT#2;
{analysis +24}
```

Beispiel: Mehrere Regeln ignorieren

Das folgende Codebeispiel zeigt, wie mehrere Regeln gleichzeitig deaktiviert und wieder aktiviert werden können:

Programm:

```
{analysis -10, -24, -18}
...
{analysis +10, +24, +18}
```

- SA0010: Arrays mit nur einer Komponente
- SA0024: Nicht getypte Literale
- SA0018: Unübliche Bitzugriffe

Programmierobjekte mit Pragma von der statischen Analyse ausschließen

Die Regel "SA0004: Mehrfacher Schreibzugriff auf Ausgang" kann nicht über Pragma deaktiviert werden!

Syntax:

```
{attribute 'analysis' := '-<Regelnummer>[,<weitere negative Regelnummern, kommasepariert>']}
```

Wenn Sie das Attributpragma im Deklarationsteil eines Programmierobjekts einfügen, werden die angegebenen Regeln für das ganze Programmierobjekt ausgeschlossen.

Wenn mehrere Regeln ausgeschlossen werden, sind die Regeln jeweils kommasepariert mit Minus und Nummer anzugeben. Eine Pragma-Anweisung für ein Einschalten wird nicht benötigt.

Beispiele:

Codebeispiel um die Regeln SA0033 und SA0031 in der Struktur `LocalData` zu ignorieren.

```
{attribute 'analysis' := '-33, -31'}
TYPE LocalData :
STRUCT
  iLocal : INT;
  uiLocal : UINT;
  udiLocal : UDINT;
END_STRUCT
END_TYPE
```

Codebeispiel um die Regel SA0100 im ARRAY `big` zu ignorieren.

```
{attribute 'analysis' := '-100'}
big: ARRAY[1..10000] OF DWORD;
```

Siehe auch

- [Kap. "Dialog 'Projekteinstellungen- 'Static Analysis Light'" auf Seite 761](#)

9.12 Orientieren und Navigieren

9.12.1 Verwendungsstellen mit der Querverweisliste finden

Sie können sich die Verwendungsstellen von Symbolen, das heißt einer Variablen, einer POU (Programm, Funktionsbaustein, Funktion) oder eines DUT, in einer so genannten "Querverweisliste", ausgeben lassen. Aus der Liste kann an die jeweiligen Stellen im Projekt gesprungen werden.

Es gibt zwei Typen der Suche nach den Verwendungsstellen für ein Symbol:

- Reine Textsuche: Sie geben dazu in der Ansicht "Querverweisliste" händisch einen Text (Symbolname, Platzhalter) ein.
- Suche nach einer konkreten Deklaration:
 - Sie befinden sich in der Ansicht "Querverweisliste" und wählen die Deklaration aus der Eingabehilfe aus.
 - Der Fokus ist im Editor eines Bausteins in einem Symbolnamen und Sie starten die Querverweissuche über einen Kontextmenübefehl.
 - Der Fokus ist im Editor eines Bausteins in einem Symbolnamen, die Ansicht "Querverweisliste" ist geöffnet, und die Querverweissuche erfolgt automatisch.
 - Sie befinden sich in der Ansicht "Querverweisliste", die bereits Verwendungsstellen für mehrere Deklarationen auflistet, und Sie schränken diese Ergebnisse auf eine bestimmte Deklaration ein.

Querverweise über Textsuche nach dem Symbolnamen

Voraussetzung: Die Ansicht "Querverweisliste" ist geöffnet.

1. Geben Sie im Feld neben nach Namen eine Zeichenfolge ein, beispielsweise den Bezeichner der Variablen, für die Sie die Verwendungsstellen im Projekt finden wollen. Beispiel: "iCounter".

Sie können für die Textsuche die Platzhalter "*" (beliebig viele Zeichen), oder "?" (genau ein beliebiges Zeichen) in Kombination mit einer Teilzeichenkette eines Variablenbezeichners verwenden.

Verwenden Sie "%" wenn Sie nach IEC-Adressen suchen wollen. Beispiele: "%MW8", "%M*".

2. Klicken Sie auf die Schaltfläche , um eine Textsuche im Projekt zu starten.

Die Ansicht "Querverweisliste" öffnet sich und zeigt die Verwendungsstellen für die Variable `iCounter`. Immer werden dabei die Deklarationsstelle und darunter eingerückt die Verwendungsstellen im Projekt dargestellt.

3. Doppelklicken Sie in der Querverweisliste auf eine Verwendungsstelle. Das entsprechende Objekt wird im Editor geöffnet und die Verwendungsstelle wird markiert.

Querverweise für eine konkrete Symboldeklaration ...aus dem Bausteineditor, mit Menübefehl

Voraussetzung: Ein Baustein ist im Editor geöffnet.

1. Setzen Sie den Cursor auf den Bezeichner des Symbols (Variable, Bausteinname) in der Deklaration oder in der Implementierung.
2. Wählen Sie im IndraWorks Menü den Befehl **Bearbeiten ► Durchsuchen ► Querverweise ausgeben**, oder im Kontextmenü der Deklaration **Durchsuchen ► Querverweise ausgeben**.

Die Ansicht **Querverweisliste** öffnet sich und zeigt die Verwendungsstellen der Variablen oder der POU.

...aus dem Bausteineditor, automatisch

Aktivieren Sie die Option "Querverweise automatisch bei Selektionsänderung aktualisieren" in den IndraLogic Optionen: Menü **Extras** ► **Optionen** ► **IndraLogic 2G** ► **SmartCoding**.

Die Querverweisliste zeigt automatisch die Verwendungsstellen für das jeweilige Symbol.

Selektieren Sie dann den Namen des Symbols im Editorfenster des Bausteins, oder setzen Sie den Cursor in den Namen.

...in der Ansicht Querverweisliste, mit Eingabehilfe

Geben Sie in der Ansicht "Querverweisliste" über die Eingabehilfe  einen Symbolnamen ins Feld neben "nach Deklaration" ein.

Die Querverweisliste zeigt die Verwendungsstellen für das Symbol.

...in der Ansicht Querverweisliste, Einschränken auf bestimmte Deklaration

Wenn in der Ansicht "Querverweisliste" mehrere Deklarationen für ein Symbol aufgelistet sind, kann die Anzeige auf eine einzige reduziert werden, beispielsweise nach einer Textsuche mit vielen Treffern:

Selektieren Sie die Zeile mit der gewünschten Deklaration und klicken auf die

Schaltfläche , oder wählen Sie den Kontextmenübefehl "Ergebnisse auf aktuelle Deklaration beschränken".

Die Querverweisliste enthält jetzt nur noch die Verwendungsstellen für die selektierte Deklaration.

Siehe auch

- [Kap. "Befehl 'Querverweisliste'" auf Seite 628](#)
- [Kap. "Befehl 'Querverweise in der klassischen Ansicht ausgeben'" auf Seite 632](#)
- [Kap. "Dialog 'Optionen- 'Intelligentes Codieren'" auf Seite 775](#)

9.12.2 Deklaration finden - 'Gehe zu Definition'

IndraLogic bietet die Möglichkeit, das gesamte Projekt nach der Definitionsstelle einer Variablen oder Funktion zu durchsuchen. Dabei wird der Baustein, der die Definition enthält, im Editor geöffnet und die Deklaration markiert.

Finden der Deklaration einer Variablen

Voraussetzung: Es ist eine POU im Editor geöffnet.

1. Setzen Sie den Cursor auf einen Bezeichner in der Implementierung.
2. Wählen Sie den Befehl **Bearbeiten** ► **Durchsuchen** ► **Gehe zu Definition**.

Die POU mit der Deklaration wird im Editor geöffnet und die Definition der Variablen wird markiert. Wenn die Definition in einer "übersetzten" Bibliothek liegt, wird der entsprechende Baustein im Bibliotheksverwalter geöffnet.



Sie können den Befehl im Offline- und Online-Modus verwenden.

Beispiele

Der folgende Baustein enthält eine Funktionsbaustein-Definition (`fbinst`), einen Programmaufruf (`prog_y()`) und einen Funktionsbaustein-Aufruf (`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 Funktionsbaustein `fb1` in seinem Editor.

Siehe auch

- [Kap. "Befehl 'Gehe zur Definition'" auf Seite 615](#)

9.12.3 Lesezeichen setzen und verwenden

Sie können Lesezeichen verwenden, um das Navigieren in langen Programmen zu erleichtern. Lesezeichen können in allen Editoren der Programmiersprachen außer AS (Ablaufsprache) verwendet werden. Über Befehle können Sie direkt zu den markierten Programmstellen navigieren.

Symbolleiste: Lesezeichen

Setzen/Löschen von Lesezeichen

Voraussetzung: Die POU ist im Editor geöffnet

1. Setzen Sie den Cursor an eine beliebige Programmzeile.
2. Wählen Sie den Befehl **Bearbeiten** ► **Lesezeichen** ► **Lesezeichen umschalten**.

An dieser Programmstelle wird ein Lesezeichen gesetzt. Dies wird durch das Lesezeichensymbol gekennzeichnet .

3. Setzen Sie mehrere Lesezeichen an verschiedenen Programmstellen.
4. Setzen Sie den Cursor an eine Programmzeile mit Lesezeichen.
5. Wählen Sie den Befehl **Bearbeiten** ► **Lesezeichen** ► **Lesezeichen umschalten**.

Das Lesezeichen wird wieder entfernt. Das Lesezeichensymbol wird gelöscht.

Alternativ dazu können Sie in der Ansicht "Lesezeichen" ein oder mehrere Lesezeichen über die Schaltfläche  löschen. Hierfür müssen die entsprechenden Lesezeichen in der Ansicht "Lesezeichen" selektiert sein.



Wählen Sie den Befehl **Bearbeiten** ► **Lesezeichen** ► **Lesezeichen löschen** um alle Lesezeichen zu entfernen.



Um alle Lesezeichen eines Projekts zu löschen, wählen Sie den Befehl "Alle Lesezeichen löschen".
Der Befehl ist nicht im Standard Befehlsumfang der IndraWorks Oberfläche enthalten.
Der Befehl kann über den Menübefehl **Extras ▶ Anpassen... ▶ Befehle ▶ Bearbeiten** in ein Menü eingefügt werden, damit er zur Verfügung steht.

Siehe auch

- [Kap. "Befehl 'Lesezeichen umschalten'" auf Seite 614](#)
- [Kap. "Befehl 'Lesezeichen löschen'" auf Seite 614](#)

Springen zu Lesezeichen innerhalb einer POU

Voraussetzung: Die POU ist im Editor geöffnet. Mehrere Lesezeichen sind gesetzt.

1. Wählen Sie den Befehl **Bearbeiten ▶ Lesezeichen ▶ Nächstes Lesezeichen**.
Abhängig von der aktuellen Cursor-Position springt der Cursor zum nächsten Lesezeichen (nach unten).
2. Wählen Sie den Befehl **Bearbeiten ▶ Lesezeichen ▶ Vorheriges Lesezeichen**.
Abhängig von der aktuellen Cursor-Position springt der Cursor zum vorherigen Lesezeichen (nach oben).

Siehe auch

- [Kap. "Befehl 'Nächstes Lesezeichen'" auf Seite 614](#)
- [Kap. "Befehl 'Vorheriges Lesezeichen'" auf Seite 614](#)

9.13 Projektweites Suchen und Ersetzen

In IndraLogic können Sie Zeichenketten in einzelnen Objekten oder im ganzen Projekt suchen und wenn gewünscht durch eine andere Zeichenkette ersetzen.

1. Aktivieren Sie im Menü **Bearbeiten ▶ Suchen und Ersetzen** den Befehl **Suchen**.
Es öffnet der Dialog **Suchen**.
2. Geben Sie im Feld **Suchen nach** die Zeichenkette ein, nach der Sie suchen wollen.
3. Wählen Sie die Suchoptionen aus.
4. Legen Sie in der Auswahlliste **Suchen in** fest, in welchen Objekten gesucht werden soll.
5. Klicken Sie auf die Schaltfläche **Weitersuchen**.
Der erste Treffer wird angezeigt.
6. Klicken Sie auf die Schaltfläche **Ersetzen**, wenn Sie den gesuchten Begriff durch einen anderen ersetzen möchten.
7. Klicken Sie auf **Alle suchen**, wenn Sie einen Übersicht über alle Treffer erhalten wollen.

9.14 Refactoring

Im Allgemeinen ist Refactoring eine Methode, bereits geschriebene Software im Design zu verbessern, ohne ihr Verhalten zu ändern.

Refactoring in IndraLogic bietet Funktionalitäten zum Umbenennen von Objekt- und Variablennamen und zum Aktualisieren von Bausteinanschlüssen.

Sie können alle Verwendungsstellen umbenannter Objekte und Variablen anzeigen lassen und diese dann gesamt oder einzeln ausgewählt umbenennen.

Zusätzlich können Sie in **Extras ► Optionen ► IndraLogic 2G** konfigurieren, ob und an welchen Stellen IndraLogic Sie automatisch zum Refactoring auffordert.

Globale Variable umbenennen

Voraussetzung: Sie haben ein Projekt geöffnet, das mindestens einen Funktionsbaustein **FB** und eine globale Variablenliste beinhaltet. Die globale Variablenliste **GVL** ist in ihrem Editor geöffnet und enthält die Deklaration einer Variablen, beispielsweise `iGlobal`. **FB** verwendet `iGlobal`.

Globale Variable projektweit umbenennen

1. Selektieren Sie den Namen einer globalen Variablen, zum Beispiel `iGlobal` in **GVL**.
2. Wählen Sie im Kontextmenü den Befehl **Refactoring ► iGlobal umbenennen**.
3. Geben Sie im Dialog **Umbenennen** in das Eingabefeld **Neuer Name** einen neuen Namen ein, zum Beispiel `iGlobalOK` und klicken Sie auf **<OK>**.

Der Dialog **Refactoring** öffnet sich. Im linken Fenster, im Projekttexplorer, sind die Objekte **GVL** und **FB** farblich rot gekennzeichnet und gelb hinterlegt. Im rechten Fenster ist **FB** in seinem Editor geöffnet und `iGlobal` ist bereits in `iGlobalOK` umbenannt.

4. Klicken Sie auf **<OK>**.
5. In Ihrem Projekt ist keine globale Variable `iGlobal`. Überall ist nun `iGlobalOK`.

Eine globale Variable projektweit umbenennen (mit Ausnahme einer POU)

1. Selektieren Sie den Namen einer globalen Variablen, zum Beispiel `iGlobal`.
2. Wählen Sie im Kontextmenü den Befehl **Refactoring ► iGlobal umbenennen**.
3. Geben Sie im Dialog **Umbenennen** in das Eingabefeld **Neuer Name** einen neuen Namen ein, zum Beispiel `iGlobalTest` und klicken Sie auf **<OK>**.

Der Dialog **Refactoring** öffnet sich. Im linken Fenster, im Projekttexplorer, sind die Objekte **GVL** und **FB** farblich rot gekennzeichnet und gelb hinterlegt. Im rechten Fenster ist der Funktionsbaustein **FB** in seinem Editor geöffnet. Statt `iGlobal` ist `iGlobalTest` aufgelistet.

4. Positionieren Sie den Cursor in das rechte Fenster und öffnen Sie das Kontextmenü.

5. Wählen Sie den Befehl **Dieses Objekt verwerfen** und klicken Sie auf <OK>

In Ihrem Projekt ist die globale Variable `iGlobal` in **FB** vorhanden. In den Objekten, in denen die Variable sonst vorkam, ist nun die Variable `iGlobalTest` angegeben.

Im Meldungsfenster erscheint eine Fehlermeldung, dass der Bezeichner `iGlobal` nicht definiert ist.

Siehe auch

- [Kap. "Befehl 'Refactoring Umbenennen'" auf Seite 617](#)
- [Kap. "Befehl 'Betreffende Anschlüsse aktualisieren'" auf Seite 620](#)

Hinzufügen und Entfernen einer Eingangsvariablen

Sie können im Deklarationsteil von Bausteinen über Refactoringbefehle Eingangs- oder Ausgangsvariablen hinzufügen oder entfernen.

An den Verwendungsstellen/Aufrufstellen der Bausteine aktualisiert IndraLogic entsprechend, wobei Sie dies pro Verwendungsstelle akzeptieren oder verwerfen können. Sie erhalten dazu den Dialog **Refactoring**.

Voraussetzung: Sie haben die POU `FCT` des Typs Funktion im Editor geöffnet. Die Funktion besitzt bereits Eingangsvariablen `input1` und `input2` und `inputx`. Sie wird in den Programmen `PlcProg` und POU aufgerufen.

1. Setzen Sie den Fokus in den Deklarationsteil der Funktion `FCT`.
2. Wählen Sie Befehl **Refactoring ► Variable hinzufügen**.
Der Standarddialog zum Deklarieren einer Variablen erscheint.
3. Deklarieren Sie Variable `input_3` mit Gültigkeitsbereich `VAR_INPUT` und Datentyp `INT`. Schließen Sie den Dialog mit <OK>.
Der Dialog **Refactoring** erscheint (siehe Abbildung unten). Die betroffenen Stellen sind gelb markiert (1)+(2).
4. Wählen Sie rechts oben die Option **Eingänge mit Platzhaltertext hinzufügen** (3).
5. Setzen Sie im linken Fenster den Cursor auf einen der gelb hinterlegten Objekte, beispielsweise `PLC_PRG`. Wählen Sie im Kontextmenü den Befehl **Das gesamte Projekt annehmen**, um die neue Variable an den Verwendungsstellen von `FCT` im gesamten Projekt hinzuzufügen.
Im rechten Fenster sehen Sie die Änderung im Implementierungsteil von `PLC_PRG`: Platzhalter `_REFACTOR_` erscheint an der Stelle, an der die neue Variable eingefügt wurde.
6. Schließen Sie den Dialog **Refactoring** mit <OK>.
7. Wählen Sie Befehl **Bearbeiten ► Suchen**. Suchen Sie im Projekt nach `"_REFACTOR_"`, um die betroffenen Stellen zu überprüfen und entsprechend zu bearbeiten.
- 8.



Alternativ können Sie die neue Variable direkt mit einem gewünschten Initialisierungswert einfügen, ohne zuerst mit einem Platzhalter zu arbeiten.

In diesem Fall wählen Sie bei Schritt 4 die Option **„mit folgendem Wert hinzufügen“** und tragen den Wert rechts davon ein.

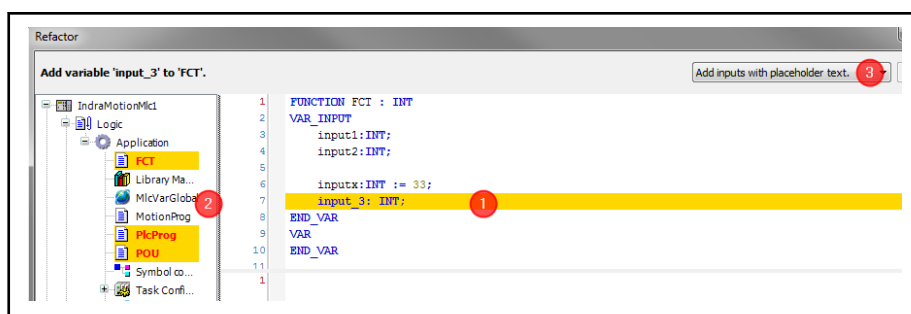


Abb. 9-21: Beispiel für neue Variable mit Platzhaltertext

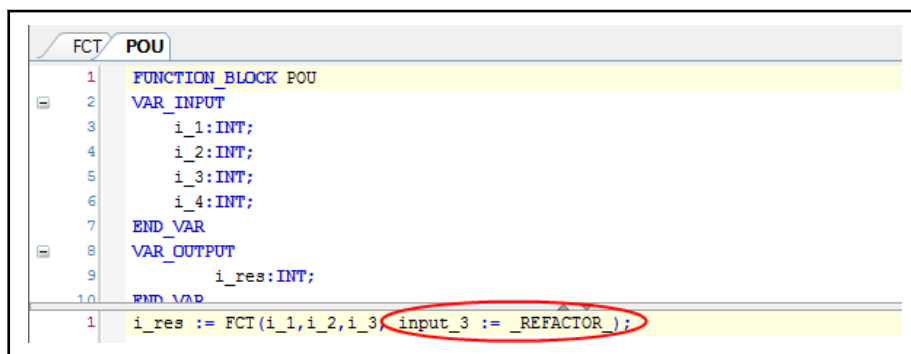


Abb. 9-22: Beispiel für neue Variable mit Platzhaltertext in einem ST-Baustein:

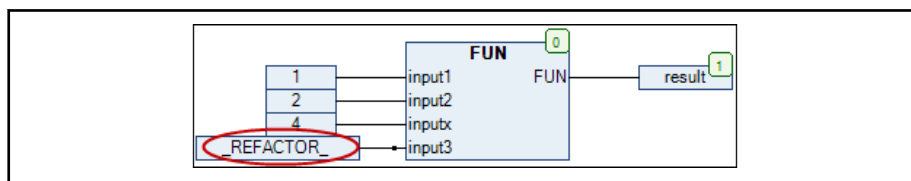


Abb. 9-23: Beispiel für neue Variable mit Platzhaltertext in einem CFC-Baustein



Beachten Sie auch die Möglichkeit, Variablen über Refactoring zu entfernen.

Siehe auch

- [Kap. "Befehl 'Variable hinzufügen'" auf Seite 618](#)
- [Kap. "Befehl 'Variable entfernen...'" auf Seite 619](#)

Variablen in der Deklaration neu anordnen

Im Deklarationsteil von Bausteinen können Sie über Refactoring die Reihenfolge von Deklarationen verändern.

Dies ist möglich bei Deklarationen der Gültigkeitsbereiche `VAR_INPUT`, `VAR_OUTPUT` oder `VAR_IN_OUT`.

Voraussetzung: Sie haben den Deklarationsteil einer POU geöffnet, der beispielsweise folgende Deklarationen enthält:

```
VAR_INPUT
invar2 : INT;
invar1 : INT;
in : DUT;
bvar : BOOL;
invar3 : INT;
```

1. Setzen Sie den Cursor in diesen Deklarationsblock und drücken die rechte Maustaste, um das Kontextmenü zu öffnen.
2. Wählen Sie den Befehl **Refactoring** ► **Variablen neu ordnen**.

Der Dialog "Neu ordnen" erscheint mit einer Liste der VAR_INPUT-Variablen.

3. Selektieren Sie beispielsweise den Eintrag `invar1 : INT;` und ziehen ihn mit der Maus vor den Eintrag `invar2`.

Die Deklaration von `invar1` steht jetzt an oberster Stelle.

4. Schließen Sie den Dialog mit <OK>.

Der Dialog "Refactoring" erscheint. Die betroffenen Stellen sind gelb markiert (siehe Abbildung oben).

5. Bestätigen Sie mit <OK> , um die neue Reihenfolge in den Baustein zu übernehmen.

Variablendeklaration ändern und Refactoring automatisch anwenden

Sie werden in der Deklaration beim Umbenennen von Variablen (mit Hilfe der Autodeklaration) von Refactoring unterstützt.

Voraussetzungen:

- Folgende Refactoring Optionen sind aktiviert:
 - "Beim Umbenennen von Variablen"
 - "Für das Hinzufügen oder Entfernen von Variablen, oder für das Ändern des Namensraums"

Aufruf: Menü **Extras** ► **Optionen** ► **IndraLogic 2G** ► **Refactoring**

- POU mit Funktionsbaustein, im Beispiel `fb_A`
- Eine deklarierte Variable, im Beispiel `iTest`

Beispiel: Ändern eines Variablennamens

1. Doppelklicken Sie den Funktionsbaustein `fb_A`.
2. Selektieren Sie in der Deklaration von `fb_A` eine Variable, beispielsweise `iTest`. Alternativ können Sie den Cursor vor oder in die Variable setzen.
3. Öffnen Sie den Dialog "Variable deklarieren" über den Menübefehl **Bearbeiten** ► **Variable deklarieren**, oder über die Tastenkombination [Umschalt]+[F2].

Der Dialog "Variable deklarieren" öffnet sich, er enthält die Einstellungen der Variable `iTest`.

Im Dialog ist die Option "Änderungen mit Hilfe von Refactoring" aktiviert.

4. Ändern Sie den Namen von `iTest` nach `iCounter_A` und bestätigen Sie die Änderung mit "OK".

Der Dialog "Refactoring Umbenennen von iTest in iCounterA" öffnet sich. Dort sind alle von der Variablenumbenennung betroffenen Stellen markiert.

5. Beenden Sie den Dialog mit "OK".

Die Änderungen werden übernommen.

Siehe auch:

- [Kap. "Befehl 'Refactoring Umbenennen'" auf Seite 617](#)
- [Kap. "Befehl 'Betreffende Anschlüsse aktualisieren'" auf Seite 620](#)
- [Kap. "Befehl 'Variable hinzufügen'" auf Seite 618](#)
- [Kap. "Befehl 'Variable entfernen...'" auf Seite 619](#)

9.15 Taskkonfiguration

In der Taskkonfiguration definieren Sie eine oder mehrere Tasks zur Steuerung und Ausführung des Applikationsprogramms auf der Steuerung.

Jede Applikation muss ein Objekt **Taskkonfiguration** besitzen.

Eine Task ist eine zeitliche Ablaufeinheit für ein IEC-Programm.

Sie ist definiert durch einen Namen, eine Priorität und einen Typ, der festlegt, welche Bedingung den Start der Task auslöst. Diese Bedingung können Sie entweder zeitlich (Zyklusintervall, freilaufend) definieren oder durch ein internes oder ein externes Ereignis definieren, bei dessen Eintreten die Task ausgeführt werden soll. Ein Ereignis ist beispielsweise die steigende Flanke einer globalen Projektvariable oder ein Interrupt-Ereignis der Steuerung.

Eine Task ruft einen oder mehrere Programmbausteine (POUs) auf. Diese Programme können sein: applikationsspezifisch, das bedeutet im Projektextplorer unterhalb der Applikation liegende Objekte, oder projektspezifisch, das bedeutet im Ordner "Allgemeiner Baustein" verfügbare Objekte. Im letzteren Fall instanziiert die Applikation das projektglobale Programm.

Wenn IndraLogic die Task im aktuellen Zyklus bearbeitet, werden die Programme für die Länge eines Zyklus ausgeführt.

Durch die Kombination von Priorität und Bedingung legen Sie fest, in welcher zeitlichen Abfolge die Tasks ausgeführt werden. Für jede Task können Sie eine Zeitüberwachung (Watchdog) konfigurieren. Zusätzlich haben Sie die Möglichkeit, beispielsweise Start, Stop, Reset direkt mit der Ausführung eines Projektbausteins zu koppeln.

Regeln für die Abarbeitungsreihenfolge der definierten Tasks:

- Wenn die Bedingung der Task erfüllt ist, arbeitet IndraLogic die Task ab.
- Wenn mehrere Tasks gleichzeitig die Bedingung zur Abarbeitung erfüllen, arbeitet IndraLogic zuerst diejenige mit der höchsten Priorität ab.
- Wenn mehrere Tasks gleichzeitig die Bedingung zur Abarbeitung erfüllen und gleiche Prioritätsstufe haben, arbeitet IndraLogic zuerst diejenige ab, die bereits am längsten wartet.
- Die Abarbeitung der Programmaufrufe erfolgt jeweils in der Reihenfolge, in der die Programme im Konfigurationsdialog der Task angeordnet sind.
- Wenn ein Programm aufgerufen wird, das mit gleichem Namen sowohl im Projektextplorer der Applikation als auch in einer Bibliothek oder projektglobal im Ordner "Allgemeiner Baustein" vorliegt, wird das Programm der Applikation verwendet.

Wichtige Hinweise

Alle Tasks teilen sich 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 müssen Sie selbst dafür sorgen, dass die Applikation bei Konflikten die Eingangsdaten in einen sicheren Bereich kopiert, bei Ausgängen gilt dasselbe.

Möglichkeiten, die Konsistenz- und Synchronisierungsprobleme zu lösen, bieten beispielsweise die Bausteine der Bibliothek SysSem.library.

Auch beim Zugriff auf andere globale Objekte, wie zum Beispiel globale Variablen oder Bausteine, können Konsistenzprobleme auftreten. Konsistenzprobleme treten immer auf, wenn mehrere Tasks eine Variable lesend und schreibend verwenden. Zur Lösung stehen hier wiederum beispielsweise die Bausteine der Bibliothek `SysSem` zur Verfügung.

Siehe auch

- [Kap. 18.1.25 "Objekt 'Taskkonfiguration'" auf Seite 598](#)
- [Kap. 18.1.26 "Objekt 'Task'" auf Seite 602](#)

9.15.1 Erstellen einer Taskkonfiguration

Voraussetzung: Das geöffnete Projekt enthält eine POU vom Typ Programm und im Projektexplorer ist unter **Application** eine **Task-Konfiguration** mit einer **Task** eingefügt.

1. Doppelklicken Sie im Projektexplorer auf das Task-Objekt unter der **Task-Konfiguration**.
Es öffnet sich die Registerkarte **Konfiguration** des Task-Objekts.
2. Wählen Sie in der Auswahlliste **Typ** den Eintrag **Zyklisch** aus.
Das Eingabefeld **Intervall** (z.B. **t#200ms**) ist sichtbar.
3. Geben Sie in das Feld **Intervall** (z.B. **t#200ms**) den Wert **t#300ms** ein.
4. Klicken Sie auf die Schaltfläche **+Aufruf hinzufügen**.
Die Eingabehilfe erscheint.
5. Wählen Sie in der **Eingabehilfe** ► **Kategorien** aus der **Kategorie Programm** die gewünschte POU aus und klicken Sie auf **OK**.
IndraLogic fügt die ausgewählte POU in der POU-Liste der Registerkarte **Konfiguration** und im Projektexplorer unterhalb des Task-Objekts ein.
Bei Ausführung der Applikation auf der Steuerung führt IndraLogic die im Task-Objekt ausgewählte POU zyklisch im Intervall von 300 ms aus.

- [Kap. "Registerkarte 'Konfiguration'" auf Seite 602](#)

9.16 Datenpersistenz

Die Lebensdauer von Variablen und deren Daten reicht vom Zeitpunkt, an dem eine Variable erzeugt wird, bis zu dem Zeitpunkt, zu dem sie gelöscht und ihr Speicherplatz wieder freigegeben wird. Der Zeitpunkt der Variablenerzeugung, Initialisierung oder Instanziierung hängt vom deklarierten Gültigkeitsbereich ab. Der Zeitpunkt der Speicherfreigabe üblicherweise auch. Beispielsweise wird der Speicher von globalen Variablen mit Beenden der Applikation freigegeben.

Sie können Daten länger als üblich erhalten. Dafür stehen Ihnen folgende Mechanismen zur Verfügung.

Mechanismen zur Datenerhaltung:

- (A): Persistente globale Variablenliste  mit dem Schlüsselwort `PERSISTENT RETAIN`
Die persistenten Variablen behalten bei einem erneuten Laden der Applikation ihren Wert. Auch nach einem Download, Warmstart, oder Kaltstart werden die Werte wiederhergestellt.
- (B): Retain-Variablen mit dem Schlüsselwort `RETAIN`

Die Retain-Variablen behalten ihren Wert nach einem Warmstart, nicht aber bei einem erneuten Laden der Applikation, einem Download oder Kaltstart.

- (C): Rezepturvariablen

Rezepturvariablen und ihre Werte werden in der Rezepturdatei gesichert.

Mechanismen im Vergleich

Welcher Mechanismus ist bei welchem Anwendungsfall geeignet?

In der folgenden Tabelle werden einige, übliche Anwendungsfälle aus dem Bereich von Haussteuerungen betrachtet.

Gegenüberstellung von Mechanismen und Anwendungsfällen

	Anwendungsfall	(A) Persistente Variablen	(B) Retain-Variablen	(D) Rezepturvariablen
1	Die Applikation muss Geräteeinstellungen erhalten. Beispiel: Die Haussteuerung muss nach einem Stromausfall die Information zur Verfügung haben, wie viel Zeit eine Jalousie für ihr Hochfahren benötigt.	Geeignet¹ Bevorzugter Anwendungsfall Sie können in diesem Fall statt persistenter Variablen auch Retain-Variablen verwenden. Das ist bei Variablen vorteilhaft, deren Deklaration oft geändert wird.	Geeignet Bevorzugter Anwendungsfall Retain-Variablen sind vorteilhaft, wenn deren Deklarationen oft geändert werden.	Möglich, aber sehr umständlich und deshalb nicht empfehlenswert.
2	Die Applikation muss Werte auch nach Programmänderungen oder Erweiterungen erhalten.			
	2a: Seltene Erweiterungen Beispiel: Ein Applikationsprogrammierer erweitert das Programm um einen neuen Schalter und baut ein neues Licht ein. Die Haussteuerung muss bis dahin gesicherte Werte danach noch zur Verfügung haben.	Geeignet¹ Bevorzugter Anwendungsfall	Geeignet	Möglich, aber umständlich
	2b: Freiere Änderungen, auch Löschen oder Ändern des Datentyps von Variablen Die Haussteuerung läuft und ist persistiert. Wenn ein Applikationsprogrammierer die Steuerung um eine neue Funktionalität und deswegen in einem Funktionsbaustein um eine weitere persistente Variable erweitert, müssen die bis dahin gesicherten Werte erhalten bleiben. Beispielsweise wird das Programm in einem Funktionsbaustein um eine Variable, die das automatische Ausschalten einer bisher ungesteuerten Lampe nach einer bestimmten Zeit steuert, erweitert. Die Haussteuerung muss nach der Erweiterung die Zeiten aller gesteuerten Lampen zur Verfügung haben.	Nicht geeignet	Geeignet Daten von Retain-Variablen bleiben nach einem Online-Change möglichst erhalten.	Möglich, wenn textuell, aber umständlich
	2c: Die Applikation muss Werte nach einem Download erhalten.	Geeignet	Nicht geeignet	Geeignet
3	Die Applikation muss unterschiedliche Wertesätze verwenden können. Beispiel: Die Betriebseinstellungen für Sommer, Winter und Urlaub müssen gespeichert und nach Bedarf wieder einge spielt werden können.	Nicht geeignet	Nicht geeignet	Geeignet Bevorzugter Anwendungsfall

	Anwendungsfall	(A) Persistente Variablen	(B) Retain-Variablen	(D) Rezepturvariablen
4	Die Applikation muss Einstellungen von einem anderen System verwenden können. Das Übertragen von Einstellungen auf eine andere Anlage, die ähnliche Variablen verwendet, muss möglich sein.	Nicht geeignet	Nicht geeignet	Geeignet²
5	Die Applikation muss menschenlesbare Daten zur Verfügung stellen. Der Benutzer muss die Daten lesen, vergleichen und editieren können.	Nicht geeignet	Nicht geeignet	Geeignet²

1 Vorteil: Schnelligkeit; empfohlener Anwendungsfall: 1 und 2a
 2 Vorteil: Extern bearbeitbar, Übertragbarkeit. Nachteil: Umständlich

Lebensdauer von Variablen bei Aufruf von Online-Befehlen

Benutzereingaben im Menü Online	Variable mit üblicher Lebensdauer (weder RETAIN noch PERSISTENT)	RETAIN	PERSISTENT RETAIN PERSISTENT PERSISTENT RETAIN
Befehl "Online-Change"	X	X	X
Befehl "Reset warm"	-	X	X
Befehl "Reset kalt"	-	-	X
Befehl "Laden"	-	-	X ¹
Befehl "Reset Ursprung"	-	-	-

X Variable behält ihren Wert
 - Variable wird initialisiert
 1 Hinweis: Zur Struktur von persistenten Daten beachten Sie bitte die Informationen unter "[Mechanismus beim Download](#)" auf [Seite 148](#)

Siehe auch:

- [Kap. "Befehl 'Online-Change'" auf Seite 657](#)
- [Kap. "Befehl 'Reset Ursprung'" auf Seite 663](#)
- [Kap. "Befehl 'Reset Kalt'" auf Seite 662](#)
- [Kap. "Befehl 'Laden'" auf Seite 656](#)

Lebensdauer von Variablen beim Laden eines Bootprojekts

Die Werte von normalen Variablen verlieren ihren Wert und werden neu initialisiert.

Die Werte von persistenten Variablen bleiben erhalten, wenn:

- Die Struktur der persistenten Variablen im Speicher zur Struktur in der persistenten Datenliste passt

Die Werte von Retain-Variablen bleiben erhalten, wenn:

- Die Struktur der persistenten Variablen im Speicher zur Struktur in der persistenten Datenliste passt
- Die persistenten Variablen zur Applikation passen (GUID muss übereinstimmen)

Wenn beim Booten der Applikation die Voraussetzungen für das Wiederherstellen der Werte von Retain-Variablen und persistenten Variablen nicht erfüllt sind, erfolgt ein "Retain mismatch".



Zur Struktur von persistenten Daten beachten Sie bitte die Informationen unter "[Mechanismus beim Download](#)" auf Seite 148

9.16.1 Daten mit persistenten Variablen erhalten

Persistente Variablen behalten ihren Wert bei einem erneuten Laden der Applikation, nach einem Download, Warmstart oder Kaltstart.

Das Sichern der Daten erfolgt im nicht flüchtigen NVRAM-Speicher der Steuerung und erfordert keinen zusätzlichen Zeitbedarf.

Verhalten *Werterhalt bei:*

- Unkontrolliertem Beenden
- Warmstart durch Aufruf des Befehls "Reset warm"
- Kaltstart durch Aufruf des Befehls "Reset kalt"
- Erneutem Download der Applikation

Neuinitialisierung bei:

- Aufruf des Befehls "Reset Ursprung"

Persistente Variablen werden also nur dann neu initialisiert, wenn Sie die Steuerung in den Auslieferungszustand zurücksetzen, beispielsweise wenn Sie den Menübefehl **Debuggen ▶ Reset Ursprung** wählen.

Wenn Sie dagegen einen erneuten Download der Applikation ausführen, bleiben die persistierten Daten nach Möglichkeit erhalten. Das hängt davon ab, wie tiefgreifend die Änderungen waren, die zum Download führten. Eine Änderung des Applikationsnamens führt immer zu einer vollständigen Neuinitialisierung. Änderungen der Implementierungen führen nie zu einer Neuinitialisierung, die Datenpersistenz bleibt vollständig erhalten. Änderungen der Deklarationen führen nur zu einer Initialisierung der neuen Variablen bei Persistenz der bestehenden Variablen, wenn Folgendes beachtet wird: Sie ändern die Deklarationen so, dass die persistente Variablenliste konsistent bleibt. Das ist der Fall, wenn Sie eine neue Variable hinzufügen, oder eine bestehende löschen. Inkonsistenzen können entstehen, wenn Sie die Bezeichner, oder die Datentypen der vorher deklarierten, persistenten Variablen bearbeiten und ändern.

Mechanismus beim Download

Wenn Sie im Persistenzeditor die Variablenliste bearbeiten, bewirken Sie, dass die Variablenliste nicht so wie im Editor dargestellt gespeichert wird, sondern automatisch nachbearbeitet wird, bevor sie gespeichert wird.

Bei der Nachbearbeitung wird eine Variable, die Sie entfernt haben, durch eine Platzhaltervariable mit dem gleichen Speicherbedarf ersetzt. Das führt dazu, dass im Prozessabbild die nachfolgenden Variablen ihre Adressen beibehalten. Außerdem wird eine Variable, die Sie hinzufügen, ans Ende der Liste verschoben. Durch die Nachbearbeitung können Änderungen, die zu Persistenzverlust führen würden, neutralisiert werden. Aber Sie erzeugen Lücken, die zusätzlichen Speicher belegen.

Beim Download wird auf der Steuerung der CRC-Wert der Variablenliste und die Länge der Liste (Anzahl der Variablen) hinterlegt. Beim erneuten Download wird der neue Prüfwert mit dem aktuell auf der Steuerung befindlichen Prüfwert verglichen. Dann wird die Variablenliste sukzessive bis zu der angegebenen Länge verglichen. Wenn Sie eine Deklaration, also den Namen oder den Datentyp, bearbeitet haben, wird die Variable neu initialisiert. Ansonsten bleibt ihr Wert erhalten. Beim erneuten Download überprüft

IndraLogic, ob die im Persistenzeditor deklarierte Variablenliste noch konsistent zur bereits auf der Steuerung liegenden Variablenliste ist.

Der Mechanismus funktioniert zuverlässig, wenn die Variablen selbst nicht gravierend geändert werden. Weitreichende Änderungen bei den Bezeichnungen und den Datentypen führen zu einer Neuinitialisierung und dem Verlust der Persistenz. Wenn aufgrund Anwendungsanforderungen mit häufigen Änderungen zu rechnen ist, ist eine Variablenliste grundsätzlich nicht empfehlenswert. Außerdem ist eine persistente Variable bei einem Online-Change nach einer Datentypänderung weniger robust als eine übliche Variable ohne Persistenz.

Es wird empfohlen die Variablenliste von Lücken zu bereinigen, verwenden Sie hierzu den Kontextbefehl "Liste neu ordnen und Lücken bereinigen" im Deklarationsteil des Persistenzeditors.



Das Bereinigen der Variablenliste führt zu einer Initialisierung aller persistenten Variablen, da die bereinigte Liste nicht mehr mit der Liste auf der Steuerung übereinstimmt.

Durch die Initialisierung geht die Persistenz aller Variablen auf der Steuerung verloren.

Daten retten über den Rezepturverwalter

Um die globale persistente Variablenliste zu bereinigen, ohne die Persistenz zu verlieren, können die Daten mit Hilfe des Rezepturverwalters in einer Rezeptur gesichert werden. Dabei wird im Rezepturverwalter eine Liste für alle Variablen der persistenten Variablenliste erzeugt und gleichzeitig werden deren aktuellen Werte von der Steuerung als Rezeptur abgespeichert. Wählen Sie anschließend im Deklarationsteil des Persistenzeditors den Kontextbefehl "Liste neu ordnen und Lücken bereinigen" aus und führen Sie danach einen erneuten Download aus. Wenn Sie danach den Kontextbefehl "Werte aus Rezeptur wiederherstellen" auswählen, werden die in der Rezeptur gespeicherten Werte wiederhergestellt.

Änderung einer bestehenden Deklaration in der Persistenten Variablenliste

Wenn Sie den Namen oder den Datentyp einer Variablen ändern, gilt die Änderung als neue Deklaration und bewirkt eine Initialisierung der Variablen beim nächsten Online-Change, oder Download. Bei komplexen Datentypen liegt eine Änderung bereits dann vor, wenn eine neue Komponente hinzugefügt wurde, oder wenn in der Tiefe einer verwendeten Struktur beispielsweise der Typ einer Variablen von INT nach UINT geändert wird.

Grundsätzlich sind komplexe, benutzerdefinierte Datentypen nicht für die Verwaltung in einer persistenten Variablenliste geeignet, da selbst kleine Änderung eine Initialisierung der Variable mit allen Komponenten bewirkt.

Doppelallokation von Speicher bei Instanzpfaden

Die Persistierung von Variablen kann erfolgen bei:

- Globalen Variablen
- Lokal in einem Funktionsbaustein deklarierten Variablen
- Im Programm deklarierten Variablen

Ergänzen Sie hierzu die Variablen-Deklaration mit dem Schlüsselwort `PERSISTENT`.

Zusätzlich müssen Sie in der persistenten globalen Variablenliste den Instanzpfad auf die persistente Variable einfügen, indem Sie im Persistenzeditor den Kontextbefehl "Alle Instanzpfade hinzufügen" ausführen.

Die Persistenz wird über folgenden Mechanismus garantiert:

- Es wird ermittelt, in welchen zyklischen Tasks ein Zugriff auf die Variable erfolgt.

- Am Ende der ersten zyklischen Task (in jedem Zyklus) wird die Variable in die persistente globale Variablenliste kopiert.
- Nach dem Neustart der Steuerung wird der Wert von der persistenten Variable in die normale Variable kopiert.

Der Nachteil dieses Mechanismus ist, dass sowohl an der Deklarationsstelle als auch an der Stelle des Instanzpfades Speicher allokiert wird. Die persistente Variable belegt doppelten Speicherplatz. Außerdem werden die Daten in jedem Zyklus an beide Stellen kopiert. Der Kopiervorgang kann zeitintensiv sein, insbesondere wenn es sich um große, strukturierte Werte handelt.

Speicherlokation bei persistenten Funktionsbaustein-Instanzen

Eine Funktionsbausteininstanz liegt immer vollständig im Speicher. Das ist notwendig, damit der gleiche Code auf verschiedenen Instanzen arbeiten kann. Wenn bereits eine einzige Variable in einem Funktionsbaustein mit `PERSISTENT` gekennzeichnet wird, wird die Funktionsbausteininstanz vollständig mit allen Variablen im remanenten Speicher abgelegt, obwohl nur die eine Variable als persistent behandelt wird. Nichtflüchtiger Speicher steht aber nicht in gleichem Umfang zur Verfügung wie Arbeitsspeicher.

Ein Funktionsbaustein, der als Variable einen Pointer auf eine Instanz im NVRAM hat, wird nicht im gesicherten Speicher abgelegt.

Siehe:

- [Kap. 17.2.11 "Persistente Variable - PERSISTENT" auf Seite 308](#)
- [Kap. "Befehl 'Alle Instanzpfade hinzufügen'" auf Seite 722](#)

9.16.2 Daten mit Retain-Variablen erhalten

Retain-Variablen behalten ihren Wert nach einem Warmstart. Der Grad der Werteerhaltung ist bei persistenten Variablen aber höher.

Das Sichern der Retain-Variablen erfolgt im nicht flüchtigen NVRAM-Speicher der Steuerung und erfordert keinen zusätzlichen Zeitbedarf.

Deklaration Um eine Retain-Variable zu deklarieren, fügen Sie einer Variablendeklaration das Schlüsselwort `RETAIN` hinzu.

Verhalten *Werterhalt bei:*

- Unkontrolliertem Beenden
- Aufruf des Befehls "Reset warm"

Neuinitialisierung bei:

- Erneutem Download der Applikation
- Aufruf des Befehls "Reset kalt" (im Gegensatz zu persistenten Variablen)
- Aufruf des Befehls "Reset Ursprung"

Wenn Sie eine Applikation neu starten, werden deren Variablen üblicherweise mit explizit vorgegebenen Initialwerten, oder mit einem Standardwert initialisiert.

Variablen, die mit dem Schlüsselwort `RETAIN` gekennzeichnet sind, werden im NVRAM-Speicher der Steuerung verwaltet und behalten ihren Wert auch dann, wenn die Steuerung z. B. bei einem Stromausfall ausfällt. Ein Anwendungsfall für die Verwendung von Retain-Variablen stellt z. B. ein Stückzähler in einer Fertigungsanlage dar, der auch nach einem Stromausfall korrekt weiterzählen muss.

Speicherlokation von persistenten Funktionsbaustein-Instanzen

Funktionsbaustein-Instanzen liegen "am Stück" im Speicher. Das ist notwendig, damit der gleiche Code auf verschiedenen Instanzen arbeiten kann. Wenn nun eine Variable in einem Funktionsbaustein mit `RETAIN` gekennzeichnet ist, dann wird jede Instanz des Funktionsbausteins mit allen Variab-

len gesichert. Auch die Variablen des Funktionsbausteins, die nicht so gekennzeichnet sind. Nichtflüchtiger Speicher steht aber nicht in gleichem Maß zur Verfügung wie Arbeitsspeicher.

Ein Funktionsbaustein, der als Variable einen Pointer auf eine Instanz im SRAM hat, wird nicht im gesicherten Speicher abgelegt.

Siehe [Kap. 17.2.11 "Persistente Variable - PERSISTENT" auf Seite 308](#)

9.16.3 Daten mit Rezepturen erhalten

Im Rezepturverwalter werden Variablen persistent verwaltet. Der Rezepturverwalter benötigt auf der Steuerung keinen speziellen Speicher, um Werte und Daten zu erhalten.

Deklaration Eine Rezepturdefinition besteht aus einem Satz an Variablen mit Werten und wird im Objekt "Rezepturverwalter" erstellt, bearbeitet und als Datei gesichert.

- Funktionalität**
- Sie können eine Variable in mehreren Rezepturen mit jeweils unterschiedlichen Werten einbinden.
 - Sie können im Onlinebetrieb aus der Steuerung die Istwerte der Variablen einlesen und als Rezepturwert (Sollwert) speichern.
 - Sie können mit Hilfe der Bibliothek "Recipe Management" die Erstellung und Verarbeitung einer Rezeptur programmatisch implementieren.
 - Sie können eine Rezeptur als Rezepturdatei speichern und sichern.

Siehe

- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)
- [Kap. "Befehl 'Aktuelle Werte in Rezeptur speichern'" auf Seite 721](#)
- [Kap. "Befehl 'Liste neu ordnen und Lücken bereinigen'" auf Seite 721](#)
- [Kap. "Befehl 'Werte aus Rezeptur wiederherstellen'" auf Seite 722](#)
- [Kap. 18.1.21 "Objekt 'Rezepturverwalter'" auf Seite 583](#)
- [Kap. 18.1.22 "Objekt 'Rezepturdefinition'" auf Seite 587](#)

9.16.4 VAR PERSISTENT-Variablen deklarieren

Die folgenden Schritte geben ein Beispiel, wie eine Persistente Variablenliste erstellt und eine persistente Variable in der persistenten Variablenliste und in einer POU deklariert wird.

Voraussetzungen:

- Ein Projekt ist geöffnet und enthält eine Programm-POU
- In den IndraWorks Optionen **Extras ► Optionen ► IndraLogic 2G ► Deklarationseditor** ist die Option für die textuelle Ansicht aktiviert

Beispiel: Persistente Variablenliste anlegen und persistente Variable deklarieren

1. Fügen Sie dem Applikationsobjekt mit dem Kontext-Menübefehl **Hinzufügen** das Objekt **Globale Persistente Variablenliste...** hinzu.
Der Dialog Persistente Variablen hinzufügen öffnet sich, hier können Sie einen Namen für die Persistente Variablenliste vergeben (der Defaultname ist "PersistentVars").
2. Bestätigen Sie den Dialog mit "Fertigstellen".

IndraLogic fügt die persistente Variablenliste  **PersistentVars** unter dem Applikationsobjekt im Projektextplorer ein.

3. Öffnen Sie den Editor der Persistenten Variablenliste **PersistentVars** durch Doppelklick im Projektextplorer.
4. Geben Sie im Editor zwischen `VAR_GLOBAL PERSISTENT RETAIN` und `END_VAR` eine Variablendeklaration ein,
zum Beispiel `ivarpersist1 : INT;`.
5. Doppelklicken Sie auf die POU im Projektextplorer, in der die persistente Variable deklariert werden soll.
Der Editor der POU öffnet sich.
6. Geben Sie im Deklarationsteil die folgende Deklaration ein:

```
VAR PERSISTENT
    ivarpersist2 : INT;
END_VAR
```
7. Wählen Sie den Menübefehl **Erstellen ► Übersetzen**.
Das Meldungsfenster öffnet sich. Wenn IndraLogic die Applikation fehlerfrei übersetzt hat, schließen Sie das Meldungsfenster und fahren Sie mit dem nächsten Schritt fort. Ansonsten beheben Sie den/die Fehler und wählen erneut den Menübefehl **Erstellen ► Übersetzen**.
8. Wechseln Sie in den Editor der Persistenten Variablenliste **PersistentVars**, um die persistente Variablenliste als aktiven Editor zu wählen.
Setzen Sie hierzu den Cursor in den Deklarationsteil und wählen Sie den Kontext-Menübefehl **Alle Instanzpfade hinzufügen**.
IndraLogic fügt die persistente Variable der persistenten Variablenliste "PersistentVars" der POU hinzu:

```
// Instanzpfad der persistenten Variablen erzeugt
POU.ivarpersist2: INT
```

Siehe auch

- [Kap. 17.2.11 "Persistente Variable - PERSISTENT" auf Seite 308](#)
- [Kap. 18.1.12 "Objekt 'Persistente Variablenliste'" auf Seite 546](#)
- [Kap. "Befehl 'Alle Instanzpfade hinzufügen'" auf Seite 722](#)

9.16.5 Speichern der Werte einer persistenten Variablenliste in einer Rezeptur

Voraussetzung: Ein Projekt ist geöffnet und unter einem Applikationsobjekt existiert eine persistente Variablenliste mit Deklarationen persistenter Variablen.

1. Selektieren Sie Ihr Applikationsobjekt im Projektextplorer und wählen Sie den Kontextmenü-Befehl **Aktive Applikation setzen**.
Das Applikationsobjekt ist fett dargestellt.
2. Wählen Sie den Menübefehl **Debuggen ► Einloggen**.
Ihre Applikation ist auf der Steuerung eingeloggt und das Applikationsobjekt im Projektextplorer ist grün hinterlegt.

3. Doppelklicken Sie auf die persistente Variablenliste und wählen Sie den Befehl **Deklarationen ▶ Aktuelle Werte in Rezeptur speichern**.

IndraLogic erzeugt unter dem Applikationsobjekt die Objekte **Rezeptur-verwalter** und **PersistentVariables**.

4. Wählen Sie den Menübefehl **Debuggen ▶ Ausloggen**.

Die Applikation ist aus der Steuerung ausgeloggt.

Siehe auch

- [Kap. "Befehl 'Aktuelle Werte in Rezeptur speichern'" auf Seite 721](#)
- [Kap. 9.16 "Datenpersistenz" auf Seite 144](#)

Häufige Fragen

F: Beim (erstmaligen) Abspeichern aktueller Werte in eine Rezeptur wird dies mit einer Fehlermeldung verweigert.

A: Überprüfen Sie, ob in Ihrem Projekt ein Bibliotheksinfo-Objekt (Ordner "Allgemeiner Baustein") als "Projektinformation" verwendet wird ([Kap. 18.1.20 "Objekt 'Bibliotheksinfo'" auf Seite 579](#)). Schalten Sie die Metadatum "Freigegeben" unter "Zusammenfassung" ab und wiederholen das Abspeichern.

9.17 Bausteine für implizite Prüfung verwenden

IndraLogic stellt spezielle POUs zur Verfügung, die implizite Überwachungsfunktionen implementieren. Diese überprüfen während der Laufzeit die Grenzen von Arrays oder Unterbereichstypen, die Gültigkeit von Pointer-Adressen oder eine Division durch 0.

1. Selektieren Sie das Objekt **Applikation** im Projektextplorer.

Wählen Sie den Befehl **Hinzufügen ▶ POU Vorlagen ▶ IndraMotion MLC Base**

Es erscheint der Dialog **POU für implizite Prüfungen hinzufügen**.

2. Aktivieren Sie die gewünschten Funktionen.
3. Klicken Sie auf die Schaltfläche **Hinzufügen**.

Die ausgewählten POUs werden unterhalb der **Applikation** im Projektextplorer eingefügt.

4. Öffnen Sie die POUs im Editor.
5. Passen Sie den Implementierungsvorschlag Ihren Anforderungen an.

VORSICHT

Funktionalität der Überwachungsfunktionen

Um die Funktionalität der Überwachungsfunktionen zu erhalten, dürfen Sie den Deklarationsteil nicht verändern. Als einzige Ausnahme dürfen Sie lokale Variablen hinzufügen.

Siehe auch

- [Kap. 18.1.18 "Überwachung des SPS-Programms der Applikation während der Laufzeit" auf Seite 576](#)

9.18 Objektorientiert programmieren

IndraLogic unterstützt die objektorientierte Programmierung mit Funktionsbausteinen und stellt dafür folgende Funktionalitäten und Objekte zur Verfügung:

- POU-Methoden, Schnittstellen-Methoden
- POU-Eigenschaften, Schnittstellen-Eigenschaften
- Schnittstellen
- Vererbung
- Methodenaufruf, virtueller Funktionsaufruf
- Definition von Funktionsbausteinen als Erweiterungen anderer Funktionsbausteine

Siehe auch

- [Kap. "Objekt 'Methode' " auf Seite 564](#)
- [Kap. "Objekt 'Schnittstelle\(INTERFACE\)" auf Seite 562](#)

9.18.1 Erweitern eines Funktionsbausteins

Die Erweiterung eines Funktionsbausteins basiert auf dem Konzept der Vererbung in der objektorientierten Programmierung. Ein abgeleiteter Funktionsbaustein "erweitert" zu diesem Zweck einen Basis-Funktionsbaustein und erhält dadurch die Eigenschaften des Basis-Funktionsbausteins zusätzlich zu seinen eigenen Eigenschaften.

Die Erweiterung eines Funktionsbausteins bedeutet:

- Der abgeleitete Funktionsbaustein enthält alle Daten und Methoden, die vom Basis-Funktionsbaustein definiert sind.

Eine Instanz des abgeleiteten Funktionsbausteins können Sie in jedem Kontext verwenden, in dem IndraLogic einen Funktionsbaustein vom Typ des Basis-Funktionsbausteins erwartet.

- Der abgeleitete Funktionsbaustein darf die Methoden überschreiben, die Sie im Basis-Funktionsbaustein definiert haben.

Das bedeutet:

Der abgeleitete Funktionsbaustein kann eine Methode mit demselben Namen, denselben Eingängen und denselben Ausgängen definieren, wie vom Basis-Funktionsbaustein definiert ist.



Für dieses Überschreiben der vom Basisbaustein geerbten Methoden, Aktionen, Eigenschaften, Transitionen erhalten Sie folgendermaßen Unterstützung:

Wenn Sie eine Methode, Aktion etc. unterhalb des abgeleiteten Bausteins einfügen, erhalten Sie im Dialog "Objekt hinzufügen" eine Combobox mit einer Auswahl der im Basisbaustein verwendeten Methoden, Aktionen etc. und können diese übernehmen, um sie dann nur noch entsprechend anzupassen.

- Der abgeleitete Funktionsbaustein darf keine Funktionsbaustein-Variablen mit demselben Namen enthalten, wie Sie der Basis-Funktionsbaustein verwendet. Dies meldet der Compiler als Fehler.

Einzige Ausnahme: Wenn Sie eine Variable im Basis-Funktionsbaustein als `VAR_TEMP` deklariert haben, dann darf der abgeleitete Funktions-

baustein eine Variable mit demselben Namen definieren. In diesem Fall kann der abgeleitete Funktionsbaustein auf die Variable des Basis-Funktionsbausteins nicht mehr zugreifen.

- Sie können die Variablen und Methoden des Basis-Funktionsbausteins innerhalb des Gültigkeitsbereichs des abgeleiteten Funktionsbausteins durch die Verwendung des **SUPER**-Zeigers direkt ansprechen.



Mehrfache Vererbung ist nicht erlaubt!

Ausnahme: Ein Funktionsbaustein kann mehrere Schnittstellen implementieren und eine Schnittstelle kann andere Schnittstellen erweitern.

Siehe auch

- [Kap. 17.2.14 "SUPER" auf Seite 312](#)
- [Kap. 17.2.15 "THIS" auf Seite 313](#)
- [Kap. "Objekt 'Methode' " auf Seite 564](#)
- [Kap. "Objekt 'Eigenschaft'" auf Seite 571](#)
- [Kap. "Objekt 'Aktion'" auf Seite 574](#)
- [Kap. "Objekt 'Transition'" auf Seite 576](#)

Erweitern eines Basis-Funktionsbausteins durch einen neuen Funktionsbaustein

Voraussetzung: Das aktuell geöffnete Projekt besitzt einen Basis-Funktionsbaustein, zum Beispiel **POU_1(FB)**, der durch einen neuen Funktionsbaustein erweitert werden soll.

1. Selektieren Sie das Objekt **Application** im Projektexplorer und wählen Sie den Menübefehl **Hinzufügen ► POU**.

Der Dialog **Objekt Hinzufügen Assistent** öffnet sich.

2. Geben Sie einen Namen für den neuen Funktionsbaustein in das Eingabefeld **Name** ein, beispielsweise **POU_Ex**.
3. Wählen Sie **Funktionsbaustein**.
4. Wählen Sie **Erweitert**: und klicken Sie auf die Schaltfläche .
5. Wählen Sie in der Eingabehilfe aus der Kategorie **Funktionsbausteine** unter **Application** die POU(FB) aus, die als Basis-Funktionsbaustein dienen soll, zum Beispiel **POU_1** und klicken Sie auf **OK**.
6. Optional können Sie einen **Zugriffsmodifizierer** für den neuen Funktionsbaustein aus der Combobox auswählen.
7. Wählen Sie aus der Combobox **Implementierungssprache** zum Beispiel **Strukturierter Text(ST)** aus.
8. Klicken Sie auf **Hinzufügen**.

IndraLogic fügt den Funktionsbaustein **POU_Ex** in den Projektexplorer ein, und der Editor öffnet sich.

In der ersten Zeile steht:

```
FUNCTION_BLOCK POU_Ex EXTENDS POU_1
```

Der Funktionsbaustein **POU_Ex** erweitert den Basis-Funktionsbaustein **POU_1**.

Erweitern eines Basis-Funktionsbausteins durch einen bestehenden Funktionsbaustein

Voraussetzung: Das aktuell geöffnete Projekt besitzt einen Basis-Funktionsbaustein, beispielsweise **POU_1 (FB)**, und einen weiteren Funktionsbaustein, beispielsweise **POU_Ex (FB)**. Der Funktionsbaustein **POU_Ex (FB)** soll auch die Eigenschaften des Basis-Funktionsbausteins erhalten. Das bedeutet: **POU_Ex (FB)** soll **POU_1 (FB)** erweitern.

1. Doppelklicken Sie im Projektextplorer auf den Funktionsbaustein `POU_Ex (FB)`
Der Editor des Funktionsbausteins öffnet sich.
2. Erweitern Sie den bestehenden Eintrag der obersten Zeile `FUNCTION_BLOCK POU_Ex` mit `EXTENDS POU_1`
Der Funktionsbaustein `POU_Ex` erweitert den Basis-Funktionsbaustein `POU_1`

Siehe auch

- [Kap. "Objekt 'Funktionsbaustein' \(FUNCTION_BLOCK\)" auf Seite 556](#)

9.18.2 Schnittstellen implementieren (IMPLEMENTS)

Die Implementierung von Schnittstellen basiert auf dem Konzept der objektorientierten Programmierung. Über gemeinsame Schnittstellen können Sie verschiedene aber gleichartige Funktionsbausteine auch gleichartig verwenden.

Ein Funktionsbaustein, der eine Schnittstelle implementiert, muss alle Methoden und Eigenschaften enthalten, die in dieser Schnittstelle definiert sind (Schnittstellen-Methoden und Schnittstellen-Eigenschaften).

Das bedeutet: Name, Eingänge und Ausgang der jeweiligen Methode oder Eigenschaft müssen exakt gleich sein.

Wenn Sie einen neuen Funktionsbausteins anlegen, der eine Schnittstelle implementiert, fügt IndraLogic automatisch alle Methoden und Eigenschaften dieser Schnittstelle unterhalb des neuen Funktionsbausteins im Explorer ein.



Wenn Sie danach weitere Methoden der Schnittstelle hinzufügen, fügt IndraLogic diese Methoden nicht automatisch auch in den betreffenden Funktionsbausteinen hinzu.

Für die Aktualisierung müssen Sie explizit den Befehl **Schnittstellen implementieren** auswählen.

Bei abgeleiteten Funktionsbausteinen müssen Sie darauf achten, dass Methoden oder Eigenschaften, die durch "Vererbung" einer Schnittstelle übernommen wurden, die gewünschte Implementierung erhalten oder gelöscht werden, falls die in der Basis vorliegende Implementierung verwendet werden soll. Sie erhalten entsprechende Übersetzungsfehlermeldungen oder Warnungen, provoziert durch automatisch eingefügte Pragmaattribute. Sehen Sie bitte [Kap. "Befehl 'Schnittstellen implementieren'" auf Seite 742](#).



- Einer Variablen vom Typ einer Schnittstelle müssen Sie die Schnittstelle eines Funktionsbausteins zuweisen, bevor eine Methode über die Variable aufgerufen werden kann.
- Eine Variable vom Typ einer Schnittstelle ist immer eine Referenz der zugewiesenen Funktionsbausteininstanz.

Eine Variable vom Typ einer Schnittstelle ist eine Referenz auf Instanzen von Funktionsbausteinen. Eine solche Variable kann auf jeden Funktionsbaustein verweisen, der die Schnittstelle implementiert. Wenn keine Zuweisung auf die Variable erfolgt ist, dann enthält die Variable im Online-Betrieb den Wert 0.

INTERFACE ITF_1 enthält die Methode GetName:

```
METHOD GetName : STRING
```

Funktionsbausteine FB_A und FB_B implementieren jeweils die Schnittstelle ITF_1:

```
FUNCTION_BLOCK FB_A IMPLEMENTS ITF_1
END_FUNCTION_BLOCK
```

```
FUNCTION_BLOCK FB_B IMPLEMENTS ITF_1
END_FUNCTION_BLOCK
```

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

Die Implementation ist für jeden FB getrennt auszuführen.

Funktionsbausteine FB_A, Methode GetName (Beispiel)

```
METHOD GetName : STRING
  GetName:= 'FB_A';
```

Funktionsbausteine FB_B, Methode GetName (Beispiel)

```
METHOD GetName : STRING
  GetName:= 'FB_B';
```

Betrachten Sie nun die Funktion DeliverName mit ihrem Eingang einer Variable vom Typ der Schnittstelle ITF_1:

Funktion DeliverName: STRING

```
FUNCTION DeliverName : STRING
VAR_INPUT
  I_i: ITF_1;
END_VAR
DeliverName:= I_i.GetName(); // in the case, it depends on the "actual" type
                             // if A.GetName or B.GetName is called.
```

Diese Eingabevariable kann alle Funktionsbausteine empfangen, die die "Schnittstelle ITF_1" implementieren.

- [Kap. "Befehl 'Schnittstellen implementieren'" auf Seite 742](#)

**Implementieren einer Schnittstelle
in einen neuen Funktionsbaustein**

Voraussetzung: Das aktuell geöffnete Projekt besitzt mindestens ein Schnittstellen-Objekt.

1. Selektieren Sie **Application** im Projektextplorer und wählen Sie den Menübefehl **Hinzufügen ► POU**.
Der Dialog **Objekt Hinzufügen Assistent** öffnet sich.
2. Geben Sie einen Namen für den neuen Funktionsbaustein in das Eingabefeld **Name** ein, zum Beispiel **POU_Im**.
3. Wählen Sie **Funktionsbaustein**.
4. Wählen Sie **Implementiert** und klicken Sie auf die Schaltfläche .
5. Wählen Sie in der Eingabehilfe aus der Kategorie **Schnittstellen** die Schnittstelle zum Beispiel **ITF1** und klicken Sie auf **OK**.
6. Um eine weitere Schnittstellen einzufügen, klicken Sie erneut auf  und wählen Sie eine weitere Schnittstelle aus.
7. Optional können Sie einen **Zugriffsmodifizierer** für den neuen Funktionsbaustein aus der Auswahlliste auswählen.

8. Wählen Sie aus der Auswahlliste **Implementierungssprache** zum Beispiel **Strukturierter Text(ST)** aus.
9. Klicken Sie auf **Hinzufügen**.

IndraLogic fügt den Funktionsbaustein **POU_Im** in den Projektextplorer ein und der Editor öffnet sich. In der ersten Zeile steht:

```
FUNCTION_BLOCK POU_Im IMPLEMENTS ITF1
```

Die Schnittstelle und ihre Methoden und Eigenschaften sind jetzt im Projektextplorer unter dem Funktionsbaustein eingefügt und Sie können nun Programmcode im Implementierungsteil der Schnittstelle und ihrer Methoden eingeben.

Implementieren einer Schnittstelle in einen bestehenden Funktionsbaustein

Voraussetzung: Das aktuell geöffnete Projekt besitzt einen Funktionsbaustein, zum Beispiel **POU_Im** und mindestens ein Schnittstellen-Objekt, zum Beispiel **ITF1**.

1. Doppelklicken Sie im Projektextplorer auf die POU **POU_Im(FB)**.
Der Editor der POU öffnet sich.
2. Erweitern Sie den bestehenden Eintrag der obersten Zeile `FUNCTION_BLOCK POU_Im` mit `IMPLEMENTS ITF1`.
Der Funktionsbaustein **POU_Im** implementiert die Schnittstelle **ITF1**.

9.18.3 Erweitern einer Schnittstelle

Sie können Schnittstellen ebenso wie Funktionsbausteine erweitern. Die Schnittstelle erhält dann auch die Schnittstellen-Methoden und Schnittstellen-Eigenschaften der Basis-Schnittstelle, zusätzlich zu ihren eigenen.

Erzeugen einer Schnittstelle, die eine andere Schnittstelle erweitert.

1. Selektieren Sie das Objekt **Application** im Projektextplorer.
 2. Wählen Sie den Befehl **Hinzufügen ► Schnittstelle**.
Der Dialog **Schnittstelle hinzufügen** öffnet sich.
 3. Geben Sie einen Namen für die neue Schnittstelle ein.
 4. Aktivieren Sie die Option **Erweitert** und klicken Sie auf die Schaltfläche .
 5. Die Eingabehilfe öffnet sich.
 6. Wählen Sie aus der Kategorie **Schnittstellen** die Schnittstelle aus, die durch die neue Schnittstelle erweitert werden soll.
- [Kap. "Objekt 'Schnittstelle\(INTERFACE\)'" auf Seite 562](#)

9.18.4 Methodenaufruf

Um einen Methodenaufruf zu implementieren, werden den Schnittstellenvariablen die tatsächlichen Parameter (Argumente) übergeben. Dabei kann alternativ auf die Parameternamen verzichtet werden.

Je nach deklarierter Zugriffsmodifizierer kann eine Methode nur innerhalb des eigenen Namensraums (**INTERNAL**), nur innerhalb des eigenen Programmierbausteins und seinen Ableitungen (**PROTECTED**), oder nur innerhalb des eigenen Programmierbausteins (**PRIVATE**) aufgerufen werden. Bei **PUBLIC** kann die Methode überall aufgerufen werden.

Innerhalb der Implementierung kann sich eine Methode selbst rekursiv aufrufen, entweder direkt mit Hilfe des **THIS**-Pointers, oder mit Hilfe einer lokalen Variablen für den zugeordneten Funktionsbaustein.

Methodenaufruf als virtueller Funktionsaufruf

Durch Vererbung kann es zu virtuellen Funktionsaufrufen kommen.

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

In folgenden Fällen wird der Methodenaufruf dynamisch gebunden:

- Sie rufen eine Methode über einen Pointer auf einen Funktionsbaustein auf (zum Beispiel `pFub^.method`).
Der Pointer kann in dieser Situation auf Instanzen vom Typ des Funktionsbausteins und auf Instanzen von allen abgeleiteten Funktionsbausteinen zeigen.
- Sie rufen eine Methode einer Schnittstellen-Variablen auf (zum Beispiel `interface1.method`).
Die Schnittstelle kann auf alle Instanzen von Funktionsbausteinen verweisen, die diese Schnittstelle implementieren.
- Eine Methode ruft eine andere Methode desselben Funktionsbausteins auf. Die Methode kann in dem Fall auch die Methode eines abgeleiteten Funktionsbausteins mit gleichem Namen aufrufen.
- Der Aufruf einer Methode erfolgt über eine Referenz auf einen Funktionsbaustein. Die Referenz kann in dieser Situation auf Instanzen vom Typ des Funktionsbausteins und auf Instanzen von allen abgeleiteten Funktionsbausteinen zeigen.
- Sie weisen `VAR_IN_OUT`-Variablen eines Basis-Funktionsbaustein-Typen einer Instanz eines abgeleiteten FB-Typen zu.
Die Variable kann in dieser Situation auf Instanzen vom Typ des Funktionsbausteins und auf Instanzen von allen abgeleiteten Funktionsbausteinen zeigen.

Beispiel: Methoden überladen

Die Funktionsbausteine `fub1` und `fub2` erweitern den Funktionsbaustein `fubbase` und implementieren die Schnittstelle `interface1`. Es gibt die Methoden `method1` und `method2`.

```

PROGRAM PlcProg
VAR INPUT
  b : BOOL;
END_VAR
VAR pInst : POINTER TO fubbase;
  instBase : fubbase;
  inst1 : fub1;
  inst2 : fub2;
  instRef : REFERENCE to fubbase;
END_VAR
IF b THEN
  instRef REF= inst1;          (* reference to fub1 *)
  pInst := ADR(instBase);
ELSE
  instRef REF= inst2;          (* reference to fub2 *)
  pInst := ADR(inst1);
END IF
pInst^.method1(); (* If b is TRUE, fubbase.method1 will be called,
                    otherwise fub1.method1 is called *)
instRef.method1(); (* If b ist TRUE, fub1.method1 will be called,
                    otherwise fub2.method1 is called*)

```

Unter der Annahme, dass `fubbase` des oberen Beispiels zwei Methoden `method1` und `method2` enthält, überschreibt `fub1` `method2`, aber nicht `method1`. Der Aufruf von `method1` erfolgt folgendermaßen:

```
pInst^.method1();
```

Wenn `b` `TRUE` ist, ruft IndraLogic `fubbase.method1` auf, ansonsten `fub1.method1`.

Siehe auch

- [Kap. "Objekt 'Methode' " auf Seite 564](#)
- [Kap. 17.2.15 "THIS" auf Seite 313](#)
- [Kap. 17.2.14 "SUPER" auf Seite 312](#)
- [Kap. "Objekt 'Funktion\(FUNCTION\)" auf Seite 558](#)

Zusätzliche Ausgänge

Gemäß der Norm IEC 61131-3 können Methoden so wie normale Funktionen zusätzliche Ausgänge deklariert haben. Beim Methodenaufruf weisen Sie den zusätzlichen Ausgängen Variablen zu.

Genaue Informationen hierzu finden Sie beim Thema "Funktion".

Syntax beim Aufruf

Programm:

```
<function block name>.<method name>(<first input
name> := <value> (, <further input assignments>)+ , <first output name> => <first output variable name> (, <further out
```

Beispiel:

Deklaration

```
METHOD PUBLIC DoIt : BOOL
VAR_INPUT
  iInput_1 : DWORD;
  iInput_2 : DWORD;
  sInput_3 : STRING(12);
END_VAR
```

Aufruf

```
bFinishedMethod := fbInstance.DoIt(sInput_3 := 'Hello World ', iInput_2 := 16#FFFF, iInput_1 := 16);
```

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

Bei Aufruf der Methode werden die Werte der Methodenausgänge auf die lokal deklarierte Ausgabevariablen geschrieben.

In der Gerätebeschreibung kann definiert sein, dass eine bestimmte Funktionsbausteininstanz (eines Bibliotheksbausteins) eine bestimmte Methode immer taskzyklisch aufruft. Wenn die Methode die Eingabeparameter des folgenden Beispiels enthält, arbeitet IndraLogic die Methode auch dann ab, wenn die aktive Applikation gerade im STOP-Status ist:

Beispiel

```
VAR_INPUT
  pTaskInfo : POINTER TO DWORD;
  pApplicationInfo : POINTER TO _IMPLICIT_APPLICATION_INFO;
END_VAR
(*Now the status of the application can be queried via pApplicationInfo
and the instructions can be implemented: *)

IF pApplicationInfo^.state = RUNNING THEN <instructions> END_IF;
```

Methode rekursiv aufrufen



Verwenden Sie Rekursionen vorwiegend zur Bearbeitung von rekursiven Datentypen wie beispielsweise verketteten Listen. Allgemein ist es ratsam bei der Verwendung von Rekursion vorsichtig zu sein. Bei einer unerwartet tiefen Rekursion kann es zu einem Stacküberlauf und damit zu einem Maschinenstillstand kommen!

Innerhalb der Implementierung kann sich eine Methode selbst aufrufen:

- Direkt mit Hilfe des THIS-Pointers

- Indirekt mit Hilfe einer lokalen Funktionsbausteininstanz des Basisfunktionsbausteins

Üblicherweise wird bei einem rekursiven Aufruf eine Compilerwarnung ausgegeben. Wenn die Methode mit dem Pragma `{attribute 'estimated-stack-usage' := '<estimated stack size in bytes>'}` versehen ist, wird die Compilerwarnung unterdrückt. Im Kapitel "Attribut 'estimated-stack-usage'" finden Sie ein Implementierungsbeispiel.

Siehe auch:

- [Kap. "Attribut 'estimated-stack-usage'" auf Seite 409](#)
- [Kap. 17.2.15 "THIS" auf Seite 313](#)
- [Kap. 17.2.14 "SUPER" auf Seite 312](#)
- [Kap. "Objekt 'Methode' " auf Seite 564](#)
- [Kap. "Objekt 'Eigenschaft'" auf Seite 571](#)

10 Mit Steuerungsnetzwerken arbeiten

IndraLogic unterstützt die Kommunikation zwischen Steuerungsgeräten (SPS) und das Unterordnen einer Sicherheitssteuerung unter ein Steuerungsgerät durch folgende Funktionalitäten:

- **Symbolkonfiguration:** Für die Variablen in einer Applikation erzeugt IndraLogic Symbole mit bestimmten Zugriffsrechten. Über diese Symbole können Sie von extern auf die Variablen zugreifen, beispielsweise von einem OPC-Server aus.
- **Datenquellenverwalter:** Verwaltet die Verbindungseinstellung und die Datenübertragung zu entfernten Geräten (Datenquellen). Die übertragenen Daten werden in Datenquellenvariablen abgebildet, auf die in der Visualisierung, oder der lokalen Applikation zugegriffen werden kann. Ein Beispiel dafür ist ein Controlpanel, das entfernte Geräte steuert und den Zustand der Geräte als HMI-Applikation visualisiert.
- **Netzwerkvariablen:** Netzwerkvariablen sind Variablen, deren Werte verschiedenen Steuerungsgeräten im Netzwerk zugänglich sind. Die Variablen müssen dazu in starren, identischen Listen sowohl im Sendergerät als auch im Empfängergerät definiert sein. Diese Listen sind Applikationen zugeordnet, können aber in verschiedenen Projekten liegen.
- Im Projektextplorer kann eine Sicherheitssteuerung unter die Steuerung eingefügt werden. Die Kommunikationsverbindungen der Sicherheitssteuerung zu den Feldgeräten, Steuerungsnetzwerken und zum Programmiersystem werden dann über diese Steuerung geroutet.

Siehe auch

- [Kap. 10.2 "Symbolkonfiguration" auf Seite 167](#)
- [Kap. 10.4.1 "Datenquellenverwalter Grundlagen" auf Seite 174](#)
- [Kap. 10.3 "Netzwerkvariablen" auf Seite 168](#)
- [Kap. 10.1.1 "Netzwerktopologie" auf Seite 163](#)
- [Kap. 10.1.2 "Kommunikation im Steuerungsverbund über Gateway" auf Seite 164](#)
- [Kap. 10.5 "Untergeordnete Sicherheitssteuerung" auf Seite 184](#)

10.1 Netzwerk und Adressierung

Bauen Sie ein Steuerungsnetzwerk hierarchisch auf, damit weitgehende Selbstkonfiguration möglich ist.

In IndraLogic wird die Netzwerk-Topologie auf eindeutige Adressen abgebildet und der Routing-Algorithmus wird durch strukturierte Adressen einfach gehalten. Es gibt direkte und relative Adressierung und automatische Adressbestimmung während des Hochfahrens des Systems.

Siehe auch

- [Kap. 10.1.1 "Netzwerktopologie" auf Seite 163](#)
- [Kap. 10.1.2 "Kommunikation im Steuerungsverbund über Gateway" auf Seite 164](#)

10.1.1 Netzwerktopologie

Informationen und Empfehlungen zur Topologie eines Steuerungsnetzwerks

Empfohlen wird, ein Netzwerksystem so einzurichten, dass Folgendes möglich wird:

- Weitreichende Selbstkonfiguration (Adresszuweisung)

- Transparente Unterstützung jedes Kommunikationsmediums
- Beförderung von Datenpaketen zwischen verschiedenen Netzwerken

Dabei sollte der Routing-Mechanismus so einfach sein, dass jeder Netzwerkknoten Datenpakete umleiten kann, auch wenn er geringe Speicherkapazität hat. Vermeiden Sie deshalb umfangreiche Routing-Tabellen, komplexe Berechnungen oder Anfragen während der Laufzeit.

Bauen Sie das Steuerungsnetzwerk hierarchisch auf. Jeder Knoten darf einen Vaterknoten und eine beliebige Anzahl an Kindknoten besitzen. Ein Knoten ohne Vater ist ein "Top-Level"-Knoten. Zyklen sind nicht erlaubt, das heißt 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 Sub-Netzen konfiguriert werden, für die der Knoten jeweils als Vater fungiert.

Ein Netzwerkbereich darf nur einen einzigen Vaterknoten haben. Deshalb ist eine Konfiguration, in der ein Netzwerkbereich gleichzeitig als Subnetz von mehreren Knoten definiert wird, ungültig.

Siehe auch

- [Kap. 10.1.2 "Kommunikation im Steuerungsverbund über Gateway" auf Seite 164](#)

10.1.2 Kommunikation im Steuerungsverbund über Gateway

Die SPS-Kommunikation von einem Engineering-PC zu einem oder mehreren Steuerungsgeräten (System MLC oder MTX) erfolgt immer über ein IndraLogic-Gateway.

Das folgende Schaubild erläutert die Kommunikationsbeziehung.

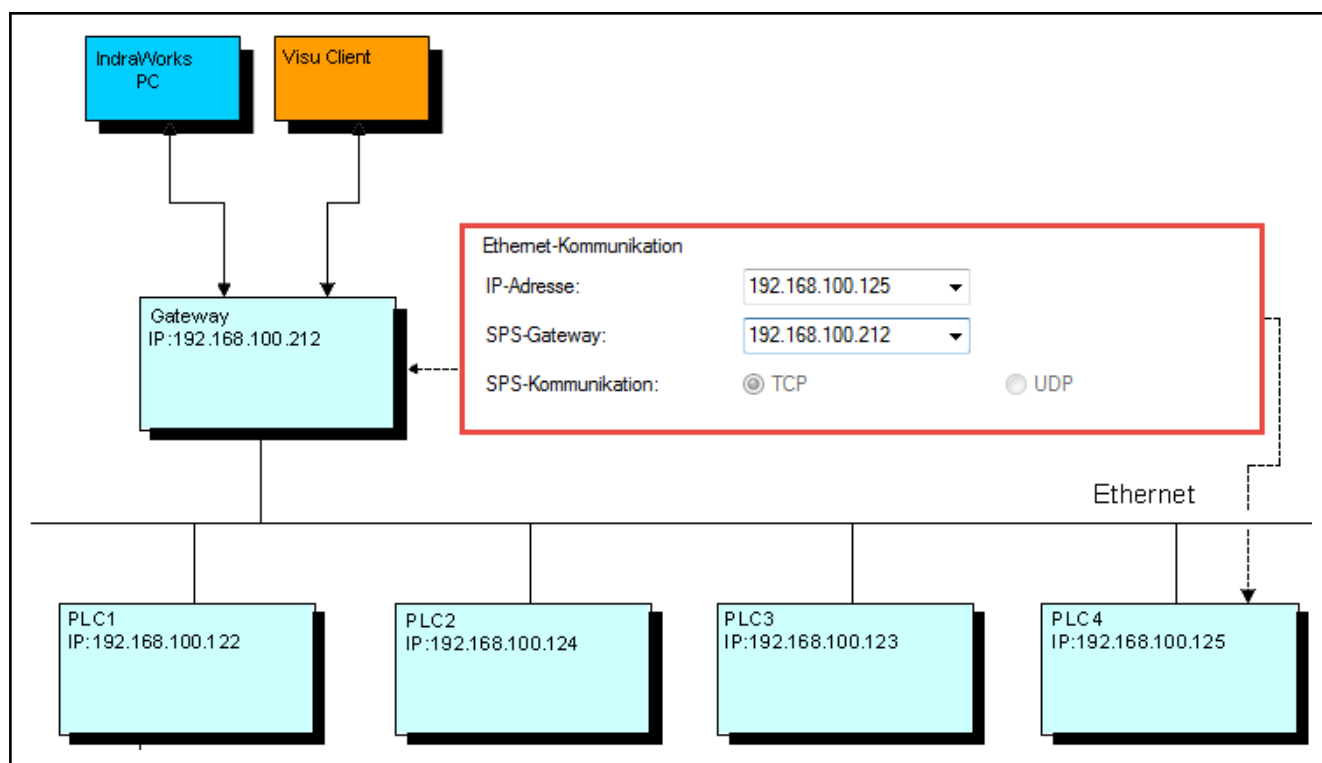


Abb. 10-1: Kommunikationsbeziehungen

Die notwendigen Einstellungen erfolgen im jeweiligen Geräteassistenten.

Ein IndraLogic-Gateway kann entweder auf dem eigenen Engineering-PC (Einstellung: localhost bzw. 127.0.0.1) oder abgesetzt im Steuerungsnetzwerk (Einstellung: IP-Adresse des Gateway-PCs) laufen.

Abb. 10-2: Einstellung von IP- und Gateway-Adresse

Nach einem erfolgreichen Verbindungstest wird die IP-Adresse zur Steuerung angezeigt (hier 192.168.100.125). Die TCP-Adresse kann direkt im Visualisierungclient und OPC Server eingestellt werden.

Ist die SPS-Kommunikation nicht erfolgreich sind die folgenden Fehlerursachen zu untersuchen. Die Fehlerbehebung wird exemplarisch unter Windows XP gezeigt. Die Lösung kann auf anderen Windows Versionen differieren.



Eine SPS-Kommunikation zum Gerät ist nur über die Geräte-Engineering-Schnittstelle möglich!

Verbindung zum Gateway überprüfen

Bei fehlender Gatewayverbindung wird nach einem Verbindungstest eine der folgenden Meldung ausgegeben:

- IndraLogic Gateway nicht gefunden.
- IndraLogic-Gateway: keine Kommunikation. Gateway offline?

Stellen Sie sicher das der Gateway-Server läuft.

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 Programmsymbol bietet über die rechte Maustaste ein Kontextmenü mit Start- und Stop-Befehle, worüber der Dienst jederzeit vom Anwender gestoppt und gestartet werden kann.

Stellen Sie außerdem sicher das in der Windows **Systemsteuerung** unter **Verwaltung** - **Dienste** sowohl der <IndraLogic Service Control>-Dienst als auch der <IndraLogic V14 Gateway>-Dienst laufen.

Es darf jeweils nur ein <IndraLogic Service Control>- und <IndraLogic V14 Gateway>-Dienst laufen.

Beenden Sie evtl. weitere Dienste und stellen Sie den Autostarttyp dieser Dienste auf "Deaktiviert".

Falls der Verbindungstest weiterhin fehlschlägt, ändern Sie die Einstellungen für das IndraLogic-Gateway von <localhost> in <127.0.0.1> und dann in Ihre eigene IP-Adresse.

Alternativ können Sie das Gateway eines anderen Engineering-PC durch die Eingabe seiner IP-Adresse unter dem IndraLogic-Gateway im Geräteassistenten verwenden.

Verbindung zum Gerät überprüfen

Bei fehlender Geräteverbindung wird nach einem Verbindungstest eine der folgenden Meldungen ausgegeben:

- Keine Verbindung zum Gerät. Gerät offline?
- Keine Verbindung zum Gerät. Gerät offline? Fehler: <>

Überprüfen Sie zunächst die Steuerung. Ist die Steuerung abgestürzt? Booten Sie die Steuerung neu.

Ist ein Firewall aktiv? Insbesondere wenn die Steuerung auf einem PC läuft. Deaktivieren Sie den Firewall.

Wurden Änderungen an der Gateway.cfg oder GWClient.cfg Datei durchgeführt? Verwenden Sie bitte die Original-Dateien ohne Änderung.

Wurde CoDeSys parallel zu IndraWorks installiert? Beenden Sie die CoDeSys-Simulation.

Fehlerhafte Antwort vom Gerät

Kann grundsätzlich mit dem Gerät kommuniziert werden, kann es sein, dass der Gerätetyp oder die Geräteversion nicht übereinstimmen. In diesem Fall werden bei einem Verbindungstest die folgenden Meldungen ausgegeben:

Unterschiedliche Gerätetypen:

- Das gewählte Zielsystem stimmt nicht mit dem verbundenen Gerät überein.

ID-Nichtübereinstimmung:

angefordert=1001 0003, online=1001 0103

Unterschiedliche Geräteversionen:

- Das gewählte Zielsystem stimmt nicht mit dem verbundenen Gerät überein.

Nichtübereinstimmung der Version:

angefordert=14.12.0.0, online=13.11.4.0

Aktualisieren Sie über die IndraWorks Firmwareverwaltung die Firmware ihres Gerätes.

10.2 Symbolkonfiguration

Verwenden Sie die Symbolkonfiguration, um für Projektvariablen Symbole mit bestimmten Zugriffsrechten bereitzustellen.

Über diese Symbole können Sie von extern auf die Variablen zugreifen, beispielsweise von einem OPC-Server aus.

IndraLogic erzeugt bei der Codegenerierung eine Symboldatei (xml-Datei), die die Beschreibung der Symbole enthält.

Die Symboldatei wird im Projektverzeichnis gespeichert.

Der Name der Symboldatei setzt sich wie folgt zusammen: <project name>.<device name>.<application name>.xml

Beispiel:

```
proj_xy.PLC1.application.xml
```



Sie können die Symboldatei auch durch den Befehl **Erstellen ► Code erzeugen** erzeugen. Dies ist sehr hilfreich, wenn ein Download auf die SPS nicht möglich ist.

Die Variablen, die Sie als Symbole exportieren, können Sie entweder im Symbolkonfigurationseditor, oder in der Variablendeklaration über das Pragmaattribut {attribute 'symbol'} definieren.

Eine weitere Möglichkeit bietet der AS-Editor:

Hier können Sie im Eigenschafteneditor eines Elements die implizit erzeugten Elementvariablen bestimmen, die in die Symbolkonfiguration exportiert werden.

Der Name eines Symbols wird in der Symbolkonfiguration gemäß folgender Syntax erzeugt: <Applikationsname>.<POU-Name>.<Variablennamen>.

Beim Zugriff auf die Variable müssen Sie den Symbolnamen immer vollständig in dieser Syntax angeben.

Beispiel:

```
MyApplication.PlcProg.a oder MyApplication.SymFB.a
```



Auf Symbole für Eingangsadressen beziehungsweise für Variablen, die auf Eingangskanäle abgebildet sind, kann grundsätzlich nur lesend zugegriffen werden.

Nur im Simulationsbetrieb ist zu Testzwecken auch schreibender Zugriff möglich.

Die Symboldatei wird zusammen mit der Applikation auf die Steuerung geladen.

Siehe auch

- [Kap. 18.1.24 "Objekt 'Symbolkonfiguration'" auf Seite 588](#)
- [Kap. "Attribut 'symbol'" auf Seite 436](#)
- [Kap. "AS-Elementeigenschaften" auf Seite 270](#)
- [Kap. "Registerkarte 'Applikationen'" auf Seite 521](#)

Anlegen einer Symbolkonfiguration

Voraussetzung: Das Projekt kann fehlerfrei übersetzt werden

Das Objekt **Symbolkonfiguration** wird normalerweise automatisch mit angelegt. Es kann jedoch -aus welchen Gründen immer- gelöscht worden sein. Die Neuanlage läuft dann wie folgt ab:

1. Selektieren Sie im Projektextplorer das Objekt **Applikation**.
2. Wählen Sie den Kontextmenü-Befehl **Hinzufügen ► Symbolkonfiguration**.

Das Objekt **Symbolkonfiguration** wurde zum Projektextplorer hinzugefügt und der Editor des Objekts ist geöffnet.

3. Öffnen Sie das Menü **Ansicht** des Editors und aktivieren Sie die Kategorien von Variablen, die Sie im Konfigurationseditor angeboten haben möchten: Wählen Sie dann den Befehl **Erstellen** im Symbolkonfigurationseditor.

Alle verfügbaren Variablen gemäß der aktuell eingestellten Filter im Menü **Ansicht** werden in einer Baumstruktur angezeigt.

4. Wählen Sie einzelne Variablen durch Aktivieren der zugehörigen Check-box aus.



Beachten Sie die aktuellen Einstellungen; siehe Schaltfläche "Einstellungen" in der Menüleiste des Editors.

Im Feld unterhalb der Menüleiste des Editors erhalten Sie Informationen zur aktuellen Situation und eine passende Anleitung sowie Schaltflächen zur Durchführung von Korrekturen.

5. Folgen Sie der Aufforderung im Feld unterhalb der Menüleiste. Im vorliegenden Fall sollte dies nur die Information sein, dass die geänderte Symbolkonfiguration mit dem nächsten Download oder Online-Change übertragen wird.

Wählen Sie im Hauptmenü den Befehl **Erstellen ► Code erzeugen**.

Die Datei `<project name>.<device name>.<application name>.xml` wird im Projektverzeichnis erzeugt.

Beim Download der Applikation oder beim Online Change überträgt IndraLogic die Symbolkonfiguration auf die Steuerung.

Siehe auch

- [Kap. 18.1.24 "Objekt 'Symbolkonfiguration'" auf Seite 588](#)

10.3 Netzwerkvariablen

Netzwerkvariablen sind Variablen, deren Werte zwischen verschiedenen Steuerungsgeräten im Netzwerk ausgetauscht werden können. Die Variablen

müssen dazu in starren, identischen Listen sowohl im Sendergerät als auch im Empfängergerät definiert sein, wobei nur eine Applikation eines Geräts Netzwerkvariablen definieren darf. Die Listen können innerhalb eines Projekts oder in verschiedenen Projekten liegen.

Die Netzwerkvariablenliste im **Sender** ist eine globale Variablenliste, in deren Objekteigenschaften bestimmte Protokoll- und Übertragungsparameter definiert sind.

Durch Hinzufügen dieser Eigenschaften erzeugen Sie aus einer "normalen" **GVL** eine "Globale Netzwerkvariablenliste (Sender)".

Sie können aber auch direkt ein Objekt  **Globale Netzwerkvariablenliste (Sender)** im Projektextplorer einfügen, für das diese Parameter bereits gesetzt sind.

Die Netzwerkvariablenliste im **Empfänger** ist vom Typ  **Globale Netzwerkvariablenliste (Empfänger)**.

Beim Anlegen wählen Sie die zugehörige Netzwerkvariablenliste des Senders aus. Alternativ können Sie diese Variablenliste aus einer Exportdatei einlesen, die zuvor aus der Senderliste erzeugt wurde. Eine Exportdatei ist auf jeden Fall nötig, wenn die Senderliste in einem anderen Projekt definiert ist.

Die Übertragung von Netzwerkvariablen erfolgt als **Broadcasting** immer nur in eine Richtung, vom Sender zum Empfänger. Es ist allerdings möglich, dass ein Gerät sowohl Sender- als auch Empfängerlisten enthält.

Ab Version 3.5.7.0 der Bibliothek `NetVarUdp` wird kein Empfangskanal mehr belegt, wenn kein bestätigter Transfer gewählt wird. Dadurch ist auch Netzwerkvariablen austausch zwischen zwei Steuerungen auf einem Hardwaregerät möglich.



- Wenn die austauschenden Geräte als **Sender UND Empfänger** dienen sollen, müssen die Variablenlistenkennungen der Variablenlisten eindeutig sein, um fehlerhaftes Verhalten zu vermeiden! Die Variablenlistenkennungen sind im Dialog **Eigenschaften** eines Objekts GVL definiert.
- Datenaustausch über Netzwerkvariablen ist in folgenden Fällen nicht möglich:
 - Wenn es nicht vom Gerät (Zielsystem) unterstützt wird
 - Wenn eine Firewall die Kommunikation blockiert
 - Wenn ein anderer Client oder eine andere Applikation den UDP-Port benutzt, der in den Eigenschaften der Netzwerkvariablenliste eingestellt ist
- Nur Arrays, deren Grenzen mit einem Literal oder einer Konstante definiert sind, werden zur Empfänger-Applikation übertragen!
Konstante Ausdrücke sind für diesen Zweck nicht erlaubt!
Beispiel:

```
arrVar : ARRAY[0..g_iArraySize-1] OF INT ;
```

wird nicht übertragen;

```
arrVar : ARRAY[0..10] OF INT ;
```

wird übertragen.
- Die maximal erlaubte Größe einer Netzwerkvariable ist 255 Bytes. Die mögliche Anzahl von Netzwerkvariablen ist unbegrenzt.
- Wenn die Größe der globalen Variablenliste (GVL) die maximale Länge des Netzwerktelegramms übersteigt, werden die Daten auf mehrere Telegramme aufgeteilt. Abhängig von der Konfiguration kann dies zu Dateninkonsistenzen führen.



Eine Kommunikation mit Hilfe von Netzwerkvariablen ist auch möglich, wenn die teilnehmenden Steuerungen mit Applikationen aus **verschiedenen Versionen des Programmiersystems** arbeiten (IndraLogic 1x).

In diesem Fall können Sie jedoch für das Anlegen der genau übereinstimmenden Variablenlisten im Sender- und Empfängerprojekt nicht den Export/Import-Mechanismus verwenden.

Grund:

Eine aus IndraLogic 1x erzeugbare Variablen-Exportdatei (*.exp) enthält nicht den erforderlichen Umfang an Informationen, der nötig ist, um in IndraLogic 2G eine empfangende NVL anzulegen. Es fehlt die entsprechende Konfiguration der Netzwerkparameter in Form einer *.gvl-Datei, die Sie zuvor aus dem Sender exportiert haben.

Um diese zu erhalten, müssen Sie die 1x-NVL zuerst in 2G "nachbauen".

Daraus können Sie eine Exportdatei erzeugen und auf ihrer Basis eine empfangende NVL in 2G anlegen.



Eine Alternative zum Datenaustausch zwischen Steuerungen ist die Verwendung von **Datenquellen**. Im Unterschied zur Broadcasting-Methode beim Netzwerkvariablen austausch werden dabei definierte Punkt-zu-Punkt-Verbindungen zwischen einer Applikation und einer entfernten Datenquelle angelegt.

Siehe auch

- [Kap. "Dialog 'Eigenschaften- 'Netzwerkvariablen'" auf Seite 751](#)
- [Kap. 18.1.11 "Objekt 'GVL'- Globale Variablenliste" auf Seite 544](#)

10.3.1 Konfigurieren eines Netzwerkvariablen-Austauschs

1. Anlegen einer Netzwerkvariablenliste im Sender-Gerät und Erzeugen einer Exportdatei

Folgende Schritte sind nötig, um Netzwerkvariablen zwischen Sender-Gerät und Empfänger-Gerät austauschen zu können:

Voraussetzungen: Im Projektextplorer des als Sender eingesetzten Steuerungsgeräts (Sender-Gerät) ist eine Applikation eingefügt.

1. Selektieren Sie die Applikation und fügen Sie ein Objekt **Globale Netzwerkvariablenliste (Sender)** ein.

Stellen Sie im Dialog **Globale Netzwerkvariablenliste (Sender)** hinzufügen Folgendes ein:

Netzwerktyp: UDP, Beispiel:  NVL_Sender.

2. Öffnen Sie mit Doppelklick auf das NVL-Objekt den zugehörigen Editor und tragen die Deklarationen der Netzwerkvariablen ein.

Beispiel für NVL_Sender

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

3. Selektieren Sie das NVL-Objekt im Baum und wählen Sie im Kontextmenü **Eigenschaften**. Öffnen Sie im Dialog **Eigenschaften** die Registerkarte **Netzwerkvariablen**.

Sie zeigt folgende Einstellungen:

Netzwerktyp UDP, Variablenlistenkennung 1, Variablen packen, Zyklische Übertragung alle 50 ms.

- 4.



Sie können auch eine bestehende GVL zur Netzwerkvariablenliste machen, indem Sie ihre Netzwerkvariablen-Eigenschaften entsprechend konfigurieren.

5. Wechseln Sie im Dialog **Eigenschaften** des NVL_Sender zur Registerkarte **Verknüpfung mit Datei**.

Definieren Sie einen Dateinamen `<export>.gvl` und einen Ablageort für die Exportdatei der GVL im Dateisystem.

Aktivieren Sie die Option **Export vor Übersetzen**.

6. Übersetzen Sie die Applikation mit dem Befehl **Erstellen ► Übersetzen**. Die Exportdatei für die Netzwerkvariablenliste liegt nun im definierten Ordner.

Siehe auch

- [Kap. "Dialog 'Eigenschaften- 'Netzwerkvariablen'" auf Seite 751](#)

2. Anlegen einer zugehörige Netzwerkvariablenliste im Empfänger-Gerät

Voraussetzungen:

Zusätzlich zum Sender-Gerät gibt es ein Empfänger-Gerät im Projektexplorer.

Unter beiden Geräten ist eine Applikation mit einer Task-Konfiguration eingefügt.

Unter dem Sender-Gerät ist eine NVL oder GVL als zu sendende Netzwerkvariablenliste angelegt.

1. Selektieren Sie die Applikation des Empfänger-Geräts im Explorer und wählen Sie **Hinzufügen ► Globale Netzwerkvariablenliste (Empfänger)**.

Der Dialog **Globale Netzwerkvariablenliste (Empfänger) hinzufügen** erscheint.

2. Wählen Sie im Dialog die vorher angelegte NVL des Sendergeräts aus und tragen einen Namen ein:

Beispiel:  NVL_Receiver.

IndraLogic füllt diese Empfängerliste automatisch mit den Variablendeklarationen der Senderliste.

Automatisch erzeugt: NVL_Receiver

```
//This global variable list is received via the network.
//Sender: NVL_Sender [IndraMotionMlc1: Logic: Application]
//Protocol: UDP
```

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



Alternativ können Sie im Dialog die Option **Import aus Datei** wählen und die zuvor aus der Sender-Liste erzeugte Exportdatei laden.

3. Testen des Netzwerkvariablen-Austauschs

Voraussetzungen: Es gibt eine Netzwerkvariablenliste (Sender) im Sender-Gerät und eine Netzwerkvariablenliste (Empfänger) mit identischen Variablendeklarationen im Empfänger-Gerät.

1. Erstellen Sie unter der Applikation im Sender-Gerät ein Programm, das eine Netzwerkvariable hochzählt.

Zum Beispiel: NVL_Sender.iglobvar:=NVL_Sender.iglobvar+1;

2. Konfigurieren Sie die Task der Applikation so, dass sie dieses Programm aufruft.
3. Erstellen Sie ein Programm unter der Applikation im Empfänger-Gerät, welches den Wert dieser Netzwerkvariablen auf eine lokale Variable schreibt.

Zum Beispiel: ilocalvar:=NVL_Receiver.iglobvar;

4. Konfigurieren Sie die Task der Applikation so, dass sie dieses Programm aufruft.
5. Laden Sie beide Applikationen auf die Steuerungen und starten Sie diese. (Applikation aktiv setzen, Einloggen, Laden, Starten.)
6. Prüfen Sie in den Online-Ansichten des Editors beider Programme, ob die Werte von iglobvar im Empfänger mit denen im Sender übereinstimmen.

Konfigurieren einer Netzwerkvariablen-Kommunikation zwischen 1.x- und 2G-Steuerungen

Voraussetzungen: In einem 1.x-Projekt liegt eine Globale Variablenliste mit Netzwerkvariablen, die Sie in Ihrem 2G-Projekt lesen möchten.

Im 2G-Projekt gibt es im Projektextplorer unter der Steuerung eine Applikation mit einem Programm, das die Variablen verwendet.

1. Öffnen Sie den Dialog **Eigenschaften** der GVL im 1.x-Projekt (Sender).
2. Fügen Sie im Projekt eine weitere Steuerung als Dummy ein, mit einer Applikation und einer Task. Darunter wird jetzt die 1.x-GVL nachgebaut:
3. Fügen Sie unter der Dummy-Applikation ein Objekt **Globale Netzwerkvariablenliste (Sender)** mit Namen "dummy1x" ein.
4. Öffnen Sie über das Kontextmenü von **dummy1x** den Dialog **Eigenschaften** und wählen die Registerkarte **Netzwerkeigenschaften**. Tragen Sie dort die gleichen Netzwerkeigenschaften ein, wie sie die GVL im 1x-Projekt hat.
5. Füllen Sie **dummy1x** mit den gleichen Variablendeklarationen wie sie in der 1.x-GVL vorliegen.
6. Definieren Sie in den **Eigenschaften** von **dummy1x** in der Registerkarte **Mit Datei verknüpfen** einen Ablageort für eine Exportdatei "dummy1x.gvl".
7. Setzen Sie die Applikation der Dummy-Steuerung aktiv und übersetzen Sie sie (<F11>). IndraLogic legt die Exportdatei an.
8. Setzen Sie die Applikation der realen Steuerung aktiv und wählen Sie im Kontextmenü **Hinzufügen**.
9. Wählen Sie **Globale Netzwerkvariablenliste (Empfänger)** und vergeben Sie einen Namen, z.B. "NVL_from_1x".
10. Aktivieren Sie die Option **Aus Datei öffnen** und wählen dann die vorher erzeugte **dummy1x.gvl** aus.
11. Prüfen Sie die neu angelegte **NVL_from_1x**. Sie muss die gleichen Variablendeklarationen enthalten wie die im 1.x-Projekt liegende Sender-GVL.

Siehe auch

- [Kap. 18.1.15 "Objekt 'Globale Netzwerkvariablen-Liste \(Sender\)'" auf Seite 551](#)
- [Kap. 18.1.16 "Objekt 'Globale Netzwerkvariablen-Liste \(Empfänger\)'" auf Seite 552](#)

10.4 Datenquellenverwalter

Eine Applikation kann mit Hilfe des Datenquellenverwalters eine Punkt-zu-Punkt-Verbindung zu einer oder mehreren entfernten Datenquelle(n) aufbauen.

So können Variablenwerte aus anderen Steuerungsapplikationen in der "lokalen" Applikation verwendet werden, typischerweise in einer Visualisierung.

Der Datenquellenverwalter aktualisiert die Werte der Variablen in konfigurierbaren Zeitintervallen.

Siehe auch

- [Kap. 10.4.1 "Datenquellenverwalter Grundlagen" auf Seite 174](#)
- [Kap. 10.4.2 "Datenquelle initial hinzufügen" auf Seite 175](#)
- [Kap. 10.4.3 "Datenquellenvariablen bearbeiten" auf Seite 179](#)
- [Kap. 10.4.5 "Datenschnittstelle aktualisieren" auf Seite 182](#)

- [Kap. 10.4.6 "Entfernte Daten verwenden" auf Seite 183](#)
- [Kap. 18.1.3 "Objekt 'Datenquellenverwalter'" auf Seite 510](#)

10.4.1 Datenquellenverwalter Grundlagen

Um auf entfernte Geräte und die dort laufenden Applikationen lesend und schreibend zuzugreifen, können Sie Ihre Applikation um einen Datenquellenverwalter mit einer oder auch mehreren Datenquellen erweitern.

Die Funktionalität des Datenquellenverwalters ermöglicht den Verbindungsaufbau und die Kommunikation zu entfernten Geräten und stellt über Datenquellenvariablen deren Daten zur Verfügung. Dabei kommunizieren die Partner über eine Punkt-zu-Punkt-Verbindung.

Während der Entwicklungsphase können Sie mit Hilfe von lokalen Symbolkonfigurationsdateien eine Variablenliste offline erstellen. Somit können Sie offline ohne Verbindung zur Datenquelle auf Basis der Symbolinformation eine lokale Applikation entwickeln.

Siehe [Kap. 18.1.3 "Objekt 'Datenquellenverwalter'" auf Seite 510](#)

Datenquellentyp

Die Kommunikation zwischen entfernten und lokalen Geräten erfolgt über so genannte Datenquellentypen. In IndraLogic steht Ihnen folgender Datenquellentyp zur Verfügung:

"CODESYS ApplikationV3": Der Datenzugriff erfolgt über Adressmonitoring.

Das erfordert, dass die Adressinformationen zwischen entfernter Steuerung und lokalem Gerät zusammenpassen. Nachteil: Bei einer Änderung der entfernten Applikation muss anschließend die lokale Applikation, beispielsweise die HMI-Applikation, aktualisiert werden! Von Vorteil ist, dass keine Symbolkonfiguration in der entfernten Applikation erforderlich ist.



Verwenden Sie diese Kommunikation bei Embedded- oder Kleinststeuerungen, wenn keine Ressourcen für die Symbolkonfiguration zur Verfügung stehen.

Siehe ["Dialog 'Datenquelle hinzufügen'" auf Seite 510](#)

Verbindungstyp

Die Funktionalität des Datenquellenverwalters ermöglicht den Aufbau einer Punkt-zu-Punkt-Verbindung von einem lokalen Gerät zu einem oder mehreren entfernten Geräten, auch über Netzwerke hinweg. Hierzu unterstützt IndraLogic den Verbindungstyp "CODESYS Applikation V3".

Bei diesem Verbindungstyp, benötigt das Laufzeitsystem der lokalen Applikation gültige Kommunikationsparameter, um die Verbindung aufzubauen.

Bei der Konfiguration der Datenquelle können Sie sich von der Funktionalität des Netzwerks-Scans unterstützen lassen.

Siehe [Kap. 18.1.6 "Registerkarte 'Kommunikation'" auf Seite 514](#)

Datenübertragung

Zur Laufzeit der lokalen Applikation werden die Datenquellenvariablen, die im Datenquelleneditor in Registerkarte "Variablen" erscheinen, in konfigurierbaren Zeitintervallen aktualisiert. Die entfernte Applikation wird dabei auch ausgeführt. Variablen, die in der Visualisierung, im Trend, als Alarm oder bei Rezepturen konfiguriert sind, werden automatisch übertragen und gespeichert. Wenn auf eine Variable nur im IEC-Code zugegriffen wird, wird die Variable nicht automatisch aktualisiert. In diesem Fall müssen Sie die Option "Immer aktualisieren" im Datenquelleneditor in Registerkarte "Variablen" aktivieren.

Die Datenquellentypen unterstützen den (lesenden oder schreibenden) Datenzugriff auf Variablen der Quellsteuerung bei folgenden Datentypen:

- Skalarer Wert auf oberster Ebene Beispiel `PLC_PRG.hugo`

- Eigenschaft auf ein Programm oder auf eine GVL über einen Aufruf, wenn dieser mit `{attribute monitoring := ,call'}` markiert ist.
Beispiel: `PLC_PRG.PropertyCall`
- Variable, die auf Bitadressen gemappt ist. Beispiel in `PLC_PRG`: `x AT %MX0.5 : BOOL;`
- Variable (Typ BIT) in einem Funktionsbaustein Beispiel: Deklaration in `DUT`: `x, y : BIT`, Bitzugriff: `PLC_PRG.dutInst.y`
- Strukturiert erreichbare Variable Beispiel: `PLC_PRG.outerInst.innerInst.dwVar`
- Eigenschaft auf Bausteininstanz, wenn diese mit `{attribute monitoring := ,call'}` markiert ist.
Beispiel: `PLC_PRG.instance.PropertyCall`
- Eigenschaft sowohl auf oberster Ebene als auch auf einer Instanz, wenn diese mit `{attribute monitoring := ,variable'}` markiert ist.
Hinweis: Diese kann weder vom Monitoring noch von den Datenquellen beschrieben werden.
- Array-Zugriff mit Litteralem Index Beispiel: `PLC_PRG.arrOfInts[3]`, `PLC_PRG.inst.arrOfBool[1]`
- Verschachtelter Zugriff (beispielsweise Array von Strukturen oder Struktur mit Arrays)

VORSICHT

In Visualisierungen verwendete Bitzugriffe, die über eine Datenquellenverbindung übertragen werden, funktionieren nur, wenn sie literale Offset-Angaben enthalten.

Eine Offsetangabe über definierte Konstanten kann eine Visualisierung nicht verarbeiten.

siehe ["Registerkarte 'Variablen'" auf Seite 511](#)

10.4.2 Datenquelle initial hinzufügen

Für den Datenaustausch Ihres lokalen Geräts mit einem entfernten Gerät fügen Sie im Projektexplorer unter dem Knoten "Application" ein Objekt „Datenquellenverwalter“ und dann ein Objekt „Datenquelle“ hinzu. Ein Assistent führt Sie durch die Konfiguration der Datenquelle. Danach können Sie die Einstellungen im Editor des Objekts jederzeit ändern.

Beim Hinzufügen einer Datenquelle wählen Sie einen Datenquellennamen. Dann geben Sie die Verbindungseinstellungen der Punkt zu- Punkt-Verbindung zu dem entfernten Gerät an. Idealerweise ist währenddessen das entfernte Gerät in Ausführung und IndraLogic kann die Verbindung dorthin sogleich aufbauen. In diesem Fall werden alle verfügbaren Datenquellenvariablen aus der entfernten Steuerung angezeigt. Wählen Sie die Variablen aus, die übertragen werden sollen. Sie können auch alle Variablen auswählen. Dann wird die Datenquelle automatisch initialisiert, die Datenquellenvariablen unter dem Ordner „DataSources_Objects“ erzeugt, und unter dem Datenquellenverwalter eine weitere Datenquelle hinzugefügt.

Im Editor der Datenquelle sind die Initialen Einstellungen jederzeit änderbar.

Siehe [Kap. 18.1.3 "Objekt 'Datenquellenverwalter'" auf Seite 510](#)

Geräte mit Adressmonitoring initial verbinden**Voraussetzungen:**

Das entfernte Gerät und dessen Applikation ist in Ausführung.

Auf ihrem Entwicklungsrechner befindet sich das IndraLogic Projekt des entfernten Geräts.

Die dort projektierte Applikation enthält keine Symbolkonfiguration.

1. Fügen Sie im Projektexplorer unter dem Knoten "Application" ein Objekt "Datenquellenverwalter" ein.
2. Selektieren Sie das Objekt "Datenquellenverwalter" und wählen Sie im Kontextmenü den Befehl **Hinzufügen ► Datenquelle**.

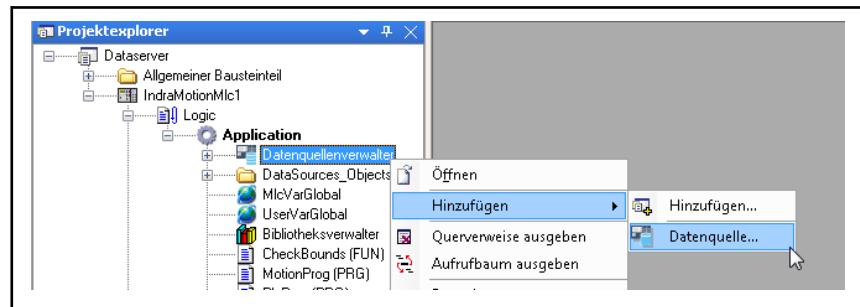


Abb. 10-3:

Der Dialog „Datenquelle hinzufügen“ öffnet sich.

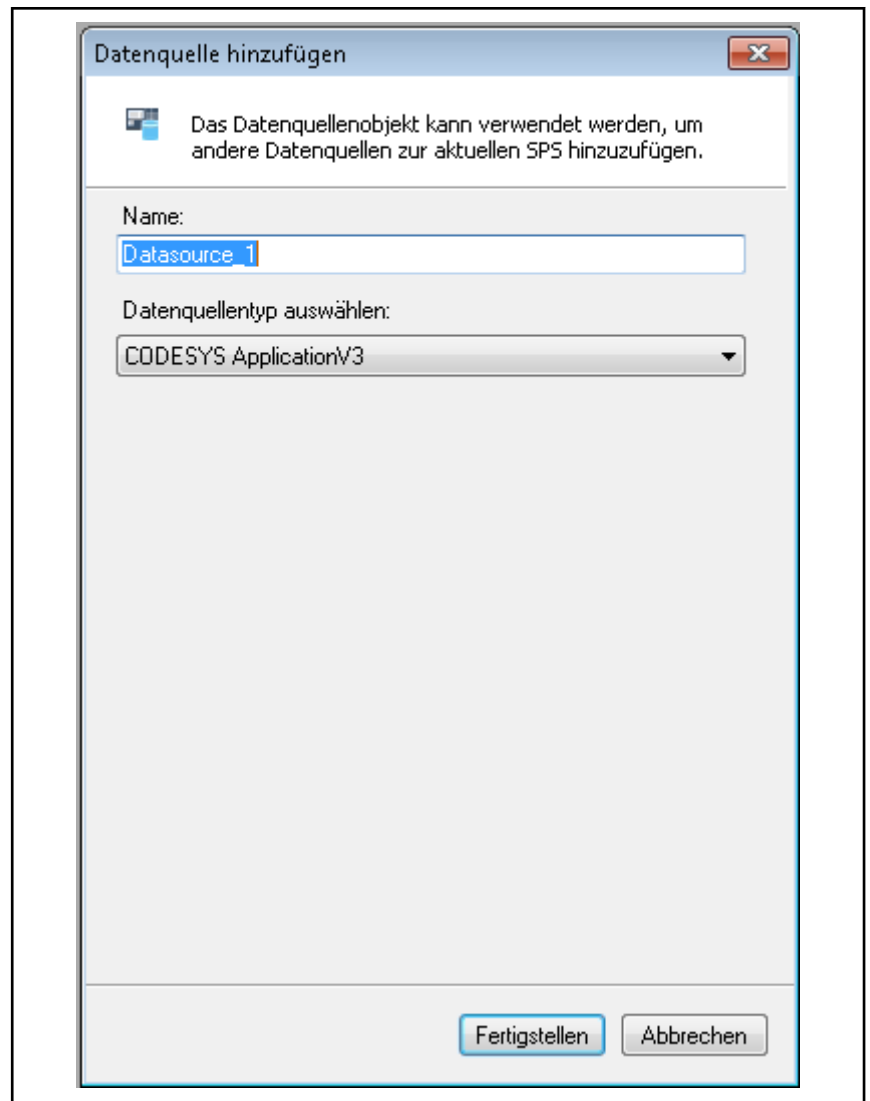


Abb. 10-4:

3. Geben Sie in "Name" den Datenquellennamen an.
Die Auswahl des Datenquellentyps ist auf "CODESYS ApplicationV3" voreingestellt und kann nicht geändert werden.
In dieser Einstellung erfolgt die Datenübertragung per Adressmonitoring.
Bestätigen Sie mit "Fertigstellen".
Der Dialog "Datenquellenassistenten initialisieren - Providereinstellungen" öffnet sich.
4. Wählen Sie bei "Projekttyp auswählen" die Option "Anderes Projekt" (1).

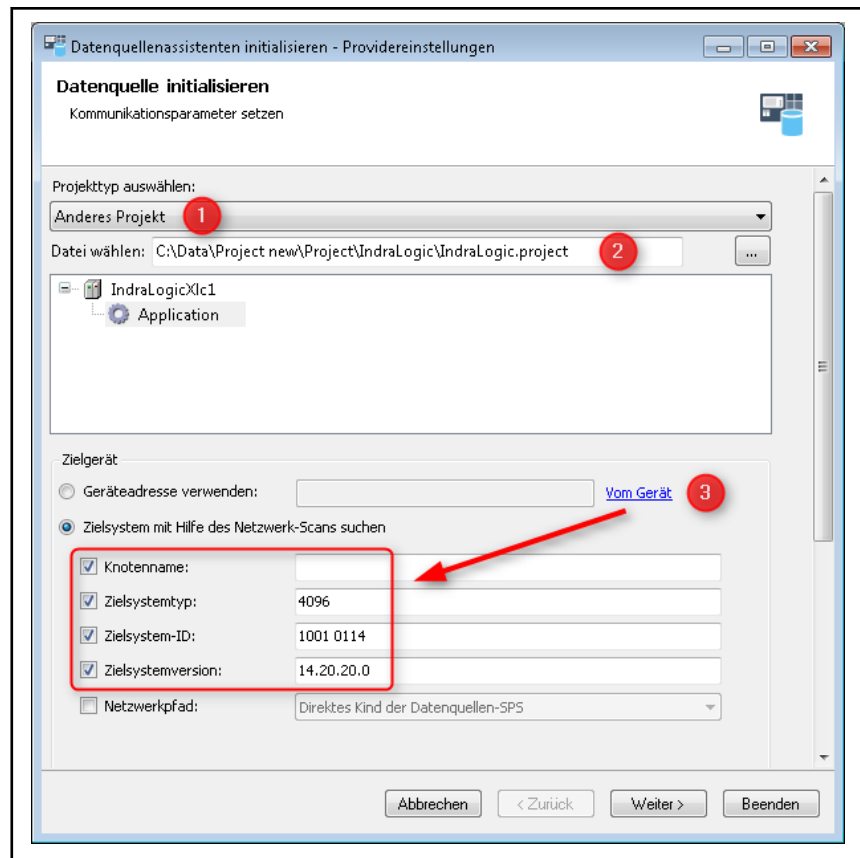


Abb. 10-5:

5. Geben Sie bei "Datei wählen" die Datei und den Speicherort des IndraLogic Projekts des entfernten Geräts an (2).

Beispiel: C:\Data\Projects\Project\IndraLogic\IndraLogic.project.

Hinweis: Das IndraLogic Projekt ist Teil des IndraWorks Projekts

Im Fenster darunter wird in Baumansicht das entfernte Gerät mit Applikationsnamen dargestellt, folglich wurde die Verbindung aufgebaut.

6. Wählen sie den Link "Vom Gerät" (3).

Die Verbindungsparameter zum entfernten Gerät werden eingelesen und im Dialog dargestellt. Die Verbindung ist konfiguriert.

7. Wählen Sie "Weiter >".

Der Dialog "Datenquellenassistenten initialisieren - Datenquellenvariablen durchsuchen" öffnet sich.

Unter dem Eintrag "Variablen" werden die entfernten Steuervariablen in Baumansicht dargestellt.

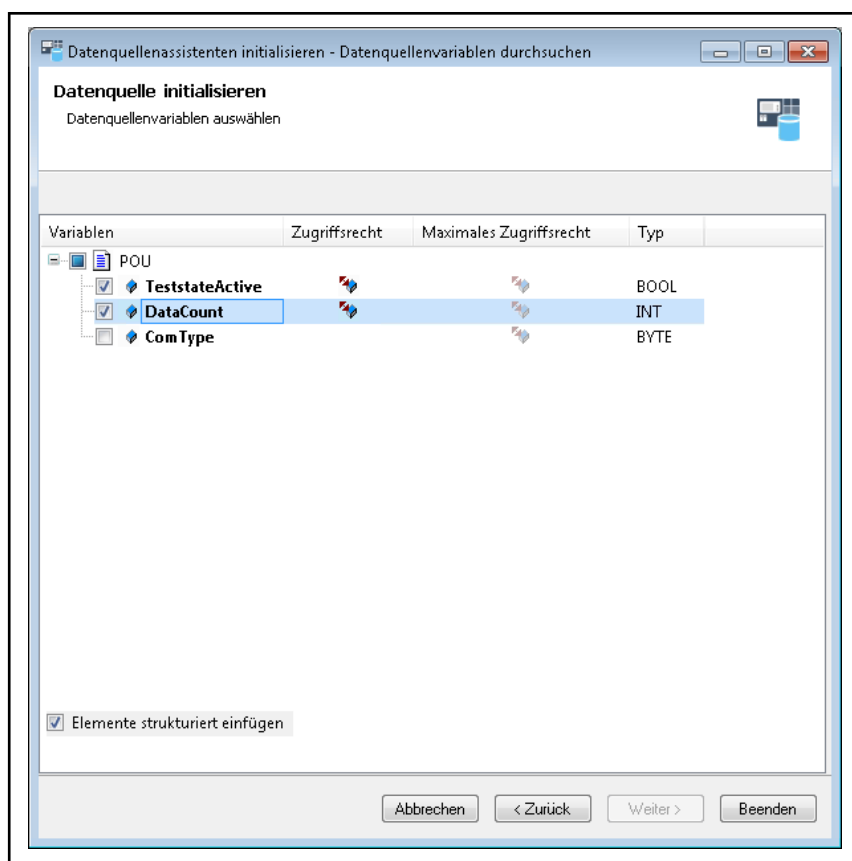


Abb. 10-6:

8. Wählen Sie in der Baumansicht aus welche Steuervariablen übertragen werden sollen und wählen Sie dann "Beenden"

Die Datenquelle ist konfiguriert. Eine Verbindung ist aufgebaut. Die Einstellungen sind im Objekt gespeichert und können im Editor des Objekts geändert werden.

Siehe:

- [Kap. 10 "Mit Steuerungsnetzwerken arbeiten" auf Seite 163](#)
- [Kap. 18.1.6 "Registerkarte 'Kommunikation'" auf Seite 514](#)

10.4.3 Datenquellenvariablen bearbeiten

Zur Laufzeit werden in den Datenquellenvariablen die entfernten Daten gespeichert. Im Datenquelleneditor unter Registerkarte "Variablen" sind die Datenquellenvariablen und deren Zuordnung zur entfernten Variable dargestellt.

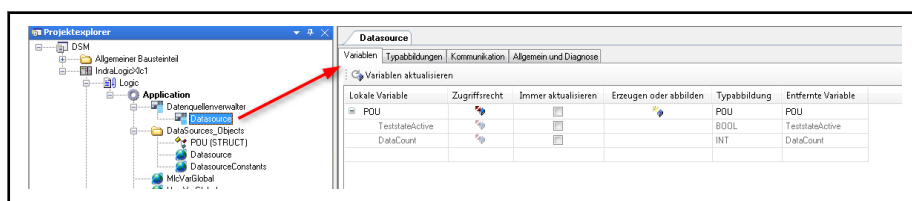


Abb. 10-7:

Wenn die lokalen Variablen den gleichen Namen und den gleichen Datentyp wie die entfernten Variable haben, dann werden die Daten 1:1 abgebildet. Die Variablen und die Datentypen werden automatisch erzeugt. Das ist die übliche Vorgehensweise.

Sie können aber auch auf bereits bestehende Variablen abbilden. Dies ist beispielsweise nötig, wenn eine Visualisierung einen Datentyp in der Schnittstelle enthält. Es muss dann auch der gleiche Datentyp an diese Visualisierung übergeben werden. Dann haben die bereits deklarierte lokale Variable und die entfernte Variable den gleichen Datentyp, beispielsweise aus einer Bibliothek. Außerdem können Sie auf eine entfernte Variable auf eine lokale Variable mit konformen Datentyp abbilden. Den Datentyp können Sie unter Registerkarte „Typabbildungen“ erzeugen.

Unter dem Ordner "DataSources_Objects" sind die eigens erzeugten Variablen und Datentypen deklariert.

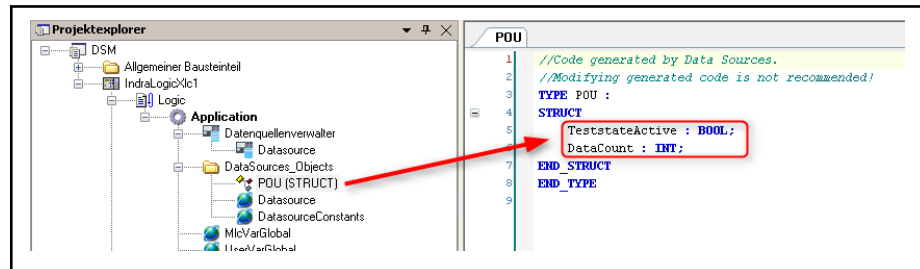


Abb. 10-8:

Dort ist pro Datenquelle eine globale Variablenliste unter dem gleichen Namen wie die Datenquelle deklariert. Außerdem haben die Datenquellenvariablen üblicherweise den gleichen oder wenigstens typkonformen Datentyp wie die entfernte Steuervariable und sind als benutzerdefinierter Datentypen (DUT-Objekte) deklariert. Dabei wird unter Berücksichtigung aller Datenquellen vermieden, dass gleiche Datentypen mehrmals deklariert sind.



Editieren Sie die Datenschnittstelle unter dem "Ordner DataSources_Objects" nicht händisch.

Sie wird beim Hinzufügen einer Datenquelle initial erzeugt.

Änderungen können Sie danach im Editor der Datenquelle vornehmen.

Siehe

- [Kap. 10 "Mit Steuerungsnetzwerken arbeiten" auf Seite 163](#)
- [Kap. 10.4.5 "Datenschnittstelle aktualisieren" auf Seite 182](#)
- [Kap. 18.1.4 "Objekt 'Datenquelle'- Registerkarte 'Variablen'" auf Seite 511](#)

Variablen für Datenübertragung auswählen

Die Auswahl der Datenquellenvariablen kann bearbeitet werden.

Voraussetzungen:

Das entfernte Gerät und dessen Applikation ist in Ausführung.

Unter der lokalen Applikation ist bereits ein Datenquellenverwalter mit einer Datenquelle eingefügt.

1. Öffnen Sie den Editor der Datenquelle.
2. Wählen Sie die Registerkarte "Variablen".
3. Wählen Sie den Befehl "Variablen aktualisieren".
Der Dialog "Variable wählen" erscheint.
4. Aktivieren Sie die Variablen, die übertragen werden sollen und beenden Sie den Dialog mit "OK".

Die Datenquellenvariablen wird entsprechend der Auswahl geändert. Die Deklaration von Variablen und Datentypen ist ebenfalls angepasst. Die Registerkarte "Variablen" stellt die geänderte Auswahl dar. Außerdem ist unter der Spalte "Entfernte Variable" die zugeordnete, entfernte Variable aufgelistet.

Siehe

- [Kap. 10 "Mit Steuerungsnetzwerken arbeiten" auf Seite 163](#)
- [Kap. 10.4.5 "Datenschnittstelle aktualisieren" auf Seite 182](#)
- [Kap. 18.1.4 "Objekt 'Datenquelle'- Registerkarte 'Variablen'" auf Seite 511](#)

Entfernte Variable auf eine lokale Variable mit konformen Datentyp abbilden

Symbol: 

Erzeugen Sie erst einen konformen Datentyp und verwenden Sie ihn dann für eine Variable.

Voraussetzung:

Im Projekt der lokalen Applikation ist ein Datenquellenverwalter, darunter eine Datenquelle.

Im Editor der Datenquelle in Registerkarte "Variable" sind die entfernte Daten, die übertragen werden, dargestellt.

1. Öffnen Sie den Editor der Datenquelle.
2. Wählen Sie die Registerkarte "Typabbildung".
3. Selektieren Sie in der List der Datentypen denjenigen, den Sie bearbeiten wollen.

Im Fenster unter der Datentypenliste sind die Elemente des Datentyps aufgelistet.

4. Wählen Sie für den Datentyp einen Namen.
Wählen Sie für den entfernten Datentypen den Namen zu dem der lokale Typ konform sein soll.
5. Bearbeiten Sie ihn im Fenster unter der Datentypenliste und entfernen die Elemente, die nicht für die Datenübertragung benötigt werden.
6. Wählen Sie für diesen Datentypen unter der Spalte "Erzeugen oder abbilden" das Symbol  aus.

Der Datentyp ist unter dem Ordner "DataSources_Objects" deklariert.

7. Wählen Sie die Registerkarte "Variablen".
8. Geben Sie unter der Spalte "Lokale Variable" einen Namen ein.
9. Wählen Sie unter der Spalte "Erzeugen oder abbilden" das Symbol .
10. Geben Sie unter der Spalte "Typabbildung" den in Schritt 6 erstellten Datentyp an.
11. Wählen Sie die entfernte Variable aus, deren Daten übertragen werden sollen. Verwenden Sie dazu die Eingabeunterstützung.

Die in Schritt 8 definierte, lokale Variable wird automatisch mit dem benutzerdefinierten Datentyp der entfernten Variable deklariert. Bei laufender Datenübertragung enthält sie die Daten der zugeordneten entfernten Variablen.

Der Arbeitsablauf zeigt, wie Variablen der Datenquelle erzeugt werden. Dabei werden neue Variablen erzeugt, Daten auf bereits existierende Datentypen und deren Variablen abgebildet und neue Datentypen mit typkonformer Abbildung erzeugt.

Siehe [Kap. 10 "Mit Steuerungsnetzwerken arbeiten"](#) auf Seite 163

10.4.4 Kommunikation bearbeiten

Die Verbindungsparameter einer Datenquelle können im Datenquelleneditor in der Registerkarte "Kommunikation" bearbeitet werden. Die Statuszeile im unteren Rand des Editors informiert über den Datenquelentyp, das IndraLogic Projektverzeichnis und den aktuellen Verbindungstyp. Der Verbindungstyp kann nachträglich nicht geändert werden.

Siehe

- [Kap. 10.4.2 "Datenquelle initial hinzufügen"](#) auf Seite 175
- [Kap. 18.1.6 "Registerkarte 'Kommunikation'"](#) auf Seite 514

10.4.5 Datenschnittstelle aktualisieren

Die Datenquellenvariablen werden zur Laufzeit zyklisch aktualisiert. Dabei werden nur die Daten aktualisiert, die entweder in der aktuellen Visualisierung verwendet werden oder die die Eigenschaft "Immer aktualisieren" haben.

Sie können das Zeitintervall bestimmen. Außerdem können Sie Variablen bestimmen, deren Daten bei jedem Aktualisierungsintervall übertragen und damit immer aktualisiert werden. Um Variablen zu aktualisieren, die nicht im Applikationscode verwendet werden, können Sie mit Hilfe von Schnittstellenfunktionen des Datenquellverwalter programmatisch eine Aktualisierung implementieren.



Bei zu hohem Datenaufkommen zwischen entferntem und lokalem Gerät wird automatisch die Aktualisierungsrate herabgesetzt. Das kann zu einer unvollständigen Übertragung führen!

Siehe [Kap. 18.1.4 "Objekt 'Datenquelle'- Registerkarte 'Variablen'"](#) auf Seite 511

Aktualisierungsrate setzen

1. Öffnen Sie den Editor der Datenquelle.
2. Wählen Sie die Registerkarte "Allgemein und Diagnose".
3. Geben Sie im Feld "Aktualisierungsrate" einen Wert ein.

Beispiel: 100

Die Daten vom entfernten Gerät zum lokalen Gerät werden alle 100 ms übertragen.

Siehe [Kap. 18.1.7 "Objekt 'Datenquelle'- Registerkarte 'Allgemein und Diagnose'"](#) auf Seite 517

Variable 'Immer aktualisieren' auswählen



Vermeiden Sie, zu viele Variablen immer zu aktualisieren.

Jede Aktualisierung produziert zusätzlichen Datenverkehr auf der Verbindung zwischen entferntem und lokalem Gerät.

Bei zu hohem Datenaufkommen wird automatisch die Aktualisierungsrate herabgesetzt.

Das kann zu einer unvollständigen Übertragung führen!

1. Öffnen Sie den Editor der Datenquelle.

2. Aktivieren Sie für eine Variable die Option "Immer aktualisieren".

Die Daten der Variablen werden bei jedem Aktualisierungszyklus übertragen, auch wenn die Daten sich nicht geändert haben.

Siehe ["Registerkarte Variablen" auf Seite 511](#)

Daten programmatisch aktualisieren

Der Datenquellenverwalter stellt in der Bibliothek `Datasources` Schnittstellenfunktionen zur Verfügung. Wenn ein Datenquellenverwalter im Applikationscode eingebunden ist, wird automatisch die globale Variable `g_Datasources` instanziiert. Sie bietet Zugriff auf die Schnittstellenfunktionen.

Sie können damit einzelne Variablen aktualisieren, die nicht in der gerade aktiven Visualisierung aufgerufen werden.

Beispiel:

Die Variable `ivar` wird über Methoden der Bibliothek `Datasources` aktiviert (und deaktiviert), so dass ihr Wert übertragen wird. Darüber hinaus können Sie konfigurieren, dass die Variable nur über eine definierte Zeitdauer hinweg aktualisiert wird, um Übertragungskapazität zu sparen.

Programm:

```
//Synchronize with DatasourcesTask and block
until access is possible
//Regard the feedback in ERR_OK or in
ERR_DS_MULTITASKING_LOCKED
g_Datasources.BeginDataConfiguration(TRUE);
// Activate variable
g_DataServer.UseData(ADR('RemoteDevice.Applicat
ion.PLC_PRG.iVar'));
// Deactivate variable
//
g_DataServer.ReleaseData(ADR('RemoteDevice.Appl
ication.PLC_PRG.iVar'));
g_DataServer.EndDataConfiguration();
```

Die Datenkonfiguration wird mit `BeginDataConfiguration(TRUE)` gestartet und somit die Synchronisierung der Task `DatasourceTask` mit der Applikationstask initiiert. Mit `TRUE` wird die Abarbeitung blockiert, bis der Zugriff auf die Variable möglich wird, mit `FALSE` werden wiederholte Zugriffsversuche ermöglicht ohne zu blockieren. Die Rückgabewerte `ERR_OK` oder `ERR_DS_MULTITASKING_LOCKED` geben eine Rückmeldung zu den Zugriffsversuchen.

Nach erfolgreicher Synchronisierung wird die Variable mit Hilfe der Methode `UseData` aktiviert. Danach wird die Datenkonfiguration mit der Methode `EndDataConfiguration` abgeschlossen und die Synchronisierung mit der Task `DatasourceTask` wieder gelöst.

Auf gleiche Weise wird die Methode `ReleaseData` verwendet, um die Variable zum gewünschten Abarbeitungszeitpunkt wieder zu deaktivieren.

Siehe [Kap. 10 "Mit Steuerungsnetzwerken arbeiten" auf Seite 163](#)

10.4.6 Entfernte Daten verwenden

Sie können die Variablen, die im Datenquelleneditor in der Registerkarte "Variablen" aufgelistet sind (und unter dem Ordner "DataSources_Objects" deklariert), wie IEC-Variablen in Ihrer Applikation verwenden.

Ein möglicher Anwendungszweck ist die Anzeige der Variable über ein Visualisierungselement.



Wenn mehrere Datenquellen vorhanden sind und dadurch bezüglich der Eindeutigkeit der Variablennamen Konflikte auftreten, müssen Sie den Datenquellenname als Präfix mit angeben. Wenn keine Konflikte auftreten, ist das nicht nötig und Sie können die Variablen ohne ein Datenquellenpräfix zuweisen.

<Datenquellenname>.<Funktionsbausteinname>.<Variablenname>

Variablenwert eines entfernten Geräts in einer Visualisierung ausgeben

Vorgehensweise um den Variablenwert einer entfernten Steuerung (z. B. `iTemp`) in der lokalen Applikation über eine IndraLogic Visualisierung mit einem Visualisierungselement auszugeben (mit Datenquellenverwalter).

Ausgangssituation:

Unter dem lokalen Datenquellenverwalter ist eine Datenquelle "dsRemotePLC" angelegt, in der die Verbindung mit dem entfernten Gerät konfiguriert ist. Die Variable `iTemp` ist außerdem im Datenquelleneditor unter der Registerkarte "Variablen" ausgewählt.

1. Selektieren Sie das Visualisierungselement im Editorfenster. Wählen Sie im Fenster "Eigenschaften" die Eigenschaft "Textvariablen" - "Textvariable".
2. Wählen Sie die Variable `iTemp` aus.
Die Variablenzuweisung ist qualifiziert.
Beispiel: `dsRemotePLC.PLC_PRG.iTemp`.
3. Wählen Sie die Eigenschaft "Text" des Visualisierungselements und tragen Sie Folgendes ein: `Temperature: %s`
Der Wert der Variablen `iTemp` des entfernten Geräts `RemoteDevice` wird ausgegeben.
4. Laden und starten Sie die entfernte Applikation.
5. Laden und starten Sie die lokale Applikation.
Die Visualisierung startet und gibt den Istwert von `iTemp` aus.



Die in IndraLogic integrierte Visualisierung zeigt Istwerte von Variablen nicht an, die über einen Datenquellenverbindung übertragen werden! Die integrierte Visualisierung stellt nur die Initialisierungswerte oder die zuletzt anderweitig gelieferten Werte dar, da sie keine Verbindung zu den Datenquellen aufbaut.



Wenn Variablen verwendet werden, die nicht im Visualisierungscodem aufgerufen werden, dann muss die Aktualisierung der Variablen im Applikationscode über Funktionen der Datenquellenschnittstelle erfolgen.

10.5 Untergeordnete Sicherheitssteuerung

Wenn unter der Standardsteuerung eine Sicherheitssteuerung projektiert ist, erfolgt die Kommunikation und der Datenaustausch zwischen Sicherheitssteuerung und dem Programmiersystem über die Standardsteuerung.

Die Ausführung von Befehlen, welche die Standardsteuerung betreffen, können die Kommunikationsverbindungen der Sicherheitssteuerung unterbrechen!

Hierzu finden Sie bei den jeweiligen Befehlen einen entsprechenden Hinweis.

Mögliche Unterbrechungen

- **Temporäre Unterbrechung:**

Während der Ausführung des Befehls (zum Beispiel Download) werden die Verbindungen mit der Sicherheitssteuerung zuerst unterbrochen, stehen danach aber automatisch wieder zur Verfügung.

Wenn die Unterbrechungszeit zu lang ist, kann es zu sicherheitsgerichteten Reaktionen in den Ausgangsgeräten und verbundenen Netzwerkvariablen-Empfänger-Sicherheitssteuerungen kommen.

Es müssen dann in der Sicherheitssteuerung die entsprechenden Kommunikationsfehler quittiert werden, um die sicherheitsgerichteten Reaktionen zu beenden (falls die Quittierung nicht automatisch erfolgt).

Dies betrifft die Verbindung zu Feldgeräten der Sicherheitssteuerung und Netzwerkvariablen-Empfänger-Verbindungen zu anderen Sender-Sicherheitssteuerungen. Im Fall einer angeschlossenen Sicherheitssteuerung mit Netzwerkvariablen-Sendern müssen die Kommunikationsfehler in den anderen Sicherheitssteuerungen quittiert werden.

- **Dauerhafte Unterbrechung:**

Die Ausführung des Befehls (z. B. Löschen) führt zu einer Unterbrechung, die erst durch eine weitere Aktion (z. B. Download) wieder beendet wird.

Durch die Unterbrechung kann es zu sicherheitsgerichteten Reaktionen in den Ausgangsgeräten und verbundenen Netzwerkvariablen-Empfänger-Sicherheitssteuerungen kommen!

Nach Beendigung der Unterbrechung müssen die entsprechenden Kommunikationsfehler in der Sicherheitssteuerung quittiert werden, um die sicherheitsgerichteten Reaktionen zu beenden (falls die Quittierung nicht automatisch erfolgt).

Bei einer untergeordneten Sicherheitssteuerung läuft das Routing über "<Name der SafetyApp> Mapping". Es kann unter gewissen Umständen vorkommen, dass diese Applikation für den Anwender im Projektexplorer sichtbar ist.

** VORSICHT**

Auf der Applikation "<Name der SafetyApp> Mapping" dürfen keine Befehle ausgeführt werden!

Siehe auch

[Kap. 10 "Mit Steuerungsnetzwerken arbeiten" auf Seite 163](#)

11 Applikation auf die SPS übertragen

Um Ihre Applikation auf die Steuerung zu übertragen, muss das Programm fehlerfrei übersetzt und die Verbindungseinstellungen für die Steuerung konfiguriert sein.

Wenn diese Voraussetzungen erfüllt sind, wird die Applikation beim Einloggen auf die Steuerung geladen.

11.1 Verbindung zur SPS konfigurieren

Die Verbindung zur Steuerung erfolgt über ein so genanntes Gateway. Dieses Gateway kann Ihr Entwicklungsrechner oder sonst ein mit der Steuerung verbundener Rechner im Netzwerk sein.

Die Einstellung des Kommunikationsweges kann beim **Hinzufügen** der Steuerung zum Projekt im Dialog "Schritt 2: Hardware / Kommunikation" oder zu einem späteren Zeitpunkt über **<steuerung> ► Eigenschaften** Registerblatt "Hardware / Kommunikation" erfolgen.



Überprüfen Sie die Verbindung durch **Ausführen** des Verbindungstestes.

Siehe auch

- [Kap. 10.1.2 "Kommunikation im Steuerungsverbund über Gateway" auf Seite 164](#)

11.2 Applikationscode erzeugen

Der Applikationscode bezeichnet den Maschinencode, den eine Steuerung ausführt, wenn Sie eine Applikation starten.

IndraLogic erzeugt den Applikationscode aus dem im Entwicklungssystem geschriebenen Quellcode automatisch vor dem Download der Applikation auf die Steuerung. Dabei wird vor dem Erzeugen des Applikationscodes eine Prüfung der Zuweisungen, der Datentypen und der Verfügbarkeit von Bibliotheken durchgeführt. Weiterhin werden beim Erzeugen des Applikationscodes die Speicheradressen vergeben.

Sie können diesen Vorgang auch explizit über den Befehl **Erstellen ► Code erzeugen** ausführen. Damit können Sie mögliche Fehler in Ihrem Quellcode finden, auch wenn die Steuerung noch nicht verbunden ist. Die Fehler werden im Meldungsfenster in Kategorie "Übersetzen" ausgegeben.

Explizites Erzeugen von Applikationscode

Voraussetzung: Die Applikation lässt sich fehlerfrei übersetzen.

1. Wählen Sie den Befehl **Erstellen ► Code erzeugen**.

Der Applikationscode wird erzeugt. In dem Meldungsfenster werden Detailinformationen zur Speicherbelegung ausgegeben.

Siehe auch

- [Kap. 11.4 "Erzeugen einer Bootapplikation" auf Seite 189](#)
- [Kap. "Befehl 'Code erzeugen'" auf Seite 649](#)

Meldungen beim Erzeugen des Applikationscodes

Wenn Sie den Applikationscode erzeugen, gibt IndraLogic Informationen zur Speicherbelegung in der Meldungsansicht aus. Da aufgrund der inkrementellen Kompilierung der Speicher nur für geänderte und neue Bausteine und Variablen neu vergeben wird, entstehen Lücken im Speicher. Denselben Effekt haben Online Changes. Diese Fragmentierung verringert den verfügbaren

ren freien Speicher. Sie können in diesem Fall den Speicher mit Hilfe des Befehls **Erstellen ► Bereinigen** komplett neu vergeben und somit den freien Speicher wieder vergrößern.

Weitere Informationen zu den Meldungen bei der Codegenerierung: Syntaxfehler und Fehler, die IndraLogic während der Codegenerierung und Speichervergabe feststellt, erscheinen im Meldungsfenster in Kategorie "Übersetzen".

Nach einem "Bereinigen" der Applikation wird der Speicher komplett neu vergeben. In diesem Fall kommt es durch das vorgegebene "Alignment" (in der Regel 8) eventuell zu kleinen Lücken. Größere Lücken entstehen bei der Änderung eines Datums ohne "Bereinigen", beispielsweise Vergrößern eines Arraybereichs. In diesem Fall werden nur die betroffenen Bausteine neu übersetzt. Denselben Effekt haben Online-Changes.

Beachten Sie hierzu auch die Optionen zur Applikationserzeugung.

Siehe auch

- [Kap. "Dialog 'Eigenschaften - 'Optionen Applikationserzeugung'" auf Seite 751](#)

11.3 Applikationscode laden, Einloggen und SPS starten

Um den Quellcode Ihrer Applikation auf die Steuerung zu laden, müssen Sie sich mit der Applikation auf der Steuerung einloggen. Falls Sie mehrere Applikationen (auf mehreren Steuerungen) in Ihrem Projekt haben, müssen Sie die gewünschte Applikation zuerst aktiv schalten.

Wenn Sie eine Applikation in die Steuerung laden, führt IndraLogic folgende Prüfungen durch:

Die Liste der Applikationen auf der Steuerung wird mit den im Projekt verfügbaren verglichen. Bei Nicht-Übereinstimmung werden Sie gefragt, ob Sie die noch nicht auf der Steuerung vorhandenen Applikation laden oder bereits vorhandene Applikationen löschen wollen.

Für "extern implementierte" Bausteine in der zu ladenden Applikation prüft IndraLogic, ob diese auf der Steuerung verfügbar sind. Wenn dies nicht der Fall ist, erscheint eine Meldung ("unaufgelöste Referenz(en)") in einer Meldungsbox und im Meldungsfenster.

Anschließend vergleicht IndraLogic die Parameter (Variablen) der Bausteine in der zu ladenden Applikation mit denen der gleichnamigen Bausteine in der bereits auf der Steuerung vorliegenden Applikation (Signaturen¹⁾-Prüfung). Bei Unstimmigkeiten wird eine entsprechende Meldung ("ungültige Signatur(en)") in einer Meldungsbox und im Meldungsfenster ausgegeben.

Wenn für eine Applikation unter **Eigenschaften**, Registerblatt **Optionen Applikationserzeugung** die Option "Download der Applikationsinfo" aktiviert ist, werden zusätzlich Informationen über den Inhalt der Applikation auf die SPS geladen.

Siehe auch

- [Kap. 18.1.2 "Objekt 'Applikation'" auf Seite 508](#)

Übertragen der Applikation und Starten des Programms

Voraussetzung: Die Applikation ist fehlerfrei und die Kommunikationseinstellungen der Steuerung sind korrekt. Die Applikation befindet sich noch nicht auf der Steuerung.

1. Selektieren Sie die gewünschte Applikation im Projektexplorer. Falls Sie nur eine Applikation haben, machen Sie mit Schritt 3 weiter.

¹⁾ Name/ Typ des Bausteines, Namen/ Typen/ Reihenfolge der Ein- und Ausgänge

2. Wählen Sie den Befehl **Aktive Applikation setzen**.
Die Applikation wird fett dargestellt.
3. Wählen Sie den Befehl **Debuggen ► Einloggen**.
Es erscheint ein Dialog mit der Abfrage, ob Sie die Applikation auf der Steuerung anlegen wollen.
4. Bestätigen Sie den Dialog mit **Ja**.
Die Applikation wird auf die Steuerung geladen.
5. Wählen Sie den Befehl **Debuggen ► Start** oder Drücken Sie **<Alt>+<F5>**.
Die Applikation läuft auf der Steuerung.

11.4 Erzeugen einer Bootapplikation

Eine Bootapplikation ist die Applikation, die automatisch gestartet wird, wenn die Steuerung eingeschaltet oder gestartet wird.

Dazu muss die Applikation auf der Steuerung in einer Datei `<Applikationsname>.app` vorhanden sein.

Für jede auf der Steuerung laufende Applikation kann dort auch eine Bootapplikation abgelegt sein.

Standardmäßig erzeugt IndraLogic die Bootapplikation automatisch beim Download einer Applikation und überträgt sie mit auf die SPS. Die Voreinstellungen zur automatischen Erzeugung liegen in den **Eigenschaften** des Applikationsobjekts in Kategorie **Bootapplikation**. Beim Einloggen mit einer geänderten Applikation erhalten Sie aber immer nochmals die Möglichkeit, zu entscheiden, ob nun eine neue Bootapplikation erzeugt werden soll, oder nicht.

Außerdem können Sie im Online-Betrieb jederzeit explizit eine (neue) Bootapplikation anlegen, über den Befehl **Debuggen ► Bootapplikation erzeugen**.

Auch im Offline-Betrieb können Sie eine Bootapplikation erzeugen und lokal speichern. Diese Applikation können Sie dann mit externen Tools auf die Steuerung kopieren. Somit können Sie eine Applikation auf eine Steuerung übertragen, auch wenn keine Verbindung zu IndraLogic besteht.

Bootapplikation auf der Steuerung automatisch und explizit erzeugen

Voraussetzung: Offline-Betrieb, Ihre Applikation ist fehlerfrei übersetzt. Die Verbindung zur Steuerung ist konfiguriert, die Steuerung läuft. Die Applikation ist die aktive. Folgende Schritte demonstrieren die Möglichkeiten:

1. Wählen Sie den Befehl **Debuggen ► Einloggen**.

Auf der Steuerung wird die Boot-Applikationsdatei `<Applikationsname>.app` angelegt, daneben die Prüfsumme der Bootapplikation `<Applikationsname>.crc`.
2. Wählen Sie nun explizit den Befehl **Debuggen ► Bootapplikation erzeugen**.
Die Dateien auf der Steuerung werden durch neue ersetzt.
3. Loggen Sie aus.
4. Verändern Sie die Applikation. Loggen Sie erneut auf die Steuerung ein.
Sie werden gefragt, ob ein Online-Change durchgeführt werden soll. Im gleichen Dialog sehen Sie die Option **Bootapplikation aktualisieren**.

Standardmäßig ist sie deaktiviert, die Voreinstellung kann in den **Eigenschaften** des Applikationsobjekts in Kategorie **Bootapplikation** verändert werden.

5. Lassen Sie die Option deaktiviert und setzen Sie das Einloggen fort.
Keine neuen Boot-Applikationsdateien werden angelegt.
6. Loggen Sie wieder aus. Schließen Sie das Projekt. Stoppen Sie die Steuerung. Starten Sie die Steuerung erneut.
Auf der Steuerung läuft die oben angelegte Bootapplikation.



Sie können die Bootapplikation verschlüsselt auf der Steuerung ablegen. Die Einstellungen hierzu sind in den **Eigenschaften** des Applikationsobjekts, Kategorie **Optionen Applikationserzeugung** definiert.

Siehe auch

- [Kap. "Dialog 'Eigenschaften - 'Optionen Applikationserzeugung'" auf Seite 751](#)
- [Kap. "Dialog 'Eigenschaften- 'Boot-Applikation'" auf Seite 748](#)
- [Kap. 14.1 "Ausführen eines Online Changes" auf Seite 229](#)
- [Kap. "Befehl 'Bootapplikation erzeugen'" auf Seite 656](#)

Bootapplikation im Offline-Betrieb anlegen

Voraussetzung: Offline-Betrieb, Ihre Applikation ist fehlerfrei übersetzt. Sie möchten für eine Applikation eine Bootapplikation erzeugen und im Dateiverzeichnis ablegen, um sie später mit Hilfe externer Tools auf die Steuerung zu bringen, ohne IndraLogic zu benötigen.

1. Wählen Sie den Befehl **Debuggen ► Bootapplikation erzeugen**.
Ein Dialog zum Angeben eines Speicherorts im lokalen Dateisystem erscheint.
2. Wählen Sie einen Ablagepfad und klicken Sie **Speichern**.
3. Wenn sich die Applikation seit dem letzten Erzeugen der Bootapplikation geändert hat, erscheint eine Abfrage, ob Sie einen neuen Code für die Bootapplikation verwenden möchten. Klicken Sie in diesem Fall auf **Ja**.
Der Dialog **Speichern unter** erscheint.
4. Wählen Sie ein Verzeichnis aus und klicken Sie auf **Speichern**.
Im angegebenen Pfad wird die Boot-Applikationsdatei <Applikationsname>.app angelegt.
Sie werden gefragt, ob die Übersetzungsinformation für die Bootapplikation gespeichert werden soll.
5. Klicken Sie **Ja**.
Die Übersetzungsinformation wird als Datei <Applikationsname.compileinfo> im Projektverzeichnis gespeichert (z. B. C:\ProgramData\IW-Projects\Ix\Project\IndraLogic\IndraLogic.IndraMotionMlc1.Application.*.compileinfo). Sie ist beim nächsten Aktualisieren der Applikation Voraussetzung für einen möglichen Online-Change.



Der Befehl **Erstellen ▶ Bereinigen** löscht diese Datei.

SPS-Quellen zusammen mit Bootapplikation übertragen

Die Übertragung der SPS-Quellen (POUs, Tasks usw.) gemeinsam mit der Bootapplikation auf die Steuerung erfolgt mit dem Ziel, im Online-Vergleich zusätzlich einen Detailvergleich für SPS-Elemente durchführen zu können.

Die Option kann über das Kontextmenü der Steuerung **Eigenschaften ▶ SPS** aktiviert werden.

Siehe auch

- [Kap. "Befehl 'Bootapplikation erzeugen'" auf Seite 656](#)

12 Testen und Fehler beheben

IndraLogic bietet Ihnen verschiedene Möglichkeiten, Ihre Applikation zu testen und Fehler zu finden. So können Sie im Simulationsbetrieb Ihre Applikation auch ohne angeschlossene Hardware starten. Mit Haltepunkten und Befehlen zum schrittweisen Abarbeiten des Programms können Sie ganz gezielt bestimmte Programmstellen untersuchen. Durch das Schreiben von Variablenwerten können Sie das laufende Programm beeinflussen.

Es stehen Ihnen Befehle zur Verfügung, die Ihre Applikation unterschiedlich stark zurücksetzen. Dies reicht vom **Reset warm** (Alle Variablen außer RETAIN- und PERSISTENT-Variablen werden zurückgesetzt) über **Reset kalt** (Alle Variablen außer PERSISTENT-Variablen werden zurückgesetzt) bis zum **Reset Ursprung** (Alle Variablen werden zurückgesetzt).

Siehe auch

- [Kap. 12.1 "Testen im Simulationsbetrieb" auf Seite 193](#)
- [Kap. 12.2 "Verwenden von Haltepunkten" auf Seite 193](#)
- [Kap. 12.3 "Schrittweises Abarbeiten eines Programms \(Stepping\)" auf Seite 196](#)
- [Kap. 12.5 "Reset der Applikation durchführen" auf Seite 200](#)

12.1 Testen im Simulationsbetrieb

Verwenden Sie den Simulationsbetrieb zum Testen und Debuggen Ihres Programms, wenn Sie kein reales Zielgerät zur Verfügung haben. In diesem Modus wird die Applikation auf einem simulierten Zielgerät gestartet.

Der Befehl ist nur im ausgeloggten Zustand verfügbar.

Voraussetzung: Sie haben ein fehlerfreies Programm (Compiler-Fehlermeldungen oder Übersetzungsfehler) und sind nicht eingeloggt.

1. *Aktivieren Sie den Simulationsbetrieb:*

- Selektieren Sie die Steuerung im Projektexplorer und wählen Sie aus dem Kontextmenü den Befehl **Simulation**.

In der Statuszeile erscheint "Simulation" mit rotem Hintergrund.
Der Menübefehl **Simulation** ist angehakt.

2. Wählen Sie den Befehl **Debuggen ► Einloggen**.

3. Beim ersten Einloggen mit der aktiven Applikation erfolgt die Abfrage, ob die Applikation "Sim.<devicename>.<applicationname>" angelegt und geladen werden soll. Bestätigen Sie mit **Ja**.

Die Applikation ist auf der Steuerung eingeloggt.

4. Sie können nun den Programmablauf prüfen und korrigieren. Dazu stehen Ihnen entsprechende Befehle im Menü **Debuggen** zur Verfügung.

5. Loggen Sie sich aus der Steuerung aus und beenden Sie den Simulationsbetrieb.

12.2 Verwenden von Haltepunkten

Haltepunkte (Breakpoints) werden üblicherweise zur Fehlersuche im Programm genutzt. IndraLogic unterstützt Haltepunkte in allen IEC-Editoren.

Sie können Haltepunkte an bestimmten Positionen im Programm setzen, um dort einen Ausführungsstopp zu erzwingen und die Variablenwerte zu beobachten.

Der Stopp am Haltepunkt kann an zusätzliche Bedingungen geknüpft werden.



Die Ansicht **Haltepunkte** gibt eine Übersicht über alle definierten Haltepunkte. Darin stehen Ihnen zusätzliche Befehle zum gleichzeitigen Ändern mehrerer Haltepunkte zur Verfügung.

Im Editor wird der Status von Haltepunkten mit folgenden Symbolen markiert:

- Haltepunkt aktiviert
- Haltepunkt deaktiviert
- Stop an Haltepunkt
- Haltepunkt mit Bedingung aktiviert
- Haltepunkt mit Bedingung deaktiviert

Siehe auch

- [Kap. "Befehl 'Haltepunkte'" auf Seite 664](#)

Haltepunkte in Applikationen mit mehreren Tasks

Grundsätzlich gilt: Debugging ist nicht auf mehreren Tasks gleichzeitig möglich. Solange Sie auf einer Task mit Haltepunkten oder schrittweiser Abarbeitung arbeiten, werden Haltepunkte in anderen Tasks ignoriert.

Wenn ein Baustein, in dem ein Haltepunkt gesetzt ist, von mehreren Tasks verwendet wird, dann wird nur die Task ("Debug-Task") angehalten, die zuerst auf den Haltepunkt trifft. Alle anderen Tasks laufen weiter.

Im Dialog **Aufrufliste** sehen Sie, welche Task gerade angehalten wird.

Wenn Sie wollen, dass nur eine bestimmte Task von einem Haltepunkt betroffen sein soll, können Sie dies in den Haltepunkteigenschaften festlegen.

Haltepunkte funktionieren für jede Applikation separat, so dass ein "Halt auf BP" keine andere Applikation betrifft.



Die E/As, die von der "Debug-Task" bedient werden, werden bei einem Stop im Haltepunkt nicht aktualisiert, auch wenn Sie die Option **E/As aktualisieren im Stop** in den SPS-Einstellungen aktiviert haben.



Wenn die Applikation auf der SPS an einem Haltepunkt steht, erzeugt ein Online Change oder Download ein Stoppen aller Tasks, was ein Stop der SPS bedeutet!

In diesem Fall zeigt IndraLogic eine entsprechende Meldung an, und Sie können entscheiden, ob Sie mit dem Login fortfahren wollen oder nicht.

Siehe auch

- [Kap. "Befehl 'Aufrufliste'" auf Seite 667](#)

Setzen eines einfachen Haltepunkts (Beispiel ST-Editor)

Voraussetzung: Die Applikation ist im Online-Betrieb und läuft. Der Operating Mode ist **Debuggen**.

1. Öffnen Sie eine POU in der Sprache ST im Editor.

2. Setzen Sie den Cursor in die Zeile, in der ein Haltepunkt gesetzt werden soll.
3. Wählen Sie den Befehl **Debuggen ► Haltepunkt umschalten** oder drücken Sie die Tasten <ALT>+<F9>.

Die Zeile wird rot markiert und mit dem Icon  (Haltepunkt aktiviert) gekennzeichnet. Wenn das Programm an dem Haltepunkt steht, wird die Zeile mit dem Icon  (Stop an Haltepunkt) gekennzeichnet. Die Abarbeitung des Programms stoppt. Dies wird in der Statuszeile mit dem rot hinterlegten Status `HALT AUF BP` signalisiert.

4. Wählen Sie den Befehl **Debuggen ► Start** oder drücken Sie die Taste <ALT>+<F5>.

Das Programm läuft weiter.

5. Setzen Sie weitere Haltepunkte und überprüfen Sie an den Haltepositionen die Werte von Variablen.
6. Setzen Sie den Cursor in eine Zeile, in der ein Haltepunkt gelöscht werden soll.
7. Wählen Sie den Befehl **Debuggen ► Haltepunkt umschalten** oder drücken Sie die Taste <ALT>+<F9>.

Die Markierung verschwindet. Der Haltepunkt ist gelöscht.

Siehe auch

- [Kap. "Befehl 'Haltepunkt umschalten'" auf Seite 666](#)

Festlegen einer Haltepunkt-Bedingung (Beispiel ST-Editor)

1. Öffnen Sie eine POU in der Sprache ST im Editor.
2. Setzen Sie den Cursor in die Zeile, in der ein Haltepunkt gesetzt werden soll.
3. Wählen Sie den Befehl **Debuggen ► Neuer Haltepunkt**.
Es öffnet sich der Dialog **Neuer Haltepunkt**.
4. Wählen Sie die Registerkarte **Bedingung**.
5. Wählen Sie im Abschnitt **Trefferzahl** die Option **Anhalten, wenn die Trefferzahl ein Vielfaches ist von**.
Geben Sie im Feld rechts daneben den Wert 5 ein.
6. Definieren Sie zusätzlich noch eine boolesche Bedingung, wann der Haltepunkt aktiv sein soll. Aktivieren Sie dazu die Option **Halt, wenn TRUE**. Geben Sie im Feld rechts daneben eine boolesche Variable ein.
7. Aktivieren Sie die Option **Haltepunkt sofort aktivieren**.
8. Schließen Sie den Dialog.

Die Zeile wird rot markiert und mit dem Icon  gekennzeichnet

Beobachten Sie nun das laufende Programm. Solange die boolesche Variable für die Bedingung `FALSE` ist, ist die Bedingung für den Haltepunkt nicht erfüllt und das Programm läuft. Wenn Sie die Variable auf `TRUE` setzen, ist die Bedingung erfüllt und das Programm bleibt bei jedem 5. Durchlauf an diesem Haltepunkte stehen.

Siehe auch

- [Kap. 18.3.2 "Dialog 'Eigenschaften Haltepunkt'" auf Seite 745](#)

12.3 Schrittweises Abarbeiten eines Programms (Stepping)

Mit Hilfe des "Stepping" können sie eine Applikation Schritt für Schritt ausführen lassen und dabei durch den Programmcode navigieren. Die Abarbeitung in Schritten ist hilfreich, um den Status des Programmcodes zur Laufzeit zu ermitteln. Sie können dabei z. B. den Aufrufablauf untersuchen, Variablenwerte verfolgen oder Fehler ermitteln.

Zur schrittweisen Abarbeitung stehen im Menü "Debuggen" Schrittbefehle zur Verfügung. Die Befehle stehen erst dann zur Verfügung, wenn während des Onlinebetriebs Haltepunkte gesetzt sind und die Ausführung an einem Haltepunkt anhält:

→ Die Applikation ist im Zustand "HALT AUF BP" (Debugbetrieb). Während des Debugbetriebs wird die aktuelle Halteposition gelb hinterlegt und in den Texteditoren mit dem Symbol  gekennzeichnet.

In den Debugbetrieb wechseln

1. Loggen Sie die Applikation im Projektextplorer ein.
Falls erforderlich wird dabei der Applikationscode auf die Steuerung geladen.
→ Die Applikation ist jetzt im Onlinebetrieb. Der Betriebszustand der Steuerung wird im Projektextplorer angezeigt.
2. Setzen Sie in den Programmierbausteinen Haltepunkte an den Stellen im Code, die Sie untersuchen wollen.
3. Starten Sie die Applikation im Projektextplorer über den Kontext-Menübefehl **Start**
→ Die Applikation startet, der Code wird abgearbeitet bis zum ersten Haltepunkt.
Die Applikation ist jetzt im Debugbetrieb.
Im Projektextplorer ist die Applikation mit "[Halt auf BP]" gekennzeichnet.
Die Statuszeile informiert über den Betriebszustand:

HALT AUF BP

Der Editor mit der aktuellen Halteposition ist geöffnet. Die Codezeile mit dem aktiven Haltepunkt, an dem die Programmausführung angehalten wurde, ist gelb hinterlegt und mit dem Symbol  gekennzeichnet. Die gelb gekennzeichnete Anweisung wurde noch nicht ausgeführt.
Jetzt können Sie Schrittbefehle wählen, oder sich den Aufrufbaum anzeigen lassen.



Alternativ können Sie erst die Applikation starten und dann einen Haltepunkt setzen.

Verhalten der Schrittbefehle des Menüs "Debuggen"

Die folgende Tabelle zeigt die verfügbaren Schrittbefehle und deren Auswirkung auf die Bearbeitung des Programmcodes.

Prozedurschritt (step over) Symbol: 	Die Anweisung an der Halteposition wird ausgeführt. Vor der nächsten Anweisung im Programmierbaustein wird angehalten. Wenn in der Anweisung ein Aufruf enthalten ist (von einem Programm, einer Funktionsbaustein-Instanz, einer Funktion, einer Methode, oder einer Aktion), wird der untergeordnete Programmierbaustein in einem Schritt vollständig durchlaufen.
Einzelschritt (step into) Symbol: 	Die Anweisung an der Halteposition wird ausgeführt. Vor der nächsten Anweisung wird angehalten. Wenn in der Anweisung ein Aufruf enthalten ist (von einem Programm, einer Funktionsbaustein-Instanz, einer Funktion, einer Methode, oder einer Aktion), wird in diesen untergeordneten Programmierbaustein gesprungen. Die erste Anweisung im untergeordneten Programmierbaustein wird ausgeführt und vor der nächsten Anweisung wird angehalten. Die neue aktuelle Halteposition ist dann im aufgerufenen Programmierbaustein.
Ausführen bis Rücksprung (step out) Symbol: 	Der Befehl führt den Programmierbaustein von der aktuellen Halteposition bis zum Bausteinende aus und springt dann zurück in den aufrufenden Programmierbaustein. An der Aufrufstelle wird angehalten (in der Zeile mit dem Aufruf). Wenn die aktuelle Halteposition im Hauptprogramm ist, wird der Programmierbaustein bis zum Ende durchlaufen. Dann wird zurück an den Anfang gesprungen und dort angehalten (an den Programmstart an die erste Codezeile im Programmierbaustein).
Ausführen bis Cursor (run to cursor) Symbol: 	Setzen Sie zunächst den Cursor an eine beliebige Codezeile und wählen Sie dann den Befehl. Das Programm wird ab der aktuellen Halteposition ausgeführt und hält an der aktuellen Cursorposition an, ohne den Code dieser Zeile auszuführen.
Nächste Anweisung festlegen (set next statement) Symbol: 	Setzen Sie zunächst den Cursor an eine beliebige Codezeile (auch vor der aktuellen Halteposition) und wählen Sie dann den Schrittbefehl "Nächste Anweisung festlegen". Die mit dem Cursor gekennzeichnete Anweisung wird als nächstes ausgeführt. Alle Anweisungen dazwischen werden ignoriert und übersprungen.
Nächste Anweisung anzeigen (show next statement) Symbol: 	Wenn Sie die aktuelle Halteposition nicht sehen, wählen Sie den Schrittbefehl "Nächste Anweisung anzeigen". Dann wird das Fenster mit der aktuellen Halteposition aktiv und die Halteposition sichtbar.

Tab. 12-1: IndraLogic - unterstützte Schrittbefehle

Um den vollständigen bisherigen Aufrufpfad für die gerade in der Programmarbeitung erreichte Position anzuzeigen, wählen Sie den Menübefehl **Debuggen ► Aufrufliste**



Die Ansicht "Aufrufliste" zeigt jederzeit wo sich der Baustein in der Aufrufstruktur des Programms befindet, auch bereits vor dem Kompilieren (Übersetzen) der Applikation.

Siehe auch

- [Kap. "Befehl 'Einzelschritt'" auf Seite 669](#)
- [Kap. "Befehl 'Prozedurschritt'" auf Seite 669](#)
- [Kap. "Befehl 'Ausführen bis Rücksprung'" auf Seite 669](#)
- [Kap. "Befehl 'Ausführen bis Cursor'" auf Seite 670](#)
- [Kap. "Befehl 'Nächste Anweisung festlegen'" auf Seite 670](#)
- [Kap. "Befehl 'Nächste Anweisung anzeigen'" auf Seite 670](#)
- [Kap. "Befehl 'Aufrufbaum'" auf Seite 631](#)

12.4 Forcen und Schreiben von Variablen

Sie können in IndraLogic den Wert von Variablen auf der Steuerung im Online-Betrieb verändern. Dabei wird unterschieden zwischen Forcen und Schreiben.

Beim Forcen schreibt IndraLogic den Wert in jedem Zyklus, so dass die Variable permanent auf dem geforcen Wert gehalten wird.



Bitte beachten Sie:

1. Geforcte Ausgänge halten ihren Wert auch im Stop und nach dem Auftreten einer IEC-Exception
2. Geforcte Variablen müssen explizit durch den Anwender aufgehoben werden

Beim Schreiben setzt IndraLogic den Wert einmalig. Der Wert kann somit jederzeit durch das Programm überschrieben werden.

Falls beim Ausloggen noch geforcte Variablen vorhanden sind, erscheint ein Dialog.

Forcen im Deklarationsteil

Voraussetzung: Ihre Applikation enthält einen Baustein mit Deklarationen. Die Applikation befindet sich im Onlinebetrieb.

1. Öffnen Sie den Baustein im Editor.
2. Führen Sie im Deklarationsteil des Editors einen Doppelklick in der Spalte **Vorbereiteter Wert** einer Variablen aus ①.

Das Feld wird editierbar und Sie können einen Wert eingeben.

Device.Application1.PLC_PRG			
Expression	Type	Value	Prepared value
iColorR	INT	0	100 ①
iColorY	INT	0	200
iColorG	INT	0	300

Abb. 12-1:

3. Führen Sie den Schritt 2 für weitere Variablen aus.
4. Wählen Sie den Befehl **Debuggen ► Werte forcen**.

Die Werte der Variablen werden mit den vorbereiteten Werten überschrieben. Die Werte sind mit dem Symbol gekennzeichnet.

Device.Application1.PLC_PRG			
Expression	Type	Value	Prepared value
iColorR	INT	100	
iColorY	INT	200	
iColorG	INT	300	

Abb. 12-2:



Sie können Werte von Variablen auch in der Ansicht **Überwachen** forcen.

Forcen im Implementierungsteil

Voraussetzung: Die Applikation befindet sich im Onlinebetrieb.



Bei eingeschalteter Ablaufkontrolle (**Debuggen** ► **Ablaufkontrolle**) ist die Wertvorbereitung weder für Schreiben noch für Forcen in der Implementation möglich.

1. Öffnen Sie den Baustein im Editor.
2. Führen Sie im Implementierungsteil des Editors einen Doppelklick auf ein Inline-Monitoring-Feld aus ①.
Es öffnet sich der Dialog **Werte vorbereiten**.

3. Geben Sie im Feld **Einen neuen Wert für die nächste Schreib- oder Force-Operation vorbereiten** den neuen Wert ein.

Der vorbereitete Wert erscheint im Inline-Monitoring-Feld (1).

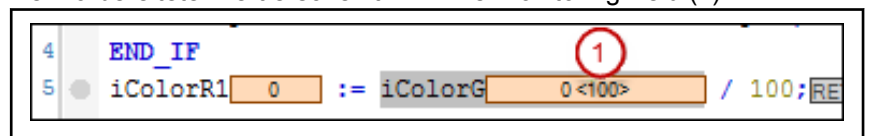


Abb. 12-3:

4. Wählen Sie den Befehl **Debuggen** ► **Werte forcen**.
Der Wert der Variablen wird mit den vorbereiteten Werten überschrieben. Die Werte sind mit dem Symbol **F** gekennzeichnet.

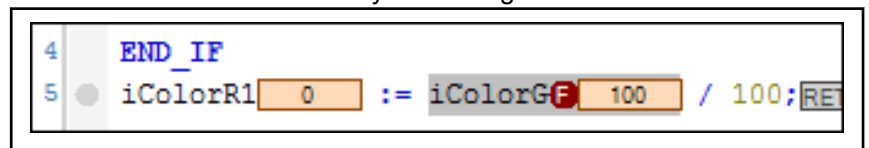


Abb. 12-4:

Alle geforcen Variablen in einer Liste sehen und bearbeiten

Voraussetzung: Die Applikation befindet sich im Onlinebetrieb. Mehrere Variablen sind im geforcen Zustand.

1. Wählen Sie Befehl **Debuggen** ► **Überwachung** ► **Alle Forces anzeigen**.
Die Ansicht **Alle Forces anzeigen** erscheint. Sie enthält in der Form einer Überwachen-Liste alle aktuell geforcen Variablen der Applikation.
2. Selektieren Sie alle Zeilen der Liste und wählen Sie in der Auswahlliste links oben in der Ansicht **Force aufheben** ► **Forcen aufheben und alle ausgewählten Werte beibehalten**.

Das Forcen für die Variablen wird aufgehoben, sie erhalten die Werte, die sie vor dem Forcen hatten.

Siehe auch

- [Kap. 18.3.3 "Dialog 'Wert vorbereiten'" auf Seite 745](#)
- [Kap. "Befehl 'Werte forcen'" auf Seite 671](#)
- [Kap. "Befehl 'Forcen für alle Werte aufheben'" auf Seite 672](#)
- [Kap. "Befehl 'Werte schreiben'" auf Seite 672](#)
- [Kap. 13.1.2 "Überwachungslisten verwenden" auf Seite 211](#)

12.5 Reset der Applikation durchführen

Ein Reset der Applikation stoppt das Programm und setzt die Variablen auf ihren Initialisierungswert zurück. Abhängig von der Art des Reset werden auch Retain-Variablen und Persistent-Variablen zurückgesetzt.

- **Reset warm:** Alle Variablen außer RETAIN- und PERSISTENT-Variablen werden zurückgesetzt
- **Reset kalt:** Alle Variablen außer PERSISTENT-Variablen werden zurückgesetzt
- **Reset Ursprung:** Alle Variablen werden zurückgesetzt

Das kleine Beispielprogramm und die nachfolgende Handlungsanweisungen verdeutlichen Ihnen das Verhalten der verschiedenen Resets.

Siehe auch

- [Kap. 17.2.11 "Persistente Variable - PERSISTENT" auf Seite 308](#)

Beispielprogramm

Deklaration

```
VAR
    iVar: INT := 13;
END VAR
VAR RETAIN
    iVarRetain: INT := 14;
END VAR
VAR PERSISTENT
    iVarPersistent : INT:= 15;
END VAR
```

Implementierung

```
iVar := iVar + 1;
iVarRetain := iVarRetain + 1;
iVarPersistent := iVarPersistent + 1;
```

1. Fügen Sie das Objekt **PersistenteVars** unterhalb der Applikation ein und öffnen es im Editor.
2. Führen Sie den Befehl **Erstellen ► Übersetzen** aus.
3. Führen Sie den Befehl **Deklaration ► Alle Instanzpfade hinzufügen** aus. Der Instanzpfad der persistenten Variablen wird hinzugefügt.
4. Laden Sie die Applikation auf die Steuerung.

Voraussetzung: Das Beispielprogramm läuft auf der Steuerung.

1. Wählen Sie den Befehl **Debuggen ► Einloggen**, um in den Online-Betrieb zu wechseln.
2. Starten Sie Ihr Programm (**Debuggen ► Start** oder Kontextmenü der Applikation)
3. Beobachten Sie die Variablen iVar, iVarRetain und iVarPersistent.
4. Wählen Sie den Befehl **Debuggen ► Reset warm**.
Es erscheint eine Abfrage, ob Sie den Befehl wirklich durchführen wollen.
5. Bestätigen Sie den Dialog mit **Ja**.

Ausführen eines "Reset warm", "Reset kalt" und "Reset Ursprung"

Die Applikation ist zurückgesetzt. Die Variable `iVar` wird auf den Initialisierungswert (im Beispiel 13) gesetzt. Die beiden anderen Variablen behalten ihren Wert.

6. Starten Sie Ihr Programm erneut.
7. Wählen Sie den Befehl **Debuggen ▶ Reset kalt**.
Es erscheint eine Abfrage, ob Sie den Befehl wirklich durchführen wollen.
8. Bestätigen Sie den Dialog mit **Ja**.
Die Applikation ist zurückgesetzt. Die Variable `iVar` und `iVarRetain` werden auf den Initialisierungswert gesetzt (im Beispiel 13 und 14). Die Variablen `iVarPersistent` behält ihren Wert.
9. Starten Sie Ihr Programm erneut.
10. Wählen Sie den Befehl **Debuggen ▶ Reset Ursprung**.
Es erscheint eine Abfrage, ob Sie den Befehl wirklich durchführen wollen.
11. Bestätigen Sie den Dialog mit **Ja**.
Die Applikation wird zurückgesetzt (Not loaded). Die Werte der Variablen können nicht mehr gelesen werden. Das Programm muss neu geladen werden (**Debuggen ▶ Laden**) und kann danach wieder gestartet werden (**Debuggen ▶ Start**).

Siehe auch

- [Kap. "Befehl 'Reset Warm'" auf Seite 661](#)
- [Kap. "Befehl 'Reset Kalt'" auf Seite 662](#)
- [Kap. "Befehl 'Reset Ursprung'" auf Seite 663](#)

12.6 Ablaufkontrolle

Mit der Ablaufkontrolle können Sie die Abarbeitung des Applikationsprogramms verfolgen. Die Ablaufkontrolle ist für die Spracheditoren ST, FUP, KOP und AWL verfügbar.

Mit aktivierter Ablaufkontrolle stellt IndraLogic die Werte von Variablen, die Ergebnisse von Funktionsaufrufen und Operationen an der jeweiligen Abarbeitungsposition und zum jeweiligen Abarbeitungszeitpunkt dar. Dabei werden exakt diejenigen Codezeilen bzw. Netzwerke farblich markiert, die im aktuellen Zyklus durchlaufen werden.

Zum Vergleich:

Beim Standard-Monitoring liefert IndraLogic nur den Wert, den eine Variable zwischen zwei Abarbeitungszyklen hat.

Die Ablaufkontrolle arbeitet in allen gerade sichtbaren Teilen der gerade geöffneten Editorfenster. Dabei wird **Ablaufkontrolle aktiviert** in der Statuszeile angezeigt, solange die Funktion aktiv ist und in einem Editorfenster Ablaufkontrollpositionen (durchlaufene Teile des Codes) sichtbar sind.

Sie können Werte nur im Deklarationsteil schreiben und forcen. (Durch einen Doppelklicke auf das Werteanzeigefeld wird der Dialog "Wert vorbereiten" geöffnet, wo der gewünschte Wert eingegeben werden kann).



Wenn Sie die Ablaufkontrolle aktivieren, verlängert sich die Laufzeit der Applikation!

Das Verwenden von Haltepunkten und die schrittweise Abarbeitung des Programms sind bei aktivierter Ablaufkontrolle nicht möglich.

Siehe auch

- [Kap. "Befehl 'Ablaufkontrollmodus umschalten'" auf Seite 673](#)

Darstellung der Ablaufkontrolle in den verschiedenen Spracheditoren:

Standardmäßig stellt IndraLogic die Ablaufkontrollpositionen der durchlaufenen Codeteile als grüne Felder dar. Nichtdurchlaufene Codeteile werden weiß dargestellt.



Beachten Sie, dass der angezeigte Wert einer nicht-durchlaufenen Codeposition ein "normaler" Monitoringwert ist. Dies ist der Wert, der zwischen zwei Task-Zyklen vorliegt.

```

1 i 1619 := i 1619 + 1;
2 b 0 := NOT b 0;
3 IF str 'abcdefghij' = str1 ' ' THEN
4 f12 1.5 := f1 1.23;
5 ELSE
6 f12 1.5 := 1.5;
7 D 6.5E+04 := B255 * B255;
8 END_IF;
9 IF D 6.5E+04 < 0.0 THEN

```

Abb. 12-5:

In Netzwerkeditoren markiert IndraLogic die durchlaufenen Netzwerke am linken Rand durch Balken in der 'Ablaufkontrollfarbe'.

Im KOP stellt IndraLogic die aktuell durchlaufenen Verbindungslinien in grün dar, die anderen in grau. Der aktuelle Wert der Verbindung wird ebenfalls dargestellt: TRUE durch fette blaue, FALSE durch fette schwarze Linien, unbekannte oder analoge Werte durch dünne schwarze Linien. Das kann durch Kombination der jeweiligen Informationen zu gestrichelten Linien führen.

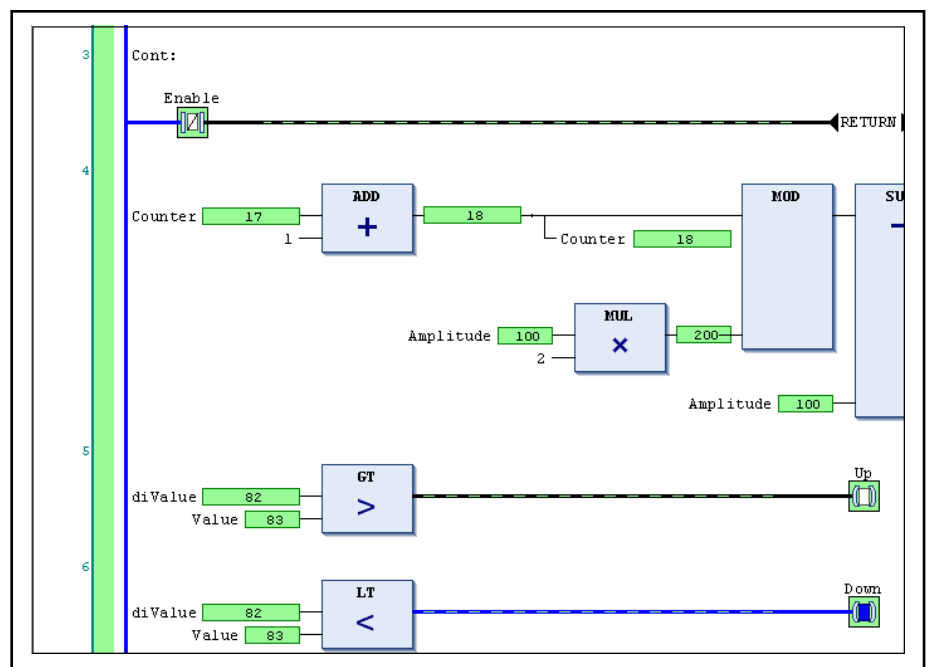


Abb. 12-6:

In AWL verwendet IndraLogic für jede Anweisung zwei Felder für die Anzeige der Istwerte. Eins links des Operators mit dem aktuellen Akkumulatorwert, eins rechts des Operanden mit dem Operandenwert.

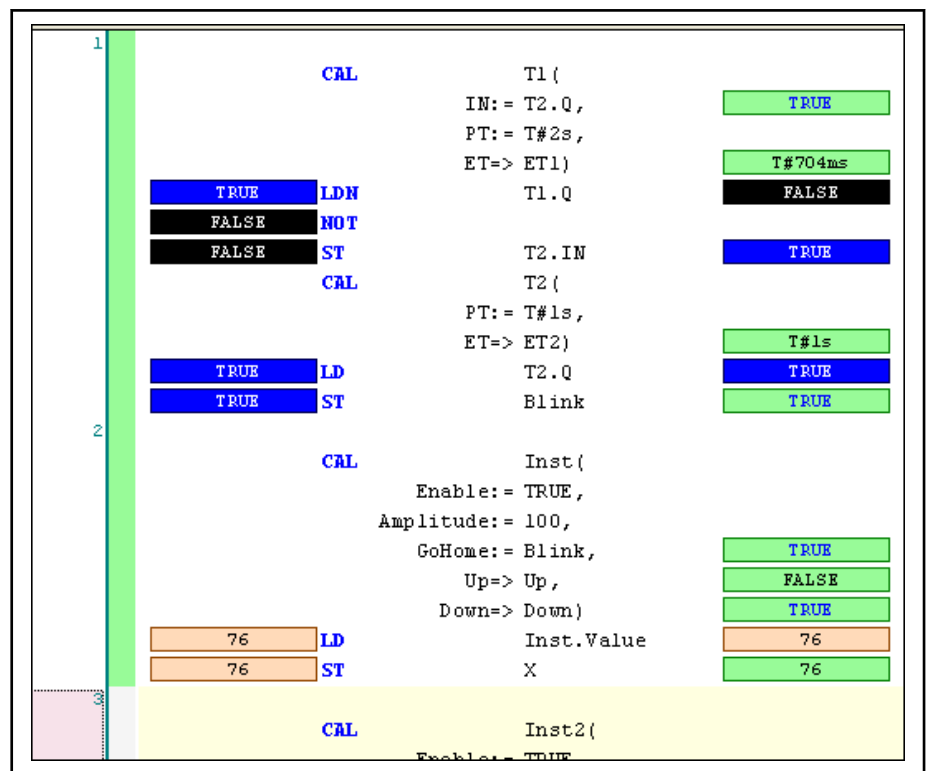


Abb. 12-7:

12.7 Aktuelle Abarbeitungsposition mit der Aufrufliste bestimmen

Mit Hilfe der Aufrufliste können Sie die aktuelle Position der Programmabarbeitung bestimmen. Diese Funktion ist bei der schrittweisen Abarbeitung des Programms sehr nützlich.

Voraussetzung: Die Applikation ist im Online-Betrieb. Das Programm steht an einem Haltepunkt oder Sie führen es schrittweise aus.

1. Öffnen Sie die Aufrufliste mit dem Befehl **Debuggen ► Aufrufliste**

Die Aufrufliste wird geöffnet. Die Liste zeigt die aktuell erreichte Position mit vollständigem Aufrufpfad.

Die Aufrufliste ist auch im Offline-Betrieb verfügbar oder im normalen Online-Betrieb (ohne gerade die Debugging-Funktionen zu benützen). In diesem Fall enthält sie die zuletzt während einer schrittweisen Ausführung angezeigte Position, allerdings in "gegrauter" Schrift.

Siehe auch

- [Kap. "Befehl 'Aufrufliste'" auf Seite 667](#)
- [Kap. 12.3 "Schrittweises Abarbeiten eines Programms \(Stepping\)" auf Seite 196](#)

12.8 Taskaufstellung überprüfen

Die Registerkarte **Taskaufstellung** des Geräteeditors zeigt in einer Übersicht, welche Tasks mit welcher Priorität die einzelnen Eingänge und Ausgänge des E/A-Abbilds Ihrer Applikation bearbeiten. Sie können hier überprüfen, ob nicht ein unbeabsichtigtes Überschreiben von Werten entsteht, was zu undefinierten Werten führen kann.

1. Erzeugen Sie für die Applikation Code: Wählen Sie dazu beispielsweise den Befehl **Erstellen ► Code erzeugen**.
2. Öffnen Sie mit einem Doppelklick auf das Geräteobjekt im Projektextplorer den Geräteeditor. Wählen sie die Registerkarte **Taskaufstellung**.

Sie erhalten eine Darstellung der Eingänge und Ausgänge Ihrer Applikation und die Zuordnung der Tasks und deren Prioritäten. Sehen Sie für Details die Beschreibung der Registerkarte **Taskaufstellung**.

Siehe auch

- [Kap. "Registerkarte 'Task-Aufstellung'" auf Seite 542](#)
- [Kap. 8.1 "Projektextplorer und Geräteeditor" auf Seite 71](#)
- ["Allgemeines zum E/A-Abbild \(Mapping\)" auf Seite 76](#)

13 Applikation zur Laufzeit

Wenn die Applikation auf der SPS läuft, gibt es in IndraLogic Funktionalitäten zum Überwachen und Ändern der Variablenwerte, sowie zum Aufzeichnen und Speichern ihres Verlaufs.

Es gibt auch Möglichkeiten, gewisse Informationen aus der SPS abzurufen, das SPS-Logbuch einzusehen, einen Coredump anzuzeigen und das Zeitverhalten der Tasks zu beobachten.

Beachten Sie auch die Möglichkeit, den Zugriff auf die laufende Applikation über Online-Befehle aus dem Entwicklungssystem in kritischen Zuständen der Maschine einzuschränken.

13.1 Monitoring von Werten

Sie können zur Laufzeit die aktuellen Werte von Variablen eines Programmierobjekts an verschiedenen Stellen im Projekt beobachten. Wir nennen dies "Monitoring":

- Online-Ansicht des Programmiereditor eines Objekts: "Inline-Monitoring"
- Online-Ansicht des Deklarationseditors eines Objekts
- Objektunabhängige, konfigurierbare Überwachungslisten

Sie können die Ergebnisse von Funktionsaufrufen und die aktuellen Werte von Variablen in Objekten des Typs Eigenschaft (PROPERTY) monitoren, wenn Sie das Pragma `{attribute 'monitoring' ...}` einsetzen.



Zusätzliche Möglichkeiten, die aktuellen Variablenwerte zu erfassen:

- Lesen und Speichern von "Rezepturen"
- Aufzeichnen der Werte über eine Zeitachse, um unmittelbar oder später den Verlauf darzustellen: Trace-Funktionalitäten

Siehe auch

- [Kap. "Attribut 'monitoring'" auf Seite 419](#)
- [Kap. 13.1.2 "Überwachungslisten verwenden" auf Seite 211](#)
- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)
- [Kap. 13.3 "Datenerfassung mit 'Trace'" auf Seite 216](#)

13.1.1 Monitoring in Programmierobjekten aufrufen

Die Monitoring-Funktion ermöglicht die Darstellung der aktuellen Variablenwerte einer Programmorganisationseinheit (POU) während des Online-Betriebs.

Voraussetzungen:

- Die IndraLogic-Option "Inline-Monitoring aktivieren" ist aktiviert.
Die Optionseinstellung finden Sie über das Indraworks Menü **Extras** ► **Optionen** ► **IndraLogic 2G**, Kategorie **Texteditor - Monitoring Inline-Monitoring aktivieren**.
- Eine Applikation ist auf die Steuerung geladen und die Steuerung läuft
- Die Applikation ist Eingeloggt (Online)
- Die POU ist im Editor geöffnet

Darstellungsformat einstellen und Monitoring einer POU starten

1. Um das Darstellungsformat auf Dezimalwerte einzustellen wählen Sie den Menüeintrag **Debuggen ► Anzeigemodus ► Dezimal**.
2. Laden Sie eine Applikation auf die Steuerung und starten Sie die Applikation.
3. Loggen Sie die Applikation ein (Online-Betrieb).
4. Öffnen Sie die POU mittels Doppelklick im Projektextplorer.
→ Der Editor öffnet sich.
Sowohl in der Deklaration als auch in der Implementierung werden Istwerte der Variablen angezeigt und ständig aktualisiert.

Im Deklarationseditor monitoren

Der aktuelle Wert einer Variablen wird in der Spalte **Wert** (3) dargestellt. Sie können den in Spalte **Vorbereiteter Wert** (4) eingetragenen Wert schreiben und forcen. Geforcen Werten ist ein rotes Symbol **F** vorangestellt.

Expression	Type	Value	Prepared value	Address	Comment
iVar	INT	F 2411		%MW9	
bVar	BOOL	TRUE	FALSE	%QX0.3	
myStruct	TestStruct				Check address defined in "TestStru
nTest1	INT	2411		%MW0	
nTest2	INT	0		%MW2	
aVar	ARRAY [0..3] OF INT			%MW5	
fbinst	FB1				instance of function block FB1
fbin	INT	0			
fbout	INT	0			
fhvar	INT	n			

- ① Ausdruck
- ② Datentyp
- ③ Wert
- ④ Vorbereiteter Wert
- ⑤ Adresse
- ⑥ Kommentar

Abb. 13-1:

Der Istwert eines Ausdrucks ① wird in der Spalte „Wert“ ③ angezeigt.

Sie können einen in der Spalte Vorbereiteter Wert ④ eingetragenen Wert schreiben und forcen. Während des Forcen wird der Istwert mit dem roten Symbol **F** gekennzeichnet.

Der Ausdruck einer Schnittstellenreferenz ist aufklappbar. Wenn die Schnittstelle auf eine globale Instanz zeigt, wird diese globale Instanz als erster Eintrag unter der Referenz angezeigt. Wenn sich danach die Schnittstellenreferenz ändert, klappt die angezeigte Referenz zu.

In der Implementierung monitoren (Inline-Monitoring)

Die Darstellung des aktuellen Variablenwerts in der Implementierung wird als "Inline-Monitoring" bezeichnet.

Abhängig von der Programmiersprache gibt es folgende Darstellungen im Implementierungsteil:

- Variablen haben hinter ihrem Namen ein Fenster mit dem aktuellen Wert eingeblendet: `nResult2` 17.

Wenn Sie für Variablen Werte zum Forcen oder Schreiben vorbereitet haben, werden diese innerhalb des Inline-Monitoring-Fensters hinter dem aktuellen Wert in spitzen Klammern dargestellt.

Nach dem Forcen werden die betroffenen Werte mit dem Symbol **F** gekennzeichnet.

- **Netzwerkeditoren und im CFC-Editor:**
Verbindungslinien (Signalleitungen) sind entsprechend ihrem booleschen Istwert farblich gekennzeichnet: Blau bedeutet **TRUE**, schwarz bedeutet **FALSE**.
- **KOP-Editor:**
Zusätzlich sind die Kontakt- und Spulenelemente gekennzeichnet.
Bei Kontakten oder Spulen wird ein vorbereiteter Wert (**TRUE** oder **FALSE**) in einem kleinen Fenster neben dem Element angezeigt.
- **AS-Editor:**
Transitionen mit Wert **TRUE** sind entsprechend ihrem booleschen Istwert farblich gekennzeichnet: Blau bedeutet **TRUE**, schwarz bedeutet **FALSE**.
Aktive Schritte sind blau gekennzeichnet.
In der Implementierung sind geforcete Transitionswerte rot gekennzeichnet.
- **AWL-Tabelleneditor:**
Aktuelle Werte werden in einer eigenen Spalte dargestellt.

Beispiel:

Monitoring im ST-Editor

```

1 iVar 13945 := iVar 44 + 1; (* counter *)
2 bVar TRUE := TRUE;
3 myStruct.nTest1 13945 := iVar 139 <39>;
4 aVar[2] 13945 := iVar 13945;
5 erg 0 := fbinst.fbout 0; RETURN

```

Abb. 13-2:

Beispiel:

Monitoring im KOP-Editor

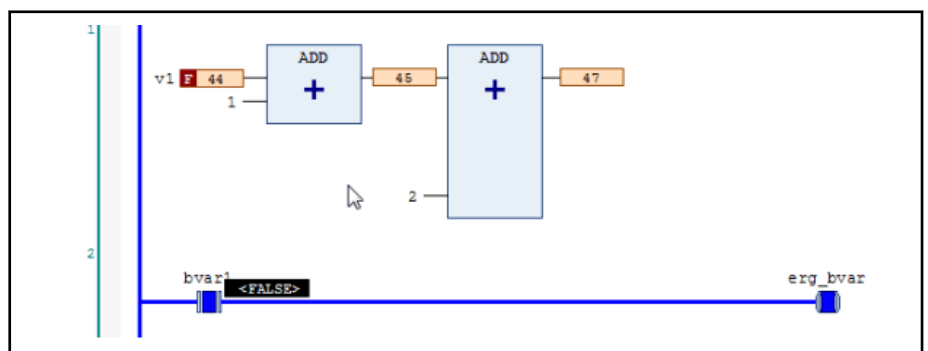


Abb. 13-3:

Beispiel:

Monitoring im AS-Editor

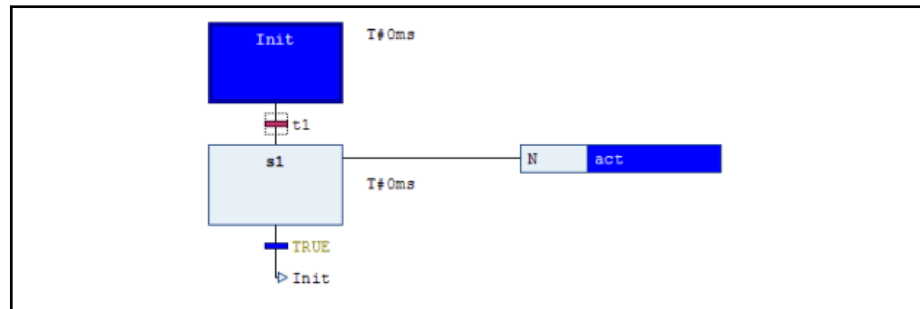


Abb. 13-4:



Sie können die "Inline Monitoring"-Funktion hier deaktivieren: **Extras** ► **Optionen** ► **IndraLogic 2G**, Kategorie **Texteditor** / **Monitoring**.

Siehe auch

- [Kap. "ST-Editor im Online-Betrieb" auf Seite 242](#)
- [Kap. "FUP/KOP/AWL-Editor im Online-Betrieb" auf Seite 277](#)
- [Kap. "AS-Editor im Online-Betrieb" auf Seite 253](#)
- [Kap. "CFC-Editor im Onlinebetrieb" auf Seite 293](#)

Array teilweise monitorieren

Ein Array, das aufgeklappt ist kann die Istwerte von bis zu 1000 Elementen anzeigen, was bei großen Arrays unübersichtlich sein kann.

Außerdem kann ein Array auch mehr als 1000 Elemente umfassen und damit die Höchstmenge an darstellbaren Elementen überschreiten. In Diesem Fall ist es hilfreich, den Bereich der angezeigten Elemente zu begrenzen, wie im folgendem Absatz beschrieben.

Monitoringbereich begrenzen

Voraussetzung: Eine Applikation läuft. Darin ist eine mehrdimensionale Array-Variable mit mehr als 1000 Elementen deklariert.

Beispiel: `arrBig : ARRAY [0..100, -9..10, -19..20] OF INT;`

1. Klicken Sie bei der Variablen `arrBig` in das Feld der Spalte Datentyp.
→ Der Dialog "Monitoringbereich" öffnet sich.
2. Geben Sie bei "Start" den Wert `[1, -9, -19]` ein.
3. Geben Sie bei "Ende" den Wert `[1, 10, 20]` ein.
→ Die Istwerte von 800 Array-Elemente werden im Deklarationseditor angezeigt. Der Bereich ist begrenzt auf Elemente des Index `[1, <i>, <j>]` mit `i` von -9 bis 10 und `j` von -19 bis 20.

Siehe auch:

- [Kap. 9.2.1 "Deklarationseditor verwenden" auf Seite 87](#)
- [Kap. 12.4 "Forcen und Schreiben von Variablen" auf Seite 198](#)
- [Kap. 18.3.5 "Dialog 'Monitoringbereich'" auf Seite 748](#)

Funktionsbaustein monitorieren

Wenn Sie während des Onlinebetriebs die Editoransicht eines Funktionsbausteins doppelklicken, erscheint ein Dialog, in dem Sie zwischen der Ansicht der Basisimplementierung oder einer bestimmten Instanz wählen können.

Wenn Sie die Basisimplementierung wählen, dann erscheint im Editor der Code, ohne dass aktuelle Werte angezeigt werden. Setzen Sie nun in der Basisimplementierung einen Haltepunkt. Wenn die Ausführung dort anhält, werden die aktuellen Werte derjenigen Instanz angezeigt, die im Programmablauf zuerst abgearbeitet wird. Sie können nun sukzessive durch alle Instanzen steppen.

Wenn Sie eine der Instanzen wählen, dann öffnet der Editor mit dem Code der Funktionsbaustein-Instanz. Die aktuellen Werte werden in der Deklaration und gegebenenfalls in der Implementierung angezeigt und ständig aktualisiert.

Siehe auch:

- [Kap. "Objekt 'Funktionsbaustein' \(FUNCTION_BLOCK\)" auf Seite 556](#)
- [Kap. 12.2 "Verwenden von Haltepunkten" auf Seite 193](#)

Eigenschaft monitoren

Sie können in einem Eigenschaftsobjekt  Variablen monitoren, wenn Sie während des Onlinebetriebs einen Haltepunkt in der Funktion setzen. Wenn dort angehalten wird, werden die aktuellen Werte angezeigt.

Zusätzlich zu den eigenen Werten werden automatisch die Werte der Variablen der übergeordneten Instanz angezeigt. Im Deklarationsteil der Eigenschaft erscheint dann in der ersten Zeile der Pointer `THIS`, der auf die übergeordnete Instanz zeigt, mit den aktuellen Datentypangaben und Werten.

Beispiel

```
FUNCTION_BLOCK FB_BaseAlfa
VAR
    iBaseLocal : INT;
    sBaseLocal : STRING;
END_VAR
iBaseLocal := iBaseLocal + 1;
sBaseLocal := 'Testing Text';

FB_BaseAlfa.PorpBeta.Get
iBaseLocal := iBaseLocal + 1;
IF iBaseLocal > 0 THEN
    PropBeta := TRUE;
END_IF

FB_BaseAlfa.PorpBeta.Set
IF PropBeta = TRUE THEN
    iBaseLocal := 0;
    sBaseLocal := 'Tested IF';
END_IF

PROGRAM PLC_PRG
VAR
    fb_BaseAlfa : FB_BaseAlfa;
END_VAR

fb_BaseAlfa();

IF fb_BaseAlfa.PropBeta = TRUE THEN
    xResult := TRUE;
END_IF
IF xReset THEN
    fb_BaseAlfa.PropBeta := TRUE;
    xReset := FALSE;
END_IF
```

Siehe auch [Kap. "Objekt 'Eigenschaft'" auf Seite 571](#)

Eigenschaftenzugriff im übergeordneten Programmierobjekt monitoren

Sie können in einem Funktionsbaustein oder Programm zusätzlich zu den Variablenwerten die Werte von untergeordneten Eigenschaften  monitoren.

Fügen Sie dafür in dem untergeordneten Eigenschaftenobjekt in der Deklaration entweder das Pragma {attribute 'monitoring' = 'variable'}, oder das Pragma {attribute 'monitoring' = 'call'} ein.

Wenn Sie zur Laufzeit die übergeordnete Programminstanz, oder übergeordnete Funktionsbaustein-Instanz öffnen, dann werden im Editor zusätzlich zu den aktuellen Variablenwerte die aktuellen Eigenschaftenwerte angezeigt.

Siehe auch:

- [Kap. "Objekt 'Eigenschaft'" auf Seite 571](#)
- [Kap. "Attribut 'monitoring'" auf Seite 419](#)

Methode monitoren

Sie können in einem Methodenobjekt  Variablen monitoren, wenn Sie während des Onlinebetriebs einen Haltepunkt in der Methode setzen. Wenn dort angehalten wird, werden die aktuellen Werte angezeigt.

Zusätzlich zu den eigenen Werten werden automatisch die Werte der Variablen der übergeordneten Instanz angezeigt. Im Deklarationsteil der Methode erscheint dann in der ersten Zeile der Pointer `THIS`, der auf die übergeordnete Instanz zeigt, mit den aktuellen Datentypangaben und Werten.

Beispiel

```
FUNCTION_BLOCK FB_BaseAlfa
VAR
    iBaseLocal : INT;
    sBaseLocal : STRING;
END_VAR
iBaseLocal := iBaseLocal + 1;
sBaseLocal := 'Testing Text';

METHOD MethBaseAlfa : BOOL // Method of FB_BaseAlfa
VAR_INPUT
END_VAR
VAR
    iMethLocal : INT;
END_VAR
iMethLocal := iMethLocal + 1;

PROGRAM PLC_PRG
VAR
    fb_BaseAlfa : FB_BaseAlfa;
END_VAR
fb_BaseAlfa();
fb_BaseAlfa.MethBaseAlfa();
```

Siehe auch:

- [Kap. "Objekt 'Eigenschaft'" auf Seite 571](#)
- [Kap. "Objekt 'Methode'" auf Seite 564](#)

Funktion monitoren

Sie können in einem Funktionsobjekt Variablen monitoren, wenn Sie während des Onlinebetriebs einen Haltepunkt in der Funktion setzen. Wenn dort angehalten wird, werden die aktuellen Werte angezeigt.

Rückgabewert eines Funktionsaufrufs monitoren

Im ST-Editor einer POU wird an der Stelle, an der eine Funktion aufgerufen wird, der aktuelle Rückgabewert als Inline-Monitoring angezeigt.

Dafür muss Folgendes erfüllt sein:

- Beim Rückgabewert handelt sich um einen Typ, der als 4-Byte-numerischer Wert interpretiert werden kann. Beispielsweise `INT`, `SINT` oder `LINT`.
- In der Funktion ist das Pragma {attribute 'monitoring' := 'call'} eingefügt.

Siehe auch

- [Kap. "Objekt 'Funktion\(FUNCTION\)" auf Seite 558](#)
- [Kap. "Attribut 'monitoring'" auf Seite 419](#)

13.1.2 Überwachungslisten verwenden

Was ist eine Überwachungsliste?

Eine Überwachungsliste ist eine benutzerdefinierte Liste von Projektvariablen, die zum Zweck des Monitorings ihrer Werte in einer Ansicht zusammengefasst werden.

Im Onlinebetrieb können Sie in einer Überwachungsliste Variablenwerte Schreiben und Forcen.

Monitoring, Schreiben und Forcen sind wie im Onlinebetrieb des Deklarationseditors zu handhaben.

Sie können das Format der Darstellung von Gleitpunktwerten in den Optionen für Monitoring anpassen.

In einem Projekt stehen vier Überwachungslisten `Überwachungsliste <n>` im Menü **Debuggen** ► **Überwachung** zur Verwendung bereit.



Wenn der Ausdruck eine Schnittstellenreferenz ist, ist sie aufklappbar. In dem Fall, dass die Schnittstelle auf eine globale Instanz zeigt, wird als erster Eintrag unter der Referenz diese globale Instanz angezeigt. Wenn sich die Schnittstellenreferenz ändert, wird die angezeigte Referenz zugeklappt.

Siehe:

- [Kap. 17.1.1 "Deklarationseditor" auf Seite 239](#)
- [Kap. 18.3.5 "Dialog 'Monitoringbereich'" auf Seite 748](#)

Anlegen und Bearbeiten einer Überwachungsliste (Offline- oder Onlinebetrieb)

Voraussetzung: Ein Projekt ist im Offline- oder Onlinebetrieb geöffnet. Das Projekt enthält eine Applikation, in der Variablen deklariert sind, die Sie in einer der vier möglichen Überwachungslisten haben möchten.

1. Wählen sie den Befehl **Debuggen** ► **Überwachung** ► **Überwachungsliste <n>**.

Die Ansicht `Überwachungsliste <n>` erscheint. Sie enthält eine leere Tabellenzeile.

2. Geben Sie nach einem Doppelklick auf das Feld in Spalte **Ausdruck** eine zu überwachende Variable ein, händisch oder über die Eingabehilfe.

Syntax:

`<Gerätename>.<Applikationsname>.<Objektnamen>.<Variablenname>`

Beispiel:

`IndraMotionM1c1.Application.PlcProg.ivar`

Wenn Sie den Namen einer strukturierten Variablen eintragen, werden im Onlinebetrieb die einzelnen Komponenten automatisch in weiteren Zeilen angezeigt.

3. Definieren Sie nacheinander alle mit dieser Liste zu überwachenden Variablen. Die Reihenfolge können Sie mit Drag&Drop verändern.

Die Felder **Ausführungszeitpunkt**, **Datentyp**, **Adresse**, **Kommentar** werden automatisch gemäß der Variablendeklaration gefüllt.

Das Symbol vor dem Ausdruck zeigt an, um welche Art von Variable es sich handelt:

-  Eingangsvariable
-  Ausgangsvariable
-  "normale" Variable



Im Onlinebetrieb können Sie eine Überwachungsliste auch mit Hilfe des Kontextmenü-Befehls **Zur Überwachungsliste hinzufügen** neu anlegen oder eine bestehende bearbeiten.

Siehe auch

- [Kap. "Befehl 'Überwachung- 'Überwachungsliste <n>' auf Seite 667](#)

Hinzufügen von Variablen mit Hilfe des Befehls 'Zur Überwachungsliste hinzufügen' (Onlinebetrieb)

Voraussetzung: Ein Projekt ist im Betrieb geöffnet. Es enthält eine Applikation, in der bereits Variablen deklariert sind, die Sie in eine Überwachungsliste aufnehmen möchten.

1. Öffnen Sie die gewünschte Überwachungsliste mit Befehl **Debuggen ► Überwachung ► Überwachungsliste <n>..**
2. Stellen Sie den Cursor im Deklarations- oder Implementierungsteil einer POU auf die Variable und wählen Sie aus dem Kontextmenü den Befehl **Zur Überwachungsliste hinzufügen**.

Der Liste wird ein Eintrag für die gewählte Variable hinzugefügt.

3. Sie können weitere Variablen auf diese Weise hinzufügen, oder durch Eintragen in der Liste im Feld **Ausdruck**, wie oben beschrieben ("Anlegen und Bearbeiten...").

Die Überwachungslisten werden unmittelbar aktualisiert.



Wenn keine Überwachungsliste geöffnet ist, während Sie den Befehl **Zur Überwachungsliste hinzufügen** auf eine Variable anwenden, wird diese automatisch der Liste **Überwachungsliste 1** hinzugefügt.



Schreiben und Forcen der Variablenwerte ist auch in den Überwachungslisten möglich. Im Onlinebetrieb erscheint dazu die Spalte **Vorbereiteter Wert**.

Siehe auch

- [Kap. "Befehl 'Zu Überwachungsliste hinzufügen'" auf Seite 742](#)
- [Kap. 13.1.1 "Monitoring in Programmierobjekten aufrufen" auf Seite 205](#)
- [Kap. 12.4 "Forcen und Schreiben von Variablen" auf Seite 198](#)

13.2 Werte ändern mit Rezepturen

Sie verwenden Rezepturen, um gleichzeitig die Werte für einen bestimmten Satz an Variablen (Rezepturdefinition) auf der Steuerung zu verändern oder auszulesen.

Die Grundeinstellungen zu Rezepturen, wie der Speicherort und das Speicherformat, legen Sie im Objekt **Rezepturverwalter** fest. Unterhalb dieses Objekts fügen Sie eine oder mehrere Rezepturdefinitionen ein. Eine Rezeptur-

definition umfasst eine oder mehrere Rezepturen für die enthaltenen Variablen. Die Rezeptur besteht aus bestimmten Variablenwerten.

Sie können die Rezepturen in Dateien speichern, oder direkt von Dateien in die Steuerung schreiben.

Rezepturen können über die Programmieroberfläche von IndraLogic, über Visualisierungselemente oder über das Applikationsprogramm geladen werden.



Verwendung von Rezepturen auf entfernten Geräten

Die Variablenwerte aus Rezepturen werden automatisch zu, oder von einer anderen Steuerung übertragen, wenn es sich um Datenquellenvariablen handelt und ein Datenquellenaustausch konfiguriert ist.

Das Lesen/Schreiben erfolgt synchron. Also aktualisiert IndraLogic alle in einer Rezeptur vorhandenen Variablen zum selben Zeitpunkt.

Durch einen Aufruf von (`g_recipeManager.LastError = 0`) nach dem Lesen/Schreiben können Sie überprüfen, ob die Übertragung funktioniert hat.

Siehe auch

- [Kap. 18.1.21 "Objekt 'Rezepturverwalter'" auf Seite 583](#)
- [Kap. 18.1.22 "Objekt 'Rezepturdefinition'" auf Seite 587](#)

Handhaben von Rezepturen in der IndraLogic-Benutzeroberfläche

Die Programmieroberfläche von IndraLogic bietet Ihnen Befehle zum Erzeugen von Rezepturen sowie zum Lesen/Schreiben im Online-Betrieb.

Siehe auch

- [Kap. 18.2.14 "Menü 'Rezepturen'" auf Seite 723](#)

Verwenden von Rezepturen in der Applikation

Sie können Rezepturen zur Laufzeit im Anwenderprogramm oder über Visualisierungselemente nutzen.

Im Anwenderprogramm verwenden Sie die Methoden des Funktionsbausteins `RecipeManCommands` aus der Bibliothek `RecipeManagement` (**Bibliothek hinzufügen ► Intern**). In der Visualisierung erfolgt die Verwendung von Rezepturen über die Eingabekonfiguration (Internes Kommando) von Visualisierungselementen.



Die Rezepturverwaltung liest beim Initialisierungsvorgang die Werte der Variablen, die in der Rezepturdefinition festgelegt sind. Der Vorgang findet am Ende der Initialisierungsphase der Applikation statt. Zu diesem Zeitpunkt sind alle Initialwerte der Applikationsvariablen gesetzt. Der Vorgang wird durchgeführt, um fehlende Werte aus Rezepturdateien richtig initialisieren zu können.

Siehe auch

- [Kap. 19.2 "Bibliothek 'Recipe Management'" auf Seite 789](#)
- [Kap. "Eingabekonfiguration" auf Seite 849](#)

Erstellen einer Rezeptur

1. Selektieren Sie im Projektexplorer das Objekt **Application**.
2. Wählen Sie den Befehl **Hinzufügen ► Rezepturverwalter**.
IndraLogic fügt den Rezepturverwalter zum Projektexplorer hinzu.
3. Selektieren Sie im Projektexplorer das Objekt **Rezepturverwalter**.
4. Wählen Sie den Befehl **Hinzufügen ► Rezepturdefinition**.

IndraLogic fügt die Rezepturdefinition unterhalb des Rezepturverwalters hinzu.

5. Öffnen Sie den Editor der Rezepturdefinition durch einen Doppelklick auf das Objekt.
6. Führen Sie einen Doppelklick im Editor auf das leere Feld unterhalb **Variable** aus. Geben Sie den Namen einer Variablen an, für die Sie eine Rezeptur definieren wollen. Die Eingabehilfe steht dazu bereit: Schaltfläche .
7. Wählen Sie den Befehl **Rezepturen ► Rezeptur einfügen** und geben Sie einen Namen für die neue Rezeptur an.
Im Editor erscheint eine Spalte mit dem Rezepturnamen.
8. Geben Sie in diesem Feld den Wert der Variable für diese Rezeptur ein.
9. Fügen Sie bei Bedarf weitere Variablen ein.
10. Selektieren Sie einen Variablenwert der Rezeptur und führen Sie den Befehl **Rezepturen ► Rezeptur speichern** aus. Wählen Sie einen Speicherort und einen Dateinamen aus.

IndraLogic speichert die Rezeptur im dem Format, das im Rezepturverwalter definiert ist.

Siehe auch

- [Kap. 18.1.21 "Objekt 'Rezepturverwalter'" auf Seite 583](#)
- [Kap. 18.1.22 "Objekt 'Rezepturdefinition'" auf Seite 587](#)
- [Kap. 18.2.14 "Menü 'Rezepturen'" auf Seite 723](#)

Laden einer Rezeptur aus einer Datei

Voraussetzung: In der Applikation gibt es eine Rezepturverwaltung. In einer Rezepturdefinition gibt es eine Rezeptur **myRec** mit Variablenwerten. Eine Rezepturdatei `myRec.txt`, die Einträge für diese Rezeptur enthält, liegt im Dateisystem vor.

Beispiel der Rezepturdatei:

```
PlcProg.bVar:=0
PlcProg.iVar:=2
PlcProg.dwVar:=35232
PlcProg.stVar:='first'
PlcProg.wstVar:="123443245"
```

1. Öffnen Sie mit Doppelklick auf das Objekt **Rezepturdefinition** im Projek Explorer den Tabelleneditor für die Definition der einzelnen Rezepturen.
Sie sehen eine Spalte `myRec` mit den aktuellen Werte für diese Rezeptur.
2. Bearbeiten Sie die Datei `myRec.txt` in einem externen Texteditor und ersetzen die Variablenwerte durch andere, die sie in die Rezepturdefinition in IndraLogic laden möchten. Speichern Sie die Datei.
3. Klicken Sie in der Rezepturdefinition in die Spalte von **myRec** und wählen im Kontextmenü den Befehl **Rezeptur laden**.

Ein Hinweis auf die mögliche Notwendigkeit, eines Online-Change beim nächsten Einloggen erscheint. Online Change wird nötig, falls Sie durch das Laden der Rezeptur die aktuellen Werte der Rezepturvariablen verändern.

4. Schließen Sie den Hinweisdialog mit **Ja** (Fortfahren). Wählen Sie die Datei `myRec.txt` aus dem Datei-Explorer zum Laden aus.

Die Rezepturwerte in der Rezepturdefinition werden entsprechend den in der Datei gelesenen Werten aktualisiert.



Wenn Sie nur einzelne Variablen der Rezeptur mit neuen Werten überschreiben möchten, entfernen Sie vor dem Laden in der Rezepturdatei die Werte für die restlichen Variablen. Einträge ohne Wertangabe werden bei der Aktualisierung nicht eingelesen und bleiben auf der Steuerung und im Projekt unberührt.

Bei Werten vom Datentyp `REAL/LREAL` wird in manchen Fällen auch der Hexadezimalwert in die Rezepturdatei geschrieben. Das ist notwendig, damit bei der Rückkonvertierung der exakt identische Wert wiederhergestellt wird. In diesem Fall ändern Sie den Dezimalwert und löschen Sie den Hexadezimalwert.

Siehe auch

- [Kap. "Befehl 'Rezeptur laden'" auf Seite 724](#)
- [Kap. "Befehl 'Rezeptur laden und schreiben'" auf Seite 725](#)

Rezepturverwaltung auf der Steuerung Speicherverbrauch

Wenn Sie die Option **Rezepturverwaltung in der SPS** deaktiviert haben, verbraucht der Rezepturverwalter und die Rezepturdefinitionen keinen Speicher auf der Steuerung.

Wenn Sie die Option aktiviert haben, so wird für den Rezepturverwalter und alle Rezepturdefinitionen Code erzeugt, der auf der Steuerung Speicher belegt. Die Größe des Speicherverbrauchs hängt hauptsächlich von der Anzahl der Rezepturen und deren Variablen sowie vom Datentyp der Variablen ab. Dabei ist ebenfalls von Bedeutung, ob die Felder der Rezepturdefinition mit Werten besetzt sind oder nicht. Der Speicherverbrauch von Rezepturen kann nicht berechnet werden - dieser muss bei Bedarf empirisch ermittelt werden. Die nachfolgende Tabelle soll hier nur ein Anhaltspunkt sein.

	Codegröße (Byte)	Datengröße (Byte)	Gesamt (Byte)
Eine Rezepturdefinition mit 100 INT-Variablen	194406	79400	267352
Eine Rezepturdefinition mit 200 INT-Variablen	238318	121284	459344
Eine Rezepturdefinition mit 300 INT-Variablen	282230	163084	543856
Eine Rezepturdefinition mit 100 BOOL-Variablen	192742	69884	343168
Eine Rezepturdefinition mit 200 BOOL-Variablen	235446	101568	436872
Eine Rezepturdefinition mit 300 BOOL-Variablen	278146	133284	510072
Eine Rezepturdefinition mit 100 String-Variablen	203278	870084	1154000

	Codegröße (Byte)	Datengröße (Byte)	Gesamt (Byte)
Eine Rezepturdefinition mit 200 String-Variablen	255570	1709784	2973296
Eine Rezepturdefinition mit 300 String-Variablen	307886	2549484	2964112

Tab. 13-1: Speicherverbrauch für IndraLogic 3.5.6 mit Steuerung im Simulationsmodus

13.3 Datenerfassung mit 'Trace'

Ein **Trace** ermöglicht Ihnen auf der Steuerung den Werteverlauf von Variablen ähnlich einem digitalen Sampling-Oszilloskop zu erfassen.

Mit einem Trace können Sie den Werteverlauf von Variablen auf der Steuerung verfolgen, ähnlich einem digitalen Abtast-Oszilloskop.

Zur Laufzeit einer Applikation mit Trace werden innerhalb des Taskzyklus zuerst alle Anweisungen ausgeführt. Danach startet die Datenerfassung und die Speicherung der Werte mit Zeitstempel.

Der Zeitstempel ist relativ und bezieht sich auf den Startzeitpunkt der Datenerfassung.

Die Daten ergeben ein zeitdiskretes Signal, dessen Verlauf IndraLogic im Traceeditor darstellt.

Wert und Zeitstempel bilden einen "Sample", also einen Datensatz. Das Laufzeitsystem schreibt die Samples in einen Puffer von einstellbarer Größe.

IndraLogic fordert die Daten an, speichert sie im Trace-Editorpuffer und stellt sie als Funktion über die Zeit im Tracediagramm dar. Sie können den Werteverlauf der konfigurierten Variablen kontinuierlich beobachten, da IndraLogic die neuesten Daten darstellen kann.

Sie können die Datenspeicherung mit einem Trigger auslösen. Bei Auslösen des Triggers speichert die Applikation die Daten mit Bezug zum Triggerzeitpunkt und IndraLogic stellt die Daten um den Triggerzeitpunkt herum dar.

Trace Konfigurieren

Die Konfiguration sowie die Darstellung eines Trace ist im Projekt über den Traceeditor möglich.

Fügen Sie hierzu dem Applikationsobjekt im Projektextplorer das Objekt "Trace" hinzu.

Das Trace-Objekt enthält eine rein applikationsspezifische Tracekonfiguration. Die Tracekonfiguration wird auf die Steuerung geladen und läuft dort parallel zur Applikation.



Eine laufende Datenerfassung mit Trace kann zu einem signifikanten Anstieg der Zykluszeit von IEC-Tasks führen.



Eine Datenerfassung mit Trace läuft auch nach dem Ausloggen vom Gerät weiter.

Laufzeitsystemkomponente CmpTraceMgr

Die Gerätebeschreibung bei einem LZS mit Tracemanager enthält unter der Sektion `TargetSettings` den Eintrag `tracemanager`.

Beispiel

Gerätebeschreibung

```
<ts:section xmlns:ts="http://www.3s-software.com/schemas/TargetSettings-0.1.xsd" name="trace">
```



```
<ts:setting name="tracemanager" type="boolean" access="visible">
  <!-- With this option, the new tracemanager is activated in IL 2G.
        With this option, the CmpTraceMgr component is needed in the runtime system! -->
  <ts:value>1</ts:value>
</ts:setting>
</ts:section>
```

Dann überträgt IndraLogic beim Laden der Applikation auf die Steuerung nur die Tracekonfiguration. Wenn Sie den Trace starten, interpretiert die Applikation die Konfiguration, führt die Datenerfassung aus und puffert die Datensätze auf der Steuerung.

Die Laufzeitsystemkomponente `CmpTraceMgr` bietet, im Gegensatz zu einer Datenerfassung mit IEC-Code, eine erweiterte Funktionalität.

So ist die Datenerfassung von folgenden Werten möglich:

- Parameter der Steuerung, wie zum Beispiel die Temperaturkurve einer CPU oder eines Akku
- Signale des Geräts, wie zum Beispiel dem Stromverlauf eines Antriebs
- Systemvariablen einer anderen Laufzeitsystemkomponente

Sie können Parameter so wie IEC-Variablen im Dialog **Tracekonfiguration** unter **Variableneinstellungen** konfigurieren.

Siehe auch

- [Kap. 13.3.4 "Zugriff auf alle Traces der Steuerung" auf Seite 223](#)
- ["Teildialog 'Variableneinstellung'" auf Seite 786](#)

Datenerfassung nach Auslösen eines Triggers

Wenn Sie Daten beobachten wollen, die von einem Ereignis oder einer Bedingung abhängen, können Sie die Datenerfassung abhängig von einem Trigger freigeben.

Die Applikation in Laufzeit prüft, ob das Ereignis eingetreten oder die Bedingung erfüllt ist und puffert die Daten dann entsprechend.

Die Tracekonfiguration ermöglicht das Auslösen eines Triggers über:

- eine Triggervariable, die ein Ereignis abbildet
- eine Bedingung als Ausdruck
- eine Kombination von Triggervariable und Bedingung

Samples in Datei speichern

Sie können die Samples auf dem Dateisystem des Entwicklungssystems als Datei speichern.

Die Datei kann zusätzlich die Tracekonfiguration enthalten.

Dateierweiterung	Dateityp	Beschreibung
*.trace:	Tracedatei	Enthält die Samples und die Tracekonfiguration in einem XML-basierten Speicherformat. Sie können die Datei mit dem Befehl Trace laden offline in den Traceeditor laden und die Samples ohne Betrieb einer Steuerung analysieren.
*.txt	Textdatei	Enthält die Samples im ASCII-Format. Sie können die Datei mit einem externen Tool weiterverarbeiten.

Dateierweiterung	Dateityp	Beschreibung
*.trace.csv	Tracedumpdatei	Datei im CSV-Format, die die Tracekonfiguration und optional die Samples enthält. Sie können die Datei erstellen, indem Sie den Befehl Symbolische Tracekonfiguration exportieren wählen. Sie können diese Datei auf die Steuerung übertragen und in die Applikation laden. Dann können Sie in IndraLogic mit dem Befehl Trace laden diese im Traceeditor darstellen. Sie können aber auch den Befehl Trace ► Trace speichern und dort den Dateityp *.trace.csv wählen. Sie können diese Datei auf eine Steuerung übertragen und mit einer HMI laden und analysieren.
.traceconfig	Symbolische Tracekonfiguration	Enthält die Tracekonfiguration im CSV-Format. Sie können die Datei erstellen, indem Sie den Befehl Symbolische Tracekonfiguration exportieren wählen. Die Laufzeitsystemkomponente CmpTraceMgr kann die Datei lesen.

Tab. 13-2: Mögliche Dateiformate

Siehe auch

- [Kap. 18.1.27 "Objekt 'Trace'" auf Seite 605](#)
- [Kap. "Befehl 'Trace speichern'" auf Seite 738](#)
- [Kap. "Befehl 'Trace laden'" auf Seite 735](#)
- [Kap. "Befehl 'Symbolische Tracekonfiguration exportieren'" auf Seite 734](#)
- [Kap. 18.1.24 "Objekt 'Symbolkonfiguration'" auf Seite 588](#)

13.3.1 Erste Schritte mit Trace

PlcProg

```

PROGRAM PlcProg
VAR
    rSin : REAL;
    rVar : REAL;
END_VAR
rVar := rVar + 0.1;
rSin := 30 * SIN(rVar);

```

Voraussetzung: Die Applikation läuft mit dem Programm PlcProg auf der Steuerung.

1. Selektieren Sie im Projektfexplorer die Applikation und fügen Sie mit dem Befehl **Hinzufügen** ein neues Traceobjekt hinzu.
Der zugehörige Traceeditor öffnet sich, und die Befehle des Menüs **Trace** stehen zur Verfügung.
2. Wählen Sie den Befehl **Trace ► Konfiguration**.
Der Dialog **Tracekonfiguration** erscheint.
3. Wählen Sie eine Task aus, in der die Tracefunktionalität laufen wird. Im Normalfall ist das die gleiche Task, in der PlcProg läuft (PlcTask).
4. Fügen Sie mit **Variable hinzufügen** einen Eintrag in der Baumansicht der Tracekonfiguration hinzu und weisen Sie ihm eine IEC-Variable zu, zum Beispiel PlcProg.rSin

5. Loggen Sie sich über den Kontextmenü-Befehl der Applikation **Einloggen** ein und starten Sie die Applikation (Kontextmenü-Befehl der Applikation **Start**).
6. Wählen Sie den Befehl **Trace ► Trace Download**.

IndraLogic lädt die Tracekonfiguration auf die Steuerung. Die Applikation startet die Datenerfassung und überträgt die Daten an IndraLogic. Dort werden sie im Tracediagramm als Graph angezeigt. Ihnen stehen nun die Befehle zur Verfügung, um in den Samples zu navigieren und die Datenerfassung zu steuern.

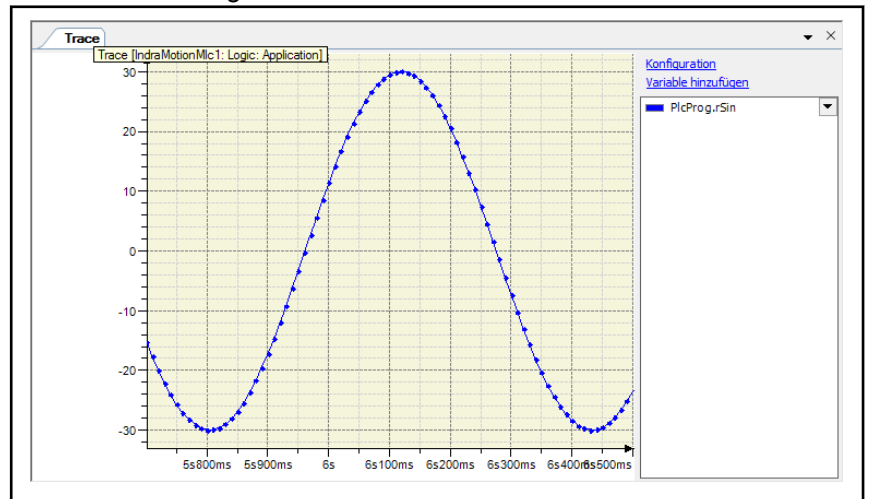


Abb. 13-5: Erfassen der sinusförmigen Daten der IEC-Variablen "PlcProg.rSin"

Siehe auch

- [Kap. 18.1.27 "Objekt 'Trace'" auf Seite 605](#)

13.3.2 Tracekonfiguration erstellen

Bestimmen Sie für eine vollständige Tracekonfiguration wenigstens eine Task und eine Variable.

Um die Datenerfassung zu triggern, aktivieren Sie die Triggeroption und wählen eine Triggervariable, oder bestimmen eine Aufzeichnungsbedingung.

Siehe

- [Kap. "Dialog 'Erweiterte Traceeinstellungen'" auf Seite 782](#)
- [Kap. "Dialog 'Tracekonfiguration'" auf Seite 783](#)

Task zuweisen

In dieser Task wird zur Laufzeit die Datenerfassung ausgeführt. Üblicherweise wird die selbe Task gewählt, in der auf die Variablen geschrieben wird.

1. Doppelklicken Sie auf das Traceobjekt.
Der Traceeditor öffnet sich und die Befehle des Menüs **Trace** stehen Ihnen zur Verfügung.
2. Wählen Sie den Befehl **Trace ► Konfiguration**.
Der Dialog **Tracekonfiguration** öffnet sich. In der Baumansicht **Traceaufzeichnung** ist der oberste Eintrag selektiert und rechts ist der Teildialog **Aufzeichnungseinstellungen** verfügbar.
3. Klicken Sie im Eingabefeld von **Task** auf ▼.

Eine Auswahlliste mit allen Tasks, die applikationsweit vorhanden sind, erscheint.

4. Wählen Sie eine Task für den Trace aus.

Siehe auch

- [Kap. "Dialog 'Tracekonfiguration'" auf Seite 783](#)

Tracevariable konfigurieren

1. Doppelklicken Sie auf das Traceobjekt.
Der Traceeditor öffnet sich. Die Befehle des Menüs **Trace** sind verfügbar.
2. Wählen Sie den Befehl **Variable hinzufügen**.
Der Dialog **Tracekonfiguration** öffnet sich. Rechts ist der Teildialog **Variableneinstellung** verfügbar.
3. Klicken Sie im Eingabefeld der Einstellung **Variable** auf  und wählen Sie im Dialog "Eingabehilfe" eine Tracevariable aus.
Die Variable ist für die Datenerfassung konfiguriert. Der Traceaufzeichnungsbaum und der Darstellungsbaum wurden um die Variable erweitert.
4. Wählen Sie den Link **Variable hinzufügen**.
Der Traceaufzeichnungsbaum und der Darstellungsbaum enthalten eine weitere, neue Variable. Rechts sind die Einstellungen der Variablen verfügbar.
5. Wählen Sie eine Tracevariable aus.
6. Beenden Sie den Dialog mit **OK**.
Die Variablen sind Tracevariablen und werden in der Tracevariablenliste dargestellt.

Tracevariable entfernen

1. Doppelklicken Sie auf das Traceobjekt.
2. Klicken Sie im Traceaufzeichnungsbaum auf eine Variable.
Die Variable ist selektiert.
3. Klicken Sie auf den Link **Variable entfernen**, oder verwenden Sie <Entf>.
4. Beenden Sie den Dialog mit **OK**.
Die Variable ist aus der Tracevariablenliste entfernt.

Parameter tracen

1. Doppelklicken Sie auf das Traceobjekt.
2. Wählen Sie den Befehl **Variable hinzufügen**.
Der Dialog **Tracekonfiguration** öffnet sich. Rechts ist der Teildialog **Variableneinstellung** verfügbar.
3. Klicken Sie auf , rechts neben der Einstellung **Variable**.
4. Wählen Sie in der Auswahlliste die Option **Parameter**.
5. Klicken Sie auf  und wählen Sie im Dialog **Eingabehilfe** einen Parameter aus.

6. Konfigurieren Sie die Darstellung des Parameters.
7. Beenden Sie den Dialog mit **OK**.
Der Parameter wird jetzt in der Tracevariablenliste dargestellt.

Triggerauslösung konfigurieren

1. Doppelklicken Sie auf das Traceobjekt.
Der Traceeditor öffnet sich und die Befehle des Menüs **Trace** stehen zur Verfügung.
2. Wählen Sie den Befehl **Trace ► Konfiguration**.
Der Dialog **Tracekonfiguration** erscheint. Rechts ist der Teildialog **Aufzeichnungseinstellungen** verfügbar.
3. Aktivieren Sie die Option **Trigger aktivieren**.
4. Wählen Sie die Task, in der die Datenerfassung ausgeführt werden wird.
5. Wählen Sie im Eingabefeld von **Triggervariable** eine Variable aus.
6. Beenden Sie den Dialog mit **OK**.
Die Datenerfassung wird getriggert werden.

Der Triggerzeitpunkt wird im Diagramm zur Laufzeit als schwarze Linie gekennzeichnet.

1. Laden Sie die Applikation und starten Sie sie.
2. Wählen Sie **Trace ► Download Trace**
Die Tracekonfiguration wird geladen. Nach Auslösen des Triggers speichert das Laufzeitsystem den Werteverlauf der Tracevariablen. Die Daten werden im Traceeditor angezeigt. Dabei wird der Zeitpunkt der Triggerauslösung im Diagramm als schwarze Linie dargestellt.

Anzeige der Zeitachse konfigurieren

1. Doppelklicken Sie auf das Traceobjekt.
2. Klicken Sie auf den Link **Konfiguration**, der sich oberhalb des Konfigurationsbaums befindet.
Der Dialog **Tracekonfiguration** öffnet sich.
3. Selektieren Sie im Darstellungsbaum unter **Darstellung ► Diagramme** den Eintrag **Zeitachse**.
Rechts erscheinen die Anzeigeeinstellungen der Zeitachse.
4. Ändern Sie die Voreinstellungen und klicken Sie auf den Link **Vorschau**.
Die Änderungen werden in der Vorschau des Koordinatensystems visualisiert.
5. Selektieren Sie im Darstellungsbaum einen Eintrag **Y-Achse**. Den Eintrag **Y-Achse** gibt es unter jedem konfigurierten Diagramm. Somit ist die Anzeige der Werteachse für jedes Diagramm separat einstellbar.
Rechts ist der Teildialog **Anzeigeeinstellungen** der selektierten Achse verfügbar.
6. Ändern Sie die gewünschte Einstellung
Die Änderungen werden in der Vorschau des Koordinatensystems angewendet.

	Beenden Sie den Dialog Tracekonfiguration mit OK. Die Anzeigeänderungen werden in den betroffenen Diagrammen sichtbar.
Anzeige der Tracevariablen konfigurieren	- Doppelklicken Sie auf das Traceobjekt. - Klicken Sie auf den Link Konfiguration. Der Dialog Tracekonfiguration öffnet sich. - Wählen Sie unter Traceaufzeichnung eine Variable aus. Rechts ist der Teildialog Variableneinstellungen der selektierten Variable verfügbar. - Ändern Sie eine Einstellung, beispielsweise die Einstellung Linientyp. - Beenden Sie den Dialog mit OK. Die Anzeigeänderungen werden in den betroffenen Diagrammen sichtbar.
Puffer für Daten auf dem LZS konfigurieren	- Doppelklicken Sie auf ein Traceobjekt. - Wählen Sie den Befehl Trace ► Konfiguration. Der Dialog Tracekonfiguration öffnet sich. Rechts ist der Teildialog Aufzeichnungseinstellungen verfügbar. - Klicken Sie auf die Schaltfläche Erweitert. Der Dialog Erweiterte Trace-Einstellungen öffnet sich. - Ändern Sie die Einstellung Messung in jedem n-ten Zyklus, oder Größe des LZS-Puffers in Samples. - Beenden Sie den Dialog mit OK. Die Puffereinstellungen sind neu konfiguriert. Nach dem nächsten Laden der Tracekonfiguration auf das LZS werden sie angewendet werden.
Tracekonfiguration zur Laufzeit bearbeiten	Voraussetzung: Auf der Steuerung läuft die Applikation und eine Tracekonfiguration ist geladen. - Doppelklicken Sie im Traceaufzeichnungsbaum auf eine Variable Der Dialog Tracekonfiguration öffnet sich. - Ändern Sie beispielsweise die Farbe. Die Variable wird in den betroffenen Diagrammen mit neuer Farbe dargestellt, ohne die Ausführung der Applikation zu unterbrechen.



Wenn Sie wesentliche Einstellungen ändern, beispielsweise eine Tracevariable, müssen Sie die Tracekonfiguration neu auf die Steuerung laden.

Siehe auch

- [Kap. "Dialog 'Erweiterte Traceeinstellungen'" auf Seite 782](#)
- [Kap. "Dialog 'Tracekonfiguration'" auf Seite 783](#)

13.3.3 Datenerfassung steuern

Sie können die Datenerfassung über Menübefehle steuern.

Voraussetzung: Die Applikation ist auf dem Laufzeitsystem geladen und ein Trace ist konfiguriert.

Menübefehle zu Trace

- [Kap. "Befehl 'Download Trace'" auf Seite 734](#)
- [Kap. "Befehl 'Trace starten'" auf Seite 739](#)
- [Kap. "Befehl 'Trace stoppen'" auf Seite 739](#)
- [Kap. "Befehl 'Trigger zurücksetzen'" auf Seite 738](#)

13.3.4 Zugriff auf alle Traces der Steuerung

Sie können über Menübefehle Zugang auf die Laufzeitsystemkomponente `CmpTraceManager` erhalten.

Voraussetzung: Die Applikation ist im Online-Betrieb.

Zugriff auf die Traces der Steuerung CmpTraceMgr bieten die Trace-Befehle

- [Kap. "Befehl 'Onlineliste'" auf Seite 737](#)
- [Kap. "Befehl 'Trace hochladen'" auf Seite 740](#)

13.3.5 Navigieren in den Tracedaten

Sie können über Menübefehle in den Daten des Tracediagramms navigieren.

Voraussetzung: Die Applikation ist im Onlinebetrieb.

Menübefehle Trace

- [Kap. "Befehl 'Cursor'" auf Seite 733](#)
- [Kap. "Befehl 'Zoomen mit Maus'" auf Seite 736](#)
- [Kap. "Befehl 'Ansicht zurücksetzen'" auf Seite 738](#)
- [Kap. "Befehl 'Automatisch skalieren'" auf Seite 732](#)
- [Kap. "Befehl 'Komprimieren'" auf Seite 732](#)
- [Kap. "Befehl 'Strecken'" auf Seite 739](#)
- [Kap. "Befehl 'in Mehrkanal konvertieren'" auf Seite 736](#)
- [Kap. "Befehl 'in Einkanal konvertieren'" auf Seite 736](#)

13.3.6 Trace verwalten

Sie können über Menübefehle Traces in verschiedenen Formaten laden und speichern.

Menübefehle Trace

- [Kap. "Befehl 'Trace speichern'" auf Seite 738](#)
- [Kap. "Befehl 'Trace laden'" auf Seite 735](#)
- [Kap. "Befehl 'Symbolische Tracekonfiguration exportieren'" auf Seite 734](#)

13.3.7 Statistik anzeigen

Funktion: IndraLogic wertet die erfassten Daten statistisch aus, zeigt Sie Ihnen an und ermöglicht eine Ablage im Zwischenspeicher.

Aufruf: Befehl muss installiert werden über **Extras ► Anpassen - Trace ► Statistik**

Siehe auch

- [Kap. "Befehl 'Statistik'" auf Seite 740](#)

13.4 Tasks überwachen

Im Online-Betrieb können Sie einige statistische Werte der Tasks im Laufzeitsystem anzeigen. Diese Informationen können bei Taktzeituntersuchungen oder Problemen im Zeitverhalten sehr hilfreich sein.

1. Wechseln Sie in den Online-Betrieb.
2. Selektieren Sie die **Task-Konfiguration** im Projektextplorer und öffnen Sie diese per Doppelklick.

Die Task-Konfiguration wird im Editor geöffnet.

3. Wählen Sie die Registerkarte **Überwachung**.

Siehe auch

- [Kap. "Registerkarte 'Überwachung'" auf Seite 601](#)

13.5 Logbuch der SPS lesen

IndraLogic bietet Ihnen die Möglichkeit, die in der Steuerung geloggtten Ereignisse und Fehlermeldungen anzuzeigen.

Logbuch lesen Voraussetzung: Die Steuerung läuft.

1. Selektieren Sie im Projektextplorer die Steuerung.
2. Wählen Sie den Kontextmenü-Befehl **Öffnen**.

Der Geräteeditor öffnet sich.

3. Wählen Sie die Registerkarte **Log**.

4. Klicken Sie auf  zum Aktualisieren der Ansicht.

Es wird eine Verbindung zur Steuerung aufgebaut.
Alle Log-Informationen der Steuerung werden angezeigt.

5. Klicken Sie auf  zum Löschen der aktuellen Liste.

6. Filtern Sie die Ansicht, indem Sie auf die gewünschte Kategorie klicken (zum Beispiel "Information").

7. Speichern Sie die Log-Einträge. Klicken Sie dazu auf  und wählen Sie einen Dateinamen.

Siehe auch

- [Kap. "Registerkarte 'Log'" auf Seite 525](#)

13.6 Fehleranalyse mit Core Dump

Ein Core Dump ist ein Speicherabzug der Applikationsdaten.

Im Fall eines Ausnahmefehlers auf der Steuerung legt diese automatisch einen Core Dump ab.

Der Core Dump wird in einer Datei `<Applikationsname>.core` im Applikationsverzeichnis gespeichert.

Im Online-Betrieb können Sie auch explizit einen Core Dump erzeugen lassen, wenn die Applikation gerade an einem Haltepunkt steht oder ein Ausnahmefehler aufgetreten ist.

In diesem Fall legt IndraLogic die Core-Dump-Datei nur im Projektverzeichnis und nicht auf der Steuerung ab (z. B. C:\ProgramData\IW-Projects*\Project\IndraLogic\IndraLogic.IndraMotionMlc1.Application.*.core).

Im Offline-Betrieb der Applikation können Sie den Core Dump von der Steuerung ins Projekt laden. Sie erhalten dann eine Online-Ansicht der Applikation mit den Daten und Werten zum Zeitpunkt des Ausnahmefehlers.

Zusätzlich können Sie im Projekt eine Ansicht der Logdatei der Steuerung zum Zeitpunkt der Core-Dump-Erzeugung aufrufen.



Das Laufzeitsystem muss die Funktionalität voll unterstützen, um eine korrekte Anzeige von Variablenwerten in der Coredumpan-sicht sicherzustellen.

In der Online-Ansicht der Applikation, die IndraLogic beim Laden des Core Dumps ins Projekt erzeugt, erscheinen Menübefehle als verfügbar, die in diesem Status nicht wirksam sind. Bei Anwählen eines solchen Befehls erhalten Sie eine entsprechende Meldung.

VORSICHT

Einschränkung (CDS-37572):

Das Exceptionhandling im Laufzeitsystem überschreibt möglicherweise Teile der Variablenwerte im Callstack. Dadurch gehen die ursprünglichen Werte verloren und beim Monitoring wird für Variablen in Funktionen und Methoden unter Umständen keine sinnvolle Information mehr dargestellt.



Für das Erstellen eines Projektarchivs: Wenn Sie einen Core Dump explizit mit dem Befehl **Core Dump erzeugen** erstellt haben, steht automatisch eine Option in der Archivkonfiguration dafür bereit.

Wenn Sie dagegen den Core Dump von der SPS kopiert haben, können Sie ihn nur über die Funktion **Zusätzliche Dateien** zu einem Projektarchiv hinzufügen.

Warnung:

Ein Projektarchiv mit Core Dump muss unbedingt die Download-information enthalten, sonst kann IndraLogic den Dump später nicht mehr verwenden.

Core Dump und Geräte-Log zur Analyse ins Projekt laden

Voraussetzung: Sie haben das Projekt mit der Applikation geöffnet, die auf der Steuerung einen Ausnahmefehler produziert hat. Das Laufzeitsystem ist in der Lage, Core Dumps zu erstellen. Die Applikation im Projekt befindet sich im Offline-Betrieb.

1. Laden Sie mit Befehl **Debuggen ► Core Dump ► Core Dump laden** den gewünschten Core Dump von der Steuerung.

IndraLogic stellt eine Online-Ansicht der Applikation dar. Sie sehen die Variablenwerte und Aufrufliste zur Fehlerzeit (siehe unten). Dies betrifft auch die Werte der E/A-Variablen im Abbild-Dialog des Gerätekonfigurators und die Task-Konfiguration. In der Statuszeile steht "Core Dump geladen".

IndraLogic kopiert die Core-Dump-Datei mit der Benennung <Projektname>.<Gerätename>.<Applikationsname>.<Applikati-

ons-Guid>.core ins lokale Projektverzeichnis (z. B. \...\IndraLogic.IndraMotionMlc1.Application.*.core).

2. Laden Sie mit Befehl **Debuggen ▶ Core Dump ▶ Geräte-Logbuch vom Core Dump laden** den Geräte-Log zum Fehlerzeitpunkt ins Projekt.
Die Log-Ansicht erscheint - wie im Online-Betrieb des Geräte-Editors - mit den Ereignissen zum Zeitpunkt des Core Dumps.
3. Nach Abschluss der Core-Dump-Analyse wählen Sie Befehl **Debuggen ▶ Core Dump ▶ Core Dump schließen**.
IndraLogic schließt die Coredumpansicht der Applikation. Das Projekt erscheint wieder mit den Ansichten des normalen Offline-Betriebs.

Manuelles Erzeugen eines Core Dumps der laufenden Applikation

Voraussetzung: Eine Applikation ist in IndraLogic im Online-Betrieb. Das Laufzeitsystem unterstützt die Core-Dump-Funktionalität.

1. Wählen Sie Befehl **Debuggen ▶ Core Dump ▶ Core Dump erzeugen**.
IndraLogic startet die Erzeugung eines neuen Core-Dumps. In der Statusleiste erscheinen ein Fortschrittsbalken und eine Schaltfläche zum Abbrechen der Erzeugung.
IndraLogic legt eine Core-Dump-Datei <Projektname>.<Gerätename>.<Applikationsname>.<Applikations-Guid>.core im lokalen Projektverzeichnis an (z. B. \...\IndraLogic.IndraMotionMlc1.Application.*.core).

Siehe auch

- [Kap. "Befehl 'Core Dump erzeugen'" auf Seite 674](#)
- [Kap. "Befehl 'Core Dump laden'" auf Seite 675](#)
- [Kap. "Befehl 'Core Dump schließen'" auf Seite 675](#)
- [Kap. "Befehl 'Geräte-Logbuch vom Core Dump laden'" auf Seite 675](#)

13.7 SPS-Shell zur Abfrage von Informationen verwenden

Die "SPS-Shell" in IndraLogic ist ein textbasierter Steuerungsmonitor auf einer Registerkarte des Geräteeditors.

Sie können dort Kommandos zur Abfrage bestimmter Informationen eingeben sowie Aktionen ausführen wie Starten, Stoppen oder Laden von Applikationen. Auch eine Beschreibung zur Bedeutung und Syntax der möglichen Kommandos können Sie direkt aus der SPS-Shell abrufen.

Die möglichen Kommandos sind herstellerabhängig. Der Umfang an Kommandos ist über den sogenannten `SPS Shell Command Handler` entweder in einer C-Komponente im Laufzeitsystem, oder als Funktionsbaustein in einer IEC-Bibliothek oder einem Projekt erweiterbar.

Die Antwort aus der SPS erscheint auf der Registerkarte als Zeichenfolge in einem Ergebnisfenster.

Abfrage von Information zur Applikation auf der Steuerung

Voraussetzung: Ihr Projekt ist mit einer Steuerung verbunden;

Beispiel Steuerung, auf der eine Applikation `App1` läuft.

1. Öffnen Sie den Geräteeditor mit einem Doppelklick auf das Objekt <steuerung> im Projektextplorer und wählen die Registerkarte **SPS-Shell**.

Die Registerkarte erscheint mit einem leeren Anzeigefenster. Darunter finden Sie eine Eingabezeile für ein Kommando

2. Klicken Sie auf die Schaltfläche .
Der Dialog **Standardkommando eingeben** erscheint mit einer Liste mit Kommandos.
3. Wählen Sie das Kommando **?** und klicken auf die Schaltfläche **Ausführen**.
Der Dialog schließt und im Ausgabefenster erscheint eine Liste der verfügbaren Kommandos mit den möglichen Parametern. Sie zeigt jeweils die Syntax der Kommandoeingabe.
4. Klicken Sie erneut auf  und wählen Sie das Kommando **pid**.
Ergänzen Sie in der Eingabezeile folgendermaßen: `pid Appl`. Drücken Sie die Eingabetaste.
Im Ausgabefenster erscheint Folgendes (GUIDs nur als Beispiele):

```
pid Appl
Project Identification
Applikation: Appl
Code GUID:0x08a893c0
Data GUID:0x762d0e90
```
5. Drücken Sie die Schaltfläche  in der Kommandoeingabezeile.
Das Kommando `pid Appl` ist in der Historie der eingegebenen Kommandos aufgenommen.

Siehe auch

- [Kap. "Registerkarte 'SPS-Shell'" auf Seite 532](#)

13.8 SPS-Operationssteuerung über Systemvariablen

VORSICHT

SPS-Operationssteuerung über Systemvariablen

Es liegt in Ihrer Verantwortung, dass Laufzeitsystemdienste in sicheren Applikationszuständen aktiviert und nur in kritischen deaktiviert sind.

Eine Anlage oder Applikation kann zur Laufzeit in einen sensiblen Zustand kommen, in dem störende Aktionen die gesamte Maschine oder Anlage gefährden können.

Sie können aber in diesem Zustand bestimmte Befehle unterdrücken und gefährliche Aktionen verhindern.

Hierfür steht Ihnen der Funktionsbaustein **PlcOperationControl** der Bibliothek **Component Manager** zur Verfügung.

Beispiele von IndraLogic-Befehlen, deren Ausführung unterdrückt werden kann

- Online-Change, Download
- Haltepunkt aktivieren
- Applikation Reset, Applikation Stop
- Dateien übertragen
- Werte forcen, Werte schreiben

Applikation zur Laufzeit

Um immer eine Rückfalllösung zu haben, können Sie die Befehle **Reset Ursprung** und **Löschen** der Applikation nicht unterdrücken.

Wenn zur Laufzeit der Applikation ein Laufzeitsystemdienst angefordert wird, der aber gerade deaktiviert ist, erhalten Sie in IndraLogic eine Meldung darüber. Sie können dann mit einer passenden Gegenmaßnahmen darauf reagieren.

Funktionsbaustein PlcOperationControl zur Operationssteuerung

Der Funktionsbaustein dient der Aktivierung und Deaktivierung von Operationen.

Name	Datentyp	Initialwert	Beschreibung
xDisableAll	BOOL	FALSE	TRUE: Alle Operationen werden unterdrückt
xDisableApplicationBreakpoint	BOOL	FALSE	TRUE: Setzen eines Haltepunkts wird unterdrückt
xDisableApplicationDownload	BOOL	FALSE	TRUE: Download wird unterdrückt
xDisableApplicationForce	BOOL	FALSE	TRUE: Forcen von Variablen wird unterdrückt
xDisableApplicationOnline-Change	BOOL	FALSE	TRUE: Online-Change wird unterdrückt
xDisableApplicationReset	BOOL	FALSE	TRUE: Reset der Applikation (nicht "Reset Ursprung!") wird unterdrückt
xDisableApplicationStop	BOOL	FALSE	TRUE: Applikationsstopp wird unterdrückt
xDisableApplicationWrite	BOOL	FALSE	TRUE: Schreiben von Variablen wird unterdrückt Kann auch über PLCHandler/IecVarAccess unterbunden werden.
xDisableFileTransferDownload	BOOL	FALSE	TRUE: Dateiübertragung (Download) wird unterdrückt
xDisableFileTransferUpload	BOOL	FALSE	TRUE: Dateiübertragung (Upload) wird unterdrückt

Tab. 13-3: Eigenschaft (PROPERTY)

Operationssteuerung implementieren

Voraussetzung

- Compilerversion ab 3.4.3.0
- In der Gerätebeschreibung ist die SPS-Operationssteuerung über Systemvariablen aktiviert.

Beispiel

Deklarieren Sie eine Instanz des Funktionsbausteins

```
VAR
  PlcOpCtrl_Inst : PlcOperationControl;
END_VAR
```

Unterdrücken Sie einen Befehl, indem Sie der zugehörigen Eigenschaft TRUE zuweisen.

Beispiel "Applikation stoppen"

```
PlcOpCtrl_Inst.xDisableApplicationStop := TRUE;
```

Siehe auch

- [Kap. "Befehl 'Online-Change'" auf Seite 657](#)
- [Kap. "Befehl 'Werte forcen'" auf Seite 671](#)
- [Kap. 12.4 "Forcen und Schreiben von Variablen" auf Seite 198](#)

14 Applikation auf der SPS aktualisieren

IndraLogic bietet Ihnen grundsätzlich zwei Möglichkeiten, eine geänderte Applikation auf die Steuerung zu übertragen: "Download" und "Online Change".

Ein **Download** führt zu einem erneuten Übersetzen der Applikation. Dabei wird neben einer Syntax-Prüfung auch Applikationscode erzeugt und auf die Steuerung geladen. Dies führt zu einem Stopp des laufenden Programms. Ein Download ist die empfohlene Art der Datenübertragung, da aufgrund des Programmstopps und der Neuinitialisierung immer ein definierter Ausgangszustand geschaffen wird.

Bei einem **Online Change** werden nur die geänderten Teile neu in die Steuerung geladen. Dabei wird ein laufendes Programm nicht angehalten. Führen Sie einen Online Change nur bei kleinen Änderungen der Applikation aus. Bei umfangreichen Änderungen kann das Verhalten eines Programms nicht sicher vorhergesagt werden. Lesen Sie dazu bitte die Hinweise in der Beschreibung des Befehls **Online Change**.

Siehe auch

- [Kap. 14.1 "Ausführen eines Online Changes" auf Seite 229](#)
- [Kap. "Befehl 'Online-Change'" auf Seite 657](#)
- [Kap. 14.2 "Ausführen eines Downloads" auf Seite 230](#)
- [Kap. "Befehl 'Laden'" auf Seite 656](#)

14.1 Ausführen eines Online Changes

IndraLogic bietet Ihnen automatisch einen "Online Change" an, wenn Sie mit einer Applikation einloggen, die bereits auf der Steuerung vorhanden ist, aber seit dem letzten Download im Programmiersystem verändert wurde.

Bei diesem Vorgang werden nur die geänderten Teile neu in die Steuerung geladen. Ein laufendes Programm auf der Steuerung wird beim Online Change nicht angehalten.



Beachten Sie beim Online Change die Hinweise in der Beschreibung des Befehls **Online Change**.

Siehe auch

- [Kap. "Befehl 'Online-Change'" auf Seite 657](#)
- [Kap. "Befehl 'Einloggen'" auf Seite 652](#)
- [Kap. 11.2 "Applikationscode erzeugen" auf Seite 187](#)

Online Change beim Einloggen

Voraussetzung: Die Verbindungseinstellungen der Steuerung sind korrekt eingestellt. Die Applikationen im Projekt und auf der Steuerung sind identisch. Das Programm auf der Steuerung läuft. Die Applikation ist ausgeloggt.

1. Verändern Sie Ihre Applikation.
2. Wählen Sie den Befehl **Debuggen ► Einloggen**

Es erscheint ein Dialog mit dem Hinweis, dass die Applikation seit dem letzten Download geändert wurde.

3. Klicken Sie auf die Schaltfläche **Details...**
4. Prüfen Sie die Angaben in der Registerkarte **Applikationsinformationen**.

Wenn Sie seit der letzten Änderung keinen Code erzeugt haben, ist der Befehl **Applikation nicht aktuell. Code jetzt erzeugen?** am unteren Rand des Dialogs sichtbar. Klicken Sie in diesem Fall auf diesen Befehl.

Sie erhalten eine Vergleichsansicht der Objekte - rot markierte Objekte sind unterschiedlich.

5. Schließen Sie den Dialog.
6. Wählen Sie die Option **Mit Online Change einloggen** und Klicken Sie auf **OK**.

Die Änderung wird auf die Steuerung geladen. Das laufende Programm auf der Steuerung wird dabei nicht angehalten. Die Applikation ist eingeloggt.

Online Change im eingeloggten Zustand (Online-Betrieb)

Voraussetzung: Die Verbindungseinstellungen der Steuerung sind korrekt eingestellt. Die Applikationen im Projekt und auf der Steuerung sind identisch. Das Programm auf der Steuerung läuft. Die Applikation ist eingeloggt.

1. Selektieren Sie ein Objekt im Projektexplorer. Hier wählen Sie am besten eine POU oder GVL.
2. Wählen Sie den Kontextmenü-Befehl **Objekt (offline) bearbeiten**. Das Objekt öffnet im Editor.
3. Verändern Sie das Objekt. Hier können Sie beispielsweise eine neue Variable deklarieren oder eine Wertzuweisung ändern.
4. Wählen Sie den Befehl **Debuggen ► Online Change**.
Sie erhalten eine Abfrage, ob Sie den Online Change wirklich ausführen wollen.
5. Bestätigen Sie den Dialog mit **Ja**.
Die Änderung wird auf die Steuerung geladen.

14.2 Ausführen eines Downloads

Ein Download der Applikation bewirkt ein Übersetzen der aktiven Applikation. Dabei wird neben einer Syntax-Prüfung auch Applikations-Code erzeugt und auf die Steuerung geladen. Ein laufendes Programm auf der Steuerung wird beim Download angehalten.



Beachten Sie beim Download die Hinweise in der Beschreibung des Befehls **Debuggen ► Laden**.

Siehe auch

- [Kap. "Befehl 'Laden'" auf Seite 656](#)
- [Kap. "Befehl 'Einloggen'" auf Seite 652](#)
- [Kap. 11.2 "Applikationscode erzeugen" auf Seite 187](#)

Download beim Einloggen

Voraussetzung: Die Verbindungseinstellungen der Steuerung sind korrekt eingestellt. Die Applikationen im Projekt und auf der Steuerung sind identisch. Das Programm auf der Steuerung läuft. Die Applikation ist ausgeloggt.

1. Verändern Sie Ihre Applikation.
2. Wählen Sie den Befehl **Debuggen ► Einloggen**

Es erscheint ein Dialog mit dem Hinweis, dass die Applikation seit dem letzten Download geändert wurde.

3. Wählen Sie die Option **Mit Download einloggen** und Klicken Sie auf **OK**.
Das laufende Programm auf der Steuerung wird angehalten und die Änderung auf die Steuerung geladen. Die Applikation ist eingeloggt.

Download im eingeloggten Zustand (Online-Betrieb)

Voraussetzung: Die Verbindungseinstellungen der Steuerung sind korrekt eingestellt. Die Applikationen im Projekt und auf der Steuerung sind identisch. Das Programm auf der Steuerung läuft. Die Applikation ist eingeloggt.

1. Selektieren Sie eine Objekt im Projektextplorer. Hier wählen Sie am besten eine POU oder GVL.
2. Wählen Sie den Kontextmenü-Befehl **Objekt (offline) bearbeiten**
Das Objekt öffnet im Editor.
3. Verändern Sie das Objekt. Hier können Sie beispielsweise eine neue Variable deklarieren oder eine Wertzuweisung ändern.
4. Wählen Sie den Befehl **Debuggen ► Laden**.
Sie erhalten eine Abfrage, ob Sie den Download wirklich ausführen wollen.
5. Bestätigen Sie den Dialog mit **Ja**.
Das laufende Programm auf der Steuerung wird angehalten und die Änderung auf die Steuerung geladen.

15 Dateien von/zur SPS kopieren

IndraWorks/ IndraLogic erlaubt es Dateien zwischen Ihrem lokalen Dateisystem und der Steuerung in beiden Richtungen kopieren.

Voraussetzung: Im Projektextplorer Ihres Projekts ist die Verbindung mit dem Steuerungsgerät konfiguriert. Das Gerät läuft.

1. Öffnen Sie das Kontextmenü **<steuerung> ▶ Gerätedaten ▶ Explorer** im Projektextplorer.

2. Stellen Sie in der linken Dialoghälfte (**PC**) bei **Adresse** den Pfad in Ihrem lokalen Dateisystem ein, aus dem oder in das Dateien kopiert werden sollen. Beispiel: D:\FileTransferWithPLC. Wenn nötig, legen Sie über die Schaltfläche  direkt ein neues Verzeichnis an.

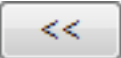
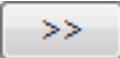
Wie aus dem Datei-Manager bekannt, werden die Verzeichnisse und

Dateien dargestellt. Mit der Schaltfläche  können Sie die Darstellung aktualisieren.

3. Stellen Sie auf gleiche Weise in der rechten Dialoghälfte (**Steuerung**) das gewünschte Verzeichnis für den Dateitransfer ein.

IndraLogic zeigt die auf der Steuerung vorliegenden Dateien an.

4. Wählen Sie nun für einen Dateitransfer im jeweiligen Dateisystembaum die gewünschten Dateien aus. Mehrfachauswahl ist möglich. Sie können auch ein Verzeichnis auswählen, wenn alle enthaltenen Dateien übertragen werden sollen.

5. Betätigen Sie die Schaltfläche  beziehungsweise  zwischen den Dialoghälften.

IndraLogic kopiert die ausgewählten Dateien unmittelbar ins andere Dateisystem. Wenn eine Datei im Zielverzeichnis noch nicht vorhanden ist, wird sie dort neu angelegt. Wenn sie bereits vorliegt, wird sie überschrieben, außer sie ist schreibgeschützt. Dann erscheint eine entsprechende Meldung.

Beispiele: *Siehe auch:*

- RIL_ProfinetIO.library/ IL_PNIODDeviceBackup
- RIL_ProfinetIO.library/ IL_PNIOIoLinkBackupDevice
- RIL_EthernetIP.library/ IL_ENIPAdapterBackup

16 Bibliotheken erzeugen und verwalten

Bibliotheksrepository Das Bibliotheksrepository ist der Ablageort für Bibliotheken und den zugehörigen Metadaten auf dem Entwicklungssystem. Sie können die dort installierten Bibliotheken in Ihr Projekt in einen Bibliotheksverwalter einbinden. Außerdem werden die Bibliotheken versioniert installiert, um ein Aktualisieren der Bibliotheken zu ermöglichen.

Sie können neben dem vorinstallierten Repository System weitere Repositories anlegen und bearbeiten.

Siehe [Kap. "Befehl 'Bibliotheks-Repository'" auf Seite 676](#)

Bibliotheksverwalter Der Bibliotheksverwalter verwaltet die im Projekt eingebundenen Bibliotheken. Er stellt die Bibliotheken entsprechend dem jeweiligen Bibliothekstypen und den zugehörigen Eigenschaften dar. Über den Bibliotheksverwalter können weitere Bibliotheken eingebunden, entfernt, oder deren Eigenschaften bearbeitet werden.

Der Bibliotheksverwalter kann sowohl im Projekt-Ordner "Allgemeiner Baustein", oder im "Application" Knoten der Steuerung platziert sein. Somit kann ein Projekt pro Applikation einen Bibliotheksverwalter haben und zusätzlich einen applikationsübergreifenden im Projekt-Ordner "Allgemeiner Baustein". Die Bibliotheksbausteine der eingebundenen Bibliotheken im Projektextplorer-Ordner "Allgemeiner Baustein" können applikationsunabhängig aufgerufen werden. Die Bibliotheksbausteine der eingebundenen Bibliotheken im Steuerungsknoten "Application" können nur im zugehörigen Applikationscode aufgerufen werden. Außerdem verhalten sich Platzhalterbibliotheken beim Laden einer Applikation abhängig von Ihrer Platzierung unterschiedlich.

Bibliotheken, die mit einer bestimmten Version in das Projekt eingebunden sind, haben zusätzlich einen Platzhalter auf diese Version (Platzhalterbibliothek). Sie können spezielle Platzhalteraufösungen definieren. Sie können aber auch die Platzhalterauflösung verwenden, die für ein Gerät in der Gerätebeschreibung definiert ist oder die bei einer Bibliothek im Bibliotheksrepository abgelegt ist. Der Bibliotheksverwalter informiert über die tatsächliche Platzhalterauflösung und zeigt die Version, die beim Laden einer Applikation geladen werden wird, an (effektive Version).

Wenn einen Bibliotheksverwalter im Projekt-Ordner "Allgemeiner Baustein" applikationsübergreifend eingebunden ist, können Sie auf dessen Inhalt global zugreifen. Wenn dabei Platzhalterbibliotheken eingebunden sind, werden nur die Platzhalteraufösungen in der Gerätebeschreibung oder aus dem Bibliotheksrepository geprüft.

Üblicherweise ist ein Bibliotheksverwalter im "Application" Knoten der Steuerung eingebunden. Dann ruft nur der darunter liegende Applikationscode Bibliotheksbausteine daraus auf. Außerdem werden bei Platzhalterbibliotheken zuerst die speziellen Platzhalteraufösungen geprüft. Erst danach werden die Platzhalteraufösungen, die in der Gerätebeschreibung sind, oder aus dem Bibliotheksrepository stammen geprüft.

Siehe [Kap. 18.1.14 "Objekt 'Bibliotheksverwalter'" auf Seite 548](#)

16.1 Informationen für Bibliotheksentwickler



Für das Erstellen, das Referenzieren, das Verschlüsseln und Schützen von Bibliotheken, wie auch das Dokumentieren, sollten Sie unbedingt bestimmte Regeln einhalten, um Konsistenzprobleme zu vermeiden und den Anwender ausreichend zu unterstützen.

Die nachfolgenden Beschreibung soll Ihnen einen Überblick über die Möglichkeiten der Bibliotheksentwicklung geben. Sie können eine weiterführende detaillierte Beschreibung zu diesen Themen in den Richtlinien für Bibliotheksentwicklung "LibDevSummary" finden.

Allgemein

- Sie können für Bibliotheken Kategorien definieren. Die Bibliotheken werden dann im Bibliotheks-Repository unterhalb dieser Kategorien dargestellt.
- Sie können für eine Bibliothek einen Namensraum definieren, um einen eindeutigen Zugriff auf die eingebundenen Objekte zu ermöglichen. Durch das Hinzufügen des Namensraums vor dem Bausteinnamen wird ein Zugriff eindeutig:
`<namespace>.<variable name>`
 Beispiel: `AC.Module`
- Sie können die POUs nicht verschlüsselter Bibliotheken (`*.library`) durch einen Doppelklick auf den jeweiligen Eintrag im Bibliotheksverwalter öffnen.
- Sie können folgende Bibliothekstypen erzeugen:
 - `*.library`: Implementierungsbibliothek
 - `*.compiled-library`: Geschützte Bibliothek; der Quellcode ist nicht mehr zugänglich
 - `*._Itfs.library`: Interface-Bibliothek
 - `*._Cnt.library`: Container-Bibliothek
- Sie können externe Bibliotheken in die Applikation einbinden. Externe Bibliotheken werden außerhalb von IndraLogic in einer anderen Programmiersprache, wie beispielsweise C, programmiert.

Schutz von Bibliotheken

- Quellcode-Schutz:
 Durch das Bereitstellen einer Bibliothek im "compiled-library"-Format ist der Quellcode der Bibliotheksbausteine nach Einbinden der Bibliothek in ein Projekt nicht mehr sichtbar.
- Signierung:
 Bibliotheken können mit einem privaten Schlüssel und einem zugehörigen Token signiert werden. Die Signierung kann im Objekt "Bibliothek-sinfo" in der Registerkarte Signieren durchgeführt werden. Das Objekt "Bibliothek-sinfo" kann im Ordner "Allgemeiner Bausteinteil" im Projekt-explorer hinzugefügt werden.

Bibliotheksversionen

- Sie können gleichzeitig mehrere Versionen einer Bibliothek auf dem System installieren.

- Sie können gleichzeitig mehrere Versionen einer Bibliothek im Projekt einbinden (nicht empfohlen).

Referenzierte Bibliotheken

- Sie können eine Bibliothek in andere Bibliotheken einbinden (referenzierte Bibliotheken). Dabei kann die Verschachtelung beliebig tief sein.
- Sie können festlegen, ob referenzierte Bibliotheken im Bibliotheksverwalter sichtbar sein sollen.
- Sie können referenzierte Bibliotheken über Bibliotheks-Platzhalter einbinden. Damit vermeiden Sie Probleme, die aufgrund von Versionsabhängigkeiten oder der Notwendigkeit, herstellerspezifische Bibliotheken zu verwenden, entstehen können.



Die Verwendung von Parameterlisten ist nur dann erlaubt, wenn sich die Bibliothek an oberster Stelle in der Hierarchie befindet.

16.2 Hinzufügen einer Bibliothek zur Applikation

Die nachfolgende Anweisung beschreibt, wie Sie die Bibliothek `Util` in Ihre Applikation einbinden.

1. Selektieren Sie den Bibliotheksverwalter und öffnen Sie per Doppelklick. Der Bibliotheksverwalter öffnet im Editor.
2. Führen Sie im Bibliotheksverwalter den Befehl **Bibliothek hinzufügen** aus. Es öffnet sich der Dialog **Bibliothek hinzufügen**.
3. Suchen Sie die Bibliothek, indem Sie die Zeichenfolge "util" in das Eingabefeld oben eingeben. Es erscheint die Bibliothek `Util` in der Bibliotheksansicht.
4. Wählen Sie die Bibliothek `Util` aus und schließen Sie den Dialog mit **OK**. Die Bibliothek `Util` ist im Bibliotheksverwalter hinzugefügt.

Siehe auch

- [Kap. "Befehl 'Bibliothek hinzufügen'" auf Seite 714](#)
- [Kap. 16.3 "Hinzufügen einer Bibliothek zum Repository" auf Seite 237](#)

16.3 Hinzufügen einer Bibliothek zum Repository

Die nachfolgende Anweisung beschreibt, wie Sie eine Bibliothek in das Bibliotheks-Repository installieren.

1. Wählen Sie den Befehl **Extras ► Bibliotheken ► Bibliotheks-Repository**. Es öffnet der Dialog **Bibliotheks-Repository**.
2. Klicken Sie auf die Schaltfläche **Installieren**.
3. Wählen Sie die Bibliothek aus, die Sie installieren wollen. Dabei können Sie einen Dateifilter setzen. Klicken Sie auf **Öffnen**. Die Bibliothek wird im Repository hinzugefügt. Die Bibliothek kann nun im Bibliotheksverwalter hinzugefügt werden.

Siehe auch

- [Kap. 16.2 "Hinzufügen einer Bibliothek zur Applikation" auf Seite 237](#)

16.4 Bibliotheksdatei exportieren

Sie können eine Bibliothek aus dem Bibliotheksverwalter eines Projekts oder aus dem Bibliotheksrepository exportieren und als Datei auf der Festplatte ablegen.

1. Wählen Sie den Befehl **Extras ► Bibliotheken ► Bibliotheks-Repository**.
Es öffnet der Dialog **Bibliotheks-Repository**.
2. Selektieren Sie im Fenster der installierten Bibliotheken eine Bibliotheksversion.
3. Klicken Sie auf die Schaltfläche "Exportieren" .
Der Dialog "Bibliothek exportieren" öffnet sich.
4. Wenn die ausgewählte Bibliothek nicht nur als übersetzte Bibliothek, sondern auch im "Source-Format" im Projekt eingebunden ist, sind beide Dateitypen in der Auswahlliste bei "Dateityp", ansonsten zeigt der Filter automatisch den vorliegenden Typ an: Bibliotheken (*.library) oder Übersetzte Bibliotheken (*.compiled-library).
5. Wählen Sie den Dateityp und Ablageort und klicken Sie auf die Schaltfläche "Speichern".

Siehe auch:

[Kap. 18.1.14 "Objekt 'Bibliotheksverwalter'" auf Seite 548](#)

[Kap. "Befehl 'Bibliotheks-Repository'" auf Seite 676](#)

17 Referenz Programmierung

17.1 Programmiersprachen und ihre Editoren

Sie programmieren eine POU jeweils im Editor für die Implementierungssprache, die Sie beim Anlegen der POU ausgewählt haben. IndraLogic bietet einen Texteditor für ST und grafische Editoren für AS, FUP/KOP/AWL und CFC.

Der Editor öffnet sich beim Doppelklick auf die POU im Projektexplorer unterhalb von **Application** oder im Ordner "Allgemeiner Baustein".

Jeder der Programmiersprachen-Editoren besteht aus zwei Teilfenstern:

- Im oberen Teil nehmen Sie im "Deklarationseditor" die Deklarationen vor, je nach Einstellung in textueller oder tabellarischer Form.
- Im unteren Teil fügen Sie den Implementierungscode in der jeweiligen Sprache ein.

Die Darstellung und das Verhalten jedes Editors können Sie projektweit in der zugehörigen Registerkarte der IndraLogic-Optionen konfigurieren.

17.1.1 Deklarationseditor

Der Deklarationseditor dient der Deklaration von Variablen in Variablenlisten und Bausteinen.

Wenn der Deklarationseditor in Verbindung mit einem Programmierspracheneditor verwendet wird, öffnet er sich in einem Fenster oberhalb des Programmierspracheneditors.

Der Deklarationseditor bietet zwei mögliche Ansichten:  textuell und  tabellarisch.

Im Dialog **Extras ► IndraLogic 2G ► Optionen ► Deklarationseditor** definieren Sie, ob entweder nur die textuelle oder nur die tabellarische Ansicht verfügbar ist oder ob Sie über die Schaltflächen am rechten Rand des Editorfensters zwischen den beiden Ansichten wechseln können.

In der textuellen Ansicht des Deklarationseditors ist eine Rechteck-Selektion möglich. Die Tastenkombinationen für die Rechteckselektion finden Sie auf der Hilfeseite zum ST-Editor.

Siehe auch

- [Kap. 9.2 "Variablen deklarieren" auf Seite 85](#)
- [Kap. 17.1 "Programmiersprachen und ihre Editoren" auf Seite 239](#)
- [Kap. "Dialog 'Optionen- 'Deklarationseditor'" auf Seite 766](#)
- [Kap. "ST-Editor" auf Seite 241](#)

Deklarationseditor im Onlinebetrieb

Im Onlinebetrieb sehen Sie die tabellarische Ansicht des Editors.

Die Kopfzeile enthält immer den aktuellen Objektpfad:

`<Gerätename>.<Applikationsname>.<Objektname>.`

Die Tabelle enthält zusätzlich zum Offlinebetrieb auch die Spalten **Wert** und **Vorbereiteter Wert**.

- Die Spalte **"Wert"** zeigt den Istwert auf der Steuerung, bietet also Monitoring-Funktionalität. Wenn der Ausdruck ein Array ist, können Sie den Bereich der zu monitorenden Arrayindizes festlegen. Dazu öffnen Sie mit einem Doppelklick in die Spalte "Datentyp" den Dialog "Monitoring-Bereich" und tragen "Start-Index" und "End-Index" des gewünschten

Bereichs ein. Mit Hilfe einer Scrollleiste können Sie bequem einen Bereich von 1000 Elementen definieren.

- Die Spalte **"Vorbereiteter Wert"** enthält den Wert, den Sie gegebenenfalls zum Forcen oder Schreiben vorbereitet haben. Wenn Sie auf ein Feld in der Spalte „Vorbereiteter Wert“ doppelklicken, können Sie direkt einen Wert für das Schreiben oder Forcen eingeben. Bei Enumerationen öffnet sich eine Combobox, aus der Sie einen Wert auswählen können. Bei einer booleschen Variable können Sie den vorbereiteten Wert mit Hilfe der Tasten <Eingabe>, oder <Leerzeichen> toggeln. Wenn ein Ausdruck (Variable) von einem strukturierten Datentyp ist, beispielsweise die Instanz eines Funktionsbausteins oder eine Arrayvariable, dann ist ein Plus- oder Minuszeichen vorangestellt.

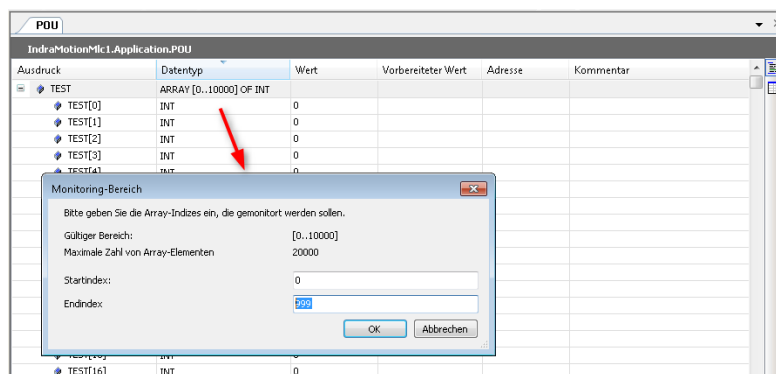
Sie können das Format der Darstellung von Gleitpunktwerten in den Optionen für Monitoring anpassen, siehe [Kap. 18.3.5 "Dialog 'Monitoringbereich'"](#) auf Seite 748.



Bei Arrays ist der Monitoring-Bereich standardmäßig auf 1000 Array-Indizes (0-999) eingestellt.

Bei Bedarf kann der Bereich auf bis zu 20000 Array-Elemente erweitert werden.

Öffnen Sie hierzu den Dialog zur Einstellung des Monitoring Bereich durch Doppelklick in der Spalte "Datentyp"



Siehe auch

- [Kap. 18.3.3 "Dialog 'Wert vorbereiten'"](#) auf Seite 745
- ["Forcen im Deklarationsteil"](#) auf Seite 198

17.1.2 Generelle Funktionalitäten in allen grafischen Editoren

Der Implementierungsteil der grafischen Editoren für FUP, KOP, CFC und AS enthält in der rechten unteren Ecke eine Symbolleiste:

	Zurück in den normalen Editiermodus: Der Mauszeiger bekommt die standardmäßige Pfeilform. Sie können wieder Elemente im Editorfenster auswählen und bearbeiten.
	Schwenkwerkzeug: Der Mauszeiger bekommt die Form zweier gekreuzter Pfeile. Sie können nun an beliebiger Stelle ins Editorfenster klicken, die Maustaste gedrückt halten und dann mit der Maus den sichtbaren Bereich des FUP/KOP/AWL-Editors oder des CFC-Diagramms innerhalb des Fensters schwenken/verschieben.

	Lupenwerkzeug: Es öffnet sich ein zusätzliches Lupenfenster in der unteren rechten Ecke des Editorfensters und der Mauszeiger bekommt die Form zweier gekreuzter Pfeile. Solange Sie nun den Mauszeiger über Ihr Diagramm bewegen, zeigt die Lupe jeweils den Bereich des Diagramms um den Zeiger 100% Größe. Hinweis: Wenn Sie nun ins Fenster klicken, wird die Lupe geschlossen und der Teil des Diagramms, der in ihr zu sehen gewesen war, in 100% Größe angezeigt. Wenn Sie den vorher eingestellten Zoomfaktor beibehalten wollen, sollten Sie deshalb  verwenden, um zurück in den normalen Editiermodus zu gelangen.
	Vergrößerungs-/Verkleinerungswerkzeug: Die Schaltfläche öffnet ein Untermenü zur Auswahl eines Zoomfaktors. Die Auswahl der drei Punkte . . . öffnet den Dialog Zoom, in dem Sie einen anderen Wert angeben können. Der aktuelle Zoomfaktor wird immer links von der Schaltfläche angezeigt.

Vergrößern/Verkleinern mit dem Mausrad: Wenn Sie die Taste <Strg> drücken, während Sie das Mausrad bewegen, wird der Zoomfaktor stufenweise um 10% verändert.

Jeder grafische Editor bringt eine Werkzeugbox (**Ansicht ► Andere Fenster ► Werkzeuge**) mit, standardmäßig rechts vom Editorfenster.

Die Werkzeugbox enthält die verfügbaren Elemente, die Sie mit der Maus auf die jeweiligen Einfügepositionen ins Editorfenster ziehen können.

Die jeweiligen Einfügepositionen hebt IndraLogic mit grauen rauten-, dreieck- oder pfeilförmigen Positionsmarken hervor. Sobald der Mauszeiger über einer dieser Marken steht, wird diese Marke grün. Wenn Sie nun die Maustaste loslassen, fügt IndraLogic das Element an der aktuellen Position ein.

Das Verschieben von Elementen im Editor ist ebenfalls mit der Maus möglich.

Sie können in den grafischen Editoren FUP, KOP und CFC Deklarationen von Funktionsbausteinen in das Editorfenster ziehen. Selektieren Sie dazu die vollständige Deklaration (Variablenname + Datentyp) und ziehen Sie diese auf eine geeignete Stelle im Editorfenster. Beim Kontaktplan können Sie zusätzlich boolesche Deklarationen in den Editor ziehen und als Kontakt einfügen.

Siehe auch

- [Kap. "AS-Editor" auf Seite 253](#)
- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "CFC-Editor" auf Seite 288](#)

17.1.3 Strukturierter Text und Erweiterter Strukturierter Text (ExST)

ST-Editor

Der ST-Editor ist ein Texteditor und dient der Implementierung von Code in strukturiertem Text (ST) und erweitertem strukturiertem Text (ExST).

Am linken Rand des Editors befindet sich die Zeilennummerierung.

Bei der Eingabe der Programmier Elemente werden Sie von der Funktionalität "Komponenten auflisten" und der Eingabehilfe [F2] unterstützt.

Wenn der Cursor auf einer Variablen steht, zeigt IndraLogic ein Tooltip mit Informationen zur Deklaration der Variablen an.

Eine Rechteckselektion kann mit folgenden Tastenkombinationen ausgeführt werden:

- [strg] + [Pfeil nach rechts]: Der selektierte Bereich wird um eine Stelle nach rechts erweitert

- [strg] + [Pfeil nach links]: Der selektierte Bereich wird um eine Stelle nach links erweitert
- [strg] + [Pfeil nach oben]: Der selektierte Bereich wird um eine Zeile nach oben erweitert
- [strg] + [Pfeil nach unten]: Der selektierte Bereich wird um eine Zeile nach unten erweitert

Das Verhalten (wie zum Beispiel Klammerung, Mausaktionen, Tabulatoren) und Aussehen des Editors konfigurieren Sie in den IndraLogic-Optionen im Menü: **Extras ▶ Optionen ▶ IndraLogic 2G** in der Kategorie **Texteditor**.

IndraLogic kann Syntaxfehler bereits während der Eingabe im Editor erkennen und eine entsprechende Meldung mittels Tooltip anzeigen. Hierzu muss in den IndraLogic-Optionen die Einstellung "Fehler im Editor unterstreichen" aktiviert sein. Sie finden die Einstellung im Menü: **Extras ▶ Optionen ▶ IndraLogic 2G** Kategorie "Intelligentes Codieren".

Siehe auch

- [Kap. "Programmieren in Strukturiertem Text \(ST\)" auf Seite 105](#)
- [Kap. "ST-Ausdrücke" auf Seite 242](#)
- [Kap. "ST - Kommentare" auf Seite 252](#)
- [Kap. 17.1.3 "Strukturierter Text und Erweiterter Strukturierter Text \(ExST\)" auf Seite 241](#)
- [Kap. "Dialog 'Optionen- 'Texteditor'" auf Seite 776](#)

ST-Editor im Online-Betrieb

Im Online-Betrieb zeigt IndraLogic im ST-Editor die verwendeten Variablen und Ausdrücke an.

Das Schreiben und Forcen der Variablen und Ausdrücke und Debugging-Funktionen (Haltepunkte, Einzelschrittabarbeitung) sind ebenfalls möglich.

Wenn Sie in der ST-Programmierung Zuweisungen als Ausdrücke verwenden, entstehen keine weiteren Haltepunkt-Positionen innerhalb einer Zeile.

Siehe auch

- [Kap. 13.1.1 "Monitoring in Programmierobjekten aufrufen" auf Seite 205](#)
- [Kap. 12.4 "Forcen und Schreiben von Variablen" auf Seite 198](#)
- [Kap. 13.1.2 "Überwachungslisten verwenden" auf Seite 211](#)
- [Kap. 12.2 "Verwenden von Haltepunkten" auf Seite 193](#)
- [Kap. 12 "Testen und Fehler beheben" auf Seite 193](#)
- [Kap. 12.3 "Schrittweises Abarbeiten eines Programms \(Stepping\)" auf Seite 196](#)
- [Kap. 12.6 "Ablaufkontrolle" auf Seite 201](#)
- [Kap. 12.7 "Aktuelle Abarbeitungsposition mit der Aufrufliste bestimmen" auf Seite 204](#)

ST-Ausdrücke

Ein Ausdruck ist ein Konstrukt, das nach seiner Auswertung einen Wert zurück liefert.

Ausdrücke sind zusammengesetzt aus Operatoren und Operanden.

Im Erweiterten Strukturierten Text (ExST) können Sie auch Zuweisungen als Ausdruck verwenden.

Ein Operand kann eine Konstante, eine Variable, ein Funktionsaufruf oder ein weiterer Ausdruck sein.

Beispiel:

Bespiele

2014	(* Konstante *)
ivar	(* Variable *)
fct(a,b)	(* Funktionsaufruf *)
(x*y)/z	(* Ausdruck *)
real_var2 := int.var;	(* in ExST: Zuweisung *) *)

Siehe auch

- [Kap. "Programmieren in Strukturiertem Text \(ST\)" auf Seite 105](#)

Auswertung von Ausdrücken

Die Auswertung eines Ausdrucks erfolgt durch Abarbeitung der Operatoren nach bestimmten Bindungsregeln.

IndraLogic arbeitet den Operator mit der stärksten Bindung zuerst ab. Operatoren mit gleicher Bindungsstärke werden von links nach rechts abgearbeitet.

Operation	Symbol	Bindungsstärke
Einklammern	(Ausdruck)	Stärkste Bindung
Funktionsaufruf	Funktionsname (Parameterliste) alle Operatoren mit Syntax: <operator> ()	
Potenzieren	EXPT	
Negieren	-	
Komplementbildung	NOT	
Multiplizieren	*	
Dividieren	/	
Modulo	MOD	
Addieren	+	
Subtrahieren	-	
Vergleichen	<,>,<=,>=	
Gleichheit	=	
Ungleichheit	<>	
Bool AND	AND AND_THEN	
Bool XOR	XOR	
Bool OR	OR OR_ELSE	Schwächste Bindung

Siehe auch

- [Kap. 17.3 "Operatoren" auf Seite 315](#)

Zuweisungen

ST-Zuweisungsoperator

Syntax:

`<operand> := <expression>`

Dieser Zuweisungsoperator führt die gleiche Funktion aus wie der `MOVE`-Operator.

Siehe auch

- [Kap. 17.3.6 "Operator 'MOVE'" auf Seite 322](#)

ST-Zuweisungsoperator für Ausgänge

Der Zuweisungsoperator `=>` weist den Ausgang einer Funktion, eines Funktionsbausteins oder einer Methode einer Variablen zu. Der Platz rechts vom Operator kann auch leer sein.

Syntax: `<output> => <variable>`

Beispiel:

```
FBcomp_Output1 => bVar1;
FBcomp_Output2 => ;
```

`FBcom_Output1` und `FB_Output2` sind Ausgänge eines Funktionsbausteins. Der Wert von `FBcom_Output1` wird der Variablen `bVar1` zugewiesen.

ExST-Zuweisung 'S='

Wenn der Operand der Set-Zuweisung auf `TRUE` schaltet, bewirkt die Zuweisung, dass der Variablen links des Operators ein `TRUE` zugewiesen wird. Die Variable wird gesetzt.

`<variable name> S= <operand name> ;`

Die Variablen und der Operand haben den Datentyp `BOOL`.

Beispiel

```
PROGRAM PLC_PRG
VAR
    xOperand: BOOL := FALSE;
    xSetVariable: BOOL := FALSE;
END_VAR

xSetVariable S= xOperand;
```

Wenn der Operand `xOperand` von `FALSE` auf `TRUE` schaltet, wird auch der Variable `xSetVariable` ein `TRUE` zugewiesen. Dann aber behält die Variable diesen Zustand, auch wenn der Operand weiterhin seinen Zustand wechselt.

Mehrfachzuweisungen



Bei Mehrfachzuweisungen innerhalb einer Codezeile werden die einzelnen Zuweisungen nicht von rechts nach links abgearbeitet, sondern alle Zuweisungen beziehen sich auf den Operanden am Codezeilenende.

Beispiel

```
FUNCTION funCompute : BOOL
VAR_INPUT
```

```

    xIn : BOOL;
END_VAR
IF xIn = TRUE THEN
    funCompute := TRUE;
    RETURN;
END_IF

PROGRAM PLC_PRG
VAR
    xSetVariable: BOOL;
    xResetVariable: BOOL := TRUE;
    xVar: BOOL;
END_VAR
xSetVariable S= xResetVariable R= funCompute(xIn := xVar);

```

Die Variable `xResetVariable` erhält die R=-Zuweisung des Rückgabewerts von `funCompute`.

Die Variable `xSetVariable` erhält die S=-Zuweisung des Rückgabewerts von `funCompute`, aber nicht von `xResetVariable`.

Siehe auch

- [Kap. 9.3.3 "Strukturierter Text \(ST\), Erweiterter Strukturierter Text \(ExST\)" auf Seite 104](#)
- [Kap. "ExST-Zuweisung 'R=' auf Seite 245](#)

ExST-Zuweisung 'R='

Wenn der Operand der Reset-Zuweisung auf `TRUE` schaltet, bewirkt die Zuweisung, dass der Variablen links des Operators ein `FALSE` zugewiesen wird. Die Variable wird zurückgesetzt.

`<variable name> R= <operand name> ;`

Beispiel

```

VAR
    xOperand: BOOL := FALSE;
    xResetVariable: BOOL := TRUE;
END_VAR

xResetVariable R= xOperand;

```

Wenn der Operand `xOperand` von `FALSE` auf `TRUE` schaltet, wird auch der Variablen `xResetVariable` ein `FALSE` zugewiesen. Dann aber behält die Variable ihren Zustand, auch wenn der Operand weiterhin seinen Zustand wechselt.

Mehrfachzuweisungen



Bei Mehrfachzuweisungen innerhalb einer Codezeile werden die einzelnen Zuweisungen nicht von rechts nach links abgearbeitet, sondern alle Zuweisungen beziehen sich auf den Operanden am Codezeilenende.

Beispiel

```

FUNCTION funCompute : BOOL
VAR_INPUT
    xIn : BOOL;
END_VAR
IF xIn = TRUE THEN
    funCompute := TRUE;
    RETURN;
END_IF

PROGRAM PLC_PRG
VAR
    xSetVariable: BOOL;
    xResetVariable: BOOL := TRUE;

```

Referenz Programmierung

```

xVar: BOOL;
END_VAR
xSetVariable S= xResetVariable R= funCompute(xIn := xVar);

```

Die Variable `xResetVariable` erhält die `R=`-Zuweisung des Rückgabewerts von `funCompute`.

Die Variable `xSetVariable` erhält die `S=`-Zuweisung des Rückgabewerts von `funCompute`, **aber nicht** von `xResetVariable`.

Siehe auch

- [Kap. 9.3.3 "Strukturierter Text \(ST\), Erweiterter Strukturierter Text \(ExST\)" auf Seite 104](#)
- [Kap. "ExST-Zuweisung 'S=' auf Seite 244](#)

ExST-Zuweisung '=' als Ausdruck

Im ExST erlaubt IndraLogic in Erweiterung zur Norm IEC 61131-3 die Verwendung von Zuweisungen als Ausdrücke.

Beispiel

<code>int_var1:= int_var2:= int_var3 + 9;</code>	(* int_var1 und int_var2 erhalten Wert von int_var3 + 9 *)
<code>real_var1:= real_var2:= int_var;</code>	(* real_var1 und real_var2 erhalten den Wert von int_var *)
<code>int_var:= real_var1:= int_var;</code>	(* fehlerhafte Zuweisung, die Datentypen stimmen nicht überein! *)
<code>IF b:= (i = 1) THEN i:= i + 1; END_IF</code>	

Siehe auch

- [Kap. 9.3.3 "Strukturierter Text \(ST\), Erweiterter Strukturierter Text \(ExST\)" auf Seite 104](#)

Zuweisungsoperator 'REF'

Der Operator erzeugt eine Referenz (Pointer) auf einen Wert.

Syntax: REF=<Variable>

Beispiel:

```

A : REFERENCE TO DUT;
B : DUT;
C : DUT;

A REF= B; // corresponds to A := ADR(B);
A := C; // corresponds to A^ := C;

```

Siehe auch [Kap. 17.5.13 "Datentyp 'REFERENCE'" auf Seite 384](#)

Anweisungen

ST-Anweisung 'IF'

Die `IF`-Anweisung verwenden Sie, um eine Bedingung zu prüfen und bei erfüllter Bedingung die nachfolgenden Anweisungen auszuführen.

Eine Bedingung wird als Ausdruck codiert, der einen booleschen Wert zurück liefert. Wenn der Ausdruck `TRUE` liefert, ist die Bedingung erfüllt und die zugehörigen Anweisungen nach `THEN` werden ausgeführt. Wenn der Ausdruck `FALSE` liefert, werden die folgenden Bedingungen, die mit `ELSIF` gekennzeichnet sind, ausgewertet. Wenn eine `ELSIF`-Bedingungen `TRUE` liefert, werden die Anweisungen nach dem zugehörigen `THEN` ausgeführt. Wenn al-

le Bedingungen `FALSE` sind, werden die Anweisungen nach `ELSE` ausgeführt.

Es wird also höchstens ein Zweig der `IF`-Anweisung ausgeführt. Die `ELSIF`-Zweige und der `ELSE`-Zweig sind optional.

Syntax:

```
IF <condition> THEN
  <statements>
( ELSIF <condition> THEN
  <statements> )*
( ELSE
  <statements> )?
END IF;
// ( ... ) * None, once or several times
// ( ... ) ? Optional
```

Beispiel

```
PROGRAM PLC_PRG
VAR
  iTemp: INT;
  xHeatingOn: BOOL;
  xOpenWindow: BOOL;
END_VAR

IF iTemp < 17 THEN
  xHeatingOn := TRUE;
ELSIF iTemp > 25 THEN
  xOpenWindow := TRUE;
ELSE xHeatingOn := FALSE;
END_IF;
```

Zur Laufzeit wird das Programm folgendermaßen durchlaufen:

- Wenn die Auswertung des Ausdrucks `iTemp < 17 = TRUE` ergibt, wird die nachfolgende Anweisung ausgeführt und die Heizung eingeschaltet (`xHeatingOn := TRUE`).
- Wenn die Auswertung des Ausdrucks `iTemp < 17 = FALSE` ergibt, wird die nachfolgenden `ELSIF`-Bedingungen `iTemp > 25` ausgewertet.
Wenn die Bedingung `iTemp > 25` wahr ist, wird die Anweisungen unter `ELSIF` ausgeführt und das Fenster geöffnet (`xOpenWindow := TRUE`).
- Wenn alle Bedingungen `FALSE` sind, wird die Anweisung unter `ELSE` ausgeführt und die Heizung wird ausgeschaltet (`xHeatingOn := FALSE`).

Siehe auch

- [Kap. "ST-Anweisung 'CASE'" auf Seite 248](#)

ST-Anweisung 'FOR'

Die `FOR`-Schleife verwenden Sie, um Anweisungen mit einer bestimmten Anzahl von Wiederholungen auszuführen.

Syntax:

```
FOR <counter> := <start value> TO <end value> {BY <increment> } DO
  <instructions>
END_FOR;
```

Der Abschnitt innerhalb der geschweiften Klammern `{ }` ist optional.

IndraLogic führt die `<instructions>` solange aus, wie der `<counter>` nicht größer, oder - bei negativer Schrittgröße `increment` - kleiner als der

<end value> ist. Dies wird vor der Ausführung der <instructions> geprüft.

Immer wenn die Anweisungen <instructions> ausgeführt worden sind, wird der Zähler <counter> automatisch um die Schrittgröße <increment> erhöht. Die Schrittgröße <increment> kann jeden ganzzahligen Wert haben. Wenn Sie keine Schrittgröße angeben, ist die Standard-Schrittgröße 1.

Beispiel

```
FOR iCounter := 1 TO 5 BY 1 DO
    iVar1 := iVar1*2;
END_FOR;
Erg := iVar1;
```

Wenn Sie iVar1 mit 1 vorbelegt haben, hat iVar1 nach der FOR-Schleife den Wert 32.



VORSICHT

Endlosschleife

Der Endwert <end value> darf nicht den gleichen Wert erhalten, wie die Obergrenze des Datentyps des Zählers.

Wenn der Endwert des Zählers gleich der Obergrenze des Datentyps des Zählers ist, entsteht eine Endlosschleife.

Beispielsweise entsteht im obigen Beispiel eine Endlosschleife, wenn iCounter vom Datentyp SINT und der <end value> gleich 127 ist, da der Datentyp SINT die Obergrenze 127 hat.

Die EXIT-Anweisung verwenden Sie, um die Schleife zu beenden, ungeachtet anderer Abbruchbedingungen.

In Erweiterung zum Standard IEC 61131-3 können Sie innerhalb der FOR-Schleife die CONTINUE-Anweisung verwenden.

Siehe auch

- [Kap. 17.5.2 "Ganzzahlige Datentypen" auf Seite 375](#)
- [Kap. "ST-Anweisung 'EXIT'" auf Seite 251](#)
- [Kap. "ST-Anweisung 'IF'" auf Seite 246](#)
- [Kap. "ST-Anweisung 'REPEAT'" auf Seite 250](#)
- [Kap. "ST-Anweisung 'WHILE'" auf Seite 249](#)
- [Kap. "EXST-Anweisung 'CONTINUE'" auf Seite 252](#)

ST-Anweisung 'CASE'

Die CASE-Anweisung verwenden Sie, um mehrere bedingte Anweisungen mit derselben Bedingungsvariablen in einem Konstrukt zusammenzufassen.

Syntax:

```
CASE <Var1> OF
<value1>:<instruction1>
<value2>:<instruction2>
<value3, value4, value5>:<instruction3>
<value6 ... value10>:<instruction4>
...
<value n>:<instruction n>

{ELSE <ELSE-instruction>}

END_CASE;
```

Der Abschnitt innerhalb der geschweiften Klammer { } ist optional.

Abarbeitungsschema einer CASE-Anweisung

- Wenn die Variable <Var1> den Wert <<value i> hat, wird die Anweisung <instruction i> ausgeführt.
- Wenn die Variable <Var1> keinen der angegebenen Werte hat, dann wird die <ELSE-instruction> ausgeführt.
- Wenn für mehrere Werte der Variablen dieselbe Anweisung auszuführen ist, können Sie diese Werte durch Kommata getrennt hintereinander schreiben.

Beispiel

```

CASE iVar OF
1, 5: bVar1 := TRUE;
      bVar3 := FALSE;

2: bVar2 := FALSE;
   bVar3 := TRUE;

10..20: bVar1 := TRUE;
        bVar3 := TRUE;
ELSE
    bVar1 := NOT bVar1;
    bVar2 := bVar1 OR bVar2;
END_CASE;

```

Siehe auch

- [Kap. "ST-Anweisung 'IF'" auf Seite 246](#)

ST-Anweisung 'WHILE'

Die WHILE-Schleife verwenden Sie wie die FOR-Schleife, um Anweisungen mehrfach auszuführen, bis die Abbruchbedingung zutrifft. Die Abbruchbedingung einer WHILE-Schleife ist ein boolescher Ausdruck.

Syntax:

```

WHILE <boolean expression> DO
  <instructions>
END_WHILE;

```

IndraLogic führt die Anweisungen <instructions> solange wiederholt aus, wie der boolesche Ausdruck <boolean expression> TRUE ergibt.

Wenn der boolesche Ausdruck bereits bei der ersten Auswertung FALSE ist, dann führt IndraLogic die Anweisungen niemals aus.

Wenn der boolesche Ausdruck niemals den Wert FALSE annimmt, werden die Anweisungen endlos wiederholt, wodurch ein Laufzeitfehler entsteht.

Beispiel

```

WHILE iCounter <> 0 DO
  Var1 := Var1*2
  iCounter := iCounter-1;
END_WHILE;

```



Sie müssen programmatisch sicherstellen, dass keine Endlosschleife verursacht wird.

Die WHILE- und die REPEAT-Schleifen 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 ist es somit nur möglich, mit diesen beiden Schleifenarten zu arbeiten. Wenn jedoch die Anzahl der Schleifendurchläufe klar ist, dann ist eine FOR-Schleife zu bevorzugen, um Endlosschleifen zu vermeiden.

Die `EXIT`-Anweisung verwenden Sie, um die Schleife zu beenden, ungeachtet anderer Abbruchbedingungen.

In Erweiterung zum Standard IEC 61131-3 können Sie innerhalb der `WHILE`-Schleife die `CONTINUE`-Anweisung verwenden.

Siehe auch

- [Kap. "ST-Anweisung 'EXIT'" auf Seite 251](#)
- [Kap. "ST-Anweisung 'IF'" auf Seite 246](#)
- [Kap. "ST-Anweisung 'FOR'" auf Seite 247](#)
- [Kap. "ST-Anweisung 'REPEAT'" auf Seite 250](#)
- [Kap. "EXST-Anweisung 'CONTINUE'" auf Seite 252](#)

ST-Anweisung 'REPEAT'

Die `REPEAT`-Schleife verwenden Sie wie die `WHILE`-Schleife, jedoch mit dem Unterschied, dass IndraLogic die Abbruchbedingung erst nach dem Ausführen der Schleife überprüft. Dieses Verhalten hat zur Folge, dass die `REPEAT`-Schleife mindestens einmal durchlaufen wird, egal wie die Abbruchbedingung lautet.

Syntax:

```
REPEAT
  <instructions>
UNTIL <boolean expression>
END_REPEAT;
```

IndraLogic führt die Anweisungen `<instructions>` solange aus, bis der boolesche Ausdruck `<boolean expression>` `TRUE` ergibt.

Wenn der boolesche Ausdruck bereits bei der ersten Auswertung `TRUE` ergibt, dann führt IndraLogic die Anweisungen genau einmal aus.

Wenn der boolesche Ausdruck niemals den Wert `TRUE` annimmt, werden die Anweisungen endlos wiederholt, wodurch das Programm einen Laufzeitfehler verursacht.

Beispiel

```
REPEAT
  Var1 := Var1*2;
  iCounter := iCounter-1;
UNTIL
  iCounter = 0
END_REPEAT;
```

Die `WHILE`- und die `REPEAT`-Schleifen sind in gewissem Sinne mächtiger als die `FOR`-Schleife, da die Anzahl der Schleifendurchläufe nicht bereits vor Ausführung der Schleife bekannt sein muss. In manchen Fällen können Sie nur mit diesen beiden Schleifenarten arbeiten. Wenn jedoch die Anzahl der Schleifendurchläufe klar ist, dann ist eine `FOR`-Schleife zu bevorzugen, um Endlosschleifen zu vermeiden.

Die `EXIT`-Anweisung verwenden Sie, um die Schleife zu beenden, ungeachtet anderer Abbruchbedingungen.

In Erweiterung zum Standard IEC 61131-3 können Sie innerhalb der `WHILE`-Schleife die `CONTINUE`-Anweisung verwenden.

Siehe auch

- [Kap. "ST-Anweisung 'EXIT'" auf Seite 251](#)
- [Kap. "ST-Anweisung 'FOR'" auf Seite 247](#)
- [Kap. "ST-Anweisung 'IF'" auf Seite 246](#)

- [Kap. "ST-Anweisung 'WHILE'" auf Seite 249](#)
- [Kap. "EXST-Anweisung 'CONTINUE'" auf Seite 252](#)

ST-Anweisung 'RETURN'

Die `RETURN`-Anweisung verwenden Sie, um einen Funktionsbaustein zu verlassen. Dies können Sie beispielsweise von einer Bedingung abhängig machen.

Beispiel

```
IF xIsDone = TRUE THEN
  RETURN;
END_IF;

iCounter := iCounter + 1;
```

Wenn der Wert von `xIsDone` gleich `TRUE` ist, wird der Funktionsbaustein sofort verlassen und die Anweisung `iCounter := iCounter + 1;` wird nicht ausgeführt.

Siehe [Kap. "ST-Anweisung 'IF'" auf Seite 246](#)

ST-Anweisung 'JMP'

Die `JMP`-Anweisung wird verwendet, um einen unbedingten Sprung zu einer Programmzeile auszuführen, die durch eine Sprungmarke gekennzeichnet ist.

Syntax:

```
<label>: <instructions>

JMP <label>;
```

Die Sprungmarke `<label>` ist ein beliebiger, eindeutiger Bezeichner, den Sie an den Anfang einer Programmzeile stellen.

Bei Erreichen der `JMP`-Anweisung erfolgt ein Rücksprung zu der Programmzeile mit der `<Sprungmarke>`.

Beispiel

```
iVar1 := 0;

_label1: iVar1 := iVar1+1;

(*instructions*)

IF (iVar1 < 10) THEN
  JMP _label1;
END_IF;
```



Sie müssen programmatisch sicherstellen, dass keine Endlosschleife verursacht wird. Beispielsweise können Sie den Sprung an eine Bedingung knüpfen.

ST-Anweisung 'EXIT'

Die `EXIT`-Anweisung verwenden Sie in einer `FOR`-, `WHILE`- oder `REPEAT`-Schleife, um die Schleife zu beenden, ungeachtet anderer Abbruchbedingungen.

Siehe auch

- [Kap. "ST-Anweisung 'FOR'" auf Seite 247](#)
- [Kap. "ST-Anweisung 'WHILE'" auf Seite 249](#)
- [Kap. "ST-Anweisung 'REPEAT'" auf Seite 250](#)

EXST-Anweisung 'CONTINUE'

CONTINUE ist eine Anweisung des Erweiterten Strukturierten Texts (ExST).

Die Anweisung verwenden Sie innerhalb von FOR-, WHILE-, und REPEAT-Schleifen, damit die Ausführung zum Anfang des nächsten Schleifendurchlaufs springt.

Beispiel

```
// SUM= 15 IF FLAG= FALSE
// SUM= 9 IF FLAG= TRUE
SUM:= 0;
FOR I:= 1 TO 3 DO
  FOR J:= 1 TO 2 DO
    SUM:= SUM + 1;
    IF FLAG THEN
      CONTINUE;
    END IF;
    SUM:= SUM + 1;
  END FOR;
  SUM:= SUM + 1;
END FOR;
```

Siehe auch

- [Kap. "ST-Anweisung 'FOR'" auf Seite 247](#)
- [Kap. "ST-Anweisung 'WHILE'" auf Seite 249](#)
- [Kap. "ST-Anweisung 'REPEAT'" auf Seite 250](#)

ST - Aufruf Funktionsbaustein

Syntax

```
<FB-instance>(<FB input variable>:=<value or address> | <other FB input variables>);
```

Beispiel

```
TMR:TON;

TMR (IN:=%OX5, PT:=T#300ms);
varA:=TMR.Q;
```

Der Timer-Funktionsbaustein TON wird in TMR:TON; instanziiert und mit Zuweisungen für die Parameter IN und PT aufgerufen.

Der Ausgang Q wird mit TMR.Q angesprochen und der Variablen varA zugewiesen.

Siehe auch

- [Kap. "Objekt 'Funktionsbaustein' \(FUNCTION_BLOCK\)" auf Seite 556](#)

ST - Kommentare

Kommentar	Beschreibung	Beispiel
Einzeilig	Beginnt mit // und endet am Ende der Zeile.	// Dies ist ein Kommentar
Mehrzeilig	Beginnt mit (* und endet mit *).	(* Dies ist ein mehrzeiliger Kommentar *)
Verschachtelt	Beginnt mit (* und endet mit *). Innerhalb dieses Kommentars können weitere (*. . . . *) Kommentare enthalten sein.	(* a:=inst.out; (* 1.Kommentar *) b:=b+1; (* 2.Kommentar *) *)

17.1.4 Ablaufsprache (AS)

AS-Editor

Der AS-Editor ist ein grafischer Editor. Eine neu hinzugefügte AS-POU enthält einen Init-Schritt und eine nachfolgende Transition.

Sie können im AS-Editor die einzelnen Elemente über Befehle des Menüs "SFC", das Kontextmenü oder über die Werkzeug-Ansicht in das Diagramm einfügen.

Beim Einfügen über einen Menübefehl stehen die an der aktuell selektierten Position einfügbaren Elemente zur Auswahl.

Vor dem Einfügen von Verzweigungen parallel zu mehreren Aktionen und Transitionen, müssen Sie diese Aktionen und Transitionen durch Mehrfachauswahl selektieren.

Sie können AS-Elemente auch per Drag&Drop aus der Ansicht "Werkzeuge" im Diagramm platzieren.

Sobald Sie ein Element in den Editor ziehen, markiert IndraLogic alle möglichen Einfügepositionen mit grauen Rechtecken.

Wenn Sie die Maus über eine mögliche Einfügeposition bewegen, dann ändert sich die Farbe des Rechtecks auf grün. Wenn Sie die Maustaste nun loslassen, wird das Objekt an dieser Stelle eingefügt.

Wenn Sie eine Verzweigung per Drag&Drop einfügen, müssen Sie den Beginn und das Ende der Verzweigung mit der Maus kennzeichnen. Den Beginn der Verzweigung markieren Sie durch Loslassen der Maus auf einer Einfügeposition: Dabei ändert sich die Farbe des Rechtecks auf rot.

Durch Anklicken einer zweiten Einfügeposition legen Sie das Ende der Verzweigung fest. Danach fügt IndraLogic eine Verzweigung um die Objekte zwischen Beginn-Markierung und Ende-Markierung herum ein.

Für das Kopieren von Schritt- und Transitionselementen, die Aktionsobjekte oder Transitionsobjekte aufrufen, können zwei unterschiedliche Duplizierungsmodi eingestellt werden. Entweder werden die Referenzen mit kopiert, oder die referenzierten Objekte werden "eingebettet" und beim Kopieren dupliziert.

Das Verhalten und Aussehen des Editors definieren Sie in den IndraLogic-Optionen in der Kategorie **AS-Editor**.

Siehe auch

- [Kap. 17.1.2 "Generelle Funktionalitäten in allen grafischen Editoren" auf Seite 240](#)
- [Kap. 17.1.4 "Ablaufsprache \(AS\)" auf Seite 253](#)
- [Kap. "Programmieren in Ablaufsprache \(AS / SFC\)" auf Seite 106](#)
- [Kap. "Dialog 'Optionen- 'AS-Editor'" auf Seite 773](#)

AS-Editor im Online-Betrieb

Im AS-Editor können Sie zur Laufzeit die verwendeten Variablen und Ausdrücke auf der Steuerung anzeigen.

Sie können die Variablen und Ausdrücke auch schreiben und forcen.

Eine Debugging-Funktionalität (Haltepunkte, schrittweise Ausführung etc.) ist noch nicht verfügbar.

Einstellungen bezüglich der Online-Darstellung der AS-Elemente und Attribute können Sie in den Optionen des AS-Editors vornehmen.

Falls Sie AS-Flags explizit deklariert haben, werden diese im Online-Betrieb im Deklarationsteil angezeigt - im Offline-Betrieb werden sie nicht angezeigt.



Bitte beachten Sie die Abarbeitungsreihenfolge der Elemente in einem AS-Diagramm.

IndraLogic stellt aktive Schritte im Online-Betrieb blau dar.

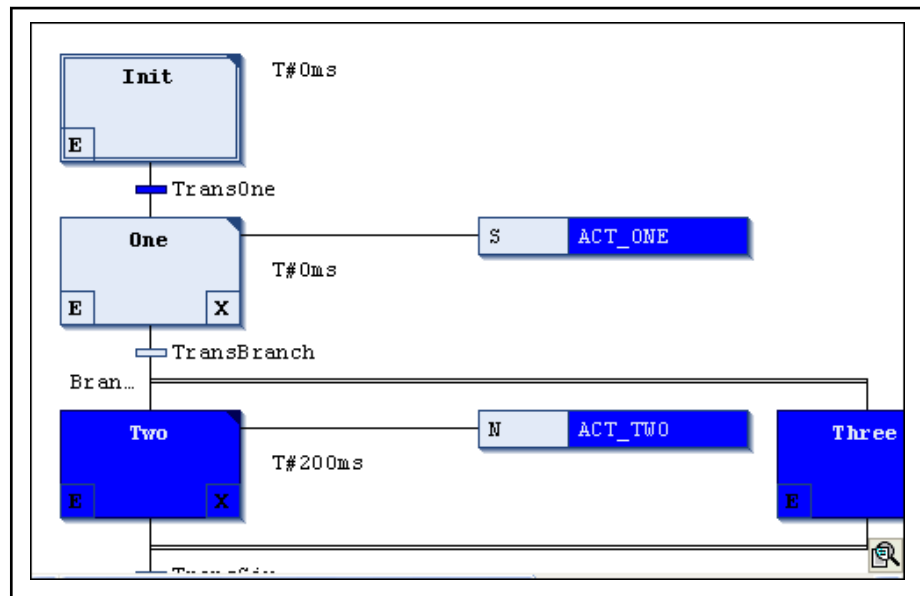


Abb. 17-1:

Siehe auch

- [Kap. "Implizite Variablen" auf Seite 257](#)
- [Kap. 9.2.1 "Deklarationseditor verwenden" auf Seite 87](#)
- [Kap. "Abarbeitungsreihenfolge in AS" auf Seite 254](#)
- [Kap. "Dialog 'Optionen- 'AS-Editor'" auf Seite 773](#)

Abarbeitungsreihenfolge in AS

Grundsätzliches Verhalten der Elemente

- **Aktiver Schritt:** Ein aktiver Schritt ist ein Schritt, dessen Aktionen gerade ausgeführt werden. Im Online-Betrieb stellt IndraLogic aktive Schritte in blau dar.
- **Initialschritt:** Im ersten Zyklus nach Aufruf einer AS-POU wird der Initialschritt automatisch aktiv und die Schrittaktion wird ausgeführt.
- **IEC-Aktionen** führt IndraLogic mindestens zweimal aus. Das erste Mal nach Aktivwerden des Schrittes, das zweite Mal - aber erst im nächsten Zyklus - nach Deaktivieren des Schrittes.
- **Alternative Verzweigungen:** Wenn der Schritt vor der Verzweigung aktiv ist, wertet IndraLogic jeweils die erste Transition jedes alternativen Zweiges von links nach rechts aus.
Im ersten Zweig, in dem IndraLogic eine Transition findet, 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, aktiviert IndraLogic die ersten Schritte in allen Zweigen. 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.

Abarbeitungsreihenfolge:

1. Reset der IEC-Aktionen

IndraLogic setzt die internen Aktionskontroll-Flags der Aktions-Qualifizierer (N, R, S, L, D, P, SD, DS, SL) zurück. Das sind die Flags, die eine IEC-Aktion steuern. Die Flags, die innerhalb von Aktionen aufgerufen werden, werden jedoch nicht zurückgesetzt!

2. Ausgangsaktionen ausführen

IndraLogic überprüft daraufhin alle Schritte, ob sie die Bedingung für die Ausführung der Ausgangsaktionen erfüllen. Die Reihenfolge der Prüfung entspricht der Anordnung im AS-Diagramm: von oben nach unten und von links nach rechts.

IndraLogic führt eine Ausgangsaktion aus, wenn der Schritt deaktiviert wird, d.h. wenn seine Eingangs- und Schritttaktionen (falls vorhanden) im vorangegangenen Zyklus ausgeführt worden sind und die Bedingung für den nachfolgenden Schritt `TRUE` liefert.

3. Eingangsaktionen ausführen

IndraLogic überprüft daraufhin alle Schritte, ob sie die Bedingung für die Ausführung der Eingangsaktionen erfüllen.

Die Reihenfolge der Prüfung entspricht der Anordnung im AS-Diagramm: von oben nach unten und von links nach rechts.

Wenn dies der Fall ist, führt IndraLogic die Eingangsaktionen aus.

IndraLogic führt eine Eingangsaktion aus, sobald die Abarbeitung bei der dem Schritt vorangehenden Transition angelangt ist und diese Transition `TRUE` liefert, der Schritt also aktiviert wird.

4. Zeitenprüfung, Schritttaktionen ausführen

IndraLogic führt für alle Schritte in der Reihenfolge, in der sie im AS-Diagramm angeordnet sind, Folgendes durch:

- IndraLogic kopiert die verstrichene Zeit des aktiven Schritts in die entsprechende implizite Schrittvariable `<Schrittname>.t.` (noch nicht implementiert)
- Bei einer Zeitüberschreitung bedient IndraLogic die entsprechenden Fehler-Flags. (noch nicht implementiert)
- Bei nicht-IEC-Schritten führt IndraLogic die Schritttaktion nun aus.

5. IEC-Aktionen ausführen

IndraLogic führt die IEC-Aktionen in alphabetischer Reihenfolge aus. Dies erfolgt in zwei Durchläufen durch die Liste der Aktionen. Im ersten Lauf führt IndraLogic die IEC-Aktionen aller Schritte aus, 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 ggf. definierte Minimalzeit des Schrittes abgelaufen ist), wird der nachfolgende Schritt aktiviert.



Bei der Implementierung von Aktionen müssen Sie folgendes beachten:

Eine Aktion kann innerhalb desselben Zyklus mehrfach ausgeführt werden, weil Sie sie in mehreren AS-Diagrammen verwenden. 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 Sie die gleiche IEC-Aktion gleichzeitig auf verschiedenen Ebenen eines AS-Diagramms verwenden, kann das zu unvorhersehbaren Effekten während der Abarbeitung führen. Deshalb gibt IndraLogic eine entsprechende Fehlermeldung aus. Diese Fehlermeldung kann zum Beispiel erscheinen, wenn mit Projekten gearbeitet wird, die mit einer älteren Version des Entwicklungssystems erstellt wurden.



Beachten Sie die Möglichkeit, Implizite Variablen zu verwenden, um den Abarbeitungsstatus von Schritten und Aktionen zu überwachen bzw. die Abarbeitung zu kontrollieren.

Siehe auch

- [Kap. "Implizite Variablen" auf Seite 257](#)
- [Kap. "Qualifizierer für Aktionen in AS" auf Seite 256](#)

Qualifizierer für Aktionen in AS

Sie weisen Qualifizierer IEC-Schritten zu. Qualifizierer beschreiben, in welcher Art eine zum Schritt gehörige Aktion ausgeführt wird.

Die Qualifizierer werden vom Funktionsbaustein `SFCActionControl` der Bibliothek `IecSfc.library` verarbeitet. Die Bibliothek wird automatisch im Projekt eingebunden.

N	Non-stored	Die Aktion ist solange aktiv wie der Schritt.
R	overriding Reset	Die Aktion wird deaktiviert.
S	Set (Stored)	IndraLogic führt die Aktion aus, sobald der Schritt aktiv wird. Die Aktion wird weiter ausgeführt, auch wenn der Schritt schon deaktiviert wurde, bis sie einen Reset erhält.
L	time Limited	IndraLogic führt die Aktion aus, sobald der Schritt aktiv wird. Die Aktion wird solange ausgeführt, bis der Schritt inaktiv wird oder die gegebene Zeitspanne abgelaufen ist.
D	time Delayed	IndraLogic startet die Ausführung der Aktion 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	IndraLogic führt die Aktion genau zweimal aus: Einmal, wenn der Schritt aktiv wird und ein weiteres Mal, wenn der Schritt deaktiviert wird.
SD	Stored and time Delayed	IndraLogic startet die Ausführung der Aktion erst, wenn nach Aktivwerden des Schrittes die gegebene Verzögerungszeit abgelaufen ist. Die Aktion wird solange ausgeführt, bis sie einen Reset erhält.

DS	Delayed and Stored	IndraLogic startet die Ausführung der Aktion erst, wenn nach Aktivwerden des Schrittes die gegebene Verzögerungszeit abgelaufen ist und der Schritt immer noch aktiv ist. Die Aktion wird solange ausgeführt, bis sie einen Reset erhält.
SL	Stored and time limited	IndraLogic führt die Aktion aus, sobald der Schritt aktiviert wird. Sie wird solange ausgeführt, bis die gegebene Zeit abgelaufen ist oder sie einen Reset erhält.

Tab. 17-1: Verfügbare Qualifizierer

Sie müssen die Zeitangaben bei den Qualifizierern L, D, SD, DS und SL im Format einer TIME Konstante angeben.



Wenn eine IEC-Aktion deaktiviert wird, wird sie ein weiteres Mal ausgeführt. Dies bedeutet, dass IndraLogic eine solche Aktion mindestens zweimal ausführt. Das betrifft auch Aktionen mit dem Qualifizierer P.

Siehe auch

- [Kap. "Programmieren in Ablaufsprache \(AS / SFC\)" auf Seite 106](#)

Implizite Variablen

Jedes AS-Objekt stellt implizite Variablen bereit, mit denen Sie zur Laufzeit den Status von Schritten und IEC-Aktionen überwachen können.

Diese impliziten Variablen werden von IndraLogic für jeden Schritt und jede IEC-Aktion automatisch angelegt.

Die impliziten Variablen sind Strukturinstanzen vom Typ `SFCStepType` (bei Schritten) bzw. `SFCActionType` (bei Aktionen).

Die Variablen haben den Namen des Elementes, z.B. "step1" für einen Schritt mit Schrittnamen "step1".

Die Strukturkomponenten beschreiben den Status eines Schritts bzw. einer Aktion oder die aktuell bereits abgelaufene Zeit in einem aktiven Schritt.



In den Elementeigenschaften können Sie angeben, ob IndraLogic für dieses Flag eine Symboldefinition in die Symbolkonfiguration exportieren soll.

Siehe auch

- [Kap. "AS-Elementeigenschaften" auf Seite 270](#)

Schritt- und Aktionsstatus

Die Syntax für die implizit durchgeführte Variablendeklaration:

```
<stepname>:SFCStepType;
_  
<actionname>:SFCActionType;
```

Schritt	
<Schrittname>.x	Zeigt den Aktivierungszustand im aktuellen Zyklus. Wenn <Schrittname>.x = TRUE, führt IndraLogic den Schritt im aktuellen Zyklus aus.
<Schrittname>._x	Zeigt den Aktivierungszustand für den nächsten Zyklus. Wenn <Schrittname>._x = TRUE und <Schrittname>.x = FALSE, führt IndraLogic den Schritt im nächsten Zyklus aus, d.h. <Schrittname>._x wird zu Beginn eines Zyklus in <Schrittname>.x kopiert.

<Schrittname>.t	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 AS-Flag SFCError.
<Schrittname>._t	Verwendung nur für interne Zwecke
IEC-Aktion	
_<Aktionsname>.x	TRUE, wenn die Aktion ausgeführt wird.
_<Aktionsname>._x	TRUE, wenn die Aktion aktiv ist.

Tab. 17-2: Folgende implizite Variablen stehen Ihnen für Schritt- oder IEC-Aktionsstatus zur Verfügung:



Sie können die oben beschriebenen Variablen benutzen, um einen bestimmten Status für einen Schritt zu erzwingen (forcen), d.h. um einen Schritt aktiv zu setzen. Beachten Sie jedoch, dass dadurch ein unkontrollierter Status des gesamten SFCs herbeigeführt werden kann!

Siehe auch

- [Kap. "AS-Flags" auf Seite 258](#)

Zugriff auf implizite Variablen

Syntax für den Zugriff:

Innerhalb der POU weisen Sie die implizite Variable direkt zu:

<Variablenname>:=<Schrittname>.<implizite Variable> bzw.

<Variablenname>:=_<Aktionsname>.<implizite Variable>

Beispiel:

```
status:=step1._x;
```

Von einem anderen Baustein aus mit POU-Name:

<Variablenname>:=<POU-Name>.<Schrittname>.<implizite Variable> bzw.

<Variablenname>:=<POU-Name>._<Aktionsname>.<implizite Variable>

Beispiel:

```
status:=SFC_prog.step1._x;
```

Symbolerzeugung

Sie können in den Elementeigenschaften eines Schritts oder einer Aktion definieren, ob IndraLogic für das Schritt- bzw. Aktions-Flag eine Symboldefinition hinzufügen soll.

Dazu müssen Sie in der Ansicht **Eigenschaften** das gewünschte Zugriffsrecht in Spalte **Symbol** aktivieren.

Siehe auch

- [Kap. "AS-Elementeigenschaften" auf Seite 270](#)

AS-Flags

AS-Flags sind implizit erzeugte Variablen mit vordefinierten Namen. Sie können sie verwenden, um die Abarbeitung eines AS-Diagramms zu beeinflussen. Mit diesen Flags können Sie zum Beispiel Zeitüberschreitungen anzeigen oder Schrittketten zurücksetzen. Weiterhin können Sie zum Beispiel den

Implizite Erzeugung der AS-Flags

Tipp-Betrieb anschalten, um Transitionen gezielt zu schalten. Damit Sie auf diese Variablen Zugriff haben, müssen Sie sie deklarieren und aktivieren.

AS-Flags werden von IndraLogic automatisch deklariert, wenn Sie die entsprechende Option aktiviert haben. Diese Option können Sie im Eigenschaftendialog **AS-Einstellungen** für eine einzelne POU oder im Projekteinstellungsdialog **AS** für alle AS-POUs im Projekt setzen.



Die AS-Einstellungen der AS-Flags von einzelnen POU's sind nur wirksam, wenn Sie die Option **Voreinstellungen verwenden** nicht aktiviert haben.

Wenn Sie diese Option aktiviert haben, dann gelten die Einstellungen, die in den Projekteinstellungen definiert sind.



AS-Flags, die Sie im AS-Einstellungen-Dialog deklariert haben, sind nur in der Online-Ansicht des AS-Bausteins sichtbar !

Siehe auch

- ["Variablen \("AS-Flags"\)" auf Seite 756](#)

Explizite Erzeugung der AS-Flags

Händische Deklaration, wie es in IndraLogic 1x nötig war, ist nur noch erforderlich um Schreibzugriff von einem anderen Baustein aus zu ermöglichen.

In diesem Fall müssen Sie Folgendes beachten:

Wenn Sie das Flag in einer globalen Variablenliste deklariert haben, müssen Sie dessen Einstellung **Deklarieren** im AS-Einstellungen-Dialog deaktivieren.

Andernfalls wird implizit ein lokales AS-Flag deklariert, das IndraLogic dann anstelle der globalen Variable verwendet!

Anwendungsbeispiel für SFCError

Beispiel:

Sie haben einen AS-Baustein namens `sfc1` angelegt, der einen Schritt `s1` enthält. In den Eigenschaften des Schritts haben Sie Zeitbegrenzungen definiert.

Die Auswahl der anzuzeigenden Zeiten erfolgt in den Optionen **Extras ▶ Optionen ▶ IndraLogic2G ▶ AS-Editor "Ansicht"**

Die Zeitüberwachung wird durch Forcen der Variablen `SFCEnableLimit` aktiviert.

Wenn der Schritt `s1` länger aktiv bleibt als in seinen Zeiteigenschaften erlaubt ist (Zeitüberschreitung), setzt IndraLogic das AS-Flag `SFCError`, auf das die Applikation zugreifen kann.

Um diesen Zugriff zu erlauben, müssen Sie das AS-Flag in den AS-Einstellungen aktivieren und deklarieren. Wenn Sie es nur deklarieren, wird das AS-Flag zwar in der Online-Ansicht von `sfc1` im Deklarationsteil angezeigt, ist aber ohne Funktion.

☐	SFCReset	☐
☒	SFCError	☒
☒	SFCEnableLimit	☒
☒	SFCErrorStep	☒
☐	SFCErrorPOU	☐

Abb. 17-2:

Nun können Sie das AS-Flag innerhalb der POU, z.B. in einer Aktion (2) oder von außerhalb des Bausteins (1) ansprechen.

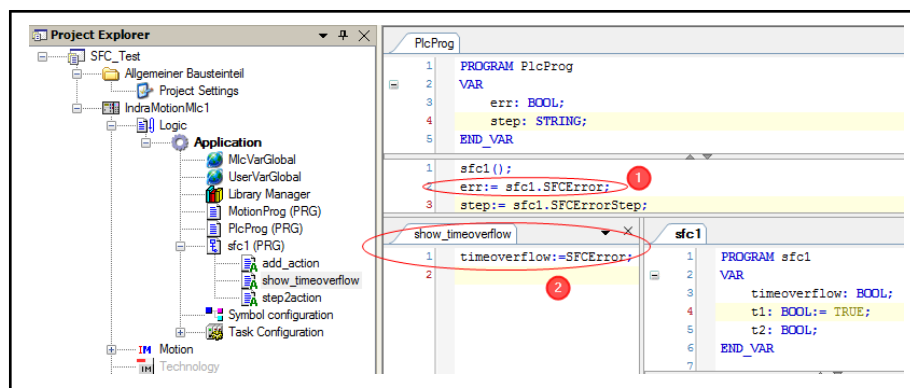


Abb. 17-3:

Online Ansicht von AS-Baustein sfc1.

Minimale Schrittzeit von Schritt s1: T#0s, maximale Schrittzeit von Schritt s1: T#5s.

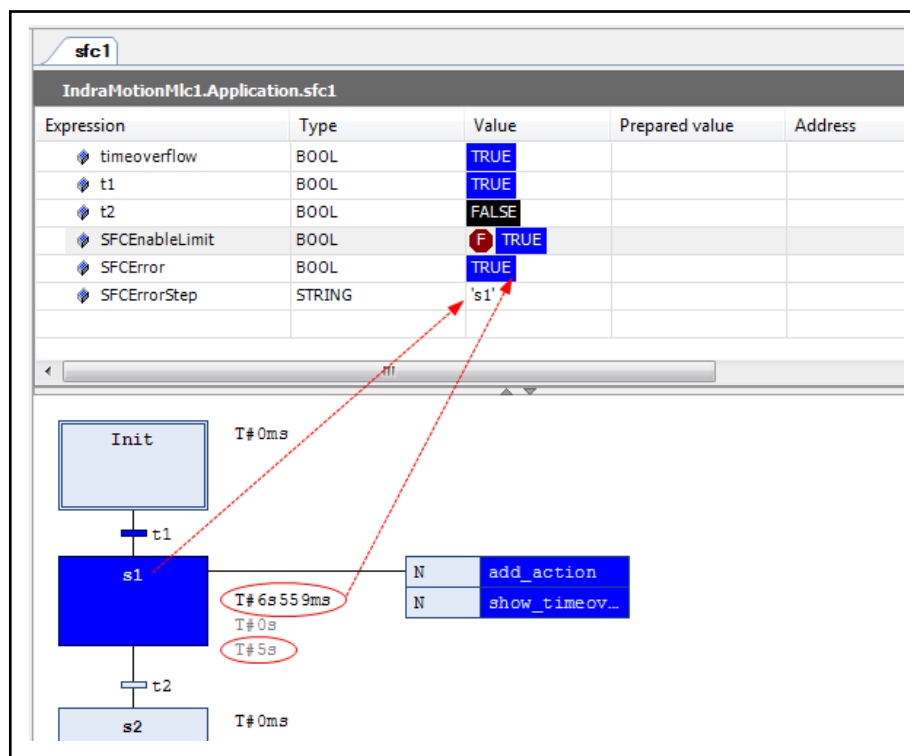


Abb. 17-4:

SFCError wird TRUE, sobald eine Zeitüberschreitung innerhalb von Schritt s1 auftritt.

Zugriff auf die Flags

Syntax für den Zugriff:

Innerhalb der POU weisen Sie das Flag direkt zu: <Variablenna-me>:=<AS-Flag>

Beispiel:

Beispiel

```
checkerror:=SFCError;
```

Von einem anderen Baustein aus mit POU-Name: <Variablenna-me>:=<POU-Name>.<AS-Flag>

*Beispiel:***Beispiel:**

```
checkerror:=SFC_prog.SFCError;
```

Falls Sie einen Schreibzugriff von einem anderen Baustein aus benötigen, müssen Sie das AS-Flag zusätzlich explizit als eine VAR_INPUT Variable im AS-Baustein oder global in einer GVL deklarieren.

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
```

Schreibzugriff auf SFCinit in SFC_prog

```
PROGRAM PlcProg
VAR
    setinit: BOOL;
END_VAR
SFC_prog.SFCinit:=setinit;
```

AS-Flags

Name	Datentyp	Beschreibung
SFCInit	Bool	TRUE: IndraLogic setzt den Ablauf auf den Initialschritt zurück. Die anderen AS-Flags werden ebenfalls zurückgesetzt (Initialisierung). Solange die Variable TRUE ist, bleibt der Initialschritt gesetzt (aktiv), wird aber nicht ausgeführt. Erst wenn Sie SFCInit wieder auf FALSE setzen, wird der Baustein normal weiter bearbeitet.
SFCReset	Bool	Verhält sich ähnlich wie SFCInit. Im Unterschied zu diesem arbeitet IndraLogic allerdings nach der Initialisierung den Initialschritt weiter ab. So könnten Sie beispielsweise im Init-Schritt das SFCReset-Flag gleich wieder auf FALSE setzen.
SFCError	Bool	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 Sie die Variable SFCError vorher nicht wieder zurückgesetzt haben. Die Deklaration von SFCError ist Voraussetzung für das Funktionieren der anderen Flag-Variablen zur Kontrolle des zeitlichen Ablaufs (SFCErrorStep, SFCErrorPOU, SFCQuitError).
SFCEnableLimit	Bool	Dient dem gezielten Einschalten (TRUE) und Ausschalten (FALSE) der Kontrolle für Zeitüberschreitungen in Schritten durch SFCError. Wenn Sie diese Variable deklarieren und aktivieren (AS-Einstellungen), müssen Sie sie auch auf TRUE setzen, damit SFCError arbeitet, ansonsten werden Zeitüberschreitungen ignoriert. Die Verwendung kann beispielsweise bei Inbetriebnahme oder Handbetrieb nützlich sein. Wenn Sie die Variable nicht deklarieren, arbeitet SFCError automatisch. Voraussetzung ist natürlich die Deklaration von SFCError.
SFCErrorStep	String	Speichert den Namen des Schritts, der eine durch SFCError registrierte Zeitüberschreitung verursacht hat. Voraussetzung ist die Deklaration von SFCError.

Name	Datentyp	Beschreibung
SFCErrorPOU	String	Speichert im Falle einer Zeitüberschreitung den Namen des Bausteins, in dem eine durch <code>SFCError</code> registrierte Zeitüberschreitung aufgetreten ist. Voraussetzung ist die Deklaration von <code>SFCError</code> .
SFCQuitError	Bool	Solange diese boolesche Variable <code>TRUE</code> ist, hält IndraLogic die Abarbeitung des AS-Diagramms an, eine eventuelle Zeitüberschreitung in der Variablen <code>SFCError</code> wird dabei zurückgesetzt. Wenn Sie die Variable wieder auf <code>FALSE</code> setzen, werden alle bisherigen Zeiten in den aktiven Schritten zurückgesetzt. Voraussetzung ist die Deklaration von <code>SFCError</code> .
SFCPause	Bool	Solange diese Variable <code>TRUE</code> ist, hält IndraLogic die Abarbeitung des AS-Diagramms an.
SFCTrans	Bool	Wird <code>TRUE</code> , wenn eine Transition schaltet.
SFCCurrent-Step	String	Zeigt den Namen 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	Erlauben den "Tipp-Betrieb" des AS-Bausteins. Wenn Sie den Tipp-Betrieb durch <code>SFCtipMode=TRUE</code> aktivieren, kann nur zum nächsten Schritt weiter geschaltet werden, indem Sie <code>SFCtip</code> auf <code>TRUE</code> setzen. Solange Sie <code>SFCtipMode</code> auf <code>FALSE</code> setzen, kann zusätzlich auch über die Transitionen weiter geschaltet werden.

Elemente

AS-Elemente 'Schritt' und 'Transition'

Symbol Schritt , Symbol Transition 

IndraLogic fügt Schritte und Transitionen grundsätzlich als Kombination ein.

Wenn Sie einen Schritt ohne Transition oder eine Transition ohne Schritt einfügen, führt dies beim Übersetzen zu einem Fehler.

Durch einen Doppelklick auf den Namen können Sie diesen verändern.

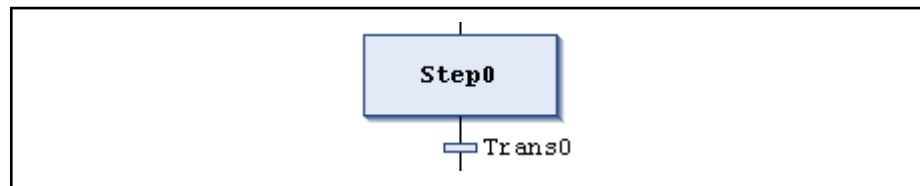


Abb. 17-5:



Schrittnamen müssen innerhalb des Gültigkeitsbereichs des "Vater"-Bausteins eindeutig sein.

Bedenken Sie dies insbesondere, wenn Aktionen verwendet werden, die ebenfalls in AS programmiert sind.

Beachten Sie, dass Sie einen Schritt mit dem Befehl **Initialschritt** oder durch Setzen der entsprechenden Eigenschaft in den AS-Elementeigenschaften zum Initialschritt machen können.

Jeder Schritt wird durch die Schritteigenschaften definiert, die Sie abhängig von den eingestellten Optionen in der Ansicht **Eigenschaften** anzeigen und bearbeiten können.

Aktionen, die ausgeführt werden sollen, wenn der Schritt aktiv ist, müssen Sie dem Schritt hinzufügen. Es gibt "IECAktionen" und "Schrittaktionen". Details hierzu finden Sie im Kapitel zum AS-Element "Aktion".

Eine Transition muss die Bedingung enthalten, unter der der nachfolgende Schritt aktiv werden soll, sobald der Wert der Bedingung TRUE ist. Somit muss eine Transitionsbedingung den Wert TRUE oder FALSE haben.

Sie kann auf folgende zwei Arten definiert werden:

- (1) Direkt ("Inlinebedingung"): Sie ersetzen den Standard-Transitionsnamen durch den Namen einer booleschen Variablen, einer booleschen Adresse, einer booleschen Konstante oder einer Anweisung mit booleschem Ergebnis.

Sie dürfen hier keine Programme, Funktionsbausteine oder Zuweisungen angeben.

- (2) Verwendung eines separaten Transitions- oder Eigenschaftenobjekts ("multi-use condition"):

Sie ersetzen den Standard-Transitionsnamen durch den Namen eines Transitions- oder Eigenschaften-Objekts (📄, 🏠).

Sie legen diese Objekte über den Befehl **Hinzufügen** an. Dies erlaubt eine Mehrfachverwendung ("multiple-use") von Transitionen

Das Objekt kann wie eine "inline-Bedingung" eine boolesche Variable, Adresse, Konstante oder Anweisung enthalten, aber auch Mehrfachanweisungen mit beliebigem Code.

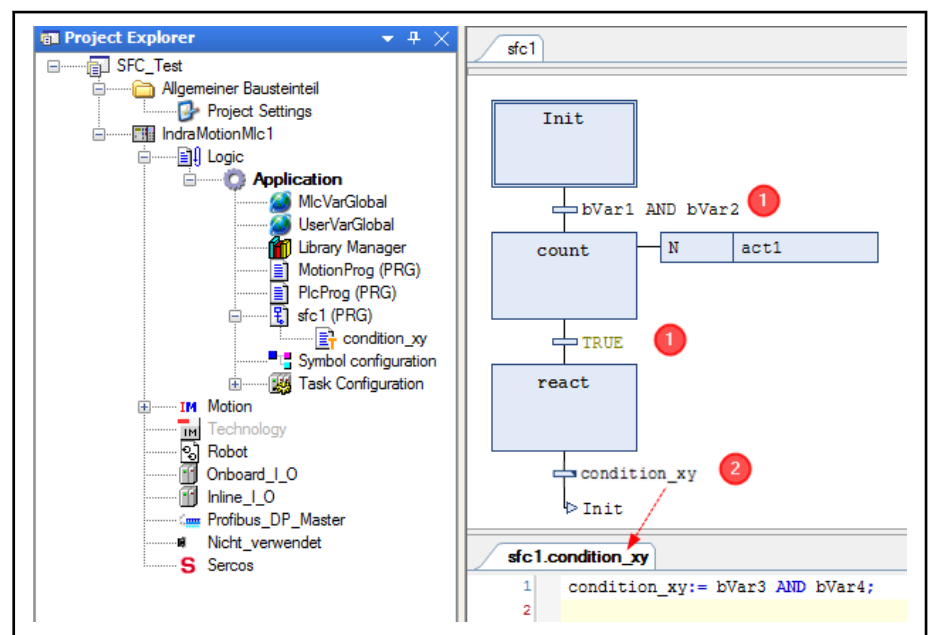


Abb. 17-6:



Es liegt in der Verantwortung des Benutzers, den gewünschten Ausdruck einer Transitionsvariable zuzuweisen, falls die Transition mehrfache Anweisungen enthält!

Transitionen, die ein Transitions- oder Eigenschaftenobjekt referenzieren, werden durch ein kleines Dreieck in der rechten oberen Ecke des Transitionsrechtecks gekennzeichnet.

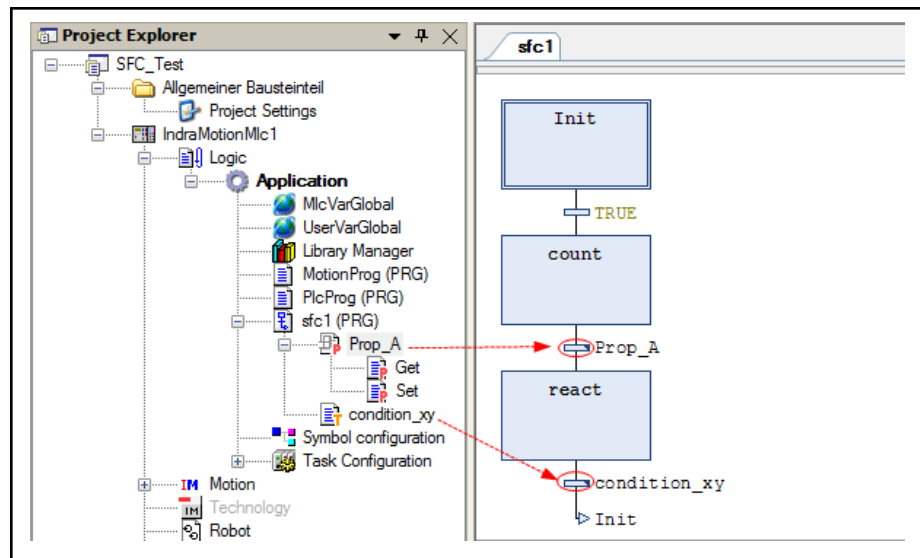


Abb. 17-7:

Im Gegensatz zu IndraLogic 1x behandelt IndraLogic 2G eine Transitionsbedingung wie einen Methodenaufruf.

Die Eingabe erfolgt nach folgender Syntax:

```
<Transitionsname>:=<Transitionsbedingung>
```

(Beispiel `trans1:= a=100`)

oder nur

```
<Transitionsbedingung>
```

(beispielsweise `a=100`)

Sie finden ein Beispiel (`condition_xy`) in der Abbildung oben.

Siehe auch

- [Kap. "Programmieren in Ablaufsprache \(AS / SFC\)" auf Seite 106](#)
- [Kap. "Befehl 'Schritt-Transition einfügen'" auf Seite 683](#)
- [Kap. "AS-Element 'Aktion'" auf Seite 264](#)
- [Kap. "Befehl 'Initialschritt'" auf Seite 682](#)
- [Kap. "AS-Elementeigenschaften" auf Seite 270](#)
- [Kap. 9.18.4 "Methodenaufruf" auf Seite 158](#)

AS-Element 'Aktion'

Symbol



Eine Aktion enthält eine oder mehrere Anweisungen in einer der gültigen Programmiersprachen. Sie können eine Aktion einem Schritt zuweisen.

Aktionen, die Sie in AS-Schritten verwenden, müssen als Bausteine im Projekt angelegt sein.

Die Variablen, die in den Aktionen verwendet werden, sind im Deklarations teil der "Vater"-POU enthalten (im Beispiel `sfc1`).

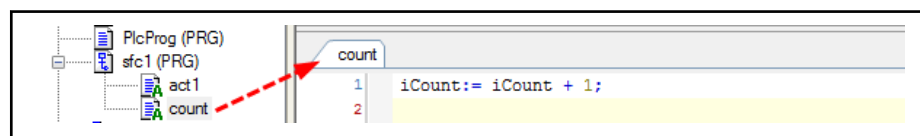


Abb. 17-8:

Ausnahme: Im Fall von IEC-Aktionen, die Sie als Aktionsassoziation einem Schritt hinzufügen, können Sie anstelle eines Aktionsobjekts auch eine boolesche Variable angeben. Der Wert dieser Variablen wird bei jeder Ausführung der "Aktion" zwischen `FALSE` und `TRUE` umgeschaltet.



Sie müssen Schrittnamen innerhalb des Gültigkeitsbereichs des "Vater"-Bausteins eindeutig definieren.

Eine in AS geschriebene Aktion darf keinen Schritt enthalten, der den gleichen Namen hat wie der Schritt, dem die Aktion zugewiesen ist.

Es gibt "IEC-Aktionen" und "Schrittaktionen":

IEC-Aktionen IEC-Aktionen sind Aktionen gemäß der Norm IEC1131-3. Sie werden entsprechend ihres Qualifizierers ausgeführt.

IEC-Aktionen werden zweimal ausgeführt: das erste Mal, wenn der Schritt aktiv wird und ein zweites Mal, wenn er deaktiviert wird. Wenn Sie einem Schritt mehrere Aktionen zuweisen, wird die Aktionenliste von oben nach unten abgearbeitet.

Jede Aktionenbox enthält in der ersten Spalte den Qualifizierer und in der zweiten den Aktionsnamen. Beide können direkt editiert werden.

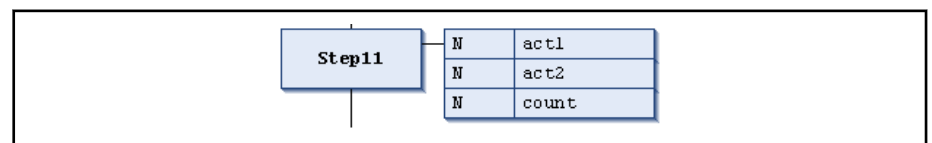


Abb. 17-9:



Wenn in verschiedenen AS-Bausteinen dieselbe globale boolesche Variable als IEC-Aktion assoziiert ist, kann dies zu nicht erwünschten Überschreibungseffekten führen!

Sie können für IEC-Aktionen im Gegensatz Schrittaktionen unterschiedliche Qualifizierer verwenden. Ein weiterer Unterschied zu den Schrittaktionen ist, dass jede IEC-Aktion mit einem Kontroll-Flag versehen ist. Dies bewirkt, dass IndraLogic die Aktion - auch wenn sie gleichzeitig von einem weiteren Schritt aufgerufen wird - nur einmal zur selben Zeit ausführt. Dies kann für Schrittaktionen nicht garantiert werden.

Sie weisen IEC-Aktionen mit dem Befehl **SFC ► Aktionsassoziation einfügen** einem Schritt zu.

Siehe auch

- [Kap. "Befehl 'Aktionsassoziation einfügen'" auf Seite 686](#)
- [Kap. "Qualifizierer für Aktionen in AS" auf Seite 256](#)

Schrittaktionen *Dies sind Aktionen, die Sie in Erweiterung des IEC-Standards verwenden können.*

- **Eingangsaktion:**

IndraLogic führt diese Aktion einmal aus, nachdem der Schritt aktiviert wurde und bevor die Hauptaktion ausgeführt wird.

Sie referenzieren eine neue oder bereits unter dem AS-Objekt angelegte Aktion aus einem Schritt heraus über die Elementeneigenschaft **Eingangsaktion** (`sCount_entry`). Eine neue Aktion können Sie dem Schritt auch mit dem Befehl „Eingangsaktion hinzufügen“ hinzufügen. Die Ein-

gangsaktion wird durch ein "E" in der unteren linken Ecke der Schritt-Box angezeigt.

- **Hauptaktion:**

IndraLogic führt diese Aktion aus, nachdem der Schritt aktiviert wurde und eine eventuelle Eingangsaktion bereits ausgeführt wurde. Allerdings wird sie im Gegensatz zu einer IEC-Aktion (s.o.) kein zweites Mal ausgeführt, wenn der Schritt wieder deaktiviert wird. Sie können hier auch keine Qualifizierer verwenden.

Eine bereits vorliegende Aktion fügen Sie einem Schritt über die Elementeigenschaft **Hauptaktion** (sCount_active) hinzu. Eine neue Aktion können Sie nach einem Doppelklick auf das Schrittelement anlegen und hinzufügen. Eine Hauptaktion wird durch ein gefülltes Dreieck in der rechten oberen Ecke der Schritt-Box angezeigt.

- **Ausgangsaktion:**

IndraLogic führt diese Aktion einmal aus, wenn der Schritt deaktiviert wurde. Beachten Sie allerdings, dass die Ausführung nicht mehr im gleichen Zyklus, sondern zu Beginn des nächsten erfolgt!

Sie referenzieren eine neue oder bereits unter dem AS-Objekt angelegte Aktion aus einem Schritt heraus über die Elementeigenschaft **Ausgangsaktion** (sCount_exit). Eine neue Aktion können Sie dem Schritt auch mit dem Befehl „Ausgangsaktion hinzufügen“ hinzufügen. Die Ausgangsaktion wird durch ein "X" in der unteren rechten Ecke der Schritt-Box angezeigt.

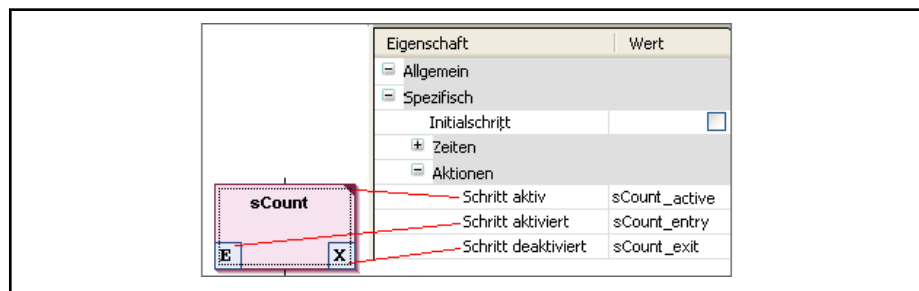


Abb. 17-10:

Siehe auch

- [Kap. "AS-Elementeigenschaften" auf Seite 270](#)

Unterschied IEC-Aktion - Schrittaktion

Der wesentliche Unterschied zwischen Schrittaktionen und IEC-Aktionen mit Qualifizierer "N" besteht darin, dass die IEC-Aktion immer zweimal ausgeführt wird: ein erstes Mal, wenn der Schritt aktiv ist und ein zweites Mal, wenn er deaktiviert wird. Siehe dazu folgendes Beispiel:

Beispiel:

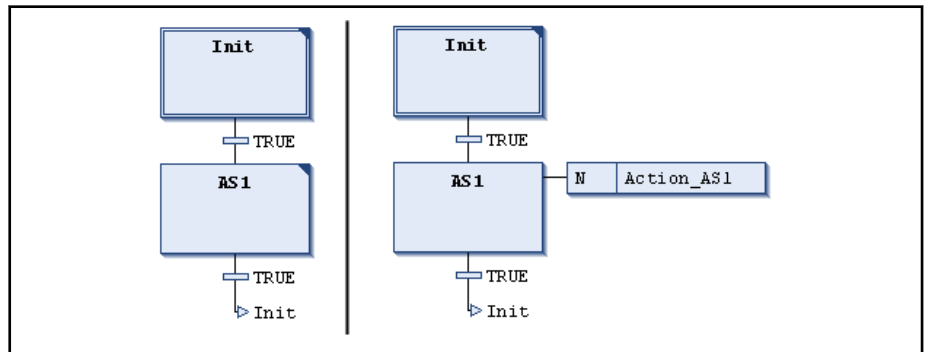


Abb. 17-11:

Sie haben dem Schritt AS1 die Aktion Action_AS1 einmal als Schritttaktion (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.

Angenommen, 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.

Ein weiterer Unterschied ist, dass Schritttaktionen quasi "eingebettet" werden können. In diesem Fall können sie nur noch von dem betreffenden Schritt aufgerufen werden. Wenn Sie diesen Schritt kopieren, erzeugt IndraLogic automatisch neue Aktionsobjekte und kopiert jeweils den Implementierungscode. Ob eine Schritttaktion "eingebettet" wird, definieren Sie entweder beim Einfügen der ersten Aktion im Schritt, oder auch später noch über die Elementeigenschaft "Beim Kopieren duplizieren". Generell kann dieses Verhalten auch in den AS-Optionen voreingestellt werden.

Außerdem kann bei IEC-Aktionen anstelle eines Aktionsobjekts eine boolesche Variable angegeben werden. Dies ist bei Schritttaktionen nicht möglich.

AS-Element 'Verzweigung'

Symbol

Sie verwenden Verzweigungen, wenn Sie in der Ablaufsprache parallele oder alternative Abläufe programmieren.

Bei alternativen Verzweigungen führt IndraLogic immer nur einen der Zweige aus, abhängig von der vorausgehenden Transitionsbedingung. Parallele Verzweigungen werden gleichzeitig ausgeführt.

Siehe auch

- [Kap. "Abarbeitungsreihenfolge in AS" auf Seite 254](#)
- [Kap. "Programmieren in Ablaufsprache \(AS / SFC\)" auf Seite 106](#)
- [Kap. "Befehl 'Verzweigung rechts einfügen'" auf Seite 684](#)

Parallel Verzweigung

In einer parallelen Verzweigung müssen die Zweige mit Schritten beginnen und enden. Parallele Zweige können weitere Verzweigungen beinhalten.

Die horizontale Linie vor und nach der Verzweigung ist eine doppelte Linie.

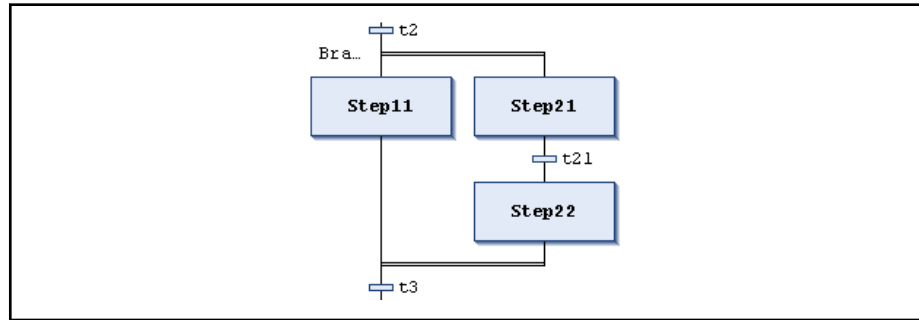


Abb. 17-12:

Abarbeitung im Online-Betrieb:

Wenn die vorausgehende Transition (t2 im dargestellten Beispiel) **TRUE** ist, werden die ersten Schritte in allen parallelen Zweigen aktiv (Step11 und Step21).

IndraLogic arbeitet die einzelnen Zweige parallel ab und wertet erst dann die nachfolgende Transition aus (t3).

Der horizontalen Linie, die den Beginn einer Verzweigung bildet, wird automatisch eine Sprungmarke "Branch<n>" hinzugefügt. Sie können diese Marke als Sprungziel angeben.

Beachten Sie, dass Sie eine parallele Verzweigung mit dem Befehl **Alternativ** in eine alternative Verzweigung umwandeln können.

Siehe auch

- [Kap. "Befehl 'Alternativ'" auf Seite 684](#)

Alternative Verzweigung

Die horizontale Linie vor und nach der Verzweigung ist eine einfache Linie.

In einer alternativen Verzweigung müssen die Zweige mit Transitionen beginnen und enden. Die Zweige können weitere Verzweigungen beinhalten.

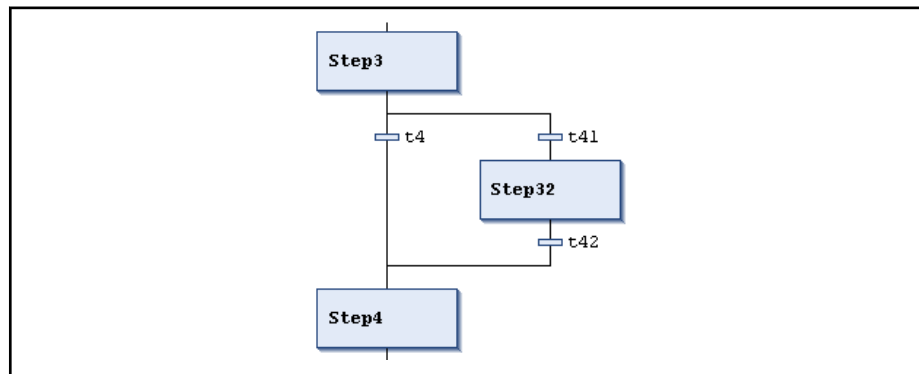


Abb. 17-13:

Wenn der Schritt, der der Verzweigung vorangeht, aktiv ist, wertet IndraLogic jeweils die erste Transition der alternativen Zweige von links nach rechts aus. Bei der ersten Transition, die **TRUE** liefert, wird der zugehörige Zweig 'geöffnet', d.h. der der Transition folgende Schritt aktiviert.

Beachten Sie, dass Sie eine alternative Verzweigungen mit dem Befehl **Parallel** in eine parallele Verzweigung umwandeln können.

Siehe auch

- [Kap. "Befehl 'Parallel'" auf Seite 684](#)

AS-Element 'Sprung'

Symbol 

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 Sie Sprünge nur am Ende einer Verzweigung eingeben.

Das Ziel eines Sprungs wird durch den hinzugefügten Textstring definiert, den Sie direkt editieren können. Das Sprungziel kann ein Schrittname oder die Marke einer parallelen Verzweigung sein.

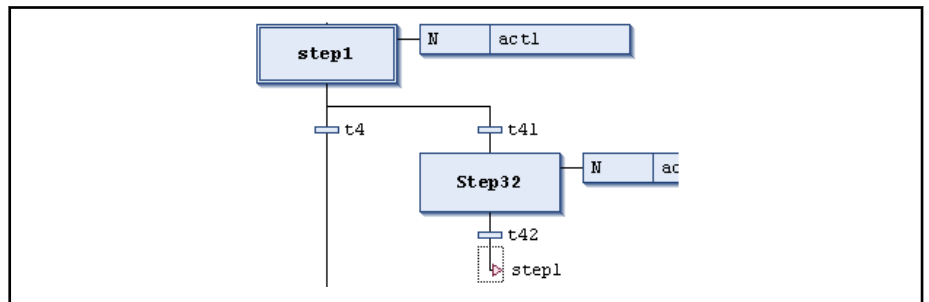


Abb. 17-14:

Siehe auch

- [Kap. "Programmieren in Ablaufsprache \(AS / SFC\)" auf Seite 106](#)
- [Kap. "Befehl 'Sprung einfügen'" auf Seite 687](#)

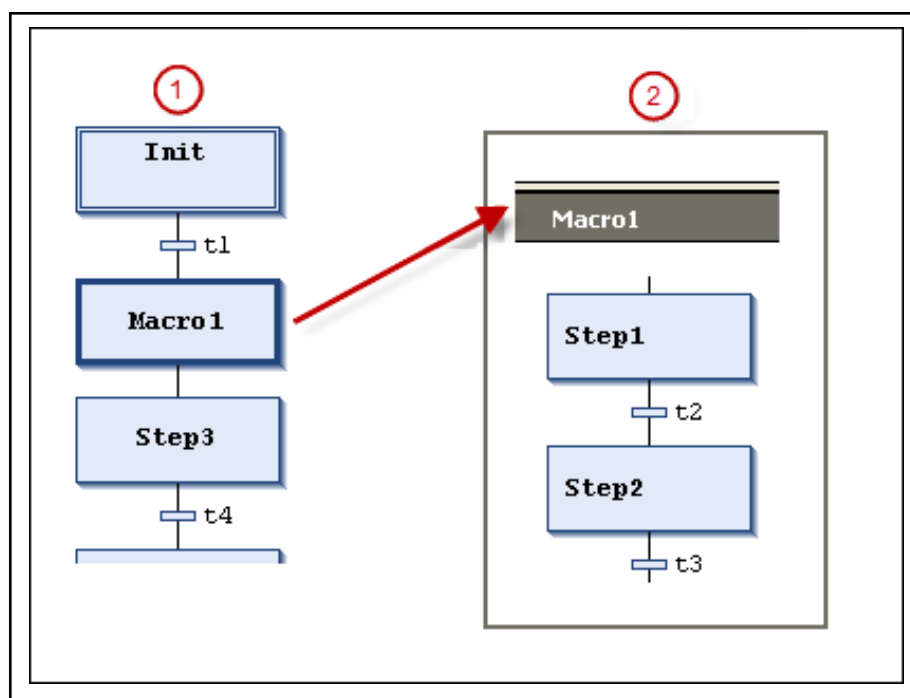
AS-Element 'Makro'

Symbol 

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. Makros dienen nur dazu, bestimmte Teile des Diagramms auszublenden, um zum Beispiel die Übersichtlichkeit zu erhöhen.

Sie öffnen den Makro-Editor mit einem Doppelklick auf die Makro-Box oder mit dem Befehl **SFC ► Makro anzeigen**. Sie können hier genauso programmieren wie in der Hauptansicht des AS-Editors. Um den Makro-Editor wieder zu schließen, verwenden Sie den Befehl **SFC ► Makro verlassen**.



- ① Hauptansicht im SFC-Editor
 ② Ansicht im Makro-Editor für Macro1

Abb. 17-15:

Makros können wiederum Makros enthalten. Die Titelzeile des Makro-Editors zeigt immer, welchen Pfad das gerade geöffnete Makro innerhalb des Diagramms hat.

Beispiel:

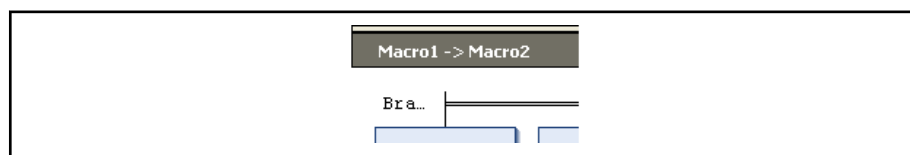


Abb. 17-16:

Siehe auch

- [Kap. "Programmieren in Ablaufsprache \(AS / SFC\)" auf Seite 106](#)
- [Kap. "Befehl 'Makro anzeigen'" auf Seite 688](#)
- [Kap. "Befehl 'Makro verlassen'" auf Seite 688](#)

AS-Elementeigenschaften

Sie bearbeiten die Eigenschaften eines AS-Elements in der Ansicht **Eigenschaften**.

Sie öffnen diese Ansicht mit dem Befehl **Ansicht ► Elementeigenschaften**. Es hängt vom gerade ausgewählten Element ab, welche Eigenschaften dargestellt werden.



Welche Eigenschaften neben dem Element im AS-Diagramm angezeigt werden, hängt von den Einstellungen in den SFC-Editor-Optionen, Registerkarte **Ansicht** ab.

Allgemein

Eigenschaft	Beschreibung Wert
Name	Elementname, standardmäßig "<Element><fortlaufende Zahl>" , z.B. Schrittname "Schritt0", "Schritt1", Verzweigungsname "Verzweigung0" etc.
Kommentar	Elementkommentar (Text), z.B. "Zähler Reset". Sie können Zeilenumbrüche mit <Strg>+<Eingabe> einfügen.
Symbol	Für jedes AS-Element legt IndraLogic eine implizite Variable an, mit dem gleichen Namen wie das Element. Konfiguration ob diese Flag-Variable in die Symbolkonfiguration exportiert werden soll und welches Zugriffsrecht für das Symbol in der Steuerung gelten soll. • 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. • leer: Es wird kein Symbol in die Symbolkonfiguration exportiert.

Spezifisch

Eigenschaft	Beschreibung Wert
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. Hinweis: Wenn Sie diese Eigenschaft für einen anderen Schritt aktiviert haben, muss sie für den Schritt, der die Eigenschaft vorher hatte, deaktiviert werden, um Übersetzungsfehler zu vermeiden.
Beim Kopieren duplizieren	Diese Option ist für Schritte verfügbar, die eine Schrittaktion (Eingangsaktion, Hauptaktion oder Ausgangsaktion) enthalten, und für Transitionen, die mit einem Transitionsojekt verknüpft sind. : Beim Kopieren des Schritts/der Transition wird für die aufgerufenen Aktionen/Transitionen jeweils ein neues Objekt angelegt. Es enthält eine Kopie des Implementierungscode des kopierten Objekts. : Beim Kopieren des Schritts oder der Transition wird für die zugehörigen Aktionen/Transitionen die Verknüpfung zum jeweiligen aufgerufenen Objekt beibehalten. Es entstehen keine neuen Objekte. Quelle und Kopien des Schritts oder der Transition rufen dieselbe Aktion/Transition auf.

Eigenschaft	Beschreibung Wert
Zeiten - Minimal aktiv - Maximal aktiv	Minimale Zeit, die der Schritt aktiv ist, auch wenn die nachfolgende Transition TRUE ist. Maximale Zeit, die der Schritt aktiv sein darf. Wird die Zeit überschritten, setzt IndraLogic die implizite Variable <code>SFCError</code> auf TRUE. Zeitangaben gemäß der IEC-Syntax (z.B. <code>t#8s</code>) oder TIME Variable; Default: <code>t#0s</code>.
Aktionen - Eingangsaktion - Schrittaktion - Ausgangsaktion	Eingangsaktion: IndraLogic führt diese Aktion aus, nachdem der Schritt aktiviert wurde. Schrittaktion: IndraLogic führt diese Aktion aus, wenn der Schritt aktiv ist und eine eventuelle Eingangsaktion bereits abgearbeitet ist. Ausgangsaktion: Wenn der Schritt deaktiviert wird, führt IndraLogic diese Aktion im darauf folgenden Zyklus aus. Hinweis: Beachten Sie die Abarbeitungsreihenfolge.



Mit Hilfe der entsprechenden impliziten AS-Variablen und -Flags können Sie Informationen zum Status eines Schritts oder einer Aktion oder zu Zeitüberschreitungen erhalten.

Siehe auch

- [Kap. "Dialog 'Optionen- 'AS-Editor'" auf Seite 773](#)
- [Kap. "Implizite Variablen" auf Seite 257](#)

17.1.5 Funktionsplan/Kontaktplan/Anweisungsliste (FUP/KOP/AWL)

FUP/KOP/AWL-Editor

Der FUP/KOP/AWL-Editor ist ein kombinierter Editor der Programmiersprachen FUP, KOP und AWL.



AWL kann bei Bedarf aktiviert werden, über **Extras ▶ Optionen ▶ IndraLogic 2G ▶ FUP, KOP und AWL ▶ AWL ▶ Ansicht ▶ AWL aktivieren**

Es gibt einen gemeinsamen Satz an Befehlen und Elementen und IndraLogic konvertiert die 3 Programmiersprachen automatisch intern ineinander.

Der Code im Implementierungsteil ist in allen drei Sprachen mit Hilfe von Netzwerken strukturiert.

Das Menü **FUP/KOP/AWL** enthält Befehle für das Arbeiten im Editor.

Im Offline- und Online-Betrieb können Sie jederzeit über den Befehl zwischen den 3 Editoransichten umschalten.

Das Verhalten des FUP/KOP/AWL-Editors ist durch die Einstellungen in **Extras ▶ Optionen ▶ IndraLogic 2G**, Kategorie **FUP, KOP und AWL** definiert.



Es gibt einige spezielle Elemente, die IndraLogic nicht konvertieren kann und somit nur in der passenden Sprache anzeigt.

Ebenso gibt es Konstrukte, die zwischen AWL und FUP nicht eindeutig konvertierbar sind und deshalb bei der Rückumwandlung in FUP "normalisiert", also wieder aufgehoben werden. Dies betrifft: Negation von Ausdrücken und explizite/implizite Zuweisung von Funktionsbaustein-Eingängen und -Ausgängen.

Eine fehlerfreie Konvertierung zwischen den Sprachen setzt syntaktisch korrekten Code voraus. Anderenfalls können Teile der Implementierung verloren gehen.

Siehe auch

- [Kap. 9.3.1 "FUP/KOP/AWL" auf Seite 92 \(Programmierung\)](#)
- [Kap. 18.2.9 "Menü 'FUP/KOP/AWL'" auf Seite 702 \(Befehle\)](#)
- [Kap. "Dialog 'Optionen- 'FUP, KOP und AWL'" auf Seite 768](#)
- [Kap. 17.1.2 "Generelle Funktionalitäten in allen grafischen Editoren" auf Seite 240](#)

FBD- und KOP-Editor

Einfügen und Anordnen von Elementen

Sie können Elemente durch Ziehen mit der Maus aus der Ansicht **Werkzeuge** (Toolbox) in den Implementierungsteil des Editors ziehen. Alternativ können Sie die Befehle des Kontextmenüs oder des Menüs **FUP/KOP/AWL** verwenden.

Einstellungen zur Anzeige und Oberfläche definieren Sie in den IndraLogic-Optionen, Kategorie **FUP/KOP/AWL**.

Wenn Sie ein Element mit der Maus über ein Netzwerk im Editor ziehen, werden alle möglichen Einfügepositionen mit grauen rauten-, dreieck- oder pfeilförmigen Positionsmarken angezeigt. Sobald der Mauszeiger über einer dieser Marken steht, wird die Marke grün und durch Loslassen der Maustaste fügt IndraLogic das Element an der Position ein.

Beispiel:

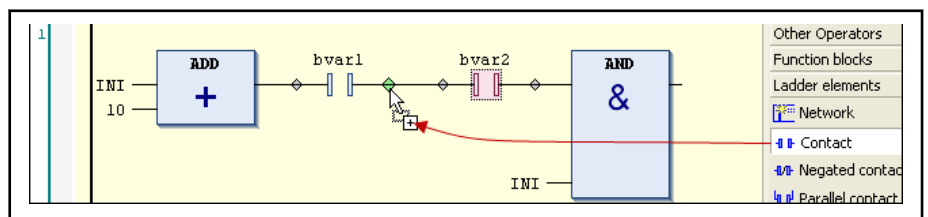


Abb. 17-17:

Wenn Sie einen Funktionsbaustein oder Operator aus der Toolbox oder einem Netzwerk an den linken Rand des Netzwerks auf einen der beiden Pfeile ziehen, geschieht Folgendes: IndraLogic erzeugt automatisch ein neues Netzwerk und fügt dort das Element ein.

Um ein Element zu ersetzen, ziehen Sie mit der Maus ein passendes anderes Element auf seine Position. Elemente, die durch das neue Element ersetzt werden können, kennzeichnet IndraLogic im Editor mit Textfeldern, zum Beispiel **Ersetzen, Eingang anhängen**.

Für das Ausschneiden, Kopieren, Einfügen und Löschen von Elementen können Sie die üblichen Befehle des Menüs **Bearbeiten** verwenden. Kopieren funktioniert auch mit Drag&Drop bei gedrückt gehaltener Taste <Strg>.



Die Operatoren mit EN/ENO-Funktionalität können nur im FUP- und AWL-Editor eingefügt werden.

Selektieren von Elementen

Mit einem Mausklick auf einen Baustein oder eine Verbindungslinie im Editor selektieren Sie diese(n), setzen also den Fokus darauf. Mehrfachselektion ist möglich, während die Taste <Strg> gedrückt gehalten wird. Ein selektiertes Element erhält eine rote Schattierung.

Tooltip

Wenn der Cursor auf bestimmte Elemente zeigt, beispielsweise auf eine Variable oder auf einen Eingang, erscheint ein Tooltip mit Informationen zu diesem Element.

Von rot unterkringelten Elementen zeigt der Tooltip die Precompile-Fehlermeldung des Fehlers, der bei diesem Element auftritt.

Navigieren im Editor

Mit Hilfe der im Folgenden beschriebenen Tasten und Befehle können Sie den Fokus innerhalb des Editors auf eine andere Cursorposition legen. Der Wechsel zwischen den Positionen funktioniert auch netzwerkübergreifend.	
<→> <←>	Wechsel zur benachbarten Cursorposition, entlang des Signalflusses, also von links nach rechts und umgekehrt.
<↑> <↓>	Wechsel zur nächsten Cursorposition oberhalb oder unterhalb der aktuellen Position, wenn diese Nachbarposition zur gleichen logischen Gruppe gehört. Eine logische Gruppe bilden beispielsweise alle Anschlüsse eines Bausteins. Wenn eine solche logische Gruppe nicht existiert: Wechsel zur ersten Cursorposition im nächsten oberen oder unteren Nachbar-element. Im Fall von parallel verbundenen Elemente erfolgt die Navigation entlang des ersten Zweiges.
<Strg> + <Pos 1>	Wechsel ins erste Netzwerk; dieses wird selektiert.
<Strg> + <Ende>	Wechsel ins letzte Netzwerk; dieses wird selektiert.
<Bild ↑>	Blättern um 1 Seite nach oben; das oberste Netzwerk auf dieser Seite wird selektiert.
<Bild ↓>	Blättern um 1 Seite nach unten; das unterste Netzwerk auf dieser Seite wird selektiert.
Befehl Gehe zu...	Wechsel in ein bestimmtes Netzwerk.

Tab. 17-3: Navigieren im Editor

Funktionsbaustein öffnen:

Wenn ein Funktionsbaustein im Editor eingefügt ist, können Sie dessen Implementierung durch einen Doppelklick oder mit dem Befehl **Symbol suchen ► Gehe zur Definition** des Kontextmenüs öffnen.

Siehe auch

- [Kap. 9.3.1 "FUP/KOP/AWL" auf Seite 92](#) (FUP, KOP)
- [Kap. "Programmieren in Funktionsplan \(FUP\)" auf Seite 95](#)

- [Kap. "Programmieren in Kontaktplan \(KOP\)" auf Seite 97](#)
- [Kap. "Dialog 'Optionen- 'FUP, KOP und AWL'" auf Seite 768](#)
- [Kap. "FUP/KOP/AWL-Editor im Online-Betrieb" auf Seite 277](#)
- [Kap. "Befehl 'Gehe zu'" auf Seite 714](#)

AWL-Editor**Einfügen und Anordnen von Elementen:**

Sie können Elemente mit Hilfe der Befehle des Menüs **FUP/KOP/AWL** des Kontextmenüs einfügen. Ein neues Netzwerk können Sie auch zusätzlich per Drag&Drop aus der Werkzeug-Box in den Implementierungsteil des Editors ziehen.

Für das Ausschneiden, Kopieren, Einfügen und Löschen von Elementen können Sie die üblichen Befehle des Menüs **Bearbeiten** verwenden. Kopieren funktioniert auch mit Drag&Drop bei gedrückt gehaltener Taste <Strg>.



Beachten Sie, dass Operatoren mit EN/ENO Funktionalität nur im FUP- und AWL-Editor eingefügt werden können.

Jede Programmzeile wird in einer Tabellenzeile eingetragen.

1. Zeile: Netzwerktitel		
Voraussetzung: Die Option ist in den IndraLogic-Optionen aktiviert		
2. Zeile: Netzwerkkommentar		
Voraussetzung: Die Option ist in den IndraLogic-Optionen aktiviert		
ab 3. Zeile:		
Spalte	Inhalt	Beschreibung
1	Operator	Enthält den AWL-Operator (LD, ST, CAL, AND, OR etc.) oder einen Funktionsnamen. Wenn Sie einen Funktionsbaustein aufrufen, müssen Sie hier zusätzlich die entsprechenden Parameter angeben; im vorhergehenden Feld müssen Sie in diesem Fall := oder => eingeben.
2	Operand	Enthält genau einen Operanden oder den Namen einer Sprungmarke. Bei mehreren Operanden, müssen Sie diese in mehreren Zeilen eingeben und dabei direkt hinter den einzelnen Operanden ein Komma einfügen. (Beispiel siehe unten)
3	Adresse	Enthält die Adresse des Operanden, wie bei dessen Deklaration definiert. nicht editierbar Sie können die Anzeige über die Option Sym-boladresse anzeigen aktivieren/deaktivieren. Wählen Sie dazu den Befehl Extras ► Optionen ► IndraLogic 2G und in der Kategorie FUP, KOP und AWL die Registerkarte Allgemeines .

4	Symbolkommentar	Enthält den Kommentar, der gegebenenfalls für den Operanden in der Deklaration angegeben wurde. nicht editierbar Sie können die Anzeige über die Option Symbolkommentar anzeigen aktivieren/deaktivieren, wenn Sie den Befehl Extras ► Optionen ► IndraLogic 2G auswählen und in der Kategorie FUP, KOP und AWL die Registerkarte Allgemeines auswählen.
5	Operandenkommentar	Kommentar für die aktuelle Programmzeile. Sie können die Anzeige über die Option Operandenkommentar aktivieren/deaktivieren, wenn Sie den Befehl Extras ► Optionen ► IndraLogic 2G auswählen und in der Kategorie FUP, KOP und AWL die Registerkarte Allgemeines auswählen.

Tab. 17-4: Struktur eines Netzwerks im AWL-Editor:

Beispiel:

Beispiel

Spalte:				
①	②	③	④	⑤
CAL	tonInst1{			
	IN:= bVar,			
	PT:= t1,			
	ET=> tOut1}			
LD	tonInst1.Q		is TRUE, PT seconds after IN had a r...	
ST	tonInst2.IN		starts timer with rising edge, reset...	
CAL	tonInst2{			
	PT:= t2,			
	Q=> bReady,	\$QB1		for tonInst2
	ET=> tOut2}			

Abb. 17-18:

Taste(n)/Befehl	Cursorbewegung
<↑>	Springt zum oberhalb/unterhalb liegenden Feld.
<↓>	
<Tab>	Springt nach rechts zum nächsten Feld innerhalb der Zeile.
<Umsch> + <Tab>	Springt nach links zum vorhergehenden Feld innerhalb der Zeile
<Leerzeichen>	Öffnet den Editerrahmen für das selektierte Feld. Alternativ können Sie einen Mausklick auf das Feld ausführen. Gegebenenfalls steht die Schaltfläche für den Eingabehilfe-Dialog zur Verfügung.
<Strg> + <Eing>	Fügt eine neue Zeile unterhalb der aktuellen Zeile ein.
<Entf>	Löscht die aktuelle Zeile.
<Strg> + <Pos1>	Setzt den Fokus auf den Dokumentanfang und markiert das erste Netzwerk.

Taste(n)/Befehl	Cursorbewegung
<Strg><Ende>	Setzt den Fokus auf das Dokumentende und markiert das letzte Netzwerk.
<Bild nach unten>	Blättert eine Seite hoch und markiert das oberste Rechteck.
<Bild nach oben>	Blättert eine Seite nach unten und markiert das oberste Rechteck.

Tab. 17-5: Navigieren im Editor

Siehe auch

- [Kap. 9.3.1 "FUP/KOP/AWL" auf Seite 92 \(AWL\)](#)
- [Kap. "Programmieren in Anweisungsliste \(AWL\)" auf Seite 98](#)
- [Kap. "Modifikatoren und Operatoren in AWL" auf Seite 278](#)
- [Kap. "Dialog 'Optionen- 'FUP, KOP und AWL'" auf Seite 768](#)
- [Kap. "FUP/KOP/AWL-Editor im Online-Betrieb" auf Seite 277](#)

FUP/KOP/AWL-Editor im Online-Betrieb

FUP/KOP/AWL-Editor im Online-Betrieb

Im Online-Betrieb wird im Editor hinter jeder Variablen ihr aktueller Wert dargestellt. Schreiben/Forcen und das Setzen von Haltepunkten ist möglich.

Wenn die Variable gerade geforct ist, wird dies durch **F** direkt vor dem geforcten Wert angezeigt. Wenn ein Wert für das Schreiben oder Forcen vorbereitet ist, wird dieser Wert direkt hinter dem aktuellen Wert in eckigen Klammern <value> angezeigt.

Beispiel:

Geforcte Variable:

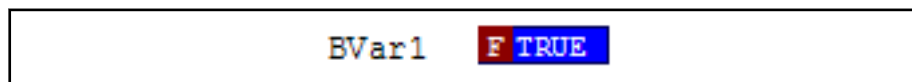


Abb. 17-19:

Vorbereiteter Wert

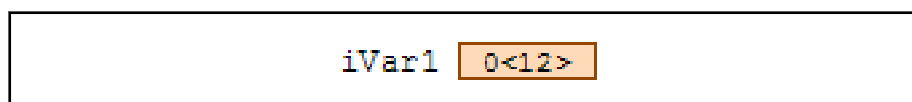


Abb. 17-20:

In der Online-Ansicht eines Kontaktplans (KOP) sind die Anschlussleitungen farblich gekennzeichnet: Verbindungen mit dem Wert **TRUE** werden als fette blaue Linie dargestellt, Verbindungen mit dem Wert **FALSE** als fette schwarze Linie. Dagegen werden Verbindungen von unbekanntem oder analogem Wert normal dargestellt werden (dünne schwarze Linie).



Beachten Sie, dass der Wert der Verbindungen aus den gemonitor-ten Variablen berechnet wird. Es handelt sich dabei nicht um eine echte Ablaufkontrolle.

Haltepunkte

Mögliche Positionen für Haltepunkte sind grundsätzlich die Positionen, an denen sich Variablenwerte ändern können (Anweisungen), an denen sich das Programm verzweigt oder an denen ein anderer Baustein aufgerufen wird.

Mögliche Haltepunktpositionen:

- Auf dem gesamten Netzwerk: Bewirkt, dass der Haltepunkt an der ersten möglichen Position im Netzwerk gesetzt wird.
- Auf einer Bausteinbox, wenn der Baustein eine Zuweisung beinhaltet. Nicht möglich bei Operatoren-Bausteinen, beispielweise ADD, DIV.
- 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.



Gegenwärtig können Sie auf den ersten Baustein im Netzwerk nicht direkt einen Haltepunkt setzen. Wenn Sie jedoch einen Haltepunkt auf das gesamte Netzwerk setzen, wird diese Haltepunktmarkierung im Online-Betrieb automatisch auf den ersten Baustein übertragen.

**Haltepunkte in Methoden:**

IndraLogic setzt automatisch einen Haltepunkt in allen Methoden, die aufgerufen werden können. Wenn also eine durch eine Schnittstelle verwaltete Methode aufgerufen wird, werden Haltepunkte in allen Methoden gesetzt, die in Funktionsbausteinen vorkommen, die diese Schnittstelle implementieren; ebenso in allen abgeleiteten Funktionsbausteinen, die die Methode verwenden. Wenn eine Methode durch einen Zeiger auf einen Funktionsbaustein aufgerufen wird, setzt IndraLogic die Haltepunkte in der Methode des Funktionsbausteins und in allen abgeleiteten Funktionsbausteinen, die die Methode verwenden.

Siehe auch

- [Kap. 12.4 "Forcen und Schreiben von Variablen" auf Seite 198](#)
- [Kap. 12.2 "Verwenden von Haltepunkten" auf Seite 193](#)

Modifikatoren und Operatoren in AWL

Modifikator	Kombiniert mit Operator	Beschreibung
C	JMP, CAL, RET	Die Anweisung wird nur ausgeführt, wenn das Ergebnis des vorhergehenden Ausdrucks TRUE ist.
N	JMPC, CALC, RETC	Die Anweisung wird nur ausgeführt, wenn das Ergebnis des vorhergehenden Ausdrucks FALSE ist.
N	ansonsten	Negation des Operanden (nicht des Akkus).

Tab. 17-6: *Modifikatoren*

Operator	N	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

Operator	N	Bedeutung	Beispiel
S		Setzt den Operanden (Typ <code>BOOL</code>) auf <code>TRUE</code> , wenn der Inhalt des Akkumulators <code>TRUE</code> ist.	S bVar1
R		Setzt den Operanden (Typ <code>BOOL</code>) auf <code>FALSE</code> , wenn der Inhalt des Akkumulators <code>TRUE</code> ist.	R bVar1
AND	N, (	Bitweises <code>AND</code> des Akkumulatorwerts und des (negierten) Operanden	AND bVar2
OR	N, (	Bitweises <code>OR</code> des Akkumulatorwerts und des (negierten) Operanden	OR xVar
XOR	N, (	Bitweises, exklusives <code>OR</code> des Akkumulatorwerts und des (negierten) Operanden	XOR N, (bVar1,bVar2)
NOT		Bitweise Negation des Akkumulatorwerts	
ADD	(	Addition des Akkumulatorwerts und des Operanden; Ergebnis wird in den Akkumulator geschrieben.	ADD iVar1
SUB	(	Subtraktion des Operanden vom Akkumulator-Wert; Ergebnis wird in den Akkumulator geschrieben.	SUB iVar2
MUL	(	Multiplikation von Akkumulatorwert und Operand; Ergebnis wird in den Akkumulator geschrieben.	MUL iVar2
DIV	(	Division des Akkumulatorwerts durch den Operanden; Ergebnis wird in den Akkumulator geschrieben.	DIV 44
GT	(	Prüft, ob der Akkumulatorwert größer als der Operandenwert ist; Ergebnis (<code>BOOL</code>) wird in den Akkumulator geschrieben; >	GT 23
GE	(	Prüft, ob der Akkumulatorwert größer oder gleich dem Operandenwert ist; Ergebnis (<code>BOOL</code>) wird in den Akkumulator geschrieben.	GE iVar2
EQ	(	Prüft, ob der Akkumulatorwert gleich dem Operandenwert ist; Ergebnis (<code>BOOL</code>) wird in den Akkumulator geschrieben.	EQ iVar2
NE	(	Prüft, ob der Akkumulatorwert ungleich dem Operandenwert ist; Ergebnis (<code>BOOL</code>) wird in den Akkumulator geschrieben.	NE iVar1
LE	(	Prüft, ob der Akkumulatorwert kleiner oder gleich dem Operandenwert ist; Ergebnis (<code>BOOL</code>) wird in den Akkumulator geschrieben.	LE 5
LT	(	Prüfen, ob der Akkumulator-Wert kleiner als der Operandenwert ist; Ergebnis (<code>BOOL</code>) wird in den Akkumulator geschrieben.	LT cVar1
JMP	CN	Unbedingter (bedingter) Sprung zur angegebenen Sprungmarke	JMPN next
CAL	CN	(Bedingter) Aufruf eines Programms oder Funktionsbausteins (wenn der Akkumulator-Wert <code>TRUE</code> ist)	CAL prog1
RET		Verlassen des Bausteins und Rückkehr zum aufrufenden Baustein	RET
RET	C	Wenn der Akkumulatorwert <code>TRUE</code> ist: Verlassen des Bausteins und Rückkehr zum aufrufenden Baustein.	RETC

Referenz Programmierung

Operator	N	Bedeutung	Beispiel
RET	CN	Wenn der Akkumulatorwert <code>FALSE</code> ist: Verlassen des Bausteins und Rückkehr zum aufrufenden Baustein.	RETCN
)		Auswerten der zurückgestellten Operation	

Tab. 17-7: Operatoren mit den möglichen Modifikatoren

Beispiel:

1	<code>AND</code>	<code>TRUE</code>	load <code>TRUE</code> to accumulator
	<code>ANDN</code>	<code>bVar1</code>	execute <code>AND</code> with negated value of <code>bVar1</code>
	<code>JMPC</code>	<code>m1</code>	if <code>accum. is TRUE</code> , jump to label " <code>m1</code> "
	<code>LDN</code>	<code>bVar2</code>	store negated value of <code>bVar2</code> ...
	<code>ST</code>	<code>bRes</code>	... in <code>bRes</code>
2	<code>m1:</code>		
	<code>LD</code>	<code>bVar2</code>	store value of <code>bVar2</code> ...
	<code>ST</code>	<code>bRes</code>	... in <code>bRes</code>

Abb. 17-21:

Anwendung	Beschreibung	Beispiele																				
Mehrere Operanden für 1 Operator	<i>Möglichkeiten</i> - Sie geben die Operanden in aufeinander folgenden Zeilen, getrennt durch Komma in der 2. Spalte ein. - Sie wiederholen den Operator in aufeinander folgenden Zeilen.	Variante 1: <table><tr><td>LD</td><td>2</td></tr><tr><td>ADD</td><td>3,</td></tr><tr><td></td><td>4,</td></tr><tr><td></td><td>6</td></tr><tr><td>ST</td><td>iVAR</td></tr></table> Variante 2: <table><tr><td>LD</td><td>2</td></tr><tr><td>ADD</td><td>3</td></tr><tr><td>ADD</td><td>4</td></tr><tr><td>ADD</td><td>6</td></tr><tr><td>ST</td><td>iVAR</td></tr></table>	LD	2	ADD	3,		4,		6	ST	iVAR	LD	2	ADD	3	ADD	4	ADD	6	ST	iVAR
LD	2																					
ADD	3,																					
	4,																					
	6																					
ST	iVAR																					
LD	2																					
ADD	3																					
ADD	4																					
ADD	6																					
ST	iVAR																					
Komplexe Operanden	Für einen komplexen Operanden geben Sie in der ersten Spalte die öffnende Klammer (an. Die schließende Klammer geben Sie in der ersten Spalte in einer separaten Zeile im Anschluss an die Operandeneinträge der folgenden Zeilen ein.	Ein String wird jeden Zyklus um ein Zeichen rotiert: <table><tr><td>LD</td><td>stRotate</td></tr><tr><td>RIGHT(</td><td>stRotate</td></tr><tr><td>LEN</td><td></td></tr><tr><td>SUB</td><td>1</td></tr><tr><td>)</td><td></td></tr><tr><td>CONCAT(</td><td>stRotate</td></tr><tr><td>LEFT</td><td>1</td></tr><tr><td>)</td><td></td></tr><tr><td>ST</td><td>stRotate</td></tr></table>	LD	stRotate	RIGHT(	stRotate	LEN		SUB	1	)		CONCAT(	stRotate	LEFT	1	)		ST	stRotate		
LD	stRotate																					
RIGHT(	stRotate																					
LEN																						
SUB	1																					
)																						
CONCAT(	stRotate																					
LEFT	1																					
)																						
ST	stRotate																					

Anwendung	Beschreibung	Beispiele												
Funktionsbaustein-Aufruf, Programmaufruf	Spalte 1: Operator <code>CAL</code> oder <code>CALC</code> Spalte 2: Name der Funktionsbausteins-Instanz oder des Programms und öffnende Klammer <code>(</code>. Wenn keine Parameter folgen, wird hier die schließende Klammer <code>)</code> eingegeben. darauf folgende Zeilen: Spalte 1: Parametername gefolgt von <code>:=</code> für Eingabeparameter oder <code>=></code> für Ausgabeparameter Spalte 2: Parameterwert gefolgt von Komma <code>,</code> wenn weitere Parameter folgen. Nach dem letzten Parameter, wird die schließende Klammer <code>)</code> eingegeben. Als Einschränkung bezüglich des IEC-Standards können hier komplexe Ausdrücke nicht verwendet werden. Solche Konstrukte müssen Sie dem Funktionsbaustein oder dem Programm noch vor dem Aufruf zuweisen.	<table><tr><td>CAL</td><td>POUToCall (</td></tr><tr><td></td><td> iCounter:= 1,</td></tr><tr><td></td><td> iDecrement:= 1000,</td></tr><tr><td></td><td> wError=> wResult)</td></tr><tr><td>LD</td><td>POUToCall.bError</td></tr><tr><td>ST</td><td>bErr</td></tr></table>	CAL	POUToCall (		iCounter:= 1,		iDecrement:= 1000,		wError=> wResult)	LD	POUToCall.bError	ST	bErr
CAL	POUToCall (													
	iCounter:= 1,													
	iDecrement:= 1000,													
	wError=> wResult)													
LD	POUToCall.bError													
ST	bErr													
Funktionsaufruf	Zeile 1: Spalte 1: <code>LD</code> Spalte 2: Eingabevariable Zeile 2: Spalte 1: Funktionsname Spalte2: weitere Eingabeparameter durch Komma getrennt. IndraLogic schreibt den Rückgabewert in den Akkumulator. Zeile 3: Spalte 1: <code>ST</code> Spalte 2: Variable, in die der Rückgabewert geschrieben wird	<table><tr><td>LD</td><td>X7</td></tr><tr><td>GeomAverage</td><td>25</td></tr><tr><td>ST</td><td>Ave</td></tr></table>	LD	X7	GeomAverage	25	ST	Ave						
LD	X7													
GeomAverage	25													
ST	Ave													
Aktionsaufruf	Wie Funktionsbaustein- oder Programmaufruf. Der Aktionsname wird an den FB-Instanz- oder Programmnamen angehängt.	<table><tr><td>CAL</td><td>POU1.ResetAction</td></tr></table>	CAL	POU1.ResetAction										
CAL	POU1.ResetAction													
Sprung	Spalte 1: Operator <code>JMP</code> oder <code>JMPC</code>. Spalte 2: Name der Sprungmarke des Ziel-Netzwerks. Bei einem unbedingten Sprung muss die voraus gehenden Anweisungsfolge mit einem der folgenden Kommandos enden: <code>ST,STN, S, R, CAL, RET, JMP</code> Bei einem bedingten Sprung hängt die Ausführung des Sprungs vom geladenen Wert ab.	<table><tr><td>LD</td><td>BVar1</td></tr><tr><td>JMPC</td><td>Label1</td></tr></table>	LD	BVar1	JMPC	Label1								
LD	BVar1													
JMPC	Label1													

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

- [Kap. 9.3.1 "FUP/KOP/AWL" auf Seite 92 \(AWL\)](#)
- [Kap. "Programmieren in Anweisungsliste \(AWL\)" auf Seite 98](#)

Elemente

FUP/KOP/AWL-Element 'Netzwerk'

Symbol: 

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 kann Folgendes enthalten: logische und arithmetische Ausdrücke, Programm-, Funktions- und Funktionsbausteinaufrufe, eine Sprung- oder Return-Anweisung.

Ein AWL-Programm besteht aus mindestens einem Netzwerk. Dieses Netzwerk kann alle AWL-Anweisungen des Programms enthalten.

Sie können jedes Netzwerk mit einem Titel, einem Kommentar oder einer Marke versehen. In den IndraLogic-Optionen, Kategorie **FUP, KOP und AWL** können Sie definieren, ob Netzwerktitel, Netzwerkkommentar und Trennlinien zwischen den einzelnen Netzwerken im Editor angezeigt werden.

Um einen Netzwerktitel einzugeben, klicken Sie auf die 1. Zeile des Netzwerks. Zur Eingabe eines Kommentars klicken Sie auf die 2. Zeile des Netzwerks.

Siehe auch

- [Kap. "Dialog 'Optionen- 'FUP, KOP und AWL'" auf Seite 768](#)
- [Kap. "Befehl 'Netzwerk einfügen'" auf Seite 702](#)

FUP/KOP/AWL-Element 'Baustein'

Symbol: 

Ein Baustein und dessen Aufruf kann zusätzliche Funktionen repräsentieren, beispielsweise IEC-Funktionsbausteine, IEC-Funktionen, Bibliotheksbausteine, Operatoren.

Ein Baustein kann beliebige Ein- und Ausgänge haben.

Wenn der Baustein eine Bilddatei mitliefert, wird das Bausteinsymbol innerhalb des Bausteins angezeigt. Voraussetzung ist, dass in den IndraLogic-Optionen, Kategorie **FUP, KOP und AWL** die Option **Bausteinsymbol anzeigen** aktiviert ist.

Wenn Sie die Bausteinschnittstellen geändert haben, können Sie die Bausteinparameter mit dem Befehl **FUP/KOP/AWL ► Parameter aktualisieren** aktualisieren, ohne dass Sie den Baustein neu einfügen müssen.

Siehe auch

- [Kap. "Befehl 'Bausteinaufruf einfügen'" auf Seite 703](#)
- [Kap. "Dialog 'Optionen- 'FUP, KOP und AWL'" auf Seite 768](#)
- [Kap. "Befehl 'Parameter aktualisieren'" auf Seite 712](#)

FUP/KOP/AWL-Element 'Zuweisung'

Symbol: 

Der FUP-Editor zeigt eine neu eingefügte Zuweisung als Linie mit nachfolgenden 3 Fragezeichen an. Der KOP-Editor zeigt eine neu eingefügte Zuweisung als Spule mit darüber liegenden 3 Fragezeichen an.

Nach den Einfügen können Sie den Platzhalter ??? durch den Namen der Variablen ersetzen, der das von links kommende Signal zugewiesen werden soll. Dafür steht Ihnen die Eingabehilfe zur Verfügung.



In AWL wird eine Zuweisung über die Operatoren LD und ST programmiert.

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "Befehl 'Zuweisung einfügen'" auf Seite 703](#)

FUP/KOP/AWL-Element 'Baustein mit EN/ENO'

Symbol:

Das Element steht nur im FUP- und im KOP-Editor zur Verfügung.

Der Baustein entspricht generell dem FUP/KOP/AWL-Element **Baustein**; zusätzlich enthält dieser Baustein jedoch einen EN-Eingang und einen ENO-Ausgang. EN und ENO haben den Datentyp `BOOL`.

Funktion des EN-Eingangs und ENO-Ausgangs: Wenn der Eingang EN zum Zeitpunkt des Bausteinaufrufs den Wert `FALSE` hat, werden die im Baustein definierten Operationen nicht ausgeführt. Andernfalls, also wenn EN den Wert `TRUE` hat, werden diese Operationen ausgeführt. Der ENO-Ausgang hat den gleichen Wert wie der EN-Eingang.

Siehe auch

- [Kap. "Befehl 'Baustein mit EN/ENO einfügen'" auf Seite 704](#)
- [Kap. "FUP/KOP/AWL-Element 'Baustein'" auf Seite 282](#)

FUP/KOP/AWL-Element 'Eingang'

Symbol:

Die maximale Anzahl von Eingängen hängt vom Bausteintyp ab.

Ein neu hinzugefügter Eingang ist zunächst mit ??? belegt. Sie können die Zeichenfolge ??? mit einer Variablen oder einer Konstanten ersetzen.

Siehe auch

- [Kap. "Befehl 'Bausteineingang einfügen'" auf Seite 705](#)
- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

FUP/KOP/AWL-Element 'Sprungmarke'

Die Marke ist ein optionaler Kennzeichner für ein Netzwerk in FUP und KOP, den Sie als Ziel für einen Sprung angeben können.

Wenn Sie in einem Netzwerk eine Sprungmarke einfügen, wird diese im Netzwerk als editierbares Feld **Label** hinzugefügt.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "Befehl 'Sprungmarke einfügen'" auf Seite 705](#)

FUP/KOP/AWL-Element 'Sprung'

Symbol:

In FUP oder KOP wird ein Sprung abhängig von der aktuellen Cursorposition entweder direkt vor einem Eingang, direkt nach einem Ausgang oder am Ende des Netzwerks eingefügt.

Direkt hinter dem Sprungelement geben Sie eine Sprungmarke als Sprungziel ein.

In AWL programmieren Sie einen Sprung mit der Anweisung `JMP`.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "Befehl 'Sprung einfügen'" auf Seite 705](#)
- [Kap. "FUP/KOP/AWL-Element 'Sprungmarke'" auf Seite 283](#)

FUP/KOP/AWL-Element 'Return'

Das Element dient dazu, die Abarbeitung des Baustein sofort abubrechen, wenn der Eingang des `RETURN`-Elements `TRUE` wird.

In einem FUP- oder KOP-Netzwerk können Sie die Return-Anweisung parallel zu oder anschließend an die vorausgehenden Elemente platzieren.

In AWL steht Ihnen für diesen Zweck die Anweisung `RET` zur Verfügung.

Siehe auch

- [Kap. "Befehl 'Return einfügen'" auf Seite 705](#)
- [Kap. "Modifikatoren und Operatoren in AWL" auf Seite 278](#)

FUP/KOP/AWL-Element 'Leitungsverzweigung'

Symbol: 

Das Element steht im KOP- und FUP-Editor zur Verfügung und stellt eine offene Leitungsverzweigung dar. Eine Leitungsverzweigung spaltet die Abarbeitungslinie ab der aktuellen Cursor-Position in 2 Subnetzwerke, die von oben nach unten nacheinander ausgeführt werden. Jedes Subnetzwerk können Sie weiter verzweigen, wodurch innerhalb eines Netzwerks Mehrfachverzweigungen entstehen.

Jedes Subnetzwerk erhält am Verzweigungspunkt ein Marker-Symbol (Rechteck), das Sie auswählen können, um weitere Befehle auszuführen.



Die Befehle **Kopieren**, **Ausschneiden** und **Einfügen** stehen für Subnetzwerke nicht zur Verfügung.

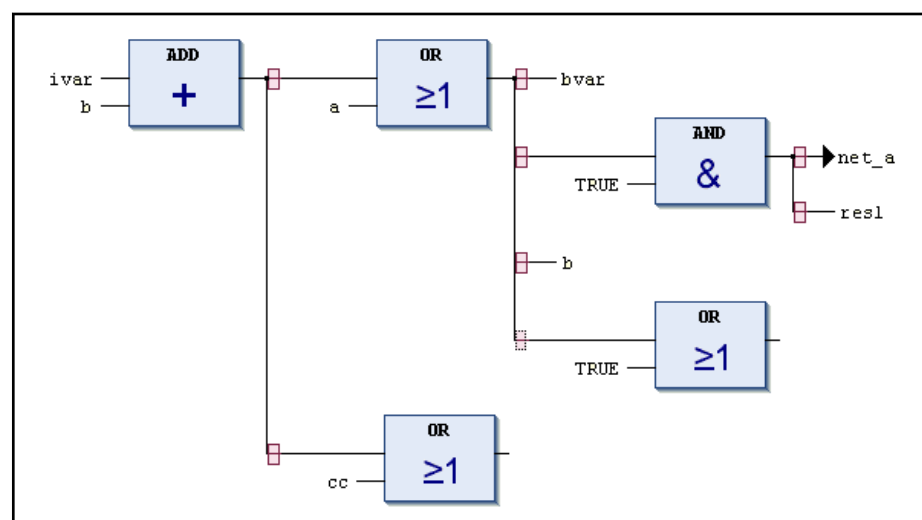


Abb. 17-22:

Um ein Subnetzwerk zu löschen, müssen Sie zuerst alle Elemente des Netzwerks und anschließend das Markersymbol des Subnetzwerks löschen.

Siehe auch

- [Kap. "Befehl 'Leitungsverzweigung einfügen'" auf Seite 710](#)
- [Kap. "Befehl 'Leitungsverzweigung oberhalb einfügen'" auf Seite 711](#)
- [Kap. "Befehl 'Leitungsverzweigung unterhalb einfügen'" auf Seite 711](#)

FUP/KOP/AWL-Element 'Execute'

Symbol: 

Das Element ist ein Baustein, der dazu dient, dass Sie im FUP- und im KOP-Editor direkt ST-Code eingeben können.

Sie können das Element **Execute** aus der Ansicht **Werkzeuge** mit der Maus in den Implementierungsteil Ihrer POU ziehen.

Wenn Sie im Baustein auf **Geben Sie hier ST-Code ein...** klicken, öffnet sich ein Eingabefeld, wo Sie mehrzeiligen ST-Code eingeben können.

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

KOP-Element 'Kontakt'

Symbol: , im Editor 

Das Element ist nur im KOP-Editor verfügbar.

Ein Kontakt gibt das Signal `TRUE` (ON) oder `FALSE` (OFF) von links nach rechts weiter, bis das Signal schließlich eine Spule im rechten Teil des Netzwerks erreicht. Zu diesem Zweck wird dem Kontakt eine boolesche Variable zugewiesen, die das Signal enthält. Dazu ersetzen Sie den Platzhalter `???` oberhalb des Kontakts mit dem Namen einer booleschen Variablen

Sie können mehrere Kontakte sowohl in Reihe als auch parallel anordnen. Bei zwei parallelen Kontakten muss nur einer den Wert `TRUE` erhalten, damit ON nach rechts weitergegeben wird. Wenn Kontakte hintereinander geschaltet sind, müssen alle Kontakte den Wert `TRUE` erhalten, damit vom letzten Kontakt der Reihe ON nach rechts weitergegeben wird. Sie können also mit KOP elektrische Parallel- und Reihenschaltungen programmieren.

Ein negierter Kontakt  gibt das Signal `TRUE` weiter, wenn der Variablenwert `FALSE` ist. Sie können einen eingefügten Kontakt mit Hilfe des Befehls **FUP/KOP/AWL ► Negation** negieren oder einen negierten Kontakt aus der Ansicht **Werkzeuge** einfügen.

Wenn Sie bei selektiertem Netzwerk mit der Maus auf einen Kontakt gehen und die linke Maustaste gedrückt halten, erscheint im Netzwerk die Schaltfläche **In Spule umwandeln**. Wenn Sie nun mit der weiterhin gedrückten Maustaste den Mauszeiger auf diese Schaltfläche bewegen und über der Schaltfläche die Maustaste loslassen, wandelt IndraLogic den Kontakt in eine Spule um.

Siehe auch

- [Kap. "Befehl 'Kontakt einfügen'" auf Seite 706](#)
- [Kap. "Befehl 'Negierten Kontakt einfügen'" auf Seite 708](#)
- [Kap. "Befehl 'Kontakt einfügen \(rechts\)'" auf Seite 706](#)
- [Kap. "Befehl 'Kontakt parallel einfügen \(oberhalb\)'" auf Seite 707](#)
- [Kap. "Befehl 'Kontakt parallel einfügen \(unterhalb\)'" auf Seite 707](#)

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

KOP-Element 'Spule'

Symbol: , im Editor 

Das Element ist nur im KOP-Editor verfügbar.

Eine Spule nimmt den Wert auf, der von links geliefert wird und speichert ihn in der ihr zugewiesenen booleschen Variablen.

Ihr Eingang kann den Wert `TRUE` (ON) oder `FALSE` (OFF) haben.

Mehrere Spulen in einem Netzwerk können nur parallel angeordnet werden.

In einer negierten Spule  wird der negierte Wert des eingehenden Signals in der booleschen Variablen gespeichert, die der Spule zugewiesen ist.

Set-Spule, Reset-Spule

Symbol: , , im Editor: , 

Set-Spule: Wenn der Wert `TRUE` an einer Set-Spule ankommt, behält die Spule den Wert `TRUE`. Solange die Applikation läuft, kann der Wert an dieser Stelle nicht mehr überschrieben werden.

Reset-Spule: Wenn der Wert `TRUE` an einer Reset-Spule ankommt, behält die Spule den Wert `FALSE`. Solange die Applikation läuft, kann der Wert an dieser Stelle nicht mehr überschrieben werden.

Sie können eine eingefügte Spule mit Hilfe des Befehls **FUP/KOP/AWL ▶ Set/Reset** als Set- oder Reset-Spule definieren oder als Elemente **Set-Spule** und **Reset-Spule** aus der Ansicht **Werkzeuge** einfügen.

Siehe auch

- [Kap. "Befehl 'Spule einfügen'" auf Seite 706](#)
- [Kap. "Befehl 'Reset-Spule einfügen'" auf Seite 706](#)
- [Kap. "Befehl 'Negation'" auf Seite 709](#)
- [Kap. "Befehl 'Set/Reset'" auf Seite 710](#)

KOP-Element 'Leitungsverzweigung Start/Ende'

Symbol: 

Das Element dient der geschlossenen Leitungsverzweigung.

Siehe auch

- [Kap. "Geschlossene Leitungsverzweigung" auf Seite 286](#)
- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "Befehl 'Leitungsverzweigung Startpunkt setzen'" auf Seite 711](#)
- [Kap. "Befehl 'Leitungsverzweigung Endpunkt setzen'" auf Seite 711](#)

Geschlossene Leitungsverzweigung

Eine geschlossene Leitungsverzweigung steht nur im KOP zur Verfügung und enthält einen Start- und einen Endpunkt. Sie dient der Implementierung einer parallelen Auswertung von logischen Elementen.

Einfügen einer geschlossenen Leitungsverzweigung

- Befehl **FUP/KOP/AWL ▶ Kontakt parallel einfügen (unterhalb)**
- Befehl **FUP/KOP/AWL ▶ Kontakt parallel einfügen (oberhalb)**

Geschlossene Leitungsverzweigung an einem Kontakt

Geschlossene Leitungsverzweigung an einem Baustein, OR-Auswertung, Kurzschlussauswertung

• Befehl FUP/KOP/AWL ► Verzweigung Start-/ Endpunkt setzen

Wenn Sie einen oder mehrere Kontakte markiert haben und Sie einen Befehl „Kontakt parallel einfügen“ ausführen, wird eine parallele Verzweigung mit einem einfachen vertikalen Strich eingefügt. Bei dieser Verzweigung durchläuft der Signalfluss beide Zweige. Es handelt sich um ein OR-Konstrukt der beiden Zweige.

Neu: Wenn Sie einen Baustein markiert haben und Sie einen Befehl "Kontakt parallel einfügen" ausführen, wird eine parallele Verzweigung mit einem doppelten vertikalen Strich eingefügt. Dies signalisiert, dass eine Kurzschlussauswertung ("Short Circuit Evaluation" - "SCE") implementiert ist. Die Kurzschlussauswertung ermöglicht es, die Ausführung eines Funktionsbausteins mit einem booleschen Ausgang zu umgehen, wenn eine bestimmte Bedingung `TRUE` ist. Die Bedingung kann im KOP-Editor durch einen dem Funktionsbaustein parallel geschalteten Zweig dargestellt werden. Die Kurzschlussbedingung wird durch einen oder mehrere Kontakte in diesem Zweig definiert, die parallel oder sequentiell verschaltet sind.

Funktionsweise:

Zuerst werden die Zweige betrachtet, die nicht den Funktionsbaustein enthalten. Wenn IndraLogic für einen dieser Zweige der Wert `TRUE` ermittelt, dann wird der Funktionsbaustein im parallelen Zweig nicht aufgerufen. In diesem Fall wird der Wert, der am Eingang des Funktionsbausteins anliegt, unmittelbar auf den Ausgang übertragen. Wenn IndraLogic für die Kurzschlussauswertungsbedingung `FALSE` ermittelt, dann wird der Baustein aufgerufen und das boolesche Ergebnis seiner Abarbeitung wird weitergegeben. Wenn alle Zweige Funktionsbausteine enthalten, werden sie von oben nach unten ausgewertet und ihre Ausgänge mit logischen OR-Operationen verknüpft. Wenn es keine Zweige mit Funktionsbausteinen gibt, werden normale OR-Operationen durchgeführt.

Beispiel

Die Funktionsbaustein-Instanz `x1` (TON) hat einen booleschen Eingang und einen booleschen Ausgang. Die Ausführung von `x1` wird ausgelassen, wenn für die Bedingung in der parallelen Leitungsverzweigung `TRUE` ermittelt wurde. Der Bedingungswert resultiert aus den OR- und AND-Operationen, die die Kontakte `cond1`, `cond2` und `cond3` verknüpfen.

Wenn der Bedingungswert aus der Verknüpfung der Kontakte `cond1`, `cond2` und `cond3` `FALSE` ist, wird `x1` ausgeführt.

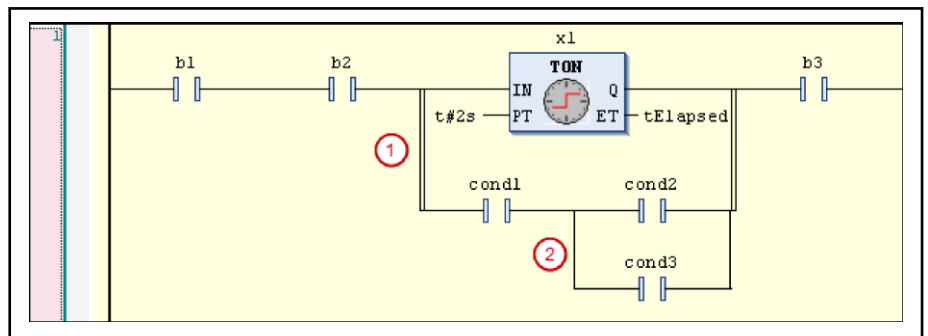


Abb. 17-23:

(1) zeigt durch die doppelt gezeichneten Vertikalverbindungen an, dass es sich um ein Konstrukt handelt, das einer Kurzschlussauswertung unterliegt.

(2) zeigt durch die einfach gezeichneten Vertikalverbindungen an, dass es sich um ein OR-Konstrukt handelt.

Im Folgenden wird das oben gezeigte KOP-Beispiel als ST-Code gezeigt. Dabei sind P_IN und P_OUT die booleschen Werte am Eingang (Aufspaltungspunkt) und Ausgang (Wiedervereinigungspunkt) der parallelen Leitungsverzweigung.

```
P_IN := b1 AND b2;

IF ((P_IN AND cond1) AND (cond2 OR cond3)) THEN
  P_OUT := P_IN;
ELSE
  x1(IN := P_IN, PT := t#2s);
  tElapsed := x1.ET;
  P_OUT := x1.Q;
END_IF
bRes := P_OUT AND b3;
```

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "Befehl 'Leitungsverzweigung Startpunkt setzen'" auf Seite 711](#)
- [Kap. "Befehl 'Leitungsverzweigung Endpunkt setzen'" auf Seite 711](#)
- [Kap. "Befehl 'Kontakt parallel einfügen \(oberhalb\)'" auf Seite 707](#)
- [Kap. "Befehl 'Kontakt parallel einfügen \(unterhalb\)'" auf Seite 707](#)

17.1.6 Continuous Function Chart (CFC) und Seitenorientierter CFC

Ein Funktionsplan besteht von außen betrachtet aus Eingängen und Ausgängen und dazwischen werden Daten verarbeitet. Intern besteht ein Funktionsplan aus Bausteinen und deren Verbindungen, die Daten (Signale) repräsentieren und wie Zuweisungsoperatoren in ST fungieren. Das Gesamtverhalten setzt sich zusammen aus dem Verhalten der eingefügten Bausteine, die andere Programmierbausteine oder Bibliotheksbausteine aufrufen.

Code in der Implementierungssprache "Continuous Function Chart (CFC)" veranschaulicht vor allem den Datenfluss durch das System. Ein Continuous Function Chart wird deshalb auch als Signalflussplan bezeichnet.

Sie können im seitenorientierten CFC-Editor Programmierbausteine miteinander verdrahten und anschauliche Funktionspläne auf Seiten verteilt erstellen. Der seitenorientierte Editor verhält sich wie der CFC-Editor, unterstützt Sie aber durch folgende Funktionalität:

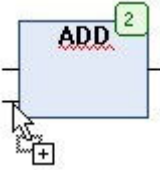
- Erstellen von Seiten
- Einstellen der Seitengröße
- Kopieren und Einfügen von Seiten im Seitennavigator
- Kopieren der Implementierung eines Programmierbausteins der Implementierungssprache CFC und Einfügen in einer Seite
- Übersichtliches und platzsparendes Arrangieren von Eingängen, Ausgängen und Verbindungsmarken in den Randbereichen
- Verbindung über Seiten hinweg mit Verbindungsmarken

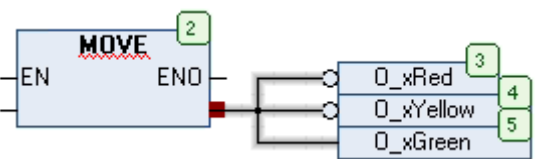
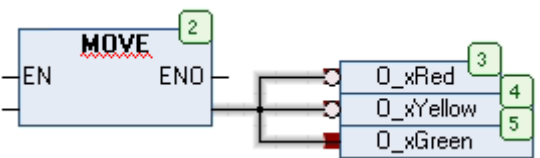
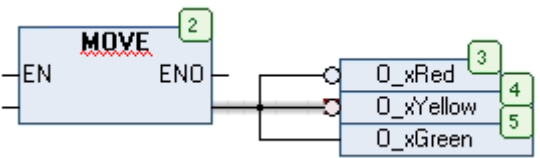
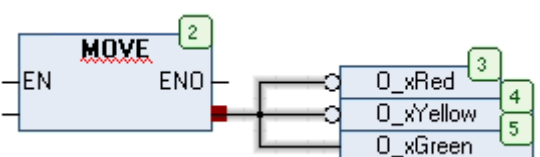
CFC-Editor

Editor konfigurieren

Sie können das Aussehen, Verhalten und Drucken in den IndraWorks-Optionen unter **Extras ► Optionen ► IndraLogic 2G** Kategorie "CFC-Editor" projektweit konfigurieren. Beispielsweise können Sie in Registerkarte "Ansicht" die Farbe der Verbindungslinien abhängig vom Datentyp konfigurieren.

Editieren

Cursorsymbol 	Voraussetzung: In der Ansicht "Werkzeuge" ist "Zeiger" selektiert (Aufruf der Ansicht durch Doppelklick in den Arbeitsbereich des CFC-Editors). Das Symbol weist darauf hin, dass Sie im Editor editieren können. Selektieren Sie Elemente oder Verbindungen, um sie zu verschieben oder um Befehle auszuwählen.
Cursorsymbol 	Voraussetzung: In der Ansicht "Werkzeuge" ist ein Element selektiert (Aufruf der Ansicht durch Doppelklick in den Arbeitsbereich des CFC-Editors). Mit einem Mausklick in den Editor wird das selektierte Element eingefügt. Alternativ können Sie mit Drag&Drop ein Element in den Editor ziehen.
Drag&Drop einer Funktionsbausteininstanz aus der Deklaration in den Editor	Voraussetzung: In der Deklaration des CFCs ist eine Deklarationszeile selektiert. Die Instanz wird als Baustein mit Bausteinname und Bausteintyp und allen Anschlüssen eingefügt.
Drag&Drop einer Variablen aus der Deklaration in den Editor an einen Bausteinanschluss	Die Variable wird als Eingang oder Ausgang mit Verbindung zum fokussierten Bausteineanschluss eingefügt. Tipp: Der Cursor zeigt an, ob Sie eine gültige Stelle für eine Variable fokussieren: 
[Strg] + Klick in den Programmierbereich	Voraussetzung: In der Ansicht "Werkzeuge" ist ein Element selektiert (Aufruf der Ansicht durch Doppelklick in den Arbeitsbereich des CFC-Editors). Solange Sie [Strg] gedrückt halten, wird bei jedem Klick in den Programmierbereich ein selektiertes Element erzeugt.

[Strg] + [Pfeil nach rechts]	Voraussetzung: Im CFC Programm ist bei einem Element genau ein Ausgangsanschluss (Ausgangsspin) selektiert.  Die Selektion wird so verschoben, dass der Eingangsanschluss (Eingangspin) am Ende der Verbindungslinie selektiert ist. Bei mehreren Anschlüssen sind alle selektiert. 
[Strg] + [Pfeil nach links]	Voraussetzung: Im CFC Programm ist bei einem Element genau ein Eingangsanschluss (Eingangspin) selektiert.  Die Selektion wird so verschoben, dass der Ausgangsanschluss am Anfang der Verbindungslinie selektiert ist. Bei mehreren Anschlüssen sind alle selektiert. 

Tab. 17-8: Funktionen im CFC-Editor

Siehe auch [Kap. 17.1.2 "Generelle Funktionalitäten in allen grafischen Editoren"](#) auf Seite 240

Verbinden

Sie können zwischen Elementanschlüssen Verbindungslinien einfügen. Verbindungslinien werden mit Autorouting eingefügt, so dass Verbindungslinien automatisch optimal und möglichst kurz sind. Die Verbindungslinien werden auf Kollisionen überprüft.

Drag&Drop von einem Anschluss zu einem anderen	Zwischen den 2 Elementanschlüssen wird eine Verbindungslinie eingefügt.
Befehl "Selektierte Anschlüsse verbinden"	Voraussetzung: Sie haben mehrere Anschlüsse selektiert und die zu verbindenden Anschlüsse sind rot gekennzeichnet.

Eingefügtes Element so verschieben, dass es den Anschluss eines anderen Elements berührt.	Voraussetzung: Option "Automatisches Verknüpfen aktivieren" ist aktiviert. Die sich berührenden Anschlüsse werden automatisch miteinander verbunden.
Verbindungssymbol 	Das Verbindungssymbol befindet sich in der rechten oberen Ecke im Editor. Ein grün umrandetes Symbol weist auf kollisionsfreie Verbindungen hin. Ein rot umrandetes Symbol weist auf Kollisionen hin. Mit Klick auf das Symbol öffnet sich ein Kontextmenü mit Befehlen zur Kollisionsbearbeitung, beispielsweise der Befehl "Nächste Kollision zeigen".
Befehl "Verbindungsmarke" 	Voraussetzung: Sie haben eine Verbindung selektiert und den Befehl "Verbindungsmarke" gewählt. Statt einer langen Verbindungslinie wird eine Verbindung durch Verbindungsmarken dargestellt.

Tab. 17-9: Funktionen im CFC-Editor

Siehe auch [Kap. "Befehl 'Nächste Kollision zeigen'" auf Seite 696](#)

Befehle beim Editieren

Siehe [Kap. 18.2 "Menübefehle" auf Seite 610](#)

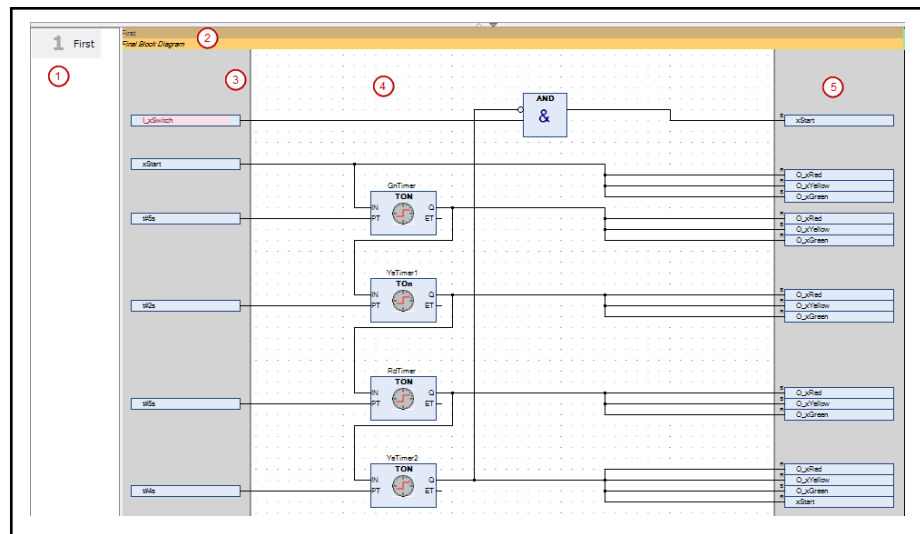
Weiterführende Dokumentationen:

- [Kap. 17.1.2 "Generelle Funktionalitäten in allen grafischen Editoren" auf Seite 240](#)
- [Kap. "Im CFC-Editor programmieren" auf Seite 100](#)
- [Kap. 18.2.8 "Menü 'CFC'" auf Seite 689](#)

CFC-Editor seitenorientiert



Sie können Programmierbausteine, die in der Implementierungssprache "Continuous Function Chart (CFC) - seitenorientiert" erzeugt wurden, nicht in Programmierbausteine der Implementierungssprache "Continuous Function Chart (CFC)" umwandeln und umgekehrt.



- ① Seitennavigator
- ② Kopfzeile der Seite mit Seitennamen und Seitenbeschreibung
- ③ Linker Randbereich reserviert für Eingänge und Zielverbindungsmarken
- ④ Programmbereich
- ⑤ Rechter Randbereich reserviert für Ausgänge und Quellverbindungsmarken

Abb. 17-24:

Editieren Ein neu erstelltes, seitenorientiertes CFC-Objekt enthält standardmäßig eine Seite zur Erstellung des Programms.

Zusätzliche Seiten können per Drag&Drop in die Seitennavigation hinzugefügt werden. Öffnen Sie hierzu die Ansicht "Werkzeuge" mittels Doppelklick im CFC-Editor und ziehen sie das Element "Seite" aus der Ansicht in die Seitennavigation.

Bestehende Seiten können vervielfacht werden. Selektieren Sie hierzu die gewünschte Seite in der Seitennavigation und führen sie den Kontextbefehl **Kopieren** und danach **Einfügen** aus.

Verbindungen über Seiten hinweg erstellen Sie mit Hilfe der Elemente "Verbindungsmarke - Quelle" und "Verbindungsmarke - Ziel". Wenn Sie eine Verbindungslinie von einem Eingangspin oder einem Ausgangspin auf den Randbereich ziehen, wird automatisch eine neue Verbindungsmarke erzeugt. Vorteilhaft ist, dass die Eingabeunterstützung "Komponenten auflisten" alle bereits definierten Verbindungsmarken-Quellen anbietet.

Wenn Sie im Editor ein Element selektiert haben, können Sie mit Hilfe der Pfeiltasten die Selektion von einem zum nächsten Element umsetzen und so durch die Schaltung navigieren. Wenn Sie dann eine Verbindungsmarke selektiert haben und noch eine Pfeiltasten drücken, wird sogar die zugehörige Verbindungsmarke der nächsten/vorherigen Seite selektiert.

Sie können Netzwerke aus einem CFC-Programmieraustausch mit den Kontextbefehlen "Kopieren" und "Einfügen" (über die Zwischenablage) in den Programmbereich eines seitenorientierten CFCs übertragen. Alternativ können Sie Drag&Drop verwenden.

Ausführungsreihenfolge Die Ausführungsreihenfolge wird automatisch nach der Reihenfolge der Seiten festgelegt, so wie sie im Seitennavigator des Editors sortiert sind. Innerhalb einer Seite verhält sich ein seitenorientiertes CFC-Objekt so wie ein CFC-Objekt.

Siehe auch:

- [Kap. 17.1.2 "Generelle Funktionalitäten in allen grafischen Editoren" auf Seite 240](#)
- [Kap. "Im CFC-Editor programmieren" auf Seite 100](#)
- [Kap. 18.2.8 "Menü 'CFC'" auf Seite 689](#)

CFC-Editor im Onlinebetrieb

Im Onlinebetrieb können Sie Variablenwerte der Steuerung überwachen und diese auch verändern. Weiterhin können Sie die Debugging-Funktionalitäten wie Haltepunkte und schrittweises Ausführen nutzen.

Monitoring

Im Onlinebetrieb ermöglicht die Inline-Monitoring Funktion die Anzeige der aktuellen Variablenwerte im Deklarationsteil und Implementierungsteil des CFC-Editors. Bei einem Funktionsbaustein ist das Inline-Monitoring nur dann möglich, wenn eine Instanz des Funktionsbausteins geöffnet ist. In der Ansicht der Basisimplementierung werden keine Werte angezeigt.

Boolesche Variable monitoren

Die Verbindungen zwischen booleschen Variablen werden ihrem Istwert entsprechend farblich dargestellt: **TRUE** in blau und **FALSE** in schwarz. Die Elementanschlüsse werden mit dem Istwert ausgezeichnet.

Beispiel:

Eine Applikation enthält einen CFC-Programmierbaustein. Dort wird eine interne boolesche Variable umgeschaltet: Die Variable `xToggle` wechselt mit jedem Buszyklus ihren Zustand von **TRUE** auf **FALSE**.

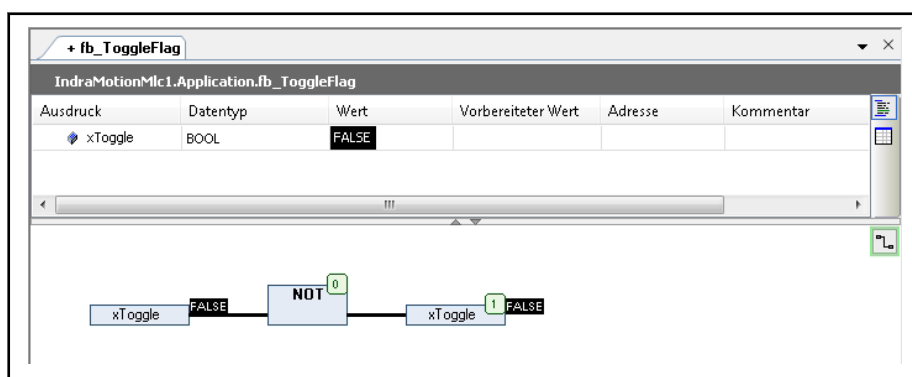


Abb. 17-25:

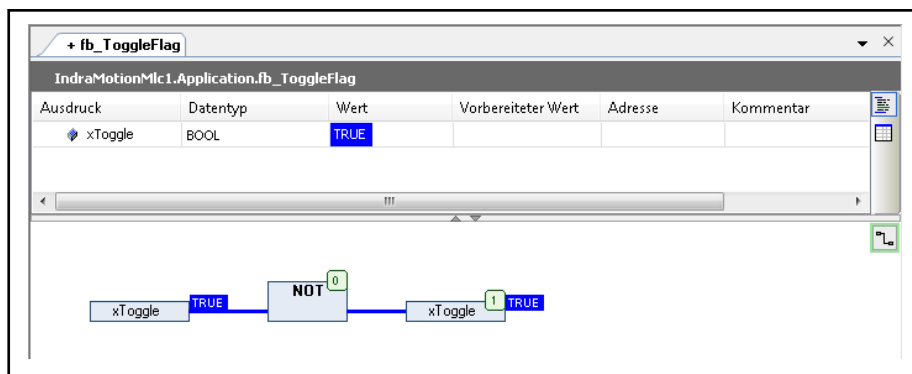


Abb. 17-26:

Skalare Variable monitoren

Bei skalaren Variablen werden die Elementanschlüsse mit den Variablen-Istwerten ausgezeichnet.

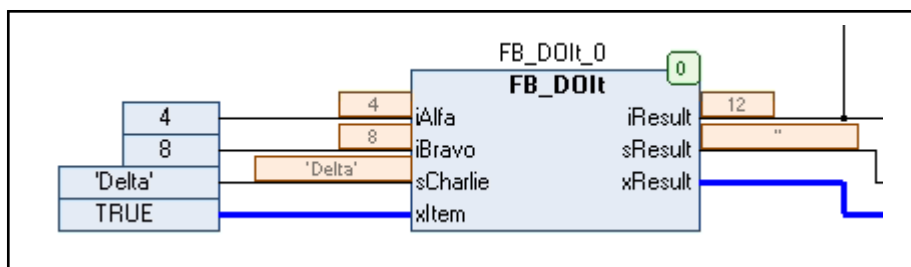


Abb. 17-27:

Forcen und Schreiben von Variablen

Im Onlinebetrieb können Sie im Deklarationseditor einen Wert für das Forcen oder Schreiben einer gemonitorten Variable vorbereiten.

Im Implementierungsteil können ebenfalls Werte vorbereitet werden, Voraussetzung hierfür ist, dass die IndraLogic-Option "Werte im Implementierungsteil vorbereiten" aktiviert ist.

Sie finden die Option im IndraWorks Menü **Extras ▶ Optionen ▶ IndraLogic 2G** in der Kategorie **CFC-Editor**.

Zum Vorbereiten von Werten öffnen Sie den Dialog **Wert vorbereiten** durch einen Doppelklick auf die auf die Monitoring-Box neben dem jeweiligen Element, oder direkt auf das Element.

Bei booleschen Variablen erscheint kein Dialog, hier kann der Wert durch Mausklick in das Feld "Vorbereiteter Wert" gesetzt werden. Mit jedem Mausklick wird zwischen **TRUE** und **FALSE** gewechselt.

Vorbereitete Werte werden in spitzen Klammern angezeigt. Nach dem Ausführen des Schreibens oder Forcens erscheint in der Monitoring-Box ein rotes "F".

Konstante Eingangsparameter von Funktionsbaustein-Instanzen ändern

Sie können Eingangsparameter von Funktionsbaustein-Instanzen vom Typ **VAR_INPUT CONSTANT** im Onlinebetrieb schreiben und so die Parameter justieren. Nach dem Ausloggen übernehmen Sie dann die Parameter mit dem Befehl "Vorbereitete Parameterwerte übernehmen" in Ihr Projekt.

Voraussetzung: Ein CFC-Editor ist aktiv. Es ist ein Funktionsbaustein instanziiert, der in seiner Deklaration **VAR_INPUT CONSTANT**-Variablen hat.

1. Öffnen Sie den Baustein mit dem Aufruf der Funktionsbaustein-Instanz im Editor.

→ Die Deklaration von **FB_DOIt** wurde ergänzt um die Konstante **MAXIMUM**.

Programm:

```
FUNCTION_BLOCK FB_DoIt
VAR_INPUT
  iAlfa : INT;
  iBravo: INT;
  sCharlie : STRING := 'Charlie';
  xItem : BOOL;
  iDelta : INT;
END_VAR
VAR_INPUT CONSTANT
  MAXIMUM : INT := 12;
END_VAR
VAR_OUTPUT
  iResult : INT;
  sResult : STRING;
  xResult : BOOL;
```

Die grafische Darstellung der Funktionsbaustein-Instanzen enthält jetzt die Schaltfläche "Parameter":

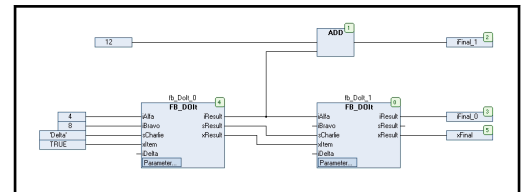


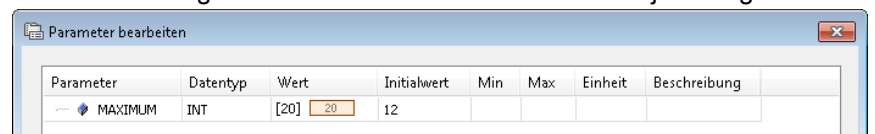
Abb. 17-28:

2. Loggen Sie sich auf der Steuerung ein
3. Klicken Sie auf die Schaltfläche "Parameter" bei einer der Funktionsbausteininstanzen.
→ Der Dialog "Parameter bearbeiten" öffnet sich.
4. Klicken Sie in der Spalte "Wert" in ein Inline-Monitoring-Feld eines Parameters.
→ Der Dialog "Wert vorbereiten" öffnet sich.
5. Geben Sie beispielsweise den Wert 20 in das Feld "Einen neuen Wert für die nächste Schreib- oder Force-Operation vorbereiten" ein.
6. Bestätigen Sie die Eingabe mit "OK".

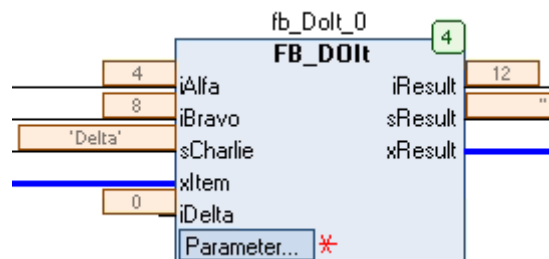
Der vorbereitete Wert wird neben dem aktuellen Wert in spitzen Klammern dargestellt (im Beispiel <20>)



7. Schreiben Sie den vorbereiteten Wert mit dem Befehl **Debuggen ► Werte schreiben**.
→ Der vorbereitete Wert wird geschrieben. Der Parameter ist geändert und wird in eckigen Klammern hinter dem Wert im Projekt dargestellt.



Neben dem Parameterfeld der Funktionsbausteininstanz wird der Unterschied der beiden Werte durch ein rotes Kreuz dargestellt



8. Schließen Sie den Dialog "Parameter bearbeiten" und Loggen Sie sich aus.

9. Wählen Sie den Befehl **CFC ▶ Vorbereitete Parameterwerte übernehmen**.

Der geänderte Parameterwert wird in das Projekt übernommen. Der Stern neben dem Parameterfeld verschwindet.

Haltepunkt-Positionen Mögliche Position eines Haltepunkts:

- Element "Ausgang": Variablen werden beschrieben.
- Element "Baustein": Programmierbausteine werden aufgerufen.
- Element "RETURN": Der Programmverlauf verzweigt sich.
- Element "Selektor": Strukturelemente werden beschrieben.

Wählen Sie den Befehl Debuggen → Haltepunkt umschalten, um einen Haltepunkt zu setzen oder einen bestehenden zu löschen. Rote Kreise im Blockschaltbild stellen einen aktivierten Haltepunkt dar

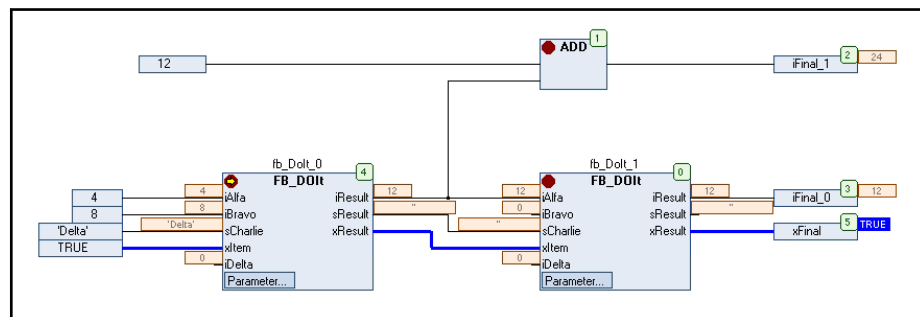


Abb. 17-29:



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 Funktionsbausteinen, die diese Schnittstelle implementieren, Haltepunkte gesetzt. Dies gilt ebenfalls in allen abgeleiteten Funktionsbausteinen, die die Methode definieren.

Programmierbaustein schrittweise ausführen

Sie können einen Programmierbaustein im Debugbetrieb schrittweise abarbeiten. Ein Programmierbaustein, der aufgerufen wird, wird intern ergänzt um einen RETURN am Anfang vor dem Element mit der Nummer 0 und am Ende nach dem letzten Element. Bei der schrittweisen Abarbeitung werden diese automatisch angesprungen.

Befehle im Onlinebetrieb

Siehe:

- [Kap. "Befehl 'FB-Eingang forcen'" auf Seite 700](#)
- [Kap. "Befehl 'Funktionsbausteinelement für Forcen vorbereiten'" auf Seite 699](#)
- [Kap. "Befehl 'Parameter bearbeiten'" auf Seite 694](#)
- [Kap. "Befehl 'Vorbereitete Parameter im Projekt speichern'" auf Seite 695](#)

Siehe auch:

- [Kap. 13.1 "Monitoring von Werten" auf Seite 205](#)

- [Kap. 12.4 "Forcen und Schreiben von Variablen" auf Seite 198](#)
- [Kap. 12.2 "Verwenden von Haltepunkten" auf Seite 193](#)
- [Kap. 12.3 "Schrittweises Abarbeiten eines Programms \(Stepping\)" auf Seite 196](#)

Elemente

CFC-Element 'Seite'

Symbol: 

Das Element fügt eine neue Seite in den Editor ein. Es ist nur im seitenorientierten CFC-Editor verfügbar. Die Nummer der Seite wird automatisch nach ihrer Position vergeben. Sie können den Namen und die Beschreibung der Seite in die orangene Kopfzeile eingeben. Die Seitengröße passen Sie mit dem Befehl **Seitengröße bearbeiten** an.

Siehe auch

- [Kap. "Befehl 'Seitengröße bearbeiten'" auf Seite 689](#)

CFC-Element 'Kontrollpunkt'

Symbol: 

Verwenden Sie einen Kontrollpunkt, um Punkte einer Verbindung zu fixieren, bevor Sie die Linienführung anpassen. Dazu ziehen Sie das Element an die gewünschte Position auf einer Verbindungslinie. Verbindungslinien mit Kontrollpunkten werden nicht mehr automatisch geroutet.

Siehe auch

- [Kap. "Im CFC-Editor programmieren" auf Seite 100](#)
- [Kap. "Befehl 'Kontrollpunkt erzeugen'" auf Seite 698](#)
- [Kap. "Befehl 'Kontrollpunkt entfernen'" auf Seite 698](#)

CFC-Element 'Eingang'

Symbol: 

IndraLogic fügt ein Eingangselement standardmäßig mit dem Text "???" ein. Sie können dieses Feld durch Anklicken direkt editieren und einen konstanten Wert oder einen Variablennamen eingeben. Alternativ können Sie durch Anklicken von  den Eingabeassistenten öffnen um eine Variable auszuwählen.

CFC-Element 'Ausgang'

Symbol: 

IndraLogic fügt ein Ausgangselement standardmäßig mit dem Text "???" ein. Sie können dieses Feld durch Anklicken direkt editieren und einen konstanten Wert oder einen Variablennamen eingeben. Alternativ können Sie durch Anklicken von  den Eingabeassistenten öffnen um eine Variable auszuwählen.

CFC-Element 'Baustein'

Symbol: 

Sie verwenden das Element, um einen Operator, eine Funktion, einen Funktionsbaustein oder ein Programm einzufügen. IndraLogic fügt das Element

standardmäßig mit dem Namen ??? ein. Sie können dieses Feld durch Anklicken direkt editieren und einen Bausteinnamen eingeben. Alternativ können Sie durch Anklicken von  den Eingabeassistenten öffnen und einen Baustein auswählen.

Im Fall eines Funktionsbausteins zeigt IndraLogic zusätzlich oberhalb des Bausteinsymbols ein Eingabefeld an (???). Diesen Namen müssen Sie durch den Namen der Funktionsbaustein-Instanz ersetzen. Wenn Sie einen Funktionsbaustein mit konstanten Eingangsparametern instanzieren, zeigt das Bausteinelement in der linken unteren Ecke das Feld "Parameter..." an. Die Parameter bearbeiten Sie durch Anklicken dieses Felds.

Um eine existierende Box zu ersetzen, ersetzen Sie nur den aktuell eingefügten Bezeichner mit dem gewünschten neuen Namen. Beachten Sie dabei, dass IndraLogic die Anzahl an Ein- und Ausgangspins entsprechend der Definition der POU anpasst und deshalb möglicherweise existierende Zuweisungen gelöscht werden.

Siehe auch

- [Kap. "Befehl 'Parameter bearbeiten'" auf Seite 694](#)

CFC-Element 'Sprung'

Symbol: 

Sie verwenden das Element, um eine Position festzulegen, an der die Programmabarbeitung fortsetzen soll. Diese Zielposition müssen Sie durch eine Marke definieren. Dazu geben Sie den Namen der Marke in das Eingabefeld ??? ein. Falls Sie die entsprechende Marke bereits eingefügt haben, können Sie diese auch über den Eingabeassistenten () auswählen.

Siehe auch

- [Kap. "CFC-Element 'Marke'" auf Seite 298](#)

CFC-Element 'Marke'

Symbol: 

Eine Marke definiert eine Position, zu der die Programmabarbeitung mit Hilfe eines Sprung-Elements springt.

Im Online-Modus fügt IndraLogic automatisch eine RETURN-Marke am Ende eines CFC-Bausteins ein.

Siehe auch

- [Kap. "CFC-Element 'Sprung'" auf Seite 298](#)

CFC-Element 'Return'

Symbol: 

Verwenden Sie das Element, um den Baustein zu verlassen.

Beachten Sie, dass im Online-Modus im CFC-Editor ein Return-Element automatisch vor der ersten Zeile und nach dem letzten Element eingefügt wird. Bei der schrittweisen Abarbeitung springt IndraLogic das Return-Element am Ende automatisch an, bevor der Baustein verlassen wird.

CFC-Element 'Kompositor'

Symbol: 

Das Komposer-Element dient der Handhabung von Strukturkomponenten. Dabei werden Ihnen die einzelnen Komponenten einer Struktur als Eingang zur Verfügung gestellt. Zu diesem Zweck müssen Sie das Kompositorelement wie die betreffende Struktur nennen (Ersetzen der ???).

Das Kompositorelement ist das Gegenstück zum Selektorelement.

Siehe auch

- [Kap. "CFC-Element 'Selektor'" auf Seite 299](#)

CFC-Element 'Selektor'

Symbol: 

Das Selektorelement dient der Handhabung von Strukturkomponenten. Dabei werden Ihnen die einzelnen Komponenten einer Struktur als Ausgang zur Verfügung gestellt. Zu diesem Zweck müssen Sie das Selektorelement wie die betreffende Struktur nennen (Ersetzen der ???)

Das Selektorelement ist das Gegenstück zum Kompositorelement.

Siehe auch

- [Kap. "CFC-Element 'Kompositor'" auf Seite 298](#)

CFC-Element 'Kommentar'

Symbol: 

Mit diesem Element geben Sie einen Kommentar im CFC-Editor ein. Ersetzen Sie den Platzhaltertext im Element durch den Kommentartext. Ein Zeilenumbruch kann mit Hilfe der Tastenkombination <Strg> + <Eingabe> eingefügt werden.

CFC-Element 'Verbindungsmarke-Quelle/Ziel'

Symbol: 

Sie können Verbindungsmarken anstelle einer Verbindungslinie zwischen Elementen verwenden. Das hilft Ihnen, komplexe Diagramme übersichtlicher darzustellen.

Für eine gültige Verbindung müssen Sie ein Element **Verbindungsmarke - Quelle** mit dem Ausgang eines Elements und ein Element **Verbindungsmarke - Ziel** mit dem Eingang eines anderen Elements verbinden. Beide Marken müssen den gleichen Namen tragen. Dabei wird die Groß-/Kleinschreibung nicht berücksichtigt.

Hinweise zur Namensgebung

- Der Standardnamen für Verbindungsmarken ist C-<nr>. Dabei ist <nr> eine fortlaufende Nummer, beginnend mit 1.
- Sie können den Standardnamen umbenennen. Dabei müssen Sie beachten, dass die Quellmarke und die Zielmarke den gleichen Namen haben.
- Wenn Sie den Namen der Quellmarke verändern, wird der Zielname automatisch umbenannt.
- Wenn Sie den Namen der Zielmarke verändern, bleibt der Quellname erhalten.



Beachten Sie den Befehl **Verbindungsmarke** zur automatischen Umwandlung einer bestehenden Verbindung.

Siehe auch

- [Kap. "Befehl 'Verbindungsmarke'" auf Seite 699](#)
- [Kap. "CFC-Editor" auf Seite 288](#)

CFC-Element 'Bausteineingang'

Symbol: 

Abhängig vom Bausteintyp können Sie einem eingefügten Bausteinelement weitere Eingänge hinzufügen. Dazu müssen Sie das Bausteinelement selektieren und das Bausteineingangelement auf den Bausteinrumpf ziehen.



Sie können eine Eingangs- oder Ausgangsverbindung bei gedrückt gehaltener Taste <Strg> auf eine andere Position am Baustein ziehen.

Siehe auch

- [Kap. "CFC-Element 'Bausteingang'" auf Seite 300](#)

CFC-Element 'Bausteingang'

Symbol: 

Abhängig vom Bausteintyp können Sie einem eingefügten Bausteinelement weitere Ausgänge hinzufügen. Dazu müssen Sie das Bausteinelement selektieren und das Bausteingangselement auf den Bausteinrumpf ziehen.



Sie können eine Eingangs- oder Ausgangsverbindung bei gedrückt gehaltener Taste <Strg> auf eine andere Position am Baustein ziehen.

Siehe auch

- [Kap. "CFC-Element 'Bausteingang'" auf Seite 300](#)

17.2 Variablen

Der Gültigkeitsbereich einer Variablen definiert, wie und wo Sie die Variable verwenden können. Sie legen den Gültigkeitsbereich bei der Variablendeklaration fest.

17.2.1 Lokale Variablen - VAR

Die Deklaration lokaler Variablen erfolgt im Deklarationsteil von Programmobjekten zwischen den Schlüsselwörtern `VAR` und `END_VAR`.

Sie können auf lokale Variablen über den Instanzpfad lesend zugreifen.

Sie können lokale Variablen mit einem Attribut-Schlüsselwort erweitern.

Beispiel

```
VAR
    iVar1 : INT;
END_VAR
```

Siehe auch [Kap. 9.16 "Datenpersistenz" auf Seite 144](#)

17.2.2 Eingabevariablen - VAR_INPUT

Eingabevariablen sind Eingangsvariablen für einen Baustein.

VAR_INPUT-Variablen deklarieren Sie im Deklarationsteil von Programmierobjekten zwischen den Schlüsselwörtern VAR_INPUT und END_VAR.

Sie können Eingabevariablen mit einem Attribut-Schlüsselwort erweitern.

Beispiel

```
VAR_INPUT
  iIn1:INT; (* 1st input variable *)
END_VAR
```

Siehe auch [Kap. 9.16 "Datenpersistenz" auf Seite 144](#)

17.2.3 Ausgabevariablen - VAR_OUTPUT

Ausgabevariablen sind Ausgangsvariablen eines Bausteins.

VAR_OUTPUT-Variablen deklarieren Sie im Deklarationsteil von Programmierobjekten zwischen den Schlüsselwörtern VAR_OUTPUT und END_VAR. IndraLogic liefert die Werte dieser Variablen an den aufrufenden Baustein zurück. Dort können Sie die Werte abfragen und weiter verwenden.

Sie können Ausgabevariablen mit einem Attribut-Schlüsselwort erweitern.

Beispiel:

```
VAR_OUTPUT
  iOut1:INT; (*1st output variable *)
END_VAR
```

Siehe auch [Kap. 9.16 "Datenpersistenz" auf Seite 144](#)

Ausgabevariablen in Funktionen und Methoden

Gemäß der Norm IEC 61131-3 können Funktionen (und Methoden) zusätzliche Ausgänge haben. Die zusätzlichen Ausgänge müssen Sie beim Aufruf der Funktion wie im folgenden Beispiel zuweisen.

Beispiel:

```
fun(iIn1 := 1, iIn2 := 2, iOut1 => iLoc1, iOut2 => iLoc2);
```

17.2.4 Ein- und Ausgabevariable - VAR_IN_OUT

Eine VAR_IN_OUT-Variable ist eine Ein- und Ausgabevariable, die Teil einer Bausteinschnittstelle ist und als formaler Durchgangsparameter dient.

Syntax Deklaration

Programm:

```
<keyword> <POU name>
VAR_IN_OUT
  <variable name> : <data type> ( := <initialization value> )? ;
END_VAR
<keyword> : FUNCTION | FUNCTION_BLOCK | METHOD | PRG
```

Sie können eine Ein- und Ausgabevariable in den Programmierbausteinen PRG, FUNCTION_BLOCK, METHOD oder FUNCTION im Deklarationsabschnitt VAR_IN_OUT deklarieren. Als Initialisierungswert kann optional eine Konstante des deklarierten Datentyps zugewiesen werden. Die VAR_IN_OUT-Variable kann gelesen und beschrieben werden.

Verwendung

- **Aufruf:** Bei Aufruf des Programmierbausteins erhält die formale VAR_IN_OUT-Variable als Argument die tatsächliche Variable, die so genannte "Durchgangsvariable". Zur Laufzeit bei der Parameterübergabe wird dann keine Kopie erzeugt, sondern die formale Variable erhält eine Referenz auf die von außen übergebene tatsächliche Variable. Die referenziellen Variablen enthalten intern als Wert eine Speicheradresse auf den eigentlichen Wert (Übergabe als Pointer, Call-by-Reference). Es

ist nicht möglich als Argument direkt eine Konstante (Literal) oder eine Bitvariable anzugeben.

- **Schreib-/Lesezugriff innerhalb des Programmierbausteins:** Wenn innerhalb des Programmierbausteins auf die Variable geschrieben wird, wirkt das auf die übergebene Variable durch. Wenn der Programmierbaustein verlassen wird, bleiben vorgenommene Änderungen somit erhalten. Das bedeutet, dass ein Programmierbaustein seine VAR_IN_OUT-Variablen so verwendet, wie der aufrufende Programmierbaustein seine Variablen verwendet. Ein Lesezugriff ist immer erlaubt.
- **Schreib-/Lesezugriff von außen:** VAR_IN_OUT-Variablen können nicht via <function block instance name>.<variable name> direkt von außen gelesen oder beschrieben werden, dies funktioniert nur bei VAR_INPUT und VAR_OUTPUT-Variablen.
- **Übergabe von Stringvariablen:** Wenn als Argument eine Stringvariable übergeben wird, sollten die tatsächliche Variable und die formale Variable die gleiche Länge haben. Der übergebene String kann sonst unbeabsichtigt manipuliert werden. Bei VAR_OUTPUT CONSTANT-Parametern tritt dieses Problem nicht auf.
- **Übergabe von Bitvariablen:** Eine Bitvariable kann nicht direkt an eine VAR_IN_OUT-Variable übergeben werden, sondern benötigt eine Zwischenvariable.



Wenn ein String als Variable oder auch als Konstante an eine formale VAR_IN_OUT CONSTANT-Variable übergeben wird, dann wird dieser automatisch vollständig übergeben. Sie müssen nicht auf die Stringlänge achten.

Beispiel:

Übergabe eines Arrays

```

TYPE DUT_A :
STRUCT
  xA: BOOL;
  iB: INT;
END_STRUCT
END_TYPE

FUNCTION_BLOCK FB_SetArray
VAR_IN_OUT
  aData_A : ARRAY[0..1] OF DUT_A; // Formal variable
END_VAR
aData_A[0].xA := TRUE;
aData_A[0].iB := 100;

PROGRAM PLC_PRG
VAR
  fbSetA : FB_SetArray;
  aSpecialData : ARRAY[0..1] OF DUT_A; // Actual variable
END_VAR
fbSetA(aData_A := aSpecialData);

```

Expression	Type	Value	Prepared value	Address	Comment
fbSetA	FB_SetArray				
aSpecialData	ARRAY [0..1] OF DUT_A				Actual variable
aSpecialData[0]	DUT_A				
xA	BOOL	TRUE			
iB	INT	100			
aSpecialData[1]	DUT_A				
xA	BOOL	FALSE			
iB	INT	0			

Abb. 17-30:

Stringübergabe

```

{attribute 'qualified_only'}
VAR_GLOBAL
  g_sDEV_STATUS : STRING(25) := 'Device_A';
END_VAR

FUNCTION_BLOCK FB_SetStatus
  VAR_IN_OUT
    sDeviceStatus : STRING(25); // Formal parameter
  END_VAR
  sDeviceStatus := CONCAT(sDeviceStatus, ' Activ');
END_FUNC

PROGRAM PLC_PRG
  VAR
    fbDoB : FB_SetStatus;
  END_VAR
  fbDoB(sDeviceStatus := GVL.g_sDEV_STATUS); //Call with actual parameter

```

Die Variable `sDeviceStatus` ist Teil der Bausteinschnittstelle von `FB_B`. Bei Aufruf von `fbDoB` wird dem String zunächst ein Gerätenamen zugewiesen, dann wird der String manipuliert.

Expression	Type	Value	Prepared value	Address	Comment
fbDoB	FB_SetStatus				Instance
sDeviceStatus	REFERENCE TO STRING(25)	'Device_A Activ Activ Acti'			Formal parameter

Abb. 17-31:

Übergabe einer Bitvariablen

```

VAR_GLOBAL
  xBit0 AT %MX0.1 : BOOL;
  xTemp : BOOL;
END_VAR

FUNCTION_BLOCK FB_DoSomething
  VAR_INPUT
    xIn : BOOL;
  END_VAR
  VAR_IN_OUT
    xInOut : BOOL;
  END_VAR
  IF xIn THEN
    xInOut := TRUE;
  END_IF

PROGRAM PLC_PRG
  VAR
    xIn : BOOL;
    DoSomething_1 : FB_DoSomething;
    DoSomething_2 : FB_DoSomething;
  END_VAR

  // The following line of code causes a compiler error:
  // C0201: Typ 'BIT' is not equal to type 'BOOL' of VAR_IN_OUT 'xInOut'
  DoSomething_1(xIn := xIn, xInOut := xBit0);

  // Workaround

```

```
xTemp := xBit0;
DoSomething_2(xIn := xIn, xInOut := xTemp);
xBit0 := xTemp;
```

Das Programm ruft die Funktionsbaustein-Instanzen DoSomething_1 und DoSomething_2 auf. Wegen der direkten Zuweisung der Bitvariablen xBit0 an den VAR_IN_OUT-Eingang wird beim Aufruf der Instanz DoSomething_1 ein Compilerfehler erzeugt. Der Aufruf der Instanz DoSomething_2 mit Zuweisen einer Zwischenvariablen ist dagegen ein korrekter Code.

Übergabevariable VAR_IN_OUT CONSTANT

Eine VAR_IN_OUT CONSTANT-Variable dient als konstanter Durchgangsparmeter, an den eine Variable aber auch eine Konstante übergeben werden kann. Der Parameter kann gelesen, aber nicht beschrieben werden.

Syntax Deklaration

```
<keyword> <POU name>
VAR_IN_OUT CONSTANT
  <variable name> : <data type>; // formal parameter
END_VAR
<keyword> : FUNCTION | FUNCTION_BLOCK | METHOD | PRG
```

VAR_IN_OUT CONSTANT-Variablen werden deklariert, ohne einen Initialisierungswert zuzuweisen.

Verwendung:

- Bei Aufruf des Programmierbausteins kann eine konstante Variable oder ein Literal übergeben werden. Dabei sind alle Datentypen erlaubt. Folglich sind Schreibzugriffe nicht erlaubt.
- Bei Parameterübergabe einer Stringkonstanten: Die Stringlänge der Konstanten kann beliebig sein und hängt nicht von der Stringlänge der VAR_IN_OUT CONSTANT-Variablen ab.



Wenn in den Projekteinstellungen in Kategorie Compile-Optionen die Option "Konstanten ersetzen" aktiviert ist, dann erzeugt die Parameterübergabe einer Konstanten mit Basisdatentyp oder einer konstanten Variablen mit Basisdatentyp einen Compilerfehler.

Beispiel

Parameterübergabe von Stringkonstanten und Stringvariablen

```
FUNCTION funManipulate : BOOL
VAR_IN_OUT
  sReadWrite : STRING(16); (* Can be read or written here in POU *)
  dwVarReadWrite : DWORD; (* Can be read or written here in POU *)
END_VAR
VAR_IN_OUT CONSTANT
  c_sReadOnly : STRING(16); (* Constant string variable can only be read here in POU *)
END_VAR

sReadWrite := 'String from POU';
dwVarReadWrite := STRING_TO_DWORD(c_sReadOnly);

PROGRAM PRG_A
VAR
  sVarFits : STRING(16);
  sValFits : STRING(16) := '1234567890123456';
  dwVar : DWORD;
END_VAR

// The following line of code causes the compiler error C0417:
// C0417: VAR_IN_OUT parameter 'sReadWrite' needs a variable with write access as input.
funManipulate(sReadWrite:= '1234567890123456', c_sReadOnly:= '1234567890123456', dwVarReadWrite := dwVar);
```



```
// Correct code
funManipulate(sReadWrite := sValFits, c_sReadOnly := '23', dwVarReadWrite := dwVar);
funManipulate(sReadWrite := sVarFits, c_sReadOnly := sValFits, dwVarReadWrite := dwVar);
```

Im Code werden Strings über verschiedene VAR_IN_OUT-Variablen an die Funktion funManipulate übergeben. Bei Übergabe eines Literals an eine VAR_IN_OUT-Variable wird ein Compilerfehler ausgegeben. Bei Übergabe einer konstanten Variablen an eine VAR_IN_OUT CONSTANT-Variable wird korrekter Code erzeugt, ebenso bei der Übergabe von Stringvariablen.

Deklaration des Funktionsbausteins MB_LoadCamData

```
FUNCTION BLOCK MB_LoadCamData EXTENDS IL_ExecuteDoneBase
VAR_INPUT
    Permanent: BOOL:= FALSE; (*Setting, if cam is written permanent or temporary*)
    CamTableID: MC_CAM_ID:= CAM_TABLE_1;(*Selection of camtable from drive or controller*)
END_VAR

VAR_IN_OUT
    CamTable: MB_CAMTABLE; (*Destination of value, contains the values of cam table
                             after finishing without an error and the number of elements*)
END_VAR

VAR_IN_OUT CONSTANT
    Slave: AXIS_REF; (*Selection of reference*)
END_VAR
```

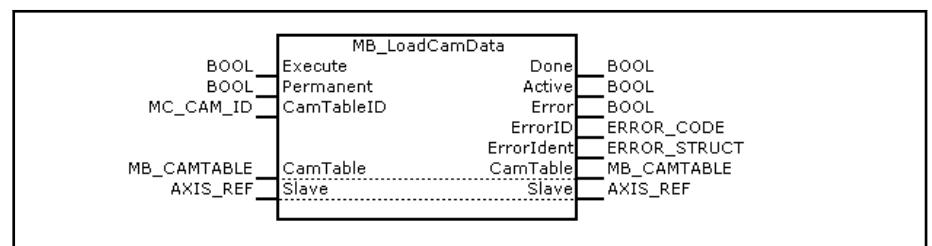


Abb. 17-32: Funktionsbaustein mit VAR_IN_OUT "CamTable" und VAR_IN_OUT CONSTANT "Slave"

Beispiel "CamTable":

- Über den Eingang "CamTable" vom Typ MB_CAMTABLE ist der Speicherbereich, auf den schreibend oder lesend zugegriffen werden soll, direkt zu erreichen.

Beispiel "Slave":

- Über den Eingang "Slave" vom Typ AXIS_REF erhält der Baustein Informationen zur gewünschten Steuerung und der von dieser Steuerung gesteuerten Achse.

Unter Verwendung dieser Informationen wird der gewünschte Speicherbereich adressiert und (schreibend und/oder lesend) auf die entsprechenden Daten zugegriffen.

Siehe auch:

- [Kap. 9.2 "Variablen deklarieren" auf Seite 85](#)
- [Kap. "Dialog 'Projekteinstellungen- 'Compileroptionen'" auf Seite 759](#)
- [Kap. "Objekt 'Funktion\(FUNCTION\)' auf Seite 558](#)
- [Kap. "Objekt 'Funktionsbaustein' \(FUNCTION_BLOCK\)" auf Seite 556](#)
- [Kap. "Objekt 'Methode' " auf Seite 564](#)
- [Kap. "Objekt 'Schnittstelle\(INTERFACE\)' auf Seite 562](#)
- [Kap. "Objekt 'Schnittstellenmethode'" auf Seite 567](#)
- [Kap. 17.2.10 "Konstante Variablen - VAR CONSTANT" auf Seite 308](#)

17.2.5 Globale Variablen - VAR_GLOBAL

Globale Variablen sind "normale" Variablen, Konstanten, externe oder remanente Variablen, die im gesamten Projekt bekannt sind.

Sie deklarieren globale Variablen in globalen Variablenlisten oder im Deklarationsteil von Programmierobjekten zwischen den Schlüsselwörtern `VAR_GLOBAL` und `END_VAR`.

Das System erkennt eine globale Variable, wenn Sie dem Variablennamen ein Punkt voranstellen, zum Beispiel `.iGlobVar1`.



Eine lokal in einem Baustein deklarierte Variable, die denselben Namen hat wie eine globale Variable, hat innerhalb des Bausteins Vorrang.



Ab Compiler-Version 3.2.0.0 initialisiert IndraLogic globale Variablen immer vor den lokalen Variablen von POU's.

Beispiel:

```
VAR_GLOBAL
    iVarGlob1:INT;
END_VAR
```

Siehe auch

- [Kap. 18.1.11 "Objekt 'GVL'- Globale Variablenliste" auf Seite 544](#)
- [Kap. 17.3.67 "Operator - Globaler Namensraum" auf Seite 359](#)

17.2.6 Temporäre Variablen - VAR_TEMP

Diese Funktionalität ist eine Erweiterung bezüglich der Norm IEC 61131-3.

Sie deklarieren temporäre Variablen lokal zwischen den Schlüsselwörtern `VAR_TEMP` und `END_VAR`.

`VAR_TEMP`-Deklarationen sind nur in Programmen und Funktionsbausteinen möglich.

IndraLogic initialisiert temporäre Variablen bei jedem Aufruf des Bausteins neu.

Auf die temporären Variablen kann die Applikation nur im Implementierungsteil eines Programms oder eines Funktionsbausteins zugreifen.

Beispiel:

```
VAR_TEMP
    iVarTmp1:INT; (*1st temporary variable *)
END_VAR
```

17.2.7 Statische Variablen - VAR_STAT

Diese Funktionalität ist eine Erweiterung bezüglich der Norm IEC 61131-3.

Statische Variablen deklarieren Sie lokal zwischen den Schlüsselwörtern `VAR_STAT` und `END_VAR`.

Die statischen Variablen initialisiert IndraLogic beim ersten Aufruf des jeweiligen Bausteins.

Auf statische Variablen können Sie nur innerhalb des Namensraums, in dem die Variablen deklariert sind, zugreifen (wie bei statischen Variablen in C). Jedoch behalten statische Variablen ihren Wert, wenn die Applikation den

Baustein verlassen hat. Sie können statische Variablen beispielsweise als Zähler für Funktionsaufrufe verwenden.

Sie können statische Variablen mit einem Attribut-Schlüsselwort erweitern.

Beispiel:

```
VAR_STAT
    iVarStat1:INT;
END_VAR
```

Siehe auch [Kap. 9.16 "Datenpersistenz" auf Seite 144](#)

17.2.8 Externe Variablen - VAR_EXTERNAL

Externe Variablen sind globale Variablen die in einen Baustein "importiert" werden.

Die Variablen deklarieren Sie zwischen den Schlüsselwörtern VAR_EXTERNAL und END_VAR. Wenn die globale Variable nicht existiert, wird eine Fehlermeldung ausgegeben.



In IndraLogic ist es nicht notwendig, Variablen als extern zu deklarieren. Das Schlüsselwort existiert, um die Kompatibilität zu IEC 61131-3 zu wahren.

Syntax *Programm:*

```
<POU keyword> <POU name>
VAR_EXTERNAL
    <variable name> : <data type>;
END_VAR
```

Eine Initialisierung ist nicht erlaubt.

Beispiel

```
FUNCTION_BLOCK FB_DoSomething
VAR_EXTERNAL
    iVarExt1 : INT; (* 1st external variable *)
END_VAR
```

Siehe auch:

- [Kap. 18.1.11 "Objekt 'GVL'- Globale Variablenliste" auf Seite 544](#)

17.2.9 Instanzvariablen - VAR_INST

IndraLogic legt eine VAR_INST-Variable einer Methode nicht auf dem Methoden-Stack, sondern auf dem Stack der Funktionsbaustein-Instanz ab.

Dies bedeutet, dass sich die VAR_INST-Variable wie andere Variablen der Funktionsbaustein-Instanz verhält und nicht bei jedem Aufruf der Methode neu initialisiert wird.

VAR_INST-Variablen sind nur in Methoden erlaubt und Sie können nur innerhalb der Methode auf eine solche Variable zugreifen.

Die Variablenwerte von Instanzvariablen monitoren Sie im Deklarationsteil der Methode.

Sie können Instanzvariablen nicht mit einem Attribut-Schlüsselwort erweitern.

Beispiel:

```
METHOD meth_last : INT
VAR_INPUT
    iVar : INT;
END_VAR
VAR_INST
```

```

    iLast : INT := 0;
END_VAR
meth_last := iLast;
iLast := iVar;

```

Siehe auch:

- [Kap. "Objekt 'Methode' " auf Seite 564](#)

17.2.10 Konstante Variablen - VAR CONSTANT

Die Deklaration konstanter Variablen erfolgt in den globalen Variablenlisten oder im Deklarationsteil von Programmierobjekten. In Implementierungen kann auf konstante Variablen über den Instanzpfad lesend zugegriffen werden, aber nicht schreibend.

Syntax *Programm:*

```

<scope> CONSTANT
  <identifier> : <data type> :=
  <initialization> ;
END_VAR

<scope> : VAR | VAR_INPUT | VAR_STAT | VAR_GLOBAL

```



Weisen Sie bei der Deklaration einer konstanten Variable immer einen Initialisierungswert zu. Danach kann die Konstante nicht mehr beschrieben werden.

Beispiel *Deklaration*

```

VAR CONSTANT
  c_rTAXFACTOR : REAL := 1.19;
END_VAR

```

Aufruf

```

rPrice := rValue * c_rTAXFACTOR;

```

Sie können in Implementierungen auf konstante Variablen nur lesend zugreifen.

Konstante Variablen stehen rechts vom Zuweisungsoperator.

Siehe auch [Kap. 17.2.4 "Ein- und Ausgabevariable - VAR_IN_OUT" auf Seite 301](#)

17.2.11 Persistente Variable - PERSISTENT

Persistente Variablen werden in der persistenten globalen Variablenliste im Deklarationsabschnitt VAR_GLOBAL RETAIN PERSISTENT deklariert. Für Variablen, die außerhalb des Persistenzeditors mit dem Schlüsselwort PERSISTENT gekennzeichnet sind, werden dort Instanzpfade hinzugefügt.

Syntax der Deklaration in der globalen persistenten Variablenliste
PersistentVars

```

VAR_GLOBAL PERSISTENT RETAIN
  <identifier>: <data type> (: <initialization>)?;
  <instance path to POU variable>
END_VAR

```

Syntax der Deklaration in POUs

```

<scope> PERSISTENT RETAIN
  <identifier>: <data type> ( := <initialization> )?; // (...)? : Optional

```

```
END_VAR
<scope> : VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | VAR_STAT | VAR_GLOBAL
```



Eine Zuweisung von Ein-, Ausgabe oder Speicheradressen mit dem Schlüsselwort **AT** ist nicht erlaubt.

Verwenden Sie in persistenten Variablenlisten nie den Datentyp **POINTER TO**, denn bei erneutem Download der Applikation können sich die Adressen verändern. Im Meldungsfenster werden entsprechende Compilerwarnungen ausgegeben.

Wenn Sie die Namen, oder die Datentypen einer remanenten Variablen häufig ändern, ist es empfehlenswert die remanenten Variablen als Retain-Variablen nur mit dem Schlüsselwort **RETAIN** zu deklarieren.

Deklarieren Sie Variablen direkt in der Liste der persistente Variablen und verzichten Sie auf das Einfügen von Instanzpfaden, denn durch die Verwendung von Instanzpfaden wird doppelt so viel Speicherplatz belegt und die Zykluszeit erhöht!

Beispiel *Deklaration in der persistenten Variablenliste PersistentVars:*

```
{attribute 'qualified_only'}
VAR_GLOBAL PERSISTENT RETAIN
  g_iCounter : INT;
  // Generated instance path of persistent variable
  PLC_PRG.fb_A.iPersistentCounter_A: INT;
END_VAR
```

Deklaration im Funktionsbaustein FB_A:

```
FUNCTION_BLOCK FB_A
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR_PERSISTENT
  iPersistentCounter_A : INT;
END_VAR
```

Deklaration im Programm PLC_PRG:

```
VAR
  fb_A1 : FB_A;
END_VAR
```

Mögliche Deklarationsstellen

Deklarationsstelle:	Auswirkung:
Direkt in der persistenten globalen Variablenliste	Die Variable ist persistent und liegt im geschützten Speicherbereich.
Lokal in einem Programm mit Instanzpfad in der persistenten Variablenliste	Die Variable ist persistent und liegt im geschützten Speicherbereich und im Speicher (Doppelbelegung).
Lokal in einem Funktionsbaustein mit Instanzpfad in der persistenten Variablenliste	

Deklarationsstelle:	Auswirkung:
Nur lokal in einem Programm	Variable ist nicht persistent. Im Meldungsfenster wird eine Warnung ausgegeben. Tipp: Wählen Sie den Befehl "Deklarationen" → "alle Instanzpfade hinzufügen", um die Variablen in die persistente Variablenliste aufzunehmen.
Nur lokal in einem Funktionsbaustein	
Lokal in einer Funktion	Diese Deklaration hat keine Auswirkung. Die Variable ist nicht persistent.

Tab. 17-10:

Wählen Sie im Persistenzeditor den Befehl "Alle Instanzpfade hinzufügen", wenn lokale Variablen mit `PERSISTENT` gekennzeichnet sind.



Vermeiden Sie, Variablen, die lokal in einem Funktionsbaustein deklariert sind, mit `PERSISTENT` zu kennzeichnen. Dabei wird nämlich die Funktionsbausteininstanz vollständig im remanenten Speicher abgelegt und nicht nur die gekennzeichnete Variable.

Siehe auch:

- [Kap. 9.16 "Datenpersistenz" auf Seite 144](#)
- [Kap. 9.16.4 "VAR PERSISTENT-Variablen deklarieren" auf Seite 151](#)
- [Kap. "Befehl 'Alle Instanzpfade hinzufügen'" auf Seite 722](#)
- [Kap. 9.16.1 "Daten mit persistenten Variablen erhalten" auf Seite 148](#)

17.2.12 Retain-Variable - RETAIN

Retain-Variablen werden deklariert, indem in Programmierobjekten im Gültigkeitsbereich `VAR`, `VAR_INPUT`, `VAR_OUTPUT`, `VAR_IN_OUT`, `VAR_STAT` oder `VAR_GLOBAL` das Schlüsselwort `RETAIN` hinzugefügt wird.

Syntax bei der Deklaration

```
<scope> RETAIN
  <identifier>: <data type> ( := <initialization> )? // (...)?: Optional
END_VAR
<scope> : VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | VAR_STAT | VAR_GLOBAL
```



Eine Zuweisung von Ein-, Ausgabe oder Speicheradressen mit dem Schlüsselwort `AT` ist nicht erlaubt.

Beispiel *In einer POU:*

```
VAR RETAIN
  iVarRetain: INT;
END_VAR
```

In einer GVL:

```
VAR_GLOBAL RETAIN
  g_iVarRetain: INT;
END_VAR
```

Mögliche Deklarationsstellen

Deklarationsstelle:	Auswirkung:
Lokal in einem Programm	Nur die Variable liegt im Retain-Speicherbereich.
Global in einer globalen Variablenliste	Nur die Variable liegt im Retain-Speicherbereich.
Lokal in einem Funktionsbaustein	Die komplette Instanz des Funktionsbausteins mit all seinen Daten liegt im Retain-Speicherbereich. Nur die deklarierte Retain-Variable wird geschützt.
Lokal in einer Funktion	Selbst die Variable liegt nicht im Retain-Speicherbereich. Diese Deklaration hat keine Auswirkung.
Lokal und persistent in einer Funktion	Selbst die Variable liegt nicht im Retain-Speicherbereich. Diese Deklaration hat keine Auswirkung.

Tab. 17-11:



Vermeiden Sie, Variablen eines Funktionsbausteins mit `RETAIN` zu kennzeichnen.

Siehe auch:

- [Kap. 9.16 "Datenpersistenz" auf Seite 144](#)
- [Kap. 9.16.4 "VAR PERSISTENT-Variablen deklarieren" auf Seite 151](#)
- [Kap. "Befehl 'Alle Instanzpfade hinzufügen'" auf Seite 722](#)
- [Kap. 9.16.1 "Daten mit persistenten Variablen erhalten" auf Seite 148](#)

17.2.13 Konfigurationsvariablen - VAR_CONFIG

Konfigurationsvariablen dienen dazu, Variablen, die in Funktionsbausteinen mit unvollständigen Adressangaben deklariert sind und auf Geräte-E/As abgebildet werden sollen, die vollständige Adresse zuzuweisen.

Sie deklarieren die Variablen in einer globalen Variablenliste zwischen den Schlüsselwörtern `VAR_CONFIG` und `END_VAR`. Diese globale Variablenliste wird "Variablenkonfiguration" genannt.

In der "Variablenkonfiguration" müssen Sie die Konfigurationsvariablen mit vollständigem Instanzpfad und mit der korrekten Adresse angeben.

Beispiel

Deklaration der Variablen `xLocIn` mit unvollständiger Adresse `%I*` in einem Funktionsbaustein:

```
FUNCTION_BLOCK locio
VAR
  xLocIn AT %I*: BOOL := TRUE;
END_VAR
```

Der Funktionsbaustein `locio` wird im Programm `PlcProg` verwendet:

```
PROGRAM PlcProg
VAR
  locioVar1: locio;
END_VAR
```

Die korrekte Variablenkonfiguration in der globalen Variablenliste lautet:

```
VAR_CONFIG
  P1cProg.locioVar1.xLocIn AT %IX1.0 : BOOL;
END_VAR
```

Siehe auch

- [Kap. 9.10.1 "Variablenkonfiguration - VAR_CONFIG" auf Seite 128](#)

17.2.14 SUPER

SUPER ist eine spezielle Variable und dient der objektorientierten Programmierung.

SUPER ist der Zeiger eines Funktionsbausteins auf die Basis-Funktionsbaustein-Instanz, aus die der Funktionsbaustein erzeugt wurde. Der **SUPER**-Zeiger erlaubt somit auch den Zugriff auf die Implementierung der Methoden des Basis-Funktionsbausteins (Basisklasse). Für jeden Funktionsbaustein steht automatisch ein **SUPER**-Zeiger zur Verfügung.

Sie können **SUPER** nur in Methoden- und in den zugehörigen Funktionsbaustein-Implementierungen verwenden.

Dereferenzierung des Zeigers: **SUPER^**

Verwendung des **SUPER**-Zeigers: Mit Hilfe des Schlüsselworts **SUPER** rufen Sie eine Methode auf, die in der Instanz der Basisklasse oder Elternklasse gültig ist.

Beispiel:

Beispiele

ST:

```
SUPER^.METH_DoIt();
```

FUP/CFC/LD

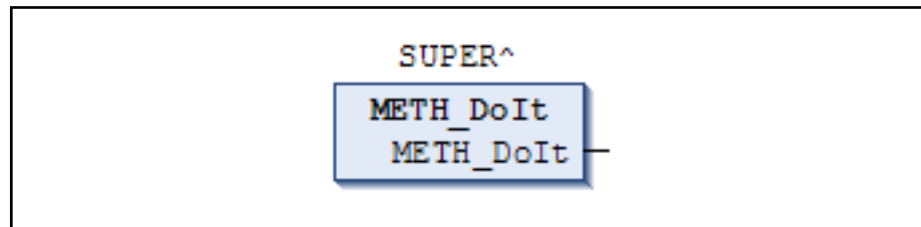


Abb. 17-33:



SUPER ist für die Anweisungsliste (AWL) noch nicht implementiert.

Beispiele:

*Verwendung von **SUPER**- und **THIS**-Zeiger*

```
FUNCTION_BLOCK FB_Base
VAR_INPUT
END_VAR
VAR_OUTPUT
  iCnt : INT;
  iRes : INT;
END_VAR
VAR
END_VAR
END_VAR

THIS^.METH_DoIt();
THIS^.METH_DoAlso();

METHOD METH_DoIt : BOOL
```



```

VAR
END_VAR
iCnt := -1;
METH_DoIt := TRUE;

METHOD METH_DoAlso : BOOL
VAR
END_VAR
iRes := -5;
METH_DoAlso := TRUE;

FUNCTION_BLOCK FB_1 EXTENDS FB_Base
VAR_INPUT
END_VAR
VAR_OUTPUT
    iCnt_Base: INT;
    iCnt_THIS: INT;
    iRes_Base: INT;
    iRes_THIS: INT;
END_VAR
VAR
END_VAR
// Calls the method defined under FB_1
THIS^.METH_DoIt();
THIS^.METH_DoAlso();
iCnt_THIS:= iCnt;
iRes_THIS:= iRes;
// Calls the method defined under FB_Base
SUPER^.METH_DoIt();
SUPER^.METH_DoAlso();
iCnt_Base:= iCnt;
iRes_Base:= iRes;

METHOD METH_DoIt : BOOL
VAR
END_VAR
    iCnt := 1111;
    METH_DoIt := TRUE;

PROGRAM PlcProg
VAR
    Myfb_1: FB_1;
    iFB: INT;
    iBase: INT;
END_VAR
Myfb_1();
iBase := Myfb_1.iCnt_Base;
iFB := Myfb_1.iCnt_THIS;

```

Siehe auch

- [Kap. 17.5 "Datentypen" auf Seite 374](#)
- [Kap. 17.2.15 "THIS" auf Seite 313](#)

17.2.15 THIS

THIS ist eine spezielle Variable und dient der objektorientierten Programmierung.

THIS ist der Zeiger eines Funktionsbausteins auf seine eigene Funktionsbaustein-Instanz. Für jeden Funktionsbaustein steht automatisch ein THIS-Zeiger zur Verfügung.

Sie können THIS nur in Methoden und in Funktionsbausteinen verwenden. THIS steht für die Implementierung in der Eingabehilfe in der Kategorie **Schlüsselwörter** zur Verfügung.

Dereferenzierung des Zeigers: THIS^

Verwendung des THIS-Zeigers

- Wenn in einer Methode eine lokale Variable eine Funktionsbaustein-Variable verschattet, können Sie die Funktionsbaustein-Variable mit dem THIS- Zeiger setzen. Siehe unten Beispiel (1)

- Wenn der Zeiger auf die eigene Funktionsbaustein-Instanz zur Verwendung in einer Funktion referenziert wird. (Siehe unten Beispiel (2))

Beispiel:

Beispiele

ST:

```
THIS^.METH_DoIt();
```

FUP/CFC/LD:

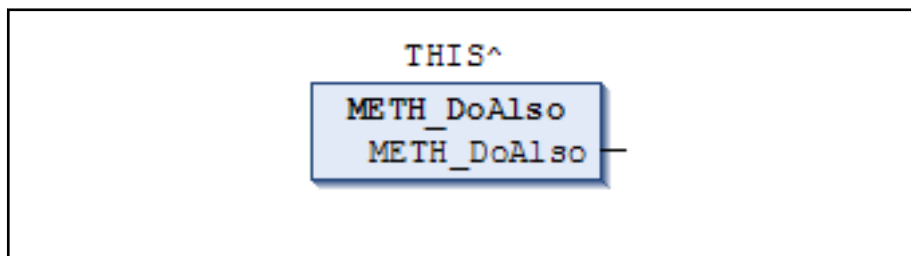


Abb. 17-34:



THIS ist für die Anweisungsliste (AWL) noch nicht implementiert.

Beispiele

Die lokale Variable iVarB verschattet die Funktionsbaustein-Variable iVarB:

```
FUNCTION_BLOCK fbA
VAR_INPUT
    iVarA: INT;
END_VAR
iVarA := 1;

FUNCTION_BLOCK fbB EXTENDS fbA
VAR_INPUT
    iVarB: INT := 0;
END_VAR
iVarA := 11;
iVarB := 2;

    METHOD DoIt : BOOL
    VAR_INPUT
    END_VAR
    VAR
        iVarB: INT;
    END_VAR
    iVarB := 22; // Here the local iVarB is set.
    THIS^.iVarB := 222; // Here the function block variable iVarB is set,
                        // although iVarB is overloaded.

PROGRAM PlcProg
VAR
    MyfbB: fbB;
END_VAR

MyfbB(iVarA:=0 , iVarB:= 0);
MyfbB.DoIt();
```

Ein Funktionsaufruf benötigt die Referenz auf die eigene Instanz:

```
FUNCTION funA : BOOL
VAR_INPUT
    pFB: fbA;
END_VAR
...;

FUNCTION_BLOCK fbA
VAR_INPUT
    iVarA: INT;
END_VAR
```

```

...;

FUNCTION BLOCK fbB EXTENDS fbA
VAR_INPUT
    iVarB: INT := 0;
END_VAR
iVarA := 11;
iVarB := 2;

    METHOD DoIt : BOOL
    VAR_INPUT
    END_VAR
    VAR
        iVarB: INT;
    END_VAR
    iVarB := 22;           // Here the local iVarB is set.
    funA(pFB := THIS^); // Here funA is called with THIS^

PROGRAM PlcProg
VAR
    MyfbB: fbB;
END_VAR
MyfbB(iVarA:=0 , iVarB:= 0);
MyfbB.DoIt();

```

Siehe auch

- [Kap. 17.5.12 "Datentyp 'POINTER'" auf Seite 383](#)
- [Kap. 17.2.14 "SUPER" auf Seite 312](#)

17.3 Operatoren

IndraLogic 2G unterstützt alle Operatoren der Norm IEC-61131-3.

Diese Operatoren sind implizit überall im Projekt bekannt.

Neben den IEC-Operatoren unterstützt IndraLogic auch einige Operatoren, die nicht in der Norm IEC 61131-3 beschrieben sind.

Operatoren werden in einem Baustein wie Funktionen benutzt.



Informationen zur Abarbeitungsreihenfolge (Bindungsstärke) der ST-Operatoren finden Sie unter [Kap. "ST-Ausdrücke" auf Seite 242](#).

HINWEIS

Operationen mit Gleitkomma-Datentypen

Bei Operationen mit Gleitkomma-Datentypen ist das Rechenergebnis abhängig von der verwendeten Zielsystem-Hardware.

HINWEIS

Bei Operationen mit Überlauf oder Unterlauf

Bei Operationen mit Überlauf oder Unterlauf im Datentyp ist das Rechenergebnis abhängig von der verwendeten Zielsystemhardware.

Überlauf/Unterlauf im Datentyp

Der IndraLogic-Compiler generiert Code für den jeweiligen Steuerungstyp und berechnet dabei Zwischenergebnisse immer mit der nativen Größe, die durch die Steuerung vorgegeben ist. Dies bietet deutliche Vorteile bei der Rechengeschwindigkeit und produziert häufig auch das erwartete Ergebnis. Dies bedeutet aber auch, dass ein Überlauf oder ein Unterlauf im Datentyp unter Umständen nicht abgeschnitten wird.

Beispiel1 Das Ergebnis dieser Addition wird nicht abgeschnitten und das Ergebnis in dwVar ist 65536.

```

VAR
  wVar : WORD;
  dwVar: DWORD;
END_VAR

wVar := 65535;
dwVar := wVar + 1;

```

Beispiel2 Der Überlauf und Unterlauf im Datentyp wird nicht abgeschnitten und die Ergebnisse (bVar1, bVar2) beider Vergleiche sind FALSE.

```

VAR
  wVar1 : WORD;
  wVar2 : WORD;
  bVar1 : BOOL;
  bVar2 : BOOL;
END_VAR

wVar1 := 65535;
wVar2 := 0;
bVar1 := (wVar1 + 1) = wVar2;
bVar2 := (wVar2 - 1) = wVar1;

```

Beispiel3 Durch die Zuweisung auf wVar3 wird der Wert auf den Zieldatentyp WORD abgeschnitten und das Ergebnis bvar1 ist TRUE.

```

VAR
  wVar1 : WORD;
  wVar2 : WORD;
  wVar3 : WORD;
  bVar1 : BOOL;
END_VAR

wVar1 := 65535;
wVar2 := 0;
wVar3 := (wVar1 + 1);
bVar1 := wVar3 = wVar2;

```

Beispiel4 Um den Compiler zu einem Abschneiden des Zwischenergebnisses zu zwingen, kann eine Konvertierung eingefügt werden.

Durch die Typkonvertierung wird sichergestellt, dass beide Vergleiche nur 16 Bit vergleichen und die Ergebnisse (bVar1, bVar2) beider Vergleiche jeweils TRUE sind.

```

VAR
  wVar1 : WORD;
  wVar2 : WORD;
  bVar1 : BOOL;
  bVar2 : BOOL;
END_VAR

wVar1 := 65535;
wVar2 := 0;
bVar1 := TO_WORD(wVar1 + 1) = wVar2;
bVar2 := TO_WORD(wVar2 - 1) = wVar1;

```

Arithmetische Operatoren [Kap. 17.3.1 "Operator 'ADD'" auf Seite 319](#)

[Kap. 17.3.3 "Operator 'SUB'" auf Seite 320](#)

	Kap. 17.3.2 "Operator 'MUL'" auf Seite 319
	Kap. 17.3.4 "Operator 'DIV'" auf Seite 320
	Kap. 17.3.5 "Operator 'MOD'" auf Seite 321
	Kap. 17.3.6 "Operator 'MOVE'" auf Seite 322
	Kap. 17.3.7 "Operator 'INDEXOF'" auf Seite 322
	Kap. 17.3.8 "Operator 'SIZEOF'" auf Seite 322
Bitstring-Operatoren	Kap. 17.3.10 "Operator 'AND'" auf Seite 323
	Kap. 17.3.11 "Operator 'OR'" auf Seite 323
	Kap. 17.3.12 "Operator 'XOR'" auf Seite 324
	Kap. 17.3.9 "Operator 'NOT'" auf Seite 323
	Kap. 17.3.13 "Operator 'AND_THEN'" auf Seite 324
	Kap. 17.3.14 "Operator 'OR_ELSE'" auf Seite 325
Bitshift-Operatoren	Kap. 17.3.15 "Operator 'SHL'" auf Seite 325
	Kap. 17.3.16 "Operator 'SHR'" auf Seite 326
	Kap. 17.3.17 "Operator 'ROL'" auf Seite 326
	Kap. 17.3.18 "Operator 'ROR'" auf Seite 327
Auswahloperatoren	Kap. 17.3.19 "Operator 'SEL'" auf Seite 328
	Kap. 17.3.20 "Operator 'MAX'" auf Seite 329
	Kap. 17.3.21 "Operator 'MIN'" auf Seite 329
	Kap. 17.3.22 "Operator 'LIMIT'" auf Seite 329
	Kap. 17.3.23 "Operator 'MUX'" auf Seite 330
Vergleichsoperatoren	Die Vergleichsoperatoren sind boolesche Operatoren, die jeweils zwei Eingänge (erster und zweiter Operand) miteinander vergleichen.
	Kap. 17.3.24 "Operator 'GT'" auf Seite 330
	Kap. 17.3.25 "Operator 'LT'" auf Seite 331
	Kap. 17.3.26 "Operator 'LE'" auf Seite 331
	Kap. 17.3.27 "Operator 'GE'" auf Seite 331
	Kap. 17.3.28 "Operator 'EQ'" auf Seite 332
	Kap. 17.3.29 "Operator 'NE'" auf Seite 332
Adressoperatoren	Kap. 17.3.30 "Operator 'ADR'" auf Seite 333
	Kap. 17.3.31 "Operator 'Inhaltsoperator'" auf Seite 333
	Kap. 17.3.32 "Operator 'BITADR'" auf Seite 334
Aufrufoperatoren	Kap. 17.3.33 "Operator 'CAL'" auf Seite 334
Typkonvertierungsoperatoren	Eine implizite Konvertierung von einem "größeren" Typ auf einen "kleineren" Typ ist nicht möglich, wie beispielsweise von INT nach BYTE oder von DINT nach WORD.
	Wenn Sie implizite Typkonvertierungen von einem "größeren" Typ auf einen "kleineren" vornehmen wollen, müssen Sie spezielle Typkonvertierungen anwenden. Grundsätzlich können Sie jeden elementaren Typ zu jedem anderen elementaren Typ konvertieren.
	Getyppte Konvertierung: <elementarer Typ1>_TO_<elementarer Typ2>
	Überladene Konvertierung: TO_<elementarer Typ2>



IndraLogic generiert bei ...TO_STRING-Konvertierungen die Zeichenfolge (String) linksbündig. Wenn die Stringlänge zu kurz definiert ist, wird von rechts her abgeschnitten.



Wenn der Eingangswert eines Typkonvertierungsoperators außerhalb des Wertebereichs des Ausgangsdatentyps liegt, ist das Ergebnis der Operation nicht definiert und abhängig vom Steuerungstyp. Dies ist zum Beispiel der Fall, wenn der Eingangswert für die Konvertierung von LREAL nach DINT negativ ist.

[Kap. 17.3.34 "Operator 'BOOL_TO'" auf Seite 335](#)

Siehe auch

[Kap. 17.3.35 "Operator 'TO_BOOL'" auf Seite 336](#)

[Kap. 17.3.37 "Operator '<INT-Typ>_TO_<INT-Typ>'" auf Seite 337](#)

[Kap. 17.3.38 "Operator 'REAL_TO-/ LREAL-TO'" auf Seite 338](#)

[Kap. 17.3.39 "Operator 'TIME_TO/TIME_OF_DAY_TO'" auf Seite 339](#)

[Kap. 17.3.40 "Operator 'DATE_TO/DT_TO'" auf Seite 340](#)

[Kap. 17.3.41 "Operator 'STRING_TO'" auf Seite 341](#)

[Kap. 17.3.42 "Operator 'TRUNC'" auf Seite 342](#)

[Kap. 17.3.43 "Operator 'TRUNC_INT'" auf Seite 342](#)

[Kap. 17.3.36 "Operator 'TO_<xxx>'" auf Seite 337](#)

Numerische Operatoren

[Kap. 17.3.44 "Operator 'ABS'" auf Seite 343](#)

[Kap. 17.3.45 "Operator 'SQRT'" auf Seite 343](#)

[Kap. 17.3.46 "Operator 'LN'" auf Seite 343](#)

[Kap. 17.3.47 "Operator 'LOG'" auf Seite 344](#)

[Kap. 17.3.48 "Operator 'EXP'" auf Seite 344](#)

[Kap. 17.3.49 "Operator 'EXPT'" auf Seite 344](#)

[Kap. 17.3.50 "Operator 'SIN'" auf Seite 345](#)

[Kap. 17.3.53 "Operator 'ASIN'" auf Seite 346](#)

[Kap. 17.3.51 "Operator 'COS'" auf Seite 346](#)

[Kap. 17.3.52 "Operator 'TAN'" auf Seite 346](#)

[Kap. 17.3.54 "Operator 'ACOS'" auf Seite 347](#)

[Kap. 17.3.55 "Operator 'ATAN'" auf Seite 347](#)

Namensraum-Operatoren

Die Namensraum-Operatoren sind eine Erweiterung der IEC 61131-3 Operatoren. Sie bieten Ihnen Möglichkeiten, den Zugriff auf Variablen oder Module eindeutig zu gestalten, auch wenn Sie den gleichen Variablen- oder Modulnamen im Projekt mehrmals verwenden.

[Kap. 17.3.67 "Operator - Globaler Namensraum" auf Seite 359](#)

[Kap. 17.3.68 "Operator - Namensraum für Globale Variablenlisten" auf Seite 360](#)

[Kap. 17.3.70 "Operator - Enumerations-Namensraum" auf Seite 361](#)

[Kap. 17.3.69 "Operator - Bibliotheks-Namensraum" auf Seite 360](#)

Weitere Operatoren

[Kap. 17.3.58 "Operator '__NEW'" auf Seite 349](#)

[Kap. 17.3.56 "Operator '__DELETE'" auf Seite 347](#)

[Kap. 17.3.57 "Operator '__ISVALIDREF'" auf Seite 349](#)

Kap. 17.3.59 "Operator '__QUERYINTERFACE'" auf Seite 351

Kap. 17.3.60 "Operator '__QUERYPOINTER'" auf Seite 352

Kap. 17.3.71 "Operator 'INI'" auf Seite 361

Kap. 17.3.61 "Operator '__TRY', '__CATCH', '__FINALLY', '__ENDTRY'" auf Seite 353

17.3.1 Operator 'ADD'

Der IEC-Operator dient der Addition von Variablen.

Erlaubte Datentypen: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, TIME_OF_DAY (TOD), DATE, DATE_AND_TIME (DT)

Mögliche Kombinationen für TIME-Datentypen: TIME+TIME = TIME, TOD+TIME = TOD, DT+TIME = DT

Besonderheit im FUP/KOP-Editor: Sie können den ADD-Operator um Bausteineingänge erweitern. Die Anzahl der zusätzlichen Bausteineingänge ist begrenzt.

Beispiele

ST:

```
var1 := 7+2+4+7;
```

FUP:

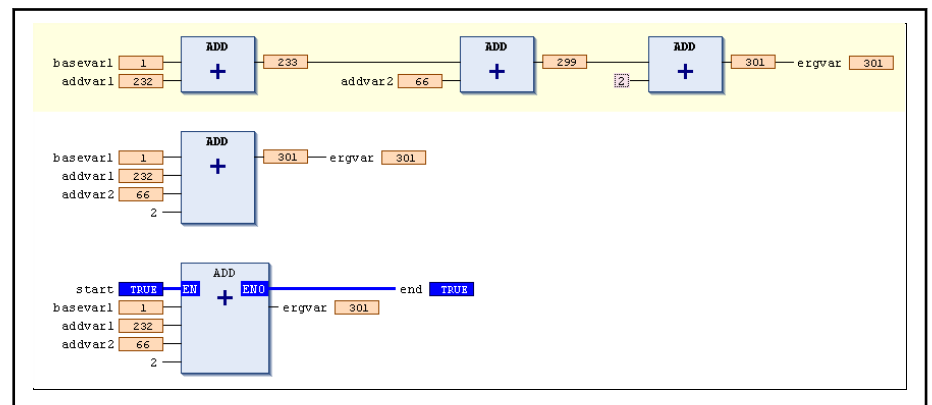


Abb. 17-35:

17.3.2 Operator 'MUL'

Der IEC-Operator dient der Multiplikation von Variablen.

Erlaubte Datentypen: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME

Besonderheit im FUP/KOP-Editor: Sie können den MUL-Operator um zusätzliche Bausteineingänge erweitern. Die Anzahl der zusätzlichen Bausteineingänge ist begrenzt.

Beispiele

ST:

```
var1 := 7*2*4*7;
```

FUP:

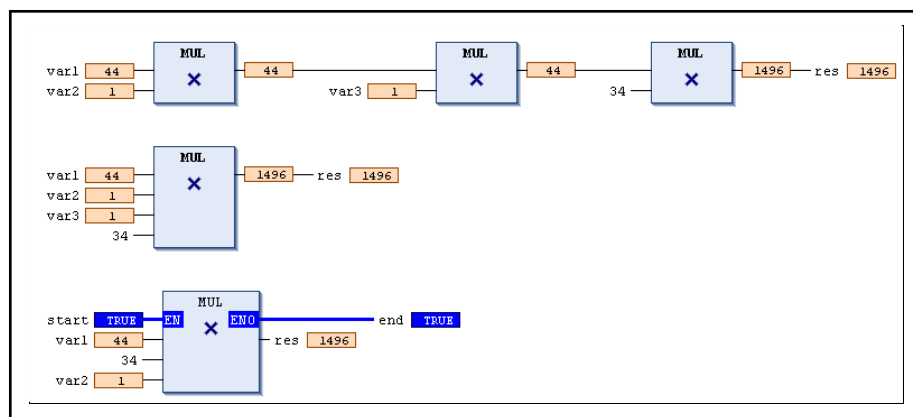


Abb. 17-36:

17.3.3 Operator 'SUB'

Der IEC-Operator dient der Subtraktion von Variablen.

Erlaubte Datentypen: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, TIME_OF_DAY (TOD), DATE, DATE_AND_TIME (DT)

Mögliche Kombinationen für TIME-Datentypen: TIME-TIME = TIME, DATE-DATE = TIME, TOD-TIME = TOD, TOD-TOD = TIME, DT-TIME = DT, DT-DT = TIME



Negative TIME-Werte sind undefiniert.

Beispiele

ST:

```
var1 := 7-2;
```

FUP:

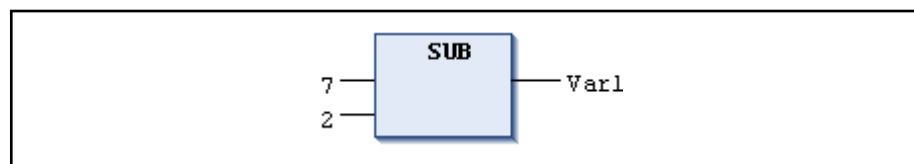


Abb. 17-37:

17.3.4 Operator 'DIV'

Der IEC-Operator dient der Division von Variablen.

Erlaubte Datentypen: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME



Das Verhalten bei einer Division durch Null kann zielsystemabhängig unterschiedlich sein!

Beispiele

ST:

```
var1 := 8/2;
```

FUP:

1. Reihe von DIV-Bausteinen, 2. Einzelner DIV-Baustein, 3. DIV-Bausteine mit EN/ENO-Parametern

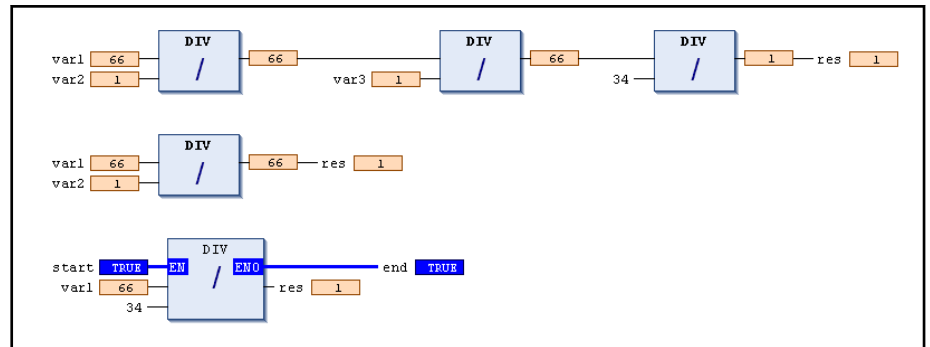


Abb. 17-38:



Beachten Sie die Möglichkeit, eine Division durch 0 während der Laufzeit durch die impliziten Überwachungsfunktionen `CheckDivInt`, `CheckDivLint`, `CheckDivReal` und `CheckDivLReal` zu überwachen.

Siehe auch

- [Kap. 18.1.18 "Überwachung des SPS-Programms der Applikation während der Laufzeit" auf Seite 576](#)
- ["CheckDivInt" auf Seite 578](#)
- ["CheckDivLint" auf Seite 578](#)
- ["CheckDivReal" auf Seite 579](#)
- ["CheckDivLReal" auf Seite 578](#)

17.3.5 Operator 'MOD'

Der IEC-Operator dient der Modulo-Division.

Das Ergebnis der Funktion ist der ganzzahlige Rest der Division.

Erlaubte Datentypen: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT



Das Verhalten bei einer Division durch Null kann zielsystemabhängig unterschiedlich sein!

Beispiele

Ergebnis in Var1 ist 1

ST:

```
var1 := 9 MOD 2;
```

FUP:

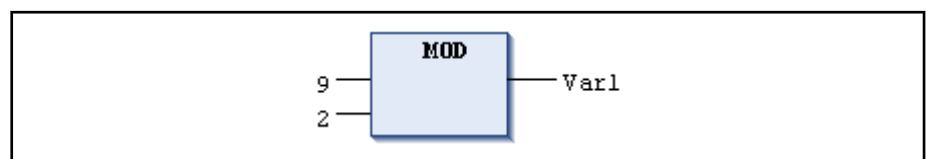


Abb. 17-39:

17.3.6 Operator 'MOVE'

Der IEC-Operator dient der Zuweisung einer Variablen auf eine andere Variable eines entsprechenden Typs.

Da `MOVE` in den CFC-, FUP- und KOP-Editoren als Baustein verfügbar ist, können Sie dort die EN/ENO-Funktionalität auch auf eine Variablenzuweisung anwenden.

Beispiel:

CFC: in Verbindung mit der EN/ENO Funktion:

Nur wenn `en_i` `TRUE` ist, weist IndraLogic den Wert der Variablen `var1` Variable `var2` zu.

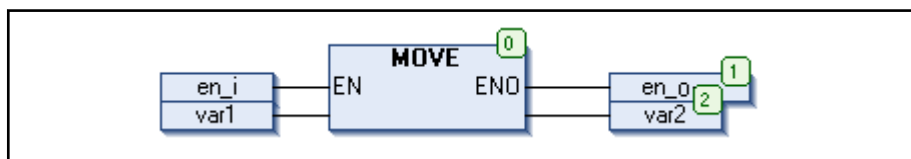


Abb. 17-40:

ST:

```
ivar2 := MOVE(ivar1);
```

dies entspricht:

```
ivar2 := ivar1;
```

17.3.7 Operator 'INDEXOF'

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

Anstelle des Operators `INDEXOF` steht in IndraLogic 2G der `ADR`-Operator zur Verfügung, um einen Zeiger auf den Index eines Bausteins zu erhalten.

Siehe auch

- [Kap. 17.3.30 "Operator 'ADR'" auf Seite 333](#)

17.3.8 Operator 'SIZEOF'

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

Der Operator dient dazu, die Anzahl Bytes zu bestimmen, die die angegebene Variable `x` benötigt. Der `SIZEOF`-Operator liefert immer einen vorzeichenlosen Wert. Der Typ der Rückgabewerte passt sich der gefundenen Größe von Variable `x` an.

Rückgabewert von <code>SIZEOF(x)</code>	Datentyp der Konstanten, die IndraLogic implizit für die gefundene Größe verwendet.
$0 \leq \text{Größe von } x < 256$	USINT
$256 \leq \text{Größe von } x < 65536$	UINT
$65536 \leq \text{Größe von } x < 4294967296$	UDINT
$4294967296 \leq \text{Größe von } x$	ULINT

Beispiele

Ergebnis in `var1` ist 10.

Deklaration

```
arr1 : ARRAY[0..4] OF INT;
var1 : INT;
```

ST:

```
var1 := SIZEOF(arr1); (* var1:=USINT#10; *)
```

17.3.9 Operator 'NOT'

Der IEC-Operator dient dem bitweisen NOT eines Bit-Operanden.

Wenn das entsprechende Eingangsbit 0 ist, ist das Ausgangsbit 1, und umgekehrt.

Erlaubte Datentypen: BOOL, BYTE, WORD, DWORD, LWORD

Beispiele

Ergebnis in Var1 ist 2#0110_1100

ST:

```
Var1 := NOT 2#1001_0011
```

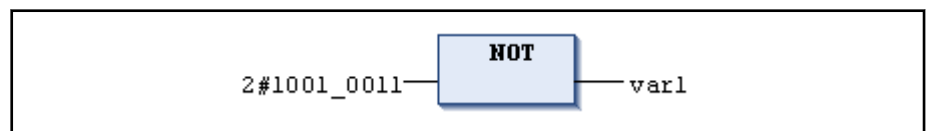
FUP:

Abb. 17-41:

17.3.10 Operator 'AND'

Der IEC-Operator dient dem bitweisen AND von Bit-Operanden.

Wenn die Eingangs-Bits jeweils 1 sind, ist das Ausgangs-Bit 1, ansonsten 0.

Erlaubte Datentypen: BOOL, BYTE, WORD, DWORD, LWORD

Beispiele

Ergebnis in Var1: 2#1000_0010

ST:

```
var1 := 2#1001_0011 AND 2#1000_1010
```

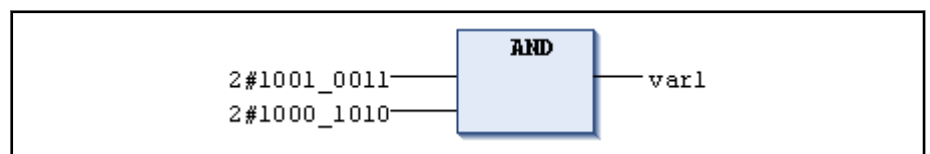
FUP:

Abb. 17-42:

17.3.11 Operator 'OR'

Der IEC-Operator dient dem bitweisen OR von Bit-Operanden.

Wenn mindestens eines der Eingangs-Bits 1 ist, ist das Ausgangs-Bit 1, ansonsten 0.

Erlaubte Datentypen: BOOL, BYTE, WORD, DWORD, LWORD

Beispiele

Ergebnis in Var1 ist 2#1001_1011

ST:

```
Var1 := 2#1001_0011 OR 2#1000_1010
```

FUP:

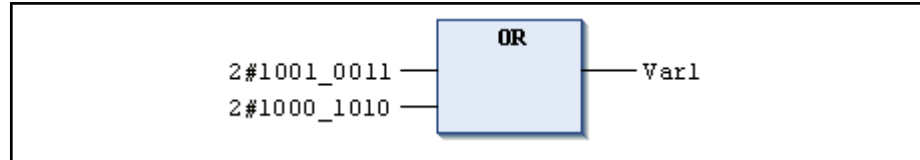


Abb. 17-43:

17.3.12 Operator 'XOR'

Der IEC-Operator dient der bitweisen 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 Datentypen: BOOL, BYTE, WORD, DWORD, LWORD



Beachten Sie folgendes Verhalten des XOR-Bausteins in erweiterter Form, also bei mehr als 2 Eingängen: IndraLogic vergleicht die Eingänge paarweise und die jeweiligen Ergebnisse dann wiederum gegeneinander (entspricht der Norm, jedoch nicht unbedingt der Erwartung).

Beispiele

Ergebnis in var1 ist 2#0001_1001

ST:

```
var1 := 2#1001_0011 XOR 2#1000_1010
```

FUP:

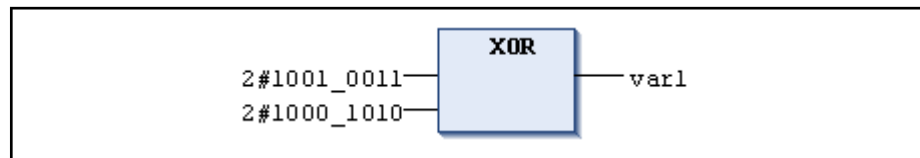


Abb. 17-44:

17.3.13 Operator 'AND_THEN'

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

Der Operator AND_THEN ist nur für die Programmierung im Strukturiertem Text bei folgender Operation erlaubt: AND-Operation von Operanden des Typs BOOL und BIT mit Kurzschluss-Auswertung. Das bedeutet:

Wenn alle Operanden TRUE sind, ist das Ergebnis der Operation ebenfalls TRUE, ansonsten FALSE.

Aber: Nur wenn der erste Operand des AND_THEN Operators TRUE ist, führt IndraLogic die Ausdrücke an weiteren Operanden ebenfalls aus. Dies kann beispielsweise in Bedingungen wie IF (ptr <> 0 AND_THEN ptr^ = 99) THEN... Probleme mit Null-Pointern vermeiden.

Im Unterschied dazu wertet IndraLogic bei Verwendung des IEC-Operators AND immer alle Operanden aus.

Siehe auch

- [Kap. 17.3.10 "Operator 'AND'" auf Seite 323](#)

17.3.14 Operator 'OR_ELSE'

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

Der Operator `OR_ELSE` ist nur für die Programmierung im Strukturiertem Text erlaubt: OR-Operation von Operanden des Typs `BOOL` und `BIT`, mit Kurzschluss-Auswertung. Das bedeutet:

Wenn mindestens einer der Operanden `TRUE` ist, ist das Ergebnis der Operation ebenfalls `TRUE`, ansonsten `FALSE`.

Im Unterschied zur Verwendung des IEC-Operators `OR`, werden bei `OR_ELSE`, sobald einer der Operanden mit `TRUE` ausgewertet wurde, die Ausdrücke an allen weiteren Operanden nicht mehr ausgeführt

Beispiel:

```
VAR
  bEver : BOOL;
  bX    : BOOL;
  dw    : DWORD := 16#000000FF;
END_VAR

bEver := FALSE;
bX := dw.8
  OR_ELSE dw.1
  OR_ELSE dw.1
  OR_ELSE (bEver := TRUE);
```

`dw.8` ist `FALSE` und `dw.1` ist `TRUE`, somit ist das Ergebnis `bX` der Operation `TRUE`. Der Ausdruck am dritten Eingang wird jedoch nicht ausgeführt, `bEver` bleibt `FALSE`. Wenn stattdessen die Standard-OR-Operation verwendet würde, wäre `bEver` auf `TRUE` gesetzt worden.

Siehe auch

- [Kap. 17.3.11 "Operator 'OR'" auf Seite 323](#)

17.3.15 Operator 'SHL'

Der IEC-Operator dient dem bitweisen Verschieben eines Operanden nach links.

`erg := SHL (in, n)`

`in`: Operand, der nach links verschoben wird.

`n`: Anzahl der Bits, um die `in` nach links verschoben wird.



Wenn `n` die Datentyp-Breite überschreitet, hängt es vom Zielsystem ab, wie `BYTE`, `WORD`, `DWORD` und `LWORD` Operanden aufgefüllt werden. Die Zielsysteme bewirken Auffüllen mit Nullen oder mit `n MOD <Registerbreite>`.



Beachten Sie, dass Sie die Anzahl der Bits, die IndraLogic für die Rechenoperation berücksichtigt, durch den Datentyp der Eingangsvariablen `in` vorgeben.

Beispiele

Die Ergebnisse für `erg_byte` und `erg_word` sind unterschiedlich, obwohl die Werte der Eingangsvariablen `in_byte` und `in_word` gleich, die Datentypen der Eingangsvariablen jedoch unterschiedlich sind.

ST:

```

PROGRAM shl_st
VAR
  in_byte  : BYTE:=16#45; (* 2#01000101 )
  in_word  : WORD:=16#0045; (* 2#0000000001000101 )
  erg_byte : BYTE;
  erg_word : WORD;
  n: BYTE  :=2;
END_VAR

erg_byte := SHL(in_byte,n); (* Result is 16#14, 2#00010100 *)
erg_word := SHL(in_word,n); (* Result is 16#0114, 2#0000000100010100 *)

```

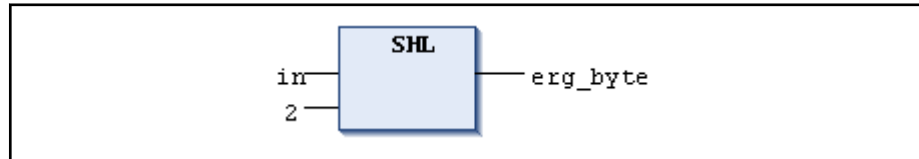
FUP:

Abb. 17-45:

17.3.16 Operator 'SHR'

Der IEC-Operator dient dem bitweisen Verschieben eines Operanden nach rechts.

`erg := SHR (in, n)`

`in`: Operand, der nach rechts verschoben wird.

`n`: Anzahl der Bits, um die `in` nach rechts verschoben wird.



Wenn `n` die Datentyp-Breite überschreitet, hängt es vom Zielsystem ab, wie BYTE-, WORD-, DWORD- und LWORD-Operanden aufgefllt werden. Die Zielsysteme bewirken Auffllen mit Nullen oder mit `n MOD <register width>`.

Beispiele**ST:**

```

PROGRAM shr_st
VAR
  in_byte  : BYTE :=16#45; (* 2#01000101 )
  in_word  : WORD :=16#0045; (* 2#0000000001000101 )
  erg_byte : BYTE;
  erg_word : WORD;
  n: BYTE  :=2;
END_VAR

erg_byte := SHR(in_byte,n); (* Result is 16#11, 2#00010001 *)
erg_word := SHR(in_word,n); (* Result is 16#0011, 2#0000000000010001 *)

```

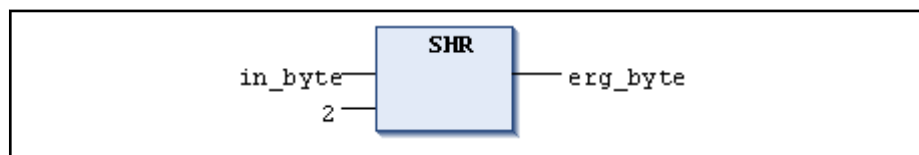
FUP:

Abb. 17-46:

17.3.17 Operator 'ROL'

Der IEC-Operator dient der bitweisen Rotation eines Operanden nach links.

Erlaubte Datentypen: BYTE, WORD, DWORD, LWORD

```
erg := ROL (in, n)
```

IndraLogic verschiebt *in* n-mal um 1 Bit nach links und fügt gleichzeitig das Bit mit der äußersten linken Position von rechts wieder ein.



Die Anzahl der Bits, die IndraLogic für die Rechenoperation berücksichtigt, geben Sie durch den Datentyp der Eingangsvariable *in* vor. Wenn es sich hierbei um eine Konstante handelt, berücksichtigt IndraLogic den kleinstmöglichen Datentyp. Der Datentyp der Ausgangsvariablen bleibt ohne Auswirkung auf die Rechenoperation.

Beispiele

Die Ergebnisse für *erg_byte* und *erg_word* sind unterschiedlich, abhängig vom Datentyp der Eingangsvariablen, obwohl die Werte der Eingangsvariablen *in_byte* und *in_word* gleich sind.

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); (* Result: 16#15 *)
erg_word := ROL(in_word,n); (* Result: 16#0114 *)
```

FUP:

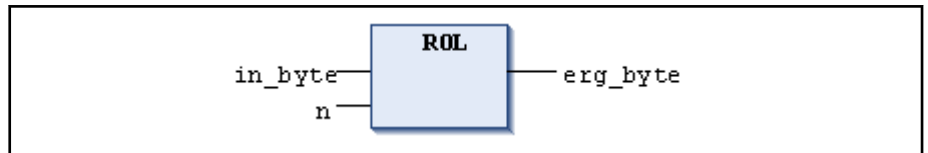


Abb. 17-47:

17.3.18 Operator 'ROR'

Der IEC-Operator dient der bitweisen Rotation eines Operanden nach rechts.

Erlaubte Datentypen: BYTE, WORD, DWORD, LWORD

```
erg := ROR(in,n)
```

IndraLogic verschiebt *in* n-mal um 1 Bit nach rechts und fügt gleichzeitig das Bit mit der äußersten rechten Position von links wieder ein.



Die Anzahl der Bits, die IndraLogic für die Rechenoperation berücksichtigt, wird durch den Datentyp der Eingangsvariablen *in* vorgegeben. Handelt es sich hierbei um eine Konstante, berücksichtigt IndraLogic den kleinstmöglichen Datentyp. Der Datentyp der Ausgangsvariable bleibt ohne Auswirkung auf die Rechenoperation.

Beispiele

Die Ergebnisse für *erg_byte* und *erg_word* sind unterschiedlich, abhängig vom Datentyp der Eingangsvariablen, obwohl die Werte der Eingangsvariablen *in_byte* und *in_word* gleich sind.

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); (* Result: 16#51 *)
erg_word := ROR(in_word,n); (* Result: 16#4011 *)

```

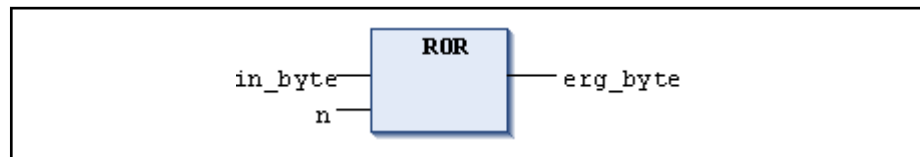
FUP:

Abb. 17-48:

17.3.19 Operator 'SEL'

Der IEC-Operator dient der bitweisen Auswahl.

`OUT := SEL(G, IN0, IN1)` bedeutet:

`OUT := IN0; wenn G = FALSE`

`OUT := IN1; wenn G = TRUE`

Erlaubte Datentypen:

`IN0, ..., INn` und `OUT`: Beliebiger, identischer Datentyp. Achten Sie speziell bei der Verwendung von benutzerdefinierten Datentypen darauf, dass an allen drei Positionen Variablen mit identischem Typ verwendet werden.

Der Compiler prüft die Typgleichheit und gibt Übersetzungsfehler aus. Insbesondere die Zuweisung von Instanzen eines Funktionsbausteins an Schnittstellen(variablen) wird nicht unterstützt.

G: BOOL



IndraLogic berechnet einen Ausdruck, der `IN0` vorgeschaltet ist, nicht, wenn `G TRUE` ist.

IndraLogic berechnet einen Ausdruck, der `IN1` vorgeschaltet ist, nicht, wenn `G FALSE` ist.

Achtung: Bei grafischen Programmiersprachen werden unabhängig vom Eingang `G` die Ausdrücke an `IN0` und `IN1` berechnet, wenn ein "Baustein", ein "Sprung", ein "Return", eine "Leitungsverzweigung" oder eine "Flankenerkennung" vorgeschaltet ist.

Beispiele

ST:

```

Var1 := SEL(TRUE,3,4); (* Result: 4 *)

```

FUP:

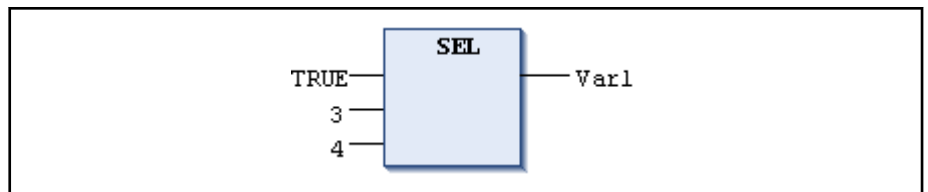


Abb. 17-49:

17.3.20 Operator 'MAX'

Der IEC-Operator dient der Maximumsfunktion. Er liefert von zwei Werten den größten Wert.

OUT := MAX(IN0, IN1)

Erlaubte Datentypen: alle

Beispiele

ST, Ergebnis: 90

Var1 := MAX(30, 40);

Var1 := MAX(40, MAX(90, 30));

FUP:

Ergebnis: 90

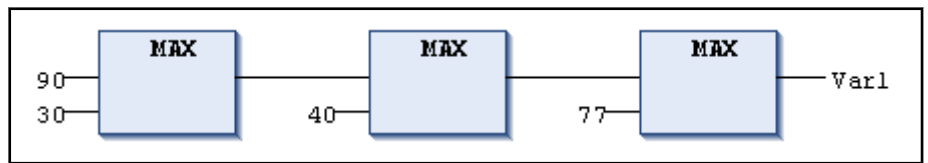


Abb. 17-50:

17.3.21 Operator 'MIN'

Der IEC-Operator dient der Minimumsfunktion. Er liefert von zwei Werten den kleinsten Wert.

OUT := MIN(IN0, IN1)

Erlaubte Datentypen: alle

Beispiele

Ergebnis: 30

ST

Var1 := MIN(90, 30);

Var1 := MIN(MIN(90, 30), 40);

FUP:

Ergebnis: 30

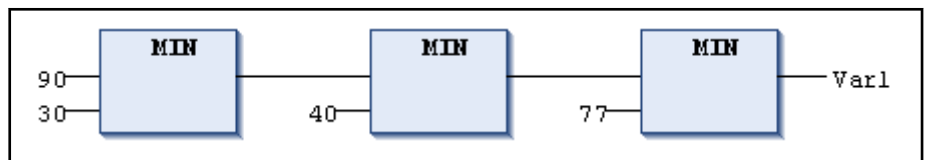


Abb. 17-51:

17.3.22 Operator 'LIMIT'

Der IEC-Auswahloperator dient der Limitierung.

```
OUT := LIMIT(Min, IN, Max)
```

bedeutet: `OUT := MIN (MAX (IN, Min), Max)`

`Max` ist die obere, `Min` ist 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`.

Erlaubte Datentypen für `IN` und `OUT`: alle

Beispiele

Ergebnis in `Var1` ist 80

ST:

```
Var1 := LIMIT(30,90,80);
```

17.3.23 Operator 'MUX'

Der IEC-Operator dient als Multiplexer

```
OUT := MUX(K, IN0, ..., INn)
```

bedeutet `OUT = IN_K`

Erlaubte Datentypen für `K`: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, LINT, ULINT, UDINT.

`IN0, ..., INn` und `OUT`: Beliebiger, identischer Datentyp. Achten Sie speziell bei der Verwendung von benutzerdefinierten Datentypen darauf, dass an allen drei Positionen Variablen mit identischem Typ verwendet werden.

Der Compiler prüft die Typgleichheit und gibt Übersetzungsfehler aus. Insbesondere die Zuweisung von Instanzen eines Funktionsbausteins an Schnittstellen(variablen) wird nicht unterstützt.

`MUX` wählt aus einer Menge von Werten den `K`-ten aus.

Der erste Wert entspricht `K=0`. Wenn `K` größer als die Anzahl der weiteren Eingänge (`n`) ist, so gibt IndraLogic den letzten Wert weiter (`INn`).



Zum Zweck der Laufzeitoptimierung berechnet IndraLogic nur den Ausdruck, den Sie `IN_K` vorgeschaltet haben. In der Simulation dagegen berechnet IndraLogic alle Zweige.

Beispiele

Ergebnis in `Var1` ist 30.

ST:

```
Var1 := MUX(0,30,40,50,60,70,80);
```

17.3.24 Operator 'GT'

Der IEC-Operator dient der Funktion "Größer als".

Erlaubte Datentypen der Operanden: beliebiger Basisdatentyp.

Wenn der erste Operand größer als der zweite Operand ist, liefert der Operator das Ergebnis `TRUE`, ansonsten `FALSE`.

Beispiele

Ergebnis: `FALSE`

ST:

```
VAR1 := 20 > 30;
```

FUP:

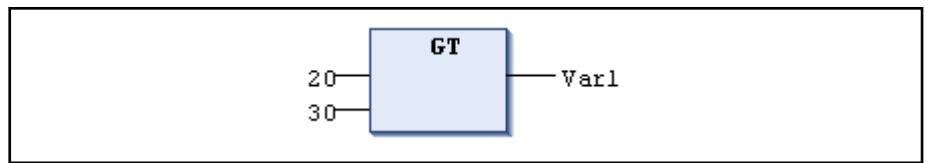


Abb. 17-52:

17.3.25 Operator 'LT'

Der IEC-Operator dient der Funktion "Kleiner als".

Erlaubte Datentypen der Operanden: beliebiger Basisdatentyp.

Wenn der erste Operand kleiner als der zweite ist, liefert der Operator das Ergebnis **TRUE**, ansonsten **FALSE**.

Beispiele

Ergebnis: **TRUE**

ST:

```
Var1 := 20 < 30;
```

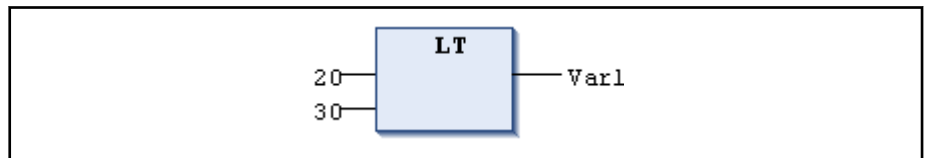


Abb. 17-53:

17.3.26 Operator 'LE'

Der IEC-Operator dient der Funktion "Kleiner oder gleich".

Erlaubte Datentypen der Operanden: beliebiger Basisdatentyp.

Wenn der erste Operand kleiner als der zweite Operand oder gleich groß wie der zweite Operand ist, liefert der Operator das Ergebnis **TRUE**, ansonsten **FALSE**.

Beispiele

Ergebnis in Var1: **TRUE**

ST:

```
Var1 := 20 <= 30;
```

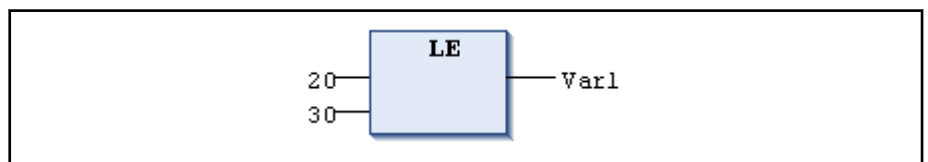


Abb. 17-54:

17.3.27 Operator 'GE'

Der IEC-Operator dient der Funktion "größer oder gleich".

Erlaubte Datentypen der Operanden: beliebiger Basisdatentyp.

Wenn der erste Operand größer als der zweite Operand oder gleich groß wie der zweite Operand ist, liefert der Operator das Ergebnis **TRUE**, ansonsten **FALSE**.

Beispiele

Ergebnis: **TRUE**

ST:

```
Var1 := 60 >= 40;
```

FUP:

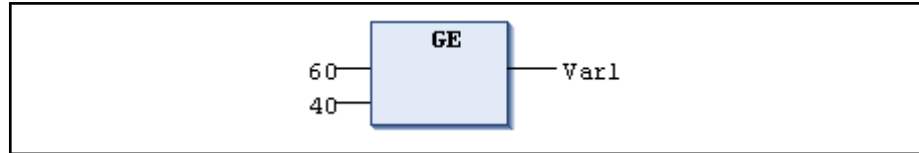


Abb. 17-55:

17.3.28 Operator 'EQ'

Der IEC-Operator dient der Funktion "Gleichheit".

Zulässige Datentypen der Operanden: beliebiger Basisdatentyp, zielsystem- und compilerversionsabhängig: Strukturdatentyp.

Wenn die Operanden gleich sind, liefert der Operator das Ergebnis **TRUE**, ansonsten **FALSE**.

Beispiele

Ergebnis: **TRUE**

ST:

```
Var1 := 40 = 40;
```

FUP:

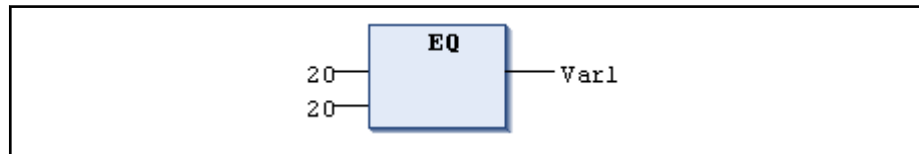


Abb. 17-56:

17.3.29 Operator 'NE'

Der IEC-Operator dient der Funktion "Ungleichheit".

Zulässige Datentypen der Operanden: beliebiger Basisdatentyp, zielsystem- und compilerversionsabhängig: Strukturdatentyp.

Wenn die Operanden ungleich sind, liefert der Operator das Ergebnis **TRUE**, ansonsten **FALSE**.

Wenn das Zielsystem den Datentyp unterstützt, können ab Compilerversion **>= 3.5.7.0** auch Operanden vom Typ **STRUCT** (Struktur) verglichen werden.

Beispiel: **IF (stStruct1 <> stStruct2) THEN....**

Beispiele

Ergebnis in Var1 ist **FALSE**

ST:

```
Var1 := 40 <> 40;
```

FUP:

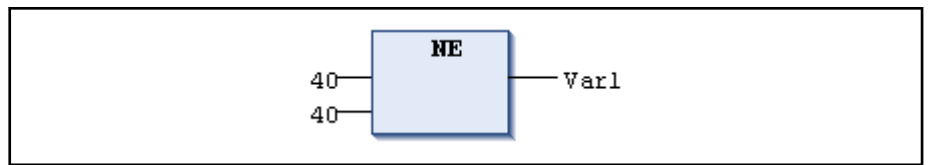


Abb. 17-57:

17.3.30 Operator 'ADR'

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

ADR liefert die Adresse seines Arguments in einem `DWORD`. Diese Adresse können Sie an Herstellerfunktionen übergeben oder innerhalb des Projekts einem Pointer zuweisen.



Im Unterschied zu IndraLogic 1x können Sie den ADR-Operator mit Funktions-, Programm-, Funktionsbaustein- und Methodennamen verwenden.

ADR ersetzt damit den `INDEXOF`-Operator.

Beachten Sie bei der Verwendung von Funktionszeigern Folgendes:

Sie können Funktionszeiger zwar an externe Bibliotheken weitergeben, es gibt jedoch keine Möglichkeit, einen Funktionszeiger innerhalb von IndraLogic aufzurufen!

Um einen Systemaufruf (Laufzeitsystem) zu ermöglichen, müssen Sie die entsprechende Objekteigenschaft (Registerkarte **Übersetzen**) für das Funktionsobjekt setzen.

HINWEIS

Wenn Sie Pointer auf Adressen verwenden, können sich beim Online-Change Inhalte von Adressen verschieben!

Wenn Sie direkte Adressen verwenden, verschieben sich Inhalte von Adressen beim Online-Change nicht!

Siehe auch [Kap. "Befehl 'Online-Change'" auf Seite 657](#).

⚠ VORSICHT

Pointer-TO

Pointer-TO -Variablen von Funktionen und Methoden sollten Sie nicht an den Aufrufer zurückgeben oder globalen Variablen zuweisen.

Beispiele:

ST:

```
dwVar := ADR(bVAR);
```

Siehe auch

- [Kap. 17.5.12 "Datentyp 'POINTER'" auf Seite 383](#)
- [Kap. 17.5.13 "Datentyp 'REFERENCE'" auf Seite 384](#)

17.3.31 Operator 'Inhaltsoperator'

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

Der Operator ermöglicht das Dereferenzieren eines Pointers. Sie hängen den Operator als `^` an den Pointer-Bezeichner an.

HINWEIS

Wenn Sie Pointer auf Adressen verwenden, können sich beim Online-Change Inhalte von Adressen verschieben!

Wenn Sie direkte Adressen verwenden, verschieben sich Inhalte von Adressen beim Online-Change nicht!

Siehe auch [Kap. "Befehl 'Online-Change'" auf Seite 657](#).

Beispiel

ST:

```
pt      : POINTER TO INT;
var_int1 : INT;
var_int2 : INT;
pt      := ADR(var_int1);
var_int2 := pt^;
```

17.3.32 Operator 'BITADR'

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

BITADR liefert den Bit-Offset innerhalb des Segments in einem `DWORD`.



Der Offset hängt davon ab, ob die Option Byteadressierung in den Zielsystemeinstellungen aktiviert ist oder nicht.

Der höchstwertige Nibble (4 Bit) in diesem `DWORD` beschreibt den Speicherbereich:

Merker: 16#4000 0000

Eingang: 16#8000 0000

Ausgang: 16#C000 0000

HINWEIS

Wenn Sie Pointer auf Adressen verwenden, können sich beim Online-Change Inhalte von Adressen verschieben!

Wenn Sie direkte Adressen verwenden, verschieben sich Inhalte von Adressen beim Online-Change nicht!

Siehe auch [Kap. "Befehl 'Online-Change'" auf Seite 657](#).

Beispiele

ST:

```
VAR
  var1 AT %IX2.3 : BOOL;
  bitoffset      : DWORD;
END_VAR

bitoffset := BITADR(var1); (* Result if byte
  addressing = TRUE: 16#80000013,
  if byte addressing = FALSE: 16#80000023 *)
```

17.3.33 Operator 'CAL'

Der IEC Operator dient dem Aufruf eines Funktionsbausteins.

CAL ruft in AWL die Instanz eines Funktionsbausteins auf.

Syntax

```
CAL <function block>
<input variable1> := <value>, <input variableN> := <value>
```

Beispiel

Aufruf der Instanz `Inst` eines Funktionsbausteins mit Belegung der Eingangsvariablen `Par1`, `Par2` auf 0 oder `TRUE`.

ST:

```
CAL Inst(Par1 := 0, Par2 := TRUE) ;
```

17.3.34 Operator 'BOOL_TO'

Der IEC-Operator dient der Konvertierung von Datentyp `BOOL` zu einem anderen Datentyp.

```
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` oder `FALSE`.

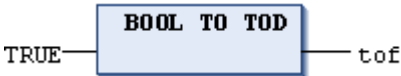
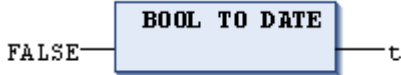
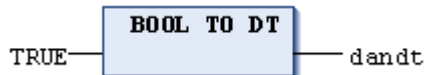


Wenn der Eingangswert eines Typkonvertierungsoperators außerhalb des Wertebereichs des Ausgangsdatentyps liegt, ist das Ergebnis der Operation nicht definiert und abhängig von der Plattform. Dies ist zum Beispiel der Fall, wenn der Eingangswert für die Konvertierung von `LREAL` nach `DINT` negativ ist.

Beispiel:

ST-Code	Ergebnis
<code>i := BOOL_TO_INT(TRUE);</code>	1
<code>str := BOOL_TO_STRING(TRUE);</code>	TRUE
<code>t := BOOL_TO_TIME(TRUE);</code>	T#1ms
<code>tof := BOOL_TO_TOD(TRUE);</code>	TOD#00:00:00.001
<code>dat := BOOL_TO_DATE(FALSE);</code>	D#1970
<code>dandt := BOOL_TO_DT(TRUE);</code>	DT#1970-01-01-00:00:01

FUP-Code	Ergebnis
	1
	TRUE
	T#1ms

FUP-Code	Ergebnis
	TOD#00:00:00.001
	D#1970-01-01
	DT#1970-01-01-00:00:01

17.3.35 Operator 'TO_BOOL'

Der IEC-Operator dient der Konvertierung von einer Variablen anderen Typs nach **BOOL**.

<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**.

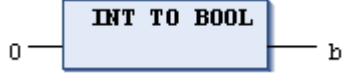


Wenn der Eingangswert eines Typkonvertierungsoperators außerhalb des Wertebereichs des Ausgangsdatentyps liegt, ist das Ergebnis der Operation nicht definiert und abhängig von der Plattform. Dies ist zum Beispiel der Fall, wenn der Eingangswert für die Konvertierung von **LREAL** nach **DINT** negativ ist.

Beispiel:

Beispiele

ST-Code	Ergebnis
b := BYTE_TO_BOOL(2#11010101);	TRUE
b := INT_TO_BOOL(0);	FALSE
b := TIME_TO_BOOL(T#5ms);	TRUE
b := STRING_TO_BOOL('TRUE');	TRUE

FUP-Code	Ergebnis
	TRUE
	FALSE

FUP-Code	Ergebnis
	TRUE
	TRUE

17.3.36 Operator 'TO_<xxx>'

Der IEC-Operator dient der Konvertierung von Variablen eines Typs in einen anderen Typ, wobei Sie den Eingangstyp nicht explizit angeben müssen. ("überladene Konvertierung").

TO_<Datentyp>

Es gilt dabei jeweils, was für die einzelnen getypten Konvertierungen beschrieben ist.



Wenn der Eingangswert eines Typkonvertierungsoperators außerhalb des Wertebereichs des Ausgangsdantentyps liegt, ist das Ergebnis der Operation nicht definiert und abhängig von der Plattform. Dies ist zum Beispiel der Fall, wenn der Eingangswert für die Konvertierung von LREAL nach DINT negativ ist.

Beispiele

Implementierungssprache ST:

```

VAR
  iVar : INT;
  bVar : BOOL;
  sVar : STRING;
  rVar : REAL;
END_VAR

iVar := TO_INT(4.22);      (* Result: 4 *)
bVar := TO_BOOL(1);       (* Result: TRUE *)
sVar := TO_STRING(342);   (* Result: '342' *)
rVar := TO_WORD('123');  (* Result: 123 *)

```

17.3.37 Operator '<INT-Typ>_TO_<INT-Typ>'

Konvertierung eines ganzzahligen Datentyps zu einem anderen ganzzahligen Datentyp.

<INT-Datentyp>_TO_<INT-Datentyp>



Bei der Typkonvertierung von größeren zu kleineren Datentypen können Informationen verloren gehen. Wenn die zu konvertierende Zahl die Bereichsgrenze überschreitet, berücksichtigt IndraLogic die ersten Bytes der Zahl nicht.



Wenn der Eingangswert eines Typkonvertierungsoperators außerhalb des Wertebereichs des Ausgangsdantentyps liegt, ist das Ergebnis der Operation nicht definiert und abhängig von der Plattform. Dies ist zum Beispiel der Fall, wenn der Eingangswert für die Konvertierung von LREAL nach DINT negativ ist.

Beispiele

Ergebnis in `si`: 127

ST:

```
si := INT_TO_SINT(4223);
```

Wenn Sie die Integer-Zahl 4223 (16#107f in Hexadezimaldarstellung) in eine `SINT`-Variable speichern, dann enthält diese Variable die Zahl 127 (16#7f in Hexadezimaldarstellung).

FUP:

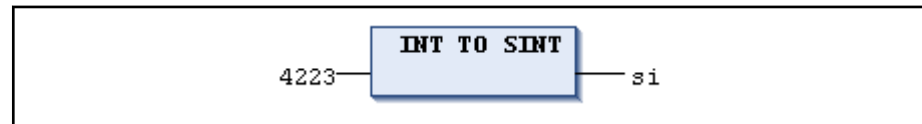


Abb. 17-58:

17.3.38 Operator 'REAL_TO-/ LREAL-TO'

Der IEC-Operator dient der Konvertierung vom Typ `REAL` und `LREAL` zu einem anderen Typ.

`REAL_TO_<Datentyp>`

`LREAL_TO_<Datentyp>`

IndraLogic rundet den Real-Wert des Operanden nach oben oder unten auf einen ganzzahligen Wert und wandelt in den entsprechenden Typen um. Ausgenommen davon sind die Typen `STRING`, `BOOL`, `REAL` und `LREAL`.



Wenn Sie eine Zahl vom Typ `REAL` oder `LREAL` zu `SINT`, `USINT`, `INT`, `UINT`, `DINT`, `UDINT`, `LINT` oder `ULINT` konvertieren und der Wert der `REAL/LREAL`-Zahl außerhalb des Wertebereichs des Integers liegt, erhalten Sie ein undefiniertes, zielsystemabhängiges Ergebnis. Auch ein Ausnahmefehler ist dann möglich! Um zielsystem-unabhängigen Code zu erhalten, müssen Sie Wertebereich-Überschreitungen über die Applikation abfangen. Wenn die `REAL/LREAL`-Zahl innerhalb des Bereichs liegt, arbeitet die Konvertierung auf allen Systemen gleich.

Beachten Sie bei der Konvertierung zu dem Typ `STRING`, dass die Gesamtkommastellen-Zahl 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, schneidet IndraLogic von rechts her entsprechend ab.



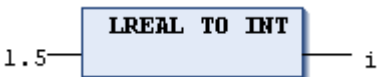
Bei der Typkonvertierung von größere auf kleinere Typen können Informationen verloren gehen.



Wenn der Eingangswert eines Typkonvertierungsoperators außerhalb des Wertebereichs des Ausgangsdatentyps liegt, ist das Ergebnis der Operation nicht definiert und abhängig von der Plattform. Dies ist zum Beispiel der Fall, wenn der Eingangswert für die Konvertierung von `LREAL` nach `DINT` negativ ist.

Beispiel:

ST-Code	Ergebnis
i := REAL_TO_INT(1.5);	2
j := REAL_TO_INT(1.4);	1
i := REAL_TO_INT(-1.5);	-2
j := REAL_TO_INT(-1.4);	-1

FUP-Code	Ergebnis
	2

17.3.39 Operator 'TIME_TO/TIME_OF_DAY_TO'

Der IEC-Operator dient der Konvertierung von Typ `TIME` bzw. `TIME_OF_DAY` zu einem anderen Typ.

`<TIME data type>_TO_<data type>`

Intern speichert IndraLogic die Zeit in einem `DWORD` in Millisekunden ab (bei `TIME_OF_DAY` seit 00:00 Uhr). Diesen Wert konvertiert IndraLogic.

Beim Typ `STRING` ist das Ergebnis die Zeitkonstante.



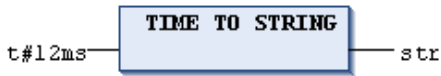
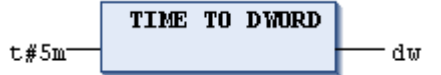
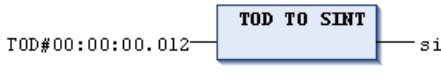
Bei der Typkonvertierung von größeren auf kleinere Typen können Informationen verloren gehen.



Wenn der Eingangswert eines Typkonvertierungsoperators außerhalb des Wertebereichs des Ausgangsdatentyps liegt, ist das Ergebnis der Operation nicht definiert und abhängig von der Plattform. Dies ist zum Beispiel der Fall, wenn der Eingangswert für die Konvertierung von `LREAL` nach `DINT` negativ ist.

Beispiel:

ST-Code	Ergebnis
str := TIME_TO_STRING(T#12ms);	T#12ms
dw := TIME_TO_DWORD(T#5m);	300000
si := TOD_TO_SINT(TOD#00:00:00.012);	12

FUP-Code	Ergebnis
	T#12ms
	30000
	12

17.3.40 Operator 'DATE_TO/DT_TO'

Der IEC-Operator dient der Konvertierung von Typ `DATE` oder `DATE_AND_TIME` zu einem anderen Typ.

`<DATE data type>_TO_<data type>`

Intern speichert IndraLogic das Datum in einem `DWORD` in Sekunden seit dem 1. Januar 1970. Diesen Wert konvertiert IndraLogic.

Beim Typ `STRING` ist das Ergebnis die Datumskonstante.



Bei der Typkonvertierung von größeren auf kleinere Typen können Informationen verloren gehen.



Wenn der Eingangswert eines Typkonvertierungsoperators außerhalb des Wertebereichs des Ausgangsdatentyps liegt, ist das Ergebnis der Operation nicht definiert und abhängig von der Plattform. Dies ist zum Beispiel der Fall, wenn der Eingangswert für die Konvertierung von `LREAL` nach `DINT` negativ ist.

Beispiel:

ST-Code	Ergebnis
<code>b := DATE_TO_BOOLEAN(D#1970-01-01);</code>	FALSE
<code>i := DATE_TO_INT(D#1970-01-15);</code>	29952
<code>byt := DT_TO_BYTE(DT#1970-01-15-05:05:05);</code>	129
<code>str := DT_TO_STRING(DT#1998-02-13-14:20);</code>	'DT#1998-02-13-14:20'

FUP-Code	Ergebnis
D#1970-01-01 — DATE TO BOOL — b	FALSE
D#1970-01-15 — DATE TO INT — i	29952
D#1970-01-15-05:05:05 — DATE TO BYTE — i	129
D#1998-02-13-14:20 — DT TO STRING — str	DT#1998-02-13-14:20

17.3.41 Operator 'STRING_TO'

Der IEC Operator dient der Konvertierung vom Typ `STRING` zu einem anderen Typ (Zieldatentyp).

`STRING_TO_<data type>`

Sie müssen den Operand vom Typ `STRING` gemäß der Norm IEC 61131-3 angeben. Der Wert muss einer gültigen Konstante (Literal) des Zieltyps entsprechen. Dies betrifft die Angabe von Exponentialwerten, infiniten Werten, Präfixen, Gruppierungszeichen ("_") und Komma.

Zusätzliche Zeichen hinter den Ziffern einer Zahl sind erlaubt, zum Beispiel: 23xy.

Zusätzliche Zeichen vor einer Zahl sind nicht erlaubt.

Der Operand muss einen gültigen Wert des Zieldatentyps darstellen.



Wenn der Datentyp des Operanden nicht zum Zieldatentyp passt oder der Wert außerhalb des Bereichs des Zieldatentyps liegt, hängt die Ergebnisausgabe vom Prozessortyp ab und ist damit undefiniert. Bei der Typkonvertierung von größeren auf kleinere Typen können Informationen verloren gehen.



Wenn der Eingangswert eines Typkonvertierungsoperators außerhalb des Wertebereichs des Ausgangsdantentyps liegt, ist das Ergebnis der Operation nicht definiert und abhängig von der Plattform. Dies ist zum Beispiel der Fall, wenn der Eingangswert für die Konvertierung von `LREAL` nach `DINT` negativ ist.

Beispiel:

ST-Code	Ergebnis
b := STRING_TO_BOOL('TRUE');	TRUE
w := STRING_TO_WORD('abc34');	0
w := STRING_TO_WORD('34abc');	34
t := STRING_TO_TIME('T#127ms');	T#127ms

ST-Code	Ergebnis
<code>r := STRING_TO_REAL('1.234');</code>	1.234
<code>bv := STRING_TO_BYTE('500');</code>	244

FUP-Code	Ergebnis
	TRUE

17.3.42 Operator 'TRUNC'

Der IEC-Operator dient der Konvertierung vom Typ `REAL` zum Typ `DINT`. IndraLogic 2G nimmt nur den ganzzahligen Anteil der Zahl.



In IndraLogic 1x konvertiert der `TRUNC`-Operator von `REAL` nach `INT`.

Wenn Sie ein IndraLogic 1x-Projekt importieren, ersetzt IndraLogic 2G `TRUNC` automatisch durch `TRUNC_INT`.

Wenn IndraLogic 2G den Eingangswert nicht durch einen `DINT` oder `INT` darstellen kann, ist das Ergebnis dieser Funktion undefiniert. Das Verhalten für solche Eingangswerte ist plattformabhängig.



Wenn der Eingangswert eines Typkonvertierungsoperators außerhalb des Wertebereichs des Ausgangsdatentyps liegt, ist das Ergebnis der Operation nicht definiert und abhängig von der Plattform. Dies ist zum Beispiel der Fall, wenn der Eingangswert für die Konvertierung von `LREAL` nach `DINT` negativ ist.

Beispiele

Ergebnis in `diVar` ist 1

ST:

```
diVar := TRUNC(1.9); (* Result: 1 *)
```

```
diVar := TRUNC(-1.4); (* Result: -1 *)
```

17.3.43 Operator 'TRUNC_INT'

Der IEC-Operator dient der Konvertierung vom Typ `REAL` zum Typ `INT`. IndraLogic 2G nimmt nur den Betrag des ganzzahligen Anteils der Zahl.



`TRUNC_INT` entspricht dem Operator `TRUNC` in IndraLogic 1x und wird beim Import von IndraLogic 1x-Projekten automatisch an dessen Stelle verwendet.

Beachten Sie die veränderte Funktion von `TRUNC` in.

Das Ergebnis dieser Funktion ist undefiniert, wenn IndraLogic 2G den Eingangswert nicht durch einen `DINT` oder `INT` darstellen kann. Das Verhalten für solche Eingangswerte ist plattformabhängig.



Wenn der Eingangswert eines Typkonvertierungsoperators außerhalb des Wertebereichs des Ausgangsdatentyps liegt, ist das Ergebnis der Operation nicht definiert und abhängig von der Plattform. Dies ist zum Beispiel der Fall, wenn der Eingangswert für die Konvertierung von LREAL nach DINT negativ ist.

Beispiele

Ergebnis in iVAR ist 1

ST:

```
iVar := TRUNC_INT(1.9); (* Result: 1 *)
```

```
iVar := TRUNC_INT(-1.4); (* Result: -1 *)
```

17.3.44 Operator 'ABS'

Der IEC-Operator liefert den Absolutwert einer Zahl.

Erlaubte Datentypen für die Eingangs- und Ausgangsvariablen und Zahlenkonstanten: beliebiger numerischer Basisdatentyp

Beispiele

Ergebnis in i: 2

ST:

```
i := ABS(-2);
```

FUP:

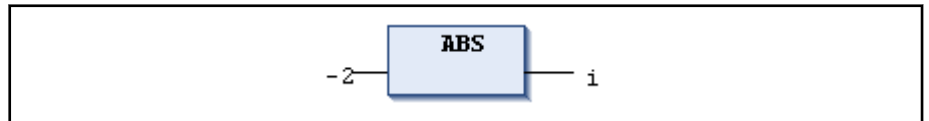


Abb. 17-59:

17.3.45 Operator 'SQRT'

Der IEC-Operator liefert die Quadratwurzel einer Zahl.

Erlaubte Datentypen für die Eingangsvariablen: beliebiger numerischer Basisdatentyp

Erlaubte Datentypen für die Ausgangsvariablen: REAL oder LREAL

Beispiele

Ergebnis in q ist 4.

ST:

```
q := SQRT(16);
```

FUP:



Abb. 17-60:

17.3.46 Operator 'LN'

Der IEC-Operator liefert den natürlichen Logarithmus einer Zahl.

Erlaubte Datentypen für die Eingangsvariablen: beliebiger numerischer Basisdatentyp

Erlaubte Datentypen für Ausgangsvariablen: `REAL` und `LREAL`

Beispiele

Ergebnis in `q` ist 3.80666

ST:

`q := LN(45);`

FUP:

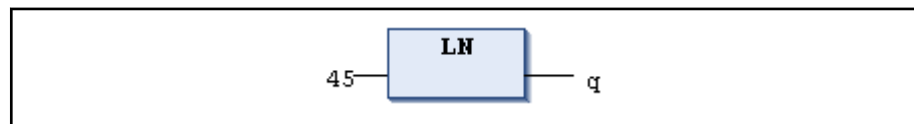


Abb. 17-61:

17.3.47 Operator 'LOG'

Der 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.

Beispiele

Ergebnis in `q` ist 2.49762

ST:

`q := LOG(314.5);`

FUP:

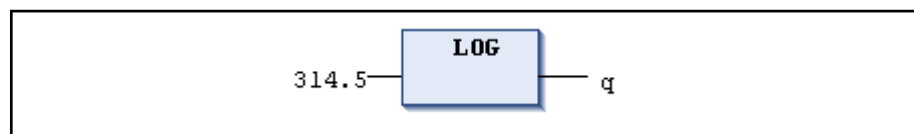


Abb. 17-62:

17.3.48 Operator 'EXP'

Der IEC-Operator liefert die Exponentialfunktion.

Erlaubte Datentypen für die Eingangsvariablen: beliebiger numerischer Basisdatentyp

Erlaubte Datentypen für die Ausgangsvariablen: `REAL` und `LREAL`

Beispiele

Ergebnis in `q` ist 7.389056099.

ST:

`q := EXP(2);`

FUP:

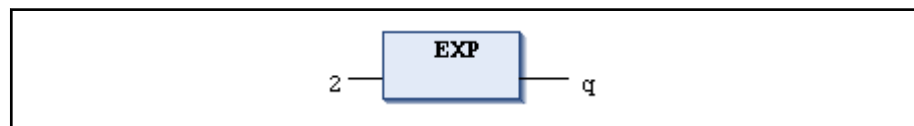


Abb. 17-63:

17.3.49 Operator 'EXPT'

Der IEC-Operator potenziert eine Zahl mit einer anderen und liefert die Potenz von Basis hoch Exponent zurück: $\text{power} = \text{base}^{\text{exponent}}$. Dabei sind so-

wohl Basis als auch Exponent Eingabewerte (Parameter). Die Potenzfunktion ist nicht definiert, wenn die Basis 0 und gleichzeitig der Exponent negativ ist. Das Verhalten in diesem Fall ist allerdings plattformabhängig.

Syntax:

```
EXPT(<Basis>,<Exponent>)
```

Erlaubte Datentypen für die Eingabewerte: Numerische Basisdatentypen (SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, BYTE, WORD, DWORD, LWORD)

Erlaubte Datentypen für den Rückgabewert: Gleitkommazahltypen (REAL, LREAL)

Beispiele:

Ergebnis in Var1: 49

Potenzfunktion mit Literalen

```
Var1 := EXPT(7,2);
```

FUP:

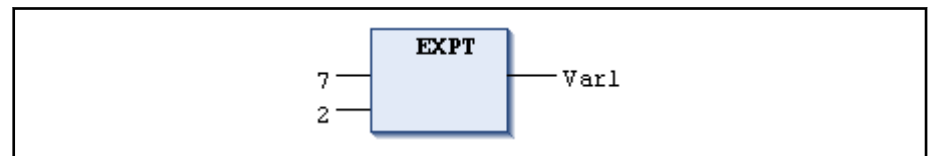


Abb. 17-64:

Rückgabewert: Var1 = 49

Potenzfunktion mit Variablen

Programm:

```
PROGRAM PLC_PRG
VAR
  lrPow      : LREAL;
  iBase      : INT := 7;
  iExponent  : INT := 2;
END_VAR

lrPow := EXPT(iBase, iExponent);
```

Rückgabewert: lrPow = 49

17.3.50 Operator 'SIN'

Der IEC-Operator liefert den Sinus-Wert für eine Zahl.

Erlaubte Datentypen für die Eingangsvariable, die den Winkel im Bogenmaß angibt: beliebiger numerischer Basisdatentyp

Erlaubte Datentypen für Ausgangsvariable: REAL und LREAL

Beispiele

Ergebnis in q ist 0.479426.

ST:

```
q := SIN(0.5);
```

FUP:

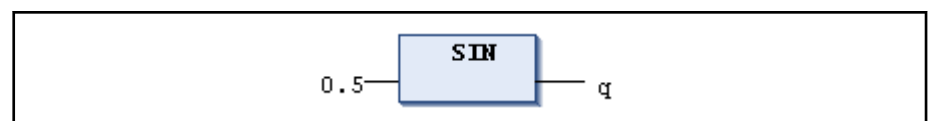


Abb. 17-65:

17.3.51 Operator 'COS'

Der IEC-Operator liefert den Cosinuswert für eine Zahl.

Erlaubte Datentypen für die Eingangsvariable, die den Winkel im Bogenmaß angibt: beliebiger numerischer Basisdatentyp

Erlaubte Datentypen für Ausgangsvariablen: REAL und LREAL

Beispiele

Ergebnis in q: 0.877583

ST:

```
q := COS(0.5);
```

FUP:

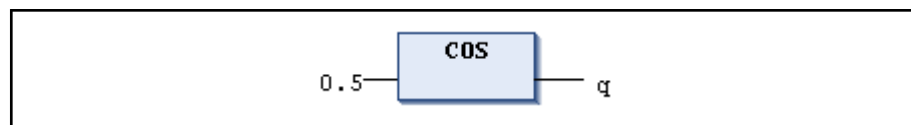


Abb. 17-66:

17.3.52 Operator 'TAN'

Der IEC-Operator liefert den Tangens-Wert für eine Zahl.

Erlaubte Datentypen für die Eingangsvariable, die den Winkel im Bogenmaß angibt: beliebiger numerischer Basisdatentyp

Erlaubte Datentypen für Ausgangsvariablen: REAL und LREAL

Beispiele

Ergebnis in q ist 0.546302

ST:

```
q := TAN(0.5);
```

FUP:

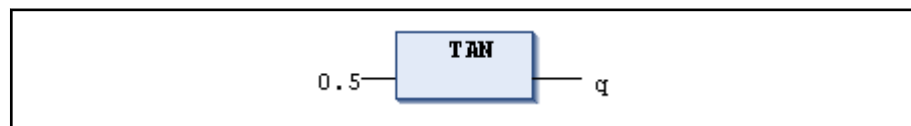


Abb. 17-67:

17.3.53 Operator 'ASIN'

Der IEC-Operator liefert den Arcussinus (Umkehrfunktion des Sinus) für eine Zahl.

Erlaubte Datentypen für die Eingangsvariable: Beliebiger numerischer Basisdatentyp

Erlaubte Datentypen für Ausgangsvariablen: REAL und LREAL

Beispiele

Ergebnis in q ist 0.523599

ST:

```
q := ASIN(0.5);
```

FUP:

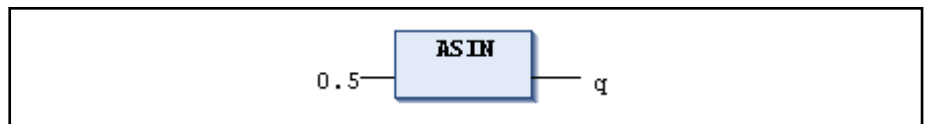


Abb. 17-68:

17.3.54 Operator 'ACOS'

Der IEC-Operator liefert den Arcuscosinus (Umkehrfunktion von Cosinus) für eine Zahl. Der Wert wird in Bogenmaß errechnet.

Erlaubte Datentypen für die Eingangsvariable, die den Winkel im Bogenmaß angibt: beliebiger numerischer Basisdatentyp

Erlaubte Datentypen für Ausgangsvariable: REAL und LREAL

Beispiele

Ergebnis in q ist 1.0472

ST:

$q := \text{ACOS}(0.5);$

FUP:

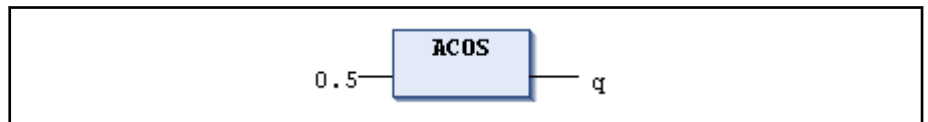


Abb. 17-69:

17.3.55 Operator 'ATAN'

Der IEC-Operator liefert den Arcustangens (Umkehrfunktion von Tangens) einer Zahl. Der Wert wird in Bogenmaß errechnet.

Erlaubte Datentypen für die Eingangsvariable, die den Winkel im Bogenmaß angibt: beliebiger numerischer Basisdatentyp

Erlaubte Datentypen für Ausgangsvariablen: REAL und LREAL

Beispiele

Ergebnis in q ist 0.463648

ST:

$q := \text{ATAN}(0.5);$

FUP:



Abb. 17-70:

17.3.56 Operator '__DELETE'

Dieser Operator wird in der Norm IEC61131-3 nicht beschrieben.

Der Operator gibt den Speicher von Instanzen wieder frei, die der Operator `__NEW` dynamisch erzeugt hat. Der Operator `__DELETE` hat keinen Rückgabewert, und der Operand wird nach dieser Operation auf 0 gesetzt.



Im Eigenschaftendialog der Applikation ist in der Registerkarte "Optionen Applikationserzeugung" die Option "Dynamische Speicherallozierung verwenden" aktiviert.

Syntax:

```
__DELETE (<Pointer>)
```



Zwei Tasks sollten nicht gleichzeitig `__DELETE` aufrufen. Entweder Sie verwenden ein Semaphore (SysSemEnter) oder eine vergleichbare Technik, um einen konkurrierenden Aufruf von `__DELETE` zu verhindern, oder Sie verwenden `__DELETE` nur in einer Task (empfohlenes Vorgehen).

Eine umfangreiche Nutzung von `__DELETE` verursacht einen höheren Jitter.

Wenn `Pointer` auf einen Funktionsbaustein zeigt, wird die zugehörige Methode `FB_EXIT` aufgerufen, bevor der Pointer auf 0 gesetzt wird.

Programm:

```
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;
```

Programm:

```
METHOD INC : INT
VAR_INPUT
    _inc : INT;
END_VAR

_inc := _inc + 1;
INC := _inc;
```

Programm:

```
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
```

Siehe auch:

- [Kap. "Dialog 'Eigenschaften - 'Optionen Applikationserzeugung'" auf Seite 751](#)
- [Kap. 17.10 "Methode 'FB_Init', 'FB_Reinitund 'FB_Exit'" auf Seite 455](#)
- [Kap. 17.3.58 "Operator '__NEW'" auf Seite 349](#)
- ["Anwendungsbeispiele" auf Seite 350](#)

17.3.57 Operator '__ISVALIDREF'

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

Der Operator dient der Überprüfung, ob eine Referenz auf einen gültigen Wert verweist. Eine Beschreibung der Anwendung und ein Beispiel finden Sie in der Beschreibung des Datentyps `REFERENCE`.

Siehe auch

- [Kap. 17.5.13 "Datentyp 'REFERENCE'" auf Seite 384](#)

17.3.58 Operator '__NEW'

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

Der `__NEW`-Operator reserviert dynamisch Speicher um Funktionsbausteine, benutzerdefinierte Datentypen oder Arrays von Standarddatentypen zu instanzieren. Der Operator gibt einen passend getypten Pointer zurück.

Voraussetzung: Im Eigenschaftendialog des Applikationsobjekts ist in der Registerkarte "Optionen Applikationserzeugung" die Option "Dynamische Speicherallozierung verwenden" aktiviert ([Kap. "Dialog 'Eigenschaften - 'Optionen Applikationserzeugung'" auf Seite 751](#)).

Syntax

```
<pointer name> := __NEW( <type> ( , <size> ) ? ) ;
__DELETE( <pointer name> );
```

```
<Type> : <function block> | <data unit type> | <standard data type>
```

Der Operator erzeugt eine Instanz des Typs `<type>` und gibt einen Pointer auf diese Instanz zurück. Anschließend wird die Initialisierung der Instanz aufgerufen. Wenn es sich bei `<type>` um einen skalaren Standarddatentyp handelt, wird zusätzlich der optionale Operand `<size>` ausgewertet. Der Operator erzeugt dann ein Array des Typs `<standard data type>` der Größe `<size>`. Wenn der Versuch Speicher zu allozieren scheitert, gibt `__NEW` den Wert 0 zurück.

Verwenden Sie den Operator innerhalb der Zuweisung ":", ansonsten wird eine Fehlermeldung ausgegeben.

Ein Funktionsbaustein oder ein benutzerdefinierter Datentyp, dessen Instanz mit `__NEW` dynamisch erzeugt wird, belegt einen fixen Speicherbereich. Dafür ist es erforderlich, dass Sie die Objekte mit dem Pragma `{attribute 'enable_dynamic_creation'}` kennzeichnen. Das ist bei Funktionsbausteinen, die Teil einer Bibliothek sind, nicht erforderlich.



Wenn Sie im Onlinebetrieb das Datenlayout des Funktionsbausteins ändern, können Sie anschließend kein Einloggen mit Online-Change ausführen. Der Speicherbereich der Funktionsbausteininstanz ist durch die Änderung ungültig geworden. Eine Änderung des Datenlayouts nehmen Sie vor, wenn Sie beim Funktionsbaustein neue Variablen hinzufügen, bestehende Variablen löschen oder Datentypen von Variablen ändern.

Anwendungsbeispiele *Array (DWORD):*

```

PROGRAM PLC_PRG
VAR
    pdwScalar : POINTER TO DWORD; //Typed pointer
    xInit : BOOL := TRUE;
    xDelete : BOOL;
END_VAR

IF (xInit) THEN
    pdwScalar := __NEW(DWORD, 16);
    // Allocates memory (16 dwords) and assigns them to pointer pdwScalar
END_IF
IF (xDelete) THEN
    __DELETE(pdwScalar); // Frees memory of pointer
END_IF

```

Funktionsbaustein

```

{attribute 'enable_dynamic_creation'}
FUNCTION BLOCK FBComputeGamma
VAR_INPUT
    iAlpha : INT;
    iBeta : INT;
END_VAR
VAR_OUTPUT
    iGamma : INT;
END_VAR
VAR
    iResult : INT;
END_VAR

iGamma := iAlpha + iBeta;

PROGRAM PLC_PRG
VAR
    pComputeGamma : POINTER TO FBComputeGamma; // Typed pointer
    xInit : BOOL := TRUE;
    xDelete : BOOL;
    iResult : INT;
END_VAR

IF (xInit) THEN
    pComputeGamma := __NEW(FBComputeGamma); // Allocates memory
    xInit := FALSE;
END_IF
pComputeGamma^.iAlpha := (pComputeGamma^.iAlpha + 1)MOD 100;
    // Sets first input of pComputeGamma
pComputeGamma^.iBeta := 10; // Sets second input of pComputeGamma
pComputeGamma^(); // Calls the FB pComputeGamma is pointing to
iResult := pComputeGamma^.iGamma; // Reads output of pComputeGamma
IF (xDelete) THEN
    __DELETE(pComputeGamma); // Frees memory
END_IF

```

Benutzerdefinierter Datentyp (DUT):

```

{attribute 'enable_dynamic_creation'}
TYPE ABCDATA :
STRUCT
    iA, iB, iC, iD : INT;
END_STRUCT
END_TYPE

PROGRAM PLC_PRG
VAR
    pABCDData : POINTER TO ABCDATA; // Typed pointer
    xInit : BOOL := TRUE;
    xDelete : BOOL;
END_VAR

IF (xInit) THEN
    pABCDData := __NEW(ABCDATA); // Allocates memory
    xInit := FALSE;
END_IF
IF (xDelete) THEN

```

```

    __DELETE(pABCDData); // Frees memory
END_IF

```

Array (BYTE)

```

PROGRAM PLC_PRG
VAR
    pbDataAlpha : POINTER TO BYTE;
    pbDataBeta  : POINTER TO BYTE;
    xInit       : BOOL := TRUE;
    xDelete     : BOOL;
    usiCnt      : USINT;
    bTestC      : BYTE;
END_VAR

IF (xInit) THEN
    pbDataAlpha := __NEW(BYTE, 16); // Allocates 16 bytes for pbDataAlpha
    pbDataBeta  := __NEW(BYTE);    // Allocates memory for pbDataBeta
    xInit := FALSE;

    FOR usiCnt := 0 TO 15 DO
        pbDataAlpha[usiCnt] := usiCnt; // Writes to new array
    END FOR
    pbDataBeta^:= 16#FF; // Writes to new data
END_IF

bTestC := pbDataAlpha[12]; // Reads new array by index access

IF (xDelete) THEN // Frees memory
    __DELETE(pbDataAlpha);
    __DELETE(pbDataBeta);
END_IF

```



Es ist nicht ratsam, zwei Tasks gleichzeitig auszuführen, die beide den `__NEW`-Operator aufrufen. Entweder Sie verwenden ein Semaphore (`SysSemEnter`) oder eine vergleichbare Technik, um einen konkurrierenden Aufruf von `__NEW` zu verhindern. Bei umfangreicher Nutzung von `__NEW` entsteht ein höherer Jitter.

Siehe auch:

- [Kap. "Dialog 'Eigenschaften - 'Optionen Applikationserzeugung'" auf Seite 751](#)
- [Kap. 18.1.8 "Objekt 'DUT'" auf Seite 517](#)
- [Kap. "Objekt 'Funktionsbaustein' \(FUNCTION_BLOCK\)" auf Seite 556](#)
- [Kap. "Attribut 'enable_dynamic_creation'" auf Seite 438](#)

17.3.59 Operator '`__QUERYINTERFACE`'

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

Der Operator führt zur Laufzeit eine Typkonvertierung einer Interface- Referenz zu einer anderen aus. Der Operator liefert ein Ergebnis vom Typ `BOOL` zurück. `TRUE` bedeutet, dass IndraLogic die Konvertierung erfolgreich durchgeführt hat.

Syntax:

```
__QUERYINTERFACE(<ITF_Source>, <ITF_Dest>);
```

1. Operand: Interface-Referenz oder FB-Instanz
2. Operand: Interface-Referenz mit gewünschten Zieltypen

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

FUNCTION_BLOCK FB1 IMPLEMENTS ItfDerived1
METHOD mbase : BOOL
    mbase := TRUE;
END_METHOD
METHOD mderived1 : BOOL
    mderived1 := TRUE;
END_METHOD
END_FUNCTION_BLOCK

FUNCTION_BLOCK FB2 IMPLEMENTS ItfDerived2
METHOD mbase : BOOL
    mbase := FALSE;
END_METHOD
METHOD mderived2 : BOOL
    mderived2 := TRUE;
END_METHOD
END_FUNCTION_BLOCK

PROGRAMM POU
VAR
    inst1 : FB1;
    inst2 : FB2;
    itfbasel : ItfBase := inst1;
    itfbase2 : ItfBase := inst2;
    itfderived1 : ItfDerived1 := 0;
    itfderived2 : ItfDerived2 := 0;
    xResult1, xResult2, xResult3, xResult4: BOOL;
END_VAR

xResult1 := __QUERYINTERFACE(itfbasel, itfderived1);
// xResult = TRUE, itfderived1 <> 0
// references the instance inst1
xResult2 := __QUERYINTERFACE(itfbasel, itfderived2);
// xResult = FALSE, itfderived2 = 0
xResult3 := __QUERYINTERFACE(itfbase2, itfderived1);
// xResult = FALSE, itfderived1 = 0
xResult4 := __QUERYINTERFACE(itfbase2, itfderived2);
// xResult = TRUE, itfderived2 <> 0
// references the instance inst2

```

17.3.60 Operator '___QUERYPOINTER'

Der Operator ist eine Erweiterung der IEC61131-3.

Der Operator 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 IndraLogic die Konvertierung erfolgreich durchgeführt hat.



Aus Kompatibilitätsgründen muss die Definition des zu konvertierenden Pointers eine Erweiterung des Basis-Interface **__SYSTEM.IQueryInterface** sein.

Syntax:

```
___QUERYPOINTER (<ITF_Source>,< Pointer_Dest>);
```

Der Operator bekommt als ersten Operanden eine Interface-Referenz oder eine FB-Instanz mit den gewünschten Zieltypen und als zweiten Operanden einen Pointer.

Nach Abarbeiten von `__QUERYPOINTER` enthält `Pointer_Dest` den Pointer auf diejenige Referenz, oder Instanz eines Funktionsbausteins, auf welche die Interface-Referenz `ITF_Source` aktuell verweist.

`Pointer_Dest` ist nicht getypt und kann auf einen beliebigen Typ gecastet werden. Sie müssen den Typ sicherstellen. Beispielsweise könnte das Interface eine Methode anbieten, die einen Typ-Code zurück liefert.

17.3.61 Operator '`__TRY`', '`__CATCH`', '`__FINALLY`', '`__ENDTRY`'

Die Operatoren sind eine Erweiterung der Norm IEC 61131-3 und dienen einem gezielten Exceptionhandling im IEC-Code.

Syntax

```

__TRY
    <statements_try>
__CATCH(exec)
    <statements_catch>
__FINALLY
    <statements_finally>
__ENDTRY
    <statements_next>

```

Wenn eine Anweisung, die unter dem Operator `__Try` steht, eine Exception produziert, hält die Applikation nicht an. Sie führt stattdessen die Anweisungen unter `__Catch` aus und startet damit das Exceptionhandling. Danach führt sie die Anweisungen unter `__FINALLY` aus. Das Exceptionhandling endet mit `__ENDTRY`. Dann führt die Applikation die anschließenden Anweisungen aus.

Eine IEC-Variable für eine Exception hat den Datentyp `__System.ExceptionCode`.

Beispiel

Wenn die Anweisung unter `__TRY` eine Exception produziert, wird die Programmabarbeitung nicht gestoppt, sondern die Anweisung unter `__CATCH` abgearbeitet. In diesem Beispiel wird also die Funktion `exc` und dann die Anweisung unter `__FINALLY` ausgeführt. Danach führt die Applikation die Anweisung unter `__ENDTRY` aus.

Beispiel

```

FUNCTION Tester : UDINT
VAR_INPUT
    count : UDINT;
END_VAR
VAR_OUTPUT
    strExceptionText : STRING;
END_VAR
VAR
    exc : __SYSTEM.ExceptionCode;
END_VAR

__TRY
Tester := tryFun(count := count, testcase := g_testcase); (*Diese Anweisung wird getestet.
                                                         Wenn sie eine Exception produziert,
                                                         werden zuerst die Anweisung
                                                         unter __CATCH
                                                         und danach die Anweisung
                                                         unter __FINALLY ausgeführt.*)

__CATCH(exc)
HandleException(exc, strExceptionText => strExceptionText);
__FINALLY
GVL.g_count := GVL.g_count + 2;
__ENDTRY

```

Siehe auch

- [Kap. "Befehl 'Ausführung an behandelten Exceptions anhalten'" auf Seite 743](#)

__System.ExceptionCode	System.ExceptionCode
	<pre> TYPE ExceptionCode : (RTSEXCPT_UNKNOWN := 16#FFFFFFFF, RTSEXCPT_NOEXCEPTION := 16#00000000, RTSEXCPT_WATCHDOG := 16#00000010, RTSEXCPT_HARDWAREWATCHDOG := 16#00000011, RTSEXCPT_IO_CONFIG_ERROR := 16#00000012, RTSEXCPT_PROGRAMCHECKSUM := 16#00000013, RTSEXCPT_FIELDBUS_ERROR := 16#00000014, RTSEXCPT_IUPDATE_ERROR := 16#00000015, RTSEXCPT_CYCLE_TIME_EXCEED := 16#00000016, RTSEXCPT_ONLCHANGE_PROGRAM_EXCEEDED := 16#00000017, RTSEXCPT_UNRESOLVED_EXTREMS := 16#00000018, RTSEXCPT_DOWNLOAD_REJECTED := 16#00000019, RTSEXCPT_BOOTPROJECT_REJECTED_DUE_RETAIN_ERROR := 16#0000001A, RTSEXCPT_LOADBOOTPROJECT_FAILED := 16#0000001B, RTSEXCPT_OUT_OF_MEMORY := 16#0000001C, RTSEXCPT_RETAIN_MEMORY_ERROR := 16#0000001D, RTSEXCPT_BOOTPROJECT_CRASH := 16#0000001E, RTSEXCPT_BOOTPROJECTTARGETMISMATCH := 16#00000021, RTSEXCPT_SCHEDULEERROR := 16#00000022, RTSEXCPT_FILE_CHECKSUM_ERR := 16#00000023, RTSEXCPT_RETAIN_IDENTITY_MISMATCH := 16#00000024, RTSEXCPT_IEC_TASK_CONFIG_ERROR := 16#00000025, RTSEXCPT_APP_TARGET_MISMATCH := 16#00000026, RTSEXCPT_ILLEGAL_INSTRUCTION := 16#00000050, RTSEXCPT_ACCESS_VIOLATION := 16#00000051, RTSEXCPT_PRIV_INSTRUCTION := 16#00000052, RTSEXCPT_IN_PAGE_ERROR := 16#00000053, RTSEXCPT_STACK_OVERFLOW := 16#00000054, RTSEXCPT_INVALID_DISPOSITION := 16#00000055, RTSEXCPT_INVALID_HANDLE := 16#00000056, RTSEXCPT_GUARD_PAGE := 16#00000057, RTSEXCPT_DOUBLE_FAULT := 16#00000058, RTSEXCPT_INVALID_OPCODE := 16#00000059, RTSEXCPT_MISALIGNMENT := 16#00000100, RTSEXCPT_ARRAYBOUNDS := 16#00000101, RTSEXCPT_DIVIDEBYZERO := 16#00000102, RTSEXCPT_OVERFLOW := 16#00000103, RTSEXCPT_NONCONTINUABLE := 16#00000104, RTSEXCPT_PROCESSORLOAD_WATCHDOG := 16#00000105, RTSEXCPT_FPU_ERROR := 16#00000150, RTSEXCPT_FPU_DENORMAL_OPERAND := 16#00000151, RTSEXCPT_FPU_DIVIDEBYZERO := 16#00000152, RTSEXCPT_FPU_INEXACT_RESULT := 16#00000153, RTSEXCPT_FPU_INVALID_OPERATION := 16#00000154, RTSEXCPT_FPU_OVERFLOW := 16#00000155, RTSEXCPT_FPU_STACK_CHECK := 16#00000156, RTSEXCPT_FPU_UNDERFLOW := 16#00000157, RTSEXCPT_VENDOR_EXCEPTION_BASE := 16#00002000, RTSEXCPT_USER_EXCEPTION_BASE := 16#00010000) UDINT; END_TYPE </pre>

17.3.62 Operator '__VARINFO'

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

Der Operator liefert Informationen zu einer Variablen zurück. Sie können die Informationen als Datenstruktur in einer Variablen des Datentyps __SYSTEM.VAR_INFO speichern.

Syntax in der Deklaration

```
<name of the info variable> : __SYSTEM.VAR_INFO; // Data
structure for info variable
```

Syntax beim Aufruf

```
<name of the info variable> := __VARINFO( <variable
name> ); // Call of the operator
```

Beispiel

```
FUNCTION_BLOCK FB_Velocity
```

```

VAR_INPUT
  rVelocity: REAL := 1.2;
END_VAR
VAR_OUTPUT
END_VAR
VAR
  infoVelocity: __SYSTEM.VAR_INFO; //Info of Velocity
END_VAR

infoVelocity := __VARINFO(rVelocity); // Gets the info of Velocity locally

PROGRAM PLC_PRG
VAR
  iCounter : INT := 0; // Counts the calls
  infoCounter : __SYSTEM.VAR_INFO; //Info of Counter
  arrA : ARRAY [1..2, 1..2, 1..2] OF INT := [0, 1, 2, 3, 4, 5, 6, 7]; // Stores the A data
  infoA : __SYSTEM.VAR_INFO; //Info of A
  fbVel : FB_Velocity;
END_VAR

iCounter := iCounter + 1;
infoCounter := __VARINFO(iCounter);
infoA := __VARINFO(arrA);
fbVel();

```

Im Applikationscode sind die Variablen `iCounter` und `arrA` bekannt. Die Variableninformationen werden in den Variablen `infoCounter` und `infoA` abgelegt. Außerdem ist der Funktionsbaustein `FB_Velocity` instanziiert.

Datentyp SYSTEM.VAR_INFO

Name	Datentyp	Initialisierung	Beschreibung
ByteAddress	DWORD	0	Adresse der Variablen. Beispiel: 16#072E35EC
ByteOffset	DWORD	0	Offset der Variablenadresse in Bytes. Beispiel: 13936 Bytes. Hinweise: - Wenn die Variable global ist, dann ist der Offset relativ zum Beginn der Area. - Wenn die Variable eine lokale Variable in einer Funktion oder Methode ist, dann ist der Offset relativ zum aktuellen Stack-Frame. - Wenn die Variable eine lokale Variable in einem Funktionsbaustein ist, dann ist der Offset relativ zur Funktionsbaustein-Instanz.
Area	DINT	0	Speicherbereichsnummer Area im Laufzeitsystem. Beispiel: -1. Bedeutet, dass die Variable nicht global im Speicher liegt, sondern relativ zu einer Instanz oder auf dem Stack. Hinweis: Die Speicherbereiche sind geräteabhängig.
BitNr	INT	0	Anzahl der Bits in Bytes. Beispiel: 16#00FF Bytes Hinweis: Wenn die Variable kein ganzzahliger Datentyp ist, gilt: BitNr = -1 = 16#FFFF
BitSize	INT	0	Speicherplatzgröße der Variablen in Bits. Beispiel: 16 Bits

Referenz Programmierung

Name	Datentyp	Initialisierung	Beschreibung
BitAdd-ress	UDINT	0	Bitadresse der Variablen. Voraussetzung: Die Variable liegt im Eingangsspeicherbereich I , Ausgangsspeicherbereich Q oder Merkerspeicherbereich M . Ansonsten ist der Wert undefiniert.
TypeClass	TY-PE_CLASS	TYPE_BOOL	Datentypklasse der Variablen. Beispiel: TYPE_INT, TYPE_ARRAY Hinweis: Bei benutzerdefinierten Datentypen oder Funktionsbaustein-Instanzen wird als Datentypklasse TYPE_USERDEF ausgegeben.
TypeName	STRING(79)	' '	Datentypname der Variablen als STRING(79). Hinweis: Bei benutzerdefinierten Datentypen ist der Funktionsbausteinname oder der DUT-Name angegeben. Beispiel: 'INT', 'ARRAY'
NumElements	UDINT	0	Anzahl der Arrayelemente. Voraussetzung: Die Variable hat den Datentyp ARRAY. Beispiel: 8
BaseType-Class	TY-PE_CLASS	TYPE_BOOL	Elementarer Basisdatentyp der Arrayelemente. Voraussetzung: Die Variable hat den Datentyp ARRAY. Beispiel: TYPE_INT bei arrA : ARRAY [1..2,1..2,1..2] OF INT;
ElemBit-Size	UDINT	0	Speicherplatzgröße des Arrayelements in Bits. Voraussetzung: Die Variable hat den Datentyp ARRAY. Beispiel: 16 Bits bei arrA : ARRAY [1..2,1..2,1..2] OF INT;
MemoryArea	MEMO-RY_AREA	MEM_MEMO-RY	Information zum Speicherbereich: - MEM_GLOBAL: Globaler Speicherbereich Beispielsweise in Area 0 - MEM_LOCAL: Lokaler Speicherbereich in Area -1 - MEM_MEMORY: Merkerspeicherbereich %M Beispielsweise in 16#10 in Area 1 - MEM_INPUT: Eingangsspeicherbereich %I Beispielsweise in 16#04 in Area 2 - MEM_OUTPUT: Ausgangsspeicherbereich %Q Beispielsweise in 16#08 in Area 3 - MEM_RETAIN: Retain-Speicherbereich Beispielsweise in 16#20 in Area 0 Beispiel: MEM_GLOBAL Hinweis: Die Speicherbereichskonfiguration ist geräteabhängig.
Symbol	STRING(39)	' '	Variablenname als STRING(39) Beispiel: 'iCounter', 'arrA'
Comment	STRING(79)	' '	Kommentar der Variablendeklaration Beispiel: 'Counts the calls' oder 'Stores the A data'

Tab. 17-12:

17.3.63 Operator '__CURRENTTASK'

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

Der Operator liefert zur Laufzeit Informationen über die gerade ausgeführte IEC-Task.



Der Operator wird nur auf Zielsystemen unterstützt, bei denen die Zielsystemeinstellung `memory-layout\max-stack-size` auf einen Wert `> 0` gesetzt ist.

Der Operator ermöglicht Zugriff auf eine Struktur mit zwei Variablen:

- `TaskIndex`: Nullbasierter Index, der die Task identifiziert.
- `pTaskInfo`: detaillierte Informationen über die gerade ausgeführte Task, kann einem `POINTER TO Task_Info2` aus der Bibliothek `CmpIecTask` zugewiesen werden.

Der Operator kann nicht in der Deklaration eines Programmbausteins verwendet werden, dies führt zu einer Fehlermeldung. Wenn die aktuelle Task nicht bestimmt werden kann, ist der `TaskIndex` `-1` und die `pTaskInfo` gleich null.

Beispiel

```
//Declaration
VAR
  idx : INT;
  pInfo : POINTER TO Task_Info2;
END_VAR

//Program code

idx := __CURRENTTASK.TaskIndex;
pInfo := __CURRENTTASK.pTaskInfo;
```

17.3.64 Operator '__COMPARE_AND_SWAP'

Der Multicore-Operator ist eine Erweiterung der Norm IEC 61131-3.

Der Operator kann zur Realisierung einer Semaphore verwendet werden, beispielsweise um den exklusiven Zugriff auf eine Variable zu garantieren, die von verschiedenen Tasks geschrieben wird.

Funktionsweise

`__COMPARE_AND_SWAP` bekommt als Eingabe einen Pointer auf eine Variable vom Datentyp `__XWORD`, einen alten Wert und einen neuen Wert (beispielsweise `bMutex := __COMPARE_AND_SWAP(ADR(dwSynch), dwOld, dwNew);`).

Der alte und neue Wert können auch Variablen vom Datentyp `__XWORD` sein.

Die referenzierte `__XWORD`-Variable wird mit dem alten Wert verglichen und wenn beide gleich sind, dann wird der neue Wert geschrieben.

Das Ergebnis der Funktion ist `TRUE`, wenn der neue Wert geschrieben werden konnte.



Der Datentyp `__XWORD` wird vom Compiler automatisch bei 32-Bit-Systemen durch `DWORD` und bei 64-Bit-Systemen durch `LWORD` ersetzt.

Die Operation verläuft atomar, sie kann also nicht von einer anderen Task unterbrochen werden.

Beispiel Das folgende Beispiel zeigt eine typische Verwendung. Es soll ein exklusiver Zugriff auf eine Variable vom Typ `STRING` implementiert werden, die über den Pointer `pstrOutput` angesprochen wird.

Der Zugriff auf einen String ist nicht atomar. Wenn mehrere Tasks gleichzeitig auf denselben String schreiben, dann kann der Inhalt inkonsistent sein.

Mit der im Beispiel gezeigten Funktion ist es möglich in verschiedenen Tasks dieselbe `STRING`-Variable zu schreiben:

```

FUNCTION ExclusiveStringWrite : BOOL
VAR_INPUT
    strToWrite : STRING;
    pstrOutput : POINTER TO STRING;
END_VAR
VAR_STAT
    dwSynch : __XWORD;
END_VAR
VAR
    bMutex: BOOL;
END_VAR

bMutex:= __COMPARE_AND_SWAP(ADR(dwSynch), 0, 1);
(* compare with 0 and write 1 as atomic operation *)
IF bMutex THEN
    pstrOutput^ := strToWrite;    // Now you can write safely on the string
    dwSynch := 0;                // The __XWORD variable must be reset.
    ExclusiveStringWrite := TRUE; // Writing was successful
ELSE
    ExclusiveStringWrite := FALSE; // Writing was not successful
END_IF

```

17.3.65 Operator '`__XADD`'

Der Multicore-Operator ist eine Erweiterung der Norm IEC 61131-3.

Der Operator kann zur Realisierung eines atomaren Zählers verwendet werden.

Funktionsweise Wenn eine Integer-Variable mit einer normalen Addition, zum Beispiel `iTest := iTest + 1`; hochgezählt wird, dann wird diese Operation nicht atomar ausgeführt. Zwischen dem Lesen und dem Schreiben der Variable könnte ein weiterer Zugriff auf die Variable erfolgen.

Wenn der Zähler in mehreren Tasks hochgezählt wird, kann das Ergebnis der Zählung kleiner sein als die Anzahl der Zählvorgänge. Wenn also zwei Tasks den obigen Code einmal ausführen und die Variable zuvor den Wert 0 hatte, dann kann die Variable anschließend den Wert 1 haben. Das ist insbesondere dann problematisch, wenn Arrays in mehreren Tasks bearbeitet werden und in jeder Bearbeitung ein eindeutiger Index für das Array benötigt wird.

Der Operator `__XADD` erhält beim Aufruf als ersten Summanden einen Pointer auf eine Variable vom Typ `DINT` und als zweiten Summanden einen Wert vom Typ `DINT`. `__XADD` liefert den alten Wert des ersten Summanden zurück und addiert im selben Schritt den zweiten Summanden auf den ersten Summanden.

Beispielsweise kann der Aufruf der Funktion wie folgt aussehen: `diOld := __XADD(ADR(diVar), deAdd);`

Beispiel Das folgende Beispiel zeigt eine typische Verwendung. Ein Array soll aus zwei Tasks gefüllt werden, dabei sollen alle Positionen im Array verwendet werden und keine Position überschrieben werden.

Mit der im Beispiel gezeigten Funktion können mehrere Tasks ein boolsches Array füllen:

```

FUNCTION WriteToNextArrayPosition : BOOL
VAR_EXTERNAL
    g_diIndex : DINT; // Index and array are globally defined and used by multiple tasks
    g_boolArray : ARRAY [0..1000] OF BOOL;
END_VAR
VAR_INPUT
    bToWrite : BOOL;
END_VAR
VAR
    diIndex : DWORD;
END_VAR

diIndex := __XADD(ADR(g_diIndex), 1); // Returns a unique index
WriteToNextArrayPosition := FALSE;
IF (diIndex >= 0 AND diIndex <= 1000) THEN
    g_boolArray[diIndex] := bToWrite; //Writes to unique index
    WriteToNextArrayPosition := TRUE; // TRUE: Array was not full yet
END_IF

```

17.3.66 Operator 'TEST_AND_SET'

Der Multicore-Operator ist eine Erweiterung der Norm IEC 61131-3.

Der Operator kann zur Realisierung einer Semaphore verwendet werden. Beispielsweise um einen exklusiven Zugriff auf eine Variable zu garantieren, die von verschiedenen Tasks geschrieben wird.

Funktionsweise

TEST_AND_SET bekommt als Eingabe eine Variable vom Typ DWORD. Für diese Variable muss ein Schreibzugriff möglich sein. Die Variable wird auf 1 gesetzt und der vorherige Wert wird als Ergebnis zurückgeliefert.

Die Operation verläuft atomar, das bedeutet, dass sie nicht von einer anderen Task unterbrochen werden kann. Dies gilt auch auf Multicore-Plattformen.

Der Aufruf im Programm ist beispielsweise `dwOldValue := TEST_AND_SET(dw);`, wobei die Variablen `dwOldValue` und `dw` vom Datentyp `DWORD` sein müssen.

Beispiel

Das folgende Beispiel zeigt eine typische Verwendung. Es soll ein exklusiver Zugriff auf eine Variable vom Typ `STRING` implementiert werden, die über den Pointer `pstrOutput` angesprochen wird. Der Zugriff auf einen String ist nicht atomar. Wenn mehrere Tasks gleichzeitig auf denselben String schreiben, dann kann der Inhalt inkonsistent sein. Mit der Funktion `TEST_AND_SET` ist es nun möglich, in verschiedenen Tasks dieselbe `STRING`-Variable zu schreiben.

```

FUNCTION ExclusiveStringWrite : BOOL
VAR_INPUT
    strToWrite : STRING;
    pstrOutput : POINTER TO STRING;
END_VAR
VAR_STAT
    dwSynch : DWORD;
END_VAR
VAR
    dwOldValue: DWORD;
END_VAR

dwOldValue := TEST_AND_SET(dwSynch); // Write the 1 and read the old value at the same time
IF dwOldValue = 0 THEN                // 0 means: no other task is currently writing
    pstrOutput^ := strToWrite;         // Now you can write safely on the string
    dwSynch := 0;                     // The DWORD must be reset
    ExclusiveStringWrite := TRUE;      // Writing was successful
ELSE
    ExclusiveStringWrite := FALSE;     // Writing was not successful

```

17.3.67 Operator - Globaler Namensraum

Der Operator ist eine 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, sprechen Sie mit .<varname> die globale Variable an.

17.3.68 Operator - Namensraum für Globale Variablenlisten

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

Den Namen einer globalen Variablenliste (GVL) können Sie als Namensraum-Bezeichner für die in der Liste definierten Variablen verwenden. Dadurch ist es möglich, Variablen mit gleichen Namen in verschiedenen globalen Variablenlisten zu verwenden und dennoch eindeutig auf eine bestimmte Variable zuzugreifen. Dem Variablennamen stellen Sie den Namen der globalen Variablenliste getrennt durch einen Punkt . voran.

Syntax

```
<global variable list name>.<variable>
```

Beispiel

ST:

```
globlist1.varx := globlist2.varx;
```

Die globalen Variablenlisten globlist1 und globlist2 enthalten jeweils eine Variable varx.

IndraLogic kopiert die globale Variable varx aus Liste globlist2 in varx aus Liste globlist1.



Wenn Sie eine Variable, die in mehreren globalen Variablenlisten deklariert ist, ohne den vorangestellten Listennamen referenzieren, erscheint eine Fehlermeldung.

17.3.69 Operator - Bibliotheks-Namensraum

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

Syntax:

```
<Bibliotheksnamensraum>.<Bibliotheksbausteinbezeichner>
```

Beispiel:

```
LIB_A.FB_A
```

Ein Bibliotheksbausteinbezeichner wird (als Präfix mit einem Punkt getrennt) um den Bibliotheksnamensraum erweitert, um eindeutig und qualifiziert auf den Bibliotheksbaustein zuzugreifen. Üblicherweise stimmen Namensraum und Name einer Bibliothek überein.

Beispiel:

Eine Bibliothek ist in einem Projekt eingebunden und enthält einen Baustein FB_A. In dem Projekt ist aber bereits lokal ein Funktionsbaustein mit gleichem Namen instanziiert. Bezeichnen Sie den Bibliotheksbaustein mit LIB_A.FB_A, um nicht auf den lokalen Funktionsbaustein sondern auf den Bibliotheksbaustein zuzugreifen.

ST:

```
var1 := FB_A(in := 12); // Call of the project function FB_A
var2 := LIB_A.FB_A(in := 22); // Call of the library function FB_A
```

Sie können für den Namenraum einen anderen Bezeichner definieren. Dafür tragen Sie (als Bibliotheksentwickler beim Erstellen eines Bibliotheksprojekts) in den Projektinformationen einen Namenraum ein. Oder Sie bestimmen (als Applikationsentwickler beim Erstellen einer Applikation) im Bibliotheksverwalter für eine Bibliothek im Dialog "Eigenschaften" einen speziellen Namensraum.

Siehe auch:

- [Kap. 16 "Bibliotheken erzeugen und verwalten" auf Seite 235](#)
- [Kap. 18.1.14 "Objekt 'Bibliotheksverwalter'" auf Seite 548](#)

17.3.70 Operator - Enumerations-Namensraum

Der Operator ist eine Erweiterung der Norm IEC 61131-3.

Sie können den TYPE-Namen einer Enumeration für einen eindeutigen Zugriff auf eine Enumerations-Konstante verwenden. Somit können Sie Konstanten gleichen Namens in mehreren Enumerationsen verwenden.

Der Enumerationsname wird dem Konstantennamen getrennt durch einen Punkt . vorangestellt.

<Enumerationsname>.<Konstantenname>

Beispiel

Die Konstante `Blue` ist sowohl eine Komponente der Enumeration `Colors` als auch eine der Enumeration `Feelings`.

ST:

```
color:= Colors.Blue; //Access to component Blue in enum Colors
```

```
feeling:= Feelings.Blue; //Access to component Blue in enum Feelings
```

17.3.71 Operator 'INI'



Der INI-Operator ist ein Operator, welcher in IndraLogic 1.x verwendet wurde. In IndraLogic 2G ersetzt die "FB_init"-Methode den INI-Operator.

In Projekten, welche aus IndraLogic 1.x importiert wurden, kann dieser Operator weiter verwendet werden

Siehe auch

- [Kap. 17.10 "Methode 'FB_Init', 'FB_Reinitund 'FB_Exit'" auf Seite 455](#)
- [Kap. 17.2.11 "Persistente Variable - PERSISTENT" auf Seite 308](#)

17.4 Operanden

Konstanten

Sie kennzeichnen Konstanten mit dem Schlüsselwort `CONSTANT`. Sie können Konstanten lokal oder global deklarieren.

Syntax:

Programm:

```
VAR CONSTANT
  <Identifier>:<Type> := <Initialization>;
END_VAR
```

- [Kap. 17.4.1 "BOOL-Konstanten" auf Seite 362](#)
- [Kap. 17.4.2 "TIME-Konstanten" auf Seite 362](#)
- [Kap. 17.4.3 "DATE-Konstante" auf Seite 363](#)

- [Kap. 17.4.4 "DATE_AND_TIME-Konstante" auf Seite 364](#)
 - [Kap. 17.4.5 "TIME_OF_DAY-Konstante" auf Seite 365](#)
 - [Kap. 17.4.6 "Zahlenkonstanten" auf Seite 365](#)
 - [Kap. 17.4.7 "REAL/LREAL-Konstante" auf Seite 366](#)
 - [Kap. 17.4.8 "Stringkonstante" auf Seite 367](#)
 - [Kap. 17.4.9 "WSTRING-Konstanten" auf Seite 368](#)
 - [Kap. 17.4.10 "Getypte Konstanten / Typed Literals" auf Seite 369](#)
- Variablen** Sie können Variablen entweder lokal im Deklarationsteil eines Bausteins oder in einer globalen Variablenliste deklarieren.
- An welcher Stelle Sie eine Variable verwenden können, hängt von ihrem Datentyp ab.
- Siehe auch*
- [Kap. 17.4.11 "Zugriff auf Variablen von Arrays, Strukturen und Bausteinen" auf Seite 369](#)
 - [Kap. 17.4.12 "Bitzugriff auf Variablen" auf Seite 369](#)
 - [Kap. "Befehl 'Eingabehilfe'" auf Seite 624](#)
 - [Kap. "Befehl 'Variable deklarieren'" auf Seite 625](#)
- Weitere**
- [Kap. 17.4.13 "Adressen" auf Seite 371](#)
 - [Kap. 17.4.14 "Funktionen" auf Seite 373](#)

17.4.1 BOOL-Konstanten

BOOL-Konstanten sind die Wahrheitswerte `TRUE` (1) und `FALSE` (0).

Siehe auch

- [Kap. 17.5.1 "Datentyp 'BOOL'" auf Seite 375](#)

17.4.2 TIME-Konstanten

Sie können `TIME`-Konstanten benutzen, um die Standard-Timer-Module zu bedienen. Die Konstante hat eine Größe von 32 Bit und somit eine Auflösung in Millisekunden.

Außerdem steht Ihnen die Zeitkonstante `LTIME` als Zeitbasis für hochauflösende Timer zur Verfügung. Die `LTIME`-Konstante hat eine Größe von 64 Bit und somit eine Auflösung in Nanosekunden.

TIME-Konstante *Syntax*

```
<time keyword> # <length of time>
```

```
<time keyword> : TIME | time | T | t
<length of time> : ( <number of days>d )? ( <number of hours>h )? ( <number of minutes>m )?
                  ( <number of seconds>s )? ( <number of milliseconds>ms )? // ( ... )? Optional
```

Die Reihenfolge der Zeiteinheiten darf nicht verändert werden. Es ist jedoch nicht erforderlich, alle Einheiten anzugeben. Es ist erlaubt, die Einheiten in Großbuchstaben zu schreiben.

Zeiteinheiten:

- d: Tage (days)
- h: Stunden (hours)
- m: Minuten
- s: Sekunden
- ms: Millisekunden

Beispiel für eine korrekte ST-Zuweisung einer Zeitkonstanten:

```

VAR
  timLength : TIME := T#14ms;
  timLength1 : TIME := T#100s12ms; // Overflow in the highest unit is allowed.
  timLength2 : TIME := T#12h34m15s;
  timCompare : TIME;
  xIsOK : BOOL;
  timLongest := T#49D17H2M47S295MS; // 4294967295
END_VAR

IF timLength < T#15MS THEN
  IF timCompare < timLength1 THEN
    xIsOK := TRUE;
  END_IF;
END_IF;

```

Unkorrekte Verwendung:

timIncorrect := t#5m68s;	Überlauf bei einer niedrigeren Stelle
timIncorrect1 := 15ms;	Zeitkennung T# fehlt
timIncorrect2 := t#4ms13d;	Unkorrekte Reihenfolge der Zeiteinheiten

*Tab. 17-13:***LTIME-Konstante** *Syntax*

<long time keyword> # <length of high resolution time>

<long time keyword> : LTIME | ltime
 <length of high resolution time> : (<length of time>) (<number of microseconds>us)?
 (<number of nanoseconds>ns)? // (...)? Optional

Sie können für LTIME-Konstanten die gleichen Zeiteinheiten wie für TIME-Konstanten verwenden. Zusätzlich können Sie Mikrosekunden und Nanosekunden angeben, da die Zeitangabe in höherer Zeitauflösung gerechnet wird. Intern werden LTIME-Literale wie der Datentyp LWORD behandelt und der Wert deswegen in Nanosekunden aufgelöst.

Zusätzliche Zeiteinheiten:

- US | us: Mikrosekunden
- NS | ns: Nanosekunden

Beispiel für die korrekte Verwendung einer ST-Zuweisung:

```

PROGRAM PLC_PRG
VAR
  ltimLength := LTIME#1000d15h23m12s34ms2us44ns;
  ltimLength1 := LTIME#3445343m3424732874823ns;
END_VAR

```

Siehe auch

- [Kap. 17.5.5 "Zeitdatentypen" auf Seite 376](#)
- [Kap. 17.4.5 "TIME_OF_DAY-Konstante" auf Seite 365](#)

17.4.3 DATE-Konstante

Verwenden Sie DATE-Konstanten, um Datumsangaben zu machen.

Syntax

```

<date keyword>#<year>-<month>-<day>

<date keyword> : DATE | date | D | d
<Year> : <1970-2106>
<month> : <1-12>
<day> : <1-31>

```

DATE-Literale werden intern wie der Datentyp `DWORD` behandelt, was einer Obergrenze von DATE#2106-2-7 entspricht.

Beispiel

```

PROGRAM PLC_PRG
VAR
    dateStart      : DATE := D#2018-8-8;
    dateEnd        : DATE := d#2018-8-31;
    dateCompare    : DATE := DATE#1996-05-06;
    xIsDuringTheTime: BOOL;
    dateEarliest   : DATE : D#1970-1-1; // = 0
    dateLatest     : DATE : DATE#2106-2-7; // = 4294967295
END_VAR

IF dateStart < dateCompare THEN
    IF dateCompare < dateEnd THEN
        xIsDuringTheTime := TRUE;
    END_IF;
END_IF

```

Siehe auch

- [Kap. 17.5.5 "Zeitdatentypen" auf Seite 376](#)
- [Kap. 17.4.4 "DATE_AND_TIME-Konstante" auf Seite 364](#)
- [Kap. 17.4.5 "TIME_OF_DAY-Konstante" auf Seite 365](#)

17.4.4 DATE_AND_TIME-Konstante

DATE_AND_TIME-Konstanten sind kombinierte Datums- und Uhrzeitangaben.

Syntax

```

<date and time keyword>#<date and time value>

<date and time keyword> : DATE AND TIME | date and time | DT | dt
<date and time value> : <year>-<month>-<day>-<hour>:<minute>:<second>
<year> : <1970-2106>
<month> : <1-12>
<day> : <1-31>
<hour> : <0-24>
<minute> : <0-59>
<second> : <0-59>

```

DATE_AND_TIME-Literale werden intern als Datentyp `DWORD` behandelt. Die Zeit wird folglich in Sekunden verarbeitet und kann Werte von 1. Januar 1970 00:00 Uhr bis 07. Februar 2106 6:28:15 Uhr einnehmen.

Beispiel

```

PROGRAM PLC_PRG
VAR
    dtDate : DATE_AND_TIME := DATE_AND_TIME#1996-05-06-15:36:30;
    dtDate1: DATE_AND_TIME := DT#1972-03-29-00:00:00;
    dtDate2: DATE_AND_TIME := DT#2018-08-08-13:33:20.5;

```

```

dtEarliest : DATE_AND_TIME := DATE_AND_TIME#1979-1-1-00:00:00; // 0
dtLatest   : DATE_AND_TIME := DATE_AND_TIME#2106-2-7-6:28:15; // 4294967295
END_VAR

```

Siehe auch

- [Kap. 17.5.5 "Zeitdatentypen" auf Seite 376](#)
- [Kap. 17.4.3 "DATE-Konstante" auf Seite 363](#)
- [Kap. 17.4.5 "TIME_OF_DAY-Konstante" auf Seite 365](#)

17.4.5 TIME_OF_DAY-Konstante

Sie können mit dieser Konstante die Uhrzeit angeben.

Syntax

```

<time keyword>#<time value>

<time keyword> : TIME_OF_DAY | time_of_day | TOD | tod
<time value>   : <hour>:<minute>:<second>
<hour>         : <0-23>
<minute>       : <0-59>
<second>       : <0.000-59.999>

```

Sie können bei Sekunden auch Sekundenbruchteile angeben.

TIME_OF_DAY-Literale werden intern als `DWORD` behandelt und somit der Wert in Millisekunden aufgelöst.

Beispiel

```

PROGRAM POU
VAR
  todClockTime : TIME_OF_DAY := TIME_OF_DAY#15:36:30.123;
  todEarliest  : TIME_OF_DAY := TIME_OF_DAY#0:0:0.000;
  todLatest    : TOD        := TOD#23:59:59.999;
END_VAR

```

Siehe auch

- [Kap. 17.5.5 "Zeitdatentypen" auf Seite 376](#)
- [Kap. 17.4.3 "DATE-Konstante" auf Seite 363](#)
- [Kap. 17.4.4 "DATE_AND_TIME-Konstante" auf Seite 364](#)

17.4.6 Zahlenkonstanten

Zahlenwerte können als Dualzahlen, Oktalzahlen, Dezimalzahlen und Hexadezimalzahlen dargestellt werden.

Wenn ein Integerwert keine Dezimalzahl ist, dann müssen Sie seine Basis gefolgt von einem Doppelkreuz (#) vor die Integerkonstante schreiben. Die Ziffernwerte für die Zahlen 10 bis 15 bei Hexadezimalzahlen geben Sie wie üblich durch die Buchstaben A-F an.

Sie können Unterstriche innerhalb eines Zahlenwertes verwenden.

Beispiel:

Beispiele:

14	Dezimalzahl
2#1001_0011	Dualzahl

8#67	Oktalzahl
16#A	Hexadezimalzahl
DINT#16#A1	Getypter Datentyp DINT# und Basis 16# kombiniert.

Der Typ dieser Zahlenwerte kann dabei BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL oder LREAL sein.



Implizite Konvertierungen von "größeren" zu "kleineren" Typen sind nicht erlaubt.

Sie können eine DINT-Variable nicht ohne Weiteres als INT-Variable benutzen. Hierfür müssen Sie eine Typkonvertierungsfunktion verwenden.

Siehe auch

- [Kap. 17.3 "Operatoren" auf Seite 315](#)



Da Zahlenkonstanten grundsätzlich als ganzzahlige Werte behandelt werden, müssen Sie in Divisionen die Konstante im Format einer Gleitkommazahl angeben, um den Rest nicht zu verlieren.

Beispiel:

Division 1/10 ergibt 0,

Division 1.0/10 ergibt 0.1.

17.4.7 REAL/LREAL-Konstante

Sie können Gleitkommazahlen als REAL- und LREAL-Konstante in Punkt-schreibweise oder Exponentialschreibweise mit Mantisse und Exponent angeben.

Der Punkt dient als Dezimaltrennzeichen gemäß der International System of Units (English).

Syntax Exponentialdarstellung

```
<significand> e | E <exponent>
```

```
exponent : -44..38 // REAL
exponent : -324..308 // LREAL
```

Beispiel:

REAL-Literal	
7.4	Dezimalzahl Hinweis: 7,4 mit Komma liefert Compilerfehler
1/3.0	Dezimalbruch für 0.333333343 Hinweis: Bei Division von Integertypen bleibt das Ergebnis ein Integertyp. Dabei wird abgeschnitten. Beispielsweise liefert 1/3 als Ergebnis 0
1.64e+009	Exponentialdarstellung

-3.402823e+38	Kleinste Zahl
-1E-44	Größte negative Zahl
1.0E-44	Kleinste positive Zahl
3.402823e+38	Größte Zahl

LREAL-Literal	
-1.7976931348623157E+308	Kleinste Zahl
-4.94065645841247E-324	Größte negative Zahl
4.94065645841247E-324	Kleinste positive Zahl
1.7976931348623157E+308	Größte Zahl

Siehe auch [Kap. 17.5.3 "Datentyp 'REAL/ 'LREAL'" auf Seite 375](#)

17.4.8 Stringkonstante

Eine Stringkonstante ist eine von einfachen Anführungszeichen umschlossene Zeichenfolge. Die Zeichen werden gemäß Zeichensatz ISO/IEC 8859-1 codiert. Eine Stringkonstante kann folglich Leerzeichen und Umlaute enthalten, da die Zeichen teil des Zeichensatzes sind. Sie wird auch als Zeichenliteral oder einfach nur als String bezeichnet.

Beispiel: 'Hello World!'

Wenn in einer Stringkonstanten ein Dollarzeichen (\$) enthalten ist, werden die nachfolgenden zwei Zeichen als Hexadezimalcode gemäß der Codierung in ISO/IEC 8859-1 interpretiert.

Der Code entspricht dem ASCII-Code.

Beachten Sie außerdem die [Tab. "Sonderfälle" auf Seite 367](#)

String mit \$-Code	Interpretation
\$<8-Bit-Code>	8-Bit-Code: Zweistellige hexadezimale Zahl, die gemäß ISO/IEC 8859-1 interpretiert wird.
'\$41'	A
'\$9A'	©
'\$40'	@
'\$0D'	Steuerzeichen Zeilenumbruch, entspricht '\$R'
'\$0A'	Steuerzeichen neue Zeile, entspricht '\$L' und '\$N'

Sonderfälle

String mit \$-Code	Interpretation
'\$L', '\$l'	Steuerzeichen Zeilenvorschub, entspricht '\$0A'
'\$N', '\$n'	Steuerzeichen neue Zeile, entspricht '\$0A'
'\$P', '\$p'	Steuerzeichen Seitenvorschub
'\$R', '\$r'	Steuerzeichen Zeilenumbruch, entspricht '\$0D'
'\$T', '\$t'	Steuerzeichen Tabulator

'\$\$'	Dollarzeichen: \$
'\$'	Einfaches einfaches Anführungszeichen: '

Beispiel:*Konstantendeklaration*

```

VAR CONSTANT
  constA : STRING := 'Hello Allgäu';
  constB : STRING := 'Hello Allgäu $21'; //
Hello Allgäu!
END_VAR

```

Siehe auch:

- [Kap. 17.5.9 "Datentyp 'WSTRING'" auf Seite 382](#)
- [Kap. 17.5.4 "Datentyp 'STRING'" auf Seite 376](#)
- [Kap. 17.4.9 "WSTRING-Konstanten" auf Seite 368](#)

17.4.9 WSTRING-Konstanten

Ein WSTRING ist eine beliebige Zeichenkette in UNI-Code (16 Bit).

WSTRING-Konstanten werden mit Anführungszeichen 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 vier hexadezimalen Ziffern als hexadezimale Darstellung des 16 Bit-Zeichencodes interpretiert. Außerdem werden, Kombinationen von Zeichen, die mit dem Dollarzeichen beginnen, folgendermaßen interpretiert:

angegebene Kombination	Interpretation
"\$<vier hex. Zahlen>"	hexadezimale Darstellung des 16Bit-Zeichen-Codes
" "	Leerzeichen
"\$\$"	Dollarzeichen
"\$"	Einfaches Anführungszeichen
"\$L" oder "\$I"	Zeilenvorschub
"\$N" oder "\$n"	Neue Zeile
"\$P" oder "\$p"	Seitenvorschub
"\$R" oder "\$r"	Zeilenumbruch
"\$T" oder "\$t"	Tabulator

Beispiel:

Gültige Zeichenketten:

"w1Wüß?"

":-)"

"costs (\$\$)"

"Память"

"蓄能器"

Siehe auch:

- [Kap. 17.5.9 "Datentyp 'WSTRING'" auf Seite 382](#)

- [Kap. 17.5.4 "Datentyp 'STRING'" auf Seite 376](#)
- [Kap. 17.4.8 "Stringkonstante" auf Seite 367](#)

17.4.10 Getypte Konstanten / Typed Literals

Mit Ausnahme von REAL/LREAL-Konstanten (hier wird immer LREAL verwendet) verwendet IndraLogic beim Rechnen mit IEC-Konstanten den kleinstmöglichen Datentyp.

Wenn Sie einen anderen Datentyp verwenden wollen, können Sie dies mit Hilfe von Typed Literals (Getypte Konstanten) erreichen, ohne dass Sie die Konstante explizit deklarieren müssen.

Versehen Sie hierbei die Konstante mit einem Präfix, das 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. Sie müssen den Typ in Großbuchstaben schreiben.

`<Literal>` gibt die Konstante an. Die Eingabe muss zum unter `<Type>` angegebenen Datentypen passen.

Beispiel:

```
var1 := DINT#34;
```

Wenn IndraLogic die Konstante nicht ohne Datenverlust in den Zieltyp überführen kann, wird eine Fehlermeldung ausgegeben.

Sie können getypte Konstanten überall dort verwendet, wo Sie normale Konstanten verwenden können.

17.4.11 Zugriff auf Variablen von Arrays, Strukturen und Bausteinen

Schreibweise für den Zugriff auf

- Zweidimensionale Array-Komponenten: `<Array_instance_name>[Index1, Index2]`
- Strukturvariablen: `<Struktur_instance_name>.<Variable_name>`
- Funktionsbaustein- und Programmvariablen: `<Functionblock_instance_name>.<Variable_name>`, `<Program_name>.<Variable_name>`

Siehe auch

- [Kap. 17.5.14 "Datentyp 'ARRAY'" auf Seite 386](#)
- [Kap. 17.5.16 "Datentyp 'STRUCT'" auf Seite 396](#)
- [Kap. "Objekt 'Funktionsbaustein' \(FUNCTION_BLOCK\)" auf Seite 556](#)
- [Kap. "Objekt 'Programm\(PROGRAM\)'" auf Seite 554](#)

17.4.12 Bitzugriff auf Variablen

Wenn gleichzeitige Bitzugriffe durch zwei Tasks ausgeführt werden, obwohl der Prozessor Bitzugriff nicht direkt auf dem Speicher ausführen kann, gehen Sie wie folgt vor:

Verwenden ein Semaphor (`SysSemEnter`) oder eine vergleichbare Technik, um einen konkurrierenden Bitzugriff zu verhindern. Am Besten ist es aber, die Bitzugriffe innerhalb einer Task auszuführen.

Syntax bei Deklarationen

```
<variable name> : <data type> ( := <initialization> )? ;
<data type> = BYTE | WORD | DWORD | LWORD | SINT | USINT | INT | DINT | UDINT | LINT | ULINT
```

Syntax bei Aufrufen

```
<variable name>.<index>
```

Bei einem Bitzugriff wird innerhalb einer ganzzahligen Variablen auf ein einzelnes Bit lesend oder schreibend zugegriffen. Dazu erweitern Sie den Bezeichner der Variablen um einen Punkt und einen Index, der das Bit innerhalb der Variablen adressiert. Die Indizierung ist 0-basiert, der Index ist eine ganze positive Zahl und beginnt bei 0. Sie können den Index auch als konstant Variable angeben.

Beispiel:

```
PROGRAM PLC_PRG
VAR
    wA : WORD := 16#FF;
    xB : BOOL := 0;
END_VAR
wA.2 := xB;
```

Im Programm wird das dritte Bit der Variablen wA auf den Wert der Variablen xB gesetzt.

```
wA = 2#1111_1111_1111_1011 = 16#FFFB
```

Bitzugriff auf eine Variable mit einer globalen konstanten Variablen

Sie können eine globale Konstante als Index verwenden, um innerhalb einer ganzzahligen Variablen auf ein Bit zuzugreifen.

Beispiel: Bitzugriff auf eine ganzzahlige Variable

Deklaration in einer globalen Variablenliste:

```
VAR_GLOBAL CONSTANT
    enable : INT := 2;
END_VAR
```

Die Konstante c_usiENABLE adressiert das Bit innerhalb der ganzzahligen Variablen.

Deklaration einer POU:

```
VAR
    iX : INT;
END_VAR
```

Aufruf mit Bitzugriff:

```
iX.c_usiENABLE := TRUE; // The third bit in the variable iX is set TRUE
```

Bitzugriff mit BIT-Datentypen

Mit dem Datentyp BIT können Sie einzelne Bits zu einer Struktur zusammenfassen und dann einzeln darauf zugreifen. Das Bit wird dabei mit dem Komponentennamen adressiert.

Beispiel: Bitzugriff auf BIT Datentypen

Deklaration der Struktur:

```

TYPE ControllerData :
STRUCT
    Status_OperationEnabled : BIT;
    Status_SwitchOnActive : BIT;
    Status_EnableOperation : BIT;
    Status_Error : BIT;
    Status_VoltageEnabled : BIT;
    Status_QuickStop : BIT;
    Status_SwitchOnLocked : BIT;
    Status_Warning : BIT;
END_STRUCT
END_TYPE

```

Deklaration der POU:

```

VAR
    ControllerDrive1 : ControllerData;
END_VAR

```

Bitzugriff:

```

ControllerDrive1.OperationEnabled := TRUE;

```

Siehe auch

- [Kap. 17.5.2 "Ganzzahlige Datentypen" auf Seite 375](#)
- [Kap. 17.5.10 "Datentyp 'BIT'" auf Seite 382](#)
- [Kap. 17.5.16 "Datentyp 'STRUCT'" auf Seite 396](#)

17.4.13 Adressen

HINWEIS

Wenn Sie Pointer auf Adressen verwenden, können sich beim Online-Change Inhalte von Adressen verschieben!

Wenn Sie direkte Adressen verwenden, verschieben sich Inhalte von Adressen beim Online-Change nicht!

Siehe auch [Kap. "Befehl 'Online-Change'" auf Seite 657](#).

Syntax

```

%<memory area prefix> ( <size prefix> )? <memory position>

```

```

<memory area prefix> : I | Q | M

```

```

<size prefix> : X | B | W | D

```

```

<memory position> : <number> (.<number>)* //Depends on the target system

```

Bei der Angabe einer Adresse werden die Position im Speicher und die Größe mittels spezieller Zeichenfolgen ausgedrückt. Eine Adresse ist gekennzeichnet mit dem Prozentzeichen %, dann folgt der Speicherbereichspräfix, der optionale Größenpräfix und die Speicherposition. Die Nummerierung, mit der Sie die Speicherposition adressieren, ist zielsystemabhängig.

Unterstützte Speicherbereichspräfixe

I	Eingangs-Speicherbereich ("Inputs", physikalische Eingänge über Eingangstreiber, "Sensoren")
Q	Ausgangs-Speicherbereich ("Outputs", physikalische Ausgänge über Ausgangstreiber, "Aktoren")
M	Merker-Speicherbereich

Unterstützte Größenpräfixe:

Größenpräfix	Datentyp	Datenbreite
Kein Größenpräfix	BIT = Bit	Einzelnes Bit
X	BIT = Bit	Einzelnes Bit
B	BYTE = Byte	8 Bits
W	WORD = Word	16 Bits
D	DWORD = Double word	32 Bits
L	LWORD = Long word	64 Bits

Beispiel:

%QX7.5 %Q7.5	Einzelbitadresse des Ausgangsbits 7.5
%IW215	Wortadresse des Eingangsworts 215
%QB7	Bytesdresse des Ausgangsbytes 7
%MD48	Adresse eines Doppelworts an Speicherstelle 48 im Merkerbereich.
%IW2.5.7.1	Wortadresse eines Eingangsworts, Interpretation abhängig von der aktuellen Steuerungskonfiguration.
VAR wVar AT %IW0 : WORD; END_VAR	Variablendeklaration mit Adressangabe eines Eingangsworts Kap. 9.10.2 "AT-Deklaration" auf Seite 130
VAR xActuator AT %QW0 : BOOL; END_VAR	Boolesche Variablendeklaration Hinweis: Für boolesche Variablen wird intern ein Byte alloziert, wenn keine Einzelbitadresse angegeben ist. Eine Wertänderung von xActuator betrifft folglich den Bereich von QX0.0 bis QX0.7.
VAR xSensor AT IX7.5 : BOOL; END_VAR	Boolesche Variablendeklaration mit expliziter Angabe einer Einzelbitadresse. Beim Zugriff wird nur das Eingangsbit 7.5 gelesen.

Speicherposition

So stellen Sie sicher, dass die Adresse gültig ist:

Um in einer Applikation eine gültige Adresse zuzuweisen, muss Ihnen zunächst die geforderte Position im Prozessabbild bekannt sein; das bedeutet der zuständige Speicherbereich: Eingangs- (I), Ausgangs- (Q) oder Merkerspeicherbereich (M), siehe oben. Weiterhin müssen Sie die erforderliche Größe angeben: BIT, BYTE, WORD, DWORD, LWORD(siehe oben: X, B, W, D, L)

Eine entscheidende Rolle spielen die aktuell verwendete Gerätekonfiguration und -Einstellungen (Hardware-Struktur, Gerätebeschreibung, E/A-Einstellungen). Beachten Sie hier besonders die Unterschiede in der Interpretation von Bit-Adressen, die es zwischen den Geräten mit "Byte-Adressierungsmodus" und denen mit "word-orientiertem IEC-Adressierungsmodus" gibt:

Beispielweise spricht das erste Element der Bit-Adresse %IX5.5 in einem byte-adressierenden Gerät Byte 5 an, während es in einem word-adressierenden Gerät Word 5 erreicht.

Dagegen ist die Adressierung mit Word- oder Byte-Adressen unabhängig vom Gerätetyp:

Mit %IW5 wird immer Word 5 angesprochen und mit Byte-Adresse %IB5 immer Byte 5.

Abhängig von der Größe und dem Adressierungsmodus können Sie deshalb mit der gleichen Adressangabe unterschiedliche Speicherzellen ansprechen.

Die nachfolgende Tabelle zeigt die Gegenüberstellung der Byte-Adressierung und Word-orientierten IEC-Adressierung für BIT, BYTE, WORD und DWORD Variablen. Auch zeigt sie die überlappenden Speicherbereiche, die im Falle der Byte-Adressierung vorliegen (siehe auch das Beispiel im Anschluss an die Tabelle).

Beachten Sie bei der Schreibweise, dass der IEC-Adressierungsmodus immer word-orientiert ist. Dabei steht vor dem Punkt die Word-Nummer und nach dem Punkt die Bit-Nummer.

DWords / Words				Bytes	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		W1	B1	X1.7	...	X1.0	X0.15	...	X0.8
...	W2		W2	B2	...			X1.7	...	X1.0
	W3		W3	B3				X1.15	...	X1.8
	W4	D1	W2	B4						
	...		W3	B5						
			W4	B6						
		D2	W4	B7						
		...	...	B8						
		...	...	...						
		...	...	...						
D(n-3)		D(n/4)	...							
	W(n-1)		W(n/2)							
				Bn	Xn.7	...	Xn.0	X(n/2).15	...	X(n/2).8

Abb. 17-71: Gegenüberstellung von byte- oder wort-orientierter Adressierung der Adressformate D, W, B und X

n = Byte-Nummer; word oriented = wortorientiert, addressing = Adressierung

Beispiel:

Speicherbereich-Überlappung im Falle des Byte-Adressierungsmodus

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 Sie W1 bzw. D1, D2, D3 nicht für die Adressierung verwenden!



IndraLogic alloziert boolesche Werte byteweise, wenn Sie nicht explizit eine Einzelbit-Adresse angeben.

Beispiel: Eine Wertänderung von varbool1 AT %QW0 betrifft den Bereich von QX0.0 bis QX0.7.

17.4.14 Funktionen

In ST können Sie einen Funktionsaufruf als Operand verwenden.

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 ST:

```
systemtime := TIME();
```

Siehe auch

- [Kap. "Objekt 'Funktion\(FUNCTION\)" auf Seite 558](#)

17.5 Datentypen

Beim Programmieren in IndraLogic können Sie unterschiedliche Datentypen oder Funktionsbausteintypen verwenden.

Dabei ordnen Sie jedem Bezeichner einen Typ zu.

Der Typ legt fest, wie viel Speicherplatz IndraLogic reserviert und wie diese Werte interpretiert werden.

Folgende Gruppen von Typen stehen Ihnen zur Verfügung:

Standard Datentypen IndraLogic unterstützt alle in der Norm IEC 61131-3 beschriebenen Datentypen.

[Kap. 17.5.1 "Datentyp 'BOOL'" auf Seite 375](#)

[Kap. 17.5.2 "Ganzzahlige Datentypen" auf Seite 375](#)

[Kap. 17.5.3 "Datentyp 'REAL/ 'LREAL'" auf Seite 375](#)

[Kap. 17.5.4 "Datentyp 'STRING'" auf Seite 376](#)

[Kap. 17.5.5 "Zeitdatentypen" auf Seite 376](#)

Erweiterungen zur Norm IEC 61131-3

[Kap. 17.5.6 "Datentyp 'UNION'" auf Seite 377](#)

[Kap. 17.5.7 "Datentyp 'ANY' und 'ANY <type>'" auf Seite 378](#)

[Kap. 17.5.8 "Datentyp 'LTIME'" auf Seite 381](#)

[Kap. 17.5.9 "Datentyp 'WSTRING'" auf Seite 382](#)

[Kap. 17.5.10 "Datentyp 'BIT'" auf Seite 382](#)

[Kap. 17.5.11 "Spezialdatentypen ' __UXINT' und ' __XWORD'" auf Seite 382](#)

[Kap. 17.5.13 "Datentyp 'REFERENCE'" auf Seite 384](#)

[Kap. 17.5.12 "Datentyp 'POINTER'" auf Seite 383](#)

Benutzerdefinierte Datentypen

Benutzerdefinierte Datentypen können Sie auf zwei Arten erzeugen:

- DUT-Objekt unterhalb von "Application" oder im Ordner "Allgemeiner Baustein" anlegen
- Datentyp innerhalb eines Bausteins in der Deklaration anlegen



Beachten Sie bitte die Empfehlungen zur Namensvergabe für Objekte

[Kap. 17.5.14 "Datentyp 'ARRAY'" auf Seite 386](#)

[Kap. 17.5.16 "Datentyp 'STRUCT'" auf Seite 396](#)

[Kap. 17.5.17 "Enumerationen" auf Seite 398](#)

[Kap. 17.5.13 "Datentyp 'REFERENCE'" auf Seite 384](#)

[Kap. 17.5.12 "Datentyp 'POINTER'" auf Seite 383](#)

[Kap. 17.5.18 "Unterbereichstypen" auf Seite 401](#)

[Kap. 17.7 "Bezeichner" auf Seite 445](#)

17.5.1 Datentyp 'BOOL'

Datentyp	Werte	Speicherplatz
BOOL	TRUE (1), FALSE (0)	8 Bit

Siehe auch

- [Kap. 17.4.1 "BOOL-Konstanten" auf Seite 362](#)

17.5.2 Ganzzahlige Datentypen

In IndraLogic stehen Ihnen folgende ganzzahlige Datentypen zur Verfügung.

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
UDINT	0	4294967295	32 Bit
LINT	-2^{63}	$2^{63}-1$	64 Bit
ULINT	0	$2^{64}-1$	64 Bit



Bei Typkonvertierung von größeren zu kleineren Typen können Informationen verloren gehen.

Siehe auch

- [Kap. 17.4.6 "Zahlenkonstanten" auf Seite 365](#)

17.5.3 Datentyp 'REAL' / 'LREAL'

Die Datentypen REAL und LREAL sind Gleitpunkttypen nach IEEE 754.

Sie sind nötig bei der Verwendung von Dezimalzahlen und Gleitpunktzahlen in Punktdarstellung oder Exponentialdarstellung.

Datentyp	Betragsmäßig kleinste Zahl	Betragsmäßig größte Zahl	Speicherplatz
REAL	1.0E-44	3.402823E+38	32 Bit
LREAL	4.94065645841247E-324	1.7976931348623157E+308	64 Bit

Beispiel

```

PROGRAM PLC_PRG
VAR
  rMax: REAL      := 3.402823E+38;      // Largest number
  rPosMin : REAL  := 1.0E-44;          // Smallest positive number
  rNegMax: REAL   := -1.0E-44;         // Largest negative number
  rMin: REAL      := -3.402823E+38;     // Smallest number

  lrMax: LREAL    := 1.7976931348623157E+308; // Largest number
  lrPosMin : LREAL := 4.94065645841247E-324; // Smallest positive number
  lrNegMax: LREAL  := -4.94065645841247E-324; // Largest negative number
  lrMin: LREAL     := -1.7976931348623157E+308; // Smallest number
END_VAR

```



Wenn der Wert der REAL/LREAL-Zahl außerhalb des Wertebereichs des Integers liegt, wird bei einer Datentypkonvertierung von REAL oder LREAL nach SINT, USINT, INT, UINT, DINT, UDINT, LINT oder ULINT ein undefiniertes Ergebnis geliefert. Das Ergebnis ist vom Zielsystem abhängig. Auch ein Ausnahmefehler ist möglich! Um zielsystemunabhängigen Code zu erhalten, muss die Applikation Wertebereichsüberschreitungen abfangen.

Wenn die REAL/LREAL-Zahl innerhalb des Wertebereichs des ganzzahligen Datentyps liegt, arbeitet die Konvertierung auf allen Systemen gleich.

Siehe auch

- [Kap. 17.4.7 "REAL/LREAL-Konstante" auf Seite 366](#)

17.5.4 Datentyp 'STRING'

Eine Variable vom Datentyp STRING kann eine beliebige Zeichenkette aufnehmen. Die Größenangabe zur Speicherplatz-Reservierung bei der Deklaration bezieht sich auf Zeichen und steht in runden oder eckigen Klammern. Ist keine Größe angegeben, nimmt IndraLogic standardmäßig 80 Zeichen an.

IndraLogic begrenzt die String-Länge grundsätzlich nicht, allerdings verarbeitet die String-Funktion nur Längen von 1-255!

Wenn eine Variable mit einem String initialisiert wird, der zu lang ist für den Datentyp der Variablen, schneidet IndraLogic den String von hinten her entsprechend ab.



Der für eine STRING-Variable benötigte Speicherplatz ist immer 1 Byte pro Zeichen + 1 zusätzliches Byte, beispielsweise 81 Bytes im Falle einer "STRING[80]" Deklaration.

Beispiel:

Beispiel einer Stringdeklaration mit 35 Zeichen:

```
str : STRING(35) := 'This is a String';
```

Siehe auch

- [Kap. 17.5.9 "Datentyp 'WSTRING'" auf Seite 382](#)
- [Kap. 17.4.8 "Stringkonstante" auf Seite 367](#)
- [Kap. 17.4.9 "WSTRING-Konstanten" auf Seite 368](#)

17.5.5 Zeitdatentypen

IndraLogic behandelt die Zeitdatentypen intern wie DWORD.

Bei TIME und TIME_OF_DAY führt das zu einer Auflösung in Millisekunden, bei DATE_AND_TIME in Sekunden.

Datentyp	Untergrenze 0	Obergrenze 4294967295	Speicherplatz	Auflösung
TIME	T#0d0h0m0s0ms	T#49d17h2m47s295ms	32 Bit	Millisekunden
TIME_OF_DAY TOD	TOD#0:0:0.000 (00:00:00:000 Uhr)	TOD#23:59:59.999 (23:59:59:999 Uhr)	32 Bit	Millisekunden
DATE	D#1970-1-1 (01.01.1970)	DATE#2106-2-7 (07.Februar 2106)	32 Bit	Tag
DATE_AND_TIME DT	DT#1979-1-1-00:00:00 (01.01.1970, 00:00 Uhr)	DT#2106-2-7-6:28:15 (07.Februar 2106, 6:28:15)	32 Bit	Sekunden

Siehe auch

- [Kap. 17.5.8 "Datentyp 'LTIME'" auf Seite 381](#)
- [Kap. 17.4.2 "TIME-Konstanten" auf Seite 362](#)
- [Kap. 17.4.3 "DATE-Konstante" auf Seite 363](#)
- [Kap. 17.4.4 "DATE_AND_TIME-Konstante" auf Seite 364](#)
- [Kap. 17.4.5 "TIME_OF_DAY-Konstante" auf Seite 365](#)

17.5.6 Datentyp 'UNION'

Ein UNION ist eine Datenstruktur, die meist unterschiedliche Datentypen enthält.

Die Syntax für Uniondeklarationen ist folgende:

SYNTAX:

```

TYPE <union_name>:
  UNION
    <variable_declaration 1>
    ...
    <variable_declaration n>
  END UNION
END_TYPE

```

<union_name> ist der Typname, der im gesamten Gültigkeitsbereich erkannt wird und wie ein Standard-Datentyp verwendet werden kann.

In einer Union haben alle Komponenten die gleiche Basisadresse, d.h. sie belegen denselben Speicherplatz.

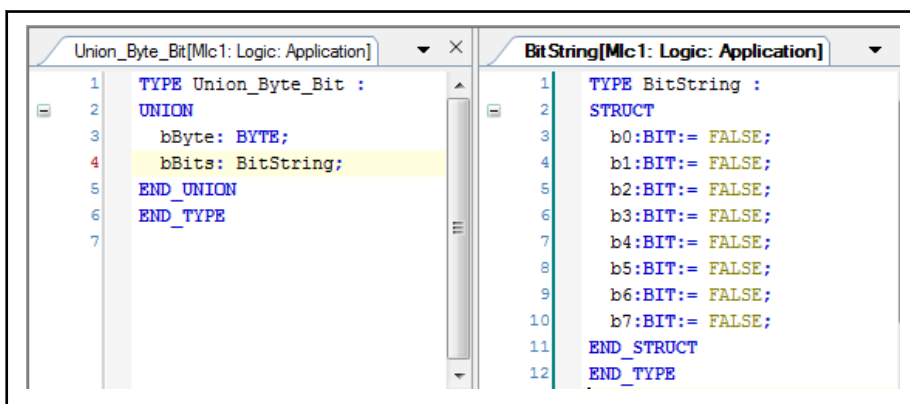


Abb. 17-72: Überlagerung eines Bytes und eines Bitstrings (Struktur)

Weist man dem Byte (union_01.bByte) nun einen Wert zu, ist auch der Bitstring betroffen.

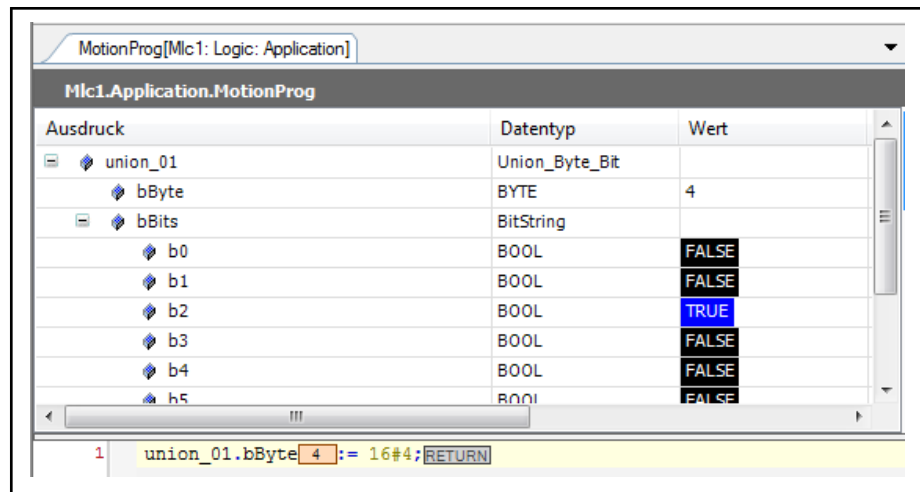


Abb. 17-73: Verwendung einer Union

Ein komplexes Beispiel enthält die Überlagerung eines Aufzählungstypes und einer Struktur.

Der Anwender kann dabei auf die Betriebsart per Enumeration oder über die Bitstruktur zugreifen.

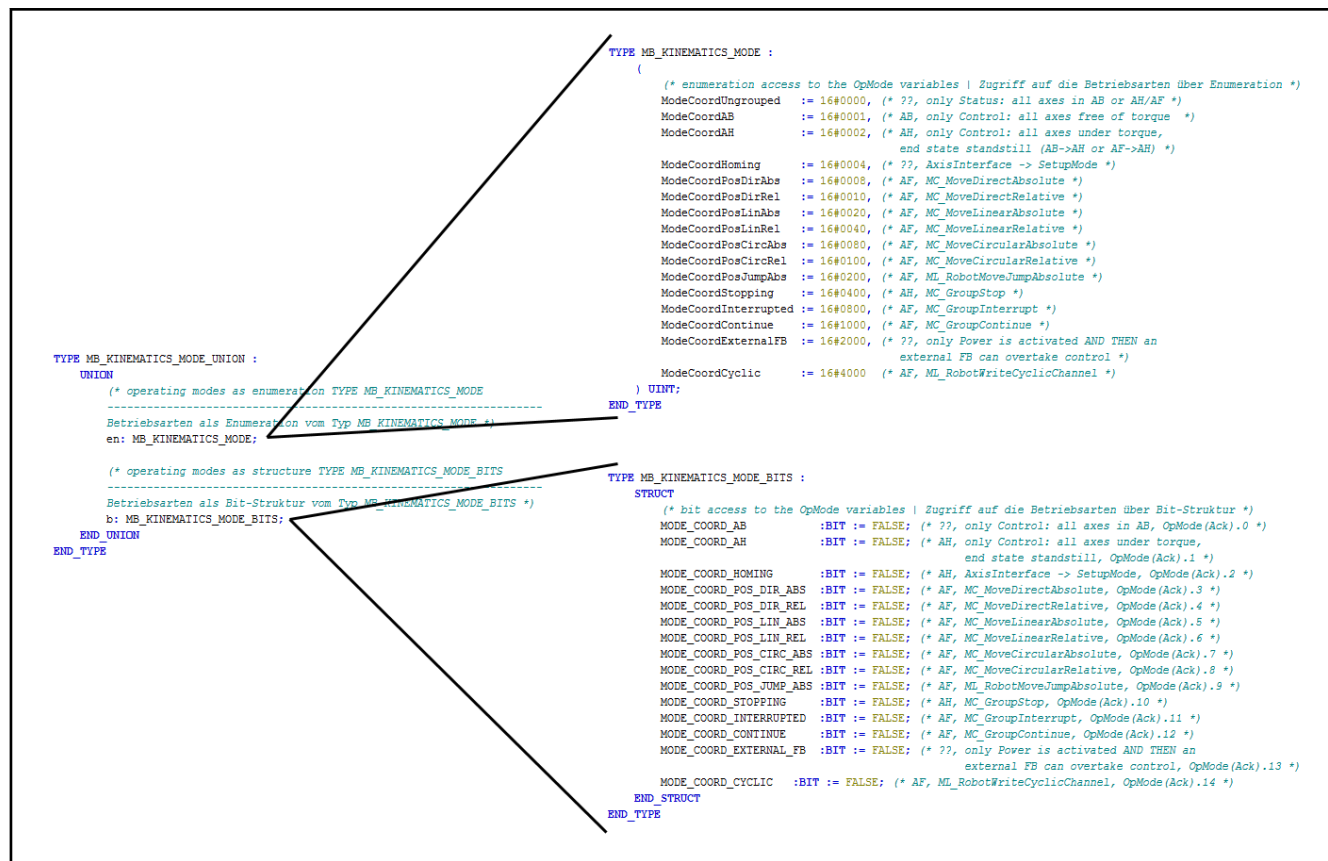


Abb. 17-74: Datentyp MB_KINEMATICS_MODE_UNION mit MB_KINEMATICS_MODE und MB_KINEMATICS_MODE_BITS

17.5.7 Datentyp 'ANY' und 'ANY <type>'

Bei der Implementierung einer Funktion oder einer Methode können Sie Eingänge (VAR_INPUT) als Variablen mit generischem IEC-Datentyp deklarieren.

ren (ANY oder ANY <type>). Folglich können Sie Aufrufe implementieren, deren Aufrufparameter sich beim Datentyp unterscheiden.

Zur Laufzeit können Sie innerhalb des Programmierbausteins für die Eingangsvariable den übergebenen Wert und dessen Typ über eine vordefinierte Struktur abfragen.

Ab Compilerversion >=3.5.1.0 werden die unten dargestellten generischen IEC-Datentypen für die Verwendung bei Funktionseingängen unterstützt. Die Tabelle stellt dar, welcher generischer Datentyp welchen elementaren Datentypen erlaubt.

Hierarchie der generischen Datentypen			Elementare Datentypen
ANY	ANY_BIT		BYTE, WORD, DWORD, LWORD
	ANY_DATE		DATE_AND_TIME, DT, DATE, TIME_OF_DAY, TOD
	ANY_NUM	ANY_REAL	REAL, LREAL
		ANY_INT	USINT, UINT, UDINT, ULINT SINT, INT, DINT, LINT
	ANY_STRING		STRING, WSTRING

Beispiel

```

FUNCTION AnyBitFunc : BOOL
VAR_INPUT
  _value : ANY_BIT;
END_VAR
FUNCTION AnyDateFunc : BOOL
VAR_INPUT
  _value : ANY_DATE;
END_VAR
FUNCTION AnyFunc : BOOL
VAR_INPUT
  _value : ANY;
END_VAR
FUNCTION AnyIntFunc : BOOL
VAR_INPUT
  _value : ANY_INT;
END_VAR
FUNCTION AnyNumFunc : BOOL
VAR_INPUT
  _value : ANY_NUM;
END_VAR
FUNCTION AnyRealFunc : BOOL
VAR_INPUT
  _value : ANY_REAL;
END_VAR
FUNCTION AnyStringFunc : BOOL
VAR_INPUT
  _value : ANY_STRING;
END_VAR

PROGRAM PLC_PRG
VAR
  xValue   : BOOL           := TRUE;
  byValue  : BYTE           := 16#AB;
  wValue   : WORD           := 16#1234;
  dwValue  : DWORD          := 16#6789ABCD;
  lwValue  : LWORD          := 16#0123456789ABCDEF;
  sValue   : STRING         := 'xyz';
  wsValue  : WSTRING        := "abc";
  dtValue  : DATE_AND_TIME  := DT#2017-02-20-11:07:00;
  dValue   : DATE           := D#2017-02-20;
  todValue : TIME_OF_DAY    := TOD#11:07:00;
  rValue   : REAL           := 42.24;
  lrValue  : LREAL          := 24.42;
  usiValue : USINT          := 12;
  uiValue  : UINT           := 1234;
  udiValue : UDINT          := 12345;
  uliValue : ULINT          := 123456;
  siValue  : SINT           := -12;
  iValue   : INT            := -1234;

```

Referenz Programmierung

```

        diValue  : DINT          := -12345;
        liValue  : LINT          := -123456;
END_VAR

AnyFunc(xValue);
AnyFunc(byValue);
AnyFunc(wValue);
AnyFunc(dwValue);
AnyFunc(lwValue);
AnyFunc(sValue);
AnyFunc(wsValue);
AnyFunc(dtValue);
AnyFunc(todValue);
AnyFunc(rValue);
AnyFunc(lrValue);
AnyFunc(usiValue);
AnyFunc(uiValue);
AnyFunc(udiValue);
AnyFunc(uliValue);
AnyFunc(siValue);
AnyFunc(iValue);
AnyFunc(diValue);
AnyFunc(liValue);

AnyBitFunc(byValue);
AnyBitFunc(wValue);
AnyBitFunc(dwValue);
AnyBitFunc(lwValue);

AnyStringFunc(sValue);
AnyStringFunc(wsValue);

AnyDateFunc(dtValue);
AnyDateFunc(dValue);
AnyDateFunc(todValue);

AnyNumFunc(rValue);
AnyNumFunc(lrValue);
AnyNumFunc(usiValue);
AnyNumFunc(uiValue);
AnyNumFunc(udiValue);
AnyNumFunc(uliValue);
AnyNumFunc(siValue);
AnyNumFunc(iValue);
AnyNumFunc(diValue);
AnyNumFunc(liValue);

AnyRealFunc(rValue);
AnyRealFunc(lrValue);

AnyIntFunc(usiValue);
AnyIntFunc(uiValue);
AnyIntFunc(udiValue);
AnyIntFunc(uliValue);
AnyIntFunc(siValue);
AnyIntFunc(iValue);
AnyIntFunc(diValue);
AnyIntFunc(liValue);

```

Wenn die Funktion ANYBIT_TO_BCD aufgerufen wird, kann als Parameter eine Variable mit Datentyp **BOOL**, **BYTE**, **WORD**, **DWORD** oder **LWORD** an die Funktion übergeben werden.

Vordefinierte Struktur

Beim Übersetzen des Codes wird ein ANY-Datentyp intern mit folgender Struktur ersetzt:

Programm:

```

TYPE AnyType :
STRUCT
    // the type of the actual parameter
    typeclass : _SYSTEM.TYPE CLASS ;
    // the pointer to the actual parameter
    pvalue : POINTER TO BYTE;
    // the size of the data, to which the pointer points
    diSize : DINT;
END_STRUCT
END_TYPE

```

Der tatsächliche Aufrufparameter wird den Strukturelementen dann zur Laufzeit zugewiesen.

Beispiel: Hier wird verglichen, ob die 2 übergebenen Variablen den gleichen Typ und den gleichen Wert haben.

Deklarationsteil:

Programm:

```
FUNCTION Generic_Compare : BOOL
VAR_INPUT
    any1 : ANY;
    any2 : ANY;
END_VAR
VAR
    pTest : POINTER TO ARRAY [0..100] OF POINTER TO DWORD;
    iCount: DINT;
END_VAR
```

Programm:

Programm:

```
pTest := ADR(any1);
Generic_Compare := FALSE;
IF any1.typeclass <> any2.typeclass THEN
    RETURN;
END_IF
IF any1.diSize <> any2.diSize THEN
    RETURN;
END_IF
// Byte comparison
FOR iCount := 0 TO any1.diSize-1 DO
    IF any1.pvalue[iCount] <> any2.pvalue[iCount] THEN
        RETURN;
    END_IF
END_FOR
Generic_Compare := TRUE;
RETURN;
// END_FUNCTION
```

17.5.8 Datentyp 'LTIME'

Sie können den Datentyp LTIME als Zeitbasis für hochauflösende Timer verwenden.

Ein hochauflösender Timer hat eine Auflösung in Nanosekunden.

Datentyp	Untergrenze	Obergrenze	Speicherplatz
LTIME	0	213503d23h34m33s709ms551us615ns	64 Bit

Syntax:

LTIME#<Zeitdeklaration>

Die Zeitdeklaration kann die Zeiteinheiten enthalten, die für die TIME Konstante gelten, und zusätzlich:

- "µs" : Mikrosekunden
- "ns" : Nanosekunden

Beispiel:

```
LTIME1 := LTIME#1000d15h23m12s34ms2us44ns
```

Siehe auch

- [Kap. 17.5.5 "Zeitdatentypen" auf Seite 376](#)

17.5.9 Datentyp 'WSTRING'

Der Datentyp `WSTRING` ist im Gegensatz zum Datentyp `STRING` (ASCII) in Unicode-Format interpretiert. Durch diese Codierung hängt bei `WSTRING` die Anzahl der darstellbaren Zeichen von den Zeichen ab. Eine Länge von 10 heißt bei `WSTRING`, dass die Länge des `WSTRING` maximal 10 `WORD` belegen kann. In Unicode werden aber für einige Zeichen mehrere `WORD` für die Codierung eines Zeichens benötigt, so dass die Anzahl der Zeichen nicht der Länge des `WSTRING` (hier 10) entsprechen muss. Der Datentyp benötigt 1 `WORD` pro Character und 1 `WORD` extra an Speicherplatz. Ein `STRING` benötigt jeweils nur 1 Byte. Der Datentyp `WSTRING` ist mit einer 0 terminiert.

Beispiel:

```
wstr : WSTRING := "This is a WString";
```

Siehe auch

- [Kap. 17.5.4 "Datentyp 'STRING'" auf Seite 376](#)
- [Kap. 17.4.8 "Stringkonstante" auf Seite 367](#)
- [Kap. 17.4.9 "WSTRING-Konstanten" auf Seite 368](#)

17.5.10 Datentyp 'BIT'

Sie können den Datentyp `BIT` nur für einzelne Variablen innerhalb von Strukturen oder in einem Funktionsbaustein verwenden.

Die möglichen Werte sind `TRUE` (1) und `FALSE` (0).

Ein `BIT`-Element benötigt 1 Bit Speicherplatz und Sie können damit einzelne Bits einer Struktur über ihren Namen ansprechen.

`BIT`-Elemente, die nacheinander deklariert sind, werden in Bytes zusammengefasst.

Dadurch können Sie im Vergleich zu `BOOL`-Typen, die jeweils mindestens 8 Bits belegen, den Speicherverbrauch optimieren.

Der Bitzugriff benötigt wesentlich mehr Zeit. Deshalb sollten Sie den Datentyp `BIT` nur verwenden, wenn Sie die Daten in genau vorgegebenem Format definieren wollen.

Siehe auch

- [Kap. 17.5.16 "Datentyp 'STRUCT'" auf Seite 396](#)

17.5.11 Spezialdatentypen '__UXINT' und '__XWORD'

IndraLogic unterstützt Systeme mit Adressregistern von 32 Bit Breite und 64 Bit Breite.

Um den IEC-Code soweit als möglich vom Zielsystem unabhängig zu machen, können Sie die "Pseudo"-Datentypen `__UXINT` und `__XWORD` verwenden.

Der Compiler prüft, welcher Zielsystemtyp aktuell verwendet wird und konvertiert diese Datentypen entsprechend in die passenden Standard-Datentypen.

`__UXINT` wird auf 64-Bit Plattformen zu `ULINT` konvertiert, auf 32-Bit Plattformen zu `UDINT`.

`__XWORD` wird auf 64-Bit Plattformen zu `LWORD` konvertiert, auf 32-Bit Plattformen zu `DWORD`.

17.5.12 Datentyp 'POINTER'

Pointer speichern die Adressen von Variablen, Programmen, Funktionsbausteinen, Methoden und Funktionen, während ein Applikationsprogramm läuft. Ein Pointer zeigt auf eines der genannten Objekte oder auf eine Variable mit beliebigem Datentyp.

Syntax einer Pointer-Deklaration:

<Kennzeichner>: POINTER TO <Datentyp | Funktionsbaustein | Programm | Methode | Funktion>;

Beim Dereferenzieren eines Pointers wird der Wert der Adresse ermittelt, auf die der Pointer zeigt. Um einen Pointer zu dereferenzieren, hängen Sie den Inhaltsoperator an den Pointer-Bezeichner an, siehe `pt^` im unten gezeigten Beispiel.

Mit dem Adressoperator `ADR` können Sie einem Pointer die Adresse einer Variablen zuweisen.

Beispiel:

```
VAR
  pt:POINTER TO INT;  (* declaration of pointer pt *)
  var_int1:INT := 5;  (* declaration of variables var_int1 and var_int2 *)
  var_int2:INT;
END_VAR

pt := ADR(var_int1); (* pointer pt is assigned to address of var_int1 *)
var_int2:= pt^;      (* value 5 of var_int1 is assigned to variable var_int2 by dereferencing of pointer pt *)
```



Wenn ein Pointer auf einen Geräteeingang verwendet wird, gilt der Zugriff (beispielsweise `pTest := ADR(input);`) als schreibender Zugriff.

Dies führt bei der Codeerzeugung zu einer Compilerwarnung "...kein gültiges Zuordnungsziel".

Wenn Sie ein Konstrukt dieser Art benötigen, müssen Sie den Eingangswert (`input`) zuerst auf eine Variable mit Schreibzugriff kopieren.

Siehe auch

- [Kap. 17.3.31 "Operator 'Inhaltsoperator'" auf Seite 333](#)
- [Kap. 17.3.30 "Operator 'ADR'" auf Seite 333](#)

Funktionszeiger auf externe Funktionen

IndraLogic unterstützt Funktionszeiger, die den `INDEXOF` Operator ersetzen. Sie können diese Zeiger an externe Bibliotheken weiterreichen. IndraLogic bietet jedoch keine Möglichkeit, einen Funktionszeiger innerhalb einer Applikation im Programmiersystem aufzurufen!

Die Funktion des Laufzeitsystems zur Registrierung von Callback-Funktionen (Systembibliotheksfunktion) erwartet den Funktionszeiger. Je nachdem welcher Callback registriert wurde, ruft das Laufzeitsystem implizit die betreffende Funktion auf (zum Beispiel bei STOP).

Damit ein solcher Systemaufruf möglich ist (Laufzeitsystem), müssen Sie die entsprechende Objekteigenschaft in der Registerkarte **Übersetzen** setzen.

Sie können den `ADR`-Operator für Funktionen, Programme, Funktionsbausteine und Methoden verwenden. Da Funktionen ihren Wert nach einem Online-Change ändern können, gibt IndraLogic nicht die Adresse der Funktion aus,

sondern die Adresse eines Pointers auf die Funktion. Diese Adresse gilt solange die Funktion auf dem Zielsystem existiert.

Siehe auch

- [Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749](#)
- [Kap. 17.3.30 "Operator 'ADR'" auf Seite 333](#)

Indezzugriff auf Pointer

In IndraLogic ist der Indezzugriff "[]" auf Variablen vom Typ `POINTER`, `STRING` oder `WSTRING` zulässig.

- `pint[i]` gibt den Basisdatentyp zurück
- Der Indezzugriff auf Pointer erfolgt arithmetisch:
Wenn Sie den Indezzugriff bei einer Variablen vom Typ `POINTER TO` verwenden, berechnet IndraLogic den Offset durch
$$\text{pint}[i] = (\text{pint} + i * \text{SIZEOF}(\text{base type}))^\wedge.$$

Der Indezzugriff bewirkt auch eine implizite Dereferenzierung des Pointers. Der resultierende Datentyp ist der Basisdatentyp des Pointers.
Beachten Sie, dass $\text{pint}[7] \neq (\text{pint} + 7)^\wedge!$
- Wenn Sie den Indezzugriff bei einer Variablen vom Typ `STRING` verwenden, erhalten Sie das Zeichen am Offset des Indexausdrucks.
Das Ergebnis ist vom Typ `BYTE`. `str[i]` und gibt das "i-te" Zeichen der Zeichenkette als `SINT` (ASCII) zurück.
- Wenn Sie den Indezzugriff bei einer Variablen vom Typ `WSTRING` verwenden, erhalten Sie das Zeichen am Offset des Indexausdrucks.
Das Ergebnis ist vom Typ `WORD`. `wstr[i]` gibt das "i-te" Zeichen der Zeichenkette als `INT` (Unicode) zurück.



Das Ergebnis der Differenz zweier Pointer ist vom Typ `DWORD`, auch auf 64-Bit-Plattformen, wenn die Pointer 64-Bit-Pointer sind.



Beachten Sie die Möglichkeit, Referenzen zu verwenden, die im Unterschied zu Zeigern einen Wert direkt steuern.



Beachten Sie die Möglichkeit, den Speicherzugriff von Zeigern während der Laufzeit durch die implizite Überwachungsfunktion `CheckPointer` zu überwachen

Siehe auch

- [Kap. 17.3.30 "Operator 'ADR'" auf Seite 333](#)
- [Kap. 17.5.13 "Datentyp 'REFERENCE'" auf Seite 384](#)
- ["CheckPointer" auf Seite 579](#)

17.5.13 Datentyp 'REFERENCE'

Eine `REFERENCE` ist ebenfalls ein Pointer, hat jedoch gegenüber dem `POINTER` einige Vorteile:

- Einfachere Nutzung: Die Referenz muss nicht extra dereferenziert werden (mit $^\wedge$), um auf die Inhalte des referenzierten Objekts zuzugreifen.
- Schöner Syntax bei der Übergabe von Werten: Ist ein Input ein `REFERENCE TO` dann muss man bei `a(refInput := value)` nicht extra `ADR(value)` schreiben.

- **Typsicherheit:** Im Gegensatz zum Pointer prüft der Compiler bei Referenzen, ob bei der Zuweisung von zwei Referenzen der Basistyp gleich ist.

Sie deklarieren eine Referenz gemäß folgender Syntax:

Syntax

<Bezeichner> : REFERENCE TO <Datentyp>

Beispiel

```
A : REFERENCE TO DUT;
B : DUT;
C : DUT;
```

```
A REF= B; // corresponds to A := ADR(B);
A := C; // corresponds to A^ := C;
```



Sie können Referenzen nicht folgendermaßen deklarieren:

- REFERENCE TO REFERENCE
- oder ARRAY OF REFERENCE
- oder POINTER TO REFERENCE.

Weiterhin können Sie eine Referenz nicht auf eine Bit-Variable deklarieren.



Mit Compilerversion >= V3.3.0.0 initialisiert IndraLogic Referenzen (mit 0).



Wenn eine Referenz auf einen Geräteeingang verwendet wird, gilt der Zugriff (beispielsweise `ref REF= input;`) als schreibender Zugriff. Dies führt bei der Codeerzeugung zu einer Compilerwarnung "...kein gültiges Zuordnungsziel".

Wenn Sie ein Konstrukt dieser Art benötigen, müssen Sie den Eingangswert (`input`) zuerst auf eine Variable mit Schreibzugriff kopieren

Prüfung auf gültige Referenzen

Sie können den Operator `__ISVALIDREF` verwenden, um zu prüfen, ob eine Referenz auf einen gültigen Wert weist, das heißt auf einen Wert ungleich 0.

Syntax:

<boolesche Variable> := `__ISVALIDREF`(mit REFERENCE TO <datatype> deklarierter Kennzeichner);

<boolesche 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= ivar;
ref_int0 REF= 0;
testref := __ISVALIDREF(ref_int);
(* becomes TRUE, because ref_int points to ivar, which is non-zero *)
testref := __ISVALIDREF(ref_int0);
(* becomes FALSE, because ref_int0 is set to 0 *)

```



Die implizite Überwachungsfunktion "Checkpoint" wirkt ab Compiler Version 3.5.7.40 auch auf Variablen vom Typ REFERENCE in gleicher Weise wie auf Zeigervariablen (Kap. 18.1.18 "Überwachung des SPS-Programms der Applikation während der Laufzeit" auf Seite 576).

Siehe auch:

- [Kap. 17.3.57 "Operator '__ISVALIDREF'" auf Seite 349](#)
- [Kap. 17.3.30 "Operator 'ADR'" auf Seite 333](#)
- [Kap. 17.5.12 "Datentyp 'POINTER'" auf Seite 383](#)

17.5.14 Datentyp 'ARRAY'

Ein Array ist eine Sammlung von Datenelementen des gleichen Datentyps. IndraLogic unterstützt ein- und mehrdimensionale Arrays von fester oder variabler Länge.

Array mit fester Länge

Sie können Arrays im Deklarationsteil einer POU oder in globalen Variablenlisten deklarieren.

Syntax der Deklaration eines eindimensionalen Arrays

```

<variable_Name> : ARRAY[ <dimension> ] OF <data type> ( := <initialization> )? ;
<dimension> : <lower index bound>..<upper index bound>
<data type> : elementary data types | user defined data types | function block types
// (...)?: Optional

```

Syntax der Deklaration eines mehrdimensionalen Arrays

```

<variable_Name> : ARRAY[ <1st dimension> (, <next dimension> )+ ] OF <data type> ( := <initialization> )? ;
<1st dimension> : <1st lower index bound>..<1st upper index bound>
<next dimension> : <next lower index bound>..<next upper index bound>
<data type> : elementary data types | user defined data types | function block types
// (...) : One or more further dimensions
// (...)?: Optional

```

Die Indexgrenzen sind ganzzahlige Zahlen, maximal des Datentyps DINT.

Syntax bei Datenzugriff

```

<variable_Name>[ <index of 1st dimension> (, <index of next dimension> )* ]
// (...) : 0, one or more further dimensions

```



Beachten Sie die Möglichkeit, während der Laufzeit die Einhaltung der Indexgrenzen durch die implizite Überwachungsfunktion CheckBounds () zu kontrollieren.

Beispiel: Eindimensionales Array von 10 Integerelementen*Deklaration*

```
VAR
```

```

    aiCounter : ARRAY[0..9] OF INT;
END_VAR

```

Untere Indexgrenze: 0

Obere Indexgrenze: 9

Initialisierung

```

aiCounter : ARRAY[0..9] OF INT := [0, 10, 20, 30, 40, 50, 60, 70, 80, 90];

```

Datenzugriff

```

iLocalVariable := aiCounter[2];

```

Der lokalen Variablen wird der Wert 20 zugewiesen.

Beispiel: 2-dimensionales Array

Deklaration

```

VAR
    aiCardGame : ARRAY[1..2, 3..4] OF INT;
END_VAR

```

1te Dimension: 1 bis 2

2te Dimension: 3 bis 4

Initialisierung

```

aiCardGame : ARRAY[1..2, 3..4] OF INT := [2(10), 2(20)];
// Short notation for [10, 10, 20, 20]

```

Datenzugriff

```

iLocal_1 := aiCardGame[1, 3]; // Assignment of 10
iLocal_2 := aiCardGame[2, 4]; // Assignment of 20

```

Beispiel: 3-dimensionales Array

Deklaration

```

VAR
    aiCardGame : ARRAY[1..2, 3..4, 5..6] OF INT;
END_VAR

```

1te Dimension: 1 bis 2

2te Dimension: 3 bis 4

3te Dimension: 5 bis 6

$2 * 2 * 2 = 8$ Arrayelemente

Initialisierung

```

aiCardGame : ARRAY[1..2, 3..4, 5..6] OF INT := [10, 20, 30, 40, 50, 60, 70, 80];

```

Datenzugriff

```

iLocal_1 := aiCardGame[1, 3, 5]; // Assignment of 10
iLocal_2 := aiCardGame[2, 3, 5]; // Assignment of 20
iLocal_3 := aiCardGame[1, 4, 5]; // Assignment of 30
iLocal_4 := aiCardGame[2, 4, 5]; // Assignment of 40
iLocal_5 := aiCardGame[1, 3, 6]; // Assignment of 50
iLocal_6 := aiCardGame[2, 3, 6]; // Assignment of 60
iLocal_7 := aiCardGame[1, 4, 6]; // Assignment of 70
iLocal_8 := aiCardGame[2, 4, 6]; // Assignment of 80

```

Initialisierung

```

aiCardGame : ARRAY[1..2, 3..4, 5..6] OF INT := [2(10), 2(20), 2(30), 2(40)];
// Short notation for [10, 10, 20, 20, 30, 30, 40, 40]

```

Datenzugriff

```

iLocal_1 := aiCardGame[1, 3, 5]; // Assignment of 10
iLocal_2 := aiCardGame[2, 3, 5]; // Assignment of 10
iLocal_3 := aiCardGame[1, 4, 5]; // Assignment of 20
iLocal_4 := aiCardGame[2, 4, 5]; // Assignment of 20
iLocal_5 := aiCardGame[1, 3, 6]; // Assignment of 30
iLocal_6 := aiCardGame[2, 3, 6]; // Assignment of 30
iLocal_7 := aiCardGame[1, 4, 6]; // Assignment of 40
iLocal_8 := aiCardGame[2, 4, 6]; // Assignment of 40

```

Beispiel: 3-dimensionalen Arrays einer benutzerdefinierten Struktur

```

TYPE DATA_A
STRUCT
  iA_1 : INT;
  iA_2 : INT;
  dwA_3 : DWORD;
END_STRUCT
END_TYPE

PROGRAM PLC_PRG
VAR
  aData_A : ARRAY[1..3, 1..3, 1..10] OF DATA_A;
END_VAR

```

Das Array aData_A besteht aus insgesamt $3 * 3 * 10 = 90$ Arrayelementen des Datentyps DATA_A.

Teilweise initialisieren

```

aData_A : ARRAY[1..3, 1..3, 1..10] OF DATA_A := [(iA_1 := 1, iA_2 := 10, dwA_3 := 16#00FF),
(iA_1 := 2, iA_2 := 20, dwA_3 := 16#FF00), (iA_1 := 3, iA_2 := 30, dwA_3 := 16#FFFF)];

```

Im Beispiel werden nur der ersten 3 Elemente explizit initialisiert. Elemente, denen explizit kein Initialisierungswert zugewiesen wird, werden intern mit dem Standardwert des Basisdatentyps initialisiert. Somit werden die Strukturkomponenten ab dem Element aData_A[2, 1, 1] mit 0 initialisiert.

Datenzugriff

```

iLocal_1 := aData_A[1,1,1].iA_1; // Assignment of 1
dwLocal_2 := aData_A[3,1,1].dwA_3; // Assignment of 16#FFFF

```

Beispiel: Array eines Funktionsbausteins

```

FUNCTION BLOCK FBOBJECT_A
VAR
    iCounter : INT;
END_VAR

PROGRAM PLC_PRG
VAR
    aObject_A : ARRAY[1..4] OF FBOBJECT_A;
END_VAR

```

Das Array aObject_A besteht aus 4 Elementen. Jedes Element instanziert einen Funktionsbaustein FBOBJECT_A.

Funktionsaufruf

```
aObject_A[2]();
```

Beispiel: Implementierung von FB_Something mit Methode FB_Init

```

FUNCTION_BLOCK FB_Something
VAR
    _nId : INT;
    _lrIn : LREAL;
END_VAR

METHOD FB_Init : BOOL
VAR_INPUT
    bInitRetains : BOOL;
    bInCopyCode : BOOL;
    nId : INT;
    lrIn : LREAL;
END_VAR

    _nId := nId;
    _lrIn := lrIn;

```

Der Funktionsbaustein FB_Something hat eine Methode FB_Init, die 2 Parameter benötigt.

Instanziierung des Arrays mit Initialisierung

```

PROGRAM PLC_PRG
VAR
    fb_Something_1 : FB_Something(nId := 11, lrIn := 33.44);
    a_Something : ARRAY[0..1, 0..1] OF FB_Something[(nId := 12, lrIn := 11.22),
        (nId := 13, lrIn := 22.33), (nId := 14, lrIn := 33.55), (nId := 15, lrIn := 11.22)];
END_VAR

```

"Array von Arrays"

Die Deklaration eines "Array von Arrays" ist eine alternative Schreibweise für multidimensionale Arrays. Eine Sammlung von Elementen wird dabei verschachtelt, statt die Elemente zu dimensionieren. Die Schachteltiefe kann beliebig sein.

Syntax bei Deklaration

```

<variable name> : ARRAY[<first>] ( OF ARRAY[<next>] )+ OF <data type> ( := <initialization> )? ;

<first> : <first lower index bound>..<first upper index bound>
<next> : <lower index bound>..<upper index bound> // one or more arrays
<data type> : elementary data types | user defined data types | function block types
// (...) + : One or more further arrays
// (...) ? : Optional

```

Syntax bei Datenzugriff

```
<variable name>[<index of first array>] ( [<index of next array>] )+ ;
// (...) * : 0, one or more further arrays
```

Beispiel

```
PROGRAM PLC_PRG
VAR
  aiPoints : ARRAY[1..2,1..3] OF INT := [1,2,3,4,5,6];
  ai2Boxes : ARRAY[1..2] OF ARRAY[1..3] OF INT := [ [1, 2, 3], [ 4, 5, 6] ];
  ai3Boxes : ARRAY[1..2] OF ARRAY[1..3] OF ARRAY[1..4] OF INT :=
    [ [ [1, 2, 3, 4], [5, 6, 7, 8] ], [9, 10, 11, 12] ],
    [ [13, 14, 15, 16], [ 17, 18, 19, 20], [21, 22, 23, 24] ] ];
  ai4Boxes : ARRAY[1..2] OF ARRAY[1..3] OF ARRAY[1..4] OF ARRAY[1..5] OF INT;
END_VAR

aiPoints[1, 2] := 1200;
ai2Boxes[1][2] := 1200;
```

Die Variablen `aiPoints` und `ai2Boxes` sammeln die gleichen Datenelemente, aber die Schreibweisen bei der Deklaration und beim Datenzugriff unterscheiden sich.

Array mit variabler Länge

In Funktionsbausteinen, Funktionen oder Methoden können Sie im Deklarationsabschnitt `VAR_IN_OUT` Arrays mit variabler Länge deklarieren. Um zur Laufzeit die Indexgrenzen des tatsächlich verwendeten Arrays zu ermitteln, stehen Ihnen die Operatoren `LOWER_BOUND` und `UPPER_BOUND` zur Verfügung.

Syntax der Deklaration eines eindimensionalen Arrays mit variabler Länge

```
<variable name> : ARRAY[*] OF <data type> ( := <initialization> )? ;

<data type> : elementary data types | user defined data types | function block types
// (...) ? : Optional
```

Syntax der Deklaration eines mehrdimensionalen Arrays mit variabler Länge

```
<variable name> : ARRAY[* (, *)+ ] OF <data type> ( := <initialization> )? ;

<data type> : elementary data types | user defined data types | function block types
// (...) + : One or more further dimensions
// (...) ? : Optional
```

Syntax der Operatoren zur Grenzindexberechnung

```
LOWER_BOUND( <variable name> , <dimension number> )
UPPER_BOUND( <variable name> , <dimension number> )
```

Beispiel:

Die Funktion `SUM` addiert die Integerwerte der Arrayelemente und gibt als Ergebnis die ermittelte Summe zurück. Die Summe wird über alle zur Laufzeit vorhandenen Arrayelemente berechnet. Da die tatsächliche Anzahl an Arrayelementen erst zur Laufzeit bekannt sein wird, wird die lokale Variable als eindimensionales Array variabler Länge deklariert.

```
FUNCTION SUM: INT;
VAR_IN_OUT
```

```

    aiData : ARRAY[*] OF INT;
END_VAR
VAR
    diCounter, diResult : DINT;
END_VAR

diResult := 0;
FOR diCounter := LOWER_BOUND(aiData, 1) TO UPPER_BOUND(aiData, 1) DO
    // Calculates the length of the current array
    diResult := diResult + A[i];
END_FOR;
SUM := diResult;

```

Siehe auch

- ["Checkbounds" auf Seite 578](#)
- [Kap. 9.2.3 "Array deklarieren" auf Seite 89](#)

17.5.15 Datentyp '__VECTOR'



Vektoroperationen werden nur auf 64-Bit-Prozessoren nativ unterstützt und bieten nur auf diesen einen Geschwindigkeitsvorteil. Das Datenblatt der Steuerung informiert Sie über den auf der Steuerung verwendeten Prozessor.

Aktuell werden Vektoroperationen auf den Plattformen x86-64Bit mit SSE2 und ARM64 mit NEON nativ unterstützt.

Auf allen anderen Plattformen werden Vektoroperationen in einzelne Anweisungen übersetzt. Beispielsweise wird dann eine Vektoraddition mit mehreren einzelnen Additionen ausgeführt.

Die Befehlssatzerweiterungen der Prozessoren sind SIMD-Erweiterungen. SIMD (Single Instruction, Multiple Data) steht für eine Rechnerarchitektur, bei der mit einem Befehlsaufruf gleichzeitig mehrere gleichartige Datensätze parallel und damit schneller verarbeitet werden. Bei Vektoroperationen können dann beispielsweise 4 Zahlenpaare gleichzeitig addiert werden.

Syntax

```

<variable name> : __VECTOR[ <vector size> ] OF <element type> ( := <initialization> )? ;

<vector size> : 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8
<element type> : REAL | LREAL
// (...)? : Optional

```

Ein Vektordatentyp ist ein Gleitpunktzahl-Array mit maximal 8 Elementen. Für diesen Datentyp stehen die Operatoren `__vc<operator name>` zur Verfügung. Damit können Sie Vektoroperationen ohne zusätzliche Funktionsaufrufe implementieren.

Syntax bei Indexzugriff

```

<variable name>[ <index> ]
<index> : 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

```

Beim Indexzugriff auf eine Vektorvariable können Sie auf ein einzelnes Element des Vektors zugreifen. Der Index beginnt bei 0 und geht bis `<vector size> - 1`.

Beispiel

```

PROGRAM PLC_PRG
VAR
    vcA : __VECTOR[3] OF REAL;
END_VAR

```

```
vcA[0] := 1.1;
vcA[1] := 2.2;
vcA[2] := 3.3;
```

Optimale Vektorgröße bestimmen



Verwenden Sie als Vektorgröße die optimale, von Ihrem Zielsystem abhängige Vektorgröße, um möglichst effizienten Code zu programmieren.

Bei Zielsystemen, deren Rechnerarchitektur grundsätzlich für eine Vektorverarbeitung geeignet ist, ist es nicht ratsam Vektoren mit beliebiger Größe zu verwenden. Abhängig von der Art der Datenverarbeitung des Prozessors gibt es nämlich eine optimale Vektorgröße. Vektoren, die mit dieser Arraygröße deklariert werden, werden schnellstmöglich verarbeitet. Vektoren, die als größeres Array deklariert werden, haben keinen Geschwindigkeitsvorteil. Vektoren, die als kleineres Array deklariert werden, nutzen die Möglichkeiten des Prozessors nicht vollständig.

Sie können die optimale Größe zur Laufzeit abfragen. Die Informationen stehen Ihnen in den Konstanten `Constants.vcOptimalREAL` für Vektoren mit `REAL`-Elementen und `Constants.vcOptimalLREAL` für Vektoren mit `LREAL`-Elementen zur Verfügung. Die Konstanten haben den Datentyp `INT`. Wenn eine Konstante als optimale Größe den Wert 1 zurück liefert, bedeutet das, dass das Zielsystem keine beschleunigte Vektorverarbeitung unterstützt.

Operator '___VCADD'

Der Operator berechnet die Summe zweier Vektoren.

Syntax

```
<vector variable> := <1st vector operand> ___VCADD <2nd vector operand>;
```

Beispiel: Addieren

```
FUNCTION_BLOCK FB_ADD
VAR
  vcA      : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
  vcB      : __VECTOR[3] OF REAL := __VCSET_REAL(1, 2, 3);
  vcResult : __VECTOR[3] OF REAL;
END_VAR

vcResult := vcA ___VCADD vcB;
```

Operator '___VCSUB'

Der Operator berechnet die Differenz zweier Vektoren.

Syntax

```
<vector variable> := <vector minuend> ___VCADD <vector subtrahend>;
```

Beispiel: Subtrahieren

```
FUNCTION_BLOCK FB_SUB
VAR
  vcA      : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
  vcB      : __VECTOR[3] OF REAL := __VCSET_REAL(1, 2, 3);
  vcResult0 : __VECTOR[3] OF REAL;
  vcResult1 : __VECTOR[3] OF REAL;
END_VAR

vcResult0 := vcA ___VCSUB vcB;
vcResult1 := vcB ___VCSUB vcA;
```


Operator '___VCMUL' Der Operator berechnet das Produkt zweier Vektoren oder eines Skalars (Gleitpunktzahl) mit einem Vektor.

Syntax

```
<vector variable> := <1st vector operand> ___VCMUL <2nd vector operand> |
                    <scalar operand> ___VCMUL <vector operand> |
                    <vector operand> ___VCMUL <scalar operand> ;
```

Beispiel: Multiplizieren

```
FUNCTION_BLOCK FB_MUL
VAR
    rScalar    : REAL := 1.1;
    vcA        : __VECTOR[3] OF REAL;
    vcB        : __VECTOR[3] OF REAL;
    vcResult0  : __VECTOR[3] OF REAL;
    vcResult1  : __VECTOR[3] OF REAL;
    vcResult2  : __VECTOR[3] OF REAL;
END_VAR

vcResult0 := vcA ___VCMUL vcB;
vcResult1 := rScalar ___VCMUL vcB;
vcResult2 := vcA ___VCMUL 3.3;
```

Operator '___VCDIV' Der Operator berechnet den Quotienten zweier Vektoren oder eines Vektors und eines Skalars.

Syntax

```
<vector variable> := <vector dividend> ___VCDIV <vector divisor> |
                    <vector dividend> ___VCMUL <scalar divisor> ;
```

Beispiel: Dividieren

```
FUNCTION_BLOCK FB_DIV
VAR
    iScalar    : INT := 3;
    rScalar    : REAL := 1.5;
    vcA        : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
    vcB        : __VECTOR[3] OF REAL := __VCSET_REAL(1, 2, 3);
    vcResult0  : __VECTOR[3] OF REAL;
    vcResult1  : __VECTOR[3] OF REAL;
    vcResult2  : __VECTOR[3] OF REAL;
END_VAR

vcResult0 := vcA ___VCDIV vcB;
// ERROR CODE vcResult1 := rScalar ___VCDIV vcB;
// ERROR CODE vcResult1 := iScalar ___VCDIV vcB;
// ERROR CODE vcResult1 := 3.3 ___VCDIV vcB;
vcResult2 := vcA ___VCDIV 1.5;
vcResult2 := vcA ___VCDIV iScalar;
vcResult2 := vcA ___VCDIV rScalar;
```

Operator '___VCDOT' Der Operator berechnet das Punktprodukt (Skalarprodukt) zweier Vektoren.

Syntax

```
<skalar variable> := <1st vector operand> ___VCDOT <2nd vector operand> ;
```

Beispiel: Punktprodukt

```
FUNCTION_BLOCK FB_DOT
VAR
    rResult : REAL;
    vcA     : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
```

```

    vcB      : __VECTOR[3] OF REAL := __VCSET_REAL(1, 2, 3);
END_VAR

rResult := vcA __VCDOT vcB; // = 18

```

Operator '___VCSQRT' Der Operator berechnet die Quadratwurzel für jedes Element des Vektors.

Syntax

```
<vector variable> := ___VCSQRT( <vector operand> );
```

Beispiel: Quadratwurzel

```

FUNCTION_BLOCK FB_SQRT
VAR
    vcA      : __VECTOR[3] OF REAL := __VCSET_REAL(4, 9, 16);
    vcResult0 : __VECTOR[3] OF REAL;
END_VAR

vcResult0 := ___VCSQRT(vcA);

```

Operator '___VCMAX' Der Operator berechnet den maximalen Vektor zweier Vektoren. Dabei wird elementweise das Maximum bestimmt.

Syntax

```
<vector variable> := ___VCMAX(<1st vector operand>, <2nd vector operand>);
```

Beispiel: Maximumvektor

```

FUNCTION_BLOCK FB_MAX
VAR
    vcA      : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
    vcB      : __VECTOR[3] OF REAL := __VCSET_REAL(1, 2, 6);
    vcResult0 : __VECTOR[3] OF REAL;
END_VAR

vcResult0 := ___VCMAX(vcA, vcB);

```

Operator '___VCSET_REAL' Der Operator setzt alle Elemente eines Vektors in einer Anweisung. Die Elemente haben den Datentyp REAL.

Syntax

```
<vector variable> := ___VCSET_REAL( <first literal>, ( <next literal> )+ );
( ... )+ // number of elements have to match
```

Beispiel:

```

FUNCTION_BLOCK FB_SET
VAR
    vcA : __VECTOR[3] OF REAL := ___VCSET_REAL(3, 3, 3);
    vcB : __VECTOR[3] OF REAL := ___VCSET_REAL(1, 2, 3);
END_VAR

vcA := ___VCSET_REAL(4, 4, 4);
vcB := ___VCSET_REAL(1.1, 2.2, 3.3);

```

Operator '___VCSET_LREAL' Der Operator setzt alle Elemente eines Vektors auf einmal in einer Anweisung. Die Elemente haben den Datentyp LREAL.

Sie können den Operator überall dort verwenden, wo Variablen gültig sind, beispielsweise in Zuweisungen in Implementierungen oder als Parameter in Funktionsaufrufen.

Syntax

```
<vector variable> := __VCSET_LREAL( <first literal>, ( <next literal> )+ );
( ... )+ // number of elements have to match
```

Beispiel:

```
FUNCTION_BLOCK FB_SET
VAR
    vc1A : __VECTOR[3] OF LREAL := __VCSET_LREAL(3, 3, 3);
    vc1B : __VECTOR[3] OF LREAL := __VCSET_LREAL(1, 2, 3);
END_VAR

vc1A := __VCSET_LREAL(-1.7976931348623158E+308, 0.0, 1.7976931348623158E+308);
vc1B := __VCSET_LREAL(-1.7976931348623158E+308, 0.0, 1.7976931348623158E+308);
```

Operator '__VCLOAD_REAL'

Der Operator interpretiert einen beliebigen Speicherbereich als Vektor, beispielsweise um Vektorvariablen mit einem bestehenden Code zu verbinden.

Der Operator benötigt 2 Parameter. Der erste Parameter gibt die Anzahl der Vektorelemente an. Der zweite Parameter ist ein Pointer auf die REAL-Daten.

Syntax

```
<vector variable> := __VCLOAD_REAL( <vector size>, <pointer to data of type REAL> );
<vector size> : 2 | 3 | 4 | 5 | 6 | 7 | 8
```

Beispiel: Vektorisieren

```
FUNCTION_BLOCK FB_LOAD
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    rData0 : REAL := 1.234;
    rData1 : REAL := 5.678;
    rData2 : REAL := 9.123;
    pData : POINTER TO REAL := ADR(rData0);
    vcA : __VECTOR[3] OF REAL := __VCSET_REAL(3, 3, 3);
END_VAR

vcA := __VCLOAD_REAL(3, pData);
```

Operator '__VCLOAD_LREAL'

Der Operator interpretiert einen beliebigen Speicherbereich als Vektor, beispielsweise um Vektorvariablen mit einem bestehenden Code zu verbinden.

Der Operator benötigt 2 Parameter. Der erste Parameter gibt die Anzahl der Vektorelemente an. Der zweite Parameter ist ein Pointer auf die LREAL-Daten.

Syntax

```
<vector variable> := __VCLOAD_LREAL( <vector size>, <pointer to data of type LREAL> );
<number of vector elements> : 2 | 3 | 4 | 5 | 6 | 7 | 8
```

Beispiel: Vektorisieren

```
FUNCTION_BLOCK FB_LOAD
VAR_INPUT
END_VAR
```

```

VAR_OUTPUT
END_VAR
VAR
  lrData0 : LREAL           := -1.7976931348623158E+308;
  lrData1 : LREAL           := 1.6E+308;
  lrData2 : LREAL           := 1.7E+308;
  lrData3 : LREAL           := -1.6E+308;
  plData  : POINTER TO LREAL := ADR(lrData0);
  vc1A    : __VECTOR[4] OF LREAL := __VCSET_LREAL(4, 4, 4, 4);
END_VAR

vc1A := __VCLOAD_LREAL(4, plData);

```

Operator '__VCSTORE' Der Operator speichert bzw. kopiert den Inhalt des Vektors an die angegebene Speicheradresse.

Die Anzahl und der Typ der Elemente werden dabei automatisch von der Vektorvariablen übernommen.

Syntax

```
<vector variable> := __VCSTORE( <pointer to data>, <vector variable> );
```

Beispiel: Speichern

```

FUNCTION_BLOCK FB_STORE
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
  rData0 : REAL           := 3;
  rData1 : REAL           := 3;
  rData2 : REAL           := 3;
  pData  : POINTER TO REAL := ADR(rData0);
  lrData0 : LREAL         := 4;
  lrData1 : LREAL         := 4;
  lrData2 : LREAL         := 4;
  lrData3 : LREAL         := 4;
  plData : POINTER TO LREAL := ADR(lrData0);

  vcA : __VECTOR[3] OF REAL := __VCSET_REAL( 1.234, 5.678, 9.123);
  vc1A : __VECTOR[4] OF LREAL := __VCSET_LREAL(-1.7976931348623158E+308,
                                              1.6E+308, 1.7E+308, -1.6E+308);
END_VAR

__VCSTORE(pData, vcA);
__VCSTORE(plData, vc1A);

```

Siehe auch:

- [Kap. 9.2.3 "Array deklarieren" auf Seite 89](#)
- ["Checkbounds" auf Seite 578](#)

17.5.16 Datentyp 'STRUCT'

Sie legen eine Struktur durch ein DUT-Objekt über den Befehl **Objekt hinzufügen** im Projekt an.

Die Strukturdeklaration beginnt mit den Schlüsselwörtern TYPE und STRUCT und endet mit END_STRUCT und END_TYPE.

Syntax für Strukturdeklaration:

```

TYPE <Structure_name>:
STRUCT
  <Variable_declaration 1>
  ...

```

```

    <Variable_declaration n>
END_STRUCT
END_TYPE

```

<structure name> ist ein Typ, den IndraLogic im gesamten Projekt erkennt und den Sie wie einen Standard-Datentyp verwenden können.

Sie können auch verschachtelte Strukturen verwenden. Dabei ist die einzige Beschränkung, dass Sie Variablen nicht Adressen zuweisen dürfen (die AT-Deklaration ist nicht zulässig!).

Beispiel für eine Strukturdefinition namens Polygone

```

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

```

Siehe auch

- [Kap. 18.1.8 "Objekt 'DUT'" auf Seite 517](#)
- [Kap. 17.3.28 "Operator 'EQ'" auf Seite 332](#)

Initialisierung von Strukturen

Beispiel

```
pPoly_1 : polygonline := ( start:=[3,3], point1:=[5,2], point2 := [7,3], point3 := [9,4], point4 := [11,5], end:=[13,6] );
```

Sie dürfen keine Initialisierungen mit Variablen verwenden. Ein Beispiel für die Initialisierung eines Arrays einer Struktur finden Sie auf der Hilfeseite zu Datentyp ARRAY.

Siehe auch

- ["Beispiel: 3-dimensionales Array" auf Seite 387](#)

Zugriff auf Strukturkomponenten

Sie greifen auf eine Strukturkomponente gemäß folgender Syntax zu:

<structure name>.<component name>

Sie können also für das obige Beispiel der Struktur `polygonline` mit `pPoly_1.start` auf die Komponente `start` zugreifen.

Siehe auch

- ["Beispiel: 3-dimensionales Array" auf Seite 387](#)

Bitzugriff in Strukturen

Bit ist ein spezieller Datentyp, den Sie nur in Strukturen definieren können.

Er belegt 1 Bit Speicherplatz und ermöglicht, einzelne Bits einer Struktur über einen Namen zu adressieren.

```

TYPE <Structure_name>:
STRUCT
    <Bit_name bit1> : BIT;
    <Bit_name bit2> : BIT;
    <Bit_name bit3> : BIT;
    ...
    <Bit_name bitn> : BIT;
END_STRUCT
END_TYPE

```

Sie können mit folgender Syntax kann auf die Strukturkomponente BIT zugreifen:

<structure name>.<bit name>



Sie können keine Referenzen und Pointern auf BIT-Variablen verwenden.

Sie dürfen keine BIT-Variablen in Datenfeldern verwenden.

Siehe auch

- [Kap. 17.5.10 "Datentyp 'BIT'" auf Seite 382](#)

17.5.17 Enumerationen

Eine Enumeration oder Aufzählung ist ein benutzerdefinierter Datentyp, der sich aus einer kommaseparierten Reihe von Komponenten, auch Enumerationswerte genannt, zusammensetzt, um benutzerdefinierte Variablen zu deklarieren. Außerdem können Sie die Enumerationskomponenten wie konstante Variablen verwenden, deren Bezeichner `<enumeration name>.<component name>` global im Projekt bekannt sind.

Die Deklaration einer Enumeration nehmen Sie in einem DUT-Objekt vor, das Sie über den Befehl **Hinzufügen** im Projekt angelegt haben.

Deklaration *Syntax*

```
( {attribute 'strict'} )? // Pragma optional but recommended
TYPE <enumeration name> ;
(
  <first component deklaration>,
  (<component declaration> ,)+
  <last component deklaration>
)( <basic data type> )? ( := <default variable initialization> )?
;
END_TYPE

( ... )? : Optional
<component declaration> : <component name> ( := <component initialization> )?
<basic data type> : INT | UINT | SINT | USINT | DINT | UDINT | LINT | ULINT | BYTE | WORD | DWORD | LWORD
<variable initialization> ; <one of the component names>
```

In einer Enumerationsdeklaration sind üblicherweise mindestens 2 Komponenten deklariert, Sie können jedoch beliebig viele deklarieren. Jeder einzelnen Komponente kann eine eigene Initialisierung zugewiesen werden. Enumerationen haben automatisch den Basisdatentypen `INT`, Sie können aber einen anderen Basisdatentypen angeben. Außerdem können Sie in der Deklaration eine Komponente angeben, mit der eine Enumerationsvariable dann initialisiert wird.

Das Pragma `{attribute 'strict'}` bewirkt, dass eine strenge Typprüfung, so wie unten beschrieben, durchgeführt wird.

Beispiel

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE COLOR_BASIC :
(
  yellow,
  green,
  blue,
  black
) // Basic data type is INT, default initialization for all COLOR_BASIC variables is yellow
;
END_TYPE
```

Enumeration mit explizitem Basisdatentyp

Erweiterungen zur Norm IEC 61131-3.

Der Basisdatentyp bei einer Enumerationsdeklaration ist standardmäßig `INT`. Sie können aber auch Enumerationen deklarieren, die explizit auf einen anderen ganzzahligen Datentypen basieren.

Syntax

```
<basic data type> : INT | UINT | SINT | USINT | DINT | UDINT | LINT | ULINT | BYTE | WORD | DWORD | LWORD
```

Beispiel: Enumeration mit Basisdatentyp DWORD

```
TYPE COLOR :
(
  white := 16#FFFFFF00,
  yellow := 16#FFFF0000,
  green := 16#00FF0000,
  blue := 16#0000FFFF,
  black := 16#88000000
) DWORD := black
; // Basic data type is DWORD, default initialization for all COLOR variables is black
END_TYPE
```

Strikte Programmierregeln



Beim Deklarieren einer Enumeration wird automatisch in der ersten Zeile das Pragma `{attribute 'strict'}` hinzugefügt.

Mit dem automatischen Hinzufügen des Pragmas `{attribute 'strict'}` werden strikte Programmierregeln aktiviert.

Folgender Code wird dann als Compilerfehler eingestuft:

- Arithmetische Operationen mit Enumerationskomponenten.
Beispielsweise kann eine Enumerationsvariable nicht als Zählvariable in einer `FOR`-Schleife verwendet werden.
- Zuweisung eines konstanten Werts, der nicht einem Enumerationswert entspricht, an eine Enumerationskomponente.
- Zuweisung einer nicht-konstanten Variablen, die einen anderen Datentyp als die Enumeration hat, an eine Enumerationskomponente,

Arithmetischen Operationen können dazu führen, dass Enumerationskomponenten nicht deklarierte Werte zugewiesen werden.

Ein besserer Programmierstil nutzt `SWITCH- /CASE`-Anweisungen, um Komponentenwerte zu bearbeiten.

Deklaration und Initialisierung von Enumerationsvariablen

Syntax

Programm:

```
<variable name> : <enumeration name> ( :=<initialization> )? ;
```

Bei der Deklaration einer Enumerationsvariable mit benutzerdefiniertem Datentyp `<enumeration name>` kann diese mit einer Enumerationskomponenten initialisiert werden.

Beispiel

```
PROGRAM PLC_PRG
VAR
  colorCar: COLOR;
  colorTaxi : COLOR := COLOR.yellow;
END_VAR
```

Die Variable `colorCar` wird mit `COLOR.black` initialisiert. Das ist die Standardinitialisierung für alle Enumerationsvariablen des Typs `COLOR` und so in der Typdeklaration festgelegt. Die Variable `colorTaxi` hat eine eigene Initialisierung.

Wenn keine Initialisierungen angegeben sind, wird mit 0 initialisiert.

Beispiel

```
PROGRAM PLC_PRG
VAR
  cbFlower : COLOR_BASIC;
  cbTree: COLOR_BASIC := COLOR_BASIC.green;
END_VAR
```

Die Variable `cbFlower` wird mit `COLOR_BASIC.yellow` initialisiert. Das ist die Standardinitialisierung für alle Enumerationsvariablen des Typs `COLOR_BASIC`. Da in der Enumerationsdeklaration keine Komponente für die Initialisierung angegeben ist, wird automatisch mit der Komponente, die den Wert 0 hat, initialisiert. Das ist üblicherweise die erste der Enumerationskomponenten. Das kann aber auch eine andere Komponente sein, die nicht an erster Stelle steht, aber explizit mit 0 initialisiert wird.

Die Variable `cbTree` hat eine explizite Initialisierung.

Wenn sowohl beim Typ als auch bei der Variable kein Wert angegeben ist, dann gilt folgende Regel: Wenn eine Enumeration einen Wert für 0 enthält, dann ist dieser Wert die Standardinitialisierung, wenn nicht, dann die erste Komponente in der Liste.

Beispiel: Initialisierung mit der 0-Komponente

```
TYPE ENUM :
(
  e1 := 2,
  e2 := 0,
  e3
)
;
END_TYPE

PROGRAM PLC_PRG
VAR
  e : ENUM;
END_VAR
```

Die Variable `e` wird mit `ENUM.e2` initialisiert.

Beispiel: Initialisierung mit der ersten Komponente

```
TYPE ENUM2 :
(
  e1 := 3,
  e2 := 1,
  e3
)
;
END_TYPE

PROGRAM PLC_PRG
VAR
  e2 : ENUM2;
END_VAR
```

Die Variable `e2` wird mit `ENUM.e1` initialisiert.

Die Enumerationskomponenten können auch als konstante Variablen mit dem Bezeichner `<enumeration name>.<component name>` verwendet werden. Enumerationskomponenten sind global im Projekt bekannt und der Zugriff auf sie ist eindeutig. Deshalb kann ein Komponentennamen in unterschiedlichen Enumerationen verwendet werden.

Beispiel: Komponente blue

```
PROGRAM PLC_PRG
VAR
    cbFlower : COLOR_BASIC;
    colorCar : COLOR;
END_VAR

(* unambiguous identifiers although the component names are identical *)
cbFlower := COLOR_BASIC.blue;
colorCar := COLOR.blue;

(* invalid code *)
cbFlower := blue;
colorCar := blue;
```

Siehe auch [Kap. 17.3.70 "Operator - Enumerations-Namensraum"](#) auf Seite 361

17.5.18 Unterbereichstypen

Ein Unterbereichstyp ist ein Datentyp dessen Wertebereich eine Untermenge eines Basistypen umfasst.

Syntax der Deklaration:

Programm:

`<Name> : <Inttype> (<ug>..<og>);`

<code><Name></code>	Gültiger IEC-Bezeichner
<code><Inttype></code>	Datentyp des Unterbereichs (SINT, USINT, INT, UINT, DINT, UDINT, BYTE, WORD, DWORD, LINT, ULINT, LWORD).
<code><ug></code>	Untergrenze des Bereichs: Konstante, die mit dem Basisdatentyp kompatibel sein muss. Die Untergrenze selbst gehört zu diesem Bereich.
<code><og></code>	Obergrenze des Bereichs: Konstante, die mit dem Basisdatentyp kompatibel sein muss. Die Obergrenze selbst gehört zu diesem Bereich.

Beispiele:

```
VAR
    i : INT (-4095..4095);
    ui : UINT (0..10000);
END_VAR
```

Wenn Sie einem Unterbereichstypen in der Deklaration oder in der Implementierung ein Wert zuweisen, der nicht in diesen Bereich fällt (zum Beispiel `i:=5000`), gibt IndraLogic eine Fehlermeldung aus.



Beachten Sie die Möglichkeit, die Bereichsgrenzen eines Unterbereichstypen während der Laufzeit mit den impliziten Überwachungsfunktionen `CheckRangeSigned` und `CheckRangeUnsigned` zu überwachen.

Siehe auch

- ["CheckRangeUnsigned" auf Seite 579](#)
- ["CheckRangeSigned" auf Seite 579](#)

17.6 Pragmas

Pragma-Anweisungen beeinflussen die Eigenschaften einer oder mehrerer Variablen bezüglich der Kompilierung oder des Prä-Kompilierungsprozesses.

Dafür stehen Ihnen verschiedene Kategorien von Pragmas zur Verfügung.

17.6.1 Meldungspragmas

Meldungspragmas dienen dazu, die Ausgabe von Meldungen im Meldungsfenster während des Übersetzungsvorgangs zu erzwingen.

Einfügeort: Separate oder bereits bestehende Zeile im Texteditor einer POU.

Pragma	Meldungstyp
{text <'textstring'>}	Text: Ausgabe von <textstring>.
{info <'textstring'>}	 Information : Ausgabe von <textstring>.
{warning <'textstring'>}	 Warnung: Ausgabe von <textstring>. Im Unterschied zum Attributpragma 'obsolete' definieren Sie die Warnung lokal für die aktuelle Position.
{error <'textstring'>}	 Fehler : Ausgabe von <textstring>.

Tab. 17-14: 4 Typen von Meldungspragmas



Im Meldungsfenster von IndraLogic gelangen Sie von einer Meldung der Kategorie **Information**, **Warnung** und **Fehler** mit Hilfe der Befehle **Nächste Meldung** und **Vorherige Meldung** zur Quellposition der Meldung.

Dies bedeutet, Sie gelangen an die Position, wo das Pragma im Quellcode hinzugefügt ist.

Beispiel

```
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:

Messages			
Build			
Description	Project	Object	Position
----- Build started: Application: Res.App2 -----			
typify code ...			
❗ Compile time before typification: 0 ms			
❗ Compile time after typification: 15 ms			
❗ TODO: should get another name!	TS pragma	NewPOU	Line 3 (Decl)
⚠ This is a warning	TS pragma	NewPOU	Line 7 (Impl)
Part xy has been compiled completely	TS pragma	NewPOU	Line 10 (Impl)
Compile complete -- 1 errors, 2 warnings			

Abb. 17-75:

Siehe auch

- [Kap. 17.6.3 "Bedingte Pragmas" auf Seite 439](#)

17.6.2 Attribut-Pragmas

Attributpragmas dienen dazu die Kompilierung und die Vorkompilierung zu beeinflussen.

IndraLogic unterstützt eine Reihe von vordefinierten Attributpragmas. Zusätzlich können Sie benutzerdefinierte Pragmas verwenden, die Sie vor der Übersetzung des Projekts mit Hilfe von bedingten Pragmas abfragen können.

Attribute werden im Deklarationsteil definiert.

Ausnahme: Für die Objekte Aktion und Transition, die keinen eigenen Deklarationsteil haben, können Sie Attribute zu Beginn des Implementierungsteils definieren.



Wenn Sie eigene Attribute definieren, achten Sie bitte auf Eindeutigkeit.

Sie erreichen dies beispielsweise indem Sie den Attributnamen mit einem Präfix versehen.

OEMs können hierfür das Vendor-Präfix verwenden.

Benutzerdefinierte Attribute

Benutzerdefinierte Attribute sind beliebige applikations- oder benutzerdefinierte Attribute, die Sie auf POU's, Aktionen, Datentypdefinitionen und Variablen anwenden können. Ein benutzerdefiniertes Attribut können Sie vor der Übersetzung der Applikation mit Hilfe von bedingten Pragmas abfragen.



Sie können benutzerdefinierte Attribute mit bedingten Pragmas mit dem Operator `hasattribute` abfragen.

Genaue Informationen und Beispiele finden Sie unter 'Bedingte Pragmas'.

Syntax:

```
{attribute 'attribute'}
```

Beispiel für POU's und Aktionen

Attribut 'vision' für Funktion *fun1*

```
{attribute 'vision'}
FUNCTION fun1 : INT
VAR_INPUT
  i : INT;
END_VAR
```

Beispiel zu Variablen *Attribut 'DoCount' für Variable ivar:*

```
PROGRAM PlcProg
VAR
  {attribute 'DoCount'};
  ivar:INT;
  bvar:BOOL;
END_VAR
```

Beispiel zu Datentypen *Attribut 'aType' für Datentyp DUT_1:*

```
{attribute 'aType'}
TYPE DUT_1 :
STRUCT
  a:INT;
  b:BOOL;
END_STRUCT
END_TYPE
```

Siehe auch

- [Kap. 17.6.3 "Bedingte Pragmas" auf Seite 439](#)

Attribut 'call_after_global_init_slot'

VAR_INPUT-Deklarationen in Funktionen oder Methoden, die das Attribut verwenden, führen zu Übersetzungsfehlern!

Grund: Eingabevariablen sind in diesem Fall zum Zeitpunkt des Aufrufs, der implizit während des Online-Change erfolgt, nicht bekannt.

Das Pragma bewirkt, dass alle Funktionen und Programme, die dieses Attribut enthalten, nach der globalen Initialisierung aufgerufen werden. Die Aufrufreihenfolge legen Sie durch den Attributwert fest.

Syntax:

```
{attribute 'call_after_global_init_slot' := '<slot>'}
```

<slot>: Ganzzahliger Wert, der den Stellenwert in der Reihenfolge der Aufrufe definiert: Je niedriger der Wert ist, desto früher erfolgt der Aufruf. Wenn mehrere Bausteine denselben Stellenwert für das Attribut haben, so bleibt die Reihenfolge ihrer Aufrufe unbestimmt.

Einfügeort: Erste Zeile über dem Deklarationsteil von Funktionen und Programmen

Wenn eine Methode das Attribut besitzt, ermittelt IndraLogic alle Instanzen des entsprechenden Funktionsbausteins und ruft alle Instanzen im angegebenen Slot auf. Auf die Reihenfolge der Instanzen untereinander haben Sie in diesem Fall keinen Einfluss.

Siehe auch

- [Kap. 17.10 "Methode 'FB_Init', 'FB_Reinit' und 'FB_Exit'" auf Seite 455](#)
- [Kap. "Attribut 'global_init_slot'" auf Seite 413](#)

Attribut 'call_after_init'

VAR_INPUT-Deklarationen in Funktionen oder Methoden, die das Attribut verwenden, führen zu Übersetzungsfehlern!

Grund: Eingabevariablen sind in diesem Fall zum Zeitpunkt des Aufrufs, der implizit während des Online-Change erfolgt, nicht bekannt.

Das Pragma bewirkt, dass eine Methode implizit nach der Initialisierung einer Funktionsbaustein-Instanz aufgerufen wird. Aus Performanzgründen müssen

Sie das Attribut sowohl dem Funktionsbaustein als auch der Methode in einer eigenen ersten Zeile über dem Deklarationsteil hinzufügen.

Syntax:

```
{attribute 'call_after_init'}
```

Aufruf: Erste Zeile über dem Deklarationsteil der Methode und des Funktionsbausteins.

IndraLogic ruft die Methode nach der Methode FB_init und nachdem die Variablenwerte eines Initialisierungsausdrucks in der Instanzdeklaration gültig wurden, auf.

Diese Funktionalität wird ab Compilerversion 3.4.1.0 unterstützt.

Beispiel

Definition

```
{attribute 'call_after_init'}
FUNCTION_BLOCK FB
... <functionblock definition>

{attribute 'call_after_init'}
METHOD FB_AfterInit
... <method definition>
```

Die Definition setzt zum Beispiel folgende Deklaration in die nachfolgende Codeabarbeitung um:

```
inst : FB := (in1 := 99);
```

Codeabarbeitung

```
inst.FB_Init();
inst.in1 := 99;
inst.FB_AfterInit();
```

Somit kann in der Methode FB_AfterInit auf die benutzerdefinierte Initialisierung reagiert werden.

Siehe auch

- [Kap. 17.10 "Methode 'FB_Init', 'FB_Reinitund 'FB_Exit'" auf Seite 455](#)

Attribut 'call_after_online_change_slot'



VAR_INPUT-Deklarationen in Funktionen oder Methoden, die das Attribut verwenden, führen zu Übersetzungsfehlern!

Grund: Eingabevariablen sind in diesem Fall zum Zeitpunkt des Aufrufs, der implizit während des Online-Change erfolgt, nicht bekannt.

Das Pragma bewirkt, dass alle Funktionen und Programme, die dieses Attribut enthalten, nach einem Online-Change aufgerufen werden. Die Aufrufreihenfolge legen Sie durch den Attributwert <slot> fest.

Syntax:

```
{attribute 'call_after_online_change_slot' := '<slot>'}
```

<slot>: Ganzzahliger Wert, der den Stellenwert in der Reihenfolge der Aufrufe definiert: Je niedriger der Wert ist, desto früher erfolgt der Aufruf. Haben mehrere Bausteine denselben Stellenwert für das Attribut, so bleibt die Reihenfolge ihrer Aufrufe unbestimmt.

Aufruf: Erste Zeile über dem Deklarationsteil von Funktionen, Methoden und Programmen.

Wenn eine Methode das Attribut besitzt, dann ermittelt IndraLogic alle Instanzen des betreffenden Funktionsbausteins. IndraLogic ruft alle Instanzen in dem angegebenen Slot auf. Auf die Reihenfolge der Instanzen untereinander haben Sie in diesem Fall keinen Einfluss.



Da die Applikation während des Online-Change nicht laufen kann, kann jeder Code der in dieser Situation ausgeführt wird, zu einem Jitter führen. Halten Sie deshalb den auszuführenden Code-Umfang möglichst gering.

Siehe auch

- [Kap. "Befehl 'Online-Change'" auf Seite 657](#)

Attribut 'call_before_global_exit_slot'



VAR_INPUT-Deklarationen in Funktionen oder Methoden, die das Attribut verwenden, führen zu Übersetzungsfehlern!

Grund: Eingabevariablen sind in diesem Fall zum Zeitpunkt des Aufrufs, der implizit während des Online-Change erfolgt, nicht bekannt.

Das Pragma bewirkt, dass alle Funktionen und Programme, die dieses Attribut in einer eigenen ersten Zeile ihrer Deklaration enthalten, vor den GlobalExit aufgerufen werden.

Der GlobalExit findet vor einem erneuten Download, oder bei einem Reset statt. Betroffen sind Bausteine, die mit der Methode `FB_Exit` versehen sind. Die Aufrufreihenfolge wird durch den Attributwert festgelegt.

Syntax:

```
{attribute 'call_before_global_exit_slot' := '<slot>'}
```

Einfügeort: Erste Zeile über dem Deklarationsteil von Funktionen und Programmen.

<slot>: Ganzzahliger Wert, der den Stellenwert in der Reihenfolge der Aufrufe definiert: Je niedriger der Wert ist, desto früher erfolgt der Aufruf. Wenn mehrere Bausteine denselben Stellenwert für das Attribut haben, so bleibt die Reihenfolge ihrer Aufrufe unbestimmt.

Wenn eine Methode das Attribut besitzt, wird die Methode für alle Instanzen des betreffenden Funktionsbausteins aufgerufen. IndraLogic ruft alle Instanzen im angegebenen Slot auf. Auf die Reihenfolge der Instanzen untereinander haben Sie in diesem Fall keinen Einfluss.

Siehe auch:

- [Kap. 17.10 "Methode 'FB_Init', 'FB_Reinitund 'FB_Exit'" auf Seite 455](#)
- [Kap. 17.10 "Methode 'FB_Init', 'FB_Reinitund 'FB_Exit'" auf Seite 455](#)

Attribut 'call_on_type_change'

Mit dem Pragma kennzeichnen Sie eine Methode eines Funktionsbausteins A, die aufgerufen werden soll, sobald sich der Datentyp eines oder mehreren von A referenzierten Funktionsbausteinen B, C, ... ändert. Die Referenzierung kann über eine Zeigervariable oder über eine Variable vom Typ `REFERENCE` definiert sein.

Syntax

```
{attribute 'call_on_type_change':= '<name of the first referenced function block>'  
                                '<name of the second referenced function block>'  
                                '<name of the nth referenced function block>'
```

Einfügeort: Zeile oberhalb der ersten Zeile in der Deklaration der Methode.

Beispiel *Funktionsbaustein mit Referenzen*

```
FUNCTION_BLOCK FB_A  
...  
VAR  
  var_pt: POINTER TO FB_B;  
  var_ref: REFERENCE TO FB_C;  
END
```

Methode zur Reaktion auf eine Typänderung in den Referenzen FB_B und FB_C

```
{attribute 'call_on_type_change' := 'FB_B, FB_C'}  
METHOD METH_react_on_type_change : INT  
VAR
```

Attribut 'const_replaced', Attribut 'const_non_replaced'

Das Pragma {attribute 'const_replaced'} bewirkt, dass für alle globalen Konstanten, die dieses Attribut in der Konstantendeklaration enthalten, die Compiler-Option **Konstanten ersetzen** explizit aktiviert wird.

Das Pragma {attribute 'const_non_replaced'} fügen Sie entsprechend ein, um die Compiler-Option **Konstanten ersetzen** explizit zu deaktivieren. Dies kann beispielsweise erwünscht sein, damit eine Konstante in der Symbolkonfiguration verfügbar ist.

Die Option **Konstanten ersetzen** im Dialog **Projekteinstellungen**, Kategorie **Compiler-Optionen** ist für das ganze Projekt voreingestellt.

Syntax:

```
{attribute 'const_replaced'}  
{attribute 'const_non_replaced'}
```

Einfügeort: Zeile oberhalb der Deklarationszeile der globalen Variablen.

Beispiel

Die Konstanten iTestCon und bTestCon sind in der Symbolkonfiguration verfügbar, weil die Option **Konstanten ersetzen** deaktiviert ist.

```
VAR_GLOBAL CONSTANT  
  {attribute 'const_non_replaced'}  
  iTestCon : INT := 12;  
  {attribute 'const_non_replaced'}  
  bTestCon : BOOL := TRUE;  
  rTestCon : REAL := 1.5;  
END_VAR  
  
VAR_GLOBAL  
  iTestVar : INT := 12;  
  bTestVar : BOOL := TRUE;  
END_VAR
```

Siehe auch

- [Kap. "Dialog 'Projekteinstellungen- 'Compileroptionen'" auf Seite 759](#)
- [Kap. 10.2 "Symbolkonfiguration" auf Seite 167](#)

Attribut 'dataflow'

Mit dem Pragma steuern Sie den Datenfluss in der Abarbeitung von Funktionsbausteinen im FUP/KOP/AWL-Editor. Das Attribut legt fest, an welchem

Eingang oder Ausgang eines Funktionsbausteins die Weiterverbindung zum nächsten oder vorherigen Baustein anliegt.

Sie dürfen nur 1 Eingang und 1 Ausgang in der Deklaration eines Funktionsbausteins mit dem Attribut versehen.

Syntax:

```
{attribute 'dataflow'}
```

Einfügeort: Zeile oberhalb der Zeile mit der Deklaration der entsprechenden Variablen.

Bei Funktionsbausteinen ohne Attribut 'dataflow' bestimmt IndraLogic den Datenfluss folgendermaßen: Zunächst wird die Verbindung zwischen einem Ausgang und einem Eingang gleichen Datentyps gelegt. Dabei wird zuerst immer die oberste Eingangs- oder Ausgangsvariable der Funktionsbausteine genommen. Wenn es keine Variablen mit übereinstimmendem Datentyp gibt, verbindet IndraLogic den obersten Ausgang mit dem obersten Eingang der benachbarten Bausteine.

Beispiel

Die Verbindung zwischen FB und dem vorhergehenden Funktionsbaustein erfolgt über die Eingangsvariable `i1`. Die Verbindung zwischen FB und dem nachfolgenden Funktionsbaustein erfolgt über die Ausgangsvariable `outRes1`.

```
FUNCTION_BLOCK FB
VAR_INPUT
    r1 : REAL;
    {attribute 'dataflow'}
    i1 : INT;
    i2 : INT;
    r2 : REAL;
END_VAR

VAR_OUTPUT
    {attribute 'dataflow'}
    outRes1 : REAL;
    out1 : INT;
    g1 : INT;
    g2 : REAL;
END_VAR
```

Siehe auch

- [Kap. "Programmieren in Funktionsplan \(FUP\)" auf Seite 95](#)

Attribut 'displaymode'

Mit dem Pragma definieren Sie den Darstellungsmodus einer einzelnen Variablen.

Diese Festlegung überschreibt die globale Einstellung für die Darstellung der Monitoring-Variablen, die über die Befehle im Menü **Debuggen ► Anzeigemodus** erfolgt.

Syntax:

```
{attribute 'displaymode':=<displaymode>}
```

Folgende Definitionen sind möglich

- Binärformat
 - {attribute 'displaymode':='bin'}
 - {attribute 'displaymode':='binary'}
- Dezimalformat

- {attribute 'displaymode':='dec'}
- {attribute 'displaymode':='decimal'}
- Hexadezimalformat
 - {attribute 'displaymode':='hex'}
 - {attribute 'displaymode':='hexadecimal'}

Einfügeort: Zeile oberhalb der Zeile mit der Deklaration der entsprechenden Variablen.

Beispiel

```
VAR
{attribute 'displaymode':='hex'}
  dwVar1: DWORD;
END_VAR
```

Siehe auch

- [Kap. "Befehl 'Anzeigemodus'" auf Seite 676](#)

Attribut 'estimated-stack-usage'

Das Pragma übergibt einen Schätzwert für den Stackgrößenbedarf.

Methoden mit rekursivem Aufruf halten einer Stackprüfung nicht stand, weil der Stackverbrauch nicht ermittelt werden kann. Folglich wird eine Warnung ausgegeben. Um diese Warnung zu unterdrücken, können Sie der Methode einen Schätzwert in Bytes für den Stackgrößenbedarf mitgeben. Dann durchläuft die Methode die Stackprüfung erfolgreich.

Syntax

```
{attribute 'estimated-stack-usage' := '<estimated stack size in bytes>'}
```

Beispiel

```
{attribute 'estimated-stack-usage' := '127'} // 127 bytes
METHOD PUBLIC DoIt : BOOL
VAR_INPUT
END_VAR
```

Einfügeort: Erste Zeile über dem Deklarationsteil der Methode.

Im Kapitel "Methodenaufruf" finden Sie ein Beispiel, das dieses Pragma verwendet, siehe [Kap. 9.18.4 "Methodenaufruf" auf Seite 158](#).

Rekursiver Methodenaufruf

Innerhalb einer Implementierung kann eine Methode sich selbst aufrufen: Entweder direkt mit Hilfe des `THIS`-Pointers, oder mit Hilfe einer lokalen Variablen für den zugeordneten Funktionsbaustein.



Verwenden Sie Rekursionen vorwiegend zur Bearbeitung von rekursiven Datentypen wie beispielsweise verketteten Listen.

Allgemein ist es ratsam bei der Verwendung von Rekursion vorsichtig zu sein, da es bei unerwartet tiefen Rekursionen zu einem Stacküberlauf und damit zu einem Maschinenstillstand kommen kann.

Beispiel: Berechnung der Fakultät

Das folgende Programm `PLC_PRG` berechnet die Fakultät einer Zahl im Funktionsbaustein `FB_Factorial` auf unterschiedliche Weise in jeweils einer eigenen Methode.

Referenz Programmierung

- Methode m_Iterative: Iterativ
- Methode m_Pragmaed: Rekursiv mit Warnungsunterdrückung
- Methode m_Recursive: Rekursiv
- Methode m_Temp: Temporär mit Warnungsunterdrückung

Nur die Methode m_Recursive erzeugt eine Warnung

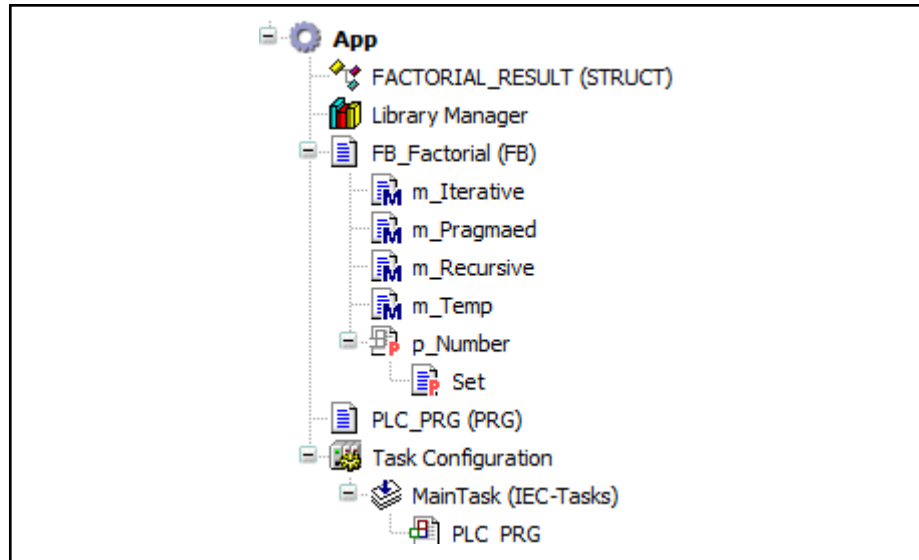


Abb. 17-76:

Programm:

```
// Contains the data of the factorial calculation of uiNumber
TYPE FACTORIAL_RESULT :
STRUCT
    uiNumber : UINT;
    udiIterative : UDINT;
    udiRecursive : UDINT;
    udiPragmaed : UDINT;
    udiTemp : UDINT;
END_STRUCT
END_TYPE

PROGRAM PLC_PRG
VAR
    fb_Factorial_A : FB_Factorial;
    factorial_A : FACTORIAL_RESULT := (uiNumber := 9, udiIterative := 0, udiRecursive := 0, udiPragmaed := 0 );
END_VAR
fb_Factorial_A.p_Number := factorial_A.uiNumber;
factorial_A.udiIterative := fb_Factorial_A.m_Iterative();
factorial_A.udiRecursive := fb_Factorial_A.m_Recursive(uiN := factorial_A.uiNumber);
factorial_A.udiPragmaed := fb_Factorial_A.m_Pragmaed(uiN := factorial_A.uiNumber);
factorial_A.udiTemp := fb_Factorial_A.m_Temp(uiN := factorial_A.uiNumber);

//Factorial calculation in different ways
FUNCTION_BLOCK FB_Factorial
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    uiN : UINT;
    udiIterative : UDINT;
    udiPragmaed : UDINT;
    udiRecursive : UDINT;
END_VAR

// Iterative calculation
METHOD PUBLIC m_Iterative : UDINT
VAR
    uiCnt : UINT;
```

```

END VAR
m_Iterative := 1;
IF uiN > 1 THEN
    FOR uiCnt := 1 TO uiN DO
        m_Iterative := m_Iterative * uiCnt;
    END FOR;
    RETURN;
ELSE
    RETURN;
END_IF;

//Recursive calculation with suppressed warning
{attribute 'estimated-stack-usage' := '99'}
METHOD PUBLIC m_Pragmaed : UDINT
VAR INPUT
    uiN : UINT;
END_VAR
VAR
END_VAR
m_Pragmaed := 1;
IF uiN > 1 THEN
    m_Pragmaed := uiN * THIS^.m_Pragmaed(uiN := (uiN - 1));
    RETURN;
ELSE
    RETURN;
END_IF;

//Recursive calculation
METHOD PUBLIC m_Recursive : UDINT
VAR INPUT
    uiN : UINT;
END_VAR
VAR
END_VAR
m_Recursive := 1;
IF uiN > 1 THEN
    m_Recursive := uiN * THIS^.m_Recursive(uiN := (uiN - 1) );
    RETURN;
ELSE
    RETURN;
END_IF;

// Called by temporary FB instance
{attribute 'estimated-stack-usage' := '99'}
METHOD PUBLIC m_Temp : UDINT
VAR INPUT
    uiN : UINT;
END_VAR
VAR
    fb_Temp : FB_Factorial;
END_VAR
m_Temp := 1;
IF uiN > 1 THEN
    m_Temp := uiN * fb_Temp.m_Temp(uiN := (uiN - 1));
    RETURN;
ELSE
    RETURN;
END_IF;

PROPERTY p_Number : UINT
uiN := p_Number; //Setter method

```

Bei der Ausführung des Programms generiert nur die Methode `m_Recursive` eine Warnung.

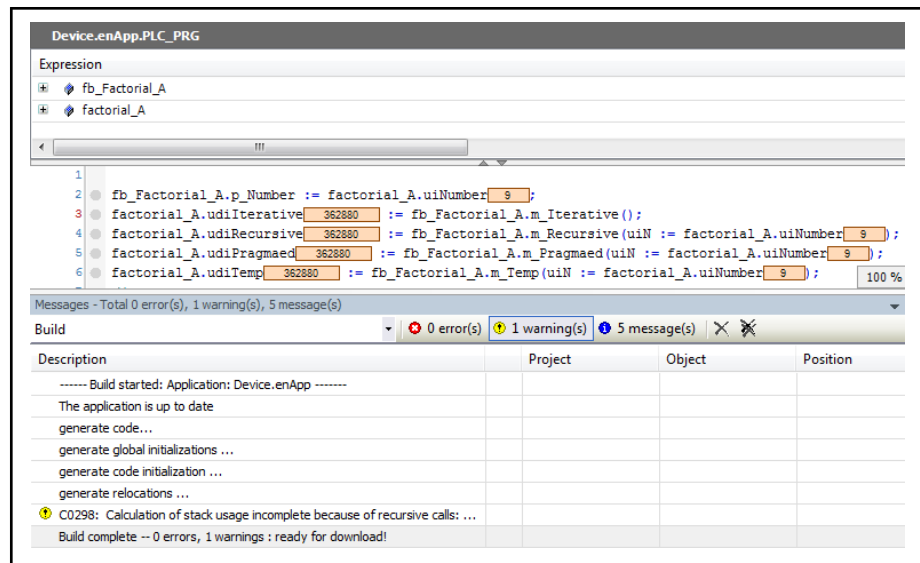


Abb. 17-77:

Siehe auch:

- [Kap. 9.18.4 "Methodenaufruf" auf Seite 158](#)
- [Kap. 9.18 "Objektorientiert programmieren" auf Seite 154](#)
- [Kap. "Objekt 'Methode' " auf Seite 564](#)
- [Kap. "Objekt 'Eigenschaft'" auf Seite 571](#)

Attribut 'ExpandFully'

Das Pragma bewirkt, dass die Komponenten eines Arrays, das als Eingabevariable für referenzierte Visualisierungen verwendet wird, im Eigenschaftendialog der Visualisierung sichtbar gemacht werden.

Syntax:

```
{attribute 'ExpandFully'}
```

Einfügeort: Die Zeile oberhalb der Zeile mit der Deklaration des Arrays.

Beispiel

Visualisierung **visu** soll in einen Frame innerhalb von Visualisierung **visu_main** eingefügt werden.

arr ist als Eingabevariable im Schnittstelleneditor von **visu** definiert und wird somit später für Zuweisungen im Eigenschaftendialog des Frames in **visu_main** bereitstehen. Um auch die einzelnen Komponenten von **arr** in diesem Eigenschaftendialog verfügbar zu bekommen, müssen Sie das Attribut 'ExpandFully' im Schnittstelleneditor von visu direkt vor **arr** einfügen.

Deklaration im Schnittstelleneditor von visu:

```
VAR_INPUT
{attribute 'ExpandFully'}
arr : ARRAY[0..5] OF INT;
END_VAR
```

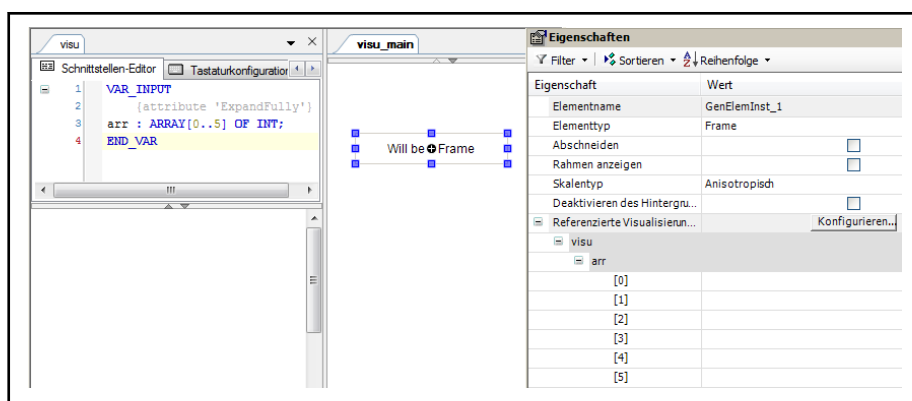


Abb. 17-78:

Attribut 'global_init_slot'

Das Pragma definiert die Reihenfolge der Initialisierung von Bausteinen und globalen Variablenlisten.

Die Reihenfolge bei der Initialisierung von Variablen aus globalen Variablenlisten ist nicht von vorn herein 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 für eine GVL ist 49990.

Der Standardwert für eine POU (Programm, Funktionsbaustein) ist 50000.

Ein niedrigerer Wert bewirkt eine frühere Initialisierung. Tragen mehrere Bausteine oder globale Variablenlisten denselben Wert für das Attribut 'global_init_slot', so bleibt die Reihenfolge ihrer Initialisierung unbestimmt.



Dieses Attribut ist nicht auf Einzelvariablen anwendbar.

Beispiel: Das Beispielprojekt enthalte zwei globale Variablen-Listen GVL_1 und GVL_2:



Abb. 17-79: Zwei globale Variablen Listen

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" von GVL_2 ist abhängig von der Variablen "A" aus GVL_1.

Initialisierungswert der Variablen "B" von GVL_2:

```
{attribute 'global_init_slot' := '350'}
VAR_GLOBAL
```

```

        B : INT:=GVL_1.A + 10;
    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 GVL_1.A+10 wohldefiniert ist zum Zeitpunkt der Initialisierung von "B".

Das Programm PlcProg verwendet beide Variablen "GVL_1.A" und "GVL_2.B".

Sein Initialisierungszeitpunkt muss also nach der Initialisierung der beiden globalen Variablenlisten liegen (also muss 'global_init_slot' größer 350 sein, im Beispiel 10000; funktioniert aber auch mit dem Standardwert 50000, also ohne 'global_init_slot').

Verwendung der Variablen

```

{attribute 'global_init_slot' := '10000'}
PROGRAM PlcProg
VAR
    ivar_1:INT:= GVL_1.A;
    ivar_2:INT:= GVL_2.B;
END_VAR

```

Attribut 'hide'

Das Pragma 'hide' verhindert, dass Variablen oder Programmierbausteine in der IndraLogic-Oberfläche angezeigt werden.

Anwendungszweck: Mit dem Pragma können Variablen oder Bausteine bewusst vor dem Anwender versteckt werden. Variablen oder Bausteine sind dann nicht im Bibliotheksverwalter und in der Eingabehilfe sichtbar und werden auch nicht bei der Funktion "Komponenten auflisten" angeboten.

Im Onlinebetrieb sind Variablen oder Bausteine im Deklarationsteil nicht sichtbar und es können auch keine Debug-Funktionalitäten darauf angewendet werden.

Syntax:

```
{attribute 'hide'}
```

Einfügeort: Bei Variablen oberhalb der Zeile mit der Deklaration der Variablen. Bei Programmierbausteinen in der ersten Zeile.

Beispiel

Der Funktionsbaustein **myPOU** verwendet das Attribut {attribute 'hide'}:

```

FUNCTION_BLOCK myPOU
VAR_INPUT
    a:INT;
{attribute 'hide'}
    a_invisible: BOOL;
    a_visible: BOOL;
END_VAR
VAR_OUTPUT
    b:INT;
END_VAR

```

Im Hauptprogramm werden 2 Instanzen des Funktionsbausteins **myPOU** definiert.

Programm:

```

PROGRAM PlcProg
VAR
    POU1, POU2: myPOU;
END_VAR

```

Während nun ein Eingabewert für `POU1` implementiert wird, zeigt die Funktion "Komponenten auflisten", die sich beim Tippen von `POU1`. (im Implementierungsteil von **PlcProg**) öffnet, die Variablen `a`, `a_visible` und `b` an, nicht aber die versteckte Variable `a_invisible`.



Mit dem Pragma `hide_all_locals` können Sie alle lokalen Variablen einer Deklaration verstecken.

Siehe auch

- [Kap. "Attribut 'hide_all_locals'" auf Seite 415](#)

Attribut 'hide_all_locals'

Das Pragma verhindert, dass alle lokalen Variablen einer Signatur in der Anzeige der Funktion "Komponenten auflisten", in der Eingabehilfe oder im Deklarationsteil im Online-Betrieb sichtbar sind.

Syntax:

```
{attribute 'hide_all_locals'}
```

Einfügeort: Erste Zeile über dem Deklarationsteil der POU.

Beispiel

Der Funktionsbaustein **myPOU** verwendet das Attribut :

```
{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 werden 2 Instanzen des Funktionsbausteins **myPOU** definiert.

Programm:

```
PROGRAM PlcProg
VAR
  POU1, POU2: myPOU;
END_VAR
```

Während nun ein Eingabewert für **POU1** implementiert wird, zeigt "Komponenten Auflisten"-Funktion, die sich beim Tippen von `POU1`. im Implementierungsteil von **PlcProg** öffnet, die Variablen `a` und `b` an, nicht aber die versteckten lokalen Variablen `c` oder `d`.

Siehe auch

- [Kap. "Attribut 'hide'" auf Seite 414](#)

Attribut 'initialize_on_call'

Das Pragma kann auf Eingangsvariablen angewendet werden. Es bewirkt, dass die Eingangsvariablen eines Funktionsbausteins bei jedem Aufruf des Funktionsbausteins initialisiert werden. Wenn ein Eingang einen Zeiger erwartet und dieser Zeiger im Zuge eines Online-Change entfernt wurde, so wird dieser Eingang auf `Null` gesetzt.

Syntax:

```
{attribute 'initialize_on_call'}
```

Einfügeort: Zeile oberhalb Zeile mit der Deklaration der entsprechenden Eingangsvariablen.

Attribut 'init_namespace'

Das Pragma bewirkt, dass eine Variable vom Typ `STRING` oder `WSTRING`, die in einem Bibliotheksbaustein mit diesem Pragma deklariert ist, bei der Verwendung im Projekt mit dem aktuellen Namensraum der Bibliothek initialisiert wird.

Syntax

```
{attribute 'init_namespace'}
```

Einfügeort: Die Zeile oberhalb der Zeile mit der Deklaration der Variablen in einem Bibliotheksbaustein.

Beispiel

Der Funktionsbaustein **POU** ist mit den nötigen Attributen versehen:

```
FUNCTION_BLOCK POU
VAR_OUTPUT
{attribute 'init_namespace'}
myStr: STRING;
END_VAR
```

Innerhalb des Hauptprogramms `PlcProg` ist eine Instanz `fb` des Funktionsbausteins **POU** definiert:

```
PROGRAM PlcProg
VAR
fb:POU;
newString: STRING;
END_VAR
newString := fb.myStr;
```

Die Variable `myStr` wird mit dem aktuellen Namensraum initialisiert, zum Beispiel `MyLib..`

Dieser Wert wird `newString` im Hauptprogramm zugewiesen.

Siehe auch

- [Kap. 18.1.14 "Objekt 'Bibliotheksverwalter'" auf Seite 548](#)

Attribut 'init_on_onlchange'

Das Pragma bewirkt, dass die Variable, auf die das Pragma angewendet wird, bei jedem Online-Change initialisiert wird.

Syntax:

```
{attribute 'init_on_onlchange'}
```

Einfügeort: Die Zeile oberhalb der Zeile mit der Deklaration der Variablen.



Ab IndraWorks 14VRS wurde eine schnelle Version des Online Change für kleine Änderungen eingeführt:

In allen Fällen, in denen nur Code geändert wurde, werden nur die betroffenen POU's übersetzt und nachgeladen, weiterer Code wird nicht ausgeführt.

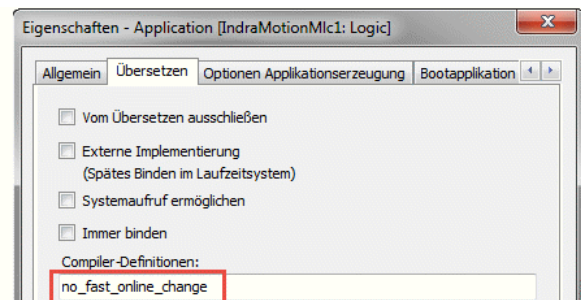
Insbesondere wird kein Initialisierungscode ausgeführt.

Das bedeutet, dass auch kein Code erzeugt wird zur Initialisierung von Variablen mit dem Attribut `init_on_onlchange`.

In der Regel wird das keine Auswirkungen haben, da das Attribut meist dazu verwendet wird, um Variablen mit Adressen zu initialisieren, es kann aber beim schnellen Online-Change nicht dazu kommen, dass eine Variable ihre Adresse ändert.



Um die Wirkung des Attributs `init_on_onlchange` auf den gesamten Applikationscode sicherzustellen, müssen Sie den schnellen Online-Change mit Hilfe der Compiler-Definition `no_fast_online_change` generell für die Applikation ausschalten.



Fügen Sie die Definition zu diesem Zweck in den "Eigenschaften" der Applikation in Registerkarte "Übersetzen" ein.

Attribut 'instance-path'

Das Pragma kann auf eine lokale STRING-Variablen angewandt werden und bewirkt, dass diese lokale STRING-Variable in Folge mit dem Projektpfad der POU, zu der sie gehört, initialisiert wird.

Dies kann für Fehlermeldungen nützlich sein.

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'}
```

Einfügeort: Die Zeile oberhalb der Zeile mit der Deklaration der STRING-Variablen.

Beispiel

Der folgende Funktionsbaustein enthält die Attribute '`reflection`', '`instance-path`' und '`noinit`'.

```
{attribute 'reflection'}
FUNCTION_BLOCK POU
VAR
  {attribute 'instance-path'}
```

```
{attribute 'noinit'}
str: STRING;
END_VAR
```

Im Hauptprogramm **PlcProg** ist eine Instanz **myPOU** des Funktionsbausteins **POU** definiert:

```
PROGRAM PlcProg
VAR
    myPOU:POU;
    myString: STRING;
END_VAR
myPOU();
myString:=myPOU.str;
```

Nach der Initialisierung von Instanz **myPOU** wird der String-Variablen **str** der Pfad der Instanz **myPOU** zugewiesen, im Beispiel **PLCWinNT.Application.PlcProg.myPOU**. Dieser Pfad wird im Hauptprogramm der Variablen **myString** zugewiesen.



Die Länge einer Zeichenkette können Sie beliebig definieren (auch >255), allerdings müssen Sie bedenken, dass die Zeichenkette (von hinten her) abgeschnitten wird, wenn sie einer Variablen zugewiesen ist, deren Datentyp zu klein dafür ist.

Siehe auch

- [Kap. "Attribut 'reflection'" auf Seite 435](#)
- [Kap. "Attribut 'noinit'" auf Seite 426](#)

Attribut 'io_function_block', 'io_function_block_mapping'

Mit dem Attribut **io_function_block** kennzeichnen Sie einen Funktionsbaustein, um ihn für die Zuweisung auf einen Kanal im E/A-Abbild der Gerätekonfiguration bereitzustellen. Er erscheint dann im Dialog Funktionsbaustein auswählen, siehe [Kap. 18.3.1 "Dialog 'Funktionsbaustein auswählen'" auf Seite 744](#).

Mit dem Attribut **io_function_block_mapping** kennzeichnen Sie in einem solchen Funktionsbaustein einen Parameter, der beim Abbilden des FBs auf einen Gerätekanal automatisch verwendet werden soll. Sie können mehrere Parameter des Bausteins mit dem Attribut versehen. Beim E/A-Mapping wird automatisch der erste verwendet, der vom Typ her zum Kanal passt (Eingang, Ausgang, Datentyp).

Syntax:

Programm:

```
{attribute 'io_function_block'}

{attribute 'io_function_block_mapping'}
```

Einfügeort: In der Deklarationszeile oberhalb des Funktionsbausteins, beziehungsweise in der Zeile oberhalb der Parameterdeklaration.

Beispiel

```
{attribute 'io_function_block'}
FUNCTION_BLOCK Scale_Output_Int
VAR_INPUT
    iInput : INT;
    iNumerator : INT;
    iDenominator : INT :=1;
    iOffset : INT := 0;
```

```

END_VAR
VAR_OUTPUT
{attribute 'io_function_block_mapping'}
  iOutput : INT;
END_VAR
VAR

```

Siehe auch [Kap. 18.3.1 "Dialog 'Funktionsbaustein auswählen'" auf Seite 744.](#)

Attribut 'linkalways'

Das Pragma bewirkt, dass das zugehörige Objekt beim Compiler markiert ist und damit immer in der Compilerinformation enthalten ist.

Dies bedeutet, dass das Objekt immer kompiliert und auf die SPS geladen wird.

Das Pragma wirkt für POU's und GVLs, die unterhalb einer Applikation oder in Bibliotheken unterhalb einer Applikation liegen oder im Ordner "Allgemeiner Bausteinteil" enthalten sind.

Die Option **Immer binden** in den Objekteigenschaften, Kategorie **Übersetzen** hat die gleich Wirkung.

Syntax:

```
{attribute 'linkalways'}
```

Einfügeort: Die Zeile oberhalb der Zeile mit dem Schlüsselwort `VAR_GLOBAL` im Deklarationsteil von POU's und GVLs.

Wenn Sie den Symbolkonfigurations-Editor verwenden, verwendet IndraLogic die markierte POU als Basis für die auswählbaren Variablen der Symbolkonfiguration verwendet.

Beispiel

Die Implementierung der globale Variablenliste **GVLMoreSymbols** benutzt das Attribut 'linkalways':

```

{attribute 'linkalways'}
VAR_GLOBAL
  g_iVar1: INT;
  g_iVar2: INT;
END_VAR

```

Durch diesen Code werden die Variablen von **GVLMoreSymbols** als auswählbare Symbole zur Verfügung gestellt.

Siehe auch

- [Kap. 10.2 "Symbolkonfiguration" auf Seite 167](#)

Attribut 'monitoring'

Das Pragma bewirkt, dass Sie Werte von Eigenschaften oder Funktionsaufrufen in der Online-Ansicht des IEC-Editors oder in einer Überwachungsliste monitoren können.

Dazu gibt es zwei mögliche Attributwerte: 'variable' und 'call'

Syntax *Programm:*

```

{attribute 'monitoring' := 'variable'}
{attribute 'monitoring' := 'call'}

```

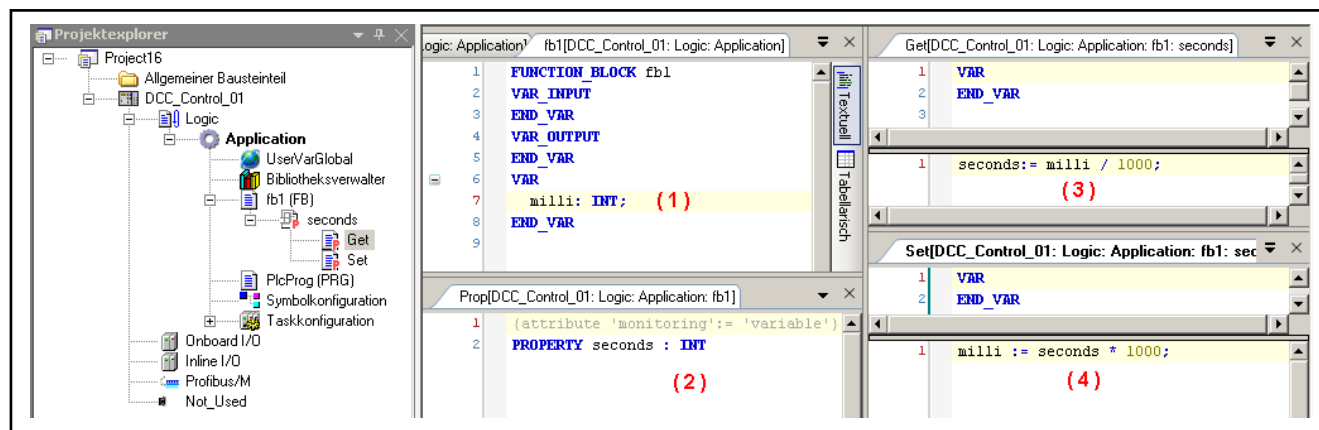
Monitoring von Eigenschaften

Das Pragma muss in der Zeile oberhalb der Definition der Eigenschaft eingefügt werden ([Kap. "Objekt 'Eigenschaft'" auf Seite 571](#)).

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 (Kap. 13.1.2 "Überwachungslisten verwenden" auf Seite 211).

In der Online-Ansicht oder der Überwachungsliste können auch Werte für das Forcen der Eigenschaften-Variablen vorbereitet werden.



- (1) Funktionsbaustein fb1 mit der lokalen Variablen 'milli'
- (2) Eigenschaft 'seconds' mit attribute-Pragma
- (3) Get der Eigenschaft 'seconds'
- (4) Set der Eigenschaft 'seconds'

Abb. 17-80: Beispiel der zum Monitoring vorbereiteten Eigenschaft 'seconds'

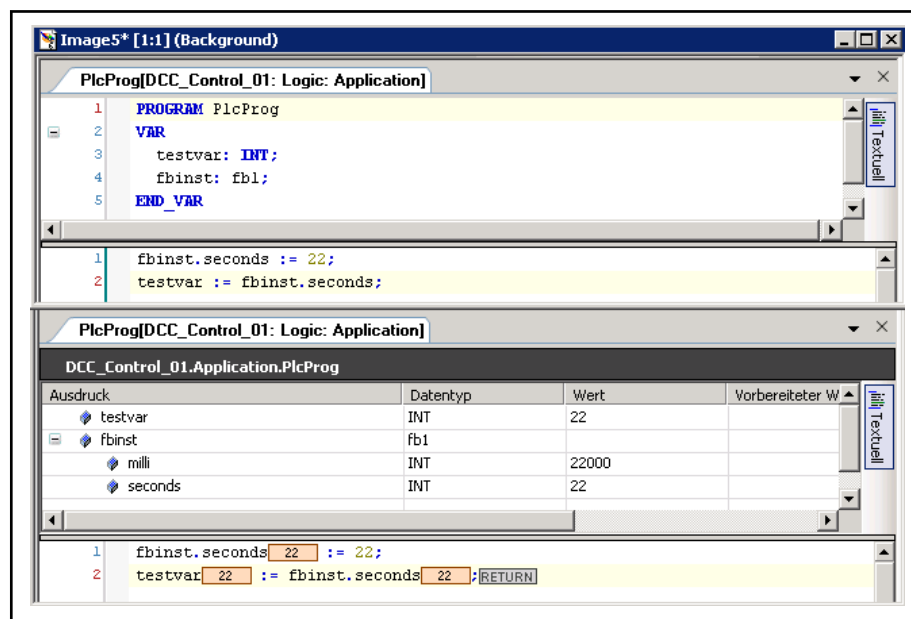


Abb. 17-81: Beispiel einer Monitoring-Ansicht mit Eigenschaft 'seconds'

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. Die Entscheidung 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 Ei-

enschaften-Wert erhält. Der zuletzt in dieser Variablen gespeicherte Wert wird im Monitoring dargestellt.

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 gelesen oder geschrieben, das bedeutet, dass der Monitoring-Dienst des Laufzeitsystems die Get- oder Set-Methode der Eigenschaft-Funktion ausführt.



Wenn dieser Monitoring-Typ gewählt wird anstelle eine implizite Variable zu verwenden (wie bei {attribute 'monitoring':='variable'}), müssen mögliche Seiteneffekte bedacht werden, die auftreten können, wenn innerhalb der Eigenschaft-Funktion zusätzliche Operationen implementiert sind.



Das Pragma `monitoring` wird auch für die Symbolkonfiguration ausgewertet. Beim Wert `variable` ist dabei nur ein Lesezugriff möglich.



Mit dem Kontextmenü-Befehl **Zur Überwachungsliste hinzufügen** wird im Online- Betrieb eine Variable, auf der gerade der Cursor steht, unmittelbar in eine Überwachungsliste aufgenommen.



Forcen oder Schreiben von Funktionen wird nicht unterstützt. Forcen können Sie jedoch implizit implementieren, indem Sie einen zusätzlichen Eingabeparameter für die jeweilige Funktion hinzufügen, der als internes Force-Flag dient.



Funktionsmonitoring ist im Kompaktlaufzeitsystem nicht möglich.

Monitoring der Ergebnisse von Funktionsaufrufen

Monitoring von Funktionsaufrufen kann für jeden konstanten Wert verwendet werden, der als 4-byte-numerischer Wert interpretiert werden kann (z.B. INT, SINT, DINT).

Für alle anderen Eingabeparameter (z.B. BOOL) muss eine Variable anstelle des konstanten Parameters verwendet werden. Das Pragma '{attribute 'monitoring':='call'}' muss in der Zeile oberhalb der Funktionsdeklaration eingefügt werden.

Dann wird in der Online-Ansicht einer POU, in der einer Variable das Ergebnis eines Funktionsaufrufs zugewiesen wird, diese Variable in der Texteditor-Ansicht gemonitored werden. Die Variable kann zum gleichen Zweck auch einer Überwachungsliste hinzugefügt werden ([Kap. 13.1.2 "Überwachungslisten verwenden" auf Seite 211](#)). Man beachte, dass dabei der Befehl "Watchliste" verwendet werden kann, um die Variable unmittelbar in einer Überwachungsliste bereitgestellt zu bekommen

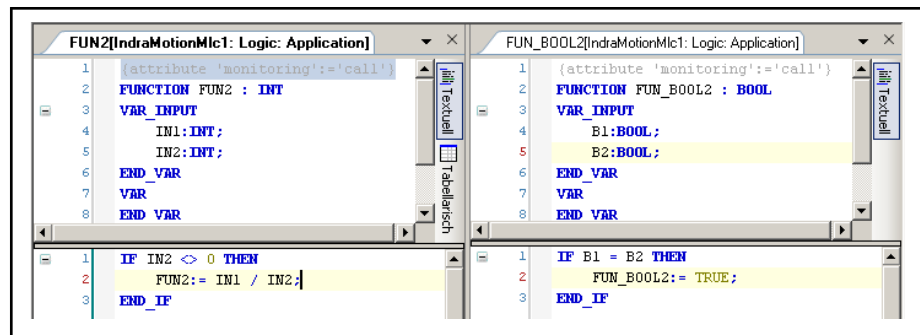
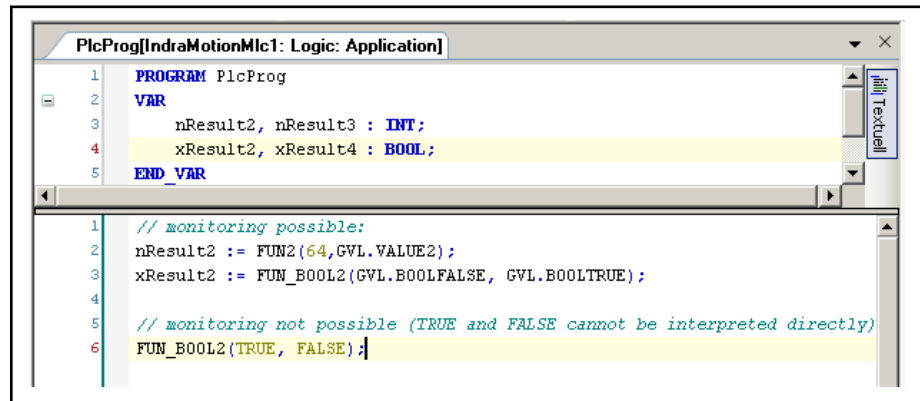
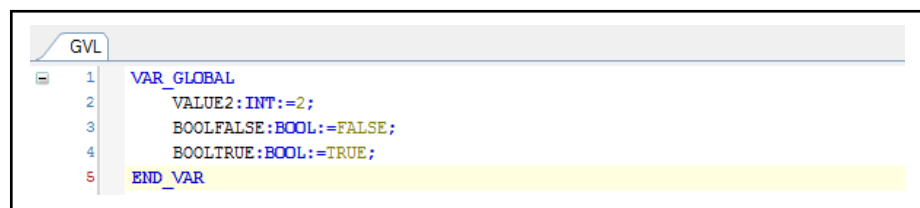
Abb. 17-82: Beispiel, Funktionen `FUN2` und `FUN_BOOL2` mit Attribut 'monitoring'Abb. 17-83: Beispiel, Aufrufe der Funktionen `FUN2` und `FUN_BOOL2` in einer Programm-POU

Abb. 17-84: Konstanten aus der Globalen Variablenliste GVL

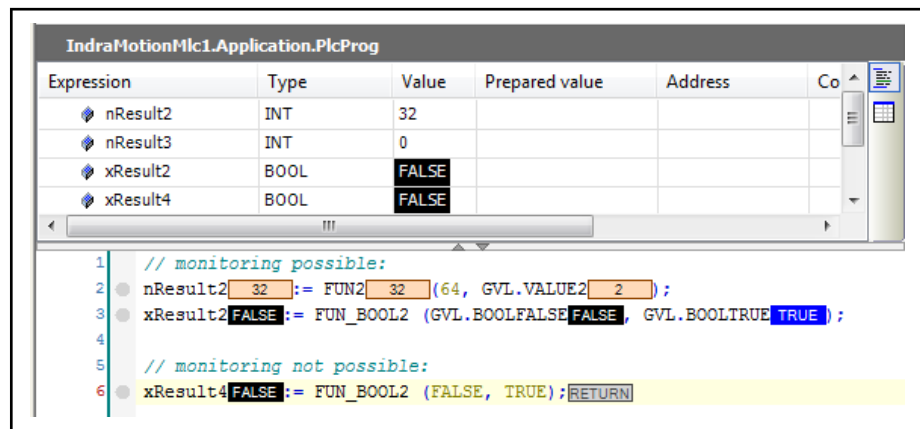


Abb. 17-85: Beispiel, Funktionsaufrufe im Online-Modus

Siehe auch

- Kap. "Objekt 'Eigenschaft'" auf Seite 571

Attribut 'namespace'

In Kombination mit dem Attribut 'symbol' erlaubt das Pragma {attribute 'namespace'} den Namensraum von Projektvariablen umzudefinieren ([Attribut 'symbol' auf Seite 436](#)).

Es kann auf komplette POU's wie GVLs oder Programme angewendet werden, nicht jedoch auf einzelne Variablen. Die betroffenen Variablen werden mit der neuen Namensraumdefinition in eine Symboldatei exportiert und sind nach einem Download auf der Steuerung verfügbar. Dies ermöglicht, auch auf Variablen von Bausteinen oder Visualisierungen zuzugreifen, die ursprünglich einen anderen Namensraum hatten. Beispielsweise kann dadurch eine IndraLogic 1.x-Visualisierung auch in der 2G-Umgebung laufen.

Für Informationen zur "Symbolkonfiguration" siehe [Objekt 'Symbolkonfiguration' auf Seite 588](#).

Eine neue Symboldatei wird bei einem Download oder Online Change des Projekts erzeugt. Sie wird zusammen mit der Applikation auf die Steuerung geladen.

Syntax:

```
{attribute 'namespace' := '<namespace>'}
```

Beispiel einer Namensraumersetzung für die Variablen eines Programms

```
{attribute 'namespace':='prog'}
PROGRAM PLC_PRG
VAR
{attribute 'symbol' := 'readwrite'}
  iVar:INT;
  bVar:BOOL;
END_VAR
```

Wenn auf iVar zuvor beispielsweise über "App1.PLC_PRG.ivar" zugegriffen wurde, ist es nun über "prog.ivar" ansprechbar.

Weitere Ersetzungsbeispiele

Ursprünglicher Namensraum	Variable	Namensraumersetzung	Variablenzugriff im aktuellen Projekt
App1.Lib2.GVL2	Var07	{attribute 'namespace' := ''}	.Var07
App1.GVL2	Var02	{attribute 'namespace' := 'Ext'}	Ext.Var02
App1.GVL2.FB1	Var02	{attribute 'namespace' := 'App1.GVL2'}	App1.GVL2.Var02

Die in der Tabelle gezeigten Ersetzungen bewirken folgende Einträge in der Symboldatei:

```
<NodeList>
  <Node name="">
    <Node name="Var07" type="T_INT" access="ReadWrite">
    </Node>
  </NodeList>

<NodeList>
  <Node name="Ext">
    <Node name="Var02" type="T_INT" access="ReadWrite">
    </Node>
  </NodeList>

<NodeList>
  <Node name="App1">
    <Node name="GVL2">
      <Node name="Var02" type="T_INT" access="ReadWrite">
      </Node>
    </Node>
  </NodeList>
```

Attribut 'no_assign', Attribut 'no_assign_warning'

Das Pragma `no_assign` bewirkt, dass Compilerfehler ausgegeben werden, wenn eine Instanz des Funktionsbausteins einer anderen Instanz desselben Bausteins zugewiesen wird. Solche Zuweisungen sollen oft vermieden werden, wenn der Funktionsbaustein Pointer enthält und Pointer zu Problemen führen, weil sie bei der Wertzuweisung mit kopiert werden.

Das Pragma `no_assign_warning` bewirkt das Gleiche wie das Pragma `no_assign` mit dem Unterschied, dass anstelle eines Compilerfehlers eine Compilerwarnung ausgegeben wird.

Syntax:

```
{attribute 'no_assign'}
```

Einfügeort: Erste Zeile im Deklarationsteil eines Funktionsbausteins.

Beispiel

Zuweisung von Funktionsbausteininstanzen, die Pointer enthalten.

In diesem Beispiel wird die Wertzuweisung der Funktionsbaustein-Instanzen zu Problemen bei der Ausführung von `fb_exit` führen:

```
VAR_GLOBAL
inst1 : TestFB;
  awsBufferLogFile : ARRAY [0..9] OF WSTRING(66); (* Area: 0, Offset: 0x1304 (4868)*)
  LogFile : SEDL.LogRecord := (sFileName := 'LogFile.log', pBuffer := ADR(awsBufferLogFile),
    udiMaxEntriesFile := UDINT#10000, udiMaxBuffered := UDINT#10,
    uiLineSize := UINT#64, wsSep := " ", xCircular := TRUE,
    siDateFormat := SINT#0, siTimeFormat := SINT#0);

END_VAR
PROGRAM PlcProg
VAR
  inst2 : TestFB := inst1;
  LogFileNew
END_VAR
```

In diesem Fall verwaltet `LogRecord` eine Liste von Pointern, für die im Fall von `fb_exit` verschiedene Aktionen ausgeführt werden. Durch die Zuweisung entstehen Probleme, weil `fb_exit` zweimal ausgeführt werden wird. Sie sollten dies verhindern, indem Sie das Attribut `'no_assign'` in der Deklaration des Funktionsbausteins **TestFB** hinzufügen:

```
{attribute 'no_assign'}
FUNCTION_BLOCK TestFB
VAR_INPUT
...
```

Dann werden die folgenden Compile-Fehler ausgegeben:

```
C0328: Assignment not allowed for type TestFB
C0328: Assignment not allowed for type LogRecord
```

Wenn anstelle des Pragmas `no_assign` das Pragma `no_assign_warning` für den Funktionsbaustein `TestFB` verwendet wird, werden ebenfalls die Meldungen `C0328` ausgegeben, jedoch nicht als Compilerfehler, sondern als Compilerwarnung.

Attribut 'no_check'

Das Pragma bewirkt, dass für die POU keine Check-Funktion (POUs für implizite Prüfungen) aufgerufen werden soll.

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.

Sie fügen das Pragma der Deklaration einer POU hinzu.

Syntax:

```
{attribute 'no_check'}
```

Einfügeort: Erste Zeile im Deklarationsteil der POU



Das Attribut wirkt automatisch auch auf die Kindobjekte einer POU!

Beispiel:

Wenn das Attribut in einem Programm eingetragen ist, werden auch für Aktionen, die diesem Programm zugewiesen sind, keine Check-Funktionen ausgeführt.

Siehe auch

- [Kap. 18.1 "Objekte" auf Seite 507](#)

Attribut 'no_copy'

Im allgemeinen erfordert ein Online-Change eine Reallozierung von Instanzen, zum Beispiel einer POU. Dabei wird der Wert der in der Instanz enthaltenen Variablen kopiert.

Das Pragma bewirkt, dass im Zuge eines Online-Changes keine Kopie des Variablenwerts der in der Instanz enthaltenen Variablen stattfindet; stattdessen wird die Variable im Zuge eines Online-Changes neu initialisiert.

Das kann für eine lokale Zeigervariable sinnvoll sein, die auf eine Variable zeigt, die gerade durch den Online-Change verschoben wurde und somit eine veränderte Adresse besitzt.

Sie fügen das Attribut im Deklarationsteil oberhalb der Zeile der Deklaration der betroffenen Variablen ein.

Syntax:

```
{attribute 'no_copy'}
```

Attribut 'no-exit'

Das Attribut unterdrückt den Aufruf der FB_exit-Methode eines Funktionsbausteins für eine bestimmte seiner Instanzen. Dazu fügen Sie das Attribut in der Zeile vor der Deklaration der Funktionsbaustein-Instanz ein.

Syntax:

```
{attribute 'no-exit'}
```

Beispiel

Dem Funktionsbaustein **POU_ex** ist die Methode **FB_exit** hinzugefügt. Im Hauptprogramm **PlcProg** werden 2 Instanzen des Funktionsbausteins **POU_ex** angelegt.

Programm:

```
PROGRAM PlcProg
VAR
POU1 : POU_ex;
{attribute 'no-exit'}
POU2 : POU_ex;
END_VAR
```

POU 1: FB_exit wird aufgerufen, POU2: FB_exit wird nicht aufgerufen.

Siehe auch

- [Kap. 17.10 "Methode 'FB_Init', 'FB_Reinit' und 'FB_Exit'" auf Seite 455](#)

Attribut 'noinit'

Das Pragma wird auf Variablen angewendet, die nicht implizit initialisiert werden sollen.

Syntax:

```
{attribute 'no_init'}
{attribute 'no-init'}
{attribute 'noinit'}
```

Einfügeort: Zeile oberhalb der Deklarationszeile der betroffenen Variablen im Deklarationsteil.

Beispiel

```
PROGRAM PlcProg
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_instance_in_retain'

Mit diesem Attribut können Sie verhindern, dass die Instanz eines bestimmten Funktionsbausteins in den Retain-Bereich gespeichert wird.

Syntax:

```
{attribute 'no_instance_in_retain'}
```

Einfügeort:

Zeile oberhalb der `FUNCTION_BLOCK`-Deklaration im Deklarationsteil des Funktionsbausteins.

Es wird eine Fehlermeldung ausgegeben, wenn eine Instanz des Bausteins als RETAIN-Variable deklariert ist.

Siehe auch

- [Kap. 17.2.11 "Persistente Variable - PERSISTENT" auf Seite 308](#)

Attribut 'no_virtual_actions'

Das Pragma wird für Funktionsbausteine angewendet, die von einem in AS implementierten Funktionsbaustein abgeleitet werden und den grundsätzlichen AS-Ablauf dieser Basisklasse nutzen. Die daraus aufgerufenen Aktionen zeigen dasselbe virtuelle Verhalten wie Methoden. Dies bedeutet, dass die Implementierungen der Aktionen in der Basisklasse von der abgeleiteten Klasse durch eigene, spezifische Implementierungen ersetzt werden können.

Wenn Sie das Pragma auf die Basisklasse anwenden, dann sind ihre Aktionen vor einem Überladen geschützt.

Syntax:

```
{attribute 'no_virtual_actions'}
```

Einfügeort: Oberste Zeile im Deklarationsteil des Funktionsbausteins

Beispiel:

Im folgenden Beispiel bildet der Funktionsblock `POU_SFC` die Basisklasse für den abgeleiteten Funktionsblock `POU_child`.

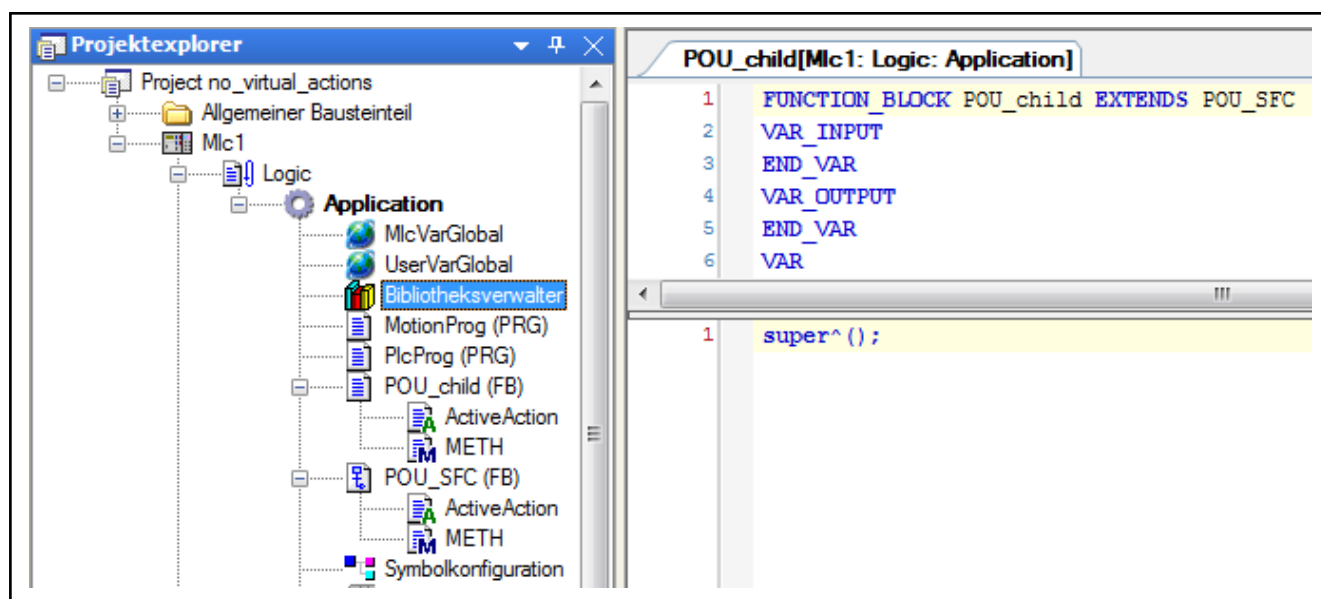


Abb. 17-86: FB POU_child erweitert FB POU_SFC

Durch Benutzung des Schlüsselworts SUPER ruft die abgeleitete Klasse (FB) POU_child den in SFC geschriebenen Ablauf der Basisklasse (Basis-FB) auf.

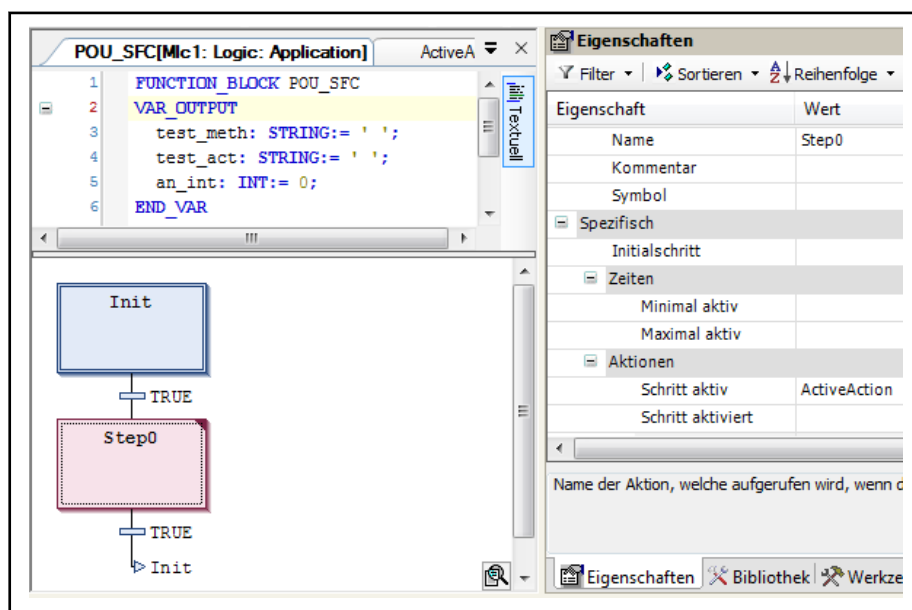


Abb. 17-87: Funktionsbaustein POU_SFC mit Aktion ActiveAction

Die beispielhafte Implementierung dieses Ablauf ist beschränkt auf den Initialschritt gefolgt von einem einzigen Schritt "Step0" mit angebundener Schrittaktion "ActiveAction", welche die Belegung der Ausgangsvariablen übernimmt:

ActiveAction (POU_SFC)

```
an_int:=an_int+1; // counting the action calls
test_act:='father_action'; // writing test_act
METH(); // Calling method METH for writing test_meth
```

METH (POU_SFC)

```
METHOD PUBLIC METH
VAR_INPUT
```

```
END_VAR
test_meth:= 'father_meth';
```

Im Fall der abgeleiteten Klasse POU_child wird die Schrittktion 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.

Die Aktion und die Methode müssen manuell POU_child namensgleich!! zugeordnet werden.

ActiveAction (POU_child)

```
an_int:=an_int+1; // counting the action calls
test_act:='child_action'; // writing test_act
METH(); // Calling method METH for writing test_meth
```

Ebenso wird die Methode "METH", welche in der Basisklasse der Variablen test_meth den String 'father_method' zuweist, dahingehend überschrieben, dass "test_meth" nun den Wert 'child_method' erhält.

METH (POU_child)

```
METHOD PUBLIC METH
VAR_INPUT
END_VAR
test_meth:= 'child_method';
```

Das Hauptprogramm "PlcProg" ruft eine Instanz des Funktionsbausteins "POU_child" namens "child" auf. Wie erwartet spiegelt der Wert der Strings den Aufruf von Aktion und Methode der abgeleiteten Klasse wider:

Ausdruck	Datentyp	Wert	Vorbereitet
child	POU_child		
test_meth	STRING	'child_method'	
test_act	STRING	'child_action'	
an_int	INT	4494	

Abb. 17-88: Hauptprogramm PlcProg online

Wird der Basis jedoch das Attribut 'no_virtual_actions' vorangestellt, so kann ein anderes Verhalten beobachtet werden:

POU_SFC mit {attribute 'no_virtual_actions'}

```
{attribute 'no_virtual_actions'}
FUNCTION_BLOCK POU_SFC
VAR_OUTPUT
    test_meth: STRING:= ' ';
    test_act: STRING:= ' ';
    an_int: INT:= 0;
END_VAR
```

Während für Methode METH weiterhin die Implementierung der abgeleiteten Klasse herangezogen wird, resultiert der Aufruf der Schrittktion nun in einem Aufruf der Aktion "ActiveAction" der Basisklasse. Daher erhält "test_act" den Wert 'father_action':

Ausdruck	Datentyp	Wert	Vorbereitete
child	POU_child		
test_meth	STRING	'child_method'	
test_act	STRING	'father_action'	
an_int	INT	449	

Abb. 17-89: Hauptprogramm PlcProg online, Nutzung von Attribut 'no_virtual_actions' für FUNCTION_BLOCK POU_SFC

Attribut 'pingroup'

Das Pragma bewirkt, dass in der Deklaration eines Funktionsbausteins die Eingangspins oder Ausgangspins (Parameter) gruppiert werden.

Im FUP/KOP-Editor kann dann eine so definierte Pingruppe am eingefügten Baustein als Einheit reduziert und erweitert angezeigt werden.

Mehrere Gruppen sind möglich und werden durch ihre Namen unterschieden.

IndraLogic speichert den jeweiligen Zustand (reduziert) pro Bausteinbox mit den Projektoptionen.

Syntax:

```
{attribute 'pingroup' := '<group name>'}
```

Einfügeort: Zeile oberhalb der Deklaration der betroffenen Eingangs- oder Ausgangsvariablen im Deklarationsteil eines Funktionsbausteins.

Beispiel

Zwei Gruppen werden definiert: general (i1, out1) und group1 (i2, g1). r1, r2, outRes1 und g2 werden immer angezeigt

```
FUNCTION_BLOCK FB
VAR_INPUT
    r1 : REAL;
    {attribute 'pingroup' := 'general'}
    i1 : INT;
    {attribute 'pingroup' := 'group1'}
    i2 : INT;
    r2 : REAL;
END_VAR
VAR_OUTPUT
    outRes1 : REAL;
    {attribute 'pingroup' := 'general'}
    out1 : INT;
    {attribute 'pingroup' := 'group1'}
    g1 : INT;
    g2 : REAL;
END_VAR
```

Attribut 'pin_presentation_order_inputs/outputs'

Die Pragmas definieren die angezeigte Reihenfolge der Ein-/ Ausgänge eines Funktionsbausteins.

Syntax Programm:

```
{attribute 'pin_presentation_order_inputs' := '<First_Input_Name>, (<Next_Input_Name>,)*
( *, )? (<Next_Input_Name>)* <Last_Input_Name>'}
{attribute 'pin_presentation_order_outputs' := '<First_Output_Name>, (<Next_Output_Name>,)*
( *, )? (<Next_Output_Name>)* <Last_Output_Name>'}
```

• *

Das Terminalzeichen dient als Platzhalter für alle Eingänge/Ausgänge, die selbst nicht in der Darstellungsreihenfolge angegeben sind. Wenn

das Terminalzeichen fehlt, werden die fehlenden Eingänge/Ausgänge an das Ende angehängt.

- (...)?
Der Inhalt der runden Klammer ist optional.
- Einfügeort: Erste Zeile im Deklarationsteil eines Funktionsbaustein



Dieses Pragma wird nicht ausgewertet, wenn das Pragma {attribute 'pingroup' := '<Group_Name>'} verwendet wird.

Beispiel

```
{attribute 'pin_presentation_order_inputs' := 'input_2,*,input_1'}
{attribute 'pin_presentation_order_outputs' := 'output_2, output_1'}
FUNCTION_BLOCK POU_BASE
VAR_INPUT
    input_1 : BOOL;
    input_2 : INT;
    input_3 : INT;
    input_4 : INT;
END_VAR
VAR_OUTPUT
    output_1 : BOOL;
    output_2 : INT;
    output_3 : INT;
    output_4 : BOOL;
END_VAR
FUNCTION_BLOCK PLC_PRG
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    pouBase_A: POU_BASE;
END_VAR
```

Die Pragmas bewirken in der Darstellung der Funktionsbaustein-Instanz pouBase_A folgende Anordnung der Eingangs- und Ausgangspins:

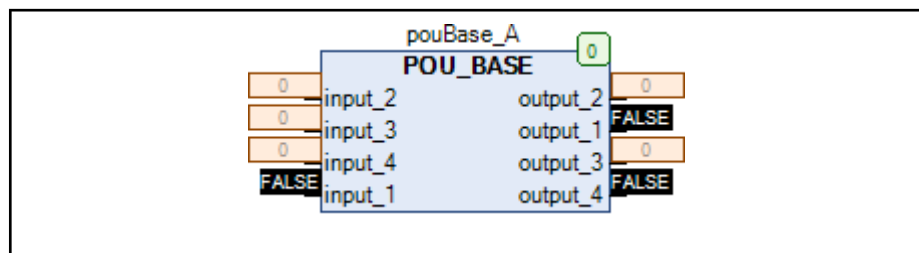


Abb. 17-90:

Siehe [Kap. "Attribut 'pingroup'"](#) auf Seite 429

Attribut 'obsolete'

Das Pragma bewirkt, dass zu einer Datentyp-Definition beim Übersetzen eine definierte Warnung ausgegeben wird, wenn der Datentyp (Struktur, Funktionsbaustein etc.) im Projekt verwendet wird. Damit können Sie beispielsweise darauf hinweisen, 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 Meldungspragma wird diese Warnung zentral für alle Instanzen eines Datentyps definiert.

Syntax:

```
{attribute 'obsolete' := 'user defined text'}
```

Einfügeort: Zeile der Datentyp-Definition oder in einer Zeile darüber.

Beispiel

Das Pragma wird in der Definition Funktionsbaustein `fb1` eingefügt:

```

{attribute 'obsolete' := 'datatype fb1 not valid!'}
FUNCTION_BLOCK fb1
VAR_INPUT
    i:INT;
END_VAR

```

Wenn Sie `fb1` als Datentyp verwenden, zum Beispiel in `fbinst:fb1`; wird beim Übersetzen des Projekts die Warnung ausgegeben: "datatype fb1 not valid".

Siehe auch

- [Kap. 17.6.1 "Meldungspragmas" auf Seite 402](#)

Attribut 'pack_mode'

Das Pragma legt fest, wie eine Datenstruktur während der Allokierung 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>'}
```

Einfügeort: oberhalb der Deklaration der Datenstruktur

Mögliche Werte für <Value>

pack_mode	Assoziierte Packungsart	Beschreibung
0	aligned	Es liegen alle Variablen auf Byte-Adressen; es treten keine Speicherlücken auf.
1	1-bytealigned	
2	2-bytealigned	<i>Es liegen</i> • 1 Byte-Variablen auf Byte-Adressen • 2 Byte-Variablen auf durch 2-teilbare Adressen. Es entsteht maximal eine Lücke von 1 Byte • 4 Byte-Variablen auf durch 2-teilbare Adressen. Es entsteht maximal eine Lücke von 1 Byte • 8 Byte-Variablen auf durch 2-teilbare Adressen. Es entsteht maximal eine Lücke von 1 Byte • Strings immer an Byte-Adressen. Es entsteht keine Lücke.

pack_mode	Assoziierte Packungsart	Beschreibung
4	4-bytealigned	<i>Es liegen</i> • 1 Byte-Variablen auf Byte-Adressen • 2 Byte-Variablen auf durch 2-teilbare Adressen. Es entsteht maximal eine Lücke von 1 Byte • 4 Byte-Variablen auf durch 4-teilbare Adressen. Es entsteht maximal eine Lücke von 3 Byte • 8 Byte-Variablen auf durch 4-teilbare Adressen. Es entsteht maximal eine Lücke von 3 Byte • Strings immer an Byte-Adressen. Es entsteht keine Lücke.
8	8-bytealigned	<i>Es liegen</i> • 1 Byte-Variablen auf Byte-Adressen • 2 Byte-Variablen auf durch 2-teilbare Adressen. Es entsteht maximal eine Lücke von 1 Byte • 4 Byte-Variablen auf durch 4-teilbare Adressen. Es entsteht maximal eine Lücke von 3 Byte • 8 Byte-Variablen auf durch 8-teilbare Adressen. Es entsteht maximal eine Lücke von 7 Byte • Strings immer an Byte-Adressen. Es entsteht keine Lücke.



Abhängig vom Aufbau der Struktur kann es somit vorkommen, dass es keinen Unterschied in der Speicheraufteilung zwischen den einzelnen Modi gibt. Somit kann die Speicherverteilung einer Struktur mit `pack_mode= 4` der von `pack_mode= 8` entsprechen.



Arrays von Strukturen:

Werden die Strukturen in Arrays zusammengefasst, so werden am Ende der Struktur Bytes eingefügt, damit die nächste Struktur wieder aligned ist.

Beispiel 1

```
{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 beispielsweise 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 0x0106.

Beispiel 2

```

STRUCT
Var1 : BOOL := 16#01;
Var2 : BYTE := 16#11;
Var3 : WORD := 16#22;
Var4 : BYTE := 16#44;
Var5 : DWORD := 16#88776655;
Var6 : BYTE := 16#99;
Var7 : BYTE := 16#AA;
Var8 : DWORD := 16#AA;
END_TYPE

```

	pack_mode= 0		pack_mode= 1		pack_mode= 2		pack_mode= 4		pack_mode= 8	
	Variable	Wert	Variable	Wert	Variable	Wert	Variable	Wert	Variable	Wert
0	Var 1	01	Var 1	01	Var 1	01	Var 1	01	Var 1	01
1	Var 2	11	Var 2	11	Var 2	11	Var 2	11	Var 2	11
2	Var 3	22	Var 3	22	Var 3	22	Var 3	22	Var 3	22
3	...	00	...	00	...	00	...	00	...	00
4	Var 4	44	Var 4	44	Var 4	44	Var 4	44	Var 4	44
5	Var 5	55	Var 5	55						
6	...	66	...	66	Var 5	55				
7	...	77	...	77	...	66				
8	...	88	...	88	...	77	Var 5	55	Var 5	55
9	Var 6	99	Var 6	99	...	88	...	66	...	66
10	Var 7	AA	Var 7	AA	Var 6	99	...	77	...	77
11	Var 8	AA	Var 8	AA	Var 7	AA	...	88	...	88
12	...	00	...	00	Var 8	AA	Var 6	99	Var 6	99
13	...	00	...	00	...	00	Var 7	AA	Var 7	AA
14	...	00	...	00	...	00				
15					...	00				
16							Var 8	AA	Var 8	AA
17							...	00	...	00
18							...	00	...	00
19							...	00	...	00

Beispiel 3

```

STRUCT
Var1 : BYTE := 16#01;
Var2 : LWORD := 16#11;
Var3 : BYTE := 16#22;
Var4 : BYTE := 16#44;
Var5 : DWORD := 16#88776655;
Var6 : BYTE := 16#99;
Var7 : BYTE := 16#AA;
Var8 : WORD := 16#AA;
END_TYPE

```

Referenz Programmierung

	pack_mode= 0		pack_mode= 1		pack_mode= 2		pack_mode= 4		pack_mode= 8	
	Variable	Wert	Variable	Wert	Variable	Wert	Variable	Wert	Variable	Wert
0	Var 1	01	Var 1	01	Var 1	01	Var 1	01	Var 1	01
1	Var 2	11	Var 2	11						
2	...	00	...	00	Var 2	11				
3	...	00	...	00	...	00				
4	...	00	...	00	...	00	Var 2	11		
5	...	00	...	00	...	00	...	00		
6	...	00	...	00	...	00	...	00		
7	...	00	...	00	...	00	...	00		
8	...	00	...	00	...	00	...	00	Var 2	11
9	Var 3	22	Var 3	22	...	00	...	00	...	00
10	Var 4	44	Var 4	44	Var 3	22	...	00	...	00
11	Var 5	55	Var 5	55	Var 4	44	...	00	...	00
12	...	66	...	66	Var 5	55	Var 3	22	...	00
13	...	77	...	77	...	66	Var 4	44	...	00
14	...	88	...	88	...	77			...	00
15	Var 6	99	Var 6	99	...	88			...	00
16	Var 7	AA	Var 7	AA	Var 6	99	Var 5	55	Var 3	22
17	Var 8	AA	Var 8	AA	Var 7	AA	...	66	Var 4	44
18	...	00	...	00	Var 8	AA	...	77		
19					...	00	...	88		
20							Var 6	99	Var 5	55
21							Var 7	AA	...	66
22							Var 8	AA	...	77
23							...	00	...	88
24									Var 6	99
25									Var 7	AA
26									Var 8	AA
27									...	00

Attribut 'qualified_only'

Das Pragma bewirkt, dass Variablen einer globalen Variablenliste nur durch Angabe des globalen Variablennamens angesprochen werden, zum Beispiel `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'}
```

Einfügeort: Zeile oberhalb von `VAR_GLOBAL` in einer GVL

Beispiel**Globale Variablenliste *GVL*:**

```
{attribute 'qualified_only'}
VAR_GLOBAL
    iVar:INT;
END_VAR
```

Innerhalb einer POU, zum Beispiel **PlcProg**, kann die globale Variable `iVar` nur unter Verwendung des Präfix `GVL` angesprochen werden:

```
GVL.iVar:=5;
```

Der folgende unvollständige Aufruf der Variablen wird hingegen einen Fehler erzeugen:

```
iVar:=5;
```

Attribut 'reflection'

Das Pragma dient dazu Bausteine zu kennzeichnen, in denen IndraLogic nach lokalen STRING-Variablen suchen soll, auf die das Attribut '`instance-path`' angewendet wird.

Syntax:

```
{attribute 'reflection'}
```

Ein Beispiel finden Sie in der Beschreibung des Attributs '`instance-path`'

Siehe auch

- [Kap. "Attribut 'instance-path'" auf Seite 417](#)

Attribut 'to_string'

Das Pragma wirkt sich darauf aus, wie das Ergebnis der Konvertierung einer Enumerationskomponente mit dem Operator `TO_STRING` ausgegeben wird: Wenn die Enumerationsdeklaration mit dem Pragma versehen ist, erscheint anstelle des numerischen Werts der Name der Enumerationskomponente als Zeichenfolge.

Syntax

```
{attribute 'to_string'}
```

Einfügeort

Erste Zeile über dem Deklarationsteil der Enumeration.

Beispiel

Deklaration der Enumeration `color`:

```
{attribute 'to_string'}
TYPE color :
(
    red := 0,
    blue := 1,
    green := 2
);
END_TYPE
```

Konvertierung mit `TO_STRING`:

```
PROGRAM PLC_PRG
VAR
    i_color: Color;
    s_show_color: STRING;
END_VAR
```

```
i_color := 1;
s_show_color := TO_STRING(i_color);
```

In diesem Fall erhält `str_show_color` als Konvertierungsergebnis den Wert 'blue' anstelle von '1'.

Siehe auch [Kap. 17.5.17 "Enumerationen" auf Seite 398](#)

Attribut 'strict'

Das Pragma sichert die korrekte Verwendung von Aufzählungstypen.

Dazu erhält ab IndraLogic 14VRS jede Enumerationen, die Sie einem Projekt neu hinzufügen, automatisch ein Attribut 'strict' in der Zeile oberhalb der TYPE-Deklaration.

Dies bewirkt, dass bei einer Compiler-Version $\geq 3.5.7.0$ in folgenden Fällen Übersetzungsfehler entstehen:

- Arithmetische Operation mit Variablen des Enumerationstyps
- Zuweisung eines konstanten Werts, der nicht einem Enumerationswert entspricht, zu einer Variablen des Enumerationstyps
- Zuweisung eines nicht-konstanten Werts, der einen anderen Datentyp hat als der Enumerationstyp, zu einer Variablen des Enumerationstyps.

Das Attribut kann auch explizit hinzugefügt oder entfernt werden.

Syntax:

```
{attribute 'strict'}
```

Attribut 'subsequent'

Das Pragma wird verwendet, um Variablen direkt hintereinander an einem Speicherplatz zu allozieren. Wenn sich die Liste ändert, wird die gesamte Variablenliste an einem neuen Speicherplatz alloziiert.

Dieses Pragma wird in Programmen und globalen Variablenlisten verwendet.

Syntax:

```
{attribute 'subsequent'}
```



`VAR_TEMP` in einem Programm mit Attribut 'subsequent' führt zu einem Compiler-Fehler.



Wenn eine Variable in der Liste 'RETAIN' ist, wird die gesamte Liste als 'RETAIN' gespeichert.

Attribut 'symbol'

Das Pragma `{attribute 'symbol'}` definiert, welche Variablen eines Programms oder einer globalen Variablenliste in die Symbolkonfiguration aufgenommen werden sollen. Das bedeutet, dass die Variablen als Symbole in eine Symbolliste exportiert werden.

Diese Symbolliste steht dann sowohl als XML-Datei im Projektverzeichnis, als auch in einer für den Anwender nicht sichtbaren Datei auf dem Zielsystem für externen Zugriff bereit.

Beispielsweise steht die Symbolliste dann für den Zugriff durch einen OPC-Server bereit. Die so mit einem Symbol ausgestatteten Variablen lädt IndraLogic auf die Steuerung, selbst wenn sie im Editor der Symbolkonfiguration nicht explizit konfiguriert oder sichtbar sind.



Beachten Sie, dass die Symbolkonfiguration als Objekt unterhalb der betreffenden Applikation im Projektnavigator angelegt sein muss.



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.

Syntax:

```
{attribute 'symbol' := 'none'|'read'|'write'|'readwrite'}  
{attribute 'symbol' := '<access possibilities>'}
```

<access possibilities>: none, read, write, readwrite.

Wenn kein Parameter angegeben wird, gilt der Standardwert `readwrite`.

Einfügeort:

- Um nur auf eine einzelne Variable zu wirken, müssen Sie das Pragma in der Zeile vor der Variablendeklaration platzieren.
- Um für alle Variablen im Deklarationsteil eines Programms wirksam zu werden, müssen Sie das Pragma in der ersten Zeile des Deklarationsteils platzieren. Auch in diesem Fall können Sie dann noch Anweisungen für einzelne Variablen explizit in der jeweiligen Zeile setzen.

Welcher **Zugriff** auf ein Symbol möglich ist, wird durch den Parameter bestimmt. Mögliche Angaben: 'none', 'read', 'write' oder 'readwrite'. Wenn kein Parameter angegeben wird, gilt der Standardwert 'readwrite'.

Beispiel Einzelvariable:

Mit der folgenden Konfiguration werden die Variablen "A" und "B" mit Lese- und Schreibrecht exportiert. Variable "D" wird mit Leserecht exportiert.

Beispiele:

```
{attribute 'symbol' := 'readwrite'}  
PROGRAM PlcProg  
VAR  
    A : INT;  
    B : INT;  
    {attribute 'symbol' := 'none'}  
    C : INT;  
    {attribute 'symbol' := 'read'}  
    D : INT;  
END_VAR
```

Beispiel Strukturelement:

Soll eine Strukturvariable in das Symbolfile exportiert werden, kann in der Struktur für jede einzelne Variable der Zugriff definiert werden.

Typdefinition einer Struktur

```
TYPE ST_Symbols :  
STRUCT  
    {attribute 'symbol' := 'read'}  
    iReadOnly: INT;  
    {attribute 'symbol' := 'write'}  
    iWriteOnly: INT;  
    {attribute 'symbol' := 'readwrite'}  
    iReadWrite: INT;  
    {attribute 'symbol' := 'none'}  
    iNoAccess: INT;  
END_STRUCT  
END_TYPE
```

Sobald in der Struktur die Zugriffsrechte definiert sind, überschreiben diese Zugriffsrechte die Zugriffsrechte der Variablen (stTest), die diese Struktur verwendet.

Es ist dabei gleichgültig, welches Attribut die Variable stTest besitzt (Attribut 'read', 'write' oder 'readwrite').

Deklaration der Strukturinstanz

```
VAR
  {attribute 'symbol' := 'read'}
  stTest: ST_Symbols;
END_VAR
```

Siehe auch

- [Kap. 9.6 "Pragmas verwenden" auf Seite 114](#)
- [Kap. 10.2 "Symbolkonfiguration" auf Seite 167](#)

Attribut 'warning disable', Attribut 'warning restore'

Das Pragma bewirkt, dass bestimmte Warnungen unterdrückt werden. Das Pragma `warning restore` bewirkt, dass eine unterdrückte Meldung wieder aktiviert wird.

Syntax:

```
{warning disable <compiler ID>}
{warning restore <compiler ID>}
```

<compiler ID>: ID, die am Anfang einer Fehler- oder Warnungsmeldung steht.

Beispiel

Compiler-Meldung:

```
typify code ...
C0196: Implicit conversion from unsigned Type 'UINT' to signed Type 'INT' : possible change of sign
Compile complete -- 0 errors
```

Das Pragma auf eine Variablendeklaration anwenden:

```
VAR
  {warning disable C0195}
  test1 : UINT := -1;
  {warning restore C0195}
  test2 : UINT := -1;
END_VAR
```

test1 erzeugt keine Fehlermeldung, test2 erzeugt eine Fehlermeldung.

Attribut 'enable_dynamic_creation'

Das Pragma `enable_dynamic_creation` wird benötigt, um den Operator `__NEW` bei Funktionsbausteinen zu verwenden.

Syntax: {attribute 'enable_dynamic_creation'}

Einfügeort: Erste Zeile in der Deklaration des Funktionsbausteins.

Siehe auch [Kap. 17.3.58 "Operator '__NEW'" auf Seite 349](#)

Attribut 'ProcessValue'

Mit dem Attribut 'ProcessValue' kennzeichnen Sie eine Komponente einer Struktur. Im CFC-Editor können Sie dann den Befehl "Attributierte Komponente als Eingang verwenden" anwenden, um diese Struktur mit einem Eingang skalaren Typs zu verbinden.

Syntax {attribute 'ProcessValue'}

Einfügeort: Zeile oberhalb der betroffenen Strukturvariablen.

Beispiel *Programm:*

```

TYPE QINT :
STRUCT
    Status : STRING;
    {attribute 'ProcessValue'}
    Value1 : INT;
    Value2 : INT;
END_STRUCT
END_TYPE

```

Siehe auch

- [Kap. "Befehl 'Attributierte Komponente als Eingang verwenden'" auf Seite 701](#)

17.6.3 Bedingte Pragmas

Bedingte Bedingte Pragmas dienen dazu, die Codegenerierung im Vorübersetzungsprozess oder Übersetzungsprozess zu beeinflussen. Die Implementierungssprache ST unterstützt diese Pragmas.



Sie verwenden bedingte Pragmas in den Implementierungen von POU's.

Wenn Sie die bedingten Pragmas im Deklarationsteil verwenden, wertet IndraLogic diese Pragmas nicht aus!

Mit bedingten Pragmas beeinflussen Sie, ob Implementierungscode für die Übersetzung berücksichtigt wird. Dies können Sie beispielsweise davon abhängig machen ob eine bestimmte Variable deklariert ist, ob ein bestimmter Baustein vorhanden ist etc.

Pragma	Beschreibung
{define <identifier> <string>}	Der Wert kann später mit <code>hasvalue</code> abgefragt und verglichen werden.
{undefine <identifier>}	Die {define}- Anweisung des Bezeichners <identifier> 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 übersetzt; die anderen Zeilen werden ignoriert. Die Reihenfolge der Abschnitte ist festgelegt. Die ELSIF und ELSE Abschnitte sind optional. Die ELSIF Abschnitte dürfen beliebig oft vorkommen. Innerhalb der Konstanten <expr> können Sie mehrere bedingte Übersetzungsoperatoren verwenden.
<expr>}	Innerhalb des konstanten Ausdrucks <expr> innerhalb eines bedingten Übersetzungspragmas {IF} oder {ELSIF} können Sie einen oder mehrere Operatoren verwenden.



Sie können Ausdrücke und `define`-Definitionen als **Compilerdefinitionen** in der Registerkarte **Übersetzen** im Eigenschaftendialog von POU's eintragen. Wenn Sie `define`-Definitionen im Eigenschaftendialog eintragen, müssen Sie das den Begriff `{define}` weglassen, im Gegensatz zur Definition im Implementierungscod. Außerdem können Sie im Eigenschaftendialog mehrere `define`-Definitionen durch Kommas getrennt angeben.

Siehe auch

- [Kap. 9.3.3 "Strukturierter Text \(ST\), Erweiterter Strukturierter Text \(ExST\)" auf Seite 104](#)

Operator defined (<identifier>)

Der Operator bewirkt, dass der Ausdruck den Wert `TRUE` erhält. Voraussetzung ist, dass 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)}
(* This code is processed in App1 *)
{info 'pdef1 defined'}
  hugo := hugo + SINT#1;
{ELSE}
(* the following code is only processed in App2 *)
{info 'pdef1 not defined'}
  hugo := hugo - SINT#1;
{END_IF}
```

Hier ist zusätzlich ein Beispiel eines Meldungspragmas enthalten: Nur die Meldung `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

Operator defined (variable: <variable>)

Der 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 `g_bTest` ist in App1 deklariert, nicht aber in App2.

```
{IF defined (variable: g_bTest)}
(* the following code is only processed in App2*)
  g_bTest := x > 300;
{END_IF}
```

Operator defined (type: <identifier>)

Der 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)}
(* the following code is only processed in App1*)
  bDutDefined := TRUE;
{END_IF}
```


**Operator defined
(pou: <pou name>)**

Der 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)}
(* the following code is only processed in App1 *)
  arrTest[CheckBounds(0,i,10)] := arrTest[CheckBounds(0,i,10)] + 1;
{ELSE}
(* the following code is only processed in App2 *)
  arrTest[i] := arrTest[i]+1;
{END_IF}
```

**Operator defined
(task: <identifier>)**

Noch nicht implementiert!

Der 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 PlcProg_Task ist in App1 definiert, nicht aber in App2.

```
IF defined (task: PLC_PRG_Task)
(* the following code is only processed in App1 *)
  erg := plc_prg.x;
{ELSE}
(* the following code is only processed in App2 *)
  erg := prog.x;
{END_IF}
```

**Operator defined
(resource: <identifier>)**

Noch nicht implementiert!

Der Operator bewirkt, dass der Ausdruck den Wert **TRUE** erhält, wenn ein Ressourcen-Objekt mit dem Namen <identifier> für die Applikation vorhanden ist; ansonsten wird **FALSE** geliefert.

Beispiel:

Voraussetzung: Es gibt zwei Applikationen App1 und App2. Ein Ressourcen-Objekt glob_var1 der Globalen Variablenliste ist für App1 vorhanden, nicht aber für App2.

```
{IF defined (resource:glob_var1)}
(* the following code is only processed in App1 *)
  gvar_x := gvar_x + ivar;
{ELSE}
(* the following code is only processed in App2 *)
  x := x + ivar;
{END_IF}
```

**Operator defined
(IsSimulationMode)**

Der Operator bewirkt, dass der Ausdruck den Wert **TRUE** erhält, wenn die Applikation auf einem simulierten Gerät läuft, also im Simulationsbetrieb ist.

Siehe auch

- [Kap. 12.1 "Testen im Simulationsbetrieb" auf Seite 193](#)

Operator defined (IsLittleEndian)

Der Operator bewirkt, dass der Ausdruck den Wert **FALSE** erhält, wenn die CPU "BigEndian (Motorola Byte Order)" ist.

**Operator defined
(IsFPUSupported)**

Wenn der Ausdruck den Wert **TRUE** liefert, erzeugt der Codegenerator bei der Berechnungen mit **REAL**-Werten einen FPU-Code (für den Floating Point Unit-Prozessor). Ansonsten emuliert IndraLogic die FPU-Operationen, was jedoch bedeutend langsamer ist.

Operator hasvalue (RegisterSize, '<register size>')	<code><register size></code>: Größe eines CPU-Registers in Bit Der Operator bewirkt, dass der Ausdruck den Wert <code>TRUE</code> liefert, wenn die Größe eines CPU-Registers gleich <code><register size></code> ist. Mögliche Werte für <code><register size></code> • 16 für 186 und C16x, • 64 für X86-64 Bit • 32 für X86-32 Bit
Operator hasvalue (PackMode, '<pack mode value>')	Der abgeprüfte PackMode hängt von der Gerätebeschreibung ab, nicht vom Pragma, das für einzelne DUTs angegeben werden kann.
Operator hasattribute (pou: <pou name>, '<attribute>')	Der Operator bewirkt, dass der Ausdruck den Wert <code>TRUE</code> erhält, wenn das Attribut <code><attribute></code> in der ersten Zeile des Deklarationsteils des Bausteins <code>pou-name</code> angegeben ist; ansonsten wird <code>FALSE</code> geliefert. Beispiel: Voraussetzung: Es gibt zwei Applikationen <code>App1</code> und <code>App2</code>. Ein Funktion <code>fun1</code> ist in <code>App1</code> und <code>App2</code> definiert, aber in <code>App1</code> ist ihr außerdem das Attribut <code>vision</code> zugewiesen. <i>in "App1":</i> <pre>{attribute 'vision'} FUNCTION fun1 : INT VAR_INPUT i : INT; END_VAR VAR END_VAR</pre> <i>In App2:</i> <pre>FUNCTION fun1 : INT VAR_INPUT i : INT; END_VAR VAR END_VAR</pre> Pragma-Anweisung: <pre>{IF hasattribute (pou: fun1, 'vision')} (* the following code is only processed in App1 *) ergvar := fun1(i); {END_IF}</pre> Siehe auch • Kap. "Benutzerdefinierte Attribute" auf Seite 403
Operator hasattribute (variable: <variable>, '<attribute>')	Der Operator bewirkt, dass der Ausdruck den Wert <code>TRUE</code> erhält, wenn der Variablen das Attribut <code>'<attribute>'</code> mit Hilfe der Anweisung <code>{attribute '<attribute>'}</code> in der Zeile vor der Variablendeklaration zugewiesen ist; ansonsten wird <code>FALSE</code> geliefert Beispiel: Voraussetzung: Es gibt zwei Applikationen <code>App1</code> und <code>App2</code>. Variable <code>g_globalInt</code> wird in <code>App1</code> und <code>App2</code> verwendet, aber in <code>App1</code> ist ihr zusätzlich das Attribut <code>'DoCount'</code> zugewiesen. Deklaration <code>g_GlobalInt</code> in <code>App1</code> <pre>VAR_GLOBAL {attribute 'DoCount'} g_globalInt : INT;</pre>

```

        g_multiType : STRING;
END_VAR

```

Deklaration g_GlobalInt in App2

```

VAR_GLOBAL
    g_globalInt : INT;
    g_multiType : STRING;
END_VAR

```

Pragmaanweisung:

```

{IF hasattribute (variable: g_globalInt, 'DoCount')}
    (* the following code is only processed in App1 *)
    g_globalInt := g_globalInt + 1;
{END_IF}

```

Siehe auch

- [Kap. "Benutzerdefinierte Attribute" auf Seite 403](#)

Operator hastype
(variable: <variable>, <type-spec>)

Der Operator bewirkt, dass der Ausdruck den Wert TRUE erhält, wenn die Variable <variable> vom Datentyp <type-spec> ist, ansonsten wird FALSE zurückgeliefert.

Mögliche Datentypen für <type-spec>:

- 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.

Operator hasvalue (<define-ident>, '<char-string>')	<pre>{IF (hastype (variable: g_multitype, LREAL))} (* the following code is only processed in App1 *) g_multitype := (0.9 + g_multitype) * 1.1; {ELSIF (hastype (variable: g_multitype, STRING))} (* the following code is only processed in App2 *) g_multitype := 'this is a multitalent'; {END_IF}</pre> Der 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 zurückgeliefert. 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 den Wert 2.
Operator hasconstantvalue (<variable>, <literal expression>)	<pre>{IF hasvalue(test, '1')} (* the following code is only processed in App1 *) x := x + 1; {ELSIF hasvalue(test, '2')} (* the following code is only processed in App2 *) x := x + 2; {END_IF}</pre> Mit diesem Operator kann man den deklarierten Wert einer Konstanten abfragen. Beispiel: <i>Voraussetzung:</i>
Operator NOT <operator>	<pre>{IF hasconstantvalue(test, '1')} (* the following code is only processed in App1 *) x := x + 1; {ELSIF hasconstantvalue(test, '2')} (* the following code is only processed in App2 *) x := x + 2; {END_IF}</pre> Der Ausdruck erhält den Wert TRUE, wenn der Umkehrwert von <operator> den Wert TRUE liefert. <operator> kann einer der in diesem Kapitel beschriebenen Operatoren sein. Beispiel: Voraussetzung: Es gibt zwei Applikationen App1 und App2. PlcProg1 ist in App1 und App2 vorhanden, POU CheckBounds gibt es nur in App1.
Operator <operator> AND <operator>	<pre>{IF defined (pou: PLC_PRG1) AND NOT (defined (pou: CheckBounds))} (* the following code is only processed in App2 *) bANDNotTest := TRUE; {END_IF}</pre> Der Ausdruck erhält den Wert TRUE, wenn die beiden angegebenen Operatoren TRUE liefern. <operator> kann einer der in diesem Kapitel beschriebenen Operatoren sein. Beispiel: Voraussetzung: Es gibt zwei Applikationen App1 und App2. PlcProg1 ist in App1 und App2 vorhanden, die POU CheckBounds nur in App1.
	<pre>{IF defined (pou: PLC_PRG1) AND (defined (pou: CheckBounds))}</pre>

```
(* the following code is only processed in App1 *)
bANDTest := TRUE;
{END_IF}
```

Operator <operator> OR <operator>

Der Ausdruck liefert TRUE, wenn einer der beiden angegebenen Operatoren TRUE liefert. <operator> kann einer der hier beschriebenen Operatoren sein.

Beispiel:

Voraussetzung: Es gibt zwei Applikationen App1 und App2.

PlcProg1 ist in App1 und App2 vorhanden, POU CheckBounds nur in App1.

```
{IF defined (pou: PLC_PRG1) OR (defined (pou: CheckBounds))}
(* the following code is only processed in App1 and in App2 *)
bORTest := TRUE;
{END_IF}
```

Operator (<operator>)

() klammert die Operatoren.

Siehe auch

- [Kap. 9.6 "Pragmas verwenden" auf Seite 114](#)
- [Kap. "Benutzerdefinierte Attribute" auf Seite 403](#)

17.6.4 Region-Pragma

Das Pragma dient dazu, in einem Texteditor mehrere Zeilen zu einem Block zusammenzufassen. Der Block kann mit einem Namen versehen werden. Region-Pragmas können auch ineinander verschachtelt sein.

Code mit Region-Pragma: Erweiterte und reduzierte Ansicht

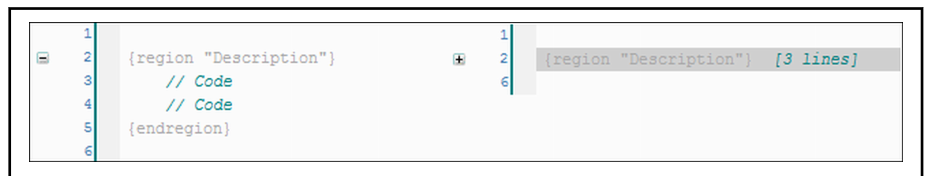


Abb. 17-91:

Das Pragma kann im ST-Editor und allen Deklarationseditoren verwendet werden. Die Syntaxhervorhebung kann in den Optionen angepasst werden.

Siehe auch:

[Kap. "Befehl 'Alles erweitern'" auf Seite 622](#)

[Kap. "Befehl 'Alles reduzieren'" auf Seite 621](#)

17.7 Bezeichner

Regeln für die Bezeichnervergabe

Regeln für den Bezeichner einer Variablen

- Ein Bezeichner darf keine Leerzeichen oder Sonderzeichen enthalten
- IndraLogic beachtet die Groß-/Kleinschreibung nicht, das heißt: VAR1 und var1 bezeichnen dieselbe Variable
- IndraLogic erkennt den Unterstrich, dies bedeutet, dass zum Beispiel A_BCD und AB_CD als zwei verschiedene Bezeichner behandelt werden.
Mehrfach aufeinander folgende Unterstriche sind nicht erlaubt.
- Die Länge eines Bezeichners ist unbegrenzt

Regeln für die Mehrfachverwendung von Bezeichnern (Namensräume)

- Ein Bezeichner darf lokal nicht doppelt deklariert werden
- Ein Bezeichner darf nicht identisch mit einem Schlüsselwort (zum Beispiel Variablentyp) sein.
- Sie dürfen einen Bezeichner global mehrfach verwenden. Wenn eine lokale Variable denselben Namen hat wie eine globale Variable, hat die lokale Variable innerhalb einer POU den Vorrang.
- Eine Variable, die in einer globalen Variablenliste definiert ist, kann denselben Namen haben wie eine Variable, die in einer anderen GVL definiert ist.

IndraLogic bietet folgende norm-erweiternden Funktionalitäten bezüglich des Namensraums/Gültigkeitsbereichs von Variablen an:

- Globaler Namensraum-Operator: Ein Instanzpfad, der mit einem Punkt beginnt, öffnet immer einen globalen Namensraum. Wenn es eine lokale Variable, zum Beispiel `ivar`, gibt, die denselben Namen hat wie eine globale Variable, sprechen Sie mit `.ivar` die globale Variable an.
- Der Name einer globalen Variablenliste kann den Namensraum für die enthaltenen Variablen eindeutig definieren. Somit können Sie Variablen mit demselben Namen in verschiedenen globalen Variablenlisten deklarieren und dennoch durch das Voranstellen des Listennamens eindeutig ansprechen.

Beispiel: `globlist1.ivar := globlist2.ivar; (* ivar aus GVL globlist2 wird auf ivar in GVL globlist1 kopiert *)`

- Variablen, die in der globalen Variablenliste einer ins Projekt eingebundenen Bibliothek definiert sind, können Sie gemäß der Syntax `<Namensraum Bibliothek>.<Name der GVL>.<Variablenname>` eindeutig ansprechen.

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 können Sie einen Bibliotheksbaustein oder eine Bibliotheksvariable über `<Namensraum Bibliothek>.<Bausteinname|Variablenname>` eindeutig ansprechen.

Beachten Sie im Fall von verschachtelten Bibliotheken, dass die Namensräume aller beteiligten Bibliotheken in Folge anzugeben sind.

Beispiel: Wenn `Lib1` durch `Lib0` referenziert wird, so wird der in `Lib1` enthaltene Baustein `func` über `Lib0.Lib1.fun` angesprochen:

`ivar := Lib0.Lib1.fun(4, 5); (* Rückgabewert von func wird auf Variable ivar im Projekt kopiert *)`

Siehe auch

- [Kap. 17.5 "Datentypen" auf Seite 374](#)
- [Kap. 17.2 "Variablen" auf Seite 300](#)

Empfehlungen

Zusätzlich zu den Punkten, die Sie speziell bei der Variablendeklaration beachten müssen, wird empfohlen die folgenden Regeln anzuwenden. Damit

Empfehlungen für Variablennamen

erreichen Sie eine möglichst weitgehende Vereinheitlichung bei der Namensvergabe.

Die Benennung von Variablen soll in allen Applikationen und Bibliotheken möglichst angelehnt an die ungarische Notation erfolgen:

Für jede Variable sollten Sie eine sinnvolle, möglichst kurze, englische Beschreibung finden, den Basisnamen. Den jeweils erste Buchstaben eines Worts des Basisnamens sollen Sie groß, die übrigen klein schreiben (zum Beispiel: `FileSize`).

Entsprechend dem Datentyp der Variablen hängen Sie vor diesen Basisnamen Präfix(e) in Kleinbuchstaben:

Datentyp	Präfix	Kommentar
BOOL	x*	Bewusst wird als Präfix für boolesche Variablen x empfohlen, um sich von Bezeichnern des Datentyps BYTE abzugrenzen. Das Präfix verdeutlicht die Sicht eines IEC-Programmierers.
	b	reserviert
BYTE	by	Bitstring, nicht für arithmetische Operationen
WORD	w	Bitstring, nicht für arithmetische Operationen
DWORD	dw	Bitstring, nicht für arithmetische Operationen
LWORD	lw	Bitstring, nicht für arithmetische Operationen
SINT	si	
USINT	usi	
INT	i	
UINT	ui	
DINT	di	
UDINT	udi	
LINT	li	
ULINT	uli	
REAL	r	
LREAL	lr	
STRING	s	
WSTRING	ws	
TIME	tim	
LTIME	ltim	
TIME_OF_DAY	tod	
DATE_AND_TIME	dt	
DATE	date	

Datentyp	Präfix	Kommentar
ENUM	e	
POINTER	p	
ARRAY	a	
Enumeration	e	

*: bewusst wird für boolesche Variablen **x** als Präfix gewählt, um zum einen eine Abgrenzung zu **BYTE** zu finden, und zum anderen um der Sicht des IEC-Programmierers entgegenzukommen.

Beispiele

```
VAR
    bySubIndix: BYTE;
    xFlag: BOOL;
    udiCounter: UDINT;
END_VAR
```

Bezeichner für	Beschreibung	Beispiel
Verschachtelte Deklarationen	Die Präfixe werden in der Reihenfolge der Deklaration aneinander gehängt:	pabyTelegramData: POINTER TO ARRAY [0..7] OF BYTE;
Funktionsbaustein-Instanz Variable vom benutzerdefinierten Datentyp	Präfix: Kürzel für den Funktionsbaustein- oder Datentypnamen	<pre>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</pre>
Lokale Konstante Lokale konstante Variable	Präfix: c_ , gefolgt vom Typ-Präfix und dem Variablennamen	<pre>VAR CONSTANT c_uiSyncID: UINT := 16#80; END_VAR</pre>
Globale Variable	An das Bibliothekspräfix wird ein zusätzliches Präfix angehängt: g_	<pre>VAR_GLOBAL CAN_g_iTest: INT; END_VAR</pre>
Globale Konstante Globale konstante Variable	An das Bibliothekspräfix wird ein zusätzliches Präfix angehängt: gc_	<pre>VAR_GLOBAL CONSTANT CAN_gc_dwExample: DWORD; END_VAR</pre>

Empfehlungen für Variablennamen in IndraLogic 2G-Bibliotheken

Bezeichner für	Beschreibung	Beispiel
Variablennamen	Entspricht der obigen Beschreibung für Variablennamen mit der Ausnahme, dass globale Variablen und Konstanten kein Bibliothekspräfix benötigen, da der Namensraum die Funktion ersetzt.	<pre>g_iTest: INT; // Declaration CAN.g_iTest; // Implementation; call in the program</pre>

**Empfehlungen für Bezeichner für
benutzerdefinierte Datentypen
(DUT)**

Bezeichner für	Beschreibung	Beispiel
Strukturen	Bibliothekspräfix, gefolgt von einem Unterstrich und einer kurzen, aussagekräftigen Beschreibung der Struktur. Das zugehörige Präfix für angelegte Variablen dieser Struktur soll als Kommentar direkt nach dem Doppelpunkt folgen.	<pre>TYPE CAN_SDOTelegram : (* prefix: sdo *) STRUCT wIndex: WORD; bySubIndex: BYTE; byLen: BYTE; abyData: ARRAY [0..3] OF BYTE; END_STRUCT END_TYPE</pre>
Enumerationen	Bibliothekspräfix, gefolgt von einem Unterstrich und dem Bezeichner in Großbuchstaben. Zu beachten: In vergangenen IndraLogic-Versionen haben ENUM-Werte > 16#7FFF zu Fehlern geführt, da sie nicht automatisch in INT konvertiert wurden. Aus diesem Grund sollen ENUMs stets mit korrekten INT-Werten definiert werden.	<pre>TYPE CAL_Day : (CAL_MONDAY, CAL_TUESDAY, CAL_WEDNESDAY, CAL_THURSDAY, CAL_FRIDAY, CAL_SATURDAY, CAL_SUNDAY); Deklaration: eToday: CAL_Day;</pre>

**Empfehlungen für Bezeichner für
benutzerdefinierte Datentypen
(DUT) in IndraLogic 2G-Bibliotheken**

Referenz Programmierung

Bezeichner für	Beschreibung	Beispiel
DUT-Namen in IndraLogic 2G-Bibliotheken	Es wird auf das Bibliothekspräfix verzichtet, da die Funktion des Bibliothekspräfixes durch den Namensraum ersetzt wird. Auch die Enumerationswerte werden ohne Bibliothekspräfix definiert.	Bibliothek mit Namensraum "CAL" <pre>TYPE Day : (MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY) ;</pre> Deklaration: <pre>eToday: CAL.Day;</pre> Verwendung in der Applikation <pre>IF eToday = CAL.Day.MONDAY THEN</pre>

**Empfehlungen für Bezeichner für
POUs: Funktionen, Funktionsbausteine, Programme**

Bezeichner für	Beschreibung	Beispiel
POUs: Funktionen, Funktionsbausteine, Programme	Bibliothekspräfix gefolgt von einem Unterstrich und einem aussagekräftigen, möglichst kurzen Namen der POU. Wie bei den Variablen soll der jeweils erste Buchstabe eines Worts des Basisnamens groß, die übrigen klein geschrieben werden. Es wird empfohlen, den Namen der POU aus einem Verb und einem Substantiv zusammenzusetzen. Bei Funktionsbausteinen soll das zugehörige Präfix für angelegte Instanzen als Kommentar direkt nach dem Namen folgen.	<pre>FUNCTION_BLOCK CAN_SendTelegram (* prefix: canst *)</pre>
Aktionen	Nur Aktionen, die der Baustein selbst aufruft, beginnen mit <code>prv_</code>. Ansonsten erhalten Aktionen kein Präfix.	

**Empfehlungen für Bezeichner für
POUs in IndraLogic 2G-Bibliotheken**

Bezeichner für	Beschreibung	Beispiel
POUs:	Es wird auf das Bibliothekspräfix verzichtet, da der Namensraum die Funktion das Bibliothekspräfix ersetzt.	FUNCTION_BLOCK CAN_SendTelegram (* prefix: canst *)
Methoden	Nur Methoden, die der Baustein selbst aufruft, beginnen mit <code>prv_</code> . Ansonsten erhalten Methoden kein Präfix.	
Interface Schnittstelle	Beginnend mit "I"	ICANDevice

Empfehlungen für Bezeichner für Visualisierungen



Achten Sie darauf, dass eine Visualisierung nicht denselben Namen erhält wie ein anderer Baustein im Projekt, da dies bei Visualisierungswechseln zu Problemen führen würde.

Siehe auch

- [Kap. 9.2 "Variablen deklarieren " auf Seite 85](#)
- [Kap. 17.5 "Datentypen" auf Seite 374](#)
- [Kap. 17.2 "Variablen" auf Seite 300](#)

17.8 Verschattungsregeln

In IndraLogic ist es prinzipiell erlaubt, den gleichen Bezeichner für verschiedene Elemente zu verwenden. So können beispielsweise ein Baustein und eine Variable gleich benannt werden. Um Verwechslungen vorzubeugen, sollte dies jedoch vermieden werden.

Negativbeispiel

Im folgenden Codeausschnitt hat eine lokale Funktionsbaustein-Instanz den gleichen Namen wie eine Funktion:

Programm:

```
FUNCTION YYYY : INT
;
END_FUNCTION

FUNCTION_BLOCK XXX
;
END_FUNCTION_BLOCK

PROGRAM PLC_PRG
VAR
    YYYY : XXX;
END_VAR
YYYY();
END_PROGRAM
```

In einem solchen Fall ist unklar, ob im Programm die Instanz oder die Funktion aufgerufen wird.

Um sicherzustellen, dass Namen immer eindeutig sind, sollten Namenskonventionen beachtet werden, beispielsweise bestimmte Präfixe für Variablen. Regeln für die Bezeichnervergabe finden Sie im Hilfef Kapitel "Bezeichner".

Namenskonventionen können mit Hilfe der statischen Codeanalyse von IndraLogic automatisch überprüft werden. Die statische Codeanalyse könnte auch die doppelte Verwendung des Namens YYYY aufdecken und als Fehler melden.

Auch durch die konsequente Verwendung des Attributs `qualified_only` für Enumerationen und globale Variablenlisten und durch die Verwendung von qualifizierten Bibliotheken können nicht-eindeutige Situationen vermieden werden.

Verschattung

Der Compiler meldet grundsätzlich weder Fehler noch Warnungen, wenn derselbe Bezeichner für verschiedene Elemente verwendet wird. Stattdessen durchsucht der Compiler den Code in einer bestimmten Reihenfolge nach der Deklaration des Bezeichners. Wenn eine Deklaration gefunden wurde, dann sucht der Compiler nicht nach eventuellen weiteren Deklarationen an anderer Stelle. Wenn weitere Deklarationen existieren, dann sind diese für den Compiler "verschattet". Im Folgenden werden die Verschattungsregeln beschrieben, das heißt die Suchreihenfolgen, die der Compiler bei der Suche nach der Deklaration für Bezeichner verwendet. Im Abschnitt "Uneindeutige Zugriffe und qualifizierte Zugriffe" werden Möglichkeiten aufgezeigt, um uneindeutige Zugriffe zu vermeiden und die Verschattungsregeln zu umgehen.

Suchreihenfolge in der Applikation

Wenn der Compiler im Code einer Applikation auf einen einzelnen Bezeichner trifft, dann sucht er die zugehörige Deklaration in der folgenden Reihenfolge:

1. Lokale Variablen einer Methode
2. Lokale Variablen im Funktionsbaustein, Programm oder Funktion und in eventuellen Basisfunktionsbausteinen
3. Lokale Methoden des Bausteins
4. Globale Variablen in der Applikation, wenn in der Variablenliste, in der die globalen Variablen deklariert sind, nicht das Attribut `qualified_only` gesetzt ist
5. Globale Variablen in einer Vaterapplikation, sofern in der Variablenliste, in der die globalen Variablen deklariert sind, nicht das Attribut `qualified_only` gesetzt ist
6. Globale Variablen in angezogenen Bibliotheken, wenn weder die Bibliothek noch die Variablenliste qualifizierten Zugriff erfordert
7. Baustein- oder Typnamen aus der Applikation (das heißt: Namen von Globalen Variablenlisten, Funktionsbausteinen etc.)
8. Baustein- oder Typnamen aus einer Vaterapplikation
9. Baustein- oder Typnamen aus einer Bibliothek
10. Namensräume von lokal angezogenen Bibliotheken und Bibliotheken, die von Bibliotheken veröffentlicht werden
11. Globale Variablen in der Ansicht `POUs`, sofern in der Variablenliste, in der sie deklariert sind, nicht das Attribut `qualified_only` gesetzt ist
12. Baustein- oder Typnamen aus der Ansicht `POUs` (das heißt: Namen von Globalen Variablenlisten, Funktionsbausteinen etc.)



Bibliotheken, die im Bibliotheksverwalter der Ansicht `POUs` eingefügt werden, werden mit der entsprechenden Platzhalterauflösung in den Bibliotheksverwalter in allen Applikationen im Projekt gespiegelt. Diese Bibliotheken bilden dann mit den Bibliotheken in der Applikation einen gemeinsamen Namensraum. Daher gibt es keine Verschattung von Bibliotheken im Pool durch Bibliotheken in der Applikation.

Suchreihenfolge in der Bibliothek

Wenn der Compiler im Code einer Bibliothek auf einen einzelnen Bezeichner trifft, dann sucht er die zugehörige Deklaration in der folgenden Reihenfolge:

1. Lokale Variablen einer Methode

2. Lokale Variablen im Funktionsbaustein, Programm oder Funktion und in eventuellen Basis-Funktionsbausteinen
3. Lokale Methoden des Bausteins
4. Globale Variablen in der lokalen Bibliothek, wenn in der Variablenliste, in der die globalen Variablen deklariert sind, nicht das Attribut `qualified_only` gesetzt ist
5. Globale Variablen in angezogenen Bibliotheken, wenn weder die Bibliothek noch die Variablenliste qualifizierten Zugriff erfordert
6. Baustein- oder Typnamen aus der lokalen Bibliothek (das heißt: Namen von globalen Variablenlisten, Funktionsbausteinen etc.)
7. Baustein- oder Typnamen aus einer angezogenen Bibliothek
8. Namensräume von lokal angezogenen Bibliotheken und Bibliotheken, die von lokal angezogenen Bibliotheken veröffentlicht werden

Uneindeutige Zugriffe und qualifizierte Zugriffe

Trotz der oben beschriebenen Suchreihenfolgen kann es zu uneindeutigen Zugriffen kommen. Das ist zum Beispiel der Fall, wenn eine Variable mit dem gleichen Namen in zwei globalen Variablenlisten vorkommt, die nicht qualifizierten Zugriff erfordern. Einen solchen Fall meldet der Compiler als Fehler (zum Beispiel: nicht eindeutige Verwendung des Namens `xxx`).

Eine solche uneindeutige Verwendung lässt sich durch einen qualifizierten Zugriff, also beispielsweise durch den Zugriff über den Namen der globalen Variablenliste, eindeutig machen (zum Beispiel: `GVL.xxx`).

Ein qualifizierter Zugriff lässt sich auch immer dazu nutzen, um Verschattungsregeln zu umgehen.

Beispiele:

- Mit dem Namen der globalen Variablenliste kann eindeutig auf eine Variable in dieser Liste zugegriffen werden
- Mit dem Namen einer Bibliothek kann eindeutig auf Elemente in dieser Bibliothek zugegriffen werden
- Mit dem Pointer `THIS` kann eindeutig auf Variablen in einem Funktionsbaustein zugegriffen werden, auch wenn eine lokale Variable mit gleichem Namen in einer Methode des Funktionsbausteins existiert

Um jederzeit die Deklarationsstelle eines Bezeichners zu finden, wählen Sie den Befehl **Bearbeiten** ► **Symbol suchen** ► **Gehe zur Definition**. Diese Vorgehensweise kann insbesondere dann hilfreich sein, wenn der Compiler eine scheinbar unverständliche Fehlermeldung produziert.

Suchen in Instanzpfaden

Die oben beschriebenen Suchreihenfolgen gelten nicht für Bezeichner, die in einem Instanzpfad als Komponente auftauchen, oder für Bezeichner, die als Eingänge in Aufrufen verwendet werden.

Für einen Zugriff der folgenden Art `yy.component` hängt es von der Entität ab, die durch `yy` beschrieben wird, wo nach der Deklaration von `component` gesucht wird.

Wenn `yy` eine Variable mit strukturiertem Datentyp bezeichnet (also vom Typ `STRUCT` oder `UNION`), dann wird `component` in dieser Reihenfolge gesucht:

- Lokale Variablen des Funktionsbausteins
- Lokale Variablen des Basis-Funktionsbausteins
- Methoden des Funktionsbausteins
- Methoden des Basis-Funktionsbausteins

Wenn `yy` eine globale Variablenliste bezeichnet oder ein Programm, dann wird `component` nur in dieser Liste gesucht.

Wenn `yy` einen Namensraum einer Bibliothek bezeichnet, dann wird `component` in dieser Bibliothek genau so gesucht, wie es im obigen Abschnitt "Suchreihenfolge in der Bibliothek" beschrieben ist.

Erst in zweiter Instanz entscheidet der Compiler, ob der Zugriff auf das gefundene Element erlaubt ist, das heißt, ob die Variable möglicherweise nur lokal zugreifbar ist, oder ob eine Methode privat ist. Wenn der Zugriff nicht erlaubt ist, dann wird ein Fehler ausgegeben.

Siehe auch:

- [Kap. 17.7 "Bezeichner" auf Seite 445](#)
- [Kap. "Attribut 'qualified_only'" auf Seite 434](#)
- [Kap. 17.2.15 "THIS" auf Seite 313](#)

17.9 Schlüsselwörter

Sie müssen Schlüsselwörter, die beispielweise Gültigkeitsbereiche, Datentypen oder Operatoren bezeichnen, in allen Editoren in Großbuchstaben schreiben.

Schlüsselwörter können nicht als Variablennamen verwendet werden.

Beispiele

Programm:

```
VAR
END_VAR
BOOL_TO_INT
IF
THEN
ELSE
LTIME
MUL
XOR
PERSISTENT
PROGRAM
```

IndraLogic überprüft die korrekte Verwendung von Schlüsselwörtern automatisch und unterkringt Fehler unmittelbar bei der Eingabe.



Wenn IndraLogic impliziten Code anlegt, dann werden Variablen und Funktionen im Regelfall mit einem Namen versehen der `"__"` enthält, meistens damit beginnt.

Die Verwendung von doppelten Unterstrichen in Implementierungen wird automatisch verhindert.

Somit kann es nicht zu Konflikten zwischen systeminternen und den vom Programmierer vergebenen Bezeichnern kommen.

Folgende Schlüsselwörter werden im IndraLogic-Export-Format verwendet. Sie dürfen sie deshalb nicht als Bezeichner verwenden:

- ACTION
- END_ACTION
- END_FUNCTION
- END_FUNCTION_BLOCK
- END_PROGRAM

Ebenfalls als Schlüsselwörter gelten:

- VAR_ACCESS
- READ_ONLY

- READ_WRITE
- PARAMS

17.10 Methode 'FB_Init', 'FB_Reinit' und 'FB_Exit'

Die Methoden können Sie explizit einsetzen, um Einfluss auf die Initialisierung von Funktionsbaustein-Variablen und auf das Verhalten beim Beenden von Funktionsbausteinen zu nehmen.



Der Typ des Rückgabewertes für die impliziten Methoden ist `BOOL`. Der Wert wird vom System nicht ausgewertet, der Typ sollte jedoch nicht geändert werden.

`FB_Init` ist immer implizit verfügbar und wird grundsätzlich bei der Initialisierung aufgerufen. Für eine gezielte Einflussnahme können Sie die Methode außerdem explizit deklarieren und dort zusätzlichen Code zum Standardinitialisierungscode ergänzen.

`FB_Reinit` muss explizit implementiert werden. Wenn diese Methode vorliegt, wird sie aufgerufen, nachdem die Instanz des betreffenden Funktionsbausteins kopiert wurde. Das geschieht während eines Online-Changes nach Änderungen an der Funktionsbausteindeklaration (Signaturänderung), um das neue Instanzmodul zu reinitialisieren. Um eine Reinitialisierung der Basisimplementierung des Funktionsbausteins zu erreichen, müssen Sie `FB_Reinit` noch explizit aufrufen.

`FB_Exit` müssen Sie explizit implementieren. Wenn eine Implementierung vorliegt, wird die Methode aufgerufen, bevor der Code der Funktionsbaustein-Instanz von der Steuerung entfernt wird (impliziter Aufruf).

Im Folgenden sehen Sie für verschiedenen Betriebsfälle einige Anwendungsfälle dieser Methoden.

Betriebsfall "Erster Download"

Beim Download einer Applikation auf eine Steuerung, die sich im Auslieferungszustand befindet, müssen die Speicherplätze aller Variablen in den gewünschten Ausgangszustand versetzt werden. Dadurch werden die Datenbereiche von Funktionsbaustein-Instanzen mit den gewünschten Werten belegt. Durch die explizite Implementierung von `FB_Init` für Funktionsbausteine können Sie im Code der Applikation in dieser Situation gezielt reagieren. Durch die Auswertung der Methodenparameter `bInCopyCode` (`FALSE`) und `bInitRetains` (`TRUE`) können Sie diesen Betriebsfall eindeutig detektieren. (Siehe "Betriebsfall "Online-Change"" und "Betriebsfall "Erneuter Download".")

Betriebsfall "Online-Change"

Im Rahmen des Online-Changes können Sie über die Methoden `FB_Exit`, `FB_Init` und `FB_Reinit` die Initialisierung von Funktionsbaustein-Instanzen beeinflussen. Im Rahmen des Online-Changes werden die im Offlinebetrieb vollzogenen Änderungen an der Applikation in der laufenden Steuerung nachvollzogen. Deshalb werden die „alten“ Instanzen der Funktionsbausteine durch Ihre „neuen Geschwister“ möglichst stoßfrei ersetzt. Wenn vor dem Login in der Applikation keine Änderungen am Deklarationsteil eines Funktionsbausteins vorgenommen wurde, sondern nur in der Implementierung, dann findet keine Ersetzung der Datenbereiche statt. Es werden nur Codeblöcke ersetzt. Dann werden die Methoden `FB_Exit`, `FB_Init` und `FB_Reinit` nicht aufgerufen!



Wenn Sie Änderungen in der Deklaration eines Funktionsbausteins vorgenommen haben, die zum oben beschriebenen Kopiervorgang führen werden, erhalten Sie beim Online-Change eine Meldung zu den "möglicherweise unbeabsichtigten Auswirkungen". In den Details der Meldungsbox sehen Sie eine Liste aller zu kopierenden Instanzen aufgelistet.

Im Code der Methode `FB_Init` kann durch Auswertung der Parameter `bInCopyCode` (`TRUE`) ermittelt werden, ob gerade ein Online-Change ausgeführt wird.

Beim Online-Change erfolgen nacheinander folgende Aufrufe:

1. `FB_Exit`

```
old_inst.fb_Exit(bInCopyCode := TRUE);
```

Sie können den Aufruf von `FB_Exit` beim Verlassen der "alten" Instanz verwenden, um vor dem Kopiervorgang bestimmte Aufräumarbeiten anzustoßen. So können Sie die Daten für den folgenden Kopiervorgang vorbereiten und den Zustand der "neuen" Instanz beeinflussen. Andere Teile der Applikation können Sie über die bevorstehende Lageänderung im Speicher informieren. Achten Sie besonders auf Variablen vom Typ `POINTER` oder `REFERENCE`. Diese verweisen nach dem Online-Change eventuell nicht mehr auf die gewünschten Speicherstellen. Schnittstellenvariablen (`INTERFACE`) werden vom Compiler gesondert behandelt und beim Online-Change entsprechend angepasst. Externe Ressourcen wie beispielsweise Sockets, Files oder andere Handles können eventuell unverändert von der neuen Instanz übernommen werden. Sie müssen oft während des Online-Changes nicht gesondert behandelt werden. (Siehe "Betriebsfall "Erneuter Download"")

2. `FB_Init`

```
new_inst.FB_Init(bInitRetains := FALSE, bInCopyCode := TRUE);
```

Der Aufruf von `FB_Init` erfolgt vor dem Kopiervorgang und kann verwendet werden, um für den Online-Change spezifische Operationen auszuführen. Beispielsweise können Sie darüber Variablen an der "neuen" Position im Speicher entsprechend initialisieren lassen, oder andere Teile der Applikation über die neue Position bestimmter Variablen im Speicher informieren.

3. Kopiervorgang `copy`

```
copy(&old_inst, &new_inst);
```

Bestehende Werte bleiben erhalten. Zu diesem Zweck werden sie aus der alten Instanz in die neue kopiert.

4. `FB_Reinit`

```
new_inst.FB_Reinit();
```

Diese Methode wird nach dem Kopiervorgang aufgerufen und setzt die Variablen der Instanz auf definierte Werte. Beispielsweise können Sie darüber Variablen an der "neuen" Position im Speicher entsprechend initialisieren, oder andere Teile der Applikation über die neue Position bestimmter Variablen im Speicher informieren. Gestalten Sie die Implementierung unabhängig vom Online-Change. Die Methode kann nämlich auch aus der Applikation heraus jederzeit aufgerufen werden, um eine Funktionsbaustein-Instanz in den ursprünglichen Zustand zurückzusetzen.



Mit dem Attribut `{attribute 'no_copy'}` können Sie für eine einzelne Variable des Funktionsbausteins verhindern, dass diese beim Online Change kopiert wird. Sie behält dann immer den Initialwert.

Betriebsfall "Erneuter Download"

Beim Download einer Applikation wird eventuell eine bereits vorhandene Applikation auf der Steuerung ersetzt. Deshalb muss der Speicherplatz für die vorhandenen Funktionsbausteine geregelt freigegeben werden. Sie können die Methode `FB_Exit` verwenden, um die dazu notwendigen Schritte zu implementieren. Beispielsweise können Sie externe Ressourcen (mit Socket- oder File-Handles) in einen definierten Zustand versetzen.

Sie können diesen Betriebsfall detektieren, indem Sie überprüfen, ob bei der Methode `FB_Exit` der Parameter `bInCopyCode = FALSE` ist.

Betriebsfall "Start der Applikation"

Vor dem ersten Zyklus der Tasks einer Applikation werden die initialen Zuweisungen verarbeitet.

Beispiel

```
T1 : TON := (PT:=t#500ms);
```

Solche Zuweisungen werden erst nach dem Aufruf von `FB_Init` ausgeführt. Um die Auswirkungen dieser Zuweisungen kontrollieren zu können, dürfen Sie einen Funktionsbaustein oder eine Methode eines Funktionsbausteins mit dem Attribut `{attribute 'call_after_init'}` versehen. Sie müssen das Attribut sowohl über dem Deklarationsteil des Funktionsbausteinrumpfes als auch über dem Deklarationsteil der entsprechenden Methode einfügen. Ein Baustein, der einen anderen Baustein erweitert, der das Attribut `{attribute 'call_after_init'}` verwendet, muss ebenfalls mit dem Attribut versehen werden. Aus Gründen der Verständlichkeit wird empfohlen, die entsprechende Methode mit dem gleichen Namen, der gleichen Signatur und dem gleichen Attribut zu überschreiben. Das erfordert einen Aufruf von `SUPER^.MyInit`. Der Name der Methode ist frei wählbar. (Ausnahmen: `FB_Init`, `FB_Reinit` und `FB_Exit`). Die Methode wird nach der Verarbeitung der initialen Zuweisungen und vor dem Start der Tasks einer Applikation aufgerufen und kann so auf die Vorgaben des Anwenders entsprechend reagieren.

Bedenken Sie bei der Verwendung von `FB_Init`, oder `{attribute 'call_after_init'}` jedoch auch Folgendes: Das Finden von Fehlern in den Methoden `FB_Init`, oder in Methoden die mit `{attribute 'call_after_init'}` dekoriert sind, ist mühsam, weil unter Anderem das Setzen von Haltepunkten nicht die gewünschte Wirkung haben kann.



Wenn während der Ausführung der explizit definierte Initialisierungscode erreicht wird, ist die Funktionsbaustein-Instanz bereits über den impliziten Initialisierungscode vollständig initialisiert. Es darf deshalb kein `SUPER^.FB_Init`- Aufruf erfolgen



`FB_Init` ersetzt den in IndraLogic 1.x verwendeten `INI`-Operator. Die Methode ist nicht zu vergleichen mit dem Konstrukt eines Konstruktors, wie man ihn aus C#, C++ oder auch Java kennt. Das hat Konsequenzen für Funktionsbausteine, die andere Funktionsbausteine erweitern. (Siehe unten "Abgeleitete Funktionsbausteine")

Schnittstelle der Methode FB_Init

```

METHOD FB_Init : BOOL
VAR_INPUT
    bInitRetains : BOOL; // TRUE: the Retainvariables are initialized
    (reset warm / reset cold)
    bInCopyCode : BOOL; // TRUE: the instance will be copied to the
    copy-code afterward(online change)
END_VAR

```

Sie können in einer FB_Init-Methode zusätzliche Funktionsbaustein- Eingänge deklarieren. Sie müssen dann diese Eingänge in der Deklaration der Funktionsbaustein-Instanz setzen.

Beispiel

Methode FB_Init für einen Funktionsbaustein serialdevice

```

METHOD PUBLIC FB_Init : BOOL
VAR_INPUT
    bInitRetains : BOOL; // initializing of retain variable
    bInCopyCode : BOOL; // instance is copied to copy code
    iCOMnum : INT; // additional input: number of the COM interface,
    that is to be observed
END_VAR

```

Instanzierung des Funktionsbausteins serialdevice:

```

com1 : serialdevice(iCOMnum:=1);
com0 : serialdevice(iCOMnum:=0);

```

**Schnittstelle der Methode
FB_Reinit**

```

METHOD FB_Reinit : BOOL

```

Schnittstelle der Methode FB_Exit

Es gibt es den obligatorischen Parameter bInCopyCode.

```

METHOD FB_Exit : BOOL
VAR_INPUT
    bInCopyCode : BOOL; // TRUE: the exit method is called in order to leave the
    instance which will be copied afterwards (online change).
END_VAR

```

**Verhalten bei abgeleiteten
Funktionsbausteinen**

Wenn ein Funktionsbaustein von einem anderen Funktionsbaustein abgeleitet ist, muss die FB_Init-Methode des abgeleiteten Funktionsbausteins dieselben Parameter definieren wie die FB_Init-Methode des Basisfunktionsbausteins. Sie können jedoch weitere Parameter hinzufügen, um für die Instanz eine spezielle Initialisierung einzurichten.

Beispiel

Die Funktionsbausteine MainFB, SubFB und SubSubFB leiten voneinander ab. Dabei gilt: SubFB EXTENDS MainFB und SubSubFB EXTENDS SubFB.

Aufrufabfolge der Methoden FB_Exit und FB_Init:

1. fbSubSubFb.FB_Exit(...);
2. fbSubFb.FB_Exit(...);
3. fbMainFb.FB_Exit(...);
4. fbMainFb.FB_Init(...);
5. fbSubFb.FB_Init(...);
6. fbSubSubFb.FB_Init(...);

Siehe auch

- [Kap. "Objekt 'Methode' " auf Seite 564](#)
- [Kap. "Attribut 'call_after_init'" auf Seite 404](#)
- [Kap. "Attribut 'no_copy'" auf Seite 425](#)
- [Kap. 17.2.14 "SUPER" auf Seite 312](#)

17.11 Compiler Fehlermeldungen

Compilerfehler C0001	Compilerfehler C0002	Compilerfehler C0003
Compilerfehler C0004	Compilerfehler C0005	Compilerfehler C0006
Compilerfehler C0007	Compilerfehler C0008	Compilerfehler C0009
Compilerfehler C0010	Compilerfehler C0011	Compilerfehler C0013
Compilerfehler C0015	Compilerfehler C0018	Compilerfehler C0020
Compilerfehler C0022	Compilerfehler C0023	Compilerfehler C0026
Compilerfehler C0027	Compilerfehler C0030	Compilerfehler C0031
Compilerfehler C0032	Compilerfehler C0035	Compilerfehler C0037
Compilerfehler C0038	Compilerfehler C0039	Compilerfehler C0040
Compilerfehler C0041	Compilerfehler C0043	Compilerfehler C0044
Compilerfehler C0045	Compilerfehler C0046	Compilerfehler C0047
Compilerfehler C0048	Compilerfehler C0049	Compilerfehler C0050
Compilerfehler C0061	Compilerfehler C0062	Compilerfehler C0064
Compilerfehler C0066	Compilerfehler C0068	Compilerfehler C0069
Compilerfehler C0070	Compilerfehler C0072	Compilerfehler C0074
Compilerfehler C0075	Compilerfehler C0076	Compilerfehler C0077
Compilerfehler C0080	Compilerfehler C0081	Compilerfehler C0082
Compilerfehler C0084	Compilerfehler C0086	Compilerfehler C0087
Compilerfehler C0089	Compilerfehler C0090	Compilerfehler C0091
Compilerfehler C0094	Compilerfehler C0096	Compilerfehler C0097
Compilerfehler C0098	Compilerfehler C0101	Compilerfehler C0102
Compilerfehler C0104	Compilerfehler C0114	Compilerfehler C0116
Compilerfehler C0117	Compilerfehler C0118	Compilerfehler C0119
Compilerfehler C0120	Compilerfehler C0122	Compilerfehler C0125
Compilerfehler C0126	Compilerfehler C0130	Compilerfehler C0131
Compilerfehler C0132	Compilerfehler C0136	Compilerfehler C0139
Compilerfehler C0140	Compilerfehler C0141	Compilerfehler C0142
Compilerfehler C0143	Compilerfehler C0144	Compilerfehler C0145
Compilerfehler C0149	Compilerfehler C0161	Compilerfehler C0162
Compilerfehler C0168	Compilerfehler C0169	Compilerfehler C0174
Compilerfehler C0175	Compilerfehler C0177	Compilerfehler C0178
Compilerfehler C0182	Compilerfehler C0185	Compilerfehler C0189
Compilerfehler C0190	Compilerfehler C0195	Compilerfehler C0196
Compilerfehler C0197	Compilerfehler C0198	Compilerfehler C0199
Compilerfehler C0201	Compilerfehler C0203	Compilerfehler C0204
Compilerfehler C0205	Compilerfehler C0206	Compilerfehler C0208
Compilerfehler C0211	Compilerfehler C0212	Compilerfehler C0216

Compilerfehler C0217	Compilerfehler C0218	Compilerfehler C0219
Compilerfehler C0221	Compilerfehler C0222	Compilerfehler C0224
Compilerfehler C0227	Compilerfehler C0228	Compilerfehler C0234
Compilerfehler C0235	Compilerfehler C0236	Compilerfehler C0237
Compilerfehler C0238	Compilerfehler C0239	Compilerfehler C0240
Compilerfehler C0241	Compilerfehler C0242	Compilerfehler C0243
Compilerfehler C0380	Compilerfehler C0509	Compilerfehler C0511

Tab. 17-15:

17.11.1 Compilerfehler C0001

Meldung:	Konstante <constant value> zu groß für Typ <data type>
Mögliche Fehlerursache:	Eine "getypte" Konstante ist zu groß für den angegebenen Datentyp oder eine Konstante ist zu groß für jeden möglichen Datentyp.
Fehlerbehebung:	Verwenden Sie kleinere Konstanten oder einen passenden Datentyp für eine getypte Konstante.
	<i>Beispiel für den Fehler:</i>

```
PROGRAM PLC_PRG
VAR
    test1: INT;
    test2: INT;
    test3: LREAL;
END_VAR

test1 := 12345678912345566991923939292939911;
test2 := INT#123456;
test3 := 10E500;

--> C0001: Constant '12345678912345566991923939292939911' is too big for type 'ANY_INT'
--> C0001: Constant 'INT#123456' is too big for type 'INT'
--> C0001: Constant '10E500' is too big for type 'ANY_REAL'
```

17.11.2 Compilerfehler C0002

Meldung:	<operator 1> oder <operator 2> anstelle von <tag> erwartet
Mögliche Fehlerursache:	Syntaxfehler
Fehlerbehebung:	Verwenden Sie die korrekte Syntax
	<i>Beispiel für den Fehler:</i>

```
PROGRAM PLC_PRG
Fun(1;

--> C0002: ', ' or ')' expected instead of ';'


```

17.11.3 Compilerfehler C0003

Meldung:	<value> ist keine gültige Bit-Nummer für <variable>
Mögliche Fehlerursache:	Es wird versucht auf ein Bit zuzugreifen, dass außerhalb des Bereichs für einen Datentyp liegt
Fehlerbehebung:	Verwenden Sie für den Bitzugriff einen Bit-Wert, der kleiner ist als die Anzahl der Bits im Datentyp der Variable
	<i>Beispiel für den Fehler:</i>

```
PROGRAM PLC_PRG
VAR
    test1: WORD;
```

```

    test2: BOOL;
END_VAR

test1 := test2.17;

--> C0003: '17' is not a valid bit number for 'w'

```

17.11.4 Compilerfehler C0004

- Meldung:** <variable> ist keine Komponente von <structure>
- Mögliche Fehlerursache:** Komponentenzugriff mit "." auf eine Variable, die kein strukturierter Wert ist oder nicht als Komponente der Struktur existiert
- Fehlerbehebung:** Greifen Sie auf eine definierte Komponente zu, oder ändern Sie die Definition im Datentyp auf den zugegriffen wird. Die Eingabeunterstützung "Komponenten auflisten" bietet Ihnen alle gültigen Zugriffe an dieser Position an
- Beispiel für den Fehler:*

```

TYPE DUT:
STRUCT
    x, y : INT;
END_STRUCT
END_TYPE

PROGRAM PLC_PRG
VAR
    test1 : DUT;
    test2 : INT;
END_VAR

test2 := test1.z;

--> C0032: Type 'Unknown type: 'test1.z'' cannot be converted to type 'INT'
--> C0004: 'z' is to a component of 'DUT'

```

17.11.5 Compilerfehler C0005

- Meldung:** Konstantenüberlauf in Adresse <address>
- Mögliche Fehlerursache:** Mindestens eine Komponente in der Adresse passt nicht in einen 32-Bit-Integer-Wert
- Fehlerbehebung:** Verwenden Sie einen gültigen Adressausdruck
- Beispiel für den Fehler:*

```

PROGRAM PLC_PRG
VAR
    X: BYTE;
END_VAR

X := %QB5555555555;

--> C0005: Constant overflow in address '%??'

```

17.11.6 Compilerfehler C0006

- Meldung:** <operator> an Stelle von <token>
- Mögliche Fehlerursache:** Syntaxfehler
- Fehlerbehebung:** Verwenden Sie die korrekte Syntax
- Beispiel für den Fehler:*

```

PROGRAM PLC_PRG
VAR
    x: INT;
    bTest : BOOL;
END_VAR

IF bTest

```

```
      x := 9;
END_IF

--> C0006: 'THEN' expected instead of 'x'
```

17.11.7 Compilerfehler C0007

Meldung: Ausdruck an Stelle von <token> erwartet

Mögliche Fehlerursache: Syntaxfehler

Fehlerbehebung: Verwenden Sie die korrekte Syntax

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    x: INT;
    bTest : BOOL;
END_VAR

IF THEN
    x := 9;
END_IF

--> C0007: Expression expected instead of 'THEN'
```

17.11.8 Compilerfehler C0008

Meldung: Unerwartetes "End-of-file" gefunden: <operator 1>, <operator 2> oder <operator 3> erwartet

Mögliche Fehlerursache: Syntaxfehler

Fehlerbehebung: Verwenden Sie die korrekte Syntax

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    x: INT;
    bTest : BOOL;
END_VAR

IF bTest THEN
    x := 9;

--> C0008: Unexpected end-of-file found: 'ELSIF', 'ELSE' or 'END_IF' expected
```

17.11.9 Compilerfehler C0009

Meldung: Unerwartetes Token <token> gefunden

Mögliche Fehlerursache: Syntaxfehler

Fehlerbehebung: Verwenden Sie die korrekte Syntax

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
END_VAR

END_FOR;

--> C0009: Unexpected token 'END_FOR' found
```

17.11.10 Compilerfehler C0010

Meldung: Unerwartetes End-of-file gefunden: <token> erwartet

Mögliche Fehlerursache: Syntaxfehler

Fehlerbehebung: Verwenden Sie die korrekte Syntax
Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i: INT;
END_VAR

FOR i := 0 TO 2 DO
;

--> C0010: Unexpected end-of-file 'END_FOR' found
```

17.11.11 Compilerfehler C0011

Meldung: Keine CASE-Marke gefunden
Mögliche Fehlerursache: Syntaxfehler in einem CASE-Statement. Ein Statement innerhalb einer CASE-Anweisung ist keiner CASE-Marke zugeordnet
Fehlerbehebung: Fügen Sie eine CASE-Marke ein
Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i: INT;
    x: INT;
END_VAR

CASE i OF
    x := 9;
END_CASE

--> C0011: No 'CASE' label found
```

Fehlerbehebung:

```
CASE i OF
0:
    x := 9;
END_CASE
```

17.11.12 Compilerfehler C0013

Meldung: Es wird wenigstens eine Anweisung erwartet
Mögliche Fehlerursache: An einigen Stellen im Code wird mindestens eine Anweisung erwartet. Beispielsweise im THEN und im ELSE-Teil einer IF-Anweisung oder im Rumpf einer FOR-Schleife
Fehlerbehebung: Fügen Sie mindestens eine Anweisung an der ausgewiesenen Position ein. Es genügt auch die leere Anweisung ";".
Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    bTest: BOOL;
END_VAR

IF bTest THEN
END_IF

--> C0013: At least one statement is expected
```

17.11.13 Compilerfehler C0015

Meldung: Zählvariableninitialisierung erwartet

Mögliche Fehlerursache: Syntaxfehler in einer FOR-Schleife.

Die Zählvariable wird nicht korrekt initialisiert.

Fehlerbehebung: Beachten Sie die korrekte Syntax der FOR-Schleife:

Programm:

```
FOR i := 0 TO 10 DO
;
END_FOR
```

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i: INT;
END_VAR

FOR i TO 10 DO
;
END_FOR

--> C0015: Counter initialization expected
```

Fehlerbehebung:

```
FOR i := 0 TO 10 DO
;
END_FOR
```

17.11.14 Compilerfehler C0018

Meldung: <expression> ist kein gültiges Zuweisungsziel

Mögliche Fehlerursache: Auf der linken Seite einer Zuweisung steht ein Ausdruck, für den kein Schreibrecht besteht, z.B. eine Konstante

Fehlerbehebung: Weisen Sie nur auf Variablen Werte zu, auf die auch ein Schreibzugriff besteht

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i: INT;
END_VAR
VAR_CONSTANT
    j: INT := 0;
END_VAR

j := i;

--> C0018: 'j' is not a valid assignment target
```

17.11.15 Compilerfehler C0020

Meldung: <statement> ist keine gültige Anweisung

Mögliche Fehlerursache: Syntaxfehler: Beispielsweise ein Zeichen vergessen oder ein Zeichen zu viel

Fehlerbehebung: Achten Sie auf eine syntaktisch korrekte Schreibweise

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    x : INT;
END_VAR

x = 2;

--> C0020: '(x = 2); ' is no valid statement
```

Fehlerbehebung:

Example:
x := 2;

17.11.16 Compilerfehler C0022

Meldung: <operator> benötigt genau <number of operands> Operanden
Mögliche Fehlerursache: Einem Operator werden zu viele oder zu wenige Operanden zugewiesen
Fehlerbehebung: Weisen Sie dem Operator die benötigte Anzahl an Operanden zu
Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : INT;
    pt: POINTER TO INT;
END_VAR

pt := ADR(i,1);

--> C0022: 'ADR' needs exactly '1' operands
```

Fehlerbehebung:

Example:
pt := ADR(i);

17.11.17 Compilerfehler C0023

Meldung: <operator> benötigt wenigstens <number of operands> Operanden
Mögliche Fehlerursache: Einem Operator werden zu wenige Operanden zugewiesen
Fehlerbehebung: Weisen Sie dem Operanden die benötigte Anzahl an Operanden zu
Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : INT;
END_VAR

i := MUX(30,40);

--> C0023: 'MUX' needs at least '3' operands
```

Fehlerbehebung:

Example:
i := MUX(30,40,50);

17.11.18 Compilerfehler C0026

Meldung: Bezeichner an Stelle von <invalid identifier> erwartet
Mögliche Fehlerursache: Einer Methode wird ein ungültiger Bezeichner gegeben
Fehlerbehebung: Verwenden Sie gültige Bezeichner
Beispiel für den Fehler:

```
METHOD 123
VAR_INPUT
END_VAR

--> C0243: The name used in the signature is not identical to the object name
--> C0026: Identifier expected instead of '123'
```

Fehlerbehebung:

Example:
METHOD METH123

17.11.19 Compilerfehler C0027

Meldung: Länge der Zeichenkette erwartet nach "("
Mögliche Fehlerursache: Länge der Zeichenkette wird nicht angegeben
Fehlerbehebung: Geben Sie eine Zeichenlänge innerhalb der Klammern an
Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    str : STRING();
END_VAR
```

```
--> C0027: size of string expected after '('
--> C0006: ';;, :=, REF=, ( or [' expected instead of ')
```

Fehlerbehebung:

Example:
str : STRING(100);

17.11.20 Compilerfehler C0030

Meldung: Direkte Adresse nach "AT" an Stelle von <identifizier> erwartet
Mögliche Fehlerursache: Nach "AT" wird keine oder eine ungültige Adresse zugewiesen
Fehlerbehebung: Geben Sie eine gültige Adresse an
Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i AT ABC : INT;
END_VAR
```

```
--> C0030: Direct Address expected after 'AT' instead of 'ABC'
```

Fehlerbehebung:

Example:
i AT %IW0 : INT;

17.11.21 Compilerfehler C0031

Meldung: Typdefinition an Stelle von <no data type> erwartet
Mögliche Fehlerursache: Dem Bezeichner wird eine ungültige Typdefinition zugewiesen
Fehlerbehebung: Geben Sie eine gültige Typdefinition an
Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : 0;
END_VAR
```

```
--> C0031: Type definition expected instead of '0'
```

Fehlerbehebung:

Example:
i : INT;

17.11.22 Compilerfehler C0032

- Meldung:** Typ <type 1> kann nicht in Typ <type 2> konvertiert werden
- Mögliche Fehlerursache:** Eine Variable wird einer anderen Variable mit inkompatiblem Typ zugewiesen
- Fehlerbehebung:** Verwenden Sie eine Typkonvertierung
- Beispiel für den Fehler:*
-

```
PROGRAM PLC_PRG
VAR
    test1: INT;
    test2: STRING;
END_VAR

test1 := test2;

-->C0032: Type 'STRING' cannot be converted to type 'INT'
```

Fehlerbehebung:

Example:
test1 := TO_INT(test2);

17.11.23 Compilerfehler C0035

- Meldung:** Programmname, Funktion oder Funktionsbausteininstance an Stelle von <invalid function> erwartet
- Mögliche Fehlerursache:** Es wird eine Funktion aufgerufen, die nicht existiert
- Fehlerbehebung:** Achten Sie darauf nur Programmnamen, Funktionen oder Funktionsbausteininstance aufzurufen, die existieren
- Beispiel für den Fehler:*
-

```
PROGRAM PLC_PRG
VAR
END_VAR

PLC_PRG.METH1();

METHOD METH
VAR INPUT
END_VAR

--> C0004: 'METH1' is no component of 'PLC_PRG'
--> C0035: Program name, function or function block instance expected instead of 'PLC_PRG.METH1'
```

Fehlerbehebung:

Example:
PLC_PRG.METH();

17.11.24 Compilerfehler C0037

- Meldung:** <invalid input> ist kein Eingang von <function name>
- Mögliche Fehlerursache:** Beim Funktionsaufruf wird eine lokale Variable definiert
- Fehlerbehebung:** Deklarieren Sie die Variable als Input-Parameter

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    i : INT;
END_VAR
i := TEST(iVar := 1);

FUNCTION TEST : INT
VAR
    iVar : INT;
END_VAR

--> C0037: 'iVar' is no input of 'TEST'

```

Fehlerbehebung:

```

Example:
VAR_INPUT
    iVar : INT;
END_VAR

```

17.11.25 Compilerfehler C0038

Meldung: <invalid output> ist kein Ausgang von <function name>

Mögliche Fehlerursache: Beim Funktionsaufruf wird eine lokale Variable als Output behandelt

Fehlerbehebung: Deklarieren Sie die Variable als Output-Parameter

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    i : INT;
    x : INT;
END_VAR
i := TEST(iVar => x);

FUNCTION TEST : INT
VAR
    iVar : INT;
END_VAR

--> C0038: 'iVar' is no output of 'TEST'

```

Fehlerbehebung:

```

Example:
VAR_OUTPUT
    iVar : INT;
END_VAR

```

17.11.26 Compilerfehler C0039

Meldung: VAR_IN_OUT <invalid variable> muss im Aufruf von <function block name> zugewiesen werden

Mögliche Fehlerursache: Einem Funktionsbaustein, der eine IN_OUT-Variable benötigt, wird keine IN_OUT-Variable übergeben

Fehlerbehebung: Weisen Sie die IN_OUT-Variable zu

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR
inst();

```

```
FUNCTION_BLOCK FB
VAR_IN_OUT
  _inout : INT;
END_VAR
```

```
--> C0039: VAR_IN_OUT 'inout' must be assigned in call of 'FB'
```

Fehlerbehebung:

Example:
inst(inout := i);

17.11.27 Compilerfehler C0040

- Meldung:** Funktion <function name> benötigt genau <number of inputs> Eingänge
- Mögliche Fehlerursache:** Der aufgerufenen Funktion werden zu viele oder zu wenige Parameter übergeben
- Fehlerbehebung:** Übergeben Sie der Funktion genau die Anzahl von Parametern, die erwartet werden
- Beispiel für den Fehler:*

```
PROGRAM PLC_PRG
VAR
  i : INT;
END_VAR
i := TEST(1,2);

FUNCTION TEST : INT
VAR_INPUT
  _IN: INT;
END_VAR

--> C0040: Function 'TEST' requires exactly '1' inputs
```

Fehlerbehebung:

Example:
i := Test(1);

17.11.28 Compilerfehler C0041

- Meldung:** VAR_IN_OUT-Parameter <parameter name> von <function name> benötigt eine Variable mit Schreibzugriff als Eingang
- Mögliche Fehlerursache:** Der übergebene Parameter ist keine Variable mit Schreibzugriff, sondern z.B. eine Konstante
- Fehlerbehebung:** Übergeben Sie der Funktion einen VAR_IN_OUT-Parameter mit Schreibzugriff
- Beispiel für den Fehler:*

```
PROGRAM PLC_PRG
VAR
  i : INT;
  x : INT;
END_VAR
i := Test(31415);

FUNCTION TEST : INT
VAR_IN_OUT
  _in_out: INT;
END_VAR
```

```
--> C0041: VAR_IN_OUT' parameter 'in_out' of 'TEST' needs variable with write access as input
```

Fehlerbehebung:

Example:
i := Test(x);

17.11.29 Compilerfehler C0043

- Meldung:** Falscher formaler Parameter: <parameter name> an dieser Stelle erwartet
- Mögliche Fehlerursache:** Der Funktion werden die Parameter in falscher Reihenfolge explizit zugewiesen
- Fehlerbehebung:** Geben Sie die Parameter in richtiger Reihenfolge an
- Beispiel für den Fehler:*
-

```
PROGRAM PLC_PRG
VAR
    i : INT;
END_VAR
i := Test(iPar2 := 2, 5);
```

```
FUNCTION Test : INT
VAR_INPUT
    iPar1 : INT;
    iPar2 : INT;
END_VAR
```

```
--> C0043: Wrong formal parameter: 'iPar1' expected in this place
--> C0412: Multiple input assignments for parameter ''
```

Fehlerbehebung:

Example:
i := Test(5, iPar2 := 2);

17.11.30 Compilerfehler C0044

- Meldung:** Eingangszuweisung für Parameter <input variable name> in Aufruf von <function block name> fehlt
- Mögliche Fehlerursache:** Es wird ein Parameter übergeben, obwohl keine Input-Variable deklariert ist
- Fehlerbehebung:** Deklarieren Sie eine Input-Variable
- Beispiel für den Fehler:*
-

```
PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR
inst(1);
```

```
FUNCTION BLOCK FB
VAR_INPUT
END_VAR
```

```
--> C0044: Assignment to input missing for parameter '1' in call of 'FB'
```

Fehlerbehebung:

Example:
VAR_INPUT

```
in : INT;  
END_VAR
```

17.11.31 Compilerfehler C0045

- Meldung:** Verwendung von `THIS` ist in diesem Zusammenhang nicht erlaubt
- Mögliche Fehlerursache:** `THIS` kann nur in einer Methode, Aktion, Transition oder im Rumpf eines Funktionsblocks verwendet werden, um auf die aktuelle Instanz zu verweisen. An allen anderen Positionen kommt es zu dieser Fehlermeldung.
- Fehlerbehebung:** Verwenden Sie `THIS` nur in erlaubtem Kontext
- Beispiel für den Fehler:*

```
PROGRAM PLC_PRG  
VAR  
    test1: INT;  
END_VAR  
  
THIS^.test1 := 19;  
  
--> C0018: 'THIS^.test1' is not a valid assignment target  
--> C0062: 'THIS^' is not a structure variable  
--> C0045: Use of 'THIS' is not allowed in this context
```

17.11.32 Compilerfehler C0046

- Meldung:** Bezeichner <identifier name> nicht definiert
- Mögliche Fehlerursache:** Es wird ein Bezeichner verwendet, der nicht deklariert ist
- Fehlerbehebung:** Deklarieren Sie die Variablen, die Sie verwenden wollen
- Beispiel für den Fehler:*

```
PROGRAM PLC_PRG  
VAR  
  
END_VAR  
  
i := 1;  
  
--> C0018: 'i' is no valid assignment target  
--> C0046: Identifier 'i' not defined
```

Fehlerbehebung:

```
Example:  
VAR  
    i : INT;  
END_VAR
```

17.11.33 Compilerfehler C0047

- Meldung:** Indizierung mit `[]` kann nicht auf einen Ausdruck mit Typ <data type> angewendet werden
- Mögliche Fehlerursache:** Es wird ein Datentyp mit `[]` indiziert, der kein Array ist
- Fehlerbehebung:** Indizieren Sie nur Datentypen mit `[]`, wenn diese auch als Array deklariert sind
- Beispiel für den Fehler:*

```
PROGRAM PLC_PRG  
VAR  
    i : INT;  
END_VAR  
i[1];
```

--> C0047: Cannot apply indexing with '[' to an expression of type 'INT'

17.11.34 Compilerfehler C0048

- Meldung:** Datenfeld benötigt genau <number> Indizes
- Mögliche Fehlerursache:** Bei der Verwendung des Arrays werden zu viele oder zu wenige Indizes angegeben
- Fehlerbehebung:** Geben Sie so viele Indizes an, wie dem Array Dimensionen zugewiesen sind
Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    arr1 : ARRAY[1..2,1..3] OF INT;
END_VAR
arr1[1] := 5;
```

--> C0048: Array requires exactly 2 indexes

Fehlerbehebung

Example:
arr1[1,2] := 5;

17.11.35 Compilerfehler C0049

- Meldung:** Der konstante Index <index value> liegt außerhalb des Bereichs von <start index> bis <end index>
- Mögliche Fehlerursache:** Es wird ein Index angegeben, der außerhalb der Größe des Arrays liegt
- Fehlerbehebung:** Verwenden Sie nur Indizes, die innerhalb der Größe des Arrays liegen
Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    arr1 : ARRAY[1..2] OF INT;
END_VAR
arr1[3] := 1;
```

--> C0049: The constant index '3' is not within the range from '1' to '2'

Fehlerbehebung

Example:
arr1[2] := 1;

17.11.36 Compilerfehler C0050

- Meldung:** Bitzugriff benötigt ein Literal oder eine symbolische ganzzahlige Konstante
- Mögliche Fehlerursache:** Es wird beim Bitzugriff kein Literal oder eine ganzzahlige Konstante angegeben
- Fehlerbehebung:** Verwenden Sie ein Literal oder eine ganzzahlige Konstante
Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : INT;
    x : INT;
END_VAR
i.x := FALSE;
```



```
--> C0018: 'i.x' is no valid assignment target  
--> C0050: Bitaccess requires literal or symbolic integer constant
```

Fehlerbehebung

Example:
i := Test(x);

17.11.37 Compilerfehler C0061

Meldung: Bitzugriff auf einen Funktionsaufruf ist nicht erlaubt
Mögliche Fehlerursache: Es wird ein Bitzugriff auf eine Funktion ausgeführt
Fehlerbehebung: Verwenden Sie den Bitzugriff nur bei unterstützten Datentypen
Beispiel für den Fehler:

```
PROGRAM PLC_PRG  
VAR  
END_VAR  
  
Test().2;  
  
FUNCTION Test : INT  
VAR_INPUT  
  
END_VAR
```

```
--> C0061: Bitaccess on function call is not allowed
```

17.11.38 Compilerfehler C0062

Meldung: <variable name> ist keine Strukturvariable
Mögliche Fehlerursache: Eine Variable, die keine Strukturvariable ist, wird wie eine Strukturvariable behandelt
Fehlerbehebung: Achten Sie darauf, dass die Variable tatsächlich eine Strukturvariable ist
Beispiel für den Fehler:

```
PROGRAM PLC_PRG  
VAR  
    pt : PUNKT;  
    i : INT;  
END_VAR  
  
i.x := 1024;  
  
TYPE Punkt :  
    STRUCT  
        x : REAL;  
        y : REAL;  
    END_STRUCT  
END_TYPE
```

```
--> C0018: 'i.x' is no valid assignment target  
--> C0062: 'Variable' is no structured variable
```

Fehlerbehebung:

Example:
pt.x := 1024;

17.11.39 Compilerfehler C0064

- Meldung:** Dereferenzierung benötigt einen Pointer
- Mögliche Fehlerursache:** Es wird eine Variable "dereferenziert", die keine Pointervariable ist
- Fehlerbehebung:** "Dereferenzieren" Sie nur Variablen, die Pointervariablen sind

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : INT;
    pi : POINTER TO INT;
END_VAR
i^:=1;
```

```
--> C0018: 'i^' not a valid assignment target
--> C0064: Dereferencing requires a pointer
```

Fehlerbehebung:

Example:

```
pi := ADR(i);
pi^ := 1;
```

17.11.40 Compilerfehler C0066

- Meldung:** Typ <data type> kann nicht mit Typ <data type> verglichen werden
- Mögliche Fehlerursache:** Es werden zwei nicht miteinander vergleichbare Datentypen verglichen
- Fehlerbehebung:** Vergleichen Sie nur Datentypen, die miteinander vergleichbar sind

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : INT;
    re : REAL;
    str: STRING;
    b : BOOL;
END_VAR
```

```
b := i > str;
```

```
--> C0066: Cannot compare type 'INT' with type 'STRING'
```

Fehlerbehebung:

Example:

```
b := i > re;
```

17.11.41 Compilerfehler C0068

- Meldung:** Vergleich für Objekte vom Typ <data type> nicht möglich
- Mögliche Fehlerursache:** Es wird ein Vergleich mit Objekten durchgeführt, bei denen ein Vergleich nicht möglich ist
- Fehlerbehebung:** Vergleichen Sie nur Datentypen, bei denen ein Vergleich möglich ist, z.B. INT, REAL, etc.

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    b : BOOL;
    arr1 : ARRAY [1..2] OF INT;
    arr2 : ARRAY [1..2] OF INT;
```

```
END_VAR
```

```
b := arr1 > arr2;
```

```
--> C0068: Compare not possible on objects of type 'ARRAY [1..2]'
```

17.11.42 Compilerfehler C0069

Meldung: Vergleich für Objekte vom Typ <data type> oder <data type> nicht möglich

Mögliche Fehlerursache: Es wird ein Vergleich mit zwei unterschiedlichen Objekten durchgeführt, bei denen ein Vergleich nicht möglich ist

Fehlerbehebung: Vergleichen Sie nur Datentypen, bei denen ein Vergleich möglich ist, z.B. INT, REAL, etc.

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    b : BOOL;
    arr1 : ARRAY [1..2] OF INT;
    arr2 : ARRAY [1..3] OF INT;
END_VAR
```

```
b := arr1 > arr2;
```

```
--> C0069: Compare not possible on objects of type 'ARRAY [1..2]' or 'ARRAY [1..3]'
```

17.11.43 Compilerfehler C0070

Meldung: INI-Operator benötigt Funktionsbaustein oder DUT-Instanz

Mögliche Fehlerursache: Auf den INI-Operator wird keine Funktionsbaustein oder DUT-Instanz angewendet

Fehlerbehebung: Übergeben Sie dem INI-Operator nur Funktionsbaustein oder DUT-Instanzen

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    b : BOOL;
    inst : FB;
END_VAR
```

```
b := INI(b, TRUE);
```

```
FUNCTION_BLOCK FB
VAR
END_VAR
```

```
--> C0070: 'INI' operator needs function block instance or data unit type instance
```

Fehlerbehebung:

```
Example:
b := INI(inst, TRUE);
```

17.11.44 Compilerfehler C0072

Meldung: Operator <operator name> kann auf Typ <data type> nicht angewendet werden

Mögliche Fehlerursache: Ein Operator wird auf einen inkompatiblen Typ angewendet

Fehlerbehebung: Wenden Sie Operatoren nur auf kompatible Datentypen an

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : INT;
    str : STRING;
END_VAR
```

```
str := ABS(str);
```

```
--> C0072: Operator 'Abs' is not possible on type 'STRING'
```

Fehlerbehebung:

Example:
i := ABS(i);

17.11.45 Compilerfehler C0074

Meldung: Unerwartete Datenfeldinitialisierung

Mögliche Fehlerursache: Syntaxfehler bei der Datenfeldinitialisierung

Fehlerbehebung: Korrigieren Sie die Syntax

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    arr1 : INT := [1,2,3,4,5,6];
END_VAR
```

```
--> C0074: Unexpected array initialisation
```

```
--> C0032: Cannot convert type 'Unknown type: [1,2,3,4,5,6]' to type 'INT'
```

Fehlerbehebung:

Example:
arr1 : ARRAY [1..6] OF INT := [1,2,3,4,5,6];

17.11.46 Compilerfehler C0075

Meldung: Zu viele Initialisierer für Datenfeld

Mögliche Fehlerursache: Es werden zu viele Werte für die Größe des Arrays angegeben

Fehlerbehebung: Die Anzahl der zugewiesenen Werte muss der Größe des Arrays entsprechen

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    arr1 : ARRAY [1..5] OF INT := [1,2,3,4,5,6];
END_VAR
```

```
--> C0075: Unexpected array initialisation
```

Fehlerbehebung:

Example:
arr1 : ARRAY [1..6] OF INT := [1,2,3,4,5,6];

17.11.47 Compilerfehler C0076

Meldung: Unerwartete Strukturinitialisierung

Mögliche Fehlerursache: Syntaxfehler bei der Strukturinitialisierung

Fehlerbehebung: Achten Sie auf eine korrekte Syntax

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    st1 : INT := (p1 := 1);
END_VAR

--> C0076: Unexpected structure initialisation
--> C0032: Cannot convert type 'STRUCT(p1:=1)' to type 'INT'
--> C0046: Identifier 'p1' not defined
--> C0018: 'p1' is no valid assignment target
```

Fehlerbehebung:

Example:
st1 : STRUCT1 := (p1:=1,p2:=10);

17.11.48 Compilerfehler C0077

Meldung: Unbekannter Typ: <data type>

Mögliche Fehlerursache: Ungültiger Datentyp bei der Deklaration (möglicherweise ein Syntaxfehler)

Fehlerbehebung: Geben Sie nur gültige Datentypen an

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : INTEGER;
END_VAR

--> C0077: Unknown type: 'INTEGER'
```

Fehlerbehebung:

Example:
arr1 : ARRAY[1..2] OF STRUCT1 := (p1:=1,p2:=10);

17.11.49 Compilerfehler C0080

Meldung: Funktionsbaustein <function block name> muss für den Zugriff instanziiert werden

Mögliche Fehlerursache: Fehlende Funktionsbaustein-Instanziierung

Fehlerbehebung: Instanziierten Sie den Funktionsbaustein

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
END_VAR

FB();

FUNCTION_BLOCK FB
VAR
END_VAR

--> C0080: Functionblock 'FB' must be instantiated to be accessed
```

Fehlerbehebung:

Example:
 VAR
 inst : FB;
 END_VAR
 inst();

17.11.50 Compilerfehler C0081

Meldung: Unerwartetes Pragma: <pragma name> ohne zugehörige IF-Bedingung gefunden

Mögliche Fehlerursache: Bei der Pragmaverwendung fehlt die IF-Bedingung

Fehlerbehebung: Ergänzen Sie die IF-Bedingung des Pragmas

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : INT;
END_VAR

i := 5;
{END_IF}
```

--> C0081: Unexpected Pragma: 'END_IF' found without matching 'if'

Fehlerbehebung:

Example:
 {IF <expression>}
 i := 5;
 {END_IF}

17.11.51 Compilerfehler C0082

Meldung: <invalid pragma> ist keine gültige Pragma-Bedingung

Mögliche Fehlerursache: Bei der Pragmaverwendung wird in der IF-Bedingung ein ungültiger Ausdruck verwendet

Fehlerbehebung: Verwenden Sie gültige Pragma-Bedingungen

Beispiel für den Fehler:

```
VAR
    i : INT;
END_VAR

{IF abc}
i := 5;
{END_IF}
```

--> C0082: '!!!ERROR!!!' is no valid condition for pragma

Fehlerbehebung:

Example:
 {IF defined (abc)}

17.11.52 Compilerfehler C0084

Meldung: <pragma operand> ist kein gültiger Pragma-Operand

Mögliche Fehlerursache: Syntaxfehler

Fehlerbehebung: Verwenden Sie gültige Pragma-Operanden

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
END_VAR

{IF defined(0)}
{END_IF}

--> C0084: 'defined(null)' is no valid operand for pragma
```

Fehlerbehebung:

Example:
{IF defined (abc)}

17.11.53 Compilerfehler C0086

- Meldung:** Keine Definition für Schnittstelle <interface name> gefunden
- Mögliche Fehlerursache:** Es wird eine nicht definierte Schnittstelle verwendet
- Fehlerbehebung:** Definieren Sie die Schnittstelle
- Beispiel für den Fehler:*
-

```
PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR

FUNCTION_BLOCK FB IMPLEMENTS XY
VAR
END_VAR

--> C0086: No definition found for interface 'XY'
```

Fehlerbehebung:

Example:
INTERFACE XY

17.11.54 Compilerfehler C0087

- Meldung:** Keine Implementierung für Methode <method name> aus Schnittstelle <interface name>
- Mögliche Fehlerursache:** Eine Methode, die von der Schnittstelle vorgegeben ist, wird vom implementierenden Funktionsbaustein nicht zur Verfügung gestellt
- Fehlerbehebung:** Implementieren Sie alle Methoden, die von der Schnittstelle vorgegeben werden
- Beispiel für den Fehler:*
-

```
PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR

INTERFACE XY
METHOD METH1
VAR_INPUT
END_VAR

FUNCTION_BLOCK FB IMPLEMENTS XY
VAR
END_VAR
METHOD METH2
```

Referenz Programmierung

```
VAR_INPUT
END_VAR
```

```
--> C0087: There is no implementation for method 'METH1' defined in interface 'XY'
```

17.11.55 Compilerfehler C0089

Meldung: Schnittstelle der überladenen Methode {0} von Schnittstelle {1} stimmt nicht mit der Deklaration überein

Mögliche Fehlerursache: Die Signatur der implementierten Methode stimmt nicht mit der Signatur der Methode in der Schnittstelle überein

Fehlerbehebung: Achten Sie darauf, dass die gleichen Rückgabetypen und Parameter deklariert sind

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR

INTERFACE XY
METHOD METH1
VAR_INPUT
    iPar : INT;
END_VAR

FUNCTION_BLOCK FB IMPLEMENTS XY
VAR
END_VAR
METHOD METH1
VAR_INPUT
END_VAR
```

```
--> C0089: Interface of overridden method 'METH1' of interface 'XY' does not match declaration
```

17.11.56 Compilerfehler C0090

Meldung: Keine Definition für Basis <function name> gefunden

Mögliche Fehlerursache: Der als Basis angegebene Funktionsbaustein existiert nicht oder ist kein Funktionsbaustein

Fehlerbehebung: Verwenden Sie einen Funktionsbaustein als Basis

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR

FUNCTION_BLOCK FB EXTENDS POU
VAR
END_VAR

FUNCTION POU
VAR
END_VAR
```

```
--> C00090: No definition found for base class 'POU'
```

17.11.57 Compilerfehler C0091

Meldung: Rekursion in Basisfunktionsbausteinliste: <function name>

Mögliche Fehlerursache: Ein Basisfunktionsbaustein wird um sich selbst erweitert

Fehlerbehebung: Rekursion in Basisfunktionsbausteinlisten ist nicht möglich

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR
```

```
FUNCTION_BLOCK FB EXTENDS FB
VAR
END_VAR
```

--> C00091: Recursion in base function block list: FB -> FB

17.11.58 Compilerfehler C0094

Meldung: Schnittstelle der überladenen Methode <method name> von Basis <function block name> stimmt nicht mit der Deklaration überein

Mögliche Fehlerursache: Die Signatur der Methode der ersten Schnittstelle stimmt nicht mit der Signatur der Methode in der zweiten Schnittstelle, die um die erste erweitert wird, überein

Fehlerbehebung: Gleichen Sie die Signaturen an

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    xyz : XY2;
END_VAR
```

```
FUNCTION_BLOCK XY
METHOD METH1
VAR_INPUT
END_VAR
```

```
FUNCTION_BLOCK XY2 EXTENDS XY
METHOD METH1
VAR_INPUT
    iPar : BOOL;
END_VAR
```

--> C00094: Interface of the overridden method METH1 of interface XY doesn't match declaration

17.11.59 Compilerfehler C0096

Meldung: Nur ein einziger Basisfunktionsbaustein darf in der EXTENDS-Liste definiert sein

Mögliche Fehlerursache: Zwei oder mehr Basisfunktionsbausteine werden in der EXTENDS-Liste definiert

Fehlerbehebung: Definieren Sie nur einen Basisfunktionsbaustein in der EXTENDS-Liste

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    fb : FB;
END_VAR
```

```
FUNCTION_BLOCK FB EXTENDS FB2, FB3
VAR
END_VAR
```

```
FUNCTION_BLOCK FB2
VAR
END_VAR
```

```
FUNCTION_BLOCK FB3
VAR
END_VAR
```

--> C00096: Only one base function block may be defined in EXTENDS-list

17.11.60 Compilerfehler C0097

Meldung: Mehrfachdefinition von Variable <variable name> in Funktionsbaustein <function block name> und in Basis <base function block name>

Mögliche Fehlerursache: Eine Variable wird in einem Funktionsbaustein und seiner Basis mit gleichem Namen deklariert

Fehlerbehebung: Verwenden Sie unterschiedliche Variablennamen

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    fb : FB;
END_VAR
```

```
FUNCTION_BLOCK FB EXTENDS FB2
VAR
    i : INT;
END_VAR
```

```
FUNCTION_BLOCK FB2
VAR
    i : INT;
END_VAR
```

--> C00097: Duplicate definition of variable 'i' in function block 'FB' and in base 'FB2'

17.11.61 Compilerfehler C0098

Meldung: Das Schlüsselwort "FUNCTIONBLOCK" wird nicht mehr unterstützt.
Verwenden Sie stattdessen "FUNCTION_BLOCK".

Mögliche Fehlerursache: Syntaxfehler

Fehlerbehebung: Verwenden Sie das Schlüsselwort "FUNCTION_BLOCK"

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR
```

```
FUNCTIONBLOCK FB
VAR
END_VAR
```

--> C00098: The keyword "FUNCTIONBLOCK" is no longer supported.
--> Use "FUNCTION_BLOCK" instead.

17.11.62 Compilerfehler C0101

Meldung: Datenrekursion: <recursion>

Mögliche Fehlerursache: Rekursive Dateninitialisierung über zwei Funktionsbausteine

Fehlerbehebung: Vermeiden Sie Rekursionen bei Dateninitialisierungen

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    inst0 : FB1;
END_VAR
```

```
FUNCTION_BLOCK FB1  
VAR  
    inst1 : FB2;  
END_VAR
```

```
FUNCTION_BLOCK FB2  
VAR  
    inst2 : FB1;  
END_VAR
```

--> C0101: Data Recursion: FB1->FB2->FB1.

17.11.63 Compilerfehler C0102

- Meldung:** Außerhalb des Retain-Speichers: Variable <variable name>, <byte size> Bytes
- Mögliche Fehlerursache:** Es wird mehr Retain-Speicher verwendet als auf der SPS verfügbar ist.
Alternativ könnte der Retain-Speicher durch inkrementelles Übersetzen zu stark fragmentiert sein.
- Fehlerbehebung:** Verwenden Sie bei Fragmentierung des Speichers den Befehl "Bereinigen".
Damit können Sie die Reallozierung aller Daten beim nächsten Übersetzen erzwingen.

17.11.64 Compilerfehler C0104

- Meldung:** 'Nicht genügend globaler Datenspeicher: Variable <variable name>, <byte size> Bytes
- Mögliche Fehlerursache:** Es wird mehr Speicher für Daten verwendet als auf der SPS verfügbar ist.
Alternativ könnte der Speicher durch inkrementelles Übersetzen zu stark fragmentiert sein.
- Fehlerbehebung:** Verwenden Sie bei Fragmentierung des Speichers den Befehl "Bereinigen".
Damit können Sie die Reallozierung aller Daten beim nächsten Übersetzen erzwingen.

17.11.65 Compilerfehler C0114

- Meldung:** Ungültiges Ziel <jump label> für JMP
- Mögliche Fehlerursache:** Syntax- oder Tippfehler im JMP-Ziel
- Fehlerbehebung:** Korrigieren Sie den Tipp- oder Syntaxfehler
Beispiel für den Fehler:

```
PROGRAM PLC_PRG  
VAR  
END_VAR  
JMP 0;
```

--> C0114: Invalid destination 0 for 'JMP'

17.11.66 Compilerfehler C0116

- Meldung:** Marke <jump label> mehrfach definiert
- Mögliche Fehlerursache:** Eine Sprungmarke wird mehrfach definiert
- Fehlerbehebung:** Definieren Sie jede Sprungmarke nur einmal
Beispiel für den Fehler:

```
PROGRAM PLC_PRG  
VAR  
END_VAR
```

```
JMP label;
label:

label:

--> C0116: The label 'LABEL' is a duplicate
```

17.11.67 Compilerfehler C0117

- Meldung:** Marke <jump label> innerhalb des Bereichs der JMP-Anweisung nicht definiert
- Mögliche Fehlerursache:** Es wird zu einer Sprungmarke gesprungen, die nicht existiert
- Fehlerbehebung:** Definieren Sie die Sprungmarke, die Sie als Ziel angeben
- Beispiel für den Fehler:*

```
PROGRAM PLC_PRG
VAR
END_VAR
JMP A;

--> C0117: No such label 'A' within the scope of the 'JMP' statement
```

17.11.68 Compilerfehler C0118

- Meldung:** Die Sprungmarke <jump label> wird nicht referenziert
- Mögliche Fehlerursache:** Es ist eine Sprungmarke definiert, die nicht referenziert wird
- Fehlerbehebung:** Entfernen Sie die ungenutzten Sprungmarken
- Beispiel für den Fehler:*

```
PROGRAM PLC_PRG
VAR
END_VAR
LABEL:

--> C0118: The label ' LABEL' has not been referenced
```

17.11.69 Compilerfehler C0119

- Meldung:** Eine FB_init-Methode eines Funktionsbausteins oder einer Struktur benötigt zwei Eingänge bInitRetains und bInCopyCode vom Typ BOOL
- Mögliche Fehlerursache:** Von den folgenden booleschen Eingängen fehlt einer oder beide:
- bInitRetains
 - bInCopyCode
- Fehlerbehebung:** Definieren Sie die fehlenden Eingänge
- Beispiel für den Fehler:*

```
PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR

FUNCTION_BLOCK FB
METHOD FB_init
VAR_INPUT
END_VAR

--> C0119: An 'FB_init'-Method of a functionblock or struct needs two inputs
--> 'bInitRetains' and 'bInCopyCode' of type BOOL
```

Fehlerbehebung:

```

METHOD FB_init
VAR_INPUT
    bInitRetains : BOOL;
    bInCopyCode : BOOL;
END_VAR

```

17.11.70 Compilerfehler C0120

- Meldung:** Eine FB_Exit-Methode eines Funktionsbausteins oder Struktur benötigt einen Eingang bInCopyCode vom Typ BOOL
- Mögliche Fehlerursache:** Der Eingang bInCopyCode vom Typ BOOL fehlt
- Fehlerbehebung:** Definieren Sie den Eingang
- Beispiel für den Fehler:*
-

```

PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR

```

```

FUNCTION BLOCK FB
METHOD FB_exit
VAR_INPUT
END_VAR

```

```

--> C0120: An 'FB_Exit'-Method of a functionblock or struct needs an input
--> 'bInCopyCode' of type BOOL.

```

Fehlerbehebung:

```

METHOD FB_exit
VAR_INPUT
    bInCopyCode : BOOL;
END_VAR

```

17.11.71 Compilerfehler C0122

- Meldung:** Ausdruck SUPER^ ist in diesem Zusammenhang nicht zulässig
- Mögliche Fehlerursache:** SUPER^ wird außerhalb von abgeleiteten Funktionsbausteinen verwendet
- Fehlerbehebung:** Verwenden Sie SUPER^ nur in Funktionsbausteinen
- Beispiel für den Fehler:*
-

```

PROGRAM PLC_PRG
VAR
END_VAR

SUPER^.METH(TRUE, TRUE);

```

```

--> C0122: Expression 'SUPER' is not allowed in this context

```

17.11.72 Compilerfehler C0124

- Meldung:** "Initialisierung" ist keine gültige Initialisierung für eine Enumeration
- Mögliche Fehlerursache:** Für die Enum-Initialisierung wird ein Datentyp verwendet, der kein ANY_INT ist
- Fehlerbehebung:** Verwenden Sie nur ANY_INT für Enum-Initialisierungen
- Beispiel für den Fehler:*
-

```

PROGRAM PLC_PRG

```

Referenz Programmierung

```

VAR
    inst : DUT;
END_VAR

TYPE DUT :
(
    enum_member := 1.5
) DWORD;
END_TYPE

```

```

--> C0032: Cannot convert type 'LREAL' to type 'DUT'
--> C0124: 'Initialization' is no valid initialization for an enumeration

```

17.11.73 Compilerfehler C0125

Meldung: Die Konstante <constant value> ist mehr als einer Enumeration zugewiesen

Mögliche Fehlerursache: Derselbe Wert wird zwei oder mehr Enumerationen zugewiesen

Fehlerbehebung: Weisen Sie den Enumerationen verschiedene Werte zu

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    inst : DUT;
END_VAR

TYPE DUT :
(
    enum_member := 0,
    enum_member2 := 0
);
END_TYPE

```

```

--> C0125: The constant 0 is assigned to more than one enumeration

```

17.11.74 Compilerfehler C0126

Meldung: Variable vom Typ <data type> benötigt genau einen Index

Mögliche Fehlerursache: Einer Variablen mit einem Index werden mehrere Indizes zugewiesen

Fehlerbehebung: Weisen Sie nur einen Index zu

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    pi : POINTER TO INT;
END_VAR
pi[0,1] := 0;

```

```

--> C0126: Variable of type 'POINTER TO INT' requires exactly 1 Index

```

Fehlerbehebung:

```

Example:
pi[0] := 0;

```

17.11.75 Compilerfehler C0130

Meldung: <object> '<object name>' ohne Klammern () referenziert

Mögliche Fehlerursache: Eine Methode wird ohne Klammern referenziert

Fehlerbehebung: Referenzieren Sie Methoden immer mit Klammern

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR
inst.METH1
```

--> C0130: METHOD 'METH1' referenced without parentheses '()'

Fehlerbehebung:

Example:
inst.METH1();

17.11.76 Compilerfehler C0131

Meldung: <value> ist als Operand für ADR nicht erlaubt

Mögliche Fehlerursache: Dem Operator ADR wird eine Konstante als Operand gegeben

Fehlerbehebung: Verwenden Sie nur gültige Operanden für ADR

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : INT;
    pt : POINTER TO INT;
END_VAR

pt := ADR(1);
```

--> C0131: '1' is not allowed as operand for 'ADR'

Fehlerbehebung:

Example:
pt := ADR(i);

17.11.77 Compilerfehler C0132

Meldung: Keine umschließende Schleifenanweisung auf die EXIT angewendet werden kann

Mögliche Fehlerursache: EXIT wird außerhalb einer Schleife verwendet

Fehlerbehebung: Verwenden Sie EXIT nur innerhalb von Schleifen

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
END_VAR

EXIT
;
```

--> C0132: No enclosing loop of which to EXIT

17.11.78 Compilerfehler C0136

Meldung: Nicht eindeutige Verwendung des Namens <variable name>

Mögliche Fehlerursache: Eine Variable ist in mehreren globalen Variablenlisten (GVL) deklariert

Fehlerbehebung: Qualifizieren Sie die Variable mit der gewünschten GVL

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    j : INT := g_i;
END_VAR

GVL1:
VAR_GLOBAL
    g_i : INT;
END_VAR

GVL2:
VAR_GLOBAL
    g_i : INT;
END_VAR

--> C0136: ambiguous use of name 'g_i'

```

Fehlerbehebung:

Example:

```

j : INT := GVL1.g_i;

```

17.11.79 Compilerfehler C0139

Meldung: Code <code> hat keinen Effekt. Ist dies beabsichtigt?

Mögliche Fehlerursache: Der geschriebene Code ist syntaktisch korrekt, aber erfüllt keinen Zweck

Fehlerbehebung: Schreiben Sie Code, der einen Zweck erfüllt

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    i : INT;
END_VAR

i;

--> C0139: The code 'i;' has no effect. Is this the intent?

```

17.11.80 Compilerfehler C0140

Meldung: Referenzzuweisung ist nur für Variablen von Referenztypen erlaubt

Mögliche Fehlerursache: Es wird versucht, einer nicht als Referenztyp definierten Variable einen Referenzwert zuzuweisen

Fehlerbehebung: Definieren Sie die Variable als Referenztyp

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    i : INT;
    I_r : INT;
END_VAR

I_r REF= i;

--> C0140: Reference assign is only allowed to variables of Reference type

```

Fehlerbehebung:

Example:

```

I_r : REFERENCE TO INT;

```


17.11.81 Compilerfehler C0141

- Meldung:** Referenzzuweisung benötigt eine Variable mit Schreibzugriff
- Mögliche Fehlerursache:** Der Referenzzuweisung wird eine Konstante zugewiesen
- Fehlerbehebung:** Weisen Sie eine beschreibbare Variable zu

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : INT;
    I_r : REFERENCE TO INT;
END_VAR

I_r REF= 314;

--> C0141: Reference assign needs variable with write access
```

Fehlerbehebung:

Example:
I_r REF= i;

17.11.82 Compilerfehler C0142

- Meldung:** Eine lokale Variable mit dem Namen <variable name> ist bereits in <pou name> definiert
- Mögliche Fehlerursache:** Der gleiche Variablenname wird zweimal verwendet
- Fehlerbehebung:** Verwenden Sie unterschiedliche Variablennamen

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : INT;
    i : INT;
END_VAR

--> C0142: A local variable named 'i' is already defined in 'PLC_PRG'
```

17.11.83 Compilerfehler C0143

- Meldung:** Die Eigenschaft <property name> kann in diesem Zusammenhang nicht verwendet werden, da sie keine Get-Zugriffsdefinition besitzt
- Mögliche Fehlerursache:** Der Eigenschaft fehlt der Get-Zugriff
- Fehlerbehebung:** Achten Sie darauf, dass die Eigenschaft eine Get-Zugriffsdefinition besitzt

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : INT;
    inst: FB;
END_VAR

i := inst.Prop;

FUNCTION_BLOCK FB
VAR
END_VAR

PROPERTY Prop : INT
Set;

--> C0143: The property 'Prop' cannot be used in this context because it lacks the get accessor
```

17.11.84 Compilerfehler C0144

Meldung: Vererbung ist nur in Funktionsbausteinen, Schnittstellen und Strukturen erlaubt

Mögliche Fehlerursache: Es wird versucht, Vererbung in einem Objekt zu verwenden, das keine Vererbung erlaubt

Fehlerbehebung: Verwenden Sie `EXTENDS` nur in Funktionsbausteinen, Schnittstellen und Strukturen

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    inst : DUT_1;
END_VAR

TYPE DUT:
(
    enum_member := 0
);
END_TYPE

TYPE DUT_1 EXTENDS DUT:
(
    enum_memberX := 0
);
END_TYPE
```

--> C0144: Inheritance only allowed in Functionblocks, Interfaces and Structures

17.11.85 Compilerfehler C0145

Meldung: Schnittstellen können nur von Funktionsbausteinen implementiert werden

Mögliche Fehlerursache: Es wird versucht, eine Schnittstelle außerhalb eines Funktionsbausteins zu implementieren

Fehlerbehebung: Implementieren Sie Schnittstellen nur in Funktionsbausteinen

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
END_VAR

POU();

FUNCTION POU IMPLEMENTS ITF
VAR
END_VAR
```

--> C0145: Interfaces can only be implemented by Functionblocks

17.11.86 Compilerfehler C0149

Meldung: Variablendeklarationen in Schnittstellendefinitionen nicht erlaubt

Mögliche Fehlerursache: Es wird versucht, in einer Schnittstelle eine Variable zu definieren

Fehlerbehebung: Definieren Sie keine Variablen in Schnittstellen

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    inst : ITF;
END_VAR

INTERFACE ITF
VAR_INPUT
    i : INT;
END_VAR
```

--> C0149: Variable declarations are not allowed in interfaces

17.11.87 Compilerfehler C0161

- Meldung:** Datenfeldbereichsgrenze <array bound> ist kein konstanter Wert
- Mögliche Fehlerursache:** Als Arraygrenze ist eine Variable angegeben
- Fehlerbehebung:** Verwenden Sie Konstanten für die Arraygrenzen
- Beispiel für den Fehler:*

```
PROGRAM PLC_PRG
VAR
  i : INT := 3;
  arr1 : ARRAY[1..i] OF INT;
END_VAR
```

--> C0161: Border 'i' of array is no constant value

Fehlerbehebung:

Example:
arr1 : ARRAY[1..3] OF INT;

17.11.88 Compilerfehler C0162

- Meldung:** Nummer <number of array values> der Arrayinitialisierung ist kein konstanter Wert
- Mögliche Fehlerursache:** Die Initialisierung [value1,number value2(value2)] funktioniert nur mit einer Konstante für number value2
- Fehlerbehebung:** Verwenden Sie nur Konstanten
- Beispiel für den Fehler:*

```
PROGRAM PLC_PRG
VAR
  i : INT := 3;
  arr1 : ARRAY[1..4] OF INT := [1,i(7)];
END_VAR
```

--> C0162: Number 'i' of array initialisations is no constant value

Fehlerbehebung:

Example:
arr1 : ARRAY[1..4] OF INT := [1,3(7)];

17.11.89 Compilerfehler C0168

- Meldung:** VAR_CONFIG-Deklaration ist nur in VAR_CONFIG-Liste erlaubt
- Mögliche Fehlerursache:** VAR_CONFIG wird außerhalb einer VAR_CONFIG-Liste verwendet
- Fehlerbehebung:** Verwenden Sie VAR_CONFIG nur in VAR_CONFIG-Listen
- Beispiel für den Fehler:*

```
PROGRAM PLC_PRG
VAR_CONFIG
  i : INT;
END_VAR
```

--> C0168: 'VAR_CONFIG' declaration only allowed in VAR_CONFIG - list

17.11.90 Compilerfehler C0169

- Meldung:** VAR_GLOBAL-Deklaration ist nur in globalen Variablenlisten erlaubt
- Mögliche Fehlerursache:** VAR_GLOBAL wird außerhalb von globalen Variablenlisten verwendet
- Fehlerbehebung:** Verwenden Sie VAR_GLOBAL nur in globalen Variablenlisten

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR_GLOBAL
    i : INT;
END_VAR
```

```
--> C0169: 'VAR_GLOBAL' declaration only allowed in Global variable list
```

17.11.91 Compilerfehler C0174

- Meldung:** VAR_TEMP-Deklaration an dieser Stelle nicht erlaubt
- Mögliche Fehlerursache:** VAR_TEMP wird außerhalb eines Programms oder Funktionsbausteins verwendet
- Fehlerbehebung:** Verwenden Sie VAR_TEMP nur innerhalb von Programmen und Funktionsbausteinen

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
END_VAR

FUN();

FUNCTION FUN
VAR_TEMP
END_VAR
```

```
--> C0174: 'VAR_TEMP' declaration not allowed in this place
```

17.11.92 Compilerfehler C0175

- Meldung:** RETAIN oder PERSISTENT sind an dieser Stelle nicht erlaubt
- Mögliche Fehlerursache:** RETAIN oder PERSISTENT wird in einer Funktion verwendet
- Fehlerbehebung:** RETAIN oder PERSISTENT an den dafür vorgesehenen Stellen verwenden

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
END_VAR

POU_1();

FUNCTION POU_1
VAR RETAIN
END_VAR
```

```
--> C0175: 'RETAIN' or 'PERSISTENT' not allowed in this place
```

17.11.93 Compilerfehler C0177

- Meldung:** <object> ist vom Typ type und kann nicht instanziiert werden
- Mögliche Fehlerursache:** Es wird versucht, eine Funktion zu instanziiieren
- Fehlerbehebung:** Instanziiieren Sie nur Objekte, die instanziiert werden können

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    inst : POU;
END_VAR

FUNCTION POU
VAR
END_VAR

--> C0177: 'POU' is of type 'FUNCTION' and cannot be instantiated

```

17.11.94 Compilerfehler C0178

Meldung: Externer Zugriff auf VAR_IN_OUT-Parameter <parameter name> von <object name>

Mögliche Fehlerursache: Es wird versucht, von extern auf einen VAR_IN_OUT-Parameter zuzugreifen

Fehlerbehebung: Greifen Sie nicht von extern auf VAR_IN_OUT-Parameter zu

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    inst : FB;
    i : INT;
END_VAR
i := inst.in_out;

FUNCTION BLOCK FB
VAR_IN_OUT
    in_out : INT;
END_VAR

--> C0178: No external access to 'VAR_IN_OUT' parameter 'in_out' of 'FB'

```

17.11.95 Compilerfehler C0182

Meldung: Rückgabotyp ist nur für Funktionen und Methoden erlaubt

Mögliche Fehlerursache: Es wird versucht, in einem Programm einen Rückgabotyp zu definieren

Fehlerbehebung: Definieren Sie einen Rückgabotyp nur in Methoden und Funktionen

Beispiel für den Fehler:

```

PROGRAM PLC_PRG : BOOL
VAR
END_VAR

--> C0182: Return type is only possible for POU's of Type FUNCTION and METHOD

```

17.11.96 Compilerfehler C0185

Meldung: Es ist nicht möglich, auf das Ergebnis eines Funktionsaufrufs einen Komponentenzugriff <.>, einen Indexzugriff <[]> oder einen Aufruf <()> des Ergebnisses eines Funktionsaufrufs durchzuführen.

Weisen Sie zuerst das Ergebnis einer Variablen zu.

Mögliche Fehlerursache: Es wird ein Komponenten- oder Indexzugriff auf das Ergebnis eines Funktionsaufrufs durchgeführt

Fehlerbehebung: Weisen Sie für den Zugriff das Ergebnis einer Variablen zu

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    it : ITF;
END_VAR

```

```
POU_1() [0].METH1();

FUNCTION POU_1 : ARRAY[0..0] OF ITF

INTERFACE ITF

METHOD METH1

--> C0185: It is not possible to perform component access '.', index access '[]'
--> or call '()' on result of function call.
--> Assign result to help variable first.
```

17.11.97 Compilerfehler C0189

Meldung: <;> anstatt <token> erwartet

Mögliche Fehlerursache: Syntaxfehler

Fehlerbehebung: Achten Sie auf eine korrekte Syntax

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    INT
END_VAR

--> C0009: Unexpected Token '<Token>' found
--> C0189: ';' expected instead of 'INT'
```

17.11.98 Compilerfehler C0190

Meldung: <;> anstatt Ende der POU erwartet

Mögliche Fehlerursache: Syntaxfehler in POU

Fehlerbehebung: Achten Sie auf eine korrekte Syntax

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : INT;
END_VAR
i := 5

--> C0190: ';' expected instead of end of POU
```

17.11.99 Compilerfehler C0191

Meldung: Der Operator INDEXOF wird nicht mehr unterstützt. Benützen Sie stattdessen ADR.

Wird ADR auf einen POU-Namen angewendet, so wird ein Zeiger auf den POU-Code zurückgeliefert.

Mögliche Fehlerursache: Es wird der veralteten Operator INDEXOF verwendet

Fehlerbehebung: Verwenden Sie den Operator ADR

17.11.100 Compilerfehler C0195

Meldung: Implizite Konvertierung von vorzeichenbehaftetem Datentyp <data type 1> nach nicht vorzeichenbehaftetem Datentyp <data type 2>: Möglicherweise Änderung des Vorzeichens

Mögliche Fehlerursache: Bei der impliziten Konvertierung wird ein Vorzeichenkonflikt nicht berücksichtigt

Fehlerbehebung: Konvertieren Sie nur Datentypen mit dem gleichen Vorzeichen implizit

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    i : INT;
    b : UINT;
END_VAR

b := i;

--> C0195: Implicit conversion from signed Type 'INT' to unsigned Type 'UINT'
--> : possible change of sign

```

17.11.101 Compilerfehler C0196

- Meldung:** Implizite Konvertierung von nicht vorzeichenbehaftetem Datentyp <data type 1> nach vorzeichenbehaftetem Datentyp <data type 2> : Möglicherweise Änderung des Vorzeichens
- Mögliche Fehlerursache:** Bei der impliziten Konvertierung wird ein Vorzeichenkonflikt eventuell nicht berücksichtigt
- Fehlerbehebung:** Verwenden Sie explizite Konvertierungen
- Beispiel für den Fehler:*
-

```

PROGRAM PLC_PRG
VAR
    i : INT;
    b : UINT;
END_VAR

i := b;

--> C0196: Implicit conversion from unsigned Type 'UINT' to signed type 'INT'
--> : possible change of sign

```

17.11.102 Compilerfehler C0197

- Meldung:** Implizite Konvertierung von <data type 1> nach <data type 2> : Möglicher Datenverlust
- Mögliche Fehlerursache:** Es wird versucht, eine Variable vom Datentyp DINT oder LINT in den Datentyp REAL implizit zu konvertieren
- Fehlerbehebung:** Verwenden Sie für DINT den Datentyp LREAL und achten Sie bei der Konvertierung von LINT auf LREAL, dass der Wert des LINT nicht die Kapazität des LREAL übersteigt
- Beispiel für den Fehler:*
-

```

PROGRAM PLC_PRG
VAR
    i : DINT;
    b : REAL;
END_VAR

b := i;

--> C0197: Implicit conversion from 'DINT' to 'REAL': possible loss of information

```

17.11.103 Compilerfehler C0198

- Meldung:** Stringkonstante <string value> ist zu lang für Zieltyp <data type>
- Mögliche Fehlerursache:** Stringkonstante hat zu viele Zeichen
- Fehlerbehebung:** Verwenden Sie kürzere Stringkonstanten oder deklarieren Sie größere Strings

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    str : STRING(4) := '12345';
END_VAR

--> C0198: String constant '12345' too long for destination type 'STRING(4)'
```

17.11.104 Compilerfehler C0199

Meldung: Schnittstelle <interface name> muss instanziiert werden, um darauf zuzugreifen zu können

Mögliche Fehlerursache: Es wird versucht, auf eine Schnittstellenmethode zuzugreifen, ohne dass die Schnittstelle instanziiert wurde

Fehlerbehebung: Instanziiieren Sie die Schnittstelle

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
END_VAR

ITF.METH();

INTERFACE ITF

METHOD METH
VAR_INPUT
END_VAR

--> C0199: Interface 'ITF' must be instantiated to be accessed
```

Fehlerbehebung:

Example:
itest: ITF;

17.11.105 Compilerfehler C0201

Meldung: Typ <data type 1> entspricht nicht dem Typ <data type 2> von VAR_IN_OUT-Variable

Mögliche Fehlerursache: Der Datentyp, welcher der Funktion als VAR_IN_OUT-Parameter übergeben wird, stimmt nicht mit dem in ihr definierten Datentyp überein

Fehlerbehebung: Übergeben Sie eine Variable mit dem korrekten Datentyp

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    Inst: POU;
    b : BOOL;
END_VAR

inst(in_out := b);

FUNCTION_BLOCK POU
VAR_IN_OUT
    in_out : INT;
END_VAR

--> C0201: Type 'BOOL' is not equal to type 'INT' of VAR_IN_OUT 'Variable'
```

17.11.106 Compilerfehler C0203

Meldung: Nur Strukturen und Funktionsbausteine können Variablen vom Typ BIT beinhalten

Mögliche Fehlerursache: Es wird versucht, außerhalb von Strukturen und Funktionsbausteinen eine Variable vom Typ `BIT` zu deklarieren

Fehlerbehebung: Deklarieren Sie Variablen vom Typ `BIT` nur in Strukturen und Funktionsbausteinen

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    b : BIT;
END_VAR
```

--> C0203: Only Structures and Function Blocks can contain variables of type BIT.

17.11.107 Compilerfehler C0204

Meldung: Variablen vom Typ `BIT` müssen innerhalb eines `VAR_INPUT`-, `VAR_OUTPUT` oder `VAR`-Blocks definiert werden

Mögliche Fehlerursache: Es wird versucht, eine Variable vom Typ `BIT` oder als `VAR_IN_OUT`-Parameter zu definieren

Fehlerbehebung: Definieren Sie Variablen vom Typ `BIT` nur innerhalb eines `VAR_INPUT`-, `VAR_OUTPUT` oder `VAR`-Blocks

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    inst : FB;
END_VAR
```

```
FUNCTION BLOCK FB
VAR_IN_OUT
    b : BIT;
END_VAR
```

--> C0204: Variables of type BIT must be declared within a VAR_INPUT-,
--> VAR_OUTPUT or VAR-block

17.11.108 Compilerfehler C0205

Meldung: `POINTER TO BIT` ist nicht erlaubt

Mögliche Fehlerursache: Es wird versucht, einen `POINTER TO BIT` zu deklarieren

Fehlerbehebung: Deklarieren Sie keinen `POINTER TO BIT`

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    pt : POINTER TO BIT;
END_VAR
```

--> C0205: POINTER TO BIT is not allowed

17.11.109 Compilerfehler C0206

Meldung: `BIT` ist nicht als Basistyp eines Arrays erlaubt

Mögliche Fehlerursache: Es wird versucht, ein `BIT`-Array zu deklarieren

Fehlerbehebung: Deklarieren Sie keine `BIT`-Arrays

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    arr : ARRAY[1..2] OF BIT;
```

```
END_VAR
```

```
--> C0206: BIT is not allowed as base type of an array
```

17.11.110 Compilerfehler C0208

Meldung: MOD ist für REAL nicht definiert

Mögliche Fehlerursache: Es wird versucht, eine Modulo-Operation mit einer Variable vom Typ REAL durchzuführen

Fehlerbehebung: Modulo-Operationen sind nur mit Variablen vom Typ ANY_INT möglich

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    r1 : REAL;
END_VAR
r1 := r1 MOD 2;
```

```
--> C0208: 'MOD' is not defined for 'REAL'
```

17.11.111 Compilerfehler C0211

Meldung: Variablendeklaration erwartet statt <expression>

Mögliche Fehlerursache: Syntaxfehler

Fehlerbehebung: Achten Sie auf eine korrekte Syntax

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    VAR
        END_VAR
END_VAR
```

```
--> C0211: Variable declaration expected instead of VAR END_VAR
```

17.11.112 Compilerfehler C0212

Meldung: VAR, VAR_INPUT, VAR_OUTPUT oder VAR_INOUT erwartet anstelle von <expression>

Mögliche Fehlerursache: Syntaxfehler

Fehlerbehebung: Achten Sie auf eine korrekte Syntax

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
i : INT;
```

```
--> C0212: VAR, VAR_INPUT, VAR_OUTPUT or VAR_INOUT expected instead of i : INT;
```

17.11.113 Compilerfehler C0216

Meldung: Doppelte CASE-Marke

Mögliche Fehlerursache: Eine CASE-Marke wird mehr als einmal verwendet

Fehlerbehebung: Verwenden Sie jede CASE-Marke nur einmal

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
    i : INT;
END_VAR
```

```

CASE i OF
  1: i := i+1;
  1: i := i+2;
ELSE
  i := i+10;
END_CASE;

--> C0216: Case label duplicate

```

17.11.114 Compilerfehler C0217

Meldung: Die CASE-Marke <case label> ist auch im Bereich <case range begin> ... <case range end> **enthalten**

Mögliche Fehlerursache: Eine CASE-Marke ist Teil des Bereichs einer anderen CASE-Marke

Fehlerbehebung: Achten Sie auf Überschneidungen

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
  i : INT;
END_VAR
CASE i OF
  3..5: i := i+2;
  4: i := i+2;
ELSE
  i := i+10;
END_CASE;

--> C0217: Case label 4 also contained in range 3 .. 5

```

17.11.115 Compilerfehler C0218

Meldung: CASE-Marke erfordert ein Literal oder eine symbolische Ganzzahlkonstante

Mögliche Fehlerursache: Es wird versucht, eine Variable als CASE-Marke zu verwenden

Fehlerbehebung: Verwenden Sie nur Literale und symbolische Ganzzahlkonstanten

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
  i : INT;
  a : INT := 2;
END_VAR

CASE i OF
  1: i := i+1;
  a: i := i+2;
ELSE
  i := i+10;
END_CASE;

--> C0218: Case label requires literal or symbolic integer constant

```

17.11.116 Compilerfehler C0219

Meldung: CASE enthält überlappende Bereiche <case range 1 begin> .. <case range 1 end> und <case range 2 begin> .. <case range 2 end>

Mögliche Fehlerursache: Zwei Bereiche von CASE-Marken haben gleiche Elemente oder Teilmengen

Fehlerbehebung: Achten Sie auf Überschneidungen

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
  i : INT;
END_VAR

```

Referenz Programmierung

```

CASE i OF
  3..5: i := i+2;
  1..4: i := i+2;
ELSE
  i := i+10;
END_CASE;

--> C0219: Case contains overlapping range 1 .. 4 and 3 .. 5

```

17.11.117 Compilerfehler C0221

Meldung: Direkte Adresse <address> nicht korrekt dargestellt

Mögliche Fehlerursache: Eine Adresse wird unvollständig dargestellt

Fehlerbehebung: Stellen Sie die Adresse korrekt dar

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
  xVar : BOOL;
END_VAR;
xVar := %IX0;

--> C0221: Direct Address '%IX0' malformed

```

Fehlerbehebung:

Example:
xVar := %IX0.2;

17.11.118 Compilerfehler C0222

Meldung: Ausgänge können nicht vom Typ REFERENCE TO sein

Mögliche Fehlerursache: Es wird versucht, REFERENCE TO als Ausgangsparameter zu definieren

Fehlerbehebung: Verwenden Sie REFERENCE TO nicht als Ausgangsparameter

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
  inst : FB;
END_VAR

FUNCTION_BLOCK FB
VAR_OUTPUT
  re : REFERENCE TO INT;
END_VAR

--> C0222: Outputs can't be of type 'REFERENCE TO'

```

17.11.119 Compilerfehler C0224

Meldung: Aufrufrekursion: <recursion>

Mögliche Fehlerursache: Eine Funktion ruft sich selbst auf

Fehlerbehebung: Achten Sie darauf, dass Funktionen nicht rekursiv sind

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
END_VAR

POU();

FUNCTION POU
VAR
END_VAR

```

```
POU();
```

```
--> C0224: Call Recursion: POU -> POU
```

17.11.120 Compilerfehler C0227

Meldung:	Konstante <constant name> nicht mit konstantem Wert initialisiert
Mögliche Fehlerursache:	Eine Konstante wird mit einer Variable initialisiert
Fehlerbehebung:	Initialisieren Sie Konstanten nur mit konstanten Werten
	<i>Beispiel für den Fehler:</i>

```
PROGRAM PLC_PRG
VAR
  i : INT;
END_VAR
VAR CONSTANT
  k : INT := i;
END_VAR
```

```
--> C0227: Initialisation of constant variable 'k' not constant
```

17.11.121 Compilerfehler C0228

Meldung:	Kein Initialwert für konstante Variable <constant name>
Mögliche Fehlerursache:	Eine Konstante wird nicht initialisiert
Fehlerbehebung:	Initialisieren Sie die Konstante
	<i>Beispiel für den Fehler:</i>

```
PROGRAM PLC_PRG
VAR
END_VAR
VAR CONSTANT
  k : INT;
END_VAR
```

```
--> C0228: No initial value for constant variable 'k'
```

Fehlerbehebung:

Example:
k : INT := 1;

17.11.122 Compilerfehler C0234

Meldung:	Erster Operand von __QueryInterface muss eine Schnittstellenreferenz oder die Instanz eines Funktionsbausteins sein
Mögliche Fehlerursache:	Dem Operator __QueryInterface werden falsche Operanden übergeben
Fehlerbehebung:	Übergeben Sie eine Schnittstellenreferenz oder die Instanz eines Funktionsbausteins
	<i>Beispiel für den Fehler:</i>

```
PROGRAM PLC_PRG
VAR
  a : INT;
  ITFref, ITFref2 : ITF;
  ITFref2 : ITF2;
END_VAR

__QueryInterface(a ,ITFref);

INTERFACE ITF EXTENDS __SYSTEM.IQueryInterface
INTERFACE ITF2 EXTENDS ITF
```

Referenz Programmierung

--> C0234: First Operand of __QueryInterface must be an interface,
 --> reference or the instance of a function block

Fehlerbehebung:

Example:

```
__QueryInterface(ITFref2, ITFref);
```

17.11.123 Compilerfehler C0235

Meldung: Der zweite Operand von __QueryInterface muss eine Interface-Referenz sein

Mögliche Fehlerursache: Dem Operator __QueryInterface werden falsche Operanden übergeben

Fehlerbehebung: Übergeben Sie eine Schnittstellenreferenz

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
  a : INT;
  ITFref, ITFref2 : ITF;
  ITFref2 : ITF2;
END_VAR

__QueryInterface(ITFref2, a);

INTERFACE ITF EXTENDS __SYSTEM.IQueryInterface
INTERFACE ITF2 EXTENDS ITF

--> C0235: Second Operand of __QueryInterface must be an interface reference
```

Fehlerbehebung:

Example:

```
__QueryInterface(ITFref2, ITFref);
```

17.11.124 Compilerfehler C0236

Meldung: Falsche Typdefinition für VAR_EXTERNAL <constant name>

Mögliche Fehlerursache: Die Variable wird in VAR_GLOBAL / VAR_EXTERNAL als unterschiedliche Typen deklariert

Fehlerbehebung: Verwenden Sie in VAR_GLOBAL und VAR_EXTERNAL die gleiche Typdefinition

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR_EXTERNAL
  ig : STRING;
END_VAR

VAR_GLOBAL
  ig : INT;
END_VAR

--> C0236: Wrong type definition for VAR_EXTERNAL ig
```

17.11.125 Compilerfehler C0237

Meldung: Keine globale Definition für VAR_EXTERNAL <variabler name> gefunden

Mögliche Fehlerursache: Es wird versucht, in VAR_EXTERNAL eine Variable zu deklarieren, die nicht in VAR_GLOBAL existiert

Fehlerbehebung: Achten Sie darauf, dass die Bezeichner übereinstimmen

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR_EXTERNAL
  i : INT;
END_VAR

VAR_GLOBAL
  ig : INT;
END_VAR

--> C0237: No global definition found for VAR_EXTERNAL 'i'

```

17.11.126 Compilerfehler C0238

- Meldung:** Für VAR_EXTERNAL <variabler name> ist kein Initialwert erlaubt
- Mögliche Fehlerursache:** Es wird versucht, in VAR_EXTERNAL eine Variable zu initialisieren
- Fehlerbehebung:** Initialisieren Sie keine Variablen in VAR_EXTERNAL

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR_EXTERNAL
  ig : INT := 2;
END_VAR

VAR_GLOBAL
  ig : INT;
END_VAR

--> C0238: No initial value allowed for VAR_EXTERNAL ig

```

17.11.127 Compilerfehler C0239

- Meldung:** Interface <interface name 1> erweitert nicht <interface name 2>
- Mögliche Fehlerursache:** Die verwendete Schnittstelle erweitert keine andere Schnittstelle
- Fehlerbehebung:** Erweitern Sie die Schnittstelle

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
  ITFref : ITF;
  ITFref2 : ITF2;
END_VAR

__QueryInterface(ITFref2, ITFref);

INTERFACE ITF
INTERFACE ITF2 EXTENDS ITF

--> C0239: Interface ITF__Union does not extend __System.IQueryInterface

```

Fehlerbehebung:

Example:

```

INTERFACE ITF EXTENDS __System.IQueryInterface

```

17.11.128 Compilerfehler C0240

- Meldung:** Erster Operand von __QueryPointer muss eine Schnittstellenreferenz oder Instanz eines Funktionsbausteins sein
- Mögliche Fehlerursache:** Dem Operator __QueryPointer werden falsche Operanden übergeben
- Fehlerbehebung:** Übergeben Sie eine Schnittstellenreferenz oder die Instanz eines Funktionsbausteins

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
  a : INT;
  ITFref : ITF;
  pt : POINTER TO FB;
END_VAR

__QueryPointer(a,pt);

--> C0240: First Operand of __QueryPointer must be an interface reference,
--> or the instance of a function block

```

Fehlerbehebung:

Example:

```
__QueryPointer (ITFref, pt);
```

17.11.129 Compilerfehler C0241

- Meldung:** Zweiter Operand von __QueryPointer muss ein Zeiger sein
- Mögliche Fehlerursache:** Dem Operator __QueryPointer werden falsche Operanden übergeben
- Fehlerbehebung:** Übergeben Sie einen Pointer

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
  b : INT;
  ITFref : ITF;
  pt : POINTER TO FB;
END_VAR

__QueryPointer(ITFref,b);

INTERFACE ITF EXTENDS __System.IQueryInterface

--> C0241: Second Operand of __QueryPointer must be pointer

```

Fehlerbehebung:

Example:

```
__QueryPointer (ITFref, pt);
```

17.11.130 Compilerfehler C0242

- Meldung:** Operand von __DELETE muss ein Pointer sein
- Mögliche Fehlerursache:** Dem Operator __DELETE wird ein falscher Operand übergeben
- Fehlerbehebung:** Übergeben Sie einen Pointer

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
  a : INT;
  pt : POINTER TO INT;
END_VAR

__DELETE(a);

--> C0242: Operand of __DELETE must be pointer

```

Fehlerbehebung:

Example:

```
__DELETE (pt);
```

17.11.131 Compilerfehler C0243

Meldung: Der in der Schnittstelle verwendete Name ist nicht identisch mit dem Objekt-namen

Mögliche Fehlerursache: Der Objektname und der Name, der im Code verwendet wird, unterscheiden sich

Fehlerbehebung: Achten Sie auf eine Gleichheit der Namen

Beispiel für den Fehler:

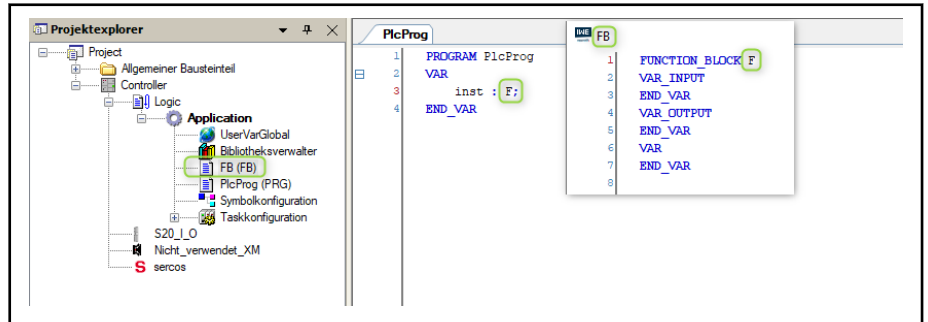


Abb. 17-92:

17.11.132 Compilerfehler C0380

Meldung: Die Operatoren LOWER_BOUND und UPPER_BOUND werden nur für Arrays mit variabler Länge unterstützt

Mögliche Fehlerursache: Einer der beiden Operatoren LOWER_BOUND oder UPPER_BOUND wird nicht für ein Array mit variabler Länge verwendet

Fehlerbehebung: Verwenden Sie die Operatoren LOWER_BOUND oder UPPER_BOUND nur für ein Array mit variabler Länge

Beispiel für den Fehler:

```
FUNCTION BLOCK POU
VAR_IN_OUT
  arrin : ARRAY [*] OF INT;
END_VAR
VAR
  arrtest : ARRAY [0..5] OF INT;
  test1: DINT;
  test2: DINT;
END_VAR

test1 := UPPER_BOUND(arrin, 0);
test2 := UPPER_BOUND(arrtest, 0);

--> C0380: The operators LOWER_BOUND and UPPER_BOUND
--> are supported only for arrays with variable length.
```

17.11.133 Compilerfehler C0509

Meldung: Mehrfachzuweisungen sind für Operator __New nicht erlaubt

Mögliche Fehlerursache: Innerhalb einer Codezeile wird mehrfach der Zuweisungsoperator := zusammen mit dem __New-Operator aufgerufen

Fehlerbehebung: Programmieren Sie für jeden Pointer, der auf einen dynamisch allozierten Speicher zeigt, die Speicherallozierung mit dem __New-Operators in einer separaten Codezeile

Beispiel für den Fehler:

```
PROGRAM PLC_PRG
VAR
  pbAlpha : POINTER TO BYTE; // Typed pointer to Alpha
  pbBeta: POINTER TO BYTE;   // Typed pointer to Beta
  xInit : BOOL := TRUE;
```

Referenz Programmierung

```

    xDelete : BOOL;
END_VAR

IF (xInit) THEN
    pbBeta := pbAlpha := __NEW(BYTE); // Incorrect code for memory allocation
END_IF

pbBeta := pbAlpha := 16#01;

IF (xDelete) THEN
    __DELETE(pbAlpha); // Frees memory of pointer
END_IF

--> C0509: Multiple assignments for operator '__New' not allowed

```

Fehlerbehebung:

```

PROGRAM PLC_PRG
VAR
    pbAlpha : POINTER TO BYTE; // Pointer to Alpha
    pbBeta:  POINTER TO BYTE;   // Pointer to Beta
    xInit : BOOL := TRUE;
    xDelete : BOOL;
END_VAR

IF (xInit) THEN
    pbAlpha := __NEW(BYTE); // Allocates memory for Alpha
    pbBeta := __NEW(BYTE); // Allocates memory for Beta
END_IF

pbBeta := pbAlpha := 16#01; // Multiple assignment

IF (xDelete) THEN
    __DELETE(pbAlpha); // Frees memory of pointer
END_IF

```

17.11.134 Compilerfehler C0511

Meldung: Der Funktionsbaustein <function block name> ist ABSTRACT und er kann nicht als Ziel für eine Zuweisung verwendet werden

Mögliche Fehlerursache: Einem abstrakten Funktionsbaustein wurde ein Wert zugewiesen. Die konkreten Funktionsbausteine haben eventuell verschiedene Typen und können deswegen nicht kopiert werden.

Fehlerbehebung: Um die Daten des Funktionsbausteins zu kopieren, müssen konkrete Funktionsbausteine verwendet werden

Beispiel für den Fehler:

```

PROGRAM PLC_PRG
VAR
    refAbstract1 : REFERENCE TO AbstractPOU;
    refAbstract2 : REFERENCE TO AbstractPOU;
END_VAR

refAbstract1 := refAbstract2;
--> C0511: The function block 'refAbstract1' is ABSTRACT and
--> cannot be used as a target for an assignment.

```

Fehlerbehebung: Verwenden Sie die Referenzzuweisung REF=, um die Referenz refAbstract1 auf den selben Funktionsbaustein wie refAbstract2 verweisen zu lassen

18 Referenz Benutzeroberfläche

18.1 Objekte

Objekte in IndraLogic 2G stellen bestimmte Funktionalitäten für die Erstellung Ihrer Applikation zur Verfügung.

Beispiele:

Applikation, Programm, Funktion, Bibliotheksverwalter, Gerät, Bildersammlung. Objekte werden im Projektextplorer verwaltet.

Sie können ein Objekt mit dem Befehl **Hinzufügen** in den Projektextplorer einfügen. Die Einfügemöglichkeiten hängen von der Position im Explorer ab.

Jedes Objekt hat Eigenschaften, die Sie über das Kontextmenü des Objekts ansehen und bearbeiten können.

Siehe auch

- [Kap. 2.2 "Objekte hinzufügen" auf Seite 8](#)
- [Kap. "Befehl 'Eigenschaften'" auf Seite 634](#)

18.1.1 Objekt 'Allgemeiner Bausteinteil'

Symbol



Aufruf

Im Projektextplorer, unterhalb des Projektknotens

Funktion

Der Ordner "Allgemeiner Bausteinteil" verwaltet die Projekteinstellungen, siehe [Kap. 18.1.19 "Objekt 'Projekteinstellungen'" auf Seite 579](#).

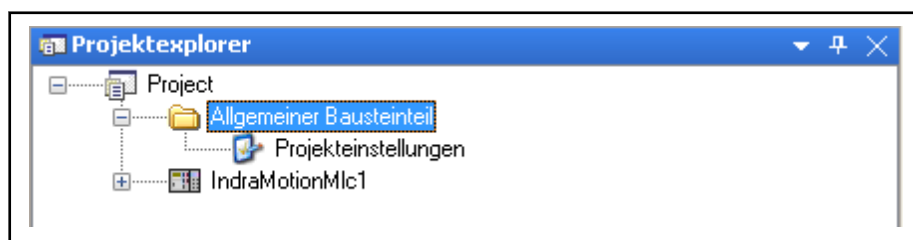


Abb. 18-1: Objekt "Allgemeiner Bausteinteil" im Projektextplorer

Zusätzlich können dem Ordner projektglobale Objekte und Instanzen über das Kontextmenü hinzugefügt werden. Die Verwendung projektglobaler Instanzen dient der objektorientierten Programmierung, siehe [Kap. 9.18 "Objektorientiert programmieren" auf Seite 154](#)

Projektglobale Objekte und Instanzen

Bibliotheksinfo...	i: Information über die Eigenschaften, Metainformationen und Informationen des Projekts, siehe Kap. 18.1.20 "Objekt 'Bibliotheksinfo'" auf Seite 579 Hinweis: Das Objekt Bibliotheksinfo kann nach dem Einfügen in das Projekt nicht mehr entfernt werden! Es ist nur eine Bibliotheksinfo im Projekt möglich.
Bibliotheksverwalter...	📁: Übersicht aller Bibliotheken, die zur Applikationserstellung im Projekt eingebunden wurden, siehe Kap. 18.1.14 "Objekt 'Bibliotheksverwalter'" auf Seite 548 Hinweis: Es ist nur ein Bibliotheksverwalter im Projekt möglich!

Bildersammlung...	 : Tabelle, in der Bilder ihren IDs zugeordnet sind, siehe Kap. 18.1.13 "Objekt 'Bildersammlung'" auf Seite 547
Datentypen...	 : Projektglobale Datentypen, siehe "Dialog 'DUT hinzufügen'" auf Seite 518
Globale Variablenliste...	 : Deklaration, Bearbeitung und Anzeige von globalen Variablen, siehe Kap. 18.1.11 "Objekt 'GVL'- Globale Variablenliste" auf Seite 544
Ordner...	 : Dem Allgemeinen Bausteinteil können weitere Unterordner hinzugefügt werden, siehe Kap. "Befehl 'Ordner hinzufügen'" auf Seite 644
POU...	 : Projektglobale Programm-Organisationseinheiten (POU), siehe: Kap. 18.1.17 "Objekt 'POU'" auf Seite 552
Parameterliste...	 : Projektglobale Parameterliste, siehe Kap. 18.1.11 "Objekt 'GVL'- Globale Variablenliste" auf Seite 544
Schnittstelle...	 : Beschreibt ein Set von Methoden- und Eigenschaft-Prototypen innerhalb der objektorientierten Programmierung, siehe Kap. "Objekt 'Schnittstelle(INTERFACE)" auf Seite 562
Textliste...	 : Liste zur Erstellung, Verwaltung und Übersetzung von Texten, siehe Kap. 18.1.23 "Objekt 'Textliste'" auf Seite 588
Visualisierung...	 : IndraLogic Visualisierung, siehe Kap. 20.1 "Visualisierung im Überblick" auf Seite 803
POU Vorlagen	SPS-Überwachungsfunktionen, siehe Kap. "POU-Vorlagen" auf Seite 578

Tab. 18-1: Allgemeiner Bausteinteil - Unterstützte Objekte und Instanzen

18.1.2 Objekt 'Applikation'

Symbol: 

Das Objekt wird als Knoten im Projektextplorer dargestellt. Es umfasst die Objekte, die für ein lauffähiges Steuerungsprogramm benötigt werden.

Das Objekt Application wird beim Anlegen einer Steuerung automatisch erzeugt und befindet sich unterhalb des **Logic-Knotens** (2).

Unterhalb des Objektes Application werden weitere Objekte automatisch angelegt:

- **Taskkonfiguration** (3): Hier konfigurieren Sie, welches Programm der Applikation von welcher Task mit welchen Einstellungen aufgerufen wird.
- **Bibliotheksverwalter** (4): Im Abhängigkeit von Ihrer Steuerungshardware wird hier automatisch ein passender Satz an Bibliotheken angelegt. Weitere Bibliotheken können Sie hinzufügen.

Entsprechend der durch Sie gewählten Steuerungshardware werden weitere Objekte erzeugt und Ihnen zur Nutzung zur Verfügung gestellt.

Sie können unter der Applikation weitere Objekte Ihres Steuerungsprogramms wie beispielsweise POU's, globale Variablenlisten usw. hinzufügen. Diese Objekte sind dann nur für diese Applikation verfügbar.

Zusätzlich kann die Applikation aber auch Instanzen projektglobaler Objekte verwenden. Projektglobale Objekte verwalten Sie im Ordner **Allgemeiner Bausteinteil** (5). Die Verwendung solcher Instanzen entspricht der Denkweise der objektorientierten Programmierung.

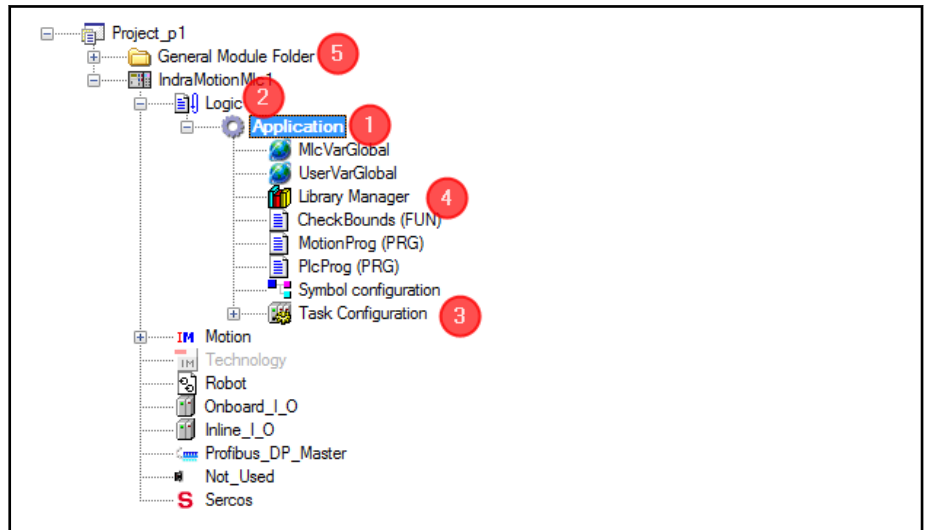


Abb. 18-2:

Die Applikation, mit der Sie im Online-Betrieb arbeiten wollen, muss als "aktive Applikation" gesetzt sein.

Sie können eine Applikation im Dialog **Eigenschaften** des Applikationsobjekts, in der Registerkarte **Optionen Applikationserzeugung**, mit speziellen Eigenschaften versehen.

Dieser Dialog ist über das Kontextmenü des Applikationsobjektes erreichbar.

Beispiel: Aktivierung dynamischer Speicherallozierung.

Beim Download der Applikation auf die SPS können Sie Informationen zum Applikationsinhalt mitgeben.

Dies ist ebenfalls eine Einstellung in Registerkarte **Optionen Applikationserzeugung**. Dann können Sie später die Applikation auf der Steuerung mit der aktiven Applikation in IndraLogic vergleichen.

Wenn Sie der Applikation individuelle Informationen zu Autor, Version und eine individuelle Kurzbeschreibung mitgeben möchten, können Sie dazu die Angaben aus den **Bibliotheksinformationen** im Dialog **Eigenschaften** in der Registerkarte **Information** modifizieren.

Wenn Sie mit einer Applikation auf das Zielgerät einloggen wollen (SPS oder Simulation), wird erst geprüft, welche Applikation aktuell auf der SPS liegt und ob die Applikationsparameter auf der Steuerung mit denen in der Projektkonfiguration übereinstimmen. Entsprechende Meldungen zeigen Nicht-übereinstimmungen und mögliche Wege des weiteren Vorgehens an.

Sie können auf diese Weise auch die Applikation auf der SPS löschen.

In der Registerkarte **Applikation** des Geräteeditors können Sie nachsehen, welche Applikationen gerade auf dem Gerät liegen. Dort können Sie auch die Applikation vom Zielsystem löschen.

Eventuell sehen Sie in der Liste zusätzlich Applikationen, die nicht durch ein separates Objekt im Projektextplorer vertreten sind, wie beispielsweise die <Applikation>_symbols.app, die eine für die Applikation erzeugte Symbolliste enthält (siehe "Symbolkonfiguration").

Siehe auch

- [Kap. 8.1 "Projektextplorer und Geräteeditor" auf Seite 71](#)
- [Kap. 18.1.25 "Objekt 'Taskkonfiguration'" auf Seite 598](#)
- [Kap. "Dialog 'Eigenschaften - 'Optionen Applikationserzeugung'" auf Seite 751](#)
- [Kap. 11 "Applikation auf die SPS übertragen" auf Seite 187](#)
- [Kap. "Registerkarte 'SPS-Einstellungen'" auf Seite 527](#)
- [Kap. 10.2 "Symbolkonfiguration" auf Seite 167](#)
- [Kap. 14.1 "Ausführen eines Online Changes" auf Seite 229](#)
- [Kap. "Befehl 'Bibliotheksinformationen'" auf Seite 719](#)

18.1.3 Objekt 'Datenquellenverwalter'

Symbol: 

Das Objekt dient als Knoten für darunter liegende Datenquellen und verwaltet diese. Dabei muss wenigstens eine Datenquelle vorhanden sein. Eine Applikation mit Datenquellenverwalter kommuniziert mit entfernten Geräten.

Siehe [Kap. 10.4.1 "Datenquellenverwalter Grundlagen" auf Seite 174](#)

Befehl 'Objekt hinzufügen' > 'Datenquelle'

Funktion: Der Befehl öffnet den Dialog "Datenquelle hinzufügen".

Aufruf:

- Den Datenquellenverwalter selektieren und das Kontextmenü öffnen
- **Hinzufügen ► Datenquelle** wählen

Dialog 'Datenquelle hinzufügen'

Name	Name der hinzuzufügenden Datenquelle
Datenquellentyp auswählen	Datenquellentyp, der zur Steuerungskonfigurationen passt, um eine Kommunikation aufbauen zu können. IndraLogic unterstützt den Treiber "CODESYS ApplicationV3" Die Kommunikation erfolgt über Adressmonitoring. Es kann kein anderer Treiber ausgewählt werden.
Hinzufügen	Öffnet den Dialog "Datenquelle initialisieren - Providereinstellungen". Der Inhalt des Dialogs hängt vom gewählten Datenquellentyp ab.



Die entfernte Steuerung sollte sich während der Initialisierung der neuen Datenquelle in Ausführung sein. Dabei sollte die entfernte Steuerungsapplikation geladen und gestartet sein.

Siehe [Kap. 18.1.4 "Objekt 'Datenquelle'- Registerkarte 'Variablen'" auf Seite 511](#)

Dialog 'Datenquellenassistenten initialisieren - Providereinstellungen' (bei Datenquellentreiber 'CODESYS ApplicationV3')



Die Einstellungen dieses Dialogs sind im Kapitel "Objekt 'Datenquelle' - Registerkarte 'Kommunikation'" beschrieben.

Dialog 'Datenquellenassistenten
initialisieren - Datenquellenvariab-
len durchsuchen'

Der Dialog dient der Initialen Konfiguration der Verbindung für den Datenquellentyp "CODESYS ApplicationV3". Die Kommunikation erfolgt über Adressmonitoring.

Nachdem die Konfigurationseinstellungen erfolgt sind, kann der Dialog über "Beenden" geschlossen werden. Die neue Datenquelle wird angelegt und initialisiert.

Die Kommunikationseinstellungen können über den Editor der neuen Datenquelle nachträglich bearbeitet werden.

Siehe [Kap. 18.1.6 "Registerkarte 'Kommunikation'" auf Seite 514](#)



Die Einstellungen dieses Dialogs sind in Kapitel "Objekt 'Datenquelle'" unter Überschrift "Dialog 'Variablen wählen'" beschrieben.

Funktion: Möglichkeit zur Auswahl der Variablen der entfernten Steuerung, die übertragen werden sollen. Gespeichert werden die übertragenen Daten in der Datenschnittstelle. Der Dialog listet dazu die entfernten Daten auf, auf die über die konfigurierte Verbindung zugegriffen werden kann.

Nach erfolgter Auswahl kann der Dialog über "Beenden" abgeschlossen werden.

Die Datenquelle wird initialisiert, die Datentypen und Variablen (Datenschnittstelle) werden unter dem Ordner „DataSources_Objects“ deklariert.

Der Editor zum Datenquellenobjekt ermöglicht Ihnen, die Einstellungen nachträglich zu bearbeiten.

Siehe ["Dialog 'Variablen wählen'" auf Seite 513](#)

18.1.4 Objekt 'Datenquelle'- Registerkarte 'Variablen'

Symbol:

Das Objekt (Typ "Datenquelle") verwaltet den Zugriff auf die Daten eines entfernten Geräts in den Registerkarten "Variablen", "Typabbildungen", "Kommunikation" und "Allgemein und Diagnose". Die immer sichtbare Statuszeile informiert über den Datenquellentyp und die wichtigsten Kommunikationseinstellungen.

Siehe

- [Kap. 10 "Mit Steuerungsnetzwerken arbeiten" auf Seite 163](#)
- [Kap. 18.1.5 "Objekt 'Datenquelle'- Registerkarte 'Typabbildungen'" auf Seite 513](#)
- [Kap. 18.1.6 "Registerkarte 'Kommunikation'" auf Seite 514](#)
- [Kap. 18.1.7 "Objekt 'Datenquelle'- Registerkarte 'Allgemein und Diagnose'" auf Seite 517](#)

Registerkarte 'Variablen'

Die Variablen, für die aus der entfernten Quelle kommenden Daten, sind in der globalen Variablenliste <Name der Datenquelle> deklariert. Die GVL fungiert als Datenschnittstelle zur entfernten Steuerung und befindet sich unterhalb der Applikation unter dem Order "DataSources_Objects".

Eine Lokale Applikation kann Offline, ohne Verbindung zur Datenquelle auf Basis der Symbolinformation entwickelt werden.

Variablen aktualisieren	Baut eine Verbindung zum entfernten Gerät auf und öffnet den Dialog "Variablen wählen".
Lokale Variable	Zugriffsrecht der Variablen. Die zugehörige entfernte Variable hat das gleiche Zugriffsrecht. : Schreibzugriff. Immer wenn sich der Wert ändert, wird die Variable auf der Steuerung aktualisiert. : Lesezugriff. Immer wenn sich der Wert auf der Steuerung ändert, wird die Variable in der Applikation aktualisiert. : Lese- und Schreibzugriff
Immer aktualisieren	: Die Steuerungsdaten werden (über die Datenquelle) automatisch aktualisiert. Eine Variable wird automatisch aktualisiert, wenn sie in der Visualisierung, als Alarm, im Trend oder in der Rezeptur verwendet wird. Hinweis: Das ist die empfohlene Einstellungsart. : Die Variable wird bei jedem Zyklus aktualisiert. Hinweis: Aktivieren Sie die Option nur dann, wenn die Variable ausschließlich im IEC-Code verwendet wird. Wenn eine Variable im Visualisierungscode verwendet wird, wird sie automatisch aktualisiert. Hinweis: Wenn eine Instanz eines Funktionsbausteins oder eines Datentyps auf diese Weise aktualisiert wird, wird die Instanz vollständig und immer übertragen!
Erzeugen oder abbilden	Abbildungsart, wie die entfernte Variable auf die lokale Variable abgebildet wird. : Abbildung auf eine eigens erzeugte Variable mit dem Datentyp der entfernten Variablen. Die Steuerdaten werden 1:1 abgebildet. Das ist die empfohlene Abbildungsart. Die Variable ist in der GVL <Name der Datenquelle> deklariert. : Abbildung auf eine bereits bestehende Variable. Das erfordert, dass die bestehende Variable den gleichem Datentyp hat. : Abbildung auf eine eigens erzeugte Variable mit typkonformen Datentyp zum entfernten Datentyp: Entfernter und lokaler Datentyp sind nicht gleich, aber kompatibel. Ein typkonformer Datentyp kann beispielsweise in einer Bibliothek zur Verfügung stehen. Die Variable ist in der GVL <Name der Datenquelle> deklariert.
Typabbildung	Datentyp der entfernten Variablen. Wenn die Variable keinen skalaren Typ hat, ist der Typ in der Registerkarte "Typabbildung" aufgelistet.
Entfernte Variable	Variable in der entfernten Steuerung

Dialog 'Variablen wählen'

Symbol: 

Funktion: Der Dialog listet die entfernten Variablen auf, auf die über die konfigurierte Verbindung zugegriffen werden kann.

Aufruf: Klick auf "Variablen aktualisieren" in der Registerkarte "Variablen".

Voraussetzung: Die entfernte Steuerung ist in Ausführung. Die Steuerungssapplikation ist geladen.

Variablen	Die entfernten Variablen werden in Baumansicht aufgelistet. Der oberste Knoten ist mit dem entfernten Applikationsnamen bezeichnet. Darunter sind deren Variablen aufgelistet. Strukturierte Daten werden mit allen ihren untergeordneten Elementen aufgelistet. Bespiel: <code>appControl_A</code> : Die Variable ist für die Übertragung zum lokalen Gerät ausgewählt. Wenn die Variable strukturiert ist, wird sie mit allen Unterelementen übernommen. Wenn die Variablen selbst ein Unterelement ist, wird nur dieses Unterelement übernommen, ohne die Struktur vollständig zu übernehmen. Rote Schrift: Wenn eine Variable mit roter Schrift dargestellt wird, dann ist die Variable in der fernen Steuerung nicht (mehr) verfügbar. : Die Variable ist nicht für die Übertragung ausgewählt.
	Die Variable verfügt über aufklappbare Elemente. Mit Klick auf das Symbol wird die Variable um ihre Elemente erweitert.
Elemente strukturiert einfügen	: Die selektierten Variablen werden, wenn sie denn strukturiert sind, mit dieser Struktur übertragen. : Die Variable ist nicht für die Übertragung ausgewählt.

18.1.5 Objekt 'Datenquelle' - Registerkarte 'Typabbildungen'

Die Registerkarte listet die nicht skalaren Datentypen auf, wie sie im Order "DataSources_Objects" aktuell vorhanden sind. Sie können die Datentypdeklaration bearbeiten, indem Sie einen Datentyp auswählen und dann im unteren Fenster die deklarierten Elemente ändern, oder entfernen. Außerdem können Sie den Namen ändern, das Zugriffsrecht neu setzen, einen anderen Typ zuweisen oder eine andere entfernte Variable auswählen.

Siehe [Kap. 10.4.3 "Datenquellenvariablen bearbeiten"](#) auf Seite 179

Registerkarte 'Typabbildungen'

Lokaler Typ	Datentyp in der lokalen Applikation
Erzeugen oder abbilden	: Abbildung auf einen neu erstellten Datentyp. Deklariert unter dem Ordner "DataSources_Objekts". : Abbildung auf einen bestehenden Datentyp : Abbildung auf einen typkonformen Datentyp. Deklariert unter dem Ordner "DataSources_Objekts".
Name der Abbildung	Name des Datentyps

Entfernter Typ	Datentyp der fernen Steuerung
Liste mit den untergeordneten Elementen des selektierten Datentyps	
Lokale Variable	Lokaler Variablenname des Elements des selektierten Datentyps
Zugriffsrecht	Zugriffsrechte auf das Element
Typabbildung	Datentyp des Elements
Entfernte Variable	Entfernter Variablenname des Elements des selektierten Datentyps

18.1.6 Registerkarte 'Kommunikation'

Die Registerkarte enthält die Kommunikationseinstellungen zur entfernten Datenquelle. Beim Hinzufügen einer Datenquelle wurde ein Datenquellentyp ausgewählt und abhängig davon die Kommunikationseinstellungen zur Datenquelle initial konfiguriert. Die Kommunikationseinstellungen werden in dieser Registerkarte verwaltet und können bearbeitet werden. Der Inhalt der Registerkarte variiert entsprechend dem Datenquellentyp und der Art der Verbindung!

Siehe

- [Kap. 10.4.2 "Datenquelle initial hinzufügen" auf Seite 175](#)
- [Kap. 18.1.3 "Objekt 'Datenquellenverwalter'" auf Seite 510](#)

Registerkarte 'Kommunikation' bei Datenquellentyp CODESYS ApplicationV3

Registerkarte des Datenquellentyps "CODESYS ApplicationV3", zur Konfiguration der Verbindungseinstellungen (die Kommunikation erfolgt über Adressmonitoring).

Projekttyp auswählen	• "Aktuelles Projekt": Die Steuerungsapplikation ist Teil des aktuellen Projekts. • "Anderes Projekt": Die Steuerungsapplikation ist als separates Projekt gespeichert und nicht Teil des aktuellen Projekts.
Datei wählen	Name und Pfad des Projekts, das die Applikation für die entfernte Steuerung enthält. Beispiel: C:\Data\Project new\Project\IndraLogic\IndraLogic.project Voraussetzung: "Projekttyp wählen" ist "Anderes Projekt".
	Fenster, das die gefundene Steuervariablen in Bauman-sicht auflistet. Wenn hier die Steuerdaten dargestellt werden, kann eine Verbindung aufgebaut werden.

"Zielgerät"

Geräteadresse verwenden:	Die Adresse des gerade verbundenen Datenquellengeräts wird eingetragen, so wie sie in dessen Kommunikationseinstellungen konfiguriert ist. Beispiel: 0000.0A34.864B.C000.2DDC.0A34.906F Folgende Fälle führen zu einer nicht korrekten Adresse: • Der Gateway, über den die Kommunikation mit dem Gerät erfolgt, läuft auf einem Rechner mit mehreren IPAdressen. • Das Gerät, auf dem die Applikation mit dem Datenquellenverwalter läuft, hat mehrere IP-Adressen oder noch andere Schnittstellen (USB, seriell, etc.) für die Kommunikation mit IndraLogic. • Das IndraLogic Laufzeitsystem und der Gateway befinden sich nicht im gleichen Ethernet-Subnetz.
Von Gerät	Mit Klick auf "Von Gerät" trägt IndraLogic die Daten des aktuell angebundenen Datenquellengeräts automatisch ein, so wie sie in dessen "Kommunikationseinstellungen" konfiguriert sind.

Zielsystem mit Hilfe des Netzwerk-Scans suchen	Mögliche Suchkriterien für das Gerät: • Knotenname • Zielsystemtyp Beispiel: 4096 • Zielsystem-ID Beispiel: 1001 0114 • Zielsystemversion Beispiel: 14.20.20.0 • Netzwerkpfad – "Direktes Kind der Datenquellen-SPS": Die gesuchte entfernte Steuerung hat eine Adresse, die mit der Adresse der lokalen Steuerung (des Datenquellenverwalters) läuft. – "Direktes Kind des Knotens mit Adresse": Adresse des Vaterknotens angegeben – "Direktes Kind der Datenquellen-SPS oder des Knotens mit Adresse": Kombination der beiden oben genannten Optionen.
Suchtyp:	• "Zuerst gefundenes Gerät": Erste Steuerung im Projektexplorer, die die angegebenen Kriterien erfüllt, wird ausgewählt • "Exakt gefundenes Gerät": Steuerung, die genau die angegebenen Kriterien erfüllt, wird ausgewählt Hinweis: Der Datenquellenverwalter wartet, bis das Durchsuchen des Netzwerks abgeschlossen ist. Üblicherweise dauert das ca. 10 Sekunden.

"Login-Konfiguration"

Wenn auf der entfernten Steuerung eine Benutzerverwaltung konfiguriert ist, werden beim Einloggen gültige Zugangsdaten benötigt.	
Typ	• "Login mit folgenden Zugangsdaten": In den Einstellungen "Benutzername" und "Passwort" sind die Zugangsdaten fest codiert. Sie werden bei jedem Verbindungsversuch verwendet. • "Login mit Zugangsdaten, die zur Laufzeit bestimmt werden": Zur Laufzeit erscheint ein Dialog, der zur Eingabe von Name und Passwort auffordert.
Benutzername	Beispiel : max.mustermann Voraussetzung: Einstellung "Typ" ist "Login mit folgenden Zugangsdaten"
Passwort	Voraussetzung: Einstellung "Typ" ist "Login mit folgenden Zugangsdaten"

"Experteneinstellung"

Experteneinstellung	 : Die Einstellung für "Standardkommunikationspuffer" ist sichtbar.

Siehe [Kap. 10 "Mit Steuerungsnetzwerken arbeiten"](#) auf Seite 163

18.1.7 Objekt 'Datenquelle' - Registerkarte 'Allgemein und Diagnose'

Die Registerkarte "Allgemein und Diagnose" informiert über den Status der Datenquellenkommunikation.

Siehe [Kap. 10.4.5 "Datenschnittstelle aktualisieren"](#) auf Seite 182

Registerkarte 'Allgemein und Diagnose'

Konfiguration aktualisieren

Aktualisierungsrate (ms)	Beispiel: 200

Verbindungsinformation

Verbindungsstatus	Beispiel: online
Fehlerinformation	Beispiel: OK

18.1.8 Objekt 'DUT'

Symbol:  für DUT ohne Textlistenunterstützung,  für DUT vom Typ Enumeration mit Textlistenunterstützung.

Ein DUT (Data Unit Typ) beschreibt einen anwenderspezifischen Datentyp. Sie können ein Objekt **DUT** unterhalb der Applikation oder im Ordner "Allgemeiner Baustein" hinzufügen.

Bereits beim Hinzufügen können Sie bestimmte Definitionen vornehmen, siehe unten: Dialog **DUT hinzufügen**.

Syntax

```
TYPE <Bezeichner> : <DUT Komponentendeklaration>
END_TYPE
```

Die Komponentendeklaration hängt vom gewählten Typ ab, Struktur oder 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

Siehe auch

- [Kap. 17.5.16 "Datentyp 'STRUCT'" auf Seite 396](#)
- [Kap. 17.5.17 "Enumerationen" auf Seite 398](#)
- [Kap. 17.5.6 "Datentyp 'UNION'" auf Seite 377](#)

Dialog 'DUT hinzufügen'

Funktion: Der Dialog dient der Konfiguration eines neuen DUT.

Der DUT (Data Unit Typ) beschreibt einen anwenderspezifischen Datentyp.

Aufruf: Menü **Hinzufügen** ► **DUT**, Kontextmenü des Objekts Applikation.

Name	Name der DUT

Datentyp

Struktur	Der DUT besteht aus einer Struktur verschiedener Datentypen. Erweitert : Die Struktur erweitert eine bereits bestehende Struktur. Somit sind die Definitionen der erweiterten DUT automatisch in der neuen gültig.
Enumeration	Der DUT besteht aus einer Reihe von Integer-Konstanten. Zur Deklarationssyntax sehen Sie bitte die Hilfeseite zu "Aufzählungen / Enumerationen". <i>"Textlistenunterstützung hinzufügen"</i> : Die Textlistenunterstützung ermöglicht eine Lokalisierung der Enumerationskomponenten. Außerdem werden in der Textausgabe eines Visualisierungselements statt der numerischen Enumerationswerte die symbolischen dargestellt. Wenn eine textlistenunterstützte Enumerationsvariable in der Eigenschaft „<i>Textvariable</i>“ eines Visualisierungselements eingetragen wird, erhält sie den Zusatz <code><Enumerationsname></code>. Mit Hilfe der Schaltflächen am rechten Editorrand können Sie zwischen  „<i>Textuelle Ansicht</i>“ und  „<i>Lokalisierungsansicht</i>“ (Textliste) wechseln. Das DUT-Objekt erscheint im Knoten "Application", oder im Ordner "Allgemeiner Baustein" mit dem Symbol . Beispiel: Sie verwenden die Variable <code>PLC_PRG.enVar</code> vom Typ <code>myEnum</code>. <code>myEnum</code> ist ein textlistenunterstützter DUT. Dann sieht der Eintrag im Eigenschafteneditor wie folgt aus: <code>PLC_PRG.enVar <myEnum></code>. Wenn der Enumerationstyp in der Applikation geändert wird, erscheint eine Eingabeaufforderung mit der Frage, ob IndraLogic die betroffenen Visualisierungen entsprechend aktualisieren soll. Hinweis: Auf einem bestehenden Enumerationsobjekt kann die Textlistenunterstützung jederzeit nachträglich hinzugefügt oder wieder entfernt werden: Dazu dienen die Befehle „<i>Textlistenunterstützung hinzufügen</i>“ oder „<i>Textlistenunterstützung entfernen</i>“ im Kontextmenü des Objekts. : Eine Enumeration ohne Textlistenunterstützung wird angelegt. Das DUT-Objekt erscheint im Knoten "Application", oder im Ordner "Allgemeiner Baustein" mit dem Symbol 

Alias	Definition eines Alias für einen Basistyp . Sie können den Basistyp direkt eingeben oder über die Eingabehilfe oder den Arrayassistenten auswählen
Union	Bei diesem Datentyp belegen alle Komponenten denselben Speicherplatz. Der Speicherplatz-Bedarf einer Union wird bestimmt durch den Speicherplatzbedarf seiner "größten" Komponente.

Siehe auch

- [Kap. 17.5.16 "Datentyp 'STRUCT'" auf Seite 396](#)
- [Kap. 17.5.17 "Enumerationen" auf Seite 398](#)
- [Kap. 17.5.6 "Datentyp 'UNION'" auf Seite 377](#)
- [Kap. "Befehl 'Textlistenunterstützung hinzufügen'" auf Seite 731](#)
- [Kap. "Befehl 'Textlistenunterstützung entfernen'" auf Seite 732](#)

18.1.9 Objekt 'Gerät' und Generischer Geräteeditor

Symbol: 

Ein Geräteobjekt repräsentiert eine bestimmte Hardware.

Beispiele:

Steuerungsgerät, Feldbusknoten, Buskoppler, Antrieb, E/A-Modul, Monitor.

Durch die Anordnung der Geräteobjekte im Projektextplorer bilden Sie die Hardwarestruktur ab.

In den Konfigurationseditoren der Geräteobjekte können Sie dann auch die E/As der Geräte mit Projektvariablen verknüpfen.

Fügen Sie ein Geräteobjekt mit Befehl **Hinzufügen** in den Projektextplorer ein (oder per Drag&Drop aus der IndraWorks-Bibliothek "Drive and Control").

IndraLogic bietet jeweils die zur Einfügeposition im Explorer passenden Geräte zur Auswahl an.

Mit einem Doppelklick auf ein Geräteobjekt im Explorer öffnen Sie den zugehörigen Geräteeditor. Dieser enthält generische und gerätespezifische Registerkarten zur Konfiguration des Gerätes.

Siehe auch

- [Kap. 8.1 "Projektextplorer und Geräteeditor" auf Seite 71](#)
- [Kap. "Generischer Geräteeditor" auf Seite 519](#)

Generischer Geräteeditor

Der generische Geräteeditor enthält Registerkarten für die Konfiguration eines SPS-Gerätes in IndraLogic. Zusätzlich gibt es gerätespezifische Registerkarten, so dass sich der Konfigurationseditor je nach Gerät aus unterschiedlich vielen Dialogen zusammensetzt.

Der Editor öffnet sich nach einem Doppelklick auf das Geräteobjekt im Projektextplorer (Ansicht **Geräte**).

In **Extras** ► **Optionen** ► **IndraLogic 2G** ► **Allgemeine Einstellungen** und ... ► **Geräteeditor** können Sie Einstellungen für einen Geräteeditor modifizieren.

Beispielsweise können Sie Registerkarten des generischen Geräteeditors ein- und ausblenden.

Ein Geräteeditor ist mit dem Namen des Geräts betitelt.

Folgende Registerkarten des generischen Geräteeditors können enthalten sein:

- **Applikationen:** Liste der Applikationen auf der Steuerung.
- **Dateien:** Konfiguration der Dateiübertragungen zwischen einem "Host"-Dateisystem und dem Gerät, Kontextmenü: **Geräte-Daten**.
- **Log:** Anzeige der SPS-Log-Datei.
- **SPS-Einstellungen:** Konfiguration der Behandlung der E/As: welche Applikation, Verhalten im Stop-Status, Aktualisierung, Buszyklus-Optionen, etc.
- **SPS-Shell:** Textbasierter Steuerungsmonitor zur Abfrage bestimmter Informationen aus der Steuerung.
- **Benutzer und Gruppen:** Benutzerverwaltung bezüglich möglicher Zugriffe auf das Gerät zur Laufzeit.
- **Zugriffsrechte:** Rechte für Zugriffe auf Objekte und Dateien auf dem Gerät.
- **Parameter:** Anzeige oder Konfiguration der Geräteparameter; Gleitkomma-Einstellungen: Verhalten bei Ausnahmen; allgemeines Verhalten bei Gleitkomma-Ausnahmen.
- **E/A-Abbild:** Mapping von IEC-Objekten
- **Task-Aufstellung:** Übersicht aller Ein- und Ausgänge, welche Tasks zugeordnet sind - nützlich zur Fehlersuche.
- **Status:** Gerätespezifische Status- und Diagnosemeldungen.
- **Information:** Allgemeine Information zum Gerät (Name, Hersteller, Version etc.)

Siehe auch:

- [Kap. 8.1 "Projektexplorer und Geräteeditor" auf Seite 71](#)
- [Kap. 8 "E/A-Anbindung konfigurieren" auf Seite 71](#)

Registerkarte 'Parameter'

In dieser Registerkarte des generischen Geräteeditors werden die gerätespezifischen Parameter in einer Tabelle angezeigt. Die Gerätebeschreibung legt fest, welche Parameter Sie in diesem Dialog bearbeiten können.

Durch einen Mausklick auf die Spaltenüberschrift können Sie die Einträge alphabetisch aufsteigend, absteigend oder in der Standardreihenfolge sortieren.

Parameter	Parameternamen, nicht editierbar
Typ	Datentyp des Parameters, nicht editierbar
Wert	Zeigt zunächst den Standardwert des Parameters, direkt oder den entsprechenden Symbolnamen. Nicht editierbare Parameter sind hellgrau dargestellt. Wenn der Parameter editierbar ist, können Sie mit einem Doppelklick auf das Tabellenfeld ein Eingabefeld, eine Auswahlliste oder einen Dateiauswahl-Dialog öffnen und darüber den Wert ändern.
Standardwert	Von der Gerätebeschreibung definierter Standardwert des Parameters, nicht editierbar
Einheit	Maßeinheit für den Wert; Beispiel: "ms" für Millisekunden; nicht editierbar
Beschreibung	Von der Gerätebeschreibung vorgegebene Kurzbeschreibung zum Parameter, nicht editierbar

Siehe auch

- [Kap. 8.1 "Projektexplorer und Geräteeditor" auf Seite 71](#)
- [Kap. "Generischer Geräteeditor" auf Seite 519](#)

Registerkarte 'Applikationen'

In dieser Registerkarte des generischen Geräteeditors sehen Sie, welche Applikationen auf dem Gerät liegen. Je nach System können Sie die Applikationen auf dem Gerät löschen oder Detailinformationen zur Applikation abrufen.

Applikationen auf der SPS	Liste der Applikationen, die beim letzten Durchsuchen des Steuerungsgeräts, über Liste aktualisieren dort gefunden wurden.
Löschen Alle löschen	Löscht die in der Liste selektierte Applikation beziehungsweise alle aufgelisteten Applikationen auf der Steuerung Hinweis: Wenn unter einer Steuerung eine Sicherheitssteuerung projektiert ist, kann dieser Befehl die Kommunikationsverbindungen der Sicherheitssteuerung zu anderen Sicherheitssteuerungen (über Safety Netzwerkvariablen), zu Feldgeräten und zum Entwicklungssystem dauerhaft unterbrechen! Die sicheren Feldgeräte bzw. die andere Sicherheitssteuerung können als Reaktion in den sicheren Zustand gehen! Die Verbindung zum Entwicklungssystem ist nur betroffen wenn die Sicherheitssteuerung über den Feldbus an die Hauptsteuerung angeschlossen ist. Siehe Kap. 10.5 "Untergeordnete Sicherheitssteuerung" auf Seite 184
Details	Öffnet den Dialog Details . Er zeigt Informationen, die im Registerkarte Informationen des Dialogs Eigenschaften für die Applikation definiert wurden.
Inhalt	Voraussetzung: In den Eigenschaften des Applikationsobjekts, in Registerblatt Optionen Applikationserzeugung ist die Option Download der Applikationsinfo aktiviert. Das bewirkt, dass zusätzlich Information über den Inhalt der Applikation auf die SPS geladen wird. Die Schaltfläche Inhalt öffnet einen Dialog mit zusätzlicher Information über die Unterschiede zwischen dem zuletzt erzeugten Code und dem bereits auf der Steuerung liegenden Code der Applikation. Die unterschiedlichen Bausteine sind in einer Vergleichsansicht dargestellt.
Liste aktualisieren	Die Steuerung wird nach Applikationen durchsucht, die Liste wird entsprechend aktualisiert

Siehe auch

- [Kap. 18.1.9 "Objekt 'Gerät und Generischer Geräteeditor' auf Seite 519](#)
- [Kap. 18.1.2 "Objekt 'Applikation'" auf Seite 508](#)
- [Kap. "Generischer Geräteeditor" auf Seite 519](#)
- [Kap. "Registerkarte 'Information'" auf Seite 543](#)
- [Kap. "Dialog 'Eigenschaften - 'Optionen Applikationserzeugung'" auf Seite 751](#)
- [Kap. "Befehl 'Einloggen'" auf Seite 652](#)
- [Kap. 10.5 "Untergeordnete Sicherheitssteuerung" auf Seite 184](#)

Registerkarte 'Sichern und Wiederherstellen'



Diese Registerkarte des generischen Geräteeditors ist ausschließlich für die Steuerungs-Hardware IndraControl VPx im System IndraMotion MLC verfügbar.

Über die Registerkarte veranlassen Sie das Sichern und Wiederherstellen der applikationsspezifischen Dateien auf der SPS. Dabei wird eine gepackte Sicherungsdatei erzeugt, oder wieder eingelesen.

Voraussetzungen:

- Die Kommunikationseinstellungen für die Verbindung zum Gerät sind korrekt
- Die Applikation, für die eine Sicherung durchgeführt werden soll, ist auf der Steuerung verfügbar.

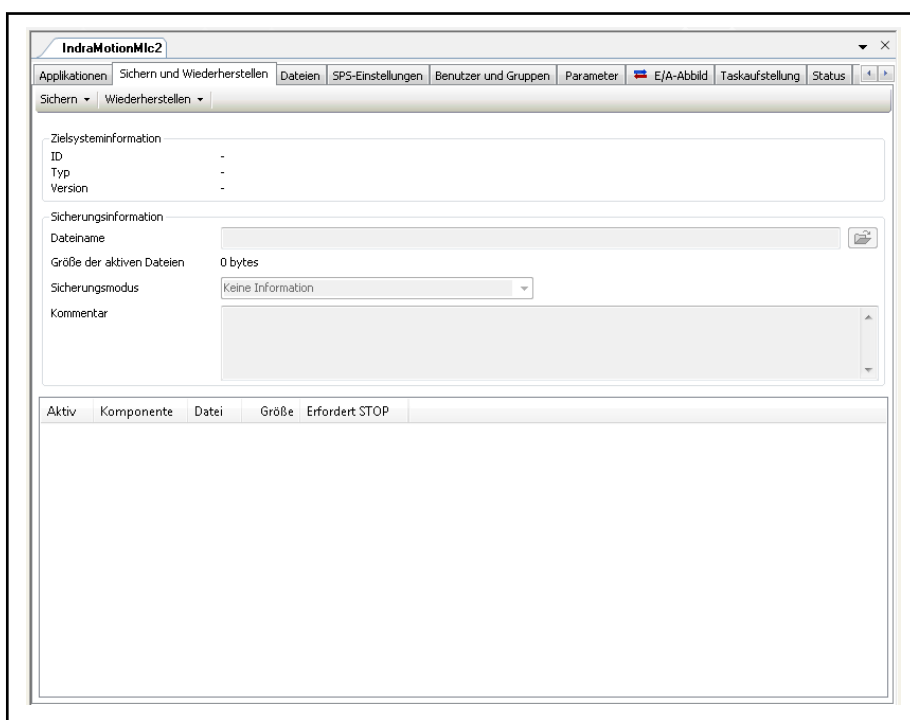


Abb. 18-3: Registerkarte "Sichern und Wiederherstellen"

Menüleiste

Sicherung	Die Schaltfläche öffnet ein Menü mit folgenden Befehlen:  "Sicherungsinformation vom Gerät lesen": Der Befehl sucht nach applikations-spezifischen Dateien vom \$PlcLogic\$-Verzeichnis der SPS und zeigt sie im unteren Teil der Registerkarte als Tabelle an.  "Sicherungsdatei erzeugen und auf Platte speichern": Voraussetzung: Mit Befehl "Sicherungsinformation vom Gerät lesen" wurden die sicherungsrelevanten Dateien ermittelt. Sie stehen jetzt in der Tabelle im unteren Teil der Registerkarte. Der Befehl packt die in der Tabelle als "Aktiv" gesetzten Dateien zusammen mit der Informationsdatei meta.info in eine Sicherungsdatei im ZIP-Format. Die Dateierweiterung ist tbf (= "Target Backup File").  "Sicherungsdatei auf Gerät speichern" : Voraussetzung: Die Sicherungsdatei wurde zuvor auf Platte gespeichert. Der Befehl speichert die Sicherungsdatei nun außerdem auf der SPS im Verzeichnis TBF.
Wiederherstellen	Die Schaltfläche öffnet ein Menü mit folgenden Befehlen:  "Sicherungsdatei von der Platte laden" : Der Befehl öffnet den Dialog "Datei öffnen" zum Durchsuchen des Dateisystems nach einer gespeicherten Sicherungsdatei. Die enthaltenen Dateien werden dann in der Tabelle im unteren Teil der Registerkarte dargestellt.  "Sicherungsdatei vom Gerät laden": Der Befehl erzeugt eine Liste aller auf der Steuerung gefundenen Sicherungsdateien. Sie wählen eine der Dateien und erhalten dann deren Inhalt auf der Registerkarte in Tabellenform. Darin können Sie für die bevorstehende Wiederherstellen- Operation noch optionale Komponenten deaktivieren und den Kommentar bearbeiten.  "Wiederherstellen auf Gerät": Der Befehl ist verfügbar, wenn mindestens eine Komponente der gerade in der Registerkarte geladenen Sicherungsdatei aktiv gesetzt ist. Er veranlasst ein entsprechendes Wiederherstellen des Applikationszustandes auf dem Gerät. Die Benutzeroberfläche ist während der Wiederherstellung gesperrt. Sie können den Vorgang abbrechen.

Zielsysteminformation

ID	ID der SPS, Beispiel: 1001 0023
Typ	Gerätetyp, Beispiel: 4096
Version	Geräteversion, Beispiel: 14.19.20.0

Sicherungsinformation

Dateiname	Speicherpfad der Sicherungsdatei. Über Schaltfläche gelangen Sie zum Dateisystem-Browser. Beispiel: PlcLogic\$/ Application/ Application.crc
Größe der aktiven Dateien	Angabe in Kilobytes: Gesamtumfang der unten in der Tabelle als aktiv gesetzten Dateien, Beispiel: 206 KB (210965 bytes)

Sicherungsmodus	Definiert den Umfang der Sicherung: "Applikation". Die applikationsrelevanten Dateien werden gepackt.
Kommentar	Optionale Eingabe eines Kommentars, der in der Datei <code>meta.info</code> der Sicherung gespeichert wird und beim Wiederherstellen wieder gelesen werden kann.

Tabelle der Dateien für die Sicherung

Aktiv	 : Optionale Dateien können Sie hier deaktivieren, damit sie nicht in die Sicherungsdatei aufgenommen werden. Bei zwingend erforderlichen Komponenten erscheint hier nur ein grünes Häkchen ohne Checkbox.
Komponente	Betroffene Komponente, Beispiel: <code>Dateisystem</code>
Datei	Name der zu sichernden Datei der Komponente, Beispiel: <code>\$PlcLogic\$/ Application/Application.app</code>
Größe	Größe der Datei in Bytes, Beispiel: <code>43280</code>
Erfordert STOP	 : Für die Komponente sollte die Applikation vor dem Sichern oder Wiederherstellen unbedingt gestoppt werden. Sie erhalten in dem Fall beim Anstoßen der Sicherung oder Wiederherstellung eine entsprechende Warnung.

Registerkarte 'Synchronisierte Dateien'

Diese Registerkarte des generischen Geräteeditors zeigt eine Liste der Dateien, die beim Applikationsdownload mit auf die SPS geladen werden. Beispielsweise sind dies externe Dateien, die einer Applikation hinzugefügt wurden.

Implizite Dateien, wie beispielsweise die Quellcode-Archivdatei erscheinen hier nur, wenn Ihr Download-Zeitpunkt entsprechend konfiguriert ist und in den IndraLogic 2G-Optionen in Kategorie "Geräteeditor" die Option "Implizite Dateien für den Applikationsdownload im Editor einer SPS anzeigen" aktiviert ist.

Dateiname	Name der Datei unterhalb der Applikation, oder direkt Name der implizit transferierten Datei. Bei Doppelklick auf den Dateinamen wird die Datei geöffnet.
Host-Pfad	Speicherort beziehungsweise Herkunftsort der Datei, Beispiel: <code>D: \Proj1\Files</code> . Bei Doppelklick auf den Pfad wird das Verzeichnis im Datei-Explorer geöffnet.
Zeitintervall	Zeitintervall der Aktualisierung der Datei auf der SPS, Beispiel: Nach Download/Online-Change der Applikation
Information	Objektabhängige Zusatzinformation, Beispiel: Objekt: Externe Datei
Herkunft	Genereller Herkunftstyp der Datei, Beispiele: Externe Dateiobjekte, Quellcode-Download Provider

Siehe auch

- [Kap. "Dialog 'Optionen- 'Geräteeditor'" auf Seite 767](#)
- [Kap. 18.1.9 "Objekt 'Gerät und Generischer Geräteeditor' auf Seite 519](#)

Registerkarte 'Dateien'

In dieser Registerkarte des generischen Geräteeditors können Sie einen Dateitransfer zwischen IndraLogic (Host) und dem Steuerungsgerät durchführen.

IndraLogic stellt die Verbindung zur Steuerung automatisch für die Zeit des Dateitransfers her, wenn die Kommunikationseinstellungen korrekt sind und die Steuerung läuft.

Linker Dialogteil, Host

Ort	Pfad im Dateisystem des Hosts. Einstellbar wie im Standard-Datei-Manager. Unterverzeichnisse und Dateien werden unten im Fenster mit Name, Größe und Änderungsdatum angezeigt.
	Ein Dialog zum Anlegen eines neuen Verzeichnisses im eingestellten Pfad erscheint.
	Löscht die ausgewählten Dateien oder Verzeichnisse.
	Aktualisiert für den eingestellten Pfad (Ort) die Liste der Dateien und Verzeichnisse.
 	Kopiert die ausgewählten Dateien und Verzeichnisse ins jeweilige andere Dateisystem. Wenn eine Datei im Zielverzeichnis 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  entspricht dem Befehl "Datei auf Steuerung schreiben"  entspricht dem Befehl "Datei von Steuerung lesen"

Siehe auch

- [Kap. "Generischer Geräteeditor" auf Seite 519](#)
- [Kap. 15 "Dateien von/zur SPS kopieren" auf Seite 233](#)

Registerkarte 'Log'

In dieser Registerkarte des generischen Geräteeditors können Sie das "Logbuch" der SPS einsehen. Es listet die Ereignisse, die auf dem Zielsystem aufgezeichnet wurden.

Dies betrifft:

- Ereignisse beim Systemstart und -beenden (geladene Komponenten mit Version)
- Applikationsdownload und Laden der Bootapplikation
- Kundenspezifische Einträge
- Log-Einträge von I/O-Treibern
- Log-Einträge von den Datenquellen

Offline-Aufzeichnung	☐ : Standardeinstellung ☒ : Die SPS zeichnet auch Aktionen auf, die sich nicht auf die Verbindung mit der Steuerung beziehen. Dies ist momentan nur für die Safety-Version von IndraLogic realisiert.
UTC-Zeit	☐ : Standardeinstellung; der Zeitstempel wird auf die lokale Zeit des Computers umgerechnet (Berechnung anhand der Zeitzonen des Betriebssystems). ☒ : Der Zeitstempel des Laufzeitsystems wird angezeigt.
Gewichtung	Für die Schwere des Ereignisses gibt es folgende Kategorien:  : Information  : Warnung  : Fehler  : Debug  : Ausnahme Sie können jede Kategorie mit Hilfe der entsprechenden Schaltfläche in der Leiste oberhalb der Liste ein- oder ausblenden. In der Schaltfläche sehen Sie jeweils die Anzahl der Log-Einträge der betreffenden Kategorie angezeigt.
Zeitstempel	Datum und Uhrzeit, Beispiel: 12.01.2007 09:48
Beschreibung	Beschreibung des Ereignisses, Beispiel: <code>Import function failed of</code>
Komponente	Name der betroffenen Laufzeitsystem-Komponente, Beispiel <code>CmpApp</code>
Auswahlliste mit Komponentennamen	Die Log-Liste zeigt nur Ereignisse an, die die ausgewählte Komponente betreffen
Logger	Auswahlliste mit allen verfügbaren Aufzeichnungen. Die Standardeinstellung ist der vom Zielsystem vorgegebene <Default Logger>, derzeit identisch mit "PlcLog" für das IndraLogic-Laufzeitsystem
	Aktualisiert die Log-Liste
	Export des Listeninhalts in eine xml-Datei. Sie können Dateinamen und Ablageverzeichnis wählen.
	Import einer Log-Liste aus einer xml-Datei. Die Liste wird dann in einem separaten Fenster dargestellt.
	Die angezeigte Log-Liste wird geleert, das heißt alle Einträge gelöscht.

Tab. 18-2:



Hinweis für die Fehlersuche:

Bei Exceptions mit der Beschreibung `*SOURCEPOSITION*` öffnet mit einem Doppelklick oder mit dem Kontextmenü-Befehl **Source-Code im Editor anzeigen** die betroffene Funktion im Editor.

Der Cursor springt an die fehlerverursachende Zeile.

Diese Diagnose können Sie auch vornehmen, wenn Ihnen das IndraLogic-Projektarchiv inklusive der "Download-Informationsdateien" und der exportierten Log-Datei vorliegt. Ist die betroffene Funktion geschützt, erscheint die Meldung "Der Quellcode ist nicht verfügbar für <Funktionsname>".



Wenn eine `VendorException` gemeldet wird, ist ein hersteller-spezifischer Ausnahmefehler im IndraLogic-Laufzeitsystem aufgetreten.

Für weitere Informationen wenden Sie sich bitte an den Rexroth Service, siehe [Kap. 21 "Service und Support" auf Seite 1083](#).

Siehe auch

- [Kap. "Generischer Geräteeditor" auf Seite 519](#)
- [Kap. 18.1.9 "Objekt 'Gerät' und Generischer Geräteeditor" auf Seite 519](#)
- [Kap. 13.5 "Logbuch der SPS lesen" auf Seite 224](#)

Registerkarte 'SPS-Einstellungen'

In dieser Registerkarte des generischen Geräteeditors setzen Sie Grundeinstellungen für die Konfiguration der SPS wie beispielsweise die Behandlung der Ein- und Ausgänge und die Buszyklustask.

Applikation für E/A-Behandlung	Applikation, die für die E/A-Behandlung zuständig ist.

SPS-Einstellungen

E/As aktualisieren im Stop	: IndraLogic aktualisiert die Werte der Ein- und Ausgangskanäle auch dann, wenn die SPS im Stop-Status ist. Wenn der Watchdog eine Fehlfunktion feststellt, werden die Ausgänge auf die vordefinierten Standardwerte gesetzt. : IndraLogic aktualisiert die Werte der Ein- und Ausgangskanäle im Stop-Status der SPS nicht. Hinweis: Die Einstellung „E/A aktualisieren im Stopp“ bezieht sich ausschließlich auf das Verhalten der SPS. Die Feldbus-Kommunikation ist unabhängig von dieser Einstellung, d.h. bei einer Feldbus-Schnittstelle werden bei SPS-in-Stopp die Daten meist auf „0“ gesetzt und/oder für „ungültig“ erklärt (bus-spezifisch).
Verhalten der Ausgänge bei Stop	<i>Behandlung der Ausgangskanäle, wenn die Steuerung in den Stop-Status geht:</i> • Werte beibehalten: Die aktuellen Werte werden beibehalten. Bitte beachten Sie, dass Ausgänge nach einer IEC-Exception ihren Wert aus dem letzten Zyklus beibehalten. • Alle Ausgänge auf Standardwert: Die Standardwerte, die aus dem E/A-Abbild resultieren, werden zugewiesen. • Programm ausführen: Sie können die Behandlung der Ausgangswerte über ein im Projekt enthaltenes Programm steuern, welches IndraLogic bei "STOP" ausführt. Geben Sie den Namen des Programms im Feld rechts an.
Variablen immer aktualisieren	<i>Globale Festlegung, ob IndraLogic die E/A-Variablen in der Buszyklustask aktualisiert. Diese Einstellung ist für E/A-Variablen der Slaves und Module nur wirksam, wenn in deren Aktualisierungseinstellungen "Deaktiviert" festgelegt ist.</i> • Deaktiviert (Aktualisierung nur wenn in einer Task verwendet): IndraLogic aktualisiert die E/A-Variablen nur, wenn sie in einer Task verwendet werden. • Aktiviert 1 (Buszyklustask verwenden, wenn in keiner Task verwendet): IndraLogic aktualisiert die E/A-Variablen in der Buszyklustask, falls sie in keiner anderen Task verwendet werden. • Aktiviert 2 (immer im Buszyklustask): IndraLogic aktualisiert alle Variablen in jedem Zyklus der Buszyklustask, unabhängig davon ob sie verwendet werden und ob sie auf einen Ein- oder Ausgangskanal abgebildet sind.

Buszyklus-Optionen

Buszyklustask	Task, die den Buszyklus steuert. Standardmäßig ist die von der Gerätebeschreibung definierte Task eingetragen. Standardmäßig gilt die Buszyklus-Einstellung des übergeordneten Busgeräts (Zykluseinstellungen des übergeordneten Busses verwenden), das heißt der Projektextplorer wird nach oben auf die nächste gültige Buszyklustask-Definition hin durchsucht. Nachfolgende Hinweise unbedingt beachten!



Bevor Sie für die Buszyklustask die Einstellung **<unspezifiziert>** wählen, sollten Sie sich über Folgendes klar sein:

<unspezifiziert> bedeutet, dass die in der Gerätebeschreibung vorgegebene Standardeinstellung wirksam wird. Prüfen Sie deshalb diese Beschreibung. Als Standard kann dort das Verwenden der Task mit der kürzesten Zykluszeit definiert sein, genauso gut aber auch das Verwenden der Task mit der längsten Zykluszeit!



Für Feldbusse ist ein festes Zyklusrastrer nötig, um ein determiniertes Verhalten sicherzustellen. Verwenden Sie deshalb für eine Buszyklustask nicht den Typ "freilaufend".

Zusätzliche Einstellungen

Force-Variablen für das E/A-Abbild	Diese Einstellung ist nur verfügbar, wenn sie vom Gerät unterstützt wird. Für jeden E/A-Kanal, der im Dialog E/A-Abbild auf eine Variable abgebildet ("gemappt") ist, erzeugt IndraLogic beim Übersetzen der Applikation zwei globale Variablen. Diese können Sie für das Forcen des Eingangs-bzw. Ausgangswerts an diesem Kanal verwenden, beispielsweise über eine HMI-Visualisierung.
Diagnose für Geräte aktivieren	IndraLogic bindet automatisch die Bibliothek <code>CAA Device Diagnosis</code> im Projekt ein und erzeugt für jedes Gerät einen impliziten Funktionsbaustein. Wenn es für das Gerät bereits einen Funktionsbaustein gibt, wird entweder ein erweiterter FB verwendet (beispielsweise bei EtherCAT) oder eine weitere FB-Instanz hinzugefügt. Diese beinhaltet dann eine allgemeine Implementierung zur Gerätediagnose. Mittels der FB-Instanzen können Sie in der Applikation von allen Geräten den Status ermitteln und Fehler auswerten. Die Bibliothek enthält außerdem Funktionen zum programmatischen Bearbeiten des Projektexplorers. Beispiel: Durchsuchen aller Kinder eines Bussystems, Springen zum Elternelement.
E/A-Warnungen als Fehler anzeigen	Warnungen, die die E/A-Konfiguration betreffen, werden als Fehler ausgegeben.

Siehe auch

- [Kap. "Generischer Geräteeditor" auf Seite 519](#)
- [Kap. "Registerkarte '<Gerätename> E/A-Abbild'" auf Seite 534](#)
- [Kap. "Registerkarte 'Parameter'" auf Seite 529](#)
- [Kap. "Befehl 'Übersetzen'" auf Seite 649](#)
- PDF-Dokument "CAA Device Diagnosis", welches Bestandteil der Bibliothek ist.

Registerkarte 'Parameter'



Voreinstellt ist, dass Gleitkomma-Ausnahmen im SPS-Programm gesperrt sind. Dies gewährleistet die Kompatibilität für bestehende Anwenderprojekte.

Bei IndraMotion MTX und IndraMotion MLD ist die Behandlung von Gleitkomma-Ausnahmen im SPS-Programm nicht konfigurierbar. Der Dialog im Geräteeditor entfällt

Im SPS-Programm können bei Gleitkomma-Berechnungen Ausnahmeforderungen (Floatingpoint-Exceptions) auftreten.

Beispiel: Division durch Null

Hierfür bietet die SPS eine Standardbehandlung an, d.h. die SPS geht in STOP und die Gleitkomma-Ausnahme wird angezeigt.

Der Anwender kann wählen, ob für sein SPS-Projekt beim Auftreten einer Gleitkomma-Ausnahme

- die SPS in Stop gehen soll (Standard-Reaktion) oder
- ob die SPS in Run bleibt und die Berechnung weiter ausführt.

Hintergrund Wenn die Behandlung von Gleitkomma-Ausnahmen gesperrt ist und bei einer Gleitkomma-Berechnung eine Ausnahme auftritt, dann setzt der Gleitkomma-Prozessor (FPU) automatisch einen Wert ein, der in vielen Fällen ein Weiterrechnen ermöglicht.

Beispiel:

Bei Gleitkomma-Berechnungen ergibt eine Division durch 0 als Ergebnis den Wert "Unendlich" (Infinity).

Dies funktioniert aber nicht immer und ist von der Art der Berechnung im Anwenderprogramm abhängig. Ein Weiterrechnen kann deshalb auch zu einer fehlerhaften Reaktion des Anwenderprogrammes führen.



Bei neuen SPS-Projekten sollte zumindest in der Entwicklungsphase die Behandlung von Gleitkomma-Ausnahmen eingeschaltet sein.

Arten der Gleitkomma-Ausnahmen

Im IEEE-Standard für Gleitkommaberechnungen sind verschiedene Arten von Gleitkomma-Ausnahmen definiert. Diese werden von den Gleitkommaprozessoren nach IEEE-Standard unterstützt.

- Overflow
- Underflow
- Divide by zero
- Invalid
- Denormalized
- Inexact (precision)

Es ist nicht für alle Arten von Gleitkomma-Ausnahmen sinnvoll eine Ausnahmebehandlung per Software durchzuführen, z.B. kommt „Inexact“ ständig vor und normalerweise rundet die FPU-Hardware nach einem voreingestelltem Algorithmus, deshalb ist für die Gleitkomma-Ausnahmen „Inexact“ keine Softwarebehandlung auswählbar.

Dialogbeschreibung, Gleitkomma-Ausnahmen

Wenn man im SPS-Projekt einen Doppel-Klick auf die Steuerung macht, öffnet sich im rechten Fenster der Gerätedialog. Hier gibt es ein neues Register „Konfiguration“.

Parameter	Typ	Wert	Standardwert	Einheit	Beschreib.
Gleitkomma Einstellungen					
Verhalten bei Ausnahme	Enumeration of BYTE	keine Behandlung	keine Behandlung		
Gleitkomma Ausnahmen					
Überlauf	Enumeration of BYTE	freigegeben	freigegeben		
Unterlauf	Enumeration of BYTE	freigegeben	freigegeben		
Division durch Null	Enumeration of BYTE	freigegeben	freigegeben		
Ungültige Operation	Enumeration of BYTE	freigegeben	freigegeben		
Denormalisiert	Enumeration of BYTE	freigegeben	freigegeben		

Abb. 18-4: Gerätedialog, hier IndraMotion MLC L65, mit Register Parameter

Im Register "Parameter" sind die Gleitkomma-Einstellungen enthalten.

Die Gleitkomma-Einstellungen gliedern sich in zwei Komponenten:

- Verhalten bei Ausnahme
- Gleitkomma-Ausnahmen

In der Komponente "Verhalten bei Ausnahme" gibt es folgende Auswahlmöglichkeiten:

- **Keine Behandlung:** Alle Ausnahmen-Bedingungen, die bei Gleitkomma-Berechnungen auftreten, werden ignoriert.
Das SPS-Programm wird weiter ausgeführt (d.h. die SPS bleibt in Run).
Die im Dialog eingestellten Gleitkomma-Ausnahmen sind in diesem Fall nicht relevant.
- **SPS in Stop:** Beim Auftreten von freigegebenen Ausnahme-Bedingungen wird die SPS in Stop geschaltet.
Berücksichtigt werden die Gleitkomma-Ausnahmen, die im Dialog auf den Zustand "freigegeben" gesetzt sind.

Die im Dialog angezeigten Gleitkomma-Ausnahmen sind nur relevant, wenn als Verhalten "SPS in Stop" eingestellt ist.

In der Voreinstellung sind alle Gleitkomma-Ausnahmen freigegeben. Der Anwender kann aber gezielt einzelne Gleitkomma-Ausnahmen sperren:



Die Gleitkomma-Ausnahme „Inexact“ wird nicht angezeigt. Sie ist immer gesperrt.

Standardverhalten

Die Reaktion der SPS beim Auftreten einer Gleitkomma-Ausnahme ist gleich zu den bisherigen Ausnahmen:

- Die SPS geht in den Zustand "STOP" und "AUSNAHMEFEHLER".
- Im Logger wird die Ursache der Ausnahme eingetragen.
- Im Monitor wird die Programmstelle angezeigt, die die Ausnahme verursacht hat.
- In der Statuszeile des Programmiersystems wird "STOP" und "AUSNAHMEFEHLER" angezeigt.

Beispiel:

Gleitkomma-Ausnahme „Division durch 0“, Ausnahme **freigegeben**

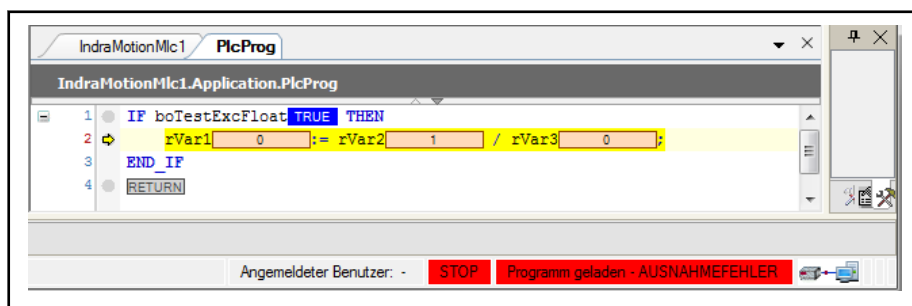


Abb. 18-5: SPS in STOP, AUSNAHMEFEHLER, Anzeige der Ursache im Programmcode

Gewicht...	Zeitstempel	Beschreibung	Komponente
!	17.02.2016 15:52:07	*SOURCEPOSITION* App=[Application] area=0, offset=396750	CmpIecTask
!	17.02.2016 15:52:07	*EXCEPTION* [UNKNOWN] occurred: App=[Application], Task=[PlcTask]	CmpIecTask
i	17.02.2016 15:51:30	Update I/O Mapping after download succeeded.	CmpOpenCtrl
i	17.02.2016 15:51:30	Parse of I/O configuration after download succeeded.	CmpOpenCtrl

Abb. 18-6: Logger-Meldung

Beispiel:

Gleitkomma-Ausnahme „Division durch 0“, Ausnahme **gesperrt**

```

IF boTestExcFloat TRUE THEN
    rVar1 Infinity := rVar2 1 / rVar3 0 ;
END_IF

```

Abb. 18-7: Programmcode, Statusanzeige bei laufendem Programm

Wenn keine Ausnahmen freigegeben sind (wie bisher), läuft das SPS-Programm weiter.

Das Ergebnis der Berechnung ist in diesem Fall der Wert "Infinity".

Registerkarte 'SPS-Shell'

Diese Registerkarte des generischen Geräteeditors enthält einen textbasierten Steuerungsmonitor zur Abfrage bestimmter Informationen aus der Steuerung.

Sie können dazu geräteabhängige Kommandos eingeben und erhalten die Antwort der Steuerung in einem Ergebnisfenster.

Voraussetzungen:

- Die Option 'SPS-Shell' ist aktiviert
- Die Applikationsknoten ist eingeloggt (Verbindung zur Steuerung / SPS ist erforderlich)

Aktivierung: IndraWorks Menü **Extras** ▶ **Optionen** ▶ **IndraLogic 2G** ▶ **Allgemeine Einstellungen**

Kommando mit möglichen Parametern	Beschreibung
?	Liste der verfügbaren SPS-Shell-Kommandos mit möglichen Parametern und Kurzbeschreibung.
getcmdlist	Liste der Namen der verfügbaren SPS-Shell-Kommandos.
mem <Speicheradresse> [<Größe>]	Liefert den Hexdump des definierten Speicherbereiches. Die Größenangabe ist optional und beschreibt die Anzahl von Bytes die ausgegeben werden sollen. Standardwert: 16. Beispiel: mem 16x0422139C 8
reflect	Wiederholt das eingegebene Kommando (zum Testen der Verbindung)
applist	Liefert eine Liste aller geladener Applikationen; die Reihenfolge in der Liste definiert, beginnend mit 0, den Applikationsindex.
pid [<Applikationsname> <Applikationsindex>] *	Liefert die GUID (Applikationsindex) einer oder aller geladenen Applikationen.

Kommando mit möglichen Parametern	Beschreibung
<code>pinf [<Applikationsname> <Applikationsindex>] *</code>	Liefert aus den Bibliotheksinformationen des Projekts die Inhalte folgender Felder: Titel, Version, Autor und Beschreibung. Voraussetzung: Die Option Automatisch POU für den Eigenschaftenzugriff erzeugen im Dialog "Bibliotheksinformationen" wurde aktiviert.
<code>startprg [<Applikationsname> <Applikationsindex>] *</code>	Startet die angegebene Applikation, oder - wenn keine Applikation angegeben ist - alle geladenen Applikationen.
<code>stopprg [<Applikationsname> <Applikationsindex>] *</code>	Stoppt die angegebene Applikation, oder - wenn keine Applikation angegeben ist - alle geladenen Applikationen.
<code>resetprg [<Applikationsname> <Applikationsindex>] *</code>	Setzt die angegebene Applikation zurück, oder - wenn keine Applikation angegeben ist - alle geladenen Applikationen.
<code>resetprgcold [<Applikationsname> <Applikationsindex>] *</code>	Führt einen Kaltstart der angegebene Applikation durch, oder - wenn keine Applikation angegeben ist - aller geladenen Applikationen.
<code>reload [<Applikationsname> <Applikationsindex>] *</code>	Lädt die Bootapplikation der angegebene Applikation, oder - wenn keine Applikation angegeben ist - die Bootprojekte aller geladenen Applikationen.
<code>getprgstat [<Applikationsname> <Applikationsindex>] *</code>	Liefert den Programmstatus der angegebene Applikation, oder - wenn keine Applikation angegeben ist - den aller geladenen Applikationen.
<code>plcload</code>	Zeigt die Prozessorauslastung der Steuerung in Prozent.
<code>rtsinfo</code>	Liefert Informationen zum Laufzeitsystem, wie beispielsweise Prozessor und Runtime-System-Version.
<code>channelinfo</code>	Liefert Informationen zum Kommunikationskanal
<code>rtc-get</code>	Liefert die Weltzeit UTC über den String DateTime
<code>listpcicards [<VendorID>]</code>	Liefert eine Liste aller oder nach <VendorID> gefilterten PCI-Karten
<code>gettaskgroups</code>	Liefert eine Liste mit allen Taskgruppen, ihrer Tasks und die CPU-Kern Bindung
<code>cert-getapplist</code>	Zeigt alle registrierten angewendeten Zertifikate an (ID der Komponente und Verwendung)
<code>cert-genselfsigned [<number for search result by "cert-getapplist"> <expdays=>]</code>	Erzeugt selbstsignierte Zertifikate. Durch <code>expdays=</code> kann die Gültigkeitsdauer des Zertifikats angegeben werden. Der Default sind hier 365 Tage.
<code>cert-gendhparams [Länge in Bits]</code>	Erzeugt die Parameter für den Diffie-Hellman Schlüsselaustausch. Vorsicht: Dieser Vorgang kann mehrere Minuten dauern!

Kommando mit möglichen Parametern	Beschreibung
<code>cert-getcertlist [<trust level>]</code>	Listet alle Zertifikate der angegebenen Vertrauensstufe auf. Wenn keine Vertrauensstufe angegeben ist, werden alle Zertifikate aufgelistet. Mögliche Vertrauensstufen: • <code>untrusted</code>: nicht vertrauenswürdige Zertifikate • <code>trusted</code>: vertrauenswürdige Zertifikate • <code>own</code>: Zertifikate der Steuerung • <code>quarantine</code>: Zertifikate, deren Vertrauensstufe (vertrauenswürdig, nicht vertrauenswürdig) nicht durch eine Validierung geklärt werden konnte. Eingehende Verbindungen wurden daher abgelehnt.
<code>cert-createcsr [<number for search result by "cert-getapplist">]</code>	Erzeugt CSR-Dateien für alle Applikationen
<code>cert-import <trust level> <filename.cer></code>	Importiert das angegebene Zertifikat
<code>cert-export <trust level> [<number for search result by "cert-getcertlist">]</code>	Exportiert das angegebene Zertifikat
<code>cert-remove <trust level> <number for search result by "cert-getcertlist" or "all"></code>	Entfernt die angegebenen Zertifikate
<code>cpuload</code>	Zeigt die Prozessorauslastung der CPU (bei Multicore pro Prozessor-kern)
<code>gettaskgroups</code>	Liefert eine Liste der definierten Taskgruppen. Zu jeder Taskgruppe werden die zugewiesenen Tasks angezeigt.
<code>getmulticoreinfo</code>	Zeigt an, ob Multicore unterstützt wird und wie viele Prozessorkerne verfügbar sind.

Tab. 18-3: SPS Shell-Standardkommandos

*:

Applikationsname: Name der Applikation im Projektexplorer;

Applikationsindex: Ergibt sich aus der Liste aller Applikationen auf der Steuerung, die Sie mit Kommando `applist` abrufen können. Index 0 für die erste Applikation in der Liste, 1 für die zweite etc.

Siehe auch

- [Kap. 13.7 "SPS-Shell zur Abfrage von Informationen verwenden" auf Seite 226](#)
- [Kap. "Generischer Geräteeditor" auf Seite 519](#)
- [Kap. "Befehl 'Bibliotheksinformationen'" auf Seite 719](#)

Registerkarte '<Gerätename> E/A-Abbild'

Geräte mit E/A-Kanälen

Diese Registerkarte erscheint in Geräteeditoren für Geräte mit E/A-Kanälen. Sie zeigt die verfügbaren Kanäle und erlaubt die Abbildung der Eingangs-, Ausgangs- und Speicheradressen des Steuerungsgeräts auf Variablen oder ganze Funktionsbausteine der Applikation.

Damit erstellen Sie das sogenannte "E/A-Abbild" ("Mapping").

Welche Applikation die E/A-Behandlung übernehmen soll, definieren Sie in der Registerkarte **SPS-Einstellungen**.



Sie können das E/A-Abbild auch im Dialog **E/A-Abbild bearbeiten** erstellen. Dort erhalten Sie eine Mapping-Liste mit Such- und Filterfunktionen für einen kompletten Projektexplorer.



Abbilden "zu großer" Datentypen

Wenn Sie eine Variable eines Datentyps, der größer ist als ein Byte, auf eine Byte-Adresse abbilden, wird dort der Wert der Variablen auf Byte-Größe abgeschnitten!

Für das Monitoring des Variablenwerts im Dialog **E/A-Abbild** bedeutet dies:

Im Wurzelement 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.

Die Registerkarte enthält eine Tabelle zum Bearbeiten der E/A-Abbilds. Die dargestellten Informationen zu den Eingängen und Ausgängen stammen aus der Gerätebeschreibung.

Suche	Eingabefeld für eine Zeichenfolge, nach der die Mapping-Tabelle durchsucht werden soll. Die Suchergebnisse werden gelb markiert.
Filter	Auswahlliste mit Filtern für die in der Mapping-Tabelle dargestellten E/A Zuweisungen: • „Alles anzeigen“ • „Nur Ausgänge anzeigen“ • „Nur Eingänge anzeigen“ • „Nur nicht-gemappte Variablen anzeigen“ • „Nur gemappte Variablen anzeigen“ • „Nur Mappings auf bestehende Variablen anzeigen“ • „Nur Mappings auf neue Variablen anzeigen“
 FB für E/AKanal hinzufügen	Geräteabhängig verfügbar, wenn ein Kanaleintrag in der Mapping-Tabelle selektiert ist. Öffnet den Dialog "Funktionsbaustein auswählen" zur Auswahl des Bausteins, der mit dem Kanal direkt verknüpft werden soll.
 Gehe zu Instanz	Verfügbar, wenn ein Eintrag in der Mapping-Tabelle selektiert ist. Springt zum entsprechenden Eintrag in der Registerkarte <Gerätename> IEC-Objekte

Variable	Abhängig vom Gerät erscheinen die Eingänge und Ausgänge des Geräts als Knoten, darunter eingerückt die zugehörigen Kanäle, oder je nach Gerät nur die implizit angelegte Geräteinstanz. Das Symbol zeigt den Kanaltyp an:  : Eingang  : Ausgang Ein Doppelklick auf die Zelle öffnet ein Eingabefeld. - Möglichkeit 1: Die Variable existiert bereits; kompletten Pfad eintragen: <Applikationsname>.<Bausteinname>.<Variablenname>, Beispiel: <code>appl.PlcPrg.ivar</code>; Eingabehilfe über . - Möglichkeit 2: Die Variable existiert noch nicht; einfachen Namen eingeben; wird automatisch intern als globale Variable angelegt. Abhängig vom Gerät können Eingänge oder Ausgänge direkt mit einem Funktionsbaustein verknüpft werden. In diesem Fall ist die Schaltfläche  "FB für E/A-Kanal hinzufügen" verfügbar, siehe oben.
Mapping	Typ des Abbilds:  : Bereits existierende Variable  : Neue Variable
Kanal	Symbolischer Name des Kanals.
Adresse	Adresse des Kanals, beispielsweise <code>%IW0</code>. Durchgestrichene Adresse: Zeigt an, dass Sie dieser Adresse keine weitere Variable zuweisen sollten. Grund: Die hier angegebene Variable wird - als bereits bestehende - zwar an einem anderen Speicherplatz verwaltet, aber es könnten Mehrdeutigkeiten beim Schreiben der Werte entstehen, speziell bei Ausgängen.  : Zeigt an, dass diese Adresse bearbeitet und fixiert wurde. Wenn sich die Anordnung der Geräteobjekte im Projektplorer ändert, passt IndraLogic diese Adresse nicht automatisch an.
Typ	Datentyp des Kanals, beispielsweise <code>BOOL</code>. In der Gerätebeschreibung definierte Strukturen oder Bitfelder werden nur angezeigt, wenn sie Teil der IEC-Norm sind und als IEC-Datentyp in der Gerätebeschreibung gekennzeichnet sind. Ansonsten bleibt die Tabellenzelle leer. Beim Abbilden von strukturierten Variablen verhindert der Editor, dass Sie sowohl die Strukturvariable (beispielsweise auf <code>%QB0</code>) als auch einzelne Strukturelemente (beispielsweise auf <code>%QB0.1</code> und <code>%QB0.2</code>) eintragen können. Wenn es einen Haupteintrag mit einem Unterbaum von Bitkanal-Einträgen in der Mapping-Tabelle gibt gilt deshalb: Sie können entweder in der Zeile des Haupteintrags eine Variable eingeben, oder in den Zeilen der Unterelemente (Bitkanäle), aber nicht in beiden.

Standardwert	Standardwert des Parameters, der auf dem Kanal anliegt: Erscheint nur, wenn in den SPS-Einstellungen für das Verhalten der Ausgänge bei Stop die Option Alle Ausgänge auf Standardwert setzen aktiviert ist. Hinweis: Ab IndraWorks 14V22 wird für ein Mapping auf eine bestehende Variable automatisch der Initialisierungswert der Variablen als Standardwert verwendet. Das Feld "Standardwert" können Sie nur noch bearbeiten, wenn Sie ein Mapping auf eine neu erzeugte Variable vornehmen, oder wenn kein Mapping eingetragen ist. In älteren Versionen musste der Anwender explizit dafür sorgen, dass Standardwert und Initialisierungswert identisch sind.
Einheit	Einheit für den Parameterwert, beispielweise ms für Millisekunden.
Beschreibung	Kurzbeschreibung des Parameters.
Aktueller Wert	Istwert des auf dem Kanal anliegenden Parameters, erscheint nur im Online-Betrieb.

Mapping zurücksetzen	IndraLogic setzt die Mapping-Einstellungen auf die durch die Gerätebeschreibungsdatei definierten Standardwerte zurück.
Variablen aktualisieren	Festlegung für das Geräteobjekt bezüglich der Aktualisierung der E/A-Variablen. Der Standardwert ist in der Gerätebeschreibung definiert: • Einstellungen des übergeordneten Geräts verwenden: Aktualisierung gemäß den Einstellungen des übergeordneten Geräts. • Aktiviert 1 (Buszyklustask verwenden, wenn in keiner Task verwendet): IndraLogic aktualisiert die E/A-Variablen in der Buszyklustask, falls sie in keiner anderen Task verwendet werden. • Aktiviert 2 (immer in Buszyklustask): IndraLogic aktualisiert alle Variablen in jedem Zyklus der Buszyklustask, unabhängig davon ob sie verwendet werden und ob sie auf einen Ein- oder Ausgangskanal abgebildet sind.



Wenn eine UNION durch E/A-Kanäle im Mapping-Dialog vertreten ist, hängt es vom Gerät ab, ob auch ein Abbilden auf das Wurzelement möglich ist.

Siehe auch

- [Kap. "Generischer Geräteeditor" auf Seite 519](#)
- [Kap. "Befehl 'E/A-Abbild bearbeiten'" auf Seite 647](#)
- [Kap. "Registerkarte 'SPS-Einstellungen'" auf Seite 527](#)

Registerkarte '<Gerätename> IEC-Objekte'



Bitte beachten Sie, dass das manuelle Anlegen einer weiteren Instanz des Geräteobjekts zu Fehlfunktionen führen kann.

In dieser Registerkarte des generischen Geräteeditors sind "Objekte" aufgelistet, die einen Zugriff aus der IEC-Applikation auf das Gerät ermöglichen. Im Onlinebetrieb dient sie als Monitoring-Ansicht.

Bei Geräten, für die implizit eine Funktionsbausteininstanz erzeugt wird, erscheint mindestens diese Instanz als Objekt in der Tabelle. Diese Instanz kann beispielsweise verwendet werden, um aus der Applikation heraus einen Bus neu zu starten oder Informationen abzufragen.

Ob und mit welchen Zugriffsmöglichkeiten eine solche Geräteinstanz verfügbar ist, hängt vom Gerätetyp ab. Beachten Sie dazu bitte die Hilfe zur speziellen Gerätekonfiguration.

Auch Instanzen von Funktionsbausteinen, die mit Eingängen oder Ausgängen des Geräts verknüpft sind, werden hier dargestellt. Das Mapping eines Funktionsbausteins auf einen Kanal ist auf der Registerkarte "<Gerätename>

E/A-Abbild" definiert. Mit dem Befehl  "Gehe zu Instanz" gelangen Sie von dort direkt hierher zum betreffenden Objekt.

Zusätzlich können Sie in der Tabelle weitere Objekte anlegen, die zunächst noch nicht mit einem Gerätekanal verknüpft sind.

Im Onlinebetrieb können Sie die Tabelle der IEC-Objekte als Monitoring-Ansicht verwenden. Die Ansicht zeigt dann auch den aktuellen Wert, die Adresse und den Kommentar zur am Kanal anliegenden Funktionsbausteinvariablen, und bietet die Möglichkeit zum Schreiben und Forcen von Werten.

 Hinzufügen  Bearbeiten	Öffnet jeweils den Dialog "Funktionsbaustein auswählen" zum Anlegen einer neuen Instanz oder zum Bearbeiten der gerade in der Tabelle selektierten Instanz
 Löschen	Löscht den selektierten Eintrag
 Gehe zu Variable	Vom selektierten Eintrag springt man direkt zum zugehörigen Mapping in der Registerkarte "<Gerätename> E/A-Abbild"
"Variable"	Der Objektname setzt sich aus dem Gerätenamen und dem Funktionsbausteinnamen zusammen. Beispiel: EL2004_Relay. Ein Ändern des Gerätenamens im Projekt wirkt sich sofort entsprechend aus. Der Teil des Namens hinter dem Gerätenamen ist editierbar.
"Mapping"	Mapping-Typ, wie auf Registerkarte "<Gerätename> E/A-Abbild"
"Typ"	Datentyp, beispielsweise der Name des Funktionsbausteins
"Wert" "Vorbereiteter Wert" "Adresse" "Kommentar"	Nur im Onlinebetrieb verfügbar: Anzeige des aktuellen Werts, der Adresse und des Kommentars für die am Kanal anliegende Variable. Möglichkeit, einen Wert zum Schreiben oder Forcen der Variablen einzutragen.

Siehe auch

- [Kap. "Registerkarte '<Gerätename> E/A-Abbild'" auf Seite 534](#)

Registerkarte 'Benutzer und Gruppen'



Empfehlungen zur Datensicherheit

Um das Risiko von Datensicherheitsverletzungen zu minimieren, empfehlen wir die folgenden organisatorischen und technischen Maßnahmen für das System, auf dem Ihre Applikationen laufen:

Vermeiden Sie soweit als möglich, die SPS und Steuerungsnetzwerke offenen Netzwerken und dem Internet auszusetzen.

Verwenden Sie zum Schutz zusätzliche Sicherungsschichten wie ein VPN für Remote-Zugriffe und installieren Sie Firewall-Mechanismen. Beschränken Sie den Zugriff auf autorisierte Personen, ändern Sie evtl. vorhandene Standard-Passwörter bei der ersten Inbetriebnahme und auch weiterhin regelmäßig.

In dieser Registerkarte des generischen Geräteeditors bearbeiten Sie die "Online-Benutzerverwaltung" des Steuerungsgeräts.

Abhängig davon wie dies vom Gerät unterstützt wird, können Sie Benutzerkonten und Benutzergruppen definieren. In Kombination mit der Konfiguration auf der Registerkarte **Zugriffsrechte** steuern Sie damit den Zugriff auf Steuerungsobjekte und Dateien zur Laufzeit.

Voraussetzung: Die Steuerung hat eine Benutzerverwaltung und Sie haben Zugangsdaten, um sich auf der Steuerung einloggen zu können.

Grundsätzlich ist die Handhabung des Dialogs **Benutzer und Gruppen** dem der Projekt-Benutzerverwaltung sehr ähnlich. Es gibt sogar die Möglichkeit, Benutzerkonto-Definitionen aus der Projekt-Benutzerverwaltung zu "importieren".

Benutzer

In Baumstruktur werden alle aktuell definierten Benutzer und darunter ihre Mitgliedschaften zu den Benutzergruppen aufgelistet.

 Hinzufügen	Öffnet den Dialog Benutzer hinzufügen . Definieren Sie einen neuen Benutzernamen und ein Passwort. Bestätigen Sie das Passwort und erstellen mit Klicken auf OK das neue Benutzerkonto.
 Import	Öffnet den Dialog Benutzer importieren . Er zeigt alle in der Projekt-Benutzerverwaltung definierten Benutzerkonten. Wählen Sie die gewünschten Einträge aus und klicken Sie OK , um sie in die gerätespezifische Benutzerverwaltung zu übernehmen. ACHTUNG: Die Passwörter werden NICHT übernommen!
 Bearbeiten	Öffnet den Dialog Benutzer <Benutzername> bearbeiten . Er entspricht dem Dialog Benutzer hinzufügen und Sie können das Benutzerkonto verändern.
 Löschen	Löscht das Konto des gerade selektierten Benutzers.

Gruppen

In Baumstruktur werden alle aktuell definierten Gruppen und darunter die ihnen zugeordneten Benutzer aufgelistet.

 Hinzufügen	Öffnet den Dialog Gruppe hinzufügen . Definieren Sie einen neuen Gruppennamen. Wählen Sie aus der Liste der definierten Benutzer diejenigen, die der Gruppe angehören sollen. Bestätigen Sie die Auswahl mit OK . Die Gruppe erscheint im Baum.

 Import	Öffnet den Dialog Benutzer importieren . Er zeigt alle in der Projekt-Benutzerverwaltung definierten Benutzergruppen. Wählen Sie die gewünschten Einträge aus und klicken Sie OK , um sie in die geräte-spezifische Benutzerverwaltung zu übernehmen.
 Bearbeiten	Öffnet den Dialog Gruppe <Gruppenname> bearbeiten . Er entspricht dem Dialog Gruppe hinzufügen und Sie können die Gruppendefinition verändern.
 Löschen	Löscht die gerade selektierte Gruppe.

Siehe auch

- [Kap. 18.1.9 "Objekt 'Gerät und Generischer Geräteeditor' auf Seite 519](#)
- [Kap. "Generischer Geräteeditor" auf Seite 519](#)
- [Kap. "Registerkarte 'Zugriffsrechte'" auf Seite 540](#)

Registerkarte 'Zugriffsrechte'



Empfehlungen zur Datensicherheit

Um das Risiko von Datensicherheitsverletzungen zu minimieren, empfehlen wir die folgenden organisatorischen und technischen Maßnahmen für das System, auf dem Ihre Applikationen laufen:

Vermeiden Sie soweit als möglich, die SPS und Steuerungsnetzwerke offenen Netzwerken und dem Internet auszusetzen.

Verwenden Sie zum Schutz zusätzliche Sicherungsschichten wie ein VPN für Remote-Zugriffe und installieren Sie Firewall-Mechanismen. Beschränken Sie den Zugriff auf autorisierte Personen, ändern Sie eventuell vorhandene Standard-Passwörter bei der ersten Inbetriebnahme und auch weiterhin regelmäßig.

In dieser Registerkarte des Geräteeditors definieren Sie für Gerätebenutzer die Rechte für Zugriffe auf Objekte auf der Steuerung.

Voraussetzungen:

- Auf dem Steuerungsgerät muss eine Benutzerverwaltung eingerichtet sein.
- Die Option "Registerkarte "Zugriffsrechte" anzeigen" in den IndraLogic-Optionen muss aktiviert sein.

Aktivierung: IndraWorks Menü **Extras** ▶ **Optionen** ▶ **IndraLogic 2G** ▶ **Geräteeditor**



Diese IndraLogic-Option wird nicht von jedem Steuerungstyp unterstützt und kann gegebenenfalls durch die Gerätebeschreibung überschrieben werden!

Werkzeugleiste der Registerkarte

 Synchronisierung	Schaltet die Synchronisierung zwischen Editor und Benutzerverwaltung auf dem Gerät ein und aus. Wenn die Schaltfläche nicht "gedrückt" ist, ist der Editor leer, oder enthält eine Konfiguration, die Sie von der Festplatte geladen haben. Wenn die Schaltfläche "gedrückt" ist, synchronisiert IndraLogic die Anzeige im Editor laufend mit der aktuellen Benutzerverwaltung auf dem verbundenen Gerät. Wenn Sie die Synchronisierung einschalten, während der Editor eine noch nicht mit dem Gerät synchronisierte Benutzerkonfiguration enthält, erhalten Sie zunächst eine Abfrage, was mit dem Editorinhalt geschehen soll. Optionen: • "Vom Gerät hochladen und den Inhalt im Editor überschreiben": Die auf dem Gerät vorliegende Konfiguration wird in den Editor geladen und überschreibt den aktuellen Inhalt. • "Inhalt des Editors auf das Gerät laden und dort die Benutzerverwaltung überschreiben": Die im Editor enthaltene Konfiguration wird auf das Gerät übertragen und dort wirksam.
 Importieren von Festplatte	Öffnet den Standarddialog zum Auswählen und Importieren einer Benutzerverwaltungskonfiguration von der Festplatte. Wenn Sie die Schaltfläche in der Registerkarte "Benutzer und Gruppen" klicken, ist der Datei-Typ "Gerätebenutzerverwaltungsdateien (*.dum)" (Device User Management).
 Exportieren auf Festplatte	Öffnet den Standarddialog zum Speichern einer Datei auf der Festplatte. Sie legen damit die Benutzerverwaltungskonfiguration in einer XML-Datei ab. Der Datentyp ist "Geräte-Rechteverwaltungsdateien (*.drm)".
Gerätebenutzer:	Benutzername des gerade auf dem Gerät eingeloggten Benutzers.

Objekte

In der Baumstruktur sind die Objekte aufgelistet, auf die zur Laufzeit Aktionen ausgeführt werden können. Die Objekte sind jeweils ihrer Objektquelle zugeordnet und teilweise in Objektgruppen sortiert. Für ein selektiertes Objekt können Sie im Rechte-Fenster die Zugriffsmöglichkeit für eine Benutzergruppe konfigurieren.
Objektquelle (Knoten auf oberster Ebene im Objektbaum) • Dateisystemobjekte • Laufzeitsystemobjekte
Objektgruppen und Objekte (eingerückt). Beispielsweise: Device mit Kindobjekten Logger, PlcLogic, Settings, UserManagement.

Rechte

Die Tabelle gilt jeweils für das gerade im Baum ausgewählte Objekt. Sie zeigt für jede Benutzergruppe die aktuell konfigurierten Rechte für die möglichen Aktionen auf dieses Objekt:

- Hinzufügen/Entfernen
- Ändern
- Ansehen
- Ausführen

Die Symbole stellen die Zugriffsrechte dar:

-  : Zugriff (Aktion) ist explizit erlaubt.
-  : Zugriff (Aktion) ist explizit verboten.
-  : Ausgegrautes Plus- oder Minussymbol: Das Recht wurde vom Vaterobjekt übernommen.
-  : Das Recht wurde nicht explizit gesetzt oder verweigert, auch nicht im Vaterobjekt. Ein Zugriff ist nicht möglich.
- kein Symbol: Es sind mehrere Objekte ausgewählt, die keine einheitlichen Zugriffsrechte haben.

Mit einem Mausklick auf das Symbol ändern Sie das Recht.

Siehe auch

- [Kap. "Generischer Geräteeditor" auf Seite 519](#)
- [Kap. "Registerkarte 'Benutzer und Gruppen'" auf Seite 539](#)

Registerkarte 'Task-Aufstellung'

Dieser Unterdialog des Geräteeditors zeigt eine Tabelle der Ein- und Ausgänge sowie ihre Zuordnung zu den definierten Tasks.

Die Information ist erst sichtbar, nachdem für die Applikation Code erzeugt wurde. Sie dient der Fehlersuche, weil sie aufzeigt, wo Eingänge oder Ausgänge in mehreren Tasks mit unterschiedlichen Prioritäten verwendet werden. Mehrfache Verwendung kann zu undefinierten Werten durch Überschreiben führen.

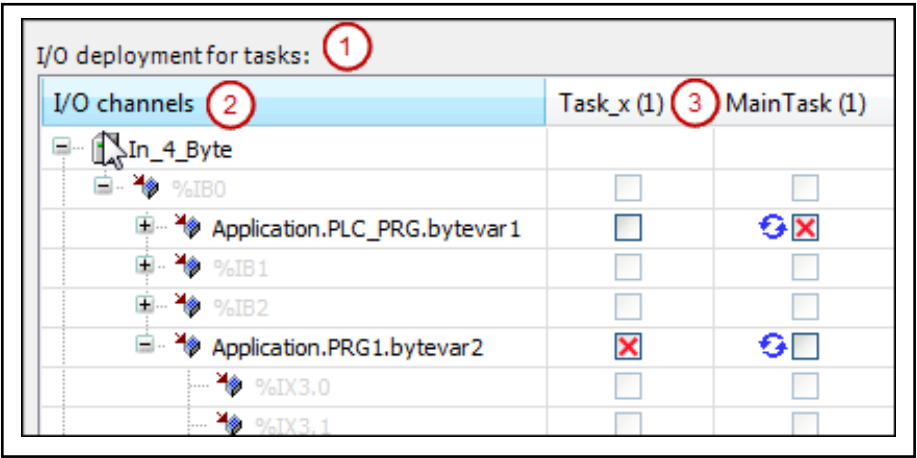


Abb. 18-8:

E/A-Kanäle (2)	Alle Eingänge und Ausgänge der angebundenen Geräte. Die Darstellung entspricht der im Dialog E/A-Abbild des Geräteeditors. Durch einen Doppelklick auf einen Eingang bzw. Ausgang können Sie den zugehörigen E/A-Abbild-Editor öffnen.
<Task-Name> (3)	Für jede in der Taskkonfiguration definierte Task erscheint eine Spalte. Im Titel stehen Taskname und Priorität. Von der ersten zur letzten Spalte sinkt die Priorität der Tasks. Für Eingänge und Ausgänge, die durch eine Task geschrieben oder gelesen werden, erscheint ein rotes Kreuz im Kästchen: . Zusätzlich ist an diesen Stellen die Task, die in den SPS-Einstellungen des Geräteeditors als Buszyklus-Task definiert ist, mit einem blauen Doppelpfeil-Symbol  gekennzeichnet. Nach einem Mausklick auf die Titelzelle werden nur die E/As angezeigt, welche dieser Task zugeordnet sind. Nach einem Mausklick auf die Zelle E/A-Kanäle werden danach wieder alle Kanäle eingeblendet.

Tab. 18-4: **E/A-Aufstellung für Tasks (1)**

Siehe auch

- [Kap. "Generischer Geräteeditor" auf Seite 519](#)
- [Kap. "Registerkarte '<Gerätename> E/A-Abbild'" auf Seite 534](#)
- [Kap. "Registerkarte 'SPS-Einstellungen'" auf Seite 527](#)
- [Kap. 9.15.1 "Erstellen einer Taskkonfiguration" auf Seite 144](#)

Registerkarte 'Status'

In dieser Registerkarte des generischen Geräteeditors erscheinen Status-Informationen, beispielsweise "Running" oder "Stopped", und spezifische Diagnosemeldungen aus dem jeweiligen Gerät, auch zur verwendeten Karte und dem internen Bussystem.

Siehe auch

- [Kap. "Generischer Geräteeditor" auf Seite 519](#)

Registerkarte 'Information'

In dieser Registerkarte des generischen Geräteeditors erscheinen allgemeine Informationen, die aus der Gerätebeschreibungsdatei stammen: Name, Hersteller, Kategorien, Version, Bestellnummer, Beschreibung, gegebenenfalls eine Abbildung.

Siehe auch

- [Kap. "Generischer Geräteeditor" auf Seite 519](#)

18.1.10 Objekt 'GlobalTextList'

Symbol: 

Das Objekt dient der Verwaltung und Übersetzung von Texten, die im Projekt in den Visualisierungen als statischer Text verfasst sind.

Das Objekt enthält eine Tabelle mit diesen Texten. Wenn Sie in einer Visualisierung in einem Element unter der Eigenschaft **Texte** einen Text verfassen, fügt IndraLogic automatisch eine Zeile in der Tabelle hinzu. Sie können hier keinen neuen Text verfassen, sondern nur einen bestehenden Text bearbeiten.

IndraLogic stellt außerdem folgende Befehle zur Verfügung, um die **Global-TextList** zu konsolidieren:

- **Visualisierungs-Text-IDs prüfen**
- **Visualisierungs-Text-IDs aktualisieren**
- **Nicht verwendete Textlisteneinträge entfernen**

Das Objekt liegt im Ordner "Allgemeiner Baustein" und ist höchstens einmal vorhanden.

ID	Eindeutiger Identifikator des Texts
Standard	Ausgangstext als Zeichenkette mit maximal einer Formatierungsangabe, zum Beispiel <code>Information A: %i Möglichkeiten</code> . Wenn unter einer Sprachspalte keine Übersetzung verfasst ist, verwendet IndraLogic diesen Text. Doppelklicken Sie in das Feld, um den Text zu editieren.
Die Tabelle enthält beliebig viele von Ihnen hinzugefügte Sprachspalten. Eine Sprachspalte ist mit einem Sprachkürzel betitelt, das Sie beim Erstellen der Spalte mit dem Befehl Sprache einfügen angegeben haben.	
<Sprachkürzel>	Name der Sprache als Sprachkürzel. Zum Beispiel <code>en-US</code> . Diese Spalte enthält die Übersetzung des Texts, der unter Standard verfasst ist. Wenn im Visualisierungsmanager das Sprachkürzel als Sprache ausgewählt ist, gibt eine Visualisierung im laufenden Betrieb die Übersetzung aus. Eine Visualisierung im laufenden Betrieb kann auf Anforderung eines Benutzers auf eine andere Sprache umschalten. Doppelklicken Sie in das Feld, um den Text zu editieren.

Siehe auch

- [Kap. "Befehl 'Sprache einfügen'" auf Seite 727](#)
- [Kap. "Befehl 'Sprache entfernen'" auf Seite 730](#)
- [Kap. "Befehl 'Import/Export Textlisten'" auf Seite 729](#)
- [Kap. "Befehl 'Globale Textliste anlegen'" auf Seite 727](#)
- [Kap. "Befehl 'Nicht verwendete Textlisteneinträge entfernen'" auf Seite 731](#)
- [Kap. "Befehl 'Visualisierungs-Text-IDs prüfen'" auf Seite 731](#)
- [Kap. "Befehl 'Visualisierungs-Text-IDs aktualisieren'" auf Seite 731](#)
- [Kap. 18.1.23 "Objekt 'Textliste'" auf Seite 588](#)
- [Kap. 9.8 "Text in einer Textliste verwalten" auf Seite 117](#)

18.1.11 Objekt 'GVL'- Globale Variablenliste

Symbol: 

Eine globale Variablenliste dient der Deklaration, der Bearbeitung und der Anzeige von globalen Variablen.

Sie fügen eine GVL mit dem Befehl **Hinzufügen ► Globale Variablenliste** der Applikation oder dem Ordner "Allgemeiner Baustein" hinzu.

- Wenn Sie eine GVL unter einer Applikation im Projektextplorer einfügen, sind die Variablen innerhalb dieser Applikation gültig.
- Wenn Sie eine GVL im Ordner "Allgemeiner Baustein" hinzufügen, sind die Variablen projektweit gültig.

Sie können im Dialog **Extras ► Optionen** in den Kategorien **Deklarationseditor** und **Texteditor** Einstellungen für den Editor des Objekts vornehmen.

GVL für konfigurierbare Konstanten ("Parameterliste") in Bibliotheken

Sie können die Variablen einer GVL in Netzwerkvariablen umwandeln, und somit für den Datenaustausch mit anderen Geräten im Netzwerk verwenden. Dazu müssen Sie entsprechende Eigenschaften für die GVL im Dialog **Eigenschaften**, Registerkarte **Netzwerkvariablen** definieren.

Es kann gewollt sein, dass man den Wert einer globalen Konstanten, die über eine Bibliothek bereitgestellt wird, durch einen von der Applikation definierten Wert ersetzen kann.

Für diesen Zweck muss die Konstante in der Bibliothek in einer "Parameterliste" deklariert sein. Ihr Wert kann dann, wenn die Bibliothek in der Applikation eingebunden ist, im Registerblatt "Parameterliste" des Bibliotheksverwalters der Applikation bearbeitet werden.

Handhabung von Parameterlisten

Eine Bibliothek `lib1.library` stellt eine Array-Variable `g_Array` bereit, deren Größe durch die globale Konstante `g_c_Arraysize` definiert ist.

Die Bibliothek wird in verschiedene Applikationen eingebunden, die jeweils eine andere Array-Größe verwenden möchten. Also soll die globale Konstante aus der Bibliothek mit einem applikationsspezifischen Wert überschrieben werden.

Wie ist vorzugehen:

Beim Erstellen von `lib1.library` wird die globale Konstante `g_c_Arraysize` innerhalb eines speziellen Typs von globaler Variablenliste (GVL) definiert, in der sogenannten "Parameterliste".

Dazu fügt man über Befehl **Hinzufügen ► Parameterliste** ein "Parameterliste"-Objekt, hier im Beispiel mit dem Namen "Param", ein.

Im Editor dieses Objekts, der identisch ist mit dem Standard-GVL-Editor, wird die Deklaration der Variablen `g_c_Arraysize` eingetragen.

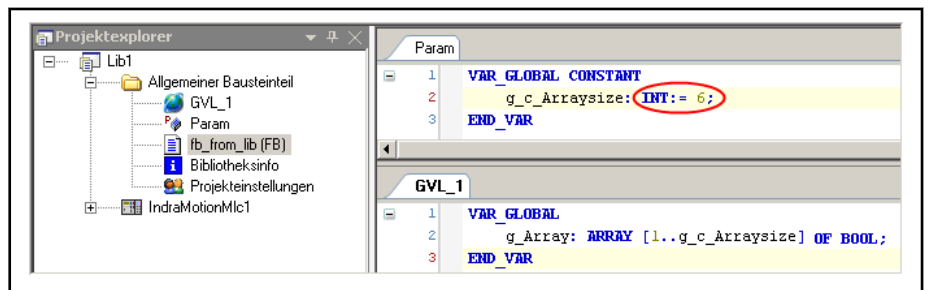


Abb. 18-9: Parameterliste Param in Bibliothek Lib1.library

Die Bibliothek `Lib1.library` wird erzeugt. Das Feld `GVL_1.g_Array` ist auf 6 Elemente voreingestellt.

Dann wird `lib1.library` im Bibliotheksverwalter eines Applikationsprojekts eingefügt:

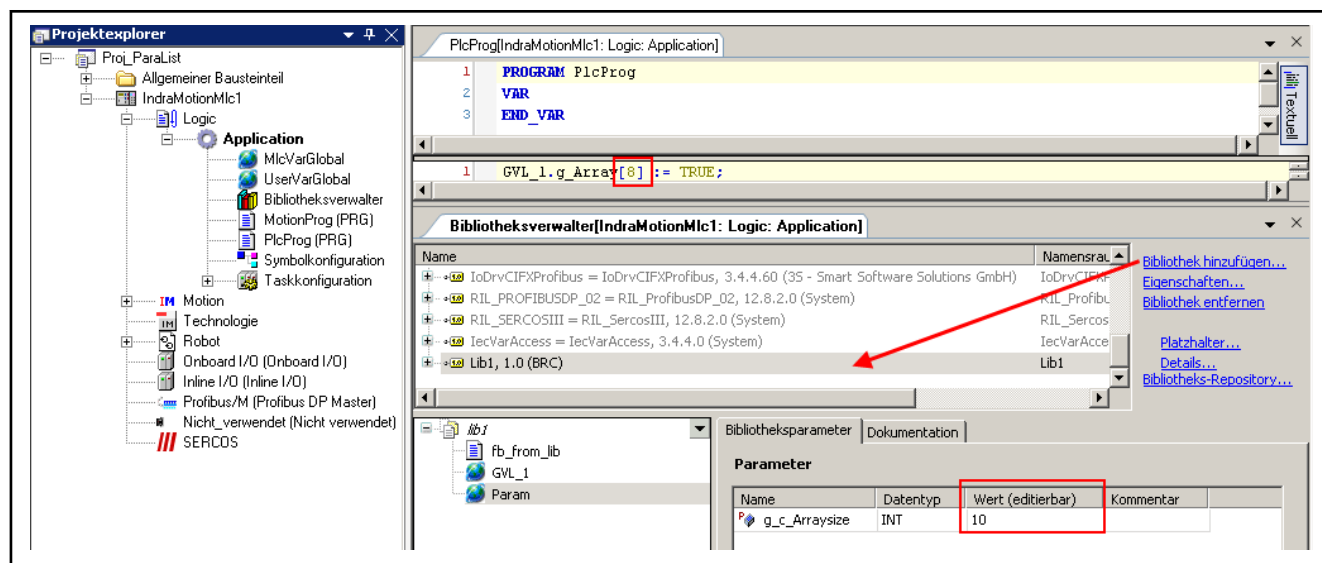


Abb. 18-10: Bearbeiten des Parameters *g_c_Arraysize* im Bibliotheksverwalter eines Projekts

Wählen Sie die Bibliothek im oberen Fenster des Bibliotheksverwalters aus, um den Bausteinbaum angezeigt zu bekommen.

Wählen Sie dort *Param* aus, um das Registerblatt "Parameterliste" zu erhalten, das die Deklarationen anzeigt.

Selektieren Sie die Zelle in Spalte „Wert (editierbar)“ und drücken die Leertaste um ein Eingabefeld zu öffnen. Geben Sie den gewünschten neuen Wert für *g_c_Arraysize* ein. Er wird im lokalen Gültigkeitsbereich der Bibliothek angewendet, sobald das Eingabefeld geschlossen wird.

Im obigen Beispiel ist der Wert jetzt auf 10 eingestellt, im Programm *PlcProg* wird Feldelement 8 belegt.



Bei geschachtelten Bibliotheken hat sich die parametrierbare Bibliothek zu oberst zu befinden.

Siehe auch

- Kap. 9 "Applikation programmieren" auf Seite 85
- Kap. 17.2.11 "Persistente Variable - PERSISTENT" auf Seite 308
- Kap. "Dialog 'Eigenschaften- 'Netzwerkvariablen'" auf Seite 751
- Kap. "Dialog 'Optionen- 'Deklarationseditor'" auf Seite 766
- Kap. "Dialog 'Optionen- 'Texteditor'" auf Seite 776

18.1.12 Objekt 'Persistente Variablenliste'

Symbol:

Das Objekt enthält die Deklaration von globalen persistenten Variablen im Deklarationsabschnitt `VAR_GLOBAL PERSISTENT RETAIN .. END_VAR`.

Die Variablen werden in einem speziellen nichtflüchtigen Speicher gesichert.

Der Persistenzeditor stellt die Variablen als Liste dar. Die dargestellte Liste wirkt sich nicht auf das Persistenzverhalten der Variablen aus, sondern nur die intern im Prozessabbild abgelegte Liste. Die Liste enthält alle jemals deklarierten Variablen in der Reihenfolge, wie sie historisch deklariert wurden. Variablen, die Sie entfernt haben, werden mit einem Platzhalter ersetzt und bestehen als Lücke weiter.

Der Deklarationsabschnitt kann außerdem Instanzpfade enthalten, die auf lokal deklarierte persistente Variablen verweisen und mit dem Befehl **Deklarationen ▶ Alle Instanzpfade hinzufügen** erzeugt wurden.



Vor der Entscheidung, auf welchem Weg Sie für eine Applikation Persistenz einrichten, ist es hilfreich, die in Kapitel Datenpersistenz beschriebenen Anwendungsfälle zu kennen. Außerdem ist es hilfreich, die Mechanismen von persistenten Variablen, Retain-Variablen, Variablen des Persistence Managers und Rezepturvariablen unterscheiden zu können.

Befehle Im Persistenzeditor stehen folgende Befehle zur Verfügung:

- Befehl **Deklarationen ▶ Alle Instanzpfade hinzufügen**
- Befehl **Deklarationen ▶ Liste neu ordnen und Lücken bereinigen**

Siehe auch

- [Kap. "Befehl 'Liste neu ordnen und Lücken bereinigen'" auf Seite 721](#)
- [Kap. 9.16.1 "Daten mit persistenten Variablen erhalten" auf Seite 148](#)
- [Kap. 9.16.2 "Daten mit Retain-Variablen erhalten" auf Seite 150](#)
- [Kap. "Befehl 'Alle Instanzpfade hinzufügen'" auf Seite 722](#)

18.1.13 Objekt 'Bildersammlung'

Das Objekt **Bildersammlung** beinhaltet eine Tabelle, in der Bilder IDs zugeordnet sind.

ID	ID des Bilds. Über diese ID referenzieren Sie zum Beispiel in der Visualisierung das Bild.
Dateiname	Dateipfad der Bilddatei. Wenn Sie auf die Schaltfläche  klicken, öffnet sich der Dialog Bildauswahl .
Bild	Zeigt eine verkleinerte Ansicht des Bilds.
Verknüpfungstyp	Öffnet den Dialog Bildauswahl . Darin definieren Sie die Art der Verknüpfung.

Dialog 'Bildauswahl'

Bilddatei	Verzeichnis und Name der Bilddatei, zum Beispiel. "C:\Programme\images\logo.bmp". IndraLogic unterstützt die Bildformate BMP, EMF, GIF, ICO, JPG, PNG, SVG and TIFF. Beachten Sie, dass eine Steuerung nicht immer alle Formate unterstützt. Die Verwendbarkeit von Bildern im skalierbaren Vektorformat *.svg hängt vom Betriebssystem ab. Notwendige Informationen stehen in den Gerätebeschreibungen Ihres Hardwareanbieters zur Verfügung.

Was soll mit der Bilddatei geschehen?

Speicherung der Verknüpfung	IndraLogic speichert lediglich die Verknüpfung. Somit aktualisiert IndraLogic eine Änderung der Bilddatei automatisch in der Bildersammlung. Sie müssen dabei sicherstellen, dass sich der Pfad der Bilddatei nicht ändert. Bei Speicherung des Projekts als Projektarchiv bettet IndraLogic die Bilddatei in das Projektarchiv ein.
Speicherung der Verknüpfung und Einbettung ins Projekt	IndraLogic kopiert das Bild in die Bildersammlung, die Verknüpfungsinformation bleibt jedoch erhalten. Somit kann IndraLogic erkennen, wenn sich die Bilddatei ändert und dann ggf. das Bild in der Bildersammlung aktualisieren. Das Verhalten wird durch die Optionen der nachfolgenden Tabelle geregelt. Eingebettete Bilddateien erhöhen den Speicherbedarf des Projekts.
Einbettung ins Projekt	IndraLogic kopiert das Bild in die Bildersammlung. Wenn sich die Bilddatei danach nochmal ändert, wird dies im Projekt nicht aktualisiert. Bei Bibliotheken müssen Sie die Bilder in das Projekt einbetten. Eingebettete Bilddateien erhöhen den Speicherbedarf des Projekts.

Wenn sich die Bilddatei ändert

Die Optionen sind nur verfügbar, wenn Sie in obiger Auswahl Speicherung der Verknüpfung und Einbettung ins Projekt gewählt haben.	
Die Datei automatisch neu laden	IndraLogic aktualisiert die Bilddatei automatisch, ohne Nachfragen, im Projekt.
Nachfragen, ob die Datei neu geladen werden soll	Wenn sich die Bilddatei geändert hat, können Sie durch eine Abfrage entscheiden, ob die Bilddatei im Projekt aktualisiert werden soll.
Nichts tun	IndraLogic aktualisiert die Bilddatei in der Bildersammlung nicht.

Siehe auch

- [Kap. 9.9 "Bildersammlungen verwenden" auf Seite 126](#)

18.1.14 Objekt 'Bibliotheksverwalter'

Symbol: 

Der Bibliotheksverwalter listet alle Bibliotheken auf, die zur Applikationserstellung im Projekt eingebunden wurden und informiert jeweils über ihren Typ, ihre Eigenschaften und ihren Inhalt.

Sie können in der Liste der eingebundenen Bibliotheken weitere hinzufügen oder welche entfernen, sowie deren Eigenschaften ändern.

Der Bibliotheksverwalter ist in 3 Fensterbereiche aufgeteilt.

- Oberer Fensterbereich: Liste der eingebundenen Bibliotheken
- Unterer, linker Fensterbereich: Baumstruktur mit allen Bausteinen einer im oberen Fensterbereich selektierten Bibliothek
- Unterer, rechter Fensterbereich: Dokumentation zu einem im Baum selektierten Baustein

Siehe auch

- [Kap. 16 "Bibliotheken erzeugen und verwalten" auf Seite 235](#)
- [Kap. 16.2 "Hinzufügen einer Bibliothek zur Applikation" auf Seite 237](#)

Liste der eingebundenen Bibliotheken

Liste aller aktuell im Projekt eingebundenen Bibliotheken. Wenn eine Bibliothek von anderen Bibliotheken abhängt (referenzierte Bibliotheken), werden diese referenzierten Bibliotheken automatisch eingebunden.

Darstellung in grauer Schrift	Die Bibliothek wurde automatisch durch ein Plug-in in das Projekt eingefügt.
Darstellung in schwarzer Schrift	Die Bibliothek wurde mit dem Befehl "Bibliothek hinzufügen" in das Projekt eingefügt.
"Name"	Darstellung der eingebundenen Bibliothek gemäß folgender Syntax: "<Platzhaltername> = <Bibliotheksname>, <Version> (<Firma>)": "<Platzhaltername>": Wenn es sich bei der Bibliothek um eine Platzhalterbibliothek handelt, steht der Platzhaltername vor einem = "<Bibliotheksname>": Name der Bibliothek, unter dem sie im Bibliotheksrepository verwaltet wird. "<Version>": Version, die beim erstmaligen Einbinden referenziert wurde. "(<Firma>)": Herstellerfirma (Optional)
Namensraum	Namensraum für einen eindeutigen Zugriff auf den Inhalt der Bibliothek. Der Namensraum wird dazu als Präfix vor einem Bausteinbezeichner verwendet: <Namensraum>.<Bibliotheksbausteinbezeichner> Üblicherweise stimmt der Namensraum mit dem Bibliotheksnamen überein. Hinweis: Wenn die Bibliothek die Eigenschaft <code>LanguageModelAttribute</code> "qualified-access-only" hat, müssen Sie im Applikationscode über den Namensraum auf den Bibliotheksbaustein zugreifen. Es wird ein qualifizierter (eindeutiger) Zugriff erzwungen. Sie können den Standardnamensraum für den lokalen Gebrauch (innerhalb des Projekts) im Dialog Eigenschaften ändern.
Effektive Version	Version der Bibliothek nach der Auflösung. Diese Version wird aktuell im Projekt verwendet. Voraussetzung: Wenn der Bibliotheksverwalter unter dem Geräteknoten "Application" liegt und eine Platzhalterbibliothek selektiert ist, dann wird hier die effektive Version dargestellt. Beispiel: 15.xx.0.0 Eine Platzhalterbibliothek, die unter einer Applikation eingebunden ist, wird folgendermaßen aufgelöst: Wenn Sie der Platzhalterbibliothek eine spezielle Auflösung im Dialog "Platzhalter" zugewiesen haben, wird die dort ausgewählte Bibliothek geladen. Andere Auflösungen werden ignoriert. Wenn keine spezielle Auflösung angegeben ist, wird erst in der Gerätebeschreibung, dann im Bibliotheksprofil der Applikation geprüft, ob eine Auflösung angegeben ist. Der erste Treffer ist gültig.
	Symbol, dessen Tooltip über die aktuelle geräteabhängige Auflösung informiert. Voraussetzung: Wenn der Bibliotheksverwalter im Ordner "Allgemeiner Baustein" liegt und eine Platzhalterbibliothek selektiert ist, dann erscheint das Symbol. Einer Platzhalterbibliothek, die im Ordner "Allgemeiner Baustein" eingebunden ist, wird folgendermaßen aufgelöst: Es wird applikationsabhängig geprüft, ob in deren Gerätebeschreibung eine Auflösung angegeben ist. Dann wird das Bibliotheksprofil geprüft. Die erste gefundene Auflösung wird verwendet. Wenn Sie der Platzhalterbibliothek eine spezielle Auflösung im Dialog "Platzhalter" zugewiesen haben, wird diese immer ignoriert. Das Ergebnis erscheint im Tooltip des Symbols
	Bibliothek, die mit einem vertrauenswürdigen Zertifikat signiert ist
	Bibliothek, die mit einem vertrauenswürdigen Zertifikat signiert ist, aber mindestens eine unsignierte Bibliothek referenziert
	Bibliothek, die mit einem privaten Schlüssel + Token signiert ist

	Bibliothek, die nicht signiert ist oder die mit einem nicht vertrauenswürdigen bzw. mit einem abgelaufenen Zertifikat signiert ist. Im Fall eines nicht vertrauenswürdigen Zertifikats steht im Kontextmenü der Befehl "Zertifikat vertrauen" zur Verfügung.
	Bibliothek, die als optional definiert ist und die aktuell nicht verfügbar ist
	Bibliothek, für die der Status gerade noch bestimmt wird
	Lizenzierte Bibliothek, für die gerade keine gültige Lizenz vorliegt
	Bibliothekssymbol für eine Bibliothek, die nicht geladen werden kann, weil ihre Signatur (Verschlüsselung) nicht verifiziert werden konnte

Befehle im Bibliotheksverwalter

 Bibliothek hinzufügen	Öffnet einen Dialog zum Auswählen einer Bibliothek. Es werden alle im Bibliotheks-Repository installierten Bibliotheken angeboten
 Bibliothek löschen	Entfernt die gerade selektierte Bibliothek aus dem Projekt
 Eigenschaften	Öffnet einen Dialog zur Anzeige und Bearbeitung der Eigenschaften der gerade selektierten Bibliothek.
 Details	Öffnet einen Dialog mit Details zur gerade selektierten Bibliothek (allgemeine Informationen, Inhalt, Eigenschaften, Lizenzinformation)
 Versuchen Bibliothek neu zu laden	Wenn Sie eine als nichtgefunden markierte Bibliothek selektieren, können Sie mit diesem Befehl ein erneutes Laden ins Projekt versuchen
 Download fehlender Bibliotheken	IndraLogic durchsucht die in den Projektoptionen angegebenen Download-Servern nach fehlenden Bibliotheken. Sie können danach einen Download durchführen und die Bibliothek installieren.
 Platzhalter	Dialog Platzhalter öffnet sich. Dort wird die aktuelle Auflösung dargestellt und kann bearbeitet werden
 Bibliotheks-Repository	Öffnet den Dialog Bibliotheks-Repository zum Installieren und Deinstallieren von Bibliotheken sowie zum Definieren von Bibliotheks-Ablageorten
 Icon-Legende	Öffnet den Dialog Information mit einer Legende zu den Icons, die in der Liste der eingebundenen Bibliotheken (siehe oben) den aktuellen Status einer Bibliothek anzeigen
 Zertifikat vertrauen	Nur im Kontextmenü einer im Bibliotheksverwalter selektierten Bibliothek, die mit einem nicht vertrauenswürdigen Zertifikat signiert wurde. Der Befehl macht das Zertifikat zu einem vertrauenswürdigen, das vorangestellte Icon ändert sich von  zu 
Bibliothek exportieren	Verfügbar im Kontextmenü einer im Bibliotheksverwalter selektierten Bibliothek. Öffnet den Standarddialog zum Speichern der Bibliotheksdatei im Dateisystem.

Siehe auch

- [Kap. "Befehl 'Bibliothek hinzufügen'" auf Seite 714](#)
- [Kap. "Befehl 'Eigenschaften'" auf Seite 716](#)
- [Kap. "Befehl 'Bibliotheks-Repository'" auf Seite 676](#)
- [Kap. "Befehl 'Platzhalter'" auf Seite 718](#)

Baumstruktur aller Bausteine einer selektierten Bibliothek

Alle Bibliotheksbausteine, die mit der Bibliothek eingebunden wurden, sind in Baumstruktur aufgelistet.

Voraussetzung: Eine Bibliothek ist im oberen Fensterbereich selektiert.



Die üblichen Sortier- und Suchfunktionen stehen in der Menüleiste zur Verfügung.

Dokumentation des im linken unteren Fensterbereich ausgewählten Bibliotheksbausteins

Registerkarte "Eingänge/ Ausgänge"	Schnittstelle (Eingänge/Ausgänge) des Bibliotheksbausteins
Registerkarte "Graphisch"	Graphische Darstellung der Schnittstelle
Registerkarte "Dokumentation"	Dokumentation des Bibliotheksbausteins Hinweis: Beachten sie Als Bibliotheksentwickler unbedingt die Regeln zur Dokumentationsmitgabe in "Richtlinien zur Bibliotheksentwicklung".
Registerkarte "Parameterliste"	Voraussetzung: Das Bibliotheksprojekt enthält eine Parameterliste. Sie können die Werte dieser Parameter in der Spalte "Wert (editierbar)" ändern.



Die Verwendung von Parameterlisten ist nur dann erlaubt, wenn sich die Bibliothek an oberster Stelle in der Hierarchie befindet.

Siehe auch

- [Kap. 16.1 "Informationen für Bibliotheksentwickler" auf Seite 236](#)

18.1.15 Objekt 'Globale Netzwerkvariablen-Liste (Sender)'

Symbol:

Eine Netzwerkvariablen-Liste (Sender) dient der Deklaration und Auflistung der globalen Variablen, die an Netzwerkvariablen-Listen (Empfänger) anderer Geräte oder Projekte des Netzwerks gesendet werden sollen.

Das Objekt wird im Projektexplorer mit dem Befehl **Hinzufügen ► Globale Netzwerkvariablenliste (Sender)** einer Applikation hinzugefügt.

Die Protokoll- und Übertragungsparameter können Sie im Dialog **Globale Netzwerkvariablen-Liste (Sender) hinzufügen** oder im Dialog **Eigenschaften** des Objekts in der Registerkarte **Netzwerkvariablen** konfigurieren.

Dialog 'Globale Netzwerkvariablen-Liste (Sender) hinzufügen'

Funktion: In diesem Dialog definieren Sie die Netzwerkeigenschaften für die Sender-NVL. Wenn Sie den Dialog schließen, fügt IndraLogic die Sender-NVL der Applikation im Projektexplorer hinzu.

Aufruf: Befehl **Hinzufügen ► Globale Netzwerkvariablen-Liste (Sender)**, wenn im Projektexplorer das Applikationsobjekt selektiert ist.

Der Dialog entspricht der Registerkarte **Netzwerkvariablen** im Dialog **Eigenschaften** des Objekts Netzwerkvariablen-Liste.

Siehe auch

- [Kap. 10.3 "Netzwerkvariablen" auf Seite 168](#)
- [Kap. "Dialog 'Eigenschaften- 'Netzwerkvariablen'" auf Seite 751](#)
- [Kap. 18.1.16 "Objekt 'Globale Netzwerkvariablen-Liste \(Empfänger\)'" auf Seite 552](#)
- [Kap. 10.3.1 "Konfigurieren eines Netzwerkvariablen-Austauschs" auf Seite 171](#)

18.1.16 Objekt 'Globale Netzwerkvariablen-Liste (Empfänger)'

Symbol: 

Das Objekt dient der Auflistung der empfangenen Netzwerkvariablen und der Anzeige der Informationen: Netzwerk- und Übertragungsinformationen und Sender.

Einer Applikation fügen Sie das Objekt mit dem Befehl **Hinzufügen ► Globale Netzwerkvariablenliste (Empfänger)** hinzu.

Die Netzwerkvariablen-Liste (Empfänger) listet die empfangenen Netzwerkvariablen auf.

Diese empfangenen Netzwerkvariablen wurden in der als Sender konfigurierten Netzwerkvariablen-Liste (Sender) eines anderen Geräts oder Projekts deklariert. Sie können die Netzwerkvariablen im Editor des Objekts nicht verändern.

Der Editor des Objekts besteht aus den 2 Bereichen:

- Informationen zum Sender und Übertragungsprotokoll der Liste
- Auflistung der Deklarationen der Netzwerkvariablen

Dialog 'Globale Netzwerkvariablen-Liste (Empfänger) hinzufügen'

Funktion: Der Dialog definiert die Empfänger-NVL zu einer Sender-NVL und fügt die Empfänger-NVL dem Applikationsobjekt im Projektnavigator hinzu.

Aufruf: Menübefehl **Hinzufügen ► Globale Netzwerkvariablen-Liste (Empfänger)**, wenn Sie das Applikationsobjekt selektiert haben.

Task	Task der aktuellen Applikation, die die zu empfangenen Variablen kontrolliert.
Sender	<i>Auswahlliste</i> • Zur Verfügung stehende Sender-NVLs eines anderen Geräts des Projekts • Import aus Datei: Notwendig, wenn die gewünschte Sender-NVL in einem anderen Projekt definiert ist. Dazu muss im anderen Projekt die gewünschte Sender-NVL im Eigenschaftendialog der NVL in der Registerkarte Verknüpfung mit Datei als GVL export file *.gvl erzeugt worden sein.
Import aus Datei	Angabe des Dateinamens im Format GVL export file *.gvl , wenn Sie bei Sender Import aus Datei ausgewählt haben.

Siehe auch

- [Kap. 10.3 "Netzwerkvariablen" auf Seite 168](#)
- [Kap. 10.3.1 "Konfigurieren eines Netzwerkvariablen-Austauschs" auf Seite 171](#)
- [Kap. 18.1.15 "Objekt 'Globale Netzwerkvariablen-Liste \(Sender\)'" auf Seite 551](#)

18.1.17 Objekt 'POU'

Symbol: 

Ein Objekt vom Typ **POU** ist eine Programm-Organisationseinheit (Programming Organization Unit) in einem IndraLogic-Projekt. In POUs schreiben Sie den Quellcode für Ihr Steuerungsprogramm.

Es gibt folgende Typen von POU:

- Programm
- Funktion
- Funktionsbaustein

Sie fügen ein Objekt **POU** über den Kontextmenü-Befehl **Hinzufügen** zur Applikation oder zum Ordner "Allgemeiner Baustein" ein. Beim Hinzufügen einer POU legen Sie den POU-Typ und die Implementierungssprache fest.

Andere Programmierobjekte (Methode, Aktion, usw.) können Sie diesen Objekten wiederum als Objekte hinzufügen.

POUs aufrufen

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

Beim Aufruf von POUs über den Namensraum durchsucht IndraLogic das Projekt nach der aufzurufenden POU gemäß folgender Reihenfolge:

1. Aktuelle Applikation
2. **Bibliotheksverwalter** der aktuellen Applikation
3. Ordner "Allgemeiner Baustein"
4. **Bibliotheksverwalter** im Ordner "Allgemeiner Baustein"



Wenn Sie eine POU aufrufen, die mit gleichem Namen sowohl in einer in der Applikation verwendeten Bibliothek, als auch als Objekt im Ordner "Allgemeiner Baustein" existiert, gilt Folgendes:

Es gibt keine Syntax, nach der Sie die POU im Ordner "Allgemeiner Baustein" nur über ihren Namen aufrufen können.

Bei Namensgleichheit von POUs im Ordner "Allgemeiner Baustein" und in einer Bibliothek fügen Sie den Namensraum der Bibliothek zum Namen der POU hinzu.

Siehe auch

- [Kap. 18.1.17 "Objekt 'POU'" auf Seite 552](#)
- [Kap. 18.3.6 "Dialog 'Eigenschaften'" auf Seite 748](#)

Dialog 'POU hinzufügen'

Funktion: Der Dialog dient der Konfiguration einer neuen POU im Sinne der Norm IEC 61131-3. Dies bedeutet, dass eine POU ein Programm, eine Funktion oder ein Funktionsbaustein sein kann.

Aufruf: Kontextmenü Applikation, Kontextmenü Ordner "Allgemeiner Baustein".

Name	Name der POU

Programm	
Funktionsbaustein	•  Erweitert: Angabe oder Auswahl eines Basis-Funktionsbausteins im Sinne der objektorientierten Programmierung. Wird mit Schlüsselwort EXTENDS in der Funktionsbausteindeklaration angegeben. •  Implementiert: Angabe oder Auswahl einer Schnittstelle im Sinne der objektorientierten Programmierung. Wird mit Schlüsselwort IMPLEMENTS in der Funktionsbausteindeklaration angegeben. •  Final: Abgeleiteter Zugriff ist nicht erlaubt. Das bedeutet, dass Sie den Funktionsbaustein nicht durch einen anderen Funktionsbaustein erweitern können. Dies ermöglicht optimierte Codegenerierung. •  Abstrakt: Kennzeichnet, dass der Funktionsbaustein eine fehlende oder unvollständige Implementierung hat und nicht instanziiert werden kann. Abstrakte FBs dienen ausschließlich als Basis-Funktionsbausteine und die Implementierung erfolgt typischerweise in einem abgeleiteten FB. • Zugriffsmodifizierer – PUBLIC: Entspricht der Angabe keines Zugriffsmodifizierers – INTERNAL: Der Zugriff auf den Funktionsbaustein ist auf den Namensraum (die Bibliothek) beschränkt. • Methoden-Implementierungssprache: Wenn Sie die Option Implementiert ausgewählt haben, können Sie hier eine Implementierungssprache für alle Methodenobjekte auswählen, die IndraLogic über die Implementierung der Schnittstelle erzeugt. Die Methoden-Implementierungssprache ist unabhängig von der Implementierungssprache des Funktionsbausteins.
Funktion	Hinweis: Nicht verfügbar, wenn in der Auswahlliste Implementierungssprache die Sprache Ablaufsprache (AS) ausgewählt ist. Rückgabebetyp: Auswahlliste für den Datentyp des Rückgabewerts
Implementierungssprache	Auswahlliste für die Implementierungssprache der POU

Siehe auch

- [Kap. "Objekt 'Funktionsbaustein' \(FUNCTION_BLOCK\)" auf Seite 556](#)
- [Kap. "Objekt 'Programm\(PROGRAM\)'" auf Seite 554](#)
- [Kap. "Objekt 'Funktion\(FUNCTION\)'" auf Seite 558](#)
- [Kap. 9.18.1 "Erweitern eines Funktionsbausteins" auf Seite 154](#)
- [Kap. 9.18.2 "Schnittstellen implementieren \(IMPLEMENTS\)" auf Seite 156](#)

Objekt 'Programm' (PROGRAM)

Ein Programm ist eine POU, die bei der Ausführung einen oder mehrere Werte liefert. Alle Werte bleiben nach einer Ausführung des Programms bis zur nächsten Ausführung erhalten. Die Aufrufreihenfolge der Programme innerhalb einer Applikation definieren Sie in Taskobjekten.

Sie fügen der Applikation oder dem Ordner "Allgemeiner Bausteinteil" ein Programm über den Befehl **Hinzufügen ► POU** hinzu.

Im Projektextplorer, unterhalb der Applikation bzw. im Ordner "Allgemeiner Bausteinteil", haben die Programm-POUs das Suffix (**PRG**).

Der Editor eines Programms besteht aus dem Deklarationsteil und dem Implementierungsteil.

Die oberste Zeile des Deklarationsteils enthält folgende Deklaration:

Programm:

```
PROGRAM <program>
```

Ein Programm aufrufen

Programme und Funktionsbausteine können ein Programm aufrufen. In einer Funktion ist ein Programmaufruf nicht erlaubt. Von Programmen gibt es keine Instanzen.

Wenn eine POU ein Programm aufruft und sich dadurch Werte des Programms verändern, bleiben diese Änderungen bis zum nächsten Programmaufruf erhalten. Die Werte des Programms bleiben auch dann erhalten, wenn der erneute Aufruf durch eine andere POU erfolgt.

Dies unterscheidet sich vom Aufruf eines Funktionsbausteins.

Beim Funktionsbaustein-Aufruf ändern sich nur die Werte der jeweiligen Instanz des Funktionsbausteins. Die Änderungen sind nur zu beachten, wenn eine POU dieselbe Instanz erneut aufruft.

Sie können für ein Programm die Eingabe- und/oder Ausgabeparameter auch direkt beim Aufruf setzen.

Syntax:

```
<program>(<input variable> := <value>, <output variable>
=> <value>):
```

Wenn Sie einen Programmaufruf über die Eingabehilfe einfügen und dabei in der Eingabehilfe die Option **Mit Argumenten einfügen** aktiviert ist, fügt IndraLogic dem Programmaufruf Eingabe- und/oder Ausgabeparameter gemäß der Syntax zu.

Beispiele

Aufrufe:

AWL:

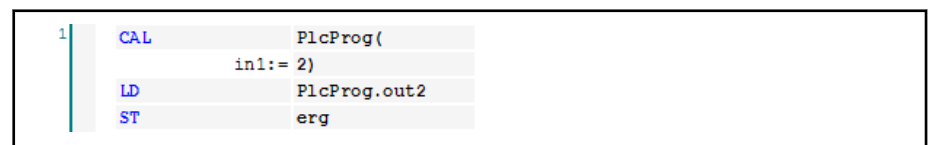


Abb. 18-11:

Mit Zuweisung der Parameter:

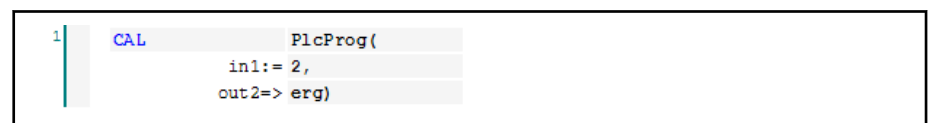


Abb. 18-12:

ST:

```
PlcProg()
erg := PlcProg.out2;
```

Mit Zuweisung der Parameter:

```
PlcProg(in1:=2, out1=>erg);
```

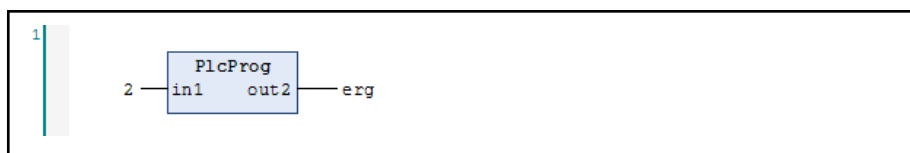


Abb. 18-13:

Siehe auch

- [Kap. 18.1.17 "Objekt 'POU'" auf Seite 552](#)
- [Kap. 9.15 "Taskkonfiguration" auf Seite 143](#)

Objekt 'Funktionsbaustein' (FUNCTION_BLOCK)

Ein Funktionsbaustein ist eine POU, die bei der Ausführung einen oder mehrere Werte liefert.

Sie fügen das Objekt der Applikation oder dem Ordner "Allgemeine Bausteinteile" über den Befehl **Hinzufügen ► POU** hinzu.

Im Projektextplorer oder in dem Ordner "Allgemeine Bausteinteile" haben Funktionsbaustein-POUs das Suffix **(FB)**.

Sie rufen einen Funktionsbaustein immer über eine Instanz auf, die eine Kopie des Funktionsbausteins ist.

Der Editor eines Funktionsbausteins besteht aus dem Deklarationsteil und dem Implementierungsteil.

Die Werte der Ausgabevariablen und der internen Variablen bleiben nach einer Ausführung bis zur nächsten erhalten. Dies bedeutet, dass der Funktionsbaustein bei mehrmaligem Aufruf mit denselben Eingabevariablen nicht unbedingt dieselben Ausgabewerte liefert.

Zusätzlich zu der in der IEC 61131-3 beschriebenen Funktionalität können Sie Funktionsbausteine in IndraLogic auch für folgende Funktionalitäten der objektorientierten Programmierung verwenden:

- Erweitern eines Funktionsbausteins
- Implementieren von Schnittstellen
- Methoden
- Eigenschaften

Die oberste Zeile des Deklarationsteils enthält folgende Deklaration:

```
FUNCTION_BLOCK <access specifier> <function block> | EXTENDS <function block> |
    IMPLEMENTS <comma-separated list of interfaces>
```

Einen Funktionsbaustein aufrufen

Der Aufruf erfolgt immer über eine Instanz des Funktionsbausteins. Beim Aufruf eines Funktionsbausteins ändern sich nur die Werte der jeweiligen Instanz

Deklaration der Instanz:

```
<instance>: <function_block>;
```

Auf eine Variable des Funktionsbausteins greifen Sie im Implementierungsteil wie folgt zu:

```
<instance>.<variable>
```



Beachten Sie Folgendes:

- Sie können nur auf Eingabe- und Ausgabevariablen eines Funktionsbausteins von außerhalb der Funktionsbaustein-Instanz zugreifen, nicht auf die internen Variablen.
- Der Zugriff auf eine Funktionsbaustein-Instanz ist auf die POU begrenzt, in der die Instanz deklariert ist, außer Sie haben die Instanz global deklariert.
- Sie können beim Aufruf der Instanz den Funktionsbausteinvariablen die gewünschten Werte zuweisen.

Beispiel

Zugriff auf Funktionsbausteinvariablen:

Der Funktionsbaustein **FB1** hat die Eingabevariable `iVar1` vom Typ `INT` und die Ausgabevariable `out1`.

Im Folgenden wird die Variable `iVar1` aus dem Programm **Prog** aufgerufen.

```
PROGRAM Prog
VAR
  inst1:FB1;
END_VAR
```

```
inst1.iVar1 := 33; (* FB1 is called and the value 33 is assigned to the variable iVar1 *)
inst1();          (* FB1 is called, that's necessary for the following access *)
ires := inst1.out1 (* the output variable out1 of the FB1 is read *)
```

In FUP:

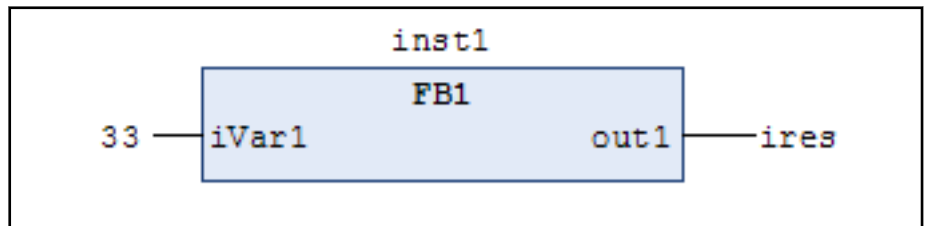


Abb. 18-14:

Variablenwerte beim Aufruf zuweisen:

In den textuellen Sprachen AWL und ST können Sie Werte beim Aufruf des Funktionsbausteins direkt an Eingabe- und/oder Ausgabevariablen zuweisen.

Die Zuweisung eines Werts an einen Eingabevariable erfolgt mit `:=`

Die Zuweisung eines Wert an eine Ausgabevariable erfolgt mit `=>`

Beispiel

Die Instanz `CMD_TMR` des Timer-Funktionsbausteins wird mit Zuweisungen für die Eingabevariable `IN` und `PT` aufgerufen. Anschließend wird die Ausgabevariable `Q` des Timers der Variablen `A` zugewiesen

```
CMD_TMR (IN := %IX5, PT := 100=);
A:=CMD_TMR.Q
```



Wenn Sie eine Funktionsbaustein-Instanz über die **Eingabehilfe** einfügen und im Dialog **Eingabehilfe** die Option **Mit Argumenten einfügen** aktiviert ist, fügt IndraLogic den Aufruf mit allen Eingabe- und Ausgabevariablen ein.

Sie müssen dann nur die gewünschten Wertzuweisung einfügen.

Im obigen Beispiel fügt IndraLogic den Aufruf wie folgt ein:

```
CMD_TMR (IN:= , PT:= , Q=> ).
```

Siehe auch

- [Kap. 18.1.17 "Objekt 'POU'" auf Seite 552](#)
- [Kap. 9.18.1 "Erweitern eines Funktionsbausteins" auf Seite 154](#)
- [Kap. 9.18.2 "Schnittstellen implementieren \(IMPLEMENTS\)" auf Seite 156](#)
- [Kap. "Objekt 'Methode' " auf Seite 564](#)
- [Kap. "Objekt 'Eigenschaft'" auf Seite 571](#)

Objekt 'Funktion' (FUNCTION)

Eine Funktion ist eine POU, die bei der Ausführung genau ein Datenelement liefert und dessen Aufruf in textuellen Sprachen als Operator in Ausdrücken vorkommen kann. Das Datenelement kann auch ein Array oder eine Struktur sein.

Sie fügen das Objekt der Applikation oder dem Ordner "Allgemeiner Bausteinteil" über den Befehl **Hinzufügen ► POU** hinzu.

Im Projektextplorer unterhalb der "Application" oder im Ordner "Allgemeiner Bausteinteil" haben Funktions-POUs das Suffix **(FUN)**.



Funktionen haben keine interne Statusinformation, das bedeutet, dass Funktionen die Werte ihrer Variablen nicht bis zum nächsten Aufruf speichern.

Aufrufe einer Funktion mit den selben Eingabevariablen-Werten liefern immer denselben Ausgabewert. Deshalb dürfen Funktionen keine globalen Variablen und Adressen verwenden!

Der Editor einer Funktion besteht aus dem Deklarationsteil und dem Implementierungsteil.

Die oberste Zeile des Deklarationsteils enthält folgende Deklaration:

Syntax

```
FUNCTION <function> : <data type>
```

Darunter deklarieren Sie die Eingabe- und Funktionsvariablen.

Die Ausgabevariable einer Funktion ist der Funktionsname.



Wenn Sie eine lokale Variable in einer Funktion als **RETAIN** deklarieren, hat dies keinen Effekt!

In diesem Fall gibt IndraLogic einen Compilerfehler aus.



In IndraLogic 2G können Sie explizite und implizite Parameterzuweisungen in Funktionsaufrufen nicht mischen.

Das bedeutet dass Sie entweder nur explizite oder nur implizite Parameterzuweisungen in Funktionsaufrufen verwenden.

Die Reihenfolge der Parameterzuweisungen beim Funktionsaufruf ist beliebig.

Eine Funktion aufrufen

In ST können Sie den Aufruf einer Funktion als Operand in Ausdrücken verwenden.

In AS können Sie einen Funktionsaufruf nur innerhalb von Schritttaktionen oder Transitionen verwenden.

Beispiele

Funktion mit Deklarationsteil und einer Zeile Implementierungs-Code

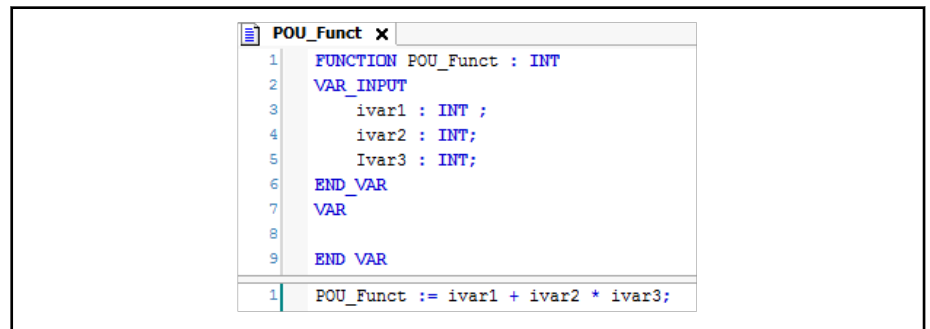


Abb. 18-15:

Funktionsaufrufe:

ST:

```
result := POU_Funct(5,3,22)
```

AWL:

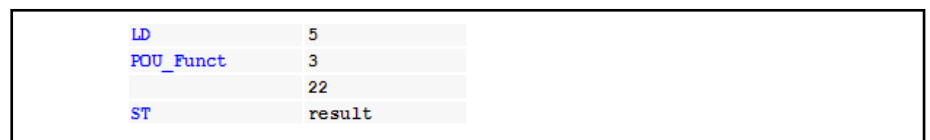


Abb. 18-16:

FUP:

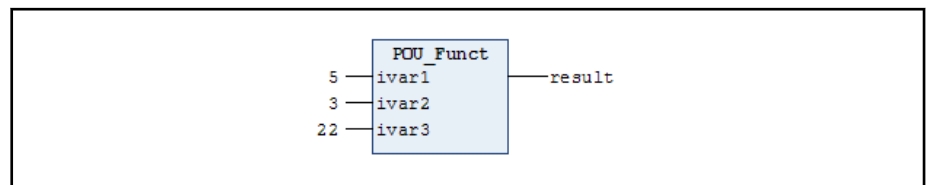


Abb. 18-17:

Funktionen mit zusätzlichen Ausgängen

Nach der Norm IEC 61131-3 können Funktionen zusätzliche Ausgänge haben. Die zusätzlichen Ausgänge deklarieren Sie in der Funktion zwischen den Schlüsselwörtern `VAR_OUTPUT` und `END_VAR`. Die Funktion rufen Sie gemäß folgender Syntax auf:

Syntax

```
<function>(<function output variable1> => <output variable 1>, <function output variable n> => <output variable n>)
```

Beispiel

Die Funktion `fun` ist mit zwei Eingabevariablen `in1` und `in2` definiert. Der Ausgabevariablen der Funktion `fun` wird auf die lokal deklarierten Ausgangsvariablen `loc1` und `loc2` geschrieben.

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

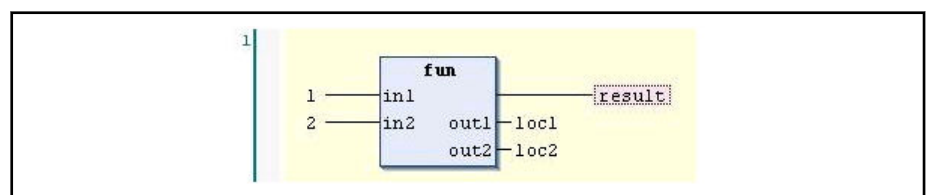


Abb. 18-18: Funktion mit Rückgabewert und zwei Ausgangsvariablen

Übergabe von Instanzdaten per VAR_INPUT

Die Deklaration einer FB-Instanz ist innerhalb einer Funktion nicht erlaubt, jedoch können Daten einer FB-Instanz per VAR_INPUT zur weiteren Verarbeitung in eine Funktion übergeben werden:

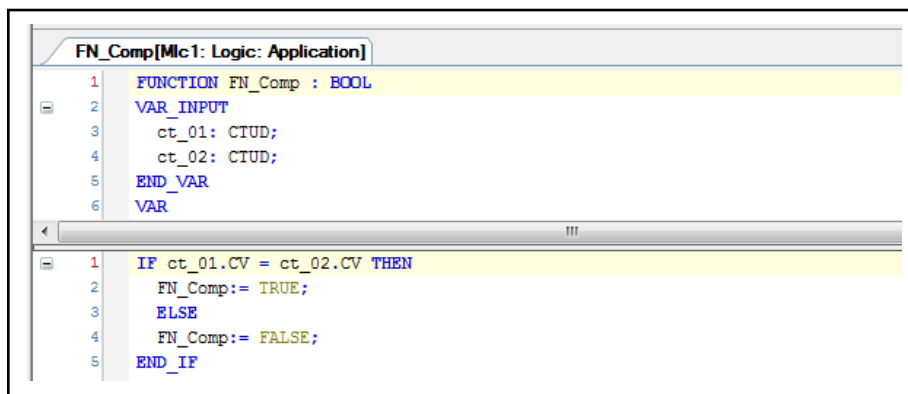


Abb. 18-19: Übergabe von zwei Instanzen per VAR_INPUT an eine Funktion

Im obigen Beispiel werden die Datensätze von zwei (Zähler-)Instanzen an die Funktion FN_Comp zum Vergleich der CV-Ausgänge übergeben. Die Bearbeitung der Instanzen selbst erfolgt in der übergeordneten POU.

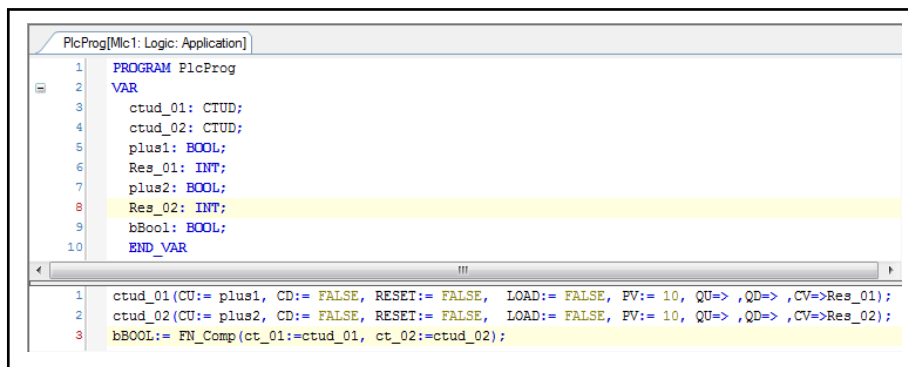
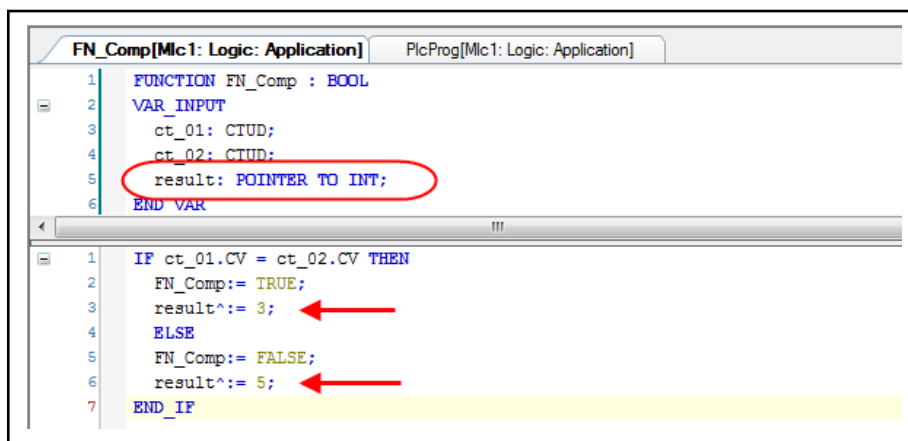


Abb. 18-20: Bearbeitung der Zähler-Instanzen und Übergabe ihrer Datensätze an die Funktion

Übergabe per POINTER in VAR_INPUT

Informationen, die in einer POU (Funktion, Funktionsbaustein) berechnet wurden, können über einen Pointer, der unter VAR_INPUT deklariert wurde, nach außen gegeben werden.



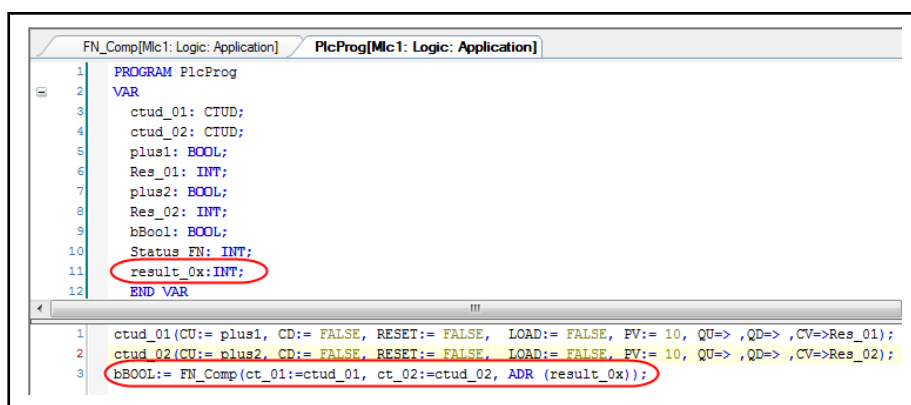


Abb. 18-22: Übernahme der Information per ADR-Operator

HINWEIS

Wenn Sie Pointer auf Adressen verwenden, können sich beim Online-Change Inhalte von Adressen verschieben!

Wenn Sie direkte Adressen verwenden, verschieben sich Inhalte von Adressen beim Online-Change nicht!

Siehe auch [Kap. "Befehl 'Online-Change'" auf Seite 657](#).

Siehe auch:

- [Kap. 17.5.12 "Datentyp 'POINTER'" auf Seite 383](#)

Übergabe per **REFERENCE TO**
in **VAR_INPUT**

Informationen, die in einer POU (Funktion, Funktionsbaustein) berechnet wurden, können über **REFERENCE TO**, die unter **VAR_INPUT** deklariert wurde, nach außen gegeben werden.

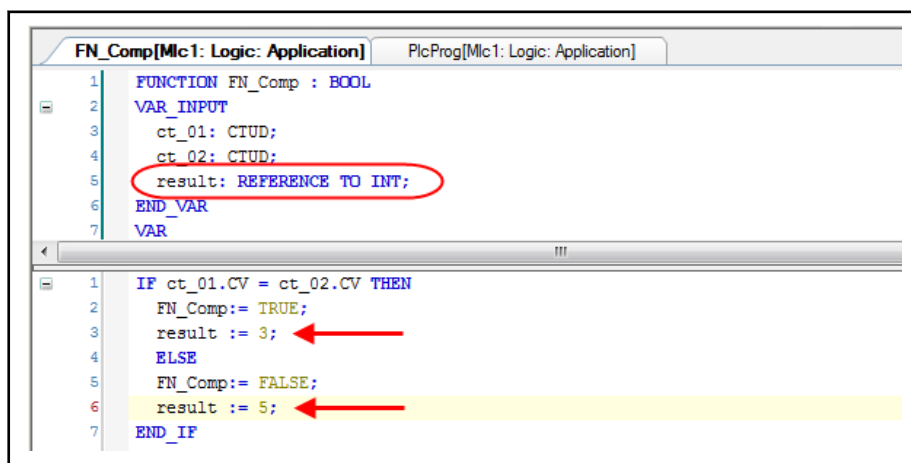


Abb. 18-23: REFERENCE TO an der Schnittstelle einer Funktion

FN_Comp[Mic1: Logic: Application]

PlcProg[Mic1: Logic: Application]

```
1 PROGRAM PlcProg
2 VAR
3   ctud_01: CTUD;
4   ctud_02: CTUD;
5   plus1: BOOL;
6   Res_01: INT;
7   plus2: BOOL;
8   Res_02: INT;
9   bBool: BOOL;
10  Status_FN: INT;
11  result_0x: INT;
12 END VAR

1 ctud_01(CU:= plus1, CD:= FALSE, RESET:= FALSE, LOAD:= FALSE, PV:= 10, QU=> ,QD=> ,CV=>Res_01);
2 ctud_02(CU:= plus2, CD:= FALSE, RESET:= FALSE, LOAD:= FALSE, PV:= 10, QU=> ,QD=> ,CV=>Res_02);
3 bBOOL:= FN_Comp(ct_01:=ctud_01, ct_02:=ctud_02, result:= result_0x);
```

Abb. 18-24: Übernahme der Information per Zuweisung

HINWEIS

Wenn Sie Pointer auf Adressen verwenden, können sich beim Online-Change Inhalte von Adressen verschieben!

Wenn Sie direkte Adressen verwenden, verschieben sich Inhalte von Adres-
sen beim Online-Change nicht!
Siehe auch [Kap. "Befehl 'Online-Change'" auf Seite 657](#).

- Siehe auch
- [Kap. 17.5.13 "Datentyp 'REFERENCE'" auf Seite 384](#)
- Siehe auch
- [Kap. 18.1.17 "Objekt 'POU'" auf Seite 552](#)

Objekt 'Schnittstelle' (INTERFACE)

Symbol: 

Schlüsselwort: INTERFACE

Eine Schnittstelle ist ein Mittel der objektorientierten Programmierung. Das Objekt `ITF` beschreibt ein Set von Methoden- und Eigenschaft-Prototypen. Prototyp bedeutet in diesem Zusammenhang, dass die Methoden und Eigen-
schaften nur Deklarationen und keine Implementierung enthalten.

Auf diese Weise können Sie verschiedene Funktionsbausteine, die gemein-
same Eigenschaften haben, gleichartig nutzen.

Ein Objekt `ITF` fügen Sie der Applikation oder dem Ordner "Allgemeiner Bau-
steinteil" mit dem Befehl **Hinzufügen ▶ Schnittstelle** hinzu.

Vererbung	
Name	Schnittstellename
Erweitert	 Erweitert die Schnittstelle, die Sie im Eingabefeld eingeben oder über die Eingabehilfe  auswählen. Dies bedeutet, dass alle Methoden der Schnittstelle, die die neue Schnittstelle erweitert, auch in der neuen Schnittstelle verfügbar sind.

Tab. 18-5: Schnittstelle hinzufügen

Dem Objekt `ITF` können Sie die Objekte **Schnittstellen-Eigenschaft** und **Schnittstellen-Methode** hinzufügen.

Schnittstellen-Methode dürfen nur die Deklarationen von Eingabe-, Ausgabe-, und VAR_IN_OUT-Variablen, aber keine Implementierung enthalten.

Damit Sie eine Schnittstelle auch im Programm verwenden können, muss es einen Funktionsbaustein geben, der diese Schnittstelle implementiert.

Das bedeutet:

- der Funktionsbaustein enthält in seinem Deklarationsteil die Schnittstelle in seiner IMPLEMENTS-Liste
- der Funktionsbaustein enthält für alle Methoden und Eigenschaften-Prototypen der Schnittstelle eine Implementierung

Ein Funktionsbaustein kann eine oder mehrere Schnittstellen implementieren. Sie können die gleiche Methode mit identischen Parametern, aber unterschiedlichem Implementierungscode in verschiedenen Funktionsbausteinen verwenden.

Beachten Sie Folgendes

- Sie dürfen keine Variablen innerhalb einer Schnittstelle deklarieren. Eine Schnittstelle hat keinen Implementierungsteil und keine Aktionen. Nur eine Sammlung von Schnittstellen-Methoden ist definiert und in diesen dürfen Sie nur VAR_INPUT-, VAR_OUTPUT- und VAR_IN_OUT-Variablen definieren.
- IndraLogic behandelt Variablen, die mit dem Typ einer Schnittstelle deklariert sind, immer als Referenzen.
- Ein Funktionsbaustein, der eine Schnittstelle implementiert, muss Implementierungs-Code für die Methoden der Schnittstelle enthalten. Die Methoden haben Sie genauso benannt wie in der Schnittstelle und die Methoden enthalten dieselben VAR_INPUT-, VAR_OUTPUT- und VAR_IN_OUT-Variablen wie in der Schnittstelle.



Schnittstellenreferenzen und Online-Change

Mit Compiler-Version < 3.4.1.0 kann Folgendes passieren: Wenn ein Funktionsbaustein seine Daten ändert, weil Variablen hinzugefügt oder gelöscht werden, oder weil sich der Typ von Variablen ändert, dann kopiert IndraLogic alle Instanzen des Funktionsbausteins an einen neuen Speicherplatz. In diesem Fall verweist eine Schnittstellenreferenz aber nicht auf den neuen Speicherplatz, sondern immer noch auf den alten.

Mit Compiler-Versionen >= 3.4.1.0 adressiert IndraLogic die Schnittstellen-Referenzen automatisch um, so dass IndraLogic auch im Falle eines Online-Change die korrekte Schnittstelle referenziert. Dafür benötigt IndraLogic zusätzlichen Code und mehr Zeit, sodass abhängig von der Anzahl der betroffenen Objekte Jitter-Probleme auftreten können.

Deshalb zeigt IndraLogic vor Ausführung des Online-Change die Anzahl der betroffenen Variablen und Schnittstellen-Referenzen an, und Sie entscheiden, ob der Online-Change durchgeführt oder abgebrochen wird.

Beispiel:

Definition einer Schnittstelle und Verwendung in einem Funktionsbaustein

Sie haben die Schnittstelle **ITF** unterhalb der Applikation eingefügt.

Die Schnittstelle enthält die Schnittstellen-Methoden **METHod1** und **METHod2**.

ITF, METHod1 und METHod2 enthalten keinen Implementierungscode. Nur im Deklarationsteil der Methoden fügen Sie die benötigten Variablendeklarationen ein.

Wenn Sie anschließend einen Funktionsbaustein im Projektextplorer einfügen, der die Schnittstelle ITF implementiert, fügt IndraLogic automatisch auch die Methoden METHod1 und METHod2 unter dem Funktionsbaustein ein (FUNCTION_BLOCK POU IMPLEMENTS ITF).

Hier können Sie in den Methoden funktionsbausteinspezifischen Code implementieren. Die **Schnittstellen-Methoden** werden damit zu **POU-Methoden**.

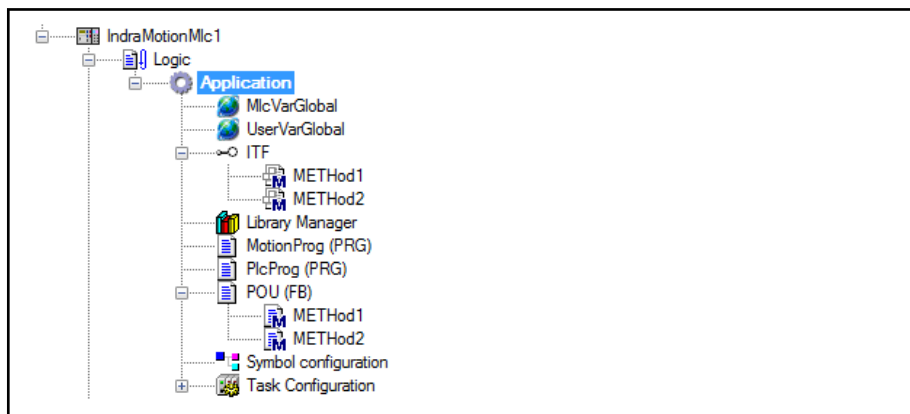


Abb. 18-25:

- [Kap. 9.18.2 "Schnittstellen implementieren \(IMPLEMENTS\)" auf Seite 156](#)
- [Kap. 9.18.3 "Erweitern einer Schnittstelle" auf Seite 158](#)
- [Kap. "Objekt 'Schnittstellenmethode'" auf Seite 567](#)
- [Kap. "Objekt 'Schnittstelleneigenschaft'" auf Seite 568](#)
- [Kap. "Objekt 'Methode' " auf Seite 564](#)
- [Kap. "Objekt 'Eigenschaft'" auf Seite 571](#)

Objekt 'Methode'

Symbol:

Schlüsselwort: METHOD

Methoden sind eine Erweiterung der Norm IEC 61131-3 und ein Mittel der objektorientierten Programmierung, das der Datenkapselung dient. Eine Methode enthält eine Deklaration und eine Implementierung. Eine Methode ist jedoch im Gegensatz zu einer Funktion kein unabhängiger Programmierbaustein, sondern ist einem Funktionsbaustein oder Programm untergeordnet. Eine Methode kann auf alle gültigen Variablen des übergeordneten Programmierbausteins zugreifen.

Zur Organisation von Methoden können Sie Schnittstellen verwenden.

Methode hinzufügen

Methoden können an Programme, oder Funktionen hinzugefügt werden.

Wählen Sie hierzu im jeweiligen Kontextmenü den Befehl **Hinzufügen ► POU-Methode**.

Der Dialog "Methode hinzufügen" öffnet sich.

Deklaration

- Die Variablen einer Methode enthalten temporäre Daten, die nur während der Ausführung der Methode gültig sind (Stack- Variablen). Alle

Variablen, die in einer Methode deklariert und implementiert sind, werden bei jedem Aufruf der Methode neu initialisiert.

- Methoden können wie Funktionen zusätzliche Ausgänge haben. Diese zusätzlichen Ausgänge müssen Sie im Methodenaufruf zuweisen.
- Je nach deklariertem Zugriffsmodifizierer kann eine Methode nur innerhalb des eigenen Namensraums (**INTERNAL**) , nur innerhalb des eigenen Programmierbausteins und seinen Ableitungen (**PROTECTED**) oder nur innerhalb des eigenen Programmierbausteins (**PRIVATE**) aufgerufen werden. Bei **PUBLIC** kann die Methode überall aufgerufen werden.

Schnittstellenmethoden können Eingabe-, Ausgabe- und **VAR_IN_OUT**-Variablen deklariert haben, enthalten aber keine Implementierung.

Siehe auch

- [Kap. "Objekt 'Schnittstellenmethode'" auf Seite 567](#)

Implementierung

- In der Implementierung der Methode ist der Zugriff auf Funktionsbaustein-Instanzen oder auf Programmvariablen erlaubt
- Der **THIS**-Pointer ermöglicht den Zugriff auf die eigene Funktionsbaustein-Instanz. Der Pointer ist somit nur in Methoden erlaubt, die unter einem Funktionsbausteins sind.
- Eine Methode kann nicht auf **VAR_TEMP**-Variablen des Funktionsbausteins zugreifen
- Eine Methode kann sich selbst rekursiv aufrufen

Dialog „Methode hinzufügen“

Funktion: Definiert eine Methode, die mit Beenden des Dialogs unterhalb des selektierten Programmierbausteins angeordnet wird.

Aufruf: Über das Kontextmenü eines Programms (PRG), oder einer Funktion (FUN)**Hinzufügen ► POU-Methode**

Name	Name der Methode, Beispielsweise <code>meth_DoIt</code>
Rückgabetyt	Standardtyp oder strukturierter Typ des Rückgabewerts. Beispiel: <code>BOOL</code>

Implementierungssprache	Beispiel: Strukturierter Text (ST)
Zugriffsmodifizierer	Regelt den Zugriff auf die Daten. • "PUBLIC" oder keine Angabe: Der Zugriff ist nicht eingeschränkt. • "PRIVATE": Der Zugriff ist auf das Programm, den Funktionsbaustein oder die GVL beschränkt. Im POU- oder Gerätebaum ist das Objekt mit (<code>private</code>) gekennzeichnet. Die Deklaration enthält das Schlüsselwort <code>PRIVATE</code>. • "PROTECTED": Der Zugriff ist auf das Programm, die GVL oder den Funktionsbaustein mit seinen Ableitungen beschränkt. Die Deklaration enthält das Schlüsselwort <code>PROTECTED</code>. Im POU- oder Gerätebaum ist das Objekt mit (<code>protected</code>) gekennzeichnet. • "INTERNAL": Der Zugriff auf die Methode ist auf den Namensraum (Bibliothek) beschränkt. Im POU- oder Gerätebaum ist das Objekt mit (<code>internal</code>) gekennzeichnet. Die Deklaration enthält das Schlüsselwort <code>INTERNAL</code>.

Spezielle Methoden für einen Funktionsbaustein

<code>FB_Init</code>	Deklarationen automatisch implizit. Auch explizite Deklaration möglich. Enthält Initialisierungscode für den Funktionsbaustein, wie im Deklarationsteil des Funktionsbausteins definiert ist.
<code>FB_Reinit</code>	Explizite Deklaration notwendig. Aufruf, nachdem die Instanz des Funktionsbausteins kopiert wurde (wie während eines Online-Change), und re-initialisiert das neue Instanzmodul.
<code>FB_Exit</code>	Explizite Deklaration notwendig. Aufruf für jede Instanz des Funktionsbausteins vor einem erneuten Download oder einem Reset oder während eines Online-Change für alle verschobenen oder gelöschten Instanzen.
POU-Eigenschaften und Schnittstellen-Eigenschaften (PROPERTY)	Bestehen jeweils aus einer <code>Set</code>- und/oder <code>Get</code>-Accessor-Methode.

Eingabeunterstützung beim Erzeugen von erbenden Bausteinen



Wenn Sie objektorientiert programmieren und die Vererbung von Bausteinen nutzen wollen, erhalten Sie folgendermaßen Unterstützung:

Wenn Sie unterhalb eines von einem Basisbaustein abgeleiteten Baustein eine Methode, Aktion etc. einfügen, erhalten Sie im Dialog "Objekt hinzufügen" eine Combobox mit einer Auswahl aller im Basisbaustein verwendeten Methoden, Aktionen etc.

Somit können Sie beispielsweise eine Methodendefinition der Basis auf einfache Weise zunächst übernehmen, um sie dann entsprechend für die abgeleitete Funktion des Bausteins anzupassen.

Methoden und Eigenschaften mit dem Zugriffsmodifizierer PRIVATE erscheinen in dieser Auswahl nicht, da sie nicht vererbt werden sollten.

Methoden und Eigenschaften mit dem Zugriffsmodifizierer PUBLIC erhalten beim Übernehmen in den abgeleiteten Baustein automatisch ein leeres Zugriffsmodifizierersfeld, was funktional das Gleiche bedeutet.

Siehe auch

- [Kap. 9.18.1 "Erweitern eines Funktionsbausteins" auf Seite 154](#)
- [Kap. 9.18 "Objektorientiert programmieren" auf Seite 154](#)

Siehe auch

- [Kap. "Objekt 'Schnittstelle\(INTERFACE\)' auf Seite 562](#)
- [Kap. 9.18.4 "Methodenaufruf" auf Seite 158](#)
- [Kap. 9.18.2 "Schnittstellen implementieren \(IMPLEMENTS\)" auf Seite 156](#)
- [Kap. "Objekt 'Schnittstelleneigenschaft'" auf Seite 568](#)
- [Kap. "Attribut 'monitoring'" auf Seite 419](#)
- [Kap. "ST-Editor im Online-Betrieb" auf Seite 242](#)
- [Kap. 13.1.2 "Überwachungslisten verwenden" auf Seite 211](#)
- [Kap. 17.10 "Methode 'FB_Init', 'FB_Reinit' und 'FB_Exit'" auf Seite 455](#)
- [Kap. 17.2.9 "Instanzvariablen - VAR_INST" auf Seite 307](#)

Objekt 'Schnittstellenmethode'

Symbol: 

Das Objekt dient der objektorientierten Programmierung.

Das Objekt **Schnittstellenmethode** fügen Sie einer Schnittstelle über den Befehl **Hinzufügen** hinzu.

Wenn eine Methode unterhalb einer Schnittstelle eingefügt ist, können Sie in dieser Methode nur Variablendeklarationen (Eingabe-, Ausgabe- und EinAusgabe-Variablen) hinzufügen und instanzieren.

Der Methode können Sie erst dann Programm-Code hinzufügen, wenn ein Funktionsbaustein die Schnittstelle, zu der die Methode gehört, "implementiert". IndraLogic fügt die Methode dann unterhalb des Funktionsbausteins ein.

Siehe auch

- [Kap. "Objekt 'Schnittstelle\(INTERFACE\)' auf Seite 562](#)
- [Kap. "Objekt 'Methode' " auf Seite 564](#)
- [Kap. 9.18.2 "Schnittstellen implementieren \(IMPLEMENTS\)" auf Seite 156](#)

Objekt 'Schnittstelleneigenschaft'

Symbol: 

Schnittstelleneigenschaften sind eine Erweiterung der Norm IEC 61131-3 und ein Mittel der objektorientierten Programmierung. Eine Schnittstelleneigenschaft deklariert die Accessormethoden `Get` und `Set` (kein Implementierungscode). Ein Funktionsbaustein, der eine Schnittstelle implementiert, erbt damit auch deren Schnittstelleneigenschaften.

Sie können im Projektextplorer bei einer Schnittstelle eine Schnittstelleneigenschaft hinzufügen, dann wird die Schnittstelle um die Accessormethoden `Get` und `Set` erweitert. Der `Get`-Accessor dient dem Lesezugriff. Der `Set`-Accessor dient dem Schreibzugriff. Sie können einen nicht benötigten Accessor löschen. Um einen Accessor hinzuzufügen, verwenden Sie den Befehl Projekt → Objekt hinzufügen → Schnittstelleneigenschaft. Dann öffnet sich der Dialog Schnittstelleneigenschaft hinzufügen.

Siehe auch:

- [Kap. "Objekt 'Schnittstelle\(INTERFACE\)' auf Seite 562](#)
- [Kap. "Objekt 'Eigenschaft'" auf Seite 571](#)

Anlegen einer Schnittstelleneigenschaft

Zum Anlegen öffnen Sie im Projektextplorer das Kontextmenü des Schnittstellenobjekts, dem Sie eine Schnittstelleneigenschaft hinzufügen wollen. Mit dem den Befehl **Hinzufügen ► Schnittstellen-Eigenschaft...** wird die Schnittstelle um die Accessormethoden `Get` und `Set` erweitert.

Der `Get`-Accessor dient dem Lesezugriff.

Der `Set`-Accessor dient dem Schreibzugriff.

Nicht benötigte Accessoren können gelöscht werden.

Um einen gelöschten Accessor wieder hinzuzufügen, markieren Sie die Schnittstelleneigenschaft und verwenden den Befehl

Hinzufügen ► Schnittstellen-Set-Zugriffdefinition... für einen `Set` Accessor, oder

Hinzufügen ► Schnittstellen-Get-Zugriffdefinition... für einen `Get` Accessor.

Anwendungsbeispiel

Deklaration und Implementierung der Schnittstelleneigenschaft `Literal_A:`

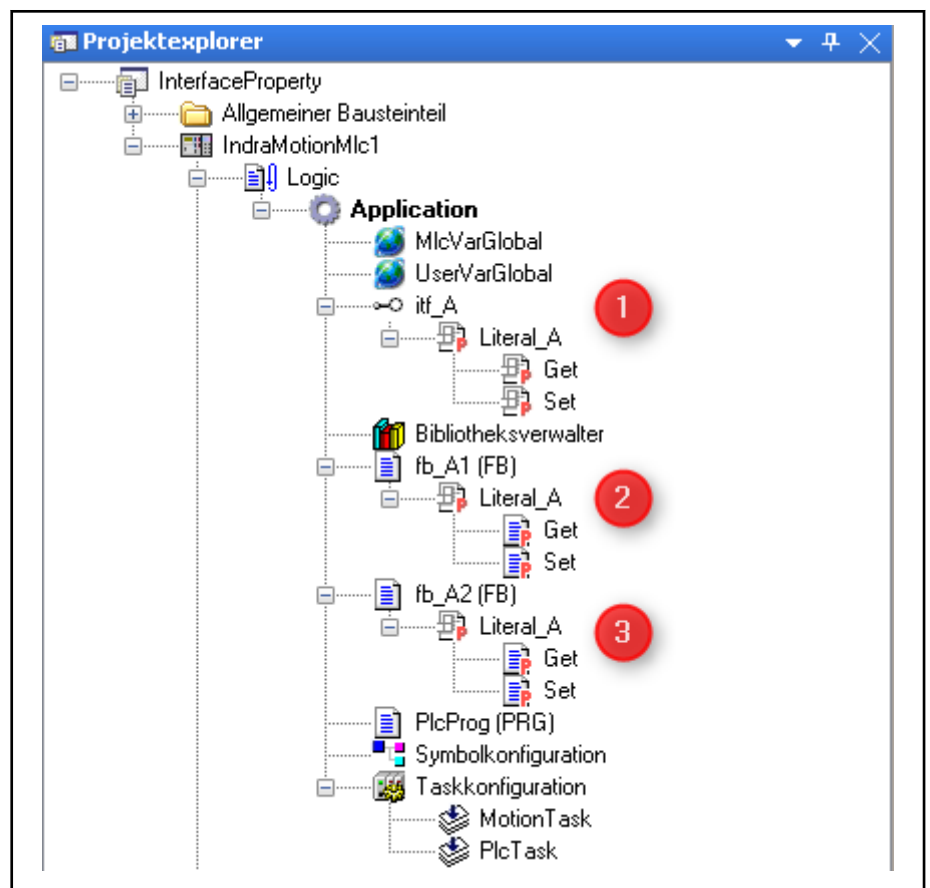


Abb. 18-26:

(1) Der Schnittstelle "itf_A" wurde im Projektexplorer die Schnittstelleneigenschaft "Literal_A" hinzugefügt (in diesem Beispiel mit dem Rückgabotyp `STRING`).

(2) & (3) Den beiden Funktionsbausteinen "fb_A1" und "fb_A2" wurde die POU-Eigenschaft "Literal_A" hinzugefügt (in diesem Beispiel mit dem Rückgabotyp `STRING` und der Implementierungssprache "Strukturierter Text").

Die Schnittstelle "itf_A" hat die Eigenschaft "Literal_A" mit den Accessormethoden "Get" und "Set".

Die Funktionsbausteine "fb_A1" und "fb_A2" implementieren die Schnittstelle "itf_A" und erben damit deren Schnittstelleneigenschaft.

Folgend, die Implementierung für Schnittstelle, Schnittstelleneigenschaft, Funktionsbausteine und Programm:

Schnittstelle itf_A

```
INTERFACE itf_A
VAR
END_VAR
```

Schnittstelleneigenschaft Literal_A

```
PROPERTY Literal_A : STRING
```

FB fb_A1

```
FUNCTION_BLOCK fb_A1 IMPLEMENTS itf_A
VAR
  str_1 : STRING;
  str_2 : STRING;
```

```

    iCnt : INT;
END_VAR

iCnt := iCnt + 1;
str_1 := 'Function block A1';

```

Accessor fb_A1.Literal_A.Get

```

VAR
END_VAR

Literal_A := CONCAT (str_1, ' and property.');
```

Accessor fb_A1.Literal_A.Set

```

VAR
END_VAR

str_2 := Literal_A;
```

FB fb_A2

```

FUNCTION_BLOCK fb_A2 IMPLEMENTS itf_A
VAR
    str_1 : STRING;
    str_2 : STRING;
    iCnt : INT;
END_VAR

iCnt := iCnt + 1;
str_1 := 'Function block A2';

```

Accessor fb_A2.Literal_A.Get

```

VAR
END_VAR

Literal_A := str_1;
```

Accessor fb_A2.Literal_A.Set

```

VAR
END_VAR

str_2 := Literal_A;
```

Programm PLC_PRG

```

PROGRAM PLC_PRG
VAR
    iCnt : INT;
    my_1 : fb_A1;
    my_2 : fb_A2;
    strName_1 : STRING;
    strName_2 : STRING;
END_VAR

iCnt := iCnt + 1;
my_1();
my_2();
strName_1 := my_1.Literal_A;
strName_2 := my_2.Literal_A;
my_1.Literal_A := 'Hello 1';
my_2.Literal_A := 'World 2';

```

Nachdem das Beispielprojekt übersetzt und auf die Steuerung übertragen wurde, werden zur Laufzeit der Applikation folgende Werte in PLC_PRG gemonitort:

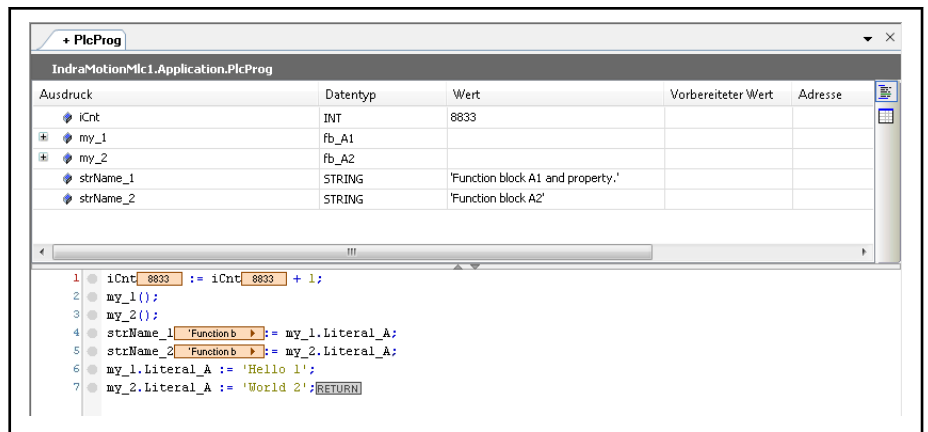


Abb. 18-27:

Objekt 'Eigenschaft'

Symbol:

Schlüsselwort: PROPERTY

Eigenschaften sind eine Erweiterung der Norm IEC 61131-3 und ein Mittel der objektorientierten Programmierung. Eigenschaften dienen der Datenkapselung, da sie von außen Zugriff auf Daten ermöglichen und gleichzeitig als Filter fungieren. Dafür stellt eine Eigenschaft die Accessormethoden „Get“ und „Set“ bereit, die Lese- und Schreibzugriff auf die Daten der Instanz ermöglichen, unter der die Eigenschaft angeordnet ist.

Unterhalb eines Programms, oder eines Funktionsbausteins kann im Projektfexplorer eine Eigenschaft mit Accessormethoden hinzugefügt werden. Öffnen Sie hierzu das Kontextmenü der jeweiligen POU und wählen den Befehl **Hinzufügen ► POU-Eigenschaft...**

Der Dialog "Eigenschaft hinzufügen" öffnet sich.

Siehe auch

- [Kap. 17.10 "Methode 'FB_Init', 'FB_Reinitund 'FB_Exit'" auf Seite 455](#)
- [Kap. "Objekt 'Schnittstelleneigenschaft'" auf Seite 568](#)

Dialog Eigenschaft hinzufügen

Funktion: Konfigurieren der Eigenschaft, die mit Beenden des Dialogs hinzugefügt wird.

Aufruf: POU Kontextmenü, Befehl **Hinzufügen ► POU-Eigenschaft...**

Voraussetzung: Im Projektfexplorer ist ein Programm (PRG), oder ein Funktionsbaustein (FUNCTION_BLOCK) selektiert.

Name	Name (Bezeichner) der Eigenschaft
Rückgabebetyp	Standardtyp oder strukturierter Typ des Rückgabewerts. Beispiel: BOOL

Implementierungssprache	Beispiel: Strukturierter Text (ST)
Zugriffsmodifizierer	<i>Regelt den Zugriff auf die Daten.</i> • PUBLIC oder keine Angabe: Der Zugriff ist nicht beschränkt. • PRIVATE: Der Zugriff ist auf das Programm, oder den Funktionsbaustein beschränkt. Im Projektextplorer ist das Objekt mit (private) gekennzeichnet. Die Deklaration enthält das Schlüsselwort <code>PRIVATE</code>. • PROTECTED: Der Zugriff ist auf das Programm, oder den Funktionsbaustein mit seinen Ableitungen beschränkt. Im Projektextplorer ist das Objekt mit (protected) gekennzeichnet. Die Deklaration enthält das Schlüsselwort <code>PROTECTED</code>. • INTERNAL: Der Zugriff ist auf den Namensraum (Bibliothek) beschränkt. Im Projektextplorer ist das Objekt mit (internal) gekennzeichnet. Die Deklaration enthält das Schlüsselwort <code>INTERNAL</code>.

Editor 'Eigenschaft'

Im Editor können Sie den Datenzugriff programmieren. Der Code darf zusätzliche lokale Variablen enthalten. Aber er darf keine zusätzlichen Eingangsvariablen oder (im Gegensatz zu einer Funktion oder Methode) keine zusätzlichen Ausgangsvariablen enthalten.

Beispiel:

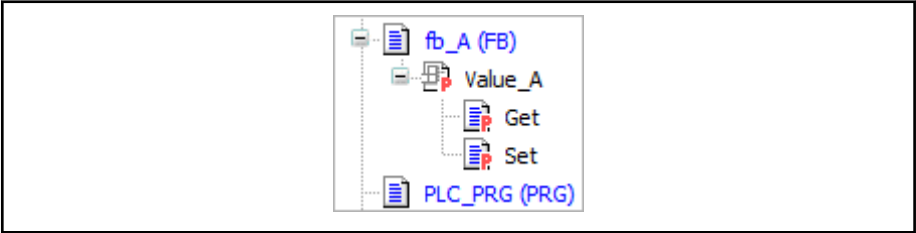


Abb. 18-28:

Funktionsbaustein fb_A

```
FUNCTION_BLOCK fb_A
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
  iA : INT;
  iB : INT;
END_VAR

iA := iA + 1;
```

Eigenschaft Value_A

```
PROPERTY PUBLIC Value_A : INT
```

Accessormethode fb_A.Value_A.Get

```
VAR
END_VAR

Value_A := iA;
```

Accessormethode fb_A.Value_A.Set

```
VAR
END_VAR

iB := Value_A;
```

Get- und Set-Accessor

Bei Aufruf des Set-Accessors wird auf die Eigenschaft geschrieben. Sie wird somit wie ein Eingabeparameter verwendet. Bei Aufruf des Get-Accessors

wird die Eigenschaft gelesen. Sie wird wie ein Ausgabeparameter verwendet. Der Zugriff wird jeweils über Zugriffsmodifizierer beschränkt, die Objekte sind dabei entsprechend gekennzeichnet.

Wenn auf eine Eigenschaft nur lesend oder nur schreibend zugegriffen wird, können Sie den nicht benötigten Accessor löschen.

Sie können explizit Accessoren hinzufügen, indem Sie im Projektextplorer eine Eigenschaft selektieren und den Kontextbefehl **Hinzufügen ► POU-Set-Zugriffsdefinition...**, oder **POU-Get-Zugriffsdefinition...** ausführen. Es öffnet sich der Dialog Set-Accessor hinzufügen, oder Get-Accessor hinzufügen. Dort können Sie die Implementierungssprache und den Zugriff einstellen.

Implementierungssprache	Beispiel: Strukturierter Text (ST)
Zugriffsmodifizierer	<i>Regelt den Zugriff auf die Daten.</i> • PUBLIC oder keine Angabe: Der Zugriff ist nicht eingeschränkt. • PRIVATE: Der Zugriff ist auf das Programm, oder den Funktionsbaustein beschränkt. Im Projektextplorer ist das Objekt mit (<code>private</code>) gekennzeichnet. Die Deklaration enthält das Schlüsselwort <code>PRIVATE</code>. • PROTECTED: Der Zugriff ist auf das Programm, oder den Funktionsbaustein mit seinen Ableitungen beschränkt. Im Projektextplorer ist das Objekt mit (<code>protected</code>) gekennzeichnet. Die Deklaration enthält das Schlüsselwort <code>PROTECTED</code>. • INTERNAL: Der Zugriff ist auf den Namensraum (Bibliothek) beschränkt. Im Projektextplorer ist das Objekt mit (<code>internal</code>) gekennzeichnet. Die Deklaration enthält das Schlüsselwort <code>INTERNAL</code>.

Monitoring von Eigenschaften im Onlinebetrieb

Für das Monitoring von Eigenschaften im Onlinebetrieb stehen folgende Pragmas zur Verfügung, die Sie an oberster Stelle in der Deklaration der Eigenschaft einfügen:

- `{attribute 'monitoring':= 'variable'}`: Bei jedem Zugriff auf die Eigenschaft speichert IndraLogic den Istwert in einer Variablen und stellt den Wert dieser Variablen dar. Dieser Wert kann unter Umständen veralten, wenn im Code kein Zugriff mehr auf die Eigenschaft erfolgt.
- `{attribute 'monitoring' := 'call'}`: Bei jeder Darstellung des Werts ruft IndraLogic den Code des Get-Accessors auf. Wenn dieser Code einen Seiteneffekt enthält, dann wird der Seiteneffekt durch das Monitoring ausgeführt!

Sie können eine Eigenschaft mit Hilfe folgender Funktionalitäten monitoren.

- Inline-Monitoring

Voraussetzung: In den Optionseinstellungen des Texteditors ist die Option "Inline-Monitoring aktivieren" angewählt.

Aufruf: IndraWorks Menü **Extras ► Optionen ► IndraLogic 2G ► Texteditor ► Monitoring**

- Überwachungsliste

Siehe auch:

- [Kap. 9.18.4 "Methodenaufruf" auf Seite 158](#)
- [Kap. "Attribut 'monitoring'" auf Seite 419](#)

Eingabeunterstützung beim Erzeugen von erbenden Bausteinen



Wenn Sie objektorientiert programmieren und die Vererbung von Bausteinen nutzen wollen, erhalten Sie folgendermaßen Unterstützung:

Wenn Sie unterhalb eines von einem Basisbaustein abgeleiteten Baustein eine Methode, Aktion etc. einfügen, erhalten Sie im Dialog "Objekt hinzufügen" eine Combobox mit einer Auswahl aller im Basisbaustein verwendeten Methoden, Aktionen etc.

Somit können Sie beispielsweise eine Methodendefinition der Basis auf einfache Weise zunächst übernehmen, um sie dann entsprechend für die abgeleitete Funktion des Bausteins anzupassen.

Methoden und Eigenschaften mit dem Zugriffsmodifizierer PRIVATE erscheinen in dieser Auswahl nicht, da sie nicht vererbt werden sollten.

Methoden und Eigenschaften mit dem Zugriffsmodifizierer PUBLIC erhalten beim Übernehmen in den abgeleiteten Baustein automatisch ein leeres Zugriffsmodifizierfeld, was funktional das Gleiche bedeutet.

Siehe auch

- [Kap. 9.18.1 "Erweitern eines Funktionsbausteins" auf Seite 154](#)
- [Kap. 9.18 "Objektorientiert programmieren" auf Seite 154](#)

Objekt 'Aktion'

Symbol:

Schlüsselwort: ACTION

In einer Aktion implementieren Sie weiteren Programmcode. Diesen Programmcode können Sie in einer anderen Sprache implementieren als die Basisimplementierung. Die Basisimplementierung ist der Funktionsbaustein oder das Programm, unter der Sie die Aktion eingefügt haben.

Eine Aktion hat keine eigenen Deklarationen und arbeitet mit den Daten der Basisimplementierung. Das bedeutet, dass die Aktion die Eingabe/Ausgabe- und lokalen Variablen ihrer Basisimplementierung verwendet.

Sie fügen eine **Aktion** einem Funktionsbaustein oder einem Programm mit dem Kontextmenü-Befehl **Hinzufügen ► Aktion** hinzu.

Aktion hinzufügen

Name	Name der Aktion
Implementierungssprache	Combobox für die Implementierungssprache

Eingabeunterstützung beim Erzeugen von ererbenden Bausteinen



Wenn Sie objektorientiert programmieren und die Vererbung von Bausteinen nutzen wollen, erhalten Sie folgendermaßen Unterstützung:

Wenn Sie unterhalb eines von einem Basisbaustein abgeleiteten Baustein eine Methode, Aktion etc. einfügen, erhalten Sie im Dialog "Objekt hinzufügen" eine Combobox mit einer Auswahl aller im Basisbaustein verwendeten Methoden, Aktionen etc.

Somit können Sie beispielsweise eine Methodendefinition der Basis auf einfache Weise zunächst übernehmen, um sie dann entsprechend für die abgeleitete Funktion des Bausteins anzupassen.

Methoden und Eigenschaften mit dem Zugriffsmodifizierer PRIVATE erscheinen in dieser Auswahl nicht, da sie nicht vererbt werden sollten.

Methoden und Eigenschaften mit dem Zugriffsmodifizierer PUBLIC erhalten beim Übernehmen in den abgeleiteten Baustein automatisch ein leeres Zugriffsmodifizierfeld, was funktional das Gleiche bedeutet.

Siehe auch

- [Kap. 9.18.1 "Erweitern eines Funktionsbausteins" auf Seite 154](#)
- [Kap. 9.18 "Objektorientiert programmieren" auf Seite 154](#)

Aufruf einer Aktion

Syntax:

`<program>.<action>` oder `<FB-instance>.<action>`

Wenn Sie eine Aktion nur innerhalb der Basisimplementierung aufrufen wollen, genügt es, nur den Aktionsnamen anzugeben.

Beispiele

Aufrufe einer Aktion **Reset** von einer anderen POU aus. Der Aufruf erfolgt also nicht in der Basisimplementierung.

Deklaration:

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

*Aufruf der Aktion **Reset** in einer AWL-POU*

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

*Aufruf der Aktion **Reset** in einer ST-POU:*

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

*Aufruf der Aktion **Reset** in einer FUP-POU:*

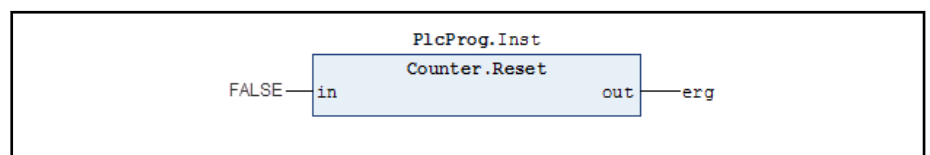


Abb. 18-29:

Objekt 'Transition'

Eingabeunterstützung beim Erzeugen von erbenden Bausteinen



Häufige Verwendung finden Aktionen in der Implementierungssprache AS.

Siehe auch

- [Kap. "AS-Element 'Aktion'" auf Seite 264](#)

Symbol:

Schlüsselwort: TRANSITION

Das Objekt kann in einem in Ablaufsprache implementierten Baustein als Transitionselement verwendet werden.

Siehe auch

- [Kap. "AS-Editor im Online-Betrieb" auf Seite 253](#)



Wenn Sie objektorientiert programmieren und die Vererbung von Bausteinen nutzen wollen, erhalten Sie folgendermaßen Unterstützung:

Wenn Sie unterhalb eines von einem Basisbaustein abgeleiteten Baustein eine Methode, Aktion etc. einfügen, erhalten Sie im Dialog "Objekt hinzufügen" eine Combobox mit einer Auswahl aller im Basisbaustein verwendeten Methoden, Aktionen etc.

Somit können Sie beispielsweise eine Methodendefinition der Basis auf einfache Weise zunächst übernehmen, um sie dann entsprechend für die abgeleitete Funktion des Bausteins anzupassen.

Methoden und Eigenschaften mit dem Zugriffsmodifizierer PRIVATE erscheinen in dieser Auswahl nicht, da sie nicht vererbt werden sollten.

Methoden und Eigenschaften mit dem Zugriffsmodifizierer PUBLIC erhalten beim Übernehmen in den abgeleiteten Baustein automatisch ein leeres Zugriffsmodifizierfeld, was funktional das Gleiche bedeutet.

Siehe auch

- [Kap. 9.18.1 "Erweitern eines Funktionsbausteins" auf Seite 154](#)
- [Kap. 9.18 "Objektorientiert programmieren" auf Seite 154](#)

18.1.18 Überwachung des SPS-Programms der Applikation während der Laufzeit

Zur Überwachung von SPS-Programmen während der Laufzeit auf Überschreitung der Grenzen von Arrays und Unterbereichstypen, der Gültigkeit von Pointernzugriffen und Division durch Null bietet IndraLogic folgende Möglichkeiten:

- [Kap. "POU-Vorlagen" auf Seite 578](#)

Über das Kontextmenü des Applikationsknoten **Hinzufügen ► POU-Vorlagen ► IndraMotion ... CheckFunctions** können gezielt Checkfunktionen ausgewählt werden. Die einzelnen Funktionen werden implizit vor dem möglichen Fehlzugriff in den Programmcode eingebunden. Im Falle eines Fehlzugriffs werden ein Eintrag ins Logbuch der Steuerung erzeugt und per Default die Abarbeitung des Programmcodes mit Ausnahmefehl-

ler gestoppt. In diesem Fall sind die Aufrufliste (**Debuggen ► Aufrufliste**) zu überprüfen und der fehlerhafte Code zu modifizieren.

Mit dem Einbinden bzw. Löschen einer Checkfunktion ist ein Einloggen nur noch mit Download möglich. Desweiteren werden zusätzliche Kompilerausgaben "Checkxxxx: Please implement your own error reaction below" erzeugt, um den Anwender darauf hinzuweisen, dass der implementierte Funktionscode an das Maschinenprogramm gegebenenfalls anzupassen ist. Je nach Applikation kann es unzulässig sein, dass die Maschine im laufenden Betrieb in Stop geschaltet wird.

In jeder Checkfunktion ist beispielhaft eine Alternative vorgegeben, die die fehlerhafte Operation so anpasst, dass diese zu keinem Maschinenstillstand führt und einen Eintrag in den Fehler- und Diagnosespeicher der Steuerung vornimmt. So wird beispielsweise eine Division durch Null gegen eine Division durch Eins ersetzt. Es sind ggf. weitere Fehlerreaktionen zu implementieren, denn die Applikation würde mit einem fehlerhaft berechneten Wert weiterlaufen.



1. Die Implementierung der Fehlerreaktion innerhalb der Checkfunktion ist an das geforderte Maschinenverhalten anzupassen!

Per Default wird die Applikation bei einem Fehler gestoppt.

2. Mit dem Einbinden bzw. Löschen der Funktion ist ein anschließendes Einloggen nur als Download möglich. Ein Onlinechange ist nicht mehr möglich.

3. Durch das Hinzubinden einer Checkfunktion kann sich die Laufzeit des SPS-Programms wesentlich verlängern.

Nach Beseitigung aller Fehler sollten die Funktionen wieder entfernt werden, um die Laufzeit wieder zu reduzieren.

• Die Bibliothek **RIL_CheckRTV.library**

Wird diese Bibliothek durch den Bibliotheksverwalter der Applikation hinzugefügt, werden ihre Funktionen durch den Compiler automatisch an den gefährdeten Stellen im Programmcode eingebunden.

Sobald die Checkfunktion einen fehlerhaften Zugriff bzw. eine fehlerhafte Operation erkennt, wird gezielt eine Exception ausgelöst, um die Applikation in den sicheren Zustand zu schalten. Eine Änderung dieses Verhaltens ist nicht möglich.

In diesem Fall sind die Aufrufliste (**Debuggen ► Aufrufliste**) zu überprüfen und der fehlerhafte Code zu modifizieren.

Sollte ein Stoppen der Applikation im laufenden Betrieb einer Maschine nicht zulässig sein, so darf diese Bibliothek nicht verwendet werden!



1. Die Applikation wird bei einem Fehler immer gestoppt.
2. Mit dem Einbinden bzw. Löschen der Funktion ist ein anschließendes Einloggen nur als Download möglich. Ein Onlinechange ist nicht mehr möglich.
3. Durch das Hinzubinden dieser Bibliothek kann sich die Laufzeit des SPS-Programms **wesentlich** verlängern.

Nach Beseitigung aller Fehler sollten die Funktionen wieder entfernt werden, um die Laufzeit wieder zu reduzieren.

POU-Vorlagen

SPS-Überwachungsfunktionen

Unter dem Menüpunkt **Extras ▶ Vorlagendatenbank** oder im Kontextmenü von ... ▶ **Logic ▶ Application ▶ Hinzufügen ▶ POU Vorlagen ▶ IndraMotion ... CheckFunctions** sind zur Überwachung des SPS-Programmcodes mehrere Funktionen aus der RIL_CheckRTV.library extrahiert worden. Die hier gelisteten Funktionen können zur Überwachung des SPS- Programmcodes gezielt implementiert werden.

Die einzelnen Funktionen werden implizit vor dem möglichen Fehlzugriff in den Programmcodes eingebunden. Im Falle eines Fehlzugriffs werden ein Eintrag ins Logbuch der Steuerung erzeugt und die Abarbeitung des Programmcodes mit Ausnahmefehler gestoppt. In diesem Fall sind die Aufrufliste (**Debuggen ▶ Aufrufliste**) zu überprüfen und der fehlerhafte Code zu modifizieren.



1. Durch das Hinzubinden der Funktion(en) kann sich die Laufzeit des SPS-Programms wesentlich verlängern.
2. Nach Beseitigung der Fehler in der Inbetriebnahme sollte(n) die Funktion(en) wieder entfernt werden.



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



Wurde eines der Objekte der Applikation hinzugefügt, ist eine durch den Bibliotheksverwalter hinzugefügte RIL_CheckRTV.library wirkungslos, nur die einzeln gewählten Bausteine kommen bei der Übersetzung zur Anwendung.

Kurzbeschreibung der Funktionen:

Checkbounds	Die Funktion "CheckBounds" prüft den korrekten Zugriff auf Arrayelemente im IEC61131.3-Programm. Aufgabe dieser Überwachungsfunktion ist eine angemessene Behandlung von Verletzungen von Feldgrenzen. Der Aufruf der Funktion erfolgt implizit sobald einer Variablen vom Typ Array Werte zugewiesen werden.
CheckDivInt	Um den Wert eines Divisors (Typ 32-Bit) zu überwachen, um also ein Teilen durch 0 zu vermeiden, wird die Funktion "CheckDivInt" verwendet. Nach der Einbindung der Funktion "CheckDivInt" in das IndraLogic-Projekt geht jeder im zugehörigen Code auftretenden Division mit 32-Bit-Datentypen automatisch der Aufruf der Funktion "CheckDivInt" voraus.
CheckDivLint	Um den Wert eines Divisors (Typ 64-Bit) zu überwachen, um also ein Teilen durch 0 zu vermeiden, wird die Funktion "CheckDivLint" verwendet. Nach der Einbindung der Funktion "CheckDivLint" in das IndraLogic-Projekt geht jeder im zugehörigen Code auftretenden Division mit 64-Bit-Datentypen automatisch der Aufruf der Funktion "CheckDivLint" voraus.
CheckDivLReal	Um den Wert eines Divisors vom Typ LREAL zu überwachen, um also ein Teilen durch 0.0 zu vermeiden, wird die Funktion "CheckDivLReal" verwendet. Nach der Einbindung der Funktion "CheckDivLReal" in das IndraLogic-Projekt geht jeder im zugehörigen Code auftretenden Division mit LREAL-Variablen automatisch der Aufruf der Funktion "CheckDivLReal" voraus.

CheckDivReal	Um den Wert eines Divisors vom Typ LREAL zu überwachen, um also ein Teilen durch 0.0 zu vermeiden, wird die Funktion "CheckDivReal" verwendet. Nach der Einbindung der Funktion "CheckDivReal" in das IndraLogic-Projekt geht jeder im zugehörigen Code auftretenden Division mit REAL-Variablen automatisch der Aufruf der Funktion "CheckDivReal" voraus.
CheckPointer	Nach der Einbindung der Funktion "CheckPointer" wird die Funktion zur Kontrolle von Lese- oder Schreibzugriffen mittels Pointer verwendet.
CheckRangeSigned	Um die Bereichsgrenzen für vorzeichenbehaftete Unterbereichstypen während der Laufzeit zu überwachen, wird die Funktion "CheckRangeSigned" verwendet. Nach der Einbindung der Funktion "CheckRangeSigned" in das IndraLogic-Projekt geht jedem Zugriff auf vorzeichenbehaftete, wertebeschränkte Variablen im zugehörigen Code automatisch der Aufruf der Funktion "CheckRangeSigned" voraus.
CheckRangeUnsigned	Um die Bereichsgrenzen für nicht vorzeichenbehaftete Unterbereichstypen während der Laufzeit zu überwachen, wird die Funktion "CheckRangeUnsigned" verwendet. Nach der Einbindung der Funktion "CheckRangeUnsigned" in das IndraLogic-Projekt geht jedem Zugriff auf nicht vorzeichenbehaftete, wertebeschränkte Variablen im zugehörigen Code automatisch der Aufruf der Funktion "CheckRangeUnsigned" voraus.

18.1.19 Objekt 'Projekteinstellungen'

Symbol 

Funktion: Das Objekt enthält die Konfiguration des Projekts.

Aufruf: Doppelklick auf das Objekt im Projektexplorer, Ordner "Allgemeiner Baustein"

IndraLogic speichert die Projekteinstellungen direkt im Projekt. Wenn Sie zum Beispiel ein Projekt auf ein anderes System übertragen, wird auch das Objekt **Projekteinstellungen** übertragen, ohne das dafür ein Projektarchiv benötigen.

Die Projekteinstellungen gelten projektweit und bieten für verschiedene Kategorien **AS**, **Compiler-Optionen / Compiler-Warnungen**, **StaticAnalysisLight** und **Visualisierung** Einstellmöglichkeiten.

Siehe auch

- [Kap. "Dialog 'Projekteinstellungen- 'AS'" auf Seite 758](#)
- [Kap. "Dialog 'Projekteinstellungen- 'Compileroptionen'" auf Seite 759](#)
- [Kap. "Dialog 'Projekteinstellungen- 'Compilerwarnungen'" auf Seite 760](#)
- [Kap. "Dialog 'Projekteinstellungen- 'Static Analysis Light'" auf Seite 761](#)
- [Kap. "Dialog 'Projekteinstellungen- 'Visualisierung'" auf Seite 763](#)

18.1.20 Objekt 'Bibliotheksinfo'

Symbol: 

Funktion: Das Objekt enthält die Eigenschaften, Metainformationen und Informationen des Projekts und ermöglicht Ihnen die Urheberschaft und Integrität des Projekts zu überprüfen.

Aufruf: Doppelklick auf das Objekt im Projektexplorer, Ordner "Allgemeiner Baustein"

Voraussetzung: IndraLogic erstellt das Objekt automatisch in Bibliotheksprojekten oder über den Kontextmenübefehl **Hinzufügen ► Bibliotheksinfo....**

Indralogic speichert die Bibliotheksinfo direkt im Projekt.

Wenn Sie zum Beispiel ein Projekt auf ein anderes System übertragen, wird auch das Objekt **Bibliotheksinfo** übertragen. Sie benötigen kein Projektarchiv.

Registerkarte 'Datei'

Die Registerkarte zeigt die Eigenschaften der Projektdatei und deren Attribute an. Sie können diese Attribute nicht editieren. Sie entsprechen den Dateieigenschaften des Windows-Explorers. Ferner wird die IndraLogic-Version angegeben, mit der die **Bibliotheksinfo** abgespeichert wurden.

Registerkarte 'Zusammenfassung'

Die Registerkarte enthält allgemeinen Informationen und Metainformationen der Projektdatei.

IndraLogic erstellt aus diesen Informationen Schlüssel in der Registerkarte **Eigenschaften**. Wenn zum Beispiel in **Firma** der Name `Company_A` eingetragen ist, ist in der Registerkarte **Eigenschaften** der Schlüssel `Company` mit dem Wert `Company_A` vorhanden.



Wenn Sie Ihr Projekt als Bibliotheksprojekt speichern, sollten Sie die Richtlinien für Bibliotheksentwickler "Library Development Summary" beachten.

Bei einem Bibliotheksprojekt muss eine **Firma**, ein **Titel** und eine **Version** eingetragen sein, um die Bibliothek installieren zu können.

Firma	Name der Firma, zum Beispiel <code>Company_A</code> .
Titel	Titel des Projekts, zum Beispiel <code>Automation_A</code>
Version	Version des Projekts, zum Beispiel <code>0.0.0.1</code>
Freigegeben	 Schutz gegen Änderung aktiviert. Folge: Wenn Sie nun das Projekt editieren, erscheint eine Eingabeaufforderung, ob das Projekt wirklich geändert werden soll. Wenn Sie diese Abfrage einmalig mit Ja beantworten, erscheinen bei weiteren Editieraktionen keine Abfrage mehr.
Kategorien	Kategorien des Bibliotheksprojekt, nach denen Sie im Dialog Bibliotheks-Repository sortieren können. Wenn keine Kategorie angegeben ist, wird der Bibliothek die Kategorie Sonstige zugewiesen. Die Kategorien stammen aus einer oder mehreren externen Beschreibungsdateien im XML-Format. Sie können aber auch aus einem bereits erstellten Bibliotheksprojekt stammen. Voraussetzung: Das Projekt ist ein Bibliotheksprojekt.
	Dialog Bibliothekskategorien öffnet sich, in dem Sie Bibliothekskategorien hinzufügen können.
Standardnamensraum	Wenn Sie hier keinen Standardnamensraum definieren, gilt automatisch der Name der Bibliotheksdatei als Namensraum.

Autor	Autor des Projekts, zum Beispiel <code>Arnold Best</code>
Beschreibung	Zum Beispiel: <code>Nur für internen Gebrauch</code>



Das Metadatum **Freigegeben** wirkt sowohl bei Bibliotheksprojekten als auch in Standardprojekten!

So wird Ihnen beispielsweise in einem freigegebenen Standardprojekt verwehrt eine neue Liste mit **Persistenten Variablen** abulegen. Dies gelingt Ihnen erst, wenn das Metadatum gelöscht wurde.

Siehe auch: [Kap. 9.16.5 "Speichern der Werte einer persistenten Variablenliste in einer Rezeptur" auf Seite 152.](#)

Liste von Kategorien	Liste der Kategorien, die dem Bibliotheksprojekt zugewiesen sind. Sie können aus mehrere Quellen stammen. Wenn Sie alle gewünschten Kategorien eingetragen haben, bestätigen Sie den Dialog mit OK .
Schaltfläche Hinzufügen	Die Befehle Von Beschreibungsdatei und Von anderer Bibliothek erscheinen.
Schaltfläche Entfernen	IndraLogic entfernt die selektierte Kategorie.
Befehl Von Beschreibungsdatei	Dialog Beschreibungsdatei auswählen erscheint, in dem Sie eine Beschreibungsdatei mit Erweiterung <code>*.libcat.xml</code> auswählen. Die Datei enthält Befehlskategorien. Wenn Sie den Dialog mit Öffnen beenden, übernimmt IndraLogic diese Kategorien.
Befehl Von anderer Bibliothek	Dialog Bibliothek auswählen erscheint, in dem Sie eine Bibliothek auswählen, deren Befehlskategorien übernommen werden sollen. Wenn Sie den Dialog mit Öffnen beenden, übernimmt IndraLogic die Kategorien.
Schaltfläche OK	IndraLogic stellt die Kategorien als Bibliotheksinformation zur Verfügung und zeigt sie im Feld Bibliothekskategorien an.

Tab. 18-6: *Dialog Bibliothekskategorien*

Siehe auch

- [Kap. 19.1 "Richtlinien für das Erstellen von Bibliotheken" auf Seite 789](#)
- [Kap. "Befehl 'Bibliotheks-Repository'" auf Seite 676](#)

Registerkarte 'Eigenschaften'

In dieser Registerkarte können Sie Schlüssel definieren, die Sie mit anwenderspezifischen externen Programmen von außen steuern können.



Wenn Sie ein Bibliotheksprojekt geöffnet haben, beachten Sie bitte die Beschreibung der relevanten Schlüssel in den Richtlinien für Bibliotheksentwickler: "Library Development Summary".

Wenn Sie als Projekt eine Symbolbibliothek geöffnet haben, muss der Schlüssel `VisuSymbolLibrary = TRUE` definiert sein. Er kennzeichnet die Bibliothek als Symbolbibliothek.

Schlüssel	Name des Schlüssels. Beliebige Zeichenfolge für neuen Schlüssel eintragen, oder einen bereits in der Tabelle Eigenschaften eingetragenen Schlüssel selektieren.
Typ	Datentyp des Schlüssels. Mögliche Typen (Text , Datum , Zahl , BOOL , Version)

Wert	Wert des Schlüssels im zulässigen Format: • "Text": Beliebige Zeichenfolge • "Datum": Beispiel: Freitag, 1. Januar 2016 00:00:00. Mindesteintrag für dieses Datum: 1.1 • "Zahl": Ganze Zahl im Integer32-Format, mit oder ohne Vorzeichen, Beispiel: -32500. • "Bool": True oder False, Groß-/Kleinschreibung nicht relevant • "Version": Beispiele: 1.1, 1.0.1.0, maximal vierstellig.
Hinzufügen	Fügt den neu definierten Schlüssel in der Tabelle Eigenschaften hinzu.
Ändern	Speichert die für den in der Tabelle "Eigenschaften" selektierten Schlüssel vorgenommene Änderung.
Entfernen	Entfernt den in der Tabelle "Eigenschaften" selektierten Schlüssel.

Eigenschaften	Liste der Eigenschaften, die als Schlüssel definiert sind. IndraLogic erstellt automatisch Schlüssel zu den Informationen der Registerkarte Zusammenfassung. Klicken Sie auf einen Schlüssel, um ihn in den Eingabefelder, die oberhalb der Liste sind, zu bearbeiten.

Automatisch Bibliotheksinfo-POUs erzeugen	 IndraLogic erstellt Programmierobjekte des Typs <code>FUNCTION</code> im Ordner "Allgemeiner Baustein", Unterordner "Library Information" und ermöglicht damit einen Zugriff in der Applikation auf die Bibliothekseigenschaften. • <code>GetLibVersion</code> (Versionsnummer als Zeichenfolge) • <code>GetLibVersionNumber</code> (Versionsnummer als Zahlenwert) • <code>IsLibReleased</code> (TRUE/FALSE)
Automatisch Projektinformationen-POUs erzeugen	 IndraLogic erstellt POU-Objekte des Typs <code>FUNCTION</code> im Ordner "Allgemeiner Baustein", Unterordner "Projektinformationen" und ermöglicht damit einen Zugriff in der Applikation auf die Projekteigenschaften. Für die Eigenschaften Firma, Titel und Version werden die nachfolgenden Funktionsbausteine erstellt: • <code>GetCompany</code> • <code>GetTitle</code> • <code>GetVersion</code> Für benutzerdefinierte Eigenschaften stehen folgende Funktionsbausteine zur Verfügung: • <code>GetBooleanProperty</code>: BOOL (TRUE/FALSE) • <code>GetNumberProperty</code>: DINT (Zahlenwert) • <code>GetTextProperty</code>: WSTRING (Zeichenfolge) • <code>GetTextProperty2</code>: POINTER TO WSTRING (ohne Längenbegrenzung) • <code>GetVersionProperty</code>: VERSION (Versionsnummer als Zeichenfolge)

Siehe auch

- [Kap. 19.1 "Richtlinien für das Erstellen von Bibliotheken" auf Seite 789](#)
- [Kap. 20.1.3 "Symbolbibliothek in der Visualisierung verwenden" auf Seite 809](#)

- [Kap. "Dialog 'Eigenschaften- 'Bildersammlung'" auf Seite 757](#)
- Registerkarte 'Statistik'** Der Dialog gibt eine statistische Auskunft, wie viele Objekte der einzelnen Typen im Projekt verwendet sind.
- Registerkarte 'Signieren'** Mit diesem Dialog können Sie ein Bibliotheksprojekt mit einem herstellerspezifischen, einmaligen Schlüssel signieren. Voraussetzung: Dieser Schlüssel liegt für Sie in Form einer "Privaten Schlüsseldatei" *.libpk mit einem zugehörigen Token bereit.

Signieren aktivieren	 IndraLogic signiert das Bibliotheksprojekt mit einem herstellerspezifischen, einmaligen Schlüssel.
Private Schlüsseldatei	Speicherort der privaten Schlüsseldatei *.libpk, zum Beispiel: D:\for lib developers only\mycomp_libkey.libpk
Token des öffentlichen Schlüssels	Zum Beispiel: 427A5701DA3CF3CF Voraussetzung: Eine privaten Schlüsseldatei ist angegeben und IndraLogic hat den Token ausgelesen und eingetragen
Private Schlüsseldatei erzeugen...	IndraLogic erzeugt eine neue private Schlüsseldatei.

Siehe auch

- ["Bibliotheksprojekt signieren" auf Seite 13](#)

Siehe auch

- [Kap. 2.5.1 "Bibliotheksinfo abfragen und bearbeiten" auf Seite 12](#)
- [Kap. 2.5.2 "Projekteinstellungen vornehmen" auf Seite 14](#)

18.1.21 Objekt 'Rezepturverwalter'

- Registerkarte 'Speicherung'** Der Rezepturverwalter bietet Funktionen zur Verwaltung von benutzerdefinierten Variablenlisten, so genannten Rezepturdefinitionen.

Speicherungstyp	textuell: IndraLogic speichert die Rezeptur in einem lesbaren Format mit den konfigurierten Spalten und Trennzeichen. binär: IndraLogic speichert die Rezeptur in einem nicht lesbaren binären Format. Dieses Format benötigt weniger Speicherplatz. Hinweis: Sie können binär gespeicherte Rezepturen nur dann wieder einlesen, wenn Sie die Variablenlisten nicht verändert haben.
Dateipfad	Relativer Pfad auf dem Laufzeitsystem. Der Pfad muss mit einem "\" enden. Beispiel MyRecipes\. Dieser Pfad wird auf dem Zielsystem im Verzeichnis der Laufzeitdateien angelegt. IndraLogic legt beim Download auf die Steuerung für jede Rezeptur eine Datei in diesem Verzeichnis ab. Voraussetzung ist, dass die Option Rezepturverwaltung in der SPS aktiviert ist. Die Dateien werden bei jedem Neustart der Applikation in den Rezepturverwalter geladen.
Dateierweiterung	Dateierweiterung für die Rezepturdatei Daraus ergibt sich der Standardname für Rezepturdateien in der Form <Rezeptur>.<Rezepturdefinition>.<Dateierweiterung>
Trennzeichen	Trennzeichen zwischen den einzelnen Werten in der gespeicherten Datei: Tab, ";", ",", ":", "=", " "

Referenz Benutzeroberfläche

Verfügbare Spalten Ausgewählte Spalten	Festlegung, welche Informationen in welcher Reihenfolge in die Rezepturdatei gespeichert werden
Als Default speichern	IndraLogic verwendet die Default-Einstellung projektübergreifend für alle weiteren Rezepturverwalter.

Registerkarte 'Allgemein'

Rezepturverwaltung in der SPS	 : Muss aktiviert sein, wenn zur Laufzeit Rezepturen durch Visualisierungselemente oder durch das Anwenderprogramm geladen werden. Falls Rezepturen ausschließlich über die IndraLogic Programmieroberfläche zur Steuerung übertragen werden, kann die Option deaktiviert werden.
Rezepturänderungen automatisch in Rezepturdateien speichern	 : Empfohlene Option, weil sie das "übliche" Verhalten einer Rezepturverwaltung bewirkt. Die Rezepturdateien werden automatisch bei jeder Änderung einer Rezeptur zur Laufzeit aktualisiert. Die Option ist nur wirksam, wenn die Option Rezepturverwaltung in der SPS aktiviert ist.
Nur bei exakter Übereinstimmung der Variablenliste laden	Rezepturdateien werden nur geladen, wenn die Datei alle Variablen aus der Variablenliste der Rezepturdefinition der Applikation enthält. Die Variablenliste in der Datei muss in derselben Reihenfolge vorliegen. Zusätzliche Einträge am Ende werden ignoriert. Andernfalls wird der Fehlerstatus <code>ERR_RECIPE_MISMATCH</code> gesetzt (<code>RecipeManCommands.GetLastError</code>).
Variablen mit übereinstimmenden Namen laden	Es werden nur die Variablen mit übereinstimmenden Variablennamen aus der Rezepturdatei geladen. Es wird kein Fehlerstatus gesetzt, wenn sich die Variablenliste in der Datei von der in der Rezepturdefinition der Applikation unterscheidet. Damit können Rezepturdateien auch dann noch geladen werden, wenn Variablen in der Datei oder in der Rezepturdefinition gelöscht wurden.

Aktionen	Im Projekt definierte Rezepturen	Zur Laufzeit definierte Rezepturen
Online Reset Warm Online Reset Kalt Download	Die Rezepturen aller Rezepturdefinitionen werden mit den Werten aus dem aktuellen Projekt belegt.	Dynamisch erzeugte Rezepturen bleiben unverändert.
Online Reset Ursprung	Die Applikation wird von der SPS entfernt. Wenn danach ein neuer Download erfolgt, werden die Rezepturen so wiederhergestellt wie bei einem Online Reset Warm	
Shutdown und Neustart der SPS	Nach dem Neustart werden die Rezepturen erneut aus den automatisch angelegten Dateien geladen. Damit wird der gleiche Zustand wie vor dem Herunterfahren wiederhergestellt.	
Online Change	Die Rezepturwerte bleiben unverändert. Während der Laufzeit kann eine Rezeptur nur über die Befehle des Funktionsbausteins <code>RecipeManCommands</code> verändert werden.	
Stop	Bei einem Stop/Start der SPS bleiben die Rezepturen unverändert.	

Tab. 18-7: Rezepturen im Online-Betrieb wenn die Option "Änderungen in Rezepturen automatisch speichern" aktiviert ist

Aktionen	Im Projekt definierte Rezepturen	Zur Laufzeit definierte Rezepturen
Online Reset Warm Online Reset Kalt Download	Die Rezepturen aller Rezepturdefinitionen werden mit den Werten aus dem aktuellen Projekt belegt. Allerdings werden diese nur im Speicher gesetzt. Um die Rezepturen in einer Datei zu speichern, muss explizit der Befehl Rezeptur speichern verwendet werden.	Dynamisch erzeugte Rezepturen gehen verloren.
Online Reset Ursprung	Die Applikation wird von der SPS entfernt. Wenn danach ein neuer Download erfolgt, werden die Rezepturen wiederhergestellt.	Dynamisch erzeugte Rezepturen gehen verloren.
Shutdown und Neustart der SPS	Nach dem Neustart werden die Rezepturen erneut aus den automatisch angelegten Dateien geladen. Damit wird der gleiche Zustand wie vor dem Herunterfahren wiederhergestellt.	
Online Change	Die Rezepturwerte bleiben unverändert. Während der Laufzeit kann eine Rezeptur nur über die Befehle des Funktionsbausteins <code>RecipeManCommands</code> verändert werden.	
Stop	Bei einem Stop/Start der SPS bleiben die Rezepturen unverändert.	

Tab. 18-8: *Rezepturen im Online-Betrieb wenn "Änderungen in Rezepturen automatisch speichern" NICHT aktiviert ist*

Einer der angegebenen "Speicherungstyp"en (textuell oder binär) sowie der Speicherort ("Dateipfad") und die "Dateierweiterung" der zu speichernden Rezeptur sind anzugeben.

Speicherungstyp "binär"



Beim Speicherungstyp "binär" werden die Daten in der zu speichernden Rezeptur kompakt, ohne Namen, abgelegt.

Für jedes Datum wird die Länge in Byte reserviert, die dem Datentyp entspricht.

Folglich muss beim Laden der Rezeptur aus einer Binärdatei die Rezepturdefinition die gleiche sein, wie sie beim Speichern war.

Folgende Fehler können beim "Rezeptur laden" auftreten:

- Ein Vertauschen von zwei typgleichen Variablen in der Rezepturdefinition kann nicht erkannt werden und führt zu fehlerhaften Variablenwerten.
- Eine Änderung des Variablentypes (auch bei gleicher Variablenlänge) wird erkannt. "Das Dateiformat ist ungültig" wird als Fehler ausgegeben.
- Zusätzliche oder weniger Variablen in der Rezepturdefinition werden erkannt. Eine Wiederherstellung ist nicht mehr möglich.

Recipes_01 GVL PlcProg							
Variable	Typ	Name	Komm...	Minim...	Maxim...	Aktueller Wert	dd
GVL.dword1	DWORD					16#87654321	2271560481
GVL.bool1	BOOL					TRUE	TRUE
GVL.dint1	DINT					16#00000021	33
GVL.bool2	BOOL					FALSE	FALSE
GVL.word1	WORD					16#4321	17185
GVL.bool3	BOOL					TRUE	TRUE
GVL.string1	STRING					'Ein Text'	'Ein Text'
GVL.bool4	BOOL					TRUE	TRUE
GVL.real1	REAL					3.14	3.14
GVL.bool5	BOOL					TRUE	TRUE

Abb. 18-30: Rezeptur, online

00000000:	04 00 05 00	21 43 65 87	01 00 00 00	01 04 00 04		!	Ce	.		
00000010:	00 21 00 00	00 01 00 00	00 00 02 00	03 00 21 43		!	.			!C
00000020:	01 00 00 00	01 00 00 00	45 69 6E 20	54 65 78		.	..			Ein Tex
00000030:	74 00 01 00	00 00 01 04	00 06 00 C3	F5 48 40 01		t	.			ÄHÖ.
00000040:	00 00 00 01					.				

Abb. 18-31: Binär gespeicherte Rezeptur

Gerahmt: Länge in Byte des danach folgenden Datum gefolgt vom Typkennzeichner;

Ohne Rahmen: Datum (Aktueller Wert)

Speicherungstyp "textuell"

Bei textueller Speicherung werden die zu speichernden Spalten durch ein **Trennzeichen** getrennt, für das eine der sechs Optionen ausgewählt werden kann, siehe oben.

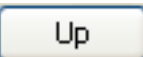
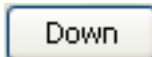
Die textuelle Variante kann extern bearbeitet und auch an andere Applikationen übergeben werden, sie benötigt jedoch mehr Speicherplatz als die binäre.

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 wird.

Mit den Schaltflächen  oder  können alle Einträge gleichzeitig verschoben werden.

Mit den Schaltflächen  und  kann dann die Reihenfolge der ausgewählten Spalten im rechten Fenster verändert werden, gemäß der die Spalten dann auch in der Speicherdatei angeordnet werden.

Recipes_01 GVL PlcProg							
Variable	Typ	Name	Komm...	Minim...	Maxim...	Aktueller Wert	dd
GVL.dword1	DWORD					16#87654321	2271560481
GVL.bool1	BOOL					TRUE	TRUE
GVL.dint1	DINT					16#00000021	33
GVL.bool2	BOOL					FALSE	FALSE
GVL.word1	WORD					16#4321	17185
GVL.bool3	BOOL					TRUE	TRUE
GVL.string1	STRING					'Ein Text'	'Ein Text'
GVL.bool4	BOOL					TRUE	TRUE
GVL.real1	REAL					3.14	3.14
GVL.bool5	BOOL					TRUE	TRUE

Abb. 18-32: Rezeptur, online

```

GVL.dword1:=2271560481
GVL.bool1:=TRUE
GVL.dint1:=33
GVL.bool2:=FALSE
GVL.word1:=17185
GVL.bool3:=TRUE
GVL.string1:='Ein Text'
GVL.bool4:=TRUE
GVL.real1:=3.14
GVL.bool5:=TRUE

```

Abb. 18-33: Textuell gespeicherte Rezeptur

Siehe auch

- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)

18.1.22 Objekt 'Rezepturdefinition'

Innerhalb der Rezepturdefinitionen (1) definieren Sie verschiedene Werte-Sets für die Variablen, welche Rezepturen (2) genannt werden.

Sie können die Ansicht der Rezepturdefinition zwischen der einfachen Ansicht (3) und der strukturierten Ansicht (4) umschalten. In der strukturierten Ansicht stellt IndraLogic die zu einer Struktur zugehörigen Variablen gruppiert dar.

RecipeDef1 x RecipeDef2									
Variable	Type	Name	Comment	Min...	Max...	Current...	R1	R2	
PLC_PRG.iColorG	INT	Green501	green Color	0	100		10	15	(3)
PLC_PRG.iColorR	INT	Red001	red Color	0	100		15	30	(4)
PLC_PRG.iColorY	INT	Yellow001	yellow color	0	100		85	55	
PLC_PRG.sVisutext	STRING						'Product A'	'Product B'	
PLC_PRG.bSpecialColor	BOOL						TRUE	FALSE	

Abb. 18-34:

Datentyp	Wird automatisch eingetragen
Name	Optional
Minimalwert Maximalwert	Wenn der Variablenwert kleiner als der Minimalwert oder größer als der Maximalwert , dann setzt IndraLogic den Wert auf den Minimalwert oder Maximalwert .
Kommentar	Zusätzliche Information, wie beispielsweise die Einheit des Wertes.
Aktueller Wert	Aktueller Variablenwert, wird nur im Online-Betrieb angezeigt.

Siehe auch

- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)

18.1.23 Objekt 'Textliste'

Symbol: 

Das Objekt dient der Erstellung, Verwaltung und Übersetzung von Texten. Es enthält eine Tabelle mit Texten, der Sie neue Texte hinzufügen können.

Einen Text, den Sie hier verfasst haben, können Sie in einer Visualisierung in einem Element in der Eigenschaft **Dynamische Texte** auswählen. Die Visualisierung im laufenden Betrieb gibt diesen Text in der ausgewählten Sprache dynamisch aus.

ID	Eindeutiger Identifikator des Texts
Standard	Ausgangstext als Zeichenkette, zum Beispiel <code>Information A: %i Möglichkeiten.</code> Mit dem Tastaturkürzel [Strg] + [Eingabe] fügen Sie einen Zeilenumbruch ein. Doppelklicken Sie in das Feld, um den Text zu editieren.
Die Tabelle enthält beliebig viele von Ihnen hinzugefügte Sprachspalten. Eine Sprachspalte ist mit einem Sprachkürzel betitelt, das Sie beim Erstellen der Spalte mit dem Befehl Sprache einfügen angegeben haben.	
<Sprachkürzel>	Name der Sprache als Sprachkürzel. Zum Beispiel <code>en-US</code> . Unter dieser Spalte ist die Übersetzung des Texts, der unter Standard verfasst ist. Unter der Voraussetzung, dass im Visualisierungsmanager das Sprachkürzel als Sprache ausgewählt ist, gibt eine Visualisierung im laufenden Betrieb den übersetzten Text aus. Wenn keine Übersetzung verfasst ist, verwendet IndraLogic den Text unter Standard . Eine Visualisierung im laufenden Betrieb kann auch eine Sprachumschaltung, die von einem Benutzer angefordert wird, durchführen.
Leere Zeile	Editieren Sie die Zeile, um einen eigenen Text hinzuzufügen.

Siehe auch

- [Kap. 9.8 "Text in einer Textliste verwalten" auf Seite 117](#)

18.1.24 Objekt 'Symbolkonfiguration'

Funktion: Mit der Symbolkonfiguration können Sie Symbolbeschreibungen für Projektvariablen erstellen.

Normalerweise wird das Objekt "Symbolkonfiguration" gemeinsam mit den Applikationsobjekt automatisch angelegt.

Fehlt das Objekt "Symbolkonfiguration" jedoch, kann es mit dem Kontextmenü-Befehl **Hinzufügen** unterhalb der Applikation eingefügt werden.

Den Symbolkonfigurationseditor öffnen Sie dann durch einen Doppelklick auf das Objekt **Symbolkonfiguration**.

Dialog 'Symbolkonfiguration hinzufügen'

Funktion: Der Dialog dient dem Definieren von Voreinstellungen für ein Objekt "Symbolkonfiguration".

Aufruf: Über das Kontextmenü des Objekts "Applikation."

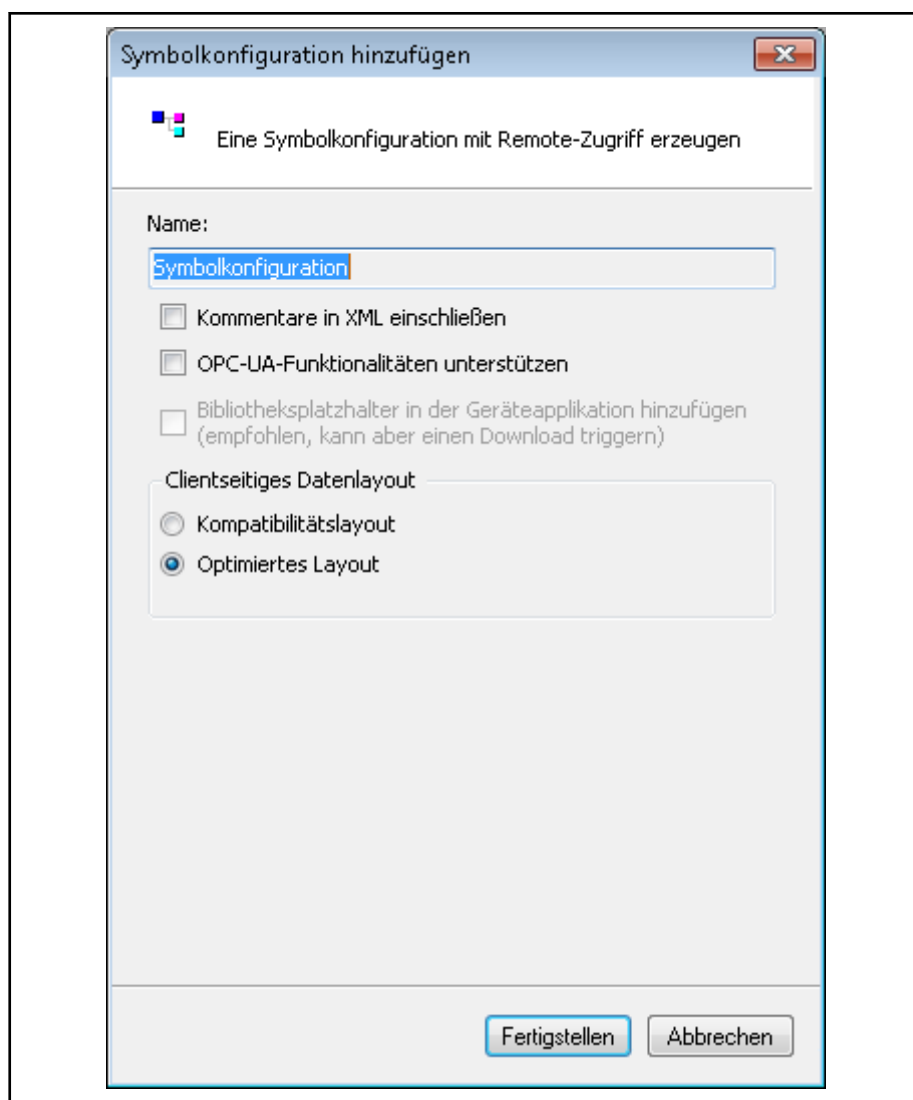


Abb. 18-35: Dialog: Symbolkonfiguration hinzufügen

Dialogelemente

Name	Name der Symbolkonfiguration
Kommentare in XML einschließen	IndraLogic exportiert die den Variablen zugewiesenen Kommentare mit in die Symboldatei.

OPC UA-Funktionalitäten unterstützen	Hinweis: Verfügbarkeit und Bearbeitbarkeit dieser Option sind geräteabhängig. : Beim Download der Symbolkonfiguration werden zusätzliche Informationen auf die Steuerung geladen, die für den Betrieb des OPC UA-Servers notwendig sind: • Basistypen von abgeleiteten Funktionsbausteinen • Inhalte von Attributen, die über Compiler-Pragmas vergeben werden • Variablentyp (INPUT / OUTPUT / IN_OUT)
Clientseitiges Datenlayout Sehen Sie ausführlichere Informationen und Beispiele zu den Layout-Optionen im nachfolgenden Abschnitt "Symbolkonfigurationseditor".	
Kompatibilitätslayout	Diese Einstellung dient der Kompatibilität im Fall von alten Projekten. Das für den Client erzeugte Datenlayout wird soweit als möglich dem intern vom Compiler erzeugten Layout angeglichen.
Optimiertes Layout	Ab IndraWorks 14V12 (Compiler Version 3.5.7.0) Empfohlen für neue Projekte. Berechnet das Ausgangslayout in der optimierten Form, entkoppelt vom internen Compilerlayout. Hinterlässt keine Lücken für nicht veröffentlichte Elemente und hält die Anforderungen an das Alignment der Datentypen strikt ein.

Siehe:

- [Kap. 10.2 "Symbolkonfiguration" auf Seite 167](#)

Symbolkonfigurationseditor

Der Editor enthält eine Tabelle mit den bereits ausgewählten Variablen, eine Menüleiste für die Bearbeitung und ein Feld mit jeweils unterstützenden Informationen und Schaltflächen für die nötigen Aktionen.

Menüleiste

Ansicht	<i>Über diese Schaltfläche können Sie folgende Variablen für die Aufnahme in den Konfigurationseditor ein- und ausschalten:</i> • Unkonfigurierte aus Projekt: Variablen, die der Symbolkonfiguration noch nicht hinzugefügt sind, aber im Projekt dafür bereitstehen. • Unkonfigurierte aus Bibliotheken: Variablen, die der Symbolkonfiguration noch nicht hinzugefügt sind, aber in eingebundenen Bibliotheken dafür bereitstehen. • Symbole via Attribut exportiert: Der Filter bewirkt, dass IndraLogic auch Variablen auflistet, die bereits über das Pragmaattribut <code>{attribute 'symbol' := 'read'}</code> für den Export in die Symboldatei gekennzeichnet sind. Diese Symbole sind grau dargestellt. Die Spalte Attribut zeigt jeweils, welches Zugriffsrecht durch das Pragma gesetzt ist.
Erstellen	Übersetzen des Projekts. Voraussetzung für eine aktuelle Bereitstellung der Variablen im Konfigurationseditor.

Einstellungen	• OPC-UA-Funktionalitäten unterstützen (Geräteabhängig): : Beim Download der Symbolkonfiguration werden zusätzliche Daten auf die Steuerung geladen, die für den Betrieb des OPC UA Servers notwendig sind. Dies beinhaltet derzeit die folgenden Informationen: – Basistypen von abgeleiteten Funktionsbausteinen – Inhalte von Attributen, die über Compiler-Pragmas vergeben werden – Variablentyp (VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT) • Kommentare in XML einbinden: IndraLogic exportiert die den Variablen zugewiesenen Kommentare mit in die Symboldatei. • Node-Flags in XML aufnehmen:  Die Namensraum- Node-Flags stellen zusätzliche Informationen über den Ursprung eines Knotens im Namensraum bereit. Die Node-Flags sind immer in der Symboltabelle, wenn OPC UA aktiviert ist. Ihre Einbindung in der XML-Datei kann jedoch deaktiviert werden, da einige fehlerhafte Parser Probleme damit haben. • Kommentare und Attribute konfigurieren: In dem Dialog "Kommentare und Attribute" konfigurieren Sie, welche Inhalte in der Symbolkonfiguration und in der XML-Datei enthalten sind. • Synchronisierung mit IEC-Tasks konfigurieren: Öffnet den Dialog "Eigenschaften - <SPS-Name>", Registerkarte "Optionen". Diese Einstellung erlaubt den symbolischen Clients (beispielsweise Visualisierungen oder Datenbankanbindungen auf Basis des PLC-Handlers) den konsistenten, mit den IEC-Tasks synchronisierten Lese- bzw. Schreibzugriff. Eine genaue Beschreibung dieser Einstellung finden Sie weiter unten im Abschnitt "Einstellung: Synchronisierung mit IEC-Tasks konfigurieren" Hinweis: Die Variablenzugriff, der synchron zu den IEC-Tasks ist, kann den Jitter für alle IEC-Applikationen auf diesem Gerät erhöhen! Der synchronisierte konsistente Zugriff kann die Echtzeitfähigkeit stören! • Auswahl, nach welchem Typ das Datenlayout für den Client der Symbolkonfiguration angelegt werden soll: Hinweis: Sehen Sie dazu auch den Abschnitt "Beispiele zu den Datenlayouttypen" am Ende dieser Hilfeseite. Kap. 18.1.24 "Objekt 'Symbolkonfiguration'" auf Seite 588 – Kompatibilitätslayout: Diese Einstellung dient der Kompatibilität im Fall von alten Projekten. Das für den Client erzeugte Datenlayout wird soweit als möglich dem intern vom Compiler erzeugten Layout angeglichen. Aufgrund der historisch gewachsenen Konfigurationsmöglichkeiten der Symbolkonfiguration kann es jedoch zu problematischen Verschiebungen kommen. Mögliche Ursachen: – Speicherlücken durch interne Pointer, oder Referenzen in Bausteinen und durch Strukturmembers, die nicht in die Symbolkonfiguration freigegeben werden – Je nach Datentyp (<code>__XINT</code>/<code>__XWORD</code> etc.) auch unterschiedliche Speicherlücken bei 32 und 64 Bit Systemen – Unabsichtlich falsches Alignment durch Verwenden der Attribute <code>pack_mode</code>, oder <code>relative_offset</code> – Optimiertes Layout (Empfohlen für neue Projekte): Berechnet das Ausgangslayout in der optimierten Form, entkoppelt vom internen Compilerlayout. Hinterlässt keine Lücken für nicht veröffentlichte Elemente und hält die Anforderungen an das Alignment der Datentypen strikt ein. Erfordert mindestens Compilerversion 3.5.7.0. ab IndraWorks 14V12 • Direkten E/A-Zugriff aktivieren : WARNUNG: Diese Funktionalität ist möglicherweise gefährlich und nicht für den Produktionsbetrieb vorgesehen. Aktivieren Sie die Einstellung nur zur Fehlersuche, für Tests, oder während der Inbetriebnahme der Maschine (beispielsweise für das Überprüfen von Kabeln und Verbindungen)

Download	Wenn Sie ein Gerät verwenden, das eine eigene Applikationsdatei für die Symbolkonfiguration unterstützt, gibt es in der Symbolleiste zusätzlich diese Schaltfläche. Wenn Sie die Symbolkonfiguration im Online-Betrieb geändert haben, können Sie damit sofort die neue <Applikationsname>._Symbols-Datei auf die Steuerung laden. siehe auch: Kap. 10.2 "Symbolkonfiguration" auf Seite 167
Tools	"XML-Schema-Datei speichern": Dieser Befehl öffnet den Standarddialog zum Speichern einer Datei im Dateisystem. Sie können damit das XSDSchema der Symboldatei beispielsweise zur Verwendung in externen Programmen bereitstellen

Symbole

Zugriffsrechte	Sie können das Zugriffsrecht für ein Symbol durch einen Mausklick auf das Symbol in der Spalte Zugriffsrechte verändern. <i>Symbole für Zugriffsrechte</i> : nur lesen : nur schreiben : lesen und schreiben : ausführen Dieses Recht ermöglicht ausführenden Zugriff auf Funktionen, Funktionsbausteine, Methoden und Programme. Voraussetzungen für die Zuweisung: Das Gerät bietet die Optionen "Aufrufe von Funktionen, FBs, Methoden und Programmen unterstützen" und "OPC UA Funktionalitäten unterstützen". Beide Optionen sind in den "Einstellungen" aktiviert.
Maximal	Zugriffsrecht, das Sie maximal vergeben können.
Typ	Ab SPS-Programmiersystem V3.5 SP6 werden auch Alias-Datentypen angezeigt. Beispiel: <code>MY_INT : INT</code> für eine Variable, die mit Datentyp <code>MY_INT</code> (Typ <code>INT</code>) deklariert wurde.
Member-Variablen	Sie können Variablen eines strukturierten Datentyps ebenfalls durch Setzen eines Häkchens in der Spalte Symbole zur Symbolkonfiguration hinzufügen. Dies bewirkt zunächst, dass IndraLogic für alle Member-Variablen-Symbole in die Symboldatei exportiert. Über die Schaltfläche  in der Spalte Member-Variablen können Sie jedoch gezielt einzelne Komponenten der Struktur auswählen. Hinweis: Diese Auswahl gilt dann für alle Instanzen dieses Datentyps, für welche die Symbole exportiert werden! Wenn kein Member eines strukturierten Typs ausgewählt ist, erscheint in den Checkboxes der Members ein Sternchen, um zu zeigen, dass alle exportierbaren Members des Typs exportiert werden.

Einstellung: Synchronisierung mit IEC-Tasks konfigurieren

Für den synchronen konsistenten Zugriff wird im Laufzeitsystem bei der Bearbeitung einer Lese- oder Schreib Anfrage vom symbolischen Client zunächst gewartet, bis ein Zeitpunkt gefunden wird, in dem keine IEC-Task ausgeführt wird. Sobald diese Lücke gefunden wird, wird der Neuanlauf der IEC-Tasks solange verhindert, bis alle Werte der Variablenliste kopiert wurden. Danach werden die IEC-Tasks wieder normal geschickt. Durch den synchronisierten konsistenten Zugriff kann es dadurch zu einem verzögerten Start von IEC-Tasks kommen, was sich als erhöhter Jitter zeigt. Da im Laufzeitsystem alle Applikationen von einem gemeinsamen Scheduler verwaltet werden, betrifft diese potentielle Verschlechterung des Echtzeitverhaltens alle Applikationen auf dem Gerät. Es sind also alle Applikationen des Geräts betroffen, unabhängig davon, ob diese eine Symbolkonfiguration enthalten oder ob diese aus einem oder mehreren IndraLogic-Projekten auf die Steuerung

rung geladen wurden. Daher lässt das Laufzeitsystem den synchronisierten konsistenten Zugriff nur zu, wenn dies alle Applikationen erlauben, die zum Zugriffszeitpunkt in der Steuerung geladen sind.



Die Einstellung ist über den Editor der Symbolkonfiguration Registerkarte "Einstellungen" erreichbar.



Nach Ändern der Einstellung müssen alle Applikationen auf dem Gerät per Download/Online Change neu geladen werden und alle Bootapplikationen aktualisiert werden.

In welchen Fällen wird ein synchronisierter konsistenter Zugriff benötigt?

In der Regel besteht bei visualisierten Werten kein Bedarf für konsistente Werte, da es bei sich ändernden Werten meist nicht relevant ist, aus welchem IEC-Task-Zyklus diese stammen. Bei sich selten ändernden Werten sowieso nicht. Auch beim Schreiben bestehen meist keine harten Konsistenzanforderungen, da sich die Maschine z.B. beim Schreiben von Rezepturen, typischerweise in einer Art von Standby-Modus befinden muss, in dem kein direkter Zugriff auf die als Rezeptur geschriebenen Werte erfolgt.

Dagegen werden insbesondere für Datenbankverbindungen zur Speicherung von Produktionsdaten konsistente Werte benötigt. Bei getakteten Maschinen müssen diese Werte aber synchron zum Produktionstakt (also pro hergestelltem Produkt ein Wertesatz) und nicht konsistent in Bezug auf eine oder mehrere IEC-Tasks sein. Die Konsistenz in Bezug auf den Maschinentakt muss aber bereits durch die IEC-Applikation gewährleistet sein. Hierfür werden typischerweise die während eines Produktionstakts anfallenden Werte in einer globalen Variablenliste gesammelt. Am Ende des Takts wird mit einer weiteren Variablen (BOOL oder Zähler) dem symbolischen Client mitgeteilt, dass der Maschinentakt beendet und somit die Werte gültig sind. Nun hat der Client die Chance die zum Produktionstakt gehörenden Werte zu archivieren. Je nach Notwendigkeit kann das erfolgreiche Lesen auch in der Gegenrichtung mittels einer freigegebenen Variablen angezeigt werden, so dass gegebenenfalls die Produktion angehalten werden kann, wenn die Produktionsdaten nicht archiviert werden konnten. Für diesen Anwendungsfall ist der synchronisierte konsistente Zugriff nicht notwendig und hilfreich, da die Synchronisierung ja auf Applikationsebene zu erfolgen hat.

Im Gegensatz dazu liegt der typische Einsatz des synchronisierten, konsistenten Zugriffs durch symbolische Clients vor allem in der Prozessindustrie mit kontinuierlich laufenden Anlagen ohne Produktionstaktung, wenn zyklische Prozesswerte konsistent in einem festen Zeitraster geschrieben werden sollen.

Dies kann entweder über Synchronisierung auf Applikationsebene analog zu getakteten Maschinen (siehe oben), oder aber auch über den des synchronisierten konsistenten symbolischen Zugriff erfolgen. Letzteres hat den Vorteil, dass keine Logik im IEC-Programm implementiert werden muss und der Zugriff rein durch den Client gesteuert werden kann.

⚠ VORSICHT

Aufgrund des erhöhten Jitters ist das synchronisierte konsistente Monitoring nicht für Motion oder echtzeitkritische Anwendungen geeignet. Aus diesen Gründen sollte der synchronisierte konsistente Zugriff nur freigegeben und verwendet werden, wenn dies unbedingt erforderlich ist.

Wenn ein Client den durch diese Einstellung freigegebenen synchronen konsistenten Zugriff verwendet, hat dies auch Auswirkungen auf den Client. Zum einen kann auch hier je nach Scheduler des Laufzeitsystems die Antwortzeit auf Lese-/ Schreibzugriffe mehr jittern, da gegebenenfalls noch auf eine Ausführungslücke der IEC-Tasks gewartet werden muss. Weiterhin können Lese- oder/und Schreibzugriffe scheitern, wenn IEC-Tasks sehr lange laufen (im Bereich von mehreren 100 ms) oder die CPU-Auslastung durch eine oder mehrere IEC-Tasks über einen längeren Zeitraum (im Bereich von mehreren 100 ms) bei nahezu 100% liegt. Die Verfügbarkeit der Werte hängt damit also auch von der Auslastung der Steuerung durch die IEC-Applikation ab.

Darüber hinaus kann der Client die Auswirkungen auf sich selbst und auf das Laufzeitsystem minimieren, wenn er bei der Definition der zu lesenden/schreibenden Variablenlisten Folgendes beachtet:

- Nur auf diejenigen Variablen synchronisiert konsistent zugreifen, die unbedingt konsistent benötigt werden
- Getrennte Variablenlisten für Variablen, die konsistent sein müssen und für Variablen, die inkonsistent sein dürfen
- Variablenlisten mit sehr vielen konsistenten Variablen in mehrere kleinere Listen aufteilen
- Leseintervalle für zyklisches Lesen von Werten so groß möglich wählen.

Unterstützung bei der aktuellen Konfiguration und mögliche Korrekturmaßnahmen

Rot markierte Einträge in der Symboltabelle zeigen Variablen an, die für den Export in die Symboldatei konfiguriert sind, aber gerade in der Applikation nicht gültig sind. Ursache kann beispielsweise sein, dass die Deklaration im Baustein entfernt wurde.

Ab Version 14VRS erhalten Sie einen Warnhinweis im Editor, wenn Variablen, für die Symbole konfiguriert sind, im IEC-Code nicht verwendet oder im Falle von E/A-Variablen nicht gemappt werden. Der Compiler weist außerdem auf Variablen hin, die in der Symbolkonfiguration aus veralteten Bibliotheksversionen referenziert werden.



Variablen von Objekten, die im Programmcode nicht verwendet werden, werden standardmäßig nicht kompiliert und sind somit nicht in der Symbolkonfiguration verfügbar.

Aber: Variablen von nicht-kompilierten Objekten werden von IndraLogic dennoch in der Symbolkonfiguration bereitgestellt, wenn eine der folgenden Bedingungen erfüllt ist:

- Die Bausteineigenschaft "Immer binden" ist aktiviert.
- Das Pragma-Attribut `{attribute 'linkalways'}` wird verwendet.

Siehe auch

- [Kap. 10.2 "Symbolkonfiguration" auf Seite 167](#)
- [Kap. 10 "Mit Steuerungsnetzwerken arbeiten" auf Seite 163](#)
- [Kap. "Dialog 'Eigenschaften- 'Übersetzen'" auf Seite 749](#)

- [Kap. "Attribut 'linkalways'" auf Seite 419](#)

Beispiele zu den Datenlayout-Typen

Im Folgenden sehen Sie Beispiele aus einer IEC-Applikation, bei denen es durch nicht veröffentlichte Symbole, durch interne "unsichtbare" Pointer, oder durch eine "pack mode"-Definition über die Gerätebeschreibung zu Lücken im clientseitigen Speicherlayout kommen kann. Mit der Einstellung "Optimiertes Layout" werden die Lücken vermieden. Die Symboldatei enthält also je nach ausgewählter Layout-Einstellung unterschiedliche Angaben für Größe und Offset von Speicherstellen.

Große Struktur

```
// Beispiel einer großen Struktur, von der nicht alle Members veröffentlicht werden :
// Example of a big structure, where not all members get published :
STRUCT
{attribute 'symbol':='readwrite'}
PublicNumber : INT;

{attribute 'symbol':='none'}
InternalData : ARRAY[0..100] OF BYTE;

{attribute 'symbol':='readwrite'}
SecondNumber : INT;

{attribute 'symbol':='none'}
MoreData : ARRAY[0..100] OF BYTE;
END_STRUCT
END_TYPE
```

Resultierende Einträge in der Symboldatei, beachten Sie "size" und "byteoffset":

Symboldatei, Große Struktur, Option Kompatibilitätslayout

```
<TypeUserDef name="T_GrosseStruktur" size="208" nativesize="208"
typeclass="Userdef" pouclass="STRUCTURE" iecname="GrosseStruktur">

<UserDefElement iecname="PublicNumber" type="T_INT" byteoffset="0"
vartype="VAR"></UserDefElement>

<UserDefElement iecname="SecondNumber" type="T_INT" byteoffset="104"
vartype="VAR"></UserDefElement>

</TypeUserDef>
```

Symboldatei, Große Struktur, Option Optimiertes Layout

```
<TypeUserDef name="T_GrosseStruktur" size="4" nativesize="208"
typeclass="Userdef" pouclass="STRUCTURE" iecname="GrosseStruktur">

<UserDefElement iecname="PublicNumber" type="T_INT" byteoffset="0"
vartype="VAR"></UserDefElement>

<UserDefElement iecname="SecondNumber" type="T_INT" byteoffset="2"
vartype="VAR"></UserDefElement>

</TypeUserDef>
```

Struktur mit ungeraden Adressen

```
// Die folgenden Mechanismen können zu Members mit nicht korrektem
// Alignment führen: / The following mechanisms can cause misalignment:
// - {attribute 'relative_offset':='...'} an einem Member / at a member
// - {attribute 'pack mode':='...'} in einer Strukturdeklaration / at a
//   structure declaration
// - target setting 'memory-layout\pack-mode' in der Gerätebeschreibung /
//   in the device description

{attribute 'pack_mode':='1'}
TYPE UngeradeAdressen :
STRUCT
{attribute 'relative_offset':='3'}
{attribute 'symbol':='readwrite'}
PublicNumber : INT;
```

```
{attribute 'symbol':='readwrite'}
PublicValue : LREAL;
END_STRUCT
END_TYPE
```

Resultierende Einträge in der Symboldatei, beachten Sie "size" und "byteoffset":

Symboldatei, Struktur mit ungeraden Adressen, Option Kompatibilitätslayout

```
<TypeUserDef name="T_UngeradeAdressen" size="13" nativesize="13"
typeclass="Userdef" pouclass="STRUCTURE" iecname="UngeradeAdressen">

<UserDefElement iecname="PublicNumber" type="T_INT" byteoffset="3"
vartype="VAR"></UserDefElement>

<UserDefElement iecname="PublicValue" type="T_LREAL" byteoffset="5"
vartype="VAR"></UserDefElement>

</TypeUserDef>
```

Symboldatei, Struktur mit ungeraden Adressen, Option Optimiertes Layout

```
<TypeUserDef name="T_UngeradeAdressen" size="16" nativesize="13"
typeclass="Userdef" pouclass="STRUCTURE" iecname="UngeradeAdressen">

<UserDefElement iecname="PublicNumber" type="T_INT" byteoffset="0"
vartype="VAR"></UserDefElement>

<UserDefElement iecname="PublicValue" type="T_LREAL" byteoffset="8"
vartype="VAR"></UserDefElement>

</TypeUserDef>
```

Funktionsbaustein

```
// Each POU contains some implicit variables, which do not get published.
// Depending on the data type these might cause memory gaps of different sizes.
// FUNCTION_BLOCK Baustein IMPLEMENTS SomeInterface.

VAR_INPUT
in : INT;
END_VAR
VAR_OUTPUT
out : INT;
END_VAR
VAR_
VAR_
END_VAR
```

Jeder Baustein enthält interne Variablen, die nicht veröffentlicht werden. Wenn es sich dabei um Datentypen wie beispielsweise `__XWORD` handelt, entstehen dadurch im clientseitigen Datenlayout unterschiedlich große Speicherlücken, je nachdem, ob auf einem 64-Bit- oder 32-Bit-System gearbeitet wird.

Resultierende Einträge in der Symboldatei für 64 Bit und 32 Bit beachten Sie "size" und "byteoffset":

Symboldatei, Funktionsbaustein, Option Kompatibilitätslayout, 64 Bit

```
<TypeUserDef name="T_Baustein" size="24" nativesize="24"
typeclass="Userdef" pouclass="FUNCTION_BLOCK" iecname="Baustein">

<UserDefElement iecname="in" type="T_INT" byteoffset="16"
vartype="VAR_INPUT"></UserDefElement>

<UserDefElement iecname="out" type="T_INT" byteoffset="18"
vartype="VAR_OUTPUT"></UserDefElement>

</TypeUserDef>
```

Symboldatei, Funktionsbaustein, Option Optimiertes Layout, 64 Bit

```
<TypeUserDef name="T_Baustein" size="4" nativesize="24"
typeclass="Userdef" pouclass="FUNCTION_BLOCK" iecname="Baustein">

<UserDefElement iecname="in" type="T_INT" byteoffset="0"
vartype="VAR_INPUT"></UserDefElement>

<UserDefElement iecname="out" type="T_INT" byteoffset="2"
vartype="VAR_OUTPUT"></UserDefElement>

</TypeUserDef>
```

Symboldatei, Funktionsbaustein, Option Kompatibilitätslayout, 32 Bit

```
<TypeUserDef name="T_Baustein" size="12" nativesize="12"
typeclass="Userdef" pouclass="FUNCTION_BLOCK" iecname="Baustein">

<UserDefElement iecname="in" type="T_INT" byteoffset="8"
vartype="VAR_INPUT"></UserDefElement>

<UserDefElement iecname="out" type="T_INT" byteoffset="10"
vartype="VAR_OUTPUT"></UserDefElement>

</TypeUserDef>
```

Symboldatei, Funktionsbaustein, Option Optimiertes Layout, 32 Bit

```
<TypeUserDef name="T_Baustein" size="4" nativesize="12"
typeclass="Userdef" pouclass="FUNCTION_BLOCK" iecname="Baustein">

<UserDefElement iecname="in" type="T_INT" byteoffset="0"
vartype="VAR_INPUT"></UserDefElement>

<UserDefElement iecname="out" type="T_INT" byteoffset="2"
vartype="VAR_OUTPUT"></UserDefElement>

</TypeUserDef>
```

18.1.25 Objekt 'Taskkonfiguration'

Symbol: 

Das Objekt dient der Definition und Anzeige der Grundeinstellungen für die Taskkonfiguration.

Das Objekt **Taskkonfiguration** muss in jeder Applikation genau einmal enthalten sein.

Registerkarten der Taskkonfiguration und ihre Funktionen

- **Eigenschaften:** Anzeige der Grundeinstellungen
- **System-Ereignisse:** Verknüpfung von Bausteinaufrufen mit Systemereignissen
- **Überwachung:** Anzeige des Status und aktueller Statistiken zu den Zykluszeiten im Online-Betrieb

Siehe auch

- [Kap. 9.15.1 "Erstellen einer Taskkonfiguration" auf Seite 144](#)
- [Kap. 18.1.26 "Objekt 'Task'" auf Seite 602](#)

Registerkarte 'Eigenschaften'

Objekt: Taskkonfiguration

In der Registerkarte definieren Sie die Grundeinstellungen der Taskkonfiguration, die das Zielsystem vorgibt, wie zum Beispiel Maximalwerte für Tasks und Watchdog-Parameter.

Registerkarte 'Systemereignisse'

Objekt: **Taskkonfiguration**

In der Registerkarte **Systemereignisse** definieren Sie, welches Ereignis welche Funktion aufruft und ob die Konfiguration aktuell aktiviert ist oder nicht. Sie verwenden die Registerkarte, wenn anstelle einer Task ein "Systemereignis" eine Funktion des Projekts aufrufen soll.

Event-Handler hinzufügen	Öffnet den Dialog Event-Handler hinzufügen .
Event-Handler löschen	Löscht die selektierte Zuweisung der Auflistung.
Ereignisinfo	Informationen aus der jeweiligen Ereignisbibliothek.
Ereignisfunktion öffnen	Öffnet den Editor der neuen Funktion der selektierten Zuweisung. Die Implementierungssprache der neuen Funktion haben Sie im Dialog Event-Handler hinzufügen ausgewählt.
Auflistung der Zuweisungen von aufzurufenden Funktionen zu Ereignissen mit: Name , Beschreibung , Aufzurufende Funktion und Aktiv (Aktivierung/Deaktivierung der Konfiguration).	

Fügt der Auflistung eine neue Zuweisung: "Ereignis - aufzurufende Funktion" hinzu.

Ereignis	Die Auswahlmöglichkeiten sind abhängig vom Zielgerät. Ereignisse, die nicht zur Verfügung stehen, kennzeichnet IndraLogic mit einem roten Symbol vor dem Namen. Eine Auflistung aller möglichen Systemereignisse befindet sich am Ende dieses Abschnitts.
Aufzurufende Funktion	Namen der Funktion (POU vom Typ FUNCTION) Sie müssen den Namen einer neuen Funktion eingeben. IndraLogic fügt die Funktion nach Bestätigung des Dialogs im Projektexplorer ein.
Gültigkeitsbereich	- Applikation: Funktion ist für die Applikation verfügbar - POUs: Funktion ist für das ganze Projekt verfügbar
Implementierungssprache	Implementierungssprache der neuen Funktion
Beschreibung	Kurzbeschreibung zum ausgewählten Ereignis

Besonderheiten im Online-Betrieb

Die Auflistung der Zuweisungen von aufzurufenden Funktionen zu Ereignissen enthält zusätzlich folgende Informationen: **Ereignisstatus** und **Anzahl Aufrufe** und die Schaltfläche **Online Reset**.

Ereignisstatus	0: Es ist kein Fehler aufgetreten. Ungleich 0: Fehlerfall. Sie müssen die entsprechende Laufzeitsystem-Dokumentation hinzuziehen.
Anzahl Aufrufe	Zeigt an, wie oft das Ereignis aufgetreten ist oder die zugehörige Funktion aufgerufen wurde.
Online Reset	IndraLogic initialisiert die Ereignislisten neu und setzt den Zähler der Ereignisse/Funktionsaufrufe zurück. Nicht korrekt initialisierte Ereignisse werden mit einer roten Statuszelle angezeigt.

Mögliche Systemereignisse

Event	Beschreibung	Task	Debuggen
PrepareStart	Aufruf vor Start der Applikation	Kommunikationstask	nein
StartDone	Aufruf nach Start der Applikation	Kommunikationstask	nein

Referenz Benutzeroberfläche

Event	Beschreibung	Task	Debuggen
PrepareStop	Aufruf vor Stop der Applikation	Kommunikationstask	nein
StopDone	Aufruf nach Stop der Applikation	Kommunikationstask	nein
PrepareReset	Aufruf vor Reset der Applikation	Kommunikationstask	nein
ResetDone	Aufruf nach Reset der Applikation	Kommunikationstask	nein
PrepareOnlineChange	Aufruf vor Online-Change der Applikation	Kommunikationstask	nein
OnlineChangeDone	Aufruf nach Online-Change der Applikation	Kommunikationstask	nein
PrepareDownload	Aufruf vor Download der Applikation	Kommunikationstask	nein
DownloadDone	Aufruf nach Download der Applikation	Kommunikationstask	nein
PrepareDelete	Aufruf vor Löschen der Applikation	Kommunikationstask	nein
DeleteDone	Aufruf nach Löschen der Applikation	Kommunikationstask	nein
PrepareExit	Aufruf vor Verlassen der Applikation	Kommunikationstask	nein
ExitDone	Aufruf nach Verlassen der Applikation	Kommunikationstask	nein
CodeInitDone	Ereignis wird nach Code-Init gesendet. Wird innerhalb der Task Safe Section aufgerufen und nur bei Online-Change! (zum Beispiel wird der Copy-Code für Online-Change hier ausgeführt.)	Kommunikationstask	nein
Exception	Ereignis wird gesendet, wenn eine Exception im Kontext einer Applikation aufgetreten ist.	Exceptionshandlingtask des Laufzeitsystems oder - wenn das Laufzeitsystem eine solche nicht hat - Task selbst	Abhängig von der Task
Login	Login eines Clients in diese Applikation	Kommunikationstask	nein
Logout	Logout eines Clients aus dieser Applikation	Kommunikationstask	nein
BeforeReadingInputs	Aufruf vor Lesen der Eingänge	IEC-Task	ja
AfterReadingInputs	Aufruf nach Lesen der Eingänge	IEC-Task	ja
BeforeWritingOutputs	Aufruf vor Schreiben der Eingänge	IEC-Task	ja
AfterWritingOutputs	Aufruf nach Schreiben der Eingänge	IEC-Task	ja
DebugLoop	Ereignis wird zyklisch in der Debug-Schleife gesendet, wenn die IEC-Task auf einem Haltepunkt steht.	Kommunikationstask	nein
PrepareShutdown	Ereignis wird unmittelbar vor dem Herunterfahren des Laufzeitsystems gesendet.	Laufzeitsystem Hauptschleife	nein
PrepareExitComm	Ereignis wird vor dem Beenden (Exit) des Kommunikationsservers während des Herunterfahrens gesendet	Laufzeitsystem Hauptschleife	nein
PrepareExitTasks	Ereignis wird gesendet vor dem Beenden (Exit) aller Tasks während des Herunterfahrens.	Laufzeitsystem Hauptschleife	nein

Registerkarte 'Überwachung'

Objekt: **Taskkonfiguration**

Die Registerkarte zeigt im Onlinebetrieb folgende Informationen: den Status für Tasks der Taskkonfiguration und einige aktuelle Messungen zu den Zyklen und Zykluszeiten. IndraLogic aktualisiert die Werte im gleichen Zeitintervall wie beim Monitoring von Werten aus der Steuerung.

Die angezeigten Werte setzen Sie durch den Kontextmenübefehl **Zurücksetzen** auf den Wert 0 zurück.

Task	Taskname, wie in der Taskkonfiguration definiert
Status	• Nicht erzeugt: Task wurde seit der letzten Aktualisierung noch nicht gestartet; speziell für Ereignis-Tasks • Erzeugt: Task ist im Laufzeitsystem bekannt, ist aber noch nicht in Betrieb • Gültig: Task ist normal in Betrieb • Ausnahme: Task hat Ausnahmezustand
Anzahl IEC-Zyklen	Anzahl der seit dem ersten Starten der Applikation ausgeführten Zyklen, in denen der IEC-Code aufgerufen wurde; 0, wenn das Zielsystem die Zählfunktion nicht unterstützt.
Anzahl Zyklen	Anzahl der bereits ausgeführter Zyklen seit dem Einloggen auf die Steuerung. Es hängt vom Zielsystem ab, ob auch Zyklen gezählt werden, in denen die Applikation nicht läuft. In diesem Fall kann die Anzahl Zyklen größer als die Anzahl IEC-Zyklen sein.
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] Jitter ist der Unterschied zwischen dem Zeitpunkt, an dem die Task starten soll und dem Zeitpunkt, ab dem die Task dann wirklich startet. Beispiel: Wenn eine Task alle 1000 µs starten soll und sie startet erst wieder nach 1100 µs, dann ist der Jitter 100 µs.
Min. Jitter (µs)	Minimaler gemessener Jitter in [µs]
Max. Jitter (µs)	Maximaler gemessener Jitter in [µs]
Core	Nummer des Prozessorkerns, auf dem die Task gerade ausgeführt wird. Beispiel: 2 Voraussetzung: Die Steuerung verfügt über einen Mehrkernprozessor.

Registerkarte 'Variablenverwendung'

Objekt: **Taskkonfiguration**

In der Registerkarte "Variablenverwendung" erhalten Sie eine Übersicht über alle Variablen und deren Verwendung. Sie sehen darin, in welchen Tasks die Zugriffe auf die Variablen erfolgen.

Bei der Verwendung von Multicore sollte der Schreibzugriff (w) auf eine Variable nur in einer Task erfolgen, da es sonst zu Inkonsistenzen kommen kann.

Über das Kontextmenü können Sie einzelne Tasks ausblenden und die Querverweisliste zu Variablen anzeigen lassen.

Variablen	Name der Variablen
Typ	Datentyp
Anzahl	Anzahl der Tasks, die auf diese Variable zugreifen.
<Taskname>	Zugriff auf die Variable (r: lesend, w: schreibend, rw: lesend/schreibend)

Siehe auch [Kap. 9.15 "Taskkonfiguration" auf Seite 143](#)

Registerkarte 'Taskgruppen'

Objekt: **Taskkonfiguration**

Voraussetzung: Multicore-Steuerung

In der Registerkarte "Taskgruppen" definieren Sie Taskgruppen. Taskgruppen können bei Multicore-Systemen auf die einzelnen Prozessorkerne verteilt werden. Tasks einer Taskgruppe laufen immer auf dem gleichen Prozessorkern.

Gruppe hinzufügen	Die Schaltfläche fügt eine neue Taskgruppe mit dem Namen NewGroup_<nr> hinzu
Gruppe entfernen	Löscht die selektierte Taskgruppe
Gruppenname	Durch einen Doppelklick in das Feld kann der Namen geändert werden
Core	Legt fest, mit welchem Prozessorkern die Tasks dieser Gruppe abgearbeitet werden. "FreeFloating": Alle Tasks sind dynamisch an verschiedene Prozessorkerne gebunden - der Anwender kann keinen Einfluss darauf nehmen, die Zuteilung erfolgt durch das Betriebssystem. - SequentialPinned: Alle Tasks sind fest an verschiedene Prozessorkerne gebunden - der Anwender kann keinen Einfluss darauf nehmen. "FixedPinned": Alle Tasks werden an einen Prozessorkern gebunden. Welcher Prozessorkern dies ist, wird im Laufzeitsystem entschieden (Standardeinstellung). "<Core-Nummer>": Fest definierter Prozessorkern. Wenn der Prozessorkern nicht vorhanden ist, wird eine Fehlermeldung ausgegeben.

Siehe auch [Kap. 9.15 "Taskkonfiguration" auf Seite 143](#)

18.1.26 Objekt 'Task'

Symbol: 

In dem Objekt definieren Sie die Bedingungen für den Start der Task und die Aufrufe der Task..

Sie fügen das Objekt im Projektextplorer unter der **Taskkonfiguration** ein.

Siehe auch

- [Kap. "Registerkarte 'Konfiguration'" auf Seite 602](#)
- [Kap. 18.1.25 "Objekt 'Taskkonfiguration'" auf Seite 598](#)

Registerkarte 'Konfiguration'

Objekt: **Task**

Priorität	Mögliche Werte: 1..20, wobei 1 die höchste Priorität darstellt
Taskgruppe (Voraussetzung: Multicore-Steuerung)	Zugeordnete Taskgruppe. Diese Zuordnung wird im Gerätebaum in Klammern angezeigt. Taskgruppen können bei Multicore bestimmten Prozessorkernen zugewiesen werden. Die Taskgruppe wird im Gerätebaum hinter der Task in Klammern angezeigt.

Typ

Zyklisch	IndraLogic arbeitet die Task zyklisch ab. Die Zykluszeit der Task ist im Feld Intervall definiert.
Ereignis	IndraLogic startet die Abarbeitung der Task, sobald die globale Variable, die im Eingabefeld Ereignis definiert ist, eine steigende Flanke erhält.
Freilaufend	IndraLogic startet die Abarbeitung der Task bei Programmstart und am Ende eines Durchlaufs automatisch neu in einer fortlaufenden Schleife. Es ist keine Zykluszeit definiert.
Extern	IndraLogic startet die Abarbeitung der Task, 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).
Intervall	Taskzykluszeit Obligatorisch für den Typ Zyklisch , wenn das Ereignis eine Zeitangabe erfordert. Zeitspanne, nach der die Task neu gestartet werden soll. Wenn Sie hier eine Zahl angeben, können Sie in der Auswahlliste hinter dem Eingabefeld die gewünschte Einheit auswählen. Wenn Sie die Einheit ms wählen, wird eine Eingabe automatisch im TIME-Format dargestellt (zum Beispiel $t\#200ms$), sobald das Fenster erneut den Fokus erhält. Sie können die Taskzykluszeit aber auch direkt im TIME-Format eingeben. Eingaben im $[\mu s]$ -Format werden immer als reine Zahl dargestellt.



Für Feldbusse ist ein festes Zyklusraster nötig, um ein determiniertes Verhalten sicherzustellen. Deshalb sollten Sie für eine Buszyklustask nicht den **Typ Freilaufend** verwenden.



Beachten Sie den folgenden Unterschied zwischen den Abarbeitungstypen **Status** und **Ereignis**:

Wenn das angegebene Ereignis den Wert **TRUE** hat, dann 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**. Wenn die Abtastrate des Task-Schedulers zu niedrig ist, können steigende Flanken des Ereignisses unbemerkt bleiben.

Watchdog

Definiert die Zeitüberwachung für eine Task. Wenn das Zielsystem eine erweiterte Watchdog-Konfiguration unterstützt, sind möglicherweise folgende Einstellungen in der Gerätebeschreibung vordefiniert:

- Obere und Untere Grenze
- Standard-Wachtdog-Zeit
- Zeitangabe in Prozent

Die hier standardmäßig verwendeten Watchdog-Einstellungen sind geräteabhängig.

Aktivieren	Watchdog ist aktiv. Wenn die Task die gerade gesetzte Zeit des Watchdogs überschreitet, wird die Task mit Fehlerstatus (Ausnahme) angehalten. Dabei werden auch die Applikation, in deren Task der Watchdog-Fehler aufgetreten ist, sowie deren Kindapplikationen gestoppt. Damit werden alle Tasks dieser betroffenen Applikationen ebenfalls gestoppt. Dabei wird auch die aktuell angegebene Empfindlichkeit mit eingerechnet. Wenn Sie in den SPS-Einstellungen der Steuerung die Option E/As aktualisieren aktiviert haben, setzt IndraLogic die Ausgänge auf die definierten Standardwerte zurück. <i>Mögliche Fälle:</i> • Mehrere aufeinander folgende Zeitüberschreitungen: Empfindlichkeit: 0, 1 - Exception in Zyklus 1 Empfindlichkeit: 2 - Exception in Zyklus 2 Empfindlichkeit: n - Exception in Zyklus n • Einmalige Zeitüberschreitung: Exception, wenn die Zykluszeit des aktuellen Zyklus länger ist als (Zeit * Empfindlichkeit). Beispiel: Zeit=t#10ms, Empfindlichkeit=5, das heißt Exception sobald die Task (einmalig) länger als 50 ms läuft.
Zeit (beispielsweise t#200ms)	Watchdog-Zeit Definiert zusammen mit der Empfindlichkeit die Zeitüberwachung für eine Task, wie bei Aktivieren beschrieben. Abhängig vom Zielsystem muss die Überwachungszeitspanne gegebenenfalls in Prozent des Taskintervalls angegeben werden. In diesem Fall ist die Auswahlliste für die Einheit inaktiv dargestellt und zeigt % an.
Empfindlichkeit	Zahl Definiert zusammen mit der Watchdog-Zeit die Zeitüberwachung für eine Task, wie bei Aktivieren beschrieben.



Sie können einen Watchdog mit Hilfe der Funktionen der Bibliothek `CmpIecTask.library` für bestimmte SPS-Zyklen abschalten. Dies ist für Zyklen nützlich, die aufgrund von Initialisierungen mehr Zeit beanspruchen können.

Beispiel

Deaktivieren/Reaktivieren des Watchdogs:

```
hIecTask := RTS_IEC_HANDLE //Declaration of the variable hIecTask
hIecTask := IecTaskGetCurrent(0);
IecTaskDisableWatchdog(hIecTask); // Watchdog disabled
...
IecTaskEnableWatchdog(hIecTask); Watchdog enabled
```

Auflistung der POU, welche die Task kontrollieren.

Die Aufrufreihenfolge entspricht der Anordnung der POU in der Liste von oben nach unten.

Aufruf hinzufügen	Definiert einen neuen Programmaufruf
Baustein öffnen	Öffnet die selektierte POU
Nach oben Nach unten	Änderung der Aufrufreihenfolge.

Siehe auch

- [Kap. 18.1.25 "Objekt 'Taskkonfiguration'" auf Seite 598](#)
- [Kap. "Registerkarte 'Taskgruppen'" auf Seite 602](#)

18.1.27 Objekt 'Trace'

Symbol: 

Das Objekt stellt Tracedaten in einem oder beliebig vielen Diagrammen dar. Zur Laufzeit der Applikation werden auf der Steuerung Werteverläufe von Tracevariablen erfasst. Diese werden in IndraLogic im Traceeditor dargestellt, so dass Sie die Werteverläufe beobachten können. Voraussetzung dafür ist die Erstellung einer Tracekonfiguration, das Übertragen der Tracekonfiguration auf die Steuerung und das Starten der Tracefunktionalität. Dann werden die erfassten Daten zum Entwicklungssystem übertragen und so wie konfiguriert in Diagrammen dargestellt. Sie können während dem Tracen in den Daten navigieren.

Mit einem Doppelklick auf das Objekt öffnet sich der Traceeditor. Die zugehörige Symbolleiste gibt Zugriff auf die wichtigsten Tracebefehle. Die Tracevariablenliste vermittelt einen Überblick über die Tracekonfiguration.

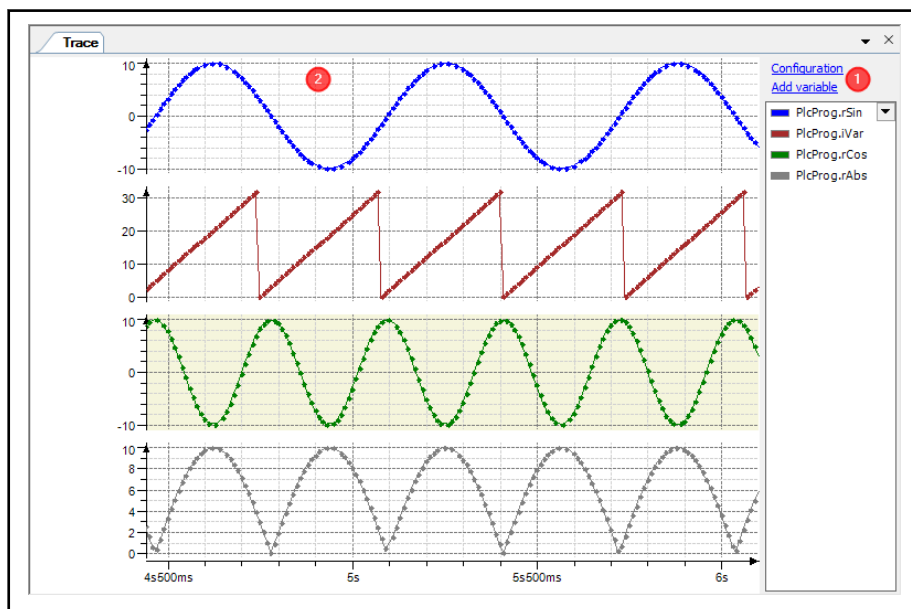


Abb. 18-36:

- (1): Ansicht der Tracekonfiguration
 - Configuration: **Konfiguration**
 - Add variable: **Variable hinzufügen**
- (2): Tracediagramm zur Laufzeit in Mehrkanal-Darstellung

Editor 'Trace'

Ansicht der Tracekonfiguration Die Ansicht vermittelt einen Überblick über die aktuelle Tracekonfiguration und ermöglicht die Bearbeitung der Konfiguration. Wenn Sie auf eine der Variablen doppelklicken, erscheint der Dialog **Tracekonfiguration** und zeigt unter **Variableneinstellung** die zugehörige Konfiguration an.

Außerdem stehen die folgenden Befehle und Kontextmenü-Befehle zur Verfügung.

Konfiguration	Dialog Tracekonfiguration erscheint mit selektiertem oberstem Knoten. Dabei ist rechts unter Aufzeichnungseinstellungen die Konfiguration der Datenerfassung angezeigt, um ein sofortiges Setzen der Einstellungen zu ermöglichen.
Variable hinzufügen	Dialog Tracekonfiguration erscheint mit einem neuen aber noch leeren Eintrag in der Tracekonfiguration. Dabei ist rechts unter Variableneinstellungen die Konfiguration des neuen Eintrags angezeigt, um ein sofortiges Setzen der Einstellungen zu ermöglichen.
	Auswahlliste mit Befehl Instanzpfade ausblenden erscheint
Instanzpfade ausblenden	<i>Umschaltung der Darstellung der Variablenliste</i> ☒ : Darstellung mit vollständigem Instanzpfad, zum Beispiel <code>PlcProg.iCounter</code>

Variable hinzufügen	Dialog Tracekonfiguration erscheint mit einem neuen aber noch leeren Eintrag in der Tracekonfiguration. Dabei ist rechts unter Variableneinstellungen die Konfiguration des neuen Eintrags angezeigt, um ein sofortiges Setzen der Einstellungen zu ermöglichen.
Variable entfernen	Die in der Ansicht selektierte Variable wird aus der Konfiguration entfernt.
Sichtbar	<i>Umschaltung der Sichtbarkeit der selektierten Variable im Tracediagramm:</i> ☒ : Graph der Variablen ist im Tracediagramm sichtbar
Anzeige	Dialog Anzeige Y-Achse erscheint.
Konfiguration	Dialog Tracekonfiguration erscheint mit den - Aufzeichnungseinstellungen .

Tab. 18-9: Kontextmenü

Siehe auch

- [Kap. "Dialog 'Tracekonfiguration'" auf Seite 783](#)

Tracediagramm zur Laufzeit

IndraLogic füllt und aktualisiert den Tracepuffer zur Laufzeit der Applikation mit den erfassten Samples und visualisiert die erfassten Daten im Tracediagramm als Graph über die Zeit. Wenn Sie den Traceeditor schließen, wird der Puffer freigegeben.

Sie können über Menübefehle die Datenerfassung steuern. Über Menübefehle, Tastaturkürzel und Mauseingaben können Sie in den Daten navigieren.

Siehe auch

- [Kap. 13.3.3 "Datenerfassung steuern" auf Seite 223](#)
- [Kap. 13.3.5 "Navigieren in den Tracedaten" auf Seite 223](#)

Navigieren im Tracediagramm mit der Maus

Sie können im Tracediagramm in den Daten navigieren, indem Sie mit Hilfe der Maus scrollen oder zoomen.

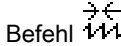
Voraussetzung	Benutzereingabe mit der Maus	Mauscursor bei Benutzereingabe	Auswirkung
	Ziehen Sie den Graphen entlang der X-Achse		Horizontales Scrollen des Tracegraphen entlang der Zeitachse
	Halten Sie <Strg> gedrückt und ziehen Sie den Graphen entlang der Y-Achse		Vertikales Scrollen des Tracegraphen entlang der Y-Achse.
Mehrkanal deaktiviert Zoomen mit Maus aktiviert:	Ziehen Sie ein Rechteck auf		IndraLogic zoomt auf das Rechteck
	Drehen Sie das Mausrad zurück		IndraLogic komprimiert die X-Achse wie beim Befehl  Komprimieren
	Drehen Sie das Mausrad vor		IndraLogic streckt die X-Achse wie beim Befehl  Strecken
	Halten Sie <Strg> gedrückt und drehen Sie das Mausrad zurück		IndraLogic komprimiert die Y-Achse
	Halten Sie <Strg> gedrückt und drehen Sie das Mausrad vor		IndraLogic streckt die Y-Achse
Mehrkanal aktiviert	Klicken Sie in eines der Tracediagramme		Das Tracediagramm ist selektiert
1 oder 2 Tracecursor vorhanden	Ziehen Sie das Dreieck eines Tracecursors auf eine andere Position		Horizontales Verschieben des Tracecursors entlang der Zeitachse
2 Tracecursor vorhanden	Klicken Sie auf den grauen Tracecursor		Grauer Tracecursor wird schwarz und ist damit selektiert

Siehe auch

- [Kap. "Befehl 'Zoomen mit Maus'" auf Seite 736](#)

Navigieren im Tracediagramm mit der Tastatur

Voraussetzung	Tastaturkürzel	Auswirkung
	<←> <→>	IndraLogic scrollt den Tracegraphen entlang der X-Achse
	<Strg> + <←> <Strg> + <→>	IndraLogic scrollt den Tracegraphen entlang der X-Achse mit großen Schritten
	<↑> <↓>	IndraLogic scrollt den Tracegraphen entlang der Y-Achse
	<Strg> + <↑> <Strg> + <↓>	IndraLogic scrollt den Tracegraphen entlang der Y-Achse mit großen Schritten
Mehrkanal aktiviert	<Pfeil nach oben> <↓>	IndraLogic scrollt den Tracegraphen im selektierten Tracediagramm entlang der X-Achse

Voraussetzung	Tastaturkürzel	Auswirkung
Mehrkanal aktiviert	<Strg> + <↑> <Strg> + <↓>	IndraLogic scrollt den Tracegraphen im selektierten Tracediagramm entlang der X-Achse mit großen Schritten
1 oder 2 Tracecursor vorhanden	<Alt> + <←> <Alt> + <→>	IndraLogic scrollt den Tracegraphen im selektierten Tracediagramm entlang der X-Achse
1 oder 2 Tracecursor vorhanden	<Strg> + <Alt> + <←> <Strg> + <Alt> + <→>	IndraLogic scrollt den Tracegraphen im selektierten Tracediagramm entlang der X-Achse mit großen Schritten
	<-->	IndraLogic komprimiert die X-Achse wie beim Befehl  Komprimieren
	<+>	IndraLogic streckt die X-Achse wie beim Befehl  Strecken
Mehrkanal deaktiviert	<Strg> + <-->	IndraLogic streckt die X-Achse
Mehrkanal deaktiviert	<Strg> + <+>	IndraLogic streckt die Y-Achse
Mehrkanal aktiviert	<Strg> + <-->	IndraLogic komprimiert die Y-Achse des selektierten Tracediagramms
Mehrkanal aktiviert	<Strg> + <+>	IndraLogic streckt die Y-Achse des selektierten Tracediagramme
Mehrkanal aktiviert	<Tab>	Selektion des nächsten unteren Tracediagramms
1 oder 2 Tracecursor vorhanden	<←> <→>	IndraLogic verschiebt den schwarzen Tracecursor
1 oder 2 Tracecursor vorhanden	<Strg> + <←> <Strg> + <→>	IndraLogic verschiebt den schwarzen Tracecursor mit großen Schritten
1 oder 2 Tracecursor vorhanden	<Umsch> + <←> <Umsch> + <→>	IndraLogic verschiebt den grauen Tracecursor
1 oder 2 Tracecursor vorhanden	<Strg> + <Umsch> + <←> <Strg> + <Umsch> + <→>	IndraLogic verschiebt den grauen Tracecursor mit großen Schritten

Siehe auch

- [Kap. "Dialog 'Tracekonfiguration'" auf Seite 783](#)
- [Kap. "Befehl 'Cursor'" auf Seite 733](#)
- [Kap. 13 "Applikation zur Laufzeit" auf Seite 205](#)

18.1.28 Objekt '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 Ver-

waltung der Zugriffsrechte wird im Rechte Dialog oder - für Objektzugriffsrechte - im Zugriffskontrolle Dialog vorgenommen. Er ist Teil des 'Eigenschaften'-Dialogs.

- Zugriffsrechte auf Objekte können "**vererbt**" werden.

Wenn ein Objekt ein "Vater"-Objekt 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 Projektexplorer dargestellt sind, und werden im 'Rechte'-Dialog gemäß der Syntax "<Vaterobjekt>.<Kindobjekt>" angezeigt.

Beispiel: Aktion 'ACT' ist dem POU-Objekt 'PlcProg' zugewiesen. Also wird 'ACT' im Projektnavigator unterhalb 'PlcProg' eingerückt dargestellt.

Im 'Rechte'-Dialog wird 'ACT' durch 'PlcProg.ACT' dargestellt, was zeigt, dass 'PlcProg' der "Vater" von 'ACT' ist.

Wenn das "modify"-Recht explizit für 'PlcProg' und eine bestimmte Benutzergruppe "verboten" wird, wird es auch beim Standardwert des "modify"-Rechts für 'ACT' "verboten".

Um ein Recht für ein SPS-Objekt umzusetzen, wird über den Objekt Kontextmenüpunkt **Eigenschaften** der Reiter **Zugriffskontrolle** selektiert. Anschließend wird das entsprechende Feld in der Tabelle selektiert und über die <Leertaste> oder einen Mausklick die Auswahlliste aufgeklappt und das gewünschte Recht ausgewählt.

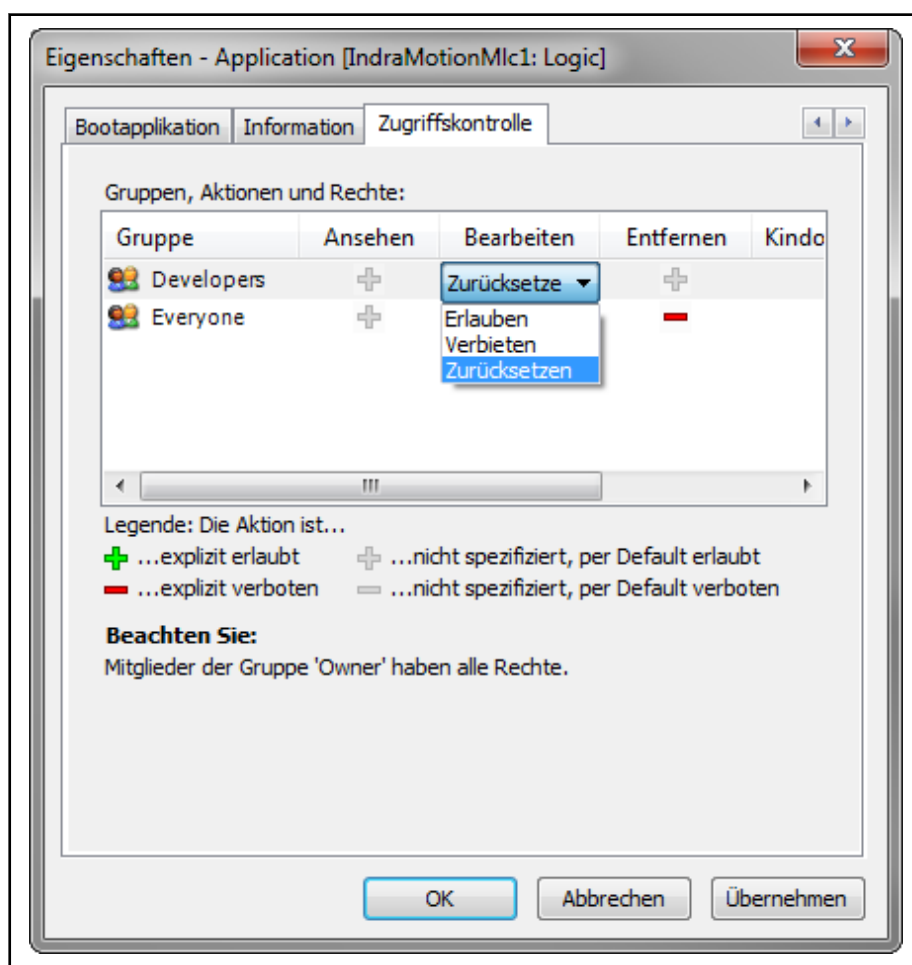


Abb. 18-37: Dialog Zugriffskontrolle in Eigenschaften einer Applikation

Zunächst sollten die Gruppenrechte für die gewünschte Zugriffskontrolle vergeben werden:

- Erlauben / Grant
- Verbieten / Deny
- Zurücksetzen / Clear

Da standardmäßig über die Gruppe "Everyone" jeder Benutzer Vollzugriff auf ein Objekt besitzt, müssen für einen wirksamen Objektschutz anschließend die Rechte für die Gruppe "Everyone" auf "Verbieten/ Deny" gesetzt werden!

18.2 Menübefehle

Standardmäßig sind die wichtigsten Befehle in der Benutzeroberfläche von IndraLogic verfügbar.

Wenn Sie die Menükonfiguration individuell anpassen möchten, wählen Sie den Befehl **Extras ► Anpassen ► Befehle**.

18.2.1 Menü 'Bearbeiten'

Standardbefehle

IndraLogic stellt Ihnen die folgenden Standardbefehle zur Verfügung:

- Rückgängig: , Tastaturkürzel: <Strg> + <Z>
- Wiederherstellen: , Tastaturkürzel: <Strg> + <Y>
- Ausschneiden: , Tastaturkürzel: <Strg> + <X>
- Kopieren: , Tastaturkürzel: <Strg> + <C>
- Einfügen: , Tastaturkürzel: <Strg> + <V>
- Löschen: , Tastaturkürzel: <Entf>
- Alles selektieren: Tastaturkürzel: <Strg> + <A>

Der Befehl "Einfügen" wird nicht von allen Editoren unterstützt oder kann in manchen nur eingeschränkt verwendet werden. In grafischen Editoren wird der Befehl nur unterstützt, wenn durch das Einfügen ein korrektes Konstrukt entsteht.

Im Projektextplorer bezieht sich der Befehl auf das gerade ausgewählte Objekt. Mehrfachauswahl ist möglich.

Suchen und Ersetzen



Die Befehlsgruppe "Suchen und Ersetzen" ist nur dann verfügbar, wenn der IndraLogic Editor geöffnet ist. Beispielsweise zum Bearbeiten von Programmorganisationseinheiten (POU's), oder Funktionsbausteinen.

Befehl 'Suchen'

Symbol: 

Funktion: Die Befehle durchsuchen das Projekt oder Teile davon nach einer bestimmten Zeichenkette.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Suchen und Ersetzen**, alternativ über das Tastaturkürzel [Strg] + [F].

Der Befehl öffnet den Dialog "Suchen", in dem die zu suchende Zeichenkette eingegeben und die Suchoptionen definiert werden kann.

Suchen nach	Zeichenkette, nach der gesucht wird.
Groß-/Kleinschreibung:	 : Die Suche beachtet Groß-/Kleinschreibung.
Nur ganzes Wort:	 : Nur Zeichenketten, die genau die gesuchte Zeichenkette wiedergeben, werden gefunden.
Nach oben suchen:	 : Der angegebene Suchraum wird nach oben durchlaufen.  : Der angegebene Suchraum wird nach unten durchlaufen
Reguläre Ausdrücke verwenden:	Über die Schaltfläche  erhalten Sie Unterstützung bei der Eingabe von regulären Ausdrücken (Regular Expressions)
Suchen in:	 : Auswahlliste mit den Bereichen des Projekts, die durchsucht werden sollen: • Aktiver Editor“ • Alle geöffneten Editoren • Ausgewählte Objekte & Subobjekte • Gesamtes Projekt • Gesamtes Projekt & nicht kompilierte Bibliotheken • Nur Selektion  : Öffnet einen Dialog, in dem Sie einstellen, in welchen Bereichen des Projekts gesucht werden soll (siehe unten)
Weitersuchen	Starten der Suche
Alle Suchen	Alle Suchergebnisse erscheinen im Meldungsfenster mit Angabe von Objektpfad, Projektname, Objektname und Objektposition. Mögliche Zusätze bei der Position: • (Dekl): Deklarationsteil des Objekts • (Impl): Implementierungsteil des Objekts Mit einem Doppelklick auf den Listeneintrag springt der Fokus zur Trefferposition im jeweiligen Objekteditor.
Ersetzen	Wechselt zum Dialog "Ersetzen"



Die Farbe der Markierung von Suchergebnissen kann in den Optionen des Texteditors angepasst werden. Dies erfolgt über den Parameter "Selektionsfarbe" - "Inaktiv" in der Registerkarte "Textbereich".

Siehe auch "[Registerkarte 'Textbereich'](#)" auf Seite 778

Dialog für die Einstellung der zu durchsuchenden Objekte

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: • Schema: Durch den Befehl Speichern wird die augenblickliche Suchkonfiguration unter dem angegebenen Namen gespeichert. Alle gespeicherten Schemata sind über die Auswahlliste () verfügbar. • Objekttypen: : Objekt wird durchsucht • Namensfilter: Namensfilter für die zu durchsuchenden Objekte. Der Platzhalter "*" kann verwendet werden. Beispiel: Filter "*CAN*": Es werden alle Objekte durchsucht, die in ihrem Namen "CAN" enthalten haben
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.
Nur Selektion	Nur der gerade in einem Objekt selektierte Text wird durchsucht.

Siehe auch

- [Kap. "Befehl 'Ersetzen'" auf Seite 612](#)
- [Kap. 9.13 "Projektweites Suchen und Ersetzen" auf Seite 138](#)

Befehl 'Ersetzen'

Symbol: 

Funktion: Die Befehle durchsuchen das Projekt oder Teile davon nach einer bestimmten Zeichenkette und ersetzen diese.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Suchen und Ersetzen**, alternativ über Tastaturkürzel [Strg] + [H].

Voraussetzung: Die Applikation ist im Online-Betrieb.

Der Befehl öffnet den Dialog "Ersetzen", in dem die zu ersetzende Zeichenkette und die neue Zeichenkette eingegeben und die Suchoptionen definiert werden.

Ersetzen mit	Eingabefeld für die neue Zeichenkette.
Ersetzen	Damit wird jeweils die nächste gefundene Zeichenkette im Editor hervorgehoben und ersetzt (schrittweises Ersetzen).
Alles Ersetzen	Alle gefundenen Zeichenketten werden auf einmal ersetzt, ohne dass sie in den Editoren angezeigt werden.
Geänderte Objekte offen lassen nach "Alles Ersetzen"	Die Editoren der gefundenen Objekte bleiben geöffnet

*Tab. 18-10: Zusätzlich zu den Optionen des Dialogs **Suchen** sind noch folgende Einstellungen möglich:*



Ein Ersetzen in referenzierten Bibliotheken ist nicht möglich.

Siehe auch

- [Kap. "Befehl 'Suchen'" auf Seite 610](#)
- [Kap. 9.13 "Projektweites Suchen und Ersetzen" auf Seite 138](#)

Befehl 'Abwärts Suchen'

Symbol: 

Funktion: Dieser Befehl selektiert bei der Suche nach einer bestimmten Zeichenkette im Projekt das nächste Suchergebnis an dessen Position im betreffenden Editor.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Suchen und Ersetzen**, alternativ über das Tastaturkürzel [F3].

Voraussetzung: Sie haben mit Hilfe der Befehle **Suchen** oder **Ersetzen** eine Suche nach einer bestimmten Zeichenkette im Projekt gestartet.

Siehe auch

- [Kap. "Befehl 'Suchen'" auf Seite 610](#)
- [Kap. "Befehl 'Ersetzen'" auf Seite 612](#)
- [Kap. 9.13 "Projektweites Suchen und Ersetzen" auf Seite 138](#)

Befehl 'Aufwärts suchen'

Symbol: 

Funktion: Dieser Befehl selektiert bei der Suche nach einer bestimmten Zeichenkette im Projekt das vorherige Suchergebnis an dessen Position im betreffenden Editor.

Aufruf: Indraworks Menü **Bearbeiten** ► **Suchen und Ersetzen**, alternativ Tastaturkürzel [Umsch] + [F3].

Voraussetzung: Sie haben mit Hilfe der Befehle "Suchen" oder "Ersetzen" eine Suche nach einer bestimmten Zeichenkette im Projekt gestartet.

Siehe auch

- [Kap. "Befehl 'Ersetzen'" auf Seite 612](#)
- [Kap. 9.13 "Projektweites Suchen und Ersetzen" auf Seite 138](#)

Befehl 'Abwärts suchen (Auswahl)'

Funktion: Der Befehl sucht nach der nächsten Zeichenkette im Projekt, die mit der übereinstimmt, die gerade selektiert ist, oder in der gerade der Cursor steht.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Suchen und Ersetzen**, alternativ über das Tastaturkürzel [Strg] + [F3].

Voraussetzung: Sie haben den Cursor in einer editierbaren Zeichenkette im Projekt platziert, oder Sie haben eine editierbare Zeichenkette selektiert.

Siehe auch

- [Kap. "Befehl 'Suchen'" auf Seite 610](#)
- [Kap. "Befehl 'Ersetzen'" auf Seite 612](#)
- [Kap. 9.13 "Projektweites Suchen und Ersetzen" auf Seite 138](#)

Befehl 'Aufwärts suchen (Auswahl)'

Funktion: Der Befehl sucht nach der vorherigen Zeichenkette im Projekt, die mit der übereinstimmt, die gerade selektiert ist, oder in der gerade der Cursor steht.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Suchen und Ersetzen**, alternativ über das Tastaturkürzel [Strg] + [Umsch] + [F3].

Voraussetzung: Sie haben den Cursor in einer editierbaren Zeichenkette im Projekt platziert, oder Sie haben eine editierbare Zeichenkette selektiert.

Siehe auch

- [Kap. "Befehl 'Suchen'" auf Seite 610](#)
- [Kap. "Befehl 'Ersetzen'" auf Seite 612](#)
- [Kap. 9.13 "Projektweites Suchen und Ersetzen" auf Seite 138](#)

Lesezeichen



Die Befehlsgruppe "Lesezeichen" ist nur dann verfügbar, wenn der IndraLogic Editor geöffnet ist. Beispielsweise zum Bearbeiten von Programmorganisationseinheiten (POU's), oder Funktionsbausteinen.

Befehl 'Lesezeichen umschalten'

Symbol:

Funktion: Der Befehl setzt oder löscht ein Lesezeichen an der aktuellen Position.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Lesezeichen**, alternativ über das Tastaturkürzel [Strg] + [F12].

Voraussetzung: Es ist eine POU im Editor geöffnet und der Cursor steht in einer Programmzeile.

Siehe auch

- [Kap. 9.12.3 "Lesezeichen setzen und verwenden" auf Seite 137](#)

Befehl 'Nächstes Lesezeichen'

Symbol:

Funktion: Der Befehl springt zum nächsten Lesezeichen.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Lesezeichen**, alternativ über das Tastaturkürzel [F12].

Voraussetzung: Ein POU ist im Editor geöffnet und der Cursor steht in einer Programmzeile.

Siehe auch

- [Kap. 9.12.3 "Lesezeichen setzen und verwenden" auf Seite 137](#)

Befehl 'Vorheriges Lesezeichen'

Symbol:

Funktion: Der Befehl springt zum vorherigen Lesezeichen.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Lesezeichen**, oder über das Tastaturkürzel [Umschalttaste] + [F12].

Voraussetzung: Es ist eine POU im Editor geöffnet und der Cursor steht in einer Programmzeile.

Siehe auch

- [Kap. 9.12.3 "Lesezeichen setzen und verwenden" auf Seite 137](#)

Befehl 'Lesezeichen löschen'

Symbol:

Funktion: Der Befehl löscht alle gesetzten Lesezeichen.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Lesezeichen**

Voraussetzung: Es ist eine POU im Editor geöffnet und der Cursor steht in einer Programmzeile.

Siehe auch

- [Kap. 9.12.3 "Lesezeichen setzen und verwenden" auf Seite 137](#)

Datei als Text einfügen



Der Befehl "Datei als Text einfügen" ist nur dann verfügbar, wenn der IndraLogic Editor geöffnet ist. Beispielsweise zum Bearbeiten von Programmorganisationseinheiten (POU's), oder Funktionsbausteinen.

Befehl 'Datei als Text einfügen'

Funktion: Der Befehl kopiert den Inhalt einer Textdatei in den aktiven Editor an die aktuelle Cursorposition.

Aufruf: IndraWorks Menü **Bearbeiten**

Voraussetzung: Die Datei muss die Dateiendung .txt haben. Der Befehl ist nur in einem Texteditor verfügbar.

Viele Entwicklungsumgebungen und Textverarbeitungsprogramme bieten die Möglichkeit, Code und Text als Textdatei zu exportieren. Der Inhalt dieser Datei kann mit diesem Befehl in den Editor kopiert werden.

Siehe auch

- [Kap. "Dialog 'Optionen- 'Texteditor'" auf Seite 776](#)

Durchsuchen

Befehl 'Gehe zur Definition'

Symbol:

Funktion: Der Befehl zeigt die Definitionsstelle einer Variablen oder Funktion.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Durchsuchen**, im Implementationsteil des Editors: Kontextmenü **Durchsuchen**

Voraussetzung: Eine POU ist im Editor geöffnet und der Cursor steht auf einer Variablen oder Funktion.

Siehe auch

- [Kap. 9.12.2 "Deklaration finden - 'Gehe zu Definition'" auf Seite 136](#)

Befehl 'Gehe zu Referenz'

Symbol:

Funktion: Der Befehl öffnet im Onlinebetrieb die Deklarationsstelle der Variablen, die von dem gerade fokussierten Pointer referenziert wird.

Aufruf:

- Über das Kontextmenü im Deklarationsteil oder Implementierungscode
- Menü **Bearbeiten** ► **Symbol suchen**

Voraussetzung: Onlinebetrieb. Ein Baustein ist im Editor geöffnet und der Cursor steht auf einem Pointer. Die referenzierte Variable liegt in einem statischen Speicher.



Wenn der Pointer nicht exakt auf den Beginn der Variablen zeigt, wird beim Wechseln zur Variablendeklaration eine entsprechende Meldung ausgegeben.

Siehe auch

- [Kap. 17.5.12 "Datentyp 'POINTER'" auf Seite 383](#)

Befehl 'Gehe zu Instanz'

Symbol: 

Funktion: Der Befehl öffnet die Instanz eines Funktionsbausteins in einem neuen Fenster

Aufruf: Menü **Bearbeiten** ► **Symbol suchen**

Voraussetzung: Die Applikation ist im Onlinebetrieb. Eine POU ist im Editor geöffnet und der Cursor steht auf der Instanz eines Funktionsbausteins.



Der Befehl ist nicht verfügbar für temporäre Instanzen oder Instanzen aus übersetzten Bibliotheken.

Siehe auch

- [Kap. 9.12.2 "Deklaration finden - 'Gehe zu Definition'" auf Seite 136](#)

Befehl 'Querverweise ausgeben'

Symbol: 

Funktion: Der Befehl öffnet die Querverweisliste, die alle Verwendungsstellen einer Variablen anzeigt.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Durchsuchen**. Alternativ über das Kontextmenü im Implementationsteil des Editors **Durchsuchen**.

Voraussetzung: Eine POU ist im Editor geöffnet und der Cursor steht in einer Variablen. Oder die Ansicht "Querverweisliste" ist geöffnet und eine Variable ist im Feld "Name" angegeben.

Siehe auch [Kap. 9.12.1 "Verwendungsstellen mit der Querverweisliste finden" auf Seite 135](#)

Befehl 'Aufrufbaum ausgeben'

Symbol: 

Funktion: Der Befehl öffnet die Ansicht "Aufrufbaum", die die Aufrufe des Bausteins sowie seine Aufrufer darstellt.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Durchsuchen**. Alternativ über das Kontextmenü des Editors im Implementationsteil **Durchsuchen**

Voraussetzung: Ein Baustein ist im Editor geöffnet und der Cursor steht in einer Variablen, oder ein Baustein ist in der Geräteknoten "Application", oder "Allgemeiner Bausteinteil" selektiert.

Siehe [Kap. "Befehl 'Aufrufbaum'" auf Seite 631](#)

Refactoring



Die Befehlsgruppe "Refactoring" ist nur dann verfügbar, wenn im Projektextplorer eine Programmorganisationseinheit (POU) selektiert ist. In diesem Fall ist nur der Befehl "Refactoring Umbenennen" verfügbar.

Alle weiteren Refactoring Befehle werden verfügbar, wenn der IndraLogic Editor geöffnet ist. Beispielsweise zum Bearbeiten von Programmorganisationseinheiten, oder Funktionsbausteinen.

Befehl 'Refactoring Umbenennen'

Symbol:

Funktion: Der Befehl öffnet einen Dialog zum projektweiten Umbenennen eines Objekts oder einer Variablen.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Refactoring**, alternativ im Kontextmenü des Editors im Implementationsteil **Refactoring**.

Voraussetzung: Ein Objekt im Projektextplorer unterhalb von Applikation oder im Ordner "Allgemeiner Bausteinteil" ist selektiert, oder im Deklarationsteil eines Programmierobjekts ist der Cursor vor oder auf einem Variablenbezeichner positioniert.

Sie können umbenennen:

- Variablen
- Programm-Organisationseinheiten (POU)
- Globale Variablenlisten (GVL)
- Methoden
- Eigenschaften
- Geräte
- Variablen und Einheitenumrechnungen innerhalb des Editors für Einheitenumrechnungen

Dialog 'Umbenennen'

Aktueller Name	Name des Objekts oder der Variablen
Neuer Name	Eingabefeld für einen neuen Namen. Wenn der eingegebene Name bereits existiert, meldet IndraLogic dies direkt unter diesem Eingabefeld.
OK	Aktivierbar, wenn Sie in Neuer Name einen gültigen Namen eingegeben haben. Öffnet den Dialog Refactoring In beiden Fenstern sind die jeweiligen Objekte und Stellen farblich markiert. In beiden Fenstern können Sie für jede Verwendungsstelle festlegen, was gemacht werden soll. Dazu stehen Ihnen im Kontextmenü folgende Befehle zur Verfügung

Dialog 'Refactoring'

Der Dialog zeigt alle Verwendungsstellen innerhalb des Projekts. In beiden Fenstern sind die jeweiligen Objekte und Stellen farblich markiert.	
Rechtes Fenster	Anzeige der jeweiligen Stelle innerhalb eines Objekts, wo Aktueller Name vorkommt.
Linkes Fenster	Navigationsbaum des Projekts mit dem jeweiligen Objekt.

Sie können in beiden Fenstern für jede Verwendungsstelle festlegen, was gemacht werden soll. Dazu stehen Ihnen im Kontextmenü folgende Befehle zur Verfügung:

Diese Änderung verwerfen	Verwerfen der einzelnen Änderung im rechten Fenster
Dieses Objekt annehmen	Annehmen aller Änderungen im betroffenen Objekt
Dieses Objekt verwerfen	Verwerfen aller Änderungen im betroffenen Objekt
Das gesamte Projekt annehmen	Annehmen aller Änderungen im Projekt
Das gesamte Projekt verwerfen	Verwerfen aller Änderungen im Projekt

IndraLogic stellt angenommene Änderungen mit gelbem Hintergrund, verworfene Änderung mit grauem Hintergrund dar.

Siehe auch

- [Kap. 9.14 "Refactoring" auf Seite 139](#)

Befehl 'Variable hinzufügen'

Symbol: 

Funktion: Der Befehl ermöglicht das Deklarieren einer neuen Variablen in einer POU und die entsprechende automatische Aktualisierung an den Verwendungsstellen der POU.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Refactoring**, alternativ über das Kontextmenü des Editors im Deklarationsteil.

Voraussetzungen: Der Fokus liegt im Deklarationsteil einer POU

Der Befehl öffnet den Standarddialog zum Deklarieren der Variablen.

Siehe auch

- ["Dialog 'Variable deklarieren'" auf Seite 626](#)

Dialog Refactoring

Nach Schließen des Deklarationsdialogs mit **OK** erscheint der zweigeteilte Dialog **Refactoring**.

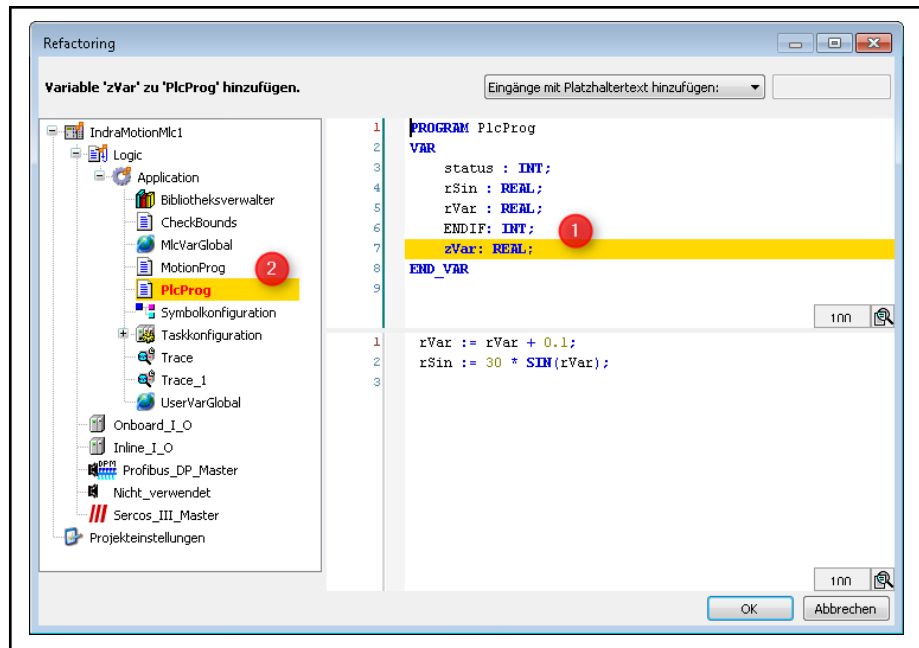


Abb. 18-38: Dialog *Refactoring*, zweigeteilt

Rechter Dialogteil	Deklarationsteil und Implementierung der POU, in deren Deklaration die Variable hinzugefügt wird Farbliche Kennzeichnung der Änderungsstellen: Neu hinzugefügte Deklaration in blauer Schrift und gelb hinterlegt (1).
Linker Dialogteil	Projektextplorer oder im Ordner "Allgemeiner Bausteinteil" des Projekts.. Farbliche Kennzeichnung der Bausteine, in denen die POU verwendet wird: Rote Schrift und gelb hinterlegt (2). Nach Doppelklick auf das POU-Objekt öffnet sich die Detailansicht.

Bevor Sie entscheiden welche Änderungen Sie an welchen Stellen übernehmen wollen, wählen Sie die gewünschte Option aus der Auswahlliste (3) rechts oben:

Eingänge mit Platzhaltertext hinzufügen	Standardplatzhaltertext: <code>_REFACTOR_</code> ; editierbar Der hier definierte Platzhaltertext erscheint an den Verwendungsstellen der neu hinzugefügten Variablen im Implementierungscode. Er dient der Suche nach den betroffenen Stellen.
Eingänge mit folgendem Wert hinzufügen	Initialisierungswert für die neue Variable

Im Kontextmenü der Änderungsstellen, sowohl im linken als auch rechten Teil des Dialogs gibt es Befehle zum Annehmen oder Ablehnen der Änderung(en).

Beispiele

1. Baustein `fun` erhält über Refactoring eine neue Eingangsvariable `input3` mit Initialisierungswert 1. Die Änderung wirkt sich folgendermaßen aus:

Vorher:

```
fun(a + b, 3, TRUE);
fun(input1:= a + b , input2 :=3 , inputx := TRUE);
```

Nachher:

```
fun(a + b, 3, 1, TRUE);
fun(input1:= a + b , input2 :=3 , _REFACTOR_ , inputx := TRUE);
```

2. Baustein `fun` erhält über Refactoring eine neue Eingangsvariable `input3` mit Platzhaltertext `"_REFACTOR_"`:

Vorher:

```
inst(input1 := a + b, input2 := 3, inputx := TRUE);
fun(a + b, 3, TRUE);
```

Nachher:

```
inst(input1 := a + b, input2 := 3, input3 := _REFACTOR_ , inputx := TRUE);
fun(a + b, 3, _REFACTOR_ , TRUE);
```

Siehe auch

- [Kap. 9.14 "Refactoring" auf Seite 139](#)
- [Kap. "Befehl 'Refactoring Umbenennen'" auf Seite 617](#)

Befehl 'Variable entfernen...'

Symbol: 

Funktion: Der Befehl entfernt eine Eingangs- oder Ausgangsvariable aus der POU und allen Verwendungsstellen der POU.

Aufruf: Indraworks Menü **Bearbeiten** ► **Refactoring**, alternativ über das Kontextmenü des editors im Deklarationsteil.

Voraussetzungen: Der Cursor steht im Bezeichner der zu entfernenden Variablen im Deklarationsteil der POU.

Der Befehl öffnet zunächst einen Dialog mit den Angaben zur gewünschten Entfernung. Nach dessen Bestätigung erscheint der Dialog **Refactoring**.

Siehe für eine Beschreibung des Dialogs **Refactoring** die Hilfeseite zu Befehl **Bearbeiten** ► **Refactoring** ► **Umbenennen**.

Wenn Sie die Änderungen in Dialog **Refactoring** akzeptieren, werden an den Verwendungsstellen der betroffenen POU die entsprechenden Eingangs- oder Ausgangsparameter gelöscht.



Im CFC wird nur die Verbindung des entfernten Eingangs oder Ausgangs zum Baustein entfernt. Der Eingang oder Ausgang selbst bleibt im Chart enthalten.

Beispiel in ST

Sie entfernen in einer POU über "Refactoring" die Eingangsvariable `input4`. An den Verwendungsstellen erfolgt eine automatische Anpassung:

Vor der Entfernung:

```
inst(input1 := a + b, input2 := 3, input4 := 1, input5 := TRUE);
fun(a + b, 3, 1, TRUE);
```

Nach der Entfernung:

```
inst(input1 := a + b, input2 := 3, input5 := TRUE);
fun(a + b, 3, TRUE);
```

Siehe auch

- [Kap. 9.14 "Refactoring" auf Seite 139](#)
- [Kap. "Befehl 'Refactoring Umbenennen'" auf Seite 617](#)

Befehl 'Variablen neu ordnen...'

Symbol: 

Funktion: Der Befehl ermöglicht im Deklarationseditor das Ändern der Reihenfolge der Variablen im gerade fokussierten Gültigkeitsbereich VAR_INPUT, VAR_OUTPUT oder VAR_IN_OUT.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Refactoring**, Kontextmenü des gerade fokussierten Gültigkeitsbereichs im Deklarationseditor.

Voraussetzung: Der Fokus steht in der Deklaration eines der oben genannten Gültigkeitsbereiche und es ist mehr als 1 Variable darin deklariert.

Der Befehl öffnet den Dialog "Neu ordnen" mit einer Liste aller Deklarationen des gerade fokussierten Gültigkeitsbereichs. Durch Ziehen mit der Maus können Sie eine selektierte Deklaration nach oben oder unten an eine andere Position ziehen.

Siehe auch

- ["Variablen in der Deklaration neu anordnen" auf Seite 141](#)

Befehl 'Betreffende Anschlüsse aktualisieren'



Der Befehl wirkt momentan nur in den Editoren für CFC und FUP/KOP/AWL. Er entspricht den Befehlen **Anschlüsse zurücksetzen** und **Parameter aktualisieren**.

Symbol: 

Funktion: Der Befehl passt in allen betroffenen Verwendungsstellen des Bausteins die Anschlüsse an die aktuelle Bausteindeklaration an.

Aufruf: Menü **Bearbeiten** ► **Refactoring**, alternativ über das Kontextmenü des Deklarationseditors.

Voraussetzung: Der Cursor steht im Namen des Bausteins in der ersten Zeile der Bausteindeklaration oder im Projektextplorer.

Siehe auch

- [Kap. 9.14 "Refactoring" auf Seite 139](#)
- [Kap. "Befehl 'Anschlüsse zurücksetzen'" auf Seite 697](#)
- [Kap. "Befehl 'Parameter aktualisieren'" auf Seite 712](#)

Erweitert



Die Befehlsgruppe "Erweitert" ist nur dann verfügbar, wenn der IndraLogic Editor geöffnet ist. Beispielsweise zum Bearbeiten von Programmorganisationseinheiten, oder Funktionsbausteinen.

Befehl 'Leerzeichen anzeigen'

Symbol: 

Funktion: Der Befehl bewirkt, dass Steuerzeichen für Leerzeichen und Tabulatoren angezeigt werden.

Aufruf: IndraWorks Menü **Bearbeiten** ► **Erweitert**

Voraussetzung: Ein Texteditor ist selektiert.

IndraLogic visualisiert Leerzeichen durch einen Punkt und Tabulatoren durch einen Pfeil.

Siehe auch

- [Kap. "Dialog 'Optionen- 'Texteditor'" auf Seite 776](#)

Befehl 'Einrückungshilfslinien anzeigen'

Funktion: Der Befehl aktiviert die Einrückungshilfslinien.

Aufruf: Menü **Bearbeiten** ► **Erweitert**

Voraussetzung: Ein Texteditor ist geöffnet.

Wenn die Einrückungshilfslinien aktiviert sind, wird für jede manuelle Einrückung im Code eine gestrichelte Linie eingefügt. Dies erleichtert den Überblick über die verschiedenen Ebenen im Code. Manuelle Einrückungen können Sie mit der Taste <Tabulator> einfügen.

Siehe auch

- [Kap. "Dialog 'Optionen- 'Texteditor'" auf Seite 776](#)

Befehl 'Alles reduzieren'

Funktion: Der Befehl reduziert alle eingerückten Code-Abschnitte (Minus-Symbol am Wurzelknoten) im Texteditor oder Fundstellen in der Querverweisliste. Dadurch werden nur noch die oberste Ebene des Codes, beziehungsweise nur die Wurzelknoten der Fundstellen angezeigt.

Voraussetzung: Ein Texteditor ist aktiv und zugleich ist im Dialog "Optionen", Kategorie "Texteditor" die Einrückung aktiviert. Alternativ ist die Querverweisliste aktiv.

Aufruf: Generell über das IndraWorks Menü **Bearbeiten ► Erweitert**, alternativ über das Kontextmenü im Editor **Erweitert**. In der Querverweisliste ebenfalls über das Kontextmenü.

Siehe auch

- [Kap. "Dialog 'Optionen- 'Texteditor'" auf Seite 776](#)
- [Kap. "Befehl 'Alles erweitern'" auf Seite 622](#)
- ["Befehle im Kontextmenü der Querverweisliste" auf Seite 630](#)

Befehl 'Alles erweitern'

Funktion: Der Befehl erweitert alle reduzierten Code-Abschnitte (Plus-Symbol) im Texteditor, oder Fundstellen in der Querverweisliste wieder auf, so dass der Code beziehungsweise alle Fundstellen wieder komplett angezeigt wird.

Voraussetzung: Ein Texteditor ist aktiv und zugleich ist im Dialog "Optionen", Kategorie "Texteditor" die Einrückung aktiviert. Alternativ ist die Querverweisliste aktiv.

Aufruf: Generell über das IndraWorks Menü **Bearbeiten ► Erweitert**, alternativ über das Kontextmenü im Editor **Erweitert**. In der Querverweisliste ebenfalls über das Kontextmenü.

Siehe auch

- [Kap. "Dialog 'Optionen- 'Texteditor'" auf Seite 776](#)
- [Kap. "Befehl 'Alles reduzieren'" auf Seite 621](#)
- ["Befehle im Kontextmenü der Querverweisliste" auf Seite 630](#)

Befehl 'Selektierte Zeilen auskommentieren'

Symbol: 

Tastaturkürzel: [Ctrl] + [O]



Der Befehl ist nicht in der Standardeinstellung der IndraWorks Oberfläche enthalten.

Sie können den Befehl bei Bedarf im folgenden Menü ergänzen:
Bearbeiten ► Erweitert...

Um den Befehl zu ergänzen, öffnen Sie das Menü **Extras ► Anpassen....**

Im angezeigten Dialog wechseln Sie in die Registerkarte "Befehle".

Selektieren Sie in der linken Spalte die Kategorie "Befehle".

Scrollen Sie in der Befehlsliste zum gewünschten Befehl und ziehen Sie ihn mittels Drag&Drop an die gewünschte Stelle im IndraWorks Menü.

Voraussetzung: Der Cursor steht im ST-Editor in einer Zeile der Implementierung oder mehrere Zeilen im ST-Editor sind selektiert.

Aufruf: Menü **Bearbeiten ► Erweitert...**

Funktion: Der Befehl fügt ein Kommentarzeichen (//) am Anfang der selektierten Zeilen ein.

Siehe auch

- [Kap. "Befehl 'Selektierte Zeilen Auskommentierung entfernen'" auf Seite 623](#)

Befehl 'Selektierte Zeilen Auskommentierung entfernen'

Symbol: 

Tastaturkürzel: [Ctrl] + [I]



Der Befehl ist nicht in der Standardeinstellung der IndraWorks Oberfläche enthalten.

Sie können den Befehl bei Bedarf im folgenden Menü ergänzen:
Bearbeiten ► Erweitert...

Um den Befehl zu ergänzen, öffnen Sie das Menü **Extras ► Anpassen....**

Im angezeigten Dialog wechseln Sie in die Registerkarte "Befehle".

Selektieren Sie in der linken Spalte die Kategorie "Befehle".

Scrollen Sie in der Befehlsliste zum gewünschten Befehl und ziehen Sie ihn mittels Drag&Drop an die gewünschte Stelle im IndraWorks Menü.

Voraussetzung: Der Cursor steht im ST-Editor in einer Zeile der Implementierung oder mehrere Zeilen im ST-Editor sind selektiert.

Aufruf: Menü **Bearbeiten ► Erweitert...**

Funktion: Der Befehl entfernt, falls vorhanden, einen Zeilenkommentar (//) am Anfang der selektierten Zeilen.

Siehe auch

- [Kap. "Befehl 'Selektierte Zeilen auskommentieren'" auf Seite 622](#)

Befehl 'Gehe zu'

Tastaturkürzel: [Ctrl] + [G]

Funktion: Mit dem Befehl springt der Cursor zu einer definierten Zeile im Code.

Aufruf: IndraWorks Menü **Bearbeiten ► Erweitert**

Voraussetzung: Ein Texteditor ist selektiert.

Der Befehl öffnet einen Dialog mit einem Eingabefeld für die gewünschte Zeilennummer.

Siehe auch

- [Kap. "Dialog 'Optionen- 'Texteditor'" auf Seite 776](#)

Befehl 'In Großbuchstaben umwandeln'

Funktion: Der Befehl wandelt alle Kleinbuchstaben im selektierten Code in Großbuchstaben um.

Aufruf: Menü **Bearbeiten ► Erweitert**, alternativ über das Tastaturkürzel [Strg] + [Shift] + [U].

Voraussetzung: Ein Texteditor ist geöffnet und Code ist selektiert, oder der Deklarationseditor ist geöffnet und Variablendeklarationen sind selektiert.

Siehe auch

- [Kap. "Dialog 'Optionen- 'Texteditor'" auf Seite 776](#)

Befehl 'In Kleinbuchstaben umwandeln'

Funktion: Der Befehl wandelt alle Großbuchstaben im selektierten Code in Kleinbuchstaben um.

Aufruf: Menü **Bearbeiten** ► **Erweitert**, alternativ über das Tastaturkürzel [Strg] + [U].

Voraussetzung: Ein Texteditor ist geöffnet und Code ist selektiert, oder der Deklarationseditor ist geöffnet und Variablendeklarationen sind selektiert.

Siehe auch

- [Kap. "Dialog 'Optionen- 'Texteditor'" auf Seite 776](#)

Befehl 'Korrespondierende Klammer'

Funktion: Der Befehl lässt den Cursor zum zugehörigen Teil der selektierten Codeklammer springen.

Aufruf: Menü **Bearbeiten** ► **Erweitert**

Voraussetzungen: Ein Texteditor ist geöffnet und eine öffnenden oder schließenden Codeklammer ist markiert.

Im Optionen-Dialog ist die Option "Zusammengehörige Klammern" in der Kategorie "Texteditor", Registerkarte "Textbereich" aktiviert.

Siehe auch

- ["Registerkarte 'Textbereich'" auf Seite 778](#)

Befehl 'Bis korrespondierende Klammer markieren'

Funktion: Der Befehl markiert den gesamten Code-Abschnitt zwischen den beiden korrespondierenden Klammern.

Aufruf: Menü **Bearbeiten** ► **Erweitert**

Voraussetzungen: Ein Texteditor ist geöffnet und der Cursor ist bei einer öffnenden oder schließenden Code-Klammer positioniert.

Im Optionen-Dialog ist die Option "Zusammengehörige Klammern" in der Kategorie "Texteditor", Registerkarte "Textbereich" aktiviert.

Siehe auch

- [Kap. "Dialog 'Optionen- 'Texteditor'" auf Seite 776](#)

Eingabehilfe

Der Befehl "Eingabehilfe" ist nur dann verfügbar, wenn der IndraLogic Editor geöffnet ist. Beispielsweise zum Bearbeiten von Programmorganisationseinheiten (POU's), oder Funktionsbausteinen.

Befehl 'Eingabehilfe'

Symbol:

Funktion: Der Befehl öffnet den Dialog "Eingabehilfe", der Sie bei der Auswahl eines an der aktuellen Cursorposition möglichen Programmierelements unterstützt.

Aufruf: IndraWorks Menü **Bearbeiten**, alternativ über das Tastaturkürzel [F2], oder über das Kontextmenü des Editors **Eingabehilfe**.

Voraussetzung: Eine Programmorganisationseinheit (POU) ist im Editor geöffnet und der Cursor steht in einer Programmzeile.

Siehe auch

- [Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112](#)

Dialog 'Eingabehilfe' - Registerkarte 'Kategorien'

Der Dialog bietet Ihnen alle Programmierelemente an, die Sie im Editor an der aktuellen Cursorposition einfügen können.

Die Elemente sind nach "Kategorien" sortiert. Sie können für die Kategorie "Variablen" zusätzlich einen "Filter" für den Variablentyp setzen, wie "Lokale Variablen", "Globale Variablen", "Konstanten" etc.

"Strukturierte Ansicht"	 : Der Dialog stellt die Elemente in einem Strukturbaum dar. Sie können die Spalten "Typ", "Adresse" und "Herkunft" mit einem Rechtsklick in den Spaltentitel in einem Untermenü aus- oder einblenden.  : Die Elemente werden in einer flachen Struktur dargestellt.
"Dokumentation anzeigen"	 : IndraLogic erweitert den Dialog um das Feld "Dokumentation".
"Mit Argumenten einfügen"	 : IndraLogic fügt Elemente, die Argumente besitzen, wie zum Beispiel Funktionen, mit diesen Argumenten an der Cursorposition ein. Beispiel: Wenn Sie den Funktionsbaustein <code>fb1</code> , der eine Eingabevariable <code>fb1_in</code> und eine Ausgabevariable <code>fb1_out</code> enthält, "mit Argumenten" einfügen, sieht dies im Editor folgendermaßen aus: <code>fb1(fb1_in:= , fb1_out=>)</code>
"Mit Namensraum-Präfix einfügen"	 : IndraLogic fügt das ausgewählte Element mit vorangestelltem Namensraum ein. Im Fall von Bibliotheksbausteinen können Sie die Checkbox nicht bedienen, wenn in den Eigenschaften der Bibliothek festgelegt ist, dass die Angabe des Namensraums zwingend ist.



Wenn Sie Objekte mit dem gleichen Namen und der gleichen Kategorie sowohl global (im "Allgemeinen Bausteinteil") als auch einer Applikation zugeordnet anlegen, erscheint nur ein Eintrag in der Eingabehilfe, da die Verwendung der üblichen Aufrufpriorität (applikationszugeordnet vor global) folgt.

Siehe auch

- ["Dialog 'Eigenschaften'" auf Seite 716](#)
- [Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112](#)

Dialog 'Eingabehilfe' - Registerkarte 'Textsuche'

In der Registerkarte können Sie nach bestimmten Objekten suchen. Sobald Sie ein oder mehrere Zeichen in das Suchfeld eingeben, listet das Trefferfenster die Namen aller Objekte auf, deren Name diese Suchzeichenfolge enthält. Mit einem Doppelklick auf das gewünschte Objekt fügen Sie es an der aktuellen Cursorposition im Editor ein.

Filter	Einschränken der Suche auf eine bestimmte Variablenkategorie.

Variable deklarieren

Der Befehl "Variable deklarieren" ist nur dann verfügbar, wenn der IndraLogic Editor geöffnet ist. Beispielsweise zum Bearbeiten von Programmorganisationseinheiten (POU's), oder Funktionsbausteinen.

Befehl 'Variable deklarieren'

Funktion: Der Befehl öffnet den Dialog "Variable deklarieren", der die Deklaration einer Variablen unterstützt.

Aufruf: IndraWorks Menü **Bearbeiten**, alternativ über das Tastaturkürzel [Shift] + [F2].

Voraussetzung: Ein Objekt oder ein Gerät des Projekts ist im Editor geöffnet.

Durch die Autodeklarationsfunktion erscheint der Dialog "Variable deklarieren" auch, wenn der Cursor im Implementierungsteil einer POU in einer Zeile steht, die den Namen einer nicht deklarierten Variablen enthält. Als Voraussetzung hierfür müssen Sie den Befehl **Extras ► Optionen ► IndraLogic 2G** ausgewählt und in der Kategorie "Intelligentes Kodieren" die Option "Unbekannte Variablen automatisch deklarieren (AutoDeclare)" aktiviert haben.

Dialog 'Variable deklarieren'

Gültigkeitsbereich	Gültigkeitsbereich der noch nicht deklarierten Variable. Beispiel: VAR (Standardeinstellung bei lokaler Variable)
Name	Noch nicht deklarierter Variablenname. Beispiel: bIsValid
Datentyp	Beispiel: BOOL  Listet die Standarddatentypen auf.  Eingabehilfe: Öffnet den Dialog "Eingabehilfe" Array-Assistent: Öffnet den Dialog "Array"
Objekt	Objekt, in dem die neue Variable deklariert wird. Standardmäßig das Objekt, das Sie gerade bearbeiten. Beispiel: fbA  : Listet die Objekte auf, in denen die Variable deklariert werden kann. Wenn für den ausgewählten Gültigkeitsbereich keine Objekte verfügbar sind, erscheint der Eintrag <Objekt anlegen>. Wenn Sie den Eintrag <Objekt anlegen> auswählen, öffnet sich der Dialog Objekt hinzufügen zur Erzeugung eines geeigneten Objekts.
Initialisierungswert	Beispiel: FALSE Wenn Sie keinen Initialisierungswert eingeben, wird die Variable automatisch initialisiert.  : Öffnet den Dialog Initialisierungswert. Diese Vorgehensweise ist für die Initialisierung von strukturierten Variablen hilfreich.
Adresse	Speicheradresse der Applikation für die noch nicht deklarierte Variable. Beispiel: %IX1.0 Hinweis: Nur möglich bei folgenden Gültigkeitsbereichen: - Lokale Variable (VAR) - Globale Variable (VAR_GLOBAL) - Persistente Variable (PERSISTENT)

Flags	Attribut-Schlüsselwörter • <code>CONSTANT</code>: Schlüsselwort für eine Konstante • <code>RETAIN</code>: Schlüsselwort für eine remanente Variable • <code>PERSISTENT</code>: Schlüsselwort für eine persistente Variable (strenger als <code>RETAIN</code>). Das ausgewählte Attribut-Schlüsselwort wird der Variablendeklaration hinzugefügt.
Kommentar	Beispiel: <code>New input In1</code> Im tabellarischen Deklarationseditor erfolgt die Anzeige des eingegebenen Kommentars in der Spalte Kommentar, im textuellen Deklarationseditor oberhalb der Variablendeklaration.
Änderungen mit Hilfe von Refactoring anwenden	 : Wenn Sie den Dialog beenden, wird die Variable noch nicht deklariert, sondern es öffnet sich zunächst der Dialog Refactoring. Dort können Sie Ihre Änderungen weiter bearbeiten. Die Option erscheint bei folgenden Gültigkeitsbereichen: • Eingabevariable (<code>VAR_INPUT</code>) • Ausgabevariable (<code>VAR_OUTPUT</code>) • <code>VAR_IN_OUT</code>-Variablen (Eingabevariable und Ausgabevariable)
OK	Variable wird deklariert und erscheint in der Deklaration. Beispiel: <pre>VAR RETAIN // New input In1 xIn1 AT %IX1.0: BOOL := FALSE; END_VAR</pre>

Dialog 'Array'

Dimensions- und Basistyp-Angabe	Definition der Feldgrößen ("Dimension") durch Eingeben der unteren und oberen Grenzen und des "Basistyps" des Arrays. Den Basistypen können Sie direkt eingeben oder mit Hilfe der Dialoge "Eingabehilfe" oder "Array", wenn Sie die Schaltfläche  anklicken.
Ergebnis	Anzeige des definierten Arrays.



IndraLogic initialisiert nur Variablen neu, wenn Sie die Initialisierungswerte der Variablen geändert haben.

Dialog 'Initialisierungswert'

Auflistung der Variablen mit Namen ("Ausdruck"), "Initialisierungswert" und "Datentyp". Geänderte Initialisierungswerte werden fett dargestellt.	
Eingabefeld unterhalb der Liste	Eingabe eines Initialisierungswerts für die selektierte Variable(n).
Wert auf ausgewählte Zeilen anwenden	Änderung des Initialisierungswerts der selektierten Zeile(n) entsprechend dem Wert des Eingabefelds.
Ausgewählte Zeilen auf Standardwerte zurücksetzen	Herstellung der Standard-Initialisierungswerte.
OK	IndraLogic übernimmt die Initialisierungswerte in den Dialog "Variable deklarieren".

Siehe auch

- [Kap. 9.2.2 "Dialog 'Variable deklarierenverwenden'" auf Seite 88](#)
- [Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112](#)
- [Kap. 9.10.2 "AT-Deklaration" auf Seite 130](#)
- [Kap. 9.16 "Datenpersistenz" auf Seite 144](#)

18.2.2 Menü 'Ansicht'

Projektexplorer

Symbol: 

Funktion: Bei aktivierter Einstellung  wird der IndraWorks Projektexplorer in der Programmieroberfläche angezeigt.

Aufruf: IndraWorks Menü **Ansicht**

Bibliothek

Symbol: 

Funktion: Bei aktivierter Einstellung  wird der IndraWorks Bibliotheksexplorer in der Programmieroberfläche angezeigt.

Aufruf: IndraWorks Menü **Ansicht**

Andere Fenster

Befehl 'Werkzeuge'

Symbol: 

Funktion: Der Befehl öffnet die Ansicht **Werkzeuge**.

Aufruf: Menü **Ansicht** ► **Andere Fenster**

Mit dem Befehl öffnen Sie die Werkzeuge-Ansicht für das gerade geöffnete Objekt. Diese Ansicht steht Ihnen standardmäßig bei einem grafischen Editor oder einer Visualisierung zur Verfügung. Sie enthält die grafischen Programmelemente, die Sie in den Editor ziehen können.

Siehe auch

- [Kap. 17.1.2 "Generelle Funktionalitäten in allen grafischen Editoren" auf Seite 240](#)

Befehl 'Querverweisliste'

Symbol: 

Funktion: Der Befehl öffnet die Ansicht "Querverweisliste".

Aufruf: Menü **Ansicht** ► **Andere Fenster**

Die Ansicht zeigt eine Liste der Querverweise im Projekt für ein Symbol. Das Symbol kann eine Variable, eine POU (Programm, Funktionsbaustein, Funktion) oder ein anwenderspezifischer Datentyp (DUT) sein. Dabei bietet die Querverweisliste grundsätzlich zwei Sucharten:

- **Textsuche:** Durch die Eingabe eines Symbolnamens werden die Querverweise aller Symbole im Projekt mit diesem Namen angezeigt. Falls mehrere Symbole mit gleichem Namen gefunden werden, kann die Anzeige über das Kontextmenü auf einzelne Deklarationen eingeschränkt werden.

- **Deklarationssuche:** Das Symbol kann über die Eingabehilfe, oder durch die Eingabe eines qualifizierten Pfads (beispielsweise `__POOL.POU.a`) ausgewählt werden. Danach werden nur die Verwendungsstellen dieses Symbols angezeigt, auch wenn noch andere Symbole mit dem gleichen Namen existieren.

Eingabefeld	Symbolname (Variablenname, Bausteinname, DUT-Name). Eingabemöglichkeiten: • Auswahl eines deklarierten Symbols über die Eingabehilfe, Schaltfläche  • Händisches Eingeben des Symbolnamens. Auslösen der Suche über die Schaltfläche , oder die [Eingabetaste]. Sie können für die Textsuche die Platzhalter "*" (beliebig viele Zeichen) oder "?" (genau ein beliebiges Zeichen) in Kombination mit einer Teilzeichenkette eines Variablenbezeichners verwenden. Verwenden Sie "%" wenn Sie nach IEC-Adressen suchen wollen. Beispiele: "%MW8", "%M*". • Automatisch, wenn der Name eines deklarierten Symbols in einem Editor selektiert ist oder der Cursor im Namen steht. Eine automatische Suche ist auch möglich, wenn das Objekt im Gerätebaum oder POU-Pool selektiert ist. Voraussetzung: Die Option Querverweise automatisch bei Selektionsänderung aktualisieren muss angewählt sein. Aufruf über das IndraWorks Menü Extras ▶ Optionen ▶ IndraLogic 2G ▶ Intelligentes Kodieren. <i>Folgende Eingaben sind gültig:</i> • Variablenname, einfach oder vollqualifiziert. Beispiele: "iVar", "PlcProg.iVar". • Bausteinname: Beispiele: "PlcProg", "myFB". • DUT-Name: Beispiel: "mySTRUCT". • Zeichenfolgen in Kombination mit Platzhalter "*" (beliebige Zeichen), oder "?" (genau ein beliebiges Zeichen). Beispiel: "iVar*" betrifft iVar1, iVar_glob2, iVar45 usw. "iVar?" betrifft iVar1, iVar2, iVarX usw., nicht aber iVar_glob2, iVar45 usw. • "%<IEC-Adresse>": IndraLogic sucht nach Variablen, die dieser Adresse zugewiesen sind, und direkte Speicherzugriffe. Beispiel: "%QB0", "%Q0 := 2".
	Textsuche durchführen <F4>: Die Suche wird durchgeführt.
	Deklaration aus der Eingabehilfe entnehmen: Der Dialog "Eingabehilfe" öffnet sich. Hier können Sie eine Variable, einen Datentyp oder einen Funktionsbaustein auswählen.
	Quellposition des vorherigen Querverweises anzeigen <Umsch>+<F4>
	Quellposition des nächsten Querverweises anzeigen <F4>
	Quellposition des selektierten Querverweises anzeigen: Der Fokus springt an die Stelle, an der die Variable deklariert ist. Dazu öffnet sich der entsprechende Deklarationseditor und die Variable wird dort selektiert.
	Ergebnisse auf aktuelle Deklaration beschränken: Verfügbar, wenn für ein Symbol mehrere Deklarationen gefunden wurden. Begrenzt die Anzeige auf die Deklaration, die Sie gerade in der Liste selektiert haben.

	Querverweisliste drucken: Der Standarddialog zum Einrichten eines Druckauftrags erscheint.
Die gefundenen Querverweise werden mit folgenden Informationen dargestellt:	
"Symbol"	Die Fundstellen für die Symbole (Variablen, POU, DUTs) werden nach ihrer Deklaration gruppiert. Die Deklarationsstelle bildet den Wurzelknoten, darunter eingerückt erscheinen die Verwendungsstellen im Projekt. Dabei wird genau der Ausdruck angezeigt, den das Symbol an der Verwendungsstelle hat. Beispiel: Gibt es im Projekt eine globale Variable <code>i</code> und in einem Baustein eine lokal deklarierte Variable <code>i</code> , dann erscheinen nach einer Textsuche nach den Querverweisen zwei Wurzelknoten-Einträge in der Liste und darunter jeweils die Verwendungsstellen der Variable <code>i</code> .
"Baustein"	Bausteinname, DUT-Name; beispielsweise auch Taskname im Falle eines Bausteinaufrufs in der Taskkonfiguration.
"Variable"	Reiner Variablenname, beispielsweise <code>iVar</code> .
"Zugriff"	Art des Zugriffs auf die Variable an der Verwendungsstelle: "Deklaration"/ "Lesen"/ "Schreiben"/ "Aufruf". Sonderfall für Pointer: Eine Zuweisung der Art <code>p := ADR(var1)</code> wird bei Suche nach <code>var1</code> als <code>Schreiben Adresse</code> angezeigt. Grund: Eventuelle Schreibzugriffe auf <code>p</code> werden bei Suche nach <code>var1</code> nicht angezeigt. Über die Pointervariable sind auch Schreibzugriffe möglich.
"Typ"	Datentyp der Variablen.
"Adresse"	IEC-Adresse, falls der Variablen zugewiesen. Beispiel: "AT %QB0".
"Position"	Position der Verwendungsstelle innerhalb des Editors der betroffenen POU: beispielsweise Zeilennummer, Netzwerknummer, Deklarationsteil oder Implementierungsteil. Beispiel: "Zeile 1, Spalte 1 (Impl)".
"Objekt"	POU-Name + in eckigen Klammern der komplette Pfad der Verwendungsstelle, falls sich diese nicht im "Allgemeinen Bausteinteil" befindet. Beispiel: "PlcProg [Device: Logic: Application]"
"Kommentar"	Kommentar, falls in der Deklaration der Variable vorhanden.

Befehle im Kontextmenü der Querverweisliste

Die Suche liefert alle Fundstellen im Projekt sowie in eingehängten, nicht kompilierten Bibliotheken.

Quellposition anzeigen: Öffnet den betroffenen Baustein und markiert die Verwendungsstelle: Für Root-Einträge die Deklaration, für darunter liegende Kindeinträge die jeweilige Verwendungsstelle. Alternativ können Sie auf eine Zeile doppelklicken.

Ergebnisse auf aktuelle Deklaration beschränken: Grenzt im Fall von mehreren gefundenen Deklarationen die Anzeige der Ergebnisse auf die ausgewählte Symboldeklaration ein.

Alles erweitern: In der Liste erscheinen alle einzelnen Fundstellen.

Alles reduzieren: In der Liste erscheinen nur die Wurzelknoten aller Fundstellen.

Siehe auch

- [Kap. 9.12.1 "Verwendungsstellen mit der Querverweisliste finden" auf Seite 135](#)
- [Kap. "Befehl 'Ergebnisse auf aktuelle Deklaration beschränken'" auf Seite 743](#)
- [Kap. "Befehl 'Alles reduzieren'" auf Seite 621](#)
- [Kap. "Befehl 'Alles erweitern'" auf Seite 622](#)

Befehl 'Aufrufbaum'

Symbol: 

Funktion: Der Befehl öffnet die Ansicht "Aufrufbaum".

Aufruf: Kontextmenü eines aufrufbaren Bausteins im Geräteknoten "Application", oder im Ordner "Allgemeiner Bausteinteil"

Ansicht 'Aufrufbaum'

Der Aufrufbaum steht jederzeit bereits vor dem Kompilieren (Übersetzen) der Applikation zur Verfügung. Er ist eine statische Darstellung der Aufrufer und der Aufrufe des Bausteins, den Sie explizit angeben. Somit enthält der Baum immer 2 Wurzelknoten, unter denen die jeweilige Aufrufabfolge als nacheinander eingerückte Einträge zu sehen ist. Rekursive Aufrufe sind in dieser Baumdarstellung schnell erkennbar.

Beispiel für einen Aufrufbaum (1) für Baustein (2) PlcProg:

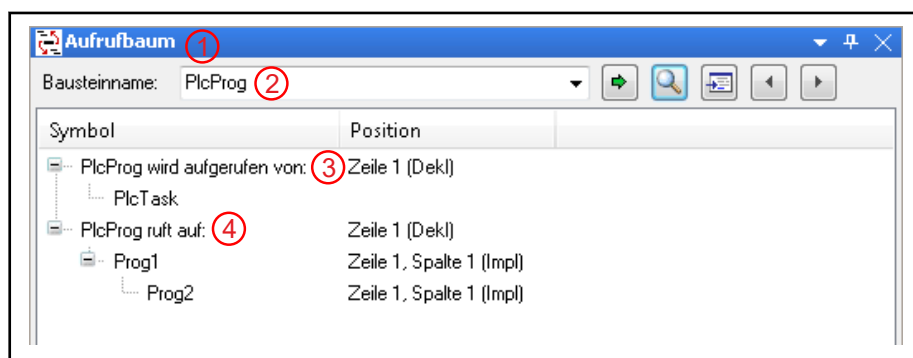


Abb. 18-39: Dialog "Aufrufbaum"

- (3) Knoten "<Bausteinname> wird aufgerufen von:"
- (4) Knoten "<Bausteinname> ruft auf"

Bausteinname	Name des Programmbausteins, kann händisch, oder durch Ziehen aus einer anderen Ansicht, oder mit Hilfe der Schaltfläche  eingegeben werden. Die Auswahlliste enthält die zuletzt eingegebenen Bausteinnamen.
Symbolleiste und Tastaturbedienung:	
 Baustein finden	IndraLogic sucht nach dem in "Bausteinname" angegebenen Baustein und stellt seine Aufrufer und seine Aufrufe dar.
 Baustein aus Eingabehilfe entnehmen	Der Dialog "Eingabehilfe" erscheint zur Auswahl eines Bausteinaufrufs, oder Instanzaufrufs. Der Aufrufbaum wird nach der Auswahl automatisch aktualisiert.
 Quellcodeposition des selektierten Bausteins anzeigen	IndraLogic springt zur Verwendungsstelle des Bausteins im Quellcode Ihres Programms.
[F4] Quellcodeposition des nächsten Bausteins anzeigen [Umsch] + [F4] Quellcodeposition des vorherigen Bausteins anzeigen	Die Selektion im Aufrufbaum springt zum nächsten, oder vorherigen Baustein in der Aufrufstruktur. Gleichzeitig wird die dazugehörige Quellcodeposition im jeweiligen Editor geöffnet. Hinweis: Ein Doppelklick auf einen Eintrag im Aufrufbaum öffnet ebenfalls die zugehörige Quellcodeposition.

Darstellung des Aufrufbaums:	
Symbol	"<Bausteinname> wird aufgerufen von": Unter diesem Knoten erscheint die Aufrufabfolge für den Baustein. Der unterste Eintrag in dieser Baumstruktur zeigt den Start der Aufrufe. "<Bausteinname> ruft auf": Unter diesem Knoten erscheinen die von dem Baustein ausgehenden Aufrufe. Der unterste Eintrag in dieser Baumstruktur zeigt das Ende der Aufrufkette.
Position	Für die Wurzelknoten im Aufrufbaum: Zeilennummer der Deklaration ("Dekl ") des Bausteins. Für die Aufrufer oder Aufrufe unter den Wurzelknoten: Je nach Implementierungssprache Zeilennummer, Spaltennummer, Netzwerknummer ihrer Position.
Kontextmenü für den gerade im Baum selektierten Eintrag:	
Alles reduzieren	Die ausgeklappten Einträge im Aufrufbaum werden bis auf die beiden Wurzelknoten zugeklappt.
Quellcodeposition anzeigen	IndraLogic springt zur Verwendungsstelle des Bausteins im Quellcode Ihres Programms.
Als neuen Wurzelknoten setzen	Der im Aufrufbaum selektierte Eintrag erscheint in "Bausteinname". Der Baum wird automatisch für die neuen Wurzelknoten angepasst.



Im Unterschied zum statischen Aufrufbaum, der jederzeit Aufrufinformationen zu einem Baustein liefert, ist die Ansicht "Aufrufliste" für unmittelbare Information während der schrittweisen Abarbeitung eines Programms vorgesehen. Die Aufrufliste zeigt immer den kompletten Aufrufpfad der gerade erreichten Position.

Siehe:

- [Kap. 12.3 "Schrittweises Abarbeiten eines Programms \(Stepping\)" auf Seite 196](#)
- [Kap. "Befehl 'Aufrufliste'" auf Seite 667](#)

Befehl 'Querverweise in der klassischen Ansicht ausgeben'

Symbol: 

Funktion: Der Befehl öffnet die Ansicht "Klassische Querverweisliste" .

Aufruf: Menü **Ansicht** ► **Andere Fenster** nach Übernahme aus **Extras** ► **Anpassen** Befehlskategorie "Quelltextnavigation"

Die Ansicht entspricht der Ansicht „Querverweisliste“ vor IndraLogic 2G, 3.5 SP6.

Die Suche liefert alle Fundstellen im Projekt sowie in eingehängten, nicht kompilierten Bibliotheken.

Befehl 'Elementeigenschaften'

Symbol: 

Funktion: Der Befehl öffnet die Ansicht "Elementeigenschaften".

Aufruf: Menü **Ansicht** ► **Andere Fenster**

Mit dem Befehl öffnen Sie die Eigenschaftenansicht für das gerade geöffnete Objekt. Diese Ansicht ist nur für wenige Objekte mit Inhalt gefüllt, wie beispielsweise Visualisierung und POU (AS).

Die Eigenschaften werden in einer strukturierten Tabelle angezeigt. Sie ändern die Werte der Eigenschaften durch einen Mausklick in das Wertefeld. Sie können die Ansicht der Eigenschaften filtern oder sortieren.

Befehl 'Speicher'

Symbol: 

Funktion: Der Befehl öffnet die Ansicht "Speicher".

Aufruf: Menü **Ansicht** ► **Andere Fenster** ► **Speicher**

Voraussetzung: Eine Applikation ist geladen und im Online-Betrieb.

Ansicht 'Speicheransicht'

"Applikation"	Auswahl der Applikation, für die die Speicheransicht dargestellt werden soll. Mit dieser Applikation müssen Sie auf der Steuerung eingeloggt sein. Es muss nicht die "aktive Applikation" sein.
"Bereich"	"Absolut": Speicher wird direkt und vollständig adressiert. Die Adresse steht im Eingabefeld "Adresse." "Area <i>": Speicherbereiche der Steuerung, beginnend mit Area 0. Speicherbereiche, die ausschließlich für Code reserviert sind, werden nicht angezeigt.
"Adresse"	Absolute Startadresse des Core Dumps Voraussetzung: Im Eingabefeld "Bereich" ist der Wert "Absolut" ausgewählt.
"Offset"	Adressversatz zum gewählten Speicherbereich in Byte, zum Beispiel 0x0200, 16#0200 oder als Dezimalzahl 512 Voraussetzung: In "Bereich" ist ein Speicherbereich ausgewählt, zum Beispiel "Area 0". IndraLogic bietet alle aktuell verwendeten Speicherbereiche zur Auswahl an. Speicherbereiche, die ausschließlich für Code reserviert sind, werden nicht angezeigt.
	Adresse für eine Variable herausfinden: Eingabeunterstützung zur Auswahl einer IEC-Variablen erscheint. Wenn Sie eine Variable ausgewählt haben, belegt IndraLogic die Startadresse mit der Variablenadresse vor.
	Speicheransicht laden/ aktualisieren
	Vorheriges Speichersegment anzeigen: Navigieren zum vorherigen Speichersegment
	Nächstes Speichersegment anzeigen: Navigieren zum nächsten Speichersegment
	Vorsicht: IndraLogic prüft nicht, ob die Änderungen zulässig sind. Sie können die Applikation durch unbedachte Änderungen zum Absturz bringen! Änderungen auf SPS laden: IndraLogic überträgt die neuen Daten auf die Steuerung. Voraussetzung: Sie haben ein oder mehrere Bytes in der Speicheransicht überschrieben.
	Speicherauszug in Datei speichern: Dialog "Speicherinhalt als Binärdatei speichern" erscheint. Wählen Sie einen Ablageort.
"Spalten"	Anzahl der Spalten der hexadezimale Darstellung des Speicherauszugs. Je Spalte werden zwei Bytes angezeigt. Bei "Auto" passt sich die Spaltenanzahl an die Fenstergröße an. Rechts im Fenster werden die Daten als Text angezeigt.

Befehl 'Ganzer Bildschirm'

Symbol: , Tastaturkürzel: <Strg> + <Alt> + <F>

Funktion: Der Befehl schaltet die Anzeige von IndraWorks in den Vollbild-Modus.

Aufruf: Menü **Ansicht**

Wenn Sie den Befehl aktivieren, wird das Hauptfenster der IndraWorks-Benutzeroberfläche im Vollbildmodus angezeigt. Sie können zur vorher eingestellten Größe zurückkehren, wenn Sie den Befehl wieder deaktivieren oder das Tastaturkürzel <Strg> + <Alt> + <F> betätigen.

Befehl 'Eigenschaften'

Symbol: 

Funktion: Der Befehl öffnet die Eigenschaften des gerade selektierten Objekts.

Aufruf: Menü **Ansicht**

Siehe auch

- [Kap. 18.3.6 "Dialog 'Eigenschaften'" auf Seite 748](#)

Befehl 'Startbild'

Symbol: 

Funktion: Der Befehl öffnet die Ansicht **Startbild**.

Aufruf: Menü **Ansicht**

Die Ansicht enthält einige Basisbefehle und eine Liste der zuletzt geöffneten Projekte.

Sie können das Startbild durch die Option "Beim Start nicht mehr anzeigen" bis auf Widerruf abschalten.

18.2.3 Menü 'Projekt'

Befehl 'Projekteinstellungen'

Symbol: 

Funktion: Der Befehl öffnet den Dialog "Projekteinstellungen".

Aufruf: Doppelklick auf das Objekt "Projekteinstellungen" im Ordner "Allgemeiner Baustein".

Voraussetzung: Ein Projekt ist geöffnet.

Siehe auch

- [Kap. 2.5.2 "Projekteinstellungen vornehmen" auf Seite 14](#)
- [Kap. 18.3.7 "Dialog 'Projekteinstellungen'" auf Seite 758](#)
- [Kap. 18.1.19 "Objekt 'Projekteinstellungen'" auf Seite 579](#)

Befehl 'Benutzerverwaltung'

In der IndraWorks-Version 13 wird die bisherige IndraWorks-Benutzerverwaltung durch eine neue Version ersetzt.

Durch die neue Benutzerverwaltung ergeben sich die folgende Vorteile:

- über die neue Benutzerverwaltung können geschützte SPS-Bibliotheken erstellt werden,

- das Debuggen in geschützten SPS-Bibliotheken wird unterstützt ,
- die Benutzerverwaltung ist integraler Bestandteil des IndraWorks-Projekts und wird mit diesem automatisch archiviert,
- die neue Benutzerverwaltung unterstützt Safety-Objekte

Beim Öffnen eines IndraWorks-Projektes < 13VRS werden diese und ein vorhandener Schutz erkannt und die folgende Meldung ausgegeben:

Meldung beim Öffnen eines IndraWorks-Projektes < 13VRS

Das Projekt liegt in einer alten Kompatibilitätsversion vor und muss in das aktuelle Format konvertiert werden.

Anschließend kann das Projekt nicht mehr in der alten Kompatibilitätsversion geöffnet werden!

Wollen Sie das Projekt jetzt konvertieren?

Meldung, wenn dieses Projekt durch die alte Benutzerverwaltung geschützt war

Im Projekt wurden Objekte gefunden, die mit der Benutzerverwaltung aus IndraWorks Version 12 geschützt werden.

Durch das Öffnen des Projektes wird dieser Schutz entfernt.

Wollen Sie den Vorgang fortsetzen?



Durch das Öffnen des Projektes werden alle bisherigen Benutzerverwaltungsinformationen entfernt.

Konfiguration der Benutzerverwaltung

Funktion: Benutzerverwaltung

Aufruf: **Projekt ► Benutzerverwaltung.** In diesem Menü stehen dem Anwender die folgenden Befehle zur Verfügung:

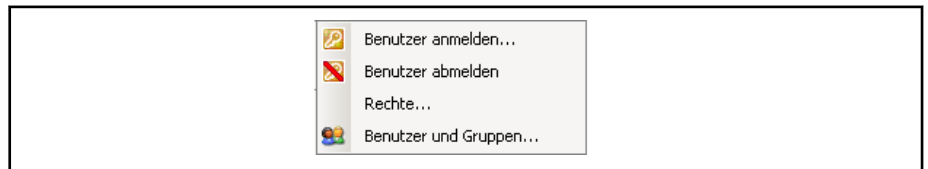


Abb. 18-40: Menü 'Benutzerverwaltung'

- [Kap. 18.1.28 "Objekt 'Zugriffskontrolle'" auf Seite 608](#)

Befehl 'Benutzer anmelden'

Symbol:

Der Befehl **Benutzer anmelden** öffnet den Dialog **Anmelden**, in dem 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.

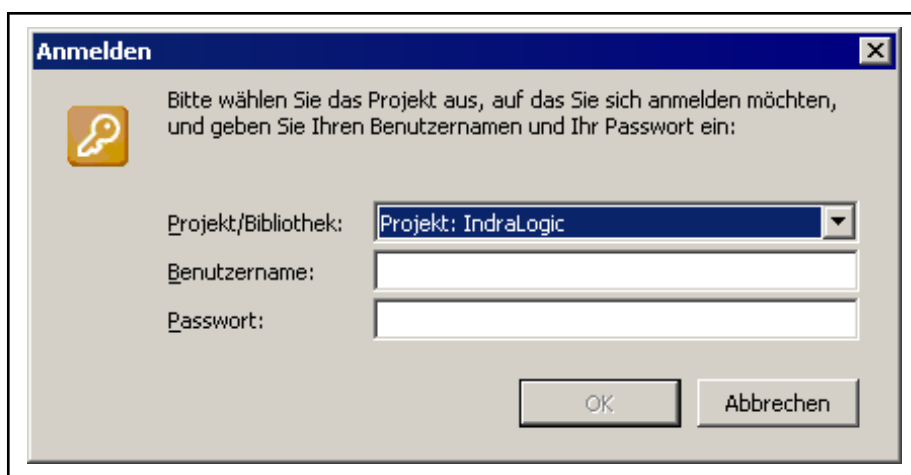


Abb. 18-41: Dialog Anmelden

Wenn bereits ein anderer Benutzer angemeldet ist, wird dieser durch die neue Anmeldung automatisch abgemeldet.

Anmelden-Dialog im Fall von nicht zulässigen Aktionen

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:

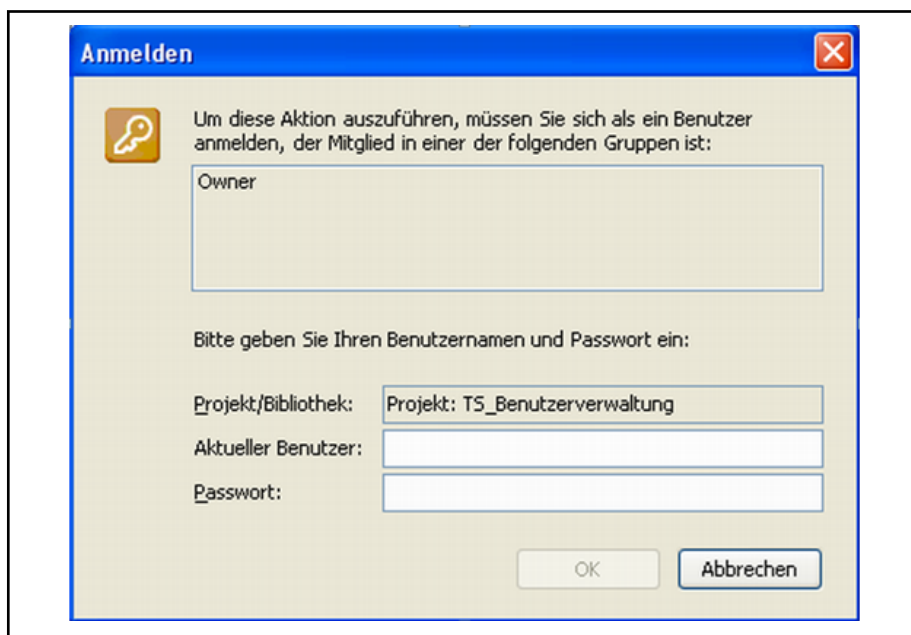


Abb. 18-42: Anmelden-Dialog im Fall einer nicht zulässigen Aktion

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.

Die Statusleiste zeigt immer den aktuell im Projekt angemeldeten Benutzer.



Abb. 18-43: Beispiel: Statusleiste

Befehl 'Benutzer abmelden'Symbol: 

Der Befehl **Benutzer abmelden** meldet den gerade im Projekt oder der Bibliothek angemeldeten Benutzer wieder ab. Nur wenn gerade gar kein Benutzer angemeldet 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:

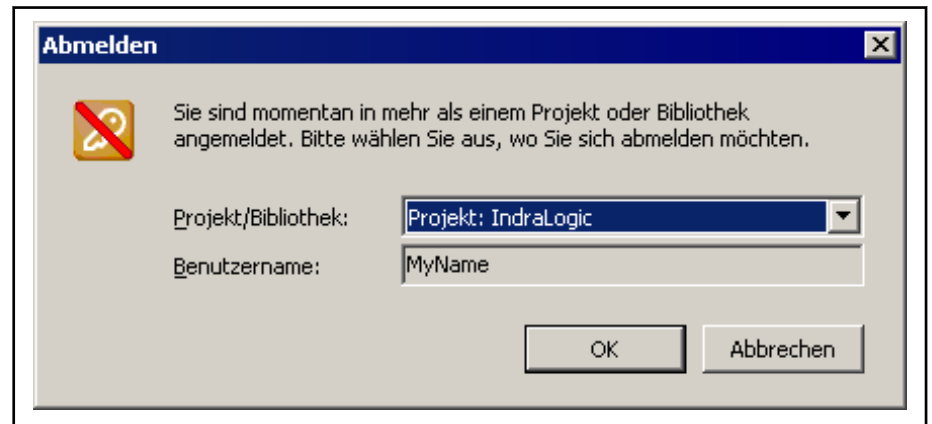


Abb. 18-44: Dialog Abmelden

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.

Befehl 'Benutzer und Gruppen'Symbol: 

Der Befehl **Benutzer und Gruppen** öffnet den Dialog **Benutzer und Gruppen**. Der Dialog besteht aus 3 Reitern Benutzer, Gruppen und Einstellungen:

Benutzer Hier werden die angelegten Benutzer und ihre Mitgliedschaft zu Gruppen angezeigt.

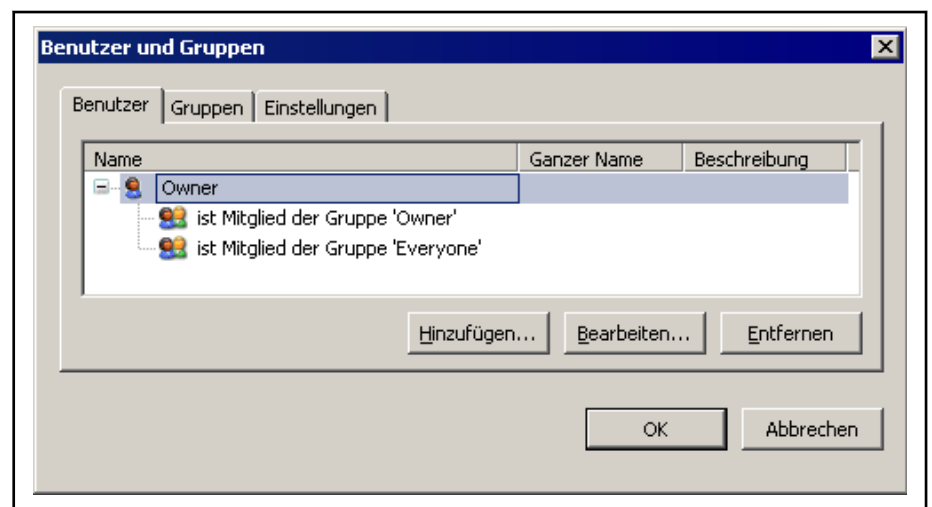


Abb. 18-45: Dialog Benutzer aus Benutzer und Gruppen

Gruppen Hier werden die angelegten Gruppen mit ihren zugehörigen Benutzern angezeigt.

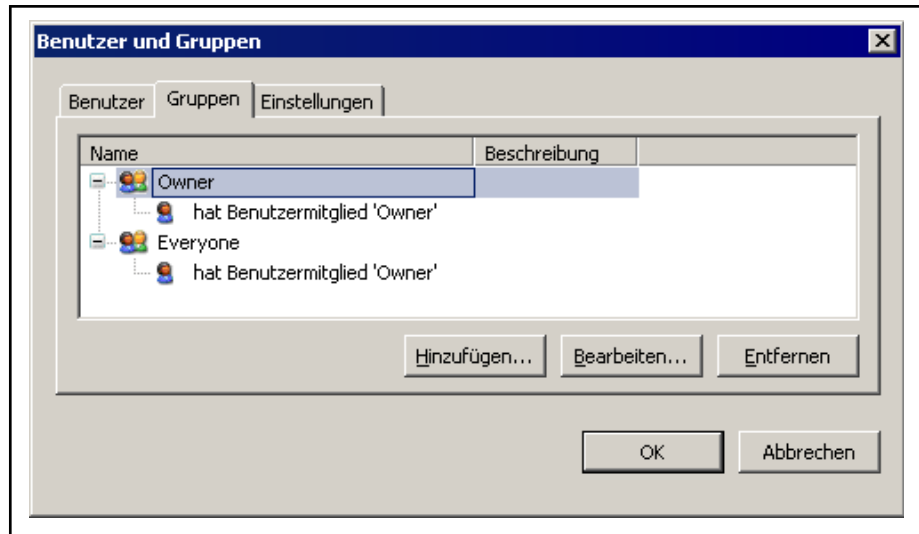


Abb. 18-46: Dialog Gruppen aus Benutzer und Gruppen

Einstellungen Hier können die Anzahl der erfolglosen Anmeldeversuche und die automatische Abmeldung nach Inaktivität konfiguriert werden.

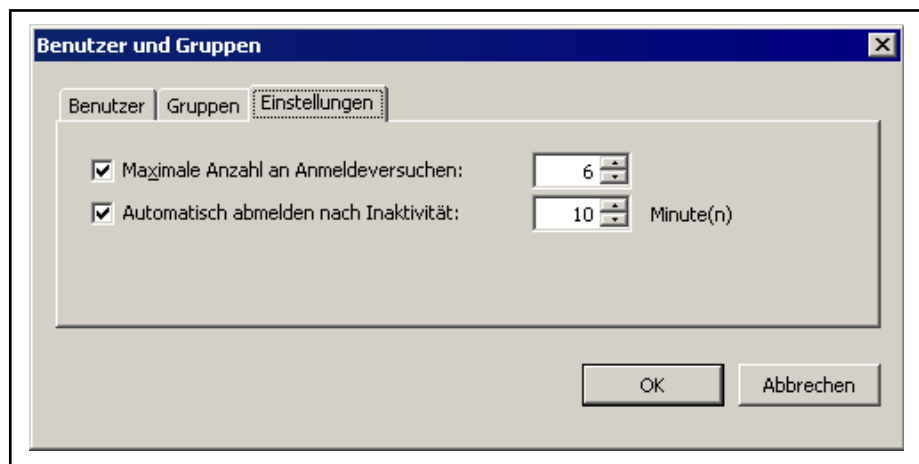


Abb. 18-47: Dialog Einstellungen aus Benutzer und Gruppen

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

Beim Anlegen von Benutzerkonten und Benutzergruppen ist folgendes zu beachten:

- 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. Die Gruppe **"Everyone"** kann umbenannt aber nicht gelöscht werden. Ferner 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 Mit-

glied 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 angemeldet.

Danach allerdings kann der Anwender sich über ein bestimmtes Benutzerkonto mit Benutzername und Passwort einloggen, um einen bestimmten Satz an Zugriffsrechten zu haben.

- **jedes Projekt hat seine eigene Benutzerverwaltung!** 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.

Befehl 'Rechte'

Der Befehl **Rechte** öffnet den Dialog **Rechte**, in dem die Zugriffsrechte auf Objekte bzw. die Rechte, bestimmte Aktionen im Projekt auszuführen, konfiguriert werden können.

Zunächst hat in einem neuen Projekt nur der "Owner" (Gruppe "Owner") das Recht, die Benutzer-, Gruppen- und Rechtekonfiguration zu ändern. Man kann ins Projekt mit Benutzername "Owner" und leerem Passwort einloggen, um dieses Recht an eine andere Gruppe zu vergeben.



Beachten Sie auch, dass alle Einstellungen im Rechte-Dialog sofort wirksam werden.

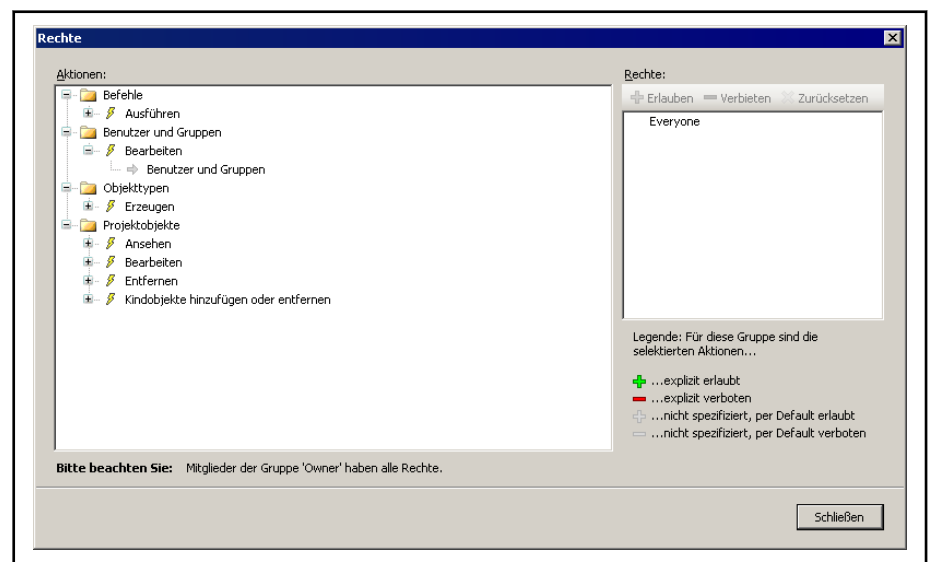


Abb. 18-48: Menü Rechte aus Benutzerverwaltung

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 **Befehlen**, das Bearbeiten von **Benutzer- und Grup-**

pen-Einstellungen, das Erzeugen von **Objekttypen** und das Ansehen, Bearbeiten, Entfernen und Handhaben von Kindobjekten von **Projektobjekten**. **Projektobjekte** beziehen sich immer auf die aktuell im Projekt angelegten Objekte. Diese Anzeige ändert sich also mit dem Hinzufügen und Löschen von Objekten.

-  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:

-  **Erlauben** Explizites Erlauben.
-  **Verbieten** Explizites Verbieten.
-  **Zurücksetzen** Das Recht wird auf den Standardwert zurückgesetzt.

Befehl 'SPS-Compilerversion'

Funktion: In IndraWorks-Versionen kleiner als 13V06 war die SPS-Compilerversion fest an die Kompatibilitätsversion von IndraWorks gebunden.

Ab der IndraWorks-Kompatibilitätsversion 13V06 kann die SPS-Compilerversion zusätzlich ausgewählt werden. Hierdurch wird es möglich den IndraWorks-Kompatibilitätsmodus anzupassen ohne das SPS-Kompilat zu verändern. Der Anwender kann sich so auch nach der Änderung des Kompatibilitätsmodus ohne erneuten Download auf das Gerät einloggen.

Aufruf: Menü **Projekt** ► **SPS-Compilerversion...**



Abb. 18-49: Dialogbox der SPS-Compilerversion

In der Dialogbox wird festgelegt, welche SPS-Compilerversion bei der Ausführung des Befehls 'Erstellen' oder während des Herunterladens zum Übersetzen des Projekts verwendet werden soll.

Hinter der SPS-Compilerversion steht jeweils in Klammern die zugehörige IndraWorks-Kompatibilitätsversion.

Die größtmöglich einzustellende SPS-Compilerversion entspricht dem aktuellen Kompatibilitätsmodus des Projekts.

Beim Neuanlegen eines Projekts wird die SPS-Compilerversion standardmäßig auf die zugehörige IndraWorks-Kompatibilitätsversion gesetzt.

Befehl 'SPS-Projektvergleich'

Funktion: Der Befehl "SPS-Projektvergleich" bietet die Möglichkeit das aktuell geladene Projekt mit einem zweiten Projekt zu vergleichen und im Bedarfsfall Bestandteile des zweiten Projekts in das erste zu mergen.

Aufruf: IndraWorks Menü **Projekt** ► **SPS-Projektvergleich**.

Wird der Befehl aktiviert, erscheint zunächst ein Auswahlfenster zur Vorgabe des zweiten zu vergleichenden Projektes.

Beide Projekte werden einander gegenüber gestellt.

Die Vergleichsstatistik gibt mögliche Unterschiede unter der Randbedingung der gewählten Vergleichsoptionen an, (**Extras ▶ Optionen ▶ Vergleich**, siehe auch IndraWorks, Optionen)

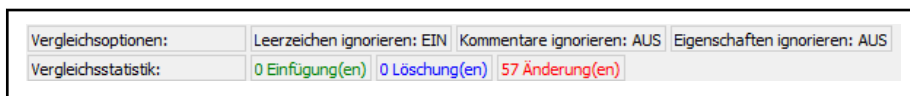
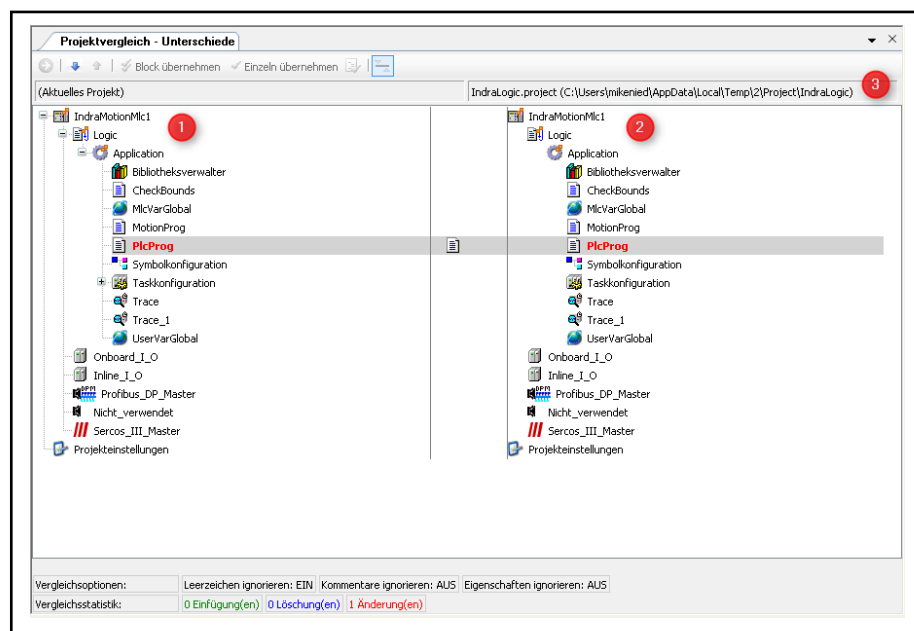


Abb. 18-50: Vergleichsstatistik

Unterscheiden sich die Projekte, kann der eigentliche Vergleich durch einen Mausklick auf eines der Objekte aktiviert werden.



- (1) das geladene aktuelle Projekt
- (2) das zweite Projekt zum Vergleich
- (3) Platz an dem sich das angewählte Objekt im Navigator befindet

Abb. 18-51:

Mit dem Anklicken eines der Objekte wird die Symbolleiste des Fensters aktiviert:

-   : erlauben den Wechsel zwischen der Ansicht "Projektunterschiede" und "innere Unterschiede des gewählten Objektpaars"
-   : gehe zum nächsten / vorherigen Unterschied
-   : erlauben die Auswahl, ob der komplette Block oder nur die markierte Zeile übernommen wird.

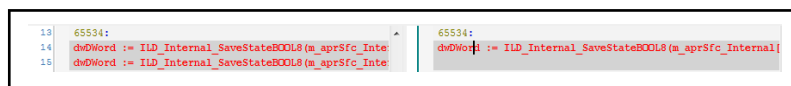


Abb. 18-52: Ausgangssituation

Zeile 14 links und rechts unterscheiden sich inhaltlich, Zeile 15 rechts wurde gelöscht.



Abb. 18-53: Block übernehmen

Zeile 14 rechts ist markiert, es werden jedoch alle Zeilen (14 und 15) des Blocks von rechts nach links zur Übernahme angeboten. Die eigentliche Übernahme erfolgt jedoch erst, wenn zusätzlich die Schaltfläche <Übernehmen> betätigt wird.



Abb. 18-54: Einzeln übernehmen

Zeile 14 rechts ist markiert, es wird nur diese Zeile des Blocks von rechts nach links zur Übernahme angeboten. Die eigentliche Übernahme erfolgt jedoch erst, wenn zusätzlich die Schaltfläche <Übernehmen> betätigt wird.

- : Ordner, Zugriffsrechte, Eigenschaften des Objektes rechts werden mit übernommen.
- : Bereiche unterschiedlich darstellen.

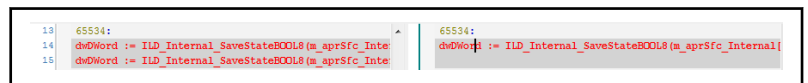


Abb. 18-55: Darstellung unterschiedliche Inhalte in Zeile 14/15



Abb. 18-56: Darstellung unterschiedliche Operationen, grün (14/15) eingefügt, blau (16) gelöscht

- : Vergleichs- und Übernahmestatus



Abb. 18-57: Inhalte der Objekte unterscheiden sich

- : Vergleichs- und Übernahmestatus

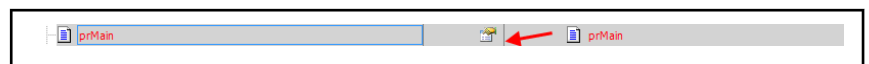


Abb. 18-58: Eigenschaften der Objekte unterscheiden sich

Befehl 'Hinzufügen'

Funktion: Der Befehl öffnet ein Untermenü mit Objekten, das abhängig von der aktuellen Position im Projektextplorer alle Objekte enthält, die eingefügt werden können.

Aufruf: Kontextmenü.

Voraussetzung: Wenn IndraLogic das Objekt im Projektextplorer einfügen soll, selektieren Sie ein bereits vorhandenes Objekt, unterhalb dessen das neue eingerückt angelegt soll.

Wenn IndraLogic das Objekt im Ordner "Allgemeiner Baustein" einfügen soll, setzen Sie den Fokus auf den Ordner.

Siehe auch

- [Kap. 2.2 "Objekte hinzufügen" auf Seite 8](#)
- [Kap. 2 "Projekt anlegen und konfigurieren" auf Seite 7](#)

Befehl 'Ordner hinzufügen'

Symbol: 

Funktion: Der Befehl öffnet einen Dialog zum Definieren eines neuen Ordners im Projektextplorer.

Aufruf: Kontextmenü



Die Anordnung der Geräteknoten und Geräteobjekte können Sie nicht durch selbst angelegte Ordner strukturieren.

Der Befehl fügt den Ordner unterhalb des gerade selektierten Objekts im Explorer ein. Wenn kein Objekt selektiert ist, fügt IndraLogic den Ordner ganz oben im Baum direkt unter dem Wurzelknoten ein.

Siehe auch

- [Kap. 2.2 "Objekte hinzufügen" auf Seite 8](#)

Befehl 'SPS-Objekte eines externen IndraWorks-Projektes anzeigen'

Symbol: 

Aufruf: Menü **Projekt ► SPS Objekte eines externen IndraLogic-Projektes anzeigen**

Funktion: Der Befehl erlaubt es, ein externes Projekt zu öffnen und aus seinem Inhalt SPS-Komponenten heraus zu öffnen, um z.B. daraus Inhalt zu kopieren.

Es öffnet sich nach Betätigen des Befehls der Standarddialog zur Auswahl eines Projektes. Der Dateifilter ist so eingestellt, dass entweder ein IndraWorks Projekt (Verzeichnisablage *.iwp oder Eindateiablage *.xiwp) oder direkt ein IndraLogic Projekt (Indralogic.project) ausgewählt werden kann. Für alle Projektarten muss der Anwender zur entsprechenden Datei navigieren, diese selektieren und den Button "Öffnen" betätigen.

Um z.B. direkt eine Indralogic-Projektdatei zu öffnen muss der Anwender zum gewünschten IndraLogic-Projekt navigieren, also z.B. **Erste Schritte Motion ► IndraLogic ► IndraLogic.project**.

Ein IndraWorks-Projekt, das in der Verzeichnisstruktur abliegt wird dadurch geöffnet, dass der Anwender z. B. die Datei **Erste Schritte Motion ► Erste Schritte Motion.iwp** selektiert.

Soll ein IndraWorks Projekt geöffnet werden, das als Eindateiablage vorliegt, muss die entsprechende Datei **<Projektname>.xiwp** ausgewählt werden.

Für alle Projektarten erhält der Anwender einen Dialog, in dem er die zu öffnenden Objekte selektieren kann:

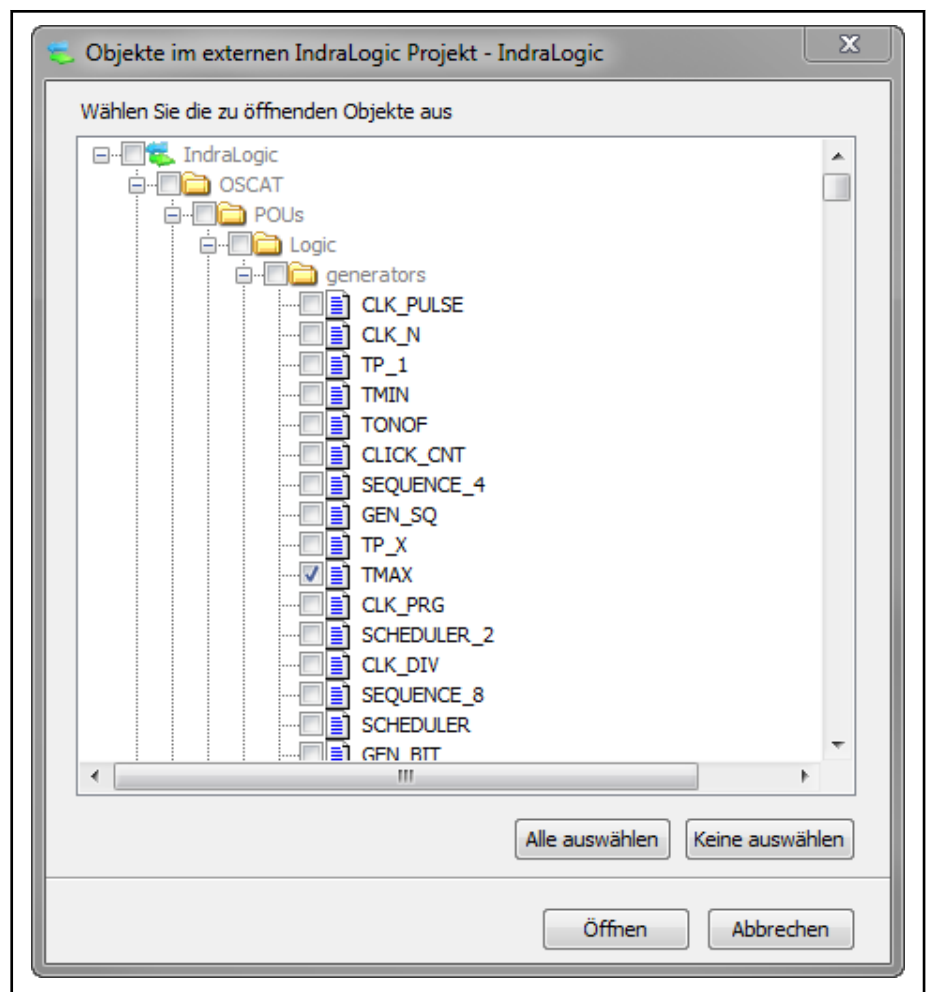


Abb. 18-59: Dialog: SPS-Objektauswahl

Objekte, die selektiert (also angezeigt) werden können, haben schwarze Schriftfarbe, alle anderen sind in grauer Schriftfarbe dargestellt.

Der Anwender kann mit den Schaltflächen <Alle auswählen> bzw. <Keine auswählen> alle Objekte zum Öffnen markieren bzw. deren Markierung wegnehmen.

Durch Klicken mit der Maus in die Schaltfläche vor einem selektierbaren Objekt kann er einzelne Objekte zum Öffnen markieren.

Betätigt der Anwender die <Öffnen>-Schaltfläche werden die selektierten Objekte für lesenden Zugriff geöffnet. Der Anwender kann jetzt in diesen Objekten z.B. Textpassagen markieren und in Objekte des geöffneten Projektes kopieren.

Mit dem Eintrag "Extern:" im Titel des Dokumentenfensters wird dem Anwender visualisiert, dass das geöffnete Objekt aus einem externen Projekt stammt.

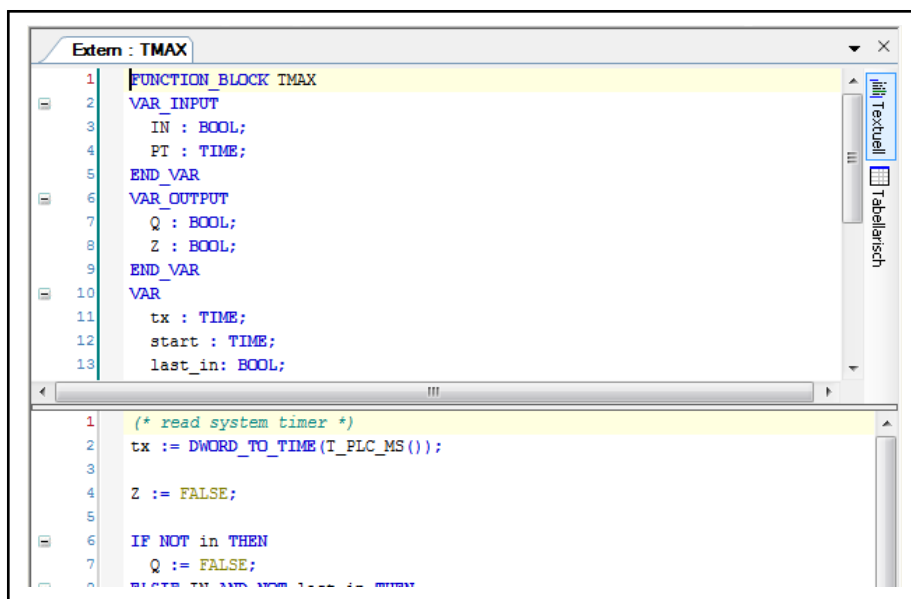


Abb. 18-60: Geöffnetes externes SPS-Objekt

Befehl 'Objekt (offline) bearbeiten'

Funktion: Der Befehl öffnet das Objekt offline in seinem Editor.

Aufruf: Kontextmenü eines offline bearbeitbaren Objektes

Voraussetzung: Die Applikation ist im Online-Betrieb. Ein Objekt im Projekexplorer oder im Ordner "Allgemeiner Baustein" ist selektiert.

Somit können Sie auch im Online-Betrieb das Objekt editieren.

Die Änderung übertragen Sie anschließend mit dem Menübefehl **Debuggen** ► **Online-Change** oder **Debuggen** ► **Laden** auf die Steuerung.

Siehe auch

- [Kap. "Befehl 'Online-Change'" auf Seite 657](#)
- [Kap. "Befehl 'Laden'" auf Seite 656](#)

Befehl 'Aktive Applikation setzen'



Online-Aktionen betreffen immer nur die aktive Applikation.

Beispiele:

- Haltepunkte
- Trace
- Forcen

Funktion Der Befehl setzt die selektierte Applikation als die "aktive Applikation".

Aufruf Im Projektextplorer über das Kontextmenü des Applikations-Objekts **Aktive Applikation setzen**.

Voraussetzungen

- Das Projekt verfügt über mindestens 2 Applikationen
- Die selektierte Applikation ist nicht aktiv

Darstellung in der Benutzeroberfläche Im Projektextplorer wird die aktive Applikation fettgedruckt dargestellt. Zusätzlich wird die aktive Applikation in der IndraLogic Symbolleiste angezeigt.

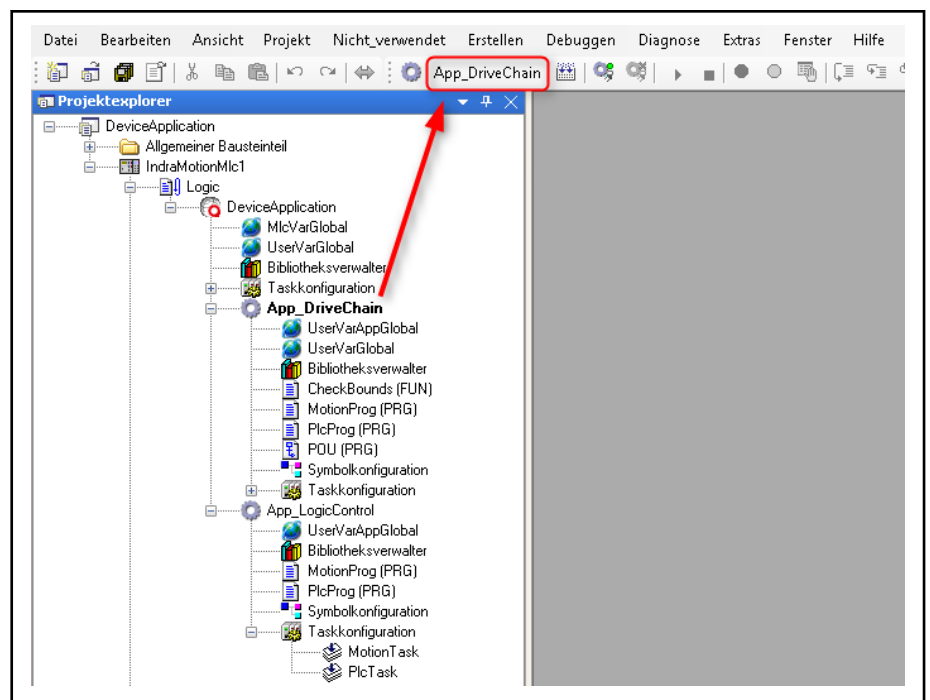


Abb. 18-61: Darstellung einer aktiven Applikation in der Benutzeroberfläche

Befehl 'E/A-Abbild bearbeiten'

Funktion: Der Befehl öffnet den Dialog **E/A-Abbild bearbeiten**. Dieser zeigt alle E/A-Abbilder ("Mappings") des gerade ausgewählten Geräteobjekts plus die E/A-Abbilder aller weiteren Geräteobjekte, die im Projektextplorer unter diesem Objekt eingehängt sind.

Aufruf: Kontextmenü eines Geräteobjekts im Projektextplorer.

Dialog 'E/A-Abbild bearbeiten'

Sie können das E/A-Abbild in diesem Dialog genauso bearbeiten wie im Dialog **E/A-Abbild** der einzelnen Geräteeditoren. Die jeweils anderen Dialoge werden unmittelbar entsprechend aktualisiert.

Suche	Eingabefeld für eine Zeichenfolge, nach der die Mapping-Tabelle durchsucht werden soll. Die Suchergebnisse werden gelb markiert.
Filter	Auswahlliste mit Filtern für die in der Mapping-Tabelle dargestellten E/A-Zuweisungen: • Alle anzeigen • Nur Ausgänge anzeigen • Nur Eingänge anzeigen • Nur gemappte Variablen anzeigen • Nur Mappings auf bestehende Variablen anzeigen • Nur Mappings auf neue Variablen anzeigen • Nur nicht-gemappte Variablen anzeigen

Im Kontextmenü finden Sie unter anderem folgende Befehle:

E/A-Abbild in CSV exportieren: Dient dem Speichern der Mappings eines Geräts und seiner Sub-Geräte in eine externe Datei. Sie wählen dazu das Gerät im Projektextplorer oder in der Mapping-Liste aus.

E/A-Abbild von CSV importieren: Dient dem Einfügen von Mappings aus einer vorher durch Export erzeugten Datei.

Siehe auch

- [Kap. 8 "E/A-Anbindung konfigurieren" auf Seite 71](#)
- [Kap. "Befehl 'E/A-Abbild in CSV exportieren'" auf Seite 648](#)
- [Kap. "Befehl 'E/A-Abbild aus CSV importieren'" auf Seite 648](#)

Befehl 'E/A-Abbild aus CSV importieren'

Funktion: Der Befehl öffnet den Standarddialog zum Suchen nach einer Datei im lokalen Dateisystem. Der Filter steht auf Dateiformat `csv`, zur Auswahl einer Datei, in die zuvor die E/A-Abbild-Konfiguration eines Geräts exportiert wurde. IndraLogic liest die Konfiguration zurück in das aktuelle Projekt.

Aufruf: Kontextmenü der Registerkarte **E/A-Abbild** im Geräteeditor eines Geräts.

Voraussetzung: Ein Projekt mit einem Gerät mit E/A-Abbild-Konfiguration ist geöffnet. Das Gerät paßt zur exportierten `csv`-Datei.



Sie können die `csv`-Datei vor dem Import extern bearbeiten. Beachten Sie Folgendes: Die Zuordnung der Einträge der Datei zu dem E/A-Abbild des Geräts erfolgt über den Gerätenamen (Device- Name) und den Kanalnamen (Parameter name). Parameternamen, die nicht eindeutig sind, werden in dieser Datei mit einer fortlaufenden Nummerierung @<n> erweitert.

Felder ohne Inhalt in der `csv`-Datei werden beim Import ignoriert. Um einen vorhandenen Eintrag im E/A-Abbild durch den Import zu entfernen, müssen Sie im betreffenden Feld in der `csv`-Datei ein Leerzeichen einfügen.

Siehe auch

- [Kap. "Befehl 'E/A-Abbild in CSV exportieren'" auf Seite 648](#)
- [Kap. 8.3 "Geräte konfigurieren und E/A Abbild konfigurieren" auf Seite 75](#)

Befehl 'E/A-Abbild in CSV exportieren'

Funktion: Der Befehl öffnet den Standarddialog zum Speichern einer Datei im lokalen Dateisystem. Der Filter steht auf Dateiformat `csv`. Nach Angabe eines Namens und Ablageorts speichert IndraLogic die E/A-Abbild-Konfiguration in eine `csv`-Datei.

Aufruf: Kontextmenü der Registerkarte **E/A-Abbild** im Geräteeditor eines Geräts.

Voraussetzung: Ein Geräteobjekt mit einer E/A-Abbild-Konfiguration ist im Projektextplorer selektiert.



Parameternamen, die nicht eindeutig sind, werden in dieser Datei mit einer fortlaufenden Nummerierung @<n> erweitert.

Siehe auch

- [Kap. "Befehl 'E/A-Abbild aus CSV importieren'" auf Seite 648](#)
- [Kap. 8.3 "Geräte konfigurieren und E/A Abbild konfigurieren" auf Seite 75](#)

Befehl 'Globale Variablenliste (GVL) in XML-Format austauschen'

Eine im IndraLogic-Projekt enthaltene Globale Variablenliste soll in les- und übersetzbarer Form als XML exportiert und wieder importiert werden können.

Ausführung	Der Aufruf erfolgt per Kontextmenü an einer selektierten Globalen Variablenliste • Daten als XML-Datei ablegen... • Datenübernahme aus XML-Datei... Das Auswählen der Dateiablage für Export und Import erfolgt über die Standard-Windows-Mechanismen. Als Dateiendung ist "*.gvl.xml" festgelegt.
Formatbeschreibung	Diese Daten werden in einem XML-Format gespeichert. Hinter dem Tag <file> wird das Tag <GVL> angelegt. Das Tag <interface> enthält den eigentlichen Inhalt der "Globalen Variablenliste" in dem Format in dem er auch im Editor in IndraWorks zu sehen ist. <i>Aufbau der XML-Datei</i> <pre> <file> <GVL> <interface>VAR_GLOBAL_CONSTANT (* define all possible axis, set to maximum axis index to preserve resources *) // set to a maximum of 64 for the L65 uiMAX_AXIS_INDEX: UINT := INT_TO_UINT(MB_MAX_AXIS_NUMBER); END VAR </interface> </GVL> <path>Application.UserConstGlobal</path> </file> </pre>

18.2.4 Menü 'Erstellen'

Der Menüpunkt enthält die Befehle zum Übersetzen, Codeerzeugen und Bereinigen. Die ersten drei Befehle sind standardmäßig enthalten, die weiteren über **Extras ► Anpassen**, Karte **Befehle**, Kategorie "Erstellen" zuordenbar.

Befehl 'Übersetzen'

Symbol: , Tastaturkürzel: <Umsch>+<F11>

Funktion: Der Befehl startet den Übersetzungsprozess für die gerade aktive Applikation.

Aufruf: Menü **Erstellen**

Bei der Übersetzung führt IndraLogic eine syntaktische Prüfung aller Objekte der Applikation durch. Dabei wird jedoch kein Code generiert, wie es bei einem Einloggen auf das Zielsystem oder beim Download der Applikation der Fall ist.

Der Übersetzungsvorgang wird automatisch immer durchgeführt, wenn Sie mit einem geänderten Programm einloggen wollen.

Nach Abschluss der Überprüfung zeigt IndraLogic eventuelle Fehlermeldungen oder Warnungen im Meldungsfenster unter der Kategorie **Übersetzen** an.

Wenn das Programm seit dem letzten fehlerfreien Übersetzungsprozess nicht mehr verändert wurde, wird es nicht neu übersetzt. Es erscheint die Meldung "Die Applikation ist aktuell". Soll die syntaktische Prüfung dennoch wiederholt werden, müssen Sie den Befehl **Neu übersetzen** verwenden.

Siehe auch

- [Kap. "Befehl 'Neu übersetzen'" auf Seite 650](#)

Befehl 'Code erzeugen'

Funktion: Der Befehl startet eine Codeerzeugung für die aktive Applikation.

Aufruf: Menü **Erstellen**

Bei der Codeerzeugung mit diesem Befehl wird wie beim "Laden der Applikation auf die Steuerung" Code erzeugt, der Code wird jedoch nicht auf die Steuerung übertragen. Dabei erfolgen weitere Prüfungen des Quellcodes.

Somit können Sie den Code auf Fehler prüfen, die vom Compiler nicht erkannt werden und diese korrigieren, bevor er im Online-Betrieb verwendet wird.

Siehe auch

- [Kap. 11.2 "Applikationscode erzeugen" auf Seite 187](#)

Befehl 'Bereinigen'

Funktion: Der Befehl löscht die Übersetzungsinformation für die gerade aktive Applikation.

Aufruf: Menü **Erstellen**

Die Übersetzungsinformation wurde beim letzten Download der Applikation im lokalen Dateisystem angelegt und in einer Datei "*.compileinfo" gespeichert (C:\ProgramData\IW-Projects\x\Project\IndraLogic\IndraLogic.IndraMotionMlc1.Application.*.compileinfo).

Nach einem Bereinigen-Prozess ist für die betroffene Applikation kein Online Change mehr möglich. Die Applikation muss wieder komplett auf die Steuerung geladen werden.

Siehe auch

- [Kap. "Befehl 'Alles bereinigen'" auf Seite 651](#)

Befehl 'Neu übersetzen'

Funktion: Der Befehl startet den Übersetzungsprozess für die gerade aktive Applikation, auch wenn sie zuletzt fehlerfrei übersetzt wurde.

Aufruf: Befehl kann über **Extras ▶ Anpassen ▶ Befehle ▶ Kategorie Erstellen** aktiviert werden.

Siehe auch

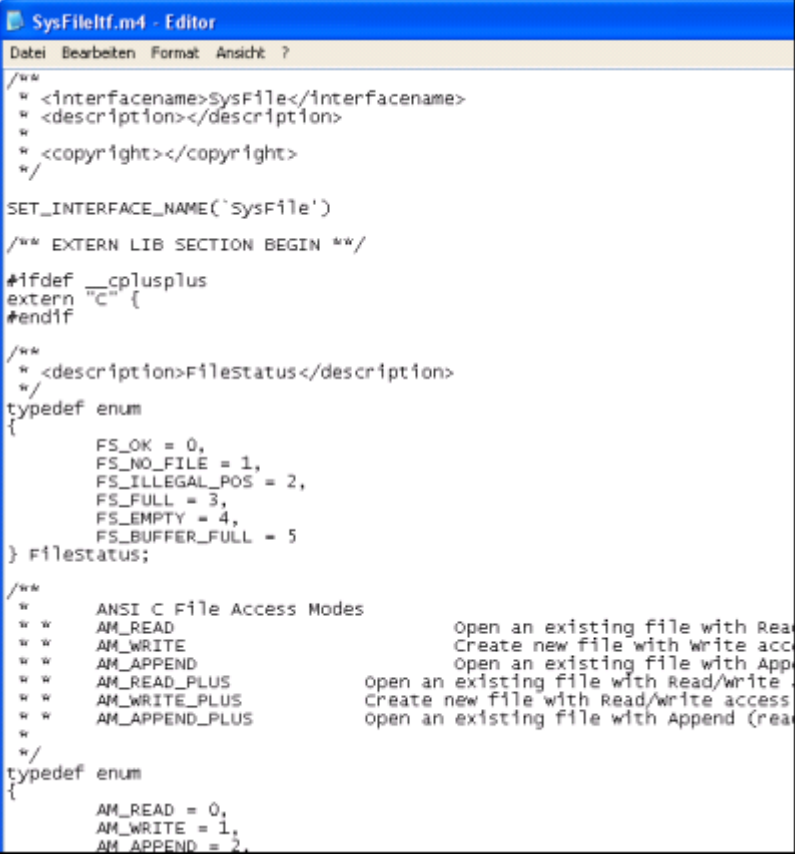
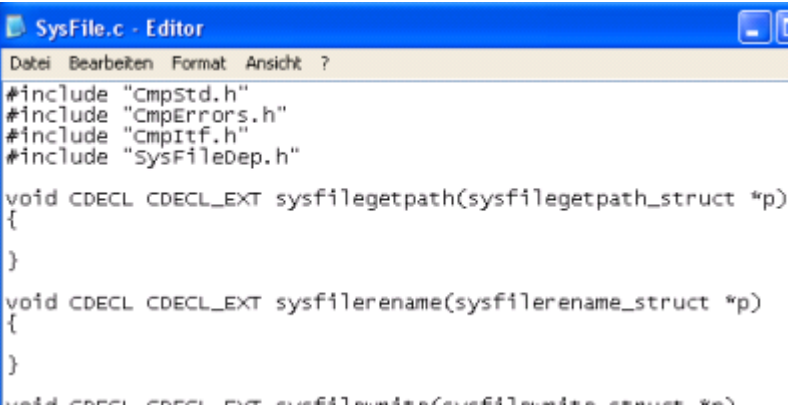
- [Kap. "Befehl 'Übersetzen'" auf Seite 649](#)

Befehl 'Laufzeitsystemdateien erzeugen'

Funktion: Der Befehl generiert aus dem aktuellen Bibliotheksprojekt eine C-Rahmendatei und eine Schnittstellendatei im M4-Dateiformat, die als Basis für die Erstellung einer externen Bibliotheksdatei dienen.

Aufruf: Der Befehl kann über **Extras ▶ Anpassen ▶ Befehle ▶ Kategorie Erstellen** aktiviert werden.

Der Befehl öffnet den Dialog **Laufzeitsystemdateien erzeugen**.

Ausgabeverzeichnis:	Verzeichnis, in dem IndraLogic die Laufzeitsystem-Dateien anlegt. Über Schaltfläche  können Sie den Standarddialog zum Browsen im Dateisystem öffnen.
M4 Interface-Datei	: Schnittstellendatei <projectname>Itf.m4 mit Definitionen. Beispiel m4-Datei:  <pre> 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 Read * 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 (read */ typedef enum { AM_READ = 0, AM_WRITE = 1, AM_APPEND = 2, </pre>
C Stub-Datei:	: Rahmendatei für die Ausprogrammierung der Bibliothek in C. Beispiel Stub-Datei:  <pre> SysFile.c - Editor Datei Bearbeiten Format Ansicht ? #include "CmpStd.h" #include "CmpErrors.h" #include "CmpItf.h" #include "SysFileDep.h" void CDECL CDECL_EXT sysfilegetpath(sysfilegetpath_struct *p) { } void CDECL CDECL_EXT sysfilerename(sysfilerename_struct *p) { } void CDECL CDECL_EXT sysfiledelete(sysfiledelete_struct *p) { } </pre>

Tab. 18-11: Welche Dateien möchten Sie erzeugen

Befehl 'Alles bereinigen'

Funktion: Der Befehl löscht die Übersetzungsinformationen für alle Applikationen des Projekts.

Aufruf: Der Befehl kann über **Extras ► Anpassen ► Befehle ► Kategorie Erstellen** aktiviert werden.

Die Übersetzungsinformation wurde beim letzten Download der Applikation im lokalen Dateisystem angelegt und in einer Datei "*.compileinfo" gespeichert (C:\ProgramData\IW-Projects\x\Project\IndraLogic\IndraLogic.IndraMotionMlc1.Application.*.compileinfo).

Der Befehl bewirkt, dass vor einem erneuten Einloggen zwingend ein Download durchgeführt werden muss. Ein Online Change ist nicht mehr möglich. Im Unterschied zum Befehl **Bereinigen** (nur der aktiven Applikation) erzeugt IndraLogic das Sprachmodell für alle Objekte neu, was zusätzlich Zeit in Anspruch nimmt.



Überlegen Sie sorgfältig, ob es nötig ist, den Befehl anzuwenden. Wenn Sie nur ein erneutes Übersetzen und einen Download der aktiven Applikation wünschen, verwenden Sie den Befehl **Bereinigen**.

Siehe auch

- [Kap. "Befehl 'Bereinigen'" auf Seite 650](#)

18.2.5 Menü 'Debuggen'

Befehl 'Einloggen'

Symbol



Funktion

Der Befehl verbindet die Applikation mit dem Zielsystem (Steuerung, oder simuliertes Gerät) und stellt den Online-Betrieb her.

Aufruf

Menü **Debuggen**, alternativ im Kontextmenü des Objekts **Applikation**, oder über das Tastaturkürzel: <Alt> + <F8>.

Voraussetzung

Die Applikation ist fehlerfrei und die Kommunikationseinstellungen sind konfiguriert.

Wenn die Kommunikationseinstellungen nicht korrekt sind, erscheint ein Dialog. Sie können dann direkt zu den **Kommunikationseinstellungen** der Steuerung wechseln.

Wenn Sie den Befehl **Einloggen** aus dem Debuggen-Menü aufrufen, wird die gerade aktive Applikation mit dem Zielsystem verbunden.

Wenn Sie den Befehl aus dem Kontextmenü aufrufen und im Projektextplorer eine Applikation selektiert haben, wird mit dieser Applikation eingeloggt, auch wenn sie nicht als "aktive Applikation" gesetzt ist.

Wenn auf dem Zielgerät eine Online-Benutzerverwaltung eingerichtet ist, werden Sie beim Einloggen aufgefordert, die Benutzerdaten anzugeben. Dazu erscheint der Dialog **Geräte-Benutzeranmeldung**.

⚠ VORSICHT**Zugriffsmöglichkeiten auf Steuerung prüfen**

Steuerungen dürfen unter keinen Umständen vom Internet oder nicht vertrauenswürdigen Netzen aus zugreifbar sein!

Im Speziellen dürfen die Programmier-Ports der Steuerung unter keinen Umständen ungeschützt aus dem Internet zugreifbar sein (meist UDP-Ports 1740..1743 und TCP-Ports 1217 + 11740 oder die steuerungsspezifischen Ports)!

Wenn ein Zugriff aus dem Internet dennoch ermöglicht werden muss, dann muss zwingend ein sicheres Verfahren gewählt werden, um sich mit der Steuerung zu verbinden (z.B. VPN).



Wenn unter einer Steuerung eine Sicherheitssteuerung projektiert ist, kann dieser Befehl die Kommunikationsverbindungen temporär unterbrechen!

Betroffen sind Verbindungen der Sicherheitssteuerung zu anderen Sicherheitssteuerungen (über Safety-Netzwerkvariablen), zu Feldgeräten und zum Entwicklungssystem. Die sicheren Feldgeräte oder auch andere Sicherheitssteuerung können als Reaktion in den sicheren Zustand gehen. Die Verbindung zum Entwicklungssystem ist nur dann betroffen, wenn die Sicherheitssteuerung über den Feldbus an die Hauptsteuerung angeschlossen ist.

Weitere Informationen finden Sie im Kapitel [Kap. 10.5 "Untergeordnete Sicherheitssteuerung"](#) auf Seite 184

Mögliche Situationen beim Einloggen:

- Auf der Steuerung liegt eine neuere Version der Gerätebeschreibung vor als im Projekt. Sie erhalten eine Warnung mit der Möglichkeit, den Vorgang abzubrechen.
- Die Applikation existiert noch nicht auf der Steuerung: Sie werden aufgefordert, den Download zu bestätigen.
- Die Applikation liegt bereits auf der Steuerung und wurde seit dem letzten Laden nicht verändert. Das Einloggen erfolgt ohne weitere Interaktion mit Ihnen.
- Die Applikation liegt bereits auf der Steuerung, wurde aber seit dem letzten Download verändert.

Sie werden aufgefordert, eine der folgenden Optionen zu wählen:

- Mit Online Change einloggen
- Mit Download einloggen
- Ohne Änderung einloggen

An dieser Stelle erhalten Sie außerdem die Möglichkeit, die Boot-Applikation auf der Steuerung zu aktualisieren.

- Eine unbekannte Version der Applikation liegt bereits auf der Steuerung. Sie werden gefragt, ob IndraLogic diese ersetzen soll.
- Eine Version der Applikation liegt bereits auf der Steuerung und läuft. Sie werden gefragt, ob IndraLogic trotzdem einloggen und die gerade laufende Applikation überschreiben soll.
- Die Applikation auf der Steuerung hält gerade an einem Haltepunkt. Sie haben ausgeloggt und das Programm geändert:

IndraLogic warnt Sie, dass in Falle eines Online Change oder Downloads die SPS komplett angehalten wird. Dies geschieht auch, wenn mehrere Tasks vorhanden sind und nur eine von dem Haltepunkt betroffen ist.

Mit der Schaltfläche **Details** in den oben beschriebenen Dialogen öffnen Sie den Dialog **Applikationsinformation**.

Siehe

- [Kap. "Befehl 'Online-Change'" auf Seite 657](#)
- [Kap. 10.5 "Untergeordnete Sicherheitssteuerung" auf Seite 184](#)

Dialog 'Applikationsinformation' (Details)

Der Dialog zeigt Informationen über die im Programmiersystem geänderte Applikation, im Vergleich zu ihrer Vorgängerversion, die gerade auf der Steuerung liegt. Es gibt zwei Registerkarten:

- **Applikationsinformation:** Die Applikationseigenschaften der **Applikation im IDE** (Integrated Development Environment = Programmiersystem) werden denen der **Applikation in der SPS** gegenüber gestellt: Projektname, Letzte Änderung, IDE-Version, Autor, Beschreibung. Zusätzlich zeigt IndraLogic die Objekte an, die sich seit dem letzten Download geändert haben.
- **Applikationsinhalt:** Wenn die Option **Download Applikation Info** aktiviert ist, kann der Inhalt der Applikation auf der SPS (2) mit dem Inhalt der Applikation im Programmiersystem (1) verglichen werden. Die Option **Download Applikation Info** finden Sie in den Applikationseigenschaften, Kategorie **Übersetzen**.

Wenn der Code im Programmiersystem nicht aktuell ist, erscheint links unten der Befehl **Applikation nicht aktuell. Code jetzt neu erzeugen?** (3). Führen Sie diesen Befehl aus, um den Quellcode in der Applikation zu aktualisieren.

Diese detaillierte Information hilft Ihnen, die Auswirkungen des Einloggens in der aktuellen Situation besser einzuschätzen und eine Entscheidung bezüglich des Downloads der neuen Applikation zu treffen.

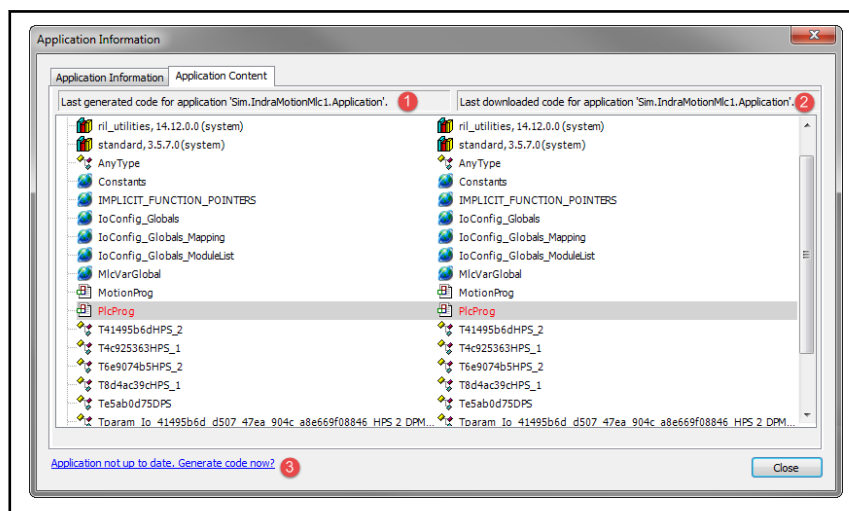


Abb. 18-62:



Der Vergleich wird auch im Geräteeeditor in der Registerkarte **Applikationen** über die Schaltfläche **Inhalt** dargestellt.

Unbekannte Applikationen auf der SPS	<i>Siehe auch</i> • Kap. "Registerkarte 'Applikationen'" auf Seite 521 • Arbeiten mit IndraWorks, Vergleichen von Projektdaten Wenn bereits eine Applikationen auf der Steuerung liegt, die nicht im Projekt vorhanden ist, öffnet IndraLogic einen Dialog mit einer Liste dieser Applikationen. Sie können darin festlegen, ob und welche Applikation vor dem Laden der aktuellen Applikation von der SPS gelöscht werden sollen.
Übersetzen des Projekts vor Einloggen	Wenn ein Applikationsprojekt seit seiner letzten Änderung noch nicht kompiliert wurde, übersetzt IndraLogic das Projekt vor dem Einloggen. Dieser Vorgang entspricht dem Befehl Übersetzen im ausgeloggten Zustand. Wenn während des Übersetzens Fehler auftreten, erscheint ein Meldungsdialog. Die Fehler werden im Meldungsfenster in Kategorie Übersetzen ausgegeben. Sie können dann entscheiden, ob Sie einloggen wollen, ohne das Programm auf die Steuerung zu laden. <i>Siehe auch</i> • Kap. "Befehl 'Übersetzen'" auf Seite 649
Fehler beim Login	Wenn während des Einloggens auf die Steuerung ein Fehler auftritt, bricht IndraLogic den Ladevorgang mit einer Fehlermeldung ab. Der Fehlerdialog bietet Ihnen die Möglichkeit, die Fehlerdetails anzuzeigen. Ist ein Ausnahmefehler aufgetreten und der Text *SOURCEPOSITION* in der Log-Meldung enthalten, können Sie mit dem Befehl Im Editor anzeigen die betroffene Funktion im Editor anzeigen. Dabei springt der Cursor an die fehlerverursachende Zeile.
Informationen zum Ladevorgang	<i>Wenn IndraLogic das Projekt beim Einloggen auf die Steuerung lädt, werden folgende Informationen im Meldungsfenster ausgegeben:</i> • Generierte Codegröße • Größe der globalen Daten • Resultierender Speicherbedarf auf der Steuerung • Eine Liste der betroffenen Bausteine (bei Online Change)
	 Im Online-Betrieb können Sie die Einstellungen von Geräten oder Modulen nicht verändern. Um Geräteparameter zu ändern, müssen Sie die Applikation ausloggen. Abhängig vom Bussystem kann es jedoch einige spezielle Parameter geben, die Sie auch im Online-Betrieb verändern können.
Befehl 'Ausloggen'	 Die Konfiguration der Ansicht speichert IndraLogic im Online- und Offline-Betrieb separat voneinander. Zusätzlich werden Ansichten, welche in einer Betriebsart nicht nutzbar sind, geschlossen. Aus diesem Grund kann sich die Ansicht beim Einloggen automatisch ändern. Symbol  Funktion Der Befehl trennt die Applikation vom Zielsystem (Steuerung oder simuliertes Gerät) und kehrt zum Offline-Betrieb zurück.

Aufruf Im Menü **Debuggen**, alternativ im Kontextmenü des Objekts Applikation, oder über das Tastaturkürzel: <Strg> + <F8>.

Befehl 'Bootapplikation erzeugen'

Funktion Der Befehl erzeugt eine Bootapplikation.

Aufruf Menü **Debuggen**

Eine Bootapplikation ist die Applikation, die automatisch gestartet wird, wenn die Steuerung eingeschaltet oder gestartet wird.

Im Offline-Betrieb können Sie die Bootapplikation in ein beliebiges Verzeichnis speichern. Im Online-Betrieb speichert IndraLogic die Bootapplikation auf dem Zielgerät.

Der Dateiname ist <Applikationsname>.app.

Siehe auch

- [Kap. 11.4 "Erzeugen einer Bootapplikation" auf Seite 189](#)

Befehl 'Aktuellen Gerätebenutzer ausloggen'

Symbol 

Funktion Der Befehl loggt den gerade auf der Steuerung eingeloggten Benutzer aus. Wenn IndraLogic noch eine Verbindung zur Steuerung hat, wird diese damit beendet.

Aufruf IndraWorks Menü **Debuggen**

Voraussetzungen

- Die Applikation ist im Online-Betrieb.
- Eine Benutzerverwaltung ist eingerichtet.



Sie können die Online-Benutzerverwaltung **Projekt ► Benutzerverwaltung** nutzen, das Zielgerät zu schützen.

Siehe auch

- IndraWorks-Benutzerverwaltung

Befehl 'Laden'

Funktion Der Befehl bewirkt ein Übersetzen der aktiven Applikation mit anschließendem Download auf die Steuerung.

Aufruf Menü **Debuggen**

Voraussetzung Die Applikation ist im Online-Betrieb.

Mit dem Befehl führt IndraLogic eine Syntax-Prüfung durch und erzeugt den Applikations-Code.

Dieser Code wird auf die Steuerung geladen.

IndraLogic erzeugt im Projektverzeichnis ein Übersetzungsprotokoll:

Beispiel:

```
<Projektname>.<Gerätename>.<Projektname>.<Applikation> .compileinfo
C:\ProgramData\IW-Projects\x\Project\IndraLogic\IndraLogic.IndraMotionMlc1.Application.*.compileinfo).
```



Beim Laden werden alle Variablen mit Ausnahme von persistenten Variablen neu initialisiert.



Wenn unter einer Steuerung eine Sicherheitssteuerung projiziert ist, kann dieser Befehl die Kommunikationsverbindungen temporär unterbrechen!

Betroffen sind Verbindungen der Sicherheitssteuerung zu anderen Sicherheitssteuerungen (über Safety-Netzwerkvariablen), zu Feldgeräten und zum Entwicklungssystem. Die sicheren Feldgeräte oder auch andere Sicherheitssteuerung können als Reaktion in den sicheren Zustand gehen. Die Verbindung zum Entwicklungssystem ist nur dann betroffen, wenn die Sicherheitssteuerung über den Feldbus an die Hauptsteuerung angeschlossen ist.

Weitere Informationen finden Sie im Kapitel [Kap. 10.5 "Untergeordnete Sicherheitssteuerung"](#) auf Seite 184

Die Beschreibung des Befehls **Einloggen** erläutert die möglichen Situationen beim Einloggen und Laden.

Wenn Sie versuchen, eine Applikation zu laden, während die gleiche Version dieser Applikation bereits auf der Steuerung liegt, erscheint die Meldung: "Programm ist unverändert. Applikation wurde nicht geladen".

IndraLogic lädt die Applikation nicht auf die SPS.

Beim Laden erscheint im Meldungsfenster in der Meldungskategorie **Übersetzen** ein Protokoll der ablaufenden Aktionen (Erzeugen des Codes, Durchführen der Initialisierungen etc.). Weiterhin werden Informationen zu den Speicherbereichen, zur Größe des Codes, der globalen Daten und des allozierten Speichers ausgegeben.

Zum Zweck der Übersichtlichkeit werden im Gegensatz zum Online Change die geänderten Bausteine nicht mehr aufgelistet.

Siehe auch

- [Kap. "Befehl 'Einloggen'"](#) auf Seite 652
- [Kap. 10.5 "Untergeordnete Sicherheitssteuerung"](#) auf Seite 184

Befehl 'Online-Change'

Funktion	Der Befehl dient dem Anstoßen eines Online-Change auf die gerade aktive Applikation. Dabei lädt IndraLogic nur die geänderten Teile einer bereits auf der Steuerung laufenden Applikation neu in die Steuerung.
Aufruf	Menü Debuggen , alternativ im Kontextmenü des Objekts Applikation .
Voraussetzung	Die Applikation ist im Projektextplorer selektiert und ist im Online-Betrieb. Der Befehl führt einen Online-Change gezielt auf die selektierte Applikation aus, auch wenn diese nicht als "aktive Applikation" konfiguriert ist. Ein Online-Change ist nicht möglich nach den Befehlen Erstellen ► Bereinigen und Erstellen ► Alles bereinigen. Der Bereinigen-Prozess löscht die Compile-Informationen (Übersetzungsprotokoll), die automatisch bei jeder Code-Generierung gespeichert werden und die die Basis für einen Online-Change sind.

⚠ VORSICHT

Online-Change verändert das laufende Applikationsprogramm und bewirkt keinen Neustart.

Stellen Sie sicher, dass der neue Applikationscode 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.



1. Ausgänge behalten ihren Wert dauerhaft aus dem letzten Zyklus, wenn im SPS-Programm eine Änderung vorgenommen wird, in dessen Folge Ausgänge nicht mehr verwendet werden und die Änderung per Online Change auf die Steuerung geladen wird.
2. Wenn ein Online-Change durchgeführt wird, werden die applikationsspezifischen Initialisierungen (Referenzfahrt etc.) nicht ausgeführt, weil die Maschine ihren Status beibehält. Aus diesem Grund hat der neue Programmcode nicht den gewünschten Effekt.
3. Zeigervariablen behalten ihren Wert aus dem letzten Zyklus. Wenn ein Pointer 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.

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.

Beim Laden listet IndraLogic im Meldungsfenster in der Kategorie "Übersetzen" auch die geänderten Schnittstellen, betroffenen Variablen und alle Bausteine, für die neuer Code generiert wurde. Wenn sich Speicherorte ändern, wird in einer Dialogbox auf mögliche Probleme in Zusammenhang mit Pointern hingewiesen.



Ab IndraWorks 14VRS wurde eine schnelle Version des Online Change für kleine Änderungen eingeführt:

In allen Fällen, in denen nur Code geändert wurde, werden nur die betroffenen POU's übersetzt und nachgeladen, weiterer Code wird nicht ausgeführt.

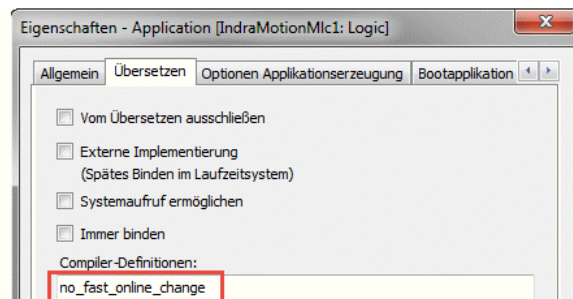
Insbesondere wird kein Initialisierungscode ausgeführt.

Das bedeutet, dass auch kein Code erzeugt wird zur Initialisierung von Variablen mit dem Attribut `init_on_onlchange`.

In der Regel wird das keine Auswirkungen haben, da das Attribut meist dazu verwendet wird, um Variablen mit Adressen zu initialisieren, es kann aber beim schnellen Online-Change nicht dazu kommen, dass eine Variable ihre Adresse ändert.



Um die Wirkung des Attributs `init_on_onlchange` auf den gesamten Applikationscode sicherzustellen, müssen Sie den schnellen Online-Change mit Hilfe der Compiler-Definition `no_fast_online_change` generell für die Applikation ausschalten.



Fügen Sie die Definition zu diesem Zweck in den "Eigenschaften" der Applikation in Registerkarte "Übersetzen" ein.

Siehe auch

- [Kap. "Attribut 'init_on_onlchange'" auf Seite 416](#)
- [Kap. 11.2 "Applikationscode erzeugen" auf Seite 187](#)

Befehl 'Start'

Symbol



Funktion

Der Befehl startet die Applikation (Zustand "LÄUFT").

Aufruf

Im Menü **Debuggen**, alternativ über das Kontextmenü des Objekts **Applikation**, oder über das Tastaturkürzel: <Alt> + <F5>.

Voraussetzung

Die Applikation ist im Online-Betrieb im Zustand "STOP".

Wenn Sie den Befehl über das Menü "Debug" aufrufen, wirkt er auf die aktive Applikation, siehe [Kap. "Befehl 'Aktive Applikation setzen'" auf Seite 646](#)

Befehl 'Stop'

Symbol



Funktion

Der Befehl stoppt die Applikation (Zustand "STOP").

Aufruf

Im Menü **Debuggen**, alternativ im Kontextmenü des Objekts **Applikation**, oder über das Tastaturkürzel: <Strg> + <F5>.

Voraussetzung

Die Applikation ist im Online-Betrieb im Zustand "LÄUFT".

Wenn Sie den Befehl über das Menü "Debug" aufrufen, wirkt er auf die aktive Applikation, siehe [Kap. "Befehl 'Aktive Applikation setzen'" auf Seite 646](#)

Befehl 'Einzelzyklus'

Funktion

Der Befehl führt die aktive Applikation für einen Zyklus aus.

Aufruf

Menü **Extras** ► **Anpassen** ► **Befehle** Kategorie "Online".

Voraussetzung: Die Applikation ist im Online-Betrieb und das Programm steht an einem Programmschritt.

Befehl 'Mehrfacher Download'

Funktion

Der Befehl bewirkt die Code-Erzeugung der im Projekt enthaltenen Applikationen, sowie das Laden der Applikationen auf die entsprechenden Steuerungen.

Aufruf Im Menü **Debuggen**.

Der Befehl öffnet einen Dialog mit einer Liste der Applikationen.

In diesem Dialog wählen Sie die Applikationen aus, die geladen werden sollen.

Danach führt IndraLogic die syntaktische Überprüfung dieser Applikationen durch und erzeugt den zugehörigen Code. Der Code wird anschließend auf die entsprechende Steuerung geladen.

Im Projektverzeichnis (C:\ProgramData\IW-Projects\x\Project\IndraLogic) erzeugt IndraLogic für jede ausgewählte Applikation ein Übersetzungsprotokoll mit dem Namen <project name>.<device name>.<application ID> .compileinfo.



Wenn unter einer Steuerung eine Sicherheitssteuerung projektiert ist, kann dieser Befehl die Kommunikationsverbindungen temporär unterbrechen!

Betroffen sind Verbindungen der Sicherheitssteuerung zu anderen Sicherheitssteuerungen (über Safety-Netzwerkvariablen), zu Feldgeräten und zum Entwicklungssystem. Die sicheren Feldgeräte oder auch andere Sicherheitssteuerung können als Reaktion in den sicheren Zustand gehen. Die Verbindung zum Entwicklungssystem ist nur dann betroffen, wenn die Sicherheitssteuerung über den Feldbus an die Hauptsteuerung angeschlossen ist.

Weitere Informationen finden Sie im Kapitel [Kap. 10.5 "Untergeordnete Sicherheitssteuerung"](#) auf Seite 184

Dialog 'Mehrfacher Download'

Bitte wählen Sie die Applikationen aus	 Auswahl der Applikationen. Dabei werden die Applikationen auch auf unterschiedliche Steuerungen geladen.
Nach oben, Nach unten	Änderung der Download-Reihenfolge der Applikationen. Die Applikationen werden in der Reihenfolge dieser Liste auf die Steuerungen geladen. Standardmäßig ist diese Liste alphabetisch sortiert. Eltern-Kind-Beziehungen von Applikationen werden dabei berücksichtigt.
OK	Überprüft alle ausgewählten Applikationen syntaktisch. Danach wird für jede Applikation die Kommunikation mit der zugehörigen Steuerung verifiziert, bevor der Download erfolgt.

Online Change Optionen

Für den Fall, dass schon eine frühere Version auf der Steuerung existiert und sich diese von der aktuellen Version unterscheidet, stehen Ihnen folgenden Optionen zur Verfügung:	
Online Change versuchen. Falls nicht möglich, Download durchführen	Standardmäßig aktiviert. Wenn ein Online-Change für eine der Applikationen nicht vollzogen werden kann, wird ein Download durchgeführt.

Online Change erzwingen. Falls nicht möglich, Operation abbrechen	Wenn ein Online-Change für (zumindest) eine der Applikationen nicht vollzogen werden kann, wird kein Download durchgeführt, sondern es wird abgebrochen; beispielsweise, wenn Sie zuvor den Befehl "Alles bereinigen" ausgeführt haben.
Immer vollen Download durchführen	Lädt ohne Berücksichtigung der bereits existierenden Versionen alle Teile der Applikationen auf die Steuerung

Für ausgewählte Applikationen, die noch nicht auf der Steuerung existieren, führt IndraLogic automatisch einen Download auf die zugehörige Steuerung durch.

Zusätzliche Optionen

Alle Applikationen auf der Steuerung löschen, die nicht im Projekt enthalten sind	 : Entsprechende Applikationen werden gelöscht
Nach Download oder Online Change alle Applikationen starten	 : Die Applikationen werden nach dem Download / Online Change gestartet.



Beachten Sie, dass Variablen mit dem Schlüsselattribut `PERSISTENT RETAIN` im Allgemeinen nicht initialisiert werden. Wenn Sie jedoch das Datenlayout verändern, werden die persistenten Variablen automatisch neu initialisiert.

Nach Abschluss des Downloads erscheint eine Auflistung aller ausgewählten Applikationen in der von Ihnen konfigurierten Reihenfolge ihres Downloads. Zusätzlich erhalten Sie für jede Applikation eine Information zum Erfolg des Downloads.

Siehe auch

- [Kap. "Befehl 'Laden'" auf Seite 656](#)
- [Kap. 11.2 "Applikationscode erzeugen" auf Seite 187](#)

Befehl 'Reset Warm'

Funktion	Der Befehl bewirkt auf der Steuerung einen Warmstart der aktiven Applikation
Aufruf	Im Menü Debuggen
Voraussetzung	Die Applikation ist im Online-Betrieb. Wenn Sie keine Initialisierungswerte festgelegt haben, setzt IndraLogic die Variablen auf den Default-Initialisierungswert zurück (beispielsweise Integer-Variablen auf 0). Als Vorsichtsmaßnahme verlangt IndraLogic 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"). Haltepunkte der Applikation, die vor dem Zurücksetzen der aktiven Applikation aktiviert waren, sind nach der Ausführung des Befehl weiterhin aktiviert. Haltepunkte die vorher deaktiviert waren, sind nach der Ausführung des Befehls weiterhin deaktiviert.

Wenn Sie den Befehl aufrufen, während der Programmablauf gerade auf einem Haltepunkt stoppt, werden Sie gefragt, ob der aktuelle Zyklus noch beendet werden soll.

Alternativ führt IndraLogic den Reset sofort aus.



Wenn unter einer Steuerung eine Sicherheitssteuerung projektiert ist, kann dieser Befehl die Kommunikationsverbindungen temporär unterbrechen!

Betroffen sind Verbindungen der Sicherheitssteuerung zu anderen Sicherheitssteuerungen (über Safety-Netzwerkvariablen), zu Feldgeräten und zum Entwicklungssystem. Die sicheren Feldgeräte oder auch andere Sicherheitssteuerung können als Reaktion in den sicheren Zustand gehen. Die Verbindung zum Entwicklungssystem ist nur dann betroffen, wenn die Sicherheitssteuerung über den Feldbus an die Hauptsteuerung angeschlossen ist.

Weitere Informationen finden Sie im Kapitel [Kap. 10.5 "Untergeordnete Sicherheitssteuerung"](#) auf Seite 184

Was bei einem Neustart mit "Reset Warm" passiert:

- Applikationscode bleibt auf der Steuerung geladen
- Variablen werden initialisiert (mit dem Initialisierungswert oder dem Standard-Initialisierungswert 0)
- Retain-Variablen bleiben mit Werten erhalten
- Persistente Variablen bleiben mit Werten erhalten
- Haltepunkte, die im Code gesetzt waren, bleiben mit Status (beispielsweise aktiviert oder deaktiviert) erhalten
- Applikation geht in den Zustand "STOP"

Sie können den Befehl auch wählen, während Sie die Applikation debuggen und diese im Zustand "HALT AUF BP" an einem Haltepunkt steht. Dann wird entweder sofort der Warmstart ausgeführt, oder es werden zunächst die noch fehlenden Anweisungen des aktuellen Zyklus fertig abgearbeitet. Deshalb erscheint ein Meldungsfenster, in dem Sie das weitere Verhalten auswählen können. Das Meldungsfenster erscheint allerdings nur, wenn das Laufzeitsystem fähig ist, einen Neustart ohne vorheriges Beenden des Zyklus auszuführen.

Sie können nach dem Reset die Applikation wie gewohnt bedienen und beispielsweise mit dem Befehl **Start** die Ausführung starten.

Siehe auch

- [Kap. 12.5 "Reset der Applikation durchführen"](#) auf Seite 200
- [Kap. 9.16.1 "Daten mit persistenten Variablen erhalten"](#) auf Seite 148
- [Kap. 12.2 "Verwenden von Haltepunkten"](#) auf Seite 193
- [Kap. 17.2.11 "Persistente Variable - PERSISTENT"](#) auf Seite 308
- [Kap. 10.5 "Untergeordnete Sicherheitssteuerung"](#) auf Seite 184
- [Kap. "Befehl 'Reset Kalt'"](#) auf Seite 662
- [Kap. "Befehl 'Reset Ursprung'"](#) auf Seite 663

Befehl 'Reset Kalt'

Funktion	Der Befehl bewirkt auf der Steuerung einen Kaltstart der aktiven Applikation.
Aufruf	Im Menü Debuggen .
Voraussetzung	Die Applikation ist im Online-Betrieb.



Wenn unter einer Steuerung eine Sicherheitssteuerung projiziert ist, kann dieser Befehl die Kommunikationsverbindungen temporär unterbrechen!

Betroffen sind Verbindungen der Sicherheitssteuerung zu anderen Sicherheitssteuerungen (über Safety-Netzwerkvariablen), zu Feldgeräten und zum Entwicklungssystem. Die sicheren Feldgeräte oder auch andere Sicherheitssteuerung können als Reaktion in den sicheren Zustand gehen. Die Verbindung zum Entwicklungssystem ist nur dann betroffen, wenn die Sicherheitssteuerung über den Feldbus an die Hauptsteuerung angeschlossen ist.

Weitere Informationen finden Sie im Kapitel [Kap. 10.5 "Untergeordnete Sicherheitssteuerung"](#) auf Seite 184

Was bei einem Neustart mit "Reset Kalt" passiert:

- Applikationscode bleibt auf der Steuerung erhalten
- Variablen werden initialisiert (mit dem Initialisierungswert oder dem Standard-Initialisierungswert 0), die bisherigen Werte gehen verloren
- Retain-Variablen werden initialisiert, die bisherigen Werte gehen verloren
- Persistente Variablen bleiben mit Werten erhalten
- Haltepunkte, die im Code gesetzt waren, bleiben mit Status (beispielsweise aktiviert oder deaktiviert) erhalten
- Applikation geht in den Zustand STOP

Sie können den Befehl auch wählen, während Sie die Applikation debuggen und diese im Zustand "HALT AUF BP" an einem Haltepunkt steht. Dann wird entweder sofort der Warmstart ausgeführt, oder es werden zunächst die noch fehlenden Anweisungen des aktuellen Zyklus fertig abgearbeitet. Deshalb erscheint ein Meldungsfenster, in dem Sie das weitere Verhalten auswählen können. Das Meldungsfenster erscheint allerdings nur, wenn das Laufzeitsystem fähig ist, einen Neustart ohne vorheriges Beenden des Zyklus auszuführen.

Sie können nach dem Reset die Applikation wie gewohnt bedienen und beispielsweise mit dem Befehl **Start** die Ausführung starten.

Siehe auch

- [Kap. 12.5 "Reset der Applikation durchführen"](#) auf Seite 200
- [Kap. 9.16.1 "Daten mit persistenten Variablen erhalten"](#) auf Seite 148
- [Kap. 12.2 "Verwenden von Haltepunkten"](#) auf Seite 193
- [Kap. 17.2.11 "Persistente Variable - PERSISTENT"](#) auf Seite 308
- [Kap. 10.5 "Untergeordnete Sicherheitssteuerung"](#) auf Seite 184
- [Kap. "Befehl 'Reset Warm'"](#) auf Seite 661
- [Kap. "Befehl 'Reset Ursprung'"](#) auf Seite 663

Befehl 'Reset Ursprung'

Funktion	Der Befehl bewirkt auf der Steuerung das Urlöschen der aktiven Applikation.
Aufruf	Im Menü Debuggen
Voraussetzung	Die Applikation ist im Online-Betrieb.



Wenn unter einer Steuerung eine Sicherheitssteuerung projektiert ist, kann dieser Befehl die Kommunikationsverbindungen **dauerhaft** unterbrechen!

Betroffen sind Verbindungen der Sicherheitssteuerung zu anderen Sicherheitssteuerungen (über Safety-Netzwerkvariablen), zu Feldgeräten und zum Entwicklungssystem. Die sicheren Feldgeräte oder auch andere Sicherheitssteuerung können als Reaktion in den sicheren Zustand gehen. Die Verbindung zum Entwicklungssystem ist nur dann betroffen, wenn die Sicherheitssteuerung über den Feldbus an die Hauptsteuerung angeschlossen ist.

Weitere Informationen finden Sie im Kapitel [Kap. 10.5 "Untergeordnete Sicherheitssteuerung"](#) auf Seite 184

Was bei einem Neustart mit "Reset Ursprung" passiert:

- Applikationscode wird gelöscht, die Applikation hat folglich keinen Zustand
- Variablen werden gelöscht, die Werte gehen verloren
- Retain-Variablen werden gelöscht, die Werte gehen verloren
- Persistente Variablen werden gelöscht, die Werte gehen verloren
- Haltepunkte, die im Code gesetzt waren, gehen verloren

Siehe auch

- [Kap. 12.5 "Reset der Applikation durchführen"](#) auf Seite 200
- [Kap. 9.16.1 "Daten mit persistenten Variablen erhalten"](#) auf Seite 148
- [Kap. 12.2 "Verwenden von Haltepunkten"](#) auf Seite 193
- [Kap. 17.2.11 "Persistente Variable - PERSISTENT"](#) auf Seite 308
- [Kap. 10.5 "Untergeordnete Sicherheitssteuerung"](#) auf Seite 184
- [Kap. "Befehl 'Reset Warm'"](#) auf Seite 661
- [Kap. "Befehl 'Reset Kalt'"](#) auf Seite 662

Befehl 'Haltepunkte'

Symbol



Funktion Der Befehl öffnet die Ansicht **Haltepunkte**.

Aufruf Im Menü **Debuggen**

Der Dialog zeigt eine Übersicht aller definierten Haltepunkte einer Applikation. Innerhalb der Ansicht stehen alle Befehle für Haltepunkte zur Verfügung.

Applikation	Wählen Sie die gewünschte Applikation aus der Liste.
POU	Name des Bausteins, der den Haltepunkt enthält.
Position	<i>Haltepunkt-Position innerhalb der POU</i> • Texteditor: Zeilen- plus Spaltennummer • Grafischer Editor: Netzwerk oder Elementnummer "(Impl)" im Fall von Funktionsbausteinen zeigt an, dass der Haltepunkt in der Implementierung des Funktionsbausteins sitzt, nicht in einer Instanz.
Instanzpfad	Vollständiger Objektpfad der Haltepunkt-Position.

Tasks	Namen der Tasks, bei deren Ausführung der Haltepunkt wirksam sein soll. Wenn keine Einschränkung gilt, steht hier "(alle)".
Bedingung	• Immer anhalten: Keine zusätzliche Aktivierungsbedingung definiert; der Haltepunkt ist immer aktiv. • Boolescher Ausdruck. Der Ausdruck muss <code>TRUE</code> liefern, damit der Haltepunkt aktiv ist.
Trefferzählbedingung	Angabe, wann (in welcher Abhängigkeit von der Trefferanzahl) der Haltepunkt wirksam werden soll.
Aktuelle Trefferanzahl	Angabe, wie oft der Haltepunkt während der Ausführung bis jetzt bereits durchlaufen ("getroffen") wurde.

Tab. 18-12: Tabelle der aktuellen Haltepunkte

	Neuer Haltepunkt	Der Befehl öffnet den Dialog Eigenschaften Haltepunkt
	Haltepunkt löschen	Entfernt den Haltepunkt; nicht zu verwechseln mit Deaktivieren!
	Haltepunkt aktivieren/deaktivieren	<i>Schaltet den Haltepunkt zwischen Status "aktiviert" und "deaktiviert" hin und her.</i> •  Haltepunkt aktiviert •  Haltepunkt deaktiviert Im Gegensatz zum "Haltepunkt löschen" bleibt ein deaktivierter Haltepunkt in der Liste erhalten und kann wieder aktiviert werden.
	Eigenschaften	Dialog Haltepunkteigenschaften erscheint zur Bearbeitung der Haltepunkt-Parameter. Der Dialog entspricht dem Dialog Neuer Haltepunkt .
	Gehe zur Quellcodeposition	Öffnet die Online-Ansicht des betreffenden Bausteins. Der Cursor steht an der Haltepunkt-Position.
	Alle Haltepunkte löschen	Löscht alle Haltepunkte der Applikation. Die Liste wird geleert. Nicht zu verwechseln mit Deaktivieren!
	Alle Haltepunkte aktivieren	Aktiviert alle gerade deaktivierten Haltepunkte.
	Alle Haltepunkte deaktivieren	Deaktiviert alle gerade aktivierten Haltepunkte. Die Punkte bleiben in der Liste und können wieder aktiviert werden.

Tab. 18-13: Werkzeugleiste

Siehe auch

- [Kap. 18.3.2 "Dialog 'Eigenschaften Haltepunkt'" auf Seite 745](#)
- [Kap. "Befehl 'Neuer Haltepunkt'" auf Seite 666](#)
- [Kap. "Befehl 'Haltepunkt aktivieren'" auf Seite 666](#)
- [Kap. "Befehl 'Haltepunkt deaktivieren'" auf Seite 666](#)
- [Kap. "Befehl 'Haltepunkt umschalten'" auf Seite 666](#)

Befehl 'Neuer Haltepunkt'

Symbol	
Funktion	Der Befehl öffnet den Dialog Eigenschaften Haltepunkt .
Aufruf	Im Menü Debuggen .
Voraussetzung	Die Applikation muss sich im Online-Betrieb befinden.

 Mit dem Befehl **Haltepunkt umschalten** können Sie im Online-Betrieb einen neuen Haltepunkt direkt an der aktuellen Cursor-Position setzen.

Siehe auch

- [Kap. 18.3.2 "Dialog 'Eigenschaften Haltepunkt'" auf Seite 745](#)
- [Kap. 12.2 "Verwenden von Haltepunkten" auf Seite 193](#)

Befehl 'Haltepunkt bearbeiten'

Symbol	
Funktion	Der Befehl öffnet den Dialog Eigenschaften Haltepunkt .
Aufruf	Im Menü Debuggen .
Voraussetzung	Die Applikation ist im Online-Betrieb und der Cursor steht auf einem Haltepunkt.

Siehe auch

- [Kap. 12.2 "Verwenden von Haltepunkten" auf Seite 193](#)
- [Kap. 18.3.2 "Dialog 'Eigenschaften Haltepunkt'" auf Seite 745](#)

Befehl 'Haltepunkt umschalten'

Funktion	Der Befehl setzt einen Haltepunkt, oder löscht einen bestehenden Haltepunkt.
Aufruf	Im Menü Debuggen , alternativ über das Tastaturkürzel <Alt>+<F9>
Voraussetzung	Die Applikation befindet sich im Online-Betrieb und der Cursor steht auf einem Haltepunkt.

Siehe auch [Kap. 12.2 "Verwenden von Haltepunkten" auf Seite 193](#)

Befehl 'Haltepunkt aktivieren'

Funktion	Der Befehl aktiviert einen deaktivierten Haltepunkt.
Aufruf	Im Menü Debuggen .
Voraussetzung	Die Applikation befindet sich im Online-Betrieb und der Cursor steht auf einem deaktivierten Haltepunkt.

Siehe auch [Kap. 12.2 "Verwenden von Haltepunkten" auf Seite 193](#)

Befehl 'Haltepunkt deaktivieren'

Funktion	Der Befehl deaktiviert einen aktivierten Haltepunkt.
Aufruf	Im Menü Debuggen .
Voraussetzung	Die Applikation befindet sich im Online-Betrieb und der Cursor steht auf einem aktivierten Haltepunkt.

Siehe auch [Kap. 12.2 "Verwenden von Haltepunkten" auf Seite 193](#)

Befehl 'Aufrufliste'

Symbol 

Funktion Der Befehl öffnet die Ansicht **Aufrufliste**.

Aufruf Im Menü **Debuggen**.

Die Ansicht "Aufrufliste" ist nützlich, wenn Sie Programme schrittweise ausführen wollen.

Die Aufrufliste zeigt die aktuell erreichte Position mit vollständigem Aufruf-Pfad.

Applikation	Name der aktiven Applikation, die den gerade erreichten Programm-baustein kontrolliert.
Task	Name der Task, die den gerade erreichten Programmbaustein kontrolliert.
POU	Name des Programmbausteins, in dem die Programmausführung steht. Die erste Zeile in der Liste beschreibt die aktuelle Ausführungsposition und ist mit einem gelben Pfeil markiert. Wenn diese Position in einem Baustein liegt, der von einem anderen Baustein 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.
Position	<i>Position innerhalb des Programmbausteins, an der die Programmausführung steht</i> • Zeilen- und Spaltennummer bei Texteditoren • Netzwerk- oder Elementnummer bei grafischen Editoren
Instanzpfad	Instanz, in der die Programmausführung steht.

Die Aufrufliste ist im Offline-Betrieb und im Online-Betrieb verfügbar, wenn keine Debugging-Funktionen angewendet werden. In diesem Fall enthält die Aufrufliste die zuletzt während einer schrittweisen Ausführung angezeigte Position, in "gegrauter" Schrift.



Die Ansicht "Aufrufbaum" liefert im Unterschied zur "Aufrufliste" jederzeit Aufrufinformationen zu einem Baustein.

Siehe auch

- [Kap. 12.2 "Verwenden von Haltepunkten" auf Seite 193](#)
- [Kap. "Befehl 'Aufrufbaum'" auf Seite 631](#)

Befehl 'Überwachung' - 'Überwachungsliste <n>'

Symbol 

Funktion Der Befehl öffnet die Ansicht **Überwachungsliste <n>**.



<n> kann 1, 2, 3, 4 sein, das heißt Sie können bis zu vier Überwachungslisten konfigurieren.

Aufruf Im Menü **Debuggen**.

Siehe auch [Kap. 13.1.2 "Überwachungslisten verwenden" auf Seite 211](#)

Die Überwachungsliste kann mit Variablen aus dem Projekt belegt werden und ermöglicht folgende Funktionen innerhalb einer einzigen Übersicht:

- Monitoren von Variablen
- Forcen von Variablen
- Schreiben von Variablen

Befehl 'Überwachung' - 'Alle Forces anzeigen'

Symbol



Funktion

Der Befehl öffnet die Ansicht **"Alle Forces anzeigen"**.

Aufruf

Im Menü **Debuggen** ► **Überwachung** ► **Alle Forces anzeigen**.

Voraussetzung

Ein Projekt ist im Projektextplorer geöffnet (Offline- oder Online-Betrieb).

Die Ansicht enthält eine Liste mit allen Variablen der Applikation, die zum Forcen vorbereitet sind, oder bereits Geforct werden.

Die Liste bietet alle Aktionen, die auch in anderen Überwachungslisten möglich sind.

Zusätzlich bietet die Liste im Auswahlmenü **Force aufheben** folgende Befehle:

- **Forcen aufheben und alle ausgewählten Werte beibehalten:**
Für alle selektierten Einträge der Liste werden die Variablen auf den geforcten Wert gesetzt und das Forcen aufgehoben.
- **Forcen aufheben und alle ausgewählten Werte wiederherstellen:**
Für alle selektierten Einträge der Liste werden die Variablen auf den Wert zurückgesetzt, den sie vor dem Forcen hatten und das Forcen wird aufgehoben.

Siehe auch

- [Kap. 12.4 "Forcen und Schreiben von Variablen" auf Seite 198](#)
- [Kap. 13.1.2 "Überwachungslisten verwenden" auf Seite 211](#)

Befehl 'Alle Forces zur Überwachungsliste hinzufügen'

Funktion

Der Befehl fügt alle Variablen der Applikation zur Überwachungsliste hinzu, die zum Forcen vorbereitet sind, oder bereits geforct sind.



Der Befehl wird nur von aktiven Überwachungslisten unterstützt!

Aufruf

Im Kontextmenü der Ansicht **"Überwachungsliste"**.

Voraussetzungen

- Die Applikation befindet sich im Online-Betrieb
- Mindestens eine Überwachungsliste ist aktiv



Die Überwachungsliste **"Alle Forces betrachten"** bietet eine Übersicht über alle Variablen einer Applikation, die zum Forcen vorbereitet wurden, oder bereits geforct sind.

Die Überwachungsliste stellt zusätzlich Befehle bereit, um das Forcen aufzuheben.

Siehe auch

- [Kap. 13.1.2 "Überwachungslisten verwenden" auf Seite 211](#)
- [Kap. "Befehl 'Überwachung- 'Überwachungsliste <n>' auf Seite 667](#)
- [Kap. "Befehl 'Überwachung- 'Alle Forces anzeigen'" auf Seite 668](#)

- [Kap. 12.4 "Forcen und Schreiben von Variablen" auf Seite 198](#)

Befehl 'Prozedurschritt'

Symbol	
Funktion	Der Befehl führt die Anweisung, an der das Programm aktuell steht, aus und hält vor der nächsten Anweisung im Programmierbaustein an.
Aufruf	Im Menü Debuggen , oder über das Tastaturkürzel <F12>.
Voraussetzung	Die Applikation ist im Onlinebetrieb und das Programm steht an der aktuellen Halteposition (Debugbetrieb). Wenn in der auszuführenden Anweisung ein Aufruf ist (von einem Programm, einer Funktionsbaustein-Instanz, einer Funktion, einer Methode oder einer Aktion), wird der untergeordnete Programmierbaustein vollständig in einem Schritt durchlaufen und zum Aufruf zurückgekehrt. Dann wird vor der nächsten Anweisung (in der nächsten Codezeile) angehalten.
	Wählen Sie den Befehl "Einzelschritt", um in einen untergeordneten Baustein zu springen, um ihn in Einzelschritten auszuführen.

Siehe auch

- [Kap. 12.3 "Schrittweises Abarbeiten eines Programms \(Stepping\)" auf Seite 196](#)
- [Kap. "Befehl 'Einzelschritt'" auf Seite 669](#)

Befehl 'Einzelschritt'

Symbol	
Funktion	Der Befehl führt die Anweisung, an der das Programm aktuell steht, aus und hält vor der nächsten Anweisung.
Aufruf	Im Menü Debuggen , oder über das Tastaturkürzel <Alt> + <F12>.
Voraussetzung	Die Applikation befindet sich im Onlinebetrieb und das Programm steht an der aktuellen Halteposition (Debugbetrieb). Wenn in der auszuführenden Anweisung ein Aufruf enthalten ist (von einem Programm, einer Funktionsbaustein-Instanz, einer Funktion, einer Methode oder einer Aktion), wird in diesen untergeordneten Programmierbaustein gesprungen. Dessen Code erscheint in einem eigenen Editor. Die erste Anweisung wird ausgeführt und vor der nächsten Anweisung wird angehalten. Die neue aktuelle Halteposition ist dann im aufgerufenen Programmierbaustein.
	Wählen Sie den Befehl "Prozedurschritt", um im aktuell aktiven Programmierbaustein zu bleiben und den Aufruf in einem Schritt zu durchlaufen.

Siehe auch

- [Kap. 12.3 "Schrittweises Abarbeiten eines Programms \(Stepping\)" auf Seite 196](#)
- [Kap. "Befehl 'Prozedurschritt'" auf Seite 669](#)

Befehl 'Ausführen bis Rücksprung'

Symbol	
Funktion	Der Befehl führt das Programm bis zum nächsten Rücksprung aus und hält danach an.
Aufruf	Im Menü Debuggen , oder über das Tastaturkürzel <Umsch> + <F10>.

Voraussetzung	Die Applikation ist im Onlinebetrieb und das Programm steht an der aktuellen Halteposition (Debugbetrieb). Wenn die aktuelle Halteposition in einem untergeordneten Programmierbaustein enthalten ist, wird dieser bis zum Ende durchlaufen. Dann wird zur Aufrufstelle im aufrufenden Programmierbaustein zurückgesprungen und dort angehalten (in der Zeile mit dem Aufruf). Wenn die aktuelle Halteposition im Hauptprogramm enthalten ist, wird der Programmierbaustein bis zum Ende durchlaufen. Dann wird zurück an den Anfang (an den Programmstart an die erste Codezeile im Programmierbaustein) gesprungen und dort angehalten. <i>Siehe auch</i> • Kap. 12.3 "Schrittweises Abarbeiten eines Programms (Stepping)" auf Seite 196

Befehl 'Ausführen bis Cursor'

Symbol	
Funktion	Der Befehl führt ein Programm bis zu einer mit dem Cursor gekennzeichneten Position aus.
Aufruf	Im Menü Debuggen .
Voraussetzung	Die Applikation ist im Onlinebetrieb und das Programm steht an der aktuellen Halteposition (Debugbetrieb). Außerdem haben Sie mit dem Cursor eine beliebige Codezeile in einem beliebigen Programmierbaustein gekennzeichnet. Die Anweisungen, die zwischen der aktuellen Halteposition und der Cursorposition liegen, werden in einem Schritt ausgeführt. Dann hält die Ausführung an der Cursorposition an, die damit zur nächsten Halteposition wird. Beachten Sie, dass die Codezeile, an der Sie den Cursor gesetzt haben, erreicht aber nicht ausgeführt wird. <i>Siehe auch</i> • Kap. 12.3 "Schrittweises Abarbeiten eines Programms (Stepping)" auf Seite 196

Befehl 'Nächste Anweisung festlegen'

Symbol	
Funktion	Der Befehl legt fest, welche Anweisung als nächstes ausgeführt wird.
Aufruf	Im Menü Debuggen .
Voraussetzung	Die Applikation ist im Onlinebetrieb und das Programm steht an der aktuellen Halteposition (Debugbetrieb). Außerdem haben Sie mit dem Cursor eine beliebige Codezeile in einem beliebigen Programmierbaustein gekennzeichnet. Die mit dem Cursor gekennzeichnete Codezeile wird zur aktuellen Halteposition, ohne die Anweisungen dazwischen oder die angesprungene Anweisung auszuführen. Siehe auch Kap. 12.3 "Schrittweises Abarbeiten eines Programms (Stepping)" auf Seite 196

Befehl 'Nächste Anweisung anzeigen'

Symbol	
Funktion	Der Befehl springt zur Programmanweisung, die im nächsten Schritt abgearbeitet wird.
Aufruf	Im Menü Debuggen .

- Voraussetzung** Die Applikation ist im Onlinebetrieb und das Programm steht an der aktuellen Halteposition (Debugbetrieb). Die Halteposition ist in einer für Sie nicht sichtbaren Codezeile.
- Der Befehl bewirkt, dass das Fenster mit der aktuellen Halteposition, die im Code gelb und mit dem Symbol  gekennzeichnet ist, aktiv und die Halteposition sichtbar wird. Das ist nützlich, wenn Sie viele Editoren geöffnet haben und die Halteposition sich verdeckt in einem nicht aktiven Editor befindet.
- Siehe auch [Kap. 12.3 "Schrittweises Abarbeiten eines Programms \(Stepping\)"](#) auf Seite 196

Befehl 'Werte forcen'

- Funktion** Der Befehl setzt den Wert einer Variablen auf der Steuerung dauerhaft auf einen vordefinierten Wert.
- Aufruf** Im Menü **Debuggen**, oder über das Tastaturkürzel <F7>.
- Voraussetzung** Die Applikation ist im Online-Betrieb.

VORSICHT

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 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.

Mit dem Befehl setzt IndraLogic eine oder mehrere Variablen der aktiven Applikation auf der Steuerung permanent auf definierte Werte.

Dieses Setzen erfolgt jeweils am Beginn und am Ende eines Abarbeitungszyklus.

Abfolge der Abarbeitung:

1. Eingänge lesen
2. Werte forcen
3. Code abarbeiten
4. Werte forcen
5. Ausgänge schreiben

Sie können Werte vorbereiten durch:

- Klick in das Feld **vorbereiteter Wert** im Deklarationsteil. Geben Sie hier den neuen Wert ein.
Bei einer booleschen Variable ändern Sie den Wert durch einfaches Klicken in das Feld.
- Klick in das Inline-Monitoring-Feld im Implementierungsteil des FUP/KOP/AWL-Editors. Geben Sie hier den neuen Wert ein
- Klick in das Feld **vorbereiteter Wert** im Überwachen-Fenster. Geben Sie hier den neuen Wert ein.

Ein "geforceter" Wert wird mit einem  Symbol gekennzeichnet.

IndraLogic führt das Forcen durch, bis es explizit durch den Anwender aufgehoben wird. Das Aufheben erfolgt durch:

- den Befehl **Forcen für alle Werte aufheben**

- das Aufheben des Forces über den Dialog **Wert vorbereiten**
- Ausloggen aus der Applikation

Siehe auch

- [Kap. 18.3.3 "Dialog 'Wert vorbereiten'" auf Seite 745](#)
- [Kap. "Befehl 'Forcen für alle Werte aufheben'" auf Seite 672](#)
- [Kap. 12.4 "Forcen und Schreiben von Variablen" auf Seite 198](#)

Befehl 'Werte schreiben'

Funktion	Der Befehl setzt den Wert einer Variablen auf der Steuerung einmalig auf einen vordefinierten Wert.
Aufruf	Im Menü Debuggen , oder über das Tastaturkürzel <Strg> + <F7>.
Voraussetzung	Die Applikation befindet sich im Online-Betrieb.

VORSICHT

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 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.

Mit dem Befehl setzen Sie eine oder mehrere Variablen der aktiven Applikation auf der Steuerung einmalig auf definierte Werte. Das Schreiben erfolgt einmal zu Beginn des nächsten Zyklus.

Sie können Werte vorbereiten durch:

- Klick in das Feld **vorbereiteter Wert** im Deklarationsteil. Geben Sie hier den neuen Wert ein.
Bei einer booleschen Variable ändern Sie den Wert durch einfaches Klicken in das Feld.
- Klick in das Inline-Monitoring-Feld im Implementierungsteil des FUP/KOP/AWL-Editors. Geben Sie den neuen Wert ein.
- Klick in das Feld **vorbereiteter Wert** im Überwachen-Fenster. Geben Sie den neuen Wert ein.

Siehe auch

- [Kap. "Befehl 'Werte forcen'" auf Seite 671](#)
- [Kap. 12.4 "Forcen und Schreiben von Variablen" auf Seite 198](#)

Befehl 'Forcen für alle Werte aufheben'

Funktion	Der Befehl setzt das Forcen aller Variablen zurück. Die Variablen erhalten dabei ihren aktuellen Wert aus der Steuerung.
Aufruf	Im Menü Debuggen , oder über das Tastaturkürzel <Alt> + <F7>.
Voraussetzung	Die Applikation befindet sich im Online-Betrieb.



Der Befehl "Forceliste aufheben" hat die gleiche Funktion wie dieser Befehl, jedoch mit dem Unterschied:

Wenn der Befehl "Forceliste aufheben" nicht für alle geforcen Variablen ausgeführt werden konnte, erscheint keine Meldung.

⚠ VORSICHT

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 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.

Siehe auch

- [Kap. "Befehl 'Werte forcen'" auf Seite 671](#)
- [Kap. 12.4 "Forcen und Schreiben von Variablen" auf Seite 198](#)

Befehl 'Ablaufkontrollmodus umschalten'

Symbol	
Funktion	Der Befehl aktiviert oder deaktiviert die Ablaufkontrolle.
Aufruf	Im Menü Debuggen
Voraussetzung	Die Applikation ist im Online-Betrieb.



Eine aktive Ablaufkontrolle verlängert die Laufzeit der Applikation!

Wenn die Option **Sicherer Online-Modus** in den Kommunikationseinstellungen aktiviert ist, erscheint beim Einschalten der Ablaufkontrolle ein Dialog, über den der Vorgang abgebrochen werden kann.

Das Verwenden von Haltepunkten und die schrittweise Abarbeitung des Programms sind bei aktivierter Ablaufkontrolle nicht möglich.

Siehe auch [Kap. 12.6 "Ablaufkontrolle" auf Seite 201](#)

Befehl 'Betriebsmodus'

Funktion	Über die Betriebsarten des Befehls "Betriebsmodus" kann die Steuerung in einen Zustand geschaltet werden, der ein versehentliches Ändern des Projekts verhindert.
Aufruf	Im Menü Debuggen ► Betriebsmodus .
Voraussetzungen	• Auf der Steuerung muss eine Bootapplikation für die Applikation vorhanden sein • Es dürfen keine aktiven Haltepunkte gesetzt sein • Die Applikation muss laufen • Es dürfen keine geforcten Werte vorhanden sein

Mit dem Befehl Betriebsmodus kann der Zustand einer Steuerung gesperrt werden. Beispielsweise um zu verhindern, dass die Steuerung den Zustand ändert, während an einer anderen Steuerung programmiert wird.

Nach Abschluss der Programmierung kann die Steuerung in einen definierten und von außen sichtbaren Zustand geschaltet werden, der sich nach einem Reboot wieder genau so einstellt.

Folgende Betriebsarten stehen zur Verfügung:

 Debug	Keine Einschränkungen
 Gesperrt	Der aktuelle Zustand des Debuggens auf der Applikation wird gelockt. Es können keine weiteren Haltepunkte gesetzt oder Variablen geforct werden. Das Schreiben von Variablen ist jedoch möglich.
 Betriebsbereit	Es kann nichts geändert werden. Haltepunkte können nicht gesetzt werden. Das Schreiben von Variablen ist jedoch möglich.

In der Statuszeile zeigen die Symbole ,  und  die aktuelle Betriebsart an. Ein Doppelklick auf eines dieser Symbole öffnet ein Hilfefenster.



Sie können die Betriebsart zwischen **Gesperrt** und **Betriebsbereit** nicht umschalten.

Befehl 'Core Dump'**Befehl 'Core Dump erzeugen'**

Funktion Der Befehl bewirkt die Erstellung einer Core-Dump-Datei.

Aufruf Im Menü **Debuggen ► Core Dump**.

Voraussetzung Die Applikation ist im Online-Betrieb.

Der Befehl bewirkt, dass IndraLogic zunächst prüft, ob bereits eine Speicherabzugsdatei auf der Steuerung verfügbar ist.

Wenn bereits eine Core-Dump-Datei vorhanden ist, bietet IndraLogic an, diese Datei ins Projektverzeichnis zu laden.

Unter folgenden Voraussetzungen veranlasst IndraLogic das Erzeugen einer neuen Core-Dump-Datei mit den aktuellen Applikationsdaten:

- Es ist noch keine Core-Dump-Datei vorhanden, oder IndraLogic lehnt das Laden der vorhandenen Core-Dump-Datei ab
- UND: Die Applikation steht gerade an einem Haltepunkt, oder es ist ein Ausnahmefehler aufgetreten

Die erzeugte Core-Dump-Datei wird direkt im Projektverzeichnis abgelegt:

Beispiel:

```
<Projectname>.<Devicename>.<Applicationname>.<Applicationguid>.core
```

(z. B. C:\ProgramData\IW-Projects\x\Project\IndraLogic\IndraLogic.IndraMotionMlc1.Application.*.core).

Sie können die Erzeugung über eine Schaltfläche in der Statusleiste abbrechen.

Wie detailliert der Core-Dump ausfällt, hängt von der Unterstützung durch das Laufzeitsystem ab. Im Falle eines Ausnahmefehlers wird immer ein Core-Dump erzeugt.

Loggen Sie dann aus der SPS aus.

Danach ist es möglich einen Core Dump zu laden.

Das Anzeigen des Core-Dumps mit Hilfe des Befehls **Core Dump laden** kann somit für die Fehleranalyse verwendet werden.

Siehe auch

- [Kap. 13.6 "Fehleranalyse mit Core Dump" auf Seite 224](#)
- [Kap. "Befehl 'Core Dump laden'" auf Seite 675](#)

Befehl 'Core Dump laden'

Funktion	IndraLogic durchsucht das Projektverzeichnis nach Core-Dump-Dateien.
Aufruf	Im Menü Debuggen ► Core Dump .
Voraussetzung	Die Applikation ist im Offline-Betrieb. Wenn ein neuer Core Dump mit Befehl Core Dump erzeugen erzwungen wird, wird die Dump-Datei automatisch von der Steuerung ins Projektverzeichnis geladen. Falls mehrere Core-Dump-Dateien vorliegen erfolgt eine Abfrage, ob die zuletzt erzeugte Datei im Projekt geöffnet werden soll. Alternativ können Sie auch eine der anderen Dateien auswählen. Das Laden ins Projekt bewirkt, dass eine Online-Ansicht der Applikation erscheint, mit dem Stand, den die Applikation zum Zeitpunkt des Erzeugens des Core Dumps hatte. Darin können Sie die Variablenwerte nachträglich ansehen. Zudem ist der Aufrufbaum verfügbar.



Sie können die Core-Dump-Ansicht nur über den Befehl **Core Dump schließen** schließen.

Der Befehl **Ausloggen** ist in dieser Ansicht nicht wirksam!

Siehe auch

- [Kap. 13.6 "Fehleranalyse mit Core Dump" auf Seite 224](#)
- [Kap. "Befehl 'Core Dump erzeugen'" auf Seite 674](#)
- [Kap. "Befehl 'Core Dump schließen'" auf Seite 675](#)

Befehl 'Geräte-Logbuch vom Core Dump laden'

Funktion	Der Befehl lädt die Log-Liste der Steuerung, die mit dem zuletzt dort erzeugten Core Dump gespeichert wurde, ins Projekt.
Aufruf	Im Menü Debuggen ► Core Dump .
Voraussetzungen	• Die Applikation ist im Offline-Betrieb • Ein Core Dump ist im Projekt geöffnet

Die Log-Liste erscheint in der gleichen Ansicht, wie sie auch im Online-Betrieb in der Registerkarte **Log** des Geräte-Editors angezeigt wird.

Siehe auch:

- [Kap. 13.6 "Fehleranalyse mit Core Dump" auf Seite 224](#)
- [Kap. "Registerkarte 'Log'" auf Seite 525](#)

Befehl 'Core Dump schließen'

Funktion	Der Befehl schließt die, im Projekt geöffnete Core-Dump-Ansicht der Applikation.
Aufruf	Im Menü Debuggen ► Core Dump .
Voraussetzungen	• Die Applikation ist im Offline-Betrieb • Eine Core-Dump-Datei wurde aus der Steuerung ins Projekt geladen

Siehe auch [Kap. 13.6 "Fehleranalyse mit Core Dump" auf Seite 224](#)

Befehl 'Anzeigemodus'

Funktion	Der Befehl "Anzeigemodus" besitzt ein Untermenü für die Anzeigeformate Binär, Hexadezimal und Dezimal. Mit der Selektion eines der drei Formate wird die Darstellung der Werte beim Monitoring im Online-Betrieb festgelegt.
Aufruf	Im Menü Debuggen .
Voraussetzung	Das Projekt ist im Offline-, oder Online-Betrieb.

Unterstützte Anzeigeformate

Binär	vorzeichenlos
Hexadezimal	vorzeichenlos
Dezimal	vorzeichenbehaftet

Siehe auch [Kap. 13.1.1 "Monitoring in Programmierobjekten aufrufen"](#) auf [Seite 205](#)

18.2.6 Menü 'Extras'

Bibliotheken

Befehl 'Bibliotheks-Repository'

Symbol: 

Funktion: Der Befehl öffnet den Dialog **Bibliotheks-Repository**. In diesem Dialog legen Sie fest, welche Bibliotheken, auf dem lokalen System installiert und damit für Ihre Applikation verfügbar sind.

Aufruf: Menü **Extras** ► **Bibliotheken**.

Dialog 'Bibliotheks-Repository'

Anzeige des Verzeichnisses auf dem lokalen System, in dem die Bibliotheksdateien liegen. Die Bibliotheken dieses **Speicherort** sind im Bereich **Installierte Bibliotheken** aufgelistet.

Bearbeiten	Öffnet den Dialog Repository-Speicherorte bearbeiten .

Tab. 18-14: **Speicherort**



1. Sie können für neue Repositorys nur leere Verzeichnisse verwenden.
2. Als Speicherort können Sie auch bestehende Repositorys verwenden.
3. Das "System"-Repository ist nicht editierbar; IndraLogic zeigt dies durch die Kursivschrift des Eintrags an.

Repository-Speicherorte bearbeiten

Auflistung der Repositorys mit Speicherort und Name .	
Hinzufügen	Legt ein neues Repository an. Öffnet den Dialog Speicherort für Repository . Das ausgewählte Verzeichnis (Eingabefeld Ort) muss leer sein oder ein bereits bestehendes gültiges Repository sein. Name ist das Eingabefeld für einen symbolischen Repository-Namen.

Bearbeiten	Öffnet den Dialog Speicherort für Repository (siehe Hinzufügen)
Entfernen	Es 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. Wenn Sie das Verzeichnis löschen wollen, müssen Sie dies bestätigen.

Installierte Bibliotheken

Liste der Bibliotheken in Baumstruktur. Darstellung jeder Bibliothek mit Kategorie, Name, Firma und Version.	
Firma:	Auswahlliste zur Filterung der angezeigten Bibliotheken.
Installieren	Öffnet den Dialog Bibliothek auswählen . Standardmäßig ist der Filter auf IndraLogic-Bibliotheken gesetzt, die standardmäßig die Erweiterung "*.library" tragen. Der Filter "IndraLogic-Bibliotheken (vor 2G)" zeigt alle Bibliotheken mit Erweiterung "*.lib".
Deinstallieren	Deinstalliert die selektierte Bibliothek.
Exportieren	Öffnet den Standarddialog zum Speichern des Bibliotheksprojekts ins lokale Dateisystem. Der Dateityp ist Bibliotheken (*.library) oder Übersetzte Bibliotheken (*.compiled-library) .
Suchen	Sucht Bibliotheken und Funktionsbausteine. Öffnet den Dialog Bibliothek suchen . Wenn Sie in das Eingabefeld eine Zeichenfolge eingeben, zeigt IndraLogic die Bibliotheken mit gefundenen Übereinstimmungen an.
Details	Öffnet für die selektierte Version einer Bibliothek den Dialog Details mit Details aus den "Bibliotheksinformationen" der Bibliothek. Über die Schaltfläche Mehr im Dialog Details erhalten Sie außerdem folgende Informationen: • Größe: Angabe in Bytes • Erzeugt: Erstellungsdatum • Geändert: Datum der letzten Änderung • Letzter Zugriff: Datum • Attribute • Eigenschaften
Abhängigkeiten	Öffnet für die ausgewählte Bibliothek den Dialog Abhängigkeiten zur Darstellung der Abhängigkeiten von anderen Bibliotheken. Für jede Bibliotheksreferenz werden Titel , Version und Firma angezeigt. Referenzen, die über Platzhalter funktionieren, sind nach Syntax: #<Platzhaltername> dargestellt.
Gruppieren nach Kategorie	• ☒: Gruppierung nach Bibliothekskategorien • ☐: Alphabetische Sortierung Die Kategorien werden durch externe Beschreibungsdateien "*.libcat.xml" definiert.

Bibliotheksprofile

Ein Bibliotheksprofil definiert, mit welcher Bibliotheksversion IndraLogic einen Bibliotheks-Platzhalter auflöst, wenn eine bestimmte Compiler-Version im Projekt gesetzt ist.	
Importieren	Importiert eine *.libraryprofile-Datei. Wenn der Import bereits vorhandene Platzhalter-Einträge beinhaltet, erscheint eine Abfrage, ob IndraLogic diese überschrieben soll.
Exportieren	Exportiert eine xml-Datei mit Erweiterung ".libraryprofile" mit den Zuordnungen der selektierten Platzhalter-Einträgen; Sie können auch nur einen einzigen Eintrag einer Compiler-Version selektieren.



Platzhalter-Auflösungen können auch im gerade verwendeten Zielgerät definiert sein und sogar durch eine spezifische lokale Angabe im Platzhalter-Dialog im Bibliotheksverwalter.

Siehe auch

- [Kap. 18.1.20 "Objekt 'Bibliotheksinfo'" auf Seite 579](#)
- [Kap. 18.1.14 "Objekt 'Bibliotheksverwalter'" auf Seite 548](#)

Erzeugen einer Bibliothek im IndraWorks

Die Bibliotheken des SPS-Programmiersystems IndraLogic enthalten vom Steuerungshersteller bereitgestellte Funktionsbausteine, Funktionen, Datentypen, Variablenlisten und Konstanten. Die Bibliotheken werden entweder bei der Installation des "IndraWorks Engineering" direkt mit installiert und sind im Bibliotheksverwalter der jeweiligen Applikation sichtbar oder können aus dem Bibliotheks-Repository hinzugefügt werden.

Siehe auch

- [Kap. 18.1.14 "Objekt 'Bibliotheksverwalter'" auf Seite 548](#)
- [Kap. "Befehl 'Bibliotheks-Repository'" auf Seite 676](#)

Allgemeines zur Bibliothekserzeugung

Funktion: Nachstehend wird beschrieben, wie der Anwender selbst eine "Bibliothek" (Library) oder "Übersetzte Bibliothek" (Compiled-Library) aus Funktionsbausteine, Funktionen, Datentypen, usw. herstellen kann, die den Bibliotheken des Herstellers gleichgestellt ist.

Aufruf: Menü **Extras** ► **Bibliotheken**.

1. Neues Projekt erstellen

Legen Sie ein neues Projekt als "Bibliotheksprojekt" an.

Geben Sie einen passenden Projektnamen vor.

Im Projekt ist der Ordner "Allgemeiner Bausteinteil" mit Bibliotheksinfo und Projekteinstellungen enthalten.

2. Inhalt der Bibliothek strukturiert anzuordnen

Funktionsbausteine, Funktionen, Datentypen, usw. sind im Ordner "Allgemeiner Bausteinteil" entsprechend dem Anwenderbedarf strukturiert anzuordnen.

3. Auswahl der Art der Herstellung der Bibliothek

Wählen Sie eine der drei nachstehenden Möglichkeiten:

- ["Bibliothek erzeugen..." auf Seite 679](#)
- ["Übersetzte Bibliothek erzeugen..." auf Seite 680](#)
- ["Übersetzte Bibliothek erzeugen und installieren..." auf Seite 681](#)



Für alle Menüpunkte gilt, dass sie nur dann verwendet werden können, wenn das geöffnete Projekt ein Bibliotheksinfo-Objekt enthält.

Siehe auch

- [Kap. 18.1.20 "Objekt 'Bibliotheksinfo'" auf Seite 579](#)

4. Ist kein Bibliotheksinfo-Objekt vorhanden und Sie wählen einen dieser Menüpunkte, so wird die folgende Meldung ausgegeben:

Fehlermeldung, Bibliotheksinfo-Objekt (noch) nicht vorhanden:

Das Bibliotheksinfo-Objekt ist nicht vorhanden.

Fügen Sie in diesem Fall per Kontextmenü ein Bibliotheksinfo-Objekt hinzu.

5. Sie müssen in diesem Objekt mindestens die Pflichtfelder "Firma", "Titel" und "Version" ausfüllen.

Im Beispiel: "BRC", "TestLib", "1.0.0"

Ist dies nicht oder nicht korrekt erfolgt, so wird dies mit der folgenden Meldung quittiert.

Meldung: Pflichtanteile des Bibliotheksinfo-Objektes fehlen

Das Bibliotheksinfo-Objekt ist nicht vollständig ausgefüllt: erforderlich sind Firma, Titel und Version.



Mit diesen Pflichtdaten meldet sich die Bibliothek später im Bibliotheksrepository.

Generell gilt, dass der Name einer Bibliothek (Library) bzw. übersetzten Bibliothek (Compiled Library) in den Details des Bibliotheksrepositorys angezeigt wird (normalerweise sollen Titel und Name übereinstimmen):

Ab jetzt unterscheidet sich der Ablauf und das Ergebnis der drei Menüpunkte:

Bibliothek erzeugen...

Symbol:

Aufruf: Kontextmenü des Ordners "Allgemeiner Baustein" **Bibliothek erzeugen...** oder

Bei markiertem Ordner **Extras ► Bibliotheken ► Bibliothek erzeugen...**

Anlage eines Ordners im Dateisystem ihres Rechners zur Aufnahme der Bibliothek

1. Wählen Sie das Verzeichnis in dem die Bibliothek abgelegt werden soll. Dieses muss sich vom Indralogic-Verzeichnis des aktuell geöffneten Projektes unterscheiden. Sofern dies nicht der Fall ist, wird der Anwender aufgefordert ein anderes Verzeichnis zu wählen. (Sie können bei der Ablage der Bibliothek deren Namen ändern.)

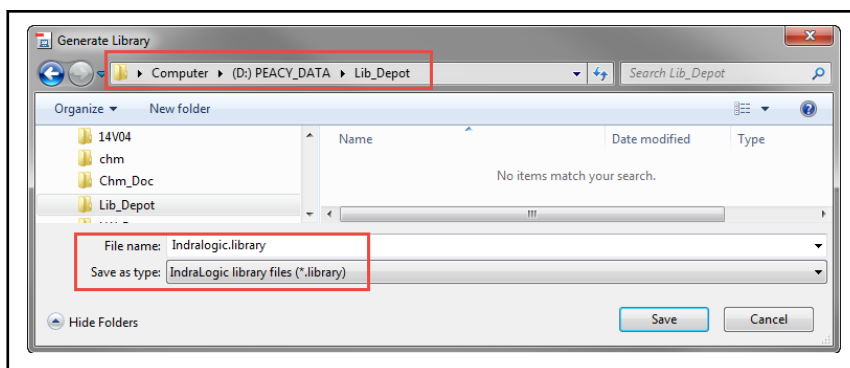


Abb. 18-63: Dialog: Wahl des Ablageortes und des Namens der zu erzeugenden Bibliothek

Nachdem Sie diesen Dialog bestätigt haben, wird zunächst das geöffnete Projekt gespeichert und anschließend in dem gewählten Verzeichnis die Bibliotheksdatei angelegt.

Pfad "D:\LibDepot" die Datei "Indralogic.library").



Der "Ablage"-Name der Bibliothek dient nur dem Wiederfinden und Verwalten außerhalb des Bibliotheksrepository. Er beeinflusst den Titel in der "Bibliotheksinfo" nicht.

Übersetzte Bibliothek erzeugen...

Symbol: 

Aufruf: Kontextmenü des Ordners "Allgemeiner Bausteinteil" **Übersetzte Bibliothek erzeugen...** oder

Bei markiertem Ordner **Extras > Bibliotheken > Übersetzte Bibliothek erzeugen...**

Anlage eines Ordners im Dateisystem ihres Rechners zur Aufnahme der Bibliothek

1. Wählen Sie das Verzeichnis in dem die Bibliothek abgelegt werden soll. Dieses muss sich vom Indralogic-Verzeichnis des aktuell geöffneten Projektes unterscheiden. Sofern dies nicht der Fall ist, wird der Anwender aufgefordert ein anderes Verzeichnis zu wählen. (Sie können bei der Ablage der Bibliothek deren Namen ändern.)

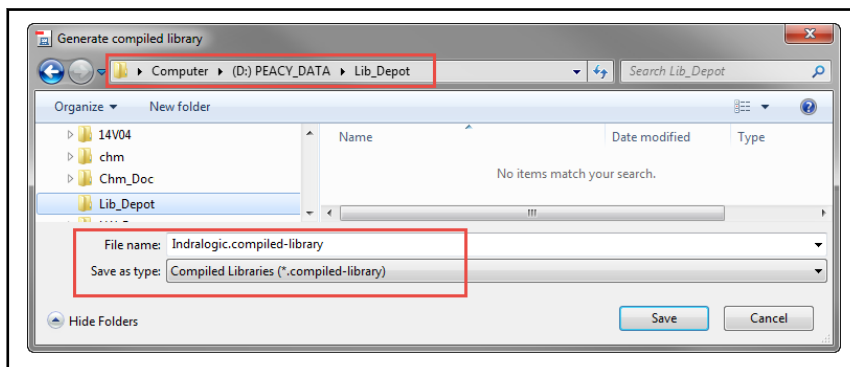


Abb. 18-64: Dialog: Wahl des Ablageortes und des Namens der zu erzeugenden "Übersetzten Bibliothek"

Nachdem Sie diesen Dialog bestätigt haben, wird zunächst das geöffnete Projekt gespeichert und anschließend in dem gewählten Verzeichnis die Bibliotheksdatei angelegt.

Pfad "D:\LibDepot" die Datei "Indralogic.compiled-library").



Der "Ablage"-Name der "Übersetzten Bibliothek" dient nur dem Wiederfinden und Verwalten außerhalb des Bibliotheksrepository. Er beeinflusst den Titel in der "Bibliotheksinfo" nicht.

Übersetzte Bibliothek erzeugen und installieren...

Symbol:

Aufruf: Kontextmenü des Ordners "Allgemeiner Bausteinteil" **Übersetzte Bibliothek erzeugen und installieren...** oder

Bei markiertem Ordner **Extras ► Bibliotheken ► Übersetzte Bibliothek erzeugen und installieren...**

Bibliothek erstellen und im Bibliotheksrepository ablegen

1. IndraLogic erzeugt die gewünschte Bibliothek und installiert sie sofort im Bibliotheks-Repository.

Die Daten werden den Pflichtfelder "Firma", "Titel" und "Version" der Bibliotheksinfo entnommen.

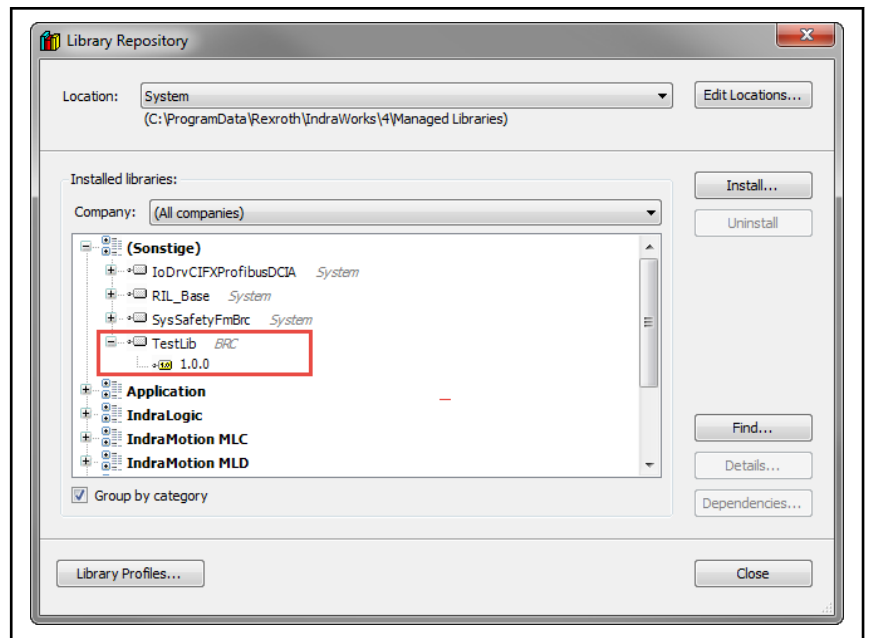


Abb. 18-65: Dialog: Bibliotheksrepository, neue Bibliothek TestLib

Befehl 'Prüfe Bibliothekskompatibilität...'

Funktion: Der Befehl startet eine Prüfung, ob das aktuell geöffnete Bibliotheksprojekt zur zuletzt installierten Version dieser Bibliothek (nächstkleinere Versionsnummer) kompatibel ist.

Aufruf: Menü **Extras ► Bibliotheken**

Voraussetzung: Ein Bibliotheksprojekt ist geöffnet. Das Bibliotheksprojekt besitzt ein Bibliotheks-Info-Objekt.

Die Prüfung achtet auf Unterschiede in den implementierten Schnittstellen (Interfaces) einer Methode. Nach der Prüfung erscheinen somit in folgenden Fällen Fehlermeldungen im Meldungsfenster:

- Hinzufügen oder Entfernen von Eingängen oder Ausgängen von Funktionsbausteinen, Funktionen oder Methoden
- Verändern des Datentyps von Eingängen oder Ausgängen
- Ändern der implementierten Schnittstellen (Interfaces) einer Methode

Siehe auch

- Anpassen
- [Kap. 16.1 "Informationen für Bibliotheksentwickler" auf Seite 236](#)

Befehl 'Anpassen'

Funktion: Der Befehl öffnet den Dialog **Anpassen**. Der Dialog "Anpassen" bietet Ihnen die Möglichkeit, Menü und Symbolleisten zu verändern. Der Dialog ist Bestandteil des Engineering-Werkzeuges IndraWorks.

Aufruf: Menü **Extras**

Befehl 'Optionen'

Funktion: Der Befehl öffnet den Dialog **Optionen** zur Konfiguration der IndraWorks-/IndraLogic-Optionen.

Diese Optionen definieren das Verhalten und Aussehen der IndraWorks-Benutzeroberfläche.

IndraWorks speichert die Einstellungen in Ihrem aktuellen Benutzerprofil auf Ihrem lokalen System. Das aktuelle Profil gibt die Standardeinstellungen vor.

Aufruf: Menü **Extras ▶ Optionen ▶ IndraLogic 2G**

Siehe auch

- [Kap. "Dialog 'Optionen- 'Allgemeine Einstellungen'" auf Seite 765](#)
- [Kap. "Dialog 'Optionen- 'CFC-Editor'" auf Seite 765](#)
- [Kap. "Dialog 'Optionen- 'Deklarationseditor'" auf Seite 766](#)
- [Kap. "Dialog 'Optionen- 'Geräteeditor'" auf Seite 767](#)
- [Kap. "Dialog 'Optionen- 'FUP, KOP und AWL'" auf Seite 768](#)
- [Kap. "Dialog 'Optionen- 'IndraLogic 1x-Konverter'" auf Seite 769](#)
- [Kap. "Dialog 'Optionen- 'Bibliotheken'" auf Seite 771](#)
- [Kap. "Dialog 'Optionen- 'Refactoring'" auf Seite 771](#)
- [Kap. "Dialog 'Optionen- 'Sichere FUP-Optionen'" auf Seite 772](#)
- [Kap. "Dialog 'Optionen- 'AS-Editor'" auf Seite 773](#)
- [Kap. "Dialog 'Optionen- 'Intelligentes Codieren'" auf Seite 775](#)
- [Kap. "Dialog 'Optionen- 'Syntaxhervorhebung'" auf Seite 776](#)
- [Kap. "Dialog 'Optionen- 'Texteditor'" auf Seite 776](#)
- [Kap. "Dialog 'Optionen- 'Visualisierung'" auf Seite 779](#)
- [Kap. "Dialog 'Optionen- 'Visualisierungsstile'" auf Seite 781](#)

18.2.7 Menü 'AS'

Befehl 'Initialschritt'

Symbol: 

Funktion: Der Befehl wandelt den gerade selektierten Schritt in einen Initialschritt um.

Aufruf: Menü **SFC**

Wenn Sie den Befehl ausführen, verändert sich der Rahmen des Schritt-Elements in eine Doppellinie. Der Schritt, der zuvor Initialschritt war, wird automatisch zu einem "normalen" Schritt und wird mit einfachem Rahmen dargestellt.

Sie können die Eigenschaft **Initialschritt** auch im Eigenschaften-Dialog eines Schritt-Elementes aktivieren oder deaktivieren, dabei passt IndraLogic jedoch die Einstellungen der anderen Schritte nicht automatisch an.

Der Befehl kann nützlich sein, wenn Sie das Diagramm umstellen wollen. Wenn Sie ein neues AS-Objekt anlegen, enthält es automatisch einen Initialschritt, gefolgt von einer Transition (TRUE) und einem Sprung zurück zum Initialschritt.



Beachten Sie die Möglichkeit, im Online-Betrieb das Diagramm mit Hilfe der AS-Flags `SFCInit` bzw. `SFCReset` auf den Initialschritt zurückzusetzen.

Siehe auch

- [Kap. "AS-Flags" auf Seite 258](#)

Befehl 'Schritt-Transition einfügen'

Symbol:

Funktion: Der Befehl fügt einen Schritt und eine Transition vor der gerade selektierten Position ein.

Aufruf: Menü **SFC**

Wenn Sie einen Schritt selektiert haben, fügt IndraLogic eine neue Schritt-Transition-Kombination ein. Wenn Sie eine Transition selektiert haben, wird eine neue Transition-Schritt-Kombination eingefügt.

Der neue Schritt wird per Default `Step<n>` genannt. `n` ist eine fortlaufende Zahl, beginnend mit 0 für den ersten Schritt, der zusätzlich zum Initialschritt eingefügt wird.

Die neue Transition wird entsprechend per Default `Trans<n>` genannt. Sie können die Default-Namen direkt durch Mausklick auf den Namen editieren.

Siehe auch

- [Kap. "Befehl 'Initialschritt'" auf Seite 682](#)
- [Kap. "Befehl 'Schritt-Transition danach einfügen'" auf Seite 683](#)
- [Kap. "AS-Elemente 'Schritt und 'Transition'" auf Seite 262](#)

Befehl 'Schritt-Transition danach einfügen'

Symbol:

Funktion: Der Befehl fügt einen Schritt und eine Transition nach der gerade angewählten Position ein.

Aufruf: Menü **SFC**

Wenn Sie einen Schritt selektiert haben, fügt IndraLogic eine neue Transition-Schritt-Kombination ein. Wenn Sie eine Transition selektiert haben, wird eine neue Schritt-Transition-Kombination eingefügt.

Der neue Schritt wird standardmäßig `Step<n>` genannt. `n` ist eine fortlaufende Zahl, beginnend mit 0 für den ersten Schritt, der zusätzlich zum Initialschritt eingefügt wird. Die neue Transition wird entsprechend per Default `Trans<n>` genannt. Sie können die Default-Namen direkt nach Mausklick auf den Namen editieren.

Siehe auch

- [Kap. "Befehl 'Schritt-Transition einfügen'" auf Seite 683](#)

Befehl 'Parallel'

Symbol: 

Funktion: Der Befehl wandelt die gerade selektierte alternative Verzweigung in eine parallele Verzweigung um.

Aufruf: Menü **SFC**

Voraussetzung: Die horizontale Verbindungslinie einer Verzweigung ist selektiert.

Beachten Sie, dass Sie nach der Umwandlung einer Verzweigung die Anordnung der Schritte und Transitionen vor und nach der Verzweigung prüfen und anpassen müssen.

Siehe auch

- [Kap. "Befehl 'Alternativ'" auf Seite 684](#)

Befehl 'Alternativ'

Symbol: 

Funktion: Der Befehl wandelt die gerade selektierte parallele Verzweigung in eine alternative Verzweigung um.

Aufruf: Menü **SFC**

Voraussetzung: Die horizontale Verbindungslinie einer Verzweigung ist selektiert.

Beachten Sie, dass Sie nach der Umwandlung einer Verzweigung die Anordnung der Schritte und Transitionen vor und nach der Verzweigung prüfen und anpassen müssen.

Siehe auch

- [Kap. "Befehl 'Parallel'" auf Seite 684](#)

Befehl 'Verzweigung einfügen'

Symbol: 

Funktion: Der Befehl fügt eine Verzweigung links von der gerade angewählten Position ein.

Aufruf: Menü **SFC**

Das Verhalten des Befehls entspricht dem Befehl **Verzweigung rechts einfügen**.

Siehe auch

- [Kap. "AS-Element 'Verzweigung'" auf Seite 267](#)
- [Kap. "Befehl 'Verzweigung rechts einfügen'" auf Seite 684](#)

Befehl 'Verzweigung rechts einfügen'

Symbol: 

Funktion: Der Befehl fügt eine Verzweigung rechts von der gerade angewählten Position ein.

Aufruf: Menü **SFC**

Der Typ der eingefügten Verzweigung hängt von dem ausgewählten Element ab:

- Wenn das oberste Element der gerade ausgewählten Elemente eine Transition oder eine alternative Verzweigung ist, fügt IndraLogic eine alternative Verzweigung ein.

- Wenn das oberste Element der gerade ausgewählten Elemente ein Schritt, ein Makro, ein Sprung oder eine parallele Verzweigung ist, fügt IndraLogic eine parallele Verzweigung mit Sprungmarke `Branch<x>` ein, wobei x eine fortlaufende Zahl ist.

Sie können diesen Default-Name der Sprungmarke editieren.

Sie können die Sprungmarke als Ziel eines Sprungs angeben.

- Wenn gerade ein gemeinsames Element einer bestehenden Verzweigung selektiert ist (horizontale Linie), fügt IndraLogic die neue Verzweigung ganz rechts als weiteren Zweig hinzu.

Wenn gerade ein gesamter Zweig einer bestehenden Verzweigung selektiert ist, fügt IndraLogic die neue Verzweigung direkt rechts davon als neuen Zweig hinzu.



Beachten Sie, dass Sie eine Verzweigung mit Befehl **Alternativ** bzw. **Parallel** in den jeweils anderen Typ umwandeln können.

Beispiel:

Parallele Verzweigung:

In der folgenden Abbildung sehen Sie eine neu eingefügte parallele Verzweigung, erzeugt mit Befehl **Verzweigung rechts einfügen** während `step11` ausgewählt war. IndraLogic fügt automatisch einen Schritt (`Step2` im Beispiel) ein.

Abarbeitung im Online-Betrieb: Wenn `t2 TRUE` liefert, führt IndraLogic `Step2` sofort nach `step11` aus, bevor `t3` ausgewertet wird.

IndraLogic führt also im Gegensatz zu alternativen Verzweigungen beide Zweige aus.

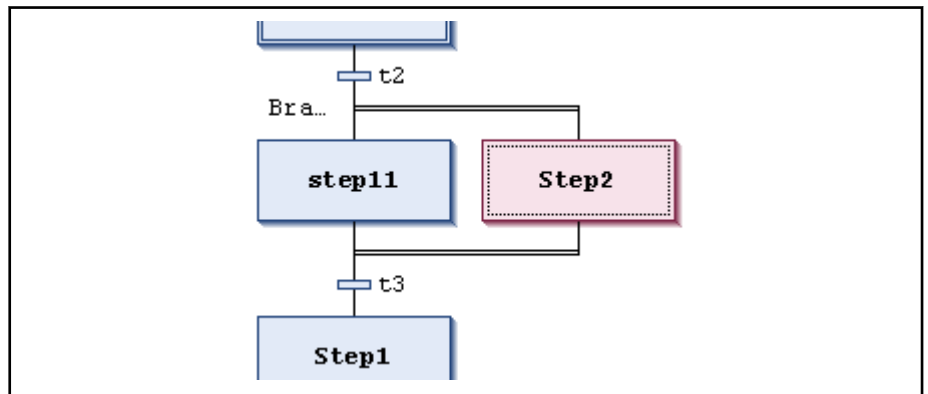


Abb. 18-66:

Beispiel:

Alternative Verzweigung

In der folgenden Abbildung sehen Sie eine neu eingefügte alternative Verzweigung, erzeugt mit Befehl **Verzweigung rechts einfügen** während Transition `t4` ausgewählt war. IndraLogic fügt automatisch einen Schritt (`Step32` im Beispiel), eine vorausgehende und eine nachfolgende Transition (`t41`, `t42`) ein.

Abarbeitung im Online-Betrieb: Wenn `Step3` aktiv ist, wertet IndraLogic die nachfolgenden Transitionen (`t4`, `t41`) von links nach rechts aus. Der erste Zweig der Verzweigung, in dem die erste Transition `TRUE` liefert, wird ausge-

führt. Somit wird im Gegensatz zu einer parallelen Verzweigung nur ein Zweig ausgeführt.

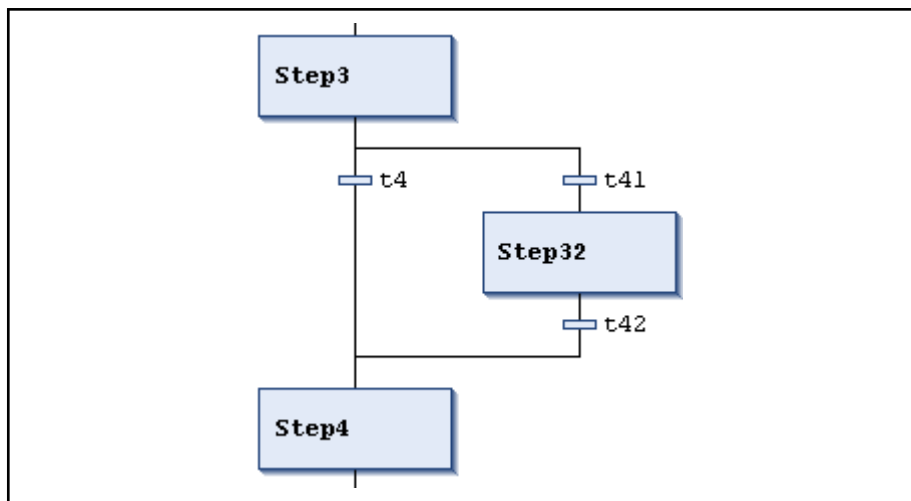


Abb. 18-67:

Siehe auch

- [Kap. "AS-Element 'Verzweigung'" auf Seite 267](#)
- [Kap. "Befehl 'Verzweigung einfügen'" auf Seite 684](#)
- [Kap. "Befehl 'Parallel'" auf Seite 684](#)
- [Kap. "Befehl 'Alternativ'" auf Seite 684](#)

Befehl 'Aktionsassoziation einfügen'

Symbol:

Funktion: Der Befehl weist einem Schritt eine IEC-Aktion zu.

Aufruf: Menü **SFC**

Voraussetzung: Es ist ein Schritt selektiert.

IndraLogic fügt das Aktion-Element rechts neben dem gerade selektierten Schritt-Element ein.

Wenn Sie dem Schritt bereits eine oder mehrere Aktionen zugewiesen haben, werden diese in einer "Aktionenliste" untereinander dargestellt.

Die neue Aktion wird dann folgendermaßen platziert:

- Wenn Sie das Schritt-Element selektiert haben, als erste Aktion des Schrittes, d.h. an oberster Position der Aktionenliste.
- Wenn Sie eine der vorhandenen Aktionen in der Aktionenliste des Schrittes selektiert haben, direkt vor, d.h. oberhalb der Aktion.

Der linke Teil des Aktion-Elements enthält den Qualifizierer, standardmäßig **N**, im rechten Teil geben Sie den Aktionsnamen ein. Dazu klicken Sie in die Box, um einen Editerrahmen zu erhalten. Sie müssen diese Aktion vorher im Projekt als POU angelegt haben.

Sie können den Qualifizierer ebenfalls editieren. Eine Liste der gültigen Qualifizierer ist unter "Qualifizierer für Aktionen in AS" beschrieben.

Siehe auch

- [Kap. "Befehl 'Aktionsassoziation danach einfügen'" auf Seite 687](#)
- [Kap. "Qualifizierer für Aktionen in AS" auf Seite 256](#)

Befehl 'Aktionsassoziation danach einfügen'

Symbol: 

Funktion: Der Befehl weist einem Schritt eine IEC-Aktion zu.

Aufruf: Menü **SFC**

Voraussetzung: Es ist ein Schritt selektiert.

Der Befehl entspricht der Beschreibung von **Aktionsassoziation einfügen**. Der Unterschied besteht darin, dass IndraLogic die neue Aktion nicht an der ersten, sondern an der letzten Position der Aktionenliste platziert.

Wenn Sie in der Aktionsliste eine Aktion selektiert haben, platziert IndraLogic die neue Aktion nicht oberhalb, sondern unterhalb.

Siehe auch

- [Kap. "Befehl 'Aktionsassoziation einfügen'" auf Seite 686](#)
- [Kap. "Qualifizierer für Aktionen in AS" auf Seite 256](#)

Befehl 'Sprung einfügen'

Symbol: 

Funktion: Der Befehl fügt ein Sprung-Element vor dem gerade selektierten Element ein.

Aufruf: Menü **SFC**

Voraussetzung: Es ist ein Schritt selektiert.

IndraLogic fügt den Sprung automatisch mit Sprungziel `step` ein. Sie müssen dieses Sprungziel anschließend noch durch ein reales Sprungziel ersetzen. Sie können das Ziel mit der Eingabehilfe auswählen.

Siehe auch

- [Kap. "AS-Element 'Sprung'" auf Seite 269](#)
- [Kap. "Befehl 'Sprung danach einfügen'" auf Seite 687](#)

Befehl 'Sprung danach einfügen'

Symbol: 

Funktion: Der Befehl fügt ein Sprung-Element nach dem gerade selektierten Element ein.

Aufruf: Menü **SFC**

IndraLogic fügt den Sprung automatisch mit Sprungziel `step` ein. Sie müssen dieses Sprungziel anschließend noch durch ein reales Sprungziel ersetzen. Sie können das Ziel mit der Eingabehilfe auswählen.

Siehe auch

- [Kap. "AS-Element 'Sprung'" auf Seite 269](#)
- [Kap. "Befehl 'Sprung einfügen'" auf Seite 687](#)

Befehl 'Makro einfügen'

Symbol: 

Funktion: Der Befehl fügt ein Makro-Element vor dem gerade selektierten Element ein.

Aufruf: Menü **SFC**

Das neue Makro wird standardmäßig `Macro<x>` genannt. `x` ist eine fortlaufende Zahl, beginnend mit 0 für das erste Makro. Sie können den Default-Name direkt durch Mausklick auf den Namen editieren.

Um das Makro zu bearbeiten öffnen Sie es mit dem Befehl **Makro anzeigen** im Makro-Editor.

Siehe auch

- [Kap. "Befehl 'Makro danach einfügen'" auf Seite 688](#)
- [Kap. "Befehl 'Makro anzeigen'" auf Seite 688](#)

Befehl 'Makro danach einfügen'

Symbol: 

Funktion: Der Befehl fügt ein Makro-Element nach dem gerade selektierten Element ein.

Aufruf: Menü **SFC**

Der Befehl entspricht der Beschreibung von **Makro einfügen**.

Siehe auch

- [Kap. "Befehl 'Makro einfügen'" auf Seite 687](#)
- [Kap. "Befehl 'Makro anzeigen'" auf Seite 688](#)

Befehl 'Makro anzeigen'

Symbol: 

Funktion: Der Befehl öffnet ein Makro im Makro-Editor zur Bearbeitung.

Aufruf: Menü **SFC**

Voraussetzung: Es ist ein Makro selektiert.

Mit dem Befehl schließt IndraLogic die Hauptansicht des AS-Editors und öffnet statt dessen den Makro-Editor. Es handelt sich ebenfalls um einen AS-Editor, in dem Sie nun den Teil des AS-Diagramms bearbeiten können, der in der Hauptansicht als Makro-Box dargestellt wird.

Mit dem Befehl **Makro verlassen** kehren Sie zur Hauptansicht zurück.

Siehe auch

- [Kap. "Befehl 'Makro verlassen'" auf Seite 688](#)

Befehl 'Makro verlassen'

Symbol: 

Funktion: Der Befehl schließt den Makro-Editor und kehrt zur Hauptansicht des AS-Editors zurück.

Aufruf: Menü **SFC**

Voraussetzung: Es ist ein Makro im Macro-Editor geöffnet.

Siehe auch

- [Kap. "Befehl 'Makro anzeigen'" auf Seite 688](#)

Befehl 'Einfügen danach'

Symbol: 

Funktion: Der Befehl fügt die Elemente aus der Zwischenablage nach der gerade angewählten Position ein.

Aufruf: Menü **AS**

18.2.8 Menü 'CFC'

Befehl 'Arbeitsblatt bearbeiten'

Funktion: Der Befehl öffnet den Dialog **Arbeitsblatt bearbeiten** in dem Sie die Größe des Arbeitsblattes festlegen.

Aufruf: Menü **CFC**

Voraussetzungen: Ein CFC-Editor ist aktiv.

Dialog 'Arbeitsblatt bearbeiten'

Folgende Dimensionen verwenden	Hier stellen Sie die Größe des Arbeitsblatts ein. Ihre Änderung wird nur übernommen, wenn die Größe für das bestehende Programm ausreichend ist.
Dimensionen automatisch anpassen	Passt die Größe des Arbeitsblatts automatisch an die Größe Ihres Programms an.
Arbeitsblatt Ursprung relativ verschieben	Verschiebt das Arbeitsblatt auf der x- oder y-Achse. Die Eingabe negativer Zahlen ist erlaubt.

Befehl 'Seitengröße bearbeiten'

Funktion: Der Befehl öffnet den Dialog **Seitengröße bearbeiten**. Darin verändern Sie die Größe des seitenorientierten CFC-Editors..

Aufruf: Menü **CFC**

Voraussetzungen: Ein seitenorientierter CFC-Editor ist aktiv.

Dialog 'Seitengröße bearbeiten'

Breite	Breite der Seite (Minimum 24, Maximum 1024). Elemente außerhalb des Arbeitsbereichs werden rot markiert.
Höhe	Höhe der Seite (Minimum 24, Maximum 1024). Elemente außerhalb des Arbeitsbereichs werden rot markiert.
Randbreite	Breite der Randspalte (Minimum 6, Maximum 25% oder Seitenbreite).
Als Standard für neue CFC-Objekte setzen	 Die aktuellen Einstellungen werden als Standard für neue CFC-Objekte gewählt.

Siehe auch

- [Kap. "CFC-Element 'Seite'" auf Seite 297](#)

Befehl 'Negieren'

Symbol: 

Funktion: Der Befehl negiert den ausgewählten Bausteineingang oder Bausteinausgang.

Aufruf: Menü **CFC**, Kontextmenü

Voraussetzungen: Ein CFC-Editor ist aktiv. Ein Bausteineingang oder Bausteinausgang ist selektiert.

Befehl 'Befehl 'EN/ENO''

Symbol: 

Funktion: Der Befehl fügt dem selektierten Baustein einen booleschen Eingang **EN** (Enable) und einen booleschen Ausgang **ENO** (Enable Out) hinzu.

Aufruf: Menü **CFC**, Kontextmenü

Voraussetzungen: Ein CFC-Editor ist aktiv. Ein Baustein ist selektiert.

Der hinzugefügte Eingang **EN** aktiviert den Baustein. Nur wenn er den Wert **TRUE** hat, wird der Baustein ausgeführt. Der Wert dieses Signals wird am Ausgang **ENO** ausgegeben.

Befehl 'Kein'

Symbol: 

Funktion: Der Befehl entfernt einen Reset (R), Set (S) oder REF (Referenz-zuweisung) vom Eingang des Elements **Ausgang**.

Aufruf: Menü **CFC** ► **Einstellen/Zurücksetzen**, Kontextmenü **Einstellen/Zurücksetzen**

Voraussetzungen: Ein CFC-Editor ist aktiv. Der Eingang eines Elements **Ausgang** ist selektiert.

Befehl 'R-Reset'

Symbol: 

Funktion: Der Befehl fügt dem Eingang eines booleschen Elements **Ausgang** einen Reset hinzu.

Aufruf: Menü **CFC** ► **Einstellen/Zurücksetzen**, Kontextmenü **Einstellen/Zurücksetzen**

Voraussetzungen: Ein CFC-Editor ist aktiv. Der Eingang eines Elements **Ausgang** ist selektiert.

Wenn ein Element **Ausgang** einen Reset-Eingang hat, wird der boolesche Ausgangswert auf **FALSE** gesetzt, sobald der Wert des Eingangs **TRUE** ist. Der Wert **FALSE** am Ausgang bleibt erhalten, auch wenn sich der Eingangswert wieder ändert.

Siehe auch

- [Kap. "Befehl 'S-SET'" auf Seite 690](#)
- [Kap. "Befehl 'Kein'" auf Seite 690](#)

Befehl 'S-SET'

Symbol: 

Funktion: Der Befehl fügt dem Eingang eines booleschen Elements **Ausgang** einen Set (S) hinzu.

Aufruf: Menü **CFC** ► **Einstellen/Zurücksetzen**, Kontextmenü **Einstellen/Zurücksetzen**

Voraussetzungen: Ein CFC-Editor ist aktiv. Der Eingang eines Elements **Ausgang** ist selektiert.

Wenn ein Element **Ausgang** einen Set-Eingang hat, wird der boolesche Ausgangswert auf **TRUE** gesetzt, sobald der Wert des Eingangs **TRUE** ist. Der Wert **TRUE** am Ausgang bleibt erhalten, auch wenn sich der Eingangswert wieder ändert.

Siehe auch

- [Kap. "Befehl 'R-Reset'" auf Seite 690](#)
- [Kap. "Befehl 'Kein'" auf Seite 690](#)

Befehl 'Ref= (Referenzzuweisung)'

Symbol: 

Funktion: Der Befehl weist einem Element **Ausgang** eine Referenz zu.

Aufruf: Menü **CFC** ► **Einstellen/Zurücksetzen**, Kontextmenü **Einstellen/Zurücksetzen**

Voraussetzungen: Ein CFC-Editor ist aktiv. Der Eingang eines Elements **Ausgang** ist selektiert.

Beispiel:

Deklaration:

```
ref_int : REFERENCE TO INT;
a : INT;
```

CFC:

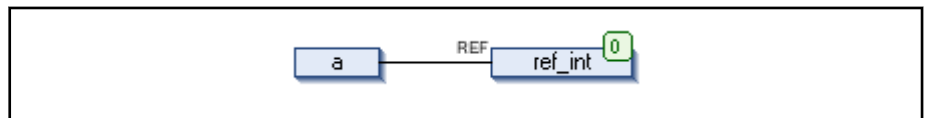


Abb. 18-68:

Dies entspricht dem ST-Code `ref_int REF= a;`

Weitere Informationen finden Sie in der Beschreibung zum Datentyps `REFERENCE TO`.

Siehe auch

- [Kap. "Im CFC-Editor programmieren" auf Seite 100](#)
- [Kap. 17.5.13 "Datentyp 'REFERENCE'" auf Seite 384](#)
- [Kap. "Befehl 'Kein'" auf Seite 690](#)

Befehl 'An den Anfang'

Symbol: 

Funktion: Der Befehl setzt die selektierten Elemente an den Beginn der Ausführungsreihenfolge.

Aufruf: Menü **CFC** ► **Ausführungsreihenfolge**, Kontextmenü **Ausführungsreihenfolge**

Voraussetzungen: Ein CFC-Editor ist aktiv und die Eigenschaft „Expliziter Ausführungsreihenfolge-Modus“ ist aktiviert. Mindestens ein Element ist selektiert.

Die selektierten Elemente erhalten die niedrigsten Nummern beginnend mit 0 unter Beibehaltung der bisherigen Reihenfolge. Die restlichen Elemente werden so nummeriert, dass ihre Ausführungsreihenfolge auch bestehen bleibt. Die topologischen Positionen der Elemente bleiben sowieso erhalten.

Siehe auch

- [Kap. "Dialog 'Eigenschaften- Registerkarte 'CFC-Ausführungsreihenfolge'" auf Seite 753](#)
- [Kap. "Befehl 'Ans Ende'" auf Seite 692](#)
- [Kap. "Befehl 'Eins vor'" auf Seite 692](#)
- [Kap. "Befehl 'Nach Datenfluss anordnen'" auf Seite 693](#)
- [Kap. "Befehl 'Topologisch anordnen'" auf Seite 694](#)

Befehl 'Ans Ende'

Symbol: 

Funktion: Der Befehl nummeriert die Elemente so, dass die selektierten Elemente am Ende der Ausführungsreihenfolge stehen.

Aufruf: Menü **CFC ▶ Ausführungsreihenfolge**, alternativ im Editor-Kontextmenü **Ausführungsreihenfolge**

Voraussetzungen: Ein CFC-Editor ist aktiv und die Eigenschaft "Expliziter Ausführungsreihenfolge-Modus" ist aktiviert. Mindestens ein Element ist selektiert.

Die selektierten Elemente erhalten die höchsten Nummern unter Beibehaltung der bisherigen Reihenfolge. Die restlichen Elemente werden so nummeriert, dass ihre Ausführungsreihenfolge auch bestehen bleibt. Die topologischen Positionen der Elemente bleiben sowieso erhalten.

Siehe auch

- [Kap. "Dialog 'Eigenschaften- Registerkarte 'CFC-Ausführungsreihenfolge'" auf Seite 753](#)
- [Kap. "Befehl 'An den Anfang'" auf Seite 691](#)
- [Kap. "Befehl 'Eins vor'" auf Seite 692](#)
- [Kap. "Befehl 'Eins zurück'" auf Seite 693](#)
- [Kap. "Befehl 'Nach Datenfluss anordnen'" auf Seite 693](#)
- [Kap. "Befehl 'Topologisch anordnen'" auf Seite 694](#)

Befehl 'Eins vor'

Symbol: 

Funktion: Der Befehl nummeriert die Elemente so, dass die selektierten Elemente um eins weiter vorne stehen.

Aufruf: Menü **CFC ▶ Ausführungsreihenfolge**, alternativ im Editor-Kontextmenü **Ausführungsreihenfolge**

Voraussetzungen: Ein CFC-Editor ist aktiv und mindestens ein Element ist selektiert. Die Eigenschaft "Expliziter Ausführungsreihenfolge-Modus" ist aktiviert.

Die selektierten Elemente erhalten eine um eins niedrigere Nummerierung unter Beibehaltung der bisherigen Reihenfolge. Die selektierten Elemente werden um eins früher abgearbeitet. Die restlichen Elemente werden so nummeriert, dass ihre Ausführungsreihenfolge auch erhalten bleibt. Die topologischen Positionen der Elemente bleiben sowieso erhalten.

Siehe auch

- [Kap. "Dialog 'Eigenschaften- Registerkarte 'CFC-Ausführungsreihenfolge'" auf Seite 753](#)
- [Kap. "Befehl 'An den Anfang'" auf Seite 691](#)
- [Kap. "Befehl 'Ans Ende'" auf Seite 692](#)
- [Kap. "Befehl 'Eins zurück'" auf Seite 693](#)
- [Kap. "Befehl 'Nach Datenfluss anordnen'" auf Seite 693](#)
- [Kap. "Befehl 'Topologisch anordnen'" auf Seite 694](#)

Befehl 'Eins zurück'

Symbol: 

Funktion: Der Befehl nummeriert die Elemente so, dass die selektierten Elemente um eins weiter hinten stehen.

Aufruf: Menü **CFC ▶ Ausführungsreihenfolge**, alternativ im Editor-Kontextmenü **Ausführungsreihenfolge**

Voraussetzungen: Ein CFC-Editor ist aktiv und mindestens ein Element ist selektiert. Die Eigenschaft "Expliziter Ausführungsreihenfolge-Modus" ist aktiviert.

Die selektierten Elemente erhalten eine um eins höhere Nummerierung unter Beibehaltung der bisherigen Reihenfolge. Die Elemente werden um eins später abgearbeitet. Die restlichen Elemente werden so nummeriert, dass ihre Ausführungsreihenfolge auch erhalten bleibt. Die topologischen Positionen der Elemente bleiben sowieso erhalten.

Siehe auch

- [Kap. "Dialog 'Eigenschaften- Registerkarte 'CFC-Ausführungsreihenfolge'" auf Seite 753](#)
- [Kap. "Befehl 'An den Anfang'" auf Seite 691](#)
- [Kap. "Befehl 'Eins vor'" auf Seite 692](#)
- [Kap. "Befehl 'Ans Ende'" auf Seite 692](#)
- [Kap. "Befehl 'Nach Datenfluss anordnen'" auf Seite 693](#)
- [Kap. "Befehl 'Topologisch anordnen'" auf Seite 694](#)

Befehl 'Nach Datenfluss anordnen'

Funktion: Der Befehl nummeriert die Elemente im Programm nach Datenfluss und bei mehreren Netzwerken nach ihrer topologischer Position im Editor.

Aufruf: Menü **CFC ▶ Nach Datenfluss anordnen**, alternativ im Editor-Kontextmenü **Nach Datenfluss anordnen**

Voraussetzungen: Ein CFC-Editor ist aktiv und die Eigenschaft "Expliziter Ausführungsreihenfolge-Modus" ist aktiviert.

Die Ausführungsreihenfolge wird nach dem Datenfluss und (bei mehreren Netzwerken) nach der topologischen Position der Netzwerke angeordnet. Alle nummerierten Elemente des Programmierbausteins werden entsprechend gesetzt. Die Ausführungsreihenfolge ist danach identisch zu der im automatischen Datenfluss-Modus. Die topologischen Positionen der Elemente bleiben sowieso erhalten.

Siehe auch

- [Kap. "Dialog 'Eigenschaften- Registerkarte 'CFC-Ausführungsreihenfolge'" auf Seite 753](#)
- [Kap. "Befehl 'An den Anfang'" auf Seite 691](#)
- [Kap. "Befehl 'Ans Ende'" auf Seite 692](#)
- [Kap. "Befehl 'Eins vor'" auf Seite 692](#)
- [Kap. "Befehl 'Eins zurück'" auf Seite 693](#)
- [Kap. "Befehl 'Nach Datenfluss anordnen'" auf Seite 693](#)
- [Kap. "Befehl 'Topologisch anordnen'" auf Seite 694](#)

Befehl 'Topologisch anordnen'

Funktion: Der Befehl ordnet die Ausführungsreihenfolge der Elemente nach ihrer topologischen Position von rechts nach links und oben nach unten an.

Aufruf: Menü **CFC** ► **Topologisch anordnen**, alternativ im Editor-Kontextmenü **Topologisch anordnen**

Voraussetzungen: Ein CFC-Editor ist aktiv und die Eigenschaft "Expliziter Ausführungsreihenfolge-Modus" ist aktiviert. Mindestens ein Element ist selektiert.

Der Befehl wirkt sich auf alle Elemente im Programm, auch wenn nicht alle Elemente beim Ausführen des Befehls selektiert sind. Die topologischen Positionen der Elemente bleiben sowieso erhalten.

Siehe auch

- [Kap. "Dialog 'Eigenschaften- Registerkarte 'CFC-Ausführungsreihenfolge'" auf Seite 753](#)
- [Kap. "Befehl 'An den Anfang'" auf Seite 691](#)
- [Kap. "Befehl 'Ans Ende'" auf Seite 692](#)
- [Kap. "Befehl 'Eins vor'" auf Seite 692](#)
- [Kap. "Befehl 'Eins zurück'" auf Seite 693](#)
- [Kap. "Befehl 'Nach Datenfluss anordnen'" auf Seite 693](#)
- [Kap. "Befehl 'Topologisch anordnen'" auf Seite 694](#)

Befehl 'Ausführungsreihenfolge setzen'

Funktion: Der Befehl öffnet einen Dialog zum Setzen der Ausführungsposition eines Elementes auf einen beliebigen Wert.

Aufruf: Menü **CFC** ► **Ausführungsreihenfolge setzen**, Kontextmenü **Ausführungsreihenfolge setzen**

Voraussetzungen: Ein CFC-Editor ist aktiv. Ein Element ist selektiert.

Mit dem Befehl ändern Sie nur die chronologische Reihenfolge im Programmablauf. Die Position der Elemente wird nicht geändert. Wenn der Befehl auf mehrere Elemente gleichzeitig angewandt wird, bleibt die Reihenfolge innerhalb der selektierten Elemente erhalten.

Siehe auch

- [Kap. "Befehl 'Nach Datenfluss anordnen'" auf Seite 693](#)
- [Kap. "Befehl 'Topologisch anordnen'" auf Seite 694](#)

Befehl 'Parameter bearbeiten'

Funktion: Der Befehl öffnet den Dialog **Parameter bearbeiten**, in dem Sie die konstanten Eingangsparameter eines Funktionsbausteins verändern.

Aufruf: Menü **CFC** ► **Parameter bearbeiten**, Kontextmenü ► **Parameter bearbeiten**, Klick auf Funktionsbaustein **Parameter**.

Voraussetzungen: Ein CFC-Editor ist aktiv. Es ist ein Funktionsbaustein instanziiert, der in seiner Deklaration VAR_INPUT CONSTANT-Variablen hat.



Diese Funktionalität wirkt nur auf Bausteine, die mit IndraLogic >= V3.5 SP4 in einem CFC eingefügt werden!

Ein Baustein mit VAR_INPUT CONSTANT -Variablen wird von IndraLogic durch das Wort **Parameter** in der linken unteren Ecke des Bausteins angezeigt.

Dialog 'Parameter bearbeiten'

Parameter	Name der Variablen
Datentyp	Datentyp der Variablen
Wert	Klicken Sie in das Feld um einen Wert einzugeben.
Initialwert	Initialisierungswert
Kategorie	<i>Zusätzliche Informationen zu den Parametern. Diese Werte werden über Attribute definiert und können in diesem Dialog nicht verändert werden</i> - parameterCategory - parameterUnit - parameterMinValue - parameterMaxValue
Einheit	
Min	
Max	
Vorbereitete Parameter löschen	Der Befehl ist aktiv, wenn Sie einen vorbereiteten Wert geschrieben haben (Befehl Debuggen ► Werte schreiben)

Nach Verlassen des Felds und Verlassen des Dialogs mit **OK** werden die Werteänderungen im Projekt angewendet.

Beispiel eines Bausteins mit konstanten Eingängen

```
FUNCTION_BLOCK FB1
VAR_INPUT CONSTANT
{attribute 'parameterCategory':='General'}
{attribute 'parameterUnit':='m/s'}
{attribute 'parameterMinValue':='0'}
{attribute 'parameterMaxValue':='100'}
fbin1:INT;
fbin2:DWORD:=24354333;
fbin3:STRING:='abc';
END_VAR
```



Diese Funktionalität und die Deklaration von Variablen mit Schlüsselwort `VAR_INPUT CONSTANT` gilt nur für den CFC-Editor. Im FUP-Editor zeigt IndraLogic immer alle Eingangsparameter am Baustein an, egal ob sie als `VAR_INPUT` oder `VAR_INPUT CONSTANT` deklariert sind. Auch für Texteditoren unterscheidet IndraLogic diesbezüglich nicht.

Siehe auch

- ["Konstante Eingangsparameter von Funktionsbaustein-Instanzen ändern" auf Seite 294](#)
- [Kap. "Befehl 'Vorbereitete Parameter im Projekt speichern'" auf Seite 695](#)

Befehl 'Vorbereitete Parameter im Projekt speichern'

Funktion: Der Befehl übernimmt die vorbereiteten Parameterwerte in das Projekt.

Aufruf: Menü **CFC**

Voraussetzungen: Ein CFC-Editor ist aktiv. Parameterwerte von Funktionsbausteininstanzen wurden im Online-Betrieb geändert. Sie sind im Offline-Betrieb.

Wenn sich die Werte von Konstanten auf der Steuerung von den Werten in der Applikation unterscheiden, wird dies durch einen roten Stern rechts neben dem Parameterfeld gekennzeichnet. Durch den Befehl **Vorbereitete Pa-**

Parameterwerte übernehmen übernehmen Sie die Werte der Steuerung in Ihre Applikation.

Siehe auch

- ["Konstante Eingangsparameter von Funktionsbaustein-Instanzen ändern" auf Seite 294](#)
- [Kap. "Befehl 'Parameter bearbeiten'" auf Seite 694](#)

Befehl 'Selektierte Anschlüsse verbinden'

Symbol: 

Funktion: Der Befehl stellt eine Verbindung zwischen den selektierten Anschlüssen her.

Aufruf: Menü **CFC ▶ Selektierte Anschlüsse verbinden**, Kontextmenü **Selektierte Anschlüsse verbinden**

Voraussetzungen: Ein CFC-Editor ist aktiv. Genau ein Ausgang und mehrere Eingänge sind selektiert.

Zum Selektieren der Anschlüsse müssen Sie die <STRG->-Taste gedrückt halten, während Sie die Anschlüsse anklicken. Danach führen Sie den Befehl aus.

Siehe auch

- [Kap. "Befehl 'Verbundene Anschlüsse selektieren'" auf Seite 697](#)

Befehl 'Verbindung lösen'

Symbol: 

Funktion: Dieser Befehl löst eine gesperrte Verbindung.

Aufruf: **CFC ▶ Verbindung lösen**, Kontextmenü **Verbindung lösen**

Voraussetzungen: Ein CFC-Editor ist aktiv. Eine Verbindung oder eine Verbindungsmarke ist selektiert.

Ein gesperrte Verbindung erhalten Sie, wenn Sie die Verbindungen des automatischen Routings verändern. Wenn Sie erneut ein automatisches Routing durchführen wollen, müssen Sie vorher eine gesperrte Verbindung lösen.



Mit einem Mausklick auf das Icon  einer gesperrten Verbindung können Sie diese Verbindung ebenfalls lösen.

Siehe auch

- [Kap. "Im CFC-Editor programmieren" auf Seite 100](#)
- [Kap. "CFC-Element 'Verbindungsmarke-Quelle/Ziel'" auf Seite 299](#)

Befehl 'Nächste Kollision zeigen'

Funktion: Der Befehl zeigt die nächsten Kollision im Editor auf und markiert die betroffene Stelle.

Aufruf: Menü **CFC ▶ Routing**, Kontextmenü **Routing**

Voraussetzungen: Ein CFC-Editor ist aktiv und es ist mindestens eine Verbindung mit Kollision vorhanden.

Diese Funktion ist sehr nützlich, wenn Sie mit großen Netzwerken arbeiten und nur einen Teilbereich sehen. Eine Kollision wird Ihnen zusätzlich durch das rot umrandete Symbol in der rechten oberen Ecke des Editors signalisiert.

Befehl 'Verbundene Anschlüsse selektieren'

Symbol: 

Funktion: Der Befehl selektiert alle Anschlüsse, die mit der gerade selektierten Leitung, oder im seitenorientierten CFC mit der gerade selektierten Verbindungsmarke verbunden sind.

Aufruf: Menü **CFC ▶ Verbundene Anschlüsse selektieren**, Kontextmenü **Verbundene Anschlüsse selektieren**

Voraussetzungen: Ein CFC-Editor, oder ein seitenorientierter CFC-Editor ist aktiv. Eine Leitung, oder eine Verbindungsmarke ist selektiert.

Siehe auch

- [Kap. "Befehl 'Selektierte Anschlüsse verbinden'" auf Seite 696](#)

Befehl 'Anschlüsse zurücksetzen'

Symbol: 

Funktion: Der Befehl stellt gelöschte Anschlüsse eines Bausteins wieder her.

Aufruf: Menü **CFC ▶ Anschlüsse zurücksetzen**, Kontextmenü **Anschlüsse zurücksetzen**

Voraussetzungen: Ein CFC-Editor ist aktiv und es ist ein Baustein selektiert.

Der Befehl stellt alle Ein- und Ausgänge des Bausteins wieder her, so wie sie in seiner Implementierung definiert sind.

Siehe auch

- [Kap. "Befehl 'Nicht verbundene Anschlüsse entfernen'" auf Seite 697](#)

Befehl 'Nicht verbundene Anschlüsse entfernen'

Symbol: 

Funktion: Der Befehl entfernt alle ungenutzten Anschlüsse des selektierten Elements.

Aufruf: Menü **CFC ▶ Nicht verbundene Anschlüsse entfernen**, Kontextmenü **Nicht verbundene Anschlüsse entfernen**

Voraussetzungen: Ein CFC-Editor ist aktiv. Ein Element ist selektiert.

Siehe auch

- [Kap. "Befehl 'Anschlüsse zurücksetzen'" auf Seite 697](#)

Befehl 'Eingangsanschluss hinzufügen'

Symbol: 

Funktion: Der Befehl fügt dem selektierten Baustein einen weiteren Eingang hinzu.

Aufruf: Konfigurierbar über **Extras ▶ Anpassen ▶ Befehle ▶ CFC**

Voraussetzungen: Ein CFC-Editor ist aktiv. Ein Baustein ist selektiert.

Siehe auch

- [Kap. "Befehl 'Ausgangsanschluss hinzufügen'" auf Seite 697](#)

Befehl 'Ausgangsanschluss hinzufügen'

Symbol: 

Funktion: Der Befehl fügt dem selektierten Baustein einen weiteren Ausgang hinzu.

Aufruf: Konfigurierbar über **Extras ▶ Anpassen ▶ Befehle ▶ CFC**

Voraussetzungen: Ein CFC-Editor ist aktiv. Ein geeigneter Baustein ist selektiert.

Siehe auch

- [Kap. "Befehl 'Eingangsanschluss hinzufügen'" auf Seite 697](#)

Befehl 'Alle Verbindungen routen'

Symbol: 

Funktion: Der Befehl macht alle manuellen Änderungen an den Verbindungen im Programm rückgängig und stellt den ursprünglichen Zustand wieder her.

Aufruf: Menü **CFC ▶ Alle Verbindungen routen**, Kontextmenü **Alle Verbindungen routen**

Voraussetzungen: Ein CFC-Editor ist aktiv.

IndraLogic kann Verbindungen, die mit Kontrollpunkten fixiert sind, nicht automatisch routen. Sie müssen die Kontrollpunkte vor dem Ausführen des Befehls entfernen. Dazu verwenden Sie den Befehl **Kontrollpunkt entfernen**. Weiterhin müssen Sie manuell veränderte Verbindungen, die mit dem Icon , gekennzeichnet sind, lösen. Dazu verwenden Sie den Befehl **Verbindung lösen**.

Siehe auch

- [Kap. "Befehl 'Kontrollpunkt entfernen'" auf Seite 698](#)
- [Kap. "Befehl 'Verbindung lösen'" auf Seite 696](#)

Befehl 'Kontrollpunkt entfernen'

Funktion: Der Befehl entfernt einen Kontrollpunkt.

Aufruf: **CFC ▶ Kontrollpunkt entfernen** oder Kontextmenü **Kontrollpunkt entfernen**

Voraussetzungen: Ein CFC-Editor ist aktiv. Sie haben eine Verbindungslinie selektiert.

Wenn Sie den Mauszeiger über eine selektierte Verbindungslinie bewegen, werden die vorhandenen Kontrollpunkte mit gelben Kreissymbolen angezeigt. Setzen Sie den Cursor auf den zu löschenden Kontrollpunkt und führen Sie den Befehl aus dem Kontextmenü aus.

Siehe auch

- [Kap. "CFC-Element 'Kontrollpunkt'" auf Seite 297](#)
- [Kap. "Befehl 'Kontrollpunkt erzeugen'" auf Seite 698](#)

Befehl 'Kontrollpunkt erzeugen'

Symbol: 

Funktion: Der Befehl erzeugt einen Kontrollpunkt an einem Verbinder.

Aufruf: **CFC ▶ Kontrollpunkt erzeugen** oder Kontextmenü **Kontrollpunkt erzeugen**

Voraussetzungen: Ein CFC-Editor ist aktiv. Der Cursor befindet sich über einer Verbindung.

Der Kontrollpunkt wird an der Stelle der Verbindung erstellt, an dem sich der Cursor bei Aufruf des Befehls befindet. Der Befehl entspricht dem Element **Kontrollpunkt** im Fenster **Werkzeuge**.

Siehe auch

- [Kap. "CFC-Element 'Kontrollpunkt'" auf Seite 297](#)
- [Kap. "Befehl 'Kontrollpunkt entfernen'" auf Seite 698](#)

Befehl 'Verbindungsmarke'

Symbol: 

Funktion: Dieser Befehl schaltet die Darstellung der Verbindung zwischen zwei Elementen zwischen einer Verbindungslinie und der Verwendung von Verbindungsmarken hin und her.

Aufruf: CFC ► **Verbindungsmarke**, Kontextmenü **Verbindungsmarke**

Voraussetzungen: Ein CFC-Editor ist aktiv. Eine Verbindung oder eine Verbindungsmarke ist selektiert.

Wenn Sie eine Verbindungslinie selektiert haben, entfernt der Befehl diese Linie und fügt eine **Verbindungsmarke - Quelle** am Ausgang des einen Elements hinzu und eine **Verbindungsmarke - Ziel** am Eingang des anderen. Beide bekommen standardmäßig den gleichen Namen "C-<n>", wobei n eine laufende Nummer ist.

Wenn Sie ein Verbindungsmarkenpaar selektieren, wandelt der Befehl diese Marken in eine Verbindungslinie um.

Siehe auch

- [Kap. "CFC-Element 'Verbindungsmarke-Quelle/Ziel'" auf Seite 299](#)

Befehl 'Gruppe erzeugen'

Symbol: 

Funktion: Der Befehl gruppiert die selektierten Elemente.

Aufruf: Konfigurieren über **Extras** ► **Anpassen** ► **Befehle** ► **CFC**

Voraussetzungen: Ein CFC-Editor ist aktiv. Mehrere Elemente sind selektiert.

Gruppierte Elemente können nur noch zusammen bewegt werden. Die Position der Elemente wird durch die Gruppierung nicht beeinflusst.

Siehe auch

- [Kap. "Befehl 'Gruppierung aufheben'" auf Seite 699](#)

Befehl 'Gruppierung aufheben'

Symbol: 

Funktion: Der Befehl entfernt eine zuvor erstellte Gruppierung.

Aufruf: Konfigurieren über **Extras** ► **Anpassen** ► **Befehle** ► **CFC**

Voraussetzungen: Ein CFC-Editor ist aktiviert. Eine Gruppierung ist selektiert.

Siehe auch

- [Kap. "Befehl 'Gruppe erzeugen'" auf Seite 699](#)

Befehl 'Funktionsbausteinelement für Forcen vorbereiten'

Funktion Der Befehl schaltet für ein Funktionsbausteinelement die "Forcebarkeit" seiner Eingänge ein oder aus.

Aufruf	Der Befehl ist nicht Teil des standard Befehlssatzes von IndraWorks, kann aber über das IndraWorks Menü Extras ▶ Anpassen ▶ Befehle ▶ CFC in die Symbolleiste gezogen werden.
Voraussetzungen	Der CFC-Editor ist im Offlinebetrieb und ein Funktionsbausteinelement ist selektiert. Nach Ausführen des Befehls ist im Onlinebetrieb der "Befehl FB-Eingang forcen" verfügbar, der einen Dialog zum Forcen der Bausteineingangswerte öffnet. Siehe auch: • Kap. "Befehl 'FB-Eingang forcen'" auf Seite 700 • Kap. 18.2 "Menübefehle" auf Seite 610 • Kap. 12.4 "Forcen und Schreiben von Variablen" auf Seite 198

Befehl 'FB-Eingang forcen'



Diese Art des Forcens verwendet intern einen Datenhaltepunkt und ist damit vom Forcen über den Befehl "Werte forcen" bzw. [F7] zu unterscheiden!

Werte, die über den Befehl "FB-Eingang forcen" geforct wurden, reagieren nicht auf die Befehle "Alle Forces anzeigen", oder "Forcen für alle Werte aufheben".

Funktion	Der Befehl öffnet den Dialog "Wert forcen" zum Forcen des selektierten Eingangs eines Funktionsbausteins. Das Forcen kann mit dem gleichen Befehl und Dialog wieder aufgehoben werden.
Aufruf:	Der Befehl ist nicht Teil des standard Befehlssatzes von IndraWorks, kann aber über das IndraWorks Menü Extras ▶ Anpassen ▶ Befehle ▶ CFC in die Symbolleiste gezogen werden.
Voraussetzungen	Die "Forcebarkeit" des Bausteins ist über den Befehl "Bausteinelement für Forcen vorbereiten" aktiviert. Der CFC-Editor ist im Onlinebetrieb und der Eingang des Funktionsbausteins ist selektiert. Im Dialog "Wert forcen" können Sie entweder einen Wert eingeben, mit dem der Eingang des Funktionsbausteins geforct werden soll, oder den aktuell geforcten Wert wieder entfernen. Nach dem Forcen erscheint der Eingang grün hinterlegt. Boolesche Eingänge erhalten zudem ein kleines Monitoringfenster mit dem geforcten Wert. Der geforcte Wert wird in der Spalte "Wert" von Monitoringansichten angezeigt, also im Deklarationsteil des Programmbausteins oder in einer Überwachungsliste.

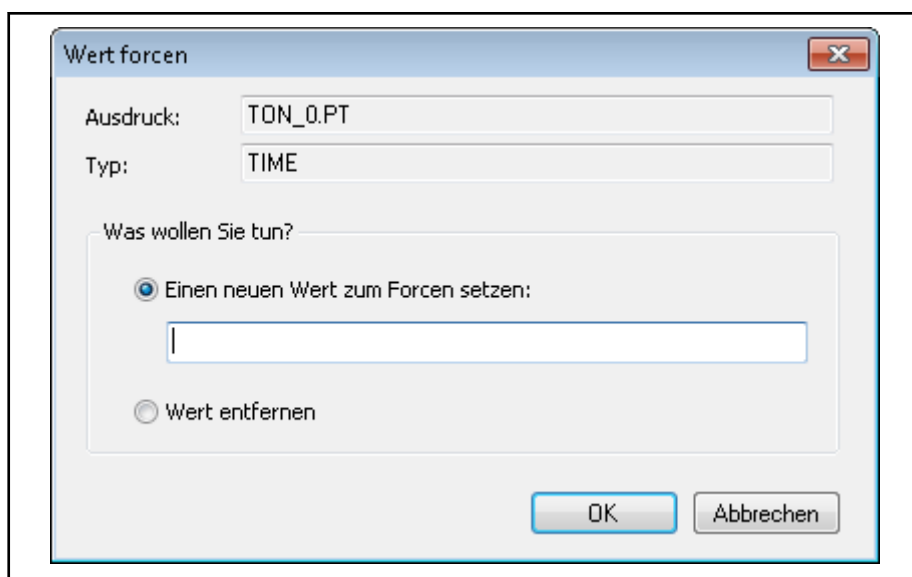


Abb. 18-69: Dialog "Wert forcen"

Dialog 'Wert forcen'

Ausdruck	Name des Funktionsbausteineingangs. Beispiel: TON_0.PT.
Typ	Datentyp des Eingangs

"Was wollen Sie tun?"

Einen neuen Wert zum Forcen setzen	 : Im Eingabefeld können Sie den gewünschten neuen Wert eintragen. Das Format muss dem Datentyp entsprechen.
Wert entfernen	 : Das Forcen am Eingang wird aufgehoben.

Siehe auch:

- [Kap. 18.2 "Menübefehle" auf Seite 610](#)
- [Kap. 12.4 "Forcen und Schreiben von Variablen" auf Seite 198](#)
- [Kap. "Befehl 'Funktionsbaustein element für Forcen vorbereiten'" auf Seite 699](#)

Befehl 'Attributierte Komponente als Eingang verwenden'

Symbol



Funktion

Der Befehl ermöglicht das Verbinden einer Strukturkomponenten mit einem Eingang skalaren Typs

Aufruf:

Menü **CFC ▶ Pins**.

Alternativ über das Kontextmenü im CFC Editor.

Voraussetzungen

Der CFC-Editor ist geöffnet und ein Bausteineingang ist selektiert

Die Komponente der Struktur, die mit dem Eingang des Folgebausteins verbunden wird, muss mit dem Attribut `{attribute 'ProcessValue'}` versehen sein.

Der Datentyp der Strukturkomponente muss zu dem Datentyp des nachfolgenden Eingangs kompatibel sein. So verbundene Eingänge werden mit dem Symbol "V" markiert.

Beispiel: Deklaration des Datentyps "QINT" mit Attribut

```
TYPE QINT :
STRUCT
  Status : STRING;
  {attribute 'ProcessValue'}
  Value1 : INT;
  Value2 : INT;
END_STRUCT
END_TYPE
```

Beispiel: Deklaration im Programm und Markierung des Eingangs "intValue"

```
PROGRAM PLC_PRG
VAR
  input1: QINT;
  output1: QINT;
  intValue: INT;
END_VAR
```

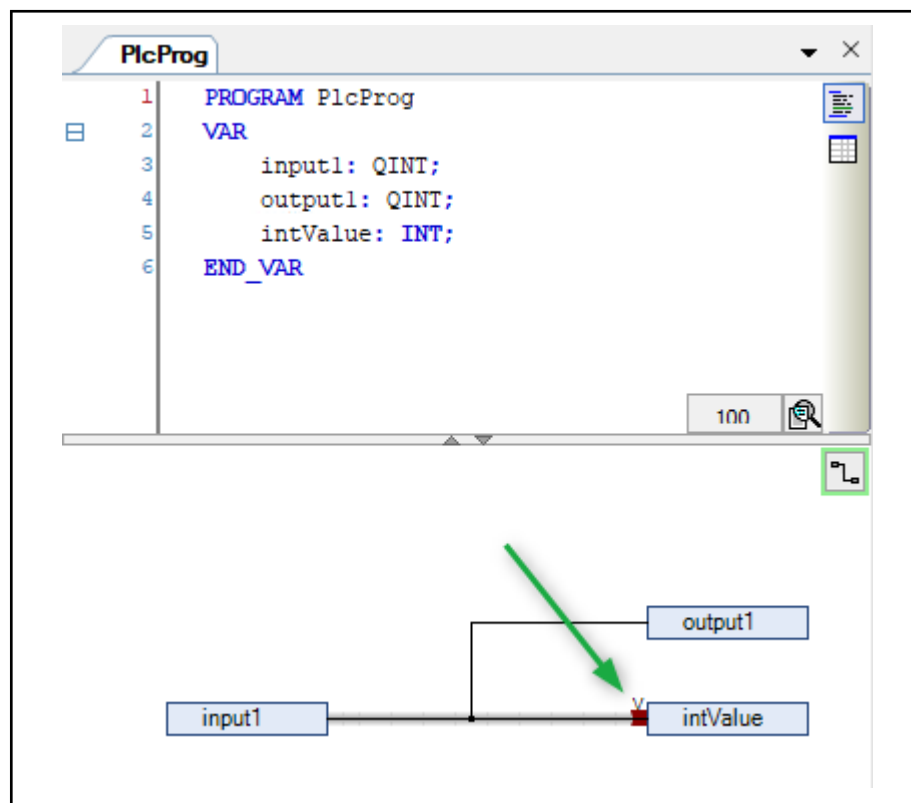


Abb. 18-70:

Siehe auch

- [Kap. "Attribut 'ProcessValue'" auf Seite 438](#)

18.2.9 Menü 'FUP/KOP/AWL'

Befehl 'Netzwerk einfügen'

Symbol: , Tastaturkürzel: <Strg> + <Umsch> +

Funktion: Der Befehl fügt ein weiteres Netzwerk im FUP/KOP/AWL-Editor ein.

Aufruf: Menü **FUP, KOP, AWL**, Kontextmenü

Voraussetzungen: Der FUP-, KOP- oder AWL-Editor ist aktiv. Kein Baustein ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Element 'Netzwerk'" auf Seite 282](#)
- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "Befehl 'Netzwerk einfügen unterhalb'" auf Seite 703](#)

Befehl 'Netzwerk einfügen unterhalb'

Symbol: , Tastaturkürzel: <Strg> + <Umsch> + <A>

Funktion: Der Befehl fügt ein weiteres Netzwerk im FUP/KOP/AWL-Editor unterhalb des selektierten Netzwerks ein.

Aufruf: Menü **FUP, KOP, AWL**, Kontextmenü

Voraussetzungen: Der FUP-, KOP- oder AWL-Editor ist aktiv. Ein Netzwerk ist selektiert. Kein Baustein ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "FUP/KOP/AWL-Element 'Netzwerk'" auf Seite 282](#)
- [Kap. "Befehl 'Netzwerk einfügen'" auf Seite 702](#)

Befehl 'Kommentierung ein/aus'

Symbol: , Tastaturkürzel: <Strg> + <Umsch> + <C>

Funktion: Der Befehl kommentiert das selektierte Netzwerk ein oder aus.

Aufruf: Menü **FUP, KOP, AWL**, Kontextmenü

Voraussetzungen: Der FUP-, KOP-, oder AWL-Editor ist aktiv. Ein Netzwerk ist selektiert. Es ist aber kein Baustein selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Zuweisung einfügen'

Symbol:  ^{VAR}, Tastaturkürzel: <Strg> + <A>

Funktion: Der Befehl fügt eine Zuweisung im FUP oder KOP Editor ein.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der FUP-, KOP- oder AWL-Editor ist aktiv. Ein Netzwerk ist selektiert, aber kein Baustein ist selektiert.



In AWL wird eine Zuweisung über die Operatoren **LD** und **ST** programmiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Bausteinaufruf einfügen'

Symbol: , Tastaturkürzel: <Strg> +

Funktion: Der Befehl fügt einen im Projekt verfügbaren Baustein am Ende des selektierten Netzwerks ein.

Aufruf: Menü **FUP, KOP, AWL**, Kontextmenü

Voraussetzungen: Der FUP-, KOP- oder AWL-Editor ist aktiv. Ein Netzwerk ist selektieren, es ist aber kein Baustein selektiert.

Wenn Sie den Befehl auswählen, öffnet die Eingabehilfe. Dort können Sie den gewünschten Baustein auswählen.

Siehe auch

- [Kap. "FUP/KOP/AWL-Element 'Baustein'" auf Seite 282](#)
- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112](#)

Befehl 'Baustein mit EN/ENO einfügen'

Symbol: , Tastaturkürzel: <Strg> + <Umschalt>+<E>

Funktion: Der Befehl fügt einen Baustein mit einem booleschen Eingang **ENable** und einen booleschen Ausgang **ENable Out** am Ende des selektierten Netzwerks ein.

Aufruf: Menü **FUP, KOP, AWL**, Kontextmenü

Voraussetzungen: Der FUP-, KOP- oder AWL-Editor ist aktiv. Ein Netzwerk ist selektiert, es ist aber kein Baustein selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112](#)

Befehl 'Leeren Baustein einfügen'

Symbol: 

Funktion: Der Befehl fügt am Ende des derzeit selektierten Netzwerks einen leeren Funktionsbaustein ein.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der FUP-, KOP- oder AWL-Editor ist aktiv. Ein Netzwerk ist selektiert, es ist aber kein Baustein selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Leeren Baustein mit EN/ENO einfügen'

Symbol: 

Funktion: Der Befehl fügt einen leeren Baustein mit einem booleschen Eingang **ENable** und einen booleschen Ausgang **ENable Out** am Ende des selektierten Netzwerks ein.

Aufruf: Menü **FUP, KOP, AWL**, Kontextmenü

Voraussetzungen: Ein FUP-Editor, KOP-Editor oder ein AWL-Editor ist aktiv. Ein Netzwerk muss selektiert sein. Kein anderer Baustein darf selektiert sein.

Wenn zum Zeitpunkt des Bausteinaufrufs **Enable** den Wert **FALSE** hat, werden die im Baustein definierten Operationen nicht ausgeführt. Andernfalls, also wenn **Enable** den Wert **TRUE** hat, werden diese Operationen ausgeführt. Der ENO-Ausgang wirkt als Repeater des EN-Eingangs.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Sprung einfügen'

Symbol:  , Tastaturkürzel: <Strg>+<J>

Funktion: Der Befehl fügt ein Sprungelement vor dem selektierten Element ein.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der FUP-, KOP- oder AWL-Editor ist aktiv. Ein Verbinder ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "FUP/KOP/AWL-Element 'Sprung'" auf Seite 283](#)

Befehl 'Sprungmarke einfügen'

Symbol: 

Funktion: Der Befehl fügt eine Sprungmarke in das aktuell selektierte Netzwerk ein.

Aufruf: Menü **FUP, KOP, AWL**, Kontextmenü

Voraussetzungen: Der FUP-, KOP- AWL-Editor ist aktiv. Ein Netzwerk ist selektiert. Keine Sprungmarke ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "FUP/KOP/AWL-Element 'Sprungmarke'" auf Seite 283](#)

Befehl 'Return einfügen'

Symbol: 

Funktion: Der Befehl fügt ein Element **Return** an der selektierten Stelle ein.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der FUP-, KOP- oder AWL-Editor ist aktiv. Ein Bausteingang ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "FUP/KOP/AWL-Element 'Return'" auf Seite 284](#)

Befehl 'Bausteineingang einfügen'

Symbol:  , Tastaturkürzel: <Strg>+<Q>

Funktion: Der Befehl fügt einem erweiterbaren Baustein (AND, OR, ADD, MUL, SEL) einen weiteren Eingang oberhalb des ausgewählten Eingangs hinzu.

Aufruf: Menü **FUP/KOP/AWL**

Voraussetzungen: Der FUP-, KOP-Editor ist aktiv. Ein Eingang eines Bausteins ist selektiert.

Wenn ein Baustein selektiert ist, ist im Kontextmenü der Befehl Namen **Eingang anhängen** verfügbar. Der Eingang wird am unteren Ende des Bausteins eingefügt.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Spule einfügen'

Symbol: , Tastaturkürzel: <Strg> + <I>

Funktion: Der Befehl fügt eine Spule im Netzwerk ein.

Aufruf: Menü **FUP**, **KOP**, **AWL**, Kontextmenü

Voraussetzungen: Der KOP-Editor ist aktiv. Ein Netzwerk, eine Spule oder ein Verbinder ist selektiert. Es ist aber kein Baustein selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "KOP-Element 'Spule'" auf Seite 286](#)

Befehl 'Set-Spule einfügen'

Symbol: 

Funktion: Der Befehl fügt eine Setspule im Netzwerk ein.

Aufruf: Menü **FUP**, **KOP**, **AWL**, Kontextmenü

Voraussetzungen: Der KOP-Editor ist aktiv. Ein Netzwerk, eine Spule oder eine Leitung ist selektiert. Es ist aber kein Baustein selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Reset-Spule einfügen'

Symbol: 

Funktion: Der Befehl fügt eine Reset-Spule im Netzwerk ein.

Aufruf: Menü **FUP**, **KOP**, **AWL**, Kontextmenü

Voraussetzungen: Der KOP-Editor ist aktiv. Ein Netzwerk, eine Spule oder eine Leitung ist selektiert. Es ist aber kein Baustein selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "KOP-Element 'Spule'" auf Seite 286](#)

Befehl 'Kontakt einfügen'

Symbol  Tastaturkürzel: <Strg> + <K>

Funktion: Der Befehl fügt einen Kontakt links neben dem gewählten Element ein.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der KOP-Editor ist aktiv. Eine Leitung oder ein Kontakt ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "KOP-Element 'Kontakt'" auf Seite 285](#)

Befehl 'Kontakt einfügen (rechts)'

Symbol: , Tastaturkürzel: <Strg> + <R>

Funktion: Der Befehl fügt einen Kontakt rechts neben dem gewählten Element ein.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der KOP-Editor ist aktiv. Eine Leitung, ein Kontakt oder ein Baustein ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "KOP-Element 'Kontakt'" auf Seite 285](#)

Befehl 'Kontakt parallel einfügen (unterhalb)'

Symbol: , Tastaturkürzel: <Strg> + <D>

Funktion: Der Befehl fügt einen Kontakt mit Leitungen parallel unterhalb des gewählten Elements ein.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der KOP-Editor ist aktiv. Eine Leitung oder ein Kontakt oder ein Baustein ist selektiert.



Geschlossene parallele Verzweigungen in einem KOP-Netzwerk können Sie als "Short Circuit Evaluation" (SCE) oder OR-Konstrukte programmieren. SCE-Zweige werden mit doppelten vertikalen Linien dargestellt, OR-Zweige mit einfachen. Sehen Sie hierzu die Hilfeseite zu "Geschlossenen Leitungsverzweigungen".

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "Befehl 'Kontakt parallel einfügen \(oberhalb\)'" auf Seite 707](#)
- [Kap. "Geschlossene Leitungsverzweigung" auf Seite 286](#)

Befehl 'Kontakt parallel einfügen (oberhalb)'

Symbol: , Tastaturkürzel: <Strg> + <E>

Funktion: Der Befehl fügt einen Kontakt mit Leitungen parallel oberhalb des gewählten Elements ein.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der KOP-Editor ist aktiv. Eine Leitung, in Kontakt oder ein Baustein ist selektiert.



Geschlossene parallele Verzweigungen in einem KOP-Netzwerk können Sie als "Short Circuit Evaluation" (SCE) oder OR-Konstrukte programmieren. SCE-Zweige werden mit doppelten vertikalen Linien dargestellt, OR-Zweige mit einfachen. Sehen Sie hierzu die Hilfeseite zu "Geschlossenen Leitungsverzweigungen".

Siehe auch

- [Kap. "KOP-Element 'Kontakt'" auf Seite 285](#)
- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "Geschlossene Leitungsverzweigung" auf Seite 286](#)

Befehl 'Parallelen Modus wechseln'

Funktion: Der Befehl wechselt eine parallele Leitungsverzweigung zwischen einem OR-Konstrukt und der "Short Circuit Evaluation" (SCE) .

Aufruf: Der Befehl ist nicht Teil des standard Befehlssatzes von IndraWorks, kann aber über das IndraWorks Menü **Extras ► Anpassen ► Befehle ► FUP/KOP/AWL** in die Symbolleiste gezogen werden.

Voraussetzungen: Der KOP-Editor ist aktiv. Eine senkrechte Linie einer parallelen Leitungsverzweigung ist selektiert.



Geschlossene parallele Verzweigungen in einem KOP-Netzwerk können Sie als "Short Circuit Evaluation" (SCE) oder OR-Konstrukte programmieren. SCE-Zweige werden mit doppelten vertikalen Linien dargestellt, OR-Zweige mit einfachen. Sehen Sie hierzu die Hilfeseite zu "Geschlossenen Leitungsverzweigungen".

Siehe auch:

- [Kap. 18.2.9 "Menü 'FUP/KOP/AWL'" auf Seite 702](#)
- [Kap. "Geschlossene Leitungsverzweigung" auf Seite 286](#)

Befehl 'Negierten Kontakt einfügen'

Symbol:

Funktion: Der Befehl fügt einen negierten Kontakt links neben dem gewählten Element ein.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der KOP-Editor ist aktiv. Eine Leitung oder ein Kontakt ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "KOP-Element 'Kontakt'" auf Seite 285](#)

Befehl 'Negierten Kontakt parallel einfügen (unterhalb)'

Symbol:

Funktion: Der Befehl fügt einen negierten Kontakt mit Leitungen parallel unterhalb des gewählten Element ein.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der KOP-Editor ist aktiv. Eine Leitung, ein Kontakt oder ein Baustein ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Kontakte einfügen: Darunter einfügen'

Tastaturkürzel: <Strg> + <Umsch> + <D>

Funktion: Der Befehl fügt einen zuvor kopierten Kontakt mit Leitungen unterhalb des selektierten Elements ein.

Aufruf: Menü **FUP/KOP/AWL ► Kontakte einfügen**, Kontextmenü

Voraussetzungen: Der KOP-Editor ist aktiv.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "KOP-Element 'Kontakt'" auf Seite 285](#)

Befehl 'Kontakte einfügen: Darüber einfügen'

Tastaturkürzel: <Strg> + <Umsch> + <E>

Funktion: Der Befehl fügt einen zuvor kopierten Kontakt mit Leitungen überhalb des selektierten Elements ein.

Aufruf: Menü **FUP/KOP/AWL ► Kontakte einfügen**, Kontextmenü

Voraussetzungen: Der KOP-Editor ist aktiv. Eine Leitung oder ein Kontakt ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "KOP-Element 'Kontakt'" auf Seite 285](#)

Befehl 'Kontakte einfügen: Rechts einfügen (danach)'

Tastaturkürzel: <Strg> + <Umsch> + <R>

Funktion: Der Befehl fügt einen zuvor kopierten Kontakt rechts neben dem selektierten Element ein..

Aufruf: Menü **FUP/KOP/AWL ► Kontakte einfügen**, Kontextmenü

Voraussetzungen: Der KOP-Editor ist aktiv. Eine Leitung oder ein Kontakt ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "KOP-Element 'Kontakt'" auf Seite 285](#)

Befehl 'AWL-Zeile danach einfügen'

Symbol: 

Funktion: Der Befehl fügt eine Anweisungszeile unterhalb der selektierten Zeile ein.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der AWL-Editor ist aktiv. Eine Zeile ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'AWL-Zeile löschen'

Symbol:  , Tastaturkürzel: <Strg>+<Entf>

Funktion: Der Befehl löscht die selektierte Anweisungszeile.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der AWL-Editor ist aktiv. Eine Zeile ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Negation'

Symbol:  , Tastenkürzel: <Strg>+ <T>

Funktion: Der Befehl negiert folgende Elemente:

- Ein-/Ausgang eines Bausteins
- Sprung
- Return
- Spule

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der FUP-, oder KOP-Editor ist aktiv. Das entsprechende Element ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Flankenerkennung'

Symbol FUP: , Symbol KOP: , Tastaturkürzel: <Strg>+ <N>

Funktion: Der Befehl fügt eine Flankenerkennung vor dem selektierten Bausteineingang oder Bausteinausgang ein.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der FUP- oder KOP-Editor ist aktiv. Ein Bausteineingang oder Bausteinausgang ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Set/Reset'

Symbol: , Tastaturkürzel <Strg>+ <Umsch><S>

Funktion: Der Befehl wechselt bei einem Element mit booleschen Ausgang zwischen Reset, Set und keiner Auszeichnung .

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der FUP- oder KOP-Editor ist aktiv. Ein Element mit booleschem Ausgang ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Weiterverschaltung festlegen'

Symbol: , Tastaturkürzel <Strg> + <Umsch> + <W>

Funktion: Der Befehl macht den selektierten Bausteinausgang zum weiterführenden Bausteinausgang.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der FUP- oder KOP-Editor ist aktiv. Einer von mehreren Bausteinausgängen ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Leitungsverzweigung einfügen'

Symbol: , Tastaturkürzel <Strg>+ <Umschalt> + <V>

Funktion: Der Befehl erstellt eine offene Leitungsverzweigung an der selektieren Leitung.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der FUP- oder KOP-Editor ist aktiv. Ein Ein- oder Ausgang eines Bausteins ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Element 'Leitungsverzweigung'" auf Seite 284](#)
- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Leitungsverzweigung oberhalb einfügen '

Symbol: 

Funktion: Der Befehl erstellt eine Leitungsverzweigung oberhalb der selektierten offenen Leitungsverzweigung.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der FUP- oder KOP-Editor ist aktiv. Eine offene Leitungsverzweigung ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "FUP/KOP/AWL-Element 'Leitungsverzweigung'" auf Seite 284](#)

Befehl 'Leitungsverzweigung unterhalb einfügen'

Symbol: 

Funktion: Der Befehl erstellt eine Leitungsverzweigung unterhalb der selektierten offenen Leitungsverzweigung.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der FUP- oder KOP-Editor ist aktiv. Eine offene Leitungsverzweigung ist selektiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "FUP/KOP/AWL-Element 'Leitungsverzweigung'" auf Seite 284](#)

Befehl 'Leitungsverzweigung Startpunkt setzen'

Symbol: 

Funktion: Der Befehl setzt den Startpunkt einer Leitungsverzweigung an der selektierten Leitung.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der KOP-Editor ist aktiv. Eine Leitung ist selektiert.

Siehe auch

- [Kap. "Geschlossene Leitungsverzweigung" auf Seite 286](#)
- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Leitungsverzweigung Endpunkt setzen'

Funktion: Der Befehl setzt den Endpunkt einer Leitungsverzweigung an der selektierten Leitung.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der KOP-Editor ist aktiv. Eine Leitung ist selektiert. Ein Startpunkt der Leitungsverzweigung wurde gesetzt.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)
- [Kap. "Geschlossene Leitungsverzweigung" auf Seite 286](#)

Befehl 'Parameter aktualisieren'

Funktion: Der Befehl pflegt Änderungen an der Deklaration des selektierten Elements in der Grafik ein.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der FUP-, KOP- oder CFC-Editor ist aktiv. Ein Baustein ist selektiert. Eine erweiternde Änderung an der Deklaration wurde vorgenommen.

Der Befehl prüft, ob ein Baustein und seine Deklaration im Deklarationseditor übereinstimmen. Die Änderung wird nur für den Baustein übernommen, wenn die Deklaration erweitert wurde. Löschungen und Überschreibungen werden nicht aktualisiert.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Nicht verwendete FB-Aufruf-Parameter entfernen'

Symbol: 

Funktion: Der Befehl löscht Eingänge und Ausgänge des selektierten Bausteins, denen keine Variable und kein Wert zugewiesen wurde. Die standardmäßigen Eingänge und Ausgänge des Bausteins bleiben aber immer erhalten.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der FUP- oder KOP-Editor ist aktiv. Ein Baustein ist selektiert. Der Baustein verfügt über Schnittstellen, denen kein Wert zugewiesen ist.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Baustein reparieren'

Symbol: 

Funktion: Der Befehl repariert interne Inkonsistenzen am selektierten Baustein.

Aufruf: Menü **FUP/KOP/AWL**, Kontextmenü

Voraussetzungen: Der FUP-, oder der KOP-Editor ist aktiv. Der fehlerhafte Baustein ist selektiert. Der Editor hat interne Inkonsistenzen im Programmierbaustein entdeckt, die eventuell automatisch aufgelöst werden können. Die Inkonsistenzen meldet IndraLogic im Meldungsfenster.

Diese Situation ist denkbar beim Bearbeiten eines Projekts, das mit einer älteren Programmiersystemversion erstellt wurde, die die betreffende Inkonsistenz noch nicht als Fehler behandelte.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Als Funktionsbausteinsprache anzeigen'

VORSICHT

Datenverlust!

Eine fehlerfreie Konvertierung setzt syntaktisch korrekten Code voraus. Anderenfalls können Teile der Implementierung verloren gehen.

Tastaturkürzel: <Strg> + <Umsch> + <F>

Funktion: Der Befehl konvertiert die aktive Anweisungsliste oder den aktiven Kontaktplan in die Funktionsbausteinsprache.

Aufruf: Menü **FUP/KOP/AWL ▶ Ansicht**

Voraussetzungen: Der KOP-Editor oder der AWL-Editor ist aktiv.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Als Kontaktplan anzeigen'

VORSICHT

Datenverlust!

Eine fehlerfreie Konvertierung setzt syntaktisch korrekten Code voraus. Anderenfalls können Teile der Implementierung verloren gehen.

Tastaturkürzel: <Strg> + <Umsch> + <L>

Funktion: Der Befehl konvertiert das aktuelle Funktionsbausteincode oder die aktive Anweisungsliste in einen Kontaktplan.

Aufruf: Menü **FUP/KOP/AWL ▶ Ansicht**

Voraussetzungen: Der FUP- oder AWL-Editor ist aktiv.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Als Anweisungsliste anzeigen'



AWL kann bei Bedarf über die IndraWorks-Optionen aktiviert werden.

Extras ▶ Optionen ▶ IndraLogic 2G ▶ FUP, KOP und AWL ▶ AWL ▶ AWL aktivieren

VORSICHT

Datenverlust!

Eine fehlerfreie Konvertierung setzt syntaktisch korrekten Code voraus. Anderenfalls können Teile der Implementierung verloren gehen.

Tastaturkürzel: <Strg> + <Umsch> + <I>

Funktion: Der Befehl konvertiert den aktiven Funktionsbausteincode oder den aktiven Kontaktplan in eine Anweisungsliste.

Aufruf: Menü **FUP/KOP/AWL ▶ Ansicht**

Voraussetzungen: Der KOP- oder FUP-Editor ist aktiv.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

Befehl 'Gehe zu'

Symbol:

Funktion: Der Befehl lässt Sie zu einem beliebigen Netzwerk springen.

Aufruf: Menü **FUP/KOP/AWL**

Voraussetzungen: Der KOP-Editor, FUP-Editor oder der AWL-Editor ist aktiv. Ein Netzwerk ist selektiert.

Der Befehl öffnet einen Dialog mit einem Eingabefeld. Geben Sie die Nummer des gewünschten Netzwerks in das Eingabefeld ein.

Siehe auch

- [Kap. "FUP/KOP/AWL-Editor" auf Seite 272](#)

18.2.10 Menü 'Bibliothek'

Die Befehle des Bibliotheksverwalters können aus seinem Editorfenster aufgerufen werden (Bibliotheksverwalter).

Wenn erforderlich, kann die Menükonfiguration über den IndraWorks **Extras ▶ Anpassen**-Dialog verändert werden.

Befehl 'Bibliothek hinzufügen'

Funktion: Der Befehl öffnet den Dialog **Bibliothek hinzufügen**. In diesem Dialog können Sie Bibliotheken zum Bibliotheksverwalter hinzufügen und so in Ihre Applikation einbinden.

Voraussetzung: Der Bibliotheksverwalter ist im Editor geöffnet.

Aufruf: Menü **Bibliotheksverwalter ▶ Bibliothek hinzufügen** oder in der Kopfzeile des Editors **Bibliothek hinzufügen**

Dialog 'Bibliothek hinzufügen'

In der Zeile oberhalb der Bibliotheksliste können Sie nach Bibliotheksnamen oder Bibliotheksbausteinen suchen, indem Sie eine entsprechende Zeichenfolge eintippen.	
Bibliothek	Liste alle im Bibliotheks-Repository installierten Bibliotheken, gruppiert in Kategorien.
Firma	Hersteller der Bibliothek.
Erweitert	Öffnet den erweiterten Dialog "Bibliothek hinzufügen"
	Die angezeigten Bibliotheken werden in Kategorien gruppiert
	Die angezeigten Bibliotheken werden alphabetisch aufgelistet
	Alle zur Verfügung stehenden Bibliotheken werden angezeigt



In der Gerätebeschreibung von Geräten können bestimmte Bibliotheken als "blacklisted" gekennzeichnet sein. Diese Bibliotheken können im Bibliotheksverwalter unterhalb dieses Geräts nicht hinzugefügt werden.

Siehe auch

- [Kap. "Befehl 'Bibliotheks-Repository'" auf Seite 676](#)

Dialog 'Bibliothek hinzufügen' - 'Erweitert'

Sie sollten diesen Dialog nur verwenden, wenn Sie fundierte Kenntnisse bezüglich Bibliotheksreferenzierung besitzen. Mit dem Dialog können Sie spezielle Versionen einbinden oder Platzhalterdefinitionen verändern.



Für das Entwickeln und Referenzieren von Bibliotheken wird dringend empfohlen, den Richtlinien zum Erstellen von Bibliotheken zu folgen.

Registerkarte 'Bibliothek'

Firma	Filterung der Liste nach Hersteller.
Gruppieren nach Kategorie	☒ : Darstellung der Bibliotheken in einer Baumstruktur in Kategorien gruppiert. ☐ : Darstellung der Bibliotheken alphabetisch in einer flachen Struktur.
Alle Versionen anzeigen	☒ : Anzeige aller Versionen der Bibliotheken. Versionsangabe "" bedeutet aktuellste im Repository verfügbare Version. ☐ : Anzeige nur der aktuellsten Versionen der Bibliotheken. In dieser Darstellung ist eine Mehrfachauswahl von Bibliotheken möglich: Halten Sie die <Umschalt>-Taste gedrückt, während Sie die Einträge auswählen.
Details	Öffnet eine detaillierte Ansicht mit den Bausteinen der Bibliothek.
Bibliotheks-Repository	Öffnet den Dialog Bibliotheks-Repository . Dort können Sie weitere Bibliotheken auf Ihrem lokalen System installieren.

Siehe auch

- [Kap. "Befehl 'Bibliotheks-Repository'" auf Seite 676](#)

Registerkarte 'Platzhalter'

Platzhaltername	Das Eingabefeld unterstützt die Eingabe mit einer Combobox der gültigen Platzhalternamen, die aus den gerade zugänglichen Gerätebeschreibungen gelesen werden. Sie können auch einen neuen Platzhalternamen eintragen, um einen "freien Platzhalter" zu definieren, der nicht vom Gerät oder vom Bibliotheksprofil aufgelöst wird.
Standardbibliothek	IndraLogic verwendet diese Bibliothek, wenn aus irgendwelchen Gründen kein Gerät verfügbar ist, das die Auflösung definiert. Somit ist in dieser Situation eine fehlerfreie Übersetzung des gerade geöffneten Projekts möglich.



Hinweis zur Platzhalterauflösung

Wenn die selektierte Bibliothek gemäß den "Richtlinien für Bibliotheksentwicklung" entwickelt wurde, ist ein Ändern der folgenden Einstellungen nicht zu empfehlen.

Siehe auch

- [Kap. "Befehl 'Platzhalter'" auf Seite 718](#)

Befehl 'Versuchen Bibliothek neu zu laden'

Funktion: Der Befehl versucht die selektierte Bibliothek erneut zu laden.

Aufruf: Menü **Bibliotheksverwalter** ► **Versuchen Bibliothek neu zu laden**.

Voraussetzung: Es ist eine Bibliothek selektiert, deren Laden fehlgeschlagen ist.

Wenn eine Bibliothek aus irgendeinem Grund nicht im definierten Repository-Pfad bereitsteht, wenn ein Projekt geöffnet wird, gibt IndraLogic eine entsprechende Fehlermeldung aus.

Nachdem Sie den Fehler behoben haben, also die Bibliothek wieder ordnungsgemäß verfügbar ist, können Sie mit dem Befehl diese Bibliothek nachladen, ohne dass Sie das Projekt verlassen müssen.

Siehe auch

- [Kap. "Befehl 'Platzhalter'" auf Seite 718](#)

Befehl 'Eigenschaften'

Funktion: Der Befehl öffnet den Dialog **Eigenschaften** für die im Bibliotheksverwalter selektierte Bibliothek.

Aufruf: Menü **Bibliotheksverwalter ▶ Eigenschaften** oder in der Kopfzeile des Editors **Eigenschaften**

Voraussetzung: Eine Bibliothek ist selektiert.



Dieser Dialog ist für Bibliotheksentwickler gedacht. Bitte benutzen Sie ihn nur, wenn Sie fundierte Kenntnisse in der Bibliotheksreferenzierung haben und beachten Sie unbedingt auch die Richtlinien für Bibliotheksentwickler.

Dialog 'Eigenschaften'

Allgemein

Namensraum	Namensraum der selektierten Bibliothek. Üblicherweise identisch mit dem Bibliotheksnamen, außer er wurde beim Erstellen der Bibliothek explizit in den Bibliotheksinformationen definiert. Sie können für das (gerade geöffnete) Projekt den Namensraum ändern. Beispiel: LA
Standard	Bibliothek, die den Platzhalter auflöst, wenn keine andere Auflösung definiert oder möglich ist. Voraussetzung: Die selektierte Bibliothek ist ein Bibliotheksplatzhalter, dann ist die Einstellung verfügbar. Hinweis: Ab Compilerversion 3.5.8.0 gilt für einen Bibliotheksplatzhalter, für den es eine Auflösung in der Gerätebeschreibung gibt, der aber im Bibliotheksverwalter der Ansicht „Allgemeiner Baustein“ liegt folgendes: > Dieser Platzhalter wird immer automatisch gemäß der Beschreibung des Geräts aufgelöst, für das die Applikation übersetzt wird.



Hinweis zur Platzhalterauflösung

Wenn die selektierte Bibliothek gemäß den "Richtlinien für Bibliotheksentwicklung" entwickelt wurde, ist ein Ändern der folgenden Einstellungen nicht zu empfehlen.

Version

Auswahl des Versionszwangs

Voraussetzung: Nur wenn die selektierte Bibliothek kein Bibliotheksplatzhalter ist, sind die Einstellungen verfügbar.

Hinweis: Container- und Interface-Bibliotheken werden automatisch mit Bibliotheksreferenzen mit entsprechendem Versionszwang erstellt.

Erstellen Sie möglichst keine Bibliothek, die Bibliotheksreferenzen mit Versionszwang enthält, sondern referenzieren Sie Bibliotheken über Platzhalter. Eine Platzhalterauflösung können Sie im Dialog „Platzhalter“ bearbeiten.

Exakte Version	 : (Über Auswahlliste gewählte) Version wird fest im Projekt eingebunden. Hinweis: Diese Option ist für Containerbibliotheken dringend empfohlen und bei diesem Bibliothekstyp üblicherweise voreingestellt.
Immer neueste Version	 : Bibliotheksrepository wird durchsucht und die neueste dort gefundene Version wird eingebunden. Hinweis: Wenn eine neuere Version der Bibliothek verfügbar ist, können sich die tatsächlich verwendeten Bibliotheksbausteine ändern. Diese Option ist für Interfacebibliotheken dringend empfohlen und bei diesem Bibliothekstyp üblicherweise voreingestellt.

Sichtbarkeit

Nur qualifizierten Zugriff auf alle Bezeichner erlauben	 : Bibliotheksbausteine (und Variablen) werden im Projekt nur mit vorangestelltem Namensraumpfad aufgerufen.
Wenn das aktuelle Projekt in einem anderen Projekt als Bibliothek referenziert wird	Hinweis: Nur wenn Sie mit Ihrem Projekt eine Bibliothek erstellen und somit ein Bibliotheksprojekt geöffnet haben, ist ein Ändern der folgenden Einstellungen sinnvoll. Die selektierte Bibliothek wird dabei in der neuen Bibliothek referenziert.

Alle IEC-Symbole im Projekt sichtbar machen, als ob diese Referenz hier direkt eingebunden worden wäre	: Die selektierte Bibliothek macht (später in einem Projekt) als Containerbibliothek den Inhalt der referenzierten Bibliothek toplevel (auf ihrer obersten Ebene) sichtbar. Voraussetzung: Sie erstellen mit dem Bibliotheksprojekt eine Containerbibliothek. Eine Containerbibliothek ist eine Bibliothek, die keine eigenen Bausteine implementiert, sondern ausschließlich andere Bibliotheken referenziert. Sie bündelt Bibliotheken. Eine Containerbibliothek kann sinnvoll eingesetzt werden, um mehrere Bibliotheken zusammen (in einer Referenz) in ein Projekt einzubinden. Für jede Bibliotheksreferenz muss dabei diese Option aktiviert sein. Symbolischer Zugriff auf Bibliotheksbausteine: <code><Namensraum Containerbibliothek>.<Bausteinname></code> : Der Inhalt der referenzierten Bibliothek ist über den Namensraum eindeutig ansprechbar. Der Pfadname setzt sich aus dem Namen der Bibliothek und dem eigenen Namen (der Bibliotheksreferenz) zusammen und wird dem Bausteinnamen vorangestellt. Voraussetzung: Sie erstellen mit dem Bibliotheksprojekt keine Containerbibliothek. Eindeutiger symbolischer Zugriff auf Bibliotheksbaustein: <code><Namensraum Bibliothek>.<Namensraum Unterbibliothek>.<Bausteinname></code>
Diese Referenz im Abhängigkeitsbaum nicht anzeigen	: Die selektierte Bibliothek wird (später in einem Projekt) nicht als Bibliotheksreferenz im Bibliotheksverwalter angezeigt. Die Bibliothek wird versteckt referenziert. Warnung: Im Fall von Übersetzungsfehlern, die auf Fehler in der versteckten Bibliothek zurückgehen, kann die Fehlersuche schwierig sein. : Die selektierte Bibliothek wird (später in einem Projekt) als Bibliotheksreferenz angezeigt.
Optional (wenn die Bibliothek fehlt, wird kein Fehler ausgegeben):	: Die selektierte Bibliothek wird als optional behandelt. Beim Laden eines Projekts, das die Bibliothek referenziert, wird kein Fehler ausgegeben, auch wenn die Bibliothek im Bibliotheksrepository nicht vorhanden ist.

Siehe auch

- [Kap. "Befehl 'Platzhalter'" auf Seite 718](#)

Befehl 'Platzhalter'

Funktion: Der Befehl öffnet den Dialog **Platzhalter**. Der Dialog informiert über die selektierte Platzhalterbibliothek und ermöglicht eine speziell für das Projekt gültige Auflösung zuzuweisen.

Aufruf: Menü Kopfzeile des Editors **Platzhalter**.

Voraussetzung: Der Bibliotheksverwalter ist im Editor geöffnet.

Eine Platzhalterbibliothek, die unter dem Geräteknoten „Application“ eingebunden ist, wird folgendermaßen aufgelöst:

- Wenn Sie der Platzhalterbibliothek eine spezielle Auflösung im Dialog "Platzhalter" zugewiesen haben, wird diese übernommen.
- Wenn keine spezielle Auflösung angegeben ist, wird geprüft, ob in der Gerätebeschreibung der Applikation eine Auflösung angegeben ist.
- Dann wird geprüft, ob im Bibliotheksprofil eine Auflösung angegeben ist.
- Das Ergebnis erscheint im Bibliotheksverwalter unter "Effektive Version".

Eine Platzhalterbibliothek, die im Ordner "Allgemeiner Bausteinteil" eingebunden ist, wird folgendermaßen aufgelöst:

- Eine im Dialog "Platzhalter" zugewiesene spezielle Auflösung wird **ignoriert**.
- Es wird applikationsabhängig geprüft, ob in der zugehörigen Gerätebeschreibung eine Auflösung angegeben ist.
- Dann wird das Bibliotheksprofil geprüft.
- Das Ergebnis erscheint im Tooltip des Symbols 

Siehe auch

- [Kap. 16.1 "Informationen für Bibliotheksentwickler" auf Seite 236](#)
- [Kap. 18.1.14 "Objekt 'Bibliotheksverwalter'" auf Seite 548](#)

Dialog 'Platzhalter'

Name	Bezeichner des Platzhalters
Bibliothek	Aktuelle Auflösung für das Projekt Doppelklicken Sie auf den Eintrag, um die Platzhalterauflösung zu bearbeiten. Es erscheint eine Auswahlliste der verfügbaren Versionen der Bibliothek. Außerdem erscheint der Befehl "Andere Bibliotheken"
Befehl "Andere Bibliothek"	Der Dialog "Bibliothek durchsuchen" zum Suchen und Installieren von Bibliotheken öffnet sich. Wählen Sie den Befehl, um nicht auf eine andere Version, sondern auf eine spezielle Bibliothek umzuleiten.
Info	Art der Platzhalterauflösung • Aufgelöst durch Gerätebeschreibung • Aufgelöst durch Bibliotheksprofil • Aufgelöst durch <spezielle Bibliothek>

Befehl 'Bibliotheksinformationen'

Symbol: 

Funktion: Der Befehl öffnet den Dialog **Bibliotheksinformationen**.

Aufruf: Ordner **Allgemeiner Baustein**

Voraussetzung: IndraLogic erstellt das Objekt automatisch in Bibliotheksprojekten. In anderen Projekten kann der Dialog über den Kontextmenübefehl **Hinzufügen ► Bibliotheksinformationen** erzeugt werden.

Siehe auch

- [Kap. 2.5.1 "Bibliotheksinfo abfragen und bearbeiten" auf Seite 12](#)
- [Kap. 2.5.2 "Projekteinstellungen vornehmen" auf Seite 14](#)
- [Kap. 18.1.20 "Objekt 'Bibliotheksinfo'" auf Seite 579](#)

18.2.11 Menü 'Bildersammlung'

Befehl 'Bilddatei einfügen'

Symbol: 

Funktion: Der Befehl fügt eine neue Zeile in eine Bildersammlung ein.

Aufruf: Menü **Imagepool** oder Kontextmenü

Voraussetzungen: Eine Bildersammlung ist aktiv und eine Zeile in der Bildersammlung ist selektiert.

Siehe auch

- [Kap. 18.1.13 "Objekt 'Bildersammlung'" auf Seite 547](#)

18.2.12 Menü 'Deklarationen'

Befehl 'Deklarationskopf editieren'

Funktion: Der Befehl öffnet den Dialog **Deklarationskopf editieren**, der im tabellarischen Deklarationseditor zur Konfiguration des Kopfteils einer POU dient.

Aufruf: Menüpunkt **<pou-name>** oder Kontextmenü des tabellarischen Deklarationseditors.

Voraussetzung: Der Fokus liegt im tabellarischen Deklarationseditor.

Siehe auch

- ["Deklarieren im tabellarischen Deklarationseditor" auf Seite 88](#)

Dialog 'Deklarationskopf editieren'

Funktion: In dem Dialog konfigurieren Sie den Kopfteil des tabellarischen Deklarationsteils einer POU.

Aufruf: Mausklick auf die Kopfleiste des tabellarischen Deklarationseditors, Kontextmenü im tabellarischen Deklarationseditor.

Deklaration	Auswahlliste zur Änderung des POU-Typs • PROGRAM • FUNCTION_BLOCK – EXTENDS: Eingabefeld für einen Basis-Funktionsbaustein – IMPLEMENTS: Eingabefeld für eine Schnittstelle • FUNCTION – Rückgabebetyp Eingabefeld mit aktuellem POU-Namen: Sie können den Namen der POU ändern.
Attribute	Der Dialog Attribute öffnet sich zur Eingabe von Attributen und Pragmas.

Siehe auch

- [Kap. 9.2.1 "Deklarationseditor verwenden" auf Seite 87](#)
- [Kap. 17.6 "Pragmas" auf Seite 402](#)

Befehl 'Nach unten verschieben'

Symbol: 

Funktion: Der Befehl verschiebt eine Variablendeklaration um eine Zeile nach unten.

Aufruf: Menüpunkt **<pou-name>** oder Kontextmenü des tabellarischen Deklarationseditors.

Voraussetzung: Im tabellarischen Deklarationseditor ist eine Zeile mit einer Variablendeklaration selektiert.

Siehe auch

- [Kap. 9.2.1 "Deklarationseditor verwenden" auf Seite 87](#)

Befehl 'Nach oben verschieben'

Symbol: 

Funktion: Der Befehl verschiebt eine Variablendeklaration um eine Zeile nach oben.

Aufruf: Menüpunkt **<pou-name>** oder Kontextmenü des tabellarischen Deklarationseditors.

Voraussetzung: Im tabellarischen Deklarationseditor ist eine Zeile mit einer Variablendeklaration selektiert.

Siehe auch

- [Kap. 9.2.1 "Deklarationseditor verwenden" auf Seite 87](#)

18.2.13 Menü 'Deklarationen' (Persistenz)

Befehl 'Liste neu ordnen und Lücken bereinigen'

Funktion: Der Befehl bereinigt die Lücken, die entstehen können, wenn Sie Änderungen in der Deklaration von persistenten Variablen vornehmen. Durch die Bereinigung reduziert sich der Speicherbedarf.

Beim Ausführen des Befehls zeigt IndraLogic eine Warnmeldung an, die den Anwender auf den möglichen Verlust von Daten hinweist.

Aufruf: **PersistentVars** oder Kontextmenü

Voraussetzung: Der Persistenzeditor (persistente Variablenliste) ist aktiv.



Vor einer Bereinigung sollten Sie in Erwägung ziehen, die aktuellen Werte der persistenten Variablen in einer Rezeptur zu speichern (Befehl **Aktuelle Werte in Rezeptur speichern**). Dann können Sie die Werte nach dem nächsten Download wieder auf die Steuerung spielen.

Häufige Fragen

- F: Beim (erstmaligen) Abspeichern aktueller Werte in eine Rezeptur wird dies mit einer Fehlermeldung verweigert.
- A: Überprüfen Sie, ob in Ihrem Projekt ein Bibliotheksinfo-Objekt (Ordner "Allgemeiner Baustein") als "Projektinformation" verwendet wird (Bibliotheksinformation). Schalten Sie die Option "Freigegeben" unter "Zusammenfassung" ab und wiederholen das Abspeichern.

Siehe auch

- [Kap. 9.16 "Datenpersistenz" auf Seite 144](#)
- [Kap. "Befehl 'Aktuelle Werte in Rezeptur speichern'" auf Seite 721](#)
- [Kap. "Befehl 'Werte aus Rezeptur wiederherstellen'" auf Seite 722](#)

Befehl 'Aktuelle Werte in Rezeptur speichern'

Funktion: Der Befehl erzeugt eine neue Rezepturdefinition im Rezepturmanager und speichert darin die aktuellen Werte der persistenten Variablen. Den Befehl sollten Sie vor dem Befehl **Liste neu ordnen und Lücken bereinigen** ausführen, um möglichen Datenverlust zu vermeiden.

Mit dem Kontextmenübefehl **Werte aus Rezeptur wiederherstellen** können Sie anschließend die Daten wiederherstellen.

Aufruf: **PersistentVars** oder Kontextmenü

Voraussetzung: Die Applikation ist Online-Betrieb und der Persistenzeditor (persistente Variablenliste) ist aktiv.



Wenn beim Speichern einer persistenten Variablenliste bereits eine Liste mit entsprechenden Namen im Rezeptur-Manager existiert, dann werden die aktuellen persistenten Variablen in diese Liste einsortiert:

- Neue persistente Variablen werden der Liste hinzugefügt
- Variablen, die sich nicht in der Liste der persistenten Variablen befinden, werden gelöscht

Es ist daher möglich, zu der Liste im Rezeptur-Manager weitere Rezepturen hinzuzufügen, die beibehalten werden. Wenn aber neue Variablen in die Liste hinzugefügt werden, dann werden diese mit der nächsten Ausführung des Befehls "Aktuelle Werte in Rezeptur speichern" gelöscht.

Siehe auch

- [Kap. 9.16 "Datenpersistenz" auf Seite 144](#)
- [Kap. "Befehl 'Liste neu ordnen und Lücken bereinigen'" auf Seite 721](#)
- [Kap. "Befehl 'Werte aus Rezeptur wiederherstellen'" auf Seite 722](#)
- [Kap. 18.1.12 "Objekt 'Persistente Variablenliste'" auf Seite 546](#)

Befehl 'Werte aus Rezeptur wiederherstellen'

Funktion: Der Befehl stellt die Werte der persistenten Variablen wieder her, die Sie mit dem Befehl **Aktuelle Werte in Rezeptur speichern** in einer Rezeptur gespeichert haben.

Den Befehl wählen Sie normalerweise nach Ausführung des Kontextmenübefehls **Liste neu ordnen und Lücken bereinigen** aus.

Aufruf: **PersistentVars** oder Kontextmenü

Voraussetzung: Der Persistenzeditor (persistente Variablenliste) ist aktiv, die Applikation ist im Online-Betrieb

Siehe auch

- [Kap. 9.16 "Datenpersistenz" auf Seite 144](#)
- [Kap. "Befehl 'Aktuelle Werte in Rezeptur speichern'" auf Seite 721](#)
- [Kap. "Befehl 'Liste neu ordnen und Lücken bereinigen'" auf Seite 721](#)
- [Kap. 18.1.12 "Objekt 'Persistente Variablenliste'" auf Seite 546](#)

Befehl 'Alle Instanzpfade hinzufügen'

Funktion:

- Wenn Sie den Befehl im Persistenzeditor auswählen, wird die Applikation nach Deklarationen persistenter Variablen mit dem Schlüsselwort `PERSISTENT` durchsucht, die sich außerhalb des Persistenzeditors befinden. Für jede gefundene Deklaration wird im Persistenzeditor ein Instanzpfad dieser Variablen hinzugefügt.
- Wenn Sie den Befehl in einer Variablenkonfiguration auswählen, wird für jede Variablen mit unvollständiger Adresse ein Instanzpfad hinzugefügt. Dabei werden alle Funktionsbausteine der Applikation berücksichtigt.

Aufruf: Menü **PersistentVars** oder alternativ im Kontextmenü des Persistenz-Editors

Voraussetzungen:

- Der Persistenzeditor (globale persistente Variablenliste) ist aktiv oder eine Variablenkonfiguration (globale Variablenliste mit `VAR_CONFIG`-Deklarationen) ist geöffnet.
- Die Applikation wurde erfolgreich übersetzt.

Siehe auch

- [Kap. 9.16 "Datenpersistenz" auf Seite 144](#)
- [Kap. 17.2.11 "Persistente Variable - PERSISTENT" auf Seite 308](#)
- [Kap. 9.10.1 "Variablenkonfiguration - VAR_CONFIG" auf Seite 128](#)

18.2.14 Menü 'Rezepturen'

Befehl 'Variable einfügen'

Symbol: 

Funktion: Der Befehl fügt eine Variable in die gerade geöffneten Rezepturdefinition vor der ausgewählten Position ein.

Aufruf: Menü **Rezepturen**

Voraussetzung: Es ist eine Rezepturdefinition im Editor geöffnet ist und die normale Ansicht gewählt

Dabei fügt IndraLogic in der Spalte **Variable** den Standardtext "NeueVariable" ein. Diesen Namen müssen Sie durch den entsprechenden gültigen Variablennamen ersetzen werden. Dazu öffnen Sie die Eingabehilfe über Schaltfläche  oder tragen Sie den Variablennamen direkt in das Tabellenfeld ein.

Siehe auch

- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)
- [Kap. 18.2.14 "Menü 'Rezepturen'" auf Seite 723](#)

Befehl 'Rezeptur einfügen'

Symbol: 

Funktion: Der Befehl öffnet einen Dialog zum Einfügen einer neuen Rezeptur (einer neuen Spalte) in die Rezepturdefinition.

Aufruf: Menü **VI-Logic Rezepturdefinition** oder Kontextmenü

Voraussetzung: Es ist eine Rezepturdefinition im Editor geöffnet.

Wenn Sie den Befehl ausführen, öffnet sich ein Dialog, in dem Sie den Namen der neuen Rezeptur festlegen. Der Dialog bietet Ihnen auch die Möglichkeit, bestehende Rezepturen in die neue Rezeptur zu kopieren.

Siehe auch

- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)
- [Kap. 18.2.14 "Menü 'Rezepturen'" auf Seite 723](#)

Befehl 'Rezeptur entfernen'

Symbol: 

Funktion: Der Befehl löscht eine Rezeptur aus der gerade geöffneten Rezepturdefinition.

Aufruf: Menü **VI-Logic Rezepturdefinition** oder Kontextmenü

Voraussetzung: Sie haben ein Feld in der Rezepturspalte einer Rezepturdefinition selektiert.

Siehe auch

- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)
- [Kap. 18.2.14 "Menü 'Rezepturen'" auf Seite 723](#)

Befehl 'Rezeptur laden'

Symbol: 

Funktion: Der Befehl lädt eine Rezeptur aus einer Datei.

Aufruf: Menü **VI-Logic Rezepturdefinition** oder Kontextmenü

Voraussetzung: Sie haben ein Feld einer Rezeptur in der Rezepturdefinition selektiert.

Wenn Sie den Befehl ausführen, werden die Werte der selektierten Rezeptur der Rezepturdefinition überschrieben.



Falls Sie die Option **Rezepturverwaltung in der SPS (Rezepturverwalter ► Allgemein)** aktiviert haben, beachten Sie bitte Folgendes:

Wenn Rezepturen im Projekt durch den Befehl **Rezeptur laden** oder **Rezeptur lesen** geändert werden, wird bei einem erneuten Einloggen ein Online Change notwendig.



Wenn Sie nur einzelne Variablen der Rezeptur mit neuen Werten überschreiben möchten, entfernen Sie vor dem Laden in der Rezepturdatei die Werte für die restlichen Variablen. Einträge ohne Wertangabe werden nicht eingelesen und somit bleiben diese Variablen auf der Steuerung und im Projekt durch die Aktualisierung unberührt.

Bei Werten vom Datentyp `REAL/LREAL` wird in manchen Fällen auch der Hexadezimalwert in die Rezepturdatei geschrieben. Dies ist notwendig, damit bei der Rückkonvertierung der exakt identische Wert wiederhergestellt wird. In diesem Fall ändern Sie den Dezimalwert und löschen Sie den Hexadezimalwert.

Siehe auch

- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)
- [Kap. 18.2.14 "Menü 'Rezepturen'" auf Seite 723](#)

Befehl 'Rezeptur speichern'

Symbol: 

Funktion: Der Befehl speichert die Werte der Variablen einer Rezeptur in eine Datei.

Aufruf: Menü **VI-Logic Rezepturdefinition** oder Kontextmenü

Voraussetzung: Sie haben den Wert einer Rezeptur in der Rezepturdefinition selektiert.

Beim Ausführen des Befehls speichert IndraLogic die Werte der selektierten Rezeptur in eine Datei. Das Format ergibt sich aus den Einstellungen des Rezepturverwalters in der Registerkarte **Speicherung**.

Siehe auch

- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)

- [Kap. 18.2.14 "Menü 'Rezepturen'" auf Seite 723](#)

Befehl 'Rezeptur lesen'

Symbol: 

Funktion: Der Befehl liest die Werte der Variablen einer Rezeptur aus der Steuerung.

Aufruf: Menü **VI-Logic Rezepturdefinition** oder Kontextmenü

Voraussetzung: Die Applikation ist im Online-Betrieb und Sie haben den Wert einer Rezeptur in der Rezepturdefinition selektiert.

Beim Ausführen des Befehls überschreibt IndraLogic die Werte der selektierten Rezeptur mit den gelesenen Werten aus der Steuerung.



Falls Sie die Option **Rezepturverwaltung in der SPS (Rezepturverwalter ► Allgemein)** aktiviert haben, beachten Sie bitte Folgendes:

Wenn Sie Rezepturen im Projekt durch den Befehl **Rezeptur laden** oder **Rezeptur lesen** ändern, wird bei einem erneuten Einloggen ein Online Change notwendig.

Siehe auch

- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)
- [Kap. 18.2 "Menübefehle" auf Seite 610](#)

Befehl 'Rezeptur schreiben'

Symbol: 

Funktion: Der Befehl schreibt die Werte einer Rezeptur in die Variablen in der Steuerung.

Aufruf: Menü **VI-Logic Rezepturdefinition** oder Kontextmenü

Voraussetzung: Die Applikation ist im Online-Betrieb und Sie haben den Wert einer Rezeptur in der Rezepturdefinition selektiert.

Beim Ausführen des Befehls überschreibt IndraLogic die Werte in der Steuerung mit den Werten der selektierten Rezeptur.

Siehe auch

- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)
- [Kap. 18.2.14 "Menü 'Rezepturen'" auf Seite 723](#)

Befehl 'Rezeptur laden und schreiben'

Symbol: 

Funktion: Der Befehl lädt eine Rezeptur aus einer Datei und schreibt die Werte auf die Variablen in der Steuerung.

Aufruf: Menü **VI-Logic Rezepturdefinition** oder Kontextmenü

Voraussetzung: Die Applikation ist im Online-Betrieb. Sie haben den Wert einer Rezeptur in der Rezepturdefinition selektiert.

Nach dem Ausführen des Befehls werden Sie gefragt, ob die Werte aus der Datei auch in die Rezeptur im Projekt, oder nur zur SPS geschrieben werden sollen. Ein Aktualisieren der Werte in der Rezeptur kann bei erneutem Einloggen einen Online Change notwendig machen.

Wenn Sie den Befehl ausführen, überschreibt IndraLogic die Werte der selektierten Rezeptur der Rezepturdefinition. Weiterhin werden die Werte der Variablen in der Steuerung mit diesen Rezepturwerten überschrieben.



Wenn Sie nur einzelne Variablen der Rezeptur mit neuen Werten überschreiben möchten, entfernen Sie vor dem Laden in der Rezepturdatei die Werte für die restlichen Variablen. Einträge ohne Wertangabe werden nicht eingelesen und somit bleiben diese Variablen auf der Steuerung und im Projekt durch die Aktualisierung unberührt.

Bei Werten vom Datentyp `REAL/LREAL` wird in manchen Fällen auch der Hexadezimalwert in die Rezepturdatei geschrieben. Dies ist notwendig, damit bei der Rückkonvertierung der exakt identische Wert wiederhergestellt wird. In diesem Fall ändern Sie den Dezimalwert und löschen Sie den Hexadezimalwert.

Siehe auch

- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)
- [Kap. 18.2.14 "Menü 'Rezepturen'" auf Seite 723](#)

Befehl 'Rezeptur lesen und speichern'

Symbol:

Funktion: Der Befehl liest die Werte der Variablen einer Rezeptur aus der Steuerung und speichert sie in eine Datei.

Aufruf: Menü **VI-Logic Rezepturdefinition** oder Kontextmenü

Voraussetzung: Die Applikation ist im Online-Betrieb und Sie haben den Wert einer Rezeptur in der Rezepturdefinition selektiert.

Nach dem Ausführen des Befehls werden Sie gefragt, ob die Variablenwerte in die Rezeptur im Projekt eingelesen oder nur gespeichert werden sollen. Ein Aktualisieren der Werte in der Rezeptur könnte bei erneutem Einloggen einen Online Change notwendig machen.

Die Werte werden gemäß den Einstellungen des Rezepturverwalters (Registerkarte **Speicherung**) mit dem Standardnamen für Rezepturdateien abgespeichert.

Siehe auch

- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)
- [Kap. 18.2.14 "Menü 'Rezepturen'" auf Seite 723](#)

Befehl 'Variable entfernen'

Symbol:

Funktion: Der Befehl entfernt in einer "Rezepturdefinition" die selektierte(n) Variablen.

Aufruf: Menü **Rezepturen** nach Übernahme aus **Extras ► Anpassen**, Befehlskategorie "Rezeptur"

Voraussetzung: Es ist eine Rezepturdefinition im Editor geöffnet ist und die normale Ansicht gewählt

Siehe auch

- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)
- [Kap. 18.2.14 "Menü 'Rezepturen'" auf Seite 723](#)

Befehl 'Strukturierte Variablen aktualisieren'

Symbol: 

Funktion: Der Befehl öffnet den Dialog **Strukturierte Variablen aktualisieren**.

Aufruf: Menü **VI-Logic Rezepturdefinition** oder Kontextmenü

In dem Dialog können Sie Rezepturdefinitionen aktualisieren, wenn sich die Deklaration einer strukturierten Variablen oder eines Bausteins geändert hat. Wenn beispielsweise die Dimension eines Arrays verändert wurde, können Sie die Einträge in der Rezepturdefinition automatisch entsprechend entfernen oder hinzufügen.

Nicht existierende Variablen entfernen	 : Variablen, die es auf Grund der Änderung eines strukturieren Elements im Projekt nicht länger gibt, werden aus der Rezepturdefinition entfernt.
Instanzen von Strukturen und Funktionsbausteinen aktualisieren	 : Wenn die Deklaration einer Struktur oder eines Funktionsbausteins erweitert wird, die/der mit einer Instanz in der Rezepturdefinition vertreten ist, werden der Rezepturdefinition die entsprechenden Variablen hinzugefügt.
Array-Dimensionen von Array-Instanzen aktualisieren	 : Wenn die Dimension eines Arrays erweitert wird, das mit einer Instanz in der Rezepturdefinition vertreten ist, werden der Rezepturdefinition die entsprechenden Variablen hinzugefügt.
Enthaltene globale Variablenlisten aktualisieren	 : Wenn die Deklaration einer globalen Variablenliste erweitert wird, die mit einer Instanz in der Rezepturdefinition vertreten sind, werden der Rezepturdefinition die entsprechenden Variablen hinzugefügt.
Enthaltene Programme aktualisieren	 : Wenn die Deklaration eines Programms erweitert wird, das in der Rezepturdefinition instanziiert wurde, werden der Rezepturdefinition die entsprechenden Variablen hinzugefügt.

Tab. 18-15: Dialog 'Strukturierte Variablen aktualisieren'

Siehe auch

- [Kap. 13.2 "Werte ändern mit Rezepturen" auf Seite 212](#)
- [Kap. 18.2.14 "Menü 'Rezepturen'" auf Seite 723](#)

18.2.15 Menü 'Textliste'

Befehl 'Sprache einfügen'

Symbol: 

Funktion: Der Befehl fügt eine weitere Sprachspalte der Textliste hinzu.

Aufruf: Menü **VI-Logic Textliste** oder Kontextmenü

Voraussetzung: Eine Textliste oder eine globale Textliste ist geöffnet und aktiv.

Geben Sie im Dialog **Sprache eingeben** ein Kürzel für die neue Sprache ein, zum Beispiel **en-US**. IndraLogic fügt das Kürzel als Spaltenüberschrift ein.

Befehl 'Globale Textliste anlegen'

Symbol: 

Aufruf: Menü **VI-Logic Visualisierung**, Kontextmenü

Voraussetzungen: Eine Visualisierung ist geöffnet.

Funktion: Das Objekt "GlobalTextList" wird automatisch im Ordner "Allgemeiner Baustein" angelegt, sobald beim Konfigurieren einer Visualisierung der erste Text definiert wird.

Der Menüpunkt "Globale Textliste anlegen" ist auch im Hauptmenü auf **VI-LogicVisualisierung** ► **Globale Textliste anlegen** verfügbar.

Alternativ ist der Menüpunkt im Kontextmenü verfügbar, wenn sich der Cursor im Arbeitsbereich des Visualisierungseditors befindet.

Siehe auch

- [Kap. 18.1.10 "Objekt 'GlobalTextList'" auf Seite 543](#)

Befehl 'Alles als Text exportieren'

Symbol: 

Funktion: Der Befehl exportiert alle Textlisten des Projekts.

Aufruf: Menü **Textliste**, Kontextmenü

Voraussetzung

- Eine Textliste oder eine globale Textliste ist geöffnet und aktiv.
- Die Visualisierung codiert die Zeichen der Texte nicht in Unicode.

IndraLogic erstellt dabei für jede Textliste eine Datei als einfachen Text im Format `.txt`. Der Name der Textliste wird zum Namen der Datei.

C:\ProgramData\IW-Projects\0\Project\IndraLogic\globaltextlist.txt bzw. C:\ProgramData\IW-Projects\0\Project\IndraLogic\application.textlist.txt

Eine Steuerung kann dieses Format lesen und verwenden. Sie können die Datei zum Beispiel auf eine Steuerung kopieren und über eine Einstellung im Visualisierungsmanager konfigurieren, dass beim Laden der Applikation die Textlisten nicht nochmals übertragen werden.

Befehl 'Alles als Unicode-Text exportieren'

Symbol: 

Funktion: Der Befehl exportiert alle Textlisten des Projekts.

Aufruf: Menü **Textliste**, Kontextmenü

Voraussetzung:

- Eine Textliste oder eine globale Textliste ist geöffnet und aktiv.
- Die Visualisierung codiert die Zeichen der Texte in Unicode.
 - Die Option **Unicode Zeichenfolge** im Visualisierungsmanager ist aktiviert.
 - Die Compiler-Anweisung `VISU_USEWSTRING` der Applikation ist gesetzt. Überprüfen Sie das, indem Sie den Befehl **Eigenschaften** im Kontextmenü der Applikation wählen. Wählen Sie dann die Registerkarte **Übersetzen**. Im Eingabefeld zu **Compiler-Definitionen** ist `VISU_USEWSTRING` eingetragen.

IndraLogic erstellt dabei für jede Textliste eine Datei als einfachen Text im Format `.txt`.

Der Name der Textliste wird zum Namen der Datei. Das Verzeichnis, in das die Dateien exportiert werden, ist in **Projekt** ► **Projekteinstellungen** ► **Visualisierung**, Kategorie **Allgemein** in **Textlistendateien** eingestellt.

C:\ProgramData\IW-Projects\0\Project\IndraLogic\globaltextlist.txt bzw. C:\ProgramData\IW-Projects\0\Project\IndraLogic\application.textlist.txt

Eine Steuerung kann dieses Format lesen und verwenden. Sie können die Datei zum Beispiel auf eine Steuerung kopieren und über eine Einstellung im Visualisierungsmanager konfigurieren, dass beim Laden der Applikation die Textlisten nicht nochmals übertragen werden.

Siehe auch

- [Kap. 9.8 "Text in einer Textliste verwalten" auf Seite 117](#)

Befehl 'Text einfügen'

Symbol: 

Funktion: Der Befehl fügt eine neue Zeile oberhalb der selektieren Zeile in die Textliste ein. Ein Eingabefeld unter **Standard** öffnet sich, in dem Sie den Ausgangstext eingeben.

Aufruf: Menü **Textliste** ► **Text einfügen**, Kontextmenü

Voraussetzung: Eine Textliste, keine **GlobalTextList**, ist geöffnet und aktiv. Ein Feld in der Tabelle ist selektiert.

Siehe auch

- [Kap. 9.8 "Text in einer Textliste verwalten" auf Seite 117](#)

Befehl 'Import/Export Textlisten'

Symbol: 

Funktion: Der Befehl exportiert eine aktive Textliste, importiert eine Datei oder gleicht eine Textlisten mit einer Datei ab. Die Datei hat CSV-Format. Der Dialog **Import/Export** bietet dafür Optionen an.

Aufruf: Menü **VI-LogicTextliste** ► **Import/Export Textlisten**, Kontextmenü

Voraussetzung: Eine Textliste oder eine globale Textliste ist geöffnet und aktiv.

Dialog 'Import/Export'

Datei für Vergleich oder Import auswählen	Datei, die IndraLogic ausliest.  öffnet den Dialog Textlisten-Datei auswählen, in dem Sie eine Datei auswählen können.
Exportdatei auswählen	Datei, in die IndraLogic schreibt.  öffnet den Dialog Textlisten-Datei auswählen, in dem Sie ein Datei und Verzeichnis auswählen können.

Import/Export-Typ

Importieren	Voraussetzung: In Datei für Vergleich oder Import auswählen ist eine Datei ausgewählt. Die Datei kann Textlisteneinträge sowohl für die globale Textliste als auch für Textlisten enthalten. <i>Globale Textliste</i> - IndraLogic liest die Datei, vergleicht für gleichen Ausgangstext die Textlisteneinträge und übernimmt Unterschiede bei den Übersetzungen. IndraLogic überschreibt gegebenenfalls die Übersetzungen im Projekt. <i>Textlisten</i> - IndraLogic liest die Datei, vergleicht für gleiche IDs die Textlisteneinträge und übernimmt Unterschiede im Ausgangstext und den Übersetzungen in das Projekt. IndraLogic überschreibt gegebenenfalls die Textlisteneinträge im Projekt. - Wenn die Datei eine neue ID enthält, wird der Textlisteneintrag in die Textliste des Projekts importiert und die Textliste ergänzt.
Ersetzungsdatei importieren	Voraussetzung: In Datei für Vergleich oder Import auswählen ist eine Ersetzungsdatei ausgewählt Die Ersetzungsdatei enthält Ersetzungen für die globale Textliste. IndraLogic arbeitet die Ersetzungsdatei zeilenweise ab und führt die spezifizierten Ersetzungen in der globalen Textliste durch.
Exportieren	Voraussetzung: In Exportdatei auswählen ist die Datei, in die IndraLogic schreibt, ausgewählt. IndraLogic exportiert alle Texte aus allen Textlisten des aktuellen Projekts. In der Exportdatei werden alle im Projekt vorhandenen Sprachen als Spalten eingefügt. Die Datei kann dazu verwendet werden die sprachenabhängigen Texte extern übersetzen zu lassen.
Nur Textunterschiede exportieren	Voraussetzung: In Datei für Vergleich oder Import auswählen ist eine Importdatei für den Vergleich ausgewählt, In Exportdatei auswählen ist eine Exportdatei, in die IndraLogic schreibt, ausgewählt. IndraLogic liest die Importdatei, vergleicht die Zeilen der aktiven Textliste damit. Wenn Zeilen übereinstimmen, ignoriert IndraLogic diese. Wenn Zeilen sich unterscheiden, schreibt IndraLogic die Zeile in die Exportdatei und übernimmt dabei gegebenenfalls Übersetzungen aus der Textliste. IndraLogic übernimmt die Übersetzungen aus der Importdatei und überschreibt Sie gegebenenfalls.

Siehe auch

- [Kap. 9.8 "Text in einer Textliste verwalten" auf Seite 117](#)

Befehl 'Sprache entfernen'

Symbol: 

Funktion: Entfernt die ausgewählte Sprachspalte aus der Textliste.

Aufruf: Menü **VI-Logic Textliste** oder Kontextmenü

Voraussetzung: Eine Textliste oder eine globale Textliste ist geöffnet und aktiv. Ein Feld in der Spalte der Sprache, die Sie entfernen wollen, ist selektiert.

Siehe auch

- [Kap. 9.8 "Text in einer Textliste verwalten" auf Seite 117](#)

Befehl 'Nicht verwendete Textlisteneinträge entfernen'

Symbol: 

Funktion: Der Befehl prüft, ob ein Textlisteneintrag im Projekt als statischer Text verwendet wird. Wenn nicht, entfernt IndraLogic ihn aus der Textliste.

Aufruf: Menü **VI-LogicTextliste** oder Kontextmenü

Voraussetzung: Die **GlobalTextList** ist geöffnet und aktiv. Ein Feld in der Tabelle ist selektiert.

Siehe auch

- [Kap. 9.8 "Text in einer Textliste verwalten" auf Seite 117](#)

Befehl 'Visualisierungs-Text-IDs prüfen'

Symbol: 

Funktion: Der Befehl prüft, ob die ID eines Textlisteneintrags im Projekt korrekt ist und meldet das Ergebnis.

Aufruf: **VI_LogicTextliste** oder Kontextmenü

Voraussetzung: Die **GlobalTextList** ist geöffnet und aktiv. Ein Feld in der Tabelle ist selektiert.

Wenn IndraLogic beim Prüfen feststellt, dass die globale Textliste und die statischen Texte der Visualisierungen nicht übereinstimmen, kann das daran liegen, dass die globale Textliste schreibgeschützt ist oder war.

Siehe auch

- [Kap. 9.8 "Text in einer Textliste verwalten" auf Seite 117](#)

Befehl 'Visualisierungs-Text-IDs aktualisieren'

Symbol: 

Funktion: Der Befehl aktualisiert alle inkonsistenten IDs in einer statischen Textliste.

Aufruf: **Textliste ► Text einfügen** Menü, Kontextmenü

Voraussetzung: Die **GlobalTextList** ist geöffnet und aktiv. Ein Feld in der Tabelle ist selektiert. Das Objekt ist schreibgeschützt.

Wenn IndraLogic beim Prüfen feststellt, dass die globale Textliste und die statischen Texte der Visualisierungen nicht übereinstimmen, kann das daran liegen, dass die globale Textliste unter Schreibschutz steht oder stand.

Siehe auch

- [Kap. 9.8 "Text in einer Textliste verwalten" auf Seite 117](#)

Befehl 'Textlistenunterstützung hinzufügen'

Symbol: 

Funktion: Der Befehl fügt dem selektierten DUT-Objekt des Typs Enumeration eine Textlistenunterstützung hinzu.

Aufruf: Kontextmenü eines Standard-DUT-Objekts vom Typ Enumeration

(Symbol: )

Die Textlistenunterstützung ermöglicht eine Lokalisierung der Enumerationskomponentenbezeichner und eine Darstellung des symbolischen Komponentenwerts in einer Textausgabe in der Visualisierung.

Siehe auch

- [Kap. 18.1.8 "Objekt 'DUT'" auf Seite 517](#)
- [Kap. "Befehl 'Textlistenunterstützung entfernen'" auf Seite 732](#)

Befehl 'Textlistenunterstützung entfernen'

Symbol: 

Funktion: Der Befehl entfernt die Textlistenunterstützung vom selektierten Enumerationsobjekt.

Aufruf: Kontextmenü eines Objekts einer textlistenunterstützten Enumeration (Symbol: )

Die Textlistenunterstützung ermöglicht eine Lokalisierung der Enumerationskomponentenbezeichner und eine Darstellung des symbolischen Komponentenwerts in einer Textausgabe in der Visualisierung.

Siehe auch

- [Kap. 18.1.8 "Objekt 'DUT'" auf Seite 517](#)
- [Kap. "Befehl 'Textlistenunterstützung hinzufügen'" auf Seite 731](#)

18.2.16 Menü 'Trace'

Befehl 'Variable hinzufügen'

Funktion: Der Befehl fügt eine Tracevariable zur Konfiguration hinzu.

Aufruf: Menü **Trace**

In der Baumansicht der Tracekonfiguration wird eine neue Variable dargestellt. Rechts erscheint die zugehörige Variablenkonfiguration unter **Variableneinstellungen**.

Siehe auch

- ["Teildialog "Variableneinstellung" auf Seite 786](#)
- [Kap. 13.3.2 "Tracekonfiguration erstellen" auf Seite 219](#)

Befehl 'Automatisch skalieren'

Symbol: 

Funktion: Der Befehl skaliert die Y-Achse der Tracediagramme so, dass alle Graphen optimal angezeigt werden und die Y-Werte in den sichtbaren Bereich der Diagramme passen. Der Befehl wirkt sowohl auf eine einkanalige als auch auf eine mehrkanalige Darstellung.

Aufruf: Menü **Trace**, Kontextmenü

Trace in Einkanalardarstellung

Wenn alle Tracevariablen in einem Diagramm angezeigt werden, ist der Trace in Einkanalardarstellung.

Trace in Mehrkanalardarstellung

Wenn die Tracevariablen in mehreren Diagramm angezeigt werden, ist der Trace in Mehrkanalardarstellung.

Siehe auch

- [Kap. "Dialog 'Tracekonfiguration'" auf Seite 783](#)

Siehe auch

- [Kap. 13.3.5 "Navigieren in den Tracedaten" auf Seite 223](#)

Befehl 'Komprimieren'

Symbol: 

Funktion: Der Befehl komprimiert den Tracegraphen, indem er den angezeigten Zeitbereich um einen festen Prozentsatz vergrößert.

Aufruf: Menü **Trace**, Kontextmenü

Siehe auch

- [Kap. "Befehl 'Strecken'" auf Seite 739](#)
- [Kap. 13.3.5 "Navigieren in den Tracedaten" auf Seite 223](#)

Befehl 'Konfiguration'

Funktion: Der Befehl öffnet den Dialog **Tracekonfiguration**, der die Konfiguration der Datenerfassung ermöglicht.

Aufruf: Menü **Trace**, Kontextmenü

Siehe auch

- [Kap. "Dialog 'Tracekonfiguration'" auf Seite 783](#)
- ["Teildialog 'Variableneinstellung'" auf Seite 786](#)

Befehl 'Cursor'

Symbol: 

Funktion: Der Befehl

- fügt einen Tracecursor ins Tracediagramm ein, wenn **kein** Tracecursor vorhanden ist
- fügt den 2-ten Tracecursor ins Tracediagramm ein, wenn **1** Tracecursor vorhanden ist
- entfernt die Tracecursor, wenn **2** Tracecursor vorhanden sind

Aufruf: Menü **Trace**, Kontextmenü

Ein Tracecursor ist ein kleines schwarzes Dreieck mit vertikaler schwarzer Linie, die parallel zur Y-Achse verläuft.

Tracediagramm ohne Tracecursor

In diesem Zustand können Sie mit dem Cursor, also dem Mauszeiger, das Tracediagramm ablaufen. Dabei wird der X-Wert, den Sie mit dem Cursor fokussieren, in der Statusleiste in normierter Schreibweise ausgegeben.

Beispiel: **Zeit: 1m23s456ms Wert: 1**

Tracediagramm mit einem Tracecursor

IndraLogic gibt den Zeitpunkt, den der Tracecursor markiert, in der Statusleiste und den Y-Wert aus.

Beispiel: **Zeit: 1m23s456ms**

Tracediagramm mit 2 Tracecursor

IndraLogic gibt die 2 Zeitpunkte und das Zeitintervall, das die 2 Tracecursor markieren, in der Statusleiste aus.

Beispiel: **Zeit: 1m23s456ms - Zeit: 1m24s456ms (Δ 1s)**

Benutzereingabe im Tracediagramm

Wenn 1 oder 2 Tracecursor vorhanden sind, können Sie diese entlang der X-Achse verschieben.

Benutzereingabe mit der Maus	Symbol	Auswirkung
Ziehen Sie das Dreieck eines Tracecursors auf eine andere Position		Solange die Maustaste gedrückt ist, kann der Cursor frei verschoben werden. In der Statusleiste erscheint immer der aktuelle Y-Wert. Sobald die Maustaste losgelassen wird, rastet der Cursor am nächstgelegenen Messpunkt ein.

Benutzereingabe mit Tastaturkürzel	Auswirkung
<Pfeil nach links> <Pfeil nach rechts>	IndraLogic verschiebt den schwarzen Tracecursor zum nächsten Messpunkt.
<Umschalttaste> + <Pfeil nach links> <Umschalttaste> + <Pfeil nach rechts>	IndraLogic verschiebt den grauen Tracecursor zum nächsten Messpunkt.

Siehe auch

- [Kap. 13.3.5 "Navigieren in den Tracedaten" auf Seite 223](#)

Befehl 'Download Trace'

Symbol: 

Funktion: Der Befehl überträgt die Tracekonfiguration auf die Steuerung an die zugeordnete Applikation und startet die Datenerfassung. Die erfassten Daten werden an das Entwicklungssystem zurück übertragen. Das Tracediagramm zeigt dann die aktuellen Samples an und läuft kontinuierlich weiter.

Aufruf: Menü **Trace**, Kontextmenü

Voraussetzung: Der Befehl ist verfügbar, wenn die zugeordnete Applikation im Onlinebetrieb ist.

Siehe auch

- [Kap. 13.3.3 "Datenerfassung steuern" auf Seite 223](#)

Befehl 'Symbolische Tracekonfiguration exportieren'

Funktion: Der Befehl exportiert eine Tracekonfiguration als Datei mit Format *.traceconfig.

Aufruf: Menü **Trace**, Kontextmenü

Voraussetzung: Die Herkunftapplikation enthält eine Symbolkonfiguration, die die konfigurierten Tracevariablen als Symbole definiert. Der Zugriff auf die IEC-Variablen, deren Daten erfasst werden, ist damit symbolisch und ermöglicht die Verwendung der Tracekonfiguration für verschiedene, ähnliche Applikationen.

Verwendung der Konfigurationsdatei

Sie können diese Datei auf ein beliebiges Laufzeitsystem übertragen. Zur Laufzeit kann dessen Laufzeitsystemkomponente `CmpTraceMgr` darauf zugreifen und darauf basierend eine Datenerfassung durchführen. Die Konfigurationsdatei enthält aber neben den Konfigurationsdaten zusätzlich Information über den Kontext der Applikation.

Die Konfigurationsdatei definiert folgenden Kontext

- Name der Applikation
- Name des Traces
- Name der Task

Die Applikation, die zur Laufzeit ausgeführt wird, muss folgende Bedingungen erfüllen

- Die Applikation ist namensgleich mit der Herkunftapplikation
- Der Trace, der in der Applikation konfiguriert ist, ist namensgleich mit dem Trace, der in der Herkunftapplikation konfiguriert ist
- Die Task, in der die Datenerfassung läuft, ist namensgleich mit der Task, die in der Herkunftapplikation konfiguriert ist



Das Laden der Konfiguration erfolgt nicht automatisch, sondern Sie müssen es befehlen.

Sie können folgendermaßen vorgehen

- Sie greifen über IEC-Code programmatisch auf den Tracemanager zu, indem Sie Bibliotheksschnittstellen verwenden
- Sie registrieren die Konfigurationsdatei beim Tracemanager. Dann lädt der Tracemanager die Konfigurationsdatei beim Start der Applikation.

Beispiel einer Konfigurationsdatei

Konfigurationsdatei *Trace_getting_started.traceconfig*

```
[key]; [value]
Version; 0x03050000
Name; Application.Trace
ApplicationName; Application
ApplicationDataGuid; 00000000-0000-0000-0000-000000000000
IecTaskName; PlcTask
Comment;
Trigger.Flags; 0
Condition.Name;
Condition.AddrFlags; 0x00000000
Condition.Class; 0
Condition.Size; 0
EveryNCycles; 1
BufferEntries; 81
Flags; 16
O.Variable; PlcProg.rSin
O.Address.AddrFlags; 0x00000101
O.Class; 14
O.Size; 4
O.GraphColor; 4278190335
O.GraphType; 1
O.MinWarningColor; 4278190080
O.MaxWarningColor; 4294901760
O.CriticalLowerLimit; 0
O.CriticalUpperLimit; 0
O.ActivateMinWarning; 0
O.ActivateMaxWarning; 0
O.YAxis; 0
O.Data;
```

Siehe auch

- [Kap. 13.3.6 "Trace verwalten" auf Seite 223](#)

Befehl 'Trace laden'

Funktion: Der Befehl ermöglicht das Laden einer Datei, die Konfiguration und Daten enthält und auf dem Dateisystem des Entwicklungssystems gespeichert wurde. Der Dialog **Trace laden** erscheint.

Aufruf: Menü **Trace**, Kontextmenü

Dialog 'Trace laden'

Dateiname	Name der Datei, die geladen wird
Dateityp	<i>Dateiformat</i> • *.trace: Tracedatei, die Tracekonfiguration und Samples enthält • *.csv: Textdatei im CSV-Dateiformat, die eine Tracekonfiguration enthält

Siehe auch

- [Kap. "Befehl 'Trace speichern'" auf Seite 738](#)
- [Kap. 13.3.6 "Trace verwalten" auf Seite 223](#)

Befehl 'Zoomen mit Maus'

Symbol:  (Befehl deaktiviert),  (Befehl aktiviert).

Funktion: Der Befehl aktiviert oder deaktiviert das Zoomen mit Maus im Tracediagramm.

Aufruf: Menü **Trace**, Kontextmenü

Benutzereingabe im Tracediagramm

Wenn der Befehl aktiviert ist, können Sie mit der Maus ein Rechteck aufziehen. Wenn Sie die Maustaste loslassen, wird in das Rechteck gezoomt und die Daten werden vergrößert dargestellt.

Siehe auch

- [Kap. "Dialog 'Tracekonfiguration'" auf Seite 783](#)
- [Kap. 13.3.5 "Navigieren in den Tracedaten" auf Seite 223](#)

Befehl 'in Mehrkanal konvertieren'

Funktion: Der Befehl schaltet die Darstellung im Tracediagramm zwischen einkanalig und mehrkanalig um.

Aufruf: Menü **Trace**, Kontextmenü

Symbol: ☐ für einkanalige Darstellung, ☒ für Mehrkanal-Darstellung

Voraussetzung: Es müssen mindestens zwei Tracevariablen in der Tracekonfiguration definiert sein.

Mehrkanaldarstellung

Mehrkanaldarstellung bedeutet, dass die Tracevariablen in mehreren Diagrammen dargestellt werden.

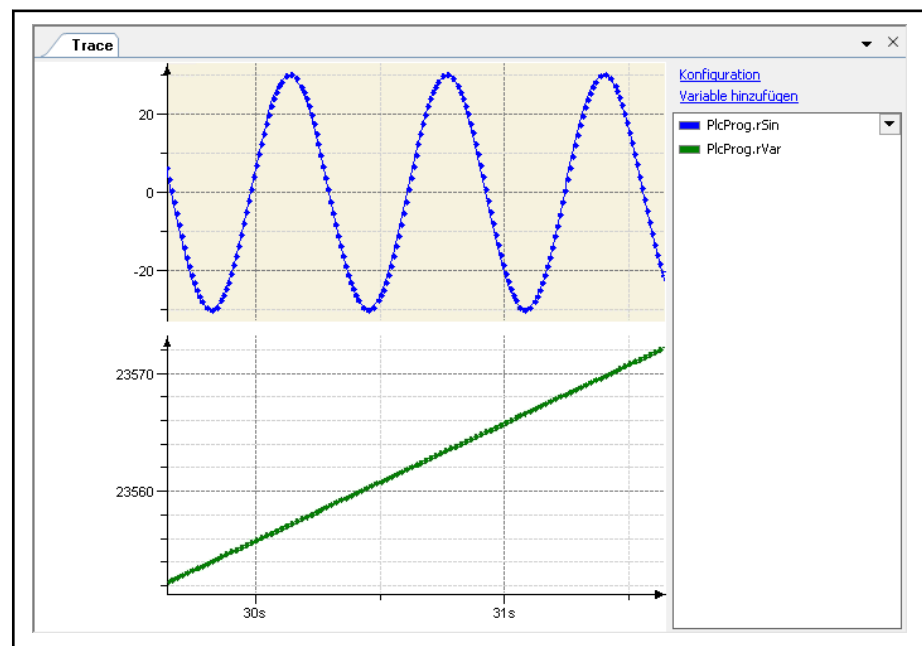


Abb. 18-71:

Siehe auch:

- [Kap. "Dialog 'Tracekonfiguration'" auf Seite 783](#)
- [Kap. 13.3.5 "Navigieren in den Tracedaten" auf Seite 223](#)

Befehl 'in Einkanal konvertieren'

Funktion: Der Befehl schaltet die Darstellung im Traceeditor von Mehrkanal- auf Einkanaldarstellung um.

Aufruf: Menü **Trace**, Kontextmenü

Symbol: ☒ (einkanalige Darstellung), ☐ (Mehrkanal-Darstellung).

Einkanaldarstellung

Wenn ein Trace einkanalig dargestellt wird, dann sind alle Tracevariablen in einem Diagramm enthalten.

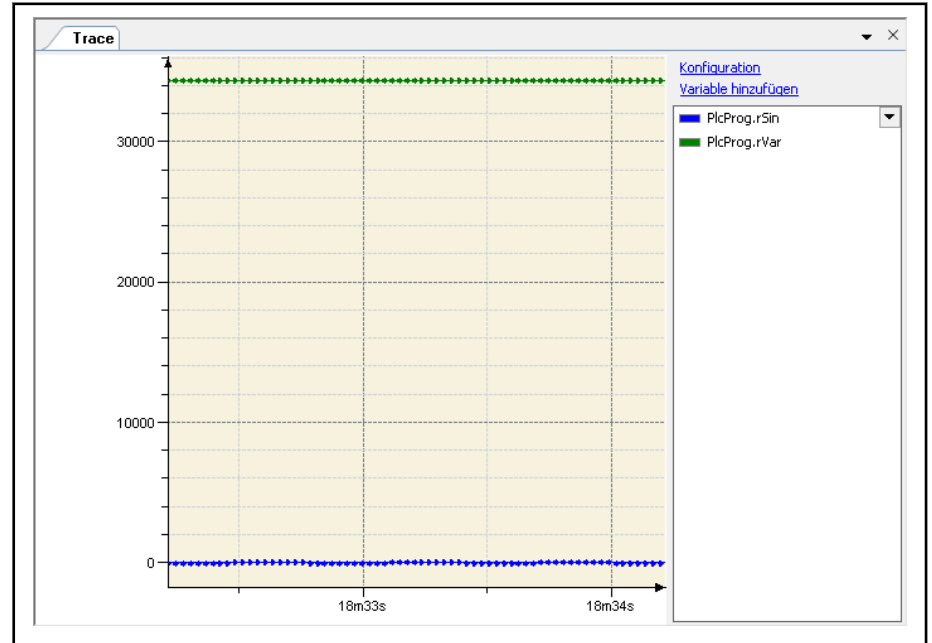


Abb. 18-72:

Siehe:

- [Kap. "Dialog 'Tracekonfiguration'" auf Seite 783](#)
- [Kap. 13.3.5 "Navigieren in den Tracedaten" auf Seite 223](#)

Befehl 'Onlineliste'

Funktion: Der Befehl öffnet den Dialog "Onlineliste". Wenn gerade der Traceeditor eines DeviceTrace-Objekts aktiv ist, werden alle Traces, die auf der Steuerung laufen, in einer Baumansicht angezeigt. Wenn gerade der Traceeditor eines applikationsbezogenen Traceobjekts aktiv ist, wird nur dieser Trace angezeigt, falls er auf der Steuerung läuft.

Aufruf: Menü **Trace**, alternativ im Kontextmenü des Traceeditors

Voraussetzungen: Das Laufzeitsystem verwendet die Komponente `CmpTraceMgr`. Eine dem Gerät zugehörige Applikation ist im Onlinebetrieb.



Das Schließen des DeviceTrace-Editors beendet die Verbindung zur Steuerung!

Beachten Sie ebenfalls, dass mit dem Schließen des letzten geöffneten DeviceTrace-Editors auch die Verbindung zur Steuerung beendet wird.

Um die Geräte-Traces wieder im Projekt dargestellt zu bekommen, müssen Sie die Geräte-Traces erneut in die DeviceTrace-Objekte laden.

Derzeit ist das Schließen des Editors das empfohlene Vorgehen, um bewusst die Verbindung zur Steuerung zu beenden. Das Ausloggen genügt nicht, um die Verbindung zur Steuerung zu beenden.

Dialog 'Online-Liste'

Vom LZS entfernen	Stoppt und entfernt den selektierten Trace aus der laufenden Applikation.
Hochladen	Der Befehl ist sichtbar, wenn im Traceeditor ein DeviceTrace geladen ist. Ein DeviceTrace ist ein Trace, der auf der Steuerung läuft: Im Gerätebaum kann er mit einem DeviceTrace-Objekt direkt unterhalb eines Geräts repräsentiert werden. Wenn Sie den Befehl auswählen, wird der in der Baumansicht selektierte Trace vom Laufzeitsystem in den Traceeditor geladen. Eine eventuell im Projekt bestehende Konfiguration wird überschrieben. Beispielsweise kann das Gerät Traces für Daten der Prozessorauslastung (cpuload, plcload) anbieten, die Sie dann im Traceeditor in IndraLogic verfolgen können. Pro Trace des Geräts, der im Projekt dargestellt werden soll, ist ein eigenes DeviceTrace-Objekt im Gerätebaum nötig.

Siehe auch

- [Kap. 13.3.4 "Zugriff auf alle Traces der Steuerung" auf Seite 223](#)
- [Kap. 18.1.27 "Objekt 'Trace'" auf Seite 605](#)
- [Kap. " Befehl 'Trace hochladen'" auf Seite 740](#)

Befehl 'Trigger zurücksetzen'

Symbol: 

Funktion: Der Befehl setzt die Tracekonfiguration nach einer ausgelösten Datenerfassung zurück. Dann kann die Applikation neue Daten erfassen und neu auf einen Trigger reagieren.

Aufruf: Menü **Trace**, Kontextmenü

Voraussetzung: Nach Auslösen eines Triggers sind die Daten vollständig im Puffer des Entwicklungssystems.

Siehe auch

- [Kap. 13.3.3 "Datenerfassung steuern" auf Seite 223](#)

Befehl 'Ansicht zurücksetzen'

Symbol: 

Funktion: Der Befehl setzt die Darstellung im Tracediagramm auf die Voreinstellung zurück.

Aufruf: Menü **Trace**, Kontextmenü

Voraussetzung: Sie haben die Darstellung im Tracediagramm durch Zoomen, Scrollen oder dem Befehl **Automatisch skalieren** geändert.

Siehe auch

- [Kap. "Befehl 'Automatisch skalieren'" auf Seite 732](#)
- [Kap. 13.3.5 "Navigieren in den Tracedaten" auf Seite 223](#)

Befehl 'Trace speichern'

Funktion: Der Befehl ermöglicht das Speichern der Daten in einer Datei auf dem Entwicklungssystems. Abhängig vom Dateiformat wird auch die Konfiguration gespeichert. Der Dialog **Trace speichern** erscheint.

Aufruf: Menü **Trace**, Kontextmenü

Dialog 'Trace speichern'

Dateiname	Name und Speicherort der Tracedatei
Dateityp	<i>Dateityp</i> • *.trace: Tracedatei, die die Daten und die Konfiguration enthält. Sie können so eine Datei mit dem Befehl Trace laden offline in den Traceeditor laden. • *.txt: Textdatei, die die erfassten Daten enthält. Sie können so eine Datei mit Tools, die das CSV-Format unterstützen, laden und verarbeiten. Sie können sie nicht offline in den Traceeditor laden, da der Traceeditor dieses Format nicht lesen kann. • *.trace.csv Trace-csv-Datei (nur Daten), die die erfassten Daten enthält. Für jede Tracevariable ist eine Adressinformation gegeben. Die erzeugte Datei kann im Laufzeitsystem eingelesen werden. Die Daten werden importiert, aber der Trace kann nicht gestartet werden, da die Adressen der Variablen nicht gespeichert werden.

Siehe auch

- [Kap. "Befehl 'Trace laden'" auf Seite 735](#)
- [Kap. 13.3.6 "Trace verwalten" auf Seite 223](#)

Befehl 'Trace starten'

Symbol: 

Funktion: Der Befehl startet die Ausführung der Datenerfassung auf der Steuerung, wenn diese gestoppt ist.

Aufruf: Menü **Trace**, Kontextmenü

Voraussetzung: Die zugeordnete Applikation auf dem Laufzeitsystem läuft und eine Tracekonfiguration ist geladen.

Siehe auch

- [Kap. 13.3.3 "Datenerfassung steuern" auf Seite 223](#)

Befehl 'Trace stoppen'

Symbol: 

Funktion: Der Befehl stoppt die Datenerfassung eines Trace.

Aufruf: Menü **Trace**, Kontextmenü

Voraussetzung: Die zugeordnete Applikation auf dem Laufzeitsystem läuft und führt einen Trace aus.

Siehe auch

- [Kap. 13.3.3 "Datenerfassung steuern" auf Seite 223](#)

Befehl 'Strecken'

Symbol: 

Funktion: Der Befehl streckt den Tracegraphen, indem er den angezeigten Zeitbereich um einen festen Prozentsatz verkleinert.

Aufruf: Menü **Trace**, Kontextmenü

Siehe auch

- [Kap. "Befehl 'Komprimieren'" auf Seite 732](#)
- [Kap. 13.3.5 "Navigieren in den Tracedaten" auf Seite 223](#)

Befehl 'Trace hochladen'

Funktion: Der Befehl stellt zunächst die Verbindung zum SPS-Gerät her, falls diese noch nicht besteht. Dann öffnet er den Dialog Onlineliste mit einer Liste der gerade auf der Steuerung laufenden Traces. Der ausgewählte Trace wird dann mit dem im Dialog verfügbaren Befehl Hochladen in den Trace-Editor geladen.

Aufruf: Menü **Trace**, alternativ im Kontextmenü des Traceeditors

Voraussetzung: Der Editor eines DeviceTrace-Objekts ist geöffnet. Das Laufzeitsystem verwendet die Komponente `CmpTraceMgr` (Tracemanager). Mindestens eine Applikation auf dem Laufzeitsystem ist in Ausführung. Im IndraLogic-Projekt sind die Kommunikationseinstellungen für die SPS korrekt konfiguriert.



Das Schließen des DeviceTrace-Editors beendet die Verbindung zur Steuerung!

Beachten Sie ebenfalls, dass mit dem Schließen des letzten geöffneten DeviceTrace-Editors auch die Verbindung zur Steuerung beendet wird.

Um die Geräte-Traces wieder im Projekt dargestellt zu bekommen, müssen Sie die Geräte-Traces erneut in die DeviceTrace-Objekte laden.

Derzeit ist das Schließen des Editors das empfohlene Vorgehen, um bewusst die Verbindung zur Steuerung zu beenden. Das Ausloggen genügt nicht, um die Verbindung zur Steuerung zu beenden.

Siehe auch

- ["Dialog 'Online-Liste'" auf Seite 738](#)
- [Kap. 13.3.4 "Zugriff auf alle Traces der Steuerung" auf Seite 223](#)

Befehl 'Statistik'

Funktion: Der Befehl öffnet den Dialog **Tracestatistik**, der eine Statistik zu jeder Tracevariablen anzeigt.

Aufruf: Menü **Trace**, Kontextmenü bzw. zum Nachinstallieren: **Extras ► Anpassen ► Befehle ► Trace**

Voraussetzung: Traceeditor enthält Samples.

Dialog 'Tracestatistik'

Im Titel ist der analysierte Zeitbereich und dessen Dauer angegeben.

Die Tabelle enthält für jedes Signal eine Zeile.

Signal	Name der Tracevariablen, zum Beispiel <code>PlcProg.S1</code>
Index	0-basierter Index der Reihenfolge der Signale
N	Anzahl der Messwerte, die den Berechnungen zugrunde liegen
Min	Minimalwert
Avg	Durchschnitt
Median	Wert, der an der mittleren Stelle steht, wenn man die Werte der Größe nach sortiert.

RMS	Quadratisches Mittel
StdDev	Standardabweichung
Max	Maximalwert
Integral	Integral
Min Δt [s]	Minimalwert der Zeitabstände aufeinanderfolgender Werte
Avg Δt [s]	Durchschnitt der Zeitabstände aufeinanderfolgender Werte
Median Δt [s]	Median der Zeitabstände aufeinanderfolgender Werte
StdDev Δt [s]	Standardabweichung der Zeitabstände aufeinanderfolgender Werte
Max Δt [s]	Maximalwert der Zeitabstände aufeinanderfolgender Werte

Klick auf den Tabellenkopf in eine bestimmte Spalte	IndraLogic sortiert die Tabelle nach der Spalte und wechselt die Reihenfolge von aufsteigend nach absteigend und umgekehrt. Voreinstellung: Die Tabelle ist aufsteigend nach der Spalte Index sortiert. Damit sind die Signale in derselben Reihenfolge sortiert, in der sie auch im Signalbaum auftauchen.
Klick in die Zeile	Die Zeile ist selektiert. Sie können mit <Umschalttaste> + <Pfeil> weitere Zeilen selektieren und auch wieder deselektieren.
<Strg> + <C>	IndraLogic kopiert die selektierten Zeilen als Text in die Zwischenablage. Die Werte der einzelnen Spalten sind mit Tabulator getrennt, die Zeilen sind mit dem Steuerzeichen <CR> oder <LF> getrennt. Voraussetzung: Sie haben eine oder mehrere Zeilen selektiert.

Siehe auch

- [Kap. 13.3 "Datenerfassung mit 'Trace'" auf Seite 216](#)

18.2.17 Sonstige

Befehl 'Simulation

Funktion: Das Kommando dient dem Ein- bzw. Ausschalten des Simulationsmodus einer Steuerung.

Aufruf: Kontextmenü der jeweiligen Steuerung

Voraussetzung: Steuerung ist ausgeloggt.

Im Simulationsbetrieb können Sie die aktive Applikation auf einem "simulierten Zielgerät" starten und debuggen.

Sie benötigen kein reales Zielgerät um das Online-Verhalten einer Applikation zu testen.

Beim ersten Einloggen erfolgt eine Abfrage, ob die Applikation angelegt und geladen werden soll.

Für ein simuliertes Gerät müssen Sie keine Kommunikationseinstellungen vornehmen.

Der Simulationsbetrieb wird in der Statuszeile durch den Eintrag "Simulation" sichtbar gemacht.



Verwenden Sie bitte die PlcTask (zyklische Task) und Programme, die dieser Task zugeordnet sind.

Nach erfolgreichem Einloggen signalisiert die IndraWorks-Statuszeile im Projektextplorer den Simulationsbetrieb. Sie können die entsprechenden Online-Kommandos benutzen, um die Applikation zu testen.

Um den Simulationsbetrieb auszuschalten loggen Sie sich zunächst aus der Steuerung aus und rufen danach erneut den Befehl **Simulation** auf.

Der Befehl betrifft immer die aktive Applikation.



Für die Peripherie der Steuerung werden zur Zeit keine Treiber bereit gestellt.

Befehl 'Zu Überwachungsliste hinzufügen'

Symbol:

Funktion: Der Befehl fügt die Variable, auf der gerade der Cursor steht, einer Überwachen-Liste hinzu, die dem Online-Monitoring dient.

Aufruf: Der Befehl ist im Kontextmenü verfügbar, wenn sich die Applikation im Online-Betrieb befindet und der Cursor in einem Editor auf einer Variable steht.

Der Befehl fügt die Variable in die gerade geöffnete Überwachen-Liste ein.

Wenn gerade keine Überwachen-Liste geöffnet ist, fügt der Befehl die Variable in Liste **Überwachen 1** ein und öffnet deren Ansicht.

Siehe auch

- [Kap. 13.1.2 "Überwachungslisten verwenden" auf Seite 211](#)
- [Kap. 13.1 "Monitoring von Werten" auf Seite 205](#)

Befehl 'Schnittstellen implementieren'

Funktion: Der Befehl aktualisiert für einen Funktionsbaustein die implementierten Schnittstellen.

Aufruf: Kontextmenü, wenn der Funktionsbaustein (FB) im Projektextplorer selektiert ist.

Voraussetzung: Der Funktionsbaustein implementiert eine Schnittstelle, die Sie verändert haben. Beispielsweise haben Sie der Schnittstelle eine weitere Methode hinzugefügt.



Wenn Sie beim objektorientierten Programmieren zum Zweck der Vererbung einen Funktionsbaustein (FB) von einem Basis-Funktionsbaustein ableiten, der eine oder mehrere Schnittstellen implementiert, gilt Folgendes:

Wenn Sie den Befehl "Schnittstellen implementieren" für den abgeleiteten FB ausführen, werden alle Schnittstellenmethoden und Schnittstelleneigenschaften des Basis-FB in Form von "Stubs" (ohne Implementierung) in den abgeleiteten FB übernommen. Sie müssen nun darauf achten, dass im abgeleiteten FB dadurch nicht eine "leere" Methode/ Eigenschaft in Konflikt mit einer im Basis-FB implementierten gerät.

Um Sie dabei zu unterstützen, geschieht Folgendes: Wenn es für eine Methode/ Eigenschaft eine Basisimplementierung gibt, fügt IndraLogic ein Pragmaattribut {error..} in der ersten Zeile der betroffenen abgeleiteten Schnittstellenmethode beziehungsweise Schnittstelleneigenschaft ein, das eine entsprechende Fehlermeldung generieren wird.

Wenn es für die Methode/Eigenschaft keine Basisimplementierung gibt, gibt es einen Pragmaattributeintrag für eine Warnung.

Nach Bearbeitung des Bausteins müssen Sie den error-Pragmaattributeintrag jeweils explizit entfernen.

Siehe auch

- [Kap. "Objekt 'Schnittstelle\(INTERFACE\)' auf Seite 562](#)
- [Kap. 9.18.2 "Schnittstellen implementieren \(IMPLEMENTS\)" auf Seite 156](#)

Befehl 'Ergebnisse auf aktuelle Deklaration beschränken'

Funktion: Wenn mehrere Deklarationen gefunden wurden, reduziert der Befehl die Darstellung in der Querverweisliste. Sie zeigt nur noch die Ergebnisse für die Deklaration, die explizit in der Liste selektiert wurden.

Aufruf: Kontextmenü

Voraussetzung: Die Querverweisliste ist aktiv. Mehrere Deklarationen für das gesuchte Symbol sind als Querverweise aufgelistet.

Siehe auch

- [Kap. 9.12.1 "Verwendungsstellen mit der Querverweisliste finden" auf Seite 135](#)

Befehl 'Ausführung an behandelten Exceptions anhalten'

Funktion: Der Befehl hält die gerade aktive Applikation an der Fehlerstelle trotz eines programmierten Exceptionhandlings an.

Aufruf: Der Befehl ist standardmäßig nicht verfügbar, aber über **Extras ► Anpassen** konfigurierbar. Der Befehl befindet sich dort in Dialog **Anpassen ► Befehle** in Kategorie **Online**.

Voraussetzung: Die Applikation enthält ein programmiertes Exceptionhandling mit den Operatoren `__TRY` und `__CATCH` und ist im Online-Betrieb.

Siehe auch

- [Kap. 17.3.61 "Operator ' __TRY', ' __CATCH', ' __FINALLY', ' __ENDTRY'" auf Seite 353](#)
- [Kap. "Befehl 'Anpassen'" auf Seite 682](#)

18.3 Dialoge

Die Dialoge der IndraLogic-Oberfläche sind grundsätzlich auf den Hilfeseiten der IndraLogic-Menübefehle oder IndraLogic-Objekte beschrieben.

Die Hilfe zu **Dialoge** enthält nur die Beschreibungen für Dialoge, die

- erst durch mehrstufig erfolgende weitere Aufrufe nach einem Menübefehl oder in einem Objekteditor erscheinen,
- oder aufgrund ihrer Komplexität nicht auf einer Befehls- oder Objekthilfeseite mit aufgenommen wurden (viele Unterdialoge).

18.3.1 Dialog 'Funktionsbaustein auswählen'

Funktion: Der Dialog dient der Auswahl eines Funktionsbausteins für E/A-Mapping. Der Funktionsbaustein soll dem auf der Registerkarte <Gerätename> E/A-Abbild selektierten E/A-Kanal, oder dem auf Registerkarte <Gerätename> IEC-Objekte selektierten Objekt zugewiesen werden.

Aufruf: Bei Objekten im Projektexplorer, die im generischen Geräteeditor die folgenden Registerkarten bereitstellen:

- **E/A-Abbild**, über die Befehlsschaltfläche  FB für E/A-Kanal hinzufügen
- **IEC-Objekte**, über die Befehlsschaltfläche  Hinzufügen

Der Dialog bietet alle Funktionsbausteine aus der aktiven Applikation und den im Projekt eingebundenen Bibliotheken an, die Folgendes erfüllen:

- Der Funktionsbaustein ist mit dem Attribut `{attribute 'io_function_block'}` versehen.
- Der Funktionsbaustein enthält Eingangs- oder Ausgangsparameter, die zum Kanaltyp (Eingang, Ausgang, Datentyp) passen und mit dem Attribut `{attribute 'io_function_block_mapping'}` versehen sind.

Wenn ein Funktionsbaustein gewählt wird, der mehrere passende Parameter bietet, wird nur der erste davon automatisch auf den Kanal abgebildet. Die weiteren können nur händisch auf der Registerkarte <Gerätename> E/A-Abbild zugewiesen werden.

Nach Zuweisen des Funktionsbausteins wird der Parameter der Funktionsbausteininstanz in der Spalte Variable der Mapping-Tabelle eingetragen. Der Pfad setzt sich folgendermaßen zusammen:

<Applikationsname>.<Gerätekanalname>_<Name des FBs>_<fortlaufende FB-Instanz-Nummer>.<FB-Parametername>

Beispiel: App1.Out_4_Int_myScale_Output_Int_1.iOutput für den Parameter iOutput der ersten eingefügten Instanz des Funktionsbausteins myScale_Output.

Dialogelemente

Suchen	Eingabefeld für eine Suche nach Bausteinnamen
Type	Zum Kanaltyp passende Funktionsbausteine in Baumstruktur. Knoten: Applikation, Bibliotheksname(n)
Dokumentation	Zeigt für die im Baum selektierte Bibliothek oder den Bibliotheksbaustein die verfügbare Dokumentation an

Siehe auch [Kap. "Attribut 'io_function_block', 'io_function_block_mapping'" auf Seite 418](#)

18.3.2 Dialog 'Eigenschaften Haltepunkt'

Funktion: Der Dialog dient in der Ansicht "Haltepunkte" zur Darstellung oder Änderung der Eigenschaften des gerade selektierten Haltepunkts.

Symbol: 

Aufruf: IndraWorks Menü **Debuggen** ► **Haltepunkt bearbeiten**, oder über das Kontextmenü des Haltepunktes **Haltepunkt bearbeiten**

Voraussetzung: Die Applikation ist im Onlinebetrieb und ein Haltepunkt ist gesetzt.

Der Dialog ist identisch mit dem Dialog "Neuer Haltepunkt", der über die entsprechenden Befehle im Menü Debug geöffnet wird, siehe [Kap. 18.3.4 "Dialog 'Neuer Haltepunkt'" auf Seite 745](#)

18.3.3 Dialog 'Wert vorbereiten'

Funktion: Der Dialog dient zur Vorbereitung eines Werts für eine bereits geforcte Variable.

IndraLogic führt die vorbereitete Aktion mit dem nächsten Forcen aus.

IndraLogic öffnet den Dialog in folgenden Situationen:

- wenn Sie im Deklarationsteil in das Feld **vorbereiteter Wert** einer geforcten Variable klicken
- wenn Sie im Implementierungsteil in das Inline-Monitoring-Feld einer geforcten Variablen klicken
- wenn Sie im Überwachen-Fenster in das Feld **vorbereiteter Wert** einer geforcten Variable klicken

Einen neuen Wert für die nächste Schreib- oder Force-Operation vorbereiten	Wert, den IndraLogic bei der nächsten Force-Operation auf die Variable schreibt
Vorbereitung mit einem Wert aufheben	IndraLogic löscht den vorbereiteten Wert.
Den Force aufheben, ohne den Wert zu verändern.	IndraLogic behält den geforcten Wert bei und beendet das Forcen. IndraLogic markiert die Variable mit <Unforce>.
Den Force aufheben und den Wert auf den Wert zurücksetzen, den er vor der Force-Operation hatte.	IndraLogic setzt den geforcten Wert zurück und beendet das Forcen. Dabei wird die Variable mit <Unforce and restore> markiert.

Siehe auch

- [Kap. "Befehl 'Werte forcen'" auf Seite 671](#)

18.3.4 Dialog 'Neuer Haltepunkt'

Funktion: In dem Dialog definieren Sie die Einstellungen für einen neuen Haltepunkt oder Datenhaltepunkt. Er ist identisch mit dem Dialog Eigenschaften Haltepunkt, der in der Ansicht Haltepunkte verwendet wird.

Symbol: 

Aufruf:

- Über das IndraWorks Menü **Debuggen** ► **Neuer Haltepunkt...**
- Über das Kontextmenü der entsprechenden Zeile im Programmcode **Neuer Haltepunkt...**

Voraussetzung: Die Applikation ist im Onlinebetrieb

Registerkarte "Bedingung"

Der Dialog legt fest, unter welchen Voraussetzungen die Programmabarbeitung am Haltepunkt stoppen soll.



Die Verwendung bedingter Haltepunkte verlangsamt die Codeausführung, auch wenn die Bedingung nicht `TRUE` ist.

Tasks	
Nur anhalten falls der Haltepunkt in einer der folgenden Tasks erreicht wird	 : IndraLogic wertet den Haltepunkt nur aus, wenn er von bestimmten Tasks erreicht wird. Die gewünschten Tasks müssen aktiviert werden. Sie können beispielsweise eine einzige "Debug Task" definieren und damit verhindern, dass beim Debugging auch andere Tasks betroffen werden
Trefferanzahl	
Immer anhalten	Das Programm stoppt immer an diesem Haltepunkt.
Anhalten, wenn die Trefferanzahl gleich	Das Programm hält am Haltepunkt an, wenn der Haltepunkt so oft getroffen wurde, wie Sie hier ausgewählt oder eingetragen haben.
Anhalten, wenn die Trefferanzahl ein Vielfaches von	
Anhalten, wenn die Trefferanzahl größer oder gleich	
Bedingung	
Halt, wenn TRUE	 : IndraLogic wertet die angegebene Bedingung aus und hält das Programm an diesem Haltepunkt nur dann an, wenn das Ergebnis <code>TRUE</code> ist. Als Bedingung können Sie gültige boolesche Ausdrücke eintragen. Beispiele: <code>x>100</code> , <code>x[y]=z</code> , <code>a AND b</code> , <code>boolVar</code> .

Registerkarte "Daten"

Voraussetzung: Es geht um die Eigenschaften eines "Datenhaltepunkts".

In der Registerkarte wird die Variable oder Speicheradresse angegeben, für die der Datenhaltepunkt gesetzt ist oder wird.	
Ausführung unterbrechen, wenn sich der Wert der angegebenen Variablen ändert	- Eingabe eines qualifizierten Variablennamens : Auswahl einer Variablen im Dialog Eingabehilfe, in der Kategorie "Watch-Variablen" Beispiele: Variable: <code>PLC_PRG.fb_DoSth.dwVariable</code>, Adresse: <code>16#12A</code>, <code>0x12A</code>, <code>129</code>
Größe	Anzahl der Bytes der oben angegebenen Variablen oder Speicheradresse, die auf Änderung hin überwacht werden sollen. Beim Neueintragen der Variablen/Adresse wird zunächst automatisch ein zum Datentyp/Speicherplatz passender Wert eingestellt. Beispiel: 4 für Datentyp <code>DWORD</code>

**Registerkarte "Ausführungspunkt
Einstellungen"**

In dieser Registerkarte kann ein bestehender Haltepunkt oder Datenhaltepunkt in einen Ausführungspunkt umgewandelt werden

Ausführungspunkt (Ausführung stoppt nicht am Haltepunkt)	 : Der Haltepunkt wird zum Ausführungspunkt. Die Ab- arbeitung hält an diesem Punkt nicht an, es wird jedoch der angegebene Code ausgeführt. - Ausführungspunkt eines Haltepunkts Aktiviert:  Deaktiviert:  - Ausführungspunkt eines Datenhaltepunkts Aktiviert:  Deaktiviert: 
"Folgenden Code ausführen"	Code, der beim Erreichen des Ausführungspunkts ausgeführt wird. Schleifenkonstruktionen (FOR, WHILE) und IF- oder CASE-Ausdrücke sind nicht möglich.
"Meldung im Geräte-Log ausgeben"	Diese Option ist nur verfügbar, wenn Sie die Option "Logging in Haltepunkten aktivieren" in den Einstellungen der Compiler-Optionen freigeschaltet haben. IndraLogic kann Variablen mit dem Platzhalter {variable name} im Meldungstext ausgeben.

Registerkarte "Ort"

Voraussetzung: Der Befehl "Neuer Haltepunkt" wurde ausgewählt.

POU	Baustein der aktiven Applikation, in dem der Haltepunkt positioniert ist
Position	Position des Haltepunkts im Baustein. Angabe in Form von Zeilen- und Spaltennummern (Texteditor) oder als Netzwerk- oder Elementnummern
Instanzen	Bei Funktionsbausteinen müssen Sie festlegen, ob der Haltepunkt in der Implementierung, oder in eine Instanz gesetzt werden soll.  : IndraLogic setzt den Haltepunkt in der Instanz. Bei dieser Option wählen Sie den "Instanzpfad" aus.  : IndraLogic setzt den Haltepunkt in der Implementierung.
Haltepunkt sofort aktivieren	 : Der Haltepunkt ist aktiviert.  : Der Haltepunkt ist nicht aktiviert. Zur späteren Aktivierung klicken Sie in der Ansicht "Haltepunkte" auf die Schaltfläche  .

Siehe auch [Kap. 12.2 "Verwenden von Haltepunkten"](#) auf Seite 193

18.3.5 Dialog 'Monitoringbereich'

Funktion: Der Dialog begrenzt den Bereich an Arrayelementen, deren Werte während des Monitorings angezeigt werden.

Aufruf: Klicken Sie in in das Feld der Spalte "Datentyp", das zur Arrayvariable gehört.

Voraussetzung: Eine POU ist im Onlinebetrieb und wird überwacht. Dabei hat eine Variable der POU den Datentyp `ARRAY`.

Gültiger Bereich	Gültigkeitsbereich der Arrayelemente, die überwacht werden. Beispiel eines dreidimensionalen Arrays: [1..10] [-3..3] [-10..10]
Maximale Zahl an Arrayelementen	Anzahl der Elemente der Arrayvariablen Beispiel: 1470
Wenn Sie eine der Einstellungen "Start", "Ende" oder "Scrollbereich von 1000 Elementen" ändern, werden automatisch die beiden anderen Einstellungen angepasst.	
Start	Index des ersten Arrayelements, dessen Wert angezeigt wird.
Ende	Index des letzten Arrayelements, dessen Wert angezeigt wird.
Scrollbereich von 1000 Elementen	Schiebebalken, um einen Bereich aus der Menge der Arrayelemente auszuwählen.

Siehe [Kap. 13.1.1 "Monitoring in Programmierobjekten aufrufen"](#) auf Seite 205

18.3.6 Dialog 'Eigenschaften'

Der Dialog dient zur Konfiguration der Eigenschaften eines Objekts in IndraLogic.

Der Dialog enthält je nach Objekt verschiedene Registerkarten, die je eine Kategorie von Eigenschaften behandeln.

Aufruf: Kontextmenü des Objekts im Projektnavigator.

Dialog 'Eigenschaften' - 'Allgemein'

Funktion: Der Dialog zeigt allgemeinen Informationen zum gerade selektierten Objekt.

Aufruf: Befehl **Ansicht ► Eigenschaften** oder Kontextmenü des Objekts, Kategorie **Allgemein**.

Voraussetzung: Ein Objekt ist im Projektextplorer unterhalb einer Steuerung oder im Ordner "Allgemeiner Baustein" selektiert.

Name	Objektname wie in POU- oder Gerätefenster
Objekttyp	Art des Objekts, zum Beispiel POU, Applikation, Schnittstelle
Öffnen mit	Typ des Editors, in dem das Objekt angezeigt oder bearbeitet wird.

Dialog 'Eigenschaften' - 'Boot-Applikation'

Funktion: Die Einstellungen in dieser Registerkarte definieren, ob und wann aus der Applikation eine Boot-Applikation erzeugt wird.

Voraussetzung: Das Gerät unterstützt die Einstellungen.

Aufruf: Applikationsobjekt selektieren, Kontextmenü **Eigenschaften**, oder Menü **Ansicht ► Eigenschaften**, Kategorie **Boot-Applikation**.

<Gruppenname>

Boot-Applikation Download	implizit	bei	Beim Download der Applikation wird automatisch eine Boot-Applikation erzeugt.
Boot-Applikation Online-Change	implizit	bei	Beim Online-Change wird automatisch eine Boot-Applikation erzeugt.
Beim Schließen des Projekts an Boot-Applikation erinnern			Vor dem Verlassen des Projekts fragt IndraLogic nach, ob die Boot-Applikation(en) (neu) angelegt werden soll(en).



Unabhängig von den hier definierten Voreinstellungen erhalten Sie beim Einloggen immer die Möglichkeit, explizit das Anlegen einer Boot-Applikation anzustoßen.

Siehe auch:

- [Kap. 11.4 "Erzeugen einer Bootapplikation" auf Seite 189](#)
- [Kap. "Befehl 'Online-Change'" auf Seite 657](#)

Dialog 'Eigenschaften' - 'Übersetzen'

Symbol

Funktion: Der Dialog enthält Optionen für das Übersetzen des Objekts.

Aufruf: Menü **Ansicht ► Eigenschaften**, Kontextmenü des Objekts im Projektor

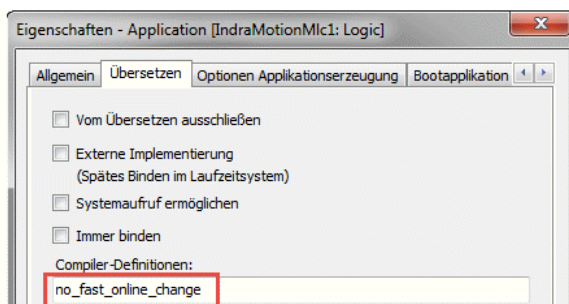
Name	Beschreibung
Vom Übersetzen ausschließen	Das Objekt und rekursiv seine Kindobjekte werden beim nächsten Übersetzungslauf nicht berücksichtigt. Der Objekteintrag in der Ansicht "Geräte" oder "POUs" wird in grüner Schrift dargestellt.
Externe Implementierung (Spätes Binden im Laufzeitsystem)	IndraLogic generiert beim Übersetzen des Projekts für dieses Objekt keinen Code. Das Objekt wird erst gelinkt, wenn das Projekt auf dem Zielsystem läuft, vorausgesetzt, es ist dort vorhanden (beispielsweise in einer Bibliothek). Der Objektname in der Ansicht "Geräte" oder "POUs" erhält den Zusatz (EXT)
Systemaufrufe ermöglichen	Ein Systemaufruf (Laufzeitsystem) für Funktionen ist möglich. Hintergrund: Im Gegensatz zu IndraLogic 1x kann in Version 2G der ADR-Operator mit Funktions-, Programm-, Funktionsbaustein- und Methodennamen verwendet werden. Er ersetzt den INSTANCE_OF-Operator. ABER: Es gibt keine Möglichkeit, Funktionszeiger innerhalb von IndraLogic aufzurufen.

Name	Beschreibung
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 Steuerung geladen wird.
Compiler-Definitionen	Hier können Sie "Defines" und Bedingungen für das Übersetzen des Objekts eintragen (bedingte Kompilierung). Sie können den Ausdruck <code>expr</code> , der in solchen Pragmas verwendet wird, auch hier eintragen. Mehrere Einträge sind in Form einer Komma separierten Liste möglich (siehe <code>{define}</code> Anweisungen). Beispiel: <code>hallo, test:='1'</code>

Anwendung von "Compiler-Definitionen"



Um die Wirkung des Attributs `init_on_onlchange` auf den gesamten Applikationscode sicherzustellen, müssen Sie den schnellen Online-Change mit Hilfe der Compiler-Definition `no_fast_online_change` generell für die Applikation ausschalten.



Fügen Sie die Definition zu diesem Zweck in den "Eigenschaften" der Applikation in Registerkarte "Übersetzen" ein.

Siehe auch

- [Kap. "Attribut 'init_on_onlchange'" auf Seite 416.](#)
- [Kap. 17.6.3 "Bedingte Pragmas" auf Seite 439](#)

Dialog 'Eigenschaften' - 'Zugriffskontrolle'

Funktion: Der Dialog dient der Festlegung, welche Benutzergruppe welche Aktionen auf das Objekt ausführen darf.

Aufruf: Kontextmenü **Eigenschaften**, Registerkarte **Zugriffskontrolle**

Voraussetzung: Im Projektextplorer unterhalb einer Steuerung oder im Ordner "Allgemeiner Baustein" ist ein Objekt selektiert.

Gruppen, Aktionen und Rechte	Tabelle, die für die bestehenden Benutzergruppen zeigt, welche Recht sie haben, eine der folgenden Aktionen auf das Objekt auszuführen: • Ansehen • Bearbeiten • Entfernen • Kindobjekte hinzufügen/löschen Mit Doppelklick auf ein Rechtesymbol öffnen Sie eine Auswahlliste aller möglichen Rechte.

Siehe auch

- [Kap. 6.3 "Objekte im Projekt über Zugriffsrechte schützen" auf Seite 62](#)

Dialog 'Eigenschaften' - 'Bitmap'

Funktion: Der Dialog dient der Festlegung, ob und mit welcher Bitmap-Datei (Bilddatei) das Objekt versehen wird. Das Bitmap wird in der grafischen Ansicht des Bibliotheksverwalters und in der Ansicht **Werkzeuge** des FUP/KOP/AWL-Editors verwendet.

Aufruf: Menü **Ansicht** ► **Eigenschaften**, Kontextmenü des Objekts.

Voraussetzung: Im Projektextplorer unterhalb einer Steuerung oder im Ordner "Allgemeiner Bausteinteil" ist ein Objekt selektiert.

Transparenzfarbe	Diese Farbe wird im Bitmap transparent dargestellt.

Siehe auch

- [Kap. 2.2 "Objekte hinzufügen" auf Seite 8](#)

Dialog 'Eigenschaften' - 'Optionen Applikationserzeugung'

Funktion: Der Dialog enthält Einstellungen, die IndraLogic bei der Erzeugung einer Bootapplikation für die Steuerung anwendet.

Aufruf: Menü **Ansicht** ► **Eigenschaften**, Kontextmenü eines Applikationsobjekts

Download der Applikationsinformation	Diese Funktionalität erfordert Compiler-Version >=3.5.0.0, Laufzeitsystemversion >= 3.5.0.0 Die Information zum Inhalt der Applikation wird mit auf die SPS geladen. Da dies eine Prüfung auf Unterschiede zwischen der aktuellen Applikation und der Applikation auf der Steuerung ermöglicht, empfehlen wir, die Option aktiviert zu lassen. Es geht um den Vergleich der Anzahl von Bausteinen, der Daten und der Speicherorte. Die Informationen zu den Unterschieden erhalten Sie dann auf der Registerkarte Applikationen des Geräteeditors über die Schaltfläche Details. Ebenso im Meldungsfenster, das erscheint, wenn Sie dabei sind, eine Applikation auf die Steuerung zu laden, die sich von der bereits dort liegenden unterscheidet.
Dynamische Speicherallozierung verwenden	Für die Applikation wird dynamisch Speicher belegt, beispielsweise bei Verwendung des Operators <code>__NEW</code>. In diesem Fall geben Sie die gewünschte Maximale Speichergröße (Bytes) an. Achtung: Nicht der gesamte Speicher ist für dynamische Objekterstellung verfügbar, sondern einen Teil davon verwendet das System immer für Verwaltungsinformationen.

Siehe auch

- [Kap. 11.2 "Applikationscode erzeugen" auf Seite 187](#)
- [Kap. 11.4 "Erzeugen einer Bootapplikation" auf Seite 189](#)
- [Kap. 18.1.2 "Objekt 'Applikation'" auf Seite 508](#)

Dialog 'Eigenschaften' - 'Netzwerkvariablen'

Symbol: 

Funktion: In diesem Dialog definieren Sie Netzwerkeigenschaften für die im Projektextplorer selektierte Variablenliste. Daraufhin stehen die in ihr deklarierten Variablen als Netzwerkvariablen zur Verfügung.

Aufruf: Kontextmenü der Variablenliste im Projektextplorer ► **Eigenschaften**, dann Auswahl der Registerkarte **Netzwerkeigenschaften**.

Netzwerktyp	UDP
Task	Task der aktuellen Applikation, die die zu sendenden Variablen kontrolliert. IndraLogic versendet die Variablen immer am Ende eines Task-Zyklus.
Variablenlistenkennung	Dient der Identifizierung der Netzwerkvariablenliste. Muss eindeutig sein.
Variablen packen	Die Größe der Pakete (Telegramme), die übertragen werden, hängt vom Netzwerktyp ab. Bei "UDP" ist ein Paket 256 Bytes groß. : IndraLogic bündelt die Variablen für die Übertragung in Pakete, um möglichst wenig Pakete zu übertragen. Bei Variablen vom Typ <code>ARRAY</code> oder strukturierten Datentypen kann dies zur Aufteilung der Variablen auf mehrere Telegramme führen. Somit sind Dateninkonsistenzen innerhalb dieser Variablen möglich, auch wenn die Variablengröße kleiner als die Paketgröße ist. : IndraLogic erzeugt ein Paket pro Variable.
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.
Bestätigung	: Für jedes erhaltene Datenpaket sendet IndraLogic eine Bestätigungsmeldung. Wenn der Sender keine Bestätigung erhalten hat, bevor er erneut sendet, wird ein Fehler in die Diagnosestruktur geschrieben. Hinweis: Ab Version 3.5.7.0 der Bibliothek <code>NetVarUdp</code> wird kein Empfangskanal mehr belegt, wenn kein bestätigter Transfer gewählt wird. Dadurch ist auch Netzwerkvariablen austausch zwischen zwei Steuerungen auf 1 Hardwaregerät möglich.
Zyklische Übertragung, Intervall	IndraLogic sendet die Variablen innerhalb des angegebenen Intervalls. Beispiel für Zeitangabe: "T#70ms".
Bei Änderung übertragen, Mindestabstand	: IndraLogic sendet die Variablen nur, wenn sich ihre Werte geändert haben; mit dem „Mindestabstand“ können Sie angeben, wieviel Zeit mindestens zwischen zwei Übertragungen liegen muss.

Bei Ereignis übertragen, Variable	 : IndraLogic sendet die Variablen, sobald die angegebene Variable TRUE wird.
Einstellungen	Protokollspezifische Einstellungen; die möglichen Einträge hängen von der jeweiligen Netzwerkbibliothek ab: Port: Nummer des Ports, den IndraLogic für den Datenaustausch mit anderen Netzwerkteilnehmern verwendet; der Vorgegebene Wert ist "1202"; den aktuellen Wert können Sie jederzeit im Feld Wert verändern: Feld auswählen, <Leertaste> drücken, Wert eingeben; Achtung: Die anderen Knoten im Netzwerk müssen 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. Den aktuellen Wert können Sie im Feld Wert verändern: Feld auswählen, <Leertaste> drücken, Adresse bzw. Adressbereich eines Sub-Netzwerks eingeben. Beispiel: "197 . 200 . 100 . 255", wenn mit allen Knoten kommuniziert werden soll, die IP-Adressen im Bereich von "197 . 200 . 100 . x" haben.

Siehe auch

- [Kap. 10.4.4 "Kommunikation bearbeiten" auf Seite 182](#)
- [Kap. 10.3.1 "Konfigurieren eines Netzwerkvariablen-Austauschs" auf Seite 171](#)
- [Kap. 18.1.16 "Objekt 'Globale Netzwerkvariablen-Liste \(Empfänger\)'" auf Seite 552](#)
- [Kap. 18.1.15 "Objekt 'Globale Netzwerkvariablen-Liste \(Sender\)'" auf Seite 551](#)

Dialog 'Eigenschaften' - 'Netzwerkeinstellungen'

Wenn die Netzwerkfunktionalität vom Gerät unterstützt wird, können die aktuellen Netzwerkeinstellungen für eine GNVL (globale Netzwerkvariablenliste) im Dialog **Eigenschaften** des Objekts angezeigt und verändert werden.

Dabei handelt es sich um die Einstellungen, die beim Einfügen der GNVL in Dialog **Netzwerkvariablen-Liste (Empfänger) hinzufügen** vorgenommen wurden.

Siehe auch:

- ["Dialog 'Globale Netzwerkvariablen-Liste \(Empfänger\) hinzufügen'" auf Seite 552](#)
- [Kap. 10.3.1 "Konfigurieren eines Netzwerkvariablen-Austauschs" auf Seite 171](#)

Dialog 'Eigenschaften' - Registerkarte 'CFC-Ausführungsreihenfolge'

Funktion	Konfiguration des Ausführungsreihenfolge-Modus in Netzwerken von CFC-Objekten.
Aufruf	Im Projektextplorer über des Kontextmenü "Eigenschaften" eines CFC-Objekts
Registerkarte 'CFC-Ausführungsreihenfolge'	Im CFC-Editor können Sie die Elemente und damit die Netzwerke frei positionieren.

Um zu vermeiden, dass die Ausführungsreihenfolge im CFC-Netzwerk durch die freie Positionierung unbestimmt ist, stehen 2 Modi zur Verfügung, um die Ausführungsreihenfolge zu definieren:

- Automatischer Datenfluss-Modus
- Expliziter Ausführungsreihenfolge-Modus

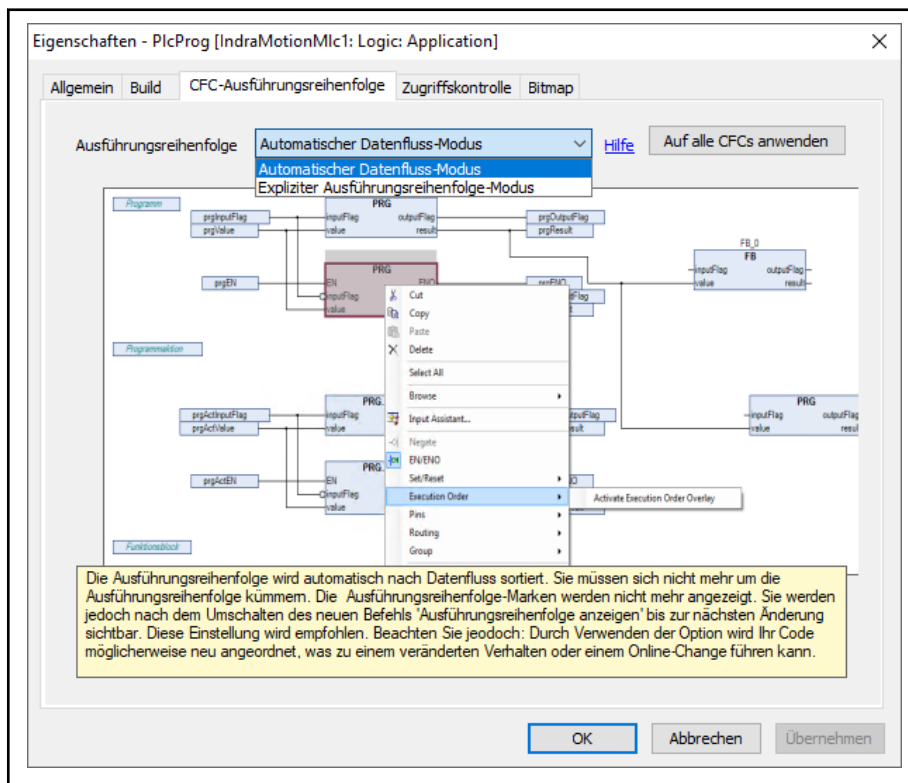


Abb. 18-73: Registerkarte 'CFC-Ausführungsreihenfolge'

Elemente der Registerkarte 'CFC-Ausführungsreihenfolge'

Modi zur Definition der Ausführungsreihenfolge in CFC Netzwerken	Automatischer Datenfluss-Modus In diesem Modus wird die Ausführungsreihenfolge automatisch nach Datenfluss und bei Mehrdeutigkeit nach Netzwerktopologie festgelegt. Die Programmierbausteine und die Ausgänge werden intern nummeriert. Dabei werden die oberen vor den unteren und die linken vor den rechten Netzwerken ausgeführt. Vorteil: Die automatische festgelegte Ausführungsreihenfolge ist zeit- und zyklusoptimiert. Sie benötigen auch während des Entwicklungsprozesses keine Information über die intern verwaltete Ausführungsreihenfolge. Im Menü CFC ▶ Ausführungsreihenfolge stehen danach folgende Befehle zur Verfügung: • Ausführungsreihenfolge anzeigen • Startpunkt der Rückkopplung setzen Die Elemente im CFC-Editor werden ohne Marken und ohne Nummerierung angezeigt. Ein manuelles Ändern der Ausführungsreihenfolge ist nicht möglich. Bei Netzwerken mit Rückkopplungen können Sie zusätzlich einen Startpunkt setzen. Expliziter Ausführungsreihenfolge-Modus In diesem Modus können Sie die Ausführungsreihenfolge explizit festlegen. Dafür werden die Elemente im CFC-Editor mit Marken und Nummerierung angezeigt und Menübefehle bereitgestellt, mit denen Sie die Reihenfolge bestimmen können. Im Menü CFC ▶ Ausführungsreihenfolge stehen danach folgende Befehle zur Verfügung: • An den Anfang • Ans Ende • Eins vor • Eins zurück • Ausführungsreihenfolge setzen • Nach Datenfluss anordnen • Topologisch anordnen Hinweis: Bis IndraWorks 14VRS war dieses Verhalten das übliche Verhalten von CFC-Programmierbausteinen. Beachten Sie, dass Sie die Ausführungsreihenfolge in Eigenverantwortung anpassen und die Konsequenzen und Effekte selbst beurteilen müssen. Die Ausführungsreihenfolge wird dafür ständig eingeblendet.
Befehl "Auf alle CFCs anwenden"	Der Befehl stellt den Modus für alle weiteren CFC-Objekte im Projekt um, auf den Modus, der in der Liste ausgewählt ist.

Tab. 18-16:

Siehe auch • [Kap. "Befehl 'An den Anfang'" auf Seite 691](#)

- [Kap. "Befehl 'Ans Ende'" auf Seite 692](#)
- [Kap. "Befehl 'Eins vor'" auf Seite 692](#)
- [Kap. "Befehl 'Nach Datenfluss anordnen'" auf Seite 693](#)
- [Kap. "Befehl 'Topologisch anordnen'" auf Seite 694](#)

Dialog 'Eigenschaften' - 'AS-Einstellungen'

Funktion: Der Dialog definiert Standardeinstellungen für alle im Projekt verwendeten POU's, die in Ablaufsprache programmiert sind.

Aufruf: Menü **Ansicht** ► **Eigenschaften**, Kontextmenü einer in Ablaufsprache programmierten POU im Projektexplorer unterhalb einer Steuerung oder im Ordner "Allgemeiner Bausteinteil".

Variablen ("AS-Flags")

Auflistung aller möglichen AS-Flags	Aktiv : Das AS-Flag wird aktiviert und somit bei der Abarbeitung berücksichtigt. Deklaration : Das AS-Flag wird automatisch deklariert. Wenn Deklarieren aktiviert ist, aber Aktiv nicht, dann wird zwar die Deklaration der Variable erfolgen, aber das Flag hat während der Abarbeitung keine Funktion. Hinweis: Wenn Sie eine manuelle Deklaration einer AS-Flag-Variable vorgenommen haben, müssen Sie für dieses Flag die Deklaration in den AS-Einstellungen deaktivieren. Sonst überschreibt das automatisch deklarierte das manuell deklarierte Flag. Hinweis: Eine automatisch deklarierte Flag-Variable ist nur im Online-Betrieb im Deklarationsteil des AS-Editors sichtbar.
Voreinstellungen verwenden	: Die hier im Dialog vorliegenden Einstellungen überschreiben die AS-Einstellungen, die innerhalb der einzelnen POU's definiert sind.

Siehe auch

- [Kap. "AS-Flags" auf Seite 258](#)

Übersetzen

Codegenerierung

Nur aktive Transitionen berechnen	: IndraLogic erzeugt nur für die gerade aktiven Transitionen Code.

Dialog 'Eigenschaften' - 'Verknüpfung mit Datei'

Funktion: Der Dialog definiert die Verknüpfung einer externen Datei mit dem Inhalt der globalen Variablenliste (GVL). Sie können die GVL entweder in eine solche Datei exportieren oder aus einer solchen Datei importieren.

Aufruf: Menü **Ansicht** ► **Eigenschaften**, Kontextmenü einer POU des Typs GVL.

Export vor Übersetzen	: Bei jedem Übersetzen des Projekts (Erstellen ► Übersetzen, bzw. <Umsch> + <F11>) speichert IndraLogic eine Datei mit Erweiterung <code>gvl</code> in dem Pfad, der im Feld Dateiname angegeben ist.
Import vor Übersetzen	: Eine vorhandene Exportdatei wird bei jeder Projektübersetzung automatisch eingelesen. Sie können somit eine GVL, die aus einem anderen Projekt exportiert wurde, importieren, um beispielsweise eine Kommunikation über Netzwerkvariablen aufzusetzen.

Siehe auch

- [Kap. 18.1.11 "Objekt 'GVL'- Globale Variablenliste" auf Seite 544](#)
- [Kap. 10.3.1 "Konfigurieren eines Netzwerkvariablen-Austauschs" auf Seite 171](#)

Dialog 'Eigenschaften' - 'Bildersammlung'

Funktion: Der Dialog ermöglicht die Konfiguration der selektierten Bildersammlung.

Aufruf: Menü **Ansicht** ► **Eigenschaften** eines Objekts des Typs **Bildersammlung**, Kontextmenü eines Objekts des Typs **Bildersammlung**

Nur verwendete Bilder auf Steuerung laden	 IndraLogic lädt nicht alle Bilder, sondern nur die in der Applikation tatsächlich verwendeten Bilder der Bildersammlung auf die Steuerung.
Intern	 IndraLogic stellt die Bildersammlung nicht in der Ansicht Werkzeuge zur Verfügung. Sie können diese Bilder nicht in eine Visualisierung ziehen.

Einstellungen Symbolbibliothek

Bibliothek als Symbolbibliothek kennzeichnen	Kennzeichnet die Bildersammlung als Symbolbibliothek zur Verwendung in einer Visualisierung. Die Symbolbibliothek erhält in den Bibliotheksinformationen als Dateieigenschaft den Schlüssel <code>VisuSymbolLibrary = TRUE</code>. Im Bibliotheksverwalter im Ordner "Allgemeiner Baustein" wird automatisch die Bibliothek <code>VisuElements</code> als Platzhalterbibliothek eingefügt. Voraussetzung: Sie haben ein Bibliotheksprojekt geöffnet. IndraLogic zeigt Symbolbibliotheken, die im Repository installiert sind, unter Projekteinstellungen, Kategorie Visualisierung, Registerkarte Symbolbibliotheken an.
Textliste für die Symbolübersetzung	Wählen Sie aus der Combobox die Textliste aus, die die lokalisierten Texte für die Bildersammlung enthält.

Siehe auch

- [Kap. 20.1.2 "Verfügbare Visualisierungselemente anzeigen und kennzeichnen" auf Seite 806](#)
- [Kap. 18.2.15 "Menü 'Textliste'" auf Seite 727](#)
- [Kap. "Dialog 'Projekteinstellungen- 'Visualisierung'" auf Seite 763](#)

Dialog 'Eigenschaften' - 'Optionen'

Optionen (Steuerung)

Funktion: Der Dialog enthält die Einstellungen für Monitoring und das Einloggen für ein Objekt des Typs Gerät (Beispiel: "Safety_CPU_CFL01_1_F1").

Welche Optionen verfügbar sind, hängt von der Gerätebeschreibung ab.

Aufruf: Kontextmenü des Geräteobjekts, oder Menü **Ansicht** ► **Eigenschaften**, wenn das Geräteobjekt selektiert ist.

Überwachungsintervall (ms)	Intervall für das Monitoring (10 ms - 1000 ms)

Interaktiver Login-Mode

Dieser Mode dient dazu, ein versehentliches Einloggen auf eine andere Steuerung zu verhindern.	
Kein	Keine Interaktion mit dem Anwender beim Einloggen. Entspricht dem bisherigen Verhalten der älteren Versionen

ID eingeben	Beim Einloggen fordert IndraLogic Sie auf, eine ID einzugeben. Die ID ist in der Steuerung hinterlegt. Ohne gültige ID ist kein Einloggen möglich Bei einem erneuten Einloggen fragt IndraLogic nicht mehr nach der ID, wenn der Computername, der Benutzername, der Geräte name und die Geräteadresse unverändert sind. Die Information wird in den Projektoptionen gespeichert.
Taste drücken	Beim Einloggen erscheint ein Dialog mit der Aufforderung eine Taste auf der Steuerung zu drücken. Der Timeout für diese Aktion ist in der Gerätebeschreibung definiert.
Blinken (LED)	Beim Einloggen blinkt die verbundene Steuerung.

Symbolkonfiguration

Variablenzugriff synchron zur IEC-Task	 : Standardeinstellung, konsistenter Zugriff nicht erlaubt  : Konsistenter Zugriff erlaubt Die Einstellung wird erst wirksam, wenn Sie alle Applikationen und Bootapplikationen neu auf die Steuerung laden. Hinweis: Wenn die Option aktiviert ist, kann sich der Jitter für alle IEC-Applikationen auf diesem Gerät erhöhen! Der konsistente Zugriff kann die Echtzeitfähigkeit stören.

Siehe auch

- [Kap. 8.1 "Projektexplorer und Geräteeditor" auf Seite 71](#)
- [Kap. 18.1 "Objekte" auf Seite 507](#)

18.3.7 Dialog 'Projekteinstellungen'

Symbol: 

Funktion: Das Objekt enthält die Grundkonfiguration des Projekts. Sie können im Dialog **Projekteinstellungen** bearbeitet werden.

Aufruf: Doppelklick auf das Objekt **Projekteinstellungen** im Projektexplorer, Ordner **Allgemeiner Baustein**.

IndraLogic speichert die Projekteinstellungen direkt im Projekt. Wenn zum Beispiel ein Projekt auf ein anderes System übertragen wird, wird auch das Objekt **Projekteinstellungen** mit übertragen, ohne dass dafür ein Projektarchiv benötigt wird.

Die Projekteinstellungen gelten projektweit. Sie bieten Einstellmöglichkeiten für verschiedene Kategorien, wie zum Beispiel **AS** oder **Compiler-Optionen / Compiler-Warnungen** bzw. **StaticAnalysisLight** und **Visualisierung**.

Dialog 'Projekteinstellungen' - 'AS'

Symbol: 

Funktion: Der Dialog dient der Konfiguration der Einstellungen für AS-Objekte. Jedes neue AS-Objekt hat die konfigurierten Einstellungen automatisch in seinen Eigenschaften.

Aufruf: Menü **Projekt ► Projekteinstellungen**, Kategorie **AS**

Voraussetzung: Ein Projekt ist geöffnet.

Registerkarte 'Variablen'

Implizit erzeugte Variablen (Flags) zur Kontrolle und zum Monitoring der Abarbeitung in einem AS-Diagramm.	
Aktiv	 : Die entsprechende Variable wird verwendet.
Deklariieren	 : Die entsprechende Variable wird automatisch angelegt. Ansonsten, falls die Verwendung beabsichtigt ist (Aktiv ist gesetzt), muss der Anwender sie selbst deklarieren.
Auf alle anwenden	In diesem Dialog vorgenommene Änderungen wendet IndraLogic auf bereits bestehende AS-Objekte an. Im Eigenschaften-Dialog der AS-POUs aktiviert IndraLogic die Option Voreinstellungen verwenden .



Eine automatisch deklarierte Flag-Variable ist nur im Online-Betrieb im Deklarationsteil des AS-Editors sichtbar.

Registerkarte 'Übersetzen'

Codegenerierung

Nur aktive Transitionen berechnen	 : IndraLogic erzeugt nur für gerade aktive Transitionen Code.

AS-Bibliothek

Dieser Teil des Dialogs ist nur für Compilerversion < 3.4.1.0 verfügbar.	
Firma Titel Version	Definiert die AS-Bibliothek, die IndraLogic standardmäßig verwendet.
Namensraum	Wird definiert, um die Bibliotheken eindeutig ansprechen zu können. Erforderlich, wenn verschiedene Versionen der Bibliothek auf dem System verfügbar sind. Achten Sie darauf, dass es keine Unstimmigkeiten zwischen dem im Bibliotheksverwalter und dem für das einzelne Objekt definierten Namensraum gibt. Als Standardeinstellungen gelten die Daten der Bibliothek <code>SfcIec.library</code> , die IndraLogic mit dem Standardprofil bereitstellt.



Jeder einzelne AS-Baustein speichert die Information über die Bibliotheksversion, die gültig war, als Sie den Baustein einfügten. Dies kann dazu führen, dass Sie mehrere Bibliotheksversionen innerhalb desselben Projekts verwenden. Um das zu vermeiden, können Sie ab Compilerversion 3.4.1.0 hier keine fixe Version der `IecSfc.library` mehr eintragen. Die Bibliotheksversion, die Sie für alle AS-Bausteine des Projekts verwenden, definieren Sie nun über einen Platzhalter.

IndraLogic löst den Platzhalter abhängig von der verwendeten Compilerversion auf. Die Zuordnung der Bibliotheksversion zur Compilerversion ist im Bibliotheksprofil definiert.

Siehe auch

- [Kap. "AS-Flags" auf Seite 258](#)

Dialog 'Projekteinstellungen' - 'Compileroptionen'

Symbol: 

Funktion: Der Dialog dient der Konfiguration der Compileroptionen.

Aufruf: Menü **Ordner "Allgemeiner Baustein"** ► **Projekteinstellungen** , Kategorie **Compileroption**.

Voraussetzung: Ein Projekt ist geöffnet.

Compilerversion

Feste Version	Definiert die Compilerversion, die IndraLogic beim Übersetzen und während des Ladens zum Übersetzen verwendet. Beispielsweise: 3.5.14.0 für Version 3.5 SP14 Die Compilerversion wird lediglich angezeigt, sie ist nicht editierbar.

Einstellungen

Unicode-Zeichen als Bezeichner zulassen	Standardmäßig deaktiviert, da die Verwendung von Unicode-Zeichen in Bezeichnern in der IEC-Norm nicht zulässig ist. Eventuell notwendig für asiatische Sprachen.
Konstanten ersetzen	 (Voreinstellung): Für jede Konstante skalaren Typs, also nicht für <code>STRING</code> , <code>ARRAY</code> oder Strukturen, lädt IndraLogic direkt den Wert. Im Online-Betrieb kennzeichnet IndraLogic die Konstanten im Deklarationseditor oder Monitoring-Fenster durch ein dem Wert vorangestelltes Symbol. In diesem Fall ist ein Zugriff, zum Beispiel über einen <code>ADR</code> -Operator, Forcen und Schreiben, nicht möglich.  : Der Zugriff auf Konstanten ist möglich, allerdings verlängert dies die Rechenzeit.
Logging in Haltepunkten aktivieren	Für als Ausführungspunkte definierte Haltepunkte können Sie im Dialog Haltepunkteigenschaften einen Meldungstext eintragen. Diesen Text gibt IndraLogic im Geräte-Log aus, wenn die Applikation am Ausführungspunkt stoppt.

Compilerwarnungen

Maximale Anzahl an Warnungen	Bezieht sich auf die Warnungen, die IndraLogic im Meldungsfenster ausgibt. Die Auswahl der angezeigten Compilerwarnungen legen Sie im Dialog Projekteinstellungen in der Kategorie Compilerwarnungen fest.

Siehe auch

- [Kap. "Dialog 'Projekteinstellungen- 'Compilerwarnungen'" auf Seite 760](#)
- [Kap. "Registerkarte 'Log'" auf Seite 525](#)
- [Kap. 18.3.2 "Dialog 'Eigenschaften Haltepunkt'" auf Seite 745](#)

Dialog 'Projekteinstellungen' - 'Compilerwarnungen'

Symbol: 

Funktion: Der Dialog dient der Auswahl der Compilerwarnungen, die IndraLogic bei einem Kompilierungslauf im Meldungsfenster anzeigt.

Aufruf: Menü **Ordner "Allgemeiner Baustein"** ► **Projekteinstellungen** , Kategorie **Compilerwarnungen**.

Voraussetzung: Ein Projekt ist geöffnet.



Die maximale Anzahl aufgelisteter Warnungen legen Sie im Dialog **Compileroptionen** fest.

Siehe auch

- [Kap. "Dialog 'Projekteinstellungen- 'Compileroptionen'" auf Seite 759](#)
- [Kap. "Befehl 'Übersetzen'" auf Seite 649](#)

Dialog 'Projekteinstellungen' - 'Static Analysis Light'

Symbol: 

Funktion: Der Dialog aktiviert die Prüfungen, die bei der Light-Version der Statischen Analyse in CODESYS bei einer Codeerzeugung jeweils durchgeführt werden.

Aufruf: Im Projektextplorer **Ordner "Allgemeiner Baustein" ► Projekteinstellungen**, Kategorie **Static Analysis Light**.



Sie können die statische Codeanalyse für Codeteile ausschließen, indem Sie den Code mit dem Pragma `{analysis ...}` oder dem Pragma `{attribute 'analysis' := '...')}` kennzeichnen.

Zusätzliche Kompilierungsprüfungen

SA033: Nicht verwendete Variablen	Ermittelt Variablen, die deklariert sind, aber innerhalb des kompilierten Programmcodes nicht verwendet werden. Hinweis für GVL-Variablen im Ordner "Allgemeiner Baustein": Wenn es in einem Projekt mehrere Applikationen gibt, berücksichtigt IndraLogic nur die Objekte unter der gerade aktiven Applikation. Wenn es nur eine einzige Applikation gibt, werden auch die Objekte im Ordner "Allgemeiner Baustein" betrachtet.
SA0028: Überlappende Speicherbereiche	Ermittelt die Stellen, durch die 2 oder mehr Variablen denselben Speicherplatz belegen. Beispielsweise tritt dies auf, wenn es die folgenden Deklarationen gibt: <pre>var1 AT %QB21: INT und var2 AT %QD5: DWORD.</pre> In diesem Fall verwenden beide Variablen Byte 21, d. h. die Speicherbereiche der Variablen überlappen.
SA0006: Schreibzugriff aus mehreren Tasks	Ermittelt Variablen, auf die von mehr als einer Task geschrieben wird.
SA0004: Mehrfacher Schreibzugriff auf Ausgang	Ermittelt Ausgänge, die an mehr als einer Position geschrieben werden. Hinweis: Es wird kein Fehler ausgegeben, wenn auf eine Ausgabevariable (VAR_IN_OUT) in verschiedenen Zweigen von IF oder CASE Anweisungen geschrieben wird. Hinweis: Sie können diese Regel nicht über Pragmas abschalten.

SA0027: Mehrfachverwendungen von Bezeichnern	Ermittelt Mehrfachverwendung des Namens/Bezeichners einer Variable oder eines Objekts (POU) innerhalb des Gültigkeitsbereichs eines Projekts. <i>Die folgenden Fälle werden abgedeckt:</i> • Der Name einer Enumerationskonstante ist derselbe, wie der in einer anderen Aufzählung innerhalb der Applikation oder einer eingebundenen Bibliothek verwendeten. • Der Name einer Variablen ist derselbe wie der eines Objekts innerhalb der Applikation oder einer eingebundenen Bibliothek. • Der Name einer Variablen ist derselbe wie der einer Enumerationskonstante in einer Enumeration in der Applikation oder einer eingebundenen Bibliothek. • Der Name eines Objekts ist derselbe wie der eines anderen Objekts in der Applikation. • Der Name einer Variablen ist derselbe wie der Name einer Methode. • Der Name eines Objekts ist derselbe wie der Name eines Objekts, das übergeordnet ist ("Parent object").
SA0167: Temporäre Funktionsbaustein-Instanzen	Die Prüfung ermittelt Funktionsbaustein-Instanzen, die als temporäre Variablen deklariert sind. Dies betrifft also Instanzen, die in einer Methode oder in einer Funktion oder als VAR_TEMP deklariert sind, und die deshalb in jedem Abarbeitungszyklus und bei jedem Bausteinaufruf neu initialisiert werden.

Beispiele

SA0003: Leere Anweisungen

```
;
(* Comment *);
iVar;
```

SA0006: Gleichzeitiger Zugriff

```
FUNCTION_BLOCK ADD_FB
g_iTemp1 := g_iTemp1 + INT#1;
```

```
PROGRAM PlcProg //controlled by MainTask
g_iTemp1 := g_iTemp1 + INT#2;
g_xTemp2 := g_iTemp1 > INT#10;
```

```
PROGRAM PlcProg_1 //controlled by SubTask
g_iTemp1 := g_iTemp1 - INT#3;
g_xTemp2 := g_iTemp1 < INT#-10;
```

SA0004: Mehrfacher Schreibzugriff auf Ausgang

```
VAR_GLOBAL
  g_xVar AT %QX0.0 : BOOL ;
  g_iTest AT %QW0 : INT ;
END_VAR
```

```
PROGRAM PlcProg
IF iCondition < INT#0 THEN
  g_xVar := TRUE;
  g_iTest := INT#12;
END_IF
```

```
CASE iCondition OF
  INT#1:
    g_xVar := FALSE;
  INT#2:
    g_iTest := INT#11;
```

```

ELSE
    g_xVar := TRUE;
    g_iTest := INT#9;
END_CASE

```

SA0006: Schreibzugriff aus mehreren Tasks

```

FUNCTION_BLOCK ADD_FB
g_iTemp1 := g_iTemp1 + INT#1;

PROGRAM PLC_PRG // Controlled by MainTask
g_iTemp1 := g_iTemp1 + INT#2;
g_xTemp2 := g_iTemp1 > INT#10;

PROGRAM PLC_PRG 1 //Controlled by SubTask
g_iTemp1 := g_iTemp1 - INT#3;
g_xTemp2 := g_iTemp1 < INT#-10;

```

SA0027: Mehrfachverwendung des Namens

```

PROGRAM PlcProg
VAR
ton : INT; // error SA0027
END_VAR

```

SA0029: Notation in Implementierung und Deklaration unterschiedlich

Im Projektextplorer gibt es einen Baustein `PLC_PRG` und einen Baustein `fnC` (Funktion).

```

PROGRAM PLC_PRG
VAR
    iVar:INT;
    _123test_var_:INT;
END_VAR
    iVar := iVar + 1; // notation different to that in
                     // the declaration part -> SA0029
    _123TEST_var_ := _123test_var_INT; // notation different to that
                                     // in the declaration part -> SA0029
    Fnc(); // notation different to that in the devices tree -> SA0029
END_VAR

```

SA0167: Temporäre Funktionsbaustein-Instanzen

```

PROGRAM PLC_PRG
VAR
END_VAR
VAR_TEMP
    yafb: AFB;
END_VAR

FUNCTION Fun : INT
VAR_INPUT
END_VAR
VAR
    funafb: AFB;
END_VAR

METHOD METH: INT
VAR_INPUT
END_VAR
VAR
    methafb: AFB;
END_VAR

```

Siehe auch

- [Kap. 9.11.2 "Code statisch analysieren" auf Seite 132](#)

Dialog 'Projekteinstellungen' - 'Visualisierung'

Symbol: 

Funktion: Der Dialog dient der Konfiguration der projektweiten Einstellungen für Objekte des Typs **Visualisierung**.

Aufruf: Ordner "Allgemeiner Baustein" ► Projekteinstellungen, Kategorie **Visualisierung**

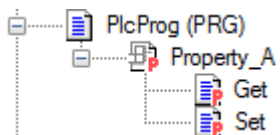
Voraussetzung: Ein Projekt ist geöffnet.

Registerkarte 'Allgemein'

Visualisierungsverzeichnisse

Textlistendateien	Verzeichnis, das Textlisten enthält, die im Projekt zur Verfügung stehen, um Texte für verschiedene Sprachen zu konfigurieren. IndraLogic verwendet das Verzeichnis zum Beispiel beim Exportieren oder Importieren von Textlisten. Mit Klick auf  erscheint der Dialog Ordner suchen, der das Auswählen eines Verzeichnisses im Dateisystem ermöglicht.
Bilddateien	Verzeichnis, das Bilddateien enthält, die im Projekt zur Verfügung stehen. Mehrere Ordner sind mit einem Strichpunkt voneinander getrennt. IndraLogic verwendet das Verzeichnis zum Beispiel beim Exportieren oder Importieren von Bilddateien. Mit Klick auf  erscheint der Dialog Ordner suchen, der das Auswählen eines Verzeichnisses im Dateisystem ermöglicht.

Erweiterte Einstellungen

Eigenschaften-Handling in allen Elementeigenschaften aktivieren	Voraussetzung: Sichtbar ist aktiviert. : Sie können ein Visualisierungselement in denjenigen seiner Eigenschaften, in denen Sie eine IEC-Variable auswählen, auch mit einer Eigenschaft  konfigurieren. IndraLogic erzeugt dann beim Übersetzen einer Visualisierung zusätzlichen Code für das Eigenschaften-Handling. Voraussetzung: Ihr IEC-Code enthält mindestens ein Objekt des Typs POU-Eigenschaft. 

Siehe auch

- [Kap. "Objekt 'Eigenschaft'" auf Seite 571](#)

Registerkarte 'Symbolbibliotheken'

Symbolbibliotheken	Liste aller installierten Symbolbibliotheken, zum Beispiel VisuSymbols
Zugewiesen	: Symbolbibliothek ist im Projekt ausgewählt und IndraLogic stellt sie in der Ansicht Werkzeuge einer Visualisierung zur Verfügung. : Symbolbibliothek ist im Bibliotheks-Repository installiert, aber IndraLogic stellt sie in der Ansicht Werkzeuge einer Visualisierung nicht zur Verfügung.

Tab. 18-17: **Visualisierungssymbolbibliotheken**

Siehe auch

- [Kap. 20.3.6 "Dialog 'Visualisierung hinzufügen'" auf Seite 842](#)

18.3.8 Dialog 'Eigenschaften' - 'Textliste'

Funktion: Der Dialog ermöglicht die Einstellung von Grundeigenschaften der selektierten Textliste.

Aufruf: Im Projektextplorer über das Kontextmenü eines Objekts des Typs "Textliste" **Eigenschaften**

Download durch Visualisierung	 : Die Textliste wird mit der Visualisierung auf die Steuerung geladen.
Intern	 : Die Textliste kann nur in einer Bibliothek verwendet werden.

Siehe auch

- [Kap. 18.1.23 "Objekt 'Textliste'" auf Seite 588](#)

18.3.9 Dialog 'Optionen'

Funktion: Der Dialog dient zur Auswahl der IndraWorks/ IndraLogic-Optionen, mit denen Sie das Verhalten und Aussehen der Benutzeroberfläche konfigurieren.

IndraWorks speichert die aktuellen Einstellungen auf dem lokalen System als Standardeinstellungen.

Aufruf: Menü **Extras** ▶ **Optionen** ▶ **IndraLogic 2G**

Dialog 'Optionen' - 'Allgemeine Einstellungen'

Funktion: Der Dialog dient der Konfiguration allgemeiner Einstellungen für das Arbeiten mit IndraLogic.

Aufruf: Menü: **Extras** ▶ **Optionen** ▶ **IndraLogic 2G** ▶ **Allgemeine Einstellungen**.

Objekt nach dem Hinzufügen öffnen:	Bei aktivierter Option wird ein neues Objekt, welches im Projektnavigator hinzugefügt wird, sofort geöffnet.
Gerätetyp im Projektextplorer anzeigen:	Der Name des Objektes wird um seinen Gerätetyp erweitert.
SPS-Logger aktivieren:	Über diese Schaltfläche kann das Registerblatt "Log" des Gerätedialogs einer jeden Steuerung aktiviert werden.
SPS-Shell aktivieren:	Bei der SPS-Shell handelt es sich um einen textbasierten Steuerungs-Monitor (Terminal). Kommandos zur Abfrage bestimmter Informationen aus der Steuerung werden in einer Eingabezeile eingegeben und als String an die Steuerung geschickt. Der zurückgelieferte Antwortstring wird in einem Ergebnisfenster des Browsers dargestellt. Diese Funktionalität dient Diagnose- und Debugging- Zwecken. Die Befehle, welche vom Zielsystem unterstützt werden, können vom jeweiligen Steuerungshersteller erweitert werden. Der SPS Shell Command Handler kann entweder in einer C-Komponente im Laufzeitsystem oder als Funktionsbaustein in einer IEC-Bibliothek oder einem Projekt implementiert sein.
SPS-Trace (IL-Trace) aktivieren (nur für Support):	

Dialog 'Optionen' - 'CFC-Editor'

Symbol: 

Funktion: Der Dialog dient der Konfiguration der Einstellungen für das Editieren und Drucken im CFC-Editor.

Aufruf: Menü **Extras** ► **Optionen** ► **IndraLogic 2G**, Kategorie **CFC-Editor**

Registerkarte 'Allgemein'

Automatisches Verknüpfen aktivieren	 : Wenn Sie ein CFC-Element auf die Arbeitsfläche des Editors ziehen und einfügen, verbindet IndraLogic automatisch unverknüpfte Pins, die sich "berühren". Achten Sie darauf, dass Sie beim Verschieben von Elementen keine unerwünschten Verknüpfungen erzeugen!
Werte im Implementierungsteil vorbereiten	 : Im Online-Betrieb können Sie auch im Implementierungsteil des CFC-Bausteins Variablenwerte für das Schreiben und Forcen vorbereiten. Außerdem zeigt IndraLogic die gerade vorbereiteten Werte in der Inlinemonitoring-Box der Variablen in spitzen Klammern an.

Registerkarte 'Ansicht'

Rasterpunkte anzeigen	 : Im Editor sind Rasterpunkte Gültigkeitsbereich, an denen Sie die Elemente positionieren können.
Bausteinsymbol anzeigen	 : stellt im CFC-Editor vorhandene Funktionsbausteine, die mit einem Bitmap verknüpft sind, als Symbol dar. Voraussetzung: Sie haben für einen Funktionsbaustein oder eine Funktion die Verknüpfung entweder in den Objekteigenschaften erstellt oder über eine Bibliothek geladen.
Linienfarben bearbeiten	Öffnet den Dialog Linienfarben bearbeiten zur Definition der Farben für die Verbindungslinien in Abhängigkeit vom anliegenden Datentyp. Die Linien erscheinen im Offline- und Online-Betrieb in diesen Farben, außer IndraLogic übermalt diese Farben durch die fette schwarze und blaue Liniendarstellung, die den booleschen Datenfluss anzeigt. • Typ hinzufügen: Fügt der Liste eine Datentyp hinzu. • Typ entfernen
Schriftart	Anzeige der Schriftart und Schaltfläche zur Änderung der Schriftart.

Siehe auch

- [Kap. "Dialog 'Eigenschaften- 'Bitmap'" auf Seite 751](#)

Registerkarte 'Drucken'

Einstellung der Layout-Optionen	
Seitenanpassung	Seite oder Poster
Skalierung	Mögliche Werte: 20 % - 200 %

Siehe auch

- [Kap. "Im CFC-Editor programmieren" auf Seite 100](#)
- [Kap. 17.1 "Programmiersprachen und ihre Editoren" auf Seite 239](#)

Dialog 'Optionen' - 'Deklarationseditor'

Symbol: 

Funktion: Der Dialog dient der Konfiguration der Anzeigeeinstellungen für den Deklarationseditor.

Aufruf Menü Extras ► Optionen ► IndraLogic 2G, Kategorie Deklarationseditor

Nur textuell	Textuelle Ansicht des Deklarationseditors
Nur tabellarisch	Tabellarische Ansicht des Deklarationseditors
Umschaltbar zwischen textuell und tabellarisch	Der Deklarationseditor bietet zwei Schaltflächen, um zwischen der textuellen und der tabellarischen Ansicht zu wechseln: : textuelle Ansicht : tabellarische Ansicht Die folgende Option definiert die Ansicht, die standardmäßig beim Öffnen eines Programmierobjekts erscheint: • Immer textuell • Immer tabellarisch • Letzte Einstellung (pro Objekt) speichern • Letzte Einstellung (global) speichern

Siehe auch

- [Kap. 9.2.1 "Deklarationseditor verwenden" auf Seite 87](#)

Dialog 'Optionen' - 'Geräteeditor'

Symbol: 

Funktion: Der Dialog enthält Einstellungen zur Darstellung des Geräteeditor.

Aufruf: IndraWorks Menü Extras ► Optionen ► IndraLogic 2G ► Geräteeditor.

Registerkarte 'Ansicht'

Generische Konfigurationseditoren anzeigen	 : Die Registerkarte mit der Liste der Geräte-Parameter ist in den Geräteeditoren parametrierbarer Geräte verfügbar.
Querverweise für IEC-Adressen erzeugen (Bereinigen erforderlich)	 : erzeugt die Querverweise für nicht abgebildete E/As.
Kommunikationsseite	• Einfacher Modus: Die Registerkarte Kommunikation erscheint so, wie es in dem entsprechenden Kapitel in der Onlinehilfe beschrieben ist. • Klassischer Modus: Die Registerkarte "Kommunikation" der Geräteeditoren erscheint als zweigeteiltes Fenster: Linker Teil zeigt die aktuell konfigurierten Gatewaykanäle in einer Baumstruktur, rechter Teil zeigt die zugehörigen Daten und Informationen.
Implizite Dateien für den Applikationsdownload im Editor einer SPS anzeigen	: Registerkarte Synchronisierte Dateien ist in den Geräteeditoren verfügbar. Synchronisierte Dateien sind Dateien, die beim Applikationsdownload mit auf die SPS geladen werden. Dies können externe Dateien sein, die der Applikation hinzugefügt wurden, sowie implizite Dateien wie beispielsweise ein Quellcode-Archiv.
Registerkarte "Zugriffsrechte" anzeigen	: Registerkarte "Zugriffsrechte" ist in den Geräteeditoren verfügbar. Hinweis: Diese Einstellung kann je nach Gerät durch die Gerätebeschreibung überschrieben werden!

Siehe auch:

- [Kap. "Generischer Geräteeditor" auf Seite 519](#)
- [Kap. "Registerkarte 'Synchronisierte Dateien'" auf Seite 524](#)

Dialog 'Optionen' - 'FUP, KOP und AWL'

Symbol: 

Funktion: Der Dialog dient der Konfiguration der Darstellungsoptionen für den FUP/KOP/AWL-Editor.

Aufruf: Menü **Extras** ► **Optionen** ► **IndraLogic 2G**, Kategorie **FUP, KOP und AWL**

Registerkarte 'Allgemeines'

Ansicht

Netzwerktitel anzeigen	Zeigt den Netzwerktitel in der oberen linken Ecke des Netzwerks an.
Netzwerkcommentar anzeigen	Zeigt den Netzwerkcommentar in der oberen linken Ecke des Netzwerks an. Wenn IndraLogic zusätzlich den Netzwerktitel darstellt, erscheint der Commentar in der Zeile unterhalb.
Bausteinsymbol anzeigen	Zeigt das Bausteinsymbol im Bausteinelement im FUP- und KOP-Editor an. Auch die Standardoperatoren haben Symbole.
Operandencommentar anzeigen	zeigt den Commentar an, den Sie einer Variablen im Implementierungsteil gegeben haben. Der Operandencommentar bezieht sich nur auf die lokale Verwendungsstelle der Variablen, im Gegensatz zum "Symbolcommentar". Der Commentar wird abhängig vom verfügbaren Platz automatisch umgebrochen. Sie können den Commentar auf eine definierte Breite begrenzen, indem Sie die Option "Feste Größe für Operandenfelder" aktivieren.
Symbolcommentar anzeigen	zeigt den Commentar oberhalb des Variablennamens an, den Sie einer Variablen oder einem Symbol bei der Deklaration gegeben haben. Zusätzlich oder anstelle des Symbolcommentars können Sie auch einen lokalen "Operandencommentar" zuweisen.
Symboladresse anzeigen	Wenn einem Symbol (Variable) eine Adresse zugewiesen ist, wird diese Adresse oberhalb des Variablennamens angezeigt.
Netzwerktrennlinien anzeigen	zwischen den einzelnen Netzwerken wird eine Trennlinie angezeigt

Verhalten

Platzhalter für neue Operanden	Operandenfeld der Pins der neuen Funktionsbausteinen wird leer gelassen (anstelle von "???").
Leere Operanden für Funktionsbaustein-Pins	Fügt anstelle von ??? leere Operanden ein.

Schriftart

Klick auf das Eingabefeld öffnet den Dialog **Schriftart**.

Feste Größe für Operandenfelder	 Operandengröße bearbeiten ist aktivierbar.
Operandengröße bearbeiten	Dialog Größe Operanden erscheint zum Einstellen der Anzahl Zeichen und Zeilen.

Registerkarte 'FUP'

Ansicht

Netzwerk mit Zeilenumbrüchen	 Darstellung der Netzwerke mit Zeilenumbrüchen, so dass IndraLogic möglichst viele Bausteine in der aktuellen Breite des Fensters darstellen kann.
Bausteine mit geraden Linien verbinden	 Linien zwischen den Elementen erhalten eine feste, kurze Länge.

Verhalten

Netzwerk Inhalt	Auswahlliste: Inhalt eines neuen Netzwerks.
Selektion nach Einfügen	Auswahlliste: Element, das IndraLogic nach dem Einfügen eines neuen Netzwerks selektiert.

Registerkarte 'KOP'**Ansicht**

Netzwerk mit Zeilenumbrüchen	 Darstellung der Netzwerke mit Zeilenumbrüchen, so dass IndraLogic möglichst viele Bausteine in der aktuellen Breite des Fensters darstellen kann.

Verhalten

Netzwerk Inhalt	Auswahlliste: Inhalt eines neuen Netzwerks.
Selektion nach Einfügen	Auswahlliste: Element, das IndraLogic nach dem Einfügen eines neuen Netzwerks selektiert.

Registerkarte 'AWL'**Ansicht**

AWL aktivieren	Die Implementierungssprache AWL steht im Entwicklungssystem zur Verfügung.

Verhalten

Netzwerk Inhalt	Auswahlliste: Inhalt eines neuen Netzwerks.
Selektion nach Einfügen	Auswahlliste: Element, das IndraLogic nach dem Einfügen eines neuen Netzwerks selektiert.

Registerkarte 'Drucken'**Layout-Optionen**

Anpassungsmethode	Auswahlliste zur Größenanpassung.
Abschneiden von Elementen vermeiden	Elemente, die nicht auf die Seite passen, werden auf der nächsten Seite gedruckt.
Verbindungen auf angrenzenden Seiten kennzeichnen	Aktivierbar, wenn Abschneiden von Elementen vermeiden aktiviert ist.

Siehe auch

- [Kap. 17.1 "Programmiersprachen und ihre Editoren" auf Seite 239](#)

Dialog 'Optionen' - 'IndraLogic 1x-Konverter'Symbol 

Funktion: Der Dialog dient der Definition des Abbilds bei der Konvertierung eines IndraLogic 1x-Projekts.

Aufruf: Menü **Extras** ► **Optionen** ► **IndraLogic 2G**, Kategorie **IndraLogic 1x-Konverter**

Geräte

Auflistung und Definition der Geräteabbilder bei einer Konvertierung.

Alle Geräteabbilder dieser Auflistung werden auch bei der nächsten Konvertierung eines IndraLogic 1x-Projekts wirksam. Somit müssen Sie die Abbilddefinition nicht neu erstellen, wenn das gleiche Gerät in mehreren Projekten verwendet wird.

Quellgerät	Doppelklick auf einen Eintrag oder leeres Feld öffnet den Dialog Ursprüngliches Gerät auswählen . IndraLogic listet dort alle Geräte auf, die der Konverter verarbeitet kann.
Zielgerät	Doppelklick auf einen Eintrag oder auf ein leeres Feld öffnet den Dialog Zielsystem auswählen . • Wählen Sie ein Zielsystem aus der Liste der installierten Geräte: Bestätigung der Auswahl mit OK. • Keines: Das Quellgerät wird im neuen Projekt nicht verfügbar sein. Dies bedeutet auch, dass applikationsspezifische Objekte wie zum Beispiel die Taskkonfiguration nicht übernommen werden.

Visualisierungen

Definiert die Handhabung der Darstellungsvariante Target-Visualisierung und Web-Visualisierung beim Öffnen eines IndraLogic 1x-Projekts.

Import der Target-Visualisierung Import der Web-Visualisierung	<i>Auswahlliste</i> • Versuche das alte Verhalten zu imitieren: Option ist verfügbar, wenn das gerade verwendete Gerät die Option unterstützt. Die Option bewirkt, dass die neuen Target-, oder Web-Visualisierungseinstellungen möglichst genau den Einstellungen im alten Projekt entsprechen. • Den Benutzer einmal fragen: Nur beim ersten Import eines "alten" Visualisierungsprojekts fordert IndraLogic den Anwender auf, die Einstellungen zu definieren. Danach wendet IndraLogic die Einstellungen automatisch bei allen weiteren Visualisierungsimports an. • Den Benutzer fragen: Bei jedem Öffnen eines "alten" Visualisierungsprojekts fordert IndraLogic den Anwender auf, die Einstellungen zu definieren • Einstellungen der Gerätebeschreibung: IndraLogic setzt die Einstellungen gemäß den Einträgen in der Gerätebeschreibung des installierten Zielgeräts. • Versuche das alte Verhalten zu imitieren: Option ist nur verfügbar, wenn das gerade verwendete Gerät die Option unterstützt. Die Option bewirkt, dass die neuen Darstellungsvarianteneinstellungen für Target- oder Webvisualisierung möglichst genau den Einstellungen im alten Projekt entsprechen.



Target-Visualisierung unterstützt nicht den Zugriff auf VAR_IN_OUT-Variablen, da dies der Compiler nicht unterstützt. Dieses Verhalten entspricht dem Verhalten von IndraLogic 1x-Target-Visualisierungen.

Siehe auch

- [Kap. 2.4 "IndraLogic 2G-Projekt öffnen" auf Seite 10](#)

Dialog 'Optionen' - 'Bibliotheken'

Symbol: 

Funktion: Der Dialog dient dem Verwalten der Abbildungen von Bibliotheksreferenzen, die IndraLogic bei der Konvertierung eines alten Projekts anwendet. Wenn Sie für eine bestimmte Bibliothek noch keine Abbildung gespeichert haben, müssen Sie bei jedem Öffnen eines alten Projekt, in das diese Bibliothek eingebunden ist, die Abbildung neu definieren.

Aufruf: Menü **Extras** ► **Optionen** ► **IndraLogic 2G**, Kategorie **Bibliotheken**

Eine Abbildung definiert, wie eine Bibliotheksreferenz nach der Konvertierung des Projekts ins aktuelle Format aussieht. Es gibt drei Möglichkeiten:

- Sie behalten die Referenz bei. Das bedeutet, dass IndraLogic die Bibliothek ebenfalls ins aktuelle Format (*.library) konvertiert und im lokalen Bibliotheks-Repository installiert.
- Sie ersetzen eine Referenz durch eine andere Referenz. Das bedeutet, dass eine der installierten Bibliotheken die bisher eingebundene Bibliothek ersetzt.
- Sie löschen die Referenz. Das bedeutet, dass das konvertierte Projekt die Bibliothek nicht mehr einbindet.

Zielsystembibliothek setzen

IndraLogic wendet alle aufgelisteten Abbildungen bei der nächsten Konvertierung eines alten Projekts auf dessen Bibliotheksreferenzen an.

Somit müssen Sie die Abbildungsdefinition wiederholen, wenn dieselbe Bibliothek wieder in einem zu konvertierenden Projekt eingebunden ist.

In der letzten Zeile können Sie eine neue Abbildung eingeben.

Quellbibliothek	Pfad der Bibliothek, die im Projekt vor der Konvertierung eingebunden ist. Ein Doppelklick auf einen Eintrag macht das Feld editierbar und die Schaltfläche für die Eingabeunterstützung erscheint.
Zielbibliothek	Name und Speicherort der Bibliothek, die nach der Konvertierung im Projekt eingebunden sein soll. Ein Doppelklick auf einen Eintrag öffnet den Dialog Zielsystembibliothek setzen .

Dialog 'Optionen' - 'Refactoring'

Symbol: 

Funktion: Der Dialog dient zur Festlegung der Operationen im Projekt, für die automatisch Refactoring vorgeschlagen wird. Die Refactoring-Funktionalität unterstützt Sie bei ihren Verbesserungswünschen.

Aufruf: Menü **Extras** ► **Optionen** ► **IndraLogic 2G** ► **Refactoring**.

Refactoring für folgende Operationen vorschlagen

Autodeklaration	
Wenn Sie in einer Deklaration den Namen einer Variablen ändern, indem Sie die Autodeklaration ([Umschalt]+[F2]) aufrufen, erscheint die aktivierte Option "Änderungen mit Hilfe von Refactoring anwenden". Dann öffnet sich der Dialog "Refactoring" und Sie können projektweit die Variable ändern.	
Beim Umbenennen von Variablen	 Beim Umbenennen von Variablen, erscheint die Eingabeaufforderung, ob IndraLogic "Automatisches Refactoring" durchführen soll.

Für das Hinzufügen oder Entfernen von Variablen oder für das Ändern des Namensraumes	 : Sie löschen in der Autodeklaration (Dialog "Variable deklarieren") den Namen und beenden den Dialog mit OK . Daraufhin öffnet sich der Dialog "Refactoring", um projektweit die Variable zu entfernen.
Editor für Einheitenumrechnung	
Beim Umbenennen von Einheitenkonvertierungen	 : Wenn Sie im Editor für Einheitenumrechnung den Namen einer Konvertierung ändern, erscheint die Eingabeaufforderung, ob IndraLogic "Automatisches Refactoring" für das Umbenennen durchführen soll.
Mapping-Editor	
Beim Umbenennen von Variablen	 : Wenn Sie im Geräteeditor, Registerkarte E/A-Abbild einen Variablennamen ändern, erscheint die Eingabeaufforderung, ob IndraLogic "Automatisches Refactoring" für das Umbenennen durchführen soll.
Navigator	
Beim Umbenennen von Objekten	 : Wenn Sie im Projektexplorer oder im Ordner "Allgemeiner Baustein" den Namen eines Objekts ändern, erscheint die Eingabeaufforderung, ob IndraLogic "Automatisches Refactoring" für das Umbenennen durchführen soll.
Tabellarischer Deklarationseditor	
Beim Umbenennen von Variablen	 : Wenn Sie im tabellarischen Deklarationseditor den Namen einer Variablen ändern, erscheint die Eingabeaufforderung, ob IndraLogic "Automatisches Refactoring" für das Umbenennen durchführen soll.

Siehe auch

- [Kap. 9.14 "Refactoring" auf Seite 139](#)
- [Kap. 9 "Applikation programmieren" auf Seite 85](#)
- [Kap. "Befehl 'Refactoring Umbenennen'" auf Seite 617](#)

Dialog 'Optionen' - 'Sichere FUP-Optionen'

Funktion: Der Dialog dient der Erweiterung der Einstellungen für das Editieren und Drucken im FUP-Editor bei Safety-Anwendungen.

Aufruf: Menü **Extras ▶ Optionen ▶ IndraLogic 2G**, Kategorie **Sichere FUP-Optionen**

Darstellung der Verbindung der FUP-Elemente	
Bausteine 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 genügend 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.
Optionale Kommentierung des Codes Zusätzlich zu den Objektkommentaren kann der Programmcode kommentiert werden. Fehlende Code-Kommentierung erzeugt keine Warnungen. Die Auswahl der optionalen Code-Kommentierung erfolgt in diesem Dialog	

Netzwerktitel anzeigen	☒ : Jedem Netzwerk des Programmcodes der POU's der Sicherheitsapplikationen eines Projekts kann ein Titel gegeben werden. Der Netzwerktitel wird in der ersten Zeile des FUP-Netzwerks editiert.
Netzwerkcomment anzeigen	☒ : Jedem Netzwerk des Programmcodes der POU's der Sicherheitsapplikationen eines Projekts kann ein Kommentar gegeben werden. Der Netzwerkcomment wird in der zweiten Zeile des FUP-Netzwerks editiert . (Ausnahme: Falls die Option "Netzwerktitel anzeigen" nicht gesetzt ist, wird der Netzwerkcomment in der ersten Zeile editiert.) Abb. 18-74: Netzwerktitel und Netzwerk-Commentar
In POU's und GVL's	Kommentar zu jeder anwenderdefinierten Variablendeklaration (optional)
Symbolcomment anzeigen	☒ : Variablencommente, die im Deklarationsfenster des FUP-Editors eingegeben wurden, werden im Implementierungsfenster oberhalb der Verwendungsstelle der Variablen angezeigt. Abb. 18-75: Symbolcomment, hier aus POU-Deklaration

Dialog 'Optionen' - 'AS-Editor'

Symbol:

Funktion: Der Dialog dient der Konfiguration der Einstellungen für den AS-Editor.

Aufruf: Menü **Extras** ► **Optionen** ► **IndraLogic 2G**, Kategorie **AS-Editor**

Siehe auch

- [Kap. "Programmieren in Ablaufsprache \(AS / SFC\)" auf Seite 106](#)
- [Kap. 18.2.7 "Menü 'AS'" auf Seite 682](#)

- [Kap. "AS-Editor" auf Seite 253](#)

Registerkarte 'Anordnung'

Elemente

Definiert die Größen der AS-Elemente Schritt, Aktion, Qualifizierer und Eigenschaft. Angabe der Werte in Rastereinheiten angegeben. 1 Rastereinheit = Fontgröße, die Sie aktuell in den Texteditor-Optionen gesetzt haben (Textbereich / Schriftart). Die Einstellungen werden immer sofort in allen gerade geöffneten AS-Editorfenstern wirksam.	
Schritthöhe	Mögliche Werte: 1-100
Schrittbreite	Mögliche Werte: 2-100
Aktionsbreite	Mögliche Werte: 2-100
Qualifiziererbreite	Mögliche Werte: 2-100
Eigenschaftenbreite	Mögliche Werte: 2-100

Zeichensatz

Der Beispielttext zeigt die aktuell eingestellte Schriftart. Klicken Sie darauf, um die Schriftart zu verändern.

Schrittaktionen

Standardeinfügemethode	• "Referenz kopieren": Die Referenz zu den Aktionsobjekten, die der Schritt aufruft, werden beim Kopieren des Schritts mit kopiert. Kopierter und neuer Schritt rufen dann dieselbe Aktion auf. • "Implementierung duplizieren": Die Referenz zu den Aktionsobjekten, die der Schritt aufruft, werden fest mit diesem Schritt verbunden. Beim Kopieren des Schrittelements werden für den neuen Schritt neue Aktionsobjekte angelegt, die Implementierung wird also dupliziert. • "Immer fragen": Beim Einfügen einer Schrittaktion werden Sie immer gefragt, ob beim Kopieren des Schrittelements seine Aktionen dupliziert werden sollen, oder die Referenz auf die bestehende Aktion übernommen werden soll. Hinweis: Wenn ein Schritt bereits eine eingebettete Aktion enthält, werden neu eingefügte Aktionen dieses Schrittes ebenfalls eingebettet. Umgekehrt werden neu eingefügte Aktionen nicht eingebettet, wenn der Schritt bereits eine nicht eingebettete Aktion enthält. In diesen Fällen werden Sie nicht mehr nach dem gewünschten Duplizierungsmodus gefragt.

Eingebettete Objekte

Aktions- und Transitionsobjekte im Navigator anzeigen	 Aktions- und Transitionsobjekte, die im AS-Baustein fest mit einem Schritt verknüpft sind ("eingebettet"), werden in der Baumansicht "Geräte" oder "POUs" dargestellt.

Registerkarte 'Ansicht'

Sichtbarkeit von Eigenschaften

Auflistung der Elementeeigenschaften der Kategorien Allgemein und Spezifisch und Definition der Anzeigoptionen.	
Eigenschaft	Definiert die Elementeeigenschaften, die neben dem Element im AS-Diagramm stehen.

Wert	 : Anzeige des Eigenschaftswerts.
Mit Namen	 : Anzeige des Eigenschaftswerts mit Namen.

Online

Schrittzeit anzeigen	 : zeigt im Online-Betrieb rechts von den Schritten die Schrittzeit an.

Dialog 'Optionen' - 'Intelligentes Codieren'

Symbol: 

Funktion: Der Dialog dient der Konfiguration der Einstellungen, die die Eingabe von Code erleichtern.

Aufruf: Menü **Extras** ► **Optionen** ► **IndraLogic 2G**, Kategorie **Intelligentes Codieren**

Unbekannte Variablen automatisch deklarieren (AutoDeclare)	 : Der Dialog Variable deklarieren öffnet sich, sobald Sie einen noch nicht deklarierten Bezeichner in einem Programmiersprachen-Editor eingegeben und die Eingabezeile verlassen haben. Damit die AutoDeclare-Funktion auch im ST-Editor zur Verfügung steht, muss zusätzlich die Option "Für den ST-Editor aktivieren" aktiviert sein.
Für den ST-Editor aktivieren	Voraussetzung: Die Option "Unbekannte Variablen automatisch deklarieren (AutoDeclare)" ist aktiviert.  : Die AutoDeclare-Funktion steht auch im ST-Editor zur Verfügung.  : Im ST-Editor steht die AutoDeclare-Funktion nicht zur Verfügung.
Alle Variablen einer Instanz in der Eingabehilfe anzeigen	 : Die Funktion "Komponenten auflisten" bietet auch die lokalen Variablen einer Funktionsbausteininstanz zur Auswahl an.  : Die Funktion "Komponenten auflisten" bietet nur die Eingangsvariablen und Ausgangsvariablen einer FB-Instanz zur Auswahl an.
Symbole von Systembibliotheken in der Eingabehilfe anzeigen	Systembibliotheken werden im Bibliotheksverwalter automatisch eingefügt und hellgrau dargestellt.  : Symbole wie globale Variablen, Datentypen, Funktionsbausteine, etc werden in der Eingabehilfe angeboten.  : Die Symbole der Systembibliotheken sind in der Eingabehilfe nicht verfügbar.
Komponenten auflisten, nachdem ein Punkt (.) eingegeben wurde	 : Aktiviert die Funktion "Komponenten auflisten". Das bedeutet: Wenn Sie an einer Stelle, an der IndraLogic einen Bezeichner erwartet, einen Punkt . eingeben, erscheint eine Auswahlliste mit Eingabemöglichkeiten.
Komponenten sofort nach Eingabe eines Zeichens auflisten	Voraussetzung: Option Komponenten auflisten, nachdem ein Punkt (.) eingegeben wurde ist aktiviert.  : Nach Eingabe einer beliebigen Zeichenfolge erscheint eine Auswahlliste der verfügbaren Bezeichner und Operatoren

	Mit Namensraum einfügen	 : Vor dem Kennzeichner fügt IndraLogic automatisch den Namensraum mit ein.
	Schlüsselworte automatisch in Großschreibung umwandeln (Autoformat)	 : IndraLogic schreibt automatisch alle Schlüsselwörter in Großbuchstaben.
	Querverweise automatisch bei Selektionsänderung aktualisieren	 : Die Querverweisliste zeigt automatisch die Referenzen der Variablen/ POU's/ DUTs an, die Sie gerade selektieren oder in welcher der Cursor steht.
	Fehler im Editor unterstreichen	 : Fehlerhafter oder unbekannter Programmcode wird unterstrichen.
	Symbole hervorheben	 : Alle Verwendungsstellen eines Symbols, auf dem der Cursor steht, werden innerhalb des Editors farblich markiert. So werden Querverweise innerhalb des Editors schnell erkannt.
	Maximaler Parallelitätsgrad	Auswahlliste für die Anzahl der parallelen Threads, die für die Precompile- Verarbeitung verwendet werden dürfen. IndraLogic ermittelt die angezeigte Anzahl Threads aus der Anzahl der CPU-Kerne des Prozessors. Diese voreingestellte Anzahl sollte nur in Ausnahmefällen verändert werden.

Siehe auch

- [Kap. 17.1 "Programmiersprachen und ihre Editoren" auf Seite 239](#)
- [Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112](#)
- [Kap. 9.12.1 "Verwendungsstellen mit der Querverweisliste finden" auf Seite 135](#)

Dialog 'Optionen' - 'Syntaxhervorhebung'

Symbol: 

Funktion: Der Dialog dient der Konfiguration der Farb- und Schrifteinstellungen für die Textelemente eines Editors (zum Beispiel Operanden, Pragmas).

Aufruf: Menü **Extras** ► **Optionen**, Kategorie **Syntaxhervorhebung**

Dialog 'Optionen' - 'Texteditor'

Symbol



Funktion Der Dialog enthält Einstellungen für das Arbeiten in einem Texteditor.

Aufruf Menü **Extras** ► **IndraLogic 2G** ► **Optionen**, Kategorie **Texteditor**

Registerkarte 'Editieren'

Anzahl von Rückgängig-Schritten	Maximale Anzahl der Bearbeitungsschritte, auf die Sie den Befehl Bearbeiten ► Rückgängig ausführen können.
Gliederung	Definiert die Strukturierung des Codes durch Einrückungen. Wenn Sie eine Einrückung auswählen, können Sie den Einrückungsabschnitt mit Hilfe eines Plus- und Minuszeichens vor der ersten Zeile des jeweiligen Abschnitts auf- oder zuklappen. • Einrückung: IndraLogic fasst alle Zeilen, die gegenüber der vorausgehenden Zeile eingerückt sind, in einer Einrückungseinheit zusammen. • Explizit: Sie kennzeichnen explizit den Code-Abschnitt mit Kommentaren, der in einer Einrückungseinheit zusammengefasst werden soll: Vor dem Abschnitt muss ein Kommentar stehen, der 3 öffnende geschweifte Klammern "{{{" enthält, nach dem Abschnitt muss ein Kommentar folgen, der 3 schließende geschweifte Klammern "}}}" enthält. Die Kommentare können zusätzlichen Text enthalten. Beispiel: <pre> 1 IF iVar1=1 2 ///comment {{{ 3 THEN 4 iVar2:=2; 5 ELSE iVar2:=10; 6 END_IF 7 ///}} 8 iVar1:=iVar1+1; </pre> <pre> 1 IF iVar1=1 2 ///comment {{{ [5 lines] 8 iVar1:=iVar1+1; </pre>
Wortumbruch	• Weich: Der Zeilenumbruch erfolgt am Rand des Editorfensters, wenn bei Umbruchsstelle 0 eingetragen ist. • Hart: Der Zeilenumbruch erfolgt nach der bei Umbruchsstelle angegebenen Anzahl von Zeichen.
Tabulatorgröße	Anzahl der Zeichen
Tabs beibehalten	☒ : Den Leerraum, den Sie mit der <Tabulator>-Taste eingefügt haben, löst IndraLogic hinterher nicht in einzelne Leerzeichen auf.

Referenz Benutzeroberfläche

Einzugsgröße	Wenn Sie die Option Automatisch Einrücken Intelligent oder Intelligent mit Code-Komplettierung aktiviert haben, fügt IndraLogic die Anzahl Leerzeichen am Beginn der Zeile ein.
Automatisch einrücken	• Nicht automatisch einrücken • Block: Eine neue Zeile übernimmt automatisch die Einrückung der Vorgängerzeile. • Intelligent: Zeilen, die einer Zeile folgen, die ein Schlüsselwort enthält (zum Beispiel <code>VAR</code>), rücken automatisch um die angegebene "Einzugsgröße" ein. • Intelligent mit Code-Komplettierung: Einrückung wie bei der Option Intelligent, zusätzlich fügt IndraLogic das abschließende Schlüsselwort ein (zum Beispiel <code>END_VAR</code>).

Registerkarte 'Textbereich'

Aktuelle Zeile hervorheben	Der Hintergrund der aktuellen Zeile wird farbig gekennzeichnet
Zusammengehörige Klammern	Wenn der Cursor vor oder nach einer Klammer innerhalb einer Code-Zeile positioniert ist, markiert IndraLogic die zugehörige schließende oder öffnende Klammer durch einen Rahmen in der eingestellten Farbe.
Zeilenendemarkierungen	Das Ende jeder Editorzeile markiert IndraLogic durch einen kleinen Querstrich in der eingestellten Farbe hinter dem letzten Zeichen (auch Leerzeichen) der Zeile.
Zeilenumbruchstelle:	Wenn ein weicher oder harter Zeilenumbruch aktiviert ist, wird die definierte Zeilenumbruchstelle durch eine senkrechte Linie in der gewählten Farbe angezeigt.
Caret-Farbe	Farbe des Cursorzeichens
Selektionsfarbe	Farbe des selektierter Textbereichs
Inaktiv	Farbe einer Selektierung, wenn das zugehörige Fenster nicht aktiv ist (Fokus liegt auf einem anderen Fenster).
Vordergrund reduzierte Zeile	Farbe der Kopfzeile eines geschlossenen, eingerückten Abschnitts im Code
Hintergrund reduzierte Zeile	Kopfzeile eines geschlossenen, eingerückten Abschnitts im Code wird in der Farbe hinterlegt.
Schriftart	Ein Klick auf das Feld öffnet den Standarddialog zum Konfigurieren der Schriftart.

Registerkarte 'Randbereich'

Einstellungen für linken Randbereich des Texteditor-Fensters, der durch eine senkrechte Linie vom Eingabebereich abgetrennt ist:	
Zeilen nummerieren	Anzeige der Zeilennummern im Deklaration- und Implementierungsteil, jeweils mit 1 beginnend
Vordergrundfarbe	Farbe der Zeilennummern
Hintergrundfarbe	Farbe des Randbereichs
Aktuelle Zeile hervorheben	Zeilennummer der aktuellen Zeile wird farbig gekennzeichnet
Klammerung	Eine Klammerung umfasst die Zeilen zwischen den Schlüsselwörtern, die ein Konstrukt öffnen und abschließen, zum Beispiel <code>IF</code> und <code>END_IF</code>. Wenn die Option aktiviert ist und der Cursor vor, nach oder in einem der Schlüsselwörter eines Konstrukts steht, wird der Klammerungsbereich durch eine eckige Klammer im Randbereich angezeigt. Die Farbe der Klammerung können Sie aus der Auswahlliste dieser Option auswählen.
Aktive Randfarbe	Farbe der Trennlinie zwischen Rand- und Eingabebereich

Inaktive Randfarbe	Farbe der Trennlinie zwischen Rand- und Eingabebereich des gerade nicht aktiven Teils des Fensters
Mausaktionen	Eine der folgenden Aktionen können Sie jeder der angegebenen Mausaktionen oder Maus-Tastenkombinationen zuordnen. IndraLogic führt die Aktionen aus, wenn Sie die Mausaktion auf das Plus- oder Minuszeichen vor der Kopfzeile eines geklammerten Bereichs ausführen: • Gliederung markieren: IndraLogic wählt alle Zeilen des geklammerten Bereichs aus. • Gliederung umschalten: IndraLogic öffnet oder schließt den geklammerten Bereich, oder wenn geschachtelte Klammerungen vorliegen, die erste Ebene des geklammerten Bereichs. • Gliederung vollst. umschalten: IndraLogic öffnet oder schließt alle Ebenen eines geschachtelt geklammerten Bereichs.

Registerkarte 'Monitoring'

Einstellungen für die Darstellung der Monitoring-Felder	
Inline-Monitoring aktivieren	Anzeige der Monitoring-Felder hinter den Variablen im Online-Modus
Monitoring Vordergrundfarbe	Darstellung des Werts im Monitoring-Feld
Monitoring Hintergrundfarbe	Darstellung des Hintergrunds im Monitoring-Feld
Ablaufkontrolle Vordergrundfarbe	Darstellung des Werts in den Monitoring-Feldern an den Ablaufkontroll-Positionen
Ablaufkontrolle Hintergrundfarbe	Darstellung des Hintergrunds der Monitoring-Felder an den Ablaufkontroll-Positionen
Anzahl der dargestellten Stellen	Anzeige-Gesamtlänge (1-20) im Monitoring-Feld, bestehend aus Vorkommastellen + Nachkommastellen. Beispiel für den Variablenwert "1234.5678": • Anzeige bei 3 Stellen (Default) : 1.23E+03 • Anzeige bei 5 Stellen: 1234.6 • Anzeige bei 7 Stellen: 1234.568
String-Länge	Maximale Länge von String-Variablenwerten im Monitoring-Feld

Siehe auch

- [Kap. "Programmieren in Strukturiertem Text \(ST\)" auf Seite 105](#)

Dialog 'Optionen' - 'Visualisierung'

Funktion: In diesem Dialog, werden allgemeine Einstellungen zum Arbeiten mit Visualisierungen in IndraLogic 2G vorgenommen.

Aufruf: Menü: **Extras ▶ Optionen ▶ IndraLogic 2G ▶ Visualisierung.**

Allgemein

Darstellungsoptionen (Visualisierungseditor im Entwicklungssystem)	
Zu definieren ist hier, wie die Bilddatei auf Änderungen der Elementrahmengröße reagieren soll.	
Fest	Das Bild behält die Originalgröße, auch wenn sich die Größe des Elementrahmens ändert.
Isotropisch	Das Bild behält seine Proportionen; d.h. das Verhältnis Höhe-Breite bleibt erhalten, auch wenn Höhe und Breite des Elementrahmens separat verändert werden.
Anisotropisch	Das Bild passt sich der Größe des Elementrahmens an; d.h. Höhe und Breite können unabhängig voneinander verändert werden.

Referenz Benutzeroberfläche

Antialiased Zeichnen	 : Aktivieren Sie diese Option, wenn beim Zeichnen der Visualisierungen im Visualisierungseditor-Fenster des Programmiersystems Antialiasing (Offline oder Online) verwendet werden soll (Kantenglättung, Verminderung von unerwünschten Effekten, die durch das begrenzte Pixelraster (Bildauflösung) entstehen können (Alias-Effekt mit dem Pixelraster) oder die durch den Treppeneffekt bei der Erzeugung einer Computergrafik (computergenerierte 2D oder 3D Grafiken) entstehen).

Bearbeitungsoptionen	
Mit Umschalt- und Tastenvariablen verknüpfen	 : Platzhalter <Umschalten-/TastenvARIABLE> in den Visualisierungselementeigenschaften aktiviert. Wenn Sie ein Element, das über die Eigenschaft Farbvariablen, Farbumschlag verfügt, in die Visualisierung ziehen, konfiguriert IndraLogic diese Eigenschaft mit dem Platzhalter <Umschalten-/TastenvARIABLE>. Folgende Elemente sind betroffen: Schaltfläche, Frame, Bild, Linie, Kreissektor, Polygon, Rechteck, Textfeld, Scrollbalken

Raster

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.

Dateioptionen

Dateioptionen	
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 (Kap. "Befehl 'Import/Export Textlisten'" auf Seite 729). Ihr Aufbau ist tabellenförmig und muss der Struktur von Textlisten folgen, die auf der Hilfeseite zu Textlisten beschrieben ist (Kap. 18.1.23 "Objekt 'Textliste'" auf Seite 588).
Visualisierungsverzeichnisse	

Textlisten-Dateien	Öffnen Sie mit der Schaltfläche  den Dialog zur Definition eines existierenden oder eines neuen Verzeichnisses für die Sprachdateien, die im Projekt zur Text- und Sprachenkonfiguration in der Visualisierung verwendet werden sollen. Die Dateien, die in diesem Verzeichnis gefunden werden, sind dann 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. Sehen Sie zu Text- und Sprachenkonfiguration auch die Beschreibung in Kap. 18.1.23 "Objekt 'Textliste'" auf Seite 588.
Bilddateien	Öffnen Sie mit der Schaltfläche  den Dialog zur Definition eines existierenden oder eines neuen Verzeichnisses für die Bilddateien, die im Projekt für die Verwendung in Visualisierungen bereitstehen sollen. Hier können Sie mehrere Verzeichnisse in einer Strichpunkt-separierten Liste angeben.

Dialog 'Optionen' - 'Visualisierungsstile'

Funktion: Der Dialog dient der Definition des Visualisierungsstils, der in Visualisierungen in Bibliotheken und globalen Visualisierungen ("Allgemeiner Bausteinteil") in Projekten verwendet werden soll.

Aufruf: Menü **Extras ▶ Optionen ▶ IndraLogic 2G**, Kategorie **Visualisierungsstile**



Im Online-Modus ist nicht die aktuelle Einstellung im Optionendialog wirksam, sondern die, die im Visualisierungsmanager vorliegen.

Stilkonfiguration für Bibliotheken und globale Visualisierungen	
Keinen Visualisierungsstil verwenden	Wenn diese Option aktiviert ist, verhalten sich Visualisierungen in Bibliotheken oder im "Allgemeinen Bausteinteil" so als ob kein Stil zugewiesen wurde.
Folgenden Visualisierungsstil verwenden	Wenn diese Option aktiviert ist, verhalten sich Visualisierungen in Bibliotheken oder im "Allgemeinen Bausteinteil" gemäß dem ausgewählten Stil. Die Auswahlliste stellt alle aktuell installierten Stile bereit.
Visualisierungsstil ableiten automatisch	Wenn diese Option aktiviert ist, versuchen Visualisierungen in Bibliotheken oder im "Allgemeinen Bausteinteil" zunächst, den im Visualisierungsmanager des aktuellen Projekts eingestellten Stil soweit als möglich anzuwenden. Wenn das nicht möglich ist, wird der in „Ausweichlösung, falls kein Visualisierungsstil abgeleitet werden konnte“ angegebene verarbeitet.



Die Stileinstellungen die für alle Visualisierungen der aktuellen Applikation gültig sind, müssen in 'Visualisierungsmanager' - 'Einstellungen' konfiguriert werden.

([Kap. "Visualisierungsmanager- 'Einstellungen'" auf Seite 1058](#))

'Stilauswahl'

Alle Versionen anzeigen (nur für Experten)

- : Die Auswahllisten für Stile enthalten den ausgewählten Stil und alle weiteren Stile, die im Visualisierungsstil-Repository installiert sind, aber nur die neueste Version. Wenn für den ausgewählten Stil eine neuere Version installiert ist, wird diese auch aufgelistet.
- : Die Auswahllisten für Stile enthalten alle installierten Stile mit allen Versionen.

Siehe auch

- [Kap. "Visualisierungsmanager- 'Einstellungen'" auf Seite 1058](#)

Stil für neue Visualisierungsmanager

Diese Option ermöglicht eine Vorauswahl des Stils, der für neue Visualisierungsapplikationen verwendet werden soll.



Bitte beachten Sie, dass sich bei einer Standardauswahl der konkret verwendete Stil, in Abhängigkeit von der Funktionalität des verwendeten Gerätes, unterscheiden kann.

Siehe auch

- [Kap. "Visualisierungsmanager- 'Einstellungen'" auf Seite 1058](#)

18.3.10 Dialog 'Tracekonfiguration'

Dialog 'Erweiterte Traceeeinstellungen'

Funktion: Der Dialog bietet erweiterte Einstellungen für die Datenerfassung.

Aufruf: Schaltfläche **Erweitert** im Dialog **Tracekonfiguration**, Teildialog **Aufzeichnungseinstellungen**.

Voraussetzung: Der Traceeditor ist geöffnet und aktiv. Der Dialog "Tracekonfiguration" wurde geöffnet und der oberste Knoten des Traceaufzeichnungsbaums ist selektiert, so dass der Teildialog "Aufzeichnungseinstellungen" verfügbar ist.



Für die Berechnung der Werte müssen Sie im Dialog "Tracekonfiguration" eine Task auswählen.

Die Größe des Puffers wird als "Anzahl an Samples" angegeben. IndraLogic berechnet die Zeitintervalle, die dieser Anzahl entsprechen und zeigt das Ergebnis rechts, außerhalb der Tabelle in normierter Schreibweise an, zum Beispiel **1h1m1s1ms**. Die Berechnung ist nur mit Hilfe von Einstellungen der Taskkonfiguration möglich und wenn die Taskzykluszeit bekannt ist.

Messung in jedem n-ten Zyklus	Datenerfassung alle n Taskzyklen Voreinstellung: 1, dann führt die Applikation in jedem Taskzyklus eine Datenerfassung durch.	Abtastintervall der Datenerfassung. Beispiel: 100ms
Empfohlene Größe des LZS-Puffers (Samples)	Voraussetzung: "Überschreiben der empfohlenen Puffergröße mit" ist deaktiviert Die von IndraLogic berechnete und empfohlene Anzahl an Samples, die die Applikation zur Laufzeit pro Tracevariable maximal speichert. IndraLogic berechnet die Anzahl aus der Taskzykluszeit und aus dem Wert in "Messung in jedem n-ten Zyklus".	Maximale Länge des Zeitintervalls, währenddessen die Applikation auf dem Laufzeitsystem die Daten erfasst. Beispiel: 2s

Überschreiben der empfohlenen Puffergröße mit	Anzahl an Samples pro Tracevariable, die die Applikation zur Laufzeit pro Tracevariable maximal speichert. Beispiel: 100 Wertebereich: ab 10 : Die Applikation verwendet nicht den von IndraLogic berechneten Wert aus Empfohlene Größe des LZS-Puffers (Samples), sondern den hier eingetragenen Wert.	Maximale Länge des Zeitintervalls, währenddessen die Applikation auf dem Laufzeitsystem die Daten erfasst. Beispiel: 6s
Traceeditor-Puffergröße pro Variable (Samples)	Anzahl der Werte, die pro Variable im Traceeditor gespeichert werden können. Beispiel: 10000	Aus der maximalen Anzahl und dem Abtastintervall der Datenerfassung ergibt sich die maximale Zeitdauer für die Anzeige im Traceeditor. Sie können maximal diese Zeit im Traceeditor zurückscrollen.

Siehe auch

- [Kap. 18.1.27 "Objekt 'Trace'" auf Seite 605](#)
- [Kap. 13.3.2 "Tracekonfiguration erstellen" auf Seite 219](#)

Dialog 'Tracekonfiguration'

Symbol: 

Funktion: Der Dialog enthält die Tracekonfiguration für die Datenerfassung.

Aufruf

- IndraWorks Menü **Trace ► Konfiguration**, Kontextmenü
- Link **Konfiguration** im Traceeditor
- Link **Variable hinzufügen** im Traceeditor

Voraussetzung: Der Editor eines Traceobjekts ist geöffnet und aktiv.

Baumansicht 'Tracekonfiguration'

Die Baumansicht listet die Variablen auf, die im Trace angezeigt werden und ermöglicht den Zugriff auf die Variableneinstellungen.

Selektierter Tracename	Rechts ist der Teildialog "Aufzeichnungseinstellungen" verfügbar.
Selektierte Tracevariable	Rechts ist der Teildialog "Variableneinstellungen" verfügbar.

Kontextmenübefehle

+ Variable hinzufügen	Fügt eine neue Tracevariable hinzu. Rechts erscheint der Teildialog Variableneinstellungen unvollständig konfiguriert. Wählen Sie im Eingabefeld der Einstellung Variable eine Variable aus, um deren Werteverlauf zu tracen
Einem Diagramm zuordnen	Listet (im Untermenü rechts) diejenigen Diagramme auf, in denen die selektierte Variable aktuell nicht dargestellt wird. Wählen Sie ein Diagramm aus, um darin die Variable darzustellen. Der Befehl ist verfügbar, wenn in der Baumansicht eine Variable selektiert ist. Tipp: Wenn der Befehl deaktiviert ist, dann wird die Variable bereits in allen Diagrammen dargestellt.
Aktiviert	Standardmäßig aktiviert Deaktivierte Variablen werden ausgegraut dargestellt. Sie werden weder dargestellt noch aufgezeichnet.

Baumansicht 'Darstellung (Diagramme)' Die Baumansicht listet die Diagramme auf, die im Traceeditor dargestellt werden und ermöglicht den Zugriff auf ihren Anzeigemodus.

Selektierter Knoten Zeitachse	Rechts ist der Teildialog Anzeigemodus für die Zeitachse verfügbar. Sie können die Zeitachsendarstellung bestimmen. Siehe unten.
Selektierter Diagrammname	Rechts erscheinen die Einstellungen für das Koordinatensystem des Diagramms und eine Vorschau. Siehe unten.
Selektierter Knoten Y-Achse	Rechts ist der Teildialog Anzeigemodus verfügbar. Sie können die Achsendarstellung bestimmen. Siehe unten.
Variablen anzeigen	
Selektierte Tracevariable	Rechts ist der Teildialog Variableneinstellungen verfügbar. Sie können die Tracevariable konfigurieren. Hinweis: Das sind die gleichen Einstellungen, auf die auch in der Baumansicht Traceaufzeichnung zugegriffen werden kann.

Kontextmenübefehle

+ Diagramm hinzufügen	Fügt ein neues Diagramm hinzu und zeigt es in der Baumansicht Darstellung (Diagramme) an.
+ Neue Variable hinzufügen	Fügt eine neue Tracevariable hinzu. Rechts erscheint im Teildialog Variableneinstellungen eine unvollständige Konfiguration. Wählen Sie im Eingabefeld der Einstellung Variable eine Variable aus, um deren Werteverlauf zu tracen. Bestimmen Sie deren Darstellung. Außerdem wird die Variable dem selektierten Diagramm zugeordnet.
Bestehende Variable hinzufügen	Listet (im Untermenü rechts) alle Tracevariablen auf, die im selektierten Diagramm aktuell nicht dargestellt werden. Wählen Sie eine Variable aus, um diese im selektierten Diagramm darzustellen. Tipp: Wenn der Befehl deaktiviert ist, dann sind bereits alle Tracevariablen im selektierten Diagramm dargestellt.

Teildialog 'Aufzeichnungseinstellungen' **Voraussetzung:** In der Baumansicht Traceaufzeichnung ist der oberste Knoten selektiert.

Trigger aktiviert	 Die Triggerauslösung ist aktiviert. Zur Laufzeit werden die Tracedaten erst dann gepuffert, wenn ein Triggersignal ausgelöst wurde. Wie das Triggersignal auslöst, kann in den Einstellungen Triggervariable , Triggerparameter , Triggerflanke , Post-Trigger in Samples und Triggerlevel bestimmt werden.  Fortlaufende Darstellung der aktuellen Datensätze
Triggervariable	Signal, das als Trigger verwendet wird. Ein vollständiger Instanzpfad ist erforderlich. Ein gültiges Triggersignal ist eine IEC-Variable, eine Eigenschaft, eine Referenz, ein Pointer, ein Arrayelement der Applikation oder ein Ausdruck. Erlaubte Typen sind alle IEC basierten Typen außer <code>STRING</code> , <code>WSTRING</code> oder <code>ARRAY</code> . Enumerationen sind erlaubt, wenn der Basistyp kein <code>STRING</code> , <code>WSTRING</code> oder <code>ARRAY</code> ist. Der Inhalt eines Pointers ist kein gültiges Signal. Wenn das LZS die Komponente <code>CmpTraceMgr</code> verwendet, können Sie eine Eigenschaft, die mit dem Attribut <code>'monitoring'</code> belegt ist, als Triggervariable auswählen.

Triggerparameter	Parameter des Systems, der als Trigger verwendet wird Der Dialog Eingabehilfe , Registerkarte Kategorien , Eintrag Parameter listet alle gültigen Parameter des Systems auf.
	Ermöglicht die Auswahl von Triggervariable oder Triggerparameter
Triggerflanke	<i>Definiert die Flankenerkennung, die die Triggerung auslöst:</i> • positiv – Bei einer boolschen Triggervariable wird der Trigger ausgelöst, sobald der Wert von FALSE nach TRUE wechselt. – Bei analogem Datentyp wird der Trigger ausgelöst, sobald der Wert, der in "Triggerlevel" definiert ist, "von unten her" erreicht wird. • negativ – Bei einer boolschen Triggervariable wird der Trigger ausgelöst, sobald der Wert von TRUE nach FALSE wechselt. – Bei analogem Datentyp wird der Trigger ausgelöst, sobald der Wert, der in "Triggerlevel" definiert ist, "von oben her" erreicht wird. • beide – Bei einer boolschen Triggervariable wird der Trigger ausgelöst, sobald sich der Wert ändert – Bei analogem Datentyp der Triggervariable wird der Trigger ausgelöst, sobald der Wert, der in Triggerlevel definiert ist, erreicht wird.
Posttrigger (Samples)	Anzahl der Datensätze pro Tracevariable, die nach Auslösung des Triggers zwischen gespeichert werden. Voreinstellung: 50. Wertebereich: 0 bis ($2^{32} - 1$)
Triggerlevel	Wert, bei dessen Erreichen der Trigger ausgelöst wird
Task	Task, in der die Datenerfassung erfolgt
Aufzeichnungsbedingung	Die Applikation prüft zur Laufzeit die Aufzeichnungsbedingung. Wenn sie erfüllt ist, werden die Tracedaten gepuffert. <i>Aufzeichnungsbedingung bei einer Datenerfassung mit Laufzeitsystemkomponente CmpTraceMgr:</i> • als Ausdruck, der nur erlaubte Operatoren und Operanden enthält Erlaubte Operatoren, die auch verschachtelt sein können: (Logisch) AND, NOT, OR, die Vergleichsoperatoren <, <=, >, >=, =, <> Erlaubte Operanden: Variablen, die für Trace gültig sind • als Variable Erlaubter Typ: BOOL, Bitzugriff, Eigenschaft. Bei TRUE oder 1 ist die Bedingung erfüllt. Der Inhalt eines Pointers ist nicht erlaubt. <i>Aufzeichnungsbedingung bei einer Datenerfassung mit IEC-Code</i> • als Ausdruck, der einen boolschen Wert zurück liefert
Kommentar	Kommentar, Beispielsweise über die Aufzeichnungsbedingung
Auflösung	Maßeinheit des Zeitstempels, der pro Datensatz erfasst wird • ms: Zeitstempel in ms • µs: Zeitstempel in µs bei einer Taskzykluszeit von 1 ms oder kleiner

Automatischer Neustart	 : Speichert dauerhaft (persistent) die Tracekonfiguration und den letzten Inhalt der LZS-Puffers auf dem Zielgerät
Erweitert	Öffnet den Dialog Erweiterte Traceeinstellungen

Siehe auch

- [Kap. "Dialog 'Erweiterte Traceeinstellungen'" auf Seite 782](#)

Teildialog 'Variableneinstellung'

Voraussetzung: In der Baumansicht Traceaufzeichnung oder in der Bauman-sicht Darstellung (Diagramme) ist eine Tracevariable selektiert.

Variable	Gültige Variable, deren Wert erfasst wird, mit vollständigem Instanzpfad <i>Gültig:</i> • IEC-Variable • Eigenschaft • Referenz • Inhalt eines Pointers • Arrayelement <i>Erlaubter Datentyp</i> • IEC basierter Typ außer <code>STRING</code>, <code>WSTRING</code> oder <code>ARRAY</code> • Enumeration, wenn der Basistyp kein <code>STRING</code>, <code>WSTRING</code> oder <code>ARRAY</code> ist Wenn das LZS die Komponente <code>CmpTraceMgr</code> verwendet, dann kann eine Eigen-schaft, die mit dem Attribut <code>'monitoring'</code> verknüpft ist, als Variable aufgezeichnet werden.
Parameter	Parameter, dessen Daten erfasst werden. Voraussetzung: Laufzeitsystem mit Komponente <code>CmpTraceMgr</code> Der Dialog Eingabehilfe, Registerkarte Kategorien, Eintrag Parameter listet alle gülti-gen Parameter des Systems auf.
	Ermöglicht Umschaltung zwischen Variable und Parameter
Farbe	Farbe der Variable im Tracediagramm
Linientyp	<i>Darstellung als Liniendiagramm</i> • Linie: Werte werden zu einer Linie verbunden • Stufe: Werte werden in Form einer Treppe verbunden • Keine: Werte werden nicht verbunden
Punkttyp	<i>Darstellung als Punktdiagramm</i> • Punkt: Wert wird als Punkt dargestellt • Kreuz: Wert wird als Kreuz dargestellt • Keiner: Wert wird nicht dargestellt
Untergrenze aktivieren	 : Warnung bei Unterschreitung der unteren Grenze
Untergrenze	Wenn der Wert der Tracevariablen die Grenze unterschreitet, wird die Variable in der Warnfarbe dargestellt.
Farbe	Warnfarbe bei Unterschreitung
Obergrenze aktivieren	 : Warnung bei Überschreitung der oberen Grenze

Obergrenze	Wenn der Wert der Tracevariablen die Obergrenze überschreitet, wird die Variable in der Warnfarbe dargestellt
Farbe	Warnfarbe bei Überschreitung

Teildialog 'Anzeigemodus'

Voraussetzung: In der Baumansicht Darstellung (Diagramme) ist eine Achse selektiert.

Anzeigemodus	Skalierung: • "Auto": Automatisch skalierte Zeitachse • "Feste Länge": Zeitachsenabschnitt mit konstanter Länge • "Fest": Zeitachsenabschnitt von Minimum bis Maximum
Minimum	Anfangswert des Zeitachsenabschnitts. Voraussetzung: Anzeigemodus ist Fest
Maximum	Endwert des Zeitachsenabschnitts. Voraussetzung: Anzeigemodus ist Fest.
Länge	Konstante Länge des Abschnitts, Anfangswert wird automatisch angepasst.
Raster	 : Diagramm mit Rasterlinie in X-Richtung. Wählen Sie die Farbe der Rasterlinie aus der Farbauswahlliste

Skalenstriche

Fester Abstand	 : Darstellung von Teilstrichen mit Abstand und Unterteilung
Abstand	Abstand der Teilstriche
Unterteilungen	Anzahl der Unterteilungen zwischen 2 Teilstrichen
Schriftart	Schriftart, verwendet für die Zeitachse
Link "Vorschau"	Zeigt die Vorschau des Diagramms an

Vorschau Diagramm

Voraussetzung: In der Baumansicht Darstellung (Diagramme) ist ein Diagramm selektiert.

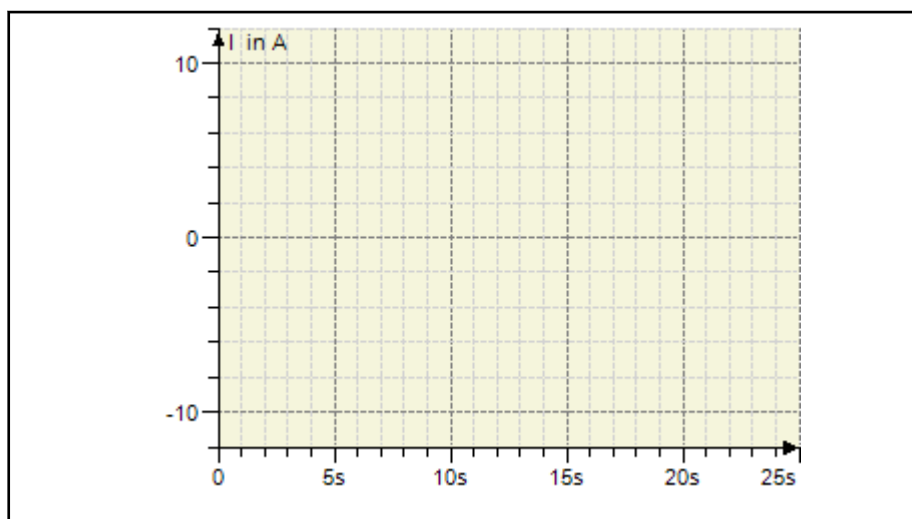


Abb. 18-76:

Hintergrundfarbe	Hintergrundfarbe des Diagramms
Hintergrundfarbe bei Selektion	Hintergrundfarbe des selektierten Diagramms

Link "Variable hinzufügen"	Fügt eine neue Tracevariable (in der Baumansicht "Traceaufzeichnung") hinzu.
Link "Variable entfernen"	Entfernt die selektierte Tracevariable (in der Baumansicht "Traceaufzeichnung").
Link "Diagramm hinzufügen"	Fügt ein neues Diagramm (in der Baumansicht "Darstellung") hinzu.
Link "Diagramm entfernen"	Entfernt das selektierte Diagramm (in der Baumansicht "Darstellung").
Link "Anzeigeeinstellungen zurücksetzen"	Setzt die Anzeigeeinstellungen entweder des selektierten Diagramms oder der selektierten Y-Achse auf die Standardwerte zurück.

Siehe auch

- [Kap. 13.3.2 "Tracekonfiguration erstellen" auf Seite 219](#)
- [Kap. 13.3 "Datenerfassung mit 'Trace'" auf Seite 216](#)
- [Kap. "Attribut 'monitoring'" auf Seite 419](#)

19 Bibliotheken

Bibliotheken dienen der Bereitstellung von Bausteinen und Funktionen zur Verwendung in IndraLogic-Applikationen. Neben den hier in der Online-Hilfe vorliegenden Beschreibungen sehen Sie bitte immer auch die in der Bibliothek enthaltene Dokumentation.

Zur Handhabung von Bibliotheken im IndraLogic-Projekt sehen Sie bitte: [Kap. 16 "Bibliotheken erzeugen und verwalten" auf Seite 235](#).

Wenn Sie selbst mit IndraLogic Bibliotheken erstellen, beachten Sie unbedingt die Richtlinien für Bibliotheksentwickler!: [Kap. 19.1 "Richtlinien für das Erstellen von Bibliotheken" auf Seite 789](#)

19.1 Richtlinien für das Erstellen von Bibliotheken

Bibliotheken müssen nach bestimmten Regeln erstellt sein um Kompatibilitätsprobleme zu vermeiden.

Die wichtigsten Punkte sind:

- Definieren Sie einen sinnvollen Bibliotheksnamen (obligatorisch)
- Verwenden Sie Vorlagen, um Konsistenz zu erreichen (optional)
- Verwenden Sie eine bekannte und soweit als möglich einheitliche Projektstruktur (optional)
- Lassen Sie einen eindeutigen globalen Bibliotheksnamensraum registrieren (obligatorisch)
- Füllen Sie die Bibliotheksinformation aus (obligatorisch)
- Referenzieren Sie andere Bibliotheken korrekt (obligatorisch)
- Implementieren Sie brauchbare externe und interne Schnittstellen (obligatorisch)
- Implementieren Sie eine benutzerfreundliche Fehlerverwaltung (obligatorisch)
- Wenden Sie die passenden Methoden (Schutz) für die Bereitstellung an (obligatorisch)
- Folgen Sie einer Namenskonvention, um sauberen Code zu erhalten (optional)
- Wenn Sie eine bestehende Bibliothek weiterentwickeln, achten Sie auf Schnittstellenkompatibilität zu vorherigen Versionen.

Siehe auch

- [Kap. 16 "Bibliotheken erzeugen und verwalten" auf Seite 235](#)

19.2 Bibliothek 'Recipe Management'

Wird ein Rezepturverwalter zu einer Applikation hinzugefügt, wird die Bibliothek `Recipe Management` dem Bibliotheksverwalter der Applikation übergeben.

In dieser Bibliothek ist der Funktionsbaustein `RecipeManCommands` enthalten. Der FB verwaltet die nachstehend beschriebenen Methoden zur Rezepturverwaltung.

Die Rückgabewerte der Methoden sind in der Globalen Variablenliste `ReturnValues` enthalten.



Beim Aufruf der Rezepturkommandos führt IndraLogic intern Dateizugriffe durch. Dies kann je nach Gerätetyp einige Millisekunden andauern.

Bitte berücksichtigen Sie, dass diese Aufrufe nicht aus einer Task durchgeführt werden, für die ein Watchdog oder eine echtzeitsensitive Task konfiguriert wurde.

Bitte beachten Sie, dass die Option **Änderungen an Rezepturen automatisch speichern** ebenfalls Dateizugriffe vornimmt, sobald Änderungen an Rezepturen durchgeführt werden (Rezepturverwalter, Registerblatt Allgemein).

Wenn die Speicherung explizit über die Applikation gesteuert werden soll, sollte diese Option ausgeschaltet sein.

Methode 'CreateRecipe'

Diese Methode erzeugt eine neue Rezeptur in der angegebenen Rezepturdefinition. Anschließend liest sie die aktuellen SPS-Werte in diese neue Rezeptur und speichert sie als Rezepturdatei mit Standardname.

Der Standardname ist <Rezeptur>.<Rezepturdefinition>.<Rezepturerweiterung>.

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
RecipeName	STRING		Name der Rezeptur

Tab. 19-1: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
CreateRecipe	DWORD		Rückgabewert, mögliche Werte - ERR_RECIPE_DEFINITION_NOT_FOUND - ERR_RECIPE_ALREADY_EXIST - ERR_RECIPE_NOMEMORY - ERR_NO_RECIPE_MANAGER_SET - ERR_OK

Tab. 19-2: Rückgabewert

Methode 'CreateRecipeNoSave'

Diese Methode erzeugt eine neue Rezeptur in der angegebenen Rezepturdefinition. Anschließend liest sie die Istwerte in die neue Rezeptur.

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
RecipeName	STRING		Name der Rezeptur

Tab. 19-3: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
CreateRecipeNoSave	DWORD		<i>Rückgabewert, mögliche Werte</i> • ERR_RECIPE_DEFINITION_NOT_FOUND • ERR_RECIPE_ALREADY_EXIST • ERR_RECIPE_NOMEMORY • ERR_NO_RECIPE_MANAGER_SET • ERR_OK

Tab. 19-4: Rückgabewert

Methode 'DeleteRecipe'

Diese Methode löscht eine Rezeptur aus der Rezepturdefinition.

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
RecipeName	STRING		Name der Rezeptur

Tab. 19-5: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
DeleteRecipe	DWORD		<i>Rückgabewert, mögliche Werte</i> • ERR_RECIPE_DEFINITION_NOT_FOUND • ERR_RECIPE_NOT_FOUND • ERR_NO_RECIPE_MANAGER_SET • ERR_OK

Tab. 19-6: Rückgabewert

Methode 'DeleteRecipeFile'

Diese Methode löscht die angegebene Rezepturdatei einer Rezeptur. Die Rezepturdatei muss im Standardnamen <Rezeptur>.<Rezepturdefinition>.<Rezepturerweiterung> gespeichert sein.

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
RecipeName	STRING		Name der Rezeptur

Tab. 19-7: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
DeleteRecipeFile	DWORD		<i>Rückgabewert, mögliche Werte</i> • ERR_RECIPE_DEFINITION_NOT_FOUND • ERR_RECIPE_NOT_FOUND • ERR_RECIPE_FILE_NOT_FOUND • ERR_NO_RECIPE_MANAGER_SET • ERR_OK

Tab. 19-8: Rückgabewert

Methode 'LoadAndWriteRecipe'

Diese Methode lädt eine Rezeptur aus der angegebenen Rezepturdatei. Die Rezepturdatei muss im Standardnamen <Rezeptur>.<Rezepturdefini-

tion>.<Rezepturerweiterung> gespeichert sein. Anschließend schreibt sie die Rezeptur in die SPS-Variablen.

 Einträge in der Rezepturdatei, die keine Wertezuweisung enthalten, werden nicht geladen und geschrieben (Kap. "Befehl 'Rezeptur laden und schreiben'" auf Seite 725).

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING STRING		Name der Rezepturdefinition
RecipeName	STRING		Name der Rezeptur

Tab. 19-9: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
LoadAndWriteRecipe	DWORD		<i>Rückgabewert, mögliche Werte</i> - ERR_RECIPE_DEFINITION_NOT_FOUND - ERR_RECIPE_NOT_FOUND - ERR_RECIPE_FILE_NOT_FOUND - ERR_RECIPE_MISMATCH - ERR_RECIPE_NOT_ALL_VARIABLES_WERE_LOADED - ERR_NO_RECIPE_MANAGER_SET - ERR_OK

Tab. 19-10: Rückgabewert

Siehe auch

- [Kap. "Befehl 'Rezeptur laden und schreiben'" auf Seite 725](#)

Methode
'LoadFromAndWriteRecipe'

Diese Methode lädt die angegebene Rezepturdatei in eine Rezeptur. Anschließend schreibt sie die Rezeptur in die SPS-Variablen.

 Einträge in der Rezepturdatei, die keine Wertezuweisung enthalten, werden nicht geladen und geschrieben (Kap. "Befehl 'Rezeptur laden und schreiben'" auf Seite 725).

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
RecipeName	STRING		Name der Rezeptur
FileName	STRING (255)		Name der Datei

Tab. 19-11: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
LoadFromAndWriteRecipe	DWORD		<i>Rückgabewert, mögliche Werte</i> • ERR_RECIPE_DEFINITION_NOT_FOUND • ERR_RECIPE_NOT_FOUND • ERR_RECIPE_FILE_NOT_FOUND • ERR_RECIPE_MISMATCH • ERR_RECIPE_NOT_ALL_VARIABLES_WERE_LOADED • ERR_NO_RECIPE_MANAGER_SET • ERR_OK

Tab. 19-12: Rückgabewert

Methode 'LoadRecipe'

Diese Methode lädt eine Rezeptur aus einer Rezepturdatei. Die Rezepturdatei muss im Standardnamen <Rezeptur>.<Rezepturdefinition>.<Rezepturerweiterung> gespeichert sein.



Einträge in der Rezepturdatei, die keine Wertezuweisung enthalten, werden nicht geladen und geschrieben (Kap. "Befehl 'Rezeptur laden und schreiben'" auf Seite 725).

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
RecipeName	STRING		Name der Rezeptur

Tab. 19-13: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
LoadRecipe	DWORD		<i>Rückgabewert, mögliche Werte</i> • ERR_RECIPE_DEFINITION_NOT_FOUND • ERR_RECIPE_NOT_FOUND • ERR_RECIPE_FILE_NOT_FOUND • ERR_RECIPE_MISMATCH • ERR_RECIPE_NOT_ALL_VARIABLES_WERE_LOADED • ERR_NO_RECIPE_MANAGER_SET • ERR_OK

Tab. 19-14: Rückgabewert

Methode 'ReadAndSaveRecipe'

Diese Methode liest die aktuellen SPS-Werte in die Rezeptur. Anschließend speichert sie die Rezeptur in eine Rezepturdatei mit Standardname. Der Standardname ist <Rezeptur>.<Rezepturdefinition>.<Rezepturerweiterung>. Der Inhalt einer eventuell existierenden Datei wird überschrieben.

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
RecipeName	STRING		Name der Rezeptur

Tab. 19-15: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
ReadAndSaveRecipe	DWORD		Rückgabewert, mögliche Werte - ERR_RECIPE_DEFINITION_NOT_FOUND - ERR_RECIPE_NOT_FOUND - ERR_RECIPE_SAVE_ERR - ERR_NO_RECIPE_MANAGER_SET - ERR_OK

Tab. 19-16: Rückgabewert

Methode 'ReadAndSaveRecipeAs'

Diese Methode liest die aktuellen SPS-Werte in die Rezeptur. Anschließend speichert sie die Rezeptur in eine angegebene Rezepturdatei. Der Inhalt einer eventuell existierenden Datei wird überschrieben.

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
RecipeName	STRING		Name der Rezeptur
FileName	STRING (255)		Name der Datei

Tab. 19-17: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
ReadAndSaveRecipeAs	DWORD		Rückgabewert, mögliche Werte - ERR_RECIPE_DEFINITION_NOT_FOUND - ERR_RECIPE_NOT_FOUND - ERR_RECIPE_SAVE_ERR - ERR_NO_RECIPE_MANAGER_SET - ERR_OK

Tab. 19-18: Rückgabewert

Methode 'SaveRecipe'

Diese Methode speichert die Rezeptur in eine Rezepturdatei mit Standardname.

Der Standardname ist

<Rezeptur>.<Rezepturdefinition>.<Rezepturerweiterung>.

Der Inhalt einer eventuell existierenden Datei wird überschrieben.

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
RecipeName	STRING		Name der Rezeptur

Tab. 19-19: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
SaveRecipe	DWORD		<i>Rückgabewert, mögliche Werte</i> • ERR_RECIPE_DEFINITION_NOT_FOUND • ERR_RECIPE_NOT_FOUND • ERR_RECIPE_SAVE_ERR • ERR_NO_RECIPE_MANAGER_SET • ERR_OK

Tab. 19-20: Rückgabewert

Methode 'ReadRecipe' Diese Methode liest die aktuellen SPS-Werte in die Rezeptur ein.

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
RecipeName	STRING		Name der Rezeptur

Tab. 19-21: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
ReadRecipe	DWORD		<i>Rückgabewert, mögliche Werte</i> • ERR_RECIPE_DEFINITION_NOT_FOUND • ERR_RECIPE_NOT_FOUND • ERR_NO_RECIPE_MANAGER_SET • ERR_OK

Tab. 19-22: Rückgabewert

Methode 'WriteRecipe' Diese Methode schreibt die Rezeptur in die SPS-Variablen.

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
RecipeName	STRING		Name der Rezeptur

Tab. 19-23: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
WriteRecipe	DWORD		<i>Rückgabewert, mögliche Werte</i> • ERR_RECIPE_DEFINITION_NOT_FOUND • ERR_RECIPE_NOT_FOUND • ERR_NO_RECIPE_MANAGER_SET • ERR_OK

Tab. 19-24: Rückgabewert

Methode 'ReloadRecipes' Diese Methode lädt die Liste der Rezepturen von einem Dateisystem. Dabei werden Rezepturen mit dem Standardnamen <Rezeptur>.<Rezepturdefinition>.<Rezepturerweiterung> berücksichtigt, die in dem im Rezepturverwalter definierten Pfad liegen.

Bibliotheken

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition

Tab. 19-25: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
ReloadRecipes	DWORD		Rückgabewert, mögliche Werte - ERR_RECIPE_DEFINITION_NOT_FOUND - ERR_NO_RECIPE_MANAGER_SET - ERR_OK

Tab. 19-26: Rückgabewert

Methode 'GetRecipeCount' Diese Methode gibt die Anzahl der Rezepturen einer Rezepturdefinition zurück.

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition

Tab. 19-27: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
GetRecipeCount	DWORD		Rückgabewert, mögliche Werte - ERR_RECIPE_DEFINITION_NOT_FOUND - ERR_NO_RECIPE_MANAGER_SET - ERR_OK

Tab. 19-28: Rückgabewert

Methode 'GetRecipeNames' Diese Methode gibt die Rezepturnamen einer Rezepturdefinition zurück.

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
pStrings	POINTER TO ARRAY[] OF STRING		Zeiger auf Zeichenfolgen, in denen die Rezepturnamen gespeichert werden.
iSize	INT		Größe des Zeichenfolgen-Arrays.
iStartIndex	INT		Der Startindex.

Tab. 19-29: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
GetRecipeNames	DWORD		Rückgabewert, mögliche Werte - ERR_RECIPE_DEFINITION_NOT_FOUND - ERR_NO_RECIPE_MANAGER_SET - ERR_OK

Tab. 19-30: Rückgabewert

Beispiel:

Es gibt zum Beispiel 50 Rezepturen. Wenn Sie eine Tabelle erzeugen wollen, die 10 Rezepturnamen gleichzeitig anzeigt, müssen Sie ein `STRING`-Array definieren:

```
strArr: ARRAY[0..9] OF STRING;
```

Korrespondierend zum `iStartIndex` erhalten Sie dann die Rezepturnamen für den spezifizierten Bereich.

```
iStartIndex := 0; // The names 0..9 are returned
iStartIndex := 20; // The names 20..29 are returned
```

In diesem Beispiel gilt:

```
iSize := 10;
```

Methode 'GetRecipeValues'

Diese Methode gibt die Werte einer Rezeptur zurück.

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
RecipeName			Name der Rezeptur
pStrings	POINTER TO ARRAY[] OF STRINGS		Zeiger auf Zeichenketten, in denen die Rezepturwerte gespeichert werden
iSize	INT		Größe des Zeichenfolgen-Arrays
iStartIndex	INT		Der Startindex
iStringLength	INT		Die Länge der Zeichenkette im Array

Tab. 19-31: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
GetRecipeValues	DWORD		Rückgabewert, mögliche Werte - ERR_RECIPE_DEFINITION_NOT_FOUND - ERR_RECIPE_NOT_FOUND - ERR_NO_RECIPE_MANAGER_SET - ERR_OK

Tab. 19-32: Rückgabewert

Beispiel:

Es gibt zum Beispiel 50 Rezepturen. Wenn Sie eine Tabelle erzeugen wollen, die 10 Rezepturwerte gleichzeitig anzeigt, müssen Sie dazu ein `STRING`-Array definieren:

```
strArr: ARRAY[0..9] OF STRING;
```

Korrespondierend zum `iStartIndex` erhalten Sie dann die Rezepturwerte für einen spezifizierten Bereich.

```
iStartIndex := 0; // The values 0..9 are returned
iStartIndex := 20; // The values 20..29 are returned
```

In diesem Beispiel gilt:

```
iStringLength := 80;
iSize := 10;
```

Methode
'GetRecipeVariableNames'

Diese Methode gibt die Namen der Rezepturvariablen einer Rezeptur wieder.

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
RecipeName			Name der Rezeptur
pStrings	POINTER TO ARRAY[] OF STRINGS		Zeiger auf STRING-Datentypen, in denen die Namen der Rezepturvariablen gespeichert werden.
iSize	INT		Größe des STRING-Arrays.
iStartIndex	INT		Der Startindex.

Tab. 19-33: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
GetRecipeVariableNames	DWORD		<i>Rückgabewert, mögliche Werte</i> - ERR_RECIPE_DEFINITION_NOT_FOUND - ERR_RECIPE_NOT_FOUND - ERR_NO_RECIPE_MANAGER_SET - ERR_OK

Tab. 19-34: Rückgabewert

Beispiel:

Es gibt zum Beispiel 50 Rezepturen. Wenn Sie eine Tabelle erzeugen wollen, die 10 Variablennamen einer Rezeptur gleichzeitig anzeigt, müssen Sie ein STRING-Array definieren:

```
strArr: ARRAY[0..9] OF STRING;
```

Korrespondierend zum iStartIndex erhalten Sie dann die Variablennamen der Rezeptur für den spezifizierten Bereich.

```
iStartIndex := 0; // The variable names 0..9 are returned
iStartIndex := 20; // The variable names 20..29 are returned
```

In diesem Beispiel gilt:

```
iSize := 10;
```

Methode 'SetRecipeValues'

Diese Methode setzt die Rezepturwerte einer Rezeptur.

Name	Datentyp	Initialwert	Beschreibung
RecipeDefinitionName	STRING		Name der Rezepturdefinition
RecipeName	STRING		Name der Rezeptur
pStrings	POINTER TO ARRAY[] OF STRINGS		Zeiger auf STRING-Datentypen, in denen die Rezepturwerte gespeichert werden.
iSize	INT		Größe des STRING-Arrays.
iStartIndex	INT		Der Startindex.

Tab. 19-35: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
SetRecipeValues	DWORD		<i>Rückgabewert, mögliche Werte</i> - ERR_RECIPE_DEFINITION_NOT_FOUND - ERR_RECIPE_NOT_FOUND - ERR_NO_RECIPE_MANAGER_SET - ERR_OK

Tab. 19-36: Rückgabewert

Beispiel:

Es gibt zum Beispiel 50 Rezepturen. Wenn Sie eine Tabelle erzeugen wollen, die 10 Rezepturwerte gleichzeitig setzt, müssen Sie ein STRING-Array definieren:

```
strArr: ARRAY[0..9] OF STRING;
```

Korrespondierend zum iStartIndex können Sie dann die Rezepturwerte für einen spezifizierten Bereich festlegen.

```
iStartIndex := 0; // The values 0..9 are set
iStartIndex := 20; // The values 20..29 are set
```

In diesem Beispiel gilt:

```
iStringLength := 80;
iSize := 10;
```

Methode 'GetLastError'

Diese Methode gibt den letzten Fehler der vorherigen Operation zurück.

Name	Datentyp	Initialwert	Beschreibung
GetLastError	DWORD		<i>Rückgabewert, mögliche Werte</i> - ERR_NO_RECIPE_MANAGER_SET - ERR_OK

Tab. 19-37: Rückgabewert

Methode 'ResetLastError'

Diese Methode setzt den letzten Fehler zurück.

Name	Datentyp	Initialwert	Beschreibung
ResetLastError	DWORD		<i>Rückgabewert, mögliche Werte</i> - ERR_NO_RECIPE_MANAGER_SET - ERR_OK

Tab. 19-38: Rückgabewert

Methode 'SetStoragePath'

Diese Methode dient dem Einstellen des Speicherpfads für die Rezepturdatei. Sie überschreibt die Pfadangabe im Dialog **Speicherung** des Rezepturverwalters.

Name	Datentyp	Initialwert	Beschreibung
stPath	STRING		<i>Dateipfad, Beispiel:</i> D:/recipefiles/

Tab. 19-39: Eingänge (VAR_INPUT)

Name	Datentyp	Initialwert	Beschreibung
SetStoragePath	BOOL		Rückgabewert, TRUE (Pfad wurde gesetzt), FALSE

Tab. 19-40: Rückgabewert

Rückgabewerte

Die Rückgabewerte der oben beschriebenen Methoden sind in der Globalen Variablenliste `ReturnValues` enthalten.

Es handelt sich um InOut-Konstanten vom Datentyp `UDINT`.

Name	Initialwert	Kommentar
ERR_OK	16#0	Die Operation wurde erfolgreich durchgeführt.
ERR_FAILED	16#1	Die Operation schlug fehl.
ERR_PARAMETER	16#2	Falscher Parameter
ERR_NOTINITIALIZED	16#3	Das DataServer-Objekt ist nicht initialisiert. Der DataServer wird benötigt, wenn die Rezepturverwaltung in Kombination mit VisuDeviceHMI verwendet wird.
ERR_NOTIMPLEMENTED	16#C	Der DataServer implementiert nicht die Schnittstelle <code>IDataServer4</code> , die benötigt wird, wenn die Rezepturverwaltung in Kombination mit VisuDeviceHMI verwendet wird..
ERR_NO_OBJECT	16#10	Nicht alle Variablen einer Rezepturdefinition können über den DataServer geschrieben werden. Nur die gültigen Variablen werden geschrieben.
ERR_NOMEMORY	16#11	Der DataServer hat nicht genügend Speicher.
ERR_RECIPE_FILE_NOT_FOUND	16#4000	Die Rezepturdatei wurde nicht gefunden.

Name	Initialwert	Kommentar
ERR_RECIPE_MISMATCH	16#4001	Der Inhalt der Rezepturdatei passt nicht zur aktuellen Rezeptur. Dieser Fehler wird nur ausgegeben, wenn der Speicherungstyp "textuell" ist (siehe Dialog Rezepturverwalter, Registerkarte Speicherung, Speicherungstyp) und wenn ein Variablenname in der Datei nicht mit dem Variablenname in der Rezepturdefinition übereinstimmt. Die Rezepturdatei wird nicht geladen, wenn dieser Fehler auftritt. Mögliche Ursachen: Eine Variable wurde aus der Rezepturdefinition im Projekt entfernt.
ERR_RECIPE_SAVE_ERR	16#4002	Der Speichervorgang schlug fehl. Mögliche Ursachen: Die Datei konnte nicht angelegt oder geöffnet werden, weil die Platte voll ist. Der konfigurierte Dateipfad existiert nicht (siehe Dialog Rezepturverwalter, Registerkarte Speicherung, Dateipfad). Die konfigurierte Dateierweiterung ist vom Laufzeitsystem nicht zugelassen (siehe Dialog Rezepturverwalter, Registerkarte Speicherung, Dateierweiterung).
ERR_RECIPE_NOT_FOUND	16#4003	Die Rezeptur existiert nicht.
ERR_RECIPE_DEFINITION_NOT_FOUND	16#4004	Die Rezepturdefinition existiert nicht.
ERR_RECIPE_ALREADY_EXIST	16#4005	Die Rezeptur existiert bereits in der Rezepturdefinition. Legen Sie eine neue Rezeptur unter einem anderen Namen an.
ERR_NO_RECIPE_MANAGER_SET	16#4006	Der globale Rezepturverwalter ist nicht angelegt. Mögliche Ursachen: Die Option Rezepturverwaltung in der SPS im Dialog Rezepturverwalter, Registerkarte Allgemein der aktuellen Applikation ist nicht gesetzt.
ERR_RECIPE_NOT_ALL_VARIABLES_WERE_LOADED	16#4007	Die Rezepturdefinition enthält mehr Variablen als die Rezepturdatei. In diesem Fall werden die Variablenwerte der Rezepturdatei auf jeden Fall geschrieben. Dies ist nur eine Information, kein tatsächlicher Fehler.
ERR_RECIPE_NOMEMORY	16#4008	Die Rezepturdefinition hat keinen freien Speicher, um eine neue Rezeptur anzulegen. Mögliche Ursachen: Die Option Rezepturänderungen automatisch in Rezepturdateien speichern im Dialog Rezepturverwalter, Registerkarte Allgemein der aktuellen Applikation ist nicht gesetzt. In diesem Fall sind nur 50 Rezepturen pro Rezepturdefinition möglich. Wenn Die Option Rezepturänderungen automatisch in Rezepturdateien speichern gesetzt ist, kann der Fehler nicht auftreten. Wenn die Platte voll ist, wird der Fehler ERR_RECIPE_SAVE_ERR ausgegeben.

Name	Initialwert	Kommentar
ERR_RECIPE_MANAGER_LOCKED_DURING_ONLINE_CHANGE	16#4009	Der Rezepturverwalter war während des Online-Change blockiert. Mögliche Ursachen: Einige der RecipeMan-Befehle, die während eines Online-Change ausgeführt werden sollten, wurden nicht ausgeführt.
ERR_SOURCE_EXHAUSTED	16#40A0	Verwendet für UTF8 Helper
ERR_TARGET_EXHAUSTED	16#40A1	Verwendet für UTF8 Helper
ERR_SOURCE_ILLEGAL	16#40A2	Verwendet für UTF8 Helper

20 IndraLogic 2G-Visualisierung

20.1 Visualisierung im Überblick

IndraWorks bietet in Abhängigkeit vom Anwendungsbereich zwei Visualisierungswerkzeuge, die mit unterschiedlicher Leistungsfähigkeit implementiert wurden:

- WinStudio, dieses Werkzeug sollten Sie für Anwendervisualisierungen, die auf Visualisierungsgeräten (BTV / Vxx) laufen, verwenden. Die Übergabe der zu visualisierenden Daten erfolgt dabei unter Nutzung des Objektes 'Symbolkonfiguration' ([Kap. 10.2 "Symbolkonfiguration" auf Seite 167](#)).
- Visualisierung onboard, dieses nachstehend beschriebene Werkzeug sollten Sie zur Unterstützung der Inbetriebnahme Ihrer eigenen Applikationen verwenden.

Werkzeuge und Editoren

Eine IndraLogic 2G-Visualisierung basiert auf einem modularen Visualisierungskonzept und stellt die folgenden Werkzeuge/Editoren zur Verfügung.

- [Kap. 20.3 "Editor" auf Seite 825](#): Hier können Visualisierungen/Masken/Panels in einem IEC 61131-3-Tool mit wenigen Klicks hergestellt werden.
- [Kap. 20.4 "Visualisierungselemente" auf Seite 843](#): Eine Vielzahl fertiger Elemente steht einsatzfähig zur Verfügung.
- [Kap. 20.9 "Visualisierungsprofil" auf Seite 1080](#): Anstelle des vorgeschlagenen Standardprofils kann ein Visualisierungsprofil aus einer Visualisierungsbibliotheken ausgewählt werden. Jedes Visualisierungsprojekt, das wenigstens eine Visualisierung enthält, basiert auf einem solchen Profil.
- [Kap. 20.7 "Visualisierungsbibliotheken" auf Seite 1078](#): Hier können Elemente zusammengefasst werden, um sie für den Einsatz in anderen Projekten zur Verfügung zu stellen.
- Mittels Visualisierungsstil wird die Elementdarstellung definiert, um ein bestimmtes Look-and-Feel der Elemente zu erhalten.
- [Kap. 20.5 "Visualisierungsmanager mit Clients" auf Seite 1058](#): Hier werden die verschiedenen Darstellungsvarianten verwaltet, die flexibel und mit geringem Aufwand für die unterschiedlichsten kundenspezifischen Anwendungen eingesetzt werden.

Visualisierungen mit IndraLogic 2G ermöglichen Ihnen:

- Visualisierungen durch Referenzen zu verschachteln und/oder zwischen verschiedenen Visualisierungen umzuschalten: [Kap. "Element 'Frame'" auf Seite 917](#)
- Schnittstellen für Parametertransfer zu verwenden, um komplexe Visualisierungen zu instanzieren: [Kap. "Grundlegendes zum Schnittstelleneditor" auf Seite 831](#)
- Unterstützung der Mehrsprachigkeit durch Textlisteneditoren: [Kap. "Text und Sprache verwenden" auf Seite 845](#)

Allgemeiner Mechanismus

[Kap. "Diagnosevisualisierung" auf Seite 1061](#): Diese Darstellungsvariante läuft nur im Programmiersystem (entspricht der reinen Online-Visualisierung in IndraLogic 1x).

Daten externer (entfernter) Datenserver können in den Visualisierungen verwendet werden. Zu diesem Zweck müssen diese Daten in die lokalen Applikation übernommen und verarbeitet werden.

Siehe auch

- [Kap. 10.4.1 "Datenquellenverwalter Grundlagen" auf Seite 174](#)

Erzeugen einer Visualisierung

Ein Visualisierungsobjekt kann im Projektextplorer unterhalb einer Applikation oder im Ordner "Allgemeiner Baustein" durch Verwendung des Befehls **Hinzufügen...** (auch im Kontextmenü) eingefügt werden.

Das Fenster eines neuen Visualisierungsobjekts zeigt zunächst einen leeren Bereich an.

Wird das erste Visualisierungsobjekt im Projekt hinzugefügt, werden die zugehörigen Visualisierungsbibliotheken automatisch vom Bibliotheksverwalter entsprechend dem aktuell aktiven Visualisierungsprofil hinzugefügt.

Eine Visualisierungsbibliothek ist immer als ein spezieller Typ einer **"Platzhalterbibliothek"** mit der Auswirkung eingerichtet, dass die exakte Version der verwendeten Bibliothek nicht aufgelöst wird, bis die Bibliothek im Projekt eingefügt wird. Nur dann definiert das aktuell aktive Profil, welche Version tatsächlich benötigt wird.

Bitte beachten Sie, dass dieser Bibliothekstyp sich von den gerätespezifischen Platzhalterbibliotheken unterscheidet. Dort werden die Platzhalter in der Gerätebeschreibung aufgelöst.

Basisbibliotheken, die standardmäßig eingefügt werden, wenn ein Visualisierungsobjekt eingefügt wird, sind `VisuElems`, `VisuElemMeter`, `VisuElemWinControls`, `VisuElemTrace`, `VisuInputs`.

Durch den Anwender können weitere Bibliotheken eingefügt werden.



Im Allgemeinen muss **keine** Visualisierungsbibliothek händisch eingefügt oder explizit im Projekt verwendet werden.

Eine Visualisierung kann als Objekt im Ordner "Allgemeiner Baustein" angelegt werden, oder direkt einer Applikation.

Sobald die erste Visualisierung unter einer Applikation hinzugefügt wurde, wird auch ein Visualisierungsmanager hinzugefügt ([Kap. 20.5 "Visualisierungsmanager mit Clients" auf Seite 1058](#)).

Eine Startvisualisierung, also eine Visualisierung, die nach einem Login auf der SPS als erstes geöffnet wird, muss im Projektnavigator unterhalb des zugehörigen Applikationsobjekts eingefügt werden.

Jede Visualisierung hat Eigenschaften entsprechend ihrer bestimmungsgemäßen Verwendung:

Visualisierung, Nummernfeld/Tastatur, Dialog oder Anzeigengröße ([Kap. 20.6 "Dialog 'Eigenschaften- 'Visualisierungsobjekt'" auf Seite 1063](#)).

Beachten Sie in diesem Zusammenhang, dass eine Visualisierung explizit für den Einsatz als Benutzereingabedialog in anderen Visualisierungen erstellt und konfiguriert werden kann ([Kap. "Eingabekonfiguration" auf Seite 849](#)).

Implizit sind auch standardmäßig ein Nummernfeld bzw. eine Tastatur verfügbar. Die Verwendung von Tastatur und Dialog kann in den Eigenschaften **Eingabekonfiguration** eines Visualisierungselements definiert werden.

Exceptionhandling zur Laufzeit

Wenn in einer Visualisierung zur Laufzeit ein Fehler oder eine Ausnahme auftritt, wird die Ausführung der Visualisierung gestoppt, ohne die Ausführung der Applikation zu stoppen.

Ein Fehlerbildschirm erscheint, in dem Sie darüber informiert werden. Der Fehlerbildschirm ermöglicht Ihnen üblicherweise die Visualisierung wieder zu starten.

Visualisierung, Dialog, virtuelles Nummernfeld oder virtuelle Tastatur?

Diese Exceptionhandling erfolgt ab Visualisierungsprofil 3.5.7.0, Compiler-version 3.5.7.0 und einem Laufzeitsystem ab Version 3.5.7.0.

Wählen Sie den Befehl **Ausführung an behandelten Ausnahmefehlern stoppen**, um die Ursache für das Auftreten von Ausnahmen und die Fehlerstelle zu untersuchen.

Siehe auch

- [Kap. "Befehl 'Ausführung an behandelten Exceptions anhalten'" auf Seite 743](#)

Online-Betrieb

Die Visualisierung arbeitet mit einem integrierten Laufzeitsystem. Deshalb erscheinen während der Arbeit im Editor gegebenenfalls Start- und Download-Meldungen.

20.1.1 Anzeigen der Vorschau auf eine Bibliotheksvisualisierung

Eine Bibliotheksvisualisierung ist ein Baustein in einer Bibliothek, der eine Visualisierung referenziert.

IndraLogic stellt im Bibliotheksverwalter eine Beschreibung zu jedem Baustein dar. Wenn es sich bei einem Baustein um eine Bibliotheksvisualisierung handelt, wird in der Registerkarte **Graphisch** eine Vorschau der Visualisierung isotropisch skaliert angezeigt.

Vorschau auf eine Bibliotheksvisualisierung anzeigen

1. Öffnen Sie den Bibliotheksverwalter.
2. Selektieren Sie eine Bibliothek, die eine Bibliotheksvisualisierung enthält, zum Beispiel *VisuDialogs*

Der Inhalt der Bibliothek wird unterhalb in einer Baumansicht dargestellt.

3. Selektieren Sie in der Baumansicht eine Bibliotheksvisualisierung, zum Beispiel *Keypad*.

Rechts von der Baumansicht der Bibliothek wird der selektierte Baustein beschrieben.

4. Klicken Sie auf **Graphisch**.

Die Vorschau der Bibliotheksvisualisierung wird angezeigt.

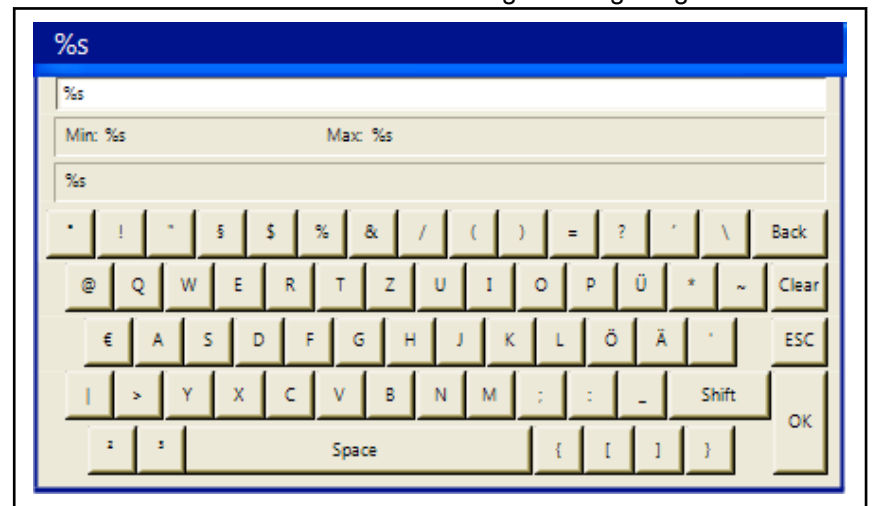


Abb. 20-1:

20.1.2 Verfügbare Visualisierungselemente anzeigen und kennzeichnen

Die Ansicht **Werkzeuge** stellt Ihnen alle Visualisierungselemente zur Verfügung, die im Projekt vorhanden sind. Das umfasst die Elemente aus dem Visualisierungsprofil, die Frame-Elemente aus Bibliotheken und die Bildelemente aus Symbolbibliotheken. Damit Sie ein Element identifizieren und beurteilen können, stellt IndraLogic es in der Ansicht **Werkzeuge** als Vorschau-Bild dar. Sie können aus den dargestellten Elementen eines auswählen und in den Visualisierungseditor ziehen. Außerdem haben Sie die Möglichkeit die Elemente zu kennzeichnen und über diese Kennzeichen zu sortieren.

Verfügbare Elemente

Die Elemente stammen aus

- dem aktuellen Visualisierungsprofil
- aus Bildern von Bildersammlungen, die im Projekt oder in einer Bibliothek enthalten sind
- aus Visualisierungen, die im Projekt oder in einer Bibliothek enthalten sind

Ein Bild, das in der tabellarischen Anordnung zur Verfügung gestellt wird, ist im Projekt in einer Bildersammlung als Referenz auf eine Bilddatei gespeichert. Allerdings ignoriert IndraLogic eine Bildersammlung, wenn in ihren Eigenschaften die Option **Intern** aktiviert ist. Ein Bild kann auch in einer Symbolbibliothek, auf die Sie zugreifen können, gespeichert sein.

Eine bestehende Visualisierung, die als Element erscheint, ist im Projekt oder in einer Bibliothek gespeichert, auf die Sie zugreifen können. Eine Bibliothek, auf die Sie nicht zugreifen können, wird von IndraLogic ignoriert und erscheint nicht in der Anzeige. Wenn Sie eine bestehende Visualisierung in die Visualisierung ziehen, hat sie den Elementtyp **Frame**.

Siehe auch

- [Kap. 9.9 "Bildersammlungen verwenden" auf Seite 126](#)
- [Kap. "Dialog 'Eigenschaften- 'Bildersammlung'" auf Seite 757](#)
- [Kap. "Element 'Bild'" auf Seite 906](#)
- [Kap. "Element 'Frame'" auf Seite 917](#)

Kennzeichen

Jedes Element ist einem oder mehreren Kennzeichen zugeordnet. Mit Hilfe der Kennzeichen können Sie die Anzahl der sichtbaren Elemente steuern. Dafür stellt IndraLogic jedes Kennzeichen als Schaltfläche in der Ansicht **Werkzeuge** zur Verfügung, die grün oder grau ist, je nachdem ob das Kennzeichen "selektiert" oder "nicht selektiert" ist. Nur wenn ein Kennzeichen selektiert und grün ist, zeigt IndraLogic in der tabellarischen Ansicht genau die Elemente an, die ihm zugeordnet sind. Sie können den Zustand des Kennzeichens umschalten, indem Sie auf die Schaltfläche klicken.

IndraLogic vergibt bestimmte Kennzeichen automatisch. Aber Sie können ein Kennzeichen auch selbst erstellen und ihm ein oder mehrere Elemente zuweisen.

Mit den Befehlen  für Einzelselektion und  für Mehrfachselektion können Sie den Selektionsmechanismus wählen. Wenn Sie auf die Schaltfläche eines Kennzeichens klicken und zusätzlich die <Umschalttaste> gedrückt halten, ist der Selektionsmechanismus für den einen Klick umgeschaltet.

Von IndraLogic bereitgestellte Kennzeichen

Jedes Element aus dem Visualisierungsprofil hat ein Kennzeichen, das kategorisiert.

Die kategorisierenden Kennzeichen sind

- **Basis**

- Allgemeine Steuerelemente
- Messgeräte
- Lampen/Schalter/Bilder
- Spezielle Steuerelemente

Ein Bild aus einer Bildersammlung hat als Kennzeichen den Namen der Bildersammlung.

Eine Visualisierung aus einer Bibliothek hat als Kennzeichen den Namen dieser Bibliothek.

Element in die Visualisierung ziehen

*Voraussetzung: Ein Projekt, das ein Objekt des Typs **Visualisierung** enthält, ist geöffnet.*

1. Doppelklicken Sie auf die Visualisierung.
Der Editor der Visualisierung und die Ansicht **Werkzeuge** öffnen sich. Die Ansicht **Werkzeuge** bietet Elemente an.
2. Ziehen Sie ein Element in die Visualisierung.
Das Element erscheint in der Visualisierung und verfügt über Eigenschaften, die Sie weiter konfigurieren können.

Kennzeichen erstellen

*Voraussetzung: Ein Projekt, das ein Objekt des Typs **Visualisierung** enthält, ist geöffnet.*

1. Doppelklicken Sie auf die Visualisierung.
Der Editor der Visualisierung und die Ansicht **Werkzeuge** öffnen sich.
2. Klicken Sie in der Ansicht **Werkzeuge** auf .
Der Dialog **Kennzeichen und Einträge konfigurieren** erscheint.
3. Klicken Sie im Dialog auf .
Der Dialog **Kennzeichen hinzufügen** erscheint.
4. Geben Sie im Eingabefeld **Name** einen Namen ein, zum Beispiel `tagA`, und beenden Sie den Dialog mit **OK**.
Im Dialog **Kennzeichen und Einträge konfigurieren** ist das neue benutzerdefinierte Kennzeichen `tagA` in der Baumansicht unten eingefügt. Es ist mit dem Symbol  versehen.
5. Klicken Sie für das neue Kennzeichen auf die Option **Aktivieren** und beenden Sie den Dialog mit **OK**.
IndraLogic fügt in der Ansicht **Werkzeuge** eine Schaltfläche `tagA` hinzu.

Kennzeichen einem Element zuordnen

*Voraussetzung: Ein Projekt, das ein Objekt des Typs **Visualisierung** enthält, ist geöffnet. In der Ansicht **Werkzeuge** ist das benutzerdefinierte Kennzeichen `tagA` erstellt.*

1. Doppelklicken Sie auf die Visualisierung.
Der Editor der Visualisierung und die Ansicht **Werkzeuge** öffnen sich.
2. Klicken Sie in der Ansicht **Werkzeuge** auf ein Element und aktivieren Sie dessen Kontextmenü.

Das Kontextmenü des Elements enthält Befehle, um dem Element ein Kennzeichen zuzuordnen.

3. Wählen Sie den Befehl **Eintrag zu Kennzeichen 'tagA' hinzufügen**.
4. Klicken Sie auf die Schaltfläche **tagA**.
Alle Elemente, die diesem Kennzeichen zugeordnet sind, erscheinen.
Auch das gerade zugeordnete Element wird angezeigt.

Kennzeichen entfernen *Voraussetzung: Ein Projekt, das ein Objekt des Typs **Visualisierung** enthält, ist geöffnet.*

1. Doppelklicken Sie auf die Visualisierung.
Der Editor der Visualisierung und die Ansicht **Werkzeuge** öffnen sich.
2. Klicken Sie in der Ansicht **Werkzeuge** auf .
Der Dialog **Kennzeichen und Einträge konfigurieren** erscheint.
3. Selektieren Sie in der Baumansicht einen benutzerdefinierten Eintrag.
4. Klicken Sie auf  oder geben Sie <Entf> ein.
Das Kennzeichen ist aus der Baumansicht entfernt.
5. Beenden Sie den Dialog mit **OK**
IndraLogic entfernt die Schaltfläche in der Ansicht **Werkzeuge**.

Kennzeichen ausblenden *Voraussetzung: Ein Projekt, das ein Objekt des Typs **Visualisierung** enthält, ist geöffnet.*

1. Doppelklicken Sie auf die Visualisierung.
Der Editor der Visualisierung und die Ansicht **Werkzeuge** öffnen sich.
2. Fokussieren Sie im Visualisierungseditor eine der Schaltflächen und aktivieren Sie deren **Kontextmenü**
Der Befehl **Kennzeichen ausblenden** erscheint.
3. Klicken Sie auf den Befehl **Kennzeichen ausblenden**.
Das Kennzeichen ist entfernt.

Kennzeichen einblenden *Voraussetzung: Ein Projekt, das ein Objekt des Typs **Visualisierung** enthält, ist geöffnet.*

1. Doppelklicken Sie auf die Visualisierung.
Der Editor der Visualisierung und die Ansicht **Werkzeuge** öffnen sich.
2. Klicken Sie in der Ansicht **Werkzeuge** auf .
Der Dialog **Kennzeichen und Einträge konfigurieren** erscheint.
3. Klicken Sie unter **Aktiv** auf ☐.
Ergebnis: 
4. Beenden Sie den Dialog mit **OK**

Das zugehörige Kennzeichen ist als Schaltfläche sichtbar.

**Kennzeichen in Einzelselektion
selektieren**

*Voraussetzung: Ein Projekt, das ein Objekt des Typs **Visualisierung** enthält, ist geöffnet.*

1. Doppelklicken Sie auf die Visualisierung.
Der Editor der Visualisierung und die Ansicht **Werkzeuge** öffnen sich.
2. Klicken Sie in der Ansicht **Werkzeuge** auf .
Die Schaltflächen sind einzeln selektierbar.
3. Klicken Sie auf eine graue Schaltfläche.
Die Schaltfläche wird grün, alle anderen Schaltflächen sind grau.
In der tabellarischen Anordnung erscheinen die der grünen Schaltfläche zugeordneten Einträge.
4. Drücken Sie die <Umschalttaste> während Sie auf eine graue Schaltfläche klicken.
Die Schaltfläche ist zusätzlich zu der bereits selektierten Schaltfläche grün. In der tabellarischen Anordnung erscheinen die zugeordneten Elemente der beiden grünen Schaltflächen. Die Schaltflächen sind trotzdem weiterhin in Einzelselektion.

**Kennzeichen in Mehrfachselektion
selektieren**

*Voraussetzung: Ein Projekt, das ein Objekt des Typs **Visualisierung** enthält, ist geöffnet.*

1. Doppelklicken Sie auf die Visualisierung.
Der Editor der Visualisierung und die Ansicht **Werkzeuge** öffnen sich.
2. Klicken Sie in der Ansicht **Werkzeuge** auf .
Das Schaltflächenfeld ist in Mehrfachselektion.
3. Klicken Sie auf eine graue Schaltfläche.
Die Schaltfläche ist grün, alle bereits selektierten Schaltflächen bleiben grün. In der tabellarischen Anordnung erscheinen die zugeordneten Einträge aller grünen Schaltflächen.
4. Drücken Sie die <Umschalttaste> während Sie auf eine graue Schaltfläche klicken.
Die Schaltfläche ist grün, alle anderen Schaltflächen sind grau. In der tabellarischen Anordnung erscheinen die zugeordneten Elemente der einen grünen Schaltfläche. Das Schaltflächenfeld ist weiterhin in Mehrfachselektion.

Siehe auch

- [Kap. 20.3.1 "Ansicht 'Werkzeuge'" auf Seite 826](#)
- [Kap. "Befehl 'Werkzeuge'" auf Seite 628](#)

20.1.3 Symbolbibliothek in der Visualisierung verwenden

Eine Symbolbibliothek stellt Ihnen Bilder aus Bildersammlungen zur Verfügung, die Sie dann im Projekt und in der Applikation referenzieren können. Vor allem wenn Sie eine Visualisierung öffnen, stellt IndraLogic den Inhalt ei-

ner installierten und ausgewählten Symbolbibliothek in der Ansicht **Werkzeuge** zur Verfügung und Sie haben über Vorschaubilder komfortablen Zugriff auf die Bilder.

Eine Symbolbibliothek enthält ein oder mehr Objekte des Typs **Bildersammlung** und kann Objekte des Typs **Textliste** enthalten. Eine Textliste ist dabei genau einer Bildersammlung zugeordnet und enthält die lokalisierten Texte der IDs der Bilder. Eine Symbolbibliothek hat die Eigenschaft `VisuSymbolLibrary = True`, die IndraLogic ausliest, um eine Symbolbibliothek als solche zu identifizieren.

Eine Symbolbibliothek kann als Package zur Verfügung gestellt werden.

Symbolbibliothek im Projekt erstmalig auswählen

Voraussetzung: Ein Projekt, das bisher keine Visualisierung und keinen Visualisierungsmanager enthält, ist geöffnet.

1. Selektieren Sie im Projektexplorer die Applikation und wählen Sie im Kontextmenü **Hinzufügen**.

2. Wählen Sie **Visualisierung**.

Der Dialog **Visualisierung hinzufügen** erscheint und listet die installierten Symbolbibliotheken auf.

3. Aktivieren Sie die Option **Aktiv**, zum Beispiel für `VisuSymbols`, wenn Sie eine der installierten Symbolbibliotheken auswählen wollen. Beenden Sie den Dialog mit **Hinzufügen**.

Die Symbolbibliotheken sind im Projekt ausgewählt.

Sie befinden sich im Ordner "Allgemeiner Baustein".

Die Ansicht **Werkzeuge** enthält unter **Symbols** die Symbole der Bibliothek.

4. Wählen Sie **Ordner "Allgemeiner Baustein" ► Projekteinstellung**, Kategorie **Visualisierung**, Registerkarte **Symbolbibliotheken**.

IndraLogic listet die installierten Symbolbibliotheken auf und informiert, ob sie für das Projekt aktiviert sind.

Symbolbibliothek im Projekt deaktivieren

Voraussetzung: Ein Projekt, dem eine Symbolbibliothek zugewiesen ist, ist geöffnet.

1. Wählen Sie **Ordner "Allgemeiner Baustein" ► Projekteinstellung**, Kategorie **Visualisierung**, Registerkarte **Symbolbibliotheken**.

IndraLogic listet die installierten Symbolbibliotheken auf und informiert, ob sie für das Projekt ausgewählt sind. Die ausgewählten Bibliotheken haben eine aktivierte Option **Aktiv**.

2. Aktivieren Sie die Option bei der Symbolbibliothek, die Sie auswählen wollen.

3. Deaktivieren Sie die Option bei der Symbolbibliothek, die Sie deaktivieren wollen.

4. Beenden Sie den Dialog mit **OK**.

Wenn Sie eine Visualisierung öffnen, stellt IndraLogic den Inhalt der ausgewählten Symbolbibliotheken in der Ansicht **Werkzeuge** zur Verfügung.

Symbolbibliothek im Projekt aktualisieren

Voraussetzung: Ein Projekt, dem eine Symbolbibliothek zugewiesen ist, ist geöffnet.

1. Wählen Sie **Ordner "Allgemeiner Baustein"** ► **Projekteinstellung**, Kategorie **Visualisierung**, Registerkarte **Symbolbibliotheken**.

IndraLogic zeigt die Symbolbibliotheken an, die veraltet sind.

2. Wählen Sie eine neuere Version aus und beenden Sie den Dialog mit **OK**.

IndraLogic aktualisiert die Symbolbibliothek.

Symbolbibliothek erstellen

1. Wählen Sie **Datei** ► **Neues Projekt**.

Der Dialog **Neues Projekt** erscheint.

2. Wählen Sie unter **Kategorien** den Eintrag **Bibliotheken**.

3. Wählen Sie **Leere Bibliothek**.

4. Gebe Sie einen Namen an, zum Beispiel `SymbolLib_A`.

5. Erstellen Sie eine Bildersammlung, zum Beispiel `ImagePool_A`.

6. Erstellen Sie eine Textliste, die der Bildersammlung zugeordnet werden kann, zum Beispiel `TextList_A`.

7. Selektieren Sie im Projektextplorer das Objekt der Bildersammlung.

8. Wählen Sie im Kontextmenü **Eigenschaften**.

Der Dialog **Eigenschaften - ImagePool_A** erscheint.

9. Wählen Sie die Registerkarte **Bildersammlung**.

10. Aktivieren Sie die Option **Nur verwendete Bilder auf Steuerung laden**.

11. Klicken Sie auf die Schaltfläche **Bibliothek als Symbolbibliothek kennzeichnen**.

Die Bibliothek ist eine Symbolbibliothek. Beim Laden einer Visualisierung lädt IndraLogic nur Bilder auf die Steuerung, die in der Visualisierung tatsächlich referenziert werden.

12. Wählen Sie in der Auswahlliste von **Textliste für Symbolübersetzung** die Textliste aus, die die IDs der Bildersammlung lokalisiert, zum Beispiel `ImagePool_A`.

13. Beenden Sie den Dialog mit **OK**.

Das Bibliotheksprojekt ist als Symbolbibliothek gekennzeichnet und enthält eine Bildersammlung mit zugeordneter Textliste.

14. Wählen Sie **Extras** ► **Bibliotheken** ► **Übersetzte Bibliotheken erzeugen und Installieren**.

Das Bibliotheksprojekt ist gespeichert und als Symbolbibliothek installiert.

Siehe auch

- [Kap. 16 "Bibliotheken erzeugen und verwalten" auf Seite 235](#)

Bildersammlung für Symbolbibliothek erstellen

Ein Bild, das in der Bildersammlung einer Symbolbibliothek enthalten ist, muss im Projekt eingebettet sein und den Verknüpfungstyp **eingebettet** haben.

Voraussetzung: Ein Bibliotheksprojekt für eine Symbolbibliothek ist geöffnet.

1. Wählen Sie **Ordner "Allgemeiner Baustein"** ► **Hinzufügen** ► **Bildersammlung**.

2. Geben Sie einen Namen ein, zum Beispiel `ImagePool_A`.
3. Doppelklicken Sie auf die Bildersammlung.
Der zugehörige Editor öffnet sich.
4. Doppelklicken Sie in ein Feld unter **Dateiname** und klicken Sie auf .
Dialog **Bildauswahl** erscheint.
5. Geben Sie im Eingabefeld **Bilddatei** den Dateinamen eines Bildes mit Pfad an, zum Beispiel `D:\Images\Image_A.bmp`.
6. Aktivieren Sie Option **Ins Projekt einbetten** und beenden Sie den Dialog mit **OK**.
Das Bild ist in der Liste der Bildersammlung mit Verknüpfungstyp **eingebettet** eingetragen.
7. Geben Sie unter **ID** eine ID an, zum Beispiel `Image_A` an.
Ein Bild ist der Bildersammlung mit den nötigen Eigenschaften zugeordnet.
8. Geben Sie weitere Bilder wie beschrieben in die Liste ein.
Die Bildersammlung ist erstellt.

Textliste zur Bildersammlung erstellen

Eine Textliste, die einer Bildersammlung zugeordnet wird, muss IDs enthalten, die in der Bildersammlung sind.

Voraussetzung: Ein Projekt mit Symbolbibliothek ist geöffnet.

1. Wählen Sie **Projekt ► Hinzufügen ► Textliste**.
2. Geben Sie einen Namen ein, zum Beispiel `TextList_A`.
3. Wählen Sie **Textliste ► Sprache einfügen**, und geben Sie den Namen der Sprache für ihre neue Spalte ein.
4. Geben Sie eine ID der Bildersammlung in der Textliste unter **ID** ein, zum Beispiel `Image_A`.
5. Geben Sie die Übersetzung in der Sprachspalte ein, zum Beispiel `Bild_A`.
Die ID `Image_A` ist lokalisiert.
6. Lokalisieren Sie möglichst alle IDs der Bildersammlung.
Die Textliste ist erstellt.

Siehe auch

- [Kap. 20.1.2 " Verfügbare Visualisierungselemente anzeigen und kennzeichnen" auf Seite 806](#)
- [Kap. 20.3.1 "Ansicht 'Werkzeuge'" auf Seite 826](#)
- [Kap. "Befehl 'Werkzeuge'" auf Seite 628](#)

20.2 Befehle

Die Visualisierungsbefehle werden für die Menükategorie **Visualisierungskommandos** zur Verfügung gestellt, die im Dialog **Extras ► Anpassen** zu fin-

den sind. Sie ermöglichen es, ein Visualisierungsobjekt im Visualisierungseditor zu editieren.

Die meisten sind standardmäßig Teil des Menüs **Visualisierung** und deswegen auch im Kontextmenü des Visualisierungseditors vorhanden. Falls nötig, ist der Dialog **Extras ▶ Anpassen** zu öffnen, um die Menükonfiguration der Kategorie **Visualisierungskommandos** anzusehen oder zu ändern.

20.2.1 Befehl 'Visualisierungselement einfügen'

Dieser Befehl (Kategorie **Visualisierungskommandos**) steht im Menü **VI-Logic Visualisierung** und im Kontextmenü zur Verfügung.

Er wird verwendet, um ein Element in der aktuellen Visualisierung einzufügen. Dies entspricht dem Einfügen per Drag & Drop in der Ansicht **Werkzeuge** (Kap. 20.3.1 "Ansicht 'Werkzeuge'" auf Seite 826).

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.

20.2.2 Befehl 'Schnittstelleneditor'

Symbol: 

Standardtastaturkürzel: <Alt> + <F6>

Dieser Befehl (Kategorie **Visualisierungskommandos**) öffnet den Schnittstelleneditor um Frameparameter in der Visualisierung zu definieren, die dazu bestimmt sind im Frame-Element einer anderen Visualisierung referenziert zu werden.

Siehe auch:

- [Kap. "Grundlegendes zum Schnittstelleneditor" auf Seite 831](#)
- [Kap. "Element 'Frame'" auf Seite 917](#)

20.2.3 Befehl 'Tastaturbedienung'

Dieser Befehl (Kategorie **Visualisierungskommandos**) öffnet den Dialog 'Tastaturbedienung' für die aktuelle Visualisierung (Kap. 20.3.4 "Registerkarte 'Tastaturkonfiguration'" auf Seite 839).

Der Befehl wird in einer Registerblattansicht im oberen Teil des Visualisierungs-Editors angezeigt.

20.2.4 Befehl 'Elementliste'

Symbol: 

Dieser Befehl (Kategorie **Visualisierungskommandos**) öffnet die Elementliste für die aktuelle Visualisierung.

Er wird in einer Registerblattansicht im oberen Teil des Visualisierungseditors angezeigt.

Siehe auch:

- [Kap. 20.3.5 "Registerkarte 'Elementliste'" auf Seite 841](#)

20.2.5 Befehl 'Tastaturbedienung aktivieren'

Symbol: 

Dieser Befehl (Kategorie **Visualisierungskommandos**) ist in der Menüleiste für eine integrierte Visualisierung verfügbar. Er aktiviert bzw. deaktiviert die Tastaturbedienung im Online-Modus einer Visualisierung.

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.

Siehe auch:

- [Kap. "Diagnosevisualisierung" auf Seite 1061](#)
- [Kap. 20.11 "Tastaturbedienung im Online-Modus" auf Seite 1081](#)

20.2.6 Ausrichtung

Befehl 'Links ausrichten'

Symbol: 

Dieser Befehl (Kategorie **Visualisierungskommandos**) bewirkt, dass alle ausgewählten Visualisierungselemente am linken Rand ihres am weitesten links positionierten Elements ausgerichtet werden.

Befehl 'Oben ausrichten'

Symbol: 

Dieser Befehl (Kategorie **Visualisierungskommandos**) bewirkt, dass alle ausgewählten Visualisierungselemente am oberen Rand ihres am weitesten oben positionierten Elements ausgerichtet werden.

Befehl 'Rechts ausrichten'

Symbol: 

Dieser Befehl (Kategorie **Visualisierungskommandos**) bewirkt, dass alle ausgewählten Visualisierungselemente am rechten Rand ihres am weitesten rechts positionierten Elements ausgerichtet werden.

Befehl 'Unten ausrichten'

Symbol: 

Dieser Befehl (Kategorie **Visualisierungskommandos**) bewirkt, dass alle ausgewählten Visualisierungselemente am unteren Rand ihres am weitesten unten positionierten Elements ausgerichtet werden.

Befehl 'Vertikal zentrieren'

Symbol: 

Dieser Befehl (Kategorie **Visualisierungskommandos**) bewirkt, dass alle ausgewählten Visualisierungselemente an ihrer gemeinsamen vertikalen Mitte ausgerichtet werden.

Befehl 'Horizontal zentrieren'

Symbol: 

Dieser Befehl (Kategorie **Visualisierungskommandos**) bewirkt, dass alle ausgewählten Visualisierungselemente an ihrer gemeinsamen horizontalen Mitte ausgerichtet werden.

Befehl 'Horizontalen Abstand gleichmachen'

Symbol: 

Der Befehl wird aktiv, wenn drei oder mehr Elemente selektiert sind.

Vorgehensweise

1. Selektieren Sie alle Elemente, die horizontal mit gleichem Abstand positioniert werden sollen.
Das erste Element ist blau markiert, die weiteren Elemente sind grau markiert.
2. Führen Sie den Befehl **Horizontalen Abstand gleichmachen** aus.
Die Elemente werden so positioniert, dass das linkeste Element und das rechteste Element seine Position behalten und die Elemente dazwischen horizontal mit gleichem Abstand ausgerichtet werden.

Befehl 'Horizontalen Abstand vergrößern'

Symbol: 

Der Befehl wird aktiv, wenn zwei oder mehr Elemente selektiert sind.

Vorgehensweise

1. Selektieren Sie alle Elemente, die horizontal mit mehr Abstand positioniert werden sollen.
Das erste Element ist blau markiert, die weiteren Elemente sind grau markiert.
2. Führen Sie den Befehl **Horizontalen Abstand vergrößern** aus.
Die Elemente werden so positioniert, dass das blaue Element seine Position behält und die weiteren Elemente horizontal mit mehr Abstand zwischen den Elementen ausgerichtet werden. Der Abstand vergrößert sich um 1 Pixel.

Befehl 'Horizontalen Abstand verkleinern'

Symbol: 

Der Befehl wird aktiv, wenn zwei oder mehr Elemente selektiert sind.

Vorgehensweise

1. Selektieren Sie alle Elemente, die horizontal mit weniger Abstand positioniert werden sollen.
2. Führen Sie den Befehl **Horizontalen Abstand verkleinern** aus.
Die Elemente werden so positioniert, dass das blaue Element seine Position behält und die weiteren Elemente horizontal mit weniger Abstand zwischen den Elementen ausgerichtet werden. Der Abstand verkleinert sich um 1 Pixel.

Befehl 'Horizontalen Abstand entfernen'

Symbol: 

Der Befehl wird aktiv, wenn zwei oder mehr Elemente selektiert sind.

Vorgehensweise

1. Selektieren Sie alle Elemente, die horizontal ohne Abstand zueinander positioniert werden sollen.

Das erste Element ist blau markiert, die weiteren Elemente sind grau markiert.

2. Führen Sie den Befehl **Horizontalen Abstand entfernen** aus.

Die Elemente werden so positioniert, dass das blaue Element seine Position behält und die weiteren Elemente horizontal ohne Abstand zueinander ausgerichtet werden.

Befehl 'Vertikalen Abstand gleichmachen'

Symbol: 

Der Befehl wird aktiv, wenn zwei oder mehr Elemente selektiert sind.

Vorgehensweise

1. Selektieren Sie alle Elemente, die vertikal mit gleichem Abstand positioniert werden sollen.

Das erste Element ist blau markiert, die weiteren Elemente sind grau markiert.

2. Führen Sie den Befehl **Vertikalen Abstand gleichmachen** aus.

Die Elemente werden so positioniert, dass das oberste Element und das unterste Element seine Position behalten und die Elemente dazwischen vertikal mit gleichem Abstand ausgerichtet werden.

Befehl 'Vertikalen Abstand vergrößern'

Symbol: 

Der Befehl wird aktiv, wenn zwei oder mehr Elemente selektiert sind.

Vorgehensweise

1. Selektieren Sie alle Elemente, die vertikal mit mehr Abstand positioniert werden sollen.

Das erste Element ist blau markiert, die weiteren Elemente sind grau markiert.

2. Führen Sie den Befehl **Vertikalen Abstand vergrößern** aus.

Die Elemente werden so positioniert, dass das blaue Element seine Position behält und die weiteren Elemente vertikal mit mehr Abstand zwischen den Elementen ausgerichtet werden. Der Abstand vergrößert sich um 1 Pixel.

Befehl 'Vertikalen Abstand verkleinern'

Symbol: 

Der Befehl ist aktiv, wenn zwei oder mehr Elemente selektiert sind.

Vorgehensweise

1. Selektieren Sie alle Elemente, die vertikal mit weniger Abstand positioniert werden sollen.

Das erste Element ist blau markiert, die weiteren Elemente sind grau markiert.

2. Führen Sie den Befehl **Vertikalen Abstand verkleinern** aus.

Die Elemente werden so positioniert, dass das blaue Element seine Position behält und die weiteren Elemente vertikal mit weniger Abstand zwischen den Elementen ausgerichtet werden. Der Abstand verkleinert sich um 1 Pixel.

Befehl 'Vertikalen Abstand entfernen'

Symbol: 

Der Befehl wird aktiv, wenn zwei oder mehr Elemente selektiert sind.

Vorgehensweise

1. Selektieren Sie alle Elemente, die vertikal ohne Abstand zueinander positioniert werden sollen.

Das erste Element ist blau markiert, die weiteren Elemente sind grau markiert.

2. Führen Sie den Befehl **Vertikalen Abstand entfernen** aus.

Die Elemente werden so positioniert, dass das blaue Element seine Position behält und die weiteren Elemente vertikal ohne Abstand zueinander ausgerichtet werden.

Befehl 'Breite gleichmachen'

Symbol: 

Der Befehl wird aktiv, wenn mehr als ein Element selektiert ist, außer ein Linien- oder Polygonelement ist selektiert.

Vorgehensweise

1. Selektieren Sie alle Elemente, die die gleiche Breite bekommen sollen.

Das erste Element ist blau markiert, die weiteren Elemente sind grau markiert.

2. Führen Sie den Befehl **Breite gleichmachen** aus.

Alle Elemente bekommen die Breite des blau markierten Elements.

Befehl 'Höhe gleichmachen'

Symbol: 

Der Befehl wird aktiv, wenn mehr als ein Element selektiert ist, außer ein Linien- oder Polygonelement ist selektiert.

Vorgehensweise

1. Selektieren Sie alle Elemente, die die gleiche Höhe bekommen sollen.

Das erste Element ist blau markiert, die weiteren Elemente sind grau markiert.

2. Führen Sie den Befehl **Höhe gleichmachen** aus.

Alle Elemente bekommen die Höhe des blau markierten Elements.

Befehl 'Größe gleichmachen'

Symbol: 

Der Befehl ist aktiv, wenn mehr als ein Element selektiert ist, außer ein Linien- oder Polygonelement ist selektiert.

Vorgehensweise

1. Selektieren Sie alle Elemente, die die gleiche Größe bekommen sollen.
Das erste Element ist blau markiert, die weiteren Elemente sind grau markiert.
2. Führen Sie den Befehl **Größe gleichmachen** aus.
Alle Elemente bekommen die Größe des blau markierten Elements.

Befehl 'Größe an Raster anpassen'

Symbol: 

Der Befehl wird aktiv, wenn mehr als ein Element selektiert ist, außer ein Linien- oder Polygonelement ist selektiert.

Vorgehensweise

1. Selektieren Sie alle Elemente, die in Position und Größe am Raster ausgerichtet werden sollen.
Das erste Element ist blau markiert, die weiteren Elemente sind grau markiert.
2. Führen Sie den Befehl **Größe an Raster anpassen** aus.
Alle Elemente werden in Größe und Position am Raster ausgerichtet.

20.2.7 Reihenfolge

Befehl 'Um Eins nach vorn legen'

Symbol: 

Dieser Befehl (Kategorie **Visualisierungskommandos**) 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.

Befehl 'Nach vorn legen'

Symbol: 

Dieser Befehl (Kategorie **Visualisierungskommandos**) positioniert das ausgewählte Element im Vordergrund der Visualisierung, d.h. auf höchster Ebene. Elemente auf tieferen Ebenen werden von denen auf höheren verdeckt.

Befehl 'Um Eins nach hinten legen'

Symbol: 

Dieser Befehl (Kategorie **Visualisierungskommandos**) 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.

Befehl 'Nach hinten legen'

Symbol: 

Dieser Befehl (Kategorie **Visualisierungskommandos**) positioniert das ausgewählte Element im Hintergrund der Visualisierung, d.h. auf tiefster Ebene.

Elemente auf tieferen Ebenen werden von denen auf höheren verdeckt.

20.2.8 Befehl 'Gruppieren'

Symbol: 

Mit diesem Befehl (Kategorie **Visualisierungskommandos**) 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:

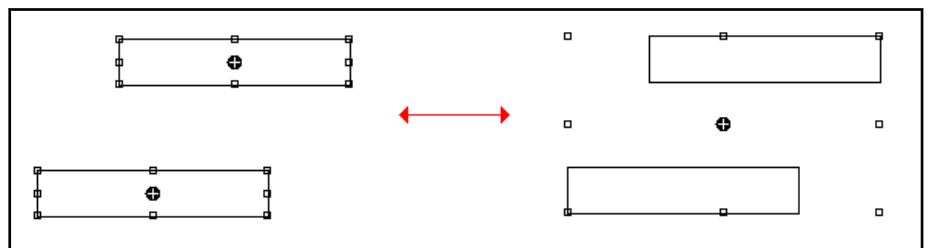


Abb. 20-2: Gruppieren und Gruppierung aufheben

Siehe auch:

- [Kap. 20.2.9 "Befehl 'Gruppierung aufheben'" auf Seite 819](#)

20.2.9 Befehl 'Gruppierung aufheben'

Symbol: 

Mit diesem Befehl (Kategorie **Visualisierungskommandos**) wird eine Gruppe von Visualisierungselementen wieder aufgelöst. Die einzelnen Elemente werden jeweils wieder einzeln selektiert dargestellt.

Siehe auch:

- [Kap. 20.2.8 "Befehl 'Gruppieren'" auf Seite 819.](#)

20.2.10 Befehl 'Frame-Auswahl'

Dieser Befehl (Kategorie **Visualisierungskommandos**) wird verwendet um einen Frame zu konfigurieren ([Kap. "Element 'Frame'" auf Seite 917](#)).

Ein Frame-Element wird verwendet, um in einer Visualisierung eine Untereinheit zu definieren, die eine oder mehrere andere Visualisierungen enthalten kann. Eine davon ist im Online-Betrieb aktiv. Zwischen den verfügbaren Visualisierungen hin- und hergeschaltet werden kann. Grundsätzlich ist zunächst die erste Visualisierung aus der Liste des Frames aktiv ("[Framevisualisierung umschalten" auf Seite 0](#)).

Somit ist ein Wechseln zwischen verschiedenen Darstellungen innerhalb einer übergeordneten Visualisierung möglich, im Gegensatz zu IndraLogic 1x-Visualisierung, wo komplett zwischen den einzelnen Visualisierungen hin- und hergesprungen werden mußte.

Visualisierungen, die in einem Frame zugewiesen wurden, sind Referenzen. Das heißt sie instanzieren eine Original-Visualisierung und können Platzhalter, die in der Original-Visualisierung definiert wurden, mit lokal geltenden Werten füllen.

Ist ein Frame im Editor selektiert, kann über den nun aktiven Befehl **Frame Auswahl** die Visualisierungsauswahl vorgenommen werden. Dafür steht der Dialog **Framekonfiguration** zur Verfügung.

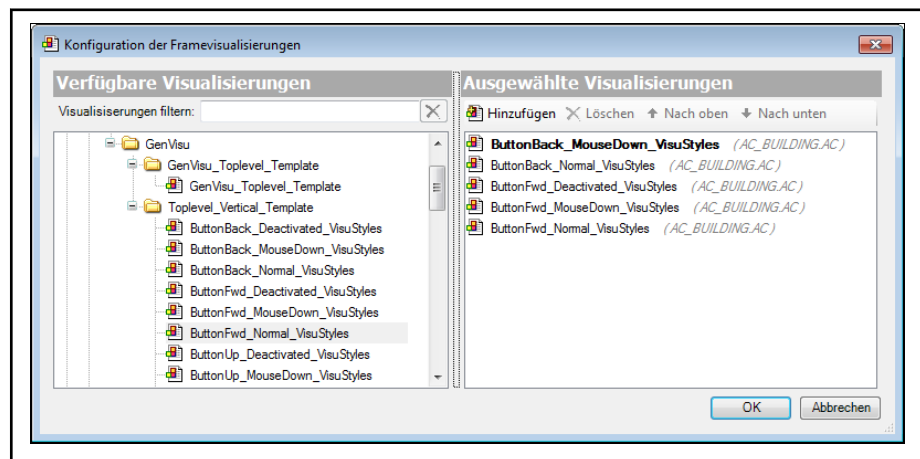


Abb. 20-3: Dialog 'Frame Konfiguration'

Auf der linken Seite sehen Sie die im Projekt **Verfügbaren Visualisierungen**. Wählen Sie diejenigen aus, die im Frame referenziert werden sollen. Durch einen Doppelklick bzw. über den Befehl **Hinzufügen** können sie zur Liste der **Ausgewählten Visualisierungen** auf der rechten Seite des Dialogs hinzugefügt werden. Ausgewählte Visualisierungen können durch einen Doppelklick bzw. durch den Befehl **Löschen** aus dieser Liste entfernt werden. Eine Mehrfachauswahl von Visualisierungen ist ebenfalls möglich. Die Reihenfolgen innerhalb der Liste kann durch die Befehle **Nach oben** und **Nach unten** verändert werden.

Empfehlenswert ist einem Frame nur Visualisierungen aus dem globalen Pool zuzuweisen. Ansonsten könnte es zu Problemen kommen, falls später Geräte- oder Applikationsobjekte umbenannt werden und dann der Pfad der zugewiesenen Visualisierungen nicht mehr gültig ist.

Die Reihenfolge der ausgewählten Visualisierungen von oben nach unten ist bestimmt durch automatisch erzeugte impliziten Indexnummern der Visualisierungen. Die oberste erhält "0", die folgenden "1", "2" usw.. Die Indexnummern werden für die Konfiguration der Umschaltfunktion eines anderen Elements benötigt. Initial wird die Visualisierung mit Index "0" dargestellt.

Beispiel

Über eine Schaltflächenleiste soll festgelegt werden, welche der Visualisierungen aktuell im Online-Modus angezeigt werden soll.

1. Einfügen eines Frame-Elements
2. Diesem Frame ist über **Frame-Auswahl** in Dialog **Framekonfiguration** alle Visualisierungen zuzuweisen, zwischen denen umgeschaltet wird.

3. Für jede Visualisierung ist ein Steuerelement einzufügen, das ermöglicht nach Benutzereingabe auf diese umzuschalten. Zum Beispiel wird eine Schaltflächenleiste mit Steuerelementen eingefügt.
4. Jedes Steuerelement wird konfiguriert, indem dessen Eingabekonfiguration konfiguriert wird:

In Eigenschaft `OnClick` wird die Folgeaktion **Framevisualisierung umschalten** hinzugefügt und dort die bestimmte Visualisierung dem bestimmten Frame zugewiesen.

Startvisualisierung

Die "Startvisualisierung", d. h. das Visualisierungsobjekt, das als erstes nach dem Einloggen mit der Applikation auf der Steuerung angezeigt werden soll, muss im Projektextplorer 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 Visualisierungsmanager ausgewählt werden. Daraufhin wird diese Option automatisch in allen anderen Visualisierungsobjekten deaktiviert.

Die Startvisualisierungen unter einer Applikation wechselt mit der Applikation in den Online-Modus.

Da Visualisierungen im Ordner "Allgemeiner Baustein" keinen direkten Bezug zu einer Applikation haben, wechseln diese nicht in den Online-Modus.

Wenn in einer Visualisierung unter der Applikation ein Visualisierungswechsel zu einer Visualisierung im Ordner "Allgemeiner Baustein" konfiguriert ist, so lässt sich diese dadurch im Online-Modus erreichen.

Siehe auch:

- [Kap. "Element 'Frame'" auf Seite 917](#)
- ["Framevisualisierung umschalten" auf Seite 0](#)
- [Kap. "Eingabekonfiguration" auf Seite 849](#)

20.2.11 Befehl 'Hintergrund'

Symbol: 

Dieser Befehl (Kategorie **Visualisierungskommandos**) 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.

20.2.12 Befehl 'Alles selektieren'

Mit diesem Befehl (Kategorie **Visualisierungskommandos**) werden alle Elemente der gerade im Editor geöffneten Visualisierung ausgewählt.

Siehe auch:

- [Kap. 20.2.13 "Befehl 'Alles deselektieren'" auf Seite 822](#)

20.2.13 Befehl 'Alles deselektieren'

Mit diesem Befehl (Kategorie **Visualisierungskommandos**) wird die aktuelle Selektion von Visualisierungselementen aufgehoben.

Siehe auch:

- [Kap. 20.2.12 "Befehl 'Alles selektieren'" auf Seite 821.](#)

20.2.14 Befehl 'Trace konfigurieren...'

Der Befehl ist verfügbar, wenn ein Element 'Trace' im Visualisierungseditor selektiert ist.

Verwenden Sie den Befehl, um den Dialog "Tracekonfiguration, Aufzeichnungseinstellungen" zu öffnen.

Siehe auch:

- [Kap. "Element 'Trace'" auf Seite 1032](#)
- [Kap. "Dialog 'Tracekonfiguration, Aufzeichnungseinstellungen'" auf Seite 1041](#)

20.2.15 Befehl 'Elemente für Trace-Kontrolle einfügen'

Dieser Befehl ist verfügbar, wenn ein Element 'Trace' im Visualisierungseditor selektiert ist ([Kap. "Element 'Trace'" auf Seite 1032](#)).

Der Befehl öffnet der Dialog [Kap. 20.2.15 "Befehl 'Elemente für Trace-Kontrolle einfügen'" auf Seite 822](#).

20.2.16 Befehl 'Frame-Parameter in <geöffnete Visualisierung> aktualisieren'

Der Befehl ist im angepassten Menü nur vorhanden und sichtbar, wenn der Visualisierungseditor geöffnet ist.

Gewöhnlich ist der Befehl dann notwendig, wenn es Compile-Fehler gibt, die Frame-Parameter betreffen. Mit dem folgenden Dialog können Sie dann die Frame-Parameter aktualisieren.

Typischerweise kommt die parametrisierte Visualisierung aus einer Bibliothek, wurde dort aktualisiert und ein automatischer Update der Parameter im Projekt fand noch nicht statt.

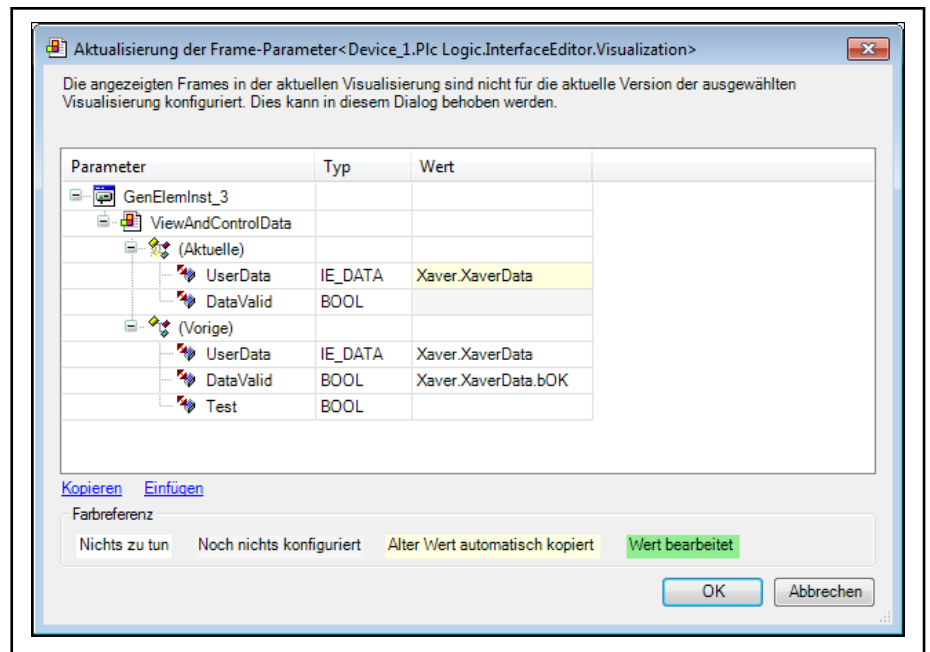


Abb. 20-4: Aktualisierung der Frame-Parameter <Name der Visualisierung>

Die Visualisierung, die neu konfiguriert werden kann, wird in Klammern <...> im Titel des Dialogs angezeigt.

Verwendung des Dialogs:

- [Kap. "Aktualisierung der Frameparameter" auf Seite 833](#)

20.2.17 Befehl 'Visualisierungselement vervielfachen'

Symbol:

Funktion: Der Befehl öffnet den Dialog **Visualisierungselement vervielfachen**, mit dem Sie das Vervielfachen konfigurieren können.

Aufruf: Menü **Visualisierung**, Kontextmenü

Voraussetzung: Die Visualisierung ist aktiv und ein Vorlagenelement ist selektiert.

Dialog 'Visualisierungselement
vervielfachen'

IndraLogic fügt beim Vervielfachen weitere Elemente ein, die dem Vorlagenelement gleichen. Sie können in diesem Dialog die Anzahl und die Anordnung, sowie die Ersetzung der Indizes konfigurieren.

Registerkarte 'Grundeinstellung'

Gesamtzahl Elemente

IndraLogic fügt die neuen Elemente als Tabelle ein. Die Zeilenanzahl ist in Horizontal eingestellt, die Spaltenanzahl in Vertikal . Das Produkt der beiden bestimmt die tatsächliche Gesamtzahl der einzufügenden Elemente.	
Horizontal	Anzahl an Elementen pro Zeile Voreinstellung: Entspricht der Anzahl der Komponenten in \$FIRSTDIM\$
Vertikal	Anzahl an Elementen pro Spalte Voreinstellung: Entspricht der Anzahl der Komponenten in \$SECONDDIM\$

Offset zwischen den Elementen

IndraLogic ordnet die Elemente in der Visualisierung als Tabelle an. Wenn Sie einen **Offset** angeben, wird zwischen den Elementen ein Abstand eingefügt.

- **0** : Die Rahmen der Elemente überlappen um ein Pixel
- **1** : Die Elemente stoßen aneinander
- **<n>** : Zwischen den Elementen ist ein sichtbarer Abstand von n-1 Pixel

Horizontal	Zeilenabstand der Elemente in Pixel
Vertikal	Spaltenabstand der Elemente in Pixel

Registerkarte 'Erweiterte Einstellung'

Erste Dimension

Startindex	Startindex für \$FIRSTDIM\$ Voreinstellung: 1, bedeutet, dass der konkrete Index für \$FIRSTDIM\$ mit 1 beginnt. Beispiel: Array[1, <\$SECONDDIM\$>]
Hochzählen	Inkrement, um das der Index hochgezählt wird Voreinstellung: 1

Zweite Dimension

Startindex	Startindex für \$SECONDDIM\$ Voreinstellung: 1, bedeutet, dass der konkrete Index für \$SECONDDIM\$ mit 1 beginnt. Beispiel Array[<\$FIRSTDIM\$>, 1]
Hochzählen	Inkrement, um das der Index hochgezählt wird Voreinstellung: 1

Siehe auch

- [Kap. "Vervielfachen eines Elements" auf Seite 865](#)

20.3 Editor

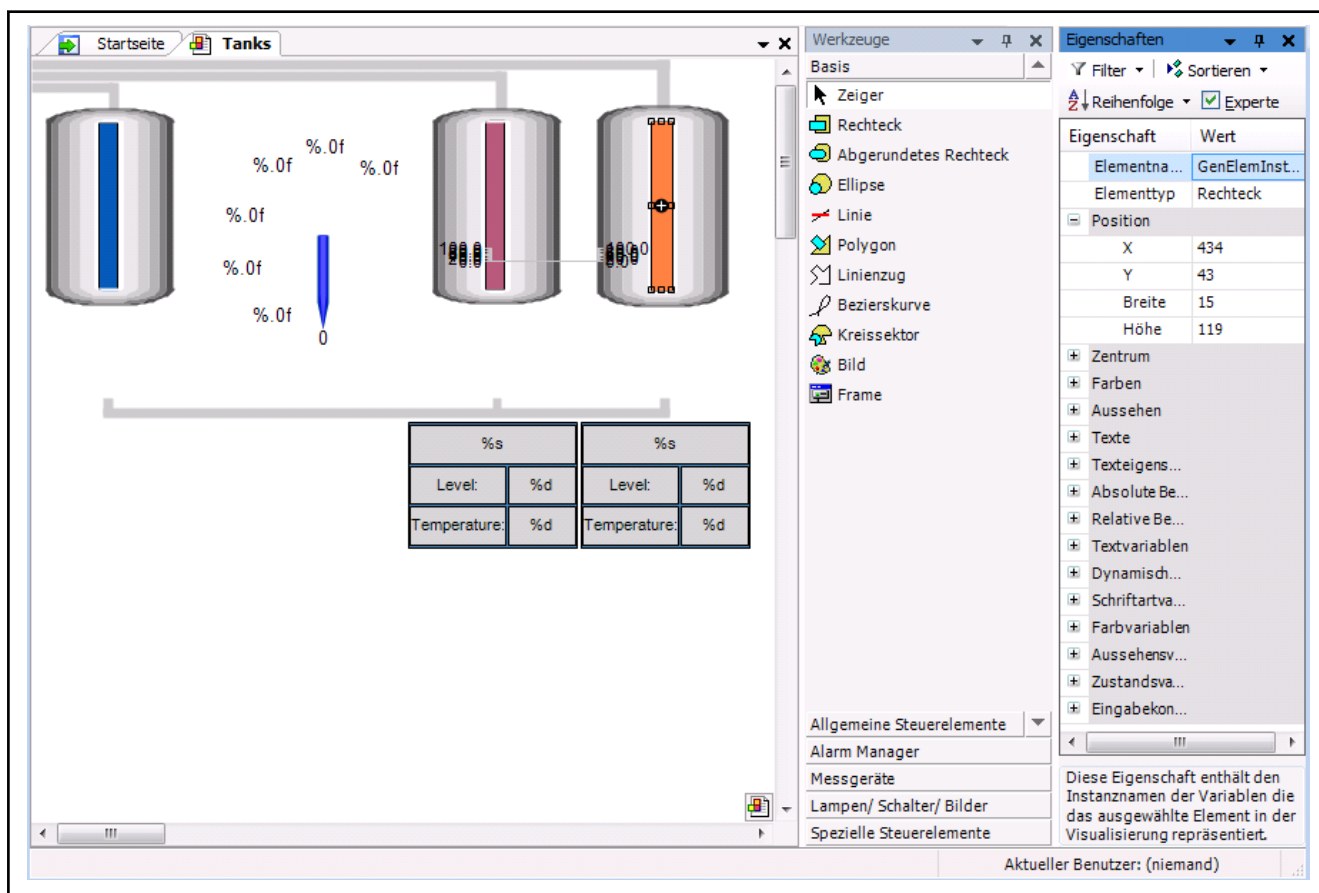


Abb. 20-5: Visualisierungeditor

Der Visualisierungeditor ist über das Plug-in 'Visualization Editor' verfügbar. Eine Visualisierung kann mit Hilfe folgender Editoren und Werkzeugen erstellt werden:

- dem Editor selbst, betitelt mit  <Name der Visualisierung>[<Objektpfad>], der die gestaltete, editierte und programmierte Visualisierung enthält.
- der Ansicht **Werkzeuge**, die alle aktuell verfügbaren Visualisierungselemente enthält.
- der Ansicht **Eigenschaften**, die alle Eigenschaften des Visualisierungselements enthält, das aktuell im Editor selektiert ist.
- den Befehlen des Menüs **Visualisierung** und denjenigen des Kontextmenüs im Editor
- dem **Schnittstelleneditor** um Platzhalter zu definieren, falls die Visualisierung in eine andere eingefügt werden soll.
- der **Tastaturkonfiguration** um Tasten oder Tastenkombinationen Folgeaktionen zuzuweisen. Zu beachten ist dabei jedenfalls, dass das Gerät, das die Visualisierung verwendet, die jeweiligen Tasten unterstützt.
- der **Elementliste**, die einen Überblick über alle Elemente und deren Position der aktuellen Visualisierung zur Verfügung stellt und Selektion, Mehrfachselektion und Löschen erlaubt.

Wenn **Schnittstelleneditor**, **Tastaturkonfiguration** oder **Elementliste** über den entsprechenden Befehl (standardmäßig im Menü **Visualisierung**) aktiviert wird, wird der Visualisierungseditor zweigeteilt und enthält im oberen Teil die entsprechenden Registerblätter.

IndraLogic 1x-Visualisierungen können importiert werden.

Die Visualisierungselemente können über eine direkte Verwendung von IndraLogic-Projektvariablen animiert werden oder in Form von Ausdrücken, das heißt kombiniert mit Operatoren und Konstanten. Dies erlaubt zum Beispiel die Variablen für eine Verwendung in der Visualisierung zu skalieren.

Innerhalb einer Visualisierung sind beliebige Ausdrücke und auch Funktionsaufrufe gültig.

20.3.1 Ansicht 'Werkzeuge'

Symbol: 

Funktion: Die Ansicht bietet in einer tabellarischen Anordnung Elemente als Vorschaubilder an, die Sie in den Visualisierungseditor ziehen können. Nur diejenigen Vorschaubilder, die mindestens ein Kennzeichen haben, das selektiert und grün ist, sind sichtbar.

Sie können die Selektion ändern, indem Sie auf die Schaltfläche eines Kennzeichens klicken. IndraLogic erstellt die Kennzeichen automatisch.

Sie können aber auch ein Kennzeichen hinzufügen. Außerdem können Sie die Größe der Vorschaubilder mit dem Schiebebalken einstellen oder nach dem Namen eines Elements über Volltextsuche suchen.

Aufruf: Menü **Ansicht ► Werkzeuge**

Voraussetzung: Eine Visualisierung ist aktiv.

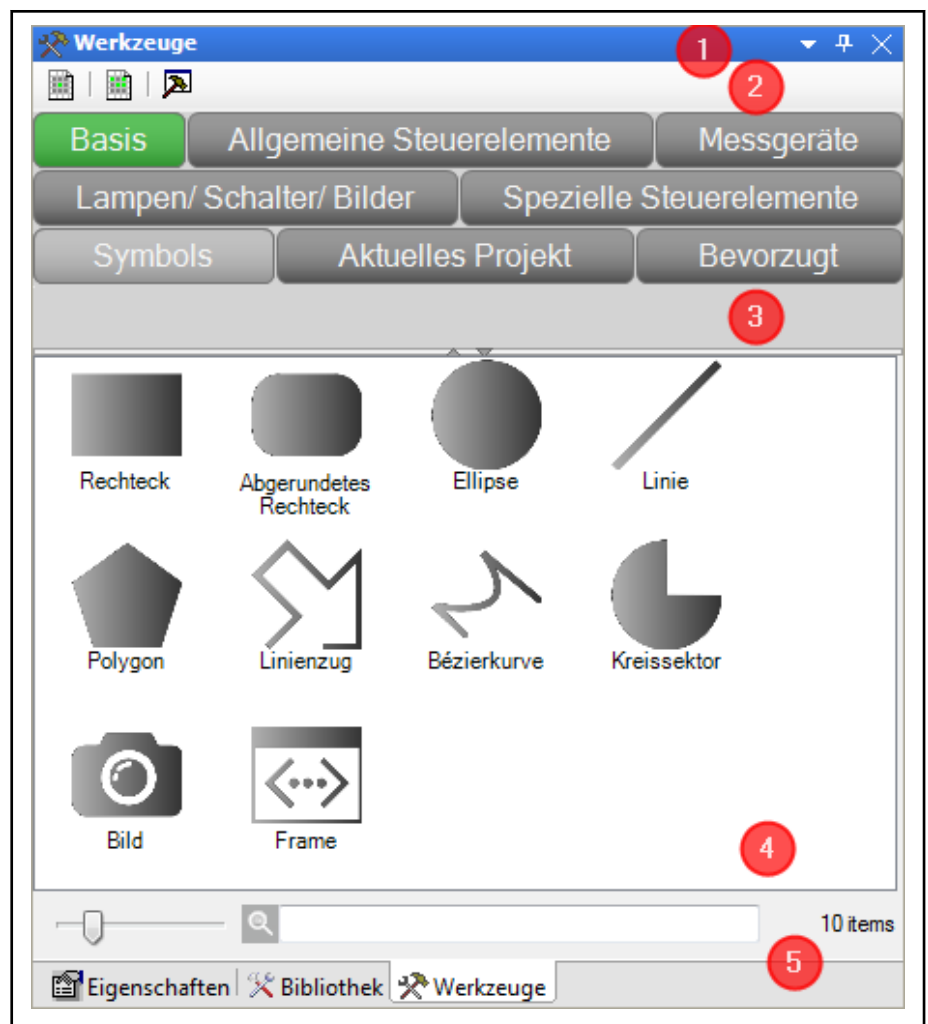


Abb. 20-6:

- (1) : Ansicht **Werkzeuge**
- (2) : Symbolleiste für Befehle
- (3) : Schaltflächen für Kennzeichen
- (4) : Tabellarische Anordnung der Vorschaubilder der gekennzeichneten Elemente
- (5) : Steuerelemente

Symbolleiste

	Selektionstyp "Einzelselektion": In den Schaltflächen für Kennzeichen ist höchstens ein Kennzeichen selektiert und grün
	Selektionstyp "Mehrfachselektion": In den Schaltflächen für Kennzeichen können mehrere Kennzeichen selektiert und grün sein.
	Dialog Kennzeichen und Einträge verwalten erscheint.

Schaltflächen der Kennzeichen

IndraLogic zeigt für jedes automatisch zugewiesene oder von Ihnen erstellte Kennzeichen eine Schaltfläche an. Jedem Kennzeichen ist mindestens ein Eintrag zugeordnet. Ein Eintrag kann aber auch mehreren Kennzeichen zugeordnet sein.

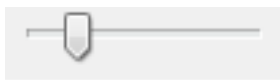
Schaltfläche	Die Schaltfläche ist beschriftet mit dem Namen des Kennzeichens. - Graue Schaltfläche, zum Beispiel . Das Kennzeichen ist nicht selektiert und die Einträge, die ihm zugeordnet sind, werden nicht angezeigt. - Grüne Schaltfläche, zum Beispiel . Das Kennzeichen ist selektiert und die Einträge, die ihm zugeordnet sind, werden angezeigt.
Klick auf eine graue Schaltfläche	Schaltfläche wird grün und das Kennzeichen ist selektiert. Die zugeordneten Einträge werden angezeigt. Das Verhalten der anderen Kennzeichen variiert, je nachdem welcher Selektionstyp aktiviert ist.
Klick auf eine grüne Schaltfläche	Voraussetzung: Mehrfachselektion ist aktiv Schaltfläche wird grau und das Kennzeichen ist nicht selektiert. Die zugeordneten Einträge werden nicht mehr angezeigt.
Klick auf eine Schaltfläche bei gedrückter <Umschalttaste>	Die Schaltfläche schaltet um. Der Selektionsverhalten ist für diesen Klick auch umgeschaltet und das Verhalten der anderen Kennzeichen variiert, je nachdem welcher Selektionstyp aktiviert ist. Wenn Einzelselektion aktiviert ist, ist das angeklickte Kennzeichen zusätzlich selektiert. Wenn Mehrfachselektion aktiviert ist, ist das angeklickte Kennzeichen als einziges selektiert.
Kennzeichen verbergen	Entfernt das Kennzeichen aus der tabellarischen Anordnung. Sie können ein Kennzeichen in Dialog Kennzeichen und Einträge konfigurieren sichtbar konfigurieren. Voraussetzung: Eine Schaltfläche ist fokussiert und das Kontextmenü ist aktiviert.

Tabellarische Ansicht der Vorschaubilder

Elemente, die IndraLogic aus dem Projekt oder den erreichbaren Bibliotheken ermittelt, werden abhängig vom Kennzeichen mit Vorschaubild angezeigt und mit Namen beschriftet.

Wenn für eine Bildersammlung die Objekteigenschaft **Intern** aktiviert wurde, zeigt IndraLogic den Inhalt der Bildersammlung nicht in der Ansicht **Werkzeuge** an.

Steuerelemente

Schieberegler 	Schieberegler um die Größe der Vorschaubilder zu ändern
 mit Eingabefeld	Für Volltextsuche. IndraLogic filtert die Namen der Einträge nach der Zeichenfolge, die Sie im Eingabefeld eingeben haben und zeigt die betroffenen Einträge an.
<Anzahl> Einträge	Anzahl der Einträge von Elementen, die Sie über die Schaltflächen der Kennzeichen und den Filter ausgewählt haben

Siehe auch

- [Kap. 20.1.2 " Verfügbare Visualisierungselemente anzeigen und kennzeichnen" auf Seite 806](#)
- [Kap. 20.3.7 "Dialog 'Kennzeichen und Einträge konfigurieren'" auf Seite 842](#)
- [Kap. "Befehl 'Werkzeuge'" auf Seite 628](#)

20.3.2 Ansicht 'Eigenschaften'

Symbol:

Das Plug-in 'Visualization Element' stellt den Befehl **Ansicht ► Elementeigenschaften** zur Verfügung, das verwendet wird um den Editor **Eigenschaften** für ein selektiertes Visualisierungselement zu öffnen. Wenn nötig, ist der Dialog **Extras ► Anpassen** zu öffnen um die Menükonfiguration in Kategorie **Ansicht** anzusehen bzw. zu ändern.

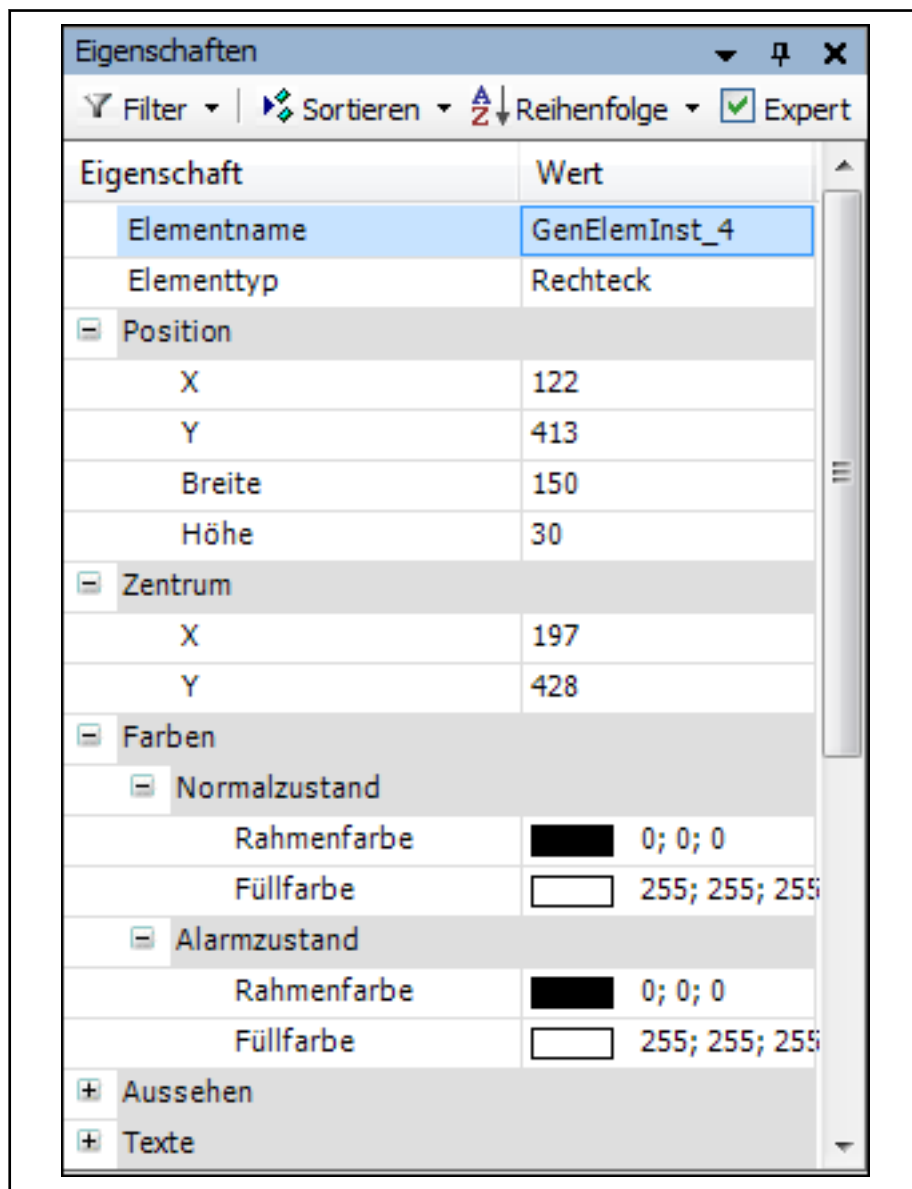


Abb. 20-7: Editor der Elementeigenschaften

Der Editor **Eigenschaften** listet alle Eigenschaften des selektierten Elements in einer Tabelle auf und stellt verschiedene Benutzereingaben zur Verfügung, um den dessen **Wert** zu ändern. Eine Beschreibung der Eigenschaften und ihre Verwendung ist in der Beschreibung der Elemente aufgelistet: [Kap. 20.4 "Visualisierungselemente"](#) auf Seite 843.

Die Menüleiste oben stellt Befehle und Optionen zur Verfügung:

- **Filter:** Auswahl an Filtereinstellungen um die Eigenschaftenliste den Bedürfnissen des Anwenders anzupassen.
 - **Alle Kategorien anzeigen:** alle Eigenschaften
 - **Einfach:** nur die grundlegenden Eigenschaften wie **Texte**, **Farben** und **Eingabekonfiguration**
 - **Standard:** die am häufigsten verwendeten Eigenschaften (die Auswahl basiert auf einen Eigenschaftensatz wie in V2.2 verwendet).
 - **Animation:** nur Eigenschaften, die zur Animation verwendet werden
 - **Farben:** nur Eigenschaften, die auf Farben verweisen
 - **Farbverlauf:** nur Eigenschaften, die auf Farbverlauf verweisen
 - **Texte:** nur Eigenschaften, die auf Text verweisen
 - **Eingaben:** nur Eigenschaften, die auf Benutzereingaben verweisen
- **Sortieren nach:**
 - **Nach Typ sortieren:** vollständige Liste der Eigenschaften in der original Reihenfolge der Kategorien
 - **Nach Name sortieren:** vollständige Liste der Eigenschaften nach alphabetischer Reihenfolge der Kategorienamen
- **Reihenfolge:**
 - **Aufsteigend sortieren:** vollständige Liste der Eigenschaften in der original Reihenfolge von unten nach oben
 - **Absteigend:** vollständige Liste der Eigenschaften in der original Reihenfolge von oben nach unten
- Checkbox **Expert:** wird diese Option aktiviert, werden alle Eigenschaften angezeigt und **Alle Kategorien anzeigen** ist gesetzt.

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.

- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)



Wenn ein Stil angewendet wird, dann hängt die Liste der auswählbaren Farben und Schriftarten von den Definitionen in der jeweiligen Stildatei ab.

In Visualisierungen, die einem Gerät zugehören, gibt es Geräte unabhängige Einschränkungen, die die Möglichkeiten der Konfiguration betreffen.

Beispiele sind: Liste der verfügbaren Elemente, Schriftarten, Farben, Bildformate, max. Anzahl von Visualisierungselementen in einer Visualisierung, max. Anzahl von Visualisierungen unterhalb eines Geräts.

20.3.3 Registerkarte 'Schnittstelleneditor'

Er wird verwendet, um Schnittstellen in Visualisierungen zu definieren, die dazu bestimmt sind in einem Frame einer anderen Visualisierungen referenziert zu werden.

Grundlegendes zum Schnittstelleneditor

Dieser Teil des  Visualisierungseditors kann mit dem Befehl 'Schnittstelleneditor' , standardmäßig im Menü **Visualisierung**, aus- oder eingeschaltet werden ([Kap. 20.2.2 "Befehl 'Schnittstelleneditor'" auf Seite 813](#)). Er erscheint im oberen Teil des Visualisierungseditors als separate Registerkarte.

Deklarieren einer Schnittstelle

Da eine Visualisierung als Funktionsbaustein behandelt wird, erscheint der Schnittstelleneditor als normaler Deklarationseditor im oberen Teil des Visualisierungsfensters. Hier können Eingabevariablen, also Frameparameter, deklariert werden.

Deklaration der Parameter durch:

- VAR_INPUT
- VAR_INPUT mit [Kap. "Attribut 'parameterstringof'" auf Seite 838](#)
- VAR_IN_OUT: Wird eine Datenstruktur übertragen, muss diese als VAR_IN_OUT definiert werden. Ohne Pragma werden die Werte der Parameter kopiert, wenn die Visualisierung geöffnet wird. Eingaben werden auf die kopierten Datenstrukturen geschrieben und mit Schließen der Visualisierung werden die Werte in die Parameter zurückgeschrieben.
- VAR_IN_OUT mit [Kap. "Attribut 'VAR_IN_OUT_AS_POINTER'" auf Seite 836](#): Hier ist es erlaubt, Objekte als Referenzen an Visualisierungen zu übergeben. Das wird bei Diagnose-Visualisierung, WebVisu und TargetVisu verwendet. Nützlich kann das sein, wenn Information an eine Visualisierung übergeben werden soll, ohne dass diese kopiert wird. Oder wenn Information von außen aktualisiert werden soll, während der Dialog geöffnet ist.

Beispiel:

Deklarieren der Schnittstelle UserData

Die Visualisierung ViewData verwendet die Schnittstelle "UserData" des Typs "IE_Data".

Das Element GenElemInst_5 verwendet "UserData" und zeigt den Variablenwert von UserData.iCounter an, so wie es in der Ansicht Eigenschaften von "GenElemInst_5" programmiert ist.

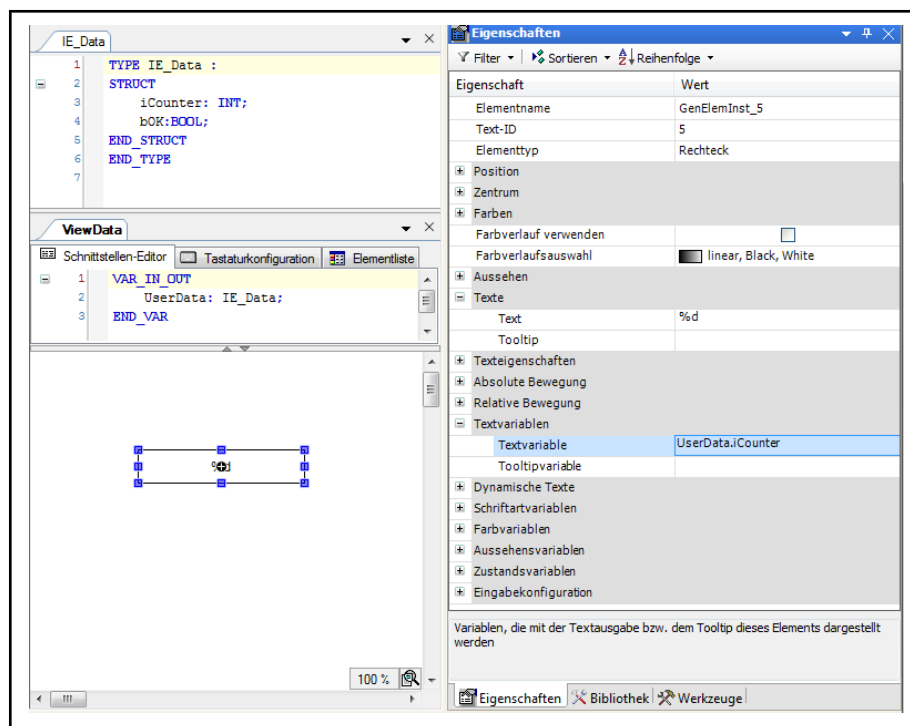


Abb. 20-8: Deklarieren der Schnittstelle

Programmieren eines Frames

1. Fügen Sie ein Frameelement in Ihrer Hauptvisualisierung hinzu.
2. Setzen Sie die Referenz auf eine Visualisierung mit Schnittstelle, die nicht nur lokal, sondern auch global oder von einer Bibliothek sein kann.
3. Weisen sie eine IEC-Variable/Ausdruck jedem Frameparameter zu.

Wenn die Visualisierung mit Schnittstelle in einer anderen Visualisierung referenziert wird, dann können diese Frameparameter programmiert werden, indem lokal verwendete Ausdrücke zugewiesen werden. Diese Ersetzungen müssen in **Eigenschaften** des Frameelements vorgenommen werden. Dort sind die eingefügten (instanzierten) Visualisierungen und ihre Eingabevariablen aufgeführt und den Variablen können Ausdrücke zugewiesen werden, die bei der lokalen Konfiguration der Instanz verwendet werden können.

Beispiel:

Programmieren eines Frames mit Referenz "ViewData"

Das Element 'Frame' referenziert die Visualisierung ViewData ([Kap. "Element 'Frame'" auf Seite 917](#)).

In der Ansicht Eigenschaften wird diese Zuweisung mit allen Frameparametern aufgeführt. Mit Klick in das Wertefeld einer solchen Variable wird eine IEC-Variable des gleichen Typs zugewiesen. Hier wird also Hans.HansData an UserData zugewiesen.

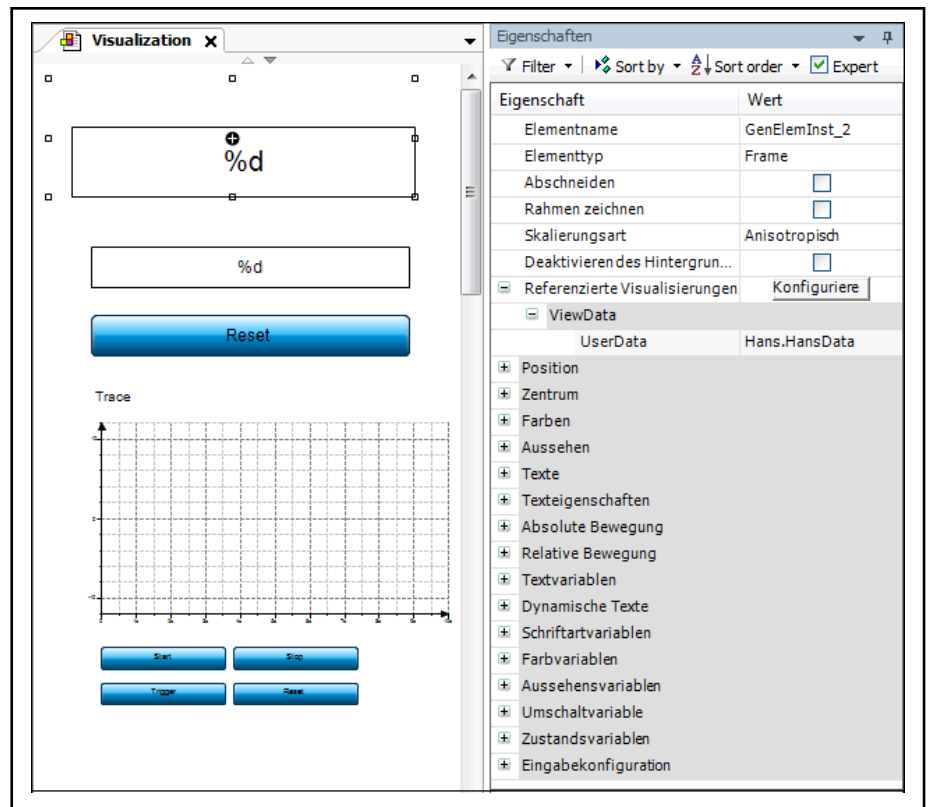


Abb. 20-9: Framereferenz "ViewData"

Frameparameter aktualisieren

Wird die Schnittstelle einer Visualisierung, die im Frame einer anderen Visualisierung referenziert wird, modifiziert, dann öffnet automatisch der Dialog 'Aktualisierung der Frameparameter', wenn Sie versuchen, das Projekt zu speichern, zu kompilieren oder ein betroffenes Objekt zu öffnen (Kap. "Aktualisierung der Frameparameter" auf Seite 833).

In diesem Dialog können Sie die Frameparameter neu konfigurieren.

Aktualisierung der Frameparameter

Wenn die Schnittstelle einer in einem Frame eingefügten Visualisierung geändert wird, wird beim Speichern oder Kompilieren des Projekts oder beim Öffnen eines betroffenen Objekts dieser Dialog automatisch geöffnet. Sie können dann in diesem Dialog die Frameparameter anpassen.

In dem Fall, dass die Schnittstelle einer Visualisierung geändert wurde, die aus einer Bibliothek stammt, müssen die Parameter in Ihrem Projekt auch aktualisiert werden.

Verwenden Sie folgenden Befehl, um den Dialog zur Aktualisierung zu öffnen:

- [Kap. 20.2.16 "Befehl 'Frame-Parameter in <geöffnete Visualisierung> aktualisieren'" auf Seite 822](#)

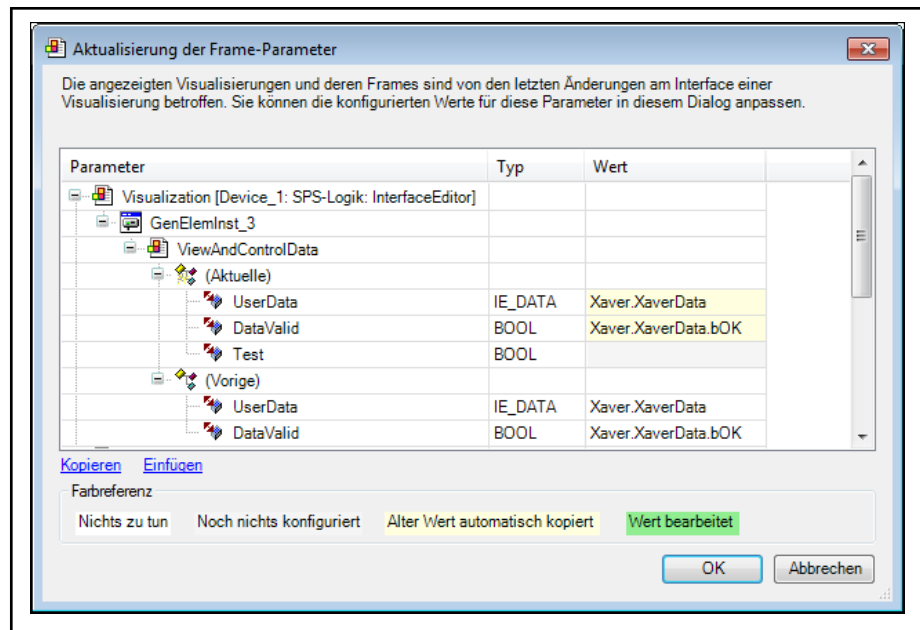


Abb. 20-10: Aktualisierung der Frameparameter

Parameter: in dieser Spalte können Sie die geänderten Parameter in Baumansicht sehen.

- Zuerst befindet sich die Visualisierung, dann der geänderte Frame.
- Dann die referenzierte Visualisierung und darunter deren Schnittstelle.
- Unter **(Aktuelle)** sind die aktuell gültigen Parameter aufgelistet.
- Unter **(Vorige)** ist die davor verwendete Deklaration zu sehen.

Typ: in dieser Spalte wird der Typ des jeweiligen Parameters angezeigt.

Wert: das Wertefeld enthält die Variablen, die dem Parameter zugewiesen wurde. Beachten Sie die Färbung des Feldes:

Farbreferenz

- Beige hinterlegt: Dieser Parameter wurde automatisch aus der bisherigen Konfiguration übernommen
- Grau hinterlegt: Diesem neuen Parameter ist noch kein Wert zugewiesen
- Grün hinterlegt: Diesem neuen Parameter ist bereits ein Wert zugewiesen.
- Weiß hinterlegt: Hier muss nichts konfiguriert werden.

Werte bearbeiten, die sich unter **(Aktuelle)** befinden:

Klicken Sie in das Wertefeld, um es zu selektieren. Nach einem weiteren Mausklick in das Feld oder nach Drücken der <Leertaste>, öffnet ein Zeileneditor, oder Sie fangen einfach an zu tippen.

Weisen Sie eine andere Variable zu, indem Sie deren Namen eintippen oder selektieren Sie eine Variable mit Hilfe des Eingabehilfsfeldes.

Ein vorhandener Werteintrag kann in ein anderes Feld kopiert werden. Selektieren Sie dafür den Eintrag, der kopiert werden soll, verwenden Sie <Kopieren> und selektieren Sie das Feld, in das der Eintrag eingefügt werden soll und verwenden Sie <Einfügen>.

Bestätigen Sie die Einstellungen im Dialog mit <OK>.

Der Dialog schließt, und die neuen Parameterzuweisungen werden im Dialog **Eigenschaften**, Kategorie **Referenzen**, beim zugehörigen Frameelement angezeigt.



Falls Sie innerhalb des referenzierten Visualisierungsbausteins vom Instanznamen einer übergebenen Variablen Gebrauch machen möchten, müssen Sie mit dem Pragma [Kap. "Attribut 'parameterstringof'" auf Seite 838](#) arbeiten, welcher einen entsprechenden String zu Verfügung stellt.

Parameter DataValid hinzufügen

Erweitern Sie die Schnittstelle der Visualisierung ViewAndControlData um die Variable **DataValid**.

Schnittstelleneditor

```
VAR_IN_OUT
  UserData: IE_Data;
  DataValid: BOOL;
END_VAR
```

Versuchen sie nun das Projekt zu kompilieren oder zu speichern, oder Visualization zu öffnen, erscheint zuerst der Dialog für die Konfiguration des neuen Parameters.

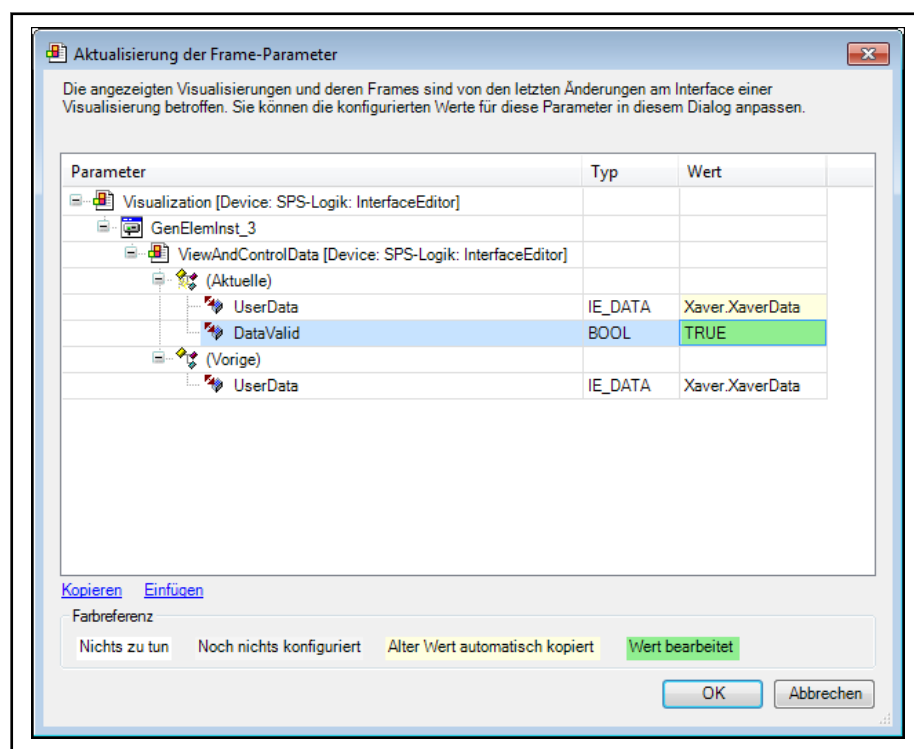


Abb. 20-11: Aktualisieren der Frameparameter

Vergleichen Sie die aktuellen mit den vorigen Parametern. Geben Sie in das Wertefeld von DataValid "TRUE" ein. Schließen sie den Dialog mit <OK>. Die neuen Parameter können in der Ansicht **Eigenschaften** in der Kategorie **Referenzen** von GenElemInst_3 nachvollzogen werden.

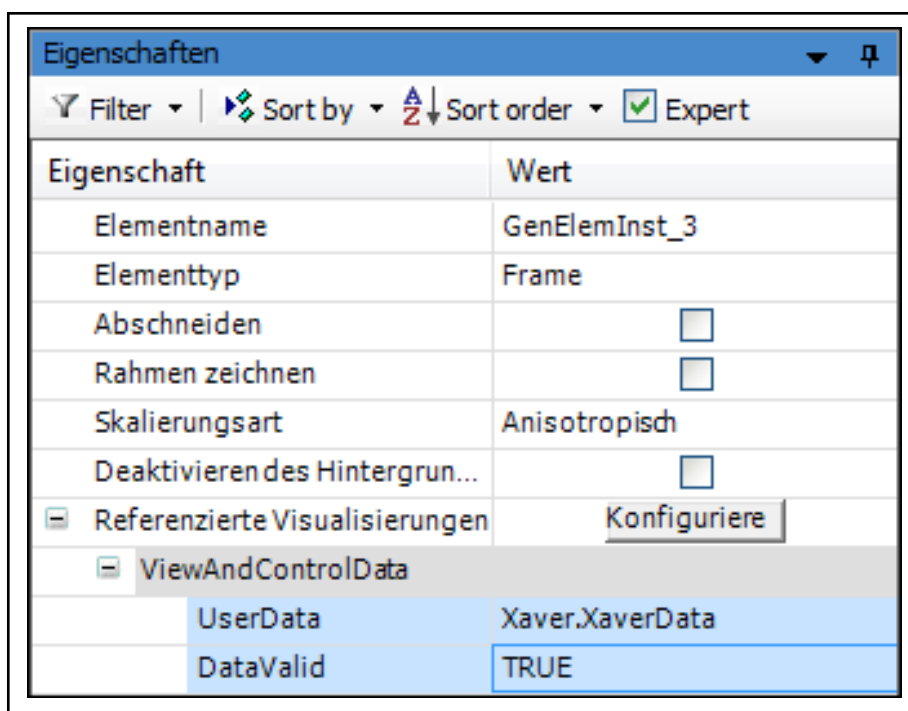


Abb. 20-12: Ansicht Eigenschaften, nachdem die Aktualisierung der Frameparameter geschlossen wurde

Pragmas

Pragmas, die im Schnittstelleneditor verwendet werden können:

Attribut 'VAR_IN_OUT_AS_POINTER'

Dieses Pragma erlaubt es, die Reference auf ein Objekt als Frame-Parameter zu verwenden. Das ist in einer Visualisierung, die als Dialog deklariert ist, möglich.

Syntax:

```
{attribute 'VAR_IN_OUT_AS_POINTER'}
```

Werden Referenzen verwendet, ist auf folgende Weise vorzugehen:

1. Deklarieren Sie ihr Benutzerdatenobjekt (DUT).
2. Deklarieren Sie einen Frame-Parameter als VAR_IN_OUT im Schnittstelleneditor eines Dialogs als eine Referenz auf das Datenobjekt.
Geben Sie diesem Frame-Parameter das Attribut 'VAR_IN_OUT_AS_POINTER'.
3. Programmieren Sie die Benutzerschnittstelle:
Verwenden Sie den Dialog in einer Visualisierung oder nutzen Sie den Dialog in der Eingabekonfiguration eines Visualisierungselements. In beiden Fällen ist ein Zugriff auf die referenzierten Daten möglich.

Verwendung einer Schnittstelle mit Pragma 'VAR_IN_OUT_AS_POINTER'

```
FUNCTION_BLOCK ControlFB
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    bOk : BOOL := TRUE;
    nCounter : INT;
```

```
nValue : INT;  
END_VAR  
nCounter := nCounter + 1;
```

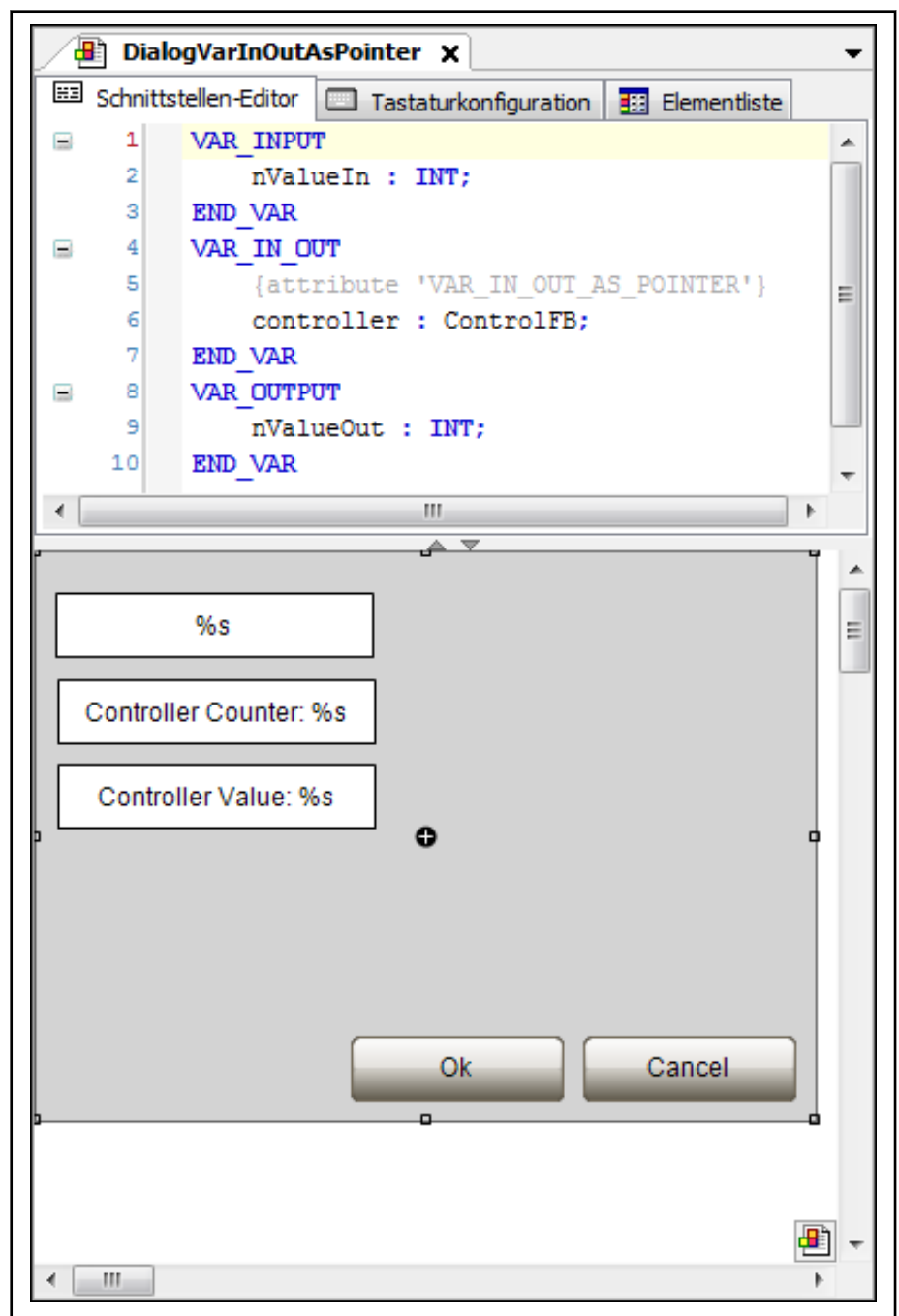


Abb. 20-13: Deklaration eines Frameparameters mit VAR_IN_OUT_AS_POINTER

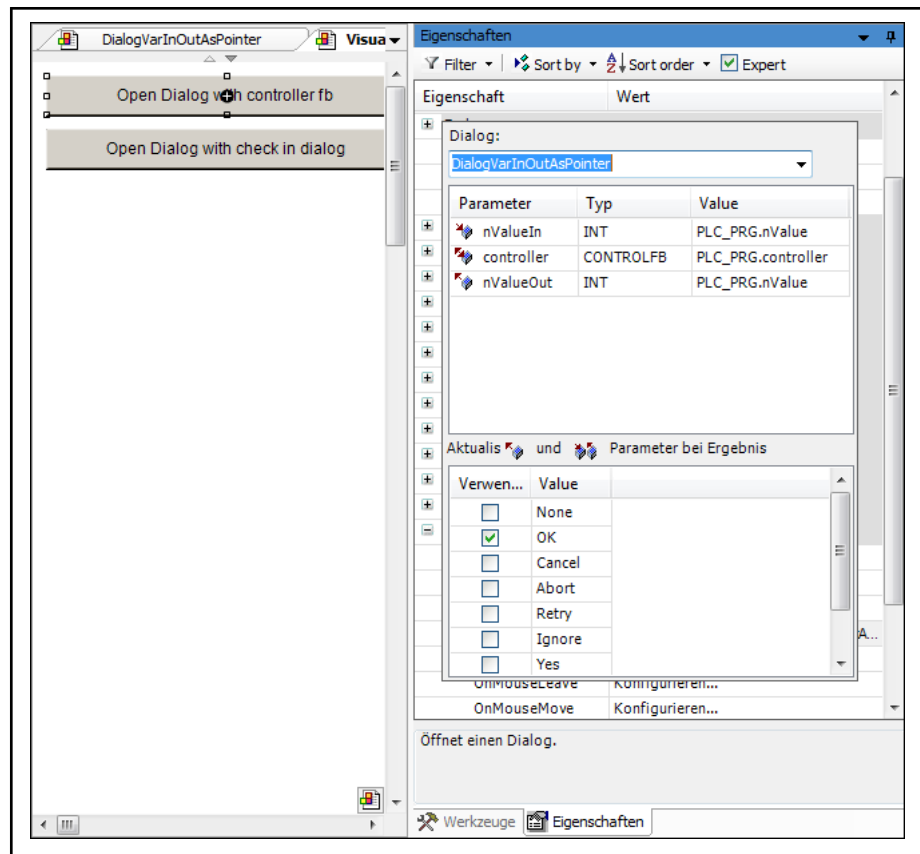


Abb. 20-14: Benutzerschnittstelle: Dialog öffnet

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 PlcProg
VAR
    myDUT: DUT;
END_VAR
```

Diese Instanz sei Eingabe eines Visualisierungsbausteins "Vis" (im Eingabeparameter "instance") welcher seinerseits von einem Frame einer anderen Visualisierung "MainVisu" referenziert wird:

Ausschnitt aus Eigenschaften eines Visualisierungselements von MainVisu



Abb. 20-15:

In dem zu "Vis" gehörenden Schnittstelleneditor finden wir natürlich die Input/Output Variable `instance`, aber auch eine weitere Input Variable genannt `instanceStr`:

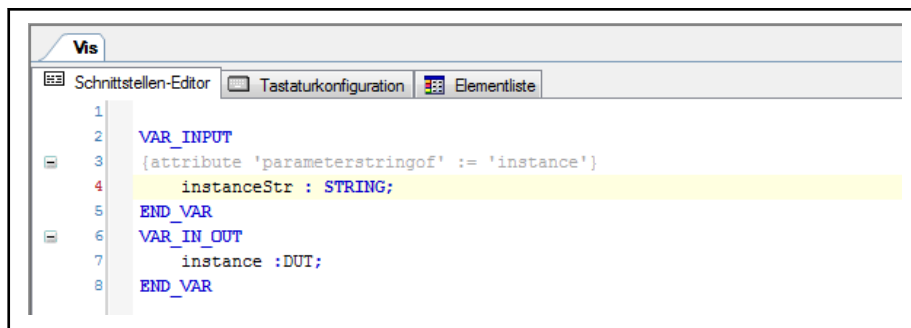


Abb. 20-16:

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-Variablen `instanceStr` wird also auf `PlcProg.myDUT` gesetzt und kann nun innerhalb der Visualisierung `Vis` verwendet werden, beispielsweise als Textvariable für einen Platzhalter `%s`.

20.3.4 Registerkarte 'Tastaturkonfiguration'

Die Tastaturkonfiguration, zu öffnen über Befehl 'Tastaturbedienung', ist im oberen Teil des Visualisierungseditors zu finden.

Siehe auch:

- [Kap. 20.2.3 "Befehl 'Tastaturbedienung'" auf Seite 813](#)
- [Kap. "Grundlegendes zum Schnittstelleneditor" auf Seite 831](#)
- [Kap. 20.3.5 "Registerkarte 'Elementliste'" auf Seite 841](#)

Grundlegendes

Zusätzlich zur Tastaturbedienung im Online-Modus können in diesem Teil des Visualisierungseditors spezielle Tastenkürzel für die Bedienung einer Visualisierung im Online-Betrieb definiert werden ([Kap. 20.11 "Tastaturbedienung im Online-Modus" auf Seite 1081](#)). Das heißt, eine Aktion kann einer bestimmten Taste(in Kombination) zugeordnet werden. Diese Tastenkonfiguration gilt genau für die vorliegende Visualisierung.

Tastenkfigurationen, die in allen Visualisierungen der Applikation verwendbar sein sollen, sollten im Visualisierungsmanager definiert werden ([Kap. "Visualisierungsmanager- 'Standardtastaturkonfiguration'" auf Seite 1060](#)).



In der **Gerätebeschreibung** ist definiert, welche Tasten in einer Visualisierung, die auf dem Gerät läuft, unterstützt werden!

Zu beachten ist die folgende Aufrufreihenfolge beim Abarbeiten von Tastenaktionen:

1. Eventhandler der Applikation, falls aktiviert (optional); z.B. [Kap. "Ereignisse und Eingabeaktionen bemerken" auf Seite 1074](#))
2. Tastaturkonfiguration im 'Visualisierungsmanager' - 'Standardtastaturkonfiguration', die für alle Visualisierungen der Applikation gilt ([Kap. "Visualisierungsmanager- 'Standardtastaturkonfiguration'" auf Seite 1060](#)).
3. Definitionen der Tastaturbedienung im Online-Modus ([Kap. 20.11 "Tastaturbedienung im Online-Modus" auf Seite 1081](#))
4. Spezielle Tastaturkonfigurationen der einzelnen Visualisierungen (verwaltet im Tastaturkonfiguration), wobei die Hauptvisualisierungen vor den in den Frames liegenden beachtet werden.

In diesem Zusammenhang ist auch die Eigenschaft **Tasten** eines Visualisierungselements zu beachten, die im Editor der Elementeigenschaften unter **Eingabekonfiguration** konfiguriert werden kann. Eine solche elementspezifische Tastenkfiguration wird ebenfalls in **Tastaturkonfiguration** verwaltet. Sie kann an beiden Stellen bearbeitet werden, wobei sie jeweils auch im anderen Editor aktualisiert wird.

Editor

Taste	Taste drücken	Umschalttaste	Ctrl	Alt	Aktionstyp	Aktion	Element ID
M	☒	☐	☐	☐	Maus-Ereignis ausführen		#3
E	☐	☒	☒	☐	ST-Code ausführen	prog();	
NUM9	☒	☒	☒	☒	Maus-Ereignis ausführen		#0

Abb. 20-17: Editor Tastaturkonfiguration

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.

Umschalt, 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: Er kann über eine Auswahlliste definiert werden, die bei einem Doppelklick auf die Zelle öffnet. Die Aktionstypen entsprechen denen, die in

Eigenschaften Kategorie **Eingabekonfiguration** eines Visualisierungselements verfügbar sind.

Aktion: Genaue Konfiguration der auszuführenden Aktion; hängt vom Aktionstyp ab und entspricht der Mausektion, wie sie in der **Eingabekonfiguration** eines Elements verwendet werden kann.

Element ID: ID des Visualisierungselements, dem die Taste zugeordnet ist (über die "Hotkey" **Eingabekonfiguration** ► **Tasten**). Eindeutige Kennung innerhalb der aktuellen Visualisierung.

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.

20.3.5 Registerkarte 'Elementliste'

Dieser Teil des Visualisierungseditors stellt eine Liste der Visualisierungselemente der aktuell geöffneten Visualisierung zur Verfügung.

Die Elementliste kann über den Befehl 'Elementliste' (standardmäßig im Menü **Visualisierung**) geöffnet werden und steht dann als separate Registerkarte neben anderen Editoren wie z.B. dem Schnittstelleneditor oder dem Editor für die Tastaturkonfiguration zur Verfügung (Kap. 20.2.4 "Befehl 'Elementliste'" auf Seite 813).

Typ	X	Y	Breite	Höhe	ID	Name	Zugriffsrechte
 #0 Schaltfläche	50	60	150	30	0	Login	
 #1 Schaltfläche	53	112	150	30	1	Logout	
 #2 Schaltfläche	51	166	150	30	3	Change PW	
 #3 Schaltfläche	50	240	150	30	5	Open User Config	
 #4 Rechteck	378	64	150	30	6	GenElemInst_7	
 #5 Rechteck	586	65	150	30	10	GenElemInst_11	
 #6 Abgerundetes Rechteck	402	145	221	52	7	GenElemInst_8	
 #7 Ellipse	413	245	167	30	8	GenElemInst_9	
 #8 Schaltfläche	50	340	150	30	9	GenElemInst_17	
 #9 Schaltfläche	50	390	150	30	13	GenElemInst_19	
 #10 Rechteck	413	352	262	30	16	GenElemInst_21	

Abb. 20-18: 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 Ansicht 'Werkzeuge' verwendet, sowie die Elementnummer, die sich zunächst aus der Reihenfolge des Einfügens ergibt (Kap. 20.3.1 "Ansicht 'Werkzeuge'" auf Seite 826).

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 **Eigenschaften** des Elements (**Elementname**) definiert

Zugriffsrechte: Ist das Verhalten eines Elements für einige Benutzergruppen eingeschränkt, dann wird dies mit dem Vorhängeschlosssymbol  gekennzeichnet.

Element(e) können durch Selektieren der entsprechenden Tabellenzeile(n) ausgewählt werden, wobei die Auswahl immer mit der im Hauptfenster des

Visualisierungseitors synchronisiert wird. Zu beachten ist, 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:

- [Kap. "Befehl 'Nach vorn legen'" auf Seite 818](#)
- [Kap. "Befehl 'Um Eins nach vorn legen'" auf Seite 818](#)
- [Kap. "Befehl 'Nach hinten legen'" auf Seite 819](#)
- [Kap. "Befehl 'Um Eins nach hinten legen'" auf Seite 818](#)

Die folgenden Befehle können ebenfalls verwendet werden:

- **Rückgängig**
- **Wiederherstellen**
- **Löschen**

20.3.6 Dialog 'Visualisierung hinzufügen'

Funktion: Der Dialog dient der Erstellung eines neuen Objekts des Typs **Visualisierung**.

Aufruf: Applikation im Projektexplorer selektieren, **Hinzufügen** auswählen, Typ **Visualisierung** auswählen

Name	Name der Visualisierung zum Beispiel Visu_A
Voraussetzung für die folgenden Einstellungen: Sie erscheinen nur, wenn Sie in einem Projekt zum ersten Mal ein Objekt des Typs Visualisierung hinzufügen.	
Symbolbibliothek	Liste aller installierten Symbolbibliotheken
Zugewiesen	 Symbolbibliothek ist ausgewählt. Wenn Sie den Dialog mit Hinzufügen beenden, weist IndraLogic die Symbolbibliothek dem Projekt zu und stellt sie in der Ansicht Werkzeuge zur Verfügung. IndraLogic verwaltet diese Einstellung in den Projekteinstellungen.

Siehe auch

- [Kap. 20.1.2 " Verfügbare Visualisierungselemente anzeigen und kennzeichnen" auf Seite 806](#)
- [Kap. "Befehl 'Hinzufügen'" auf Seite 643](#)

20.3.7 Dialog 'Kennzeichen und Einträge konfigurieren'

Funktion: Der Dialog stellt die bestehenden Kennzeichen in einer Bauman-sicht dar.

Unterhalb eines Kennzeichens befinden sich die Elemente, die dem Kennzei-chen zugeordnet sind. Sie können weitere Kennzeichen hinzufügen.

Aufruf: Klick auf Symbol  in Ansicht **Werkzeuge**

Siehe auch

- [Kap. 20.1.2 " Verfügbare Visualisierungselemente anzeigen und kennzeichnen" auf Seite 806](#)
- [Kap. 20.3.1 "Ansicht 'Werkzeuge'" auf Seite 826](#)

Bauman-sicht

Kennzeichen	Baumansicht aller Kennzeichen und deren Einträge  <Name>: Von IndraLogic bereitgestelltes Kennzeichen  <Name>: Ein von einem Benutzer erstelltes Kennzeichen. Voreinstellung: Favorit
	IndraLogic expandiert den Baum. Die Einträge, die dem Kennzeichen zugeordnet sind, werden sichtbar.
Aktiv	 : In der Ansicht Werkzeuge ist eine Schaltfläche für das Kennzeichen sichtbar. IndraLogic beschriftet die Schaltfläche mit <Name> .

Symbolleiste

	Dialog Kennzeichen hinzufügen erscheint.
 oder <Entf>	IndraLogic entfernt das Kennzeichen aus der Baumansicht. Wenn Sie den Dialog mit OK beenden, entfernt IndraLogic auch die Schaltfläche in der Ansicht Werkzeuge . Voraussetzung: Ein Kennzeichen ist selektiert

Tab. 20-1:

Dialog 'Kennzeichen hinzufügen'

Aufruf: Klick auf Symbol  im Dialog **Kennzeichen und Einträge konfigurieren**

Name	Name des Kennzeichens, zum Beispiel tagA
Beschreibung	Beschreibung des Kennzeichens

20.4 Visualisierungselemente

20.4.1 Elemente konfigurieren - allgemein

Benutzereingabe beim Editieren eines Elements

Im Visualisierungseditor können verschiedene Benutzereingaben (Mausbefehle oder Menübefehle) abhängig von der Position des Cursors im Editor und dem Zustand des Elements vorgenommen werden. Erlaubt die Cursorposition Benutzereingaben, dann ändert er seine Form um das dem Benutzer zu signalisieren.

Ein Element kann im Zustand "nicht selektiert" oder "selektiert" sein. Ein selektiertes Element wird visualisiert mit zusätzlichen kleinen Quadraten, die seine Position und Größe repräsentieren. Verwendete Befehle wirken auf dieses Element.

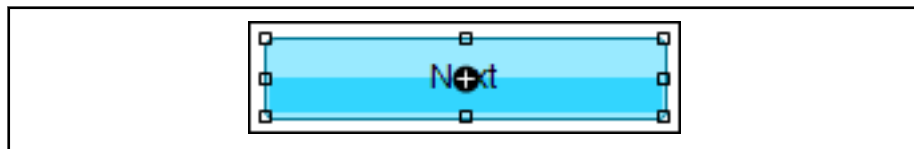


Abb. 20-19: Selektierter Button

Befehle

Abhängig vom Element und der Selektion im Editor stehen verschiedene Befehle zur Verfügung, die nicht ausgegraut sind (Kap. 20.2 "Befehle" auf Seite 812).

Editieren eines Visualisierungselements

Aktion im Editor	Auswirkung auf Eigenschaften
Ist ein Element fokussiert und ist der Cursor ein Handsymbol, wird es mit Klick auf das Element ausgewählt. Dann sind die Positions- und Größenrechtecke des Elements auf der Umrisslinie sichtbar.	Die Eigenschaften der aktuell selektierten Elements wird sofort angezeigt und kann in Eigenschaften editiert werden.
Ist ein Element fokussiert und wird der Cursor  , dann kann mit einem Klick auf das Element der Zeileneditor geöffnet und dort ein Name/Text eingegeben werden. Mit <Eingabe> wird der Zeileneditor geschlossen.	Text unter Kategorie Texte enthält den eingegebenen Text.
Mit Fokus auf das Zentrum eines selektierten Elements ändert sich die Cursorform in  und das Zentrum kann mit gedrückter Maustaste auf eine beliebige X-/Y-Position gezogen werden.	X und Y unter Kategorie Zentrum werden aktualisiert.
Mit Fokus auf das selektierte Element ändert sich die Cursorform in  und das Element kann mit gedrückter Maustaste auf eine beliebige X-/Y-Position gezogen werden.	X und Y unter Kategorie Position werden aktualisiert.
Ist das Element selektiert und der Fokus auf eines der kleinen Quadrate auf der Umrisslinie, dann ändert sich die Cursorform und das Quadrat kann auf eine neue Größe gezogen werden. Wenn die Größe mit der Maus geändert wird, wird das Zentrum zurückgesetzt. Wenn zusätzlich die <Umschalttaste>-Taste gedrückt wird, wird die Breite und Höhe proportional zueinander geändert. Wenn zusätzlich die <Strg>-Taste gedrückt wird, wird die Breite und Höhe symmetrisch zum Zentrum geändert. Wenn zusätzlich die <Umschalttaste>-Taste und die <Strg>-Taste gedrückt wird, wird die Größenänderung proportional und symmetrisch zum Zentrum ausgeführt.	Abhängig vom Elementtyp wird dessen Position und Größe auf unterschiedliche Weise unter Kategorie Position gespeichert: • X, Y • X, Y, Breite, Höhe • Kategorie Punkte [0] .. [<n>]: X, Y

Mehrfachauswahl *Mehrfachauswahl ist, wenn*

- <Umschalttaste> gedrückt wird während Elemente per Mausklick ausgewählt oder abgewählt werden.
- ein Rechteck mit gedrückter Maustaste um mehrere Elemente gezogen wird (Drag-and-drop).
- [Kap. 20.2.12 "Befehl 'Alles selektieren'"](#) auf Seite 821 verwendet wird.

Deselektion ist möglich mit [Kap. 20.2.13 "Befehl 'Alles deselektieren'"](#) auf Seite 822.

Alternativ können Elemente in der Registerkarte 'Elementliste' mit Hilfe von <Umschalttaste> und <Strg> selektiert werden ([Kap. 20.3.5 "Registerkarte 'Elementliste'"](#) auf Seite 841).



Sind mehrere Elemente selektiert, dann wirken Änderungen in **Eigenschaften** auf alle selektierten Elemente.

Verlagern der Selektion mit Tastatur

Ist ein Element selektiert, kann die Selektion zum nächsten Element gemäß der Reihenfolge beim Einfügen (Z-Reihenfolge) verlagert werden, indem

	<Tabulator> gedrückt wird. Wird <Umschalttaste> + <Tabulator> gedrückt, wird die Selektion in umgekehrter Reihenfolge verlagert.
Elemente gruppieren	Mehrfach ausgewählte Elemente können mit dem Befehl 'Gruppieren' gruppiert werden (Kap. 20.2.8 "Befehl 'Gruppieren'" auf Seite 819). Die selektierten Elemente sind in einer Gruppe festgesetzt, deren Unterelemente dann über Registerkarte 'Elementliste' selektiert werden können (Kap. 20.3.5 "Registerkarte 'Elementliste'" auf Seite 841). Mit 'Gruppierung aufheben' wird die Gruppe wieder in seine Elemente zerlegt (Kap. 20.2.9 "Befehl 'Gruppierung aufheben'" auf Seite 819).
Ändern der Reihenfolge	Um festzulegen, wo die Elemente auf der Z-Achse des Visualisierungsbereichs, das ist im Vordergrund, im Hintergrund oder irgendwo dazwischen, positioniert sein sollen, stehen Befehle unter VI-Logic Visualisierung ► Folge zur Verfügung.
Ausrichtung der Elemente	Um zwei oder mehr Elemente auszurichten, stehen Befehle unter VI-Logic Visualisierung ► Ausrichtung zur Verfügung.

Text und Sprache verwenden

Text kann einem Visualisierungselement durch eine geeignete Definition im Eigenschafteneditor oder durch direkte Eingabe im Visualisierungseditor zugewiesen werden. Für letzteres kann am selektierten Element mit der <Leertaste> ein Eingabefeld geöffnet werden.

Siehe auch

- [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Textlisten 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 Sprachversionen (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.

Siehe auch

- [Kap. 18.2.15 "Menü 'Textliste'" auf Seite 727](#)



Verzeichnisse, die Textlisten für die Verwendung in der Visualisierung bereitstellen, sind unter **Extras ► Optionen ► IndraLogic 2G ► Visualisierung**, Registerblatt "Dateioptionen" definiert.

Textarten In den Texteigenschaften eines Visualisierungselements müssen der Textlistenname 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 Formatierung beeinflusst werden ("[Textformatierung](#)" auf Seite 846).

Da die Tex-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 Eingabekonfiguration versehen sein, die eine Mausektion zur "Sprachumschaltung" definiert. ([Kap. "Eingabekonfiguration" auf Seite 849](#)).

Somit sind eine dynamische Anzeige und Sprachumschaltung für Visualisierungstexte möglich.

Grundsätzlich gibt es zwei Typen von Text in einer Visualisierung:

- **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 der Eigenschaft **Texte** konfiguriert und wird in einer automatisch angelegten Textliste 'GlobalTextList' verwaltet.

- **Dynamische Texte**

Dynamischer Text kann im Online Modus durch eine Benutzereingabe oder über eine Variable der Applikation verändert werden. Er wird in der Eigenschaft **Dynamische Texte** konfiguriert. Dabei muss der Name der dynamischen Textliste und die Text ID des gewünschten Texts angegeben werden.

Textformatierung

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 Sie ein %s in den Text-String einfügen, dann wird an dieser Stelle im Online-Modus der Text eingefügt, der in der Variable gespeichert ist, die unter der Eigenschaft **Textvariablen** zugewiesen wurde.

Falls Sie den Instanznamen einer Variablen, die einem Visualisierungsbau-stein als Eingangsparameter übergeben wurde, anzeigen lassen wollen, so ist vom dem Pragmaattribut 'parameterstringof' Gebrauch zu machen ([Kap. "Attribut 'parameterstringof'" auf Seite 838](#)).



Sie können jede Formatierungsangabe verwenden, die der Standard-C-Bibliotheksfunktion "sprintf" entspricht, wenn sie zum Datentyp der verwendeten Variable passt.

Zeichen nach "%"	Argument/Ausgabe als
d, i	Dezimalzahl
b	Binärzahl
o	Vorzeichenlose Oktalzahl (ohne vorangestellte Null)
x	Vorzeichenlose Hexadezimalzahl (ohne vorangestelltes "0x")
u	Vorzeichenlose Dezimalzahl
c	Einzelnes Zeichen

Zeichen nach "%"	Argument/Ausgabe als
s	Zeichenfolge: Dieses wird im Online-Betrieb durch den Wert der Variablen ersetzt, die im Eigenschaften-Feld Textvariablen ► Textvariable angegeben ist.
f	REAL-Wert Syntax: % [<Ausrichtung> <Mindestbreite> . <Genauigkeit>] f • <Ausrichtung>: fügen sie - + ein; Minuszeichen bedeutet eine Ausrichtung linksbündig; Pluszeichen bedeutet rechtsbündig (Default). • <Mindestbreite>: Anzahl der Stellen vor dem Komma • <Genauigkeit>: Anzahl der Stellen hinter dem Komma (Default: 6)

Tab. 20-2: Beispiele von Formatierungsangaben und ihren Argumenten:

Beispiel:

Geben Sie in Eigenschaft **Texte** ► **Text** ein: `Fill level %2.5f mm`

Geben Sie in Eigenschaft **Textvariablen** ► **Textvariable** ein: `PlcProg.fvar1`; Variable von Typ REAL, gilt den Füllstandwert an.

->Ausgabe im Online-Betrieb:

`Fill level 32.89999 mm`



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).

Die Schriftart, die im Online-Betrieb verwendet wird, kann auch im Eigenschafteneditor verwendet werden. Siehe dazu die Eigenschaften **Texteigenschaften** und **Fontvariablen**.

Die horizontale und vertikale Ausrichtung eines Textes ist im Eigenschafteneditor der Elemente definiert.

**Unterstützung des
Unicode-Formats**

Unicode kann aktiviert werden.

Siehe auch

- [Kap. 18.2.15 "Menü 'Textliste'" auf Seite 727](#)

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 Tabelle.



Import von IndraLogic 1x-Projekten:

Die in IndraLogic 1x verwendete Zeitformatierung %t wird automatisch in das neue %t[] Format konvertiert, wenn ein altes Projekt importiert wird.

Folgende Platzhalter werden nicht mehr unterstützt: %U, %W, %Z, %Z.

IndraLogic 2G-Visualisierung

ddd	Name des Wochentags, abgekürzt, z.B. "Wed"
dddd	Name des Wochentags, z.B. "Wednesday"
dddddd	Wochentag als Zahl von 1 = Montag bis 7 = Sonntag
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. "4" 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	Sekunden (00-59), mit vorangestellter Null, z.B. "06"
ms	Millisekunden(0-999), ohne vorangestellte Null, zum Beispiel "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

Tab. 20-3: Gültige Platzhalter

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

Ausgabe einer Variablen des Datentyps TIME, LTIME, DATE, TOD oder DT

In der Visualisierung können Sie auch Variablen vom Typ `TIME`, `LTIME`, `DATE`, `TOD` oder `DT` ausgeben. Dazu müssen Sie in der Eigenschaft **Textvariable** eine Variable entsprechenden Typs auswählen. Die Ausgabe erfolgt über die Eigenschaft **Text**. Dabei geben Sie `%t` und einen Platzhalter in eckigen Klammern an.

d oder dd	Tage
HH	Stunde, 24-Stunden-Format (01-23), z.B. "16"

hh	Stunde, 12-Stunden-Format (01-11), z.B. "4" für 16 Uhr
m oder mm	Minuten (0-59), ohne vorangestellte Null, z.B. "6"
s oder ss	Sekunden (0-59), ohne vorangestellte Null, z.B. "6"
ms	Millisekunden (0-999), ohne vorangestellte Null, zum Beispiel "22"
us	Mikrosekunden (0-999), ohne vorangestellte Null, zum Beispiel "22" - nur bei LTIME
ns	Nanosekunden (0-999), ohne vorangestellte Null, zum Beispiel "22" - nur bei LTIME
' '	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

Tab. 20-4: Gültige Platzhalter

Beispiel:

```
Deklaration      in      PlcProg:      LTIME1      :      LTIME      :=
LTIME#1000D23H44M12S34MS2US44NS
```

Eigenschaft **Textvariable**: PlcProg.LTIME1Eigenschaft **Text**: Time1: %t[dd.HH.m.s.ms.us.ns]

→ Anzeige im Online-Modus:

Time1: 1000.23.44.12.34.2.44

Eingabekonfiguration

Eine oder mehrere der im Folgenden beschriebenen Nachfolgeaktionen kann im Dialog **Eingabekonfiguration** für ein bestimmtes Ereignis (z. B. OnMouseClicked) des Elements definiert werden. Der Dialog öffnet auf einen Mausklick in das Wertefeld dieses Ereignisses im **Eigenschaften**-Fenster. Der Name des Ereignisses, für die diese Nachfolgeaktionen selektiert werden, ist als Titel in der grauen Zeile sichtbar.

Dialog schließen

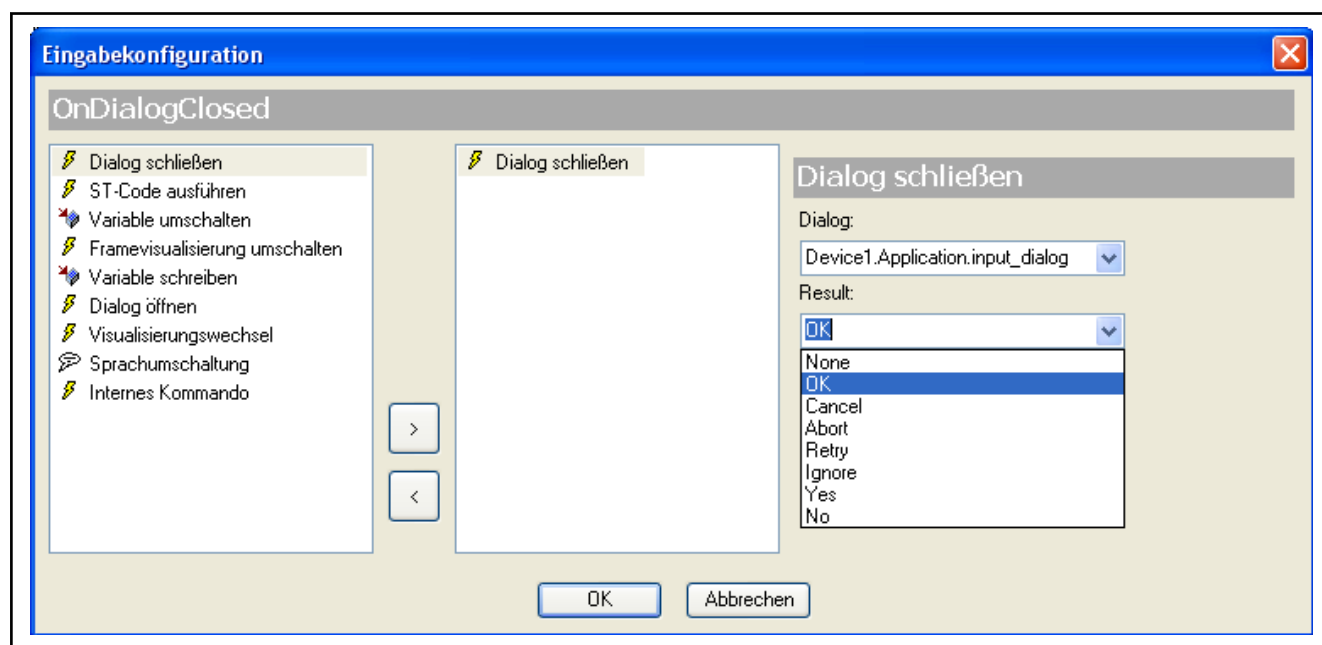


Abb. 20-20: Dialog 'Eingabekonfiguration', 'Dialog schließen' selektiert

Hier kann festgelegt werden, dass auf das Ereignis hin der angegebene Dialog mit dem angegebenen Resultat geschlossen wird. Wählen Sie den gewünschten Dialog von der Auswahlliste, die alle momentan verfügbaren [Kap. "Eingabedialoge" auf Seite 1064](#) anbietet.

Dialog: Auswählen eines Dialogs aus der Auswahlliste.

Ergebnis: Die Liste bietet die Standardoptionen, die in Dialogen verwendet werden, an, 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**.

Siehe [Kap. "Eingabedialoge" auf Seite 1064](#) für mehr Information zu den standardmäßigen und benutzerdefinierten Eingabedialogen in Visualisierungen.

Dialog öffnen

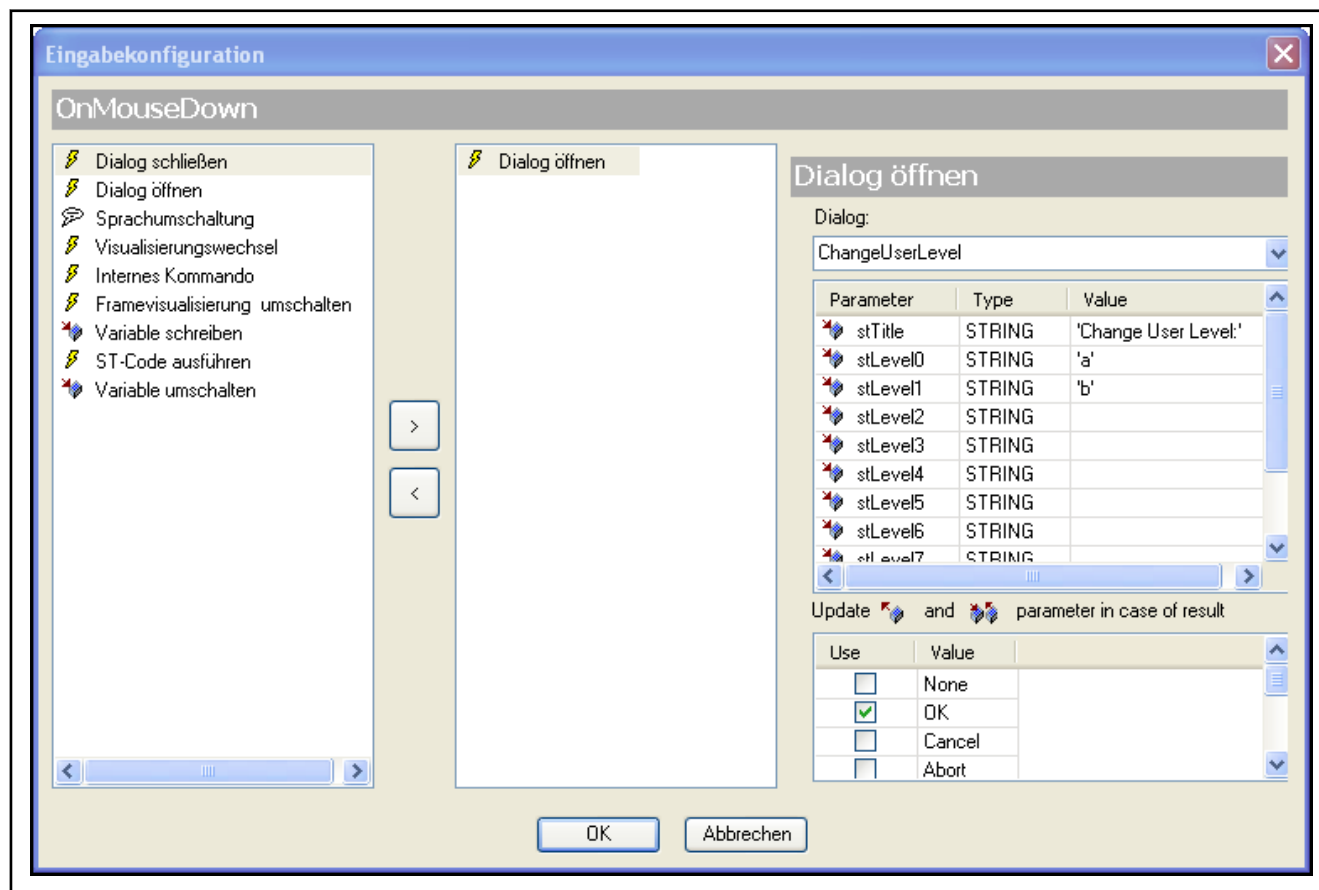


Abb. 20-21: Dialog 'Eingabekonfiguration', 'Dialog öffnen' selektiert

Hier können Sie definieren, dass auf die Mausaktion hin ein Dialog geöffnet werden soll, der durch eine an andere (standardmäßig oder vom Benutzer erstellte) Visualisierung repräsentiert wird. Die Auswahlliste bietet alle Visualisierungen an, die in ihren Dialog 'Eigenschaften' eines Visualisierungsobjekts den Verwendungszweck "Dialog" eingetragen haben ([Kap. 20.6 "Dialog 'Eigenschaften- Visualisierungsobjekt'" auf Seite 1063](#)). Somit kann ein selbst-

erstellter Dialog als Benutzereingabemaske in einer Visualisierung verwendet werden.

Zumindest das Standardobjekt "VisuDialogs.Login" und "VisuDialogs.FileOpenSave" ist verfügbar, wenn **VisuDialogs.library** im Projekt eingebunden ist.

Für die ausgewählte Dialogvisualisierung werden die Eingangs- (VAR_INPUT) und Ein-/Ausgangsparameter (VAR_IN_OUT) wie in Grundlegendes zum Schnittstelleneditor definiert dargestellt (**Parameter, Typ, Wert**) ([Kap. "Grundlegendes zum Schnittstelleneditor" auf Seite 831](#)).

Die Standardwerte (=0) können durch visualisierungsspezifische Werte ersetzt werden, die immer dann in die Dialogvisualisierung geschrieben werden, wenn der Dialog geöffnet wird.

Außerdem muss angegeben werden, auf welches **Ergebnis** der Dialogvisualisierung hin ihre Ausgangs- und Ein-/Ausgangsparameter geschrieben werden sollen: Setzen Sie einen Haken in der **Verwenden**-Spalte des gewünschten Ergebnis**Werts**. Mögliche Werte: Keinen (None), OK, Abbrechen (Cancel), Komplette abbrechen (Abort), Erneut versuchen (Retry), Ignorieren (Ignore), Ja (Yes), Nein (No).



Im Unterschied zu früheren Versionen der Visualisierung ist es für den Standardgebrauch einer Dialogvisualisierung nicht nötig, eine **OnDialogClosed**-Aktion zu definieren, um einen Dialog geschlossen zu bekommen. Jedoch steht die Aktion weiterhin als Möglichkeit für spezielle Lösungen zur Verfügung.



Man beachte, dass die Ausgangs- und Ein-/Ausgangsparameter einer Dialog-Visualisierung nicht geschrieben werden, bevor der Dialog geschlossen wird! Bis dahin werden die Werte nur auf dem Stack gespeichert, dies bedeutet, dass werden nicht als Referenzen sondern als Kopien gehandhabt.

Sprachumschaltung

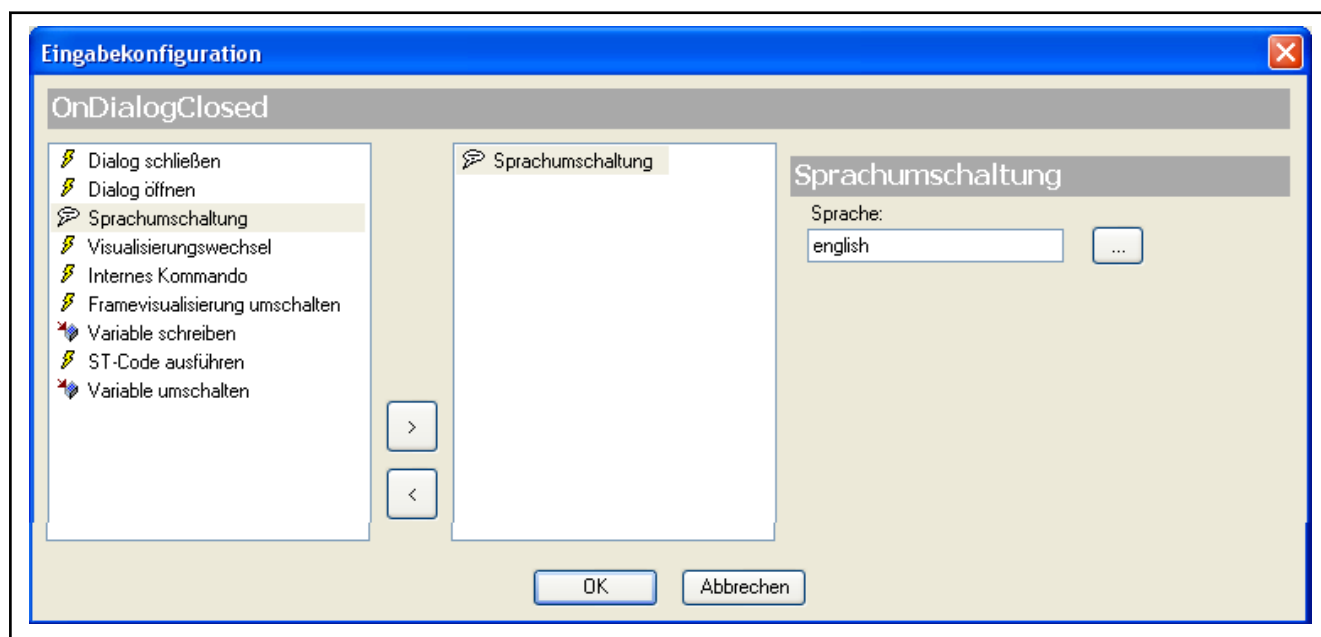


Abb. 20-22: 'Eingabekonfiguration', 'Sprachumschaltung' selektiert

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 Textlisten verwendet wird. Alternativ können Sie auf die Taste  klicken, um die gewünschte Sprache in der Eingabeunterstützung auszuwählen.

Visualisierungswechsel

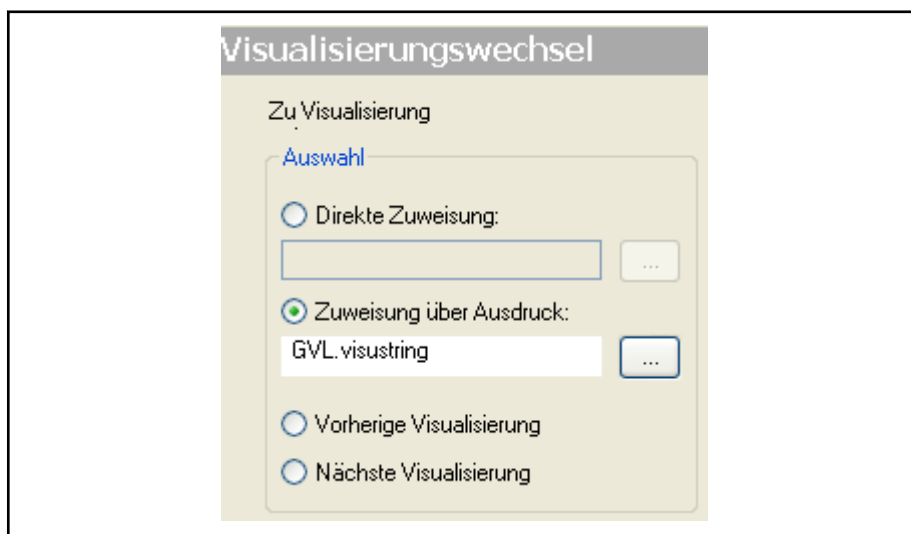


Abb. 20-23: 'Eingabekonfiguration', 'Visualisierungswechsel' selektiert

Wählen Sie eine der folgenden Optionen zur Festlegung, welche Visualisierung auf die Mausektion im Online-Betrieb 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 Eingabeunterstützung 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 vorher angezeigte Visualisierung wird wieder angezeigt. Wenn vorher 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.

Internes Kommando

Ausführen von Programmen, Drucken, und Kommandos des Rezeptverwalters.

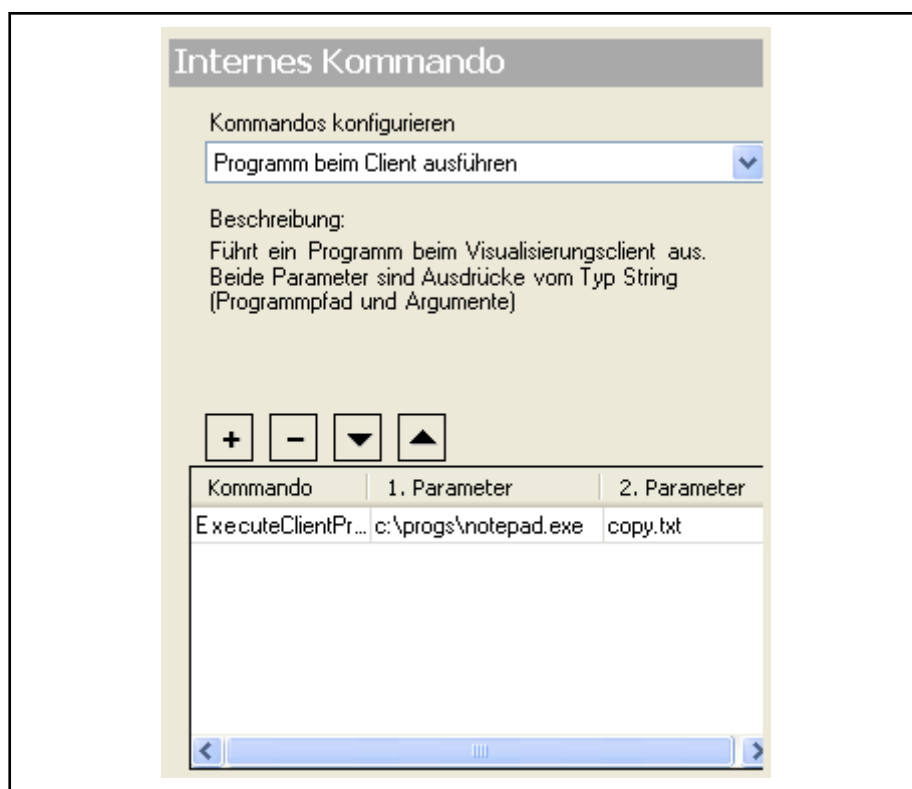


Abb. 20-24: 'Eingabekonfiguration', 'Internes Kommando' selektiert

Hier können ein oder mehrere Kommandos definiert werden, die auf die Mausektion hin ausgeführt werden sollen.

Wählen sie aus der Liste **Befehle konfigurieren** einen Befehl aus und betätigen Sie die  Schaltfläche, um ihn in der Tabelle im unteren Teil des Dialogs einzufügen. Diese Tabelle enthält alle ausgewählten Befehle der Eingabekonfiguration.

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 der gerade ausgewählte Eintrag von der Liste 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.

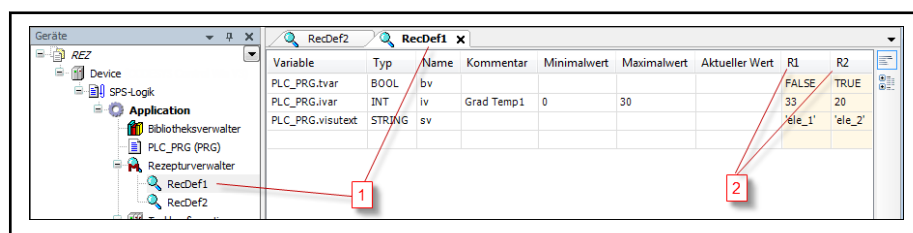
Programm auf der Steuerung ausführen Programm beim Client ausführen	Das angegebene Programm (*.exe) wird auf der Steuerung bzw. der Darstellungsvariante ausgeführt. 1. Parameter: Zeichenfolge Typ <code>STRING</code>, Pfad der Programmdatei (Beispiel: "C:\programs\notepad.exe") 2. Parameter: Zeichenfolge Typ <code>STRING</code>, 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 usw.. vorgenommen werden können. Darüber kann die aktuelle Visualisierung ausgedruckt werden. Dieser Befehl wird nur innerhalb der Target-Visualisierung unter Windows unterstützt.
Zu URL (WebVisu) navigieren	Voraussetzung: Visualisierung wird als VisuDeviceWeb ausgeführt. Bei Eintreten des Eingabeereignisses navigiert die Visualisierung zur Webseite der angegebenen URL. • 1. Parameter: Webadresse URL – Als Variable des Typs <code>STRING</code> um die Startseite programmatisch zu setzen. Zum Beispiel: <code>PlcProg.stURL</code> – Als Literal in einfachen Hochkommata. Zum Beispiel: <code>'http://de.wikipedia.org'</code> • 2. Parameter Wenn kein Parameter angegeben ist, wird die Webseite in einem neuen Fenster oder einer neuen Registerkarte dargestellt. Wenn <code>'replace'</code> angegeben ist, wird die VisuDeviceWeb durch die Webseite ersetzt.

Tab. 20-5: Befehle

Für die Kommandos des Rezeptur-Managers ist folgendes zu beachten:

Die **Rezepturdefinition** ist ein Objekt unterhalb des Rezeptur-Managers 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 Rezeptur-Manager 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.



1 Rezepturdefinition Name
2 Rezepturnamen

Abb. 20-25: Rezepturdefinition, Rezepturname

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 Istwerte 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 IndraLogic-Rezeptur-Manager angezeigt. In anderen Worten: Die in IndraLogic 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 angegebenen Rezeptur, wie sie im Rezeptur-Manager 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. Vorsicht: Die implizit verwendeten Rezeptur-Dateien, die als Zwischenablage beim Lesen und Schreiben von Rezepturen nötig sind, dürfen dabei nicht überschrieben werden, dies bedeutet, dass 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 Standarddateiauswahl-Dialog geöffnet. Der Filter ist automatisch auf Dateierweiterung "*.txtrecipe" gestellt. Nach dem Laden wird die Anzeige der betreffenden Rezeptur im IndraLogic-Rezeptur-Manager aktualisiert. 1. Parameter: Name der Rezepturdefinition 2. Parameter: Name der Rezeptur, für die die Werte aus der angegebenen Datei wieder in den Rezeptur-Manager 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.

Framevisualisierung umschalten

Voraussetzungen: In einer Visualisierung im Projekt gibt es Frame-Elemente, denen über den Befehl 'Frame-Auswahl' und Dialog **Konfiguration der Frame-visualisierungen** verschiedene Visualisierungen zugeordnet wurden ([Kap. 20.2.10 "Befehl 'Frame-Auswahl'" auf Seite 819](#)). 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.

Framevisualisierung umschalten: Im vorliegenden Dialog können Sie nun konfigurieren, dass bei Ausführen der Mausektion auf das aktuelle Visualisierungselement 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.

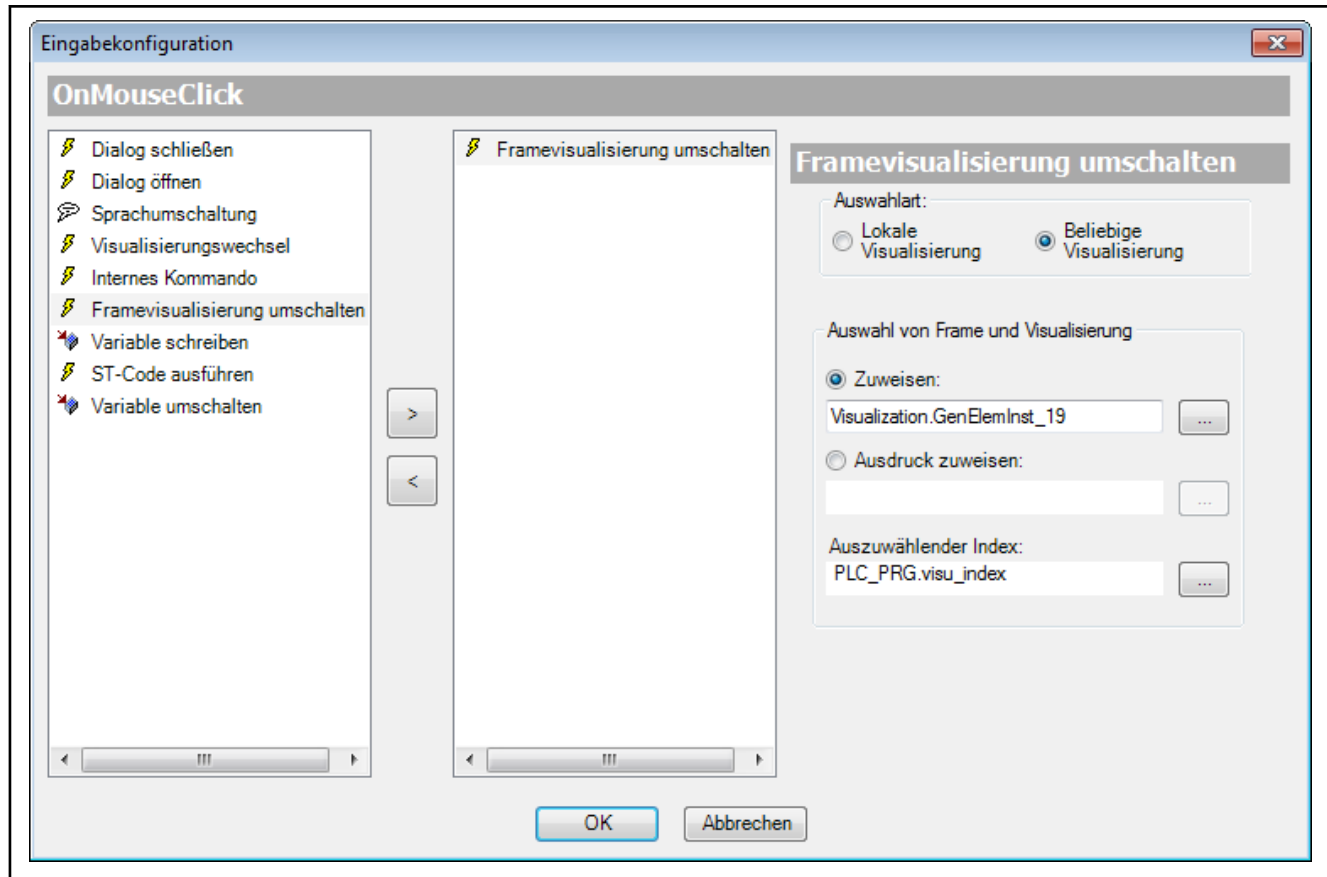


Abb. 20-26: "Eingabekonfiguration", 'Framevisualisierung umschalten' selektiert

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**.
- kann für alle im Projekt verfügbaren Visualisierungen erweitert werden, die auch dynamische Definitionen des Frames und der angezeigten Visualisierung zulassen -> Option **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.

Auswahl des Frames zeigt die lokalen Frame-Elemente und eingerückt darunter jeweils die zugeordneten Visualisierungen.

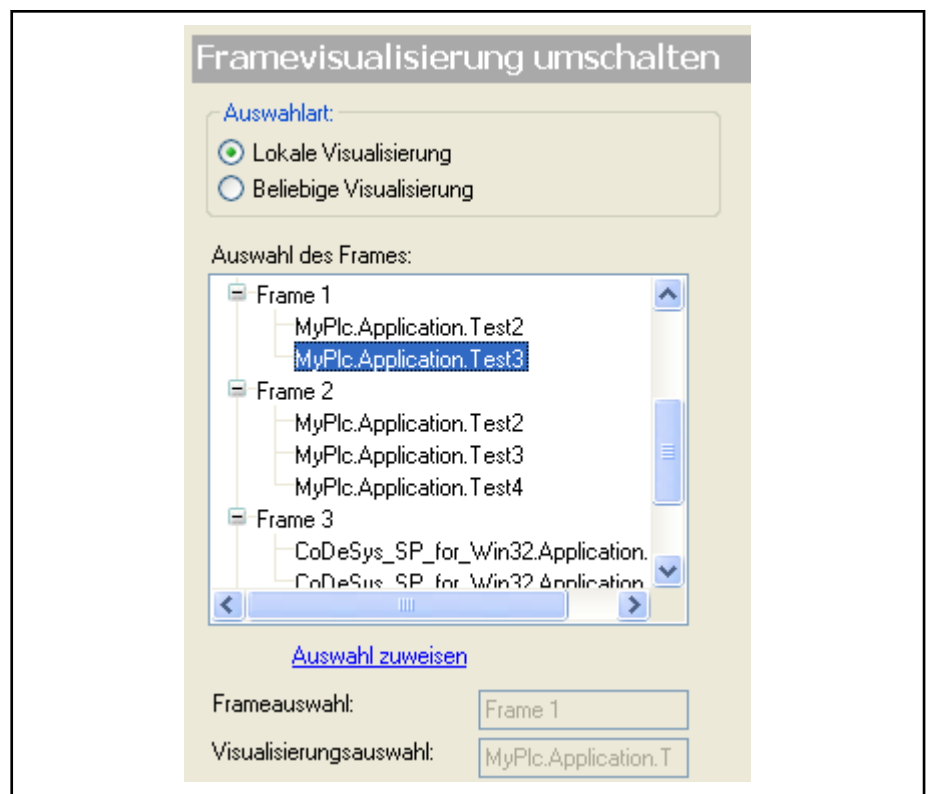


Abb. 20-27: Auswahl des Frames, lokal

Wählen Sie unterhalb des betroffenen Frames die gewünschte Visualisierung, die auf die Mausektion 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 Eingabeunterstützung dazu verwendet werden. Syntax: <Gerätename>.<Applikationsname>.<Visualisierungsname>.<Frame-Elementname>. Sehen Sie ein Beispiel in der folgenden Abbildung.

Abb. 20-28: 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 Eingabeunterstützung 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 Mausektion 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 Eingabeunterstützung zur Auswahl einer Variablen.

Variable schreiben

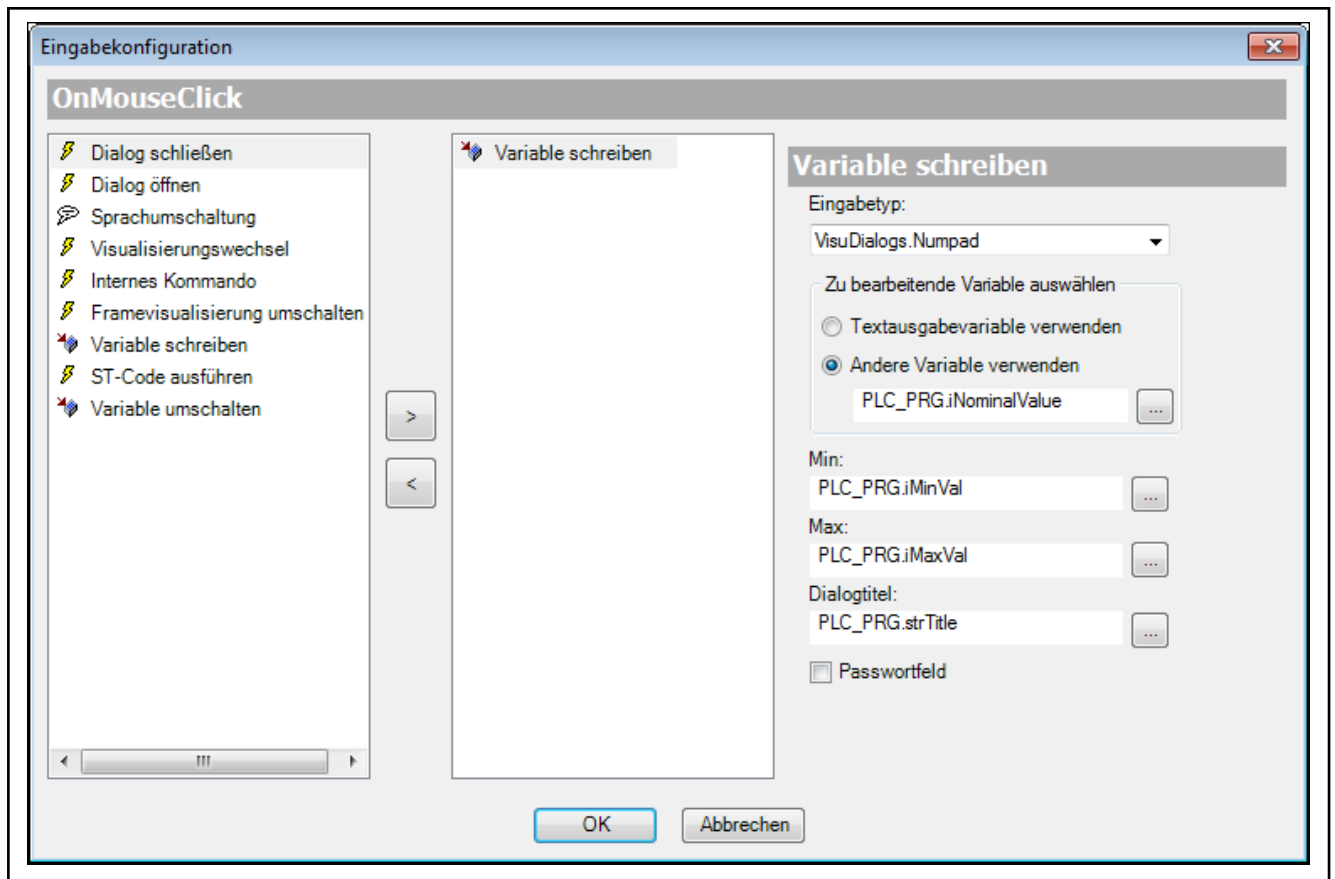


Abb. 20-29: 'Eingabekonfiguration', 'Variable schreiben'

Wenn für ein Visualisierungselement eine Eingabekonfiguration des Typs 'Variable schreiben' existiert, dann wird - sobald die entsprechende Mausaktion ausgeführt wird - das Element eine Möglichkeit bieten, einen Wert einzugeben. Der Wert kann dann als eine Zeichenfolge 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:

- **Standard:** Es wird der Standardeingabetyp verwendet. Sie können den Standard in den Einstellungen des Visualisierungsmanagers festlegen.
- **Texteingabe:** Es öffnet ein Feld für die Wert-/Texteingabe. Um Werte einzugeben, muss eine Tastatur vorhanden sein
- **Texteingabe mit Grenzen:** Es öffnet ein Feld für die Wert-/Texteingabe. Zusätzlich werden oberhalb die Eingabegrenzen angezeigt, welche bei Unterschreiten oder Überschreiten rot dargestellt werden. Um Werte einzugeben, muss eine Tastatur vorhanden sein
- **VisuDialogs.Keypad:** Die Mausaktion wird ein simuliertes Keypad (Tastaturfeld) öffnen. Sie können durch Mausklicks auf die entsprechenden Tastenfelder eine Zeichenfolge eingeben.

- **VisuDialogs.Numpad:** Die Mausektion wird ein simuliertes Numpad (Nummerntastaturfeld) öffnen. Sie können durch Mausklicks auf die entsprechenden Tastenfelder eine numerische Zeichenfolge eingeben.
- Zusätzlich zu den Standardeingabetypen werden alle Visualisierungen angeboten, welche in ihren Eigenschaften als "Nummernfeld / Tastatur / Dialog für Eingabekonfiguration" definiert wurden (siehe [Kap. 20.6 "Dialog 'Eigenschaften- 'Visualisierungsobjekt'" auf Seite 1063](#)).

Zu bearbeitende Variable auswählen

- **Textausgabevariable verwenden:** Der Wert wird in die Variable geschrieben, die als Textvariable in dem Visualisierungselement angegeben ist.
- **Andere Variable verwenden:** Angabe einer Projektvariable. Hierzu können Sie über die Schaltfläche  die Eingabeunterstützung öffnen.

Für den Wert, der in die angegebene Variable geschrieben werden soll, können Sie eine untere und obere Grenze definieren. Dazu tragen Sie entweder direkt Werte in die **Min.** und **Max.** Felder ein, oder Sie verwenden Projektvariablen.

Dialogtitel: Hier können Sie als Text oder über eine Textvariable angeben, was in der Titelleiste des Dialogs angezeigt werden soll.

Falls Sie wünschen, dass Ihre Eingabe durch "*****" (wie im Fall von Passwörtern) versteckt dargestellt werden soll, so aktivieren Sie das Check-box vor dem **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-Betrieb die Maustaste auf diesem Rechteck gedrückt wird, erscheint ein Nummerntastaturfeld mit dem Titel, der von der Variablen `PlcProg.strTitle` 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 `PlcProg.iMinVal` und `PlcProg.iMaxVal` gegeben werden. Sobald die Eingabe mit einem Mausklick auf die Taste OK bestätigt wird, wird "123" auf die Variable `PlcProg.iNominalValue` geschrieben. Wenn `iNominalValue` als STRING-Variable definiert ist, erhält es den Wert '123', wenn `iNominalValue` eine numerische Variable ist, erhält sie den Wert 123.

ST-Code ausführen

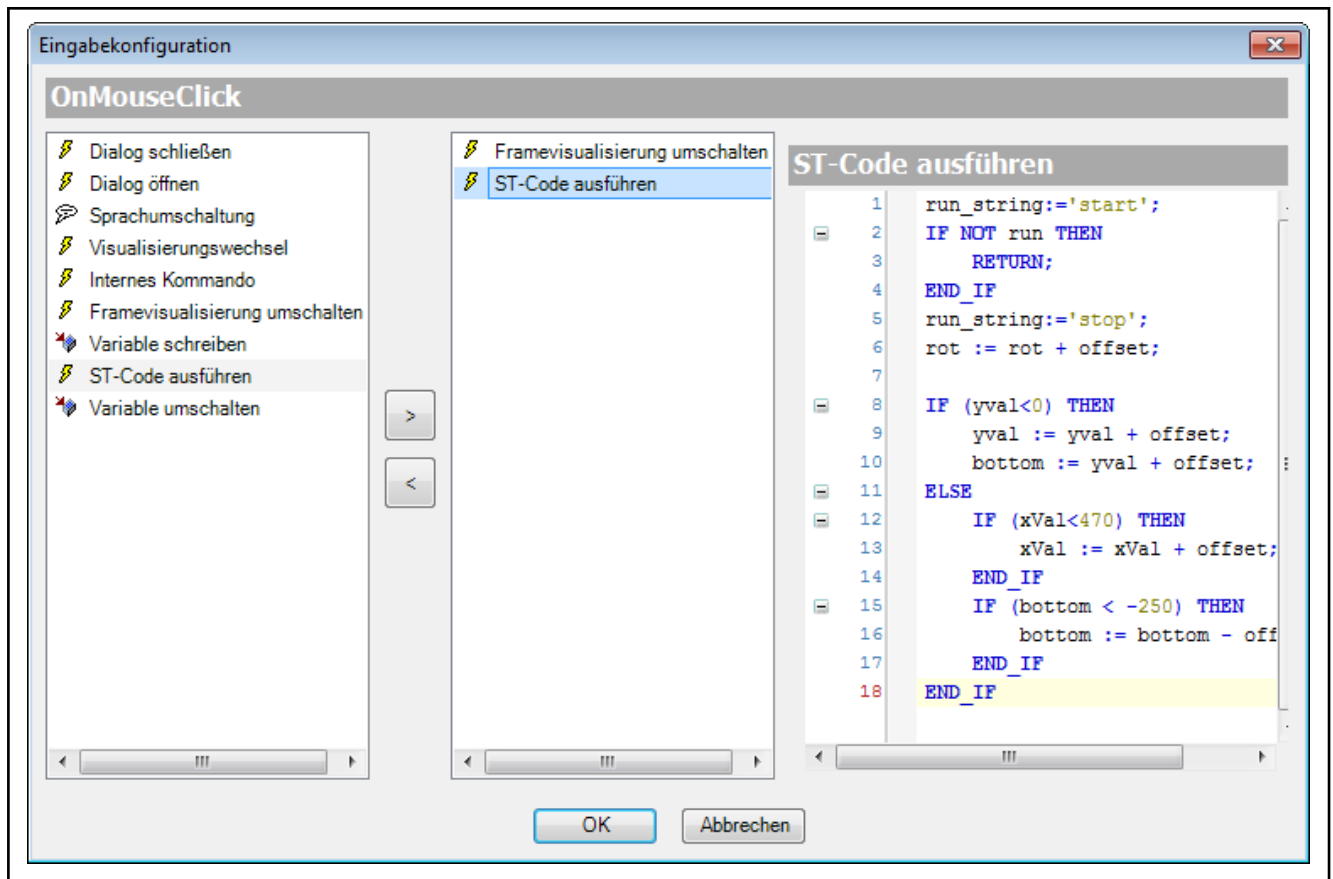


Abb. 20-30: 'Eingabekonfiguration', 'St-Code ausführen' selektiert

Im Eingabefeld kann Code in Strukturiertem Text eingegeben werden, der auf die Mausektion hin ausgeführt werden soll.

Variable umschalten

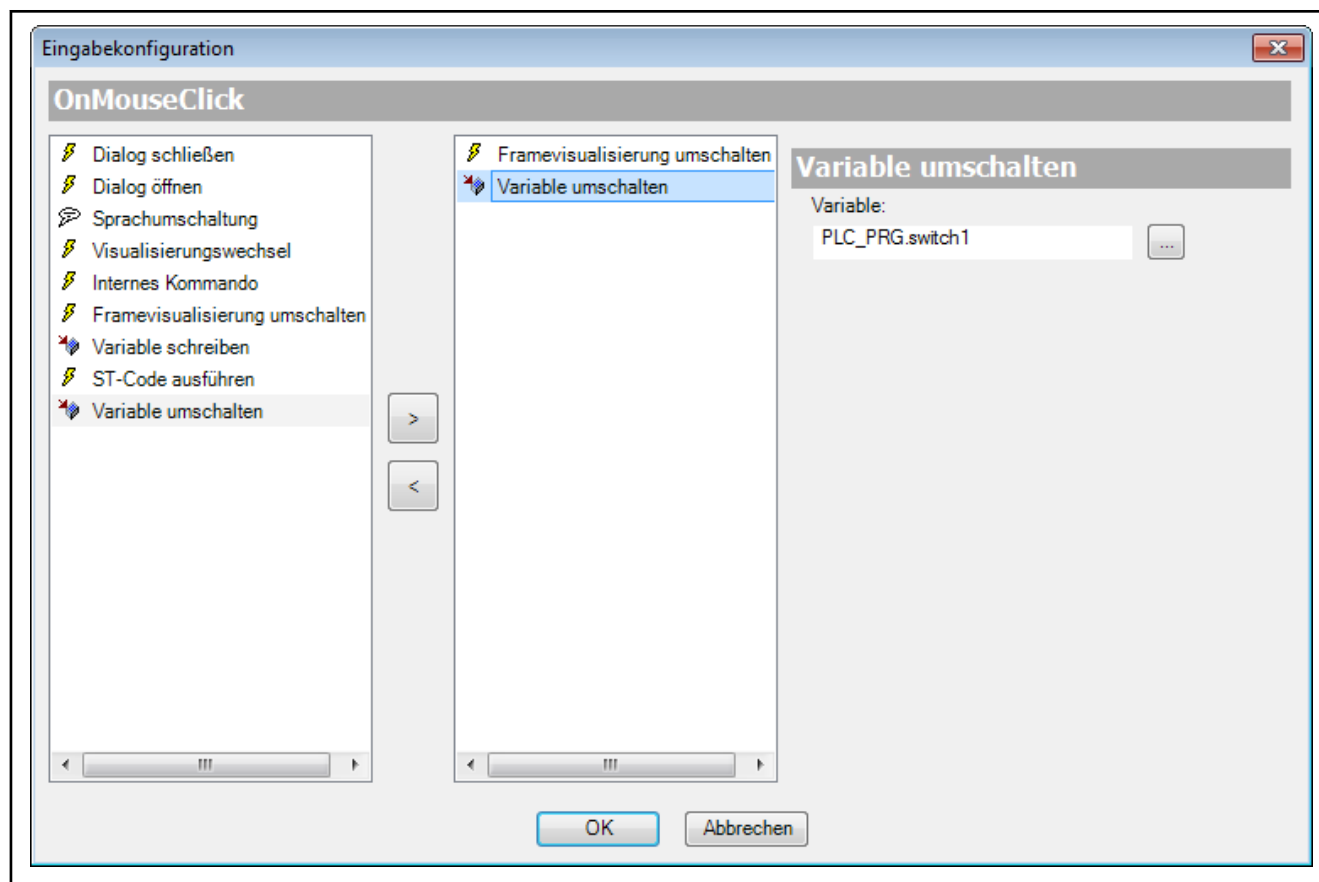


Abb. 20-31: 'Eingabekonfiguration', 'Variable umschalten' selektiert

Hier kann eine boolesche Variable eingegeben werden, die bei wiederholter Mausektion zwischen TRUE und FALSE wechseln soll. Beispiel: `PlcProg.switch1` Mit Klick auf [...] wird die Eingabeunterstützung sich öffnen.

Farbverlaufeditor

Ein Klick in das Wertefeld von **Farbverlaufauswahl** im **Eigenschaften** Fenster öffnet den **Farbverlaufeditor**.

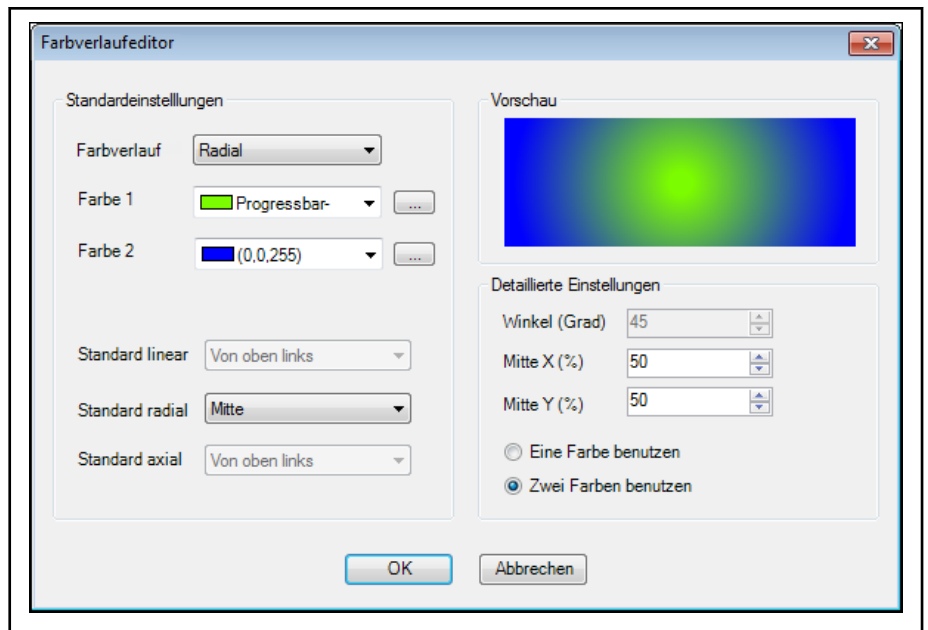


Abb. 20-32: Farbverlaufeditor

Standardeinstellungen**Farbverlauf:** Linear, Radial, Axial.**Farbe 1:** Erste Farbe des Gradienten. Sie kann aus der Combobox oder aus dem Farbdialog, zu öffnen mit , gewählt werden.**Farbe 2:** Zweite Farbe des Gradienten. Sie kann aus der Combobox oder aus dem Farbdialog, zu öffnen mit , gewählt werden.**Standard linear:** Vordefinierte Einstellungen des Farbverlauftypes **Linear** (Winkel des Gradienten): **Horizontal**, **Vertikal**, **Von oben links**, **Von oben rechts**, **Horizontale Farbe umgeschaltet**, **Vertikale Farbe umgeschaltet**, **Von unten links**, **Von unten rechts**.**Standard radial:** Vordefinierte Einstellungen des Farbverlauftypes **Radial** (Position der Mitte): **Mitte**, **Oben links**, **Oben rechts**, **Unten links**, **Unten rechts**.**Standard axial:** Vordefinierte Einstellungen des Farbverlauftypes **Axial** (Winkel des Verlaufs): **Horizontal**, **Vertikal**, **Von oben rechts**, **Von oben links**.**Detaillierte Einstellungen****Winkel (Grad):** Nur für Farbverlauftyp **Linear** und **Axial** verfügbar.**Mitte X (%):** Nur für Farbverlauftyp **Radial** verfügbar. X-Position des Mittelpunktes (0 - 100%)**Mitte Y (%):** Nur für Farbverlauftyp **Radial** verfügbar. Y-Position des Mittelpunktes (0 - 100%)**Eine Farbe benutzen:** Farbverlauftyp zwischen **Farbe 1** und der gleichen Farbe mit anderer Helligkeit. Die **Helligkeit** kann von 0 (schwarz) bis 100 (weiß) eingestellt werden.**Zwei Farben benutzen:** Farbverlauftyp zwischen den beiden ausgewählten Farben **Farbe 1** und **Farbe 2**.**Semitransparente Farbe verwenden**

Sie können ein Visualisierungselement mit semitransparenter Farbe ausstatten, indem Sie die Elementeigenschaft **Transparenz**, die eine untergeordnete Eigenschaft zu einer Farbe oder einer Schriftartfarbe ist, setzen.

Ein Wert für Transparenz kann auch im Visualisierungsstil definiert sein.

Die Hintergrundfarbe einer Visualisierung ist immer voll opak.



Die Konfiguration mit einer semitransparenten Farbe ist erst ab Compiler-Version 3.5.6.0 möglich.

Die Darstellungsvarianten verhalten sich bezüglich einer semitransparenten Visualisierung wie folgt:

- Die in IndraLogic integrierte Visualisierung zeigt immer semitransparent an

Siehe auch

- [Kap. "Semitransparente Farbe verwenden" auf Seite 863](#)
- [Kap. 20.5.2 "Die Clients und ihre Editoren" auf Seite 1061](#)

Semitransparenz abhängig von den Elementtypen

Die konfigurierbare Farbe bei Elementen der Kategorie **Allgemeine Steuerelemente** kann nur in Teilbereichen, trotzdem aber in mehreren Zeichenebenen verwendet sein. Das hat dann zur Folge, dass trotz semitransparenter Farbe der Hintergrund nicht durchscheint.

Semitransparenter Elementtyp

- Alle der Kategorie **Basis** außer
 - **Bild**
 - **Frame**
- Alle der Kategorie **Allgemeine Steuerelemente** außer
 - **Checkbox**
 - **Combobox**
 - **Fortschrittsbalken**
 - **Radiobutton**
 - **Registersteuerelement**
 - **Schieberegler**
 - **SpinControl**
- Alle der Kategorie **Datum-/Zeit-Steuerelement**

Opaker Elementtyp

- Alle der Kategorie **Messgeräte**
- Alle der Kategorie **Lampen/Schalter/Bilder**
- Alle der Kategorie **Spezielle Steuerelemente**

Semitransparenz aktivieren

Voraussetzung: Eine der Darstellungsvarianten, die im Visualisierungsmanager konfiguriert ist, kann eine Visualisierung semitransparent darstellen.

Beispiel: Die Steuerung unterstützt die Semitransparenz nicht und als Darstellungsvariante ist ausschließlich die integrierte IndraLogic-Visualisierung konfiguriert.

IndraLogic ist mit einem Projekt, das eine Visualisierung enthält, geöffnet.

1. Doppelklicken Sie auf Objekt **Visualization Manager**



Editor **Visualization Manager** öffnet sich.

2. Wählen Sie die Registerkarte **Einstellungen**.
3. Aktivieren Sie unter **Zusätzliche Einstellungen** die Option **Semitransparentes Zeichnen aktivieren**.

Der Visualisierungseditor stellt Elemente, die in ihren Eigenschaften für eine Farbeinstellung einen Transparenzwert gesetzt haben, semitransparent dar.

Siehe auch

- [Kap. "Visualisierungsmanager- 'Einstellungen'" auf Seite 1058](#)

Wert für Transparenz zuweisen

Voraussetzung: IndraLogic ist mit einem Projekt, das eine Visualisierung enthält, geöffnet.

1. Doppelklicken Sie auf die Visualisierung.
2. Wählen Sie **Ansicht ► Elementeigenschaften**.
3. Selektieren Sie ein Visualisierungselement.
4. Klicken Sie im Fenster **Eigenschaften** in Kategorie **Farben**. Expandieren Sie die Farbeinstellung bis die Eigenschaft **Transparenz** sichtbar wird. Klicken Sie in das Wertefeld und setzen Sie den Wert mit Hilfe des Schiebereglers.

Die integrierte Visualisierung stellt das Element semitransparent dar. Bei Ausführung der Applikation wird die Visualisierung als Darstellungsvariante semitransparent dargestellt, wenn die Darstellungsvariante entsprechend konfiguriert ist.

5. Klicken Sie im Fenster **Eigenschaften** in Kategorie **Texteigenschaften**. Expandieren Sie die Einstellungen bis die Eigenschaft **Transparenz** sichtbar wird. Klicken Sie in das Wertefeld und geben Sie einen Wert im Bereich von 0 bei 255 ein.

Vervielfachen eines Elements

Sie können ein Visualisierungselement so konfigurieren, dass es als Vorlage zum Vervielfachen geeignet wird. Beim Vervielfachen entsteht eine tabellarische oder zeilenweise oder spaltenweise Anordnung.

Um ein Element als Vorlage zu konfigurieren, müssen Sie

- eine Eigenschaft, die speziell ist, mit einem Array konfigurieren, das mit Platzhaltern indiziert ist. Speziell ist eine elementabhängige Eigenschaft wie zum Beispiel ein Ausgabertext, der bei jedem vervielfachten Element einen anderen Text ausgibt.
- einer Eigenschaft, die bei allen Elementen gleich ist, einen festen Wert zuweisen. Zum Beispiel schwarz für die Rahmenfarbe, wenn alle vervielfachten Elemente einen schwarzen Rahmen erhalten sollen.

Sie können bei einem Platzhalter-Array für die 1te Dimension `$FIRSTDIM$` und für die 2te Dimension `$SECONDDIM$` verwenden. In besonderen Fällen können Sie ein Array angeben, das über mehr Dimensionen verfügt, als Sie über die Platzhalter indizieren.

Zum Beispiel: `Main.strArr2Dim[1..10, $FIRSTDIM$, $SECONDDIM$, 1..10]`

Außerdem können Sie ein Element mit beliebig vielen Arrays konfigurieren, die im einfachsten Fall alle die gleiche Indizierung haben. In besonderen Fällen müssen Sie berücksichtigen, welche Dimension über Platzhalter indiziert werden soll.

Wenn Sie die Eigenschaften eines Elements bearbeiten, können Sie über Intelligentes Kodieren die Platzhalter einfügen.

Nun kann IndraLogic das Element als Vorlage erkennen und Sie können den Befehl **Visualisierungselement vervielfachen** ausführen. Bei der Konfiguration

des Befehls können Sie die Anzahl und Anordnung der einzufügenden Elemente bestimmen. Außerdem können Sie das Ersetzen der Platzhalter konfigurieren.

IndraLogic fügt dann Elemente, die dem Vorlagenelement gleichen, in tabellarischer Anordnung ein. Die in **Eigenschaften** konfigurierten Arrays erhalten statt den Platzhaltern nun konkrete Indizes.

Elemente, die Sie vervielfachen können

- Rechteck
- Abgerundetes Rechteck
- Ellipse
- Linie
- Polygon
- Linienzug
- Bézierkurve
- Bild
- Frame
- Schaltfläche
- Kreissektor
- SpinControl
- Textfeld
- Checkbox
- Bildwechsler
- Lampe
- Hebelschalter
- Energieschalter
- Drückschalter
- Drückschalter LED
- Kippschalter
- Drehschalter

Erste Schritte

Sie haben in IndraLogic eine Visualisierung geöffnet.

1. Ziehen sie ein Rechteck in die Visualisierung.
2. Konfigurieren Sie die Rahmenfarbe rot und die Füllfarbe grau.

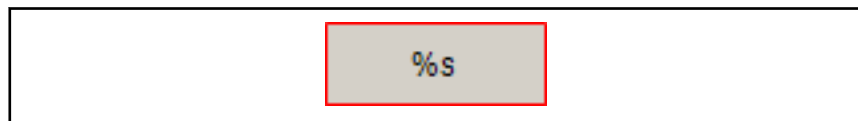


Abb. 20-33:

3. Deklarieren Sie im IEC-Code Arrays, mit denen die Eigenschaften Ausgabetext, Tooltip-Text und die Eingabekonfiguration elementabhängig konfiguriert werden können.

```

PROGRAM PLC_PRG
VAR
    strArr2Dim : ARRAY [1..5, 2..10] OF STRING;
    strArr2DimTt : ARRAY [1..5, 2..10] OF STRING;
    boolArr2Dim : ARRAY [1..5, 2..10] OF BOOL;
END_VAR

```

Abb. 20-34:

- Konfigurieren Sie das Vorlagenelement mit einem Array, indem Sie in **Textvariable** den Arraynamen eingeben. Ersetzen Sie die 1te Dimension mit dem Platzhalter \$FIRSTDIM\$ und die 2te Dimension mit dem Platzhalter \$SECONDDIM\$. Zum Beispiel: PlcProg.strArr2Dim[\$FIRSTDIM\$, \$SECONDDIM\$].

Konfigurieren Sie die Eigenschaft **Text** mit %s.

Der Ausgabetext des Vorlagenelements ist mit einem Array konfiguriert. Zur Laufzeit gibt die Visualisierung einen indexabhängigen Text aus.

- Konfigurieren Sie das Vorlagenelement mit einem Array, indem Sie in **Eingabekonfiguration, OnMouseClicked** einen Arraynamen eingeben. Zum Beispiel: PlcProg.boolArr2Dim[\$FIRSTDIM\$, \$SECONDDIM\$]

Die Eingabekonfiguration des Rechtecks ist konfiguriert. Zur Laufzeit verarbeitet die Visualisierung pro Rechteck den Mausklick.

- Wählen Sie den Befehl **Visualisierung ► Visualisierungselement vervielfachen**.

Der Dialog **Visualisierungselement vervielfachen** öffnet sich.

- Konfigurieren Sie das Vervielfachen und beenden Sie den Dialog mit **OK**.

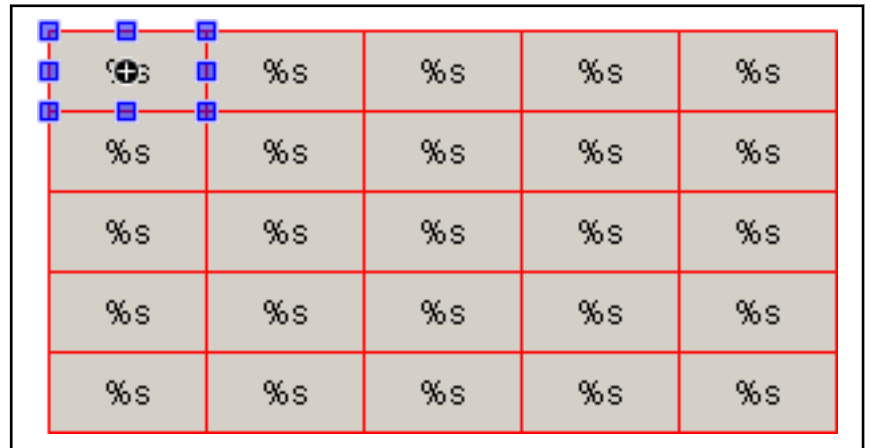


Abb. 20-35:

Siehe auch

- [Kap. 20.2.17 "Befehl 'Visualisierungselement vervielfachen'" auf Seite 823](#)
- [Kap. "Eingabekonfiguration" auf Seite 849](#)
- [Kap. "Dialog 'Optionen- 'Intelligentes Codieren'" auf Seite 775](#)

IEC-Variable auf ein Element ziehen

Sie können eine Variable an der Stelle, an der sie deklariert oder auch verwendet wird, selektieren und in eine Visualisierung ziehen. Wenn Sie dort die Variable auf ein Element ziehen, konfigurieren Sie damit eine Elementeigenschaft mit dieser Variablen.



Sie können eine Variable in einen Editor ziehen, indem Sie die Variable erst auf das zugehörige Objekt im Projektextplorer ziehen.

Oder indem Sie die Variable erst auf den Reiter des Editors ziehen, wenn dieser im Hintergrund geöffnet ist. Dann aktiviert IndraLogic den Editor im Hauptfenster.

Elemente, die Drag-and-drop unterstützen

Elemente, die Drag-and-drop unterstützen, werden in einer vorbestimmten Eigenschaft mit der Variablen konfiguriert.

*Konfigurieren als Ausgabetexts in Eigenschaft **Textvariable** bei folgenden Elementen*

- Rechteck
- Linie
- Polygon
- Kissektor
- Schaltfläche
- Bild
- Textfeld

Wenn die Eigenschaft **Text** nicht konfiguriert ist, fügt IndraLogic dort auch noch den formatierenden Text %s ein.

*Konfigurieren als Eigenschaft **Variable** bei folgenden Elementen*

- Bildwechsler
- Checkbox
- Combobox
- Fortschrittsbalken
- Lampe
- Schalter
- SpinControl
- Schieberegler
- Radiobutton

*Konfigurieren als Ausgabewert in Eigenschaft **Wert** bei folgenden Elementen*

- Balkenanzeige
- Potentiometer
- Scrollbalken
- Zeigerinstrument

*Konfigurieren als Array-Variable in Eigenschaft **Datenarray** bei folgenden Elementen*

- Tabelle
- Histogramm

Konfigurieren als Tracevariable in Eigenschaft **Trace**. Dabei können Sie beliebig viele Variablen auf das Element **Trace** ziehen.

Siehe auch

Elemente, die Drag-and-drop nicht unterstützen

- [Kap. "Element 'Trace'" auf Seite 1032](#)
- Alarmbanner
- Alarmtabelle
- Datumsbereichsauswahl
- Frame
- Groupbox
- Unsichtbare Eingabe
- Beschriftung
- Legende
- Registersteuerelement
- Textfeld
- Zeitauswahl
- Wartesymbol Blume

Erste Schritte

1. Selektieren Sie eine Variable. Zum Beispiel `strText`.
2. Ziehen Sie die Variable in eine Visualisierung und dann auf ein Element in der Visualisierung, zum Beispiel auf ein Rechteck.
Der Cursor ist mit einem Plus dekoriert und das Element ist selektiert.
3. Lassen Sie die linke Maustaste los.
IndraLogic konfiguriert das Element mit dieser Variablen. In **Textvariable** ist `PlcProg.strText` und in **Text** ist `%s` eingefügt.

20.4.2 'Basis'

Element 'Rechteck', 'Abgerundetes Rechteck' und 'Ellipse'

Rechteck, abgerundetes Rechteck und Ellipse sind vom gleichen Elementtyp. Sie können nur durch Ändern der Eigenschaft **Elementtyp** gegenseitig konvertiert werden.

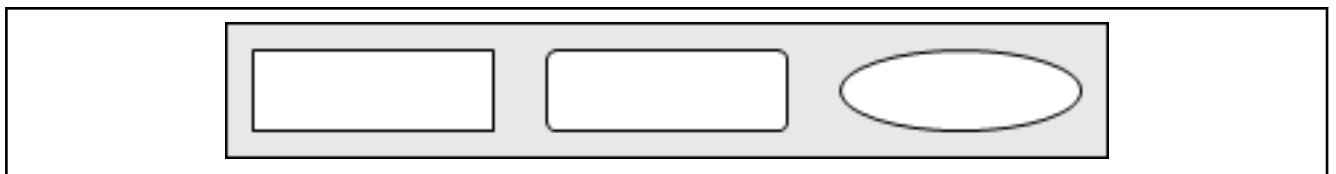


Abb. 20-36: Rechteck, abgerundetes Rechteck und Ellipse

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.

- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	• Rechteck • abgerundetes Rechteck • Ellipse

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Zentrum

Mit Bearbeiten des Wertes wird gleichzeitig das zugehörige Element  im Editor verschoben.

Eigenschaft	Beschreibung Wert
X	X-Position des Rotationspunkts des Elements
Y	Y-Position des Rotationspunkts des Elements

Farben Eine Farbe wird mittels Hexzahl definiert, die sich aus Rot/Grün/Blau (RGB) Anteilen zusammensetzt. Für jede dieser drei Farben sind 256 (0-255) Werte verfügbar. Hier sind die Einstellungen **statisch**. Die Farbe kann aus einer Auswahlliste oder dem Farbauswahldialog, der über Button  öffnet, gewählt werden. Beispiel:  0; 128; 64

Eigenschaft	Beschreibung Wert
Normalzustand - Rahmenfarbe - Füllfarbe	Auszuwählen ist eine Rahmen- und Füllfarbe für Normalzustand. Im Fall, dass die Variable definiert in Farbvariablen ► Farbumschlag FALSE ist, ist das Element in Normalzustand.
Alarmzustand - Rahmenfarbe - Füllfarbe	Auszuwählen ist eine Rahmen- und Füllfarbe, mit der das Element im Alarmzustand dargestellt wird. Dies ist der Fall, wenn die Variable in Farbvariablen ► Farbumschlag TRUE ist.

Farbverlauf verwenden	Standardmäßig ist die Checkbox deaktiviert. Falls sie aktiviert wird, wird das zugehörige Element mit Farbverlauf gezeichnet.
Farbverlaufsauswahl	Dialog öffnet. Kap. "Farbverlaufeditor" auf Seite 862

Aussehen

Eigenschaft	Beschreibung Wert
Linienstärke	Stärke der Rahmenlinie in Pixel. 0 kodiert das Gleiche wie 1 und setzt die Linienstärke auf 1 Pixel. Wird kein Rahmen gewünscht, muss man Linienart auf unsichtbar setzen.
Füllart	Auszuwählen ist eines dieser Füllarten angewendet in Füllfarbe - Ausgefüllt: Die Füllfarbe ist sichtbar - Unsichtbar: Die Füllfarbe ist unsichtbar
Linienart	<i>Auszuwählen ist eine der Linienarten angewendet bei der Umrisslinie:</i> - Durchgezogen - Striche - Punkte - Strich Punkt - Strich Punkt Punkt - Unsichtbar: Umrisslinie ist nicht sichtbar.

Elementeigenschaft 'Texte' Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Text	Geben Sie einen Text ein. Er wird verwendet um das Element zu beschriften. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Textvariable zugewiesen.
Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Tooltip-Variable zugewiesen.

Texteigenschaften Diese Eigenschaften werden für eine statische Definition der Schriftart verwendet.

Eigenschaft	Beschreibung Wert
Horizontale Ausrichtung	<i>Definieren der horizontalen Ausrichtung des Textes durch Auswahl aus:</i> • Links • Zentriert • Rechts
Vertikale Ausrichtung	<i>Definieren der vertikalen Ausrichtung des Textes durch Auswahl aus:</i> • Oben • Zentriert • Unten
Schriftart	<i>Auswahl aus vordefinierten Schriftarten:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet sich der Dialog für benutzerdefinierte Eigenschaften der Schriftart.
Farbe Schriftart	Auszuwählen ist eine Schriftartfarbe für das Element. Entweder aus der Auswahlliste oder durch Setzen im sich öffnenden Dialog, wenn  geklickt wird.

Elementeigenschaft 'Absolute Bewegung' Das Element kann bewegt werden, indem die X- und Y-Position (Pixel) der linken oberen Ecke des Elements durch eine Integer-Variable verändert wird.

Bewegung • X • Y	• X: Hier ist eine Integer-Variable einzugeben, deren Wert die aktuelle X-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in X-Richtung zu verschieben (ein positiver Wert verschiebt das Element von links nach rechts). • Y: Eingabe einer Integer-Variablen, die die aktuelle Y-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in Y-Richtung zu verschieben (ein positiver Wert verschiebt das Element von oben nach unten).
Rotation	Eingabe einer Integer-Variablen, deren Wert den Winkel (Winkelgrade) definiert, um den das Element um den Rotationspunkt Zentrum  rotiert. positiver Wert = mathematisch negativ = im Uhrzeigersinn Zu beachten ist, dass das Element, im Gegensatz zum Verhalten bei Innere Rotation, selber nicht rotiert.

Skalierung	Eingabe einer Integer-Variablen, deren Wert den aktuellen Skalenwert (Prozent) definiert. 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  .
Innere Rotation	Die hier eingetragene Integer-Variable definiert den Winkel (Winkelgrade), um den das Element um seinen Rotationspunkt gedreht wird: Zentrum  positiver Wert = mathematisch negativ = im Uhrzeigersinn. Im Gegensatz zu Rotation rotiert das Element tatsächlich.

Elementeigenschaft 'Relative Bewegung'

Das Element kann relative zu seiner fixen Position definiert in **Position** verschoben werden. Die obere linke und untere rechte Ecke des Elements werden dabei um die in den Integer-Variablen angegebenen Werte (Pixel) in X- bzw. Y-Richtung verschoben. Im Gegensatz zur absoluten Bewegung wird eine relative Position definiert, also der Abstand zur ursprünglichen Position. Das Element kann dadurch in der Form verändert werden.

Siehe **Absolute Bewegung** für das Verschieben des Elements als Ganzes.

Bewegung links-oben	
X	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die linke obere Ecke in X-Richtung verschoben wird.
Y	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die linke obere Ecke in Y-Richtung verschoben wird.
Bewegung rechts-unten	
X	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die rechte untere Ecke in X-Richtung verschoben wird.
Y	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die rechte untere Ecke in Y-Richtung verschoben wird.

Elementeigenschaft 'Textvariablen'

Sie können Text, der in einer Variablen gespeichert ist, anzeigen. Fügen Sie dazu zuerst eine Formatierungssequenz in dem Text hinzu, der unter der Eigenschaft **Texte** definiert ist. Weisen Sie dann hier eine Variable hinzu. Im Online-Betrieb wird die Formatierungssequenz durch den Inhalt der Variablen ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Beispiel:

Beispiel %f

Geben Sie **Ergebnis: %2.5f** in der Eigenschaft **Text** ein. Geben Sie **rValue** in der Eigenschaft **Textvariable** ein. Es muss eine definierte IEC-Variable sein. Dann wird im Online-Betrieb das Element mit **Ergebnis: 12.12345** beschriftet, wenn **rValue** = 12.12345 ist.

Textvariable	Geben Sie eine Variable (Typ ist Standarddatentyp) ein, der die Information enthält, die sie angezeigt haben wollen. Der Typ muss konform mit der Formatierungssequenz in Texte ► Text sein.
Tooltipvariable	Geben Sie eine STRING-Variable ein, die den Tooltiptext enthält, der angezeigt werden soll. Die Eingabe in Texte ► Tooltip muss ein %s enthalten.

Dynamische Texte

Diese Parameter dienen der Definition von dynamischen Texten, die aus Textlisten stammen (dies erlaubt Sprachumschaltung) ([Kap. 18.2.15 "Menü Textliste" auf Seite 727](#)).

Eigenschaft	Beschreibung Wert
Textliste	Name der Textliste (String) wie im POU- oder Geräte-Fenster verwendet; z.B. <code>errorlist</code>
Textindex	Index (ID) des Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. 'M1' oder String-Variable, z.B. <code>error_22</code> .
Tooltipindex	Index (ID) des Tooltip-Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. 'T1' oder String-Variable, z.B. <code>error_Tooltip_22</code> .

Elementeigenschaft 'Schriftartvariablen'

Schriftartvariablen werden bei dynamischen Schriftdefinitionen des Elementtextes über Projektvariablen verwendet. Statischen Definitionen werden unter **Texteigenschaften ► Schriftart** konfiguriert.

Schriftname	Hier ist eine Variable Typ <code>STRING</code> anzugeben, die den Schriftnamen enthält, der für die Beschriftung des Elements verwendet werden soll. (Namensangabe wie im Standard-Schrifttypen-Dialog). Beispiel: <code>PlcProg.stFontVar := 'Arial';</code>
Größe	Eingabe einer Variablen Typ <code>INT</code> , die die Größe des Elementtextes in Pixel enthält, so wie im Standarddialog Schriftart. Beispiel: <code>PlcProg.iFontHeight := 16;</code>
Flags	Eingabe einer Variable Typ <code>DWORD</code> , die mit Hilfe der unten genannten Flag-Werte die Schriftart-Darstellung festlegt. Eine kombinierte Definition kann erreicht werden, indem die jeweiligen Flag-Werte addiert werden und die Summe angegeben wird. • 1: kursiv • 2: fett • 4: unterstrichen • 8: gelöscht Beispiel für einen Text, der fett und unterstrichen dargestellt wird: <code>PlcProg.dwFontType := 6;</code>
Zeichensatz	Der Zeichensatz, der für die Schriftart verwendet werden soll, kann über die Standard-Zeichensatznummer definiert werden. Mit einer <code>DWORD</code> -Variablen kann diese Nummer angegeben werden (siehe auch die Skript -Definition im Standard-Schriftartdialog)
Farbe	Eingabe einer <code>DWORD</code> -Variable, in der die Farbe des Elementtextes festgelegt wird.
Flags für Textausrichtung	Variable mit ganzzahligem Datentyp, zum Beispiel <code>PlcProg.dwTextAlignment</code> . Die Visualisierung wertet zur Laufzeit die Variable aus und richtet dementsprechend den Text aus. <i>Textausrichtung</i> • 0: oben links • 1: horizontal zentriert • 2: rechts • 4: vertikal zentriert • 8: unten

Elementeigenschaft 'Farbvariablen'

Die Elementeigenschaft ermöglicht die dynamische Umschaltung der Elementfarben mit Hilfe von IEC-Variablen für Steuerung und Farben.

Die IEC-Variable für Farbe enthält diese als hexadezimaler Farbdefinition, die die Rot-/Grün-/Blauanteile der Farbe im RGB-Farbraum codiert: `16#00RRGGBB`. Dabei ist für jeden Farbanteil ein Byte für 256 Farbabstufungen

gen (von 0-255) reserviert. 0 bedeutet dabei, dass kein Farbanteil enthalten ist. 255 bedeutet 100% Farbanteil. Das führende Byte ist immer 16#00.

Beispiel:

Farbvariable

dwFillColor := 16#008FE03F;

Codierung: Rotanteil: 8F, Grünanteil: E0, Blauanteil: 3F

- grau: dwGrey := 16#00888888
- rot: dwRed := 16#00FF0000
- schwarz: dwBlack := 16#00000000

Farbumschlag	<i>Steuervariable</i> • als IEC-Variable des Typs <code>BOOL</code>, zum Beispiel <code>prgA.bToggleColor</code> Zur Laufzeit steuert die Visualisierung dynamisch die Farbdarstellung des Elements – <code>TRUE</code>: Element in Normalzustand, zum Beispiel in grau mit schwarzem Rahmen – <code>FALSE</code>: Element in Alarmzustand, zum Beispiel in gelb mit rotem Rahmen. • oder als Platzhalter <Umschalten-/TastenvARIABLE> Für den Farbumschlag ist dabei keine eigene Variable zugewiesen. Stattdessen wird der Farbumschlag mit der Variablen gesteuert, die in Eingabekonfiguration, Umschalten oder Tasten zugewiesen ist. Voraussetzung ist also, dass dort eine Steuervariable konfiguriert ist. Wenn sowohl in Umschalten als auch in Tasten eine Steuervariable konfiguriert ist, wird die in Tasten verwendet. Sie können den Platzhalter entfernen, indem Sie <Umschalten-/TastenvARIABLE> selektieren und <Entf> wählen. Danach erscheint  im Wertefeld.
Klick auf  im Wertefeld von Farbumschlag	IndraLogic fügt den Platzhalter <Umschalten-/TastenvARIABLE> ein.
Normalzustand	Farbvariablen des Elements während eines ungestörten Betriebs
• Rahmenfarbe	IEC-Variable des Typs <code>DWORD</code> für die Rahmenfarbe, zum Beispiel <code>prgA.dwBorderColor</code>
• Füllfarbe	IEC-Variable des Typs <code>DWORD</code> für die Füllfarbe, zum Beispiel <code>prgA.dwFillColor</code>
Alarmzustand	Farbvariablen des Elements während eines Alarms
• Rahmenfarbe	IEC-Variable des Typs <code>DWORD</code> für die Rahmenfarbe, zum Beispiel <code>prgA.dwBorderColorAlarm</code>
• Füllfarbe	IEC-Variable des Typs <code>DWORD</code> für die Rahmenfarbe, zum Beispiel <code>prgA.dwFillColorAlarm</code>

Aussehensvariablen

Anzuwenden bei einer **dynamischen** Definition des Aussehens der Kontur und der Füllung des Elements. Statische Definitionen werden in **Aussehen** festgelegt.

Eigenschaft	Beschreibung Wert
Linienstärke	Eingabe einer Integervariablen, die die Linienstärke des Elements in Pixel definiert. Dies überschreibt den fixen Wert, der in Aussehen ► Linienstärke festgelegt wurde.
Füllart	Eingabe einer DWORD-Variablen, die die Füllart des Elements definiert. Die Farbe, die bei Farbvariablen angegeben ist, kann angezeigt oder ignoriert werden: • Variablenwert = 0: gefüllt • Variablenwert > 0 : unsichtbar, das heißt keine Füllung ist sichtbar Dies überschreibt den fixen Wert, der in Aussehen ► Füllart festgelegt wurde.
Linienart	Eingabe einer DWORD-Variablen, die die Kontur der Außenlinie festlegt. <i>Folgende Werte entsprechen folgenden Linienarten:</i> • 0: Durchgezogen • 1: Strich • 2: Punkt • 3: Strich Punkt • 4: Strich Punkt Punkt • 8: Unsichtbar. Das heißt, das Element wird ohne Außenlinie dargestellt.

**Elementeigenschaft
'Zustandsvariablen'**

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Elementeigenschaft 'Eingabekonfiguration'

Hier kann definiert werden, welche Folgeaktionen ausgeführt werden, wenn der Benutzer im Online-Betrieb eine Eingabe im Element vornimmt.

Eingabeereignisse, die ein Element haben kann, sind:

- OnDialogClosed
- OnMouseClicked
- OnMouseDown
- OnMouseEnter
- OnMouseLeave
- OnMouseMove
- OnMouseUp

Mit Klick auf **Konfigurieren....** wird der Dialog [Kap. "Eingabekonfiguration" auf Seite 849](#) geöffnet, der ermöglicht, dass Folgeaktionen zugewiesen werden. Man klickt in das Feld neben der zu konfigurierenden Eingabe. Für jede selektierte Folgeaktion wird ein Eintrag in der Eigenschaftentabelle erzeugt. Jede Eingabe kann eine beliebige zugeordnete Anzahl an Folgeaktionen haben.

OnDialogClosed	Dieses Ereignis tritt ein, wenn innerhalb der Visualisierung eines der für Benutzereingaben vorher geöffneten Dialogfenster geschlossen wird.
OnMouseClicked	Dieses Mausereignis tritt ein, wenn der Cursor auf ein Element zeigt und ein vollständiger Mausklick (Drücken und Freigeben der Maustaste) ausgeführt wird.
OnMouseDown	Dieses Mausereignis tritt ein, wenn die Maustaste gedrückt wird, während der Cursor auf ein Element zeigt. (Drücken der Maustaste)
OnMouseEnter	Dieses Mausereignis tritt ein, wenn der Cursor auf das Element gezogen wird.
OnMouseLeave	Dieses Mausereignis tritt ein, wenn der Cursor das Element verlässt.
OnMouseMove	Dieses Mausereignis tritt ein, wenn der Cursor innerhalb des Elements bewegt wird.
OnMouseUp	Dieses Mausereignis tritt ein, wenn die Maustaste losgelassen wird. Das vorher ausgeführte Drücken der Maustaste geschieht dabei allerdings außerhalb des Elements.

Eingabekonfiguration: Tasten

Mit **Tasten** kann konfiguriert werden, dass beim Eintreten des Ereignisses "Tasten" der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier kann eine boolesche Variable zugewiesen werden, deren Wert TRUE wird, wenn die Maustaste gedrückt wird, während der Cursor auf das Element zeigt. Der Wert wird wieder FALSE, wenn die Maustaste wieder losgelassen wird.
FALSE tasten	Wenn diese Option aktiviert ist, wird das oben beschriebene Verhalten für die eingetragene Variable umgekehrt. Das heißt, wenn die Maustaste gedrückt wird, wird der Variablenwert auf FALSE gesetzt, wenn die Taste wieder losgelassen wird, wird der Variablenwert auf TRUE gesetzt.
Beim Betreten tasten, falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert: Wird bei gedrückter Maustaste der Elementbereich verlassen, wird die Variable FALSE und wird wieder TRUE, wenn in den Elementbereich zurückgekehrt wird. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" (also an das Element gebunden) ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Umschalten

Mit **Umschalten** kann konfiguriert werden, dass bei diesem Ereignis der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier ist eine boolesche Variable zuzuweisen. Die Eingabehilfe kann über Schaltfläche  geöffnet werden. Der Wert der Variablen wird bei jedem Mausklick auf das Element zwischen TRUE und FALSE umgeschaltet. Wird der Cursor bei gedrückter Maustaste aus dem Element herausbewegt, dann erfolgt kein Toggeln. Damit kann die begonnene Umschaltteingabe abgebrochen werden.
Umschalten beim Loslassen falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert. Wird bei gedrückter Maustaste der Elementbereich verlassen und dann die Maustaste losgelassen, wird die Variable umgeschaltet. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Hotkey

Mit **Hotkey** kann eine Taste, auch mit Zusatzaste, definiert und mit einer bestimmten Nachfolgeaktion (MouseDown, MouseUp) verknüpft werden, die bei einem Ereignis 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 unter Registerkarte 'Tastaturkonfiguration' der Visualisierung verwaltet ([Kap. 20.3.4 "Registerkarte 'Tastaturkonfiguration'" auf Seite 839](#)). Änderungen werden immer zwischen diesem und dem Elementeigenschaften-Editor synchronisiert.

Eigenschaft	Beschreibung Wert
Taste	Zuweisen einer Taste. Eine Auswahlliste stellt alle aktuell unterstützten Tasten bereit wie zum Beispiel <M>.
Eingabe(n)	Definieren des Ereignisses, das ausgeführt werden soll, wenn die Taste beziehungsweise Taste und Modifizierer verwendet werden. <i>Mögliche Werte, die in einer Auswahlliste bereitstehen:</i> - Keine Aktion - MouseDown-Aktion: OnMouseDown-Aktion, wenn die Taste gedrückt wird. - MouseUp-Aktion: OnMouseUp-Aktion, wenn die Taste freigelassen wird. - MouseDown/MouseUp-Aktion: OnMouseUp- und OnMouseDown-Aktion, wenn die Taste gedrückt oder losgelassen wird.
Umschalten	Wenn diese Option aktiviert wird, muss die Taste mit <Umschalt> kombiniert verwendet werden
Steuerung	Wenn diese Option aktiviert wird, muss die Taste mit <Strg> kombiniert verwendet werden
Alt	Wenn diese Option aktiviert wird, muss die Taste mit < Alt> kombiniert verwendet werden

Element 'Linie'

Dieses Element ist eine einfache Linie.

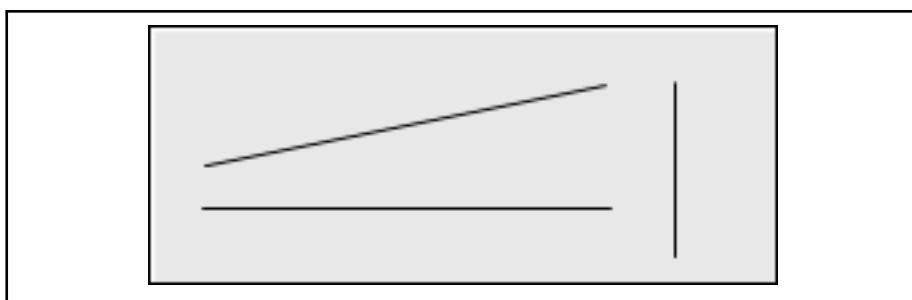


Abb. 20-37: Linie

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Linie

Elementeigenschaft 'Position' Eine Linie wird mit zwei Punkten definiert: Punkt [0] bis [1].

Punkte • [<n>] n aus {0, 1} – X – Y	• [<n>] – X: Horizontale Position in Pixel. – Y: Vertikale Position in Pixel.

Zentrum Mit Bearbeiten des Wertes wird gleichzeitig das zugehörige Element  im Editor verschoben.

Eigenschaft	Beschreibung Wert
X	X-Position des Rotationspunkts des Elements
Y	Y-Position des Rotationspunkts des Elements

Farben Eine Farbe wird mittels Hexzahl definiert, die sich aus Rot/Grün/Blau (RGB) Anteilen zusammensetzt. Für jede dieser drei Farben sind 256 (0-255) Werte verfügbar. Hier sind die Einstellungen **statisch**. Die Farbe kann aus einer Auswahlliste oder dem Farbauswahldialog, der über Button  öffnet, gewählt werden. Beispiel:  0; 128; 64

Eigenschaft	Beschreibung Wert
Farbe	Hier ist eine Farbe, mit der das Element im Normalzustand gefüllt wird, auszuwählen. Dies ist der Fall, wenn die Variable in Farbvariablen ► Farbumschlag FALSE ist. Mit Editieren des Werts ändert das Element gleichzeitig seine Farbe.
Alarmfarbe	Auszuwählen ist eine Farbe, mit der das Element gefüllt wird, wenn das Element in Alarmzustand ist. Dies ist der Fall, wenn die Variable in Farbvariablen ► Farbumschlag TRUE ist.

Aussehen

Eigenschaft	Beschreibung Wert
Linienstärke	Stärke der Rahmenlinie in Pixel. 0 kodiert das Gleiche wie 1 und setzt die Linienstärke auf 1 Pixel. Soll kein Rahmen verwendet werden, muss Linienart auf unsichtbar setzen werden.
Linienart	<i>Auszuwählen ist eine der Linienarten angewendet bei der Umrisslinie:</i> • Durchgezogen • Striche • Punkte • Strich Punkt • Strich Punkt Punkt • Unsichtbar: Dann ist die Umrisslinie unsichtbar.

Elementeigenschaft 'Texte' Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Text	Geben Sie einen Text ein. Er wird verwendet um das Element zu beschriften. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Textvariable zugewiesen.
Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Tooltip-Variable zugewiesen.

Texteigenschaften Diese Eigenschaften werden für eine statische Definition der Schriftart verwendet.

Eigenschaft	Beschreibung Wert
Horizontale Ausrichtung	<i>Definieren der horizontalen Ausrichtung des Textes durch Auswahl aus:</i> • Links • Zentriert • Rechts
Vertikale Ausrichtung	<i>Definieren der vertikalen Ausrichtung des Textes durch Auswahl aus:</i> • Oben • Zentriert • Unten
Schriftart	<i>Auswahl aus vordefinierten Schriftarten:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet sich der Dialog für benutzerdefinierte Eigenschaften der Schriftart.
Farbe Schriftart	Auszuwählen ist eine Schriftartfarbe für das Element. Entweder aus der Auswahlliste oder durch Setzen im sich öffnenden Dialog, wenn  geklickt wird.

Elementeigenschaft 'Absolute Bewegung' Das Element kann bewegt werden, indem die X- und Y-Position (Pixel) der linken oberen Ecke des Elements durch eine Integer-Variable verändert wird.

Bewegung • X • Y	• X: Hier ist eine Integer-Variable einzugeben, deren Wert die aktuelle X-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in X-Richtung zu verschieben (ein positiver Wert verschiebt das Element von links nach rechts). • Y: Eingabe einer Integer-Variablen, die die aktuelle Y-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in Y-Richtung zu verschieben (ein positiver Wert verschiebt das Element von oben nach unten).
Rotation	Eingabe einer Integer-Variablen, deren Wert den Winkel (Winkelgrade) definiert, um den das Element um den Rotationspunkt Zentrum  rotiert. positiver Wert = mathematisch negativ = im Uhrzeigersinn Zu beachten ist, dass das Element, im Gegensatz zum Verhalten bei Innere Rotation, selber nicht rotiert.

Skalierung	Eingabe einer Integer-Variablen, deren Wert den aktuellen Skalenwert (Prozent) definiert. 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  .
Innere Rotation	Die hier eingetragene Integer-Variable definiert den Winkel (Winkelgrade), um den das Element um seinen Rotationspunkt gedreht wird: Zentrum  . positiver Wert = mathematisch negativ = im Uhrzeigersinn. Im Gegensatz zu Rotation rotiert das Element tatsächlich.

Elementeigenschaft 'Relative Bewegung' Das Element kann relative zu seiner fixen Position definiert in **Position** verschoben werden.

Bewegung Punkt [<n>] mit n aus {0, 1}	X: Eingabe einer Variablen, deren Wert die Anzahl der Pixel angibt um die der Punkt[0]/Punkt[1] in X-Richtung verschoben wird Y: Eingabe einer Variablen, deren Wert die Anzahl der Pixel angibt um die der Punkt[0]/Punkt[1] in Y-Richtung verschoben wird

Elementeigenschaft 'Textvariablen' Sie können Text, der in einer Variablen gespeichert ist, anzeigen. Fügen Sie dazu zuerst eine Formatierungssequenz in dem Text hinzu, der unter der Eigenschaft **Texte** definiert ist. Weisen Sie dann hier eine Variable hinzu. Im Online-Betrieb wird die Formatierungssequenz durch den Inhalt der Variablen ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Beispiel:

Beispiel %f

Geben Sie **Ergebnis: %2.5f** in der Eigenschaft **Text** ein. Geben Sie **rValue** in der Eigenschaft **Textvariable** ein. Es muss eine definierte IEC-Variablen sein. Dann wird im Online-Betrieb das Element mit **Ergebnis: 12.12345** beschriftet, wenn **rValue = 12.12345** ist.

Textvariable	Geben Sie eine Variable (Typ ist Standardtyp) ein, der die Information enthält, die sie angezeigt haben wollen. Der Typ muss konform mit der Formatierungssequenz in Texte ► Text sein.
Tooltipvariable	Geben Sie eine STRING-Variable ein, die den ToOLTIPtext enthält, der angezeigt werden soll. Die Eingabe in Texte ► Tooltip muss ein %s enthalten.

Dynamische Texte Diese Parameter dienen der Definition von dynamischen Texten, die aus Textlisten stammen (dies erlaubt Sprachumschaltung) ([Kap. 18.2.15 "Menü 'Textliste'" auf Seite 727](#)).

Eigenschaft	Beschreibung Wert
Textliste	Name der Textliste (String) wie im POU- oder Geräte-Fenster verwendet; z.B. <code>errorlist</code>
Textindex	Index (ID) des Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. <code>'M1'</code> oder String-Variable, z.B. <code>error_22</code> .
Tooltipindex	Index (ID) des Tooltip-Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. <code>'T1'</code> oder String-Variable, z.B. <code>error_Tooltip_22</code> .

Elementeigenschaft 'Schriftartvariablen' Schriftartvariablen werden bei dynamischen Schriftdefinitionen des Elementtextes über Projektvariablen verwendet. Statischen Definitionen werden unter **Texteigenschaften** ► **Schriftart** konfiguriert.

Schriftname	Hier ist eine Variable Typ <code>STRING</code> anzugeben, die den Schriftnamen enthält, der für die Beschriftung des Elements verwendet werden soll. (Namensangabe wie im Standard-Schrifttypen-Dialog). Beispiel: <code>PlcProg.stFontVar := 'Arial';</code>
Größe	Eingabe einer Variablen Typ <code>INT</code> , die die Größe des Elementtextes in Pixel enthält, so wie im Standarddialog Schriftart. Beispiel: <code>PlcProg.iFontHeight := 16;</code>
Flags	Eingabe einer Variable Typ <code>DWORD</code> , die mit Hilfe der unten genannten Flag-Werte die Schriftart-Darstellung festlegt. Eine kombinierte Definition kann erreicht werden, indem die jeweiligen Flag-Werte addiert werden und die Summe angegeben wird. • 1: kursiv • 2: fett • 4: unterstrichen • 8: gelöscht Beispiel für einen Text, der fett und unterstrichen dargestellt wird: <code>PlcProg.dwFontType := 6;</code>
Zeichensatz	Der Zeichensatz, der für die Schriftart verwendet werden soll, kann über die Standard-Zeichensatznummer definiert werden. Mit einer <code>DWORD</code> -Variablen kann diese Nummer angegeben werden (siehe auch die Skript -Definition im Standard-Schriftartdialog)
Farbe	Eingabe einer <code>DWORD</code> -Variable, in der die Farbe des Elementtextes festgelegt wird.
Flags für Textausrichtung	Variable mit ganzzahligem Datentyp, zum Beispiel <code>PlcProg.dwTextAlignment</code> . Die Visualisierung wertet zur Laufzeit die Variable aus und richtet dementsprechend den Text aus. <i>Textausrichtung</i> • 0: oben links • 1: horizontal zentriert • 2: rechts • 4: vertikal zentriert • 8: unten

Elementeigenschaft 'Farbvariablen'

Die Elementeigenschaft ermöglicht die dynamische Umschaltung der Elementfarbe mit Hilfe von IEC-Variablen für Steuerung und Farben.

Die IEC-Variable für Farbe enthält diese als hexadezimaler Farbdefinition, die die Rot-/Grün-/Blauanteile der Farbe im RGB-Farbraum codiert: `16#00RRGGBB`. Dabei ist für jeden Farbanteil ein Byte für 256 Farbabstufungen (von 0-255) reserviert. 0 bedeutet dabei, dass kein Farbanteil enthalten ist. 255 bedeutet 100% Farbanteil. Das führende Byte ist immer `16#00`.

Beispiel:

Beispiel Farbvariable

```
dwFillColor := 16#008FE03F;
```

Codierung: Rotanteil: 8F, Grünanteil: E0, Blauanteil: 3F

- grau: `dwGrey := 16#00888888`
- rot: `dwRed := 16#00FF0000`
- schwarz: `dwBlack := 16#00000000`

Farbumschlag	<i>Steuervariable</i> - als IEC-Variable Typ <code>BOOL</code>, zum Beispiel <code>prgA.bToggleColor</code> Zur Laufzeit steuert die Visualisierung dynamisch die Farbdarstellung des Elements <code>TRUE</code>: Element wird dargestellt in Farbe, zum Beispiel in grau <code>FALSE</code>: Element wird dargestellt in Alarmfarbe, zum Beispiel in rot - oder als Platzhalter <Umschalten-/Tastenvariable> Für den Farbumschlag ist dabei keine eigene Variable zugewiesen. Stattdessen wird der Farbumschlag mit der Variablen gesteuert, die in Eingabekonfiguration, Umschalten oder Tasten zugewiesen ist. Voraussetzung ist also, dass dort eine Steuervariable konfiguriert ist. Wenn sowohl in Umschalten als auch in Tasten eine Steuervariable konfiguriert ist, wird die in Tasten verwendet. Sie können den Platzhalter entfernen, indem Sie <Umschalten-/Tastenvariable> selektieren und <Entf> wählen. Danach erscheint  im Wertefeld.
Klick auf  im Wertefeld von Farbumschlag	IndraLogic fügt den Platzhalter <Umschalten-/Tastenvariable> ein.
Farbe	IEC-Variable Typ <code>DWORD</code> für die Farbe, zum Beispiel <code>prgA.dwColor</code>
Alarmfarbe	IEC-Variable Typ <code>DWORD</code> für die Alarmfarbe, zum Beispiel <code>prgA.dwAlarmColor</code>

**Elementeigenschaft
'Zustandsvariablen'**

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert <code>TRUE</code> , dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert <code>TRUE</code>, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Elementeigenschaft 'Linienstärkenvariable'

Dynamische Definition der Dicke eines Linienelements mittels Variable

Ganzzahlwert	Eingabe einer Integervariablen, die die Linienstärke des Elements in Pixel definiert. Dies überschreibt den fixen Wert, der in Aussehen ► Linienstärke festgelegt ist. Zu berücksichtigen ist, dass 0 das Gleiche wie 1 kodiert und die Linienstärke dabei auf 1 Pixel gesetzt wird.

Elementeigenschaft 'Linienartvariable'

Dynamische Definition der Linienart mittels Variable

Ganzzahlwert	Eingabe einer Variablen mit ganzzahligem Datentyp. Der Wert definiert die Linienart. <i>Folgende Variablenwerte werden unterstützt:</i> • Durchgezogen • Striche • Punkte • Strich Punkt • Strich Punkt Punkt • Unsichtbar: Dann ist die Linie unsichtbar. Dies überschreibt den fixen Wert, der in Aussehen ► Linienart festgelegt ist.

Elementeigenschaft 'Eingabekonfiguration' Hier kann definiert werden, welche Folgeaktionen ausgeführt werden, wenn der Benutzer im Online-Betrieb eine Eingabe im Element vornimmt.

Eingabeereignisse, die ein Element haben kann, sind:

- OnDialogClosed
- OnMouseClicked
- OnMouseDown
- OnMouseEnter
- OnMouseLeave
- OnMouseMove
- OnMouseUp

Mit Klick auf **Konfigurieren....** wird der Dialog [Kap. "Eingabekonfiguration" auf Seite 849](#) geöffnet, der ermöglicht, dass Folgeaktionen zugewiesen werden. Man klickt in das Feld neben der zu konfigurierenden Eingabe. Für jede selektierte Folgeaktion wird ein Eintrag in der Eigenschaftentabelle erzeugt. Jede Eingabe kann eine beliebige zugeordnete Anzahl an Folgeaktionen haben.

OnDialogClosed	Dieses Ereignis tritt ein, wenn innerhalb der Visualisierung eines der für Benutzereingaben vorher geöffneten Dialogfenster geschlossen wird.
OnMouseClicked	Dieses Mausereignis tritt ein, wenn der Cursor auf ein Element zeigt und ein vollständiger Mausklick (Drücken und Freigeben der Maustaste) ausgeführt wird.
OnMouseDown	Dieses Mausereignis tritt ein, wenn die Maustaste gedrückt wird, während der Cursor auf ein Element zeigt. (Drücken der Maustaste)
OnMouseEnter	Dieses Mausereignis tritt ein, wenn der Cursor auf das Element gezogen wird.
OnMouseLeave	Dieses Mausereignis tritt ein, wenn der Cursor das Element verlässt.
OnMouseMove	Dieses Mausereignis tritt ein, wenn der Cursor innerhalb des Elements bewegt wird.
OnMouseUp	Dieses Mausereignis tritt ein, wenn die Maustaste losgelassen wird. Das vorher ausgeführte Drücken der Maustaste geschieht dabei allerdings außerhalb des Elements.

Eingabekonfiguration: Tasten

Mit **Tasten** kann konfiguriert werden, dass beim Eintreten des Ereignisses "Tasten" der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier kann eine boolesche Variable zugewiesen werden, deren Wert TRUE wird, wenn die Maustaste gedrückt wird, während der Cursor auf das Element zeigt. Der Wert wird wieder FALSE, wenn die Maustaste wieder losgelassen wird.
FALSE tasten	Wenn diese Option aktiviert ist, wird das oben beschriebene Verhalten für die eingetragene Variable umgekehrt. Das heißt, wenn die Maustaste gedrückt wird, wird der Variablenwert auf FALSE gesetzt, wenn die Taste wieder losgelassen wird, wird der Variablenwert auf TRUE gesetzt.
Beim Betreten tasten, falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert: Wird bei gedrückter Maustaste der Elementbereich verlassen, wird die Variable FALSE und wird wieder TRUE, wenn in den Elementbereich zurückgekehrt wird. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" (also an das Element gebunden) ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Umschalten

Mit **Umschalten** kann konfiguriert werden, dass bei diesem Ereignis der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier ist eine boolesche Variable zuzuweisen. Die Eingabehilfe kann über Schaltfläche  geöffnet werden. Der Wert der Variablen wird bei jedem Mausklick auf das Element zwischen TRUE und FALSE umgeschaltet. Wird der Cursor bei gedrückter Maustaste aus dem Element herausbewegt, dann erfolgt kein Toggeln. Damit kann die begonnene Umschaltteingabe abgebrochen werden.
Umschalten beim Loslassen falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert. Wird bei gedrückter Maustaste der Elementbereich verlassen und dann die Maustaste losgelassen, wird die Variable umgeschaltet. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Hotkey

Mit **Hotkey** kann eine Taste, auch mit Zusatz Taste, definiert und mit einer bestimmten Nachfolgeaktion (MouseDown, MouseUp) verknüpft werden, die bei einem Ereignis 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 Tastatenkonfiguration für ein Element wird auch unter Registerkarte 'Tastatenkonfiguration' der Visualisierung verwaltet ([Kap. 20.3.4 "Registerkarte 'Tastaturkonfiguration'" auf Seite 839](#)). Änderungen werden immer zwischen diesem und dem Elementeigenschaften-Editor synchronisiert.

Eigenschaft	Beschreibung Wert
Taste	Zuweisen einer Taste. Eine Auswahlliste stellt alle aktuell unterstützten Tasten bereit wie zum Beispiel <M>.
Eingabe(n)	Definieren des Ereignisses, das ausgeführt werden soll, wenn die Taste beziehungsweise Taste und Modifizierer verwendet werden. <i>Mögliche Werte, die in einer Auswahlliste bereitstehen:</i> - Keine Aktion - MouseDown-Aktion: OnMouseDown-Aktion, wenn die Taste gedrückt wird. - MouseUp-Aktion: OnMouseUp-Aktion, wenn die Taste freigelassen wird. - MouseDown/MouseUp-Aktion: OnMouseUp- und OnMouseDown-Aktion, wenn die Taste gedrückt oder losgelassen wird.
Umschalten	Wenn diese Option aktiviert wird, muss die Taste mit <Umschalt> kombiniert verwendet werden
Steuerung	Wenn diese Option aktiviert wird, muss die Taste mit <Strg> kombiniert verwendet werden
Alt	Wenn diese Option aktiviert wird, muss die Taste mit < Alt> kombiniert verwendet werden

Element 'Polygon', 'Linienzug', 'Bézierkurve'

Polygon, Linienzug und Bézierkurve sind vom gleichen Elementtyp. Sie können nur durch Ändern der Eigenschaft **Elementtyp** gegenseitig konvertiert werden.

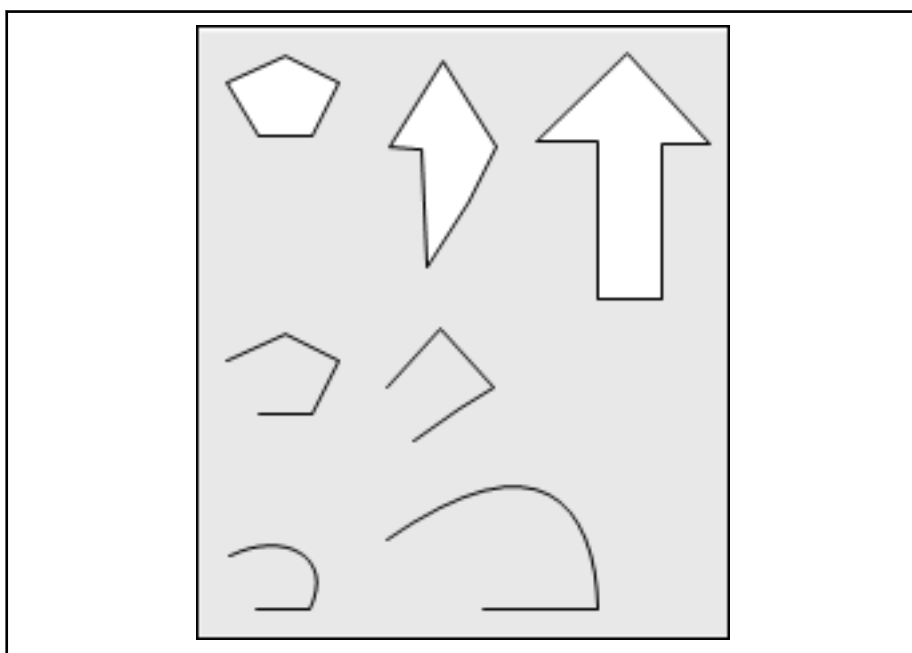


Abb. 20-38: Polygon, Linienzug, Bézierkurve

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Element spezifische Mauseingabe:

- Sie können das Element mittels Drag & Drop in den Editor ziehen. Dabei wird das Element mit fünf Punkten gezeichnet: Punkte [0] bis [4]. Zusätzliche Punkte können folgendermaßen eingefügt werden: Selektieren Sie das Element und setzen Sie den Fokus auf einen Eckpunkt (kleines



Quadrat auf der Umrisslinie). Dabei ändert sich die Cursor-Form. Wenn Sie nun <Strg> gedrückt halten, während Sie auf den fokussierten Punkt klicken, wird ein weiterer Punkt erzeugt. Sie können einen Punkt löschen, indem Sie die <Umschalttaste> + <Strg> gedrückt halten, während Sie auf den fokussierten Punkt klicken.

- Sie können das Element im Werkzeugbereich anwählen und anschließend mehrfach in den Editor klicken. Dabei wird eine Verbindungslinie von einem Punkt zum anderen gezeichnet. Beenden Sie das Element mit einem Doppelklick links oder Einfachklick rechts. Wenn Sie ein anderes Element im Werkzeugbereich anwählen, wird der Linienzug ebenfalls beendet.

In **Eigenschaften** werden gleichzeitig die Änderungen berücksichtigt und **Position ► Punkte aktualisiert**

Benutzereingabe beim Editieren der Elementeigenschaften*Eigenschaften definieren die Eigenart eines Visualisierungselements:*

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft ein Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
-  Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	<i>Auszuwählen ist einer aus:</i> • Polygon • Linienzug • Bézierkurve

Elementeigenschaft 'Position'

Solch ein Element kann mit einer beliebigen Anzahl an Punkten, die Ausschnitt und Form des Elements bestimmen, definiert werden. Standard sind fünf Punkte: Punkte [0] bis [4]. Ein Punkt ist bestimmt durch X/Y-Koordinaten in Pixel.

Punkte • [<n>] - X - Y	• [<n>]: - X: Horizontale Position in Pixel. - Y: Vertikale Position in Pixel.

Zentrum

Mit Bearbeiten des Wertes wird gleichzeitig das zugehörige Element  im Editor verschoben.

Eigenschaft	Beschreibung Wert
X	X-Position des Rotationspunkts des Elements
Y	Y-Position des Rotationspunkts des Elements

Farben

Eine Farbe wird mittels Hexzahl definiert, die sich aus Rot/Grün/Blau (RGB) Anteilen zusammensetzt. Für jede dieser drei Farben sind 256 (0-255) Werte verfügbar. Hier sind die Einstellungen **statisch**. Die Farbe kann aus einer Auswahlliste oder dem Farbauswahldialog, der über Button  öffnet, gewählt werden. Beispiel:  0; 128; 64

Eigenschaft	Beschreibung Wert
Normalzustand • Rahmenfarbe • Füllfarbe	Auszuwählen ist eine Rahmen- und Füllfarbe für Normalzustand. Im Fall, dass die Variable definiert in Farbvariablen ► Farbumschlag FALSE ist, ist das Element in Normalzustand.
Alarmzustand • Rahmenfarbe • Füllfarbe	Auszuwählen ist eine Rahmen- und Füllfarbe, mit der das Element im Alarmzustand dargestellt wird. Dies ist der Fall, wenn die Variable in Farbvariablen ► Farbumschlag TRUE ist.

Farbverlauf verwenden	Standardmäßig ist die Checkbox deaktiviert. Falls sie aktiviert wird, wird das zugehörige Element mit Farbverlauf gezeichnet.
Farbverlaufsauswahl	Dialog öffnet. Kap. "Farbverlaufeditor" auf Seite 862

Aussehen

Eigenschaft	Beschreibung Wert
Linienstärke	Stärke der Rahmenlinie in Pixel. 0 kodiert das Gleiche wie 1 und setzt die Linienstärke auf 1 Pixel. Wird kein Rahmen gewünscht, muss man Linienart auf <code>unsichtbar</code> setzen.
Füllart	Auszuwählen ist eines dieser Füllarten angewendet in Füllfarbe • <code>Ausgefüllt</code>: Die Füllfarbe ist sichtbar • <code>Unsichtbar</code>: Die Füllfarbe ist unsichtbar
Linienart	<i>Auszuwählen ist eine der Linienarten angewendet bei der Umrisslinie:</i> • <code>Durchgezogen</code> • <code>Striche</code> • <code>Punkte</code> • <code>Strich Punkt</code> • <code>Strich Punkt Punkt</code> • <code>Unsichtbar</code>: Umrisslinie ist nicht sichtbar.

Elementeigenschaft 'Texte' Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Text	Geben Sie einen Text ein. Er wird verwendet um das Element zu beschriften. Der Text kann eine Formatierungssequenz, wie zum Beispiel %, enthalten. Die zugehörige Variable ist in Textvariablen ► Textvariable zugewiesen.
Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %, enthalten. Die zugehörige Variable ist in Textvariablen ► Tooltip-Variable zugewiesen.

Texteigenschaften Diese Eigenschaften werden für eine statische Definition der Schriftart verwendet.

Eigenschaft	Beschreibung Wert
Horizontale Ausrichtung	<i>Definieren der horizontalen Ausrichtung des Textes durch Auswahl aus:</i> • <code>Links</code> • <code>Zentriert</code> • <code>Rechts</code>
Vertikale Ausrichtung	<i>Definieren der vertikalen Ausrichtung des Textes durch Auswahl aus:</i> • <code>Oben</code> • <code>Zentriert</code> • <code>Unten</code>

Eigenschaft	Beschreibung Wert
Schriftart	<i>Auswahl aus vordefinierten Schriftarten:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet sich der Dialog für benutzerdefinierte Eigenschaften der Schriftart.
Farbe Schriftart	Auszuwählen ist eine Schriftartfarbe für das Element. Entweder aus der Auswahlliste oder durch Setzen im sich öffnenden Dialog, wenn  geklickt wird.

Elementeigenschaft 'Absolute Bewegung'

Das Element kann bewegt werden, indem die X- und Y-Position (Pixel) der linken oberen Ecke des Elements durch eine Integer-Variable verändert wird.

Bewegung • X • Y	• X: Hier ist eine Integer-Variable einzugeben, deren Wert die aktuelle X-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in X-Richtung zu verschieben (ein positiver Wert verschiebt das Element von links nach rechts). • Y: Eingabe einer Integer-Variablen, die die aktuelle Y-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in Y-Richtung zu verschieben (ein positiver Wert verschiebt das Element von oben nach unten).
Rotation	Eingabe einer Integer-Variablen, deren Wert den Winkel (Winkelgrade) definiert, um den das Element um den Rotationspunkt Zentrum  rotiert. positiver Wert = mathematisch negativ = im Uhrzeigersinn Zu beachten ist, dass das Element, im Gegensatz zum Verhalten bei Innere Rotation, selber nicht rotiert.
Skalierung	Eingabe einer Integer-Variablen, deren Wert den aktuellen Skalenwert (Prozent) definiert. 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 .
Innere Rotation	Die hier eingetragene Integer-Variable definiert den Winkel (Winkelgrade), um den das Element um seinen Rotationspunkt gedreht wird: Zentrum . positiver Wert = mathematisch negativ = im Uhrzeigersinn. Im Gegensatz zu Rotation rotiert das Element tatsächlich.

Elementeigenschaft 'Dynamische Punkte'

Dynamische Definition für die Punkte, die das Element definieren.

Array von Punkten	Hier ist eine Variable einzugeben, die auf einen Array der Struktur <code>VisuElems.VisuStructPoint</code> zeigt. Die <code>VisuElems.VisuStructPoint</code>-Komponenten <code>iX</code> und <code>iY</code> enthalten die X/Y-Koordinaten eines Punktes. Der Array enthält alle Punkte des Elements und die Struktur kann dynamisch durch eine Projektvariable gefüllt werden. Die Anzahl der Elementpunkte muss explizit in Anzahl Punkte deklariert werden. Beispiel: <code>pPoints : POINTER TO ARRAY[0..100] OF VisuElems.VisuStructPoint;</code>
Anzahl Punkte	Hier ist eine Integervariable zuzuweisen, die die Anzahl der Punkte des Elements definiert. Beispiel: <code>iCount : INT:=24;</code> Das Element benötigt 24 einzelne Punkte. Diese Festlegung ist nötig, weil die Definition der einzelnen Punkte über einen Zeiger erfolgt und dies nicht erlaubt, die Anzahl der Punkte zu steuern.

Elementeigenschaft 'Textvariablen'

Sie können Text, der in einer Variablen gespeichert ist, anzeigen. Fügen Sie dazu zuerst eine Formatierungssequenz in dem Text hinzu, der unter der Eigenschaft **Texte** definiert ist. Weisen Sie dann hier eine Variable hinzu. Im Online-Betrieb wird die Formatierungssequenz durch den Inhalt der Variablen ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Beispiel:

Beispiel %f

Geben Sie `Ergebnis: %2.5f` in der Eigenschaft **Text** ein. Geben Sie `rValue` in der Eigenschaft **Textvariable** ein. Es muss eine definierte IEC-Variablen sein. Dann wird im Online-Betrieb das Element mit `Ergebnis: 12.12345` beschriftet, wenn `rValue = 12.12345` ist.

Textvariable	Geben Sie eine Variable (Typ ist Standardtyp) ein, der die Information enthält, die sie angezeigt haben wollen. Der Typ muss konform mit der Formatierungssequenz in Texte ► Text sein.
Tooltipvariable	Geben Sie eine STRING-Variablen ein, die den Tooltiptext enthält, der angezeigt werden soll. Die Eingabe in Texte ► Tooltip muss ein <code>%s</code> enthalten.

Dynamische Texte

Diese Parameter dienen der Definition von dynamischen Texten, die aus Textlisten stammen (dies erlaubt Sprachumschaltung) ([Kap. 18.2.15 "Menü 'Textliste'" auf Seite 727](#)).

Eigenschaft	Beschreibung Wert
Textliste	Name der Textliste (String) wie im POU- oder Geräte-Fenster verwendet; z.B. <code>errorlist</code>
Textindex	Index (ID) des Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. <code>'M1'</code> oder String-Variablen, z.B. <code>error_22</code> .
Tooltipindex	Index (ID) des Tooltip-Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. <code>'T1'</code> oder String-Variablen, z.B. <code>error_Tooltip_22</code> .

Elementeigenschaft 'Schriftartvariablen'

Schriftartvariablen werden bei dynamischen Schriftdefinitionen des Elementtextes über Projektvariablen verwendet. Statischen Definitionen werden unter **Texteigenschaften** ► **Schriftart** konfiguriert.

Schriftname	Hier ist eine Variable Typ <code>STRING</code> anzugeben, die den Schriftnamen enthält, der für die Beschriftung des Elements verwendet werden soll. (Namensangabe wie im Standard-Schrifttypen-Dialog). Beispiel: <code>PlcProg.stFontVar := 'Arial';</code>
Größe	Eingabe einer Variablen Typ <code>INT</code> , die die Größe des Elementtexts in Pixel enthält, so wie im Standarddialog Schriftart. Beispiel: <code>PlcProg.iFontHeight := 16;</code>
Flags	Eingabe einer Variable Typ <code>DWORD</code> , die mit Hilfe der unten genannten Flag-Werte die Schriftart-Darstellung festlegt. Eine kombinierte Definition kann erreicht werden, indem die jeweiligen Flag-Werte addiert werden und die Summe angegeben wird. • 1: kursiv • 2: fett • 4: unterstrichen • 8: gelöscht Beispiel für einen Text, der fett und unterstrichen dargestellt wird: <code>PlcProg.dwFontType := 6;</code>
Zeichensatz	Der Zeichensatz, der für die Schriftart verwendet werden soll, kann über die Standard-Zeichensatznummer definiert werden. Mit einer <code>DWORD</code> -Variablen kann diese Nummer angegeben werden (siehe auch die Skript -Definition im Standard-Schriftartdialog)
Farbe	Eingabe einer <code>DWORD</code> -Variable, in der die Farbe des Elementtextes festgelegt wird.
Flags für Textausrichtung	Variable mit ganzzahligem Datentyp, zum Beispiel <code>PlcProg.dwTextAlignment</code> . Die Visualisierung wertet zur Laufzeit die Variable aus und richtet dementsprechend den Text aus. <i>Textausrichtung</i> • 0: oben links • 1: horizontal zentriert • 2: rechts • 4: vertikal zentriert • 8: unten

Farben Eine Farbe wird mittels Hexzahl definiert, die sich aus Rot/Grün/Blau (RGB) Anteilen zusammensetzt. Für jede dieser drei Farben sind 256 (0-255) Werte verfügbar. Hier sind die Einstellungen **statisch**. Die Farbe kann aus einer Auswahlliste oder dem Farbauswahldialog, der über Button  öffnet, gewählt werden. Beispiel:  0; 128; 64

Eigenschaft	Beschreibung Wert
Normalzustand • Rahmenfarbe • Füllfarbe	Auszuwählen ist eine Rahmen- und Füllfarbe für Normalzustand. Im Fall, dass die Variable definiert in Farbvariablen ► Farbumschlag FALSE ist, ist das Element in Normalzustand.
Alarmzustand • Rahmenfarbe • Füllfarbe	Auszuwählen ist eine Rahmen- und Füllfarbe, mit der das Element im Alarmzustand dargestellt wird. Dies ist der Fall, wenn die Variable in Farbvariablen ► Farbumschlag TRUE ist.

Aussehensvariablen Anzuwenden bei einer **dynamischen** Definition des Aussehens der Kontur und der Füllung des Elements. Statische Definitionen werden in **Aussehen** festgelegt.

Eigenschaft	Beschreibung Wert
Linienstärke	Eingabe einer Integervariablen, die die Linienstärke des Elements in Pixel definiert. Dies überschreibt den fixen Wert, der in Aussehen ► Linienstärke festgelegt wurde.
Füllart	Eingabe einer DWORD-Variablen, die die Füllart des Elements definiert. Die Farbe, die bei Farbvariablen angegeben ist, kann angezeigt oder ignoriert werden: • Variablenwert = 0: gefüllt • Variablenwert > 0 : unsichtbar, das heißt keine Füllung ist sichtbar Dies überschreibt den fixen Wert, der in Aussehen ► Füllart festgelegt wurde.
Linienart	Eingabe einer DWORD-Variablen, die die Kontur der Außenlinie festlegt. <i>Folgende Werte entsprechen folgenden Linienarten:</i> • 0: Durchgezogen • 1: Strich • 2: Punkt • 3: Strich Punkt • 4: Strich Punkt Punkt • 8: Unsichtbar. Das heißt, das Element wird ohne Außenlinie dargestellt.

**Elementeigenschaft
'Zustandsvariablen'**

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Elementeigenschaft 'Eingabekonfiguration'

Hier kann definiert werden, welche Folgeaktionen ausgeführt werden, wenn der Benutzer im Online-Betrieb eine Eingabe im Element vornimmt.

Eingabeereignisse, die ein Element haben kann, sind:

- OnDialogClosed
- OnMouseClicked
- OnMouseDown
- OnMouseEnter
- OnMouseLeave
- OnMouseMove
- OnMouseUp

Mit Klick auf **Konfigurieren....** wird der Dialog [Kap. "Eingabekonfiguration" auf Seite 849](#) geöffnet, der ermöglicht, dass Folgeaktionen zugewiesen werden. Man klickt in das Feld neben der zu konfigurierenden Eingabe. Für jede selektierte Folgeaktion wird ein Eintrag in der Eigenschaftentabelle erzeugt. Jede Eingabe kann eine beliebige zugeordnete Anzahl an Folgeaktionen haben.

OnDialogClosed	Dieses Ereignis tritt ein, wenn innerhalb der Visualisierung eines der für Benutzereingaben vorher geöffneten Dialogfenster geschlossen wird.
OnMouseClicked	Dieses Mausereignis tritt ein, wenn der Cursor auf ein Element zeigt und ein vollständiger Mausklick (Drücken und Freigeben der Maustaste) ausgeführt wird.
OnMouseDown	Dieses Mausereignis tritt ein, wenn die Maustaste gedrückt wird, während der Cursor auf ein Element zeigt. (Drücken der Maustaste)
OnMouseEnter	Dieses Mausereignis tritt ein, wenn der Cursor auf das Element gezogen wird.
OnMouseLeave	Dieses Mausereignis tritt ein, wenn der Cursor das Element verlässt.
OnMouseMove	Dieses Mausereignis tritt ein, wenn der Cursor innerhalb des Elements bewegt wird.
OnMouseUp	Dieses Mausereignis tritt ein, wenn die Maustaste losgelassen wird. Das vorher ausgeführte Drücken der Maustaste geschieht dabei allerdings außerhalb des Elements.

Eingabekonfiguration: Tasten

Mit **Tasten** kann konfiguriert werden, dass beim Eintreten des Ereignisses "Tasten" der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier kann eine boolesche Variable zugewiesen werden, deren Wert TRUE wird, wenn die Maustaste gedrückt wird, während der Cursor auf das Element zeigt. Der Wert wird wieder FALSE, wenn die Maustaste wieder losgelassen wird.
FALSE tasten	Wenn diese Option aktiviert ist, wird das oben beschriebene Verhalten für die eingetragene Variable umgekehrt. Das heißt, wenn die Maustaste gedrückt wird, wird der Variablenwert auf FALSE gesetzt, wenn die Taste wieder losgelassen wird, wird der Variablenwert auf TRUE gesetzt.
Beim Betreten tasten, falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert: Wird bei gedrückter Maustaste der Elementbereich verlassen, wird die Variable FALSE und wird wieder TRUE, wenn in den Elementbereich zurückgekehrt wird. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" (also an das Element gebunden) ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Umschalten

Mit **Umschalten** kann konfiguriert werden, dass bei diesem Ereignis der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier ist eine boolesche Variable zuzuweisen. Die Eingabehilfe kann über Schaltfläche  geöffnet werden. Der Wert der Variablen wird bei jedem Mausklick auf das Element zwischen TRUE und FALSE umgeschaltet. Wird der Cursor bei gedrückter Maustaste aus dem Element herausbewegt, dann erfolgt kein Toggeln. Damit kann die begonnene Umschaltteingabe abgebrochen werden.
Umschalten beim Loslassen falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert. Wird bei gedrückter Maustaste der Elementbereich verlassen und dann die Maustaste losgelassen, wird die Variable umgeschaltet. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Hotkey

Mit **Hotkey** kann eine Taste, auch mit Zusatzstufe, definiert und mit einer bestimmten Nachfolgeaktion (MouseDown, MouseUp) verknüpft werden, die bei einem Ereignis 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 Tastatenkonfiguration für ein Element wird auch unter Registerkarte 'Tastatenkonfiguration' der Visualisierung verwaltet ([Kap. 20.3.4 "Registerkarte 'Tastaturkonfiguration'" auf Seite 839](#)). Änderungen werden immer zwischen diesem und dem Elementeigenschaften-Editor synchronisiert.

Eigenschaft	Beschreibung Wert
Taste	Zuweisen einer Taste. Eine Auswahlliste stellt alle aktuell unterstützten Tasten bereit wie zum Beispiel <M>.
Eingabe(n)	Definieren des Ereignisses, das ausgeführt werden soll, wenn die Taste beziehungsweise Taste und Modifizierer verwendet werden. <i>Mögliche Werte, die in einer Auswahlliste bereitstehen:</i> - Keine Aktion - MouseDown-Aktion: OnMouseDown-Aktion, wenn die Taste gedrückt wird. - MouseUp-Aktion: OnMouseUp-Aktion, wenn die Taste freigelassen wird. - MouseDown/MouseUp-Aktion: OnMouseUp- und OnMouseDown-Aktion, wenn die Taste gedrückt oder losgelassen wird.
Umschalten	Wenn diese Option aktiviert wird, muss die Taste mit <Umschalt> kombiniert verwendet werden
Steuerung	Wenn diese Option aktiviert wird, muss die Taste mit <Strg> kombiniert verwendet werden
Alt	Wenn diese Option aktiviert wird, muss die Taste mit < Alt> kombiniert verwendet werden

Element 'Kreissektor'

Der Kreissektor, der standardmäßig Im Visualisierungsektor eingefügt wird, ist ein Vollkreis mit der Mittelpunktion an dem Punkt, an dem das Element fallen gelassen wird.



Abb. 20-39: Kreissektor

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Element spezifische Mauseingabe:

- Anfangs- und Endpunkt des Kreissektors können mit Drag-and-Drop der kleinen Quadrate auf der Kreislinie im Editor verschoben werden. Die zugehörigen Eigenschaften **Anfang** und **Ende** werden gleichzeitig aktualisiert.

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Kreis Sektor

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Anfang	Hier ist der Anfang der Kreissektorlinie zu definieren.
Ende	Definiert das Ende der Kreissektorlinie.
Variable für Anfang	Eingabe einer Variablen, wenn der Anfang der Kreissektorlinie dynamisch mit Variable definiert werden soll. Für Hilfe vom Eingabeassistenten ist  zu klicken.
Variable für Ende	Eingabe einer Variablen, eventuell auch mit Hilfe des Eingabeassistenten, falls das Ende der Kreissektorlinie dynamisch mit Variable definiert werden soll.
Nur Kreisbogen anzeigen	Falls die Option aktiviert ist, wird der Kreissektor ohne Radiuslinie für Anfang und Ende dargestellt,

Zentrum Mit Bearbeiten des Wertes wird gleichzeitig das zugehörige Element  im Editor verschoben.

Eigenschaft	Beschreibung Wert
X	X-Position des Rotationspunkts des Elements
Y	Y-Position des Rotationspunkts des Elements

Farben Eine Farbe wird mittels Hexzahl definiert, die sich aus Rot/Grün/Blau (RGB) Anteilen zusammensetzt. Für jede dieser drei Farben sind 256 (0-255) Werte verfügbar. Hier sind die Einstellungen **statisch**. Die Farbe kann aus einer Auswahlliste oder dem Farbauswahldialog, der über Button  öffnet, gewählt werden. Beispiel:  0; 128; 64

Eigenschaft	Beschreibung Wert
Normalzustand - Rahmenfarbe - Füllfarbe	Auszuwählen ist eine Rahmen- und Füllfarbe für Normalzustand. Im Fall, dass die Variable definiert in Farbvariablen ▶ Farbumschlag FALSE ist, ist das Element in Normalzustand.
Alarmzustand - Rahmenfarbe - Füllfarbe	Auszuwählen ist eine Rahmen- und Füllfarbe, mit der das Element im Alarmzustand dargestellt wird. Dies ist der Fall, wenn die Variable in Farbvariablen ▶ Farbumschlag TRUE ist.

Farbverlauf verwenden	Standardmäßig ist die Checkbox deaktiviert. Falls sie aktiviert wird, wird das zugehörige Element mit Farbverlauf gezeichnet.
Farbverlaufsauswahl	Dialog öffnet. Kap. "Farbverlaufeditor" auf Seite 862

Aussehen

Eigenschaft	Beschreibung Wert
Linienstärke	Stärke der Rahmenlinie in Pixel. 0 kodiert das Gleiche wie 1 und setzt die Linienstärke auf 1 Pixel. Wird kein Rahmen gewünscht, muss man Linienart auf <code>unsichtbar</code> setzen.
Füllart	Auszuwählen ist eines dieser Füllarten angewendet in Füllfarbe - Ausgefüllt: Die Füllfarbe ist sichtbar - Unsichtbar: Die Füllfarbe ist unsichtbar
Linienart	<i>Auszuwählen ist eine der Linienarten angewendet bei der Umrisslinie:</i> - Durchgezogen - Striche - Punkte - Strich Punkt - Strich Punkt Punkt - Unsichtbar: Umrisslinie ist nicht sichtbar.



Die **Linienstärke** und die **Linienart** wirken sich auf Windows CE basierenden Systemen nicht aus.

Elementeigenschaft 'Texte'

Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Text	Geben Sie einen Text ein. Er wird verwendet um das Element zu beschriften. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Textvariable zugewiesen.
Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Tooltip-Variable zugewiesen.

Texteigenschaften

Diese Eigenschaften werden für eine statische Definition der Schriftart verwendet.

Eigenschaft	Beschreibung Wert
Horizontale Ausrichtung	<i>Definieren der horizontalen Ausrichtung des Textes durch Auswahl aus:</i> • Links • Zentriert • Rechts
Vertikale Ausrichtung	<i>Definieren der vertikalen Ausrichtung des Textes durch Auswahl aus:</i> • Oben • Zentriert • Unten
Schriftart	<i>Auswahl aus vordefinierten Schriftarten:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet sich der Dialog für benutzerdefinierte Eigenschaften der Schriftart.
Farbe Schriftart	Auszuwählen ist eine Schriftartfarbe für das Element. Entweder aus der Auswahlliste oder durch Setzen im sich öffnenden Dialog, wenn  geklickt wird.

Elementeigenschaft 'Absolute Bewegung'

Das Element kann bewegt werden, indem die X- und Y-Position (Pixel) der linken oberen Ecke des Elements durch eine Integer-Variable verändert werden. Hier werden absolute Koordinatenwerte verwendet.

Bewegung • X • Y	• X: Hier ist eine Integer-Variable einzugeben, deren Wert die aktuelle X-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in X-Richtung zu verschieben (ein positiver Wert verschiebt das Element von links nach rechts). • Y: Eingabe einer Integer-Variablen, die die aktuelle Y-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in Y-Richtung zu verschieben (ein positiver Wert verschiebt das Element von oben nach unten).
Skalierung	Eingabe einer Integer-Variablen, deren Wert den aktuellen Skalenwert (Prozent) definiert. 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  .
Innere Rotation	Die hier eingetragene Integer-Variable definiert den Winkel (Winkelgrade), um den das Element um seinen Rotationspunkt gedreht wird: Zentrum . positiver Wert = mathematisch negativ = Uhrzeigersinn. Im Gegensatz zu Rotation rotiert das Element tatsächlich.

Elementeigenschaft 'Textvariablen'

Sie können Text, der in einer Variablen gespeichert ist, anzeigen. Fügen Sie dazu zuerst eine Formatierungssequenz in dem Text hinzu, der unter der Eigenschaft **Texte** definiert ist. Weisen Sie dann hier eine Variable hinzu. Im Online-Betrieb wird die Formatierungssequenz durch den Inhalt der Variablen ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Beispiel:

Beispiel %f

Geben Sie **Ergebnis: %2.5f** in der Eigenschaft **Text** ein. Geben Sie **rValue** in der Eigenschaft **Textvariable** ein. Es muss eine definierte IEC-Variablen sein. Dann wird im Online-Betrieb das Element mit **Ergebnis: 12.12345** beschriftet, wenn **rValue = 12.12345** ist.

Textvariable	Geben Sie eine Variable (Typ ist Standarddatentyp) ein, der die Information enthält, die sie angezeigt haben wollen. Der Typ muss konform mit der Formatierungssequenz in Texte ▶ Text sein.
Tooltipvariable	Geben Sie eine STRING-Variable ein, die den Toolliptext enthält, der angezeigt werden soll. Die Eingabe in Texte ▶ Tooltip muss ein %s enthalten.

Dynamische Texte

Diese Parameter dienen der Definition von dynamischen Texten, die aus Textlisten stammen (dies erlaubt Sprachumschaltung) ([Kap. 18.2.15 "Menü 'Textliste'" auf Seite 727](#)).

Eigenschaft	Beschreibung Wert
Textliste	Name der Textliste (String) wie im POU- oder Geräte-Fenster verwendet; z.B. <code>errorlist</code>
Textindex	Index (ID) des Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. <code>'M1'</code> oder String-Variable, z.B. <code>error_22</code> .
Tooltipindex	Index (ID) des Tooltip-Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. <code>'T1'</code> oder String-Variable, z.B. <code>error_Tooltip_22</code> .

Elementeigenschaft 'Schriftartvariablen'

Schriftartvariablen werden bei dynamischen Schriftdefinitionen des Elementtextes über Projektvariablen verwendet. Statischen Definitionen werden unter **Texteigenschaften ▶ Schriftart** konfiguriert.

Schriftname	Hier ist eine Variable Typ <code>STRING</code> anzugeben, die den Schriftnamen enthält, der für die Beschriftung des Elements verwendet werden soll. (Namensangabe wie im Standard-Schrifttypen-Dialog). Beispiel: <code>PlcProg.stFontVar := 'Arial';</code>
Größe	Eingabe einer Variablen Typ <code>INT</code> , die die Größe des Elementtextes in Pixel enthält, so wie im Standarddialog Schriftart. Beispiel: <code>PlcProg.iFontHeight := 16;</code>
Flags	Eingabe einer Variable Typ <code>DWORD</code> , die mit Hilfe der unten genannten Flag-Werte die Schriftart-Darstellung festlegt. Eine kombinierte Definition kann erreicht werden, indem die jeweiligen Flag-Werte addiert werden und die Summe angegeben wird. 1: kursiv 2: fett 4: unterstrichen 8: gelöscht Beispiel für einen Text, der fett und unterstrichen dargestellt wird: <code>PlcProg.dwFontType := 6;</code>
Zeichensatz	Der Zeichensatz, der für die Schriftart verwendet werden soll, kann über die Standard-Zeichensatznummer definiert werden. Mit einer <code>DWORD</code> -Variablen kann diese Nummer angegeben werden (siehe auch die Skript -Definition im Standard-Schriftartdialog)

Farbe	Eingabe einer <code>DWORD</code> -Variable, in der die Farbe des Elementtextes festgelegt wird.
Flags für Textausrichtung	Variable mit ganzzahligem Datentyp, zum Beispiel <code>PlcProg.dwTextAlignment</code>. Die Visualisierung wertet zur Laufzeit die Variable aus und richtet dementsprechend den Text aus. <i>Textausrichtung</i> • 0: oben links • 1: horizontal zentriert • 2: rechts • 4: vertikal zentriert • 8: unten

Elementeigenschaft 'Farbvariablen'

Die Elementeigenschaft ermöglicht die dynamische Umschaltung der Elementfarben mit Hilfe von IEC-Variablen für Steuerung und Farben.

Die IEC-Variable für Farbe enthält diese als hexadezimaler Farbdefinition, die die Rot-/Grün-/Blauanteile der Farbe im RGB-Farbraum codiert: `16#00RRGGBB`. Dabei ist für jeden Farbanteil ein Byte für 256 Farbabstufungen (von 0-255) reserviert. 0 bedeutet dabei, dass kein Farbanteil enthalten ist. 255 bedeutet 100% Farbanteil. Das führende Byte ist immer `16#00`.

Beispiel:

Farbvariable

```
dwFillColor := 16#008FE03F;
```

Codierung: Rotanteil: 8F, Grünanteil: E0, Blauanteil: 3F

- grau: `dwGrey := 16#00888888`
- rot: `dwRed := 16#00FF0000`
- schwarz: `dwBlack := 16#00000000`

Farbumschlag	<i>Steuervariable</i> • als IEC-Variable des Typs <code>BOOL</code>, zum Beispiel <code>prgA.bToggleColor</code> Zur Laufzeit steuert die Visualisierung dynamisch die Farbdarstellung des Elements – <code>TRUE</code>: Element in Normalzustand, zum Beispiel in grau mit schwarzem Rahmen – <code>FALSE</code>: Element in Alarmzustand, zum Beispiel in gelb mit rotem Rahmen. • oder als Platzhalter <Umschalten-/TastenvARIABLE> Für den Farbumschlag ist dabei keine eigene Variable zugewiesen. Stattdessen wird der Farbumschlag mit der Variablen gesteuert, die in Eingabekonfiguration, Umschalten oder Tasten zugewiesen ist. Voraussetzung ist also, dass dort eine Steuervariable konfiguriert ist. Wenn sowohl in Umschalten als auch in Tasten eine Steuervariable konfiguriert ist, wird die in Tasten verwendet. Sie können den Platzhalter entfernen, indem Sie <Umschalten-/TastenvARIABLE> selektieren und <Entf> wählen. Danach erscheint  im Wertefeld.
Klick auf  im Wertefeld von Farbumschlag	IndraLogic fügt den Platzhalter <Umschalten-/TastenvARIABLE> ein.

Normalzustand	Farbvariablen des Elements während eines ungestörten Betriebs
• Rahmenfarbe	IEC-Variable des Typs <code>DWORD</code> für die Rahmenfarbe, zum Beispiel <code>prgA.dwBorderColor</code>
• Füllfarbe	IEC-Variable des Typs <code>DWORD</code> für die Füllfarbe, zum Beispiel <code>prgA.dwFillColor</code>
Alarmzustand	Farbvariablen des Elements während eines Alarms
• Rahmenfarbe	IEC-Variable des Typs <code>DWORD</code> für die Rahmenfarbe, zum Beispiel <code>prgA.dwBorderColorAlarm</code>
• Füllfarbe	IEC-Variable des Typs <code>DWORD</code> für die Rahmenfarbe, zum Beispiel <code>prgA.dwFillColorAlarm</code>

Aussehensvariablen Anzuwenden bei einer **dynamischen** Definition des Aussehens der Kontur und der Füllung des Elements. Statische Definitionen werden in **Aussehen** festgelegt.

Eigenschaft	Beschreibung Wert
Linienstärke	Eingabe einer Integervariablen, die die Linienstärke des Elements in Pixel definiert. Dies überschreibt den fixen Wert, der in Aussehen ► Linienstärke festgelegt wurde.
Füllart	Eingabe einer <code>DWORD</code> -Variablen, die die Füllart des Elements definiert. Die Farbe, die bei Farbvariablen angegeben ist, kann angezeigt oder ignoriert werden: • Variablenwert = 0: gefüllt • Variablenwert > 0 : unsichtbar, das heißt keine Füllung ist sichtbar Dies überschreibt den fixen Wert, der in Aussehen ► Füllart festgelegt wurde.
Linienart	Eingabe einer <code>DWORD</code> -Variablen, die die Kontur der Außenlinie festlegt. <i>Folgende Werte entsprechen folgenden Linienarten:</i> • 0: Durchgezogen • 1: Strich • 2: Punkt • 3: Strich Punkt • 4: Strich Punkt Punkt • 8: Unsichtbar. Das heißt, das Element wird ohne Außenlinie dargestellt.

**Elementeigenschaft
'Zustandsvariablen'**

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert <code>TRUE</code> , dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert <code>TRUE</code> , dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Elementeigenschaft 'Eingabekonfiguration' Hier kann definiert werden, welche Folgeaktionen ausgeführt werden, wenn der Benutzer im Online-Betrieb eine Eingabe im Element vornimmt.

Eingabeereignisse, die ein Element haben kann, sind:

- **OnDialogClosed**

- OnMouseClicked
- OnMouseDown
- OnMouseEnter
- OnMouseLeave
- OnMouseMove
- OnMouseUp

Mit Klick auf **Konfigurieren....** wird der Dialog [Kap. "Eingabekonfiguration" auf Seite 849](#) geöffnet, der ermöglicht, dass Folgeaktionen zugewiesen werden. Man klickt in das Feld neben der zu konfigurierenden Eingabe. Für jede selektierte Folgeaktion wird ein Eintrag in der Eigenschaftentabelle erzeugt. Jede Eingabe kann eine beliebige zugeordnete Anzahl an Folgeaktionen haben.

OnDialogClosed	Dieses Ereignis tritt ein, wenn innerhalb der Visualisierung eines der für Benutzereingaben vorher geöffneten Dialogfenster geschlossen wird.
OnMouseClicked	Dieses Mausereignis tritt ein, wenn der Cursor auf ein Element zeigt und ein vollständiger Mausklick (Drücken und Freigeben der Maustaste) ausgeführt wird.
OnMouseDown	Dieses Mausereignis tritt ein, wenn die Maustaste gedrückt wird, während der Cursor auf ein Element zeigt. (Drücken der Maustaste)
OnMouseEnter	Dieses Mausereignis tritt ein, wenn der Cursor auf das Element gezogen wird.
OnMouseLeave	Dieses Mausereignis tritt ein, wenn der Cursor das Element verlässt.
OnMouseMove	Dieses Mausereignis tritt ein, wenn der Cursor innerhalb des Elements bewegt wird.
OnMouseUp	Dieses Mausereignis tritt ein, wenn die Maustaste losgelassen wird. Das vorher ausgeführte Drücken der Maustaste geschieht dabei allerdings außerhalb des Elements.

Eingabekonfiguration: Tasten Mit **Tasten** kann konfiguriert werden, dass beim Eintreten des Ereignisses "Tasten" der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier kann eine boolesche Variable zugewiesen werden, deren Wert TRUE wird, wenn die Maustaste gedrückt wird, während der Cursor auf das Element zeigt. Der Wert wird wieder FALSE, wenn die Maustaste wieder losgelassen wird.
FALSE tasten	Wenn diese Option aktiviert ist, wird das oben beschriebene Verhalten für die eingetragene Variable umgekehrt. Das heißt, wenn die Maustaste gedrückt wird, wird der Variablenwert auf FALSE gesetzt, wenn die Taste wieder losgelassen wird, wird der Variablenwert auf TRUE gesetzt.
Beim Betreten tasten, falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert: Wird bei gedrückter Maustaste der Elementbereich verlassen, wird die Variable FALSE und wird wieder TRUE, wenn in den Elementbereich zurückgekehrt wird. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" (also an das Element gebunden) ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Umschalten Mit **Umschalten** kann konfiguriert werden, dass bei diesem Ereignis der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier ist eine boolesche Variable zuzuweisen. Die Eingabehilfe kann über Schaltfläche  geöffnet werden. Der Wert der Variablen wird bei jedem Mausklick auf das Element zwischen TRUE und FALSE umgeschaltet. Wird der Cursor bei gedrückter Maustaste aus dem Element herausbewegt, dann erfolgt kein Toggeln. Damit kann die begonnene Umschalteneingabe abgebrochen werden.
Umschalten beim Loslassen falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert. Wird bei gedrückter Maustaste der Elementbereich verlassen und dann die Maustaste losgelassen, wird die Variable umgeschaltet. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Hotkey

Mit **Hotkey** kann eine Taste, auch mit Zusatzaste, definiert und mit einer bestimmten Nachfolgeaktion (MouseDown, MouseUp) verknüpft werden, die bei einem Ereignis 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 unter Registerkarte 'Tastaturkonfiguration' der Visualisierung verwaltet ([Kap. 20.3.4 "Registerkarte 'Tastaturkonfiguration'" auf Seite 839](#)). Änderungen werden immer zwischen diesem und dem Elementeigenschaften-Editor synchronisiert.

Eigenschaft	Beschreibung Wert
Taste	Zuweisen einer Taste. Eine Auswahlliste stellt alle aktuell unterstützten Tasten bereit wie zum Beispiel <M>.
Eingabe(n)	Definieren des Ereignisses, das ausgeführt werden soll, wenn die Taste beziehungsweise Taste und Modifizierer verwendet werden. <i>Mögliche Werte, die in einer Auswahlliste bereitstehen:</i> - Keine Aktion - MouseDown-Aktion: OnMouseDown-Aktion, wenn die Taste gedrückt wird. - MouseUp-Aktion: OnMouseUp-Aktion, wenn die Taste freigelassen wird. - MouseDown/MouseUp-Aktion: OnMouseUp- und OnMouseDown-Aktion, wenn die Taste gedrückt oder losgelassen wird.
Umschalten	Wenn diese Option aktiviert wird, muss die Taste mit <Umschalt> kombiniert werden
Steuerung	Wenn diese Option aktiviert wird, muss die Taste mit <Strg> kombiniert werden
Alt	Wenn diese Option aktiviert wird, muss die Taste mit < Alt> kombiniert werden

Bild und dessen Dialog

Element 'Bild'

Mit diesem Element kann in der Visualisierung ein Bild eingefügt werden. Ein Bild wird dem Bildelement zugewiesen, indem **Statische ID** spezifiziert wird und, um eindeutigen Zugriff zu erhalten, um den Namen der Bildersammlung erweitert wird. Man kann auch dynamisch konfigurieren, indem man eine Projektvariable, deren Wert die aktuelle Bild-ID ist, für dynamisches Umschalten der Bilder zuweist. Verwenden Sie dazu **Bitmap ID**.



Verzeichnisse, die Bilder für die Verwendung in Visualisierungen bereitstellen, sind spezifiziert in **Extras ▶ Optionen ▶ IndraLogic 2G Kategorie Visualisierung**.

Ein **Hintergrundbild** für die gesamte Visualisierung kann mit dem Befehl [Kap. 20.2.11 "Befehl 'Hintergrund'" auf Seite 821](#) definiert werden.



Abb. 20-40: Bild "Zahnkranz"

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseeditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.

- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Element-Typ	Bild

Statische ID	Bezeichner der Bilddatei für eine 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). Zum Beispiel: imagepool2.button_image. Mit Klick auf  öffnet der Dialog Kap. "Dialog 'Eingabehilfe'" auf Seite 915 und listet alle verfügbaren Bildersammlungen mit ihren Bildern auf. Siehe Kap. 9.9 "Bildersammlungen verwenden" auf Seite 126 für weitere Informationen über das Verwalten von Bilddateien in Bildersammlungen.
Rahmen zeichnen	Wenn diese Option aktiviert ist, wird die Bilddatei mit einem Rahmen dargestellt.
Abschneiden	Wenn diese Option zusammen mit Skalierungsart : <code>Unskaliert</code> aktiviert ist, wird im Falle eines Frame-Elements nur der Teil der Visualisierung, der in den Frame passt, dargestellt.
Transparent	Ist die Option aktiviert, wird die Farbe, die in Eigenschaft Transparenzfarbe angegeben ist, transparent dargestellt.
Transparenzfarbe	Die Schaltfläche  öffnet den Farbauswahldialog zur Auswahl einer Farbe, die transparent dargestellt werden soll, wenn Option Transparent aktiviert ist.

Skalierungsart	<i>Zu definieren ist hier, wie die Bilddatei auf Änderungen der Elementrahmengröße reagieren soll.</i> • Isotropisch: Das Bild behält seine Proportionen; d.h. das Verhältnis Höhe-Breite bleibt erhalten, auch wenn Höhe und Breite des Elementrahmens separat verändert werden. Beachten Sie folgendes, wenn Sie die Ausrichtung der Elemente zueinander auch innerhalb eines skalierten Frame-Elements beibehalten möchten. Beispielsweise haben Sie eine Lampe zentriert oberhalb eines Schalters angeordnet, und sie soll in dieser horizontal zentrierten Position bleiben, auch wenn sich die Größe des Frames verändert. Um unerwünschte horizontale oder vertikale Verschiebungen zu vermeiden, definieren Sie explizit die Horizontale bzw. Vertikale Ausrichtung mit Zentriert (siehe unten). • Anisotropisch: Das Bild passt sich der Größe des Elementrahmens an; d.h. Höhe und Breite können unabhängig voneinander verändert werden. • Unskaliert: Das Bild behält die Originalgröße, auch wenn sich die Größe des Elementrahmens ändert. Zu berücksichtigen ist auch, ob auch Option Abschneiden aktiviert ist. Bei dieser Einstellung wird bei jeder Neuzuweisung einer ImageID die Elementgröße auf die Bildgröße automatisch angepasst.
Horizontale Ausrichtung	<i>Nur verfügbar, wenn die Option Experte des Eigenschafteneditors aktiviert ist und wenn die Skalierungsart des Bildes "Isotropisch" ist. Dient dazu, die horizontale Ausrichtung zu einem anderen Element (wie in der Basis-Visualisierung definiert) explizit aufrechtzuerhalten, wenn die Visualisierung innerhalb eines skalierten Frames verwendet wird. Siehe oben: Einstellung Isotropisch</i> • Links • Zentriert • Rechts
Vertikale Ausrichtung	<i>Nur verfügbar, wenn die Option Experte des Eigenschafteneditors aktiviert ist und wenn die Skalierungsart des Bildes Isotropisch ist. Dient dazu, die vertikale Ausrichtung zu einem anderen Element (wie in der Basis-Visualisierung definiert) explizit aufrechtzuerhalten, wenn die Visualisierung innerhalb eines skalierten Frames verwendet wird. Siehe oben: Einstellung Isotropisch.</i> • Oben • Zentriert • Unten

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Zentrum Mit Bearbeiten des Wertes wird gleichzeitig das zugehörige Element  im Editor verschoben.

Eigenschaft	Beschreibung Wert
X	X-Position des Rotationspunkts des Elements
Y	Y-Position des Rotationspunkts des Elements

Farben Eine Farbe wird mittels Hexzahl definiert, die sich aus Rot/Grün/Blau (RGB) Anteilen zusammensetzt. Für jede dieser drei Farben sind 256 (0-255) Werte verfügbar. Hier sind die Einstellungen **statisch**. Die Farbe kann aus einer Auswahlliste oder dem Farbauswahldialog, der über Button  öffnet, gewählt werden. Beispiel:  0; 128; 64

Eigenschaft	Beschreibung Wert
Farbe	Hier ist eine Farbe, mit der das Element im Normalzustand gefüllt wird, auszuwählen. Dies ist der Fall, wenn die Variable in Farbvariablen ► Farbumschlag FALSE ist. Mit Editieren des Werts ändert das Element gleichzeitig seine Farbe.
Alarmfarbe	Auszuwählen ist eine Farbe, mit der das Element gefüllt wird, wenn das Element in Alarmzustand ist. Dies ist der Fall, wenn die Variable in Farbvariablen ► Farbumschlag TRUE ist.

Aussehen

Eigenschaft	Beschreibung Wert
Linienstärke	Stärke der Rahmenlinie in Pixel. 0 kodiert das Gleiche wie 1 und setzt die Linienstärke auf 1 Pixel. Soll kein Rahmen verwendet werden, muss Linienart auf unsichtbar setzen werden.
Linienart	<i>Auszuwählen ist eine der Linienarten angewendet bei der Umrisslinie:</i> • Durchgezogen • Striche • Punkte • Strich Punkt • Strich Punkt Punkt • Unsichtbar: Dann ist die Umrisslinie unsichtbar.

Elementeigenschaft 'Texte' Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Text	Geben Sie einen Text ein. Er wird verwendet um das Element zu beschriften. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Textvariable zugewiesen.
Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Tooltip-Variable zugewiesen.

Texteigenschaften Diese Eigenschaften werden für eine statische Definition der Schriftart verwendet.

Eigenschaft	Beschreibung Wert
Horizontale Ausrichtung	<i>Definieren der horizontalen Ausrichtung des Textes durch Auswahl aus:</i> • Links • Zentriert • Rechts
Vertikale Ausrichtung	<i>Definieren der vertikalen Ausrichtung des Textes durch Auswahl aus:</i> • Oben • Zentriert • Unten
Schriftart	<i>Auswahl aus vordefinierten Schriftarten:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet sich der Dialog für benutzerdefinierte Eigenschaften der Schriftart.
Farbe Schriftart	Auszuwählen ist eine Schriftartfarbe für das Element. Entweder aus der Auswahlliste oder durch Setzen im sich öffnenden Dialog, wenn  geklickt wird.

Elementeigenschaft 'Bild ID Variable'

Bild ID	Hier ist eine Projektvariable, die die Bild-ID enthält, zuzuweisen. Siehe Kap. 9.9 "Bildersammlungen verwenden" auf Seite 126 für weitere Informationen über das Verwalten von Bilddateien in Bildersammlungen.

Elementeigenschaft 'Absolute Bewegung'

Das Element kann bewegt werden, indem die X- und Y-Position (Pixel) der linken oberen Ecke des Elements durch eine Integer-Variable verändert wird.

Bewegung • X • Y	• X: Hier ist eine Integer-Variable einzugeben, deren Wert die aktuelle X-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in X-Richtung zu verschieben (ein positiver Wert verschiebt das Element von links nach rechts). • Y: Eingabe einer Integer-Variablen, die die aktuelle Y-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in Y-Richtung zu verschieben (ein positiver Wert verschiebt das Element von oben nach unten).
Rotation	Eingabe einer Integer-Variablen, deren Wert den Winkel (Winkelgrade) definiert, um den das Element um den Rotationspunkt Zentrum  rotiert. positiver Wert = mathematisch negativ = im Uhrzeigersinn Zu beachten ist, dass das Element, im Gegensatz zum Verhalten bei Innere Rotation, selber nicht rotiert.

Skalierung	Eingabe einer Integer-Variablen, deren Wert den aktuellen Skalenwert (Prozent) definiert. 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  .
Innere Rotation	Die hier eingetragene Integer-Variable definiert den Winkel (Winkelgrade), um den das Element um seinen Rotationspunkt gedreht wird: Zentrum  . positiver Wert = mathematisch negativ = im Uhrzeigersinn. Im Gegensatz zu Rotation rotiert das Element tatsächlich.

Elementeigenschaft 'Relative Bewegung'

Das Element kann relative zu seiner fixen Position definiert in **Position** verschoben werden. Die obere linke und untere rechte Ecke des Elements werden dabei um die in den Integer-Variablen angegebenen Werte (Pixel) in X- bzw. Y-Richtung verschoben. Im Gegensatz zur absoluten Bewegung wird eine relative Position definiert, also der Abstand zur ursprünglichen Position. Das Element kann dadurch in der Form verändert werden.

Siehe **Absolute Bewegung** für das Verschieben des Elements als Ganzes.

Bewegung links-oben	
X	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die linke obere Ecke in X-Richtung verschoben wird.
Y	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die linke obere Ecke in Y-Richtung verschoben wird.
Bewegung rechts-unten	
X	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die rechte untere Ecke in X-Richtung verschoben wird.
Y	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die rechte untere Ecke in Y-Richtung verschoben wird.

Elementeigenschaft 'Textvariablen'

Sie können Text, der in einer Variablen gespeichert ist, anzeigen. Fügen Sie dazu zuerst eine Formatierungssequenz in dem Text hinzu, der unter der Eigenschaft **Texte** definiert ist. Weisen Sie dann hier eine Variable hinzu. Im Online-Betrieb wird die Formatierungssequenz durch den Inhalt der Variablen ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Beispiel:

Beispiel %f

Geben Sie **Ergebnis: %2.5f** in der Eigenschaft **Text** ein. Geben Sie **rValue** in der Eigenschaft **Textvariable** ein. Es muss eine definierte IEC-Variable sein. Dann wird im Online-Betrieb das Element mit **Ergebnis: 12.12345** beschriftet, wenn **rValue** = 12.12345 ist.

Textvariable	Geben Sie eine Variable (Typ ist Standardtyp) ein, der die Information enthält, die sie angezeigt haben wollen. Der Typ muss konform mit der Formatierungssequenz in Texte ► Text sein.
Tooltipvariable	Geben Sie eine STRING-Variable ein, die den Tooltiptext enthält, der angezeigt werden soll. Die Eingabe in Texte ► Tooltip muss ein %s enthalten.

Dynamische Texte

Diese Parameter dienen der Definition von dynamischen Texten, die aus Textlisten stammen (dies erlaubt Sprachumschaltung) ([Kap. 18.2.15 "Menü Textliste" auf Seite 727](#)).

Eigenschaft	Beschreibung Wert
Textliste	Name der Textliste (String) wie im POU- oder Geräte-Fenster verwendet; z.B. <code>errorlist</code>
Textindex	Index (ID) des Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. 'M1' oder String-Variable, z.B. <code>error_22</code> .
Tooltipindex	Index (ID) des Tooltip-Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. 'T1' oder String-Variable, z.B. <code>error_Tooltip_22</code> .

Elementeigenschaft 'Schriftartvariablen'

Schriftartvariablen werden bei dynamischen Schriftdefinitionen des Elementtextes über Projektvariablen verwendet. Statischen Definitionen werden unter **Texteigenschaften ► Schriftart** konfiguriert.

Schriftname	Hier ist eine Variable Typ <code>STRING</code> anzugeben, die den Schriftnamen enthält, der für die Beschriftung des Elements verwendet werden soll. (Namensangabe wie im Standard-Schrifttypen-Dialog). Beispiel: <code>PlcProg.stFontVar := 'Arial';</code>
Größe	Eingabe einer Variablen Typ <code>INT</code> , die die Größe des Elementtextes in Pixel enthält, so wie im Standarddialog Schriftart. Beispiel: <code>PlcProg.iFontHeight := 16;</code>
Flags	Eingabe einer Variable Typ <code>DWORD</code> , die mit Hilfe der unten genannten Flag-Werte die Schriftart-Darstellung festlegt. Eine kombinierte Definition kann erreicht werden, indem die jeweiligen Flag-Werte addiert werden und die Summe angegeben wird. • 1: kursiv • 2: fett • 4: unterstrichen • 8: gelöscht Beispiel für einen Text, der fett und unterstrichen dargestellt wird: <code>PlcProg.dwFontType := 6;</code>
Zeichensatz	Der Zeichensatz, der für die Schriftart verwendet werden soll, kann über die Standard-Zeichensatznummer definiert werden. Mit einer <code>DWORD</code> -Variablen kann diese Nummer angegeben werden (siehe auch die Skript -Definition im Standard-Schriftartdialog)
Farbe	Eingabe einer <code>DWORD</code> -Variable, in der die Farbe des Elementtextes festgelegt wird.
Flags für Textausrichtung	Variable mit ganzzahligem Datentyp, zum Beispiel <code>PlcProg.dwTextAlignment</code> . Die Visualisierung wertet zur Laufzeit die Variable aus und richtet dementsprechend den Text aus. <i>Textausrichtung</i> • 0: oben links • 1: horizontal zentriert • 2: rechts • 4: vertikal zentriert • 8: unten

Elementeigenschaft 'Farbvariablen'

Die Elementeigenschaft ermöglicht die dynamische Umschaltung der Elementfarbe mit Hilfe von IEC-Variablen für Steuerung und Farben.

Die IEC-Variable für Farbe enthält diese als hexadezimaler Farbdefinition, die die Rot-/Grün-/Blauanteile der Farbe im RGB-Farbraum codiert: `16#00RRGGBB`. Dabei ist für jeden Farbanteil ein Byte für 256 Farbabstufungen

gen (von 0-255) reserviert. 0 bedeutet dabei, dass kein Farbanteil enthalten ist. 255 bedeutet 100% Farbanteil. Das führende Byte ist immer 16#00.

Beispiel:

Beispiel Farbvariable

`dwFillColor := 16#008FE03F;`

Codierung: Rotanteil: 8F, Grünanteil: E0, Blauanteil: 3F

- `grau : dwGrey := 16#00888888`
- `rot: dwRed := 16#00FF0000`
- `schwarz: dwBlack := 16#00000000`

Farbumschlag	<i>Steuervariable</i> - als IEC-Variable Typ <code>BOOL</code>, zum Beispiel <code>prgA.bToggleColor</code> Zur Laufzeit steuert die Visualisierung dynamisch die Farbdarstellung des Elements <code>TRUE</code>: Element wird dargestellt in Farbe, zum Beispiel in grau <code>FALSE</code>: Element wird dargestellt in Alarmfarbe, zum Beispiel in rot - oder als Platzhalter <Umschalten-/Tastenvariable> Für den Farbumschlag ist dabei keine eigene Variable zugewiesen. Stattdessen wird der Farbumschlag mit der Variablen gesteuert, die in Eingabekonfiguration, Umschalten oder Tasten zugewiesen ist. Voraussetzung ist also, dass dort eine Steuervariable konfiguriert ist. Wenn sowohl in Umschalten als auch in Tasten eine Steuervariable konfiguriert ist, wird die in Tasten verwendet. Sie können den Platzhalter entfernen, indem Sie <Umschalten-/Tastenvariable> selektieren und <Entf> wählen. Danach erscheint  im Wertefeld.
Klick auf  im Wertefeld von Farbumschlag	IndraLogic fügt den Platzhalter <Umschalten-/Tastenvariable> ein.
Farbe	IEC-Variable Typ <code>DWORD</code> für die Farbe, zum Beispiel <code>prgA.dwColor</code>
Alarmfarbe	IEC-Variable Typ <code>DWORD</code> für die Alarmfarbe, zum Beispiel <code>prgA.dwAlarmColor</code>

Aussehensvariable

Anzuwenden bei einer dynamischen Definition des Aussehens der Kontur des Elements. Statische Definitionen werden in **Aussehen** festgelegt.

Eigenschaft	Beschreibung Wert
Linienstärke	Eingabe einer Integervariablen, die die Linienstärke des Elements in Pixel definiert. Dies überschreibt den fixen Wert, der in Aussehen ► Linienstärke festgelegt wurde.
Linienart	Eingabe einer DWORD-Variablen, die die Kontur der Außenlinie festlegt. <i>Folgende Werte entsprechen folgenden Linienarten:</i> • 0: Durchgezogen • 1: Strich • 2: Punkt • 3: Strich Punkt • 4: Strich Punkt Punkt • 8: Unsichtbar. Das heißt, das Element wird ohne Außenlinie dargestellt.

**Elementeigenschaft
'Zustandsvariablen'**

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Elementeigenschaft 'Eingabekonfiguration'

Hier kann definiert werden, welche Folgeaktionen ausgeführt werden, wenn der Benutzer im Online-Betrieb eine Eingabe im Element vornimmt.

Eingabeereignisse, die ein Element haben kann, sind:

- **OnDialogClosed**
- **OnMouseClicked**
- **OnMouseDown**
- **OnMouseEnter**
- **OnMouseLeave**
- **OnMouseMove**
- **OnMouseUp**

Mit Klick auf **Konfigurieren....** wird der Dialog [Kap. "Eingabekonfiguration" auf Seite 849](#) geöffnet, der ermöglicht, dass Folgeaktionen zugewiesen werden. Man klickt in das Feld neben der zu konfigurierenden Eingabe. Für jede selektierte Folgeaktion wird ein Eintrag in der Eigenschaftentabelle erzeugt. Jede Eingabe kann eine beliebige zugeordnete Anzahl an Folgeaktionen haben.

OnDialogClosed	Dieses Ereignis tritt ein, wenn innerhalb der Visualisierung eines der für Benutzereingaben vorher geöffneten Dialogfenster geschlossen wird.
OnMouseClicked	Dieses Mausereignis tritt ein, wenn der Cursor auf ein Element zeigt und ein vollständiger Mausklick (Drücken und Freigeben der Maustaste) ausgeführt wird.
OnMouseDown	Dieses Mausereignis tritt ein, wenn die Maustaste gedrückt wird, während der Cursor auf ein Element zeigt. (Drücken der Maustaste)
OnMouseEnter	Dieses Mausereignis tritt ein, wenn der Cursor auf das Element gezogen wird.

OnMouseLeave	Dieses Mausereignis tritt ein, wenn der Cursor das Element verlässt.
OnMouseMove	Dieses Mausereignis tritt ein, wenn der Cursor innerhalb des Elements bewegt wird.
OnMouseUp	Dieses Mausereignis tritt ein, wenn die Maustaste losgelassen wird. Das vorher ausgeführte Drücken der Maustaste geschieht dabei allerdings außerhalb des Elements.

Eingabekonfiguration: Hotkey

Mit **Hotkey** kann eine Taste, auch mit Zusatz Taste, definiert und mit einer bestimmten Nachfolgeaktion (MouseDown, MouseUp) verknüpft werden, die bei einem Ereignis 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 unter Registerkarte 'Tastaturkonfiguration' der Visualisierung verwaltet ([Kap. 20.3.4 "Registerkarte 'Tastaturkonfiguration'" auf Seite 839](#)). Änderungen werden immer zwischen diesem und dem Elementeigenschaften-Editor synchronisiert.

Eigenschaft	Beschreibung Wert
Taste	Zuweisen einer Taste. Eine Auswahlliste stellt alle aktuell unterstützten Tasten bereit wie zum Beispiel <M>.
Eingabe(n)	Definieren des Ereignisses, das ausgeführt werden soll, wenn die Taste beziehungsweise Taste und Modifizierer verwendet werden. <i>Mögliche Werte, die in einer Auswahlliste bereitstehen:</i> - Keine Aktion - MouseDown-Aktion: OnMouseDown-Aktion, wenn die Taste gedrückt wird. - MouseUp-Aktion: OnMouseUp-Aktion, wenn die Taste freigelassen wird. - MouseDown/MouseUp-Aktion: OnMouseUp- und OnMouseDown-Aktion, wenn die Taste gedrückt oder losgelassen wird.
Umschalten	Wenn diese Option aktiviert wird, muss die Taste mit <Umschalt> kombiniert verwendet werden
Steuerung	Wenn diese Option aktiviert wird, muss die Taste mit <Strg> kombiniert verwendet werden
Alt	Wenn diese Option aktiviert wird, muss die Taste mit < Alt> kombiniert verwendet werden

Dialog 'Eingabehilfe'

Die Eingabehilfe für Bilder aus Bildersammlungen öffnet automatisch, wenn ein Visualisierungselement **Bild** erstmals in einer Visualisierung eingefügt wird. Sie erreichen diesen Dialog auch, wenn Sie in der Eigenschaftenansicht eines Bildelements in das Wertefeld **Statische ID** klicken und dann  ausführen.

'Textsuche'

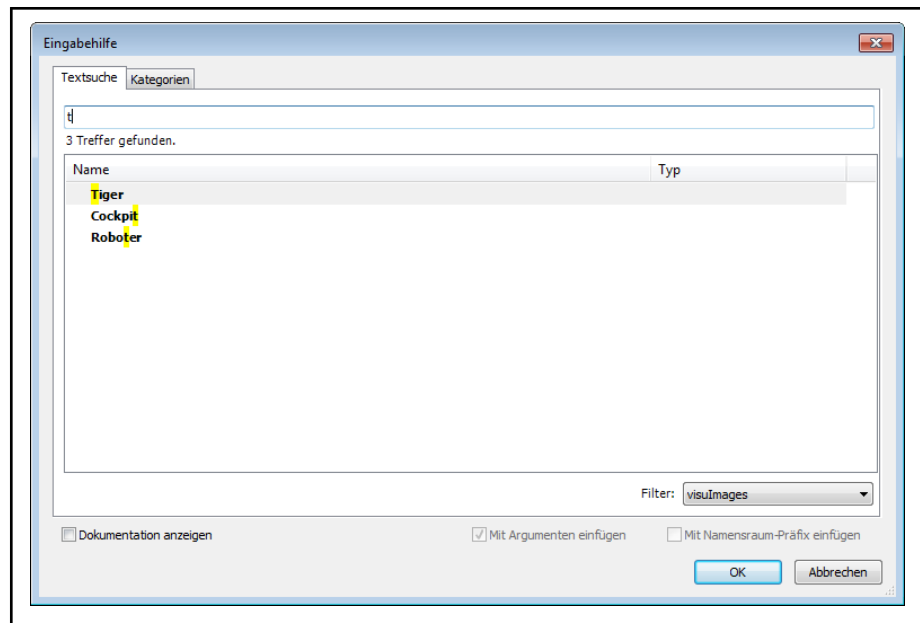


Abb. 20-41: Eingabehilfe bei Zuweisung eines Bildes

Setzen Sie zuerst den **Filter**, indem Sie in der Auswahlliste die gewünschte Bildersammlung selektieren. Geben Sie dann den Namen oder einen Teil des Namens der gesuchten Bilddatei in den oberen Zeileneditor ein. Sofort werden alle Bilddateinamen, die eine Übereinstimmung mit der Eingabe haben, angezeigt. Wählen Sie daraus ein Bild aus, selektieren es und beenden den Dialog mit <OK>. Oder Sie führen einen Doppelklick auf dem gewünschten Bild aus.

Nach Schließen des Dialogs ist in der Eigenschaft **Statische ID** das Bild mit <Name Bildersammlung>.<Name Bild> vollständig referenziert angegeben.

'Kategorien'

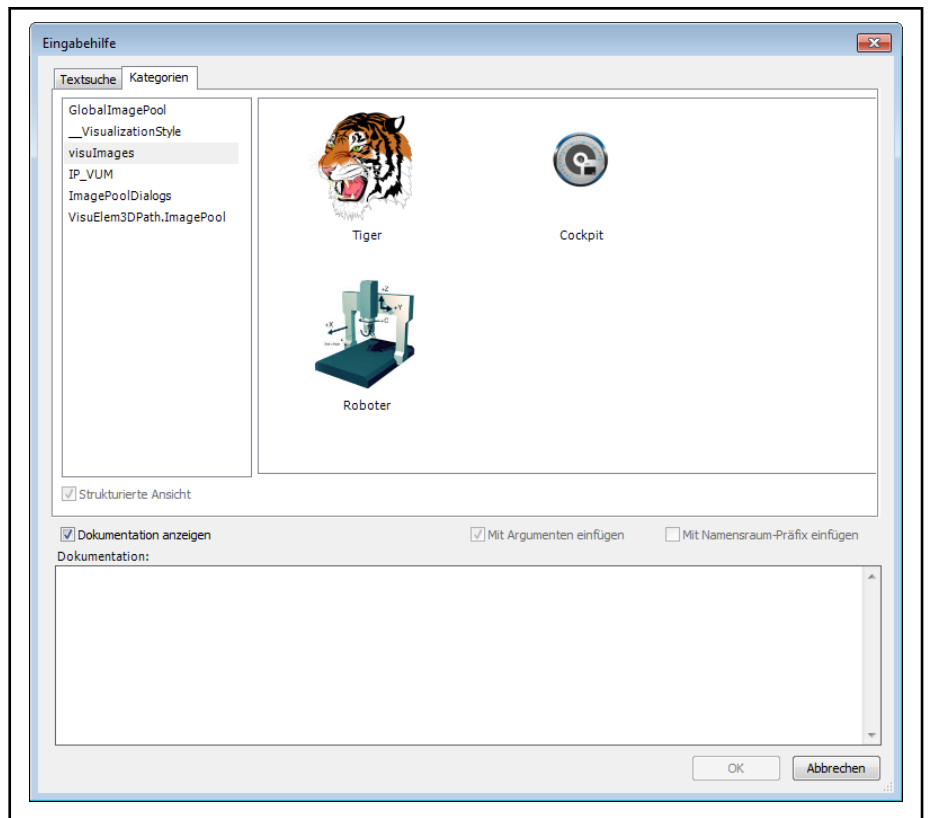


Abb. 20-42: Eingabehilfe bei Zuweisung eines Bildes

Kategorien: In der linken Spalte sind alle Bildersammlungen des Projekts aufgelistet. Selektieren Sie eine, dann werden rechts die Bilder der Sammlung in Vorschau dargestellt. Wählen Sie daraus ein Bild aus, selektieren es und beenden den Dialog mit <OK>. Oder Sie führen einen Doppelklick auf dem gewünschten Bild aus. Nach Schließen des Dialogs ist in der Eigenschaft **Statische ID** das Bild mit <Name Bildersammlung>.<Name Bild> vollständig referenziert angegeben.

Element 'Frame'

Ein Frameelement wird verwendet

- als "Referenz zu einer anderen Visualisierung" auf Seite 918
- zum "Anbieten eine Schnittstelle für Platzhalter" auf Seite 918
- zum "Umschalten zwischen verschiedenen Visualisierungen" auf Seite 918

Inhalt dieser Seite ist weiterhin:

"Elementeigenschaften" auf Seite 919

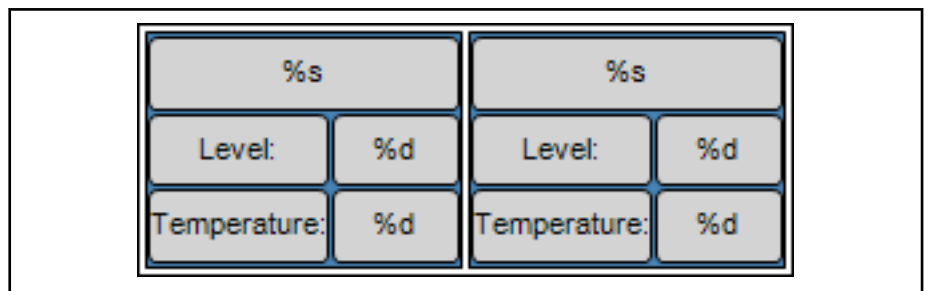


Abb. 20-43: Frame

Referenz zu einer anderen Visualisierung	Grundsätzlich kann eine Visualisierung in eine andere eingefügt und damit referenziert werden. Für diesen Zweck ist ein Frame zu verwenden, der eine oder mehrere Referenzen zu Visualisierungen einschließen kann. Welche Visualisierung aktuell zum Einfügen zur Verfügung steht, hängt von deren Position im POU- und Gerätebaum relativ zur gerade geöffneten Visualisierung ab. Siehe dazu Kap. 20.2.10 "Befehl 'Frame-Auswahl'" auf Seite 819.
Anbieten eine Schnittstelle für Platzhalter	Eine Visualisierung kann eine Schnittstelle, zu öffnen über den Befehl 'Schnittstelleneditor', anbieten, in der man die Eingabevariablen genauso wie bei einem Funktionsbaustein definieren kann (Kap. "Grundlegendes zum Schnittstelleneditor" auf Seite 831). Diese Eingabeparameter funktionieren als Platzhalter. In einer Instanz (Referenz) der Visualisierung muss ein Platzhalter dann durch einen Wert oder Ausdruck für die spezielle Verwendung im lokalen Objekt ersetzt werden. Das Ersetzen muss in Eigenschaften des Frames, der die Visualisierungsinstanzen einbindet, vorgenommen werden: Referenzierte Visualisierungen ► Referenzierte Visualisierungen. Beachten Sie dabei, dass den Eingabevariablen einer Visualisierungsinstanz, die einer bestimmten Applikation zugeordnet sind, gültige Variablen dieser Applikation zugewiesen werden müssen.
Umschalten zwischen verschiedenen Visualisierungen	Ein Frame (so wie man Frames zum Beispiel von Internetseiten her kennt) definiert einen Teilbereich der aktuellen Visualisierung, der weitere Visualisierungen enthalten kann, zwischen denen man im Online-Betrieb umschaltet. Bei diesen Visualisierungen werden deren Eingaben so konfigurieren, dass sie einen Wechsel der Anzeige dieser Referenzen im Frame bewirken. Der Dialog Kap. "Eingabekonfiguration" auf Seite 849 bietet dafür die Option Visualisierungswechsel an. Dabei ist zu berücksichtigen, dass Mauseaktionen mit Hilfe der Eigenschaft Eingabekonfiguration definiert werden. Um die Auswahl der Visualisierungen in einem Frame zu definieren ist der Dialog Frame-Konfiguration durch Selektieren des Frames und ausführen des Befehls Kap. 20.2.10 "Befehl 'Frame-Auswahl'" auf Seite 819 zu öffnen. Hier sind alle verfügbaren Visualisierungen bereitgestellt und man kann die gewünschte auswählen und damit in den Frame einfügen. Empfehlenswert ist es, nur Visualisierungen aus dem globalen POU-Pool zu verwenden, um zu vermeiden, dass später Probleme entstehen weil vielleicht ein Gerät umbenannt wird. Ein "gefüllter" Frame zeigt dann auch im Offline-Betrieb die enthaltenen Visualisierungen an. <i>Beispiel:</i> Umschalten zwischen Visualisierungen 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 IndraLogic 1x - innerhalb eines Visualisierungselements verschiedene andere im Wechsel angezeigt werden, also eine Schalttafel visualisiert werden. Dieses Konzept ersetzt das Visualisierungs- und Platzhalterkonzept in IndraLogic 1x.
Benutzereingabe beim Editieren eines Elements	<i>Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:</i> • ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: Kap. 20.2 "Befehle" auf Seite 812 • verschiedene Mausoperationen, am Element anwendbar: Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843

**Benutzereingabe beim Editieren
der Elementeigenschaften***Element spezifisches Kommando:*

- [Kap. 20.2.10 "Befehl 'Frame-Auswahl'" auf Seite 819](#)

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Element-Typ	Frame

Abschneiden	Wird diese Option zusammen mit " Skalierungsart : Unskaliert" aktiviert, dann wird nur der Teil der Visualisierung, der in den Rahmen passt, dargestellt.
Rahmen zeichnen	Falls aktiviert, wird die referenzierte Visualisierung mit einem Rahmen angezeigt.

Skalierungsart	<i>Hier ist zu spezifizieren, wie der Frame auf Größenänderungen der Visualisierung reagieren soll:</i> • Isotropisch: Der Frame behält seine Proportionen. Damit kann das Verhältnis von Höhe und Breite beibehalten werden, sogar wenn Höhe und Breite der Visualisierung unabhängig modifiziert wurden. • Anisotropisch: Der Frame richtet sich nach der Größe der Visualisierung, so dass Höhe und Breite der referenzierten Visualisierung unabhängig modifiziert werden können. • Unskaliert: Die original Größe des Frames wird beibehalten ungeachtet der Visualisierungsgröße. Falls die Option Abschneiden ebenfalls aktiviert ist, wird nur der passende Teil angezeigt. • Unskaliert und scrollbar: verwenden Sie diese Option, dann wird die referenzierte Visualisierung ohne Skalierung angezeigt. Ist sie größer als der Fensterbereich des Frames, dann wird der Frame mit Scrollbalken versehen, um den angezeigten Bereich der Visualisierung verschieben zu können. Wenn Sie die Position des Scrollbalkens mit einer Variablen setzen wollen, verwenden Sie die Eigenschaften Variable Scrollposition horizontal oder Variable Scrollposition vertikal, siehe "Elementeigenschaft 'Scrollbalken-Einstellungen'" auf Seite 925.
Deaktivieren des Hintergrundzeichnens	Um die Performance der Visualisierung zu optimieren, werden die nicht-animierten Elemente des Frame-Elements als Hintergrund Bitmap gezeichnet. Diese könnte dazu führen, dass die Elemente nicht in der erwarteten Reihenfolge dargestellt werden. Um dieses Verhalten zu vermeiden, kann die Funktion deaktiviert werden.

Referenzierte Visualisierungen

Referenzierte Visualisierungen	Der Befehl Konfiguriere öffnet den Dialog Kap. 20.2.10 "Befehl 'Frame-Auswahl'" auf Seite 819. Hier werden die referenzierten Visualisierungen des Elements aufgelistet.

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Zentrum Mit Bearbeiten des Wertes wird gleichzeitig das zugehörige Element  im Editor verschoben.

Eigenschaft	Beschreibung Wert
X	X-Position des Rotationspunkts des Elements
Y	Y-Position des Rotationspunkts des Elements

Farben Eine Farbe wird mittels Hexzahl definiert, die sich aus Rot/Grün/Blau (RGB) Anteilen zusammensetzt. Für jede dieser drei Farben sind 256 (0-255) Werte verfügbar. Hier sind die Einstellungen **statisch**. Die Farbe kann aus einer

Auswahlliste oder dem Farbauswahldialog, der über Button  öffnet, gewählt werden. Beispiel:  0; 128; 64

Eigenschaft	Beschreibung Wert
Farbe	Hier ist eine Farbe, mit der das Element im Normalzustand gefüllt wird, auszuwählen. Dies ist der Fall, wenn die Variable in Farbvariablen ► Farbumschlag FALSE ist. Mit Editieren des Werts ändert das Element gleichzeitig seine Farbe.
Alarmfarbe	Auszuwählen ist eine Farbe, mit der das Element gefüllt wird, wenn das Element in Alarmzustand ist. Dies ist der Fall, wenn die Variable in Farbvariablen ► Farbumschlag TRUE ist.

Aussehen

Eigenschaft	Beschreibung Wert
Linienstärke	Stärke der Rahmenlinie in Pixel. 0 kodiert das Gleiche wie 1 und setzt die Linienstärke auf 1 Pixel. Soll kein Rahmen verwendet werden, muss Linienart auf unsichtbar setzen werden.
Linienart	<i>Auszuwählen ist eine der Linienarten angewendet bei der Umrisslinie:</i> • Durchgezogen • Striche • Punkte • Strich Punkt • Strich Punkt Punkt • Unsichtbar: Dann ist die Umrisslinie unsichtbar.

Elementeigenschaft 'Texte'

Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Text	Geben Sie einen Text ein. Er wird verwendet um das Element zu beschriften. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Textvariable zugewiesen.
Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Tooltip-Variable zugewiesen.

Texteigenschaften

Diese Eigenschaften werden für eine statische Definition der Schriftart verwendet.

Eigenschaft	Beschreibung Wert
Horizontale Ausrichtung	<i>Definieren der horizontalen Ausrichtung des Textes durch Auswahl aus:</i> • Links • Zentriert • Rechts
Vertikale Ausrichtung	<i>Definieren der vertikalen Ausrichtung des Textes durch Auswahl aus:</i> • Oben • Zentriert • Unten
Schriftart	<i>Auswahl aus vordefinierten Schriftarten:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet sich der Dialog für benutzerdefinierte Eigenschaften der Schriftart.
Farbe Schriftart	Auszuwählen ist eine Schriftartfarbe für das Element. Entweder aus der Auswahlliste oder durch Setzen im sich öffnenden Dialog, wenn  geklickt wird.

Elementeigenschaft 'Absolute Bewegung'

Das Element kann bewegt werden, indem die X- und Y-Position (Pixel) der linken oberen Ecke des Elements durch eine Integer-Variable verändert werden. Hier werden absolute Koordinatenwerte verwendet.

Bewegung • X • Y	• X: Hier ist eine Integer-Variable einzugeben, deren Wert die aktuelle X-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in X-Richtung zu verschieben (ein positiver Wert verschiebt das Element von links nach rechts). • Y: Eingabe einer Integer-Variablen, die die aktuelle Y-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in Y-Richtung zu verschieben (ein positiver Wert verschiebt das Element von oben nach unten).
Rotation	Eingabe einer Integer-Variablen, deren Wert den Winkel (Winkelgrade) definiert, um den das Element um den Rotationspunkt Zentrum  rotiert. positiver Wert = mathematisch negativ = im Uhrzeigersinn Zu beachten ist, dass das Element, im Gegensatz zum Verhalten bei Innere Rotation, selber nicht rotiert.
Skalierung	Eingabe einer Integer-Variablen, deren Wert den aktuellen Skalenwert (Prozent) definiert. 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 .

Elementeigenschaft 'Relative Bewegung'

Das Element kann relative zu seiner fixen Position definiert in **Position** verschoben werden. Die obere linke und untere rechte Ecke des Elements werden dabei um die in den Integer-Variablen angegebenen Werte (Pixel) in X- bzw. Y-Richtung verschoben. Im Gegensatz zur absoluten Bewegung wird

eine relative Position definiert, also der Abstand zur ursprünglichen Position. Das Element kann dadurch in der Form verändert werden.

Siehe **Absolute Bewegung** für das Verschieben des Elements als Ganzes.

Bewegung links-oben	
X	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die linke obere Ecke in X-Richtung verschoben wird.
Y	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die linke obere Ecke in Y-Richtung verschoben wird.
Bewegung rechts-unten	
X	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die rechte untere Ecke in X-Richtung verschoben wird.
Y	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die rechte untere Ecke in Y-Richtung verschoben wird.

Elementeigenschaft 'Textvariablen'

Sie können Text, der in einer Variablen gespeichert ist, anzeigen. Fügen Sie dazu zuerst eine Formatierungssequenz in dem Text hinzu, der unter der Eigenschaft **Texte** definiert ist. Weisen Sie dann hier eine Variable hinzu. Im Online-Betrieb wird die Formatierungssequenz durch den Inhalt der Variablen ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Beispiel:

Beispiel %f

Geben Sie **Ergebnis: %2.5f** in der Eigenschaft **Text** ein. Geben Sie **rValue** in der Eigenschaft **Textvariable** ein. Es muss eine definierte IEC-Variable sein. Dann wird im Online-Betrieb das Element mit **Ergebnis: 12.12345** beschriftet, wenn **rValue = 12.12345** ist.

Textvariable	Geben Sie eine Variable (Typ ist Standarddatentyp) ein, der die Information enthält, die sie angezeigt haben wollen. Der Typ muss konform mit der Formatierungssequenz in Texte ▶ Text sein.
Tooltipvariable	Geben Sie eine STRING-Variable ein, die den ToOLTIPtext enthält, der angezeigt werden soll. Die Eingabe in Texte ▶ Tooltip muss ein %s enthalten.

Dynamische Texte

Diese Parameter dienen der Definition von dynamischen Texten, die aus Textlisten stammen (dies erlaubt Sprachumschaltung) ([Kap. 18.2.15 "Menü 'Textliste'" auf Seite 727](#)).

Eigenschaft	Beschreibung Wert
Textliste	Name der Textliste (String) wie im POU- oder Geräte-Fenster verwendet; z.B. <code>errorlist</code>
Textindex	Index (ID) des Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. <code>'M1'</code> oder String-Variable, z.B. <code>error_22</code> .
Tooltipindex	Index (ID) des Tooltip-Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. <code>'T1'</code> oder String-Variable, z.B. <code>error_Tooltip_22</code> .

Elementeigenschaft 'Schriftartvariablen'

Schriftartvariablen werden bei dynamischen Schriftdefinitionen des Elementtextes über Projektvariablen verwendet. Statischen Definitionen werden unter **Texteigenschaften ▶ Schriftart** konfiguriert.

Schriftname	Hier ist eine Variable Typ <code>STRING</code> anzugeben, die den Schriftnamen enthält, der für die Beschriftung des Elements verwendet werden soll. (Namensangabe wie im Standard-Schrifttypen-Dialog). Beispiel: <code>PlcProg.stFontVar := 'Arial';</code>
Größe	Eingabe einer Variablen Typ <code>INT</code> , die die Größe des Elementtexts in Pixel enthält, so wie im Standarddialog Schriftart. Beispiel: <code>PlcProg.iFontHeight := 16;</code>
Flags	Eingabe einer Variable Typ <code>DWORD</code> , die mit Hilfe der unten genannten Flag-Werte die Schriftart-Darstellung festlegt. Eine kombinierte Definition kann erreicht werden, indem die jeweiligen Flag-Werte addiert werden und die Summe angegeben wird. • 1: kursiv • 2: fett • 4: unterstrichen • 8: gelöscht Beispiel für einen Text, der fett und unterstrichen dargestellt wird: <code>PlcProg.dwFont- Type := 6;</code>
Zeichensatz	Der Zeichensatz, der für die Schriftart verwendet werden soll, kann über die Standard-Zeichensatznummer definiert werden. Mit einer <code>DWORD</code> -Variablen kann diese Nummer angegeben werden (siehe auch die Skript -Definition im Standard-Schriftartdialog)
Farbe	Eingabe einer <code>DWORD</code> -Variable, in der die Farbe des Elementtextes festgelegt wird.
Flags für Textausrichtung	Variable mit ganzzahligem Datentyp, zum Beispiel <code>PlcProg.dwTextAlignment</code> . Die Visualisierung wertet zur Laufzeit die Variable aus und richtet dementsprechend den Text aus. <i>Textausrichtung</i> • 0: oben links • 1: horizontal zentriert • 2: rechts • 4: vertikal zentriert • 8: unten

Elementeigenschaft 'Farbvariablen'

Die Elementeigenschaft ermöglicht die dynamische Umschaltung der Elementfarbe mit Hilfe von IEC-Variablen für Steuerung und Farben.

Die IEC-Variable für Farbe enthält diese als hexadezimaler Farbdefinition, die die Rot-/Grün-/Blauanteile der Farbe im RGB-Farbraum codiert: `16#00RRGGBB`. Dabei ist für jeden Farbanteil ein Byte für 256 Farbabstufungen (von 0-255) reserviert. 0 bedeutet dabei, dass kein Farbanteil enthalten ist. 255 bedeutet 100% Farbanteil. Das führende Byte ist immer `16#00`.

Beispiel:

Beispiel Farbvariable

```
dwFillColor := 16#008FE03F;
```

Codierung: Rotanteil: 8F, Grünanteil: E0, Blauanteil: 3F

- grau: `dwGrey := 16#00888888`
- rot: `dwRed := 16#00FF0000`
- schwarz: `dwBlack := 16#00000000`

Farbumschlag	<i>Steuervariable</i> - als IEC-Variable Typ <code>BOOL</code>, zum Beispiel <code>prgA.bToggleColor</code> Zur Laufzeit steuert die Visualisierung dynamisch die Farbdarstellung des Elements <code>TRUE</code>: Element wird dargestellt in Farbe, zum Beispiel in grau <code>FALSE</code>: Element wird dargestellt in Alarmfarbe, zum Beispiel in rot - oder als Platzhalter <Umschalten-/TastenvARIABLE> Für den Farbumschlag ist dabei keine eigene Variable zugewiesen. Stattdessen wird der Farbumschlag mit der Variablen gesteuert, die in Eingabekonfiguration, Umschalten oder Tasten zugewiesen ist. Voraussetzung ist also, dass dort eine Steuervariable konfiguriert ist. Wenn sowohl in Umschalten als auch in Tasten eine Steuervariable konfiguriert ist, wird die in Tasten verwendet. Sie können den Platzhalter entfernen, indem Sie <Umschalten-/TastenvARIABLE> selektieren und <Entf> wählen. Danach erscheint  im Wertefeld.
Klick auf  im Wertefeld von Farbumschlag	IndraLogic fügt den Platzhalter <Umschalten-/TastenvARIABLE> ein.
Farbe	IEC-Variable Typ <code>DWORD</code> für die Farbe, zum Beispiel <code>prgA.dwColor</code>
Alarmfarbe	IEC-Variable Typ <code>DWORD</code> für die Alarmfarbe, zum Beispiel <code>prgA.dwAlarmColor</code>

Aussehensvariable

Anzuwenden bei einer dynamischen Definition des Aussehens der Kontur des Elements. Statische Definitionen werden in **Aussehen** festgelegt.

Eigenschaft	Beschreibung Wert
Linienstärke	Eingabe einer Integervariablen, die die Linienstärke des Elements in Pixel definiert. Dies überschreibt den fixen Wert, der in Aussehen ► Linienstärke festgelegt wurde.
Linienart	Eingabe einer <code>DWORD</code>-Variablen, die die Kontur der Außenlinie festlegt. <i>Folgende Werte entsprechen folgenden Linienarten:</i> 0: Durchgezogen 1: Strich 2: Punkt 3: Strich Punkt 4: Strich Punkt Punkt 8: Unsichtbar. Das heißt, das Element wird ohne Außenlinie dargestellt.

Elementeigenschaft 'Scrollbalken-Einstellungen'

Es ist sehr empfehlenswert die Variablen Client spezifisch zu verwenden. Ändern sich die Variablen oder wird ein Scrollbalken mit Hilfe der Maus verschoben, dann wirkt sich die Änderung in diesem Fall nur auf den Frame des betroffenen Clients aus. Andernfalls werden alle Clients aktualisiert.

Variable Scrollposition horizontal	Geben Sie eine Variable ein, die die Position des horizontalen Scrollbalkens enthält.
Variable Scrollposition vertikal	Geben Sie eine Variable ein, die die Position des vertikalen Scrollbalkens enthält.

Client spezifische Variable für die Scrollposition

IEC-Code

```

PROGRAM PlcProg
VAR
    iScrollHor : ARRAY[0..20] OF INT;
    iScrollVer : ARRAY[0..20] OF INT;
END_VAR

```

Variable Scrollposition horizontal	PlcProg.iScrollHor[CURRENTCLIENTID]
Variable Scrollposition vertikal	PlcProg.iScrollVer[CURRENTCLIENTID]

Tab. 20-6: Festgelegte Eigenschaften für das Frame-Element

Umschaltvariable Diese Eigenschaft ermöglicht es, innerhalb eines Rahmens Visualisierungen umzuschalten.

Eigenschaft	Beschreibung Wert
Variable	Hier ist eine Integervariable anzugeben, deren Wert die ID der anzuzeigenden Visualisierung enthält. Die ID einer Visualisierung wird durch deren Reihenfolge in der Liste der zugewiesenen Visualisierungen im Dialog Konfiguration der Framevisualisierungen bestimmt. Zum Beispiel ergibt der erste Eintrag in dieser Liste die ID 0, der zweite Eintrag die ID 1, usw.

**Elementeigenschaft
'Zustandsvariablen'**

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Elementeigenschaft 'Eingabekonfiguration'

Hier kann definiert werden, welche Folgeaktionen ausgeführt werden, wenn der Benutzer im Online-Betrieb eine Eingabe im Element vornimmt.

Eingabeereignisse, die ein Element haben kann, sind:

- OnDialogClosed
- OnMouseClicked
- OnMouseDown
- OnMouseEnter
- OnMouseLeave
- OnMouseMove
- OnMouseUp

Mit Klick auf **Konfigurieren....** wird der Dialog [Kap. "Eingabekonfiguration" auf Seite 849](#) geöffnet, der ermöglicht, dass Folgeaktionen zugewiesen werden. Man klickt in das Feld neben der zu konfigurierenden Eingabe. Für jede selektierte Folgeaktion wird ein Eintrag in der Eigenschaftentabelle erzeugt. Jede Eingabe kann eine beliebige zugeordnete Anzahl an Folgeaktionen haben.

OnDialogClosed	Dieses Ereignis tritt ein, wenn innerhalb der Visualisierung eines der für Benutzereingaben vorher geöffneten Dialogfenster geschlossen wird.
OnMouseClicked	Dieses Mausereignis tritt ein, wenn der Cursor auf ein Element zeigt und ein vollständiger Mausklick (Drücken und Freigeben der Maustaste) ausgeführt wird.
OnMouseDown	Dieses Mausereignis tritt ein, wenn die Maustaste gedrückt wird, während der Cursor auf ein Element zeigt. (Drücken der Maustaste)
OnMouseEnter	Dieses Mausereignis tritt ein, wenn der Cursor auf das Element gezogen wird.
OnMouseLeave	Dieses Mausereignis tritt ein, wenn der Cursor das Element verlässt.
OnMouseMove	Dieses Mausereignis tritt ein, wenn der Cursor innerhalb des Elements bewegt wird.
OnMouseUp	Dieses Mausereignis tritt ein, wenn die Maustaste losgelassen wird. Das vorher ausgeführte Drücken der Maustaste geschieht dabei allerdings außerhalb des Elements.

Eingabekonfiguration: Tasten

Mit **Tasten** kann konfiguriert werden, dass beim Eintreten des Ereignisses "Tasten" der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier kann eine boolesche Variable zugewiesen werden, deren Wert TRUE wird, wenn die Maustaste gedrückt wird, während der Cursor auf das Element zeigt. Der Wert wird wieder FALSE, wenn die Maustaste wieder losgelassen wird.
FALSE tasten	Wenn diese Option aktiviert ist, wird das oben beschriebene Verhalten für die eingetragene Variable umgekehrt. Das heißt, wenn die Maustaste gedrückt wird, wird der Variablenwert auf FALSE gesetzt, wenn die Taste wieder losgelassen wird, wird der Variablenwert auf TRUE gesetzt.
Beim Betreten tasten, falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert: Wird bei gedrückter Maustaste der Elementbereich verlassen, wird die Variable FALSE und wird wieder TRUE, wenn in den Elementbereich zurückgekehrt wird. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" (also an das Element gebunden) ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Umschalten

Mit **Umschalten** kann konfiguriert werden, dass bei diesem Ereignis der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier ist eine boolesche Variable zuzuweisen. Die Eingabehilfe kann über Schaltfläche  geöffnet werden. Der Wert der Variablen wird bei jedem Mausklick auf das Element zwischen TRUE und FALSE umgeschaltet. Wird der Cursor bei gedrückter Maustaste aus dem Element herausbewegt, dann erfolgt kein Toggeln. Damit kann die begonnene Umschaltteingabe abgebrochen werden.
Umschalten beim Loslassen falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert. Wird bei gedrückter Maustaste der Elementbereich verlassen und dann die Maustaste losgelassen, wird die Variable umgeschaltet. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Hotkey

Mit **Hotkey** kann eine Taste, auch mit Zusatzstufe, definiert und mit einer bestimmten Nachfolgeaktion (MouseDown, MouseUp) verknüpft werden, die bei einem Ereignis 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 unter Registerkarte 'Tastaturkonfiguration' der Visualisierung verwaltet ([Kap. 20.3.4 "Registerkarte 'Tastaturkonfiguration'" auf Seite 839](#)). Änderungen werden immer zwischen diesem und dem Elementeigenschaften-Editor synchronisiert.

Eigenschaft	Beschreibung Wert
Taste	Zuweisen einer Taste. Eine Auswahlliste stellt alle aktuell unterstützten Tasten bereit wie zum Beispiel <M>.
Eingabe(n)	Definieren des Ereignisses, das ausgeführt werden soll, wenn die Taste beziehungsweise Taste und Modifizierer verwendet werden. <i>Mögliche Werte, die in einer Auswahlliste bereitstehen:</i> - Keine Aktion - MouseDown-Aktion: OnMouseDown-Aktion, wenn die Taste gedrückt wird. - MouseUp-Aktion: OnMouseUp-Aktion, wenn die Taste freigelassen wird. - MouseDown/MouseUp-Aktion: OnMouseUp- und OnMouseDown-Aktion, wenn die Taste gedrückt oder losgelassen wird.
Umschalten	Wenn diese Option aktiviert wird, muss die Taste mit <Umschalt> kombiniert verwendet werden
Steuerung	Wenn diese Option aktiviert wird, muss die Taste mit <Strg> kombiniert verwendet werden
Alt	Wenn diese Option aktiviert wird, muss die Taste mit < Alt> kombiniert verwendet werden

20.4.3 'Allgemeine Steuerelemente'

Element 'Text'

Das Element **Text** dient dazu Beschriftungen, Überschriften und jede andere Art von Text einer Visualisierung hinzuzufügen.



Abb. 20-44: Text

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Text

Elementeigenschaft 'Texte' Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Im Online-Modus wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt.

[Kap. "Text und Sprache verwenden" auf Seite 845](#)

Text	Geben Sie einen Text ein. Er wird verwendet um das Element zu beschriften. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Textvariable zugewiesen.

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Texteigenschaften Diese Eigenschaften werden für eine statische Definition der Schriftart verwendet.

Eigenschaft	Beschreibung Wert
Horizontale Ausrichtung	<i>Definieren der horizontalen Ausrichtung des Textes durch Auswahl aus:</i> • Links • Zentriert • Rechts
Vertikale Ausrichtung	<i>Definieren der vertikalen Ausrichtung des Textes durch Auswahl aus:</i> • Oben • Zentriert • Unten
Schriftart	<i>Auswahl aus vordefinierten Schriftarten:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet sich der Dialog für benutzerdefinierte Eigenschaften der Schriftart.
Farbe Schriftart	Auszuwählen ist eine Schriftartfarbe für das Element. Entweder aus der Auswahlliste oder durch Setzen im sich öffnenden Dialog, wenn  geklickt wird.

Zustandsvariablen

Property	Beschreibung Wert
Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Modus unsichtbar.

Element 'Combobox Integer'

Das Element erlaubt dem Visualisierungsbenutzer, einen Wert aus einer Auswahlliste auszuwählen. Der Index des selektierten Wertes wird in eine Variable geschrieben. Die Einträge können aus einer Textliste oder als Bilderliste aus der Bildersammlung stammen.

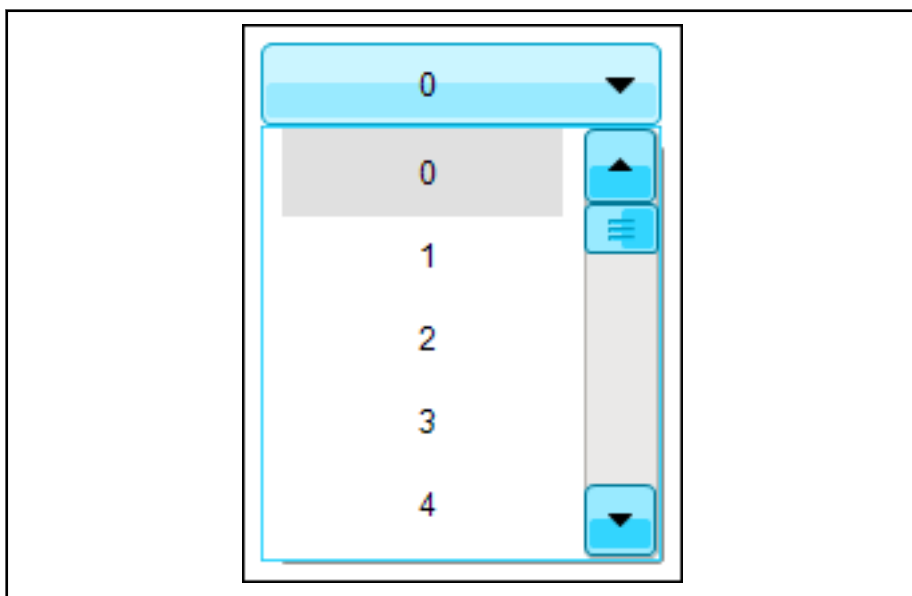


Abb. 20-45: Combobox Integer, Style 8

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseeditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,

- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Combobox Integer

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Variable	Variable des Typs Enumeration, deren Wert der Index des ausgewählten Listenelements ist.
Textliste	zugewiesene Textliste, die die Auswahlliste der Combobox bestimmt, zum Beispiel 'TextList_A'. Hinweis: Wenn keine Textliste angegeben ist, müssen Sie unter der Eigenschaft Wertebereich einen Minimalwert und einen Maximalwert angeben haben.  : Auswahlliste mit Textlisten des Projekts erscheint
Bildersammlung	Bildersammlung, die die Bilder für die Auswahlliste des Elements enthält, zum Beispiel 'ImagePool_A'.  : Auswahlliste mit Bildersammlungen des Projekts erscheint

Elementeigenschaft 'Einstellungen der Liste'

Einstellungen der Liste	Darstellung der Liste, die aufklappt, wenn ein Visualisierungsbenutzer in das Element klickt.
Zeilenhöhe	Voreinstellung: Von Stil
Anzahl sichtbarer Zeilen	Beschränkt die Anzahl der Zeilen der Auswahlliste. Voreinstellung: Von Stil
Bildhöhe	Voreinstellung: Von Stil Voraussetzung: In Bildersammlung ist eine angegeben.
Bildbreite	Voreinstellung: Von Stil Voraussetzung: In Bildersammlung ist eine angegeben.
Bildoffset	Voreinstellung: Von Stil Voraussetzung: In Bildersammlung ist eine angegeben.
Größe des Scrollbars	Breite des Scrollbalkens, der erscheint, wenn mehr Einträge in der Auswahlliste vorhanden sind, als in Anzahl sichtbarer Zeilen angegeben ist. Voreinstellung: 20

Elementeigenschaft 'Texte'

Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt.

[Kap. "Text und Sprache verwenden" auf Seite 845](#)

Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ▶ Tooltip-Variable zugewiesen.

Elementeigenschaft 'Wertebereich'

Wertebereich	Wertebereich der Auswahlliste
Grenze des Wertebereichs	 : Die Visualisierung erweitert oder beschränkt den Wertebereich der Auswahlliste im Vergleich zur Textliste, je nachdem, was Sie in Minimaler Wert und Maximaler Wert festgelegt haben. Wenn Sie dort nichts festgelegt haben, entspricht die Auswahlliste der Textliste. Voraussetzung: Nur sichtbar, wenn in Eigenschaft Textliste eine ausgewählt ist.
Minimaler Wert	Index des ersten Eintrags der Auswahlliste, zum Beispiel 0. Wenn Sie keine Textliste ausgewählt haben, müssen Sie hier eine Anzahl festlegen.
Maximaler Wert	Index des letzten Eintrags der Auswahlliste, zum Beispiel 20. Wenn Sie keine Textliste ausgewählt haben, müssen Sie hier eine Anzahl festlegen.
Fehlende Texteinträge ausfiltern	 : Die Visualisierung zeigt in der Auswahlliste der Combobox die Einträge so an, wie sie von der Textliste festgelegt sind und entfernt Einträge, wenn der Wertebereich einschränkend wirkt.  : Die Visualisierung zeigt die Auswahlliste der Combobox so an, wie sie in Wertebereich festgelegt ist und ergänzt Einträge, wenn sie in der Textliste fehlen. Voraussetzung: Nur sichtbar, wenn in Eigenschaft Textliste eine ausgewählt ist.

Elementeigenschaft 'Texteigenschaften'

Texteigenschaft	Darstellung des Texts in der Auswahlliste
Verwendung von	• Standardstilwerten • Benutzerdefinierte Einstellung: Zusätzliche Eigenschaften sind sichtbar, die eine benutzerdefinierte Darstellung der Auswahlliste ermöglicht.

Benutzerdefinierte Texteigenschaften	Voraussetzung: In Verwendung von haben Sie Benutzerdefinierte Einstellung ausgewählt.
Horizontale Ausrichtung	Horizontale Textausrichtung
Vertikale Ausrichtung	Vertikale Textausrichtung
Schriftart	Formatierung des Textes
Farbe Schriftart	Farbe des Textes
Transparent	Transparenzstufe des Textes, 255 ist deckend.

Elementeigenschaft 'Zustandsvariablen'

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Element 'Combobox Array'

Combobox Array erlaubt dem Benutzer, eine Zeile mit Werten aus einer Drop-down-Liste auszuwählen. Der Index der selektierten Zeile wird in eine Variable geschrieben.

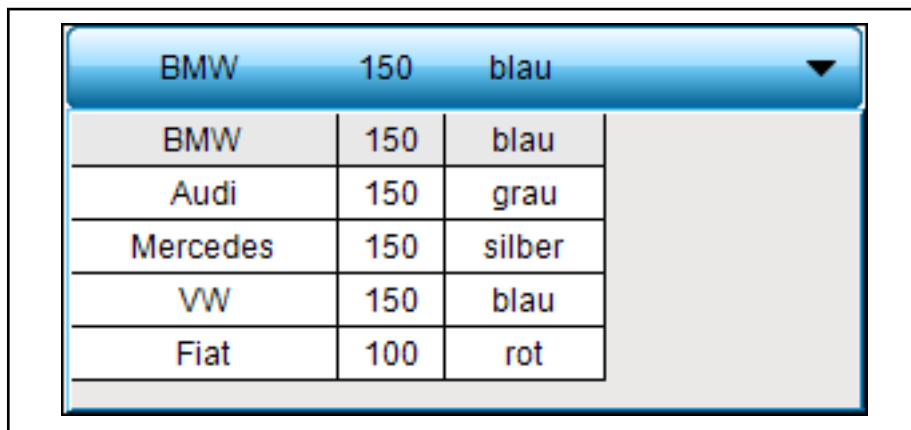


Abb. 20-46: Combobox Array, Style 8

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)

Benutzereingabe beim Editieren der Elementeigenschaften

- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabebehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Combobox ARRAY

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Variable	Variable, welche den Index der ausgewählten Zeile enthält
Daten Array	Variable vom Typ Array mit vollständigem Pfad, welche dargestellt werden soll. Die Struktur der Variablen bestimmt die Anzahl der Spalten und Zeilen der Tabelle.

Elementeigenschaft 'Spalten'

Das Element Combobox Array visualisiert eine Array- oder Strukturvariable in Tabellenansicht. Der Index von Arrayelementen bzw. von Komponenten einer Struktur wird in einer Spalte bzw. einer Zeile angezeigt. Zweidimensionale Arrays oder Arrays einer Struktur werden in mehreren Spalten angezeigt. Die visualisierte Variable wird in der Eigenschaft **Daten-Array** festgelegt. Wenn dort eine Variable zugewiesen ist, kann die Darstellung der Tabellenspalten, in welchen die jeweiligen Arrayelemente dargestellt werden, festgelegt werden. Für jede Spalte, die einem Index [<n>] zugeordnet ist, ist eine individuelle Konfiguration möglich.

Spalten [<n>]	Aufgrund der Struktur der Variablen die unter Daten-Array definiert ist, wird die Anzahl der Spalten ermittelt und über den Index <n> beschrieben. <pre>StringTabelle : ARRAY [0..2, 0..4] OF STRING := ['BMW','Audi','Mercedes','VW','Fiat','150','150','150','150','100','blau','grau','silber','blau','rot'];</pre> es werden drei Spalten gebildet [0], [1] und [2].
Max. Array-Index	Optionale numerische Variable, über die ein maximaler Array-Index festgelegt werden kann.
Zeilenhöhe	Höhe der Zeilen in Pixel.
Scrollbar Größe	Breite der vertikalen Scrollbar in Pixel.

Elementeigenschaft 'Spalten: Spalte [<n>']

Breite	Breite der Spalte in Pixel
Bildspalte	Mit dieser Option können in dieser Spalte Bilder angezeigt werden. Wenn sie aktiviert ist, werden Bilder aus der globalen Bildersammlung oder von benutzerdefinierten Bildersammlungen in dieser Spalte der Tabelle angezeigt. Dabei legen die Werte der Zellen einer Tabelle die ID des Bildes in der Bildersammlung fest.
Bildkonfiguration Füllmodus Transparenz Transparenzfarbe	Füllmodus Zelle ausfüllen: Das Bild wird auf die Zellengröße angepasst ohne Berücksichtigung des Breiten-Höhen-Verhältnisses. Zentriert: Das Bild wird unter Beibehaltung der Proportionen innerhalb der Zelle zentriert. Selbst wenn die Höhe bzw. die Breite einzeln angepasst wird, bleibt das Breiten-Höhen-Verhältnis bestehen. Wenn Transparenz aktiviert ist, wird die unter Transparenzfarbe eingestellte Farbe transparent dargestellt. Transparenzfarbe: Auswahl einer Farbe aus der Combobox oder mit Hilfe des Farbauswahldialogs, welcher über die Schaltfläche  geöffnet werden kann. Die Farbe wird transparent dargestellt wenn die entsprechende Option Transparenz aktiviert ist.
Textausrichtung in Spalte	<i>Ausrichtung der Spalten:</i> - Links - Zentriert - Rechts

Elementeigenschaft 'Texte'

Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Bei-

spiel %s, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt.

[Kap. "Text und Sprache verwenden" auf Seite 845](#)

Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Tooltip-Variable zugewiesen.

Elementeigenschaft 'Zustandsvariablen'

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Element 'Registersteuerelement'

Ein Registersteuerelement ermöglicht mehrere Formulare innerhalb eines Fensters. Registerkarte werden als Navigationswidget zum Umschalten zwischen den Formularen verwendet.

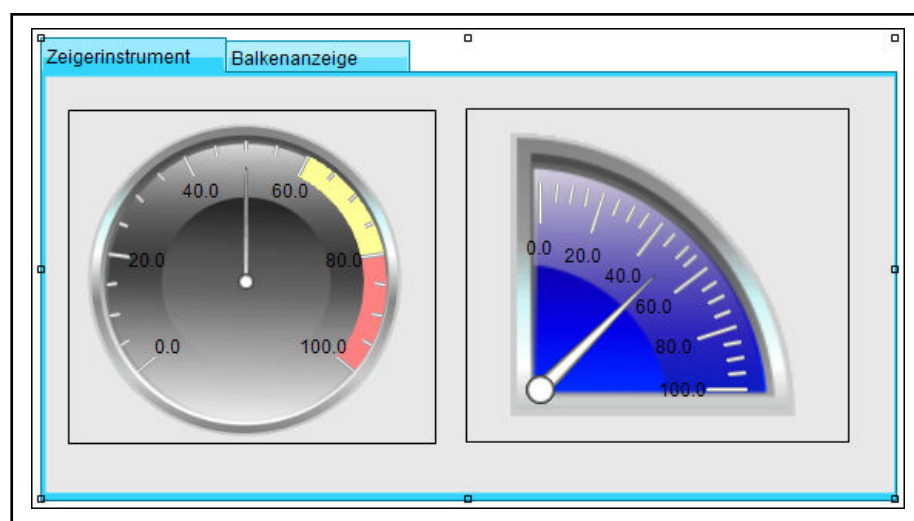


Abb. 20-47: Registersteuerelement, Stil 8

Wenn aufgrund der zu geringen Breite des Elements nicht alle Registerkarte angezeigt werden können, werden an dem Element oben rechts zwei Navigationspfeile eingeblendet.

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

**Benutzereingabe beim Editieren
der Elementeigenschaften**

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Registersteuerelement

Registerbreite	Definiert die Breite des Registerkarte in Pixel.
Skalierungsart	Hier ist zu spezifizieren, wie das Fenster auf Änderungen der Größe zu reagieren hat: Isotropisch: das referenzierte Element behält seine Proportionen; somit bleibt, auch wenn Höhe und Breite unabhängig geändert wurden, das Verhältnis von Höhe und Breite der Visualisierung unverändert. Anisotropisch: das Registersteuerelement folgt der Größe, so dass Höhe und Breite der referenzierten Visualisierung unabhängig geändert werden können. Unskaliert: Die original Größe der Visualisierung bleibt erhalten, unabhängig von der Größe des Registersteuerelements. Unskaliert und scrollbar: verwenden Sie diese Option, dann wird die referenzierte Visualisierung ohne Skalierung angezeigt. Ist sie größer als der Fensterbereich des Frames, dann wird der Frame mit Scrollbalken versehen, um den angezeigten Bereich der Visualisierung verschieben zu können. Wenn Sie die Position des Scrollbalkens mit einer Variablen setzen wollen, verwenden Sie die Eigenschaften Variable Scrollposition horizontal oder Variable Scrollposition vertikal, siehe "Elementeigenschaft 'Einstellungen Scrollbar'" auf Seite 939.
Deaktivieren des Hintergrundzeichnens	Um die Performance der Visualisierung zu optimieren, werden die nicht-animierten Elemente des Frame-Elements als Hintergrund Bitmap gezeichnet. Diese könnte dazu führen, dass die Elemente nicht in der erwarteten Reihenfolge dargestellt werden. Um dieses Verhalten zu vermeiden, kann die Funktion deaktiviert werden.

Referenzierte Visualisierungen

Referenzierte Visualisierungen	Der Befehl Konfiguriere öffnet den Dialog Kap. 20.2.10 "Befehl 'Frame-Auswahl'" auf Seite 819. Hier werden die referenzierten Visualisierungen des Elements aufgelistet.

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Elementeigenschaft 'Einstellungen Scrollbar' Es ist sehr empfehlenswert die Variablen clientspezifisch zu verwenden. Ändern sich die Variablen oder wird ein Scrollbalken mit Hilfe der Maus verschoben, dann wirkt sich die Änderung in diesem Fall nur auf den Frame des betroffenen Clients aus. Andernfalls werden alle Clients aktualisiert.

[Kap. "Element 'Frame'"](#) auf Seite 917.

Variable Scrollposition horizontal	Geben Sie eine Variable ein, die die Position des horizontalen Scrollbalkens enthält.
Variable Scrollposition vertikal	Geben Sie eine Variable ein, die die Position des vertikalen Scrollbalkens enthält.

Umschaltvariable Diese Eigenschaft ermöglicht es, innerhalb eines Rahmens Visualisierungen umzuschalten.

Eigenschaft	Beschreibung Wert
Variable	Hier ist eine Integervariable anzugeben, deren Wert die ID der anzuzeigenden Visualisierung enthält. Die ID einer Visualisierung wird durch deren Reihenfolge in der Liste der zugewiesenen Visualisierungen im Dialog Konfiguration der Framevisualisierungen bestimmt. Zum Beispiel ergibt der erste Eintrag in dieser Liste die ID 0, der zweite Eintrag die ID 1, usw.

Elementeigenschaft
'Zustandsvariablen'

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Element 'Schaltfläche'

Ein Visualisierungsbenutzer, der auf eine **Schaltfläche** klickt oder ein anderes Ereignis ausführt, löst eine definierte Folgeaktion aus. Sie können die Schaltfläche optisch anpassen, beispielsweise durch Zuweisen eines Bildes oder auch durch Konfigurieren einer Relieffansicht.

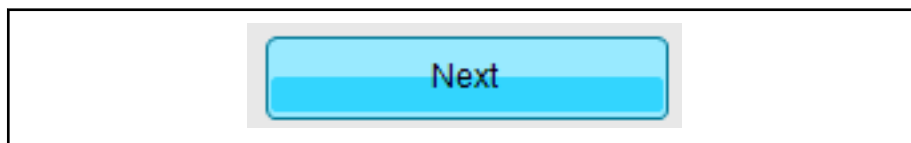


Abb. 20-48: Schaltfläche, Style 8

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,

- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Schaltfläche

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Zentrum

Mit Bearbeiten des Wertes wird gleichzeitig das zugehörige Element  im Editor verschoben.

Eigenschaft	Beschreibung Wert
X	X-Position des Rotationspunkts des Elements
Y	Y-Position des Rotationspunkts des Elements

Farben

Eine Farbe wird mittels Hexzahl definiert, die sich aus Rot/Grün/Blau (RGB) Anteilen zusammensetzt. Für jede dieser drei Farben sind 256 (0-255) Werte verfügbar. Hier sind die Einstellungen **statisch**. Die Farbe kann aus einer Auswahlliste oder dem Farbauswahldialog, der über Button  öffnet, gewählt werden. Beispiel:  0; 128; 64

Eigenschaft	Beschreibung Wert
Farbe	Hier ist eine Farbe, mit der das Element im Normalzustand gefüllt wird, auszuwählen. Dies ist der Fall, wenn die Variable in Farbvariablen ► Farbumschlag FALSE ist. Mit Editieren des Werts ändert das Element gleichzeitig seine Farbe.
Alarmfarbe	Auszuwählen ist eine Farbe, mit der das Element gefüllt wird, wenn das Element in Alarmzustand ist. Dies ist der Fall, wenn die Variable in Farbvariablen ► Farbumschlag TRUE ist.

Farbverlauf verwenden	Standardmäßig ist die Checkbox deaktiviert. Falls sie aktiviert wird, wird das zugehörige Element mit Farbverlauf gezeichnet.
Farbverlaufsauswahl	Dialog öffnet. Kap. "Farbverlaufeditor" auf Seite 862

Schaltflächenhöhe	Höhe der Reliefansicht in Pixel, zum Beispiel 

Elementeigenschaft 'Bild'

Statische ID	Referenz auf ein Bild in einer Bildersammlung im Format <Name der Bildersammlung>.<Bild-ID> zum Beispiel <code>Imagepool.GreenButton</code> Wenn das Bild aus GlobalImagePool ist, können Sie den Namen der Bildersammlung weglassen, da IndraLogic diese Sammlung immer zuerst durchsucht.  : Dialog Eingabehilfe öffnet sich und listet alle projektweit verfügbaren Bildersammlungen mit deren Bildern auf.
Skalentyp	<i>Verhalten des Bildes bei Änderung der Schaltflächengröße</i> • Isotropisch: Das Bild behält seine Proportionen. Das Verhältnis Höhe-Breite bleibt erhalten, auch wenn Sie die Höhe oder die Breite der Schaltfläche separat verändern. • Anisotropisch: Das Bild passt sich an die Größe der Schaltfläche an. • Unskaliert: Das Bild behält die Originalgröße, auch wenn Sie die Größe der Schaltfläche ändern.
Transparenz	Die Visualisierung stellt im Bild die Farbe, die in Transparenzfarbe ausgewählt ist, transparent dar.
Transparenzfarbe	Farbe, die im Bild transparent ist, zum Beispiel Weiß . Wenn der Hintergrund des Bildes, das die Statische ID referenziert, weiß ist, wird dieser Hintergrund transparent dargestellt. Mit Klick auf  öffnet sich ein Farbauswahldialog. Voraussetzung: Option Transparenz ist aktiviert.

Horizontale Ausrichtung	<i>Horizontalen Ausrichtung des Bitmaps</i> • Links • Zentriert • Rechts
Vertikale Ausrichtung	<i>Vertikalen Ausrichtung des Bitmaps</i> • Oben • Zentriert • Unten

Elementeigenschaft 'Texte' Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Text	Geben Sie einen Text ein. Er wird verwendet um das Element zu beschriften. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Textvariable zugewiesen.
Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Tooltip-Variable zugewiesen.

Texteigenschaften Diese Eigenschaften werden für eine statische Definition der Schriftart verwendet.

Eigenschaft	Beschreibung Wert
Schriftart	<i>Auswahl aus vordefinierten Schriftarten:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet sich der Dialog für benutzerdefinierte Eigenschaften der Schriftart.
Farbe Schriftart	Auszuwählen ist eine Schriftartfarbe für das Element. Entweder aus der Auswahlliste oder durch Setzen im sich öffnenden Dialog, wenn  geklickt wird.

Elementeigenschaft 'Absolute Bewegung' Das Element kann bewegt werden, indem die X- und Y-Position (Pixel) der linken oberen Ecke des Elements durch eine Integer-Variable verändert wird.

Bewegung - X - Y	X: Hier ist eine Integer-Variable einzugeben, deren Wert die aktuelle X-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in X-Richtung zu verschieben (ein positiver Wert verschiebt das Element von links nach rechts). Y: Eingabe einer Integer-Variablen, die die aktuelle Y-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in Y-Richtung zu verschieben (ein positiver Wert verschiebt das Element von oben nach unten).
Rotation	Eingabe einer Integer-Variablen, deren Wert den Winkel (Winkelgrade) definiert, um den das Element um den Rotationspunkt Zentrum  rotiert. positiver Wert = mathematisch negativ = im Uhrzeigersinn Zu beachten ist, dass das Element, im Gegensatz zum Verhalten bei Innere Rotation , selber nicht rotiert.
Skalierung	Eingabe einer Integer-Variablen, deren Wert den aktuellen Skalenwert (Prozent) definiert. 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  .
Innere Rotation	Die hier eingetragene Integer-Variable definiert den Winkel (Winkelgrade), um den das Element um seinen Rotationspunkt gedreht wird: Zentrum  . positiver Wert = mathematisch negativ = im Uhrzeigersinn. Im Gegensatz zu Rotation rotiert das Element tatsächlich.

Elementeigenschaft 'Relative Bewegung'

Das Element kann relative zu seiner fixen Position definiert in **Position** verschoben werden. Die obere-linke und untere-rechte Ecke des Elements werden dabei um die in den Integer-Variablen angegebenen Werte (Pixel) in X- bzw. Y-Richtung verschoben. Im Gegensatz zur absoluten Bewegung wird eine relative Position definiert, also der Abstand zur ursprünglichen Position. Das Element kann dadurch in der Form verändert werden.

Siehe **Absolute Bewegung** für das Verschieben des Elements als Ganzes.

Bewegung links-oben	
X	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die linke obere Ecke in X-Richtung verschoben wird.
Y	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die linke obere Ecke in Y-Richtung verschoben wird.
Bewegung rechts-unten	
X	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die rechte untere Ecke in X-Richtung verschoben wird.
Y	Integer-Variable, deren Wert die Anzahl der Pixel angibt, um die die rechte untere Ecke in Y-Richtung verschoben wird.

Elementeigenschaft 'Textvariablen'

Sie können Text, der in einer Variablen gespeichert ist, anzeigen. Fügen Sie dazu zuerst eine Formatierungssequenz in dem Text hinzu, der unter der Eigenschaft **Texte** definiert ist. Weisen Sie dann hier eine Variable hinzu. Im Online-Betrieb wird die Formatierungssequenz durch den Inhalt der Variablen ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Beispiel:

Beispiel %f

Geben Sie **Ergebnis: %2.5f** in der Eigenschaft **Text** ein. Geben Sie **rValue** in der Eigenschaft **Textvariable** ein. Es muss eine definierte IEC-Variablen sein. Dann wird im Online-Betrieb das Element mit **Ergebnis: 12.12345** beschriftet, wenn **rValue = 12.12345** ist.

Textvariable	Geben Sie eine Variable (Typ ist Standarddatentyp) ein, der die Information enthält, die sie angezeigt haben wollen. Der Typ muss konform mit der Formatierungssequenz in Texte ▶ Text sein.
Tooltipvariable	Geben Sie eine STRING-Variable ein, die den Toolliptext enthält, der angezeigt werden soll. Die Eingabe in Texte ▶ Tooltip muss ein %s enthalten.

Dynamische Texte

Diese Parameter dienen der Definition von dynamischen Texten, die aus Textlisten stammen (dies erlaubt Sprachumschaltung) ([Kap. 18.2.15 "Menü 'Textliste'" auf Seite 727](#)).

Eigenschaft	Beschreibung Wert
Textliste	Name der Textliste (String) wie im POU- oder Geräte-Fenster verwendet; z.B. <code>errorlist</code>
Textindex	Index (ID) des Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. <code>'M1'</code> oder String-Variable, z.B. <code>error_22</code> .
Tooltipindex	Index (ID) des Tooltip-Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. <code>'T1'</code> oder String-Variable, z.B. <code>error_Tooltip_22</code> .

Elementeigenschaft 'Schriftartvariablen'

Schriftartvariablen werden bei dynamischen Schriftdefinitionen des Elementtextes über Projektvariablen verwendet. Statischen Definitionen werden unter **Texteigenschaften ▶ Schriftart** konfiguriert.

Schriftname	Hier ist eine Variable Typ <code>STRING</code> anzugeben, die den Schriftnamen enthält, der für die Beschriftung des Elements verwendet werden soll. (Namensangabe wie im Standard-Schrifttypen-Dialog). Beispiel: <code>PlcProg.stFontVar := 'Arial';</code>
Größe	Eingabe einer Variablen Typ <code>INT</code> , die die Größe des Elementtextes in Pixel enthält, so wie im Standarddialog Schriftart. Beispiel: <code>PlcProg.iFontHeight := 16;</code>
Flags	Eingabe einer Variable Typ <code>DWORD</code> , die mit Hilfe der unten genannten Flag-Werte die Schriftart-Darstellung festlegt. Eine kombinierte Definition kann erreicht werden, indem die jeweiligen Flag-Werte addiert werden und die Summe angegeben wird. 1: kursiv 2: fett 4: unterstrichen 8: gelöscht Beispiel für einen Text, der fett und unterstrichen dargestellt wird: <code>PlcProg.dwFontType := 6;</code>
Zeichensatz	Der Zeichensatz, der für die Schriftart verwendet werden soll, kann über die Standard-Zeichensatznummer definiert werden. Mit einer <code>DWORD</code> -Variablen kann diese Nummer angegeben werden (siehe auch die Skript -Definition im Standard-Schriftartdialog)

Farbe	Eingabe einer <code>DWORD</code> -Variable, in der die Farbe des Elementtextes festgelegt wird.
Flags für Textausrichtung	Variable mit ganzzahligem Datentyp, zum Beispiel <code>PlcProg.dwTextAlignment</code>. Die Visualisierung wertet zur Laufzeit die Variable aus und richtet dementsprechend den Text aus. <i>Textausrichtung</i> • 0: oben links • 1: horizontal zentriert • 2: rechts • 4: vertikal zentriert • 8: unten

Elementeigenschaft 'Farbvariablen'

Die Eigenschaft ermöglicht die dynamische Umschaltung der Farbe mit Hilfe von IEC-Variablen für Steuerung und Farben.

Die IEC-Variable für Farbe enthält diese als hexadezimaler Farbdefinition, die die Rot-/Grün-/Blauanteile der Farbe im RGB-Farbraum codiert: `16#00RRGGBB`. Dabei ist für jeden Farbanteil ein Byte für 256 Farbabstufungen (von 0-255) reserviert. 0 bedeutet dabei, dass kein Farbanteil enthalten ist. 255 bedeutet 100% Farbanteil. Das führende Byte ist immer `16#00`.

Beispiel:

Beispiel Farbvariable

```
dwFillColor := 16#008FE03F;
```

Codierung: Rotanteil: `8F`, Grünanteil: `E0`, Blauanteil: `3F`

- grau: `dwGrey := 16#00888888`
- rot: `dwRed := 16#00FF0000`
- schwarz: `dwBlack := 16#00000000`

Farbumschlag	Steuervariable als IEC-Variable des Typs <code>BOOL</code>, zum Beispiel <code>prgA.bToggleColor</code>. <i>Zur Laufzeit steuert die Visualisierung dynamisch die Farbdarstellung des Elements</i> • <code>TRUE</code>: Element wird dargestellt in Farbe, zum Beispiel in grau • <code>FALSE</code>: Element wird dargestellt in Alarmfarbe, zum Beispiel in rot
Farbe	IEC-Variable des Typs <code>DWORD</code> für die Farbe, zum Beispiel <code>prgA.dwColor</code>
Alarmfarbe	IEC-Variable des Typs <code>DWORD</code> für die Alarmfarbe, zum Beispiel <code>prgA.dwAlarmColor</code>

Elementeigenschaft 'Zustandsvariablen'

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert <code>TRUE</code> , dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert <code>TRUE</code>, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Elementeigenschaft 'Schaltflächenzustandsvariable'

Boolescher Wert	<i>Steuervariable</i> - als IEC-Variable des Typs <code>BOOL</code>, zum Beispiel <code>prgA.bStateButton</code> In diesem Fall steuert die Visualisierung zur Laufzeit dynamisch den Zustand der Schaltfläche <code>TRUE</code>: Schaltfläche ist gedrückt dargestellt <code>FALSE</code>: Schaltfläche ist nicht gedrückt dargestellt - oder als der Platzhalter <Umschalten-/Tastenvariable> In diesem Fall ist keine eigene Variable für die Steuerung der Schaltfläche zugewiesen. Stattdessen wird der Zustand mit der Variablen gesteuert, die in Eingabekonfiguration, Umschalten oder Tasten zugewiesen ist. Voraussetzung ist also, dass dort eine Steuervariable konfiguriert ist. Wenn sowohl in Umschalten als auch in Tasten eine Steuervariable konfiguriert ist, wird die in Tasten verwendet. Sie können den Platzhalter entfernen, indem Sie <Umschalten-/Tastenvariable> selektieren und <Entf> wählen. Danach erscheint  im Wertefeld.
	IndraLogic fügt den Platzhalter <Umschalten-/Tastenvariable> ein.

Elementeigenschaft 'Bild-ID-Variable'

Bild-ID	IEC-Variable des Typs <code>STRING</code> für die Bild-ID
Elementeigenschaft 'Eingabekonfiguration'	<i>Siehe auch</i> Kap. 9.9 "Bildersammlungen verwenden" auf Seite 126 Hier kann definiert werden, welche Folgeaktionen ausgeführt werden, wenn der Benutzer im Online-Betrieb eine Eingabe im Element vornimmt. <i>Eingabeereignisse, die ein Element haben kann, sind:</i> OnDialogClosed OnMouseClicked OnMouseDown OnMouseEnter OnMouseLeave OnMouseMove OnMouseUp Mit Klick auf Konfigurieren.... wird der Dialog Kap. "Eingabekonfiguration" auf Seite 849 geöffnet, der ermöglicht, dass Folgeaktionen zugewiesen werden. Man klickt in das Feld neben der zu konfigurierenden Eingabe. Für jede selektierte Folgeaktion wird ein Eintrag in der Eigenschaftentabelle erzeugt. Jede Eingabe kann eine beliebige zugeordnete Anzahl an Folgeaktionen haben.
OnDialogClosed	Dieses Ereignis tritt ein, wenn innerhalb der Visualisierung eines der für Benutzereingaben vorher geöffneten Dialogfenster geschlossen wird.
OnMouseClicked	Dieses Mausereignis tritt ein, wenn der Cursor auf ein Element zeigt und ein vollständiger Mausklick (Drücken und Freigeben der Maustaste) ausgeführt wird.
OnMouseDown	Dieses Mausereignis tritt ein, wenn die Maustaste gedrückt wird, während der Cursor auf ein Element zeigt. (Drücken der Maustaste)

OnMouseEnter	Dieses Mausereignis tritt ein, wenn der Cursor auf das Element gezogen wird.
OnMouseLeave	Dieses Mausereignis tritt ein, wenn der Cursor das Element verlässt.
OnMouseMove	Dieses Mausereignis tritt ein, wenn der Cursor innerhalb des Elements bewegt wird.
OnMouseUp	Dieses Mausereignis tritt ein, wenn die Maustaste losgelassen wird. Das vorher ausgeführte Drücken der Maustaste geschieht dabei allerdings außerhalb des Elements.

Eingabekonfiguration: Tasten Mit **Tasten** kann konfiguriert werden, dass beim Eintreten des Ereignisses "Tasten" der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier kann eine boolesche Variable zugewiesen werden, deren Wert TRUE wird, wenn die Maustaste gedrückt wird, während der Cursor auf das Element zeigt. Der Wert wird wieder FALSE, wenn die Maustaste wieder losgelassen wird.
FALSE tasten	Wenn diese Option aktiviert ist, wird das oben beschriebene Verhalten für die eingetragene Variable umgekehrt. Das heißt, wenn die Maustaste gedrückt wird, wird der Variablenwert auf FALSE gesetzt, wenn die Taste wieder losgelassen wird, wird der Variablenwert auf TRUE gesetzt.
Beim Betreten tasten, falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert: Wird bei gedrückter Maustaste der Elementbereich verlassen, wird die Variable FALSE und wird wieder TRUE, wenn in den Elementbereich zurückgekehrt wird. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" (also an das Element gebunden) ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Umschalten Mit **Umschalten** kann konfiguriert werden, dass bei diesem Ereignis der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier ist eine boolesche Variable zuzuweisen. Die Eingabehilfe kann über Schaltfläche  geöffnet werden. Der Wert der Variablen wird bei jedem Mausklick auf das Element zwischen TRUE und FALSE umgeschaltet. Wird der Cursor bei gedrückter Maustaste aus dem Element herausbewegt, dann erfolgt kein Toggeln. Damit kann die begonnene Umschaltteingabe abgebrochen werden.
Umschalten beim Loslassen falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert. Wird bei gedrückter Maustaste der Elementbereich verlassen und dann die Maustaste losgelassen, wird die Variable umgeschaltet. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Hotkey Mit **Hotkey** kann eine Taste, auch mit Zusatz Taste, definiert und mit einer bestimmten Nachfolgeaktion (MouseDown, MouseUp) verknüpft werden, die bei einem Ereignis 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 unter Registerkarte 'Tastaturkonfiguration' der Visualisierung verwaltet (Kap. 20.3.4 "Registerkarte 'Tastaturkonfiguration'" auf Seite 839). Änderungen werden immer zwischen diesem und dem Elementeigenschaften-Editor synchronisiert.

Eigenschaft	Beschreibung Wert
Taste	Zuweisen einer Taste. Eine Auswahlliste stellt alle aktuell unterstützten Tasten bereit wie zum Beispiel <M>.
Eingabe(n)	Definieren des Ereignisses, das ausgeführt werden soll, wenn die Taste beziehungsweise Taste und Modifizierer verwendet werden. <i>Mögliche Werte, die in einer Auswahlliste bereitstehen:</i> - Keine Aktion - MouseDown-Aktion: OnMouseDown-Aktion, wenn die Taste gedrückt wird. - MouseUp-Aktion: OnMouseUp-Aktion, wenn die Taste freigelassen wird. - MouseDown/MouseUp-Aktion: OnMouseUp- und OnMouseDown-Aktion, wenn die Taste gedrückt oder losgelassen wird.
Umschalten	Wenn diese Option aktiviert wird, muss die Taste mit <Umschalt> kombiniert verwendet werden
Steuerung	Wenn diese Option aktiviert wird, muss die Taste mit <Strg> kombiniert verwendet werden
Alt	Wenn diese Option aktiviert wird, muss die Taste mit < Alt> kombiniert verwendet werden

Elementeigenschaft 'Zustandsvariablen'

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Element 'Groupbox'

Elemente können mit Drag-and-Drop in eine Groupbox gezogen werden um eine optische Einheit zu bilden. Sie können mehrmals verschachtelt werden.

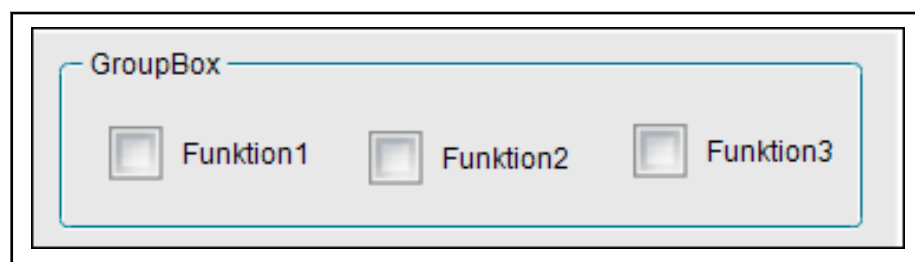


Abb. 20-49: Groupbox

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
-  Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	GroupBox

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Elementeigenschaft 'Texte'

Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Text	Geben Sie einen Text ein. Er wird verwendet um das Element zu beschriften. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Textvariable zugewiesen.
Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Tooltip-Variable zugewiesen.

Texteigenschaften

Diese Eigenschaften werden für eine statische Definition der Schriftart verwendet.

Eigenschaft	Beschreibung Wert
Schriftart	<i>Auswahl aus vordefinierten Schriftarten:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet sich der Dialog für benutzerdefinierte Eigenschaften der Schriftart.
Farbe Schriftart	Auszuwählen ist eine Schriftartfarbe für das Element. Entweder aus der Auswahlliste oder durch Setzen im sich öffnenden Dialog, wenn  geklickt wird.

Elementeigenschaft 'Zustandsvariablen'

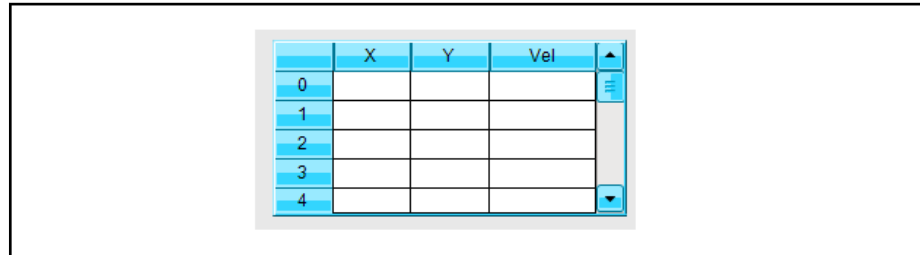
Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Element 'Tabelle'

Inhalt dieser Seite:

- ["Elementeigenschaften" auf Seite 953](#)
- ["Daten der Tabelle zuweisen" auf Seite 957](#)
- ["Eigenschaften einer Tabelle konfigurieren" auf Seite 958](#)

Eine Tabelle kann in einer Visualisierung eingefügt werden, um ein- oder zweidimensionale Arrays, Strukturen oder lokale Variablen einer POU darzustellen.



	X	Y	Vel
0			
1			
2			
3			
4			

Abb. 20-50: Tabelle ohne zugewiesenen Datenarray, Stil 8

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Element spezifische Mauseingabe:

- Die Spaltenbreite kann direkt geändert werden, indem der Teilungsstrich zwischen zwei Spalten in die gewünschte Position gezogen wird während der Cursor die Form  hat.
- Eine Tabellenspalte kann **dupliziert** werden, indem <Strg> und - während die Taste gedrückt gehalten wird - auf den Kopf der jeweiligen Spalte geklickt wird.
- Eine Tabellenspalte kann **gelöscht** werden, indem <Strg> + <Umschalt-taste> und - während die Tasten gedrückt gehalten werden - auf den Kopf der jeweiligen Spalte geklickt wird.

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,

- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)



Wenn Sie die Größe der Variablen, auf der die Tabelle basiert (Array-Grenzen, Anzahl der Strukturkomponenten etc.) im IEC-Code verändert, aber das betroffene Tabellenelement in der Visualisierung noch nicht aktualisiert haben, wird eine entsprechende Compiler-Fehlermeldung ausgegeben.

Eine Möglichkeit ist dann, die Eigenschaftenkonfiguration zu öffnen und die Tabelle z.B. durch Drücken der <Eingabe>-Taste im Wertefeld der Eigenschaft "Datenarray" anzupassen.

Eine viel komfortablere Möglichkeit ist jedoch einen Doppelklick auf die Fehlermeldung auszuführen. Dies öffnet eine Meldungsbbox, in der Sie **Ja** oder **Nein** zum unmittelbaren Aktualisieren der Tabellenkonfiguration vorgeben können.

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Tabelle

Datenarray	Hier wird die zu visualisierende Variable mit vollständigem Pfad spezifiziert. Die Struktur der Variablen bestimmt die Anzahl der Spalten und Reihen der Tabelle. Wenn die Anzahl der Array-Elemente der Variable variiert, verwenden Sie Max. Array-Index . Um die Tabelle zu aktualisieren, falls sich die Größe der Array- oder Strukturvariablen geändert hat, setzen Sie den Cursor in das Datenarray-Wertefeld und drücken <Eingabe>.
Max. Array-Index	Zuzuweisen ist eine numerische Variable, die dynamisch den maximalen Arrayindex festlegt.



Wird der **Datenarray**-Wert geändert, also eine neue Variable zugewiesen, werden die weiteren Tabelleneigenschaften auf ihren Standardwert zurückgesetzt.

Elementeigenschaft 'Spalten'

Das Tabellenelement visualisiert eine Variable in Tabellenansicht. Der Index des Arrays bzw. die Komponente der Struktur wird in einer Spalte bzw. Zeile dargestellt, bei zweidimensionalen Arrays bzw. einem Array von Strukturen wird in mehreren Spalten visualisiert. Hier wird das Aussehen der Tabellenspalten definiert, in welchen die Werte der einzelnen Arrayelemente/Struktu-

komponenten/Variablen angezeigt werden. Für jede zu einem bestimmten Index gehörenden Spalte ist eine eigene Konfiguration möglich.

Spalte [<n>]	Gemäß der Variablenstruktur, definiert in Datenarray, wird die Anzahl an Spalten identifiziert und in <n> indiziert. <code>var_data_array: ARRAY [2..8] OF INT:</code> eine Spalte Column [0] wird identifiziert. <code>var_data_array: ARRAY[1..3][1..10] OF INT:</code> drei Spalten [0], [1] und [2] werden identifiziert
Zeilenkopf anzeigen	Checkbox, um die Anzeige der Beschriftung der Zeilen in Form von zugeordneten Zeilenindizes ein- bzw. auszuschalten
Spaltenkopf anzeigen	Checkbox, um die Anzeige der Beschriftung der Spalten in Form von zugeordneten Zeilenindizes ein- bzw. auszuschalten
Zeilenhöhe	Höhe der Tabellenzeilen in Pixel
Breite Zeilenkopf	Breite der Spalte, die den Zeilenkopf enthält, in Pixel
Größe des Scrollbars	Breite des vertikalen bzw. Höhe des horizontalen Scrollbalkens in Pixel

Elementeigenschaft 'Spalten:
Spalte [<n>]'

Spaltenüberschrift	Standardmäßig wird der Name des Array/der Struktur mit dem zur Spalte gehörenden Index/Strukturkomponente als Überschrift genommen. Die Spaltenbeschriftung kann hier geändert werden, indem ein neuer Titel eingetragen wird.
Breite	Spaltenbreite in Pixel
Bildspalte	Wird diese Option aktiviert, können in der Spalte Bilder angezeigt werden. Es werden Bilder der globalen Bildersammlung oder von benutzerdefinierten Bildersammlungen verwendet. Dabei stehen in den Zellen der Tabelle die Bild-IDs, wie in der Bildersammlung definiert.
Bildkonfiguration - Füllmodus - Transparenz - Transparenzfarbe	Füllmodus <code>Zelle ausfüllen</code>: Das Bild passt sich an die Größe der Zelle ohne Berücksichtigung des Höhen-Breiten-Verhältnisses an. <code>Zentriert</code>: Das Bild wird in der Zelle zentriert und behält seine Proportionen; d.h. das Verhältnis Höhe-Breite bleibt erhalten. Ist Transparenz aktiviert, wird die Farbe, die in Transparenzfarbe angegeben ist, transparent dargestellt. Transparenzfarbe: Einzugeben ist eine Farbe. Sie kann auch mit Hilfe von , das einen Farbauswahldialog öffnet, ausgewählt werden. Sie wird transparent dargestellt, wenn Transparenz aktiviert ist.
Textausrichtung der Überschrift	<i>Ausrichtung des Spaltenkopfs:</i> - Links - Zentriert - Rechts
Vorlage verwenden	Bei Aktivierung der Option wird in jede Zelle dieser Tabellenspalte ein weiteres Visualisierungselement (vom Typ Rechteck , Abgerundetes Rechteck oder Ellipse) eingefügt. Die Eigenschaftensliste wird automatisch um die Eigenschaften dieses Elements unter Vorlage erweitert.

Textausrichtung der Überschrift aus der Vorlage	Die Aktivierung dieser Option hat nur bei gleichzeitiger Aktivierung von Vorlage verwenden eine Auswirkung. Bei Aktivierung werden die in der eingefügten Vorlage getroffenen Einstellungen für Schrift(-Größe) und Ausrichtung auch auf die Spaltenüberschrift angewendet.
Vorlage	Dieser Eintrag ist nur vorhanden, wenn Vorlage verwenden aktiviert ist. Unter Vorlage werden sukzessive die Eigenschaften aller Elemente, die der Spalte zugewiesen wurden, aufgelistet und können dort, wie unter Kap. "Element 'Tabelle'" auf Seite 952 beschrieben, editiert werden.

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Texteigenschaften Diese Eigenschaften werden für eine statische Definition der Schriftart verwendet.

Eigenschaft	Beschreibung Wert
Horizontale Ausrichtung	<i>Definieren der horizontalen Ausrichtung des Textes durch Auswahl aus:</i> • Links • Zentriert • Rechts
Vertikale Ausrichtung	<i>Definieren der vertikalen Ausrichtung des Textes durch Auswahl aus:</i> • Oben • Zentriert • Unten
Schriftart	<i>Auswahl aus vordefinierten Schriftarten:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet sich der Dialog für benutzerdefinierte Eigenschaften der Schriftart.
Farbe Schriftart	Auszuwählen ist eine Schriftartfarbe für das Element. Entweder aus der Auswahlliste oder durch Setzen im sich öffnenden Dialog, wenn  geklickt wird.

Dynamische Texte Diese Parameter dienen der Definition von dynamischen Texten, die aus Textlisten stammen (dies erlaubt Sprachumschaltung) ([Kap. 18.2.15 "Menü 'Textliste'" auf Seite 727](#)).

Eigenschaft	Beschreibung Wert
Textliste	Name der Textliste (String) wie im POU- oder Geräte-Fenster verwendet; z.B. <code>errorlist</code>
Textindex	Index (ID) des Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. <code>'M1'</code> oder String-Variable, z.B. <code>error_22</code> .
Tooltipindex	Index (ID) des Tooltip-Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. <code>'T1'</code> oder String-Variable, z.B. <code>error_Tooltip_22</code> .

Elementeigenschaft 'Schriftartvariablen' Schriftartvariablen werden bei dynamischen Schriftdefinitionen des Elementtextes über Projektvariablen verwendet. Statischen Definitionen werden unter **Texteigenschaften ► Schriftart** konfiguriert.

Schriftname	Hier ist eine Variable Typ <code>STRING</code> anzugeben, die den Schriftnamen enthält, der für die Beschriftung des Elements verwendet werden soll. (Namensangabe wie im Standard-Schrifttypen-Dialog). Beispiel: <code>PlcProg.stFontVar := 'Arial';</code>
Größe	Eingabe einer Variablen Typ <code>INT</code> , die die Größe des Elementtextes in Pixel enthält, so wie im Standarddialog Schriftart. Beispiel: <code>PlcProg.iFontHeight := 16;</code>
Flags	Eingabe einer Variable Typ <code>DWORD</code> , die mit Hilfe der unten genannten Flag-Werte die Schriftart-Darstellung festlegt. Eine kombinierte Definition kann erreicht werden, indem die jeweiligen Flag-Werte addiert werden und die Summe angegeben wird. • 1: kursiv • 2: fett • 4: unterstrichen • 8: gelöscht Beispiel für einen Text, der fett und unterstrichen dargestellt wird: <code>PlcProg.dwFontType := 6;</code>
Zeichensatz	Der Zeichensatz, der für die Schriftart verwendet werden soll, kann über die Standard-Zeichensatznummer definiert werden. Mit einer <code>DWORD</code> -Variablen kann diese Nummer angegeben werden (siehe auch die Skript -Definition im Standard-Schriftartdialog)
Farbe	Eingabe einer <code>DWORD</code> -Variable, in der die Farbe des Elementtextes festgelegt wird.
Flags für Textausrichtung	Variable mit ganzzahligem Datentyp, zum Beispiel <code>PlcProg.dwTextAlignment</code> . Die Visualisierung wertet zur Laufzeit die Variable aus und richtet dementsprechend den Text aus. <i>Textausrichtung</i> • 0: oben links • 1: horizontal zentriert • 2: rechts • 4: vertikal zentriert • 8: unten

**Elementeigenschaft
'Zustandsvariablen'**

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Elementeigenschaft 'Selektion'

Selektionsfarbe	Füllfarbe für selektierte Tabellenzellen.
Selektionstyp	<i>Definiert, wie beim einem Klick auf eine Tabellenzelle die Auswahl erfolgt:</i> - Keine Auswahl: Keine Auswahl - Zellenauswahl: nur die angeklickte Zelle ist ausgewählt. - Zeilenauswahl: Zeilenauswahl, das heißt die Zeile, in der sich die angeklickte Zelle befindet, ist selektiert. - Spaltenauswahl: Spaltenauswahl, das heißt die Spalte, in der sich die angeklickte Zelle befindet, ist selektiert. - Zeilen- und Spaltenauswahl: Zeilen- und Spaltenauswahl, das heißt die Zeile und Spalte, in der sich die angeklickte Zelle befindet, ist selektiert.
Rahmen um ausgewählte Zellen	Ist diese Option aktiviert, wird um die selektierten Zellen ein Rahmen gezeichnet.
Variable für Spaltenauswahl	Variable Typ INT, die den Array-Index der Spalte der selektierten Zelle enthält. Im Fall, dass der Datenarray auf eine Struktur zeigt, werden die Strukturkomponenten beginnend mit 0 indiziert. Zu beachten ist, dass dieser Index nur dann die korrekte Position im Array repräsentiert, wenn keine Spalten von der Anzeige in der Tabelle ausgenommen wurden.
Variable für Zeilenauswahl	Variable Typ INT, die den Array-Index der Zeile der selektierten Zelle enthält.
Variable für Gültigkeit der Spaltenauswahl	Boolesche Variable, die dann TRUE ist, wenn Variable für Spaltenauswahl einen gültigen Wert enthält.
Variable für Gültigkeit der Zeilenauswahl	Boolesche Variable, die dann TRUE ist, wenn Variable für Zeilenauswahl einen gültigen Wert enthält.

Daten der Tabelle zuweisen

Zunächst ist die in Tabellenansicht zu visualisierende Variable zu spezifizieren. Dazu wird in **Datenarray** eine Variable mit vollständigem Pfad eingetragen. Die Struktur der Variable bestimmt die Anzahl an Zeilen und Spalten der Tabelle.

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, eine Struktur wird durch eine einzige Zeile repräsentiert, wohingegen ein Array von Strukturen 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.

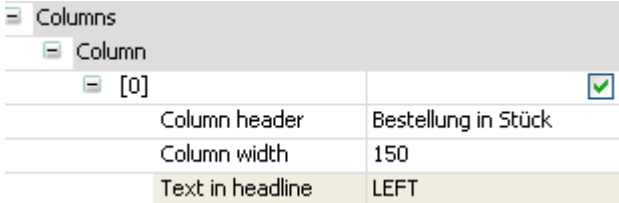
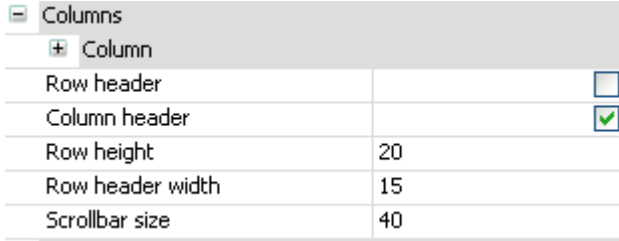
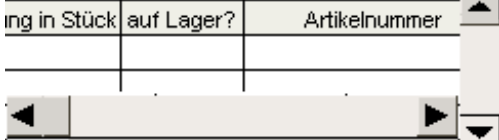
Dann ist eine Konfiguration der Tabelle mit Hilfe ihrer weiteren Eigenschaften möglich.

Deklaration der Variablen	Entstehende Tabelle																																			
arrDim1 : ARRAY[2..5] OF INT;	<table><tr><th></th><th>arrDim1[INDEX]</th></tr><tr><td>2</td><td></td></tr><tr><td>3</td><td></td></tr><tr><td>4</td><td></td></tr><tr><td>5</td><td></td></tr></table> Beachten Sie, dass die Indizierung in der Zeilenbeschriftung mit den Indizes der ersten Dimension des Arrays übereinstimmt; infolgedessen startet die Nummerierung der Zeilen in diesem Beispiel mit einer "2"!		arrDim1[INDEX]	2		3		4		5																										
	arrDim1[INDEX]																																			
2																																				
3																																				
4																																				
5																																				
arrDim2 : ARRAY[0..2, 0..3] OF INT;	<table><tr><th></th><th>arrDim2[0,INDEX]</th><th>arrDim2[1,INDEX]</th><th>arrDim2[2,INDEX]</th></tr><tr><td>0</td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td></td></tr></table>		arrDim2[0,INDEX]	arrDim2[1,INDEX]	arrDim2[2,INDEX]	0				1				2				3																		
	arrDim2[0,INDEX]	arrDim2[1,INDEX]	arrDim2[2,INDEX]																																	
0																																				
1																																				
2																																				
3																																				
TYPE MyStruct : STRUCT iNo: INT; bBool : BOOL; sText:STRING; END_STRUCT END_TYPE varStruct: MyStruct;	<table><tr><th></th><th>varStruct.iNo</th><th>varStruct.bBool</th><th>varStruct.sText</th></tr><tr><td>0</td><td></td><td></td><td></td></tr></table>		varStruct.iNo	varStruct.bBool	varStruct.sText	0																														
	varStruct.iNo	varStruct.bBool	varStruct.sText																																	
0																																				
arrStruct : ARRAY[0..3] OF MyStruct;	<table><tr><th></th><th>arrStruct[INDEX].iNo</th><th>arrStruct[INDEX].bBool</th><th>arrStruct[INDEX].sText</th></tr><tr><td>0</td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td></td></tr></table>		arrStruct[INDEX].iNo	arrStruct[INDEX].bBool	arrStruct[INDEX].sText	0				1				2				3																		
	arrStruct[INDEX].iNo	arrStruct[INDEX].bBool	arrStruct[INDEX].sText																																	
0																																				
1																																				
2																																				
3																																				
varFB: TON;	<table><tr><th></th><th>varFB.IN</th><th>varFB.PT</th><th>varFB.Q</th><th>varFB.ET</th><th>varFB.M</th><th>varFB.StartTime</th></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		varFB.IN	varFB.PT	varFB.Q	varFB.ET	varFB.M	varFB.StartTime	0																											
	varFB.IN	varFB.PT	varFB.Q	varFB.ET	varFB.M	varFB.StartTime																														
0																																				
arrFB : ARRAY[0..3] OF TON;	<table><tr><th></th><th>arrFB[INDEX].IN</th><th>arrFB[INDEX].PT</th><th>arrFB[INDEX].Q</th><th>arrFB[INDEX].ET</th><th>arrFB[INDEX].M</th><th>arrFB[INDEX].StartTime</th></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		arrFB[INDEX].IN	arrFB[INDEX].PT	arrFB[INDEX].Q	arrFB[INDEX].ET	arrFB[INDEX].M	arrFB[INDEX].StartTime	0							1							2							3						
	arrFB[INDEX].IN	arrFB[INDEX].PT	arrFB[INDEX].Q	arrFB[INDEX].ET	arrFB[INDEX].M	arrFB[INDEX].StartTime																														
0																																				
1																																				
2																																				
3																																				

Tab. 20-7: Zugewiesene Daten und die daraus entstehende Tabelle

Eigenschaften einer Tabelle konfigurieren

Neben einigen grundlegenden 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, die Auswirkungen sind jeweils in der rechten Spalte der unten stehenden Tabelle aufgezeigt.

Geänderte Eigenschaften	Daraus entstehende Tabelle
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  Es gibt zwei Möglichkeiten, die Breite der Spalten zu ändern: 1. In den Elementeigenschaften: Eintragen der gewünschten Breite in Spalte ► Breite 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 Arrayindizes beinhaltet, fehlt. Das Bild rechts zeigt zudem die verbreiterten Bildlaufleisten (Größe des Scrollbars), welche zur Tabelle hinzugefügt werden, sobald diese auf eine Größe unter der Summe der Zeilenhöhen bzw. der Spaltenbreiten geschrumpft wird. 	

Geänderte Eigenschaften

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). So 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 in **Schriftart** öffnet. Beachten Sie, dass die Vorgabe für die horizontale Textausrichtung (**Textausrichtung der Überschrift**) nur für die Zeilenbeschriftung gilt; die Ausrichtung der Texte in der einzelnen Tabellenzellen folgt in den hierfür getroffenen Einstellungen in den Spalteneigenschaften.

Daraus entstehende Tabelle

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.

Geänderte Eigenschaften	Daraus entstehende Tabelle																																																																				
Die Tabellenfelder selbst können zeilenweise editiert werden: Aktivieren Sie zu diesem Zweck für die ausgewählte Spalte das Kontrollkästchen Vorlage verwenden, woraufhin innerhalb der Spalteneigenschaften ein weiterer Punkt Vorlage eingefügt und aufgeklappt wird: <table><tr><td>Use template</td><td>☒</td></tr><tr><td>Use text alignment in headline</td><td>☐</td></tr><tr><td colspan="2">Template</td></tr><tr><td>Type of element</td><td>VISU_ST_RECTANGLE</td></tr><tr><td colspan="2">Colors</td></tr><tr><td colspan="2">Normalstate</td></tr><tr><td>Framecolor</td><td> Coral</td></tr><tr><td>Fillcolor</td><td> Bisque</td></tr><tr><td colspan="2">Alarmstate</td></tr><tr><td>Framecolor</td><td> OrangeRed</td></tr><tr><td>Fillcolor</td><td> LightSalmon</td></tr><tr><td colspan="2">Elementlook</td></tr><tr><td>Linewidth</td><td>1</td></tr><tr><td>Fillattributes</td><td>BS_SOLID</td></tr><tr><td>Frameattributes</td><td>PS_DASH</td></tr><tr><td colspan="2">Texts</td></tr><tr><td>Text</td><td>%s</td></tr><tr><td>Tooltip</td><td></td></tr><tr><td colspan="2">Text properties</td></tr><tr><td>HorizontalAlignment</td><td>RIGHT</td></tr><tr><td>VerticalAlignment</td><td>VCENTER</td></tr><tr><td>Font</td><td> Copperplate Gothic Light; 1</td></tr><tr><td colspan="2">Text variables</td></tr><tr><td>Text Variable</td><td>PLC_PRG.arrStruct[INDEX].iNo</td></tr></table>	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; 1	Text variables		Text Variable	PLC_PRG.arrStruct[INDEX].iNo	In unten stehendem Beispiel wurde für die erste und dritte Spalte ein Template benutzt, wobei das Template der ersten Spalte die links aufgezeigten Eigenschaften besitzt. <table><tr><th></th><th>Bestellung in Stück</th><th>auf Lager?</th><th>Artikelnummer</th></tr><tr><td>0</td><td>250</td><td>TRUE</td><td>DD32F7098367</td></tr><tr><td>1</td><td>480</td><td>FALSE</td><td>DD33F1782648</td></tr><tr><td>2</td><td>1500</td><td>TRUE</td><td>DD45G2637526</td></tr><tr><td>3</td><td>2500</td><td>TRUE</td><td>AD70F1829388</td></tr></table>		Bestellung in Stück	auf Lager?	Artikelnummer	0	250	TRUE	DD32F7098367	1	480	FALSE	DD33F1782648	2	1500	TRUE	DD45G2637526	3	2500	TRUE	AD70F1829388
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; 1																																																																				
Text variables																																																																					
Text Variable	PLC_PRG.arrStruct[INDEX].iNo																																																																				
	Bestellung in Stück	auf Lager?	Artikelnummer																																																																		
0	250	TRUE	DD32F7098367																																																																		
1	480	FALSE	DD33F1782648																																																																		
2	1500	TRUE	DD45G2637526																																																																		
3	2500	TRUE	AD70F1829388																																																																		

Drei Typen von Vorlagen stehen zur Auswahl: standardmäßig ist ein Rechteck eingetragen, das jedoch durch Klicken in das entsprechende Feld in ein Rechteck mit abgerundeten Ecken oder einer Ellipse geändert werden kann. Die zugehörigen Konfigurationsmöglichkeiten zeigen die Elementeneigenschaften der ausgewählten Vorlage: wie gewöhnlich können Sie die Füll- und Rahmenfarben im Normal- und Alarmzustand wählen, die auch mit Hilfe einer Variablen getoggelt 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 in **Textvariable** 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.

Geänderte Eigenschaften	Daraus entstehende Tabelle																																				
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 Selektion festgelegt: <table><tr><th colspan="2">Selection</th></tr><tr><td>Selection color</td><td> HotPink</td></tr><tr><td>Apply to columns</td><td></td></tr><tr><td>Selection type</td><td>SEL_ROW</td></tr><tr><td>Frame around selected cells</td><td></td></tr><tr><td>Variable for selection X</td><td>PLC_PRG.SelectedPosX</td></tr><tr><td>Variable for selection Y</td><td>PLC_PRG.SelectedPosY</td></tr><tr><td>Variable for valid selection</td><td>PLC_PRG.IsSelected</td></tr></table> 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.	Selection		Selection color	 HotPink	Apply to columns		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	PLC_PRG.IsSelected	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: <table><tr><th></th><th>Bestellung in Stück</th><th>auf Lager?</th><th>Artikelnummer</th></tr><tr><td>0</td><td>250</td><td>TRUE</td><td>DD32F7098367</td></tr><tr><td>1</td><td>480</td><td>FALSE</td><td>DD33F1782648</td></tr><tr><td>2</td><td>1500</td><td>TRUE</td><td>DD45G2637526</td></tr><tr><td>3</td><td>2500</td><td>TRUE</td><td>AD70F1829388</td></tr></table>		Bestellung in Stück	auf Lager?	Artikelnummer	0	250	TRUE	DD32F7098367	1	480	FALSE	DD33F1782648	2	1500	TRUE	DD45G2637526	3	2500	TRUE	AD70F1829388
Selection																																					
Selection color	 HotPink																																				
Apply to columns																																					
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	PLC_PRG.IsSelected																																				
	Bestellung in Stück	auf Lager?	Artikelnummer																																		
0	250	TRUE	DD32F7098367																																		
1	480	FALSE	DD33F1782648																																		
2	1500	TRUE	DD45G2637526																																		
3	2500	TRUE	AD70F1829388																																		

Geänderte Eigenschaften	Daraus entstehende Tabelle																																							
Wenn Sie den Schalter Anwenden drücken, so werden die Alarmfarben der speziellen Vorlage durch die in Selektionsfarber" 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	Nach Anwenden der Auswahlfarbe auch für die Templates erhält die Selektion ein verändertes Aussehen: <table><tr><th>Bestellung in Stück</th><th>auf Lager?</th><th>Artikelnummer</th></tr><tr><td>0</td><td>250</td><td>TRUE</td><td>DD32F7098367</td></tr><tr><td>1</td><td>480</td><td>FALSE</td><td>DD33F1782648</td></tr><tr><td>2</td><td>1500</td><td>TRUE</td><td>DD45G2637526</td></tr><tr><td>3</td><td>2500</td><td>TRUE</td><td>AD70F1829388</td></tr></table>	Bestellung in Stück	auf Lager?	Artikelnummer	0	250	TRUE	DD32F7098367	1	480	FALSE	DD33F1782648	2	1500	TRUE	DD45G2637526	3	2500	TRUE	AD70F1829388																				
Bestellung in Stück	auf Lager?	Artikelnummer																																						
0	250	TRUE	DD32F7098367																																					
1	480	FALSE	DD33F1782648																																					
2	1500	TRUE	DD45G2637526																																					
3	2500	TRUE	AD70F1829388																																					
Auswahlart bestimmt, welche Elemente durch Anklicken einer einzelnen Tabellenzelle selektiert werden sollen: nachdem wir Zeilenauswahl 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 Indizes des visualisierten Arrays. Diese Indizes können dann in Variablen gespeichert werden, die hierfür in den letzten drei Zeilen des Abschnitts Selektion 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.	Die verknüpften Variablen erhalten als Wert den zur ausgewählten Zelle gehörenden Arrayindex: <table><tr><th>Expression</th><th>Type</th><th>Value</th></tr><tr><td> arrDim1</td><td>ARRAY [2..5] OF INT</td><td></td></tr><tr><td> arrDim2</td><td>ARRAY [0..2, 0..3] O...</td><td></td></tr><tr><td> varStruct</td><td>MyStruct</td><td></td></tr><tr><td> arrStruct</td><td>ARRAY [0..3] OF My...</td><td></td></tr><tr><td> varFB</td><td>TON</td><td></td></tr><tr><td> arrFB</td><td>ARRAY [0..3] OF TON</td><td></td></tr><tr><td> iCounter</td><td>INT</td><td>745</td></tr><tr><td> iRow</td><td>INT</td><td>0</td></tr><tr><td> iCol</td><td>INT</td><td>7</td></tr><tr><td> SelectedPosX</td><td>INT</td><td>2</td></tr><tr><td> SelectedPosY</td><td>INT</td><td>1</td></tr><tr><td> IsSelected</td><td>BOOL</td><td>TRUE</td></tr></table>	Expression	Type	Value	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	745	iRow	INT	0	iCol	INT	7	SelectedPosX	INT	2	SelectedPosY	INT	1	IsSelected	BOOL	TRUE
Expression	Type	Value																																						
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	745																																						
iRow	INT	0																																						
iCol	INT	7																																						
SelectedPosX	INT	2																																						
SelectedPosY	INT	1																																						
IsSelected	BOOL	TRUE																																						

(noch nicht implementiert: \$ROWCONST\$, \$COLCONST\$, invisible, deactivate inputs)

Element 'Textfeld'

Mit diesem Element ist es möglich Text anzuzeigen, der direkt in den Eigenschaften oder variabel in **Textvariable** zugewiesen wird. Im Gegensatz zum Rechteck kann der Rahmen mit Schatten dargestellt werden.

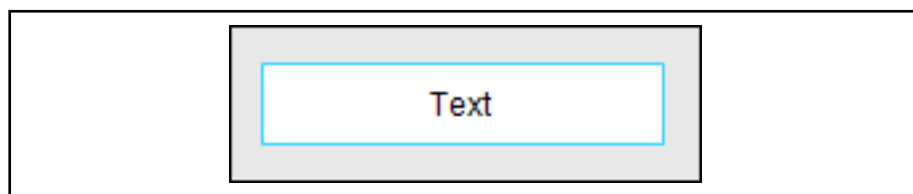


Abb. 20-51: Textfeld

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
-  Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Textfeld

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Farben Eine Farbe wird mittels Hexzahl definiert, die sich aus Rot/Grün/Blau (RGB) Anteilen zusammensetzt. Für jede dieser drei Farben sind 256 (0-255) Werte verfügbar. Hier sind die Einstellungen **statisch**. Die Farbe kann aus einer Auswahlliste oder dem Farbauswahldialog, der über Button  öffnet, gewählt werden. Beispiel:  0; 128; 64

Eigenschaft	Beschreibung Wert
Normalzustand Rahmenfarbe Füllfarbe	Auszuwählen ist eine Rahmen- und Füllfarbe für Normalzustand. Im Fall, dass die Variable definiert in Farbvariablen ► Farbumschlag FALSE ist, ist das Element in Normalzustand.
Alarmzustand Rahmenfarbe Füllfarbe	Auszuwählen ist eine Rahmen- und Füllfarbe, mit der das Element im Alarmzustand dargestellt wird. Dies ist der Fall, wenn die Variable in Farbvariablen ► Farbumschlag TRUE ist.

Aussehen

Eigenschaft	Beschreibung Wert
Linienstärke	Stärke der Rahmenlinie in Pixel. 0 kodiert das Gleiche wie 1 und setzt die Linienstärke auf 1 Pixel. Wird kein Rahmen gewünscht, muss man Linienart auf unsichtbar setzen.
Füllart	Auszuwählen ist eines dieser Füllarten angewendet in Füllfarbe - Ausgefüllt: Die Füllfarbe ist sichtbar - Unsichtbar: Die Füllfarbe ist unsichtbar
Linienart	<i>Auszuwählen ist eine der Linienarten angewendet bei der Umrisslinie:</i> - Durchgezogen - Striche - Punkte - Strich Punkt - Strich Punkt Punkt - Unsichtbar: Umrisslinie ist nicht sichtbar.

Schattenart	<i>Art des Schattens der Umrisslinie</i> - Versenkt - Kein Schatten - Erhaben - Aus Stil

Elementeigenschaft 'Texte' Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Text	Geben Sie einen Text ein. Er wird verwendet um das Element zu beschriften. Der Text kann eine Formatierungssequenz, wie zum Beispiel %, enthalten. Die zugehörige Variable ist in Textvariablen ► Textvariable zugewiesen.
Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %, enthalten. Die zugehörige Variable ist in Textvariablen ► Tooltip-Variable zugewiesen.

Texteigenschaften Diese Eigenschaften werden für eine statische Definition der Schriftart verwendet.

Eigenschaft	Beschreibung Wert
Horizontale Ausrichtung	<i>Definieren der horizontalen Ausrichtung des Textes durch Auswahl aus:</i> • Links • Zentriert • Rechts
Vertikale Ausrichtung	<i>Definieren der vertikalen Ausrichtung des Textes durch Auswahl aus:</i> • Oben • Zentriert • Unten
Schriftart	<i>Auswahl aus vordefinierten Schriftarten:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet sich der Dialog für benutzerdefinierte Eigenschaften der Schriftart.
Farbe Schriftart	Auszuwählen ist eine Schriftartfarbe für das Element. Entweder aus der Auswahlliste oder durch Setzen im sich öffnenden Dialog, wenn  geklickt wird.

Elementeigenschaft 'Textvariablen' Sie können Text, der in einer Variablen gespeichert ist, anzeigen. Fügen Sie dazu zuerst eine Formatierungssequenz in dem Text hinzu, der unter der Eigenschaft **Texte** definiert ist. Weisen Sie dann hier eine Variable hinzu. Im Online-Betrieb wird die Formatierungssequenz durch den Inhalt der Variablen ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Beispiel:

Beispiel %f

Geben Sie Ergebnis: %2.5f in der Eigenschaft **Text** ein. Geben Sie rValue in der Eigenschaft **Textvariable** ein. Es muss eine definierte IEC-Variab-

le sein. Dann wird im Online-Betrieb das Element mit Ergebnis: 12.12345 beschriftet, wenn rValue = 12.12345 ist.

Textvariable	Geben Sie eine Variable (Typ ist Standarddatentyp) ein, der die Information enthält, die sie angezeigt haben wollen. Der Typ muss konform mit der Formatierungssequenz in Texte ▶ Text sein.
Tooltipvariable	Geben Sie eine STRING-Variable ein, die den Toolliptext enthält, der angezeigt werden soll. Die Eingabe in Texte ▶ Tooltip muss ein %s enthalten.

Elementeigenschaft 'Farbvariablen'

Die Elementeigenschaft ermöglicht die dynamische Umschaltung der Elementfarben mit Hilfe von IEC-Variablen für Steuerung und Farben.

Die IEC-Variable für Farbe enthält diese als hexadezimaler Farbdefinition, die die Rot-/Grün-/Blauanteile der Farbe im RGB-Farbraum codiert: 16#00RRGGBB. Dabei ist für jeden Farbanteil ein Byte für 256 Farbabstufungen (von 0-255) reserviert. 0 bedeutet dabei, dass kein Farbanteil enthalten ist. 255 bedeutet 100% Farbanteil. Das führende Byte ist immer 16#00.

Beispiel:

Farbvariable

```
dwFillColor := 16#008FE03F;
```

Codierung: Rotanteil: 8F, Grünanteil: E0, Blauanteil: 3F

- grau: dwGrey := 16#00888888
- rot: dwRed := 16#00FF0000
- schwarz: dwBlack := 16#00000000

Farbumschlag	<i>Steuervariable</i> • als IEC-Variable des Typs <code>BOOL</code>, zum Beispiel <code>prgA.bToggleColor</code> Zur Laufzeit steuert die Visualisierung dynamisch die Farbdarstellung des Elements – <code>TRUE</code>: Element in Normalzustand, zum Beispiel in grau mit schwarzem Rahmen – <code>FALSE</code>: Element in Alarmzustand, zum Beispiel in gelb mit rotem Rahmen. • oder als Platzhalter <Umschalten-/Tastenvariable> Für den Farbumschlag ist dabei keine eigene Variable zugewiesen. Stattdessen wird der Farbumschlag mit der Variablen gesteuert, die in Eingabekonfiguration, Umschalten oder Tasten zugewiesen ist. Voraussetzung ist also, dass dort eine Steuervariable konfiguriert ist. Wenn sowohl in Umschalten als auch in Tasten eine Steuervariable konfiguriert ist, wird die in Tasten verwendet. Sie können den Platzhalter entfernen, indem Sie <Umschalten-/Tastenvariable> selektieren und <Entf> wählen. Danach erscheint  im Wertefeld.
Klick auf  im Wertefeld von Farbumschlag	IndraLogic fügt den Platzhalter <Umschalten-/Tastenvariable> ein.
Normalzustand	Farbenvariablen des Elements während eines ungestörten Betriebs
• Rahmenfarbe	IEC-Variable des Typs <code>DWORD</code> für die Rahmenfarbe, zum Beispiel <code>prgA.dwBorderColor</code>

• Füllfarbe	IEC-Variable des Typs <code>DWORD</code> für die Füllfarbe, zum Beispiel <code>prgA.dwFillColor</code>
Alarmzustand	Farbvariablen des Elements während eines Alarms
• Rahmenfarbe	IEC-Variable des Typs <code>DWORD</code> für die Rahmenfarbe, zum Beispiel <code>prgA.dwBorderColorAlarm</code>
• Füllfarbe	IEC-Variable des Typs <code>DWORD</code> für die Rahmenfarbe, zum Beispiel <code>prgA.dwFillColorAlarm</code>

**Elementeigenschaft
'Zustandsvariablen'**

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert <code>TRUE</code> , dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert <code>TRUE</code> , dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Elementeigenschaft 'Eingabekonfiguration'

Hier kann definiert werden, welche Folgeaktionen ausgeführt werden, wenn der Benutzer im Online-Betrieb eine Eingabe im Element vornimmt.

Eingabeereignisse, die ein Element haben kann, sind:

- **OnDialogClosed**
- **OnMouseClicked**
- **OnMouseDown**
- **OnMouseEnter**
- **OnMouseLeave**
- **OnMouseMove**
- **OnMouseUp**

Mit Klick auf **Konfigurieren....** wird der Dialog [Kap. "Eingabekonfiguration" auf Seite 849](#) geöffnet, der ermöglicht, dass Folgeaktionen zugewiesen werden. Man klickt in das Feld neben der zu konfigurierenden Eingabe. Für jede selektierte Folgeaktion wird ein Eintrag in der Eigenschaftentabelle erzeugt. Jede Eingabe kann eine beliebige zugeordnete Anzahl an Folgeaktionen haben.

OnDialogClosed	Dieses Ereignis tritt ein, wenn innerhalb der Visualisierung eines der für Benutzereingaben vorher geöffneten Dialogfenster geschlossen wird.
OnMouseClicked	Dieses Mausereignis tritt ein, wenn der Cursor auf ein Element zeigt und ein vollständiger Mausklick (Drücken und Freigeben der Maustaste) ausgeführt wird.
OnMouseDown	Dieses Mausereignis tritt ein, wenn die Maustaste gedrückt wird, während der Cursor auf ein Element zeigt. (Drücken der Maustaste)
OnMouseEnter	Dieses Mausereignis tritt ein, wenn der Cursor auf das Element gezogen wird.
OnMouseLeave	Dieses Mausereignis tritt ein, wenn der Cursor das Element verlässt.
OnMouseMove	Dieses Mausereignis tritt ein, wenn der Cursor innerhalb des Elements bewegt wird.
OnMouseUp	Dieses Mausereignis tritt ein, wenn die Maustaste losgelassen wird. Das vorher ausgeführte Drücken der Maustaste geschah dabei allerdings außerhalb des Elements.

Eingabekonfiguration: Tasten

Mit **Tasten** kann konfiguriert werden, dass beim Eintreten des Ereignisses "Tasten" der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier kann eine boolesche Variable zugewiesen werden, deren Wert TRUE wird, wenn die Maustaste gedrückt wird, während der Cursor auf das Element zeigt. Der Wert wird wieder FALSE, wenn die Maustaste wieder losgelassen wird.
FALSE tasten	Wenn diese Option aktiviert ist, wird das oben beschriebene Verhalten für die eingetragene Variable umgekehrt. Das heißt, wenn die Maustaste gedrückt wird, wird der Variablenwert auf FALSE gesetzt, wenn die Taste wieder losgelassen wird, wird der Variablenwert auf TRUE gesetzt.
Beim Betreten tasten, falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert: Wird bei gedrückter Maustaste der Elementbereich verlassen, wird die Variable FALSE und wird wieder TRUE, wenn in den Elementbereich zurückgekehrt wird. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" (also an das Element gebunden) ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Umschalten

Mit **Umschalten** kann konfiguriert werden, dass bei diesem Ereignis der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier ist eine boolesche Variable zuzuweisen. Die Eingabehilfe kann über Schaltfläche  geöffnet werden. Der Wert der Variablen wird bei jedem Mausklick auf das Element zwischen TRUE und FALSE umgeschaltet. Wird der Cursor bei gedrückter Maustaste aus dem Element herausbewegt, dann erfolgt kein Toggeln. Damit kann die begonnene Umschalteneingabe abgebrochen werden.
Umschalten beim Loslassen falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert. Wird bei gedrückter Maustaste der Elementbereich verlassen und dann die Maustaste losgelassen, wird die Variable umgeschaltet. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Hotkey

Mit **Hotkey** kann eine Taste, auch mit Zusatzstufe, definiert und mit einer bestimmten Nachfolgeaktion (MouseDown, MouseUp) verknüpft werden, die bei einem Ereignis 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 Tastatenkonfiguration für ein Element wird auch unter Registerkarte 'Tastatenkonfiguration' der Visualisierung verwaltet ([Kap. 20.3.4 "Registerkarte 'Tastaturkonfiguration'" auf Seite 839](#)). Änderungen werden immer zwischen diesem und dem Elementeigenschaften-Editor synchronisiert.

Eigenschaft	Beschreibung Wert
Taste	Zuweisen einer Taste. Eine Auswahlliste stellt alle aktuell unterstützten Tasten bereit wie zum Beispiel <M>.
Eingabe(n)	Definieren des Ereignisses, das ausgeführt werden soll, wenn die Taste beziehungsweise Taste und Modifizierer verwendet werden. <i>Mögliche Werte, die in einer Auswahlliste bereitstehen:</i> - Keine Aktion - MouseDown-Aktion: OnMouseDown-Aktion, wenn die Taste gedrückt wird. - MouseUp-Aktion: OnMouseUp-Aktion, wenn die Taste freigelassen wird. - MouseDown/MouseUp-Aktion: OnMouseUp- und OnMouseDown-Aktion, wenn die Taste gedrückt oder losgelassen wird.
Umschalten	Wenn diese Option aktiviert wird, muss die Taste mit <Umschalt> kombiniert verwendet werden
Steuerung	Wenn diese Option aktiviert wird, muss die Taste mit <Strg> kombiniert verwendet werden
Alt	Wenn diese Option aktiviert wird, muss die Taste mit < Alt> kombiniert verwendet werden

Element 'Scrollbalken'

Wenn ein Anwender zur Laufzeit der Visualisierung das Bildlauffeld auf eine andere Position zieht, setzt die Visualisierung die Eigenschaft **Wert** dementsprechend.

Element **Scrollbalken** im Stil `Flat` style:



Abb. 20-52:

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Element spezifische Mauseingabe:

- Wenn Sie einen Scrollbalken in den Visualisierungseditor ziehen, wird er horizontal dargestellt. Sie können ihn mit Hilfe der Maus in eine andere Größe ziehen. Sobald die Höhe die Breite übersteigt, wird der Scrollbalken vertikal dargestellt und IndraLogic aktualisiert die zugehörige Eigenschaft **Balken ► Ausrichtung** auf `Vertikal`.

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.

- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabebehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Scrollbalken

Wert	IEC-Variable mit ganzzahligem Datentyp, die die Position des Bildlaufeldes im Scrollbalken enthält
Minimaler Wert	Kleinster Wert der Bildlaufleiste
Maximaler Wert	Größter Wert der Bildlaufleiste
Seitengröße	<i>Seitengröße</i> • als fester Wert, zum Beispiel 10 • als IEC-Variable mit ganzzahligem Datentyp. Ein Klick auf  öffnet die Eingabeunterstützung. Voraussetzung: Sichtbar, wenn die Eigenschaft Zum Klick springen nicht aktiviert ist.
Zum Klick springen	Verhalten des Scrollbalkens zur Laufzeit der Visualisierung bei Klick in die Bildlaufleiste: : Der Scrollbalken springt auf die angeklickte Position. : Der Scrollbalken springt um eine Seitengröße in die Richtung, die der Klick vorgibt.

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse

verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Elementeigenschaft 'Balken'

Ausrichtung	<i>Ausrichtung</i> • Horizontal: Der Balken verläuft horizontal • Vertikal: Der Balken verläuft vertikal Sie können die Ausrichtung im Visualisierungseeditor ändern, indem Sie mit der Maus die Breite und Höhe des Scrollbalkens ändern.
Laufrichtung	Auswahlliste variiert abhängig von der Ausrichtung des Scrollbalkens. <i>Horizontal</i> • Von links nach rechts: Skalenanfang ist links. • Rechts nach links: Skalenanfang ist rechts. <i>Vertikal</i> • Unten nach oben: Skalenanfang ist unten. • Von oben nach unten: Skalenanfang ist oben.

Farben Eine Farbe wird mittels Hexzahl definiert, die sich aus Rot/Grün/Blau (RGB) Anteilen zusammensetzt. Für jede dieser drei Farben sind 256 (0-255) Werte verfügbar. Hier sind die Einstellungen **statisch**. Die Farbe kann aus einer Auswahlliste oder dem Farbauswahldialog, der über Button  öffnet, gewählt werden. Beispiel:  0; 128; 64

Eigenschaft	Beschreibung Wert
Farbe	Hier ist eine Farbe, mit der das Element im Normalzustand gefüllt wird, auszuwählen. Dies ist der Fall, wenn die Variable in Farbvariablen ► Farbumschlag FALSE ist. Mit Editieren des Werts ändert das Element gleichzeitig seine Farbe.
Alarmfarbe	Auszuwählen ist eine Farbe, mit der das Element gefüllt wird, wenn das Element in Alarmzustand ist. Dies ist der Fall, wenn die Variable in Farbvariablen ► Farbumschlag TRUE ist.

Elementeigenschaft 'Texte' Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Text	Geben Sie einen Text ein. Er wird verwendet um das Element zu beschriften. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Textvariable zugewiesen.
Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Tooltip-Variable zugewiesen.

Texteigenschaften Diese Eigenschaften werden für eine statische Definition der Schriftart verwendet.

Eigenschaft	Beschreibung Wert
Horizontale Ausrichtung	<i>Definieren der horizontalen Ausrichtung des Textes durch Auswahl aus:</i> • Links • Zentriert • Rechts
Vertikale Ausrichtung	<i>Definieren der vertikalen Ausrichtung des Textes durch Auswahl aus:</i> • Oben • Zentriert • Unten
Schriftart	<i>Auswahl aus vordefinierten Schriftarten:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet sich der Dialog für benutzerdefinierte Eigenschaften der Schriftart.
Farbe Schriftart	Auszuwählen ist eine Schriftartfarbe für das Element. Entweder aus der Auswahlliste oder durch Setzen im sich öffnenden Dialog, wenn  geklickt wird.

Elementeigenschaft 'Textvariablen'

Sie können Text, der in einer Variablen gespeichert ist, anzeigen. Fügen Sie dazu zuerst eine Formatierungssequenz in dem Text hinzu, der unter der Eigenschaft **Texte** definiert ist. Weisen Sie dann hier eine Variable hinzu. Im Online-Betrieb wird die Formatierungssequenz durch den Inhalt der Variablen ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Beispiel:

Beispiel %f

Geben Sie **Ergebnis: %2.5f** in der Eigenschaft **Text** ein. Geben Sie **rValue** in der Eigenschaft **Textvariable** ein. Es muss eine definierte IEC-Variablen sein. Dann wird im Online-Betrieb das Element mit **Ergebnis: 12.12345** beschriftet, wenn **rValue = 12.12345** ist.

Textvariable	Geben Sie eine Variable (Typ ist Standarddatentyp) ein, der die Information enthält, die sie angezeigt haben wollen. Der Typ muss konform mit der Formatierungssequenz in Texte ► Text sein.
Tooltipvariable	Geben Sie eine STRING-Variable ein, die den Toolliptext enthält, der angezeigt werden soll. Die Eingabe in Texte ► Tooltip muss ein %s enthalten.

Dynamische Texte Diese Parameter dienen der Definition von dynamischen Texten, die aus Textlisten stammen (dies erlaubt Sprachumschaltung) ([Kap. 18.2.15 "Menü 'Textliste'" auf Seite 727](#)).

Eigenschaft	Beschreibung Wert
Textliste	Name der Textliste (String) wie im POU- oder Geräte-Fenster verwendet; z.B. <code>errorlist</code>
Textindex	Index (ID) des Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. 'M1' oder String-Variable, z.B. <code>error_22</code> .
Tooltipindex	Index (ID) des Tooltip-Textes wie in der Textliste definiert (String); Direkte Angabe der ID, z.B. 'T1' oder String-Variable, z.B. <code>error_Tooltip_22</code> .

Elementeigenschaft 'Schriftartvariablen' Schriftartvariablen werden bei dynamischen Schriftdefinitionen des Elementtextes über Projektvariablen verwendet. Statischen Definitionen werden unter **Texteigenschaften ► Schriftart** konfiguriert.

Schriftname	Hier ist eine Variable Typ <code>STRING</code> anzugeben, die den Schriftnamen enthält, der für die Beschriftung des Elements verwendet werden soll. (Namensangabe wie im Standard-Schrifttypen-Dialog). Beispiel: <code>PlcProg.stFontVar := 'Arial';</code>
Größe	Eingabe einer Variablen Typ <code>INT</code> , die die Größe des Elementtextes in Pixel enthält, so wie im Standarddialog Schriftart. Beispiel: <code>PlcProg.iFontHeight := 16;</code>
Flags	Eingabe einer Variable Typ <code>DWORD</code> , die mit Hilfe der unten genannten Flag-Werte die Schriftart-Darstellung festlegt. Eine kombinierte Definition kann erreicht werden, indem die jeweiligen Flag-Werte addiert werden und die Summe angegeben wird. • 1: kursiv • 2: fett • 4: unterstrichen • 8: gelöscht Beispiel für einen Text, der fett und unterstrichen dargestellt wird: <code>PlcProg.dwFontType := 6;</code>
Zeichensatz	Der Zeichensatz, der für die Schriftart verwendet werden soll, kann über die Standard-Zeichensatznummer definiert werden. Mit einer <code>DWORD</code> -Variablen kann diese Nummer angegeben werden (siehe auch die Skript -Definition im Standard-Schriftartdialog)

Farbe	Eingabe einer <code>DWORD</code> -Variable, in der die Farbe des Elementtextes festgelegt wird.
Flags für Textausrichtung	Variable mit ganzzahligem Datentyp, zum Beispiel <code>PlcProg.dwTextAlignment</code>. Die Visualisierung wertet zur Laufzeit die Variable aus und richtet dementsprechend den Text aus. <i>Textausrichtung</i> • 0: oben links • 1: horizontal zentriert • 2: rechts • 4: vertikal zentriert • 8: unten

Elementeigenschaft 'Farbvariablen'

Farbumschlag	Boolesche Variable, die die Umschaltung der Elementfarbe steuert. <code>TRUE</code> : Visualisierung stellt Element in Alarmfarbe dar.
Farbe	Variable Typ <code>DWORD</code> für die Elementfarbe
Alarmfarbe	Variable Typ <code>DWORD</code> für die Alarmfarbe

Elementeigenschaft 'Zustandsvariablen'

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert <code>TRUE</code> , dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert <code>TRUE</code>, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Element 'Schieberegler'

Ein Visualisierungsbenutzer kann den Schieberegler mit Hilfe der Maus entlang der Schiebereglerleiste verschieben. Dabei wird einer Variablen ein Wert zugewiesen, der der Position des Reglers innerhalb der Leiste entspricht. Sie können den Wertebereich der Schiebereglerleiste mit **Skalenanfang** und **Skalenende** definieren.

Element **Schieberegler** im Stil `Flat style`:

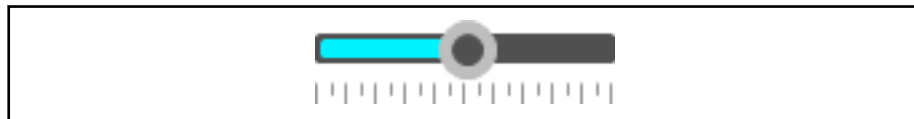


Abb. 20-53:

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Elementspezifische Mauseingabe:

- Wenn Sie einen Schieberegler in den Visualisierungseditor ziehen, wird er horizontal dargestellt. Sie können ihn mit Hilfe der Maus in eine andere Größe ziehen. Sobald die Höhe die Breite übersteigt, wird der Schieberegler vertikal dargestellt und IndraLogic aktualisiert die zugehörige Eigenschaft **Balken ▶ Ausrichtung** auf **Vertikal**.

Benutzereingabe beim Editieren der Elementeigenschaften*Eigenschaften definieren die Eigenart eines Visualisierungselements:*

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Schieberegler

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.

Eigenschaft	Beschreibung Wert
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Variable	IEC-Variable mit beliebigem numerischen Datentyp. Die Visualisierung weist ihr zur Laufzeit den Wert zu, der der Position des Schiebereglers innerhalb der Leiste entspricht. Ein Klick auf  öffnet die Eingabeunterstützung.
Seitengröße	<i>Seitengröße</i> - als fester Wert, zum Beispiel 10 - als IEC-Variable mit ganzzahligem Datentyp. Ein Klick auf  öffnet die Eingabeunterstützung. Voraussetzung: Sichtbar, wenn die Eigenschaft Zum Klick springen nicht aktiviert ist.
Zum Klick springen	Verhalten des Schiebereglers zur Laufzeit der Visualisierung bei Klick in die Schiebereglerleiste: : Der Schieberegler springt auf die angeklickte Position. : Der Schieberegler springt um eine Seitengröße in die Richtung, die der Klick vorgibt.

Elementeigenschaft 'Skala'

Skala anzeigen	 : Der Schieberegler hat eine sichtbare Skala
Skalenanfang	Wert am Skalenanfang und die untere Grenze des Wertebereichs der Schiebereglerleiste. Zum Beispiel: 0 Sie können den Wertebereich der Schiebereglerleiste mit Skalenanfang und Skalenende definieren.
Skalenende	Wert am Skalenende und die obere Grenze des Wertebereichs der Schiebereglerleiste. Zum Beispiel: 100
Hauptskala	Abstand zwischen zwei Skalenstrichen der Grobskala, zum Beispiel 10
Unterskala	Abstand zwischen zwei Strichen der Feinskala, zum Beispiel 2. Sie können die Feinskala ausblenden, indem Sie den Wert auf 0 setzen.
Skalenformat (C-Syntax)	Formatierung der Skalenbeschriftung, zum Beispiel %d s

Elementeigenschaft 'Balken'

Diagrammart	Auswahlliste variiert abhängig von der Ausrichtung des Diagramms. <i>Horizontal</i> • Oben: Skala ist über dem Regler • Unten: Skala ist unter dem Regler • Oben und unten: 2 Skalen rahmen den Regler oben und unten ein <i>Vertikal</i> • Links: Skala ist links vom Regler • Rechts: Skala ist rechts vom Regler • Links und rechts: 2 Skalen rahmen den Regler links und rechts ein
Ausrichtung	<i>Ausrichtung des Schiebereglers</i> • Horizontal • Vertikal Sie können die Ausrichtung im Visualisierungseditor ändern, indem Sie mit der Maus die Breite und Höhe des Scrollbalkens anpassen. Die Ausrichtung ist somit abhängig vom Verhältnis Breite zu Höhe.
Laufrichtung	Auswahlliste variiert abhängig von der Ausrichtung des Schiebereglers. <i>Horizontal</i> • Von links nach rechts: Skalenanfang ist links. • Rechts nach links: Skalenanfang ist rechts. <i>Vertikal</i> • Unten nach oben: Skalenanfang ist unten. • Von oben nach unten: Skalenanfang ist oben.

Elementeigenschaft
'Zustandsvariablen'

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Element 'SpinControl'

Wenn ein Anwender zur Laufzeit der Visualisierung auf die Pfeiltasten des Elements klickt, wird der Ausgabewert des Elements inkrementiert oder dekrementiert. Sie können den Ausgabewert in der Eigenschaft **Variable** konfigurieren.



Abb. 20-54:

Element **SpinControl** im Stil `Flat style`

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	SpinControl

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Variable	IEC-Variable. Wenn zur Laufzeit der Visualisierung ein Anwender auf die Pfeiltasten klickt, wird die IEC-Variable um den Wert in Intervall inkrementiert oder dekrementiert.
Zahlenformat	Formatierung für die Anzeige des Werts. Zum Beispiel <code>Value: %i</code> bei einer Variable des Typs <code>INT</code>
Intervall	Intervall, um das die Variable inkrementiert oder dekrementiert wird

Elementeigenschaft 'Wertebereich'

Minimaler Wert	<i>Untere Grenze des Ausgabewerts</i> - Als fester Wert - Als IEC-Variable des Typs <code>INT</code>:
Maximaler Wert	<i>Obere Grenze des Ausgabewerts</i> - Als fester Wert - Als IEC-Variable des Typs <code>INT</code>

Elementeigenschaft 'Texteigenschaften'

Verwendung von	Standard Stilwerten: Das Element wird mit Standard-Stileigenschaften für die Texteigenschaften gestaltet. Benutzerdefinierte Einstellung: Das Element wird mit den folgenden Eigenschaften gestaltet.

Nur wenn Sie in **Verwendung von** die Option **Benutzerdefinierte Einstellung** ausgewählt haben, erscheinen die folgenden Eigenschaften.

Horizontale Ausrichtung	- Links - Rechts - Zentriert
Vertikale Ausrichtung	- Links - Rechts - Zentriert
Schriftart	: Dialog zur Auswahl einer Schriftart erscheint - Auswahlliste bietet die im Stil definierten Stileigenschaften für Schriftarten an

Farbe Schriftart	<i>Farbe der Schrift</i> : Dialog zur Auswahl einer Farbe erscheint - Auswahlliste bietet die im Stil definierten Stileigenschaften für Farben an
Transparenz	Transparenz der Schriftfarbe als Wert von 0 bis 255

Elementeigenschaft 'Farbvariablen'

Farbumschlag	Boolesche Variable, die die Umschaltung der Elementfarbe steuert. Dabei sind Element- und Alarmfarbe als Farben im Visualisierungsstil definiert. TRUE : Visualisierung stellt Element in Alarmfarbe dar.

Elementeigenschaft 'Zustandsvariablen'

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Elementeigenschaft 'Eingabekonfiguration'

Hier kann definiert werden, welche Folgeaktionen ausgeführt werden, wenn der Benutzer im Online-Betrieb eine Eingabe im Element vornimmt.

Eingabeereignisse, die ein Element haben kann, sind:

- OnDialogClosed
- OnMouseClicked
- OnMouseDown
- OnMouseEnter
- OnMouseLeave
- OnMouseMove
- OnMouseUp

Mit Klick auf **Konfigurieren....** wird der Dialog [Kap. "Eingabekonfiguration" auf Seite 849](#) geöffnet, der ermöglicht, dass Folgeaktionen zugewiesen werden. Man klickt in das Feld neben der zu konfigurierenden Eingabe. Für jede selektierte Folgeaktion wird ein Eintrag in der Eigenschaftentabelle erzeugt. Jede Eingabe kann eine beliebige zugeordnete Anzahl an Folgeaktionen haben.

OnDialogClosed	Dieses Ereignis tritt ein, wenn innerhalb der Visualisierung eines der für Benutzereingaben vorher geöffneten Dialogfenster geschlossen wird.
OnMouseClicked	Dieses Mausereignis tritt ein, wenn der Cursor auf ein Element zeigt und ein vollständiger Mausklick (Drücken und Freigeben der Maustaste) ausgeführt wird.
OnMouseDown	Dieses Mausereignis tritt ein, wenn die Maustaste gedrückt wird, während der Cursor auf ein Element zeigt. (Drücken der Maustaste)
OnMouseEnter	Dieses Mausereignis tritt ein, wenn der Cursor auf das Element gezogen wird.
OnMouseLeave	Dieses Mausereignis tritt ein, wenn der Cursor das Element verlässt.

OnMouseMove	Dieses Mausereignis tritt ein, wenn der Cursor innerhalb des Elements bewegt wird.
OnMouseUp	Dieses Mausereignis tritt ein, wenn die Maustaste losgelassen wird. Das vorher ausgeführte Drücken der Maustaste geschieht dabei allerdings außerhalb des Elements.

Eingabekonfiguration: Hotkey

Mit **Hotkey** kann eine Taste, auch mit Zusatzstufe, definiert und mit einer bestimmten Nachfolgeaktion (MouseDown, MouseUp) verknüpft werden, die bei einem Ereignis 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 Tastatenkonfiguration für ein Element wird auch unter Registerkarte 'Tastaturkonfiguration' der Visualisierung verwaltet ([Kap. 20.3.4 "Registerkarte 'Tastaturkonfiguration'" auf Seite 839](#)). Änderungen werden immer zwischen diesem und dem Elementeigenschaften-Editor synchronisiert.

Eigenschaft	Beschreibung Wert
Taste	Zuweisen einer Taste. Eine Auswahlliste stellt alle aktuell unterstützten Tasten bereit wie zum Beispiel <M>.
Eingabe(n)	Definieren des Ereignisses, das ausgeführt werden soll, wenn die Taste beziehungsweise Taste und Modifizierer verwendet werden. <i>Mögliche Werte, die in einer Auswahlliste bereitstehen:</i> - Keine Aktion - MouseDown-Aktion: OnMouseDown-Aktion, wenn die Taste gedrückt wird. - MouseUp-Aktion: OnMouseUp-Aktion, wenn die Taste freigelassen wird. - MouseDown/MouseUp-Aktion: OnMouseUp- und OnMouseDown-Aktion, wenn die Taste gedrückt oder losgelassen wird.
Umschalten	Wenn diese Option aktiviert wird, muss die Taste mit <Umschalt> kombiniert verwendet werden
Steuerung	Wenn diese Option aktiviert wird, muss die Taste mit <Strg> kombiniert verwendet werden
Alt	Wenn diese Option aktiviert wird, muss die Taste mit < Alt> kombiniert verwendet werden

Element 'Unsichtbare Eingabe'

Das Element wird im Editor mit einer gestrichelten Linie angezeigt, aber im Online-Modus ist diese unsichtbar. Das Verhalten des Elements ist in der Eingabekonfiguration zu spezifizieren.

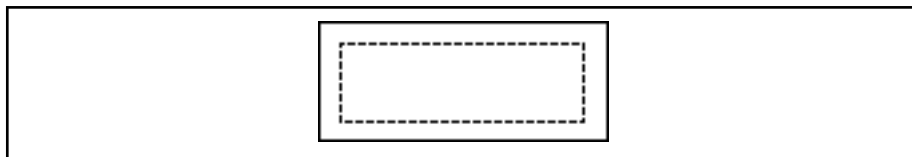


Abb. 20-55: Unsichtbare Eingabe

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungsektor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)

Benutzereingabe beim Editieren der Elementeigenschaften

- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabebehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Unsichtbare Eingabe

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Zustandsvariablen

Eingaben deaktivieren	Zuzuweisen ist eine boolesche Variable. Ist deren Wert TRUE, dann hat keine Benutzereingabe auf das Element eine Auswirkung.

Elementeigenschaft 'Eingabekonfiguration' Hier kann definiert werden, welche Folgeaktionen ausgeführt werden, wenn der Benutzer im Online-Betrieb eine Eingabe im Element vornimmt.

Eingabeereignisse, die ein Element haben kann, sind:

- OnDialogClosed
- OnMouseClicked
- OnMouseDown
- OnMouseEnter
- OnMouseLeave
- OnMouseMove
- OnMouseUp

Mit Klick auf **Konfigurieren....** wird der Dialog [Kap. "Eingabekonfiguration" auf Seite 849](#) geöffnet, der ermöglicht, dass Folgeaktionen zugewiesen werden. Man klickt in das Feld neben der zu konfigurierenden Eingabe. Für jede selektierte Folgeaktion wird ein Eintrag in der Eigenschaftentabelle erzeugt. Jede Eingabe kann eine beliebige zugeordnete Anzahl an Folgeaktionen haben.

OnDialogClosed	Dieses Ereignis tritt ein, wenn innerhalb der Visualisierung eines der für Benutzereingaben vorher geöffneten Dialogfenster geschlossen wird.
OnMouseClicked	Dieses Mausereignis tritt ein, wenn der Cursor auf ein Element zeigt und ein vollständiger Mausklick (Drücken und Freigeben der Maustaste) ausgeführt wird.
OnMouseDown	Dieses Mausereignis tritt ein, wenn die Maustaste gedrückt wird, während der Cursor auf ein Element zeigt. (Drücken der Maustaste)
OnMouseEnter	Dieses Mausereignis tritt ein, wenn der Cursor auf das Element gezogen wird.
OnMouseLeave	Dieses Mausereignis tritt ein, wenn der Cursor das Element verlässt.
OnMouseMove	Dieses Mausereignis tritt ein, wenn der Cursor innerhalb des Elements bewegt wird.
OnMouseUp	Dieses Mausereignis tritt ein, wenn die Maustaste losgelassen wird. Das vorher ausgeführte Drücken der Maustaste geschieht dabei allerdings außerhalb des Elements.

Eingabekonfiguration: Tasten Mit **Tasten** kann konfiguriert werden, dass beim Eintreten des Ereignisses "Tasten" der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier kann eine boolesche Variable zugewiesen werden, deren Wert TRUE wird, wenn die Maustaste gedrückt wird, während der Cursor auf das Element zeigt. Der Wert wird wieder FALSE, wenn die Maustaste wieder losgelassen wird.
FALSE tasten	Wenn diese Option aktiviert ist, wird das oben beschriebene Verhalten für die eingetragene Variable umgekehrt. Das heißt, wenn die Maustaste gedrückt wird, wird der Variablenwert auf FALSE gesetzt, wenn die Taste wieder losgelassen wird, wird der Variablenwert auf TRUE gesetzt.
Beim Betreten tasten, falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert: Wird bei gedrückter Maustaste der Elementbereich verlassen, wird die Variable FALSE und wird wieder TRUE, wenn in den Elementbereich zurückgekehrt wird. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" (also an das Element gebunden) ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Umschalten

Mit **Umschalten** kann konfiguriert werden, dass bei diesem Ereignis der Wert einer booleschen Projektvariable abhängig vom Mausverhalten gesetzt wird.

Eigenschaft	Beschreibung Wert
Variable	Hier ist eine boolesche Variable zuzuweisen. Die Eingabehilfe kann über Schaltfläche  geöffnet werden. Der Wert der Variablen wird bei jedem Mausklick auf das Element zwischen TRUE und FALSE umgeschaltet. Wird der Cursor bei gedrückter Maustaste aus dem Element herausbewegt, dann erfolgt kein Toggeln. Damit kann die begonnene Umschalteneingabe abgebrochen werden.
Umschalten beim Loslassen falls Maus gefangen	Ist diese Option aktiviert, verhält sich der Variablenwert wie unter Variable beschrieben, solange sich die Maus innerhalb des Elementbereichs befindet. Zusätzlich wird auf ein Ziehen der Maus reagiert. Wird bei gedrückter Maustaste der Elementbereich verlassen und dann die Maustaste losgelassen, wird die Variable umgeschaltet. Es wird also berücksichtigt, dass, solange die Maustaste gedrückt ist, die Maus "gefangen" ist, auch wenn der Elementbereich verlassen wird.

Eingabekonfiguration: Hotkey

Mit **Hotkey** kann eine Taste, auch mit Zusatz Taste, definiert und mit einer bestimmten Nachfolgeaktion (MouseDown, MouseUp) verknüpft werden, die bei einem Ereignis 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 Tastatenkonfiguration für ein Element wird auch unter Registerkarte 'Tastatenkonfiguration' der Visualisierung verwaltet ([Kap. 20.3.4 "Registerkarte 'Tastaturkonfiguration'" auf Seite 839](#)). Änderungen werden immer zwischen diesem und dem Elementeigenschaften-Editor synchronisiert.

Eigenschaft	Beschreibung Wert
Taste	Zuweisen einer Taste. Eine Auswahlliste stellt alle aktuell unterstützten Tasten bereit wie zum Beispiel <M>.
Eingabe(n)	Definieren des Ereignisses, das ausgeführt werden soll, wenn die Taste beziehungsweise Taste und Modifizierer verwendet werden. <i>Mögliche Werte, die in einer Auswahlliste bereitstehen:</i> - Keine Aktion - MouseDown-Aktion: OnMouseDown-Aktion, wenn die Taste gedrückt wird. - MouseUp-Aktion: OnMouseUp-Aktion, wenn die Taste freigelassen wird. - MouseDown/MouseUp-Aktion: OnMouseUp- und OnMouseDown-Aktion, wenn die Taste gedrückt oder losgelassen wird.
Umschalten	Wenn diese Option aktiviert wird, muss die Taste mit <Umschalt> kombiniert verwendet werden
Steuerung	Wenn diese Option aktiviert wird, muss die Taste mit <Strg> kombiniert verwendet werden
Alt	Wenn diese Option aktiviert wird, muss die Taste mit < Alt> kombiniert verwendet werden

Element 'Fortschrittsbalken'

In einem Fortschrittsbalken wird Wert einer Variablen als Balken dargestellt. Die Grenzen der Variable sowie der Anzeigestil können eingestellt werden.

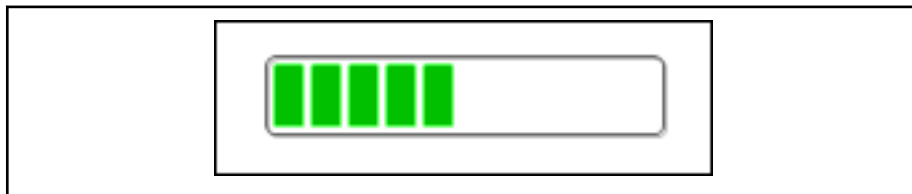


Abb. 20-56: Fortschrittsbalken

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Fortschrittsbalken

Variable	Zuzuweisen ist eine Variable, deren Wert als Fortschrittsbalken visualisiert wird.
Minimaler Wert	Hier ist der kleinste mögliche Wert anzugeben.
Maximaler Wert	Hier ist der größte mögliche Wert anzugeben.
Stil	<i>Auswahl:</i> • Blöcke • Balken

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Elementeigenschaft 'Texte'

Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Im Online-Modus wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt.

[Kap. "Text und Sprache verwenden" auf Seite 845](#)

Text	Geben Sie einen Text ein. Er wird verwendet um das Element zu beschriften. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Textvariable zugewiesen.

Zustandsvariablen

Property	Beschreibung Wert
Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Modus unsichtbar.

Element 'Radiobutton'

Ein Radiobutton erlaubt es eine beliebige Anzahl an Optionen durch Verwendung der Schaltfläche  **Neu erstellen** zu konfigurieren, um sie in einer oder mehreren Spalten anzuordnen.

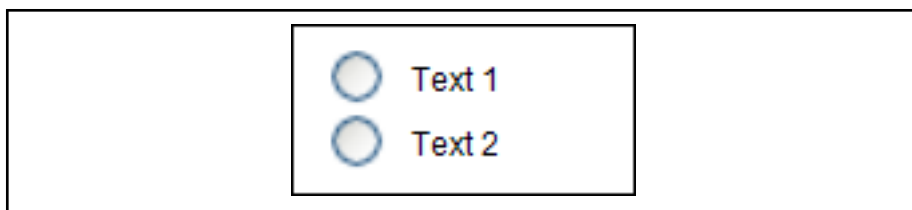


Abb. 20-57: Radiobutton

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Radiobutton

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Variable	Zuzuweisen ist eine Integer-Variable, die mit dem Index des selektierten Auswahlknopfes beschrieben wird.
Anzahl der Spalten	Hier ist die Spaltenanzahl, mit der die Radiobuttons angeordnet werden, anzugeben.
Anordnung der Radiobuttons	Von links nach rechts: Die Radiobuttons werden Zeile für Zeile angeordnet, bis die Anzahl an Spalten erreicht ist. Von oben nach unten: Die Radiobuttons werden spaltenweise angeordnet, bis die Anzahl an Spalten erreicht ist.

**Elementeigenschaft
'Zustandsvariablen'**

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

**Elementeigenschaft
'Radiobbottoneinstellungen'**

Radiobutton Bereiche [<n>]	Neu erstellen: Mit Klick auf diese Schaltfläche wird ein neuer Auswahlknopf im Editor erzeugt und ein weiterer Bereich wird im Eigenschafteneditor aufgelistet. Für jeden Radiobutton ist ein Bereich sichtbar, der die Einstellungen erfasst. [<n>]: Die Nummer indiziert den Bereich. Mit Klick auf Löschen wird der zugehörige Radiobutton mit seinen Einstellungen Text, Tooltip und Zeilenabstand in Pixel gelöscht.

Elementeigenschaft

'Radiobuttoneinstellungen:

Text Radiobutton: Bereiche: [<n>]	Hier ist der gewünschte Button-Name anzugeben. Standardwert ist Radiobutton
Tooltip	Hier ist ein Text, der als Tooltip auszugeben ist, einzugeben.
Zeilenabstand in Pixel	Hier kann der Abstand in Pixel zum oberen Button eingegeben werden.

Element 'Checkbox'

Mit einer Checkbox kann eine boolesche Variable gesetzt und rückgesetzt werden. Ist der Haken gesetzt, wird die Variable auf TRUE gesetzt.



Abb. 20-58: Checkbox

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabebehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Checkbox

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Elementeigenschaft 'Texte'

Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt ([Kap. "Text und Sprache verwenden" auf Seite 845](#)).

Text	Geben Sie einen Text ein. Er wird verwendet um das Element zu beschriften. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ▶ Textvariable zugewiesen.
Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ▶ Tooltip-Variable zugewiesen.

Elementeigenschaft 'Zustandsvariablen'

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

20.4.4 'Messgeräte'

Element 'Balkenanzeige', 'Balkenanzeige frei konfigurierbar'

Hiermit kann eine Balkenanzeige der Visualisierung hinzugefügt werden. **Balkenanzeige** hat ein vordefiniertes Design, bei dem die Hintergrundfarbe wählbar ist. Mit **Balkenanzeige frei konfigurierbar** kann ein Bild aus einer Bilder-sammlung dem Element als Hintergrund zugewiesen werden. Das ist der einzige Unterschied.

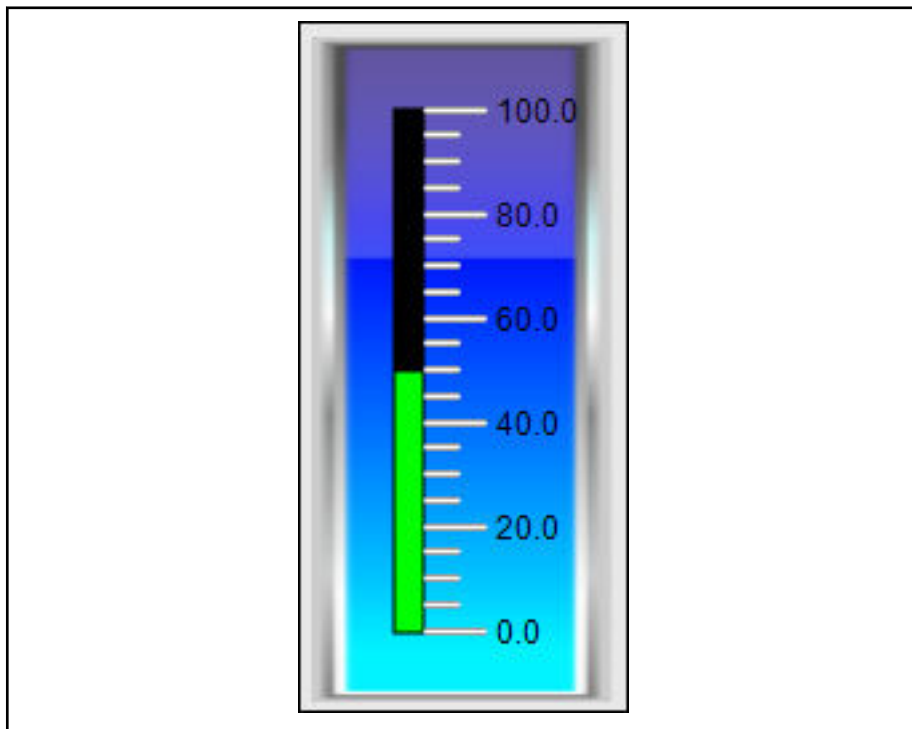


Abb. 20-59:

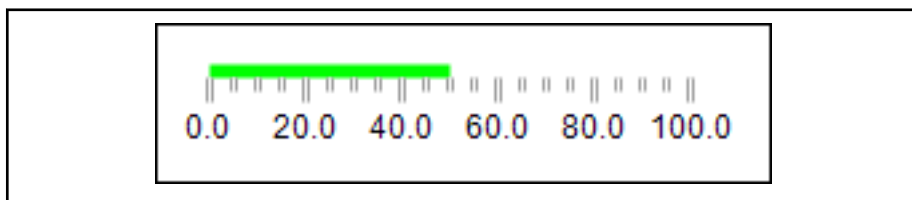


Abb. 20-60:

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Element spezifische Mauseingabe

- Standard ist, dass der Balken horizontal angezeigt wird. Wurde ein Element eingefügt, so kann dessen Form mit Hilfe der Maus geändert werden. Sobald dessen Höhe die Breite überschreitet, erscheint er vertikal und **Balken ► Ausrichtung** ist auf **Vertikal** gesetzt.

Benutzereingabe beim Editieren der Elementeigenschaften*Eigenschaften definieren die Eigenart eines Visualisierungselements:*

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Type Elementtyp	Balkenanzeige oder Balkenanzeige frei konfigurierbar

Wert	Zuzuweisen ist eine Variable, deren Wert als Balkenlänge dargestellt wird.

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Elementeigenschaft 'Hintergrund' Wird keine frei konfigurierbare **Balkenanzeige** verwendet, dann hat **Hintergrund** folgende Auswirkung:

Hintergrundbild	<i>Auszuwählen ist eine Farbe für den Balken:</i> • Gelb • Rot • Grün • Blau • Grau

Wird **Balkenanzeige frei konfigurierbar** verwendet, dann hat **Hintergrund** folgende Auswirkung:

Hintergrundbild	Zuzuweisen ist ein Bild aus dem GlobalImagePool, indem der Name der Bilddatei oder dessen ID angegeben wird.

Elementeigenschaft 'Balken'

Diagrammart	<i>Die Drop-down-Liste stellt verschiedenen Möglichkeiten zur Verfügung, wie Balken und Skale plziert werden können:</i> • Skala neben Balken • Skala im Balken • Balken in Skala • Keine Skala.
Ausrichtung	Der Balken kann Horizontal oder vertikal ausgerichtet sein. Die Ausrichtung entsteht aus dem Verhältnis von Breite zu Höhe und kann hier nicht editiert werden. Diese Änderung erfolgt im Visualisierungseditor, indem ein Eckpunkt es Elements mit der Maus "gegriffen" und horizontal oder vertikal gezogen wird (Kap. " Element 'Balkenanzeige' ", " Balkenanzeige frei konfigurierbar " auf Seite 992).
Laufrichtung	<i>Ist die Ausrichtung horizontal, kann gewählt werden zwischen:</i> • Links nach rechts • Rechts nach Links <i>Ist die Ausrichtung vertikal, kann gewählt werden zwischen:</i> • Von unten nach oben • Von oben nach unten

Elementeigenschaft 'Skala'

Skalenanfang	Wert, der die Skalierungsanzeige von unten begrenzt
Skalenende	Wert, der die Skalierungsanzeige von oben begrenzt
Hauptskala	Abstand zwischen zwei Strichen der Grobskala

Unterskala	Abstand zwischen zwei Strichen der Feinskala. Kann auf 0 gesetzt werden, falls eine weitere Unterteilung der Grobskala nicht gewünscht ist.
Skalenlinienstärke	Der Wert definiert die Dicke der Skalenlinie in Pixel.
Skalenfarbe	Die Farbe kann über Selektionsliste bzw. über Schaltfläche  aus dem Standarddialog zur Farbauswahl eingestellt werden.
Skala in 3D	Ist diese Option aktiviert, wird die Skala 3-dimensionale dargestellt.
Elementrahmen	Ist diese Option aktiviert, wird ein Rahmen um die Balkenanzeige gezogen.

Beschriftung

Eigenschaft	Beschreibung Wert
Einheit	Der eingegebene Text beschriftet das Element. String, der unterhalb des Pfeils (M) bzw. unterhalb des Mittelpunkts der Skalierung (BD) angezeigt wird (gedacht für die Eingabe der Skalierungseinheit).
Schriftart	Festzulegen ist die Schriftart für Einheit und Skalierung: <i>Auszuwählen ist:</i> • Standard • Titelzeile • Large • Title • Annotation Mit Klick auf  öffnet ein Dialog für benutzerdefinierte Einstellung der Schrifteigenschaften.
Skalenformat (C-Syntax)	Benutzen Sie die C-Syntax zur Angabe der Formatierung der Skalierungsbeschriftung. So 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"
Max. Breite der Beschriftungen	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.
Höhe der Beschriftungen	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.
Farbe Schriftart	Farbe für die Schrift kann über Selektionsliste bzw. über Schaltfläche  aus dem Standarddialog zur Farbauswahl eingestellt werden.

**Elementeigenschaft
'Positionierung'**

Horizontale Verschiebung	Hier ist der Abstand in Pixel der Skala/des Balkens zum horizontalen Elementrahmen anzugeben.
Vertikale Verschiebung	Hier ist der Abstand in Pixel der Skala/des Balkens zum vertikalen Elementrahmen anzugeben.

Elementeigenschaft 'Farben'

Farbe des Graphen	Anzugeben ist eine Farbe für den Balken.
Balkenhintergrund	Standard ist, dass die Hintergrundfarbe der Balkenanzeige weiß ist. Wird die Option aktiviert, wird die Hintergrundfarbe in schwarz geändert.
Rahmenfarbe	Farbe des Rahmens, der die Balkenanzeige umgibt, sofern das Kontrollkästchen Elementrahmen aktiviert ist.
Farbbereiche verwenden	Bei Aktivierung dieses Kontrollkästchens werden die Farbbereiche, so wie sie unter Farbbereiche definiert, angezeigt.
Gesamte Farbe umschalten	Ist diese Option aktiviert, ändert der gesamte Balken seine Farbe, wenn der Variablenwert den Anfangs-/Endwert der Skala unter- oder überschreitet.
Farbverlauf für Balken verwenden	Bei Aktivierung dieser Option wird der Balken mit Farbverlauf angezeigt.
Farbbereichsmarkierung	<i>Auswahl zwischen</i> • Keine Markierung • Markierung vorwärts • Markierung rückwärts
Farbbereiche • [<n>]	Neu erstellen: Mit Klick auf diesen Button wird ein neuer Farbbereich erzeugt und in Eigenschaften aufgelistet.

Elementeigenschaft 'Farben: Farbbereiche: Bereiche [**<n>**]'

Anzahl der Farbbereiche in aufsteigender, nicht editierbarer Reihenfolge.

Nach Anklicken von  wird der zugehörige Farbbereich gelöscht und die Nummerierung neu vergeben.

Bereichsanfang	Wert, der den Farbbereich von unten begrenzt
Bereichsende	Wert, der den Farbbereich von oben begrenzt
Farbe	Farbe, in der der Farbbereich dargestellt werden soll

Element 'Zeigerinstrument 90°'

Mit **Zeigerinstrument** kann zum Beispiel ein Drehzahlmesser der Visualisierung hinzugefügt werden. Die Nadel positioniert sich nach dem Wert der zugewiesenen Variablen. Es hat ein vorkonfiguriertes Design, in dem eine Hintergrundfarbe gesetzt werden kann.

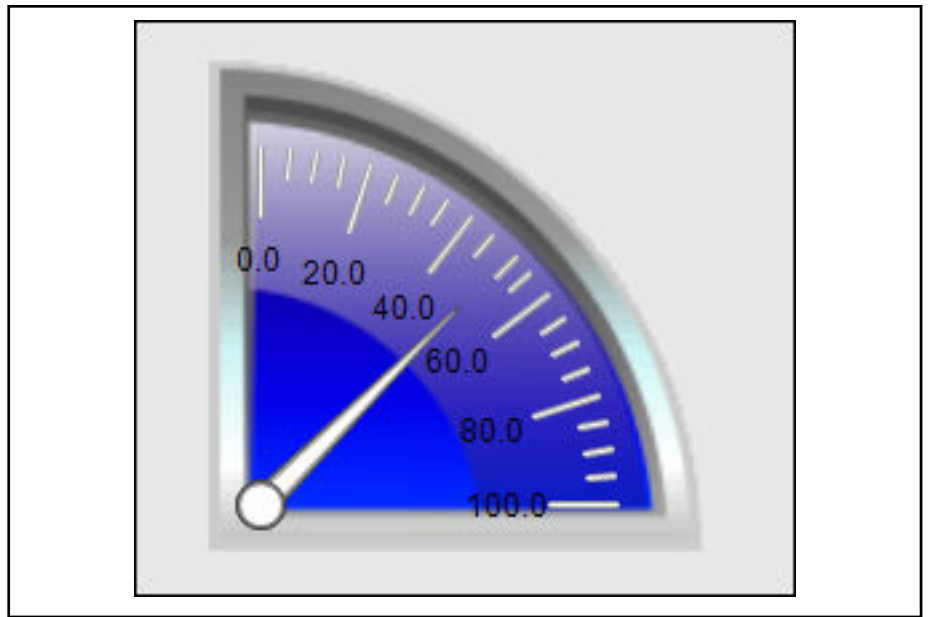


Abb. 20-61:

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Zeigerinstrument 90°

Wert	Zuzuweisen ist eine Variable, deren Wert als Ausschlag der Instrumentennadel dargestellt wird.

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Elementeigenschaft 'Hintergrund'

Hintergrundbild	<i>Auszuwählen ist eine Farbe für den Balken:</i> • Gelb • Rot • Grün • Blau • Grau

Elementeigenschaft 'Zeiger'

Zeigertyp	<i>Auszuwählen ist aus:</i> • Normaler Zeiger • Dünner Zeiger • Breiter Zeiger • Dünne Nadel • Dünner 3D-Zeiger • Dünner 3D-Nadel
Farbe	Farbe, mit der der Zeiger angezeigt wird.

Winkelbereich	<i>Auswahl eines 90°-Ausschnitts:</i> • Oben rechts • Oben links • Unten links • Unten rechts
Zusätzlicher Zeiger	Ist die Option aktiviert, ist ein zusätzlicher Zeiger auf der Skala gegenüber der Nadel sichtbar.

Skala

Eigenschaft	Beschreibung Wert
Unterskalaposition	<i>Die Unterskala kann am äußeren oder inneren Radius des Skalenrings angezeigt werden:</i> • Außen • Innen
Skalentyp	<i>Die Skala kann angezeigt werden als:</i> • Linien • Punkte • Quadrate
Skalenanfang	Wert, der die Skalierungsanzeige von unten begrenzt.
Skalenende	Wert, der die Skalierungsanzeige von oben begrenzt.
Hauptskala	Abstand zwischen zwei Strichen der Grobskala.
Unterskala	Abstand zwischen zwei Strichen der Feinskala; kann auf 0 gesetzt werden, falls eine weitere Unterteilung der Grobskala nicht gewünscht ist.
Skalenlinienstärke	Stärke der Skala in Pixel
Skalenfarbe	Die Farbe kann über Selektionsliste bzw. über Schaltfläche  aus dem Standarddialog zur Farbauswahl eingestellt werden.
Skala in 3D	Ist die Option aktiviert, wird die Skala dreidimensional dargestellt.
Skala anzeigen	Wird keine Skalendarstellung gewünscht, ist diese Option zu deaktivieren.
Innerer Rahmen	Ist diese Option aktiviert, wird der Skalenring innen mit einem Rahmen versehen. Standardmäßig deaktiviert.
Äußerer Rahmen	Ist diese Option aktiviert, wird der Skalenring außen mit einem Rahmen versehen. Standardmäßig deaktiviert.

Beschriftung

Eigenschaft	Beschreibung Wert
Beschriftung	<i>Auswählen aus</i> • Außen: Skalenwerte werden auf der Außenseite der Skala platziert. • Innen: Skalenwerte werden auf der Innenseite der Skala platziert.
Einheit	String, der unterhalb des Pfeils (M) bzw. unterhalb des Mittelpunkts der Skalierung (BD) angezeigt wird (gedacht für die Eingabe der Skalierungseinheit)

Eigenschaft	Beschreibung Wert
Schriftart	Schriftart für Darstellung von Einheit und Skalenbereich <i>Festzulegen ist die Schriftart durch auswählen von einem aus:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet ein Dialog für eine benutzerdefinierte Einstellung der Schriftart.
Skalenformat (C-Syntax)	Hier ist die C-Syntax zur Angabe der Formatierung der Skalierungsbeschriftung zu benutzen. 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"
Max. Breite der Beschriftungen	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.
Höhe der Beschriftungen	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.
Farbe Schriftart	Farbe für die Schrift kann über Selektionsliste bzw. über Schaltfläche  aus dem Standarddialog zur Farbauswahl eingestellt werden.

Elementeigenschaft 'Positionierung'

Zeigerabstand	Hier kann die Länge des Zeigers in Pixel angegeben werden.
Beschriftungsverschiebung	Hier kann ein Abstand in Pixel definiert werden um die Beschriftung zu positionieren.
Einheitenverschiebung	Hier kann ein vertikaler Abstand eingegeben werden, um den Text eingegeben in Beschriftung ► Einheit zu positionieren.

Farben

Eigenschaft	Wertebeschreibung
Farbbereiche • Dauerhafte Farbbereiche • [<n>]	Dauerhafte Farbbereiche: Dessen Auswirkung ist nur im Online-Mode sichtbar. Ist die Option aktiviert, sind die Farbbereiche dauerhaft sichtbar. Ist die Option nicht aktiviert, ist nur der Farbbereich, auf den der Pfeil zeigt, sichtbar. Neu erstellen: Mit Klick auf diesen Button wird ein weiterer Farbbereich erzeugt und angezeigt.

Farben: Farbbereiche: Bereiche: [<n>] Anzahl der Farbbereiche in aufsteigender, nicht editabler Reihenfolge.

Löschen: Ein Klick auf den Button  wird den Farbbereich von der Liste entfernen und die Reihenfolge neu arrangieren.

Eigenschaft	Beschreibung Wert
Bereichsanfang	Wert, der den Farbbereich von unten begrenzt.
Bereichsende	Wert, der den Farbbereich von oben begrenzt.
Farbe	Farbe, in der der Farbbereich dargestellt werden soll.

Element 'Zeigerinstrument 180°'

Mit **Zeigerinstrument** kann zum Beispiel ein Drehzahlmesser der Visualisierung hinzugefügt werden. Die Nadel positioniert sich nach dem Wert der zugewiesenen Variablen. Es hat ein vorkonfiguriertes Design, in dem eine Hintergrundfarbe gesetzt werden kann.

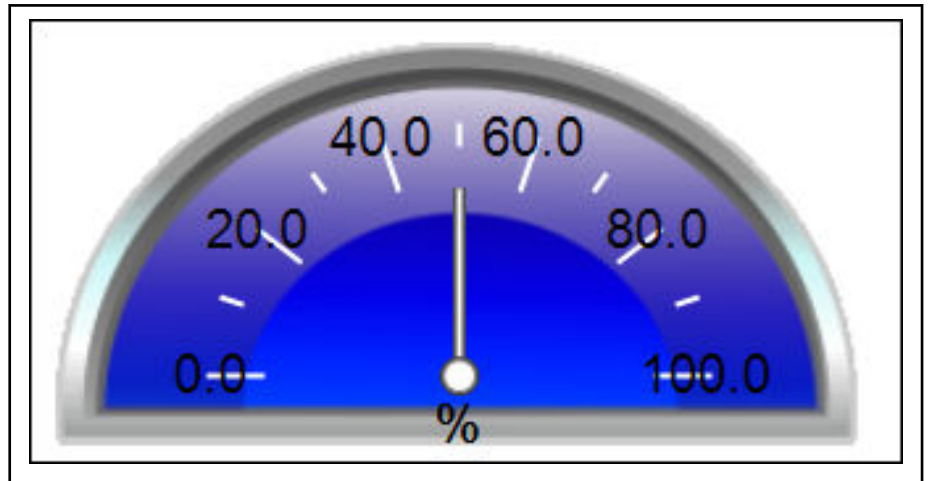


Abb. 20-62:

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Zeigerinstrument 180°

Wert	Zuzuweisen ist eine Variable, deren Wert als Ausschlag der Instrumentennadel dargestellt wird.

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Elementeigenschaft 'Hintergrund'

Hintergrundbild	<i>Auszuwählen ist eine Farbe für den Balken:</i> • Gelb • Rot • Grün • Blau • Grau

Elementeigenschaft 'Zeiger'

Zeigertyp	<i>Auszuwählen ist aus:</i> • Normaler Zeiger • Dünner Zeiger • Breiter Zeiger • Dünne Nadel • Dünner 3D-Zeiger • Dünner 3D-Nadel
Farbe	Farbe, mit der der Zeiger angezeigt wird.

Winkelbereich	<i>Auswahl eines 180°-Ausschnitts:</i> • Oben • Unten • Links • Rechts
Zusätzlicher Zeiger	Ist die Option aktiviert, ist ein zusätzlicher Zeiger auf der Skala gegenüber der Nadel sichtbar.

Skala

Eigenschaft	Beschreibung Wert
Unterskalaposition	<i>Die Unterskala kann am äußeren oder inneren Radius des Skalenrings angezeigt werden:</i> • Außen • Innen
Skalentyp	<i>Die Skala kann angezeigt werden als:</i> • Linien • Punkte • Quadrate
Skalenanfang	Wert, der die Skalierungsanzeige von unten begrenzt.
Skalenende	Wert, der die Skalierungsanzeige von oben begrenzt.
Hauptskala	Abstand zwischen zwei Strichen der Grobskala.
Unterskala	Abstand zwischen zwei Strichen der Feinskala; kann auf 0 gesetzt werden, falls eine weitere Unterteilung der Grobskala nicht gewünscht ist.
Skalenlinienstärke	Stärke der Skala in Pixel
Skalenfarbe	Die Farbe kann über Selektionsliste bzw. über Schaltfläche  aus dem Standarddialog zur Farbauswahl eingestellt werden.
Skala in 3D	Ist die Option aktiviert, wird die Skala dreidimensional dargestellt.
Skala anzeigen	Wird keine Skalendarstellung gewünscht, ist diese Option zu deaktivieren.
Innerer Rahmen	Ist diese Option aktiviert, wird der Skalenring innen mit einem Rahmen versehen. Standardmäßig deaktiviert.
Äußerer Rahmen	Ist diese Option aktiviert, wird der Skalenring außen mit einem Rahmen versehen. Standardmäßig deaktiviert.

Beschriftung

Eigenschaft	Beschreibung Wert
Beschriftung	<i>Auswählen aus</i> • Außen: Skalenwerte werden auf der Außenseite der Skala platziert. • Innen: Skalenwerte werden auf der Innenseite der Skala platziert.
Einheit	String, der unterhalb des Pfeils (M) bzw. unterhalb des Mittelpunkts der Skalierung (BD) angezeigt wird (gedacht für die Eingabe der Skalierungseinheit)

Eigenschaft	Beschreibung Wert
Schriftart	Schriftart für Darstellung von Einheit und Skalenbereich <i>Festzulegen ist die Schriftart durch auswählen von einem aus:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet ein Dialog für eine benutzerdefinierte Einstellung der Schriftart.
Skalenformat (C-Syntax)	Hier ist die C-Syntax zur Angabe der Formatierung der Skalierungsbeschriftung zu benutzen. 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"
Max. Breite der Beschriftungen	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.
Höhe der Beschriftungen	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.
Farbe Schriftart	Farbe für die Schrift kann über Selektionsliste bzw. über Schaltfläche  aus dem Standarddialog zur Farbauswahl eingestellt werden.

Elementeigenschaft 'Positionierung'

Zeigerabstand	Hier kann die Länge des Zeigers in Pixel angegeben werden.
Beschriftungsverschiebung	Hier kann ein Abstand in Pixel definiert werden um die Beschriftung zu positionieren.
Einheitenverschiebung	Hier kann ein vertikaler Abstand eingegeben werden, um den Text eingegeben in Beschriftung ► Einheit zu positionieren.

Farben

Eigenschaft	Wertebeschreibung
Farbbereiche • Dauerhafte Farbbereiche • [<n>]	Dauerhafte Farbbereiche: Dessen Auswirkung ist nur im Online-Mode sichtbar. Ist die Option aktiviert, sind die Farbbereiche dauerhaft sichtbar. Ist die Option nicht aktiviert, ist nur der Farbbereich, auf den der Pfeil zeigt, sichtbar. Neu erstellen: Mit Klick auf diesen Button wird ein weiterer Farbbereich erzeugt und angezeigt.

Farben: Farbbereiche: Bereiche: [<n>] Anzahl der Farbbereiche in aufsteigender, nicht editabler Reihenfolge.

Löschen: Ein Klick auf den Button  wird den Farbbereich von der Liste entfernen und die Reihenfolge neu arrangieren.

Eigenschaft	Beschreibung Wert
Bereichsanfang	Wert, der den Farbbereich von unten begrenzt.
Bereichsende	Wert, der den Farbbereich von oben begrenzt.
Farbe	Farbe, in der der Farbbereich dargestellt werden soll.

Element 'Zeigerinstrument 360°'

Mit **Zeigerinstrument** kann zum Beispiel ein Drehzahlmesser der Visualisierung hinzugefügt werden. Die Nadel positioniert sich nach dem Wert der zugewiesenen Variablen. Es hat ein vorkonfiguriertes Design, in dem eine Hintergrundfarbe gesetzt werden kann.

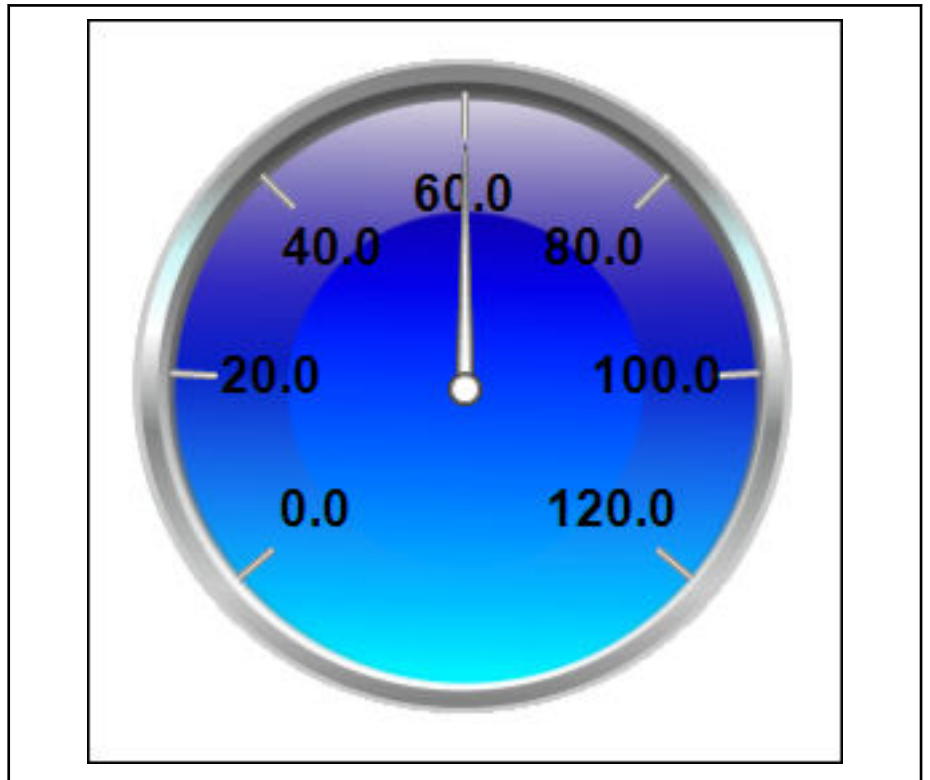


Abb. 20-63: Zeigerinstrument 360°

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,

- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Zeigerinstrument 360°

Wert	Zuzuweisen ist eine Variable, deren Wert als Ausschlag der Instrumentennadel dargestellt wird.

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Elementeigenschaft 'Hintergrund'

Hintergrundbild	<i>Auszuwählen ist eine Farbe für den Balken:</i> • Gelb • Rot • Grün • Blau • Grau

Zeiger

Eigenschaft	Beschreibung Wert
Zeigertyp	<i>Auszuwählen ist aus:</i> • Normaler Zeiger • Dünner Zeiger • Breiter Zeiger • Dünne Nadel • Dünner 3D-Zeiger • Dünner 3D-Nadel
Farbe	Farbe, mit der der Zeiger angezeigt wird.
Zeiger Anfang	Der hier anzugebenen Wert wird als Winkel in Grad interpretiert, der sich vom Nullpunkt der Skala bis zum Anfang des Skalenbereichs im Gegenuhrzeigersinn aufspannt. Nullpunkt der Skala liegt bei "3 Uhr". Der Winkel hat einen Wertebereich von 0° bis 360°. Zentrum der Drehung ist die X/Y-Position.
Zeiger Ende	Der hier anzugebenen Wert wird als Winkel in Grad interpretiert, der sich vom Nullpunkt der Skala bis zum Ende des Skalenbereichs im Gegenuhrzeigersinn aufspannt. Nullpunkt der Skala liegt bei "3 Uhr". Eingabe eines negativen Werts ist erlaubt. Zentrum der Drehung ist die X/Y-Position. Zeiger Ende muss so gesetzt werden, dass es im Uhrzeigersinn betrachtet rechts von Zeiger Anfang zu liegen kommt. Der Winkel hat einen Wertebereich von 0° bis maximal 350°.
Zusätzlicher Zeiger	Ist die Option aktiviert, ist ein zusätzlicher Zeiger auf der Skala gegenüber der Nadel sichtbar.

Skala

Eigenschaft	Beschreibung Wert
Unterskalaposition	<i>Die Unterskala kann am äußeren oder inneren Radius des Skalenrings angezeigt werden:</i> • Außen • Innen
Skalentyp	<i>Die Skala kann angezeigt werden als:</i> • Linien • Punkte • Quadrate
Skalenanfang	Wert, der die Skalierungsanzeige von unten begrenzt.
Skalenende	Wert, der die Skalierungsanzeige von oben begrenzt.
Hauptskala	Abstand zwischen zwei Strichen der Grobskala.
Unterskala	Abstand zwischen zwei Strichen der Feinskala; kann auf 0 gesetzt werden, falls eine weitere Unterteilung der Grobskala nicht gewünscht ist.
Skalenlinienstärke	Stärke der Skala in Pixel
Skalenfarbe	Die Farbe kann über Selektionsliste bzw. über Schaltfläche  aus dem Standarddialog zur Farbauswahl eingestellt werden.
Skala in 3D	Ist die Option aktiviert, wird die Skala dreidimensional dargestellt.
Skala anzeigen	Wird keine Skalendarstellung gewünscht, ist diese Option zu deaktivieren.

Eigenschaft	Beschreibung Wert
Innerer Rahmen	Ist diese Option aktiviert, wird der Skalenring innen mit einem Rahmen versehen. Standardmäßig deaktiviert.
Äußerer Rahmen	Ist diese Option aktiviert, wird der Skalenring außen mit einem Rahmen versehen. Standardmäßig deaktiviert.

Beschriftung

Eigenschaft	Beschreibung Wert
Beschriftung	<i>Auswählen aus</i> • Außen: Skalenwerte werden auf der Außenseite der Skala platziert. • Innen: Skalenwerte werden auf der Innenseite der Skala platziert.
Einheit	String, der unterhalb des Pfeils (M) bzw. unterhalb des Mittelpunkts der Skalierung (BD) angezeigt wird (gedacht für die Eingabe der Skalierungseinheit)
Schriftart	Schriftart für Darstellung von Einheit und Skalenbereich <i>Festzulegen ist die Schriftart durch auswählen von einem aus:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet ein Dialog für eine benutzerdefinierte Einstellung der Schriftart.
Skalenformat (C-Syntax)	Hier ist die C-Syntax zur Angabe der Formatierung der Skalierungsbeschriftung zu benutzen. 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"
Max. Breite der Beschriftungen	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.
Höhe der Beschriftungen	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.
Farbe Schriftart	Farbe für die Schrift kann über Selektionsliste bzw. über Schaltfläche  aus dem Standarddialog zur Farbauswahl eingestellt werden.

Positionierung

Eigenschaft	Beschreibung Wert
Zeigerabstand	Hier kann die Länge des Zeigers in Pixel angegeben werden.
Beschriftungsverschiebung	Hier kann ein Abstand in Pixel definiert werden um die Beschriftung zu positionieren.
Einheitenverschiebung	Hier kann ein vertikaler Abstand eingegeben werden, um den Text eingegeben in Beschriftung ► Einheit zu positionieren.
Ursprungsverschiebung	Dieser Wert definiert den Versatz des Elements. Er kann verwendet werden um eine exakte Positionierung relativ zum Hintergrundbild zu erreichen.

Farben

Eigenschaft	Wertebeschreibung
Farbbereiche - Dauerhafte Farbbereiche [<n>]	Dauerhafte Farbbereiche: Dessen Auswirkung ist nur im Online-Mode sichtbar. Ist die Option aktiviert, sind die Farbbereiche dauerhaft sichtbar. Ist die Option nicht aktiviert, ist nur der Farbbereich, auf den der Pfeil zeigt, sichtbar. Neu erstellen: Mit Klick auf diesen Button wird ein weiterer Farbbereich erzeugt und angezeigt.

Farben: Farbbereiche: Bereiche:
[<n>]

Anzahl der Farbbereiche in aufsteigender, nicht editierbarer Reihenfolge.

Löschen: Ein Klick auf den Button  wird den Farbbereich von der Liste entfernen und die Reihenfolge neu arrangieren.

Eigenschaft	Beschreibung Wert
Bereichsanfang	Wert, der den Farbbereich von unten begrenzt.
Bereichsende	Wert, der den Farbbereich von oben begrenzt.
Farbe	Farbe, in der der Farbbereich dargestellt werden soll.

Element 'Potentiometer'

Hier kann einer Visualisierung ein Potentiometer mit vordefiniertem Design hinzugefügt werden. Solch ein Bedienelement zeigt mittels Pfeil/Zeiger einen Variablenwert an, der über Benutzereingabe verschoben werden kann.

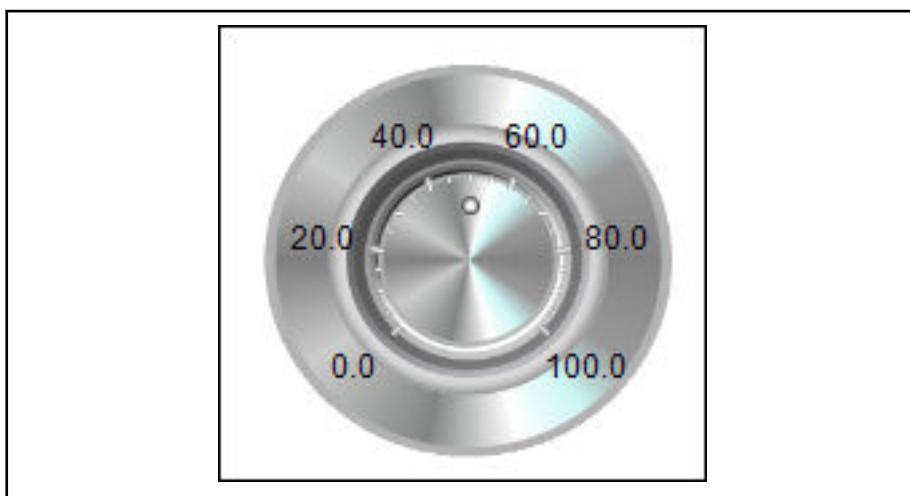


Abb. 20-64: Potentiometer

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.

- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Potentiometer

Variable	Zuzuweisen ist eine numerische Variable, die die Position des Potentiometers enthält.

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Hintergrund

Eigenschaft	Beschreibung Wert
Hintergrundbild	Zuzuweisen ist ein Bild aus dem GlobellImagePool indem entweder der Name der Bild-datei oder die Bild-ID angegeben wird.
Transparenzfarbe	Für Bilder mit transparentem Hintergrund kann eine Farbe angegeben werden.

Zeiger

Eigenschaft	Beschreibung Wert
Zeigertyp	<i>Auszuwählen ist aus:</i> • Normaler Zeiger • Dünner Zeiger • Breiter Zeiger • Dünne Nadel • Dünner 3D-Zeiger • Dünner 3D-Nadel
Farbe	Farbe, mit der der Zeiger angezeigt wird.
Zeiger Anfang	Der hier anzugebenen Wert wird als Winkel in Grad interpretiert, der sich vom Nullpunkt der Skala bis zum Anfang des Skalenbereichs im Gegenuhrzeigersinn aufspannt. Nullpunkt der Skala liegt bei "3 Uhr". Der Winkel hat einen Wertebereich von 0° bis 360°. Zentrum der Drehung ist die X/Y-Position.
Zeiger Ende	Der hier anzugebenen Wert wird als Winkel in Grad interpretiert, der sich vom Nullpunkt der Skala bis zum Ende des Skalenbereichs im Gegenuhrzeigersinn aufspannt. Nullpunkt der Skala liegt bei "3 Uhr". Eingabe eines negativen Werts ist erlaubt. Zentrum der Drehung ist die X/Y-Position. Zeiger Ende muss so gesetzt werden, dass es im Uhrzeigersinn betrachtet rechts von Zeiger Anfang zu liegen kommt. Der Winkel hat einen Wertebereich von 0° bis maximal 350°.

Skala

Eigenschaft	Beschreibung Wert
Unterskalaposition	<i>Die Unterskala kann am äußeren oder inneren Radius des Skalenrings angezeigt werden:</i> • Außen • Innen
Skalentyp	<i>Die Skala kann angezeigt werden als:</i> • Linien • Punkte • Quadrate
Skalenanfang	Wert, der die Skalierungsanzeige von unten begrenzt.
Skalenende	Wert, der die Skalierungsanzeige von oben begrenzt.
Hauptskala	Abstand zwischen zwei Strichen der Grobskala.
Unterskala	Abstand zwischen zwei Strichen der Feinskala; kann auf 0 gesetzt werden, falls eine weitere Unterteilung der Grobskala nicht gewünscht ist.
Skalenlinienstärke	Stärke der Skala in Pixel

Eigenschaft	Beschreibung Wert
Skalenfarbe	Die Farbe kann über Selektionsliste bzw. über Schaltfläche  aus dem Standarddialog zur Farbauswahl eingestellt werden.
Skala in 3D	Ist die Option aktiviert, wird die Skala dreidimensional dargestellt.
Skala anzeigen	Wird keine Skalendarstellung gewünscht, ist diese Option zu deaktivieren.
Innerer Rahmen	Ist diese Option aktiviert, wird der Skalenring innen mit einem Rahmen versehen. Standardmäßig deaktiviert.
Äußerer Rahmen	Ist diese Option aktiviert, wird der Skalenring außen mit einem Rahmen versehen. Standardmäßig deaktiviert.

Beschriftung

Eigenschaft	Beschreibung Wert
Beschriftung	<i>Auswählen aus</i> • Außen: Skalenwerte werden auf der Außenseite der Skala platziert. • Innen: Skalenwerte werden auf der Innenseite der Skala platziert.
Einheit	String, der unterhalb des Pfeils (M) bzw. unterhalb des Mittelpunkts der Skalierung (BD) angezeigt wird (gedacht für die Eingabe der Skalierungseinheit)
Schriftart	Schriftart für Darstellung von Einheit und Skalenbereich <i>Festzulegen ist die Schriftart durch auswählen von einem aus:</i> • Standard • Überschrift • Große • Titel • Anmerkung Mit Klick auf  öffnet ein Dialog für eine benutzerdefinierte Einstellung der Schriftart.
Skalenformat (C-Syntax)	Hier ist die C-Syntax zur Angabe der Formatierung der Skalierungsbeschriftung zu benutzen. 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"
Max. Breite der Beschriftungen	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.
Höhe der Beschriftungen	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.
Farbe Schriftart	Farbe für die Schrift kann über Selektionsliste bzw. über Schaltfläche  aus dem Standarddialog zur Farbauswahl eingestellt werden.

Positionierung

Eigenschaft	Beschreibung Wert
Zeigerabstand	Hier kann die Länge des Zeigers in Pixel angegeben werden.
Beschriftungsverschiebung	Hier kann ein Abstand in Pixel definiert werden um die Beschriftung zu positionieren.

Eigenschaft	Beschreibung Wert
Einheitenverschiebung	Hier kann ein vertikaler Abstand eingegeben werden, um den Text eingegeben in Beschriftung ▶ Einheit zu positionieren.
Ursprungsverschiebung	Dieser Wert definiert den Versatz des Elements. Er kann verwendet werden um eine exakte Positionierung relativ zum Hintergrundbild zu erreichen.

Farben

Eigenschaft	Wertebeschreibung
Farbbereiche - Dauerhafte Farbbereiche [<n>]	Dauerhafte Farbbereiche: Dessen Auswirkung ist nur im Online-Mode sichtbar. Ist die Option aktiviert, sind die Farbbereiche dauerhaft sichtbar. Ist die Option nicht aktiviert, ist nur der Farbbereich, auf den der Pfeil zeigt, sichtbar. Neu erstellen: Mit Klick auf diesen Button wird ein weiterer Farbbereich erzeugt und angezeigt.

Farben: Farbbereiche: Bereiche:
[<n>]

Anzahl der Farbbereiche in aufsteigender, nicht editierbarer Reihenfolge.

Löschen: Ein Klick auf den Button  wird den Farbbereich von der Liste entfernen und die Reihenfolge neu arrangieren.

Eigenschaft	Beschreibung Wert
Bereichsanfang	Wert, der den Farbbereich von unten begrenzt.
Bereichsende	Wert, der den Farbbereich von oben begrenzt.
Farbe	Farbe, in der der Farbbereich dargestellt werden soll.

Element 'Histogramm'

Inhalt dieser Seite:

- ["Elementeigenschaften" auf Seite 1014](#)
- ["Konfiguration eines Histogramms" auf Seite 1017](#)

Dieses Element fügt eine Histogrammansicht der Visualisierung zu um Werte eines Arrays darzustellen. Der minimale und maximale Anzeigewert kann angegeben werden. Für bestimmte Wertebereiche können spezielle Farben definiert werden.

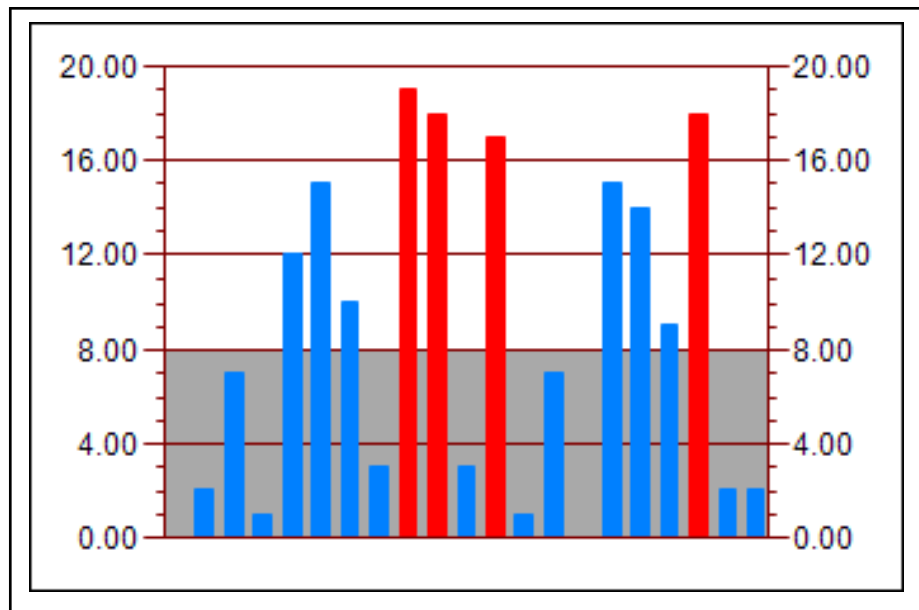


Abb. 20-65: Histogramm

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungsektor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
-  Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Histogramm

Datenarray	Zuzuweisen ist ein Datenarray, der visualisiert werden soll.

Elementeigenschaft
'Unterbereich des Arrays'

Unterbereich verwenden	Ist diese Option aktiviert, wird nur der Unterbereich des zugewiesenen Datenarrays verwendet.
Startindex	Indiziert den ersten Datensatz des zugewiesenen Datenarrays.
Endindex	Indiziert den letzten Datensatz des zugewiesenen Datenarrays.

Darstellungstyp	• Balken • Linien • Kurve
Linienstärke	Voraussetzung: Darstellungstyp ist Kurve . Linienstärke der Kurve im Histogramm. Voreinstellung: 1
Horizontale Linien anzeigen	Ist diese Option aktiviert, werden horizontale Linien an der Hauptskala gezeichnet.
Relative Balkenbreite	Dieser Wert definiert die relative Breite des Balkens bzw. der Linie in Pixel.

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Elementeigenschaft 'Skala'

Skalenanfang	Wert, der die Skalierungsanzeige von unten begrenzt
Skalenende	Wert, der die Skalierungsanzeige von oben begrenzt
Hauptskala	Abstand zwischen zwei Strichen der Grobskala

Unterskale	Abstand zwischen zwei Strichen der Feinskala. Kann auf 0 gesetzt werden, falls eine weitere Unterteilung der Grobskala nicht gewünscht ist.
Skalenfarbe	Die Farbe kann über Selektionsliste bzw. über Schaltfläche  aus dem Standarddialog zur Farbauswahl eingestellt werden.

Beschriftung

Eigenschaft	Beschreibung Wert
Einheit	Der eingegebene Text beschriftet das Element. String, der unterhalb des Pfeils (M) bzw. unterhalb des Mittelpunkts der Skalierung (BD) angezeigt wird (gedacht für die Eingabe der Skalierungseinheit).
Schriftart	Festzulegen ist die Schriftart für Einheit und Skalierung: <i>Auszuwählen ist:</i> • Standard • Titelzeile • Large • Title • Annotation Mit Klick auf  öffnet ein Dialog für benutzerdefinierte Einstellung der Schrifteigenschaften.
Skalenformat (C-Syntax)	Benutzen Sie die C-Syntax zur Angabe der Formatierung der Skalierungsbeschriftung. So 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"
Max. Breite der Beschriftungen	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.
Höhe der Beschriftungen	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.
Farbe Schriftart	Farbe für die Schrift kann über Selektionsliste bzw. über Schaltfläche  aus dem Standarddialog zur Farbauswahl eingestellt werden.

Elementeigenschaft 'Farben'

Farbe des Graphen	Definieren Sie eine Farbe für den Graphen.
Farbbereiche verwenden	Wird diese Option aktiviert, werden die spezifizierten Farbbereiche so wie unter Farbbereiche definiert dargestellt.
Farbbereiche [<n>]	Neu erstellen: Mit Klick auf diesen Button wird ein neuer Bereich erstellt und in Eigenschaften aufgelistet.

Farben: Farbbereiche: Bereiche: Anzahl der Farbbereiche in aufsteigender, nicht editierbarer Reihenfolge.
[<n>]

Löschen: Ein Klick auf den Button  wird den Farbbereich von der Liste entfernen und die Reihenfolge neu arrangieren.

Eigenschaft	Beschreibung Wert
Bereichsanfang	Wert, der den Farbbereich von unten begrenzt.
Bereichsende	Wert, der den Farbbereich von oben begrenzt.
Farbe	Farbe, in der der Farbbereich dargestellt werden soll.

Konfiguration eines Histogramms

Sehen Sie im folgenden ein Beispiel für das Einfügen und die Konfiguration eines Histogramm Elements:

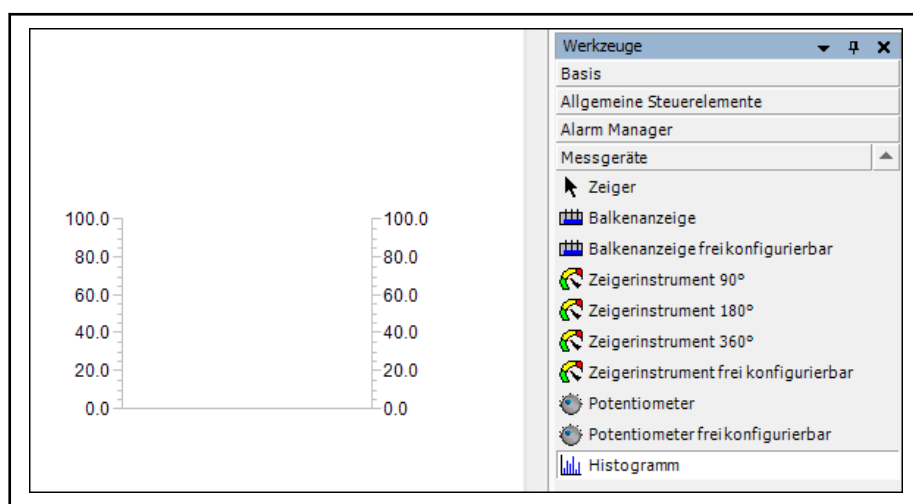


Abb. 20-66: Beispiel - Einfügen des Elements Histogramm

Das zugeordnete Datenarray - im Beispiel ein Array mit dem Namen `Histogram` - muss in den Elementeigenschaften des Histogramm-Elementes angegeben werden. Nach einem Klick in das Eingabefeld der Eigenschaft **Datenarray** 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.

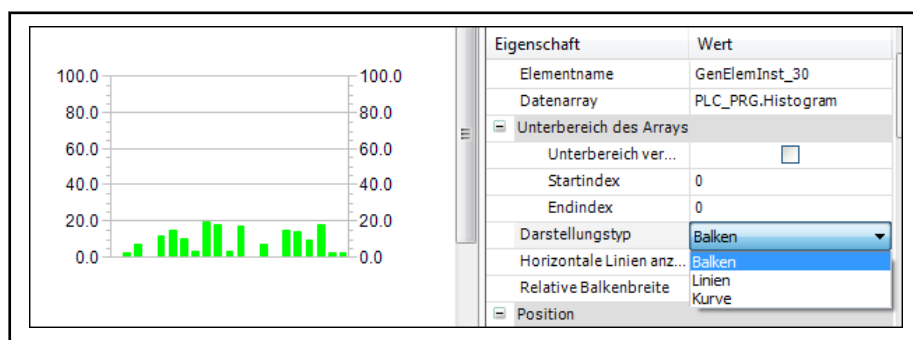
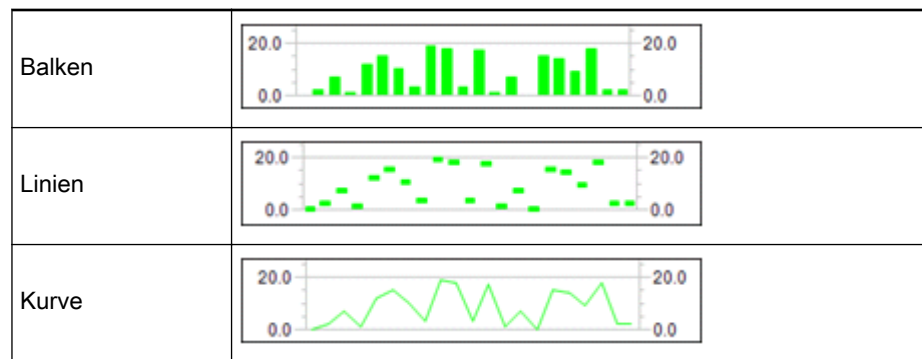


Abb. 20-67: Beispiel - Konfiguration des Darstellungstyps

Unterbereich verwenden: Diese Option wird benutzt, wenn nicht alle Elemente des Arrays angezeigt werden sollen, sondern nur die Elemente von Index **Startindex** bis **Endindex**.

Darstellungstyp: Das Datenarray kann in drei unterschiedlichen Formen dargestellt werden:



Horizontale Linien anzeigen: Mit dieser Option werden horizontale Linien an der Hauptskala gezeichnet.

Relative Balkenbreite: Dieser Wert definiert die relative Breite des Balkens bzw. der Linie

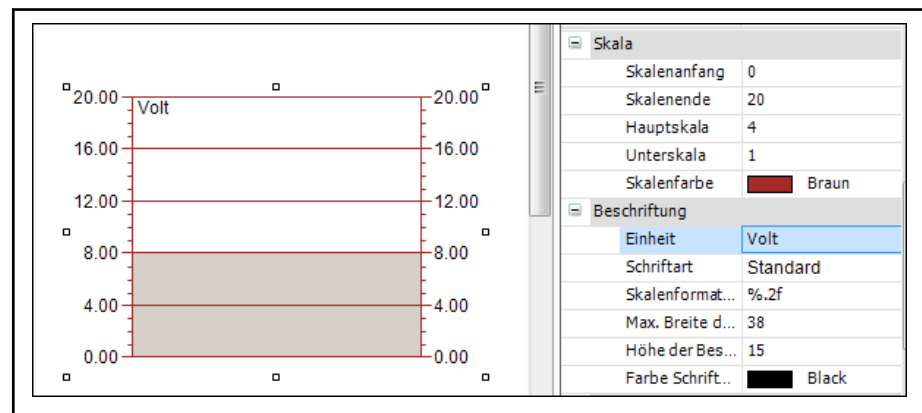


Abb. 20-68: Beispiel - Skalierung des Histogramm

In den Eigenschaften der **Skala** kann der numerische Bereich der Skalierung und die Unterteilung durch Haupt- und Unterskala definiert werden. Der Wertebereich der Skala wird von unten durch den Wert in **Skalenanfang** und von oben durch den Wert in **Skalenende** begrenzt. Der Wert von **Skalenanfang** muss also geringer sein als der von **Skalenende**. Beide Distanzangaben in **Hauptskala** wie **Unterskala** können auf 0 gesetzt werden um eine Anzeige der Skalierungsstriche auszuschalten: Wird der Wert der Grobskalierung (Hauptskala) auf 0 gesetzt, so werden unabhängig vom Wert der Feinskalierung keinerlei Skalierungsstriche gezeichnet. Wird die Feinskalierung (Unterskala) auf 0 gesetzt, so werden nur die Skalierungsstriche der groben Unterteilung gezeichnet. Um die Skalenfarbe zu ändern, muss auf das Feld der Skalenfarbe geklickt werden - es öffnet sich ein Dialog, in welchem die Farbe ausgewählt werden kann.

Im Abschnitt **Beschriftung** kann die Darstellung der Skalierung festgelegt werden. Der Eintrag im Eingabefeld **Einheit** ist zur Angabe der Einheit der Skalierung gedacht und wird links oben im Histogramm angezeigt. Nach Wahl einer geeigneten Schrift(farbe) kann das Format der Beschriftungen angepasst werden: 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).

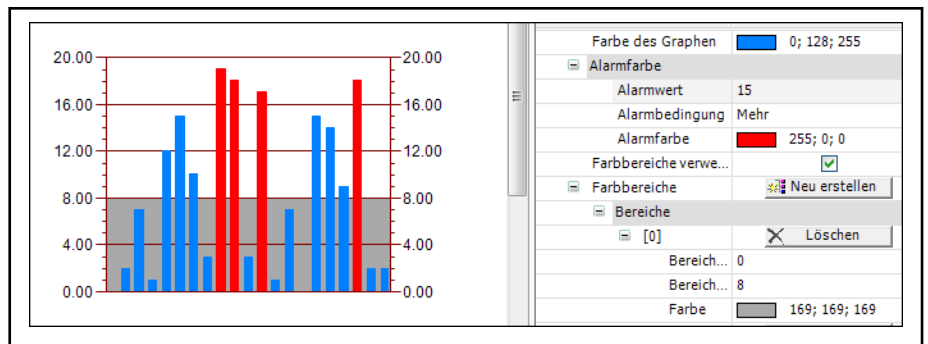


Abb. 20-69: Beispiel - Konfiguration der Farben des Histogramm

Schließlich kann die Einfärbung des Elements innerhalb des Abschnitts **Alarmfarbe** gesetzt werden:

Zuerst wird die Farbe des Histogramm ausgewählt. Innerhalb des Unterbereichs **Alarmfarbe** kann eine Alarmfarbe definiert werden, in welcher der einzelne Balken angezeigt wird, falls der aktuelle Wert des Array-Elementes die Alarmbedingung erfüllt. Diese Bedingung ergibt sich aus der Definition des Alarmwerts, welcher entweder nicht überschritten (beim Setzen der **Alarmbedingung** auf **Mehr**) oder nicht unterschritten werden darf (beim Setzen der **Alarmbedingung** auf **Weniger**).

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 Farbbereich wird innerhalb der Elementeigenschaften mit eigenen Eingabefeldern ausgestattet:

In **Bereichsanfang** und **Bereichsende** 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.

20.4.5 'Lampen/Schalter/Bilder'

Element 'Bildwechsler'

Das Element **Bildwechsler** stellt 1 Bitmap aus den 3 referenzierten Bitmaps dar. Wenn zur Laufzeit der Visualisierung eine Mausektion auftritt, wechselt das dargestellte Bild. Die Folgen von Mausektionen sind über **Elementverhalten** einstellbar.

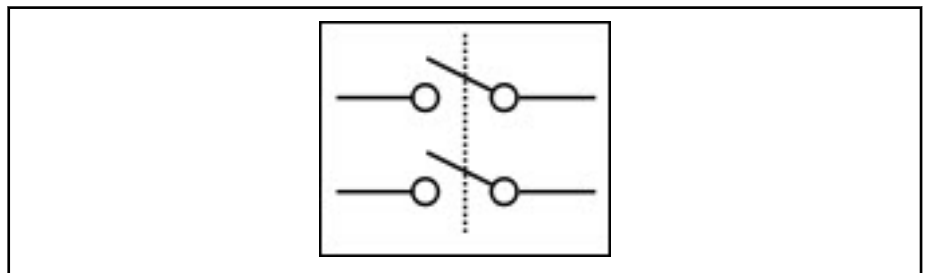


Abb. 20-70: Beispiel für "Bild an"

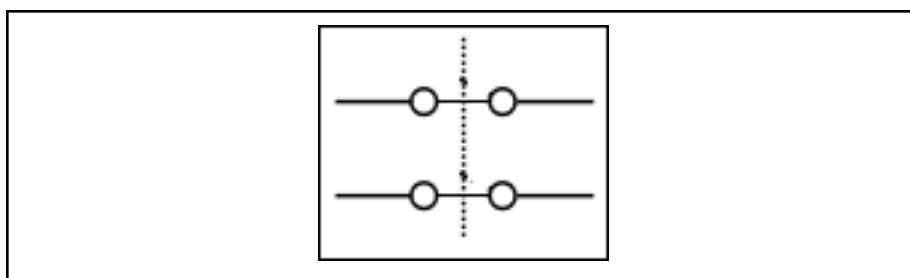


Abb. 20-71: Beispiel für "Bild aus"

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Bildwechsler

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Variable	Boolesche Variable, deren Zustand entsprechend der Benutzereingabe wechselt. Abhängig von Elementverhalten ist die Variable TRUE, solange die Maustaste gedrückt ist (Bildtaster) oder der Wert ändert sich mit jedem Mausklick (Bildumschalter).

Elementeigenschaft 'Bildeinstellungen'

Um die Einträge für die Bildernamen einheitlich zu halten, sollte der Name der Bildersammlung als Präfix angegeben werden. Dies ist nicht notwendig, wenn die Bilder im GlobalImagePool verwaltet werden, da diese sowieso als erste durchsucht wird.

Bild an	ID eines Bitmaps aus Bildersammlung. Das Bild wird im An-Zustand angezeigt. Typ STRING
Bild aus	ID eines Bitmaps aus Bildersammlung. Das Bild wird im Aus-Zustand angezeigt. Typ STRING
Bild gedrückt	ID eines Bitmaps aus Bildersammlung. Die Visualisierung zeigt zur Laufzeit das referenzierte Bitmap an, sobald das Element die Mausektion "Rechte Maustaste gedrückt" empfängt. Voraussetzung: Elementverhalten ist Bildumschalter .
Transparenz	 Bitmap ist für einen Teilbereich voll-transparent gezeichnet, und zwar für alle Pixel in der Transparenzfarbe .
Transparenzfarbe	Farbe, die im Bitmap voll-transparent dargestellt ist. Beispiel: Wenn der Hintergrund des Bitmaps weiß ist und auch die Transparenzfarbe weiß ist, dann wird der Hintergrund voll-transparent gezeichnet. Voraussetzung: Transparenz ist aktiviert.

Skalierungsart	Hier ist zu spezifizieren, wie das Element auf die Änderung der Framegröße zu reagieren hat: • Isotropisch: Höhe und Breite des Bildes werden proportional zum Frame verändert. Beachten Sie Folgendes, wenn Sie die Ausrichtung der Elemente zueinander auch innerhalb eines skalierten "Frame"-Elements beibehalten möchten. Beispielsweise haben Sie eine Lampe zentriert oberhalb eines Schalters angeordnet, und sie soll in dieser horizontal zentrierten Position bleiben, auch wenn sich die Größe des Frames verändert. Um unerwünschte horizontale oder vertikale Verschiebungen zu vermeiden, definieren Sie explizit die Horizontale bzw. Vertikale Ausrichtung mit Zentriert (siehe unten). • Anisotropisch: Das Bild füllt, unabhängig von seinen Proportionen, den ganzen Frame aus. • Unskaliert: Die Bildgröße ist festgelegt und wird nicht verändert, wenn der Frame seine Größe ändert.
Horizontale Ausrichtung	Nur verfügbar, wenn die Option "Experte" für den Eigenschaften-Editor aktiviert ist und wenn die Skalierungsart des Bildes "Isotropisch" ist. Dient dazu, die horizontale Ausrichtung zu einem anderen Element (wie in der Basis-Visualisierung definiert) explizit aufrechtzuerhalten, wenn die Visualisierung innerhalb eines skalierten Frames verwendet wird. Siehe oben: Einstellung "Isotropisch" • Links • Zentriert • Rechts
Vertikale Ausrichtung	Nur verfügbar, wenn die Option "Experte" für den Eigenschaften-Editor aktiviert ist und wenn die Skalierungsart des Bildes "Isotropisch" ist. Dient dazu, die vertikale Ausrichtung zu einem anderen Element (wie in der Basis-Visualisierung definiert) explizit aufrechtzuerhalten, wenn die Visualisierung innerhalb eines skalierten Frames verwendet wird. Siehe oben: Einstellung "Isotropisch" • Oben • Zentriert • Unten
Elementverhalten	• Bildumschalter: Der Zustand des Elements und damit Bild und Variable toggeln mit jedem Mausklick. • Bildtaster: Bild an wird angezeigt und die Variable ist <code>TRUE</code>, solange die Maustaste gedrückt gehalten wird. In diesem Zustand wird das Bild, das in Bild gedrückt referenziert ist, nicht verwendet.
FALSE Tasten	Wenn deaktiviert (Standardeinstellung), wird Variable durch das Drücken der Maustaste auf <code>TRUE</code> gesetzt. Wenn aktiviert, wird Variable auf <code>FALSE</code> gesetzt. Voraussetzung: In Elementverhalten ist Bildwechsler eingestellt.

Elementeigenschaft 'Texte'

Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel `%s`, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt.

[Kap. "Text und Sprache verwenden" auf Seite 845](#)

Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %, enthalten. Die zugehörige Variable ist in Textvariablen ▶ Tooltip-Variable zugewiesen.

Elementeigenschaft 'Zustandsvariablen'

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Element 'Lampe'

Die Lampe leuchtet auf, wenn die zugewiesene Variable gesetzt wird. In **Hintergrundbild** kann die Farbe der Lampe ausgewählt werden.

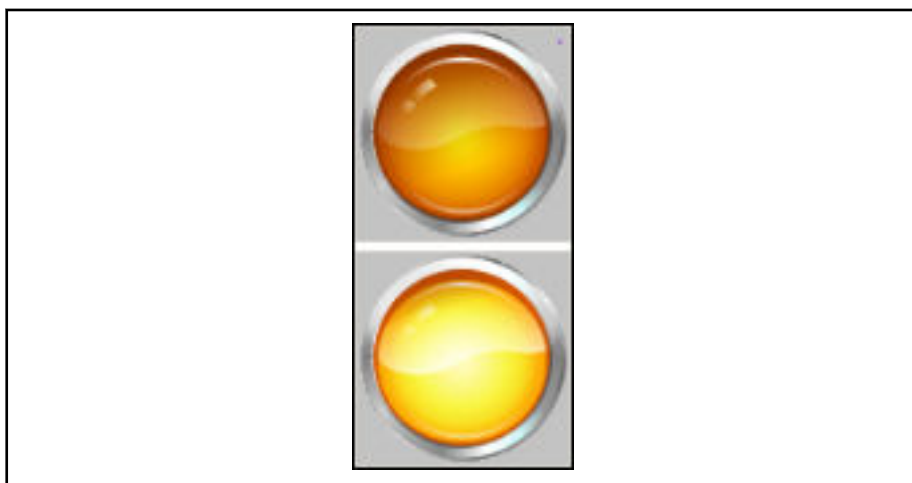


Abb. 20-72:

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Lampe

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Variable	Zuweisen einer booleschen Variable, deren Zustand den Zustand der Lampe schaltet. Bei TRUE wird die Lampe eingeschaltet und leuchtend dargestellt.

Elementeigenschaft 'Bildeinstellungen'

Skalierungsart	Hier ist zu spezifizieren, wie das Element auf die Änderung der Framegröße zu reagieren hat: • Isotropisch: Höhe und Breite des Bildes werden proportional zum Frame verändert. Beachten Sie folgendes, wenn Sie die Ausrichtung der Elemente zueinander auch innerhalb eines skalierten "Frame"-Elements beibehalten möchten. Beispielsweise haben Sie eine Lampe zentriert oberhalb eines Schalters angeordnet, und sie soll in dieser horizontal zentrierten Position bleiben, auch wenn sich die Größe des Frames verändert. Um unerwünschte horizontale oder vertikale Verschiebungen zu vermeiden, definieren Sie explizit die Horizontale bzw. Vertikale Ausrichtung mit "Zentriert" (siehe unten). • Anisotropisch: Das Bild füllt, unabhängig von seinen Proportionen, den ganzen Frame aus. • Unskaliert: Die Bildgröße ist festgelegt und wird nicht verändert, wenn der Frame seine Größe ändert.
Horizontale Ausrichtung	Nur verfügbar, wenn die Option "Experte" für den Eigenschaften-Editor aktiviert ist und wenn die Skalierungsart des Bildes "Isotropisch" ist. Dient dazu, die horizontale Ausrichtung zu einem anderen Element (wie in der Basis-Visualisierung definiert) explizit aufrechtzuerhalten, wenn die Visualisierung innerhalb eines skalierten Frames verwendet wird. Siehe oben: Einstellung "Isotropisch" • Links • Zentriert • Rechts
Vertikale Ausrichtung	Nur verfügbar, wenn die Option "Experte" für den Eigenschaften-Editor aktiviert ist und wenn die Skalierungsart des Bildes "Isotropisch" ist. Dient dazu, die vertikale Ausrichtung zu einem anderen Element (wie in der Basis-Visualisierung definiert) explizit aufrechtzuerhalten, wenn die Visualisierung innerhalb eines skalierten Frames verwendet wird. Siehe oben: Einstellung "Isotropisch" • Oben • Zentriert • Unten

Elementeigenschaft 'Texte'

Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt.

[Kap. "Text und Sprache verwenden" auf Seite 845](#)

Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ► Tooltip-Variable zugewiesen.

**Elementeigenschaft
'Zustandsvariablen'**

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Elementeigenschaft 'Hintergrund'

Hintergrundbild	<i>Auszuwählen ist eine Farbe für die Lampe:</i> • Gelb • Rot • Grün • Blau • Grau

Element 'Hebelschalter', 'Energieschalter', 'Drückschalter', 'Drückschalter LED', 'Kippschalter'

Der Schalter wird verwendet um den Wert einer booleschen Variablen zu setzen

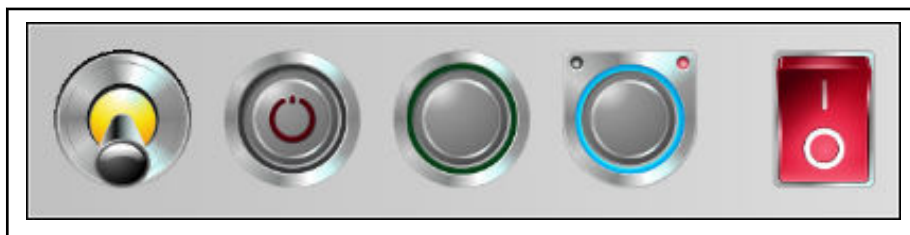


Abb. 20-73: Hebelschalter, Energieschalter, Drückschalter, Drückschalter LED, Kippschalter

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,

- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Typ des selektierten Elements: Hebelschalter, Energieschalter, Drückschalter, Drückschalter LED oder Kippschalter.

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Variable	boolesche Variable, deren Wert entsprechend der Benutzereingabe wechselt. Abhängig von Elementverhalten ist die Variable TRUE, solange die Maustaste gedrückt ist (Bildtaster) oder der Wert ändert sich mit jedem Mausklick (Bildumschalter).

Elementeigenschaft 'Bildeinstellungen'

Skalierungsart	Hier ist zu spezifizieren, wie das Element auf die Änderung der Framegröße zu reagieren hat: • Isotropisch: Höhe und Breite des Bildes werden proportional zum Frame verändert. Beachten Sie folgendes, wenn Sie die Ausrichtung der Elemente zueinander auch innerhalb eines skalierten "Frame"-Elements beibehalten möchten. Beispielsweise haben Sie eine Lampe zentriert oberhalb eines Schalters angeordnet, und sie soll in dieser horizontal zentrierten Position bleiben, auch wenn sich die Größe des Frames verändert. Um unerwünschte horizontale oder vertikale Verschiebungen zu vermeiden, definieren Sie explizit die Horizontale bzw. Vertikale Ausrichtung mit "Zentriert" (siehe unten). • Anisotropisch: Das Bild füllt, unabhängig von seinen Proportionen, den ganzen Frame aus. • Unskaliert: Die Bildgröße ist festgelegt und wird nicht verändert, wenn der Frame seine Größe ändert.
Horizontale Ausrichtung	Nur verfügbar, wenn die Option "Experte" für den Eigenschaften-Editor aktiviert ist und wenn die Skalierungsart des Bildes "Isotropisch" ist. Dient dazu, die horizontale Ausrichtung zu einem anderen Element (wie in der Basis-Visualisierung definiert) explizit aufrechtzuerhalten, wenn die Visualisierung innerhalb eines skalierten Frames verwendet wird. Siehe oben: Einstellung "Isotropisch" • Links • Zentriert • Rechts
Vertikale Ausrichtung	Nur verfügbar, wenn die Option "Experte" für den Eigenschaften-Editor aktiviert ist und wenn die Skalierungsart des Bildes "Isotropisch" ist. Dient dazu, die vertikale Ausrichtung zu einem anderen Element (wie in der Basis-Visualisierung definiert) explizit aufrechtzuerhalten, wenn die Visualisierung innerhalb eines skalierten Frames verwendet wird. Siehe oben: Einstellung "Isotropisch" • Oben • Zentriert • Unten

Elementverhalten	• Bildumschalter: Der Schalter wird als eingeschaltet angezeigt, und die Variable ist TRUE, solange die Maustaste gedrückt gehalten wird. • Bildwechsler: Die Anzeige des Schalters und Variable toggeln mit jedem Mausklick.

Elementeigenschaft 'Texte'

Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt.

[Kap. "Text und Sprache verwenden" auf Seite 845](#)

Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %, enthalten. Die zugehörige Variable ist in Textvariablen ► Tooltip-Variable zugewiesen.

**Elementeigenschaft
'Zustandsvariablen'**

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Elementeigenschaft 'Hintergrund'

Hintergrundbild	<i>Auszuwählen ist eine Farbe, die den Schalter markiert:</i> • Gelb • Rot • Grün • Blau • Grau

Element 'Drehschalter'

Der Drehschalter wird verwendet, um den Wert einer booleschen Variablen zu setzen.



Abb. 20-74:

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Drehschalter

Position

Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Variable	boolesche Variable, deren Wert entsprechend der Benutzereingabe wechselt. Abhängig von Elementverhalten ist die Variable TRUE, solange die Maustaste gedrückt ist (Bildtaster) oder der Wert ändert sich mit jedem Mausklick (Bildumschalter).

Elementeigenschaft 'Bildeinstellungen'

Skalierungsart	Hier ist zu spezifizieren, wie das Element auf die Änderung der Framegröße zu reagieren hat: - Isotropisch: Höhe und Breite des Bildes werden proportional zum Frame verändert. Beachten Sie folgendes, wenn Sie die Ausrichtung der Elemente zueinander auch innerhalb eines skalierten "Frame"-Elements beibehalten möchten. Beispielsweise haben Sie eine Lampe zentriert oberhalb eines Schalters angeordnet, und sie soll in dieser horizontal zentrierten Position bleiben, auch wenn sich die Größe des Frames verändert. Um unerwünschte horizontale oder vertikale Verschiebungen zu vermeiden, definieren Sie explizit die Horizontale bzw. Vertikale Ausrichtung mit "Zentriert" (siehe unten). - Anisotropisch: Das Bild füllt, unabhängig von seinen Proportionen, den ganzen Frame aus. - Unskaliert: Die Bildgröße ist festgelegt und wird nicht verändert, wenn der Frame seine Größe ändert.
Horizontale Ausrichtung	Nur verfügbar, wenn die Option "Experte" für den Eigenschaften-Editor aktiviert ist und wenn die Skalierungsart des Bildes "Isotropisch" ist. Dient dazu, die horizontale Ausrichtung zu einem anderen Element (wie in der Basis-Visualisierung definiert) explizit aufrechtzuerhalten, wenn die Visualisierung innerhalb eines skalierten Frames verwendet wird. Siehe oben: Einstellung "Isotropisch" - Links - Zentriert - Rechts
Vertikale Ausrichtung	Nur verfügbar, wenn die Option "Experte" für den Eigenschaften-Editor aktiviert ist und wenn die Skalierungsart des Bildes "Isotropisch" ist. Dient dazu, die vertikale Ausrichtung zu einem anderen Element (wie in der Basis-Visualisierung definiert) explizit aufrechtzuerhalten, wenn die Visualisierung innerhalb eines skalierten Frames verwendet wird. Siehe oben: Einstellung "Isotropisch" - Oben - Zentriert - Unten
Elementverhalten	- Bildumschalter: Der Drehschalter wird als eingeschaltet angezeigt, und die Variable ist TRUE, solange die Maustaste gedrückt gehalten wird. - Bildwechsler: Zustand des Drehschalters und Variable toggeln mit jedem Mausklick.
Ausrichtung	- Oben: Der Schalter ist nach oben gerichtet. - Seitlich: Der Schalter ist seitlich gerichtet.
Farbwechsel	Ist die Option nicht aktiviert, dann leuchtet der Schalter auf, wenn er auf eingeschaltet ist. Ist die Option aktiviert, dann leuchtet der Schalter nicht auf, auch wenn er auf eingeschaltet ist.

Elementeigenschaft 'Texte'

Diese Eigenschaften werden für eine statische Definition der Elementbeschriftung verwendet. Jede kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Im Online-Betrieb wird die Sequenz mit dem Inhalt der Variablen definiert in **Textvariablen** ersetzt.

[Kap. "Text und Sprache verwenden" auf Seite 845](#)

Tooltip	Geben Sie einen Text ein. Er wird als Tooltip des Elements verwendet. Er erscheint in der Visualisierung nur im Online-Betrieb. Der Text kann eine Formatierungssequenz, wie zum Beispiel %s, enthalten. Die zugehörige Variable ist in Textvariablen ▶ Tooltip-Variable zugewiesen.

Elementeigenschaft 'Zustandsvariablen'

Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Betrieb unsichtbar.
Eingaben deaktivieren	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann hat keine der Elementeingaben eine Auswirkung. Außerdem wird das Element selbst in der Visualisierung deaktiviert um zu kennzeichnen, dass keine Benutzereingabe möglich ist. In dem Fall, dass die Visualisierung die Benutzerverwaltung verwendet, werden die Elemente für Benutzergruppen mit Zugriffsrecht "nur sichtbar" deaktiviert.

Elementeigenschaft 'Hintergrund'

Hintergrundbild	<i>Auszuwählen ist eine Farbe, die den Schalter markiert:</i> • Gelb • Rot • Grün • Blau • Grau

20.4.6 'Spezielle Steuerelemente'

Element 'Trace' und seine Dialoge

Element 'Trace'

Inhalt dieser Seite:

- ["Elementeigenschaften" auf Seite 1034](#)
- ["Wie ein minimales Visualisierungselement Trace zu konfigurieren ist" auf Seite 1035](#)
- ["Auswirkung von 'Cursor anzeigen'" auf Seite 1038](#)
- ["Wie ein Visualisierungstrace mit Trigger zu konfigurieren ist" auf Seite 1039](#)

Das Element ermöglicht es, einen Tracegraphen, der permanent Variablenwerte monitort und anzeigt, in die Visualisierung zu integrieren. Der angezeigte Tracegraph wird in den Elementeigenschaften konfiguriert. Zusätzlich sind Steuerelemente, die die Tracefunktionalität steuern, entweder von Hand oder mit Hilfe des [Kap. "Dialog 'Traceassistent'" auf Seite 1048](#) zu programmieren.

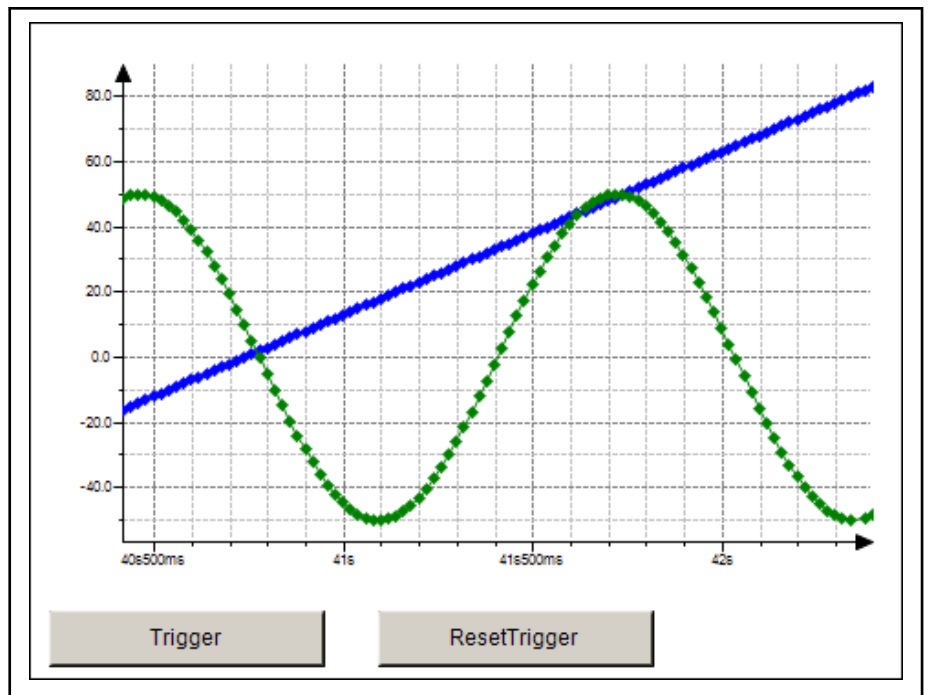


Abb. 20-75: Visualisierungstrace online

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseeditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
- : Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Applikation	Hier ist die Applikation zu spezifizieren, unter der die Tracevariablen implementiert sind. Default ist die Applikation, unter der sich die Visualisierung befindet.

Elementtyp	Trace

Trace	Mit einem Klick in das (unselektierte) Feld öffnet der Dialog Kap. "Dialog 'Tracekonfiguration, Aufzeichnungseinstellungen'" auf Seite 1041 zur Konfiguration des Tracegraphen.

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Cursor anzeigen	Wird diese Option aktiviert, werden die Trigger- und Variablenwerte, auf die der Cursor zeigt, als Tooltip angezeigt.

Bestehenden Trace überschreiben	Ist ein Trace mit gleichem Namen auf der SPS, dann wird dieser während eines Downloads mit der Konfiguration, die hier erstellt ist, überschrieben

Zahlenformat	Formatierungsangabe für den Variablenwert im Tooltip, zum Beispiel %2.5f

Siehe auch

- [Kap. "Text und Sprache verwenden" auf Seite 845](#)

Elementeigenschaft 'Steuervariablen'

Trace zurücksetzen	Geben Sie eine boolesche Variable ein. Wenn deren Wert TRUE wird, dann wird der Trigger zurückgesetzt.
Trace starten	Geben Sie eine boolesche Variable ein. Wenn deren Wert TRUE wird, dann wird der Trace gestartet.
Trace anhalten	Geben Sie eine boolesche Variable ein. Wenn deren Wert TRUE wird, dann wird der Trace angehalten.
Trace in Datei speichern Trace speichern Dateiname	Wollen Sie die aktuelle Tracekonfiguration in einer Datei zu speichern, dann geben sie eine boolesche Variable als Wert von Trace speichern ein und eine Stringvariable als Wert von Dateiname . Wird die Variable Trace speichern TRUE, dann wird die Konfiguration unter dem spezifizierten Namen gespeichert.
Trace aus Datei laden Trace laden Dateiname	Wollen Sie eine Tracekonfiguration laden, dann geben Sie eine boolesche Variable als Wert von Trace laden an und eine Stringvariable als Wert von Dateiname . Wird die Variable Trace laden TRUE, dann wird die Konfiguration der spezifizierten Datei geladen.

Wie ein minimales Visualisierungselement Trace zu konfigurieren ist

Beispiel "MinimumTrace"

Schrittweise

1. Deklarieren Sie Projektvariable für das Steuern (Trace starten und Trace anhalten) in IEC-Code. Implementieren Sie eine Tracevariable.
2. Setzen die Einstellungen in Eigenschaft **Trace** des Visualisierungselements. Ein minimaler Trace enthält eine Tracevariable
3. Weisen Sie die Projektvariablen unter **Steuervariablen** zu, mindestens für **Trace anhalten** und **Trace starten**.
4. Implementieren Sie Code für die Trace-Steuerung

In diesem Beispiel wird ein Visualisierungselement Trace über Projektvariablen mit Eingabebutten verbunden, um den Trace zu steuern und die Wertekurve der IEC-Variablen "iVar" anzuzeigen.

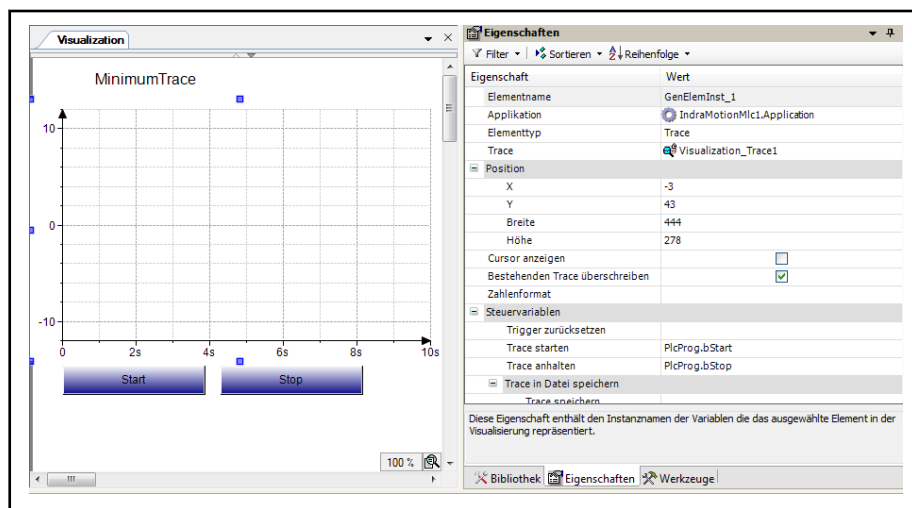


Abb. 20-76: Visualisierung "MinimumTrace"

Schritt 1: IEC-Code im Programm "PlcProg" unter Task "PlcTask" ablegen:

```
PROGRAM PlcProg
VAR
  iVar: INT;
END_VAR
VAR_INPUT
  bStart: BOOL;
```

```

bStop: BOOL;
END_VAR
iVar := iVar + 1;

```

Schritt 2: Zur Applikation das Traceobjekt "MinimumTrace" hinzufügen, die Variable "iVar" als Tracevariable unter Task "PlcTask" konfigurieren.

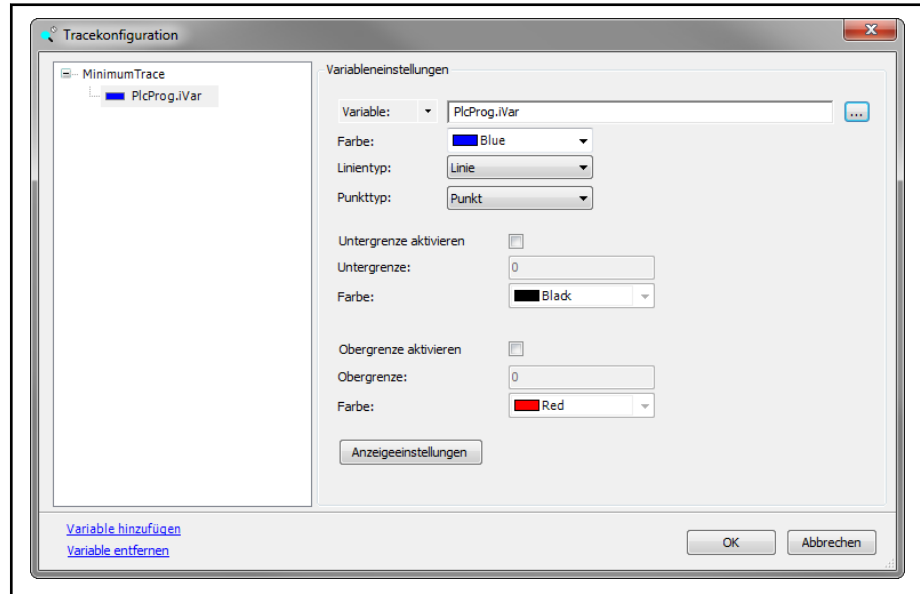


Abb. 20-77: Tracekonfiguration 'MinimumTrace', Variable hinzufügen

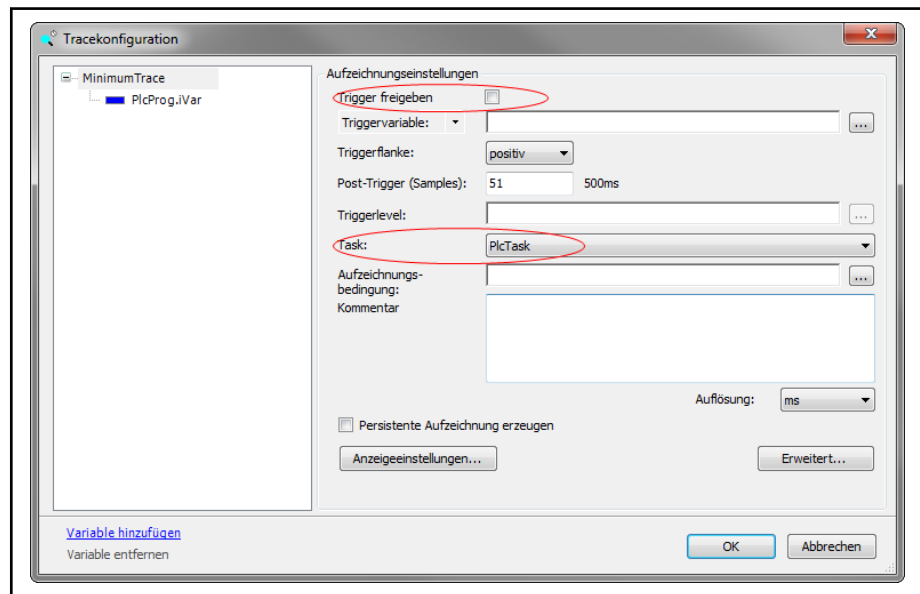


Abb. 20-78: Tracekonfiguration 'MinimumTrace', Konfiguration

Schritt 3: Zur Applikation das Visualisierungsobjekt "Visualization" hinzufügen.

Die Visualisierung besteht aus dem Textfeld "MinimumTrace", zwei Schaltflächen "Start" und "Stop" (Allgemeine Steuerelemente) und dem Visualisierungselement "Trace" (Spezielle Steuerelemente).

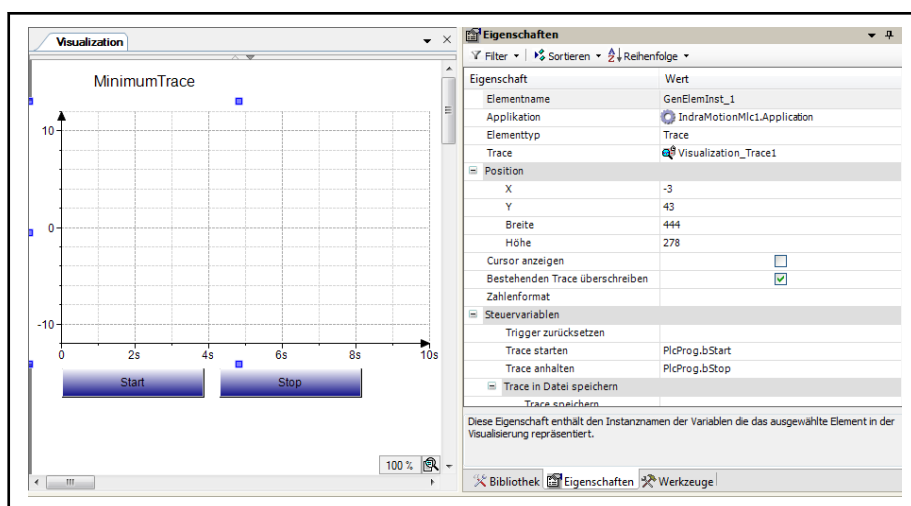


Abb. 20-79: Eigenschaften des Traceobjektes "Visualization_Trace1"

Einstellungen in Eigenschaft **Steuervariablen**

Einstellungen in Eigenschaft **Steuervariablen**

Eigenschaft	Wert
Trace starten	PlcProg.bStart
Trace anhalten	PlcProg.bStop

Schritt 4: Konfiguration der Schaltflächen "Start" und "Stop" der Trace-Steuerung

ST-Code für die Schaltfläche "Stop" programmiert in ihren **Eigenschaften** unter **Eingabekonfiguration** ► **OnMouseDown** ► **ST-Codeausführung**

```
PlcProg.bStart := FALSE;
PlcProg.bStop := TRUE;
```

ST-Code für die Schaltfläche "Start" programmiert in ihren **Eigenschaften** unter **Eingabekonfiguration** ► **OnMouseDown** ► **ST-Code ausführen**

```
PlcProg.bStart := TRUE;
PlcProg.bStop := FALSE;
```

Schritt 5: Traceobjekt und Visuobjekt Trace verbinden

Visuobjekt Trace markieren, Kontextmenü "Tace konfigurieren", Verbindung herstellen.

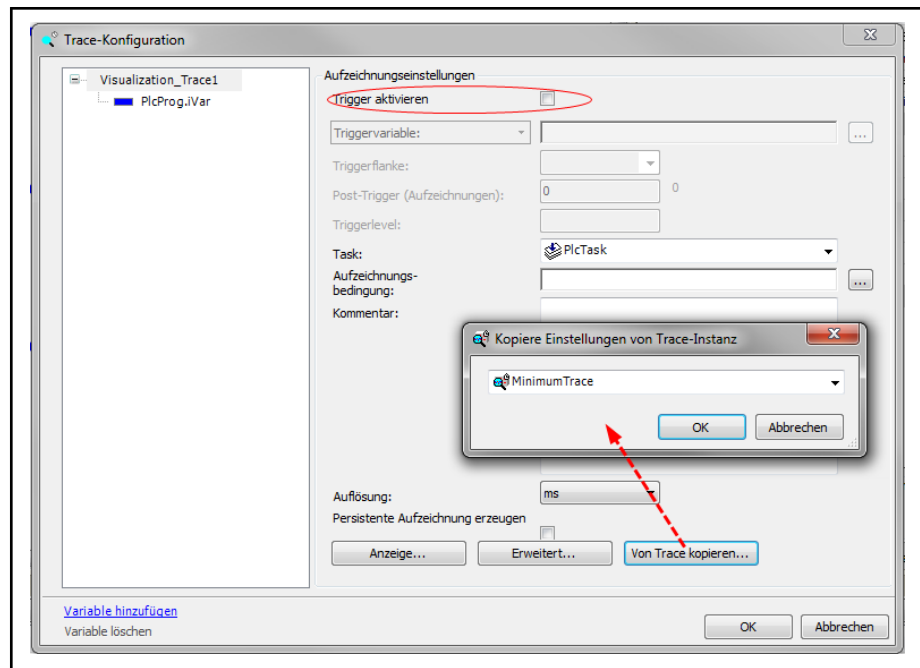


Abb. 20-80: Verbindung zur Traceinstanz "MinimumTrace" herstellen

Schritt 6: Applikation einloggen und starten; "MinimumTrace" starten und stoppen.

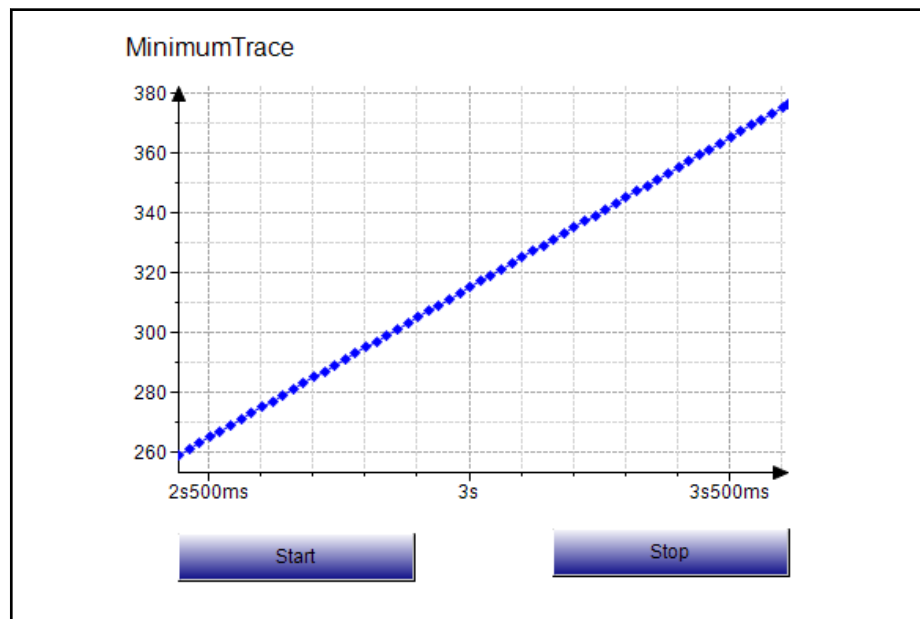


Abb. 20-81: "MinimumTrace" online

Auswirkung von 'Cursor anzeigen'

Ist die Option **Cursor anzeigen** in den Elementeneigenschaften des Trace-Visualisierungselements aktiviert, ist es umgehend möglich, im Tooltip die Werte des Triggers und der Tracevariablen an der Position, an der der Cursor im Trace-Visualisierungselement steht, angezeigt zu bekommen.

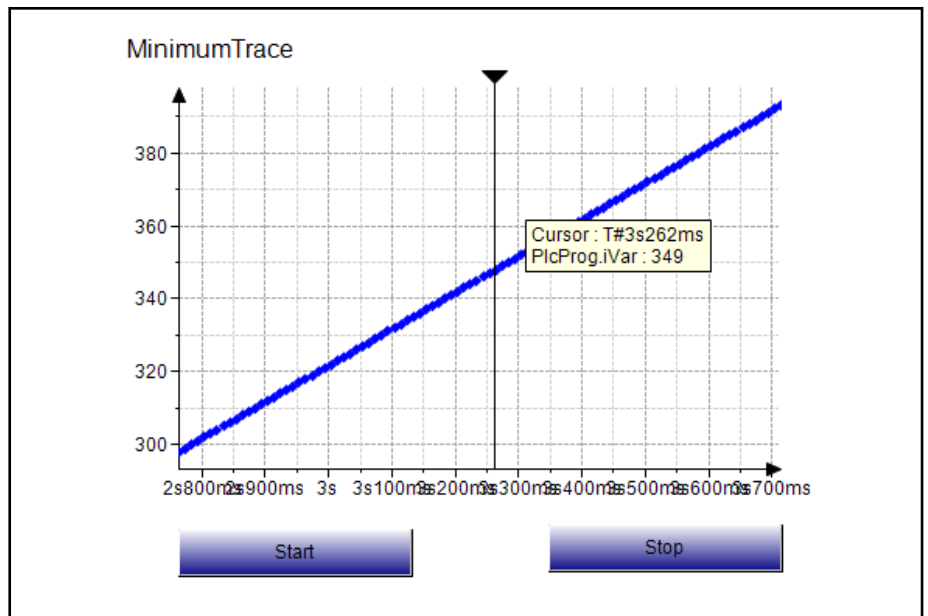


Abb. 20-82: Auswirkung der Eigenschaft 'Cursor anzeigen'

Wie ein Visualisierungstrace mit Trigger zu konfigurieren ist

Konfigurieren Sie folgende Eigenschaften

1. Deklarieren die die benötigten Projektvariablen in IEC-Code und implementieren diese um damit den Trace zu triggern und zurückzusetzen.
2. **Trace:** Konfigurieren sie das Triggersystem in Dialog [Kap. "Dialog 'Tracekonfiguration, Aufzeichnungseinstellungen'" auf Seite 1041](#)
3. **Steuervariablen:** Weisen Sie ein Projektvariabel für **Trigger zurücksetzen** zu.
4. Implementieren Sie eine Trace-Steuerung für Trigger und Zurücksetzen

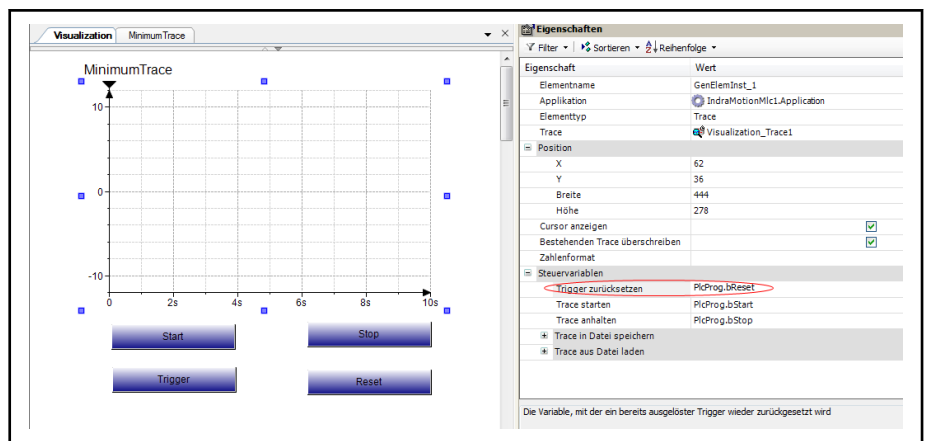


Abb. 20-83: Erweiterung des "MinimumTrace" um "Reset" und "Trigger"

Schritt 1: IEC-Code im Programm "PlcProg" unter Task "PlcTask" ablegen:

```
PROGRAM PlcProg
VAR
  iVar: INT;
END_VAR
VAR_INPUT
  bStart: BOOL;
  bStop: BOOL;
  bTrigger: BOOL;
  bReset: BOOL;
END_VAR
iVar := iVar + 1;
```

Schritt 2: Aktivieren des Triggers im Traceobjekt "MinimumTrace", Konfiguration

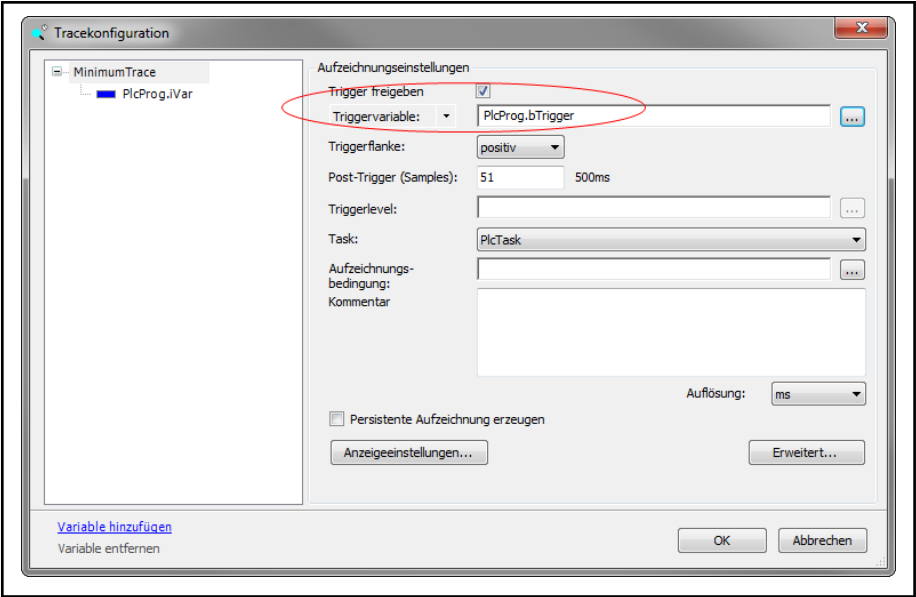


Abb. 20-84: Erweiterung der Tracekonfiguration

Schritt 3: Einstellungen in Eigenschaft Steuervariablen

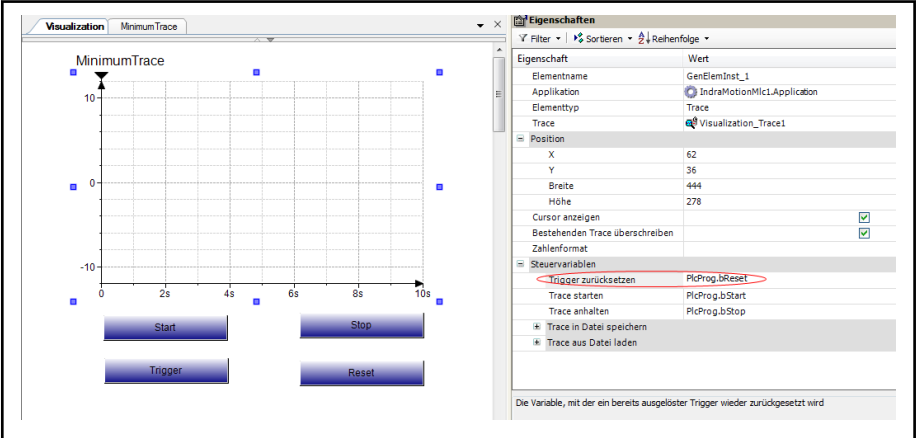


Abb. 20-85: Erweiterung des "MinimumTrace" um "Reset"

 "Cursor anzeigen", bitte deaktivieren.

Reset	PlcProg.bReset

Schritt 4: Konfiguration der Schaltflächen "Trigger" und "Reset " der Trace-Steuerung

Schaltfläche "Trigger": Triggerung des Traces durch Setzen der zugehörigen Tracevariablen auf TRUE wird aktiviert.

Schaltfläche "Reset" setzt den Trigger zurück.

ST-Code für Schaltfläche "Trigger", **Eigenschaften** unter **Eingabekonfiguration** ► **OnMouseDown** ► **ST-Code ausführen**

```
PlcProg.bReset := FALSE;
PlcProg.bTrigger := TRUE;
```


ST-Code für Schaltfläche "Reset", **Eigenschaften** unter **Eingabekonfiguration** ▶ **OnMouseDown** ▶ **ST-Code ausführen**

```
PlcProg.bTrigger:=FALSE;
PlcProg.bReset := TRUE;
```

Das folgende Bild zeigt die Trace-Visualisierung in Online-Betrieb, nachdem die Triggervariable durch Pressen der Schaltfläche **Trigger** aktiviert wurde:

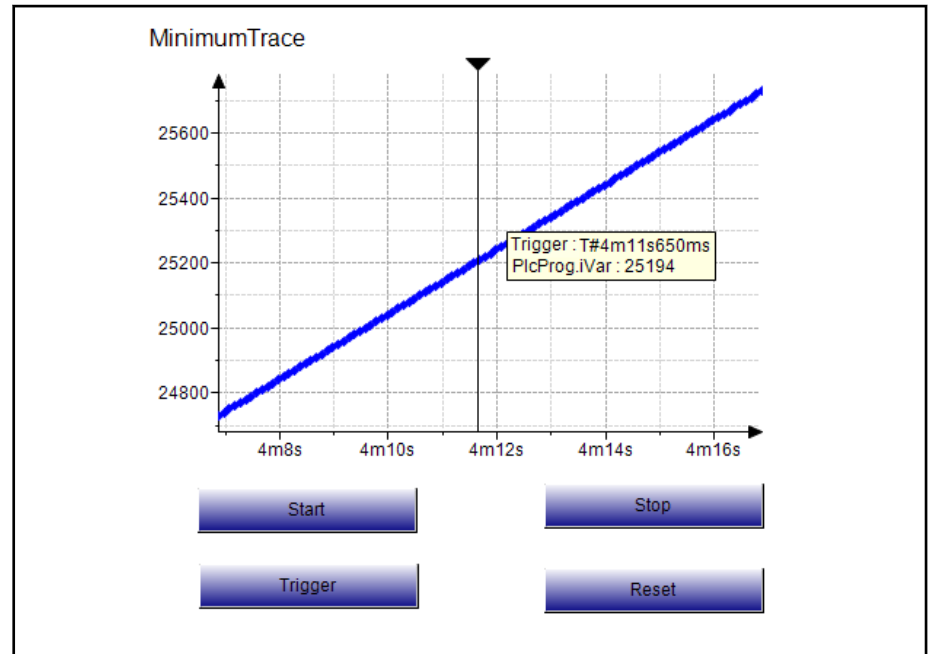


Abb. 20-86: Visualisierung in Online-Betrieb

Die Flanke der Triggervariablen ist durch eine schwarze vertikale Linie markiert. Das Aufzeichnen der Variablen wird für den speziellen Triggerlevel für das gesamte Triggerintervall fortgesetzt, nachdem die Visualisierung unterbrochen wurde.

Dialog 'Tracekonfiguration, Aufzeichnungseinstellungen'

Der Dialog **Tracekonfiguration, Aufzeichnungseinstellungen** öffnet mit Klick auf das Objekt Visualisierungstrace in **Ansicht** ▶ **Elementeigenschaften** ▶ **Trace**. Der Dialog ist nahezu gleich zu "Tracekonfiguration, Aufzeichnungseinstellungen" des Traceeditors ([Kap. "Dialog 'Tracekonfiguration'" auf Seite 783](#)).

Grundlegendes zum Trigger

Grundsätzlich kann auf folgende Weise das Aufzeichnen von Tracevariablen gesteuert werden:

- durch Starten und Stoppen der Aufzeichnung,
- durch Verknüpfen des Startens mit einer Aufzeichnungsbedingung
- durch Auslösen eines Triggers, der definiert das Aufzeichnen beendet.

Baumansicht der Tracekonfiguration

Die Baumansicht des Traces visualisiert die Tracekonfiguration und ermöglicht zu den Einstellungen zu navigieren. Der oberste Knoten enthält den Namen des Traces. Mit Klick auf diesen Eintrag erscheinen die dazugehörigen **Aufzeichnungseinstellungen**. Unter dem Knoten befinden sich Knoten für jede Tracevariable, deren Daten erfasst werden. Mit Klick auf einen Eintrag erscheinen die dazugehörigen **Variableneinstellungen**.

Tracekonfiguration exportieren	Der Dialog Tracekonfiguration exportieren erscheint. Voraussetzung: Der oberste Knoten ist selektiert und Sie haben das Kontextmenü geöffnet.
Variable hinzufügen	IndraLogic erzeugt eine weitere Tracevariable, die in der Baumansicht eingehängt wird und mit den zugehörigen Variableneinstellungen erscheint. Voraussetzung: Ein Knoten ist selektiert und Sie haben das Kontextmenü geöffnet.

Tab. 20-8: Verfügbare Befehle

Siehe auch

- [Kap. "Befehl 'Tracekonfiguration exportieren'" auf Seite 1049](#)

Festlegen und Ändern der Aufzeichnungs- (Trigger-)Einstellungen

Voraussetzung: Der Name der Tracekonfiguration, also der oberste Knoten in der Baumansicht, ist selektiert.

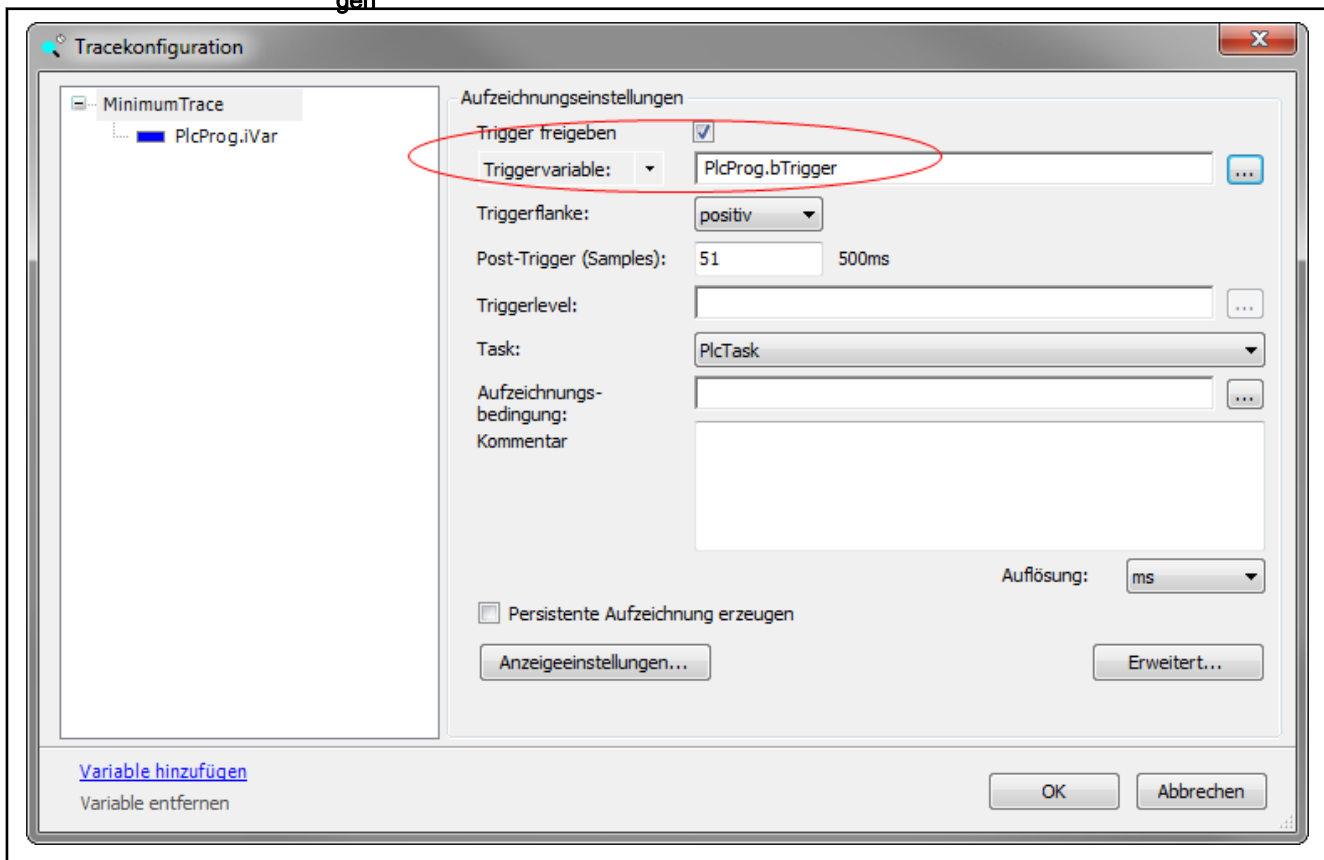


Abb. 20-87: Dialog 'Tracekonfiguration' mit 'Aufzeichnungseinstellungen'

Trigger freigeben: Markieren Sie die Checkbox um das Triggersystem entsprechend den Einstellungen zu aktivieren. Wird das Triggersystem deaktiviert, dann läuft das System unabhängig.

Triggervariable: Geben Sie an, welches Signal als Trigger verwendet wird, indem Sie Name und (und Pfad) eingeben.

Ein gültiges Triggersignal ist eine IEC-Variable, eine Eigenschaft, eine Referenz, ein Pointer, ein Arrayelement der Applikation oder ein Ausdruck. Erlaubte Typen sind alle IEC-basierte Typen **außer** STRING, WSTRING oder ARRAY. Enumerationen sind dann erlaubt, wenn der Basistyp kein STRING, WSTRING oder ARRAY ist. Der Inhalt eines Pointers ist kein gültiges Signal.

Die Eingabeunterstützung  kann verwendet werden um gültige Einträge zu erhalten.

Wollen Sie einen Geräteparameter als Trigger verwenden, klicken Sie um auf **Triggerparameter** umzuschalten. Öffnen Sie die Eingabeunterstützung mit  und selektieren Sie **"Tracebare" Parameter**. Unter **Elemente** werden dann die im System vorhandenen Parameter aufgelistet.

Triggerflanke:

positiv	Der Trigger löst bei steigender Flanke der Boolesche Triggervariable aus oder wenn bei einer analogen Triggervariablen der in Triggerlevel definierte Wert "von unten" her erreicht wird.
negativ	Der Trigger löst bei fallenden Flanke der Boolesche Triggervariable aus oder wenn bei einer analogen Triggervariablen der in Triggerlevel definierte Wert "von oben" her erreicht wird.
beide	Der Trigger löst sowohl bei fallender als auch bei steigender Flanke der Triggervariablen aus oder wenn das gesetzte Triggerlevel erreicht wird.

Post-Trigger (Aufzeichnungen): Geben Sie hier die Anzahl der Aufzeichnungen pro Eingangssignal, nachdem der Trigger gefeuert wurde, an. Standardwert ist 50, der Wertebereich reicht von 0 bis ($2^{32}-1$).

Triggerlevel: Geben Sie hier ein Literal ein, bei dessen Erreichen der Trigger ausgelöst werden soll.

Task: Selektieren Sie aus der Liste der verfügbaren Tasks die, in der die Datenerfassung stattfinden soll.

Aufzeichnungsbedingung: Geben Sie eine Variable von Typ BOOL oder Bitzugriff an. Der Inhalt eines Pointers ist kein gültiger Eintrag. Eigenschaften werden unterstützt.

Kommentar: Geben sie einen Kommentar z.B. über die aktuellen Aufzeichnungsbedingungen ein.

Auflösung: Setzen Sie die Auflösung des Trace-Zeitstempels auf "ms" oder "µs". Für jedes zu erfassende Signal werden Datenpaare, bestehend aus Wert und Zeitstempel, gespeichert und an das Programmiersystem übertragen. Die übertragenen Zeitstempel sind relativ und beziehen sich auf den Start des Tracens. Ist eine Taskzykluszeit kleiner als 1ms vorgegeben, dann wird empfohlen, die Auflösung des Zeitstempels auf "1µs" zu setzen.

Persistente Aufzeichnung erzeugen: Ist diese Option gesetzt, dann wird die Tracekonfiguration und der neueste Inhalt des Tracepuffer des Laufzeitsystems persistent auf dem Zielsystem gespeichert. Diese Option ist nur dann verfügbar, wenn im Laufzeitsystem ein Tracemanager aufzeichnet.

Anzeige....: [Kap. "Dialog 'Anzeigeeinstellungen bearbeiten'" auf Seite 1046](#)

Erweitert....: [Kap. "Dialog 'Erweiterte Trace-Einstellungen'" auf Seite 1046](#)

Vom Trace kopieren: [Kap. "Dialog 'Kopiere Einstellungen von Traceinstanz'" auf Seite 1048](#)

Dialog 'Tracekonfiguration, Variableneinstellung'

Symbol: 

Funktion: Der Dialog enthält die Variablenkonfiguration eines Traces in der Visualisierung.

Aufruf: Doppelklick auf eine Tracevariable in der Baumansicht der Tracekonfiguration

Siehe auch

- [Kap. 18.3.10 "Dialog 'Tracekonfiguration'" auf Seite 782](#), Traceobjekt

Alle Tracevariablen werden im linken Teil des Fensters in einer Baumstruktur angezeigt. Der oberste Knoten trägt den Name des Traces.

Baumansicht der Tracekonfiguration

Die Baumansicht visualisiert die Tracekonfiguration und ermöglicht darin zu navigieren. Der oberste Eintrag enthält den Namen des Traces. Mit Klick auf diesen Eintrag erscheint rechts die dazugehörige Konfiguration unter **Aufzeichnungseinstellungen**. Darunter befindet sich ein Eintrag für jede Variable, deren Daten erfasst werden. Mit Klick auf einen Eintrag erscheint rechts die dazugehörige Konfiguration unter **Variableneinstellungen**.

Befehl	Beschreibung
Variable hinzufügen	Ein neuer aber noch leerer Eintrag erscheint in der Tracekonfiguration. Dabei ist rechts unter Variableneinstellungen die Konfiguration des neuen Eintrags angezeigt, um ein sofortiges Setzen der Einstellungen zu ermöglichen.
Variable entfernen	Selektierte Variable wird entfernt

Registerkarte 'Variableneinstellung'

Enthält die Einstellungen der Variablen, die in der Baumansicht selektiert ist.

Variable	Gültige Variable mit vollständigem Instanzpfad, deren Wert erfasst wird <i>Gültig</i> • IEC-Variable mit gültigem Typ • Eigenschaft • Referenz • Inhalt eines Pointers • Arrayelement, das auf einen gültigen Typ basiert • Enumeration, die auf einen gültigen Typ basiert <i>Gültiger Datentyp: IEC-basierter Typ außer</i> • STRING • WSTRING • ARRAY Wenn das LZS die Komponente <code>CmpTraceMgr</code> verwendet, dann kann eine Eigenschaft, die mit dem Attribut 'monitoring' verknüpft ist, als Variable aufgezeichnet werden.
Parameter	Parameter, dessen Daten erfasst werden. Voraussetzung: Laufzeitsystem mit Komponente <code>CmpTraceMgr</code> Der Dialog Eingabehilfe, Registerkarte Kategorien, Eintrag Parameter listet alle gültigen Parameter des Systems auf.
	Umschaltung zwischen Variable und Parameter
Zugehörige Achse	Y-Achse des Tracediagramm für die Variable . Die Auswahlliste bietet die Standard-Y-Achse und die zusätzlich konfigurierten Y-Achsen an, die im Dialog Anzeigeeinstellungen bearbeiten konfiguriert sind.

Variablenname anzeigen	 : Die Visualisierung zeigt zur Laufzeit im Tracediagramm den Namen der IEC-Variable an. Entweder allein stehend oder in Klammer nach der Beschreibung . Voreinstellung.  : Der Name der IEC-Variable wird nicht angezeigt und erscheint nicht geklammert nach der Beschreibung . Voraussetzung: Wenn in Beschreibung ein Text eingetragen ist, können Sie die Option deaktivieren.
Beschreibung	Text des Tooltips, zum Beispiel <code>Sensor A</code> . Wenn ein Visualisierungsbenutzer im Tracediagramm den Cursor fokussiert, zeigt die Visualisierung ihn als Tooltip an. Der Text ist im Objekt GlobalTextList eingetragen und dort lokalisierbar. Wenn Variablenname anzeigen aktiviert ist, wird der Text um den geklammerten Variablenamen ergänzt. <code>Sensor A (PlcProg.iSensor_A)</code> Wenn Beschreibung keinen Text enthält, ist Variablenname anzeigen aktiviert. Der Name wird dann allein stehend und ungeklammert ausgegeben, zum Beispiel <code>PlcProg.iSensor_A</code>
Farbe	Farbe der Variable im Tracediagramm
Linientyp	<i>Darstellung als Liniendiagramm</i> • Linie: Werte zu einer Linie verbunden • Stufe: Werte in Form einer Treppe verbunden • Keine: Werte nicht verbunden
Punkttyp	<i>Darstellung als Punktdiagramm</i> • Punkt: Wert als Punkt • Kreuz: Wert als Kreuz • Keiner: Wert nicht dargestellt
Warnung bei Minimum	 : Bei Unterschreitung der unteren Grenze stellt die Visualisierung den Tracegraph in Warnfarbe dar.
Untergrenze	Wert, zum Beispiel 10
Farbe	Warnfarbe bei Unterschreitung
Obergrenze aktivieren	 : Bei Überschreitung der oberen Grenze stellt die Visualisierung den Tracegraphen in Warnfarbe dar.
Obergrenze	Wert, zum Beispiel 90
Farbe	Warnfarbe bei Überschreitung
Dynamische Anzeigeoptionen	Steuervariable, über die das Verhalten der Visualisierung zur Laufzeit programmiert ist
Variable für Sichtbarkeit	IEC-Variable des Typs <code>BOOL</code> oder <code>BIT</code> , die die Sichtbarkeit der Variablen im Tracediagramm steuert • <code>TRUE</code>: Sichtbar • <code>FALSE</code>: Unsichtbar

Siehe auch

- [Kap. "Dialog 'Anzeigeeinstellungen bearbeiten'" auf Seite 1046](#), Visuobjekt Trace

Dialog 'Erweiterte Trace-Einstellungen'

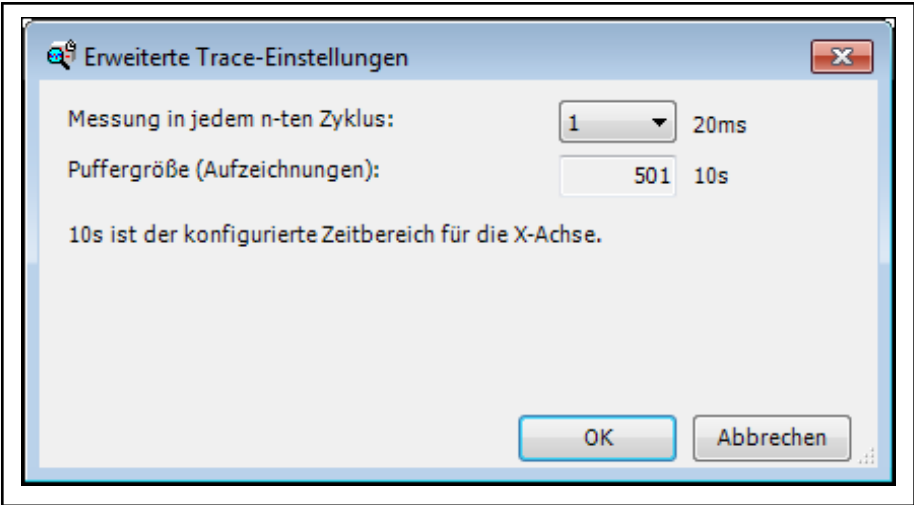


Abb. 20-88: *Erweiterte Trace-Einstellungen*

In diesem Dialog kann die Größe des Trace-Puffers im Laufzeitsystem des Visualisierungselements Trace konfiguriert werden. Die Anzahl der Messpunkte wird aus folgenden Werten berechnet:

- Zykluszeit der Task, in der der Trace läuft
- **Messung in jedem n-ten Zyklus**
- Länge der X-Achse wie es in den Anzeigeeinstellungen konfiguriert ist:
[Kap. "Dialog 'Anzeigeeinstellungen bearbeiten'" auf Seite 1046](#)

Puffergröße (Aufzeichnungen): Hier wird das Ergebnis angezeigt.

Dialog 'Anzeigeeinstellungen bearbeiten'

Funktion: Der Dialog enthält die Konfiguration der Anzeigeeinstellungen des Visualisierungselements **Trace** für die X- und die Y-Achsen zur Laufzeit der Visualisierung und bietet eine Vorschau auf das Tracediagramm

Aufruf: Klick auf die Schaltfläche **Anzeige**

Voraussetzung: Der Dialog **Tracekonfiguration - Aufzeichnungseinstellung** ist geöffnet und aktiv.

Siehe auch

- [Kap. "Dialog 'Tracekonfiguration, Aufzeichnungseinstellungen'" auf Seite 1041](#)

Registerkarte 'X-Achse'

Darstellungsmodus	• Auto : Visualisierung skaliert automatisch • Feste Länge : Visualisierung stellt einen Abschnitt mit konstanter Länge dar. Wenn neue Daten zur Verfügung stehen, läuft das Tracediagramm mit. • Fest : Visualisierung stellt einen Abschnitt von Minimum bis Maximum dar
Minimum	Anfangswert auf der X-Achse Voraussetzung: Fest ist aktiviert.

Maximum	Endwert auf der X-Achse Voraussetzung: Fest ist aktiviert.
Länge	Feste Länge mit automatisch angepasstem Anfangswert Voraussetzung: Feste Länge ist aktiviert
Gitter	 : Tracediagramm mit Rasterlinien in X-Richtung in der ausgewählten Farbe  : Farbauswahlliste des Visualisierungsstils öffnet sich

Feste Abstände	 : Achse skaliert mit Teilstrichen je Abstand und Unterteilung
Abstand	Abstand der Teilstriche, zum Beispiel 20ms
Unterteilungen	Anzahl an Teilstrichen im Abstands, zum Beispiel 5

Tab. 20-9: *Unterteilungsmarkierungen*

Schriftart	Schriftart für die Achsbeschriftung  : Schriftauswahlliste des Visualisierungsstils öffnet sich

Registerkarte 'Y-Achse'

• Auto	 : Visualisierung skaliert automatisch
• Fest:	 : Fester Bereich von Minimum bis Maximum
Minimum	Anfangswert des Bereichs Voraussetzung: Fest ist aktiviert.
Maximum	Endwert des Bereichs Voraussetzung: Fest ist aktiviert.
Gitter	 : Tracediagramm mit Rasterlinien in Y-Richtung in der ausgewählten Farbe  : Farbauswahlliste des Visualisierungsstils öffnet sich
Beschreibung	 : Text mit dem die Y-Achse beschriftet ist, zum Beispiel DC/mA

Tab. 20-10: *Darstellungsmodus*

Feste Abstände	 : Achse skaliert mit Unterteilungsstrichen je Abstand und Unterteilung
Abstand	Abstand der Skalenstriche, zum Beispiel 20ms
Unterteilungen	Anzahl an Unterteilungen des Abstands, zum Beispiel 5

Tab. 20-11: *Unterteilungsmarkierungen*

Schriftart	Schriftart für die Achsbeschriftung  : Schriftauswahlliste des Visualisierungsstils öffnet sich

Aus Visualisierungsstil	Hintergrundfarbe in der Voreinstellung, die im Visualisierungsstil definiert ist
Hintergrund zeichnen	Hintergrundfarbe, die im unteren Eingabefeld gewählt ist
Kein Hintergrund	Tracediagramm ohne Hintergrundfarbe
Hintergrundfarbe des Trenddiagramms	 : Farbauswahlliste des Visualisierungsstils öffnet sich Voraussetzung: Hintergrund zeichnen ist aktiviert

Tab. 20-12: *Hintergrund*

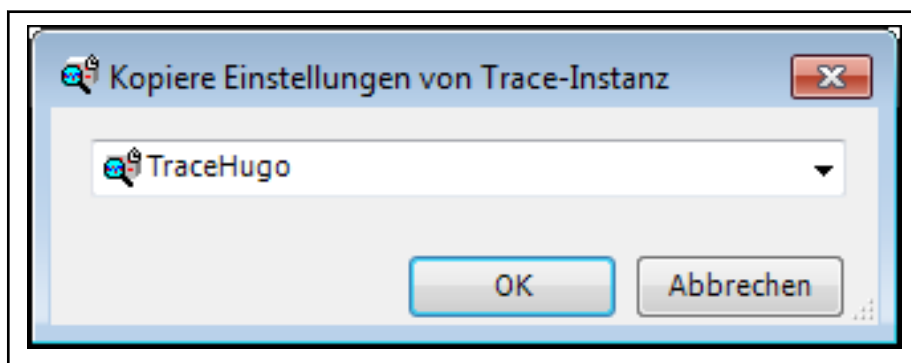
Rücksetzen	Setzt die Einstellungen des Dialogs auf Standardeinstellungen
Als Standard verwenden	Speichert Einstellungen des Dialogs als Standard

Y-Achse hinzufügen	Erweitert das Tracediagramm um eine Y-Achse Folge: Der Dialog Tracekonfiguration enthält unter Variableneinstellungen in Option Zusätzliche Achse eine erweiterte Auswahl an Y-Achsen.
Y-Achse löschen	Löscht die Y-Achse, deren Registerkarte sichtbar ist

Siehe auch

- [Kap. "Dialog 'Tracekonfiguration, Variableneinstellung'" auf Seite 1043](#)

Dialog 'Kopiere Einstellungen von Traceinstanz'

Abb. 20-89: *Kopiere Einstellungen von Trace-Instanz*

Wählen Sie aus der Liste ein Traceobjekt aus, um dessen Konfiguration in den Dialog **Trace-Konfiguration** des Visualisierungselements, das hier editiert wird, zu laden.

Dialog 'Traceassistent'

Dieser Dialog öffnet auf den Befehl **Elemente für Trace-Kontrolle einfügen...**, der sich im Kontextmenü des Visualisierungselements oder im Menü **Visualisierung** befindet. Er ist nur verfügbar, wenn ein Trace-Element selektiert ist.

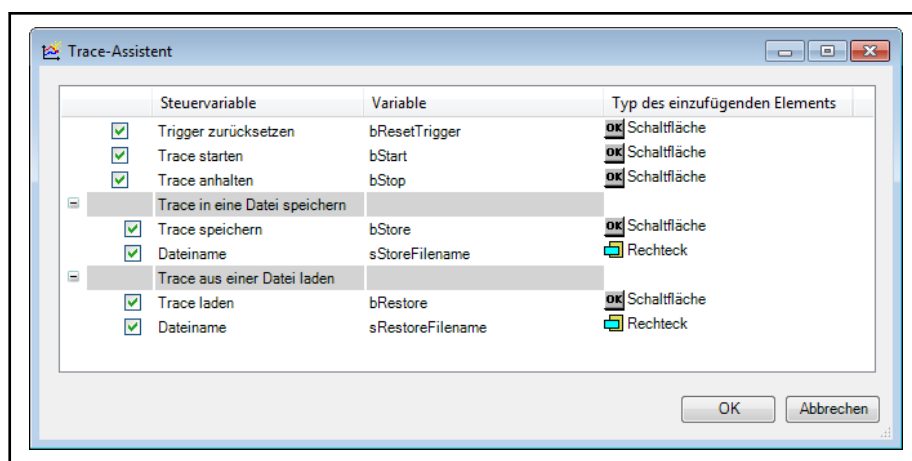


Abb. 20-90: Trace-Assistent

Steuervariable: In dieser Spalte werden die Steuervariablen des Trace-Elements aufgeführt. Diese Liste entspricht den Steuervariablen, die in der Ansicht **Eigenschaften** des Traceeditor verfügbar sind. Markieren Sie das Kontrollkästchen neben der Steuervariable, wenn ein Steuerelement für diese Variable im Visualisierungseditor erzeugt werden soll.

Variable: In dieser Spalte werden die Variablennamen, die den Trace-Steuervariablen zugewiesen sind, aufgeführt. Diese Liste entspricht den Zuweisungen, die in der Ansicht **Eigenschaften** des Traceeditor definiert wurden. Wurde bisher nichts zugewiesen, wird hier eine Vorbelegung mit Standardvariablennamen angeboten. Möchten Sie einen Namen ändern, dann öffnen Sie den Zeileneditor mit einem Doppelklick auf diesen Namen oder mit einem einfachen Klick auf das selektierte Namensfeld.

Typ des einzufügenden Elements: In dieser Spalte werden die Typen des einzufügenden Elements aufgeführt. Ist das Kontrollkästchen markiert, dann wird oder ist das hier aufgeführte Element in der Visualisierung eingefügt. Möchten Sie, dass ein anderer Elementtyp eingefügt wird, dann öffnen sie die Auswahlliste mit Doppelklick auf das Element oder mit einfachem Klick auf das selektierte Element und selektieren das gewünschte. Die Auswahlliste enthält alle Elemente, die für den Zweck geeignet sind.

Befehl 'Tracekonfiguration exportieren'

Funktion: Der Befehl öffnet den Dialog **Tracekonfiguration exportieren**.

Aufruf: Kontextmenü des obersten Knotens in der Baumansicht der Tracekonfiguration.

Voraussetzung: Der Dialog **Tracekonfiguration** ist aktiv und in der Bauman-sicht ist der Name der Tracekonfiguration selektiert, zum Beispiel visu_Trace1.

Dialog 'Tracekonfiguration exportieren'

Der Standarddialog zum Speichern einer Datei ermöglicht das Exportieren der Tracekonfiguration in eine Textdatei. Das Format der Textdatei erfüllt die Anforderungen der Laufzeitsystemkomponente CmpTraceMgr. Deshalb kann die Komponente die Textdatei lesen und anwenden.

Dateiname	Name der Textdatei, die erstellt wird
Dateityp	Tracedatei (*.trace): Format, das die Laufzeitsystemkomponente CmpTraceMgr beim Lesen erwartet

Siehe auch

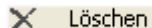
- [Kap. "Dialog 'Tracekonfiguration, Aufzeichnungseinstellungen'" auf Seite 1041](#)

Element 'ActiveX-Element'

Das ActiveX-Element erlaubt es, eine existierende ActiveX-Komponente zu implementieren.



Das ActiveX Control kann innerhalb der Target-Visualisierung unter Windows verwendet werden. Auf anderen Target-Visualisierungs-Plattformen wie WinCE und Linux ist es von dem OEM abhängig, welche Controls unterstützt werden.

Methoden und Parameter können mit den Schaltflächen  Neu erstellen und  Löschen hinzugefügt und gelöscht werden.

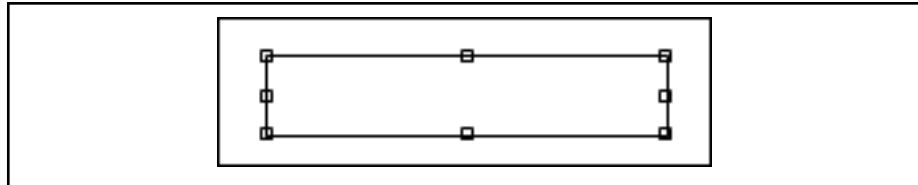


Abb. 20-91: ActiveX-Element

Benutzereingabe beim Editieren eines Elements

Beim Editieren eines Elements im Visualisierungseditor werden folgende Benutzereingaben zur Verfügung gestellt:

- ein Satz von Visualisierungskommandos, die sich wenigstens auf ein Element auswirken: [Kap. 20.2 "Befehle" auf Seite 812](#)
- verschiedene Mausoperationen, am Element anwendbar: [Kap. "Benutzereingabe beim Editieren eines Elements" auf Seite 843](#)

Benutzereingabe beim Editieren der Elementeigenschaften

Eigenschaften definieren die Eigenart eines Visualisierungselements:

- sein Aussehen: Änderungen bei Eigenschaften, die für das Aussehen eines Elements zuständig sind, werden in der Visualisierung sofort sichtbar.
- seine Anordnung in der Visualisierung.
- seine Animation: das Visualisierungselement kann durch direkte Verwendung von IndraLogic-Projektvariablen animiert werden.
- Aktionen, die nach einer Benutzereingabe (Ereignis) des Elements ausgeführt werden.

Eine Eigenschaft kann durch Editieren des Feldes **Wert** geändert werden. Dafür wird dann abhängig vom Typ der Eigenschaft eine Editierfeld, eine Auswahlliste, ein Dialog oder eine zu aktivierende Checkbox geöffnet. Solch ein Widget wird geöffnet

- nach einem Doppelklick,
- nach einem Einzelklick in ein **selektiertes** Feld,
- oder über <Leertaste>, wenn das Feld bereits selektiert war.

Wenn eine Variable zugewiesen wird:

- Geben Sie einfach deren Namen ein.
-  Öffnen Sie damit die Eingabeunterstützung um eine Variable auszuwählen. Kategorie **Variablen** listet alle Variablen auf, die im Projekt bisher definiert wurden.

[Kap. 9.5 "Eingabehilfen nutzen" auf Seite 112.](#)

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	ActiveX-Element

Element	Hier kann eine installierte ActiveX-Komponente zugewiesen werden.  ist zu verwenden, wenn eine Komponente mit Hilfe des Eingabeassistenten ausgewählt werden soll. Es ist ein STRING-Parameter einzugeben, das den Namen oder die ID der Komponenten beschreibt.

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Elementeigenschaft 'Absolute Bewegung' Das Element kann bewegt werden, indem die X- und Y-Position (Pixel) der linken oberen Ecke des Elements durch eine Integer-Variable verändert wird. Hier werden absolute Koordinatenwerte verwendet.

Bewegung - X - Y	X: Hier ist eine Integer-Variable einzugeben, deren Wert die aktuelle X-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in X-Richtung zu verschieben (ein positiver Wert verschiebt das Element von links nach rechts). Y: Eingabe einer Integer-Variablen, die die aktuelle Y-Position der linken oberen Ecke des Elements in Pixel definiert; kann verwendet werden, um das Element in Y-Richtung zu verschieben (ein positiver Wert verschiebt das Element von oben nach unten).

Zustandsvariablen

Property	Beschreibung Wert
Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Modus unsichtbar.

Elementeigenschaft 'Initiale Aufrufe' Methodenaufrufe, die während der Initialisierung ausgeführt werden, können unter dieser Eigenschaft definiert werden. Sie werden ausschließlich im ersten Zyklus ausgeführt.

Methodenaufrufe Methoden	Methoden: Mit Klick auf Neu erstellen wird ein weiterer Eintrag eines Methodenaufrufs erzeugt.

Elementeigenschaft 'Zyklische Aufrufe'

Methodenaufrufe, die in jedem Zyklus ausgeführt werden, können unter dieser Eigenschaft definiert werden. Sie werden im Aktualisierungsintervall der Visualisierung aufgerufen.

Methodenaufrufe Methoden	Methoden: Mit Klick auf Neu erstellen wird eine weiterer Eintrag eines Methodenaufrufs erzeugt.

Elementeigenschaft 'Bedingte Aufrufe'

Zusätzlich kann ein bedingter Aufrufe zugewiesen werden.

Bedingte Methodenaufrufe werden während dem Aktualisierungsintervall der Visualisierung ausgeführt und können hier definiert werden. Anders als bei initialen oder zyklischen Aufrufen wird eine Aufrufbedingung unter der Eigenschaft **Methoden** ► [**<Nummer>**] definiert. Sie werden nur bei steigender Flanke der Aufrufbedingung ausgeführt.

Methodenaufrufe - Methoden [<Nummer>]	Methoden: Mit Klick auf Neu erstellen wird eine weiterer Eintrag eines Methodenaufrufs erzeugt.

Elementeigenschaft 'Initiale/Zyklische/Bedingte Aufrufe: Methoden[<Nummer>]'	Die Nummer indiziert einen Methodenaufruf. Löschen: Löscht den zugehörigen Methodenaufruf mit allen Einstellungen: Parameter , Ergebnisparameter und alles unter Variable
Method	Der Name der aufzurufenden Methode muss hier eingegeben werden.
Parameter - Parameter - Parameter : Variable	Die Variablen der Parameter müssen hier zugewiesen werden. Neu erstellen: Mit jedem Klick wird ein weiterer Parametereintrag Variable erzeugt. Variable: Zuzuweisen ist ein Variablenname für den zugehörigen Parameter.
Ergebnisparameter Variable	Jede Methode hat einen Ergebnisparameter zur Verfügung. Variable: Zuzuweisen ist eine Variable, die das Ergebnis enthalten soll.
Aufrufbedingung	Aufrufbedingung steht nur in Bedingte Aufrufe zur Verfügung. Variable: Zuzuweisen ist eine boolesche Variable. Bei steigender Flanke der Variable wird die zugewiesene Methode ein mal aufgerufen.

Element 'Wartesymbol Blume'

Dieses animierte Element kann verwendet werden, um zu signalisieren, dass das System beschäftigt ist oder auf Daten wartet.



Abb. 20-92:

Elementeigenschaften

Im Folgenden werden alle Elementeigenschaften und deren Beschreibungen aufgelistet.

Elementname

Eigenschaft	Beschreibung Wert
Elementname	Der Elementname kann geändert werden. Standardname ist "GenElemInst_x", "x" steht für eine fortlaufende Nummerierung.

Elementtyp	Wartesymbol Blume

Position Hier ist die Position (X/Y-Koordinaten) und Größe (Breite und Höhe) des Elements jeweils in Pixel zu definieren. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse verläuft nach rechts, die positive Y-Achse verläuft nach unten. Werden die Werte editiert, wird gleichzeitig das angezeigte Element im Editor geändert.

Eigenschaft	Beschreibung Wert
X	Definiert die horizontale Position. X=0 ist der linke Fensterrand. Die positive Achse zeigt nach rechts.
Y	Vertikale Position. Y=0 ist der obere Fensterrand. Die positive Achse zeigt nach unten.
Breite	Horizontale Breite des Elements.
Höhe	Vertikale Höhe des Elements.

Elementeigenschaft 'Farben' Eine Farbe wird mittels Hexzahl definiert, die sich aus Rot/Grün/Blau (RGB) Anteilen zusammensetzt. Für jede dieser drei Farben sind 256 (0-255) Werte verfügbar. Hier sind die Einstellungen **statisch**. Die Farbe kann aus einer Auswahlliste oder dem Farbauswahldialog, der über Button  öffnet, gewählt werden. Beispiel:  0; 128; 64

Rahmenfarbe	Auswahl einer Rahmenfarbe
Füllfarbe	Auswahl einer Füllfarbe

Aussehen

Eigenschaft	Beschreibung Wert
Linienstärke	Stärke der Rahmenlinie in Pixel. 0 kodiert das Gleiche wie 1 und setzt die Linienstärke auf 1 Pixel. Wird kein Rahmen gewünscht, muss man Linienart auf <code>unsichtbar</code> setzen.
Füllart	Auszuwählen ist eines dieser Füllarten angewendet in Füllfarbe - Ausgefüllt: Die Füllfarbe ist sichtbar - Unsichtbar: Die Füllfarbe ist unsichtbar
Linienart	<i>Auszuwählen ist eine der Linienarten angewendet bei der Umrisslinie:</i> - Durchgezogen - Striche - Punkte - Strich Punkt - Strich Punkt Punkt - Unsichtbar: Umrisslinie ist nicht sichtbar.

Symbolfarbe	Auswahl einer Farbe, mit der das Blumensymbol angezeigt werden soll
Symbollinienstärke	Angabe einer Zahl, die die Stärke der Symbollinien in Pixel angibt. 0 kodiert das gleiche wie 1 und setzt die Linienstärke auf 1 Pixel.

Zustandsvariablen

Property	Beschreibung Wert
Unsichtbarkeit	Eingabe einer booleschen Variable; ist der Variablenwert TRUE, dann ist das Element im Online-Modus unsichtbar.

20.4.7 Visualisierungselementeigenschaften

Jedes Element verfügt über Elementeigenschaften, mit denen Sie es konfigurieren können. In **Ansicht ► Elementeigenschaften** sehen Sie, welche Eigenschaften ein Element hat. Die Beschreibung der Elementeigenschaften finden Sie nachstehend oder direkt bei der Beschreibung des Elements.

Eigenschaft 'Diagramm'

Diagramm	 : Wenn in das Feld geklickt wird, erscheint der Dialog Anzeigeeinstellungen bearbeiten .

Eigenschaft 'Elementname'

Elementname	Voreinstellung ist GenElemInst_<x>, x steht für eine fortlaufende Nummerierung

Eigenschaft 'Zahlenformat'

Zahlenformat	Formatierungsangabe für den Variablenwert im Tooltip, zum Beispiel %2.5f

Siehe auch

- [Kap. "Text und Sprache verwenden" auf Seite 845](#)

Eigenschaft 'Ausrichtung'

Ausrichtung	Ausrichtung des Elements, das mit Hilfe des Trendassistenten definiert wurde <i>Optionen</i> • Horizontal • Vertikal

Eigenschaft 'Position'

Die Position des Elements wird in X/Y-Koordinaten in Pixel definiert. Die Größe des Elements wird in Breite und Höhe angegeben. Der Ursprung liegt in der oberen linken Fensterecke. Die positive X-Achse zeigt nach rechts, die positive Y-Achse zeigt nach unten. Die Werte korrespondieren mit den blauen Kästchen in der Visualisierung: .

X	Horizontale Position. $x = 0$ ist der linke Fensterrand. Positive Achse zeigt nach rechts.
Y	Vertikale Position. $y = 0$ ist der obere Fensterrand. Positive Achse zeigt nach unten.
Breite	in Pixel, zum Beispiel 100
Höhe	in Pixel, zum Beispiel 50

Eigenschaft 'Rahmen anzeigen'

Rahmen anzeigen	 : Aktiviert Anzeige des Diagramms mit Rahmen

Eigenschaft 'Tooltip anzeigen'

Die Eigenschaft ist verfügbar, wenn die Eigenschaft **Cursor anzeigen** aktiviert ist.

Tooltip anzeigen	 : Aktiviert Anzeige des Tooltip. Der Tooltip enthält den Namen der IEC-Variablen und den Wert zu dem Zeitpunkt, den der Cursor im Trenddiagramm markiert. Wenn der Cursor fokussiert wird, steht der Tooltip zur Verfügung.

Eigenschaft 'Texteigenschaften'

Die Eigenschaft wirkt auf den Text, der in **Texte ► Text** und **Texte ► Tooltip** konfiguriert ist.

Horizontale Ausrichtung	<i>Horizontale Ausrichtung des Texts im Visualisierungselement</i> • Links • Zentriert: Voreinstellung • Rechts
Vertikale Ausrichtung	<i>Vertikale Ausrichtung des Texts im Visualisierungselement</i> • Oben • Zentriert: Voreinstellung • Unten
Textformat	<i>Textsteuerung im Fall, dass der Text zu lang für das Visualisierungselement ist.</i> • Standard: Der Text wird abgeschnitten und nur der Teil angezeigt, der in das Visualisierungselement passt. • Zeilenumbruch: Der Text wird in Zeilen umgebrochen. • Auslassungszeichen: Der Text wird abgeschnitten und mit Auslassungszeichen . . . ergänzt, um anzuzeigen, dass etwas fehlt.
Schriftart	<i>Schriftart des Texts. Die Auswahlliste ist im Visualisierungsstil definiert.</i> • Standard • Überschrift • Große • Titel • Anmerkung

Farbe Schriftart	Textfarbe, zum Beispiel <code>Grau</code> . Die Auswahlliste ist im Visualisierungsstil definiert.
Transparenz	Abstufungswert der Transparenz., mit dem die Visualisierung den Text darstellt 0: volltransparent und somit scheinbar verschwunden 255: deckend, Voreinstellung Voraussetzung: Die Visualisierungsvariante ist für Semitransparenz konfiguriert.

Eigenschaft 'Texteigenschaften'

Schriftart	<i>Schriftart des Texts. Die Auswahlliste ist im Visualisierungsstil definiert.</i> • Standard • Überschrift • Große • Titel • Anmerkung
Farbe Schriftart	Textfarbe, zum Beispiel <code>Grau</code> . Die Auswahlliste ist im Visualisierungsstil definiert.
Transparenz	Abstufungswert der Transparenz., mit dem die Visualisierung den Text darstellt 0: volltransparent und somit scheinbar verschwunden 255: deckend, Voreinstellung Voraussetzung: Die Visualisierungsvariante ist für Semitransparenz konfiguriert.

Eigenschaft 'Texte'

Text	Text, der links bzw. oberhalb der Schaltflächenleiste ausgegeben wird. Wenn das Element horizontal ausgerichtet ist, ist der Text links positioniert. Wenn das Element vertikal ausgerichtet ist, ist der Text oberhalb positioniert. Voreinstellung: <code>Zoom</code>

Eigenschaft 'Beschriftung der Skalenstriche'

Zweizeilige Beschriftung	<i>Optionen</i> : Ausgabe des Datums in der oberen Zeile und der Uhrzeit in der unteren : Ausgabe von Datum und Uhrzeit in einer Zeile
Unwichtige Beschriftung im Zeitstempel weglassen	: Verkürzte Darstellung des Zeitstempels. Beim ersten Skalenstrich wird das Datum ausgegeben, bei den folgenden Skalenstrichen steht nur noch die Uhrzeit. : Vollständige Ausgabe der Zeit. Die Eigenschaft Internationalisierung steht zur Verfügung.
<i>Internationalisierung (Format-Strings)</i> - Datum - Zeit	<i>Einträge</i> Datum: Definiert Datumsformat. Voreinstellung ist das Format, das in Windows eingestellt ist. "dd.MM.yyyy" ist die im deutschsprachigen Raum gebräuchliche Reihenfolge. Zeit: Definiert Zeitformat für die Uhrzeit. Voreinstellung ist das Format, das in Windows eingestellt ist. "HH:mm:ss.ms" ist die im deutschsprachigen Raum gebräuchliche Reihenfolge.

Eigenschaft 'Zeiten'

Unter **Zeiten** werden die Schaltflächen in einem Array definiert und konfiguriert, die das Element Zeitauswahl zur Laufzeit zur Verfügung stellt. Damit kann ein Benutzer ein bestimmtes Zeitintervall auswählen. Dann wird das Diagramm, das mit der Zeitauswahl gesteuert wird, auf das konfigurierte Zeitintervall umgeschaltet. Dabei wird der Endzeitpunkt der bisherigen Darstellung belassen und der Anfangszeitpunkt angepasst.

"Alles"-Auswahl zur Verfügung stellen	 : Erweitert Zeitauswahlleiste um Alles-Schaltfläche. Das Diagramm stellt ein Zeitintervall dar, das alle Zeitstempel umfasst.
Zeiten	 : Wenn auf die Schaltfläche geklickt wird, wird eine weitere Schaltfläche in die Zeitauswahlleiste eingefügt und das Array um einen Eintrag erweitert. Unter der Eigenschaft Zeiten ► Zeiten ► Zeiten ► [<neu>] ist ein weiterer Index vorhanden. Unter diesem Index befindet sich Zeit . Dort ist die Konfiguration der Schaltfläche einzutragen.
Zeiten [Index] mit Index aus {0, 1, 2,...}	Array von allen Schaltflächen der Zeitauswahlleiste. Index entspricht der Anzahl an Schaltflächen. : Wenn auf Löschen geklickt wird, wird die zugehörige Schaltfläche aus der Zeitauswahlleiste entfernt. Der Konfigurationseintrag wird aus der Eigenschaftsliste Zeiten entfernt.
[Index] Zeit	 : Zeitintervall in normierter Schreibweise. Zum Beispiel: 3M für 3 Monate, 30m für 30 Minuten. Wenn im Feld ein Zeitintervall angegeben ist, wird die Schaltfläche damit beschriftet. Wenn ein Benutzer zur Laufzeit die Schaltfläche klickt, wird der Befehl zum Umschalten des Diagramm auf dieses Zeitintervall ausgeführt. Die Voreinstellung ist leer.

20.5 Visualisierungsmanager mit Clients

20.5.1 Visualisierungsmanager

Wird eine Visualisierung im Projektextplorer, zum Beispiel unterhalb einer Applikation, eingefügt, dann wird ein Visualisierungsmanagerobjekt  automatisch hinzugefügt.

Das Objekt verwaltet allgemeine Einstellungen aller Visualisierungen, die der aktuellen Applikation durch Clients zugewiesen sind.

Mit Doppelklick auf den Eintrag im Projektextplorer kann das Objekt des Visualisierungsmanagers geöffnet werden.

Der Editor öffnet in einem Fenster betitelt mit 'VisualizationManager <Applikationspfad>', das verschiedene Registerkarten für **Einstellungen**, **Standard Tastaturkonfiguration**, **Visualisierungen** und **Schriftarteneinstellung** enthält.

Nach Änderungen werden die Einstellungen sofort angewendet.

'Visualisierungsmanager' - 'Einstellungen'

Symbol: 

Funktion: Die Registerkarte enthält Einstellungen, die applikationsweit für alle Visualisierungen gültig sind.

Aufruf: Objekt des Typs **Visualisierungsmanager**

Voraussetzung: Applikation enthält einen **Visualisierungsmanager**

'Allgemeine Einstellungen'

Unicode Zeichenfolgen verwenden	 Visualisierung kodiert Zeichenfolgen als Unicode
CurrentVisu Variable verwenden	 Die Applikation kennt und verwendet die globale Variable <code>VisuElems.CurrentVisu</code> des Typs <code>STRING</code> und enthält zur Laufzeit der Applikation den Namen der gerade aktiven Visualisierung. Die Applikation kann auf die Variable lesend zugreifen, um den Namen der gerade aktiven Visualisierung zu erhalten und schreibend, um einen Visualisierungswechsel hervorzurufen. Voraussetzung: Applikation enthält eine Visualisierung, die weitere Visualisierungen aufruft <i>Beispiel</i> • Zuweisung Variable: <code>VisuElems.CurrentVisu := strVisuName;</code> • Zuweisung Visualisierungsname: <code>VisuElems.CurrentVisu := 'visu1';</code>

Siehe auch

- [Kap. "Text und Sprache verwenden" auf Seite 845](#)

'Stilkonfiguration'

Ausgewählter Stil	• Stil aus dem Visualisierungsstile-Repository, den jede Visualisierung der Applikation anwendet, zum Beispiel Flat Style. • <Kein>: Visualisierung stellt ihre Elemente stillos oder gemäß interner Voreinstellung dar. In den Elementeigenschaften Farbe und Schriftart ermöglicht ein Standarddialog statt einer Auswahlliste eine Wertezuweisung.
	Auswahlliste mit Stilen, die im Visualisierungsstil-Repository installiert sind, erscheint.

Alle Versionen anzeigen (nur für Experten)	 : Die Auswahlliste enthält den ausgewählten Stil und alle weiteren Stile aber nur der neuesten Version. Wenn für den ausgewählten Stil eine neuere Version installiert ist, wird diese auch angezeigt.  : Die Auswahlliste enthält alle installierten Stile mit allen Versionen.
Vorschau	Die Elemente, die in der Vorschau dargestellt sind, repräsentieren den in Ausgewählter Stil angegebenen Stil.

Siehe auch

- [Kap. 20.8.2 "Dialog 'Optionen- 'Visualisierungsstile'" auf Seite 1079](#)

'Spracheinstellung'

Ausgewählte Sprache	Sprache, die die Darstellungsvariante bei Start einer Visualisierung verwendet
'Einstellungen für Standardtexteingabe'	Für ein Element mit Standardtexteingabe erscheint zur Laufzeit ein Dialog, der die Eingabe unterstützt. Sie können bestimmen, welcher Dialog erscheint.
Numpad	Dialog, den die Visualisierung aufruft, wenn zur Laufzeit ein Anwender das Eingabefeld für eine Zahl aktiviert. Der Dialog stellt einen Ziffernblock dar. Voreinstellung: VisuDialogs.Numpad
Keypad	Dialog, den die Visualisierung aufruft, wenn zur Laufzeit ein Anwender das Eingabefeld für einen Text aktiviert. Der Dialog stellt eine Tastatur dar. Voreinstellung: VisuDialogs.Keypad

'Einstellungen für Benutzerverwaltungsdialoge'

Benutzerverwaltungsdialog	Sie können Ihre Visualisierung mit einer Benutzerverwaltung konfigurieren, indem Sie eine Eingabe konfigurieren, die bewirkt, dass ein Benutzerverwaltungsdialog erscheint. Die Bibliothek <code>VisuUserManagement</code> enthält dafür fertige Visualisierungen. Sie befindet sich in Ihrem Installationsverzeichnis, zum Beispiel unter <code>C:\Program Files (x86)\3S CODESYS\CODESYS\Projects\Visu\Dialogs\VisuUserMgmtDialogs.library</code>. Sie können aber auch andere Visualisierungen als Benutzerverwaltungsdialog verwenden, indem Sie die Voreinstellungen hier ändern.
Dialog Einloggen	Benutzerverwaltungsdialog, der ein Einloggen ermöglicht, typischerweise eine Eingabeaufforderung für einen Benutzernamen und ein Passwort. Er erscheint auf ein Ereignis, das als Folgeaktion Benutzerverwaltung, Aktion Anmelden ausführt. Voreinstellung: <code>VisuUserManagement.VUM_Login</code>
Dialog Passwortänderung	Benutzerverwaltungsdialog, der eine Passwortänderung ermöglicht, typischerweise eine Eingabeaufforderung für das aktuelle und das neue Passwort. Er erscheint auf ein Ereignis, das als Folgeaktion Benutzerverwaltung, Aktion Benutzerpasswort ändern ausführt. Voreinstellung: <code>VisuUserManagement.ChangePassword</code>
Dialog Konfigurationsänderung	Benutzerverwaltungsdialog, der eine Konfigurationsänderung der Benutzerverwaltung ermöglicht, also typischerweise eine Anzeige der aktuellen Benutzerkonfiguration und eine Möglichkeit, diese zu ändern. Er erscheint auf ein Ereignis, das als Folgeaktion Benutzerverwaltung, Aktion Benutzerverwaltung öffnen ausführt. Voreinstellung: <code>VisuUserManagement.VUM_UserManagement</code>

'Zusätzliche Einstellungen'

Multitouch aktivieren	 Zur Laufzeit erwartet die Visualisierung Benutzereingaben über Gesten und Touch-Ereignisse <i>Betroffene Elemente</i> • Elemente mit Eingabekonfiguration • Element des Typs Frame • Element des Typs Registersteuerelement
Semitransparentes Zeichnen aktivieren	 Visualisierung zeichnet die Elemente semi-transparent. IndraLogic stellt für Farben die Eigenschaft Transparenz zur Verfügung. Voreinstellung: Aktiviert. Voraussetzung: Sie erstellen eine Visualisierung neu und die Darstellungsvarianten können semitransparent zeichnen

Siehe auch

- [Kap. "Semitransparente Farbe verwenden" auf Seite 863](#)

'Visualisierungsmanager' - 'Standardtastaturkonfiguration'

Die Registerkarte enthält die Tastaturkonfiguration, die für alle der Applikation zugehörigen Visualisierungen gilt.

Das heißt, die 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 wie [Kap. 20.3.4 "Registerkarte 'Tastaturkonfiguration'" auf Seite 839](#) zu bedienen.

Zusätzlich gelten, immer Geräte unabhängig, einige spezielle Standardtastaturkürzel für das Navigieren in einer Visualisierung: [Kap. 20.11 "Tastaturbedienung im Online-Modus" auf Seite 1081](#)

'Visualisierungsmanager' - 'Visualisierungen'

Die Registerkarte des Visualisierungsmanagers zeigt die gerade verfügbaren Visualisierungen an.

Standardmäßig werden die Visualisierungen, die unter der aktuellen Applikation hängen und die, die von diesen Visualisierungen referenziert werden, automatisch auf das jeweilige Zielsystem geladen.

Referenzierte Visualisierungen sind entweder global definiert (Ordner "Allgemeiner Baustein") oder kommen aus Bibliotheken.



Diese Funktionalität ersetzt die vorher mögliche Verwendung von "Visualisierungsreferenz"-Objekten. Diese Objekten können jetzt nicht mehr neu eingefügt werden; bereits eingefügte funktionieren allerdings weiterhin.

Registerkarte 'Schriftarteneinstellung'

Die Registerkarte bietet Einstellungen um sprachabhängig Schriftart und Größe eines Texts anzupassen. Angewendet werden die Einstellungen auf die Texte der Visualisierungen, die unter der selben Applikation liegen wie der Visualisierungsmanager.

Sprache	Für jede Sprache, die im Projekt verwendet wird, ist eine Spalte angelegt. Dafür werden alle Textlisten, auch die aus eingebundenen Bibliotheken durchsucht.
Schriftart	Schriftart, die die Visualisierung abhängig von der Sprache verwendet .

Größenfaktor	Der Faktor wirkt auf den Schriftgrad aller Texte in der Visualisierung. Wenn der Faktor kleiner 1 ist, führt das zu einer Verkleinerung des Schriftgrads. Wenn der Faktor 1 ist, werden alle Texte unverändert, wie in Eigenschaften definiert, dargestellt. Voreinstellung: 1
Kontextmenü, das aktiviert werden kann, wenn eine Zelle selektiert ist	• Sprache löschen: Die zugehörige Spalte wird entfernt. Das ist vor allem dann ratsam, wenn Einstellungen der Spalte rot hervorgehoben sind. • Spracheinstellung kopieren: Alle Einstellungen der Spalte werden in die Zwischenablage übernommen • Spracheinstellung einfügen: Alle Einstellungen der Spalte werden mit den Werten aus der Zwischenablage überschrieben
Rote Hervorhebung einer Zelle	Die Einstellung kann zur Laufzeit nicht angewendet werden. Die Sprache ist in den Textlisten des Projekt oder der Bibliotheken nicht mehr vorhanden

Tab. 20-13: **Sprachspezifische Schriftarten-Einstellungen**

Siehe auch

- [Kap. "Text und Sprache verwenden" auf Seite 845](#)

20.5.2 Die Clients und ihre Editoren

Diagnosevisualisierung

Inhalt dieser Seite:

- ["Einschränkungen bei Ausdrücken und Monitoring" auf Seite 1062](#)
- ["Einschränkungen bei Eingabeereignissen" auf Seite 1062](#)
- ["Visualisierungsschnittstelle" auf Seite 1063](#)
- ["Daten-Server" auf Seite 1063](#)

Diagnosevisualisierung ist ein Visualisierungsclient, der auf dem Programmiersystem läuft:

Für Diagnosezwecke kann es wünschenswert sein, eine oder mehrere Visualisierungen, die zu einer Applikation gehören, nur innerhalb des Programmiersystems laufen zu lassen, ohne den Visualisierungscode auf die Steuerung laden zu müssen.

Die Diagnosevisualisierung ist ab V3.3.2.0 verfügbar und wird automatisch verwendet, wenn kein Client-Objekt wie **Target-Visualisierung** oder **Web-Visualisierung** im Projektextplorer unterhalb des Visualisierungsmanagers eingehängt ist.

Bei einer Diagnosevisualisierung wird **kein** Visualisierungscode erzeugt und auf die Steuerung geladen.

Wenn Sie nicht die in einem Alt-Projekt enthaltenen **Target-Visualisierungs-** und/oder **Web-Visualisierungs-**Objekte unterhalb des Visualisierungsmanagers entfernen wollen, um den Diagnosemodus der Visualisierung zu erhalten, können Sie die Objekte alternativ vom Übersetzen ausschließen.

Nutzen Sie dazu die Einstellung **Vom Übersetzen ausschließen** im jeweiligen Eigenschaftendialog des Objektes ([Kap. 20.6 "Dialog 'Eigenschaften- 'Visualisierungsobjekt'" auf Seite 1063](#)).



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.



Im Unterschied zur Target-Visualisierung verhalten sich in der Diagnose-Visualisierung VAR_INPUT-Variablen wie VAR_IN_OUT.

Einschränkungen bei Ausdrücken und Monitoring

Die Diagnosevisualisierung unterstützt nur Ausdrücke, die vom Monitoring-Mechanismus des Programmiersystems verwendet werden können.

Diese sind:

- normale Variablenzugriffe wie `PlcProg.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 Standardtextfunktionen, alle Typkonvertierungsfunktionen wie INT_TO_DWORD, alle Operatoren wie SEL, MIN, ...

Einschränkungen bei Eingabeereignissen

Innerhalb des Eingabeereignisses **ST-Code ausführen** wird nur eine Liste von Zuweisungen unterstützt.

Beispiel

```
PlcProg.n := 20 * PlcProg.m;
// not allowed
IF PlcProg.n < MAX_COUNT THEN
  PlcProg.n := PlcProg.n + 1;
END_IF
//use the following statement instead:
PlcProg.n := MIN(MAX_COUNT, PlcProg.n + 1);
```



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.

Daten-Server

Die Diagnosevisualisierung ist nur dafür gedacht, die aktuelle Applikation passiv zu betrachten. Aus diesem Grund werden Werte von Variablen, die über den Daten-Server aus einer anderen Datenquelle geliefert werden, nicht automatisch aktualisiert.

Für diese Variablen werden nur die Initialwerte oder die zuletzt übertragenen Daten dargestellt.



Werden die Werte dieser Variablen in der eigenen Applikation bearbeitet, ist die Anzeige natürlich möglich.

20.6 Dialog 'Eigenschaften' - 'Visualisierungsobjekt'

Symbol:

Funktion: Der Dialog enthält die 'Eigenschaften' eines Visualisierungsobjekts

Aufruf: Menü **Ansicht** ► **Eigenschaften** bei selektiertem Visualisierungsobjekt oder Kontextmenü des Objekts im Projektextplorer

Für das aktuell angewählte Visualisierungsobjekt werden folgende Eigenschaften angezeigt und können geändert werden (Dialog 'Eigenschaften', Kategorie 'Visualisierung').

Visualisierung verwenden als:

- **Visualisierung:** Die Visualisierung wird als standardmäßiges Visualisierungsobjekt verwendet.
- **Dialog:** Die Visualisierung stellt einen Dialog dar und ist insofern automatisch in anderen Visualisierungen verfügbar, als dass sie in **Eingabekonfiguration** ► **Dialog öffnen** des Visualisierungselements zur Auswahl angeboten wird.

Siehe auch:

- [Kap. "Eingabedialoge" auf Seite 1064](#)
- [Kap. "Programmatischer Zugriff auf Eingabedialoge" auf Seite 1070](#)

- **Nummernfeld/ Tastatur/ Dialog für Eingabekonfiguration:** Die Visualisierung stellt ein Nummernfeld oder eine Tastatur dar und ist insofern automatisch in anderen Visualisierungen verfügbar, als dass sie in der Eingabekonfiguration in Kategorie **Variable schreiben** (Bearbeitungstyp) zur Auswahl steht ([Kap. "Eingabekonfiguration" auf Seite 849](#)).

Man beachte, dass die **Schnittstelle** einer solchen Nummernfeld/Tastatur-Visualisierung genauso aussehen muss, wie die des IndraLogic-Standardnummernfelds/-tastatur, das durch die **VisuDialogs.library** bereit gestellt wird.

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 einbinden** jedoch deaktiviert ist, wird das Hintergrundbild - falls es den Bereich, der alle Visualisierungselemente einschließt, überragt - abgeschnitten.

Angegebene Visualisierungsgröße verwenden: Die Visualisierungsgröße wird durch die angezeigten Werte definiert (Breite, Höhe), wobei nicht berücksichtigt wird, ob alle Visualisierungselemente und das Hintergrundbild in diesen Bereich passen, d.h. vollständig sichtbar sind.

Visualisierungsgröße: Aktuelle **Breite** und **Höhe** des Visualisierungsobjekts (Anzahl Pixel)

Intern: Wenn das Projekt als Bibliothek gespeichert wird und diese Option aktiviert ist, wird die Visualisierung nur innerhalb des Bibliotheksprojekts in der Auswahlliste in einem Frame oder für einen Visu-Wechsel angeboten.

Wenn die Bibliothek in einem Projekt eingebunden ist, ist die Visualisierung dagegen nicht mehr verfügbar.

Auch im Visualisierungsmanager auf Registerblatt "Visualisierungen" wird sie dann nicht angezeigt. Solche Visualisierungen dienen ausschließlich als interne Bausteine der Gesamtvisualisierung einer Bibliotheksfunktion.

20.6.1 Anwendungsfälle

Eingabedialoge

Inhalt dieser Seite:

- ["Beispiel für das Erstellen und Verwenden von eigenen Dialogvisualisierungen" auf Seite 1064](#)

Eine Visualisierung kann zum Zweck der späteren Verwendung als Eingabedialog entsprechend erstellt und in ihren Eigenschaften als **Dialog** definiert werden ([Kap. 20.6 "Dialog 'Eigenschaften- Visualisierungsobjekt'" auf Seite 1063](#)). Dialogvisualisierungen können dann in anderen Visualisierungen verwendet werden, um dem Benutzer eine Eingabemaske bereitzustellen. Dialogvisualisierungen werden beispielsweise bei der Konfiguration einer **OpenDialog**, **CloseDialog** oder **OnDialogClosed** der Eigenschaft **Eingabekonfiguration** eines Visualisierungselements angeboten.

VAR_INPUT- und VAR_IN_OUT-Parameter der Dialogvisualisierung müssen im Schnittstelleneditor definiert werden und müssen entsprechend von einem anderen Visualisierungselement bedient werden.

Dieses andere Visualisierungselement ist für das Öffnen des Dialogs konfigurieren ("Dialog öffnen"-Konfiguration) ([Kap. "Grundlegendes zum Schnittstelleneditor" auf Seite 831](#)).

Das heißt, die Parameter können in der Konfiguration eines anderen Elements verwendet werden.



Die VAR_INPUT-Parameter müssen für die Diagnosevisualisierung, also für die im Programmiersystem laufende Visualisierung, skalaren Typs sein ([Kap. "Diagnosevisualisierung" auf Seite 1061](#)).

Neben selbst erstellten Dialogvisualisierungen können **Standarddialogvisualisierungen** verwendet werden, wenn `VisuDialogs` im Projekt eingebunden wird. Aktuell sind dies Visualisierungen eines Login-Dialogs und eines Dialogs zum Öffnen und Speichern einer Datei (`VisuDialogs.Login`, `VisuDialogs.FileOpenSave`).

Beispiel für das Erstellen und Verwenden von eigenen Dialogvisualisierungen

Bei einem Mausklick auf eine Schaltfläche soll sich ein Dialog öffnen, in dem Zugangsdaten, in diesem Fall ein Benutzerlevel und ein Passwort, geändert

werden können. Dazu erstellen wir eine Visualisierung namens `ChangeUserLevel`, die diesen Dialog darstellt, und eine weitere namens `Visualisierung`, die die Schaltfläche darstellt.

1. Erstellen der Dialog-Visualisierung "ChangeUserLevel":

Erstellen Sie ein Visualisierungsobjekt namens `ChangeUserLevel`, das aussieht wie in der Abbildung unten gezeigt. In seinen Objekteigenschaften definieren Sie, dass es als `Dialog` verwendet werden soll. Im aktuellen Beispiel handelt es sich um einen Dialog zum Eingeben eines Benutzerlevels und eines Passworts, der nur über die OK-Schaltfläche geschlossen werden kann, wenn der angegebene Level und das Passwort zusammenpassen.

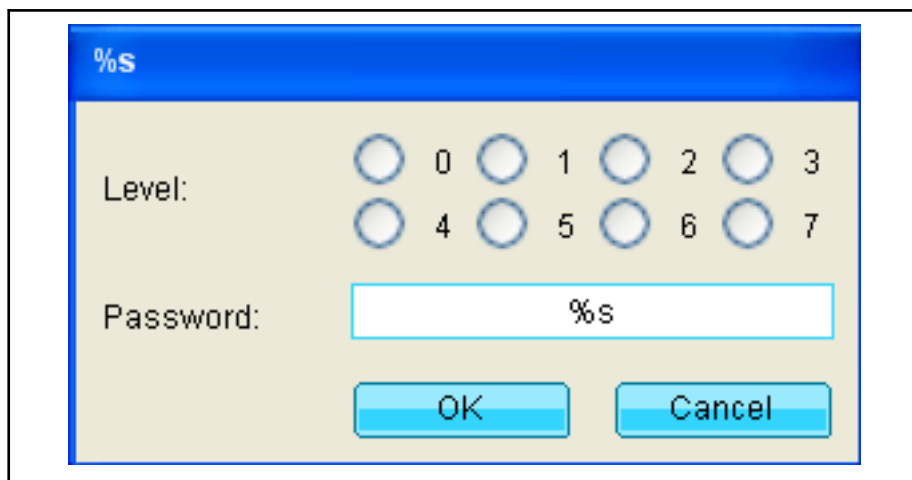


Abb. 20-93: *ChangeUserLevel Dialogvisualisierung*

Der Dialog setzt sich aus folgenden Elementen zusammen:

Element	Elementeigenschaften
Bild	Statische ID: <code>Login</code> ; Bild eines leeren Dialogs mit grauem Hintergrund und einer leeren blauen Titelleiste. Die Bilddatei kann über eine Bildersammlung bereitgestellt werden, wo sie mit ID "Login" abgelegt ist.
Rechteck (für die Anzeige des Dialognamens in der Titelleiste)	Texte: Text: <code>%s</code> ; wird dann ersetzt mit dem Wert des Parameters <code>stTitle</code> , siehe im folgenden; Textvariablen: Text Variable: <code>stTitle</code> ; diese Variable ist als Eingabeparameter in der Schnittstelle der Visualisierung definiert und ihr Wert kann im Konfigurationsdialog des Visualisierungselements angegeben werden, das für das Öffnen des Dialogs verwendet wird (im vorliegenden Beispiel ist dies die "Change User Level" Schaltfläche, siehe unten) .
Radiobutton (für die Eingabe des Benutzer-Levels)	Anzahl Spalten: 4 Anordnung der Radiobuttons: Von links nach rechts Radiobuttoneinstellungen: Definieren Sie die 8 Bereiche [0] to [7] (jeweils: Text: <code><n></code> , wobei n 0 - 7 ist) Variable: <code>iLevel</code> (Ein/Ausgang des Elements)
Textfeld (für die Passwort-Eingabe)	Texte: Text: <code>%s</code> Textvariablen: Text Variable: <code>stPwd</code> (Ein/Ausgang des Elements) Eingabekonfiguration: OnMouseDown: Variable schreiben: Bearbeitungstyp: Edit , Textausgabevervariable verwenden
Text ("Level")	Texte: Text: <code>Level:</code>
Text ("Password")	Texte: Text: <code>Password:</code>

Element	Elementeigenschaften
Schaltfläche (OK; Die Farbe zeigt an, ob Level und Passwort zusammenpassen; Ein Mausklick auf die Schaltfläche ist nur effektiv, wenn dies der Fall ist und schließt den Dialog)	Texte: Text: OK Farben: Farbe und Alarmfarbe müssen unterschiedlich sein! Farbvariablen: Farbe umschalten: <code>stPwd <> MUX(iLevel, stLevel0, stLevel1, stLevel2, stLevel3, stLevel4, stLevel5, stLevel6, stLevel7);</code> (wenn Passwort und Level nicht zusammenpassen, ändert sich die Farbe nach Grau) Zustandsvariablen: Eingaben deaktivieren: <code>stPwd <> MUX(iLevel, stLevel0, stLevel1, stLevel2, stLevel3, stLevel4, stLevel5, stLevel6, stLevel7);</code> (wenn Passwort und Level nicht zusammenpassen, bleibt eine Eingabe auf die Schaltfläche ohne Wirkung) Eingabekongfiguration: OnMouseDown: Dialog schließen: <code>ChangeUserLevel</code>, Ergebnis: OK; (zu beachten: die Eingaben auf den Dialog werden nicht auf die jeweiligen Variablen geschrieben, wenn die OK-Schaltfläche betätigt wird)
Schaltfläche, Cancel	Farben: Farbe = Farbe der OK-Schaltfläche Texte: Text: Cancel Eingabekongfiguration: OnMouseDown: Dialog schließen: <code>ChangeUserLevel</code>, Ergebnis: Cancel

Öffnen Sie den Schnittstelleneditor der Visualisierung und deklarieren folgende Variablen (die Werte der Eingangsvariablen können später in der "Dialog öffnen"-Eingabekongfiguration definiert werden, siehe **Open Dialog** Eingabekongfiguration).

```

VAR_INPUT
    stTitle: STRING;
    stLevel0: STRING;
    stLevel1: STRING;
    stLevel2: STRING;
    stLevel3: STRING;
    stLevel4: STRING;
    stLevel5: STRING;
    stLevel6: STRING;
    stLevel7: STRING;
END_VAR
VAR_IN_OUT
    iLevel: INT;
    stPwd: STRING;
END_VAR

```

Beispiel:

2. Erstellen einer Visualisierung "Visualization"

Nun wird eine Visualisierung `Visualization` mit einer Schaltfläche zum Öffnen des oben angelegten "ChangeUserLevel"-Dialogs und einer Anzeige des aktuell eingestellten Levels erstellt:

Im vorliegenden Beispiel werden die folgenden Elemente verwendet:

Element	Elementeigenschaften
Schaltfläche (zum Öffnen des "Change User Level" Dialogs)	Texte ► Text: <code>Change User Level</code> Eingabekongfiguration: OnMouseDown: Dialog öffnen: siehe Abbildung unten
Textfeld	Texte ► Text: <code>%s</code> (Dialogüberschrift); Textvariablen ► Text-Variable: <code>PlcProg.iLevel</code> (Zuweisung der Level-Eingabe zur Programmvariablen)

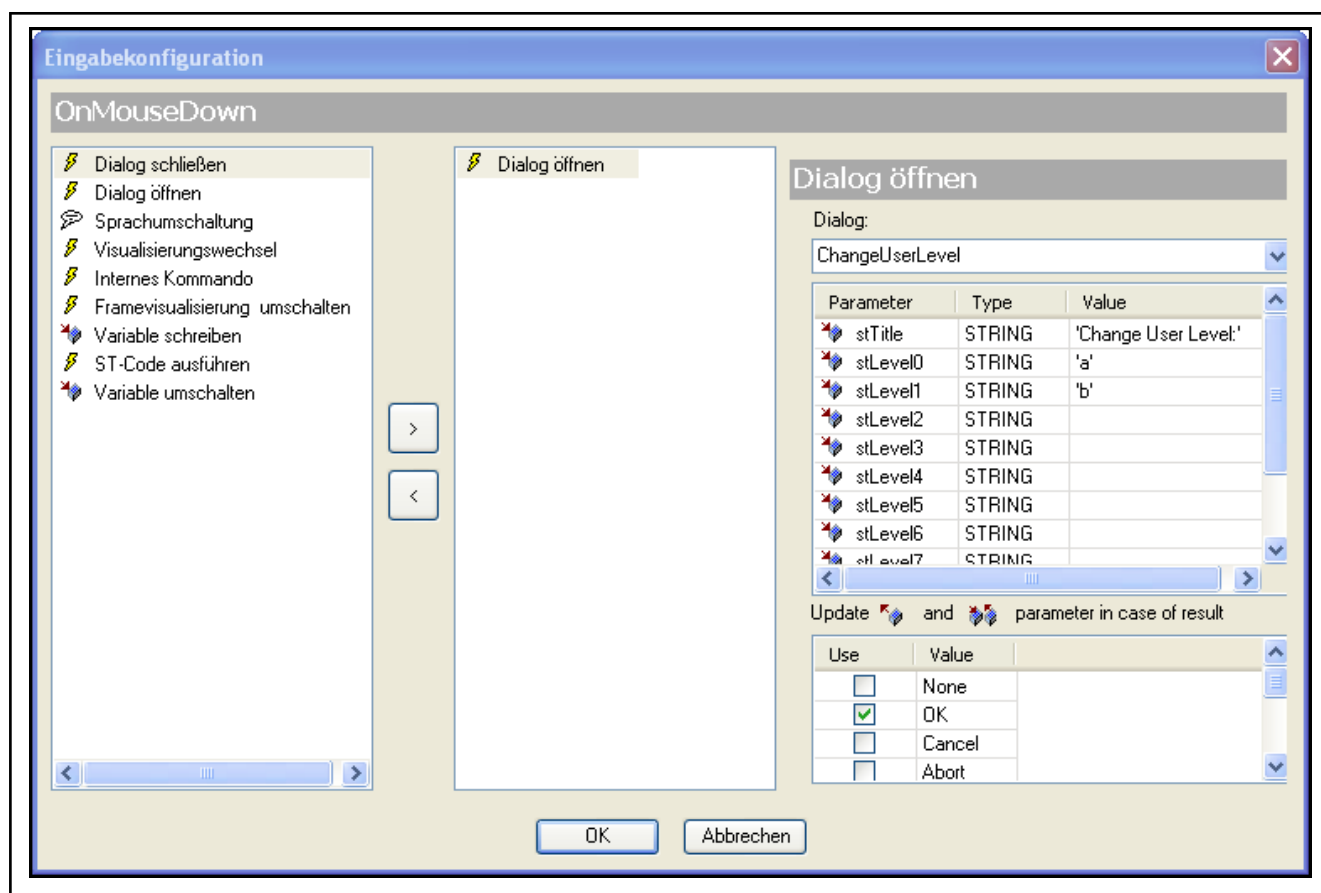


Abb. 20-94: Eingabekonfiguration der Schaltfläche, die die selbst-definierte Dialog-Visualisierung ChangeUserLevel öffnen soll

Hier muss der Dialog angegeben werden, der mit einem Mausklick auf die Schaltfläche geöffnet werden soll. Dazu wählen Sie "ChangeUserLevel" aus der Auswahlliste unterhalb von **Dialog**. Die Eingangs- (VAR_INPUT) und Ein-/Ausgangsparameter (VAR_IN_OUT) der Dialog-Visualisierung (Schnittstelle von "ChangeUserLevel") werden aufgelistet und Sie können definieren, welche Werte durch den ChangeUserLevel-Dialog immer beim Öffnen gelesen werden sollen. Beachten Sie, dass für das aktuelle Beispiel - wie in der Abbildung oben zu sehen - nur zwei Level-Passwort-Kombinationen konfiguriert sind. Wenn der Anwender Level 0 eingibt, muss er "a" als Passwort eingeben, um die OK-Schaltfläche aktiviert zu bekommen. Wenn er Level 1 wählt, muss er "b" als Passwort verwenden. Wird einer der anderen Levels ausgewählt, muss momentan kein Passwort eingegeben werden.

Definieren Sie weiterhin, bei welcher Aktion im ChangeUserLevel-Dialog dieser Dialog geschlossen und seine Ausgänge und Ein-/Ausgänge geschrieben werden sollen. In unserem Beispiel setzen Sie einen Haken für die **Verwendung** von **OK**. So wird später bei einem Mausklick auf die OK-Schaltfläche der ChangeUserLevel-Dialog geschlossen und die Variablen `iLevel` und `stPwd` (VAR_IN_OUT) werden mit den Werten aus der Benutzereingabe geschrieben.

Siehe für weitere Informationen zum "Dialog öffnen"-Konfigurationsdialog: ["Dialog öffnen" auf Seite 0](#).

*Beispiel:***3. Online-Betrieb von "Visualization"**

Wenn Sie nun die Visualisierung "Visualization" im Online-Modus laden und starten (möglich als Target-Visualisierung, Web-Visualisierung oder "Diagnose-Visualisierung"), dann kann der ChangeUserLevel-Dialog mit einem Mausklick auf die bezeichnete Schaltfläche in "Visualization" geöffnet werden:

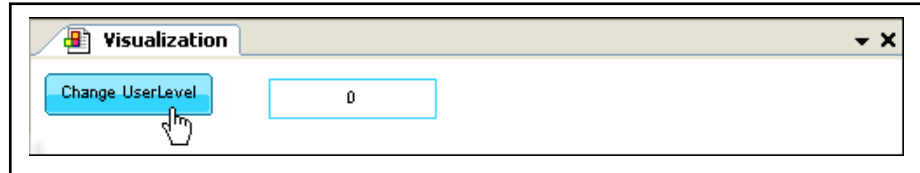


Abb. 20-95: Starte "Visualization", klicke auf Schaltfläche "Change UserLevel"

Der Dialog öffnet sich. Wählen Sie einen Level, z.B. "1" (wird in `iLevel` geschrieben) und geben Sie das passende Passwort "b" ein (wird in `stPwd` geschrieben). Diese Werte werden von der OK-Schaltfläche verglichen; solange sie nicht zusammenpassen, ist die Schaltfläche grau und kann nicht betätigt werden:

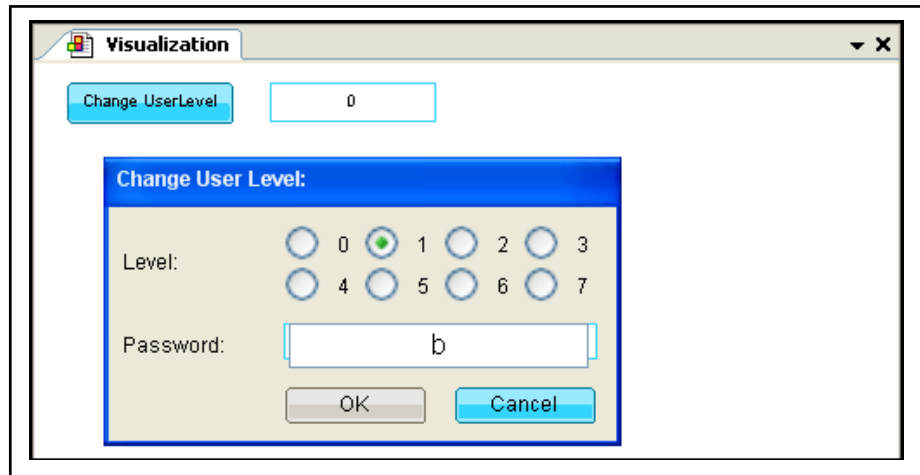


Abb. 20-96: Eingaben im Dialog "Change User Level"

Sie haben passende Werte eingegeben, somit wechselt OK nach blau und Sie können mit einem Mausklick darauf den Dialog schließen.

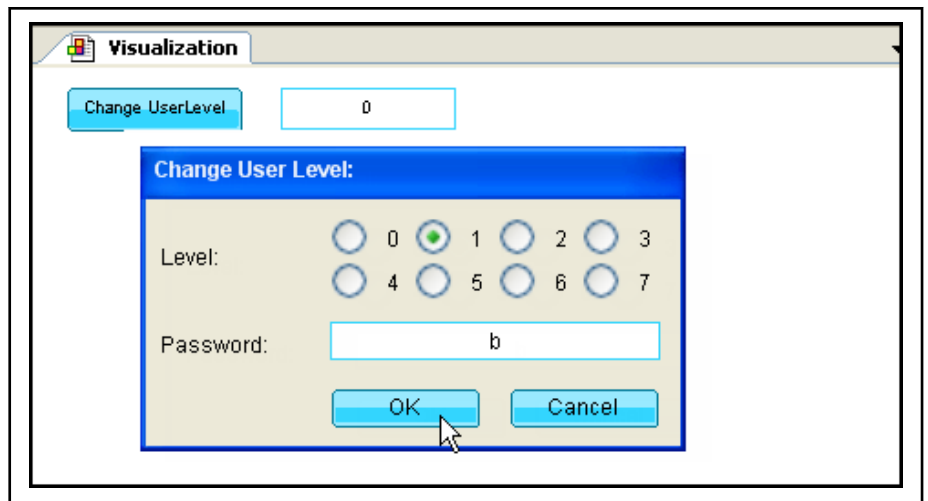


Abb. 20-97: OK zum Schließen des Dialogs verfügbar

Nun werden die Werte "tatsächlich" geschrieben (Beachten Sie, dass sie bisher nur auf dem Stack gehalten wurden!) und der geänderte Level ("1") wird angezeigt.

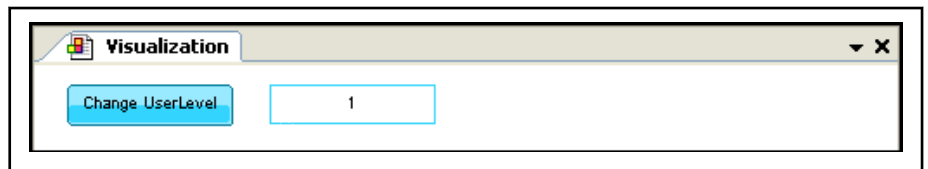


Abb. 20-98: Der neue Wert für den Benutzerlevel wird angezeigt

Globaler Dialog

Eine Visualisierung kann so konfiguriert werden, dass sie als globaler Dialog auf allen aktiven Visualisierungsgeräten erscheint, die unter dem Visualisierungsmanager konfiguriert wurden. So kann zum Beispiel ein Meldungsfenster auf allen angeschlossenen Visualisierungsgeräten erscheinen.

Eingaben bei einem der Dialoge wirken auf alle. Wenn ein Dialog beendet wird, werden alle anderen auch beendet. Davon abgesehen entspricht der Dialog einer Dialogvisualisierung oder einem Eingabedialog.

Die Schnittstelle `OpenDialog3` und `CloseDialog2` der Bibliothek `VisuElements` ermöglicht, IEC-Code zu implementieren um einen globalen Dialog zu erzeugen und zu beenden.

Implementierung eines globalen Dialogs

1. Implementieren Sie eine Funktion, die auf eine Visualisierung zugreift.
Die Visualisierung muss die Eigenschaft **Dialog** haben und sich wie ein Eingabedialog oder eine Dialogvisualisierung verhalten.

```
messageDialog := dialogMan.GetDialog('MessageDialog')
```

2. Setzen Sie den Parameter `GlobalOpenCloseDialog` des Dialogs. Dann erscheint der Dialog global auf allen Visualisierungsgeräten.

```
dialogMan.OpenDialog3(messageDialog, pClientData, TRUE, 0, 0, VisuElements Visu_InputFlags.GlobalOpenCloseDialog);
```

3. Implementieren Sie eine Funktion, die den Dialog `CloseDialog2` aufruft, um die Dialoge auf allen Visualisierungsgeräten zu beenden.

```
itfDialogManager2.CloseDialog2(messageDialog,
pClientData, VisuElems.Visu_InputFlags.GlobalOpenCloseDialog);
```

Siehe auch

- [Kap. "Eingabedialoge" auf Seite 1064](#)

Programmatischer Zugriff auf Eingabedialoge

Inhalt dieser Seite:

- ["Dialogmanager" auf Seite 1070](#)
- ["Methoden für den Dialogmanager" auf Seite 1073](#)

Programmatischer Zugriff auf einen Eingabedialog erstellt in einer IndraLogic-Visualisierung ist wie folgt möglich:

Die in der **Schnittstelle** einer Dialogvisualisierung definierten Parameter werden automatisch in eine Struktur `<Dialogname>_VISU_STRUCT` geschrieben.

(Dialogname = Name der Visualisierung).

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.

Siehe auch:

- [Kap. "Eingabedialoge" auf Seite 1064](#)
- [Kap. "Grundlegendes zum Schnittstelleneditor" auf Seite 831](#)

Dialogmanager

Alle Visualisierungen, die als **Dialog** konfiguriert sind, werden automatisch instanziiert und vom internen Dialogmanager verwaltet. Dieser Manager kann über den ebenfalls intern arbeitenden Visualisierungsmanager angesprochen werden, indem die Methode `GetDialogManager()` aufgerufen wird. Der Dialogmanager stellt einige Methoden bereit, über die man die Applikation einen Visualisierungsdialog handhaben kann.

Die verfügbaren Methoden sind beschrieben in ["Methoden für den Dialogmanager" auf Seite 1073](#).

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:

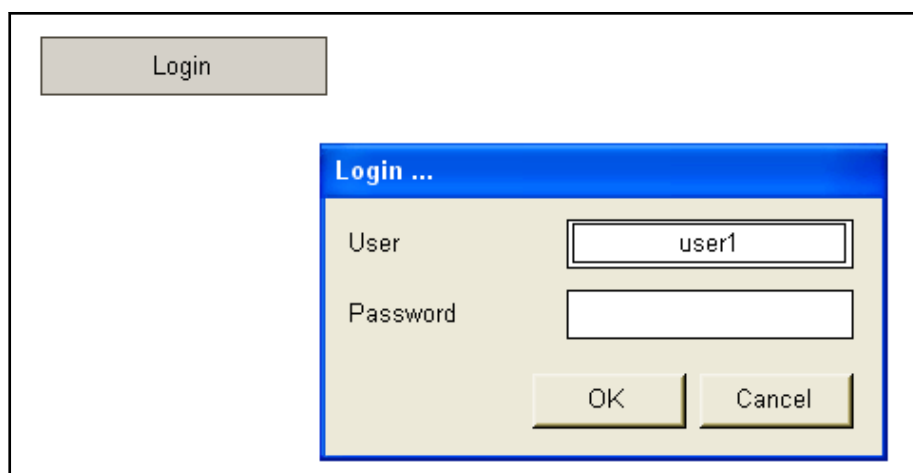


Abb. 20-99: Login-Dialog

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.

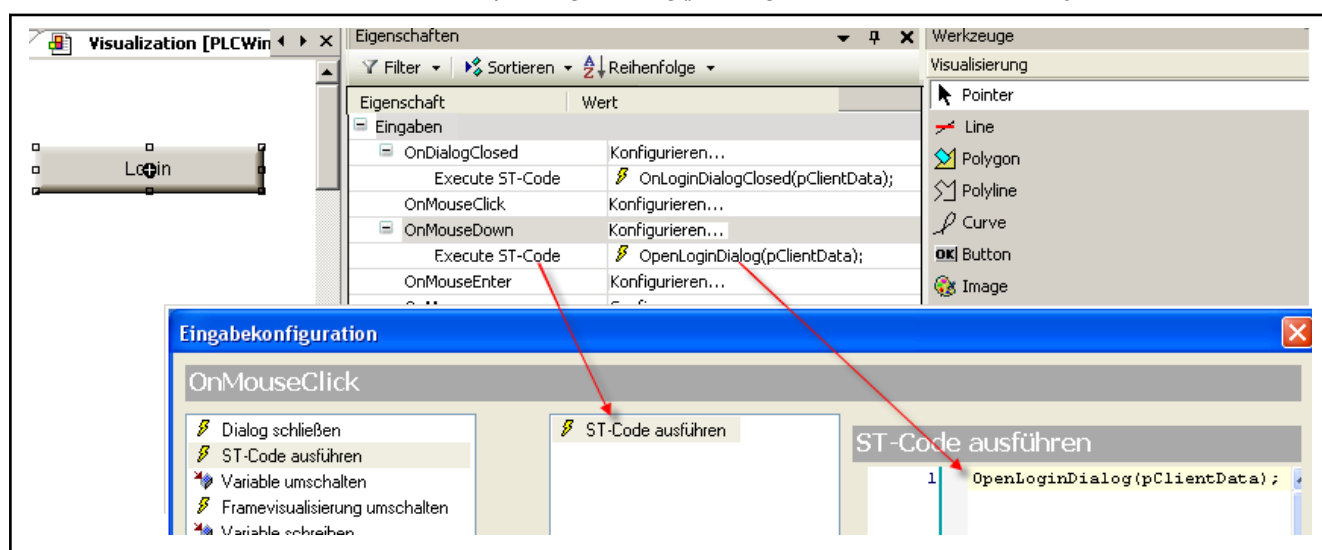


Abb. 20-100: Funktionsaufruf OpenLoginDialog()

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:

Implementierung der Funktion OpenLoginDialog()

Funktion OpenLoginDialog() wird aufgerufen wenn der Dialog geöffnet wird, z.B. bei einer OnMouseDown-Aktion auf das Schaltflächenelement.

```
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();
// The DialogManager is provided via the implicitly available VisuManager
IF dialogMan <> 0 AND pVisuClient <> 0 THEN
    loginDialog := dialogMan.GetDialog('VisuDialogs.Login'); // detects the login dialog
    IF loginDialog <> 0 THEN
        result := loginDialog.GetResult(); // gets the result (OK, Cancel) of the dialog
        IF result = Visu_DialogResult.OK THEN
            loginDialog.SetResult(Visu_DialogResult.None); // reset to default (none)
            pLoginInfo := dialogMan.GetClientInterface(loginDialog, pVisuClient);
            // structure Login_VISU_STRUCT gets read;
            // in the following the structure parameters can be set
            IF pLoginInfo <> 0 THEN
                stPassword := pLoginInfo^.stPassword; pLoginInfo^.stPassword := '';
                // reset the password
                stUsername := pLoginInfo^.stUsername;
            END_IF
            ELSEIF result = Visu_DialogResult.Cancel THEN
                loginDialog.SetResult(Visu_DialogResult.None);
                // react on 'Cancel'
            ELSE
                // nothing to do here
            END_IF
        END_IF
    END_IF
END_IF

```

Implementierung der Funktion OnLoginDialogClosed()

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();
// The DialogManager is provided via the implicitly available VisuManager
IF dialogMan <> 0 AND pVisuClient <> 0 THEN
    loginDialog := dialogMan.GetDialog('VisuDialogs.Login'); // detects the login dialog
    IF loginDialog <> 0 THEN
        result := loginDialog.GetResult(); // gets the result (OK, Cancel) of the dialog
        IF result = Visu_DialogResult.OK THEN
            loginDialog.SetResult(Visu_DialogResult.None); // reset to default (none)
            pLoginInfo := dialogMan.GetClientInterface(loginDialog, pVisuClient);
            // structure Login_VISU_STRUCT gets read;
            // in the following the structure parameters can be set
            IF pLoginInfo <> 0 THEN
                stPassword := pLoginInfo^.stPassword; pLoginInfo^.stPassword := '';
                // reset the password
                stUsername := pLoginInfo^.stUsername;
            END_IF
            ELSEIF result = Visu_DialogResult.Cancel THEN
                loginDialog.SetResult(Visu_DialogResult.None);
                // react on 'Cancel'
            ELSE
                // nothing to do here
            END_IF
        END_IF
    END_IF
END_IF

```

Methoden für den Dialogmanager

GetDialog(String stName): Liefert den Dialog, der gerade durch ein Ereignis betroffen ist, als IVisualisationDialog:

METHOD GetDialog		
Parameters:		
Name	Type	Comment
GetDialog	VAR_OUTPUT	
stName	VAR_INPUT	

Abb. 20-101:

GetClientInterface(): Liefert einen Zeiger auf die Dialogstruktur

METHOD GetClientInterface		
Parameters:		
Name	Type	Comment
GetClientInterface	VAR_OUTPUT	
dialog	VAR_INPUT	
pClient	VAR_INPUT	

Abb. 20-102:

OpenDialog(): Öffnet den Dialog für den Client

METHOD OpenDialog		
Parameters:		
Name	Type	Comment
OpenDialog	VAR_OUTPUT	
dialog	VAR_INPUT	
pClient	VAR_INPUT	
bModal	VAR_INPUT	
pRect	VAR_INPUT	

Abb. 20-103:

CloseDialog(): Schließt den Dialog für den Client

METHOD CloseDialog		
Parameters:		
Name	Type	Comment
CloseDialog	VAR_OUTPUT	
dialog	VAR_INPUT	
pClient	VAR_INPUT	

Abb. 20-104:

Ereignisse und Eingabeaktionen bemerken

Inhalt dieser Seite:

- ["Schließen eines Edit-Controls \(Schreiben einer Variable\) bemerken" auf Seite 1074](#)
- ["Tastatur-Ereignisse bemerken" auf Seite 1076](#)

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 ([Kap. 20.7 "Visualisierungsbibliotheken" auf Seite 1078](#)).

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 Funktionsbaustein-Instanz (die die Schnittstelle VisuElems.IEditBoxInputHandler aus VisuElem-Base.library implementiert) der globalen Instanz VisuElems Visu_Globals.g_VisuParamEventManager 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.

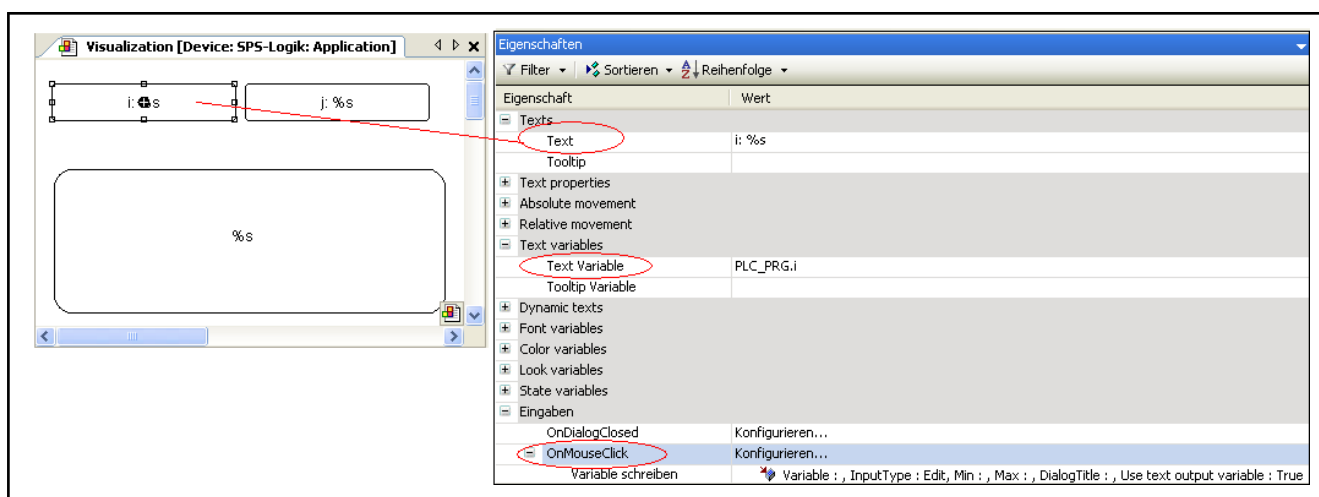


Abb. 20-105: Eingabe-Elemente

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.

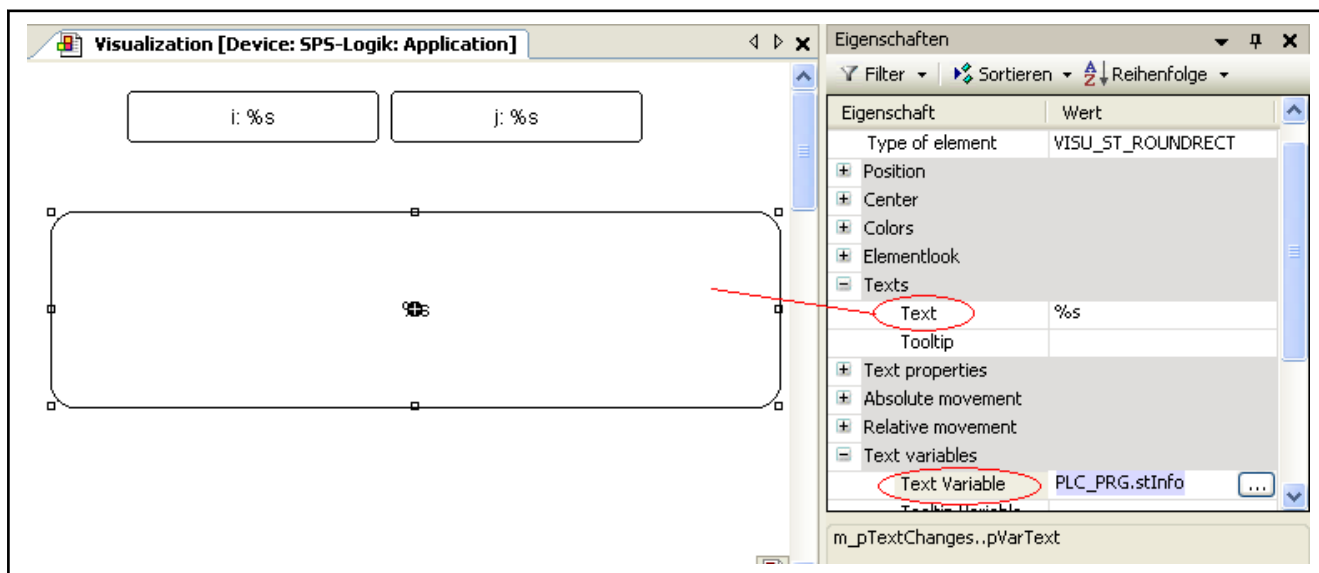


Abb. 20-106: Textausgabe-Element

Die Applikation enthält die folgenden Bausteine:

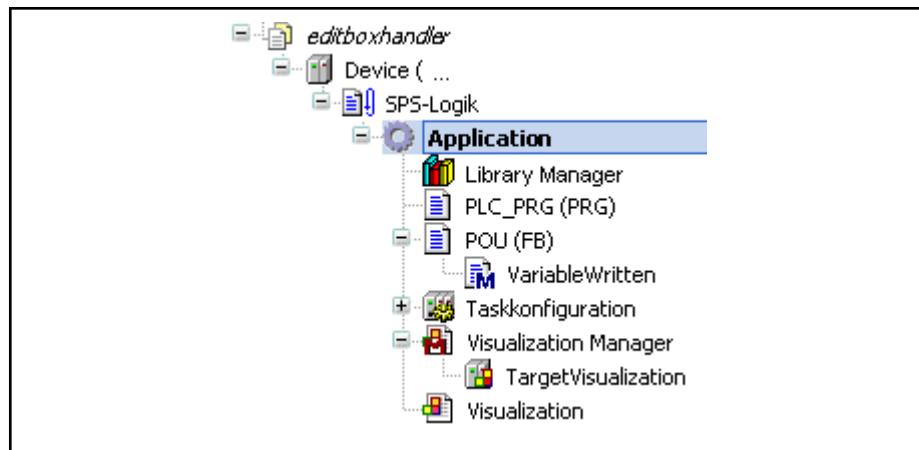


Abb. 20-107: Applikation mit Bausteinen
Programm PlcProg

```
PROGRAM PlcProg
VAR_INPUT
  i:INT; (* variable to be written by user input in visualization *)
  j:REAL; (* variable to be written by user input in visualization *)
  stInfo : STRING; (* information on the user input via the edit control field;
                    string gets composed by method VariableWritten;
                    result is displayed in the lower rectangle of the visualization *)
END_VAR
VAR
  inst : POU;
  bFirst : BOOL := TRUE;
END_VAR

IF bFirst THEN
  bFirst := FALSE;
  VisuElems.Visu_Globals.g_VisusEventManager.SetEditBoxEventHandler(inst);
  (* Call of method VariableWritten *)
END_IF
```

Funktionsbaustein POU

FUNCTION_BLOCK POU IMPLEMENTS VisuElems.IEditBoxInputHandler
(no further declarations, no implementation code)

Methode 'VariableWritten', gehört zu POU:

```
METHOD VariableWritten : BOOL
(* provides some information always when an edit control field is closed in the visual.
that is a variable gets written by user input in one of the upper rectangles *)
VAR_INPUT
  pVar : POINTER TO BYTE;
  varType : VisuElems.Visu_Types;
  iMaxSize : INT;
  pClient : POINTER TO VisuElems.VisuStructClientData;
END_VAR

// String stinfo, which will be displayed in the lower rectangle, is composed here
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 Tastenereignisse (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.



Beachten Sie die Konfigurationsmöglichkeiten für die Tastaturbedienung von Visualisierungen ([Kap. 20.11 "Tastaturbedienung im Online-Modus" auf Seite 1081](#)).

Um über solche Ereignisse informiert zu werden, kann ein Funktionsbaustein (der die Schnittstelle `VisuElem.IVisuEventManager` aus `VisuElemBase.library` implementiert) der globalen Instanz `VisuElem.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 `PlcProg.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. 20-108: Textkonfiguration des Visualisierungselements

Die Applikation enthält die folgenden Bausteine:

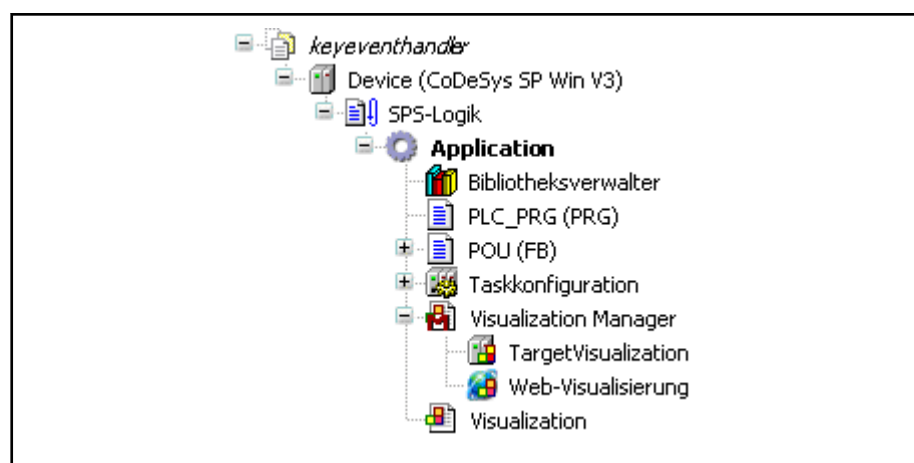


Abb. 20-109: Applikation mit POUs

Programm PlcProg

```

PROGRAM PlcProg
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 führt dazu gemeldet zu bekommen, wenn ein Tastenereignis vorliegt.

Funktionsbaustein POU

```

FUNCTION_BLOCK POU IMPLEMENTS VisuElems.IKeyEventHandler

```

Keine weiteren Deklarationen, kein Implementierungs-Code

Die Schnittstelle `VisuElems.IVisuUserEventManager` benötigt eine Methode mit folgender Signatur, die mit aller verfügbaren Information aufgerufen wird:

Method HandleKeyEvent, zur POU gehörig

```

/// This method will be called after a key event is released.
/// RETURN:
/// TRUE - When the handler has handled this event
///         and it should not be handled by someone else
/// FALSE - When the event is not handled by this handler
METHOD HandleKeyEvent : BOOL
VAR_INPUT
    /// Event type. The value is true if a key-up event was released.
    bKeyUpEvent : BOOL;
    /// Key code
    dwKey : DWORD;
    /// Modifier. Possible values:
    /// VISU_KEYMOD_SHIFT : DWORD := 1;
    /// VISU_KEYMOD_ALT : DWORD := 2;
    /// VISU_KEYMOD_CTRL : DWORD := 4;
    dwModifiers : DWORD;
    /// Pointer to the client structure where the event was released
    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,

```

20.7 Visualisierungsbibliotheken

Visualisierungselemente sind in IndraLogic 2G als Funktionsbausteine implementiert. Sie können auf diese Weise als Bibliotheken bereitgestellt werden.

Das IndraLogic 2G-Visualisierungsprofil legt auch genau fest, welche Elemente aus diesen Bibliotheken in **Werkzeuge** des Visualisierungseditors bereitstehen.

Eine Visualisierungsbibliothek ist immer als **Platzhalterbibliothek** 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 Platzhalterbibliotheken unterscheidet, bei denen die Platzhalter aus der Gerätebeschreibung aufgelöst werden.

Basisbibliotheken sind standardmäßig eingebunden, sobald ein Visualisierungsobjekt in ein Standardprojekt eingefügt wird. Sie referenzieren weitere Bibliotheken, die hier nicht aufgeführt sind. Im Standardfall müssen Sie die Visualisierungsbibliotheken nicht explizit einfügen oder in Ihren Applikationen verwenden:

Basisbibliotheken, Namespaces

- **VisuElems:** Basissatz an Visualisierungselementen
- **VisuElemMeter:** Zeigerinstrument und Balkenanzeige
- **VisuElemWinControls:** Tabelle, Textfeld, Scrollbalken
- **VisuElemTrace:** Trace
- **VisuInputs:** Handhabung der Benutzereingaben einer Visualisierung

In Visualisierungen, die einem Gerät zugewiesen sind, kann es **geräteabhängige Einschränkungen** gemäß den Bibliotheken, die das Gerät unterstützt, geben. Diese betreffen Konfigurationsmöglichkeiten, wie zum Beispiel Liste der verfügbaren Elementen, Schriftarten, Farben, Bildformate, maximale Anzahl an Visualisierungen unterhalb des Geräts.

20.8 Optionen

Standardmäßig kann der Dialog **Optionen** mit **Extras ▶ Optionen ▶ IndraLogic 2G** geöffnet werden.

20.8.1 Dialog 'Optionen' - 'Visualisierung'

Funktion: Der Dialog dient der Konfiguration des Visualisierungseditors und zur Laufzeit der Konfiguration der integrierten Visualisierung.

Aufruf: Menü **Extras ▶ Optionen ▶ IndraLogic 2G**, Kategorie **Visualisierung**

Siehe auch:

- [Kap. "Dialog 'Optionen- 'Visualisierung'" auf Seite 779](#)

20.8.2 Dialog 'Optionen' - 'Visualisierungsstile'

Funktion: Der Unterdialog des Dialogs **Optionen** dient der Definition des Visualisierungsstils, der in **Visualisierungen in Bibliotheken** und **globalen Visualisierungen** in Projekten verwendet wird.

Aufruf: Menü **Extras ▶ Optionen ▶ IndraLogic 2G**, Kategorie **Visualisierungssstile**



Im Online-Betrieb ist nicht die aktuelle Einstellung des Optionsdialogs wirksam, sondern die, die im Visualisierungsmanager vorliegen. Siehe die allgemeinen Informationen zu den Visualisierungsstilen: [Kap. 20.10 "Visualisierungsstil" auf Seite 1080](#).

Siehe auch:

- [Kap. "Dialog 'Optionen- 'Visualisierungsstile'" auf Seite 781](#)

20.9 Visualisierungsprofil

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 ([Kap. 20.7 "Visualisierungsbibliotheken" auf Seite 1078](#))
- eine Auswahl von aus den eingebundenen Bibliotheken stammenden Visualisierungselementen, die in der Ansicht 'Werkzeuge' bereitstehen sollen ([Kap. 20.3.1 "Ansicht 'Werkzeuge'" auf Seite 826](#)).

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 Visu-Style-Optionen 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 Bibliotheken. 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 IndraLogic ist dies die Bibliothek **VisuElements.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.

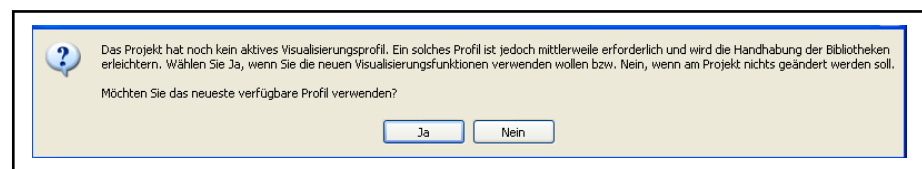


Abb. 20-110: Meldungsbox, falls Profil fehlt

Wenn Ja geantwortet wird, wird das "Neueste Profil" verwendet ([Kap. 20.8.2 "Dialog 'Optionen- 'Visualisierungsstile'" auf Seite 1079](#)).

Bei Nein wird das älteste verfügbare Profil in den Projekteinstellungen eingetragen (genannt "Kompatibilitätsprofil"), aber im Projekt werden keine weiteren Änderungen vorgenommen.

20.10 Visualisierungsstil

Ein Stil ist eine Datei, in der symbolische Namen, nämlich die Stileigenschaften, für Werte definiert sind.

Ein Stil ist somit eine Sammlung von Stileigenschaften, die entweder einen festen Wert für die Eigenschaft **Farbe** oder **Schriftart** oder eine feste Referenz auf eine Bilddatei enthalten.

Die Datei ist im XML-Format und hat die Dateierweiterung `*.visu-style.xml`.

Sie können pro Projekt einen Stil auswählen und damit bestimmen, dass dessen Stileigenschaften beim Konfigurieren der Elemente zur Verfügung stehen.

Übrigens bietet IndraLogic die Stileigenschaften kontextsensitiv an. So werden bei der Eigenschaft **Farbe** nur Stileigenschaften für Farben angeboten. Wenn Sie konsequent die Elemente mit Stil gestalteten, erhalten Sie ein einheitliches Erscheinungsbild und Sie gewinnen den Vorteil, dass Sie bei einem Änderungswunsch der Gesamtoptik nur den Stil anpassen müssen.

Der Stil, der für alle Visualisierungen, die innerhalb einer Applikation, global im Ordner "Allgemeiner Bausteinteil" oder in Bibliotheken enthalten sind, gültig ist, steht in Option **Folgenden Visualisierungsstil verwenden** in **Extras ▶ Optionen ▶ IndraLogic 2G**, Kategorie **Visualisierungsstil**.

20.11 Tastaturbedienung im Online-Modus

Für die Tastaturbedienung einer Visualisierung im Online-Modus werden einige Standardtastaturkürzel von jedem Gerät unterstützt. Zusätzlich können geräteabhängig weitere Tasten(kombinationen) anwendbar sein.

Taste(n)	Aktion
<Tab>	Gemäß der chronologischen Reihenfolge der Einfügung wird das nächste Element ausgewählt; innerhalb einer Tabelle wird jede einzelne Zelle angewählt; wenn ein Frame-Element ausgewählt ist, wird die Selektion danach an die einzelnen enthaltenen Elementen weitergegeben
<Umschalten > + <Tab>	Das vorherige Element wird ausgewählt; dabei wird in umgekehrter Reihenfolge wie mit <Tab>
<Eingabe>	Eine Eingabeaktion auf das ausgewählte Element wird ausgeführt.
<Pfeiltasten>	Das nächste Element in der durch die Pfeiltaste gegebene Richtung wird ausgewählt.

Tab. 20-14: Standardtastenkombination



Für eine Diagnosevisualisierung 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 eingegeben werden, nicht ausgeführt werden.

- [Kap. 20.2.5 "Befehl 'Tastaturbedienung aktivieren'" auf Seite 813](#)
- [Kap. "Diagnosevisualisierung" auf Seite 1061](#)

Gerätespezifische Tastaturbedienung

Außer den oben beschriebenen Standardtastenkombinationen gibt es zusätzlich geräteabhängige Kombinationen, die bei laufender Visualisierung auf diesem Gerät verwendet werden können: [Kap. "'Visualisierungsmanager-Standardtastaturkonfiguration'" auf Seite 1060](#)



Es gibt die Möglichkeit, Tastaturereignisse im Applikationscode zu behandeln: [Kap. "Ereignisse und Eingabeaktionen bemerken" auf Seite 1074.](#)

21 Service und Support

Für Ihre schnelle und optimale Unterstützung verfügen wir über ein dichtes weltweites Servicenetz. Unsere Experten stehen Ihnen mit Rat und Tat zur Seite. Sie erreichen uns täglich **rund um die Uhr - auch an Wochenenden und Feiertagen**.

Service Deutschland

Unser technologieorientiertes Competence Center in Lohr deckt alle Belange rund um den Service für elektrische Antriebe und Steuerungen ab.

Sie erreichen unsere **Service-Hotline** und unseren **Service-Helpdesk** unter:

Telefon: **+49 9352 40 5060**
Fax: **+49 9352 18 4941**
E-Mail: service.svc@boschrexroth.de
Internet: <http://www.boschrexroth.com>

Auf unseren Internetseiten finden Sie ergänzende Hinweise zu Service, Reparatur (z. B. Anlieferadressen) und Training.

Service weltweit

Außerhalb Deutschlands nehmen Sie bitte zuerst Kontakt mit Ihrem Ansprechpartner auf. Die Hotline-Rufnummern entnehmen Sie bitte den Vertriebsadressen im Internet.

Vorbereitung der Informationen

Wir können Ihnen schnell und effizient helfen, wenn Sie folgende Informationen bereithalten:

- Eine detaillierte Beschreibung der Störung und der Umstände
- Angaben auf dem Typenschild der betreffenden Produkte, insbesondere Typenschlüssel und Seriennummern
- Ihre Kontaktdaten (Telefon-, Faxnummer und E-Mail-Adresse)

Index

Symbole

__AdrInst.....	454
__BitOffset.....	454
__Cast.....	454
__CATCH.....	353, 454
__COMPARE_AND_SWAP	
Beispiel.....	358
Funktionsweise.....	357
__Copy.....	454
__CRC.....	454
__CURRENTTASK	
Beispiel.....	357
__DELETE.....	454
__ENDTRY'.....	353, 454
__FCall.....	454
__FINALLY.....	353, 454
__Init.....	454
__ISVALIDREF.....	349, 384, 454
__Lazy.....	454
__LocalOffset.....	454
__MaxOffset.....	454
__NEW.....	454
__PropertyInfo.....	454
__QUERYINTERFACE.....	351, 454
__QUERYPOINTER.....	352
__RefAdr.....	454
__RELOC.....	454
__System.ExceptionCode, Datentyp.....	353
__SystemScope.....	454
__TRY.....	353, 454
__TypeOf.....	454
__uxint.....	382, 454
__VarInfo.....	454
__Wait.....	454
__VARINFO	
Beispiel.....	354
Datentyp SYSTEM.VAR_INFO.....	355
Syntax beim Aufruf.....	354
Syntax in der Deklaration.....	354
__XADD	
Beispiel.....	358
Funktionsweise.....	358
__xword.....	382, 454
:=, ExST.....	246

0 ... 9

2-dimensionales Array	
Beispiel.....	387
Datenzugriff.....	387
Deklaration.....	387
Initialisierung.....	387
3-dimensionalen Arrays einer benutzerdefinierten Struktur	
Beispiel.....	388
3-dimensionales Array	

Beispiel.....	387
Datenzugriff.....	388
Deklaration.....	387
Initialisierung.....	387

A

Abarbeitungsreihenfolge	
in AS.....	254
in ST.....	242
Abkürzungen.....	5
Ablage	
Projekt.....	59
Ablaufkontrolle.....	201
ABS.....	343
Accessor-Methode.....	568
ACOS.....	347
ACTION.....	454
ActiveX	
Visualisierungselement.....	1050
ADD.....	319
ADR.....	333, 383
Adresse.....	371
E/A.....	81
Syntax.....	371
Adressen	
Speicherposition.....	372
Adressierung	
Bits.....	369
Aktion.....	264
Aktionsassoziation einfügen.....	686, 687
aktualisieren	
strukturierte Variablen	727
Aktualisieren	
Parameter.....	712
Symbolbibliothek.....	810
Alarmfarbe.....	880, 909, 920, 941, 972
Alarmzustand.....	871, 889, 893, 898, 965
Alias.....	518
ALIAS.....	384
Alles bereinigen.....	651
Alles deselektieren.....	822
Allgemeine Einstellungen	
Option.....	765
Allgemeiner Baustein	
Projekteinstellungen.....	507
Projektglobale Objekte und Instanzen.....	507
Alt.....	878, 886, 896, 905, 915, 928, 948, 969, 982, 985
Alternativ.....	684
AND.....	323
AND_THEN.....	324
AND, Pragma.....	444
ANDN.....	278
Anmelden, Benutzer.....	62
Anpassen.....	682

Index

Anschluss			
entfernen.....	697		
selektieren.....	697		
verbinden.....	696		
zurücksetzen.....	697		
ANSI Z535.6-2006.....	2		
Ansicht			
Querverweisliste.....	628		
Speicheransicht.....	633		
Anweisungsliste.....	94		
ANY.....	378		
ANY <type>.....	378		
Anzeigen der Vorschau auf Bibliotheksvisu- alisierung.....	805		
Applikation.....	508		
bereinigen.....	650		
Download.....	230		
Information.....	654		
Inhalt.....	654		
kompilieren.....	655		
Name.....	72		
neu übersetzen.....	650		
Reset.....	200		
übersetzen.....	649, 655		
vergleichen.....	654		
von SPS entfernen.....	655		
Applikationen, Geräteeditor.....	521, 522		
Arbeiten im CFC-Editor			
Befehle beim Editieren.....	291		
Editieren.....	289		
Verbinden.....	290		
Arbeitsblatt.....	689		
Arcuscosinus.....	347		
Arcussinus.....	346		
Arcustangens.....	347		
Array			
deklarieren.....	89		
Monitoring-Bereich einstellen.....	240		
Monitoringbereich begrenzen.....	208		
Zugriff.....	369		
ARRAY.....	386		
einer Struktur.....	386		
Array eines Funktionsbausteins			
Beispiel.....	389		
Array von Arrays			
Beispiel.....	390		
Syntax bei Datenzugriff.....	390		
Syntax bei Deklaration.....	389		
AS			
Abarbeitungsreihenfolge.....	254		
Aktion.....	264		
Aktionsqualifizierer.....	256		
Bibliothek.....	759		
Codeerzeugung.....	759		
Codegenerierung.....	756		
Eigenschaften.....	756		
Elementeigenschaften.....	270		
implizite Variablen.....	257		
Makro.....	269		
Online-Betrieb.....	253		
Projekteinstellung.....	758		
Schritt.....	262		
Schrittzeit.....	257		
Sprung.....	269		
Tipp-Betrieb.....	258		
Transition.....	262		
übersetzen.....	756		
Verzweigung.....	267		
AS-Editor.....	253		
Option.....	773		
Symbolleiste.....	240		
AS-Flag.....	258		
ASIN.....	346		
Assoziation für AS-Aktion.....	686, 687		
AT-Deklaration.....	130		
ATAN.....	347		
Attribut			
'enable_dynamic_creation'.....	438		
Attribut 'no-exit'.....	425		
Attribut			
für Pragma.....	403		
Attribut 'call_after_global_init_slot'.....	404		
Attribut 'call_after_init'.....	404		
Attribut 'call_after_online_change_slot'.....	405		
Attribut 'call_before_global_exit_slot'.....	406		
Attribut 'call_on_type_change'.....	406		
Attribut 'const_non_replaced'.....	407		
Attribut 'const_replaced'.....	407		
Attribut 'dataflow'.....	407		
Attribut 'displaymode'.....	408		
Attribut 'estimated-stack-usage'.....	409		
Rekursiver Methodenaufruf.....	409		
Attribut 'ExpandFully'.....	412		
Attribut 'global_init_slot'.....	413		
Attribut 'hide_all_locals'.....	415		
Attribut 'hide'.....	414		
Attribut 'init_namespace'.....	416		
Attribut 'init_on_onchange'.....	416		
Attribut 'initialize_on_call'.....	415		
Attribut 'instance-path'.....	417		
Attribut 'io_function_block', 'io_functi- on_block_mapping'.....	418		
Attribut 'linkalways'.....	419		
Attribut 'monitoring'.....	419		
Attribut 'namespace'.....	423		
Attribut 'no_assign_warning'.....	424		
Attribut 'no_assign'.....	424		
Attribut 'no_check'.....	424		
Attribut 'no_copy'.....	425		
Attribut 'no_instance_in_retain'.....	426		
Attribut 'no_virtual_actions'.....	426		
Attribut 'no-exit'.....	425		
Attribut 'noinit'.....	426		
Attribut 'obsolete'.....	430		
Attribut 'pack_mode'.....	431		
Attribut 'parameterstringof'.....	838		

Attribut 'pingroup'.....	429	mit EN/ENQ.....	704
Attribut 'ProcessValue'.....	438	reparieren.....	712
Attribut 'qualified_only'.....	434	Bausteinausgang	
Attribut 'reflection'.....	435	CFC.....	300
Attribut 'strict'.....	436	Bausteineingang	
Attribut 'subsequent'.....	436	CFC.....	300
Attribut 'symbol'.....	436	einfügen, FUP/KOP.....	705
Attribut 'to_string'.....	435	bedingte Kompilierung.....	439
Attribut 'VAR_IN_OUT_AS_POINTER'.....	836	Befehl	
Attribut 'warning disable'.....	438	'Aktive Applikation setzen'.....	646
Attribut 'warning restore'.....	438	'Attributierte Komponente als Eingang	
Aufrufen		verwenden'.....	701
Funktion.....	558	'Download Trace'.....	734
Funktionsbaustein.....	556	Ablaufkontrollmodus umschalten.....	673
Funktionsbaustein, ST.....	252	Aktuellen Gerätebenutzer ausloggen.....	656
Programm.....	554	Alle Forces zur Überwachungsliste hinzu-	
Aufrufliste.....	204	fügen.....	668
Aufzählung.....	398	Anzeigemodus.....	676
Aufzeichnung siehe Datenerfassung.....	784	Aufrufbaum.....	631
Ausdruck, ST.....	242	Aufrufliste.....	667
Ausführen		Ausführen bis Cursor.....	670
ST-Code in FUP/KOP/AWL.....	285	Ausführen bis Rücksprung.....	669
Ausgabevariable.....	301	Ausloggen.....	655
Ausgang		Automatisch skalieren.....	732
CFC.....	297	Betriebsmodus.....	673
Gerät.....	77	Bootapplikation erzeugen.....	656
umkehren.....	690	Core Dump.....	674
Ausgangsaktion.....	265	Core Dump erzeugen.....	674
Ausgangsspeicher.....	371	Core Dump laden.....	675
Ausnahmebehandlung siehe Exception-		Core Dump schließen.....	675
handling.....	353	Einloggen.....	652
Ausnahmefehler siehe Exception.....	353	Einzelschritt.....	669
Aussehen.....	871, 880, 889, 899, 909, 921, 965,	Einzelzyklus.....	659
	1053	Ergebnisse auf aktuelle Deklaration be-	
Aussehensvariable.....	913, 925	schränken.....	743
Aussehensvariablen.....	875, 894, 903	FB-Eingang forcen.....	700
Äußerer Rahmen.....	999, 1003, 1007, 1011	Forcen für alle Werte aufheben.....	672
Autodeklaration.....	112	Funktionsbausteinelement für Forcen	
AWL.....	94	vorbereiten.....	699
konvertieren.....	713	Geräte-Logbuch vom Core Dump laden.....	675
Option.....	768	Haltepunkt aktivieren.....	666
Zeile löschen.....	709	Haltepunkt bearbeiten.....	666
AWL-Editor		Haltepunkt deaktivieren.....	666
Online-Betrieb.....	277	Haltepunkt umschalten.....	666
		Haltepunkte.....	664
B		in Einkanal-Darstellung konvertieren.....	736
B.....	371	in Mehrkanal konvertieren.....	736
Balkenanzeige		Laden.....	656
frei konfigurierbar, Visualisierungsele-		Mehrfacher Download.....	659
ment.....	992	Nächste Anweisung anzeigen.....	670
Visualisierungselement.....	992	Nächste Anweisung festlegen.....	670
Baustein		Neuer Haltepunkt.....	666
bereinigen, FUP/KOP.....	712	Online-Change.....	657
CFC.....	297	Parallelen Modus wechseln.....	708
einfügen, FUP/KOP/AWL.....	703	Prozedurschritt.....	669
FUP/KOP/AWL.....	282	Reset Kalt.....	662
FUP/KOP/AWL, einfügen.....	704	Reset Ursprung.....	663
		Reset warm.....	661

Index

SPS-Projektvergleich.....	641	Befehl.....	715
Start.....	659	Bibliotheken	
Stop.....	659	Option.....	771
Überwachung - Alle Forces anzeigen.....	668	Bibliotheks-Repository.....	676
Überwachung - Überwachungsliste <n>.....	667	Export aus dem Bibliotheksrepository.....	238
Werte forcen.....	671	Hinzufügen einer Bibliothek.....	237
Werte schreiben.....	672	Bibliotheksentwicklung	
Beim Betreten tasten, falls Maus gefan-		Informationen.....	236
gen..... 877, 885, 895, 904, 927, 948, 969, 984		Bibliotheksinfo	
Benutzer abmelden		Funktionen für Eigenschaftenzugriff.....	12
Befehl.....	637	Bibliotheksinformationen	
Benutzer anmelden.....	62	Befehl.....	719
Befehl.....	635	Bibliotheksprojekt	
Benutzer und Gruppen		Kategorie.....	580
Befehl.....	637	signieren.....	13, 583
Geräteeditor.....	539	übersetzte Bibliothek.....	583
Benutzerdefinierte Attribute	403	Bibliotheksrepository.....	235
Benutzereingabe		Bibliotheksverwalter.....	235, 548
Element editieren.....	843	Bibliothek hinzufügen.....	714
Benutzergruppe.....	57	Eigenschaften.....	716
Benutzerkonto.....	62	Platzhalter.....	718
Benutzerverwaltung.....	57	Versuchen Bibliothek neu zu laden.....	715
Befehl.....	634	Bild.....	821
Benutzer abmelden.....	637	Visualisierungselement.....	906
Benutzer anmelden.....	635	Bildauswahl.....	547
Benutzer und Gruppen.....	637	Bilddatei	
Konfiguration.....	635	einfügen.....	719
Objekt Zugriffskontrolle.....	608	Bilddateien für Visualisierung.....	779
Projekt.....	61	Bilder ohne Pfad registrieren.....	779
Rechte.....	639	Bildersammlung.....	126, 547
Voreinstellung der Dialoge, Visualisie-		anlegen.....	127
rung.....	1059	Eigenschaften.....	757
Bereichsanfang..... 1000, 1004, 1009, 1013, 1016		globale.....	126
Bereichsende..... 1000, 1004, 1009, 1013, 1016		Bildwechsler	
Beschriftung..... 995, 999, 1003, 1008, 1012, 1016		Visualisierungselement.....	1019
Beschriftungsverschiebung..... 1008, 1012		Binärzahl.....	365
Bezeichner, Regeln.....	445	BIT.....	382, 396
Bézierkurve		BITADR.....	334
Visualisierungselement.....	887	Bitmap.....	821
Bibliothek		Bitmap, Eigenschaften.....	751
Abbildungsdefinition.....	771	Bitzugriff.....	396
Bausteine verwenden.....	116	Auf eine Variable mit einer globalen kon-	
Bibliotheksexplorer ein-/ausschalten.....	628	stanten Variablen.....	370
deinstallieren.....	676	auf Variable.....	369
erstellen.....	789	Mit BIT-Datentypen.....	370
Hinzufügen zum Projekt.....	237	Syntax bei Aufrufen.....	370
installieren.....	676	Syntax bei Deklarationen.....	370
Kompatibilität prüfen.....	681	BOOL.....	375
Namensraum.....	360	BOOL_TO.....	335
Profil.....	676	BOOL-Konstanten.....	362
Recipe Management.....	789	Bootapplikation	
Repository.....	676	Eigenschaften.....	748
Speicherort.....	676	erzeugen.....	189
Visualisierung.....	1078	Breakpoint	
Bibliothek hinzufügen		Verwenden von Haltepunkten.....	193
Befehl.....	714		
Bibliothek im IndraWorks.....	678		
Bibliothek neu laden			

Breite.....	870, 898, 908, 920, 930, 932, 935, 939, 941, 950, 955, 964, 971, 976, 979, 983, 987, 989, 991, 993, 998, 1002, 1006, 1010, 1015, 1021, 1024, 1027, 1030, 1034, 1051, 1053
Breite gleichmachen.....	817
BY.....	247
BYTE.....	375
Byte-Adressierung.....	371
C	
C Stub-Date.....	650
C-Rahmendatei für externe Bibliothek.....	650
CAL.....	334
Call Stack.....	204
Call_after_init_slot, Pragmaattribut.....	404
call_after_init, Pragmaattribut.....	404
Call_after_online_change_slot, Pragmaattribut.....	405
call_before_global_exit_slot, Pragma.....	406
call_on_type_change, Pragma.....	406
CASE.....	248
CATCH.....	353
CFC	
an den Anfang.....	691
ans Ende.....	692
Anschlüsse entfernen.....	697
Anschlüsse zurücksetzen.....	697
Arbeitsblatt bearbeiten.....	689
Ausführungsreihenfolge ändern.....	694
Ausführungsreihenfolge konfigurieren.....	753
Ausgang hinzufügen.....	697
Automatischer Datenfluss-Modus.....	755
Eingang hinzufügen.....	697
eins vor.....	692
eins zurück.....	693
Expliziter Ausführungsreihenfolge-Modus.....	755
Gruppieren.....	699
Gruppierung aufheben.....	699
Kontrollpunkt entfernen.....	698
Kontrollpunkt erzeugen.....	698
Nach Datenfluss anordnen'.....	693
Nächste Kollision zeigen.....	696
Parameter bearbeiten.....	694
Parameterwerte.....	695
Referenz.....	691
Reset.....	690
Seitengröße bearbeiten.....	689
Set.....	690
Topologisch anordnen.....	694
Verbinder zurücksetzen.....	698
Verbindung lösen.....	696
Verbindungen routen.....	698
Verbindungsmarke.....	699
verbundene Anschlüsse selektieren.....	697
CFC-Editor.....	288
Debugging.....	293
Editor konfigurieren.....	288
Haltepunkt.....	293
Monitoring.....	293
Onlinebetrieb.....	293
Option.....	765
seitenorientiert.....	291
Symbolleiste.....	240
CFC-Editor - seitenorientiert	
Ausführungsreihenfolge.....	292
Editieren.....	292
CFC-Editor Onlinebetrieb	
Befehle im Onlinebetrieb.....	296
Boolesche Variable monitoren.....	293
Forcen und Schreiben von Variablen.....	294
Haltepunkt-Positionen.....	296
Konstante Eingangsparameter von Funktionsbaustein-Instanzen ändern.....	294
Monitoring.....	293
Programmierbaustein schrittweise ausführen.....	296
Skalare Variable monitoren.....	293
cfg-Datei.....	72
Check-Funktion vermeiden.....	424
CheckBounds.....	386
Checkbox	
Visualisierungselement.....	990
CheckRangeSigned.....	401
CheckRangeUnsigned.....	401
CmpTraceMgr	
Laufzeitsystemkomponente.....	216
Code	
Einrückung.....	621
Code erzeugen.....	649
Code statisch analysieren.....	132
Codeanalyse.....	132
Combobox Array	
Visualisierungselement.....	934
Combobox Integer	
Visualisierungselement.....	931
Compiled Library siehe Übersetzte Bibliothek.....	
Compiled-Library im IndraWorks.....	678
Compiler	
Option.....	759
Warnung.....	760
Compiler-Version.....	641
Compilerfehler.....	
C0001.....	460
C0002.....	460
C0003.....	460
C0004.....	461
C0005.....	461
C0006.....	461
C0007.....	462
C0008.....	462
C0009.....	462
C0010.....	462
C0011.....	463
C0013.....	463

Index

C0015.....	463	C0122.....	485
C0018.....	464	C0124.....	485
C0020.....	464	C0125.....	486
C0022.....	465	C0126.....	486
C0023.....	465	C0130.....	486
C0026.....	465	C0131.....	487
C0027.....	466	C0132.....	487
C0030.....	466	C0136.....	487
C0031.....	466	C0139.....	488
C0032.....	467	C0140.....	488
C0035.....	467	C0141.....	489
C0037.....	467	C0142.....	489
C0038.....	468	C0143.....	489
C0039.....	468	C0144.....	490
C0040.....	469	C0145.....	490
C0041.....	469	C0149.....	490
C0043.....	470	C0161.....	491
C0044.....	470	C0162.....	491
C0045.....	471	C0168.....	491
C0046.....	471	C0169.....	492
C0047.....	471	C0174.....	492
C0048.....	472	C0175.....	492
C0049.....	472	C0177.....	492
C0050.....	472	C0178.....	493
C0061.....	473	C0182.....	493
C0062.....	473	C0185.....	493
C0064.....	474	C0189.....	494
C0066.....	474	C0190.....	494
C0068.....	474	C0191.....	494
C0069.....	475	C0195.....	494
C0070.....	475	C0196.....	495
C0072.....	475	C0197.....	495
C0074.....	476	C0198.....	495
C0075.....	476	C0199.....	496
C0076.....	476	C0201.....	496
C0077.....	477	C0203.....	496
C0080.....	477	C0204.....	497
C0081.....	478	C0205.....	497
C0082.....	478	C0206.....	497
C0084.....	478	C0208.....	498
C0086.....	479	C0211.....	498
C0087.....	479	C0212.....	498
C0089.....	480	C0216.....	498
C0090.....	480	C0217.....	499
C0091.....	480	C0218.....	499
C0094.....	481	C0219.....	499
C0096.....	481	C0221.....	500
C0097.....	482	C0222.....	500
C0098.....	482	C0224.....	500
C0101.....	482	C0227.....	501
C0102.....	483	C0228.....	501
C0104.....	483	C0234.....	501
C0114.....	483	C0235.....	502
C0116.....	483	C0236.....	502
C0117.....	484	C0237.....	502
C0118.....	484	C0238.....	503
C0119.....	484	C0239.....	503
C0120.....	485	C0240.....	503

C0241.....	504	Doppelallokation von Speicher bei Instanzpfaden.....	149
C0242.....	504	Speicherlokation bei persistenten Funktionsbaustein-Instanzen.....	150
C0243.....	505	Datenquellenverwalter	
C0380.....	505	Datenquelle initial hinzufügen.....	175
C0509.....	505	Datenquellentyp.....	174
C0511.....	506	Datenquellenvariablen bearbeiten.....	179
const_no_replaced, Pragmaattribut.....	407	Datenschnittstelle aktualisieren.....	182
const_replaced, Pragmaattribut.....	407	Datenübertragung.....	174
CONSTANT.....	361	Dialog 'Datenquellenassistenten initialisieren - Datenquellenvariablen durchsuchen'.....	511
CONTINUE.....	252	Dialog 'Datenquellenassistenten initialisieren - Providereinstellungen'.....	510
Continuous Function Chart (CFC)		Entfernte Daten verwenden.....	183
Seitenorientierter CFC.....	288	Entfernte Variable auf eine lokale Variable mit konformen Datentyp abbilden.....	181
Core Dump.....	224	Geräte mit Adressmonitoring initial verbinden.....	176
Speicheransicht öffnen.....	633	Grundlagen.....	174
COS.....	346	Kommunikation bearbeiten.....	182
CreateRecipe.....	790	Objekt 'Datenquelle' - Registerkarte 'Allgemein und Diagnose'.....	517
CreateRecipeNoSave.....	790	Objekt 'Datenquelle' - Registerkarte 'Typabbildungen'.....	513
CurrentVisu		Objekt 'Datenquelle' - Registerkarte 'Variablen'.....	511
Variable für Zugriff auf Visualisierung.....	1058	Objekt 'Datenquellenverwalter'.....	510
D		Registerkarte 'Allgemein und Diagnose'.....	517
D.....	371, 376	Registerkarte 'Kommunikation'.....	514
dar-Datei.....	540	Registerkarte 'Typabbildungen'.....	513
Darstellung von Informationen		Variablen für Datenübertragung auswählen.....	180
Bezeichnungen und Abkürzungen.....	5	Verbindungstyp.....	174
Data Unit Typ.....	517	Datensatz siehe Sample.....	216
dataflow, Pragma.....	407	Datentyp.....	374
DATE.....	376	__VECTOR.....	391
DATE_AND_TIME.....	376	ANY.....	378
DATE_AND_TIME-Konstante.....	364	ANY <type>.....	378
Beispiel.....	364	Aufzählung.....	398
Syntax.....	364	BIT.....	382
DATE_TO.....	340	Enumeration.....	398
DATE-Konstante.....	363	ganzzahlig.....	375
Beispiel.....	364	LTIME.....	381
Syntax.....	364	Meldung	430
Datei		REAL / LREAL.....	375
an Objekt knüpfen.....	756	REFERENCE.....	384
speichern.....	65	STRUCT.....	396
speichern unter.....	65	UNION.....	377
von/zu Steuerung kopieren.....	233	WSTRING.....	382
Dateien		Datentyp __VECTOR	
Geräteeditor.....	525	Beispiel.....	391
Dateien für Textlisten ohne Pfad registrieren....	779	Optimale Vektorgroße bestimmen.....	392
Daten mit persistenten Variablen erhalten		Syntax.....	391
Mechanismus beim Download.....	148	Syntax bei Indexzugriff.....	391
Datenaustausch.....	163	Datentyp 'ARRAY'.....	386
Datenerfassung		Array mit fester Länge.....	386
Trace.....	216		
Datenfluss, Pragma.....	407		
Datenpersistenz.....	144		
Änderung einer bestehenden Deklaration in der Persistenten Variablenliste.....	149		
Daten mit persistenten Variablen erhalten.....	148		
Daten retten über den Rezepturverwalter....	149		

Index

- Array mit variabler Länge..... 390
- Array von Arrays..... 389
- Beispiele..... 386, 387, 388, 389
- Syntax bei Datenzugriff..... 386
- Dauerhafte Farbbereiche..... 1000, 1004, 1009, 1013
- Deaktivieren
 - Symbolbibliothek..... 810
- Debug
 - Ablaufkontrolle..... 201
 - CFC-Editor..... 293
 - Core Dump..... 224
- Deinstallieren
 - Bibliothek..... 676
- Deklaration
 - Gehe zu..... 136
- Deklarationseditor..... 87, 239
 - Deklarationskopf editieren..... 720
 - Option..... 766
 - tabellarisch/textuell..... 87
- Deklariieren..... 85
 - Array..... 89
 - automatisch..... 775
 - globale Variable..... 90
 - Kurzformmodus..... 113
- DeleteRecipe..... 791
- DeleteRecipeFile..... 791
- Dereferenzierung..... 383
- devdesc.xml-Datei..... 72
- Dezimalzahl..... 365
- Diagnose
 - Core Dump..... 224
 - SPS-Shell..... 226
- Diagnosevisualisierung..... 1061
 - Einschränkungen..... 1061
- Dialog
 - 'Monitoringbereich'..... 748
 - 'Neuer Haltepunkt'..... 745
 - Eigenschaften Haltepunkt..... 745
 - Eigenschaften Textliste..... 765
 - globaler, Visualisierung..... 1069
 - Mehrfacher Download..... 660
 - öffnen, Visualisierung..... 849
 - Optionen..... 765
 - Optionen 'Geräteeditor'..... 767
 - schließen, Visualisierung..... 849
 - Tracekonfiguration..... 783
- Dialog 'Eigenschaften'
 - Registerkarte 'Optionen Applikationserzeugung'..... 751
- Dialoge..... 744
 - Funktionsbaustein auswählen..... 744
- Dialoge in Visualisierung..... 1064, 1070
- Dialogmanager..... 1064, 1070
- Dialogvisualisierung..... 1064, 1070
- DINT..... 375
- displaymode, Pragma..... 408
- DIV..... 320
- DO..... 247, 249
- Dokumentation
 - Änderungsverlauf..... 1
- Download
 - ausführen..... 230
- Download beim Einloggen..... 230
- Download im eingeloggten Zustand (Online-Betrieb)..... 231
- Drag-and-drop einer Variable auf ein Element..... 868
- Drehschalter
 - Visualisierungselement..... 1029
- Drückschalter
 - LED, Visualisierungselement..... 1026
 - Visualisierungselement..... 1026
- DT..... 376
- Dualzahl 365
- dum-Datei..... 539
- DUT 517
 - hinzufügen..... 518
- DWORD..... 375
- Dynamische Farbdarstellung
 - Rahmen- und Füllfarbe, Visualisierungselementeigenschaft..... 874, 902, 967
- dynamische Texte.... 873, 882, 892, 901, 911, 923, 945, 956, 974
- E**
 - E/A-Abbild..... 76
 - Adresse ändern..... 81
 - Aktualisierung..... 83
 - alle Geräte..... 84
 - bearbeiten..... 647
 - exportieren in CSV..... 648
 - Geräteeditor..... 534
 - importieren aus CSV..... 648
 - Monitoring..... 83
 - Taskaufstellung..... 204
 - Verknüpfen eines Gerätekanals mit einer Funktionsbaustein-Instanz..... 80
 - Vorgehen..... 77
 - Edit-Control in Visualisierung..... 1074
 - Editiermodus
 - grafischer Editor..... 240
 - Eigenschaften
 - Allgemein..... 748
 - AS..... 756
 - Bibliotheksverwalter..... 716
 - Bildersammlung..... 757
 - Bitmap..... 751
 - Bootapplikation..... 748
 - CFC..... 753
 - eines Objekts..... 748
 - Gerät..... 757
 - Monitoring..... 757
 - Netzwerkeinstellungen'..... 753
 - Netzwerkvariablen..... 751

Taskkonfiguration.....	598	Einrückung	
Übersetzen.....	749	Texteditor.....	621
Verknüpfung mit Datei.....	756	Element	
Visualisierungsobjekt.....	1063	Bild.....	906
Zugriffskontrolle.....	750	Frame.....	917
Ein- und Ausgabeveriablen.....	301	Polygon / Linienzug / Bézierkurve.....	887
Eindimensionales Array von 10 Integerelementen		Elemente für Trace-Kontrolle einfügen.....	822
Beispiel.....	386	Elementeigenschaften.....	632
eine Farbe benutzen.....	862	Ansicht, Visualisierung.....	829
Einfügen		AS.....	270
Anweisungszeile.....	709	Elementliste.....	813
Baustein, FUP/KOP/AWL.....	703	Elementname.....	870, 879, 888, 897, 907, 919,
Bausteineingang, FUP/KOP.....	705	929, 932, 935, 938, 941, 950, 953, 964, 971, 976,	
Bilddatei.....	719	979, 983, 987, 989, 991, 993, 998, 1002, 1006,	
Kontakt.....	706, 708, 709	1010, 1015, 1020, 1024, 1027, 1030, 1034, 1050,	
Kontakt parallel oberhalb.....	707	1052	
Kontakt parallel unterhalb.....	707	ELSE.....	246, 248
Kontakt rechts.....	706	ELSIF.....	246
leerer Baustein.....	704	EN/ENO	
leerer Baustein mit EN/ENO.....	704	FUP/KOP/AWL.....	283
Leitungsverzweigung.....	710, 711	hinzufügen.....	689
negierter Kontakt.....	708	END_ACTION.....	454
Netzwerk.....	702	END_CASE.....	248
Return, FUP/KOP/AWL.....	705	END_FOR.....	247
Set-Spule.....	706	END_FUNCTION_BLOCK.....	454
Sprung, FUP/KOP/AWL.....	705	END_IF.....	246
Sprungmarke, FUP/KOP/AWL.....	705	END_PROGRAM.....	454
Spule.....	706	END_REPEAT.....	250
Text in Textliste.....	729	END_VAR.....	300
Zuweisung, FUP/KOP/AWL.....	703	END_WHILE.....	249
Einfügen danach.....	688	ENDTRY'.....	353
Einfügeposition auswählen.....	72	Energieschalter	
Eingabedialoge.....	1064	Visualisierungselement.....	1026
Eingabehilfe.....	112	Entfernen	
Bilder.....	915	Reset.....	690
Dialog.....	625	Set.....	690
Kategorien.....	625	Enumeration.....	398
Option.....	775	Datentyp.....	398
Textsuche.....	625	Deklaration.....	398
Eingabekonfiguration Hotkey.....	878, 886, 896,	Deklaration und Initialisierung von Enu-	
905, 915, 928, 948, 969, 982, 985		merationsvariablen.....	399
Eingabekonfiguration Tasten... 877, 885, 895, 904,		Eindeutiger Zugriff auf Enumerations-	
927, 948, 969, 984		komponenten.....	400
Eingabekonfiguration Umschalten... 877, 886, 895,		Enumeration mit explizitem Basisdaten-	
904, 927, 948, 969, 985		typ.....	398
Eingabeunterstützung		Initialisierung.....	398
Komponenten auflisten.....	112	Namensraum.....	361
Kurzformmodus.....	113	Standardwert.....	398
Smart-Tag-Funktionen.....	114	Strikte Programmierregeln.....	399
Eingabeveriable.....	300	EQ.....	332
Eingang		Ereignis.... 878, 886, 896, 905, 915, 928, 948, 969,	
CFC.....	297	982, 985	
FUP/KOP/AWL.....	283	Ereignis in Visualisierung bemerken.....	1074
Gerät.....	77	Ersetzen.....	138
Eingangsaktion.....	265	Erstellen	
Eingangsspeicher.....	371	Bildersammlung für Symbolbibliothek.....	811
Einheit..... 995, 999, 1003, 1008, 1012, 1016		Symbolbibliothek.....	811
Einheitenverschiebung.....	1008, 1012	Textliste für Symbolbibliothek.....	812

Index

Erweiterter Strukturierter Text.....	105	Frameparameter	
estimated-stack-usage, Pragmaattribut.....	409	deklarieren, Visualisierung.....	831
Exception		Füllart.....	871, 889, 899, 965, 1053
Ausführung anhalten.....	743	Fülleigenschaften.....	875, 894, 903
im IEC-Code abfangen.....	353	Füllfarbe.....	871, 889, 893, 898, 965
Exceptionhandling		Funktion.....	558
Visualisierung.....	804	Aufruf.....	558
Execute.....	285	Aufruf über Ereignis.....	599
EXIT.....	251	mit externer Implementierung aufrufen.....	111
EXP.....	344	Übergabe per POINTER in VAR_INPUT.....	560
ExpandFully, Pragma.....	412	Übergabe per REFERENCE TO in VAR_IN-	
Export/Import		PUT.....	561
E/A-Abbild.....	648	Übergabe von Instanzdaten per VAR_IN-	
Textliste.....	729	PUT.....	560
EXPT.....	344	Funktion als Operand.....	373
ExST.....	105	Funktionalitäten	
Externe Implementierung.....	111	IEC 61131-3 Programmierplattform.....	1
Externe Variable.....	307	Funktionsbaustein.....	556
F		aufrufen, ST.....	252
FALSE.....	362	erweitern.....	155
FALSE tasten.....	877, 885, 895, 904, 927, 948,	mit externer Implementierung aufrufen.....	111
	969, 984	nicht verwendete Parameter, FUP/KOP.....	712
Farbbereiche.....	1000, 1004, 1009, 1013	Funktionsbausteintypen.....	374
Farbe.....	821, 862, 880, 909, 920, 941, 972, 1000,	Funktionsplan.....	92
	1004, 1006, 1009, 1011, 1013, 1016	Funktionspointer.....	383
Farbe Schriftart....	943, 951, 995, 999, 1003, 1008,	Funktionszeiger.....	383
	1012, 1016	FUP.....	92
Farben.....	871, 880, 889, 893, 898, 909, 920, 941,	konvertieren.....	713
	965, 972	Option.....	768
FarbeSchriftart.....	872, 881, 890, 899, 909, 921,	FUP/KOP/AWL	
	930, 955, 966, 973	Befehl 'Parallelen Modus wechseln'.....	708
Farbverlauf.....	862	FUP/KOP/AWL-Editor.....	272
Farbverlauf verwenden.....	871, 889, 899, 942	Leitungsverzweigung.....	711
Farbverlaufeditor.....	862	Leitungsverzweigung KOP.....	711
Farbverlaufsauswahl.....	871, 889, 899, 942	nicht verwendete Parameter, FUP/KOP.....	712
Fehleranalyse.....	224	Online-Betrieb.....	277
FINALLY.....	353	Symboleiste.....	240
Flankenerkennung.....	710	G	
Floatingpoint-Exceptions.....	529	Ganzzahliger Datentyp.....	375
FOR.....	247	konvertieren.....	337
Forcen.....	198	Gateway.....	164
Alle Forces anzeigen.....	199	GE.....	331
Formatieren		Gefahrenhinweise.....	2
Visualisierungstext.....	846	Gehe zu	
Fortschrittsbalken		Definition.....	136
Visualisierungselement.....	986	Deklaration.....	136
Frame		Netzwerk.....	714
konfigurieren.....	819	Gerät.....	519
Parameter aktualisieren.....	833	Applikation.....	521
umschalten, Visualisierung.....	849	Benutzerverwaltung.....	539, 540
Visualisierung auswählen.....	819	Dateien.....	525
Visualisierungselement.....	917	E/A-Abbild.....	534
Frame-Parameter in der geöffneten Visuali-		E/As abbilden.....	77
sierung aktualisieren.....	822	Eigenschaften.....	757
Frameelement		Interaktives Einloggen.....	757
programmieren.....	832	Konfiguration.....	519, 520

konfigurieren.....	75
Log.....	525
Optionen.....	757
Projektextplorer.....	71
Sichern und Wiederherstellen.....	522
Simulationsbetrieb.....	193, 741
SPS-Einstellungen.....	527
SPS-Shell.....	226
Gerät einfügen.....	72
Geräte	
Ansicht.....	71
Geräte-Editor	
Gleitkomma-Ausnahmen im SPS-Pro-	
gramm.....	529
Parameter.....	529
Geräte-Repository.....	72
Gerätebeschreibung.....	72
Geräteeditor	
Applikationen.....	521, 522
Benutzer und Gruppen.....	539
Dateien.....	525
E/A-Abbild.....	534
generisch.....	519
IEC-Objekte.....	537
Information.....	543
Log.....	525
Registerkarte 'Parameter'.....	520
SPS-Einstellungen.....	527
SPS-Shell.....	532
Status.....	543
Synchronisierte Dateien.....	524
Task-Aufstellung.....	542
Zugriffsrechte.....	540
Geschlossene Leitungsverzweigung, KOP.....	286
Get.....	568
GetLastError.....	799
GetRecipeCount.....	796
GetRecipeNames.....	796
GetRecipeValues.....	797
GetRecipeVariableNames.....	798
getypte Konstanten.....	369
Gleitkomma-Ausnahmen im SPS-Programm...	529
global_init_slot, Pragma.....	413
Globale Netzwerkvariablen-Liste (Empfän-	
ger).....	552
hinzufügen.....	552
Globale Netzwerkvariablen-Liste (Sender).....	551
hinzufügen.....	551
Globale Netzwerkvariablenliste.....	168
Globale Textliste	
bearbeiten.....	119
Datei importieren.....	120
exportieren.....	119
für statische Anwendung.....	122
IDs aktualisieren.....	122
mit Ersetzungsdatei bearbeiten.....	123
prüfen.....	122
Text erstellen.....	122
Textlisteneinträge entfernen.....	123
vergleichen und Unterschied exportieren....	121
Globale Variable.....	306
deklarieren.....	90
globale Variable in Diagnosevisualisierung....	1061
Globale Variablenliste.....	544
Namensraum.....	360
XML-Export.....	648
XML-Import.....	648
Globaler Namensraum-Operator.....	359
GlobalImagePool.....	126
GlobalTextList	
anlegen.....	727
Export/Import.....	729
ID aktualisieren.....	731
ID prüfen.....	731
IDprüfen.....	731, 732
nicht verwendete Einträge entfernen.....	731
Objekt.....	543
Sprache einfügen.....	727
Sprache entfernen.....	730
GlobalTextList füllen.....	118
GlobalTextList, siehe globale Textliste.....	117
Grafische Sprache	
konvertieren.....	713
Grafischer Editor Symbolleiste.....	240
Größe an Raster anpassen.....	818
Größe gleichmachen.....	818
Großschreibung	
Schlüsselwort.....	775
Groupbox	
Visualisierungselement.....	949
Gruppe	539
Benutzerverwaltung.....	57
Gruppieren.....	819
Gruppierung	
entfernen.....	699
erzeugen.....	699
Gruppierung aufheben.....	819
GT.....	330
Gültigkeitsbereiche von Objekten.....	72
GVL.....	544
Namensraum.....	360
H	
Halbtransparenz siehe Semitransparenz.....	863
Haltepunkt	
Bedingung festlegen.....	195
Eigenschaften.....	745
setzen.....	194
Verwenden von Haltepunkten.....	193
Hardware scannen.....	72
hasattribute.....	439
hastype.....	439
hasvalue.....	439
Hauptaktion.....	265
Hauptskala.....	999, 1003, 1007, 1011

Index

Hebelschalter		
Visualisierungselement.....	1026	
Helpdesk.....	1083	
Hexadezimalzahl.....	365	
hide_all_locals, Pragma.....	415	
hide, Pragma.....	414	
Hintergrund.....	821, 1010	
Hintergrundbild.....	1010	
Hinzufügen.....	643	
Ausgangsanschluss.....	697	
Eingangsanschluss.....	697	
EN/ENO.....	689	
Histogramm		
konfigurieren.....	1017	
Visualisierungselement.....	1013	
Höhe.....	870, 898, 908, 920, 930, 932, 935, 939, 941, 950, 955, 964, 971, 976, 979, 983, 987, 989, 991, 993, 998, 1002, 1006, 1010, 1015, 1021, 1024, 1027, 1030, 1034, 1051, 1053	
Höhe der Beschriftung.....	999, 1003, 1008, 1012	
Höhe der Beschriftungen.....	995, 1016	
Höhe gleichmachen.....	817	
Horizontal zentrieren.....	814	
horizontale Ausrichtung... ..	872, 881, 890, 899, 909, 921, 930, 955, 966, 973	
Horizontalen Abstand entfernen.....	815	
Horizontalen Abstand gleichmachen.....	814	
Horizontalen Abstand vergrößern.....	815	
Horizontalen Abstand verkleinern.....	815	
Hotkey.....	878, 886, 896, 905, 915, 928, 948, 969, 982, 985	
Hotline.....	1083	
I		
I.....	371	
IEC 61131-3 Programmierplattform		
Funktionalitäten.....	1	
IF.....	246	
Im CFC-Editor programmieren.....	100	
Darstellung eines Bausteins reduzieren.....	104	
Elemente einfügen und mit Verbindungslinien verschalten.....	100	
Kollisionen lösen und Verbindungslinien durch Kontrollpunkte fixieren.....	101	
Verbindungsmarken erzeugen.....	101	
Implementierung von FB_Something mit Methode FB_Init		
Beispiel.....	389	
Import		
E/A-Abbild.....	648	
INDEXOF.....	322	
IndraLogic		
Projekt Know-how-Schutz.....	57	
Projekt Schreibschutz.....	57	
IndraLogic 1x-Konverter		
Option.....	769	
IndraLogic 2G-Visualisierung.....	803	
IndraLogic-Gateway.....	164	
info.cfg-Datei.....	72	
Information		
Geräteeditor.....	543	
Inhaltsoperator.....	333	
init_namespace, Pragma.....	416	
init_on_onlchange, Pragma.....	416	
init-Schritt.....	682	
Initialisierung.....	86	
ARRAY.....	386	
Eingangsvariable, Pragma.....	415	
Namensraum, Pragma.....	416	
Online Change, Pragma.....	416	
vermeiden, Pragma.....	426	
initialize_on_call, Pragma.....	415	
Initialschritt.....	682	
Innerer Rahmen.....	999, 1003, 1007, 1011	
Installieren		
Bibliothek.....	676	
instance-path, Pragma.....	417	
Instanzvariablen.....	307	
INT.....	375	
Intelligentes Codieren		
Option.....	775	
INTERFACE.....	562	
Interface property.....	568	
INTERNAL.....	564	
io_function_block_mapping, Pragma.....	418	
io_function_block, Pragma.....	418	
J		
JMP.....	251	
K		
Kennzeichen		
in Visualisierungswerkzeuge konfigurieren.....	842	
Visualisierungselement.....	806	
Keypad		
Visualisierung.....	1059	
keyword.....	454	
Kippschalter		
Visualisierungselement.....	1026	
Kodierregeln.....	132	
Kommentar		
CFC.....	299	
ST.....	252	
Kompatibilität		
Bibliothek.....	681	
Kompilierung		
bedingt.....	439	
Komponenten auflisten.....	112	
Kompositor		
CFC.....	298	
Konfiguration		
Gerät.....	75	
Geräteeditor.....	520	

Konfigurationsvariable.....	311
Konstante.....	362
getypt.....	369
LTIME.....	363
REAL/LREAL.....	366
STRING.....	367
TIME.....	362
Zahlen.....	365
Konstante Variablen	
VAR CONSTANT.....	308
Konstanten.....	361
WSTRING.....	368
Kontakt.....	285
einfügen.....	708, 709
Kontakt einfügen	
links.....	706
negiert.....	708
negiert, parallel unterhalb.....	708
parallel oberhalb.....	707
parallel unterhalb.....	707
rechts.....	706
Kontaktplan.....	93
Kontrollpunkt	
CFC.....	297
entfernen.....	698
erzeugen.....	698
Konvertieren	
BOOL_TO.....	335
DATE_TO.....	340
ganzzahliger Datentyp.....	337
LREAL_TO.....	338
nach AWL.....	713
nach FUP.....	713
nach KOP.....	713
REAL_TO.....	338
STRING_TO.....	341
TIME_OF_DAY_TO.....	339
TIME_TO.....	339
TO_<xxx>.....	337
TO_BOOL.....	336
TRUNC.....	342
TRUNC_INT.....	342
KOP.....	93
konvertieren.....	713
Option.....	768
KOP-Editor	
Online-Betrieb.....	277
Kreisektor	
Visualisierungselement.....	896
Kurzformmodus.....	113
L	
L.....	371
Laden auf Steuerung.....	230
Lampe	
Visualisierungselement.....	1023
Laufzeitsystemdateien	
erzeugen	650
Laufzeitsystemdienst	
deaktivieren.....	227
LE.....	331
Leerer Baustein	
einfügen.....	704
Leerer Baustein mit EN/ENO	704
Leistungsverzweigung.....	711
geschlossen.....	286
offen.....	284
Leistungsverzweigung Start/Ende.....	286
Leistungsverzweigung, AWL	
Element in FUP/KOP/AWL.....	284
Lesezeichen	
setzen und verwenden.....	137
Springen zu Lesezeichen innerhalb einer POU.....	138
Library Development Summary.....	789
LIMIT.....	329
Linienart... 871, 875, 880, 889, 894, 899, 903, 909, 913, 921, 925, 965, 1053	
Linienstärke..... 871, 875, 880, 889, 894, 899, 903, 909, 913, 921, 925, 965, 1053	
Linienzug	
Visualisierungselement.....	887
linkalways, Pragma.....	419
Links ausrichten.....	814
LINT.....	132, 375
Liste neu ordnen.....	721
LN.....	343
LoadAndWriteRecipe.....	791
LoadFromAndWriteRecipe.....	792
LoadRecipe.....	793
Log	
SPS.....	224
LOG.....	344
Log der SPS	
Geräteeditor.....	525
Logbuch	
SPS.....	224
Login-Dialog in Visualisierung.....	1064, 1070
Lokale Variable.....	300
Look-and-Feel siehe Erscheinungsbild.....	1080
LREAL.....	375
LREAL_TO.....	338
LT.....	331
LTIME.....	381
Beispiel.....	363
Syntax.....	363
Lücken bereinigen.....	721
Lupenwerkzeug.....	240
LWORD.....	375
M	
M.....	371
M4-Schnittstellendatei für externe Bibliothek....	650
Makro.....	269

Index

anzeigen.....	688	Standardbefehle.....	610
einfügen.....	687	Menü 'Bearbeiten'	
einfügen danach.....	688	Befehl 'Variable deklarieren'.....	625
verlassen.....	688	Menü 'Befehle'	
Mapping siehe E/A-Abbild.....	77	Befehl 'Gehe zu Instanz'.....	616
Marke		Befehl 'Gehe zu Referenz'.....	615
CFC.....	298	Befehl 'Gehe zur Definition'.....	615
MAX.....	329	Befehl 'Vorheriges Lesezeichen'.....	614
Max. Breite der Beschriftung.....	999, 1003, 1008, 1012	Menü 'Debuggen'.....	652, 673
Max. Breite der Beschriftungen.....	995, 1016	Aktuellen Gerätebenutzer ausloggen.....	656
Mehrsprachige SPS-Kommentare.....	67	Alle Forces anzeigen.....	668
Meldungspragma.....	402	Alle Forces zur Überwachungsliste hinzu-	
Menü 'Ansicht'.....	628	fügen.....	668
Befehl 'Speicher'.....	633	Ansicht 'Haltepunkte' öffnen.....	664
Befehlsgruppe 'Andere Fenster'.....	628	Anzeigemodus umschalten (Binär, Dezi-	
Bibliothek.....	628	mal, Hexadezimal).....	676
Projektextplorer.....	628	Applikation laden.....	656
Menü 'Bearbeiten'		Applikation starten.....	659
Befehl 'Abwärts suchen (Auswahl)'.....	613	Applikation stoppen.....	659
Befehl 'Abwärts Suchen'.....	613	Aufrufliste öffnen.....	667
Befehl 'Alles erweitern'.....	622	Ausführen bis Cursor.....	670
Befehl 'Alles reduzieren'.....	621	Ausführen bis Rücksprung.....	669
Befehl 'Aufrufbaum ausgeben'.....	616	Ausloggen aus Applikation.....	655
Befehl 'Aufwärts suchen (Auswahl)'.....	613	Betriebsmodus umschalten.....	673
Befehl 'Aufwärts suchen'.....	613	Bootapplikation erzeugen.....	656
Befehl 'Betreffende Anschlüsse aktuali-		Core Dump Befehle.....	674
sieren'.....	620	Core Dump erzeugen.....	674
Befehl 'Bis korrespondierende Klammer		Core Dump laden.....	675
markieren'.....	624	Core Dump schließen.....	675
Befehl 'Datei als Text einfügen'.....	615	Dialog 'Eigenschaften Haltepunkt' öffnen....	666
Befehl 'Eingabehilfe'.....	624	Dialog 'Mehrfacher Download'.....	660
Befehl 'Ersetzen'.....	612	Einloggen auf Applikation.....	652
Befehl 'Gehe zu'.....	623	Einzelschritt durchführen.....	669
Befehl 'In Großbuchstaben umwandeln'.....	623	Einzelzyklus durchführen.....	659
Befehl 'In Kleinbuchstaben umwandeln'.....	624	Forcen für alle Werte aufheben.....	672
Befehl 'Korrespondierende Klammer'.....	624	Geräte-Logbuch vom Core Dump laden.....	675
Befehl 'Leerzeichen anzeigen'.....	621	Haltepunkt aktivieren.....	666
Befehl 'Lesezeichen löschen'.....	614	Haltepunkt bearbeiten.....	666
Befehl 'Lesezeichen umschalten'.....	614	Haltepunkt deaktivieren.....	666
Befehl 'Nächstes Lesezeichen'.....	614	Haltepunkt umschalten.....	666
Befehl 'Querverweise ausgeben'.....	616	Mehrfacher Download durchführen.....	659
Befehl 'Refactoring Umbenennen'.....	617	Nächste Anweisung anzeigen.....	670
Befehl 'Selektierte Zeilen auskommentie-		Online-Change durchführen.....	657
ren'.....	622	Prozedurschritt durchführen.....	669
Befehl 'Selektierte Zeilen Auskommentie-		Reset Kalt durchführen.....	662
rung entfernen'.....	623	Reset Ursprung durchführen.....	663
Befehl 'Suchen'.....	610	Reset warm durchführen.....	661
Befehl 'Variable deklarieren'.....	625	Überwachungsliste öffnen.....	667
Befehl 'Variable entfernen...'.....	619	Werte forcen.....	671
Befehl 'Variable hinzufügen'.....	618	Werte schreiben.....	672
Befehl 'Variablen neu ordnen...'.....	620	Menü 'Debugging'	
Befehlsgruppe 'Durchsuchen'.....	615	Nächste Anweisung festlegen.....	670
Befehlsgruppe 'Erweitert'.....	621	Menü 'Erstellen'.....	649
Befehlsgruppe 'Lesezeichen'.....	614	Menü 'Extras'.....	676
Befehlsgruppe 'Refactoring'.....	617	Bibliotheken.....	676
Befehlsgruppe 'Standardbefehle'.....	610	Menübefehle.....	610
Befehlsgruppe 'Suchen und Ersetzen'.....	610	Datei als Text einfügen.....	615
		Durchsuchen.....	615

Eingabehilfe.....	624	SFC.....	253
Erweitert.....	621	Überwachungsintervall.....	757
Lesezeichen.....	614	MOVE.....	322
Menü 'Ansicht'.....	628	MUL.....	319
Menü 'AS'.....	682	Multitouch	
Menü 'Bearbeiten'.....	610	Einstellung.....	1060
Menü 'Bibliothek'.....	714	MUX.....	330
Menü 'Bildersammlung'.....	719		
Menü 'CFC'.....	689	N	
Menü 'Deklarationen'.....	720	Nach hinten legen	819
Menü 'Deklarationen' (Persistenz).....	721	Nach oben verschieben.....	720
Menü 'FUP/KOP/AWL'.....	702	Nach unten verschieben.....	720
Menü 'Projekt'.....	634	nach vorn legen.....	818
Menü 'Rezepturen'.....	723	Namensraum	
Menü 'Textliste'.....	727	automatisch.....	775
Menü 'Trace'.....	732	Bibliothek.....	360
Refactoring.....	617	Enumeration.....	361
Sonstige.....	741	GVL.....	360
Variable deklarieren.....	625	namespace.....	423
Merkerspeicher.....	371	NE.....	332
Metainformation		Negation.....	709
zum Projekt eingeben.....	12	Negieren.....	689
Methode.....	307, 564	Negierter Kontakt.....	285
'FB_Exit'.....	455	Netzwerk.....	163
'FB_Init'.....	455	auskommentieren.....	703
'FB_Reinit'.....	455	einfügen.....	702
Aufruf.....	158	FUP/KOP/AWL.....	282
hinzufügen.....	564	springen.....	714
Methode rekursiv aufrufen.....	160	Topologie.....	163
Methodenaufruf als virtueller Funktions- aufruf.....	159	unterhalb einfügen.....	703
mit externer Implementierung aufrufen.....	111	Netzwerkeinstellungen'	
Schnittstelle.....	567	Eigenschaften.....	753
Methodenaufruf.....	158	Netzwerkvariablen.....	168
MIN.....	329	Eigenschaften.....	751
Mitte X.....	862	NEW.....	349
Mitte Y.....	862	no_assign, Pragma.....	424
MOD.....	321	no_check, Pragma.....	424
Modifikator in AWL.....	278	no_copy, Pragmaattribut.....	425
Monitoring.....	205	no_instance_in_retain, Pragma.....	426
Array teilweise monitoren.....	208	no_virtual_actions, Pragma.....	426
AS.....	253	no-exit, Pragma.....	425
CFC-Editor.....	293	noinit, Pragma.....	426
Eigenschaft.....	419	Normalzustand.....	871, 889, 893, 898, 965
Eigenschaft monitoren.....	209	NOT.....	323
Eigenschaftenzugriff im übergeordneten Programmierobjekt monitoren.....	209	NOT, Operator in Pragma.....	444
Funktion monitoren.....	210	Numpad	
Funktionsaufruf.....	419	Visualisierung.....	1059
Funktionsbaustein monitoren.....	208	NVL (Empfänger).....	552
Im Deklarationseditor monitoren.....	206	NVL (Sender).....	551
In der Implementierung monitoren (Inline- Monitoring).....	206	O	
Methode monitoren.....	210	Oben ausrichten.....	814
Monitoringbereich begrenzen.....	208	Objekt	
Pragma.....	419	(offline) bearbeiten.....	646
Rückgabewert eines Funktionsaufrufs monitoren.....	210	Aktion.....	574
		Allgemeiner Bausteinteil.....	507
		Bibliotheksinfo.....	579

Index

Eigenschaften.....	748	FUP, KOP und AWL.....	768
hinzufügen, Befehl.....	643	IndraLogic 1x-Konverter.....	769
Textliste.....	588	Intelligentes Codieren.....	775
Transition.....	576	Sichere FUP-Optionen.....	772
Verknüpfung mit Datei.....	756	Syntaxhervorhebung.....	776
Zugriffsrecht.....	59	Texteditor, ST-Editor.....	776
Objekt 'Eigenschaft'.....	571	Visualisierung.....	1079
Objekte hinzufügen.....	8	Visualisierungen.....	779
obsolete, Pragma.....	430	Visualisierungsstile.....	781
OF.....	248	OR.....	323
Oktalzahl.....	365	OR_ELSE.....	325
Online Change.....		OR, Pragma.....	445
ausführen.....	229	Ordner hinzufügen.....	644
Online Change beim Einloggen.....	229	ORN.....	278
Online Change im eingeloggten Zustand (Online-Betrieb).....	230		
Online-Betrieb.....		P	
Tasküberwachung.....	601	pack_mode, Pragma.....	431
Online-Vergleich von SPS-Objekten.....	51	Parallel.....	684
Operand.....	375	Parameter.....	
Operator.....		aktualisieren.....	712
_COMPARE_AND_SWAP.....	357	bearbeiten.....	694
_CURRENTTASK.....	357	Parameterliste.....	
_DELETE.....	347	Konfigurierbare Konstanten in Bibliothe-	
_NEW.....	349	ken.....	545
_VARINFO.....	354	parameterstringof, Pragma.....	838
_VCADD.....	392	PARAMS.....	454
_VCDIV.....	393	Passwort.....	
_VCDOT.....	393	Passwort-Manager.....	62
_VCLOAD_LREAL.....	395	Passwort-Manager.....	57, 62
_VCLOAD_REAL.....	395	PERSISTENT.....	
_VCMAX.....	394	Deklaration im Funktionsbaustein FB_A.....	309
_VCMUL.....	393	Deklaration im Programm PLC_PRG.....	309
_VCSET_LREAL.....	394	Deklaration in der persistenten Variablen-	
_VCSET_REAL.....	394	liste PersistentVars.....	309
_VCSQRT.....	394	Mögliche Deklarationsstellen.....	309
_VCSTORE.....	396	Syntax der Deklaration.....	308
_VCSUB.....	392	Syntax der Deklaration in POU.....	308
_XADD.....	358	Persistente Variable.....	144
AWL.....	278	Daten mit Rezepturen erhalten.....	151
Bindungsstärke.....	242	deklarieren.....	151
INI.....	361	PERSISTENT.....	308
Rangfolge.....	242	Persistente Variablenliste anlegen und	
SEL.....	328	persistente Variable deklarieren.....	151
TEST_AND_SET.....	359	speichern in Rezeptur.....	152
Operator __NEW.....		Persistente Variablenliste.....	546
Anwendungsbeispiele.....	350	Persistenz.....	144
Operator ' __NEW'.....		Lebensdauer von Variablen bei Aufruf	
Syntax.....	349	von Online-Befehlen.....	147
Operator 'SEL'.....	328	Lebensdauer von Variablen beim Laden	
Operatoren.....	315	eines Bootprojekts.....	147
Überlauf/Unterlauf im Datentyp.....	315	Liste neu ordnen.....	721
Optionen.....		Lücken bereinigen.....	721
Allgemeine Einstellungen.....	765	Mechanismen im Vergleich.....	145
AS-Editor.....	773	Mechanismen zur Datenerhaltung.....	144
Bibliotheken.....	771	Rezeptur.....	721
CFC-Editor.....	765	Variablenkonfiguration.....	722
Deklarationseditor.....	766	Persistenzeditor.....	546

Pfad für Bilddateien.....	779	Information.....	579
Pfad für Dateien für Textlisten.....	779	Information abfragen.....	12
pingroup, Pragma.....	429	Metainformation.....	580
Pingruppe erweitern/reduzieren.....	429	Ordner hinzufügen.....	644
Platzhalter		POUs für Schlüssel.....	581
Bibliotheksverwalter.....	718	Projekteinstellungen.....	14
PlcOperationControl.....	228	Rechteverwaltung.....	59
Pointer		Schlüssel für Metainformation.....	581
SUPER.....	312	Schlüssel VisuSymbolLibrary.....	581
THIS.....	313	Schutz.....	57
POINTER.....	383	speichern.....	65
POINTER in VAR_INPUT.....	560	speichern unter.....	65
Polygon		Statistik der Objekte.....	583
Visualisierungselement.....	887	Symbolbibliothek auswählen.....	810
Position.... 870, 898, 908, 920, 930, 932, 935, 939,		transferieren.....	15
941, 950, 955, 964, 971, 976, 979, 983, 987, 989,		Übertragen.....	15
991, 993, 998, 1002, 1006, 1010, 1015, 1021,			
1024, 1027, 1030, 1034, 1051, 1053		Projektdatei	
Positionierung.....	1008, 1012	Zugangsdaten.....	62
Potentiometer		Projekteinstellung	
Visualisierungselement.....	1009	AS.....	758
POU.....	552	Visualisierung.....	763
hinzufügen.....	553	Projekteinstellungen.....	579
Typ ändern.....	720	Befehl.....	634
POU für implizite Prüfung.....	153	Benutzerverwaltung.....	61
POU hinzufügen		vornehmen.....	14
Dialog.....	553	Projektexplorer.....	71
Pragma		Projektexplorer ein-/ausschalten.....	628
Attribut 'pin_presentation_order_inputs/		Regeln.....	72
outputs'.....	429	PROPERTY.....	571
Attribute.....	403	PROTECTED.....	564
bedingt.....	439	PUBLIC.....	564
define.....	439		
hasattribute.....	439	Q	
hastype.....	439	Q.....	371
hasvalue.....	439	qualified_only, Pragma.....	434
if.....	439	Qualifizierer für AS-Aktionen.....	256
Meldung.....	402	Quellposition anzeigen.....	630
Region-Pragma.....	445	Querverweis	
Statische Analyse.....	133	automatisch aktualisieren.....	775
undefine.....	439	Querverweise für eine konkrete Symboldek-	
verwenden.....	114	laration.....	135
verwenden im Visualisierungsschnittstel-		Querverweise in der klassischen Ansicht	
leneditor.....	831	ausgeben.....	632
PRIVATE.....	564	Querverweise über Textsuche nach dem	
Profil		Symbolnamen.....	135
Visualisierung.....	1080	Querverweisliste.....	628
PROGRAM.....	554	Ansicht.....	628
Programmieren		Ergebnisse auf aktuelle Deklaration be-	
Frameelement.....	832	schränken.....	743
Projekt.....	7	Verwendungsstelle.....	135
Ablage.....	59		
anlegen.....	8	R	
Benutzerverwaltung.....	61	Radiobutton	
Dateiinformation.....	580	Visualisierungselement.....	988
Eigenschaft mit Schlüssel erstellen.....	12	Rahmenfarbe..... 871, 889, 893, 898, 965	
exportieren.....	15	Rangfolge, ST.....	242
freigegeben.....	60	Raster	

Index

Visualisierung.....	780	Mögliche Deklarationsstellen.....	311
READ_ONLY.....	454	Syntax bei der Deklaration.....	310
READ_WRITE.....	454	Retain-Variable	
ReadAndSaveRecipe.....	793	RETAIN.....	310
ReadAndSaveRecipeAs.....	794	Retain-Variablen	
ReadRecipe.....	795	Daten mit Retain-Variablen erhalten.....	150
REAL.....	375	Speicherlokation von persistenten Funktionsbaustein-Instanzen.....	150
REAL_TO.....	338	Return	
REAL/LREAL-Konstante.....	366	CFC.....	298
Beispiel.....	366	einfügen, FUP/KOP/AWL.....	705
Syntax Exponentialdarstellung.....	366	FUP/KOP/AWL.....	284
Rechte		RETURN.....	251
Befehl.....	639	Rezeptur.....	587
Rechteck		CreateRecipe.....	790
abgerundet, Visualisierungselement	869	CreateRecipeNoSave.....	790
Ellipse, Visualisierungselement	869	DeleteRecipe.....	791
Visualisierungselement	869	DeleteRecipeFile.....	791
Rechteverwaltung.....	57	einfügen.....	723
Projekt.....	59	entfernen.....	723
Rechts ausrichten.....	814	erstellen.....	213
RecipeManCommands.....	789	GetLastError.....	799
Refactoring.....	139	GetRecipeaValues.....	797
Option.....	771	GetRecipeCount.....	796
Umbenennen.....	139	GetRecipeNames.....	796
Variable entfernen.....	140	GetRecipeVariableNames.....	798
Variable hinzufügen.....	140	laden.....	724
REFERENCE.....	384	laden aus Datei.....	214
REFERENCE TO in VAR_INPUT.....	561	laden und schreiben.....	725
Referenz Benutzeroberfläche.....	507	lesen.....	725
Referenz Programmierung.....	239	lesen und speichern.....	726
reflection, Pragmaattribut.....	435	LoadAndWriteRecipe.....	791
Registerkarte		LoadFromAndWriteRecipe.....	792
CFC-Ausführungsreihenfolge.....	753	LoadRecipe.....	793
Taskgruppen.....	602	ReadAndSaveRecipe.....	793
Variablenverwendung.....	601	ReadAndSaveRecipeAs.....	794
Registersteuerelement		ReadRecipe.....	795
Visualisierungselement.....	937	RecipeManCommands.....	789
Reihenfolge		ReloadRecipes.....	795
an den Anfang.....	691	ResetLastError.....	799
ans Ende.....	692	SaveRecipe.....	794
eins vor.....	692	schreiben.....	725
eins zurück.....	693	SetRecipeValues.....	798
ReloadRecipes.....	795	SetStoragePath.....	800
Reparieren		speichern.....	724
Baustein.....	712	Speicherverbrauch.....	215
REPEAT.....	250	Variable einfügen.....	723
Reset		Variable entfernen.....	726
Applikation.....	200	Werte ändern mit Rezepturen.....	212
Ausgang.....	690	Werte speichern.....	721
entfernen.....	690	WriteRecipe.....	795
Reset-Spule.....	286, 706	Rezepturdatei	
Reset-Zuweisung		laden.....	214
Mehrfachzuweisungen.....	245	Rezepturdefinition.....	587
R=, ExST.....	245	Rezepturverwalter.....	583
ResetLastError.....	799	ROL.....	326
RETAIN		ROR.....	327
In einer GVL.....	310		
In einer POU.....	310		

S

Sample		
darstellen.....	216	
in Tracedatei speichern.....	217	
SaveRecipe.....	794	
Scan der Hardware.....	72	
SCE		
KOP, Baustein umgehen.....	286	
Schaltfläche		
Visualisierungselement.....	940	
Schieberegler		
Visualisierungselement.....	975	
Schlüsseldatei.....	13	
Schlüsselwort		
__AdrInst.....	454	
__BitOffset.....	454	
__Cast.....	454	
__Copy.....	454	
__CRC.....	454	
__FCall.....	454	
__Init.....	454	
__Lazy.....	454	
__LocalOffset.....	454	
__MaxOffset.....	454	
__PropertyInfo.....	454	
__RefAdr.....	454	
__RELOC.....	454	
__SystemScope.....	454	
__TypeOf.....	454	
__VarInfo.....	454	
__Wait.....	454	
Großschreibung.....	775	
Schnittstelle.....	562	
deklarieren, Visualisierung.....	831	
einer Visualisierung editieren.....	831	
erweitern.....	158	
implementieren.....	156	
Schnittstellen-Eigenschaft.....	568	
Schnittstelleneditor.....	813	
Schnittstellenmethode.....	567	
Schreiben.....	198	
Schreibschutz.....	60	
Schriftart.....	872, 881, 890, 899, 909, 921, 930,	
943, 951, 955, 966, 973, 995, 999, 1003, 1008,		
1012, 1016		
Schritt.....	262	
Schritt-Transition		
einfügen.....	683	
einfügen danach.....	683	
Schrittstatus.....	257	
Schutz		
Projekt.....	57	
Schwenkwerkzeug.....	240	
Scrollbar		
Visualisierungselement.....	970	
Seite		
CFC.....	297	
Seitengröße		
bearbeiten.....	689	
seitenorientiert.....	291	
select all.....	821	
Selector		
CFC.....	299	
Semitransparenz		
Einstellung.....	1060	
verwenden in Visualisierung.....	863	
Service-Hotline.....	1083	
Set.....	568	
Ausgang.....	690	
entfernen.....	690	
Set-Spule.....	286	
einfügen.....	706	
Set-Zuweisung		
Mehrfachzuweisungen.....	244	
S=, ExST.....	244	
Set, ExST.....	244	
Set/Reset, FUP/KOP/AWL.....	710	
SetRecipeValues.....	798	
SetStoragePath.....	800	
SFC.....	106	
Abfragedialog zum Wählen des Duplizie-		
rungsmodus.....	110	
Aktionsqualifizierer.....	256	
Eigenschaften.....	756	
Elementeigenschaften.....	270	
Makro.....	269	
Online-Betrieb.....	253	
programmieren.....	106	
Schritt.....	262	
Sprung.....	269	
Transition.....	262	
SHL.....	325	
SHR.....	326	
Sichere FUP-Optionen		
Option.....	772	
Sicherheitshinweise.....	2	
Signalgrafik.....	2	
Signalwörter.....	2	
Signieren		
Bibliotheksprojekt.....	583	
Simulation.....	193	
Befehl.....	741	
zum Testen.....	193	
SIN.....	345	
SINT.....	375	
SIZEOF.....	322	
Skala.....	999, 1003, 1007, 1011	
Skala in 3D.....	999, 1003, 1007, 1011	
Skalenanfang.....	999, 1003, 1007, 1011	
Skalenende.....	999, 1003, 1007, 1011	
Skalenformat (C-Syntax).....	995, 999, 1003, 1008,	
1012, 1016		
Skalenlinienstärke.....	999, 1003, 1007, 1011	
Skalenstärke.....	999, 1003, 1007, 1011	
Smart Coding.....	112	

Index

Speicher			
Auszug anzeigen.....	633		
Speicheransicht.....	633		
Speicherbereich.....	371		
Speichern			
Projekt.....	59, 65		
Speicherverbrauch			
Rezeptur.....	215		
SpinControl			
Visualisierungselement.....	978		
Sprachdateien für Visualisierung.....	779		
Sprache			
aktuell eingestellte für Visualisierung.....	1059		
Sprache hinzufügen und übersetzen.....	119		
Sprache in Visualisierung.....	779		
Sprachumschaltung			
Visualisierung.....	849		
Sprung.....	269		
CFC.....	298		
einfügen.....	687		
einfügen danach.....	687		
einfügen, FUP/KOP/AWL.....	705		
Sprungmarke			
einfügen, FUP/KOP/AWL.....	705		
FUP/KOP/AWL.....	283		
ST.....	251		
SPS Log.....	224		
SPS-Compilerversion			
Befehl.....	641		
SPS-Compilerversion ändern.....	10		
SPS-Einstellungen			
Geräteeditor.....	527		
SPS-Kommunikation			
Gateway.....	164		
SPS-Lokalisierungsdaten.....	67		
Projektsprache auswählen.....	69		
Projektsprachen verwalten.....	67		
Übersetzungsdatei exportieren.....	68		
Übersetzungsdatei importieren.....	68		
SPS-Objekte eines externen IndraWorks-			
Projektes anzeigen			
Befehl.....	644		
SPS-Shell.....	226		
Geräteeditor.....	532		
Spule.....	286		
einfügen.....	706		
negiert.....	286		
Reset.....	706		
Reset-Spule.....	286		
Set.....	706		
Set-Spule.....	286		
SQRT.....	343		
ST.....	104		
Ausdruck.....	242		
erweitert.....	105		
ST-Anweisung 'RETURN'.....	251		
ST-Code			
ausführen, Visualisierung.....	849		
ST-Code in FUP, KOP.....	285		
ST-Editor.....	241		
Online-Betrieb.....	242		
Option.....	776		
ST-Zuweisungsoperator.....	244		
Stackprüfung rekursiver Methoden.....	409		
Standard axial.....	862		
Standard linear.....	862		
Standard radial.....	862		
Standardprojekt.....	8		
Startbild.....	634		
Startvisualisierung.....	821		
Static Analysis Light			
Projekteinstellung.....	761		
Statische Analyse			
Beispiel: Mehrere Regeln ignorieren.....	134		
Implementierungscode ausschließen.....	133		
Pragma.....	133		
Programmierobjekte mit Pragma von der			
statischen Analyse ausschließen.....	134		
Statische Variable.....	306		
Status			
AS-Aktionen.....	257		
AS-Schritten.....	257		
Geräteeditor.....	543		
Stileigenschaft.....	1080		
Strg.....	878, 886, 896, 905, 915, 928, 948, 969,		
	982, 985		
strict, Pragma.....	436		
STRING.....	376		
Datentyp.....	376		
STRING_TO.....	341		
Stringkonstante.....	367		
STRUCT.....	396		
Struktur.....	396		
Datentyp.....	396		
Zugriff.....	369		
strukturierte Variablen aktualisieren.....	727		
SUB.....	320		
subsequent, Pragma.....	436		
Suchen.....	138		
SUPER.....	312		
Support.....	1083		
Symbol suchen.....	136		
symbol, Pragma.....	436		
Symbolbibliothek.....	809		
in Visualisierung verwenden.....	809		
Symboldatei.....	588		
Symbole.....	4		
Symbolkonfiguration.....	167, 588		
hinzufügen.....	588		
Synchronisierung mit IEC-Tasks konfigu-			
rieren.....	593		
Unterstützung bei der aktuellen Konfigu-			
ration und entsprechenden Korrektur-			
maßnahmen.....	595		
Synchronisierte Dateien			
Geräteeditor.....	524		

Syntax prüfen.....	132	Schrift.....	776
Syntaxhervorhebung		Steuerung, Visualisierungselementeigen- schaft.....	1055
Option.....	776	Text und Sprache	
Systemereignis.....	599	in Visualisierung verwenden.....	845
Funktionsaufruf.....	599	Texteditor	
Systemvariable.....	227	Einrückung Code.....	621
Operationssteuerung.....	227	Option.....	776
T		Registerkarte 'Editieren'.....	776
Tabelle		Registerkarte 'Monitoring'.....	779
Daten zur Tabelle zuweisen, Beispiel.....	957	Registerkarte 'Randbereich'.....	778
Eigenschaften konfigurieren, Beispiel.....	958	Registerkarte 'Textbereich'.....	778
Visualisierungselement.....	952	Texteigenschaften....	872, 881, 890, 899, 909, 921, 930, 943, 951, 955, 966, 973
TAN.....	346	Textfeld	
Task.....	602	Visualisierungselement.....	963
Abarbeitungsreihenfolge.....	143	Textindex.....	873, 882, 892, 901, 911, 923, 945, 956, 974
Registerkarte Konfiguration.....	602	Textliste....	873, 882, 892, 901, 911, 923, 945, 956, 974
Statistik.....	224	alle als Text exportieren.....	728
überwachen.....	224	als Unicode-Text exportieren.....	728
Überwachung.....	601	bearbeiten.....	119
Zykluszeiten.....	224	Datei importieren.....	120
Task-Aufstellung		Export/Import.....	729
Geräteeditor.....	542	exportieren.....	119
Taskaufstellung		für dynamische Anwendung.....	125
überprüfen.....	204	im Visualisierungsmanager einstellen.....	1060
Taskkonfiguration.....	143	Sprache einfügen.....	727
Eigenschaften.....	598	Sprache entfernen.....	730
erstellen.....	144	Text dynamisch ausgeben.....	125
Grundeinstellungen.....	598	Text einfügen.....	729
Grundlagen.....	143	vergleichen und Unterschied exportieren....	121
Objekt.....	598	Textliste anlegen und füllen.....	118
Tasküberwachung, Online.....	601	Textliste oder globale Textliste.....	117
Tastaturbedienung		Textlistenunterstützung entfernen.....	732
Visualisierung.....	1081	Textlistenunterstützung hinzufügen.....	731
Tastaturbedienung aktivieren.....	813	THEN.....	246
Tastaturereignisse bemerken.....	1074	THIS.....	313
Tastaturkonfiguration		TIME.....	376
Visualisierung.....	839	Beispiel.....	363
Visualisierungsmanager.....	1060	Syntax.....	362
Tasten....	877, 878, 885, 886, 895, 896, 904, 905, 915, 927, 928, 948, 969, 982, 984, 985	TIME_OF_DAY.....	376
Tastenkürzel		TIME_OF_DAY Konstante.....	365
für Visualisierung konfigurieren.....	839	Beispiel.....	365
Temporäre Variable.....	306	Syntax.....	365
TEST_AND_SET		TIME_OF_DAY_TO.....	339
Beispiel.....	359	TIME_TO.....	339
Funktionsweise.....	359	TIME-Funktion.....	373
Testen und Fehler beheben		TIME-Konstanten.....	362
In den Debugbetrieb wechseln.....	196	TIME().....	373
Schrittweises Abarbeiten eines Pro- gramms (Stepping).....	196	TO.....	247
Text		TO_<xxx>.....	337
Ausrichtung, Visualisierungselementei- genschaft.....	1055	TO_BOOL.....	336
Farbe.....	776	TOD.....	376
Formatierung, Visualisierungselementei- genschaft.....	1055	Tooltip	
		anzeigen, Visualisierungselementeigen- schaft.....	1055

Index

- Tooltipindex..... 873, 882, 892, 901, 911, 923, 945, 956, 974
 - Trace..... 216
 - als Datei verwalten..... 223
 - Anzeigeeinstellung, Visualisierungselement..... 1046
 - Assistent des Visualisierungselements..... 1048
 - auf alle Traces der Steuerung zugreifen..... 223
 - Aufzeichnung über Visualisierungselement konfigurieren..... 1041
 - Datei laden 735
 - Dateiformate..... 217
 - Datenerfassung konfigurieren..... 784
 - Datenerfassung starten..... 739
 - Datenerfassung steuern..... 223
 - Datenerfassung stoppen..... 739
 - Erfassen der Daten einer IEC-Variablen..... 219
 - Erste Schritte..... 218
 - erweiterte Einstellung..... 782
 - erweiterte Einstellungen, Visualisierungselement..... 1046
 - Gerätetrace laden..... 738
 - hochladen..... 740
 - in Ausführung auflisten..... 737
 - In Datei speichern..... 738
 - in den Traceeditor laden..... 740
 - Konfiguration exportieren..... 1049
 - konfigurieren..... 733
 - konfigurierte Variable, Visualisierungselement..... 1044
 - Laufzeitsystemkomponente CmpTraceMgr..... 216
 - LZS-Puffer 782
 - Online-Liste..... 737
 - Samples in Datei speichern..... 217
 - Statistik anzeigen..... 223
 - Statistik öffnen..... 740
 - Tracecursor..... 733
 - triggern..... 217
 - Variable für Visualisierungselement konfigurieren..... 1043
 - Variable konfigurieren..... 786
 - Visualisierungselement..... 1032
 - Visualisierungselement konfigurieren, Beispiel..... 1035
 - Visualisierungselement mit Cursor, Beispiel..... 1038
 - Visualisierungselement mit Trigger konfigurieren..... 1039
 - vom LZS entfernen..... 738
 - Y-Wert auslesen..... 733
 - Trace konfigurieren..... 822
 - Tracediagramm
 - automatisch skalieren..... 732
 - in den Daten navigieren..... 223
 - in Einkanal-Darstellung konvertieren..... 736
 - in Mehrkanal-Darstellung konvertieren..... 736
 - mit der Maus navigieren..... 606
 - mit der Tastatur navigieren..... 607
 - Zoomen mit Maus..... 736
 - zur Laufzeit..... 606
 - Traceeditor..... 605
 - Tracegraph
 - Ansicht auf Voreinstellung zurücksetzen..... 738
 - komprimieren..... 732
 - strecken 739
 - Tracekonfiguration..... 219
 - Anzeige der Tracevariablen konfigurieren... 222
 - Anzeige der Zeitachse konfigurieren..... 221
 - Befehl 'Download Trace'..... 734
 - IEC-Variable hinzufügen..... 220
 - Parameter tracen..... 220
 - Puffer für Daten auf dem LZS konfigurieren..... 222
 - symbolisch exportieren..... 734
 - Task zuweisen..... 219
 - Tracekonfiguration zur Laufzeit bearbeiten..... 222
 - Tracevariable entfernen..... 220
 - Tracevariable konfigurieren..... 220
 - Triggerauslösung konfigurieren..... 221
 - Variable entfernen..... 220
 - Variable hinzufügen..... 732
 - Tracemanager siehe CmpTraceMgr..... 216
 - Transition..... 262
 - Transparenz
 - Schrift, Visualisierungselementeigenschaft..... 1055
 - Transparenzfarbe..... 1010
 - Trigger
 - Tracekonfiguration zurücksetzen..... 738
 - TRUE..... 362
 - TRUNC..... 342
 - TRUNC_INT..... 342
 - TRY..... 353
 - Typed Literals..... 369
- ## U
- Übergabevariable
 - VAR_IN_OUT CONSTANT..... 304
 - Übersetzen
 - Applikation..... 649, 650
 - Eigenschaften..... 749
 - Übersetzte Bibliothek im IndraWorks..... 678
 - Übersetzungsinformation..... 650, 651
 - Überwachen
 - Alle Forces anzeigen..... 199
 - Überwachung des SPS-Programms während der Laufzeit..... 576
 - POU-Vorlagen..... 578
 - Überwachungsfunktion..... 153
 - Überwachungsliste..... 211
 - Variable hinzufügen..... 742
 - UDINT..... 375
 - UINT..... 375

ULINT.....	375	deklarieren, textuell.....	87
Um Eins nach hinten legen.....	818	Eingabe.....	300
um Eins nach vorn legen.....	818	Extern.....	307
Umschalten..... 877, 886, 895, 904, 927, 948, 969, 985		Global.....	306
Umschalten beim Loslassen falls Maus gefangen..... 877, 886, 895, 904, 927, 948, 969, 985		initialisieren.....	86
Umschalten-/TastenvARIABLE		Kurzformmodus.....	113
Visualisierungselement Schaltfläche.....	947	lokal.....	300
Visualisierungselementeigenschaft..... 874, 902, 967		Monitoring.....	205
Umschalttaste..... 878, 886, 896, 905, 915, 928, 948, 969, 982, 985		schreiben, Visualisierung.....	849
Umschaltvariable.....	926, 939	Speicherplatz allozieren.....	436
undefine.....	439	Statisch.....	306
Unicode		Temporär.....	306
für Zeichenfolge in Visualisierung verwenden.....	1058	umschalten, Visualisierung.....	849
Textliste.....	728	verstecken, Pragma.....	414
UNION.....	377	Verwendungsstelle.....	135
unsichtbare Eingabe		Wert im Online-Betrieb.....	205
Visualisierungselement.....	982	Zugriff.....	369
Unsichtbarkeit.....	931, 988, 1051, 1054	Variable schreiben in Visualisierungen.....	1074
Unten ausrichten.....	814	Variablendeklaration.....	85
Unterbereichstypen.....	401	verschieben, nach oben.....	720
Untergeordnete Sicherheitssteuerung.....	184	verschieben, nach unten.....	720
Unterskala.....	999, 1003, 1007, 1011	Variablenkonfiguration.....	128
Unterskalaposition.....	999, 1003, 1007, 1011	Variablenname, Regeln.....	445
UNTIL.....	250	VCADD.....	392
Ursprungsverschiebung.....	1008, 1012	Beispiel: Addieren.....	392
USINT.....	375	VCDIV	
V		Beispiel: Dividieren.....	393
VAR.....	300	VCDIV.....	393
VAR CONSTANT		VCDOT.....	393
Syntax.....	308	Beispiel: Punktprodukt.....	393
VAR_ACCESS.....	454	VCLOAD_LREAL.....	395
VAR_CONFIG.....	311	Beispiel: Vektorisieren.....	395
VAR_EXTERNAL.....	307	VCLOAD_REAL.....	395
VAR_GLOBAL.....	306	Beispiel: Vektorisieren.....	395
VAR_IN_OUT.....	301	VCMAX.....	394
VAR_IN_OUT_AS_POINTER, Pragma.....	836	Beispiel: Maximumvektor.....	394
VAR_INPUT.....	300	VC MUL.....	393
VAR_INST.....	307	Beispiel: Multiplizieren.....	393
VAR_OUTPUT.....	301	VCSET_LREAL.....	394
VAR_PERSISTENT		VCSET_LREAL	
Alle Instanzpfade hinzufügen.....	722	Beispiel.....	395
VAR_STAT.....	306	VCSET_REAL.....	394
VAR_TEMP.....	306	Beispiel.....	394
Variable..... 362, 877, 885, 886, 895, 904, 926, 927, 939, 948, 969, 984, 985		VCSQRT.....	394
Adresse.....	130	Beispiel: Quadratwurzel.....	394
Ausgabe.....	301	VCSTORE	
Bitzugriff.....	369	Beispiel: Speichern.....	396
Darstellungsformat, Pragma.....	408	VCSTORE.....	396
deklarieren.....	85, 88	VCSUB.....	392
deklarieren, tabellarisch.....	88	Beispiel: Subtrahieren.....	392
		VECTOR	
		Datentyp.....	391
		VendorException.....	527
		Verbinden	
		mit Gerät.....	741
		Verbinder	
		zurücksetzen.....	698
		Verbindungsmarke	

Index

CFC.....	299	Verzweigung.....	267
Verbindung lösen.....	696	einfügen.....	684
Verbindungsmarke.....	699	einfügen rechts.....	684
Vergleich		Visualisierung	
Allgemeiner Bausteinteil.....	18	Anzeigen der verfügbaren.....	1060
Application.....	20	ausgewählt als Framereferenz.....	819
Bibliotheks Info / Library Info.....	38	Edit-Control.....	1074
Bibliotheksverwalter.....	37	Eigenschaften.....	1063
Bildersammlung.....	44	Eingabedialoge.....	1070
Datenquelle (Datenserver).....	41	Eingabekonfiguration.....	849
Datenserver.....	39	Element - Linie.....	878
Datentypen.....	25	Element - Schaltfläche.....	940
Geräte (Onboard E/A, Inline E/A, Feld- busse, ...).....	46	Element vervielfachen.....	865
Globale Variablenliste, PersistentVariab- len, Netzwerkvariablen.....	26	Elementliste.....	841
Logic.....	19	für Darstellungsvariante aktivieren.....	1060
Netzwerkvariablenlisten empfängerseitig.....	28	Kennzeichen erstellen.....	807
Ordner.....	24	Kompatibilitätsprofil.....	1080
Rezepturdefinition.....	46	Menübefehle.....	812
Rezepturverwalter.....	44	Objekt erstmalig hinzufügen.....	842
Schnittstellen, Schnittstellen-Methoden, Schnittstellen-Eigenschaften.....	36	Option.....	1079
Symbolkonfiguration.....	42	Optionen.....	1079
Task.....	24	Ordner, der Bildersammlung enthält.....	764
Taskkonfiguration.....	23	Ordner, der Textliste enthält.....	764
Textliste.....	43	Profil.....	1080
Visualisierung.....	43	Projekteinstellung.....	763
Visualisierungsmanager.....	42	Rasterlinien einblenden oder aktivieren.....	780
Vergleich von SPS-Objekten.....	17	Semitransparenz.....	863
Vergleich von SPS-Objekten und Geräteda- ten (Online).....	51	Speicherort für Bilddateien.....	780
Vergleich von SPS-Projektdaten und Gerä- tedaten (Online)		Speicherort für Datei, die die Funktion Komponentenauflisten verwendet.....	780
Detailvergleich.....	52	Speicherort für Textlisten.....	780
Optionen.....	54	Symbolbibliothek verwenden.....	809
Übersicht.....	51	Tastaturereignisse bemerken.....	1074
Vergrößern/Verkleinern		Traceassistent.....	1048
grafischer Editor.....	240	verfügbar als Framereferenz.....	819
Verschattungsregeln.....	451	Vorschau auf Bibliotheksvisualisierung.....	805
Suchen in Instanzpfaden.....	453	Werkzeugeansicht.....	826
Suchreihenfolge in der Applikation.....	452	Visualisierungen	
Suchreihenfolge in der Bibliothek.....	452	Option.....	779
Uneindeutige Zugriffe und qualifizierte Zugriffe.....	453	Visualisierungs-Text-IDs aktualisieren.....	731
Vertikal zentrieren.....	814	Visualisierungs-Text-IDs prüfen.....	731
vertikale Ausrichtung.....	872, 881, 890, 899, 909, 921, 930, 955, 966, 973	Visualisierungsbibliothek.....	1078
Vertikalen Abstand entfernen.....	817	Visualisierungseditor.....	825
Vertikalen Abstand gleichmachen.....	816	Visualisierungselement	
Vertikalen Abstand vergrößern.....	816	auswählen.....	806
vertikalen Abstand verkleinern.....	816	bedienen.....	1081
Vervielfachen eines Visualisierungsele- ments.....	865	einfügen.....	813
Verwendungsstelle		Element in die Visualisierung ziehen.....	807
Variable.....	135	für Bedienen auswählen.....	1081
Verzeichnis für Bilddateien.....	779	Kennzeichen.....	806
Verzeichnis für Textlisten.....	779	kennzeichnen.....	806
		Konfiguration mit Schnittstelleneigen- schaft.....	764
		Liste von allen.....	841
		mit statischem Text.....	122
		verfügbare anzeigen.....	806
		vervielfachen.....	823

- Ziehen einer IEC-Variable auf ein Element..... 868
 - Zuordnung in Werkzeuge ansehen..... 842
 - Visualisierungselemente..... 843
 - Allgemeine Steuerelemente..... 929
 - Basis..... 869
 - Elemente konfigurieren - allgemein..... 843
 - Lampen/Schalter/Bilder..... 1019
 - Messgeräte..... 992
 - Spezielle Steuerelemente..... 1032
 - Visualisierungselementeigenschaft..... 1054
 - Visualisierungsmanager..... 1058, 1070
 - Benutzerverwaltungsdialog..... 1059
 - Einstellung..... 1058
 - Multitouch aktivieren..... 1060
 - Tastaturkonfiguration..... 1060
 - Testlisteneinstellung..... 1060
 - Visualisierungsschnittstelle..... 1061
 - Visualisierungsstil..... 1080
 - Ausweichlösung beim Ableiten..... 1079
 - für Bibliotheksvisualisierung oder globale Visualisierung konfigurieren..... 1079
 - Option..... 1079
 - Visualisierungsstile
 - Option..... 781
 - Visualisierungswechsel..... 849
 - VisuDialogs.library..... 1064, 1070
 - VisuElems
 - Bibliothek..... 1080
 - Vollbildmodus..... 634
 - VUM_ChangePassword
 - Benutzerverwaltungsdialog, Visualisierung..... 1059
 - VUM_Login
 - Benutzerverwaltungsdialog, Visualisierung..... 1059
 - VUM_UserManagement
 - Benutzerverwaltungsdialog, Visualisierung..... 1059
- W**
- W..... 371
 - Warnhinweise..... 2
 - warning disable, Pragma..... 438
 - warning restore, Pragma..... 438
 - Wartesymbol
 - Blume, Visualisierungselement..... 1052
 - Watch-Liste..... 211
 - Weiterverschaltung festlegen, FUP/KOP..... 710
 - Werkzeug..... 628
 - Werkzeuge
 - Ansicht in der Visualisierung..... 826
 - Wert vorbereiten..... 745
 - Werte aus Rezeptur wiederherstellen..... 722
 - WHILE..... 249
 - Winkel..... 862
 - WORD..... 375
- Word-Adressierung..... 371
- WriteRecipe..... 795
- WSTRING..... 382
- WSTRING-Konstanten..... 368
- X**
- X..... 371, 870, 898, 908, 920, 930, 932, 935, 939, 941, 950, 955, 964, 971, 976, 979, 983, 987, 989, 991, 993, 998, 1002, 1006, 1010, 1015, 1021, 1024, 1027, 1030, 1034, 1051, 1053
- XOR..... 324
- XORN..... 278
- Y**
- Y..... 870, 898, 908, 920, 930, 932, 935, 939, 941, 950, 955, 964, 971, 976, 979, 983, 987, 989, 991, 993, 998, 1002, 1006, 1010, 1015, 1021, 1024, 1027, 1030, 1034, 1051, 1053
- Z**
- Zahlenkonstanten..... 365
 - Zeiger..... 383, 1006, 1011
 - SUPER..... 312
 - THIS..... 313
 - Zeigerabstand..... 1008, 1012
 - Zeigeranfang..... 1006, 1011
 - Zeigerende..... 1006, 1011
 - Zeigerinstrument 90°
 - Visualisierungselement..... 996
 - Zeigerinstrument 180°
 - Visualisierungselement..... 1001
 - Zeigerinstrument 360°
 - Visualisierungselement..... 1005
 - Zeigertyp..... 1006, 1011
 - Zeitdatentypen..... 376
 - Zentrum..... 870, 880, 889, 898, 909, 920, 941
 - Ziehen einer Variable auf ein Element..... 868
 - Zoom
 - Faktor..... 240
 - grafischer Editor..... 240
 - Zu dieser Dokumentation..... 1
 - Gültigkeit der Dokumentation..... 1
 - Zugangsdaten speichern..... 62
 - Zugriffskontrolle
 - Eigenschaften..... 750
 - Zugriffsrecht
 - Geräteeditor..... 540
 - Objekt..... 59
 - Zusätzlicher Zeiger..... 1006
 - Zustandsvariablen..... 931, 988, 1051, 1054
 - Zuweisung
 - FUP/KOP/AWL..... 282
 - Zuweisung einfügen, FUP/KOP/AWL..... 703
 - Zuweisung, ExST..... 246
 - Zuweisungen
 - ST-Zuweisungsoperator für Ausgänge..... 244
 - Zuweisungsoperator 'REF'..... 246

Zuweisungsoperator, ST.....	244
zwei Farben benutzen.....	862

Notizen

Bosch Rexroth AG

Postfach 13 57

97803 Lohr a.Main, Deutschland

Bgm.-Dr.-Nebel-Str. 2

97816 Lohr a.Main, Deutschland

Tel. +49 932 18 0

Fax +49 9352 18 8400

www.boschrexroth.com/electrics



R911396134