Micro800 Programmierbare Controller – Allgemeine Anweisungen

Katalognummern 2080-L50E, 2080-L70E, 2080-LC10, 2080-LC20, 2080-LC30, 2080-LC50, 2080-LC70

Rockwell Automation-Publikation 2080-RM001L-DE-E - November 2023
Ersetzt die Publikation 2080-RM001K-DE-E - März 2022

Rockwell Automation

Referenzhandbuch Originalanleitung

Wichtige Benutzerinformationen

Lesen Sie dieses Dokument und die im Abschnitt „Zusätzliche Ressourcen“ aufgelisteten Dokumente zu Installation, Konfiguration und Betrieb dieser Geräte, bevor Sie dieses Produkt installieren, konfigurieren, in Betrieb nehmen oder warten. Alle Benutzer müssen sich neben den Anforderungen aller entsprechenden Codes, Gesetze und Standards mit den Installations- und Verdrahtungsanweisungen vertraut machen.

Arbeitsvorgänge wie Installation, Einstellungen, Inbetriebnahme, Verwendung, Montage, Demontage und Wartung müssen von entsprechend qualifizierten Personen unter Einhaltung der entsprechenden vorgeschriebenen Vorgehensweise durchgeführt werden.

Wenn die Nutzung dieses Gerät von der Nutzungsbeschreibung des Herstellers abweicht, kann der durch das Gerät gewährleistete Schutz beeinträchtigt werden.

In keinem Fall ist Rockwell Automation, Inc. verantwortlich oder haftbar für indirekte oder Folgeschäden, die sich aus der Verwendung dieser Geräte ergeben.

Die Beispiele und Diagramme in diesem Handbuch dienen nur zur Veranschaulichung. Da eine Vielzahl von Variablen und Voraussetzungen für eine bestimmte Installation erforderlich ist, kann Rockwell Automation, Inc. keine Verantwortung oder Haftung für die tatsächliche Verwendung basierend auf den Beispielen und Diagrammen übernehmen.

Rockwell Automation, Inc. übernimmt keine Patenthaftung in Bezug auf die Verwendung der in diesem Handbuch beschriebenen Informationen, Schaltkreise, Ausrüstung oder Software.

Die Reproduzierung des Inhalts dieses Handbuchs, vollständig oder teilweise, ist ohne schriftliche Zustimmung von Rockwell Automation, Inc. nicht zulässig.

Im gesamten Handbuch verwenden wir ggf. Hinweise, um Sie auf Sicherheitserwägungen aufmerksam zu machen.

WARNUNG: Weist auf Informationen zur praktischen Anwendung oder auf Umstände hin, die in gefährlichen Umgebungen zu Explosionen führen können und Verletzungen oder Todesfolge, Beschädigungen und Umweltschäden nach sich ziehen können.

ACHTUNG: Weist auf Informationen zur praktischen Anwendung oder auf Umstände hin, die Verletzungen oder Todesfolge, Beschädigungen und Umweltschäden nach sich ziehen können. Warnungen helfen Ihnen, Gefahren zu identifizieren, zu vermeiden und die Folgen abzuwägen.

WICHTIG Bezeichnet Informationen, die für die erfolgreiche Verwendung und das Verständnis des Produkts wichtig sind.

Außen am Gerät sowie im Geräteinneren müssen sich Etikette befinden, von denen die spezifischen Vorsichtsmaßnahmen ablesbar sein müssen.

STROMSCHLAGGEFAHR: Dieses Etikett an oder in einem Gerät, z. B. einem Antrieb oder einem Motor, weist darauf hin, dass gefährliche Spannung anliegt.

BRANDGEFAHR: Dieses Etikett an oder in einem Gerät, z. B. einem Antrieb oder einem Motor, weist darauf hin, dass die Oberfläche gefährlich heiß werden kann.

GEFAHR VON LICHTBOGENBILDUNG: Dieses Etikett an oder in einem Gerät, z. B. einem Motorsteuerungszentrum, weist darauf hin, dass die Gefahr von Lichtbogenbildung besteht. Lichtbögen führen zu schweren Verletzungen und sogar zum Tod! Tragen Sie geeignete persönliche Schutzausrüstung. Befolgen Sie ALLE bestimmungsgemäßen Anforderungen hinsichtlich sicherer Arbeitsweise und der Schutzausrüstung.

Rockwell Automation erkennt, dass einige der Begriffe, die derzeit in unserer Branche und in dieser Publikation verwendet werden, nicht mit der Bewegung in Richtung einer inklusiven Sprache in der Technologie ausgerichtet sind. Wir arbeiten proaktiv mit Kollegen aus der Branche, um Alternativen zu diesen Begriffen zu finden und Änderungen an unseren Produkten und Inhalten vorzunehmen. Bitte entschuldigen Sie die Verwendung dieser Begriffe in unserem Inhalt; wir sind dabei, diese Änderungen zu implementieren.

Inhaltsverzeichnis

Vorwort

In diesem Handbuch 15
Unterstützte Controller 15
Zusätzliche Ressourcen 15
Rechtliche Hinweise 16

Kapitel 1

Suchen von Informationen zu Anweisungen und Strompfadelementen

Anweisungsblöcke 19
- Operatoren 19
- Funktionen 19
- Funktionsblöcke 21

Anweisungssatz in alphabetischer Reihenfolge 22

Kapitel 2

Sprache des Kontaktplans (KOP)

Kontaktplanprogramm (KOP) 29
Entwicklungsumgebung für KOP-Programm 29
- Anweisungs-Symbolleiste (KOP) 30
  - Hinzufügen von Anweisungselementen aus der Anweisungs-Symbolleiste 30

Kapitel 3

Elemente des Kontaktplans (KOP)

- Strompfad 33
  - Hinzufügen eines Strompfads zu einem KOP-Programm 33
  - Strompfad-Bezeichnungen 35
  - Strompfadkommentare 36
- Verzweigung 37
  - Hinzufügen einer Verzweigung zu einem KOP-Programm 38
- Anweisungsblock (KOP) 40
- Spule 40
  - Hinzufügen von Spulenelementen 41
  - Direkte Spule 43
  - Invertierter Ausgang 44
  - Spule mit Impulsabstiegflanke 45
  - Spule mit Impulsanstiegflanke 45
  - Stellspule 46
  - Rückstellspule 47
- Kontakt 48
  - Hinzufügen eines Kontakts zu einem KOP-Programm 48
  - Direkter Kontakt 51
  - Umkehrkontakt 51
  - Impuls-Kontakt bei steigender Flanke 52
  - Impuls-Kontakt bei fallender Flanke 52

Rücksprung 53
Sprung 54
Anweisungsblöcke in KOP-Programmen 55
Arbeiten im KOP-Sprach-Editor 56
Beispiele zum Kontaktplan (KOP) 57
KOP-Tastenkombinationen 58

Kapitel 4

Alarmanweisung

LIM_ALRM (Alarmgrenze) 61

Kapitel 5

Arithmetische Anweisungen

ABS (Absolutbetrag) 65
ACOS (Arkuscosinus der Quelle) 67
ACOS_LREAL (Arkuscosinus-Typ „Long Real“) 68
Addition 69
ASIN (Arkussinus) 71
ASIN_LREAL (Arkussinus-Typ „Long Real“) 72
ATAN (Arkustangens) 74
ATAN_LREAL (Arkustangens einer langen Realzahl) 75
COS (Cosinus) 76
COS_LREAL (Cosinus lange Realzahl) 78
Division 79
EXPT (Exponent) 81
LOG (Logarithmusbasis 10) 82
MOD (Modulo) 84
MOV (Verschieben) 85
Multiplikation 86
Neg (Negation) 88
POW (Potenzieren) 89
RAND (Zufallswert) 90
SIN (Sinus) 92
SIN_LREAL (Sinus langer Realzahl) 94
SQRT (Quadratwurzel) 95
Subtraktion 97
TAN (Tangens) 98
TAN_LREAL (Tangens-Typ "Long Real") 100
TRUNC (Abschneiden) 101

Kapitel 6

ASCII-Anweisungen zu seriellen Ports

ABL (ASCII-Test für Pufferzeile) 103
ACL (ASCII-Puffer löschen) 105
AHL (ASCII-Handshake-Leitungen) 107

ACB (ASCII-Zeichen im Puffer) 109
ARD (ASCII lesen) 111
ARL (ASCII-Zeile lesen) 114
AWA (ASCII-Schreibanhang) 116
AWT (ASCII-Schreibvorgang) 118
ASCII-Parameterdetails 120
Datentyp ABLACB 121
Datentyp ACL 121
Datentyp AHL ChannelSts 121
Datentyp AHLI 122
Datentyp ARDARL 122
Datentyp AWAAWT 122

Kapitel 7

Binäre Anweisungen

AND_MASK (AND-Maske) 125
BSL (Bit-Verschiebung links) 127
BSR (Bit-Verschiebung rechts) 130
NOT_MASK (Bit-zu-Bit-NOT-Maske) 133
OR_MASK (bitweise OR-Maske) 134
ROL (nach links drehen) 135
ROR (nach rechts drehen) 137
SHL (Verschiebung links) 139
SHR (Verschiebung rechts) 141
XOR_MASK (exklusive OR-Maske) 142

Kapitel 8

Boolesche Anweisungen

F_TRIG (Erkennung der Abstiegsflanke) 145
R_TRIG (Anstiegflankenerkennung) 147
RS (Zurücksetzen/Setzen) 148
OR 150
AND 150
XOR (exklusives OR) 151
NOT 152
SR (setzen/zurücksetzen) 152
TTABLE (Wahrheitstabelle) 154
MUX8B (Multiplexer mit 8 BOOL-Eingängen) 156
MUX4B (Multiplexer mit 4 BOOL-Eingängen) 159

Kapitel 9

Kommunikationsanweisungen

COM_IO_WDOG 163
MSG_CIPGENERIC (Allgemeine CIP-Nachricht) 166
Datentyp „CIPAPPCFG“ 168

Datentyp „CIPCONTROLCFG“ 168
Datentyp „CIPSTATUS“ 169
CIPSTATUS-Statusbits 169
CIPSTATUS-Fehlercodes 170
Datentyp CIPTARGETCFG 171
Zielpfad für CIP-Messaging 171
CIP/EIP-Nachrichtenverbindungen 172
Zeitüberschreitungs-Timer für CIP-Nachrichten 173
MSG_CIPSYMBOLIC (Symbolische CIP-Nachricht) 173
Datentyp „CIPSYMBOLICCFG“ 175
Symbolische Syntax für Lesen/Schreiben 176
Unterstützte Datenpaketgröße für die serielle CIP-Funktion 177
MSG_MODBUS (Modbus-Nachricht) 177
Datentyp „MODBUSLOCPARA“ 180
Nachrichtenausführungsprozess (Strompfad = TRUE) 181
Datentyp „MODBUSTARPARA“ 182
MSG_MODBUS2 (MODBUS/TCP-Nachricht) 183
Datentyp MODBUS2LOCPARA 186
Datentyp MODBUS2TARPARA 187
MSG_PCCC (Nachricht für programmierbare Controller-Kommunikationsbefehle) 189
Datentyp „CIPCONTROLCFG“ 191
Datentyp PCCCCFG 192
Datentyp TARGETCFG 193
Datentyp „CIPSTATUS“ 195
CIPSTATUS-Statusbits 195
CIP/EIP-Nachrichtenverbindungen 196
Nachricht-Zeitüberschreitung 197
MSG_PCCC-Fehlercodes 197
Beispiel: Erstellen Sie ein Programm für MSG_PCCC-Nachrichtenaustausch, um einen Wert auf eine Adresse in MicroLogix zu schreiben und von dieser zu lesen 200
Beispiel für das PCCC-Schreiben: Hinzufügen eines MSG_PCCC-Funktionsblocks, Variablen und eines Kontakts 200
Beispiel für PCCC-Lesen Hinzufügen eines PCCC_MSG-Funktionsblocks, Variablen und eines Kontakts 204
Nachrichtenausführungsprozesse und Timing-Diagramme 207
Nachrichtenausführungsprozess (allgemein) 207
Nachrichtenausführungsprozess (Strompfad = TRUE) 208
Nachrichtenausführungs-Timing-Diagramm (Strompfad = TRUE) 209
Nachrichtenausführungsprozess (Strompfad = FALSE) 210

Nachrichtenausführung-Timing-Diagramm (Strompfad = FALSE) 211
Nachrichtenausführungsprozess (Fehler) 212
Nachrichtenausführungs-Timing-Diagramm (Fehler) 213
Nachrichtenausführungs-Timing-Diagramm (Fehler bei der PCCC-Konfiguration) 213
Nachrichtenausführungs-Timing-Diagramm (Nachricht wird abgebrochen) 214
Verwenden der Kommunikationsfunktionsblöcke (Nachricht) 215
Konfigurieren von Objektdatenwerten für die explizite Nachrichtenvermittlung (MSG_CIPGENERIC) 215
Beispiel: So erstellen Sie ein MSG_CIPGENERIC-Nachrichtenprogramm zum Lesen von Daten von einem Controller 217
Ermitteln von Anfangswerten für die Eingangsvariablen, MSG_CIPGENERIC 218
Hinzufügen eines MSG_CIPGENERIC-Funktionsblocks und -Variablen 219
Konfigurieren von Anfangswerten für Variablen 220
Hinzufügen eines Kontakts und einer Spule 222
Hinzufügen eines COP-Funktionsblocks, von Variablen und eines Kontakts (MSG_CIPGENERIC) 223
Verifizieren der korrekten IP-Konfiguration auf Controller B 225
Beispiel: So erstellen Sie ein MSG_CIPSYMBOLIC-Nachrichtenprogramm zum Schreiben eines Wertes in eine Variable 225
Hinzufügen eines COP-Funktionsblocks, von Variablen und eines Kontakts (MSG_CIPSYMBOLIC) 226
Hinzufügen eines „Gleich“-Operators und einer Spule 228
Hinzufügen eines MSG_CIPSYMBOLIC-Funktionsblocks, der Variablen und eines Kontakts 229
Verifizieren der korrekten IP-Konfiguration auf Controller B 231
Erzeugen einer globalen Variable für Controller B 232
Überprüfen der vollständigen Programmergebnisse 233
Beispiel: So konfigurieren Sie die Modbus-Kommunikation zum Lesen von und Schreiben auf ein Laufwerk 234
Kommunikationsprotokollunterstützung 238
Integrierte Kommunikationskanäle 240

Kapitel 10

Vergleichsanweisungen

Gleich 241
Größer als 244

Größer als oder gleich 245
Kleiner als 246
Kleiner als oder gleich 248
Nicht gleich 249

Kapitel 11

Zähleranweisungen

CTD (Herunterzählen) 251
CTU (Hochzählen) 253
CTUD (Hoch-/Herunterzählen) 255

Kapitel 12

Anweisungen zur Datenumwandlung

ANY_TO_BOOL 257
ANY_TO_BYTE 258
ANY_TO_DATE 259
ANY_TO_DINT 261
ANY_TO_DWORD 262
ANY_TO_INT 263
ANY_TO_LINT 264
ANY_TO_LREAL 265
ANY_TO_LWORD 266
ANY_TO_REAL 267
ANY_TO_SINT 268
ANY_TO_STRING 269
ANY_TO_TIME 270
ANY_TO_UDINT 271
ANY_TO_UINT 272
ANY_TO_ULINT 273
ANY_TO_USINT 274
ANY_TO_WORD 275

Kapitel 13

Anweisungen zur Datenbearbeitung

AVERAGE 277
COP 279
Kopieren zu einem anderen Datentyp 281
Beispiel für ein COP-String-Array 282
MIN (Minimum) 284
MAX (Maximum) 285

Kapitel 14

Hochgeschwindigkeitszähler-Anweisungen (HSC)

Was ist ein Hochgeschwindigkeitszähler? 289
HSC (Hochgeschwindigkeitszähler) 290
HSCCmd-Werte 292

Datentyp HSCAPP ... 292
Datentyp HSCSTS ... 296
Datentyp PLS ... 303
Datentyp HSCE_CHANNEL ... 304
Datentyp HSCE_STS ... 305
Datentyp PLS_HSCE ... 305
HSC_SET_STS (Hochgeschwindigkeitszählerstatus festlegen) ... 306
Verwenden der Hochgeschwindigkeitszähler-Anweisungen ... 308
Aktualisieren der HSC-Anwendungsdaten ... 308
Dialogfeld zur Benutzerunterbrechung des Hochgeschwindigkeitszählers ... 309
Konfigurieren einer Hochgeschwindigkeitszähler-Benutzerunterbrechung (HSC) ... 309
Hinzufügen und Konfigurieren einer Hochgeschwindigkeitszähler-Benutzerunterbrechung (HSC) ... 310
HSC-Unterbrechungseigenschaften ... 310
HSC-Unterbrechungsparameter ... 312
Konfigurieren eines programmierbaren Endschalters (PLS) ... 313
Beispiel: So erstellen Sie ein Hochgeschwindigkeitsprogramm (HSC) ... 315
Erstellen Sie ein Kontaktplan-Diagramm und fügen Sie Variablen hinzu ... 316
Den HSC-Variablen Werte zuweisen ... 318
Zuweisen von Variablen und Erstellen des Programms ... 319
Testen des Programms und Ausführen des Hochgeschwindigkeitszählers ... 321
Add a Programmable Limit Switch (PLS) function ... 323
Beispiel: Programmierbarer Endschalter (PLS) aktiviert ... 324

Kapitel 15

HSCE-Anweisungen

HSCE ... 327
HSCE_CFG ... 330
HSCE_CFG_PLS ... 332
HSCE_READ_STS ... 334
HSCE_SET_STS ... 335
HSCE-Fehlercodes ... 338

Kapitel 16

Eingabe-/Ausgabeanweisungen

LCD ... 341
LCD_BKLT_REM (Remote-LCD-Hintergrundbeleuchtung) ... 343
LCD_REM (Remote-LCD) ... 345
RHC (Hochgeschwindigkeits-Taktgeber lesen) ... 348

RPC (Prüfsumme des Benutzerprogramms lesen) 350
DLG (Datenerfassung) 351
IIM (sofortige Eingabe) 353
IOM (sofortige Ausgabe) 355
KEY_READ (Lesen der Tasten auf LCD-Modul) 357
KEY_READ_REM (Tasten für Remote-LCD lesen) 359
MM_INFO (Speichermodulinformationen) 361
Datentyp MMINFO 363
MODULE_INFO 364
MODULE_INFO - Informationen zum Plug-in- und Erweiterungsmodul 368
MODULE_INFO-Anweisung für Timing-Diagramme 370
PLUGIN_INFO (Plug-in-Informationen) 375
PLUGIN_READ (schreibgeschütztes Plug-in) 378
PLUGIN_RESET (Plug-in zurücksetzen) 379
PLUGIN_WRITE (Plugin schreiben) 381
RCP (Rezept) 383
RTC_READ (Lesen der Echtzeituhr) 385
RTC_SET (Echtzeituhr einstellen) 387
SYS_INFO 389
Datentyp SYS_INFO 391
TRIMPOT_READ (read trimpot) 392
Definition der Trimpot-ID 393

Kapitel 17

Unterbrechungsanweisungen

STIS (Zeitlich festgelegten Start auswählen) 395
UIC (Interrupt-Lost-Bit löschen) 397
UID (Interrupt deaktivieren) 398
UIE (Interrupt aktivieren) 400
UIF (Ausstehenden Interrupt annullieren) 402

Kapitel 18

Anweisungen für Bewegungssteuerung

Allgemeine Regeln für Bewegungssteuerungs-Funktionsblöcke 406
Parameterdetails von Bewegungssteuerungs-Funktionsblock 408
Bewegungssteuerungs-Achsenstatus 408
Parameternummern von Bewegungssteuerungs-Funktionsblock 410
Fehler-IDs von Bewegungssteuerungs-Funktionsblock 411
Achsfehler-Szenarios 413
Datentyp AXIS_REF 413
Datentyp FB_AXIS_REF 414
Achsenvariablen 415
MC_AbortTrigger (Bewegungssteuerungs-Abbruchauslöser) 416

MC_Halt (Bewegungssteuerungs-Halt) 418
MC_Home (Bewegungssteuerung – Ausgangsstellung) 421
MC_MoveAbsolute (Bewegungssteuerung absolute Bewegung) 424
MC_MoveRelative (Bewegungssteuerung relative Bewegung) 428
MC_MoveVelocity (Bewegungssteuerung – Bewegungsgeschwindigkeit) 431
MC_Power (Bewegungssteuerung – Leistung) 435
MC_ReadActualPosition (Servosteuerung liest tatsächliche Position) 439
MC_ReadActualVelocity (Servosteuerung liest tatsächliche Geschwindigkeit) 442
MC_ReadAxisError (Servosteuerung liest Achsenfehler) 445
AxisErrorID-Fehlercodes 447
MC_ReadBoolParameter (Servosteuerung liest BOOL-Parameter) 449
MC_ReadParameter (Servosteuerung liest Parameter) 451
MC_ReadStatus (Servosteuerung liest Status) 453
MC_Reset (Zurücksetzen der Servosteuerung) 457
MC_SetPosition (Servosteuerung legt Position fest) 460
MC_Stop (Servosteuerungsstopp) 462
MC_TouchProbe (Servosteuerungs-Berührungssensor) 465
MC_WriteBoolParameter (Servosteuerung schreibt BOOL-Parameter) 469
MC_WriteParameter (Servosteuerung schreibt Parameter) 472

Kapitel 19

Anweisungen für die Prozesssteuerung

DERIVATE 475
FFL (FIFO laden) 477
FFL- und FFU-Anweisung für Timing-Diagramme 482
FFU (FIFO entladen) 486
HYSTER (Hysterese) 490
INTEGRAL 492
AND 497
LFL(LIFO-Last) 498
LFU(LIFO-Entlastung) 500
PWM (Impulsbreitenmodulation) 502
SCALER (Skalieren) 505
STACKINT (Ganzzahl-Stack) 507
SCL 509
Beispiele für Timingdiagramme von SCL-Anweisungen 512
TND (aktuelles Programm wird angehalten) 516
LIMIT (Grenzwerttest) 517

Kapitel 20

Anweisungen zur Programmsteuerung

AFI (Always False) 521
NOP (No Operation) 521
SUS (Ruhezustand) 521

Kapitel 21

Anweisung „Proportional-Integral-Differential“ (PID)

Was ist eine „Proportional-Integral-Differential“-Steuerung (PID)? . 525
IPIDCONTROLLER (proportionaler/integraler/differentieller Controller) 525
Datentyp GAIN_PID 529
Datentyp AT_Param 530
So implementiert der Funktionsblock IPIDController die PID-Steuerung 530
Vorgang des Funktionsblocks IPIDController 531
Verwenden der Anweisung „Proportional-Integral-Differential“ (PID) 532
Verwenden der Selbsteinstellung mit dem IPIDController-Funktionsblock 535
Selbsteinstellung in Systemen erster und zweiter Ordnung ... 535
Konfigurieren der Selbsteinstellung 536
Verwenden Sie die auswählbare, zeitlich festgelegte Unterbrechung (STI) mit Selbsteinstellung 541
Beispiel: IPIDController mit Selbsteinstellung 542
Beispiel: So erzeugen Sie eine Rückmeldungsschleife für den bearbeiteten Wert 543
Beispiel: So fügen Sie einen UDFB zu einem PID-Programm hinzu 544
Beispiel: So erstellen Sie ein IPIDController-Programm zur Temperatursteuerung 545
Beispiel: So erstellen Sie ein IPIDController-Programm zur Steuerung des Wasserversorgungspegels 547
PID (Proportional-Integral-Differential) 550
PID-Anweisung für Maschinenstatus 554
PID-Anweisung für Timing-Diagramme 554

Kapitel 22

Anweisungen für Echtzeituhr (RTC)

RTC_READ (Lesen der Echtzeituhr) 559
RTC_SET (Echtzeituhr einstellen) 561

Kapitel 23

Socket-Anweisungen

SOCKET_ACCEPT 566
SOCKET_CREATE 570
SOCKET_DELETE 575

SOCKET_DELETEALL 577
SOCKET_INFO 579
SOCKET_OPEN 584
SOCKET_READ 588
SOCKET_WRITE 592
Datentyp SOCKADDR_CFG 596
Datentyp SOCK_STATUS 596
Socket-Fehlercodes 597
Statusbits für Socket-Anweisungen 601
Socket-Anweisung für Timing-Diagramme 601
Socket-Anweisung für Transaktionsdiagramme 609
Transaktionen für TCP-Client 609
Transaktionen für TCP-Server 609
Transaktionen für UDP mit geöffneter Verbindung 609
Transaktionen für UDP ohne geöffnete Verbindungen 610
Maschinendiagramme für TCP 611
Maschinendiagramme für UDP 612

Kapitel 24

Anweisungen zur Zeichenfolgenbearbeitung

ASCII 615
CHAR (ASCII-Code zu Zeichenfolge) 617
DELETE (Löschen einer Teilzeichenfolge) 618
FIND (Teilzeichenfolge suchen) 620
INSERT (Zeichenfolge einfügen) 622
LEFT (extrahiert den linken Teil einer Zeichenfolge) 624
MID (Extrahiert den mittleren Teil einer Zeichenfolge) 626
MLEN (Zeichenfolgenlänge) 628
RIGHT (rechten Teil einer Zeichenfolge extrahieren) 629
REPLACE (Ersetzen einer Teilzeichenfolge) 631

Kapitel 25

Timer-Anweisungen

Konfiguration der Timer-Anweisungen 635
TOF (Timer, Ausschaltverzögerung) 635
TON (Timer, Einschaltverzögerung) 637
TONOFF (Zeitverzögerung, eingeschaltet, ausgeschaltet) 639
TP (zeitliche Impulssteuerung) 642
RTO (Remanenter Timer, Einschaltverzögerung) 644
DOY (Jahreseinstellung für Echtzeituhr) 646
TDF (Zeitdifferenz) 648
TOW (Woche für Echtzeituhr überprüfen) 650

Index

In diesem Handbuch

Dieses Handbuch enthält Referenzinformationen zum Befehlssatz, der für die Entwicklung von Programmen zur Verwendung in Micro800®-Steuerungssystemen verfügbar ist. Der Anweisungssatz umfasst Unterstützung für die Programmiersprachen Strukturierter Text (ST), Kontaktplan (KOP) und Funktionsblockdiagramm (FBD). Zusätzlich werden die in der Connected Components Workbench™-Entwicklungsumgebung unterstützten Kontaktplan-Elemente definiert.

Unterstützte Controller

Connected Components Workbench bietet Konfigurationsunterstützung für die folgenden Micro800-Controller:

- 2080-LC10-12AWA
- 2080-LC10-12DWD
- 2080-LC10-12QBB
- 2080-LC10-12QWB
- 2080-LC20-20AWB
- 2080-LC20-20QBB
- 2080-LC20-20QWB
- 2080-LC30-10QVB
- 2080-LC30-10QWB
- 2080-LC30-16AWB
- 2080-LC30-16QVB
- 2080-LC30-16QWB
- 2080-LC30-24QBB
- 2080-LC30-24QVB
- 2080-LC30-24QWB
- 2080-LC30-48AWB
- 2080-LC30-48QBB
- 2080-LC30-48QVB
- 2080-LC30-48QWB
- 2080-LC50-24AWB
- 2080-LC50-24QBB
- 2080-LC50-24QVB
- 2080-LC50-24QWB
- 2080-LC50-48AWB
- 2080-LC50-48QBB
- 2080-LC50-48QVB
- 2080-LC50-48QWB
- 2080-LC50-48QWB - SIM
- 2080-L50E-24AWB
- 2080-L50E-24QBB
- 2080-L50E-24QVB
- 2080-L50E-24QWB
- 2080-L50E-48AWB
- 2080-L50E-48QBB
- 2080-L50E-48QVB
- 2080-L50E-48QWB
- 2080-L70E-24AWB
- 2080-L70E-24QBB
- 2080-L70E-24QBBN
- 2080-L70E-24QWB
- 2080-L70E-24QWBN
- 2080-LC70-24AWB
- 2080-LC70-24QBB
- 2080-LC70-24QWB

Zusätzliche Ressourcen

Diese Dokumente enthalten zusätzliche Informationen zu ähnlichen Rockwell Automation®-Produkten.

Ressource	Beschreibung

Ressource	Beschreibung
Richtlinien zur industriellen automatischen Verdrahtung undErdung, Veröffentlichung 1770-4.1, verfügbar unter http://literature.rockwellautomation.com/idc/groups/literature/documents/in/1770-in041_-en-p.pdf.	Enthält allgemeine Richtlinien zur Installation von Rockwell Automation Industriesystemen.
Produktzertifizierungs-Website: http://www.ab.com	Enthält Konformitätserklärungen, Zertifikate und andere Zertifizierungsdetails.
Migrationshandbuch von MicroLogix-Controllern zu Micro800-Controllerm, verfügbar unter https://literature.rockwellautomation.com/idc/groups/literature/documents/rm/2080-rm002_-en-e.pdf	Zeigt an, wie Sie mithilfe der Softwaretools einen geeigneten Micro800-Controller auswählen und die MicroLogix-Programme für die Arbeit mit dem Micro800-Controller konvertieren.
Anfängerpaket für Micro800-Controller – Schnellstart, verfügbar unter https://literature.rockwellautomation.com/idc/groups/literature/documents/qs/2080-qs004_-en-e.pdf	Zeigt die Verwendung eines Micro800-Controllers mit einem PanelView 800-Terminal.
Konfigurieren von Micro800-Controllern auf FactoryTalk-Gateway, verfügbar unter https://literature.rockwellautomation.com/idc/groups/literature/documents/qs/2080-qs005_-en-e.pdf	Bietet Kurzanleitungen zum Konfigurieren eines Micro800-Controllers auf FactoryTalk Linx Gateway.

Sie können Veröffentlichungen unter http://www.rockwellautomation.com/literature anzeigen oder herunterladen. Um eine gedruckte Version der technischen Dokumentation zu bestellen, wenden Sie sich an Ihren Rockwell Automation Vertreter oder Vertriebsmitarbeiter vor Ort.

Rechtliche Hinweise

Rockwell Automation veröffentlicht auf der Seite Rechtliche Hinweise auf der Website von Rockwell Automation rechtliche Hinweise wie Datenschutzrichtlinien, Lizenzvereinbarungen, Bekanntmachungen von Marken sowie weitere allgemeine Geschäftsbedingungen.

Vertrag über die Software und Clouddienste

Prüfen und akzeptieren Sie den Rockwell Automation Vertrag über die Software und Clouddienste hier.

Open Source-Software-Lizenzen

Die in diesem Produkt enthaltene Software enthält urheberrechtlich geschützte Software, die unter einer oder mehreren Open-Source-Lizenzen lizenziert ist.

Eine vollständige Liste der in diesem Produkt verwendeten Open Source-Software und die entsprechenden Lizenzen finden Sie in der Datei oss_license.txt, die sich auf Ihrer Festplatte im Ordner OPENSOURCE dieses Produkts befindet. Diese Datei ist in folgende Abschnitte unterteilt:

- Komponenten
  Enthält den Namen der Open Source-Komponente, ihre Versionsnummer und ihren Lizenztyp.
- Copyright-Text
  Enthält den Namen der Open Source-Komponente, ihre Versionsnummer und die Copyright-Erklärung.
- Lizenzen
  Enthält den Namen der Lizenz, die Liste der Open Source-Komponenten, die die Lizenz angeben, sowie die Bedingungen der Lizenz.

Der Standard-Speicherort der Datei ist:

C:\Program Files (x86)\Common Files\Rockwell\Help\<*product name*>\Release Notes\OPENSOURCE\oss_licenses.txt.

Den zugehörigen Quellcode für die in diesem Produkt enthaltenen Open-Source-Pakete finden Sie auf den entsprechenden Websites des Produkts. Wenn Sie den vollständigen zugehörigen Quellcode erhalten möchten, wenden Sie sich über das **Kontaktformular** auf der Rockwell Automation-Website an Rockwell Automation:
http://www.rockwellautomation.com/global/about-us/contact/contact.page
Bitte geben Sie „Open-Source“ als Teil der Anfrage ein.

Kommerzielle Software-Lizenzen

In der folgenden Tabelle sind die kommerziell lizenzierten Softwarekomponenten in Connected Components Workbench aufgeführt.

Komponente	Urheberrecht
Actipro v17.1.0651	Copyright (c) 2002–2013 Actipro Software LLC
ag-grid v21.0.1	Copyright (c) 2015–2020 AG GRID LTD

Suchen von Informationen zu Anweisungen und Strompfadelementen

Connected Components Workbench enthält einen umfassenden Anweisungssatz mit Strukturen und Arrays sowie Entwicklungsumgebungen für Programme des Typs „Strompfadlogik“, „Strukturierter Text“, „Funktionsblockdiagramm“ und „benutzerdefinierter Baustein“.

Zusätzlich bietet Connected Components Workbench benutzerdefinierte Konfigurationstools für Micro800-Controller, PowerFlex®-Antriebe, ein Sicherheitsrelaisgerät, PanelView™ Component-Grafikterminals und Optionen für serielle und Netzwerkanschlüsse.

Um Informationen zu bestimmten Anweisungen wie eine Beschreibung, Parameterdetails und Sprachbeispiele zu erhalten, suchen Sie die Anweisung im Inhaltsverzeichnis oder unter den folgenden Referenzthemen:

- Anweisungsreferenz in alphabetischer Reihenfolge auf seite 22

Eine Beschreibung der in Connected Components Workbench verwendeten Strompfadelemente finden Sie in folgendem Abschnitt:

- Sprachreferenz des Kontaktplans (KOP) auf seite 29

Anweisungsblöcke

Der Anweisungssatz von Connected Components Workbench umfasst IEC 61131-3-kompatible Anweisungsblöcke für Micro800-Controller. Befehlsblöcke wiederum enthalten Operatoren auf seite 19, Funktionen auf seite 19 und Funktionsblöcke auf seite 21.

Operatoren

Der Connected Components Workbench-Anweisungssatz umfasst IEC 61131-3-kompatible Anweisungsblöcke für Micro800-Controller; Operatoren sind im Connected Components Workbench-Anweisungssatz enthalten.

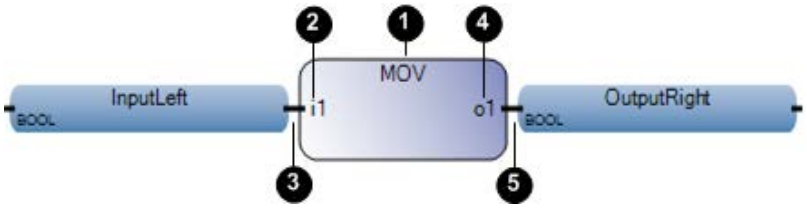
Die grundlegenden logischen Operationen wie Arithmetik, boolesche Logik, Vergleiche und Datenumwandlung.

Funktionen

Funktionen haben einen oder mehrere Eingabeparameter und einen Ausgabeparameter.

Anweisungsblockformat

Ein Anweisungsblock wird durch ein Rechteck dargestellt und verfügt über eine feste Zahl von Ein- und Ausgangsverbindungspunkten. Ein elementarer Anweisungsblock führt eine einzelne Funktion aus.

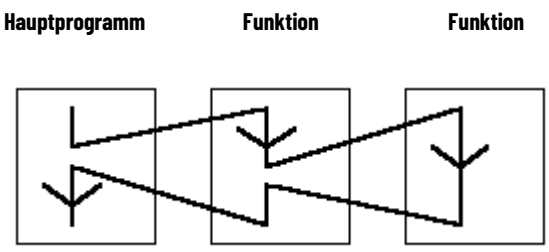


Elementnr.	Element	Beschreibung
1	Blockname	Der Name der vom Anweisungsblock auszuführenden Funktion steht in dem kleinen Rechteck (oben).
2	Eingang	Jede Eingabe eines Anweisungsblocks ist beschriftet und hat einen bestimmten Typ.
3	Eingabeverbindung	Eingaben werden am linken Rand verbunden.
4	Ausgang	Jede Ausgabe eines Anweisungsblocks ist beschriftet und hat einen bestimmten Typ.
5	Ausgabeverbindung	Ausgaben werden am rechten Rand verbunden.

Aufrufen von Funktionen

Connected Components Workbench unterstützt keine rekursiven Funktionsaufrufe. Wenn eine Funktion im Bereich „Funktionen“ sich selbst aufruft oder von einer ihrer aufrufenden Funktionen aufgerufen wird, tritt ein Laufzeitfehler auf. Zudem speichern Funktionen keine lokalen Werte ihrer lokalen Variablen. Da Funktionen nicht instanziiert sind, können sie keine Funktionsblöcke aufrufen.

- Eine Funktion kann von einem Programm, einer Funktion oder einem Funktionsblock aufgerufen werden.
- Jedes Programm in jedem Bereich kann eine oder mehrere Funktionen aufrufen. Eine Funktion kann lokale Variablen haben.
- Eine Funktion hat keine Instanz, d. h., lokale Daten werden nicht gespeichert und gehen für gewöhnlich zwischen zwei Aufrufen verloren.
- Weil die Ausführung einer Funktion von dem übergeordneten Programm gesteuert wird, wird dessen Ausführung bis zur Beendigung der Funktion ausgesetzt.



Definieren von Funktions- und Parameternamen

Die Schnittstelle einer Funktion muss explizit mit einem Typ und einem eindeutigen Namen für jeden aufrufenden Parameter (Eingabe) bzw. Rückgabeparameter (Ausgabe) definiert sein. Eine Funktion hat einen Rückgabeparameter. Der Wert eines Rückgabeparameters für einen Funktionsblock hängt von der jeweils verwendeten Programmiersprache (FBD, KOP, ST) ab.

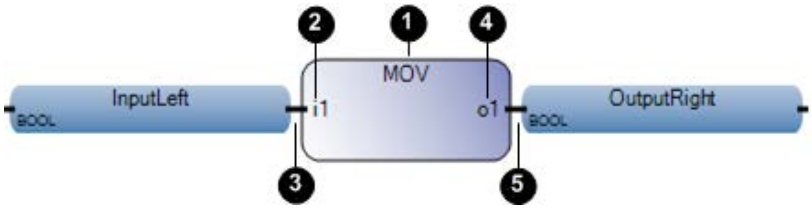
Die Namen von Funktionen und deren Parametern können bis zu 128 Zeichen enthalten. Der Name eines Funktionsparameters kann mit einem Buchstaben oder Unterstrich beginnen, gefolgt von Buchstaben, Zahlen und einzelnen Unterstrichen.

Funktionsblöcke

Ein Funktionsblock ist ein Anweisungsblock, der Eingabeparameter und Ausgabeparameter umfasst und mit internen Daten (Parametern) funktioniert. Die Funktion kann als strukturierter Text, Kontaktplan oder Funktionsblockdiagramm geschrieben werden.

Anweisungsblockformat

Ein Anweisungsblock wird durch ein Rechteck dargestellt und verfügt über eine feste Zahl von Ein- und Ausgangsverbindungspunkten. Ein elementarer Anweisungsblock führt eine einzelne Funktion aus.



Elementnr.	Element	Beschreibung
1	Blockname	Der Name der vom Anweisungsblock auszuführenden Funktion steht in dem kleinen Rechteck (oben).
2	Eingang	Jede Eingabe eines Anweisungsblocks ist beschriftet und hat einen bestimmten Typ.
3	Eingabeverbindung	Eingaben werden am linken Rand verbunden.
4	Ausgang	Jede Ausgabe eines Anweisungsblocks ist beschriftet und hat einen bestimmten Typ.
5	Ausgabeverbindung	Ausgaben werden am rechten Rand verbunden.

Aufrufen eines Funktionsblocks

Wenn in einem Programm ein Funktionsblock aufgerufen wird, wird eine Instanz des Blocks tatsächlich aufgerufen. Die Instanz verwendet denselben Code, aber die Ein- und Ausgabeparameter werden instanziiert, was bedeutet, dass für jede Instanz des Funktionsblocks lokale Variablen kopiert werden.

Die Werte der Variablen einer Funktionsblockinstanz werden von einem Zyklus in den nächsten Zyklus gespeichert.

Ein Funktionsblock kann von einem Programm oder von einem anderen Funktionsblock aufgerufen werden. Er kann nicht von Funktionen aufgerufen werden, weil Funktionen nicht instanziiert werden.

Definieren von Funktionsblock- und Parameternamen

Die Schnittstelle eines Funktionsblocks muss explizit mit einem Typ und einem eindeutigen Namen für jeden seiner aufrufenden Parameter (Eingabe) bzw. seiner Rückgabeparameter (Ausgabe) sein. Funktionsblöcke können mehrere Ausgabeparameter haben. Der Wert eines Rückgabeparameters für einen Funktionsblock hängt von der jeweils verwendeten Programmiersprache (FBD, KOP, ST) ab.

Die Namen von Funktionsblöcken und deren Parametern können bis zu 128 Zeichen verwenden. Der Name eines Funktionsblockparameters kann mit einem Buchstaben oder Unterstrich beginnen, gefolgt von Buchstaben, Zahlen und einzelnen Unterstrichen.

Anweisungssatz in alphabetischer Reihenfolge

Die folgende Tabelle enthält die in Connected Components Workbench verfügbaren Anweisungen der Micro800-Steuerung und die ihnen zugeordneten Anweisungen im Logix-Thema in alphabetischer Reihenfolge.

Anweisung	Zugewiesene Anweisung (Logix-Thema)	Kategorie	Typ	Beschreibung
- auf seite 97	SUB	Arithmetisch	Operator	Subtrahiert einen Wert vom Typ „Integer“, „Real“ oder „Time“ von einem anderen Wert vom Typ „Integer“, „Real“ oder „Time“.
* auf seite 86	MUL	Arithmetisch	Operator	Multipliziert zwei oder mehr Werte vom Typ „Integer“ oder „Real“.
/ auf seite 79	DIV	Arithmetisch	Operator	Division zweier Werte der Typen „Integer“ und „Real“.
+ auf seite 69	ADD	Arithmetisch	Operator	Addiert zwei oder mehr Werte vom Typ „Integer“, „Real“, „Time“ oder „String“.
< auf seite 246	LES	Vergleichen	Operator	Vergleicht die Eingabewerte „Integer“, „Real“, „Time“, „Date“ und „String“, um zu ermitteln, ob der erste Wert kleiner ist als der zweite Wert.
<= auf seite 248	LEQ	Vergleichen	Operator	Vergleicht die Eingabewerte „Integer“, „Real“, „Time“, „Date“ und „String“, um zu ermitteln, ob der erste Wert kleiner ist als der zweite Wert oder gleich groß.
<> auf seite 249	NEQ	Vergleichen	Operator	Vergleicht die Eingabewerte vom Typ „Integer“, „Real“, „Time“, „Date“ und „String“, um zu ermitteln, ob der erste Wert nicht gleich dem zweiten Wert ist.
= auf seite 241	EQU	Vergleichen	Operator	Testet, ob ein Wert gleich einem anderen Wert ist.
> auf seite 244	GRT	Vergleichen	Operator	Vergleicht die Eingabewerte „Integer“, „Real“, „Time“, „Date“ und „String“, um zu ermitteln, ob der erste Wert größer ist als der zweite Wert.
>= auf seite 245	GEQ	Vergleichen	Operator	Vergleicht die Eingabewerte „Integer“, „Real“, „Time“, „Date“ und „String“, um zu ermitteln, ob der erste Wert größer ist als der zweite Wert oder gleich groß.
ABL auf seite 103	ABL	Übertragungen	Funktionsblock	Ermittelt die Anzahl von Zeichen im Puffer (einschließlich Zeilenendezeichen).

Anweisung	Zugewiesene Anweisung (Logix-Thema)	Kategorie	Typ	Beschreibung
ABS auf seite 65	ABS	Arithmetisch	Funktion	Liefert den absoluten Wert einer Realzahl.
ACB auf seite 109	ACB	Übertragungen	Funktionsblock	Ermittelt die Gesamtanzahl von Zeichen im Puffer.
ACL auf seite 105	ACL	Übertragungen	Funktionsblock	Löscht die Empfangs- und Übertragungspuffer.
ACOS auf seite 67	ACS	Arithmetisch	Funktion	Berechnet den Arkuscosinus einer Realzahl.
ACOS_LREAL auf seite 68	ACOS_LREAL	Arithmetisch	Funktion	Berechnet den Arkuscosinus einer langen Realzahl.
AFI auf seite 521	AFI	Programmsteuerung	Funktion	Deaktiviert vorübergehend einen Strompfad beim Debuggen.
AHL auf seite 107	AHL	Übertragungen	Funktionsblock	Legt Handshake-Leitungen des Modems fest, oder setzt sie zurück.
AND auf seite 150	AND	Boolesche Vorgänge	Operator	Führt eine boolesche AND-Operation zwischen zwei oder mehr Werten aus.
AND_MASK auf seite 125	AND_MASK	Binäre Operationen	Funktion	Führt eine bitweise AND-Operation zwischen zwei ganzzahligen Werten aus.
ANY_TO_BOOL auf seite 257	ANY_TO_BOOL	Datenkonvertierung	Funktion	Konvertiert einen nicht booleschen Wert in einen booleschen Wert.
ANY_TO_BYTE auf seite 258	ANY_TO_BYTE	Datenkonvertierung	Funktion	Konvertiert einen Wert in ein Byte.
ANY_TO_DATE auf seite 259	ANY_TO_DATE	Datenkonvertierung	Funktion	Konvertiert die Datentypen „String“, „Integer“, „Real“ oder „Time“ in den Datentyp „Date“.
ANY_TO_DINT auf seite 261	STOD	Datenkonvertierung	Funktion	Konvertiert einen Wert in einen Wert vom Typ „Double Integer“.
ANY_TO_DWORD auf seite 262	ANY_TO_DWORD	Datenkonvertierung	Funktion	Konvertiert einen Wert in einen Wert vom Typ „Double Word“.
ANY_TO_INT auf seite 263	ACI	Datenkonvertierung	Funktion	Konvertiert einen Wert in einen Wert vom Typ „Integer“.
ANY_TO_LINT auf seite 264	ANY_TO_LINT	Datenkonvertierung	Funktion	Konvertiert einen Wert in einen Long-Integer-Wert.
ANY_TO_LREAL auf seite 265	ANY_TO_LREAL	Datenkonvertierung	Funktion	Konvertiert einen Wert in einen Wert vom Typ „Long Real“.
ANY_TO_LWORD auf seite 266	ANY_TO_LWORD	Datenkonvertierung	Funktion	Konvertiert einen Wert in einen Wert vom Typ „Long Word“.
ANY_TO_REAL auf seite 267	STOR	Datenkonvertierung	Funktion	Konvertiert einen Wert in einen Wert vom Typ „Real“.
ANY_TO_SINT auf seite 268	ANY_TO_SINT	Datenkonvertierung	Funktion	Konvertiert einen Wert in einen Short-Integer-Wert.
ANY_TO_STRING auf seite 269	DTOS	Datenkonvertierung	Funktion	Konvertiert einen Wert in einen Wert vom Typ „String“.
ANY_TO_TIME auf seite 270	ANY_TO_TIME	Datenkonvertierung	Funktion	Konvertiert einen Wert in den Datentyp „Time“.
ANY_TO_UDINT auf seite 271	ANY_TO_UDINT	Datenkonvertierung	Funktion	Konvertiert einen Wert in einen Wert vom Typ „Double Integer“ ohne Vorzeichen.
ANY_TO_UINT auf seite 272	ANY_TO_UINT	Datenkonvertierung	Funktion	Konvertiert einen Wert in einen Wert vom Typ „Integer“ ohne Vorzeichen.
ANY_TO_ULINT auf seite 273	ANY_TO_ULINT	Datenkonvertierung	Funktion	Konvertiert einen Wert in einen Long-Integer-Wert ohne Vorzeichen.
ANY_TO_USINT auf seite 274	ANY_TO_USINT	Datenkonvertierung	Funktion	Konvertiert einen Wert in einen Short-Integer-Wert ohne Vorzeichen.
ANY_TO_WORD auf seite 275	ANY_TO_WORD	Datenkonvertierung	Funktion	Konvertiert einen Wert in einen Wert vom Typ „Word“.
ARD auf seite 111	ARD	Übertragungen	Funktionsblock	Liest Zeichen aus dem Eingabepuffer und legt diese Zeichen in einer Zeichenfolge ab.
ARL auf seite 114	ARL	Übertragungen	Funktionsblock	Liest eine Zeile mit Zeichen aus dem Eingabepuffer und legt diese Zeichen in einer Zeichenfolge ab.
ASCII auf seite 615	ASCII	Zeichenfolgen-Manipulation	Funktion	Gibt den ASCII-Code für Zeichen in Zeichenfolgen zurück. Zeichen -> ASCII-Code
ASIN auf seite 71	ASN	Arithmetisch	Funktion	Berechnet den Arkussinus eines langen Real-Wertes.
ASIN_LREAL auf seite 72	ASN_LREAL	Arithmetisch	Funktion	Berechnet den Arkussinus einer langen Realzahl.
ATAN auf seite 74	ATN	Arithmetisch	Funktion	Berechnet den Arkustangens einer Realzahl.
ATAN_LREAL auf seite 75	ATAN_LREAL	Arithmetisch	Funktion	Berechnet den Arkustangens einer langen Realzahl.
AVERAGE auf seite 277	AVE	Datenbearbeitung	Funktionsblock	Berechnet einen Ausführungsdurchschnitt einer Zahl von einem bestimmten Wert.

Anweisung	Zugewiesene Anweisung (Logix-Thema)	Kategorie	Typ	Beschreibung
AWA auf seite 116	AWA	Übertragungen	Funktion	Schreibt eine Zeichenfolge mit zwei angefügten (benutzerkonfigurierten) Zeichen auf ein externes Gerät.
AWT auf seite 118	AWT	Übertragungen	Funktion	Schreibt Zeichen aus einer Quellzeichenfolge in ein externes Gerät.
BSL auf seite 127	BSL	Binäre Operationen	Funktionsblock	Verschiebt ein Bit in einem Array-Element nach links.
BSR auf seite 130	BSR	Binäre Operationen	Funktionsblock	Verschiebt ein Bit in einem Array-Element nach rechts.
CHAR auf seite 617	CHAR	Zeichenfolgen-Manipulation	Funktion	Gibt eine Zeichenfolge mit einem Zeichen für einen ASCII-Code zurück. ASCII-Code -> Zeichen
COM_IO_WDOG auf seite 163	COM_IO_WDOG	Übertragungen	Funktionsblock	Überwacht die Kommunikation mit dem Controller.
COP auf seite 279	COP	Datenkonvertierung	Funktionsblock	Kopiert Binärdaten des Quellelements in das Zielelement.
COS auf seite 76	COS	Arithmetisch	Funktion	Berechnet den Cosinus einer Realzahl.
COS_LREAL auf seite 78	COS_LREAL		Funktion	Berechnet den Cosinus einer langen Realzahl.
CTD auf seite 251	CTD	Zähler	Funktion	Zählt Ganzzahlen abwärts von einem gegebenen Wert bis 0, in Schritten von 1.
CTU auf seite 253	CTU	Zähler	Funktion	Dieser Block zählt Ganzzahlen aufwärts von 0 bis zu einem gegebenen Wert, in Schritten von 1.
CTUD auf seite 255	CTUD	Zähler	Funktion	Dieser Block zählt Ganzzahlen aufwärts von 0 bis zu einem gegebenen Wert in Schritten von 1, oder abwärts von einem gegebenen Wert bis 0 in Schritten von 1.
DELETE auf seite 618	DELETE	Zeichenfolgen-Manipulation	Funktion	Löscht Zeichen aus einer Zeichenfolge.
DERIVATE auf seite 475	DERIVATE	Prozesssteuerung	Funktionsblock	Differenzierung eines Real-Wertes über eine definierte Zykluszeit.
DLG auf seite 351	DLG	Eingabe/Ausgabe	Funktionsblock	Schreibt Variablenwerte aus der Laufzeit-Engine in eine Datenprotokollierungsdatei auf einer SD-Karte.
DOY auf seite 646	DOY	Zeit	Funktion	Aktiviert eine Ausgabe, wenn der Wert der Echtzeituhr sich im Bereich der Jahres- und Zeiteinstellung befindet.
EXPT auf seite 81	EXPT	Arithmetisch	Funktion	Berechnet den Wert des Typs „Real" einer Basiszahl potenziert mit dem Exponenten „Integer".
F_TRIG auf seite 145	OSF	Boolesche Vorgänge	Funktionsblock	Ermittelt die fallende Flanke einer booleschen Variablen.
FFL auf seite 486	FFL	Prozesssteuerung	Funktionsblock	Lädt 8-Bit-, 16-Bit-, 32-Bit- oder 64-Bit-Daten in ein vom Benutzer erzeugtes Array, einen so genannten FIFO-Stapel.
FFU auf seite 486	FFU	Prozesssteuerung	Funktionsblock	Entlädt 8-Bit-, 16-Bit-, 32-Bit- oder 64-Bit-Daten aus einem vom Benutzer erzeugten Array, einem so genannten FIFO-Stapel (First-in-First-Out), in der gleichen Reihenfolge, in der die Daten mithilfe der FFL-Anweisung geladen wurden.
FIND auf seite 620	FIND	Zeichenfolgen-Manipulation	Funktion	Sucht die Position von Teilzeichenfolgen innerhalb von Zeichenfolgen und stellt diese bereit.
HSC auf seite 290	HSC	Eingabe/Ausgabe	Funktionsblock	HSC wendet hohe und niedrige Voreinstellungen und Ausgabequellenwerte auf den Hochgeschwindigkeitszähler an.
HSC_SET_STS auf seite 306	HSC_SET_STS	Eingabe/Ausgabe	Funktionsblock	HSC_SET_STS legt den HSC-Zählstatus manuell fest oder setzt ihn zurück.
HSCE auf seite 327	HSCE	Eingabe/Ausgabe	Funktionsblock	Akkumulatorwert starten, stoppen, lesen des HSCE.
HSCE_CFG auf seite 330	HSCE_CFG	Eingabe/Ausgabe	Funktionsblock	HSCE_CFG ist die Konfiguration für Hochgeschwindigkeitszähler.
HSCE_CFG_PLS auf seite 332	HSCE_CFG_PLS	Eingabe/Ausgabe	Funktionsblock	HSCE_CFG_PLS ist die PLS-Konfiguration für Hochgeschwindigkeitszähler.
HSCE_READ_STS auf seite 334	HSCE_READ_STS	Eingabe/Ausgabe	Funktionsblock	HSCE_READ_STS liest den Status des Hochgeschwindigkeitszählers aus.
HSCE_SET_STS auf seite 335	HSCE_SET_STS	Eingabe/Ausgabe	Funktionsblock	HSCE_SET_STS stellt/setzt den Status des Hochgeschwindigkeitszählers manuell ein/zurück
HYSTER auf seite 490	HYSTER	Prozesssteuerung	Funktionsblock	Boolesche Hysterese bei Unterschieden von reellen Werten

Anweisung	Zugewiesene Anweisung (Logix-Thema)	Kategorie	Typ	Beschreibung
IIM auf seite 353	IIM	Eingabe/Ausgabe	Funktionsblock	Aktualisiert die Eingaben vor normalem Ausgabescan.
INSERT auf seite 622	INSERT	Zeichenfolgen-Manipulation	Funktion	Fügt Teilzeichenfolgen auf benutzerdefinierten Positionen innerhalb von Zeichenfolgen ein.
INTEGRAL auf seite 492	INTEGRAL	Prozesssteuerung	Funktionsblock	Integriert einen Real-Wert während der definierten Zykluszeit.
IOM auf seite 355	IOM	Eingabe/Ausgabe	Funktionsblock	Aktualisiert die Ausgaben vor normalem Ausgabescan.
IPIDCONTROLLER auf seite 525	IPIDCONTROLLER	Prozesssteuerung	Funktionsblock	Konfiguriert und steuert die Eingänge und Ausgänge für die Proportional-Integral-Differential (PID)-Logik.
KEY_READ auf seite 357		Eingabe/Ausgabe	Funktionsblock	Nur Micro810. Liest den Tastenstatus auf dem optionalen LCD-Modul, wenn die Benutzeranzeige aktiv ist.
KEY_READ_REM auf seite 359	KEY_READ_REM	Eingabe/Ausgabe	Funktionsblock	Nur Micro820. Liest den Tastenstatus auf dem optionalen Remote-LCD-Modul, wenn die Benutzeranzeige aktiv ist.
LCD auf seite 341		Eingabe/Ausgabe	Funktion	Nur Micro810. Zeigt eine Zeichenfolge oder eine Zahl auf einem LCD-Modul an.
LCD_BKLT_REM auf seite 343	LCD_BKLT_REM	Eingabe/Ausgabe	Funktion	Legt die Parameter der Remote-LCD-Hintergrundbeleuchtung in einem Benutzerprogramm fest.
LCD_REM auf seite 345	LCD_REM	Eingabe/Ausgabe	Funktion	Zeigt benutzerdefinierte Nachrichten für das Remote-LCD-Modul an.
LEFT auf seite 624	LEFT	Zeichenfolgen-Manipulation	Funktion	Extrahiert Zeichen von der linken Seite einer Zeichenfolge.
LFL(LIFO-Last) auf seite 498	LFL	Prozesssteuerung	Funktionsblock	Lädt 8-Bit-, 16-Bit-, 32-Bit- oder 64-Bit-Daten in ein vom Benutzer erzeugtes Array, einen so genannten LIFO-Stapel.
LFU(LIFO-Entlastung) auf seite 500	LFU	Prozesssteuerung	Funktionsblock	Entlädt 8-Bit-, 16-Bit-, 32-Bit- oder 64-Bit-Daten aus einem vom Benutzer erzeugten Datenfeld, einem so genannten LIFO-Stapel (Last in – First out), in der gleichen Reihenfolge, in der die Daten mithilfe der LFL-Anweisung geladen wurden.
LIM_ALRM auf seite 61	LIM	Alarm	Funktionsblock	Dieser Alarm blockiert Hysterese bei einem „Real"-Wert für Ober- und Untergrenzen.
LIMIT auf seite 517	LIMIT	Prozesssteuerung	Funktion	Beschränkt ganzzahlige Werte auf ein gegebenes Intervall.
LOG auf seite 82	LOG	Arithmetisch	Funktion	Berechnet den Logarithmus (Basis 10) einer Realzahl.
MAX auf seite 285	MAX	Datenbearbeitung	Funktion	Berechnet das Maximum zweier ganzzahliger Werte.
MC_AbortTrigger auf seite 416	MC_AbortTrigger	Bewegung	Funktionsblock	Bricht Bewegungsfunktionsblöcke ab, die mit Auslöseereignissen verbunden sind.
MC_Halt auf seite 418	MC_Halt	Bewegung	Funktionsblock	Befiehlt das Anhalten einer gesteuerten Bewegung unter normalen Betriebsbedingungen.
MC_Home auf seite 421	MAH	Bewegung	Funktionsblock	Befiehlt der Achse, die Sequenz <*search home*> auszuführen.
MC_MoveAbsolute auf seite 424	MAM	Bewegung	Funktionsblock	Befiehlt eine gesteuerte Bewegung zu einer angegebenen absoluten Position.
MC_MoveRelative auf seite 428	MC_MoveRelative	Bewegung	Funktionsblock	Befiehlt eine gesteuerte Bewegung eines angegebenen Abstands im Verhältnis zur tatsächlichen Position zum Zeitpunkt der Ausführung.
MC_MoveVelocity auf seite 431	MCD	Bewegung	Funktionsblock	Befiehlt eine gesteuerte Endlosbewegung mit einer angegebenen Geschwindigkeit.
MC_Power auf seite 435	MSO	Bewegung	Funktionsblock	Steuert die Leistungsstufe (EIN oder AUS).
MC_ReadActualPosition auf seite 442	MC_ReadActualPosition	Bewegung	Funktionsblock	Gibt die tatsächliche Position der Feedbackachse zurück.
MC_ReadActualVelocity auf seite 442	MC_ReadActualVelocity	Bewegung	Funktionsblock	Gibt die tatsächliche Geschwindigkeit der Feedbackachse zurück.
MC_ReadAxisError auf seite 445	MC_ReadAxisError	Bewegung	Funktionsblock	Liest die Achsfehler, die in keinem Zusammenhang mit den Bewegungssteuerungs-Funktionsblöcken stehen.

Anweisung	Zugewiesene Anweisung (Logix-Thema)	Kategorie	Typ	Beschreibung
MC_ReadBoolParameter auf seite 449	MC_ReadBoolParameter	Bewegung	Funktionsblock	Gibt den Wert eines anbieterspezifischen Parameters vom Typ BOOL zurück.
MC_ReadParameter auf seite 451	MC_ReadParameter	Bewegung	Funktionsblock	Gibt den Wert eines herstellerspezifischen Parameters vom Typ REAL zurück.
MC_ReadStatus auf seite 453	MC_ReadStatus	Bewegung	Funktionsblock	Gibt den Achsstatus hinsichtlich der aktuell erfolgenden Bewegung zurück.
MC_Reset auf seite 457	MAFR	Bewegung	Funktionsblock	Setzt den Achszustand von „ErrorStop" auf „StandStill". Dies geschieht durch das Zurücksetzen aller internen achsbezogenen Fehler.
MC_SetPosition auf seite 460	MRP	Bewegung	Funktionsblock	Verschiebt das Koordinatensystem einer Achse durch Bearbeiten der tatsächlichen Position.
MC_Stop auf seite 462	MAS	Bewegung	Funktionsblock	Befiehlt einen kontrollierten Bewegungsstopp und ändert den Zustand der Achse in „Stoppen".
MC_TouchProbe auf seite 465	MC_TouchProbe	Bewegung	Funktionsblock	Zeichnet eine Achsenposition bei einem Auslöseereignis auf.
MC_WriteBoolParameter auf seite 469	MC_WriteBoolParameter	Bewegung	Funktionsblock	Ändert den Wert eines anbieterspezifischen Parameters vom Typ BOOL.
MC_WriteParameter auf seite 472	MC_WriteParameter	Bewegung	Funktionsblock	Ändert den Wert eines anbieterspezifischen Parameters vom Typ REAL.
MID auf seite 626	MID	Zeichenfolgen-Manipulation	Funktion	Extrahiert Zeichen aus der Mitte einer Zeichenfolge.
MIN auf seite 284	MIN	Datenbearbeitung	Funktion	Berechnet das Minimum zweier ganzzahliger Werte.
MLEN auf seite 628	MLEN	Zeichenfolgen-Manipulation	Funktion	Berechnet die Länge der Zeichenfolge.
MM_INFO auf seite 361	MM_INFO	Eingabe/Ausgabe	Funktionsblock	Liest die Header-Informationen des Speichermoduls.
MOD auf seite 84	MOD	Arithmetisch	Funktion	Führt eine Modulberechnung für ganzzahlige Werte durch.
MODULE_INFO auf seite 364	MODULE_INFO	Eingabe/Ausgabe	Funktionsblock	Liest Modulinformationen aus einem Plug-in- oder Erweiterungsmodul.
MOV auf seite 85	MOV	Arithmetisch	Operator	Weist den Eingabewert der Ausgabe zu.
MSG_CIPGENERIC auf seite 166	MSG	Übertragungen	Funktion	Sendet eine generische explizite CIP-Nachricht.
MSG_CIPSYMBOLIC auf seite 173	MSG_CIPSYMBOLIC	Übertragungen	Funktion	Sendet eine symbolische explizite CIP-Nachricht.
MSG_MODBUS auf seite 177	MSG_MODBUS	Übertragungen	Funktion	Sendet eine Modbus-Nachricht.
MSG_MODBUS2 auf seite 183	MSG_MODBUS2	Übertragungen	Funktion	Sendet eine MODBUS-/TCP-Nachricht über einen Ethernet-Kanal.
MSG_PCCC auf seite 189	MSG_PCCC	Übertragungen	Funktion	Sendet eine PCCC-Nachricht über einen Ethernet-Kanal oder einen seriellen Anschluss.
MUX4B auf seite 159	MUX4B	Boolesch	Funktion	Multiplexer zwischen vier BOOL-Eingängen, gibt einen BOOL-Wert aus.
MUX8B auf seite 156	MUX8B	Boolesch	Funktion	Multiplexer zwischen acht BOOL-Eingängen gibt einen BOOL-Wert aus.
Neg auf seite 88	NEG	Arithmetisch	Operator	Konvertiert einen Wert in einen negativen Wert.
NOP auf seite 521	NOP	Programmsteuerung	Funktion	Fungiert als Platzhalter.
NOT auf seite 152	NOT	Boolesche Vorgänge	Operator	Konvertiert boolesche Werte in negierte Werte.
NOT_MASK auf seite 133	NOT_MASK	Binäre Operationen	Funktion	Ganzzahlige bitweise Negationsmaske; damit wird der Wert eines Parameters umgekehrt.
OR auf seite 150	OR	Boolesche Vorgänge	Operator	Boolesches OR mit mindestens zwei Werten.
OR_MASK auf seite 134	OR_MASK	Binäre Operationen	Funktion	Ganzzahlige bitweise OR-Maske, zur Aktivierung von Bits.
PID auf seite 550	PID	Prozesssteuerung	Funktionsblock	Ausgangsanweisung, die die physischen Eigenschaften, wie etwa Temperatur, Druck, Füllstand oder Durchflussrate, mithilfe von Prozessschleifen steuert.

Anweisung	Zugewiesene Anweisung (Logix-Thema)	Kategorie	Typ	Beschreibung
PLUGIN_INFO auf seite 375	PLUGIN_INFO	Eingabe/Ausgabe	Funktionsblock	Ruft die Modulinformationen von einem generischen Plug-in-Modul ab (ohne Speichermodul).
PLUGIN_READ auf seite 378	PLUGIN_READ	Eingabe/Ausgabe	Funktionsblock	Liest Daten von einem generischen Plug-in-Modul (ohne Speichermodul).
PLUGIN_RESET auf seite 379	PLUGIN_RESET	Eingabe/Ausgabe	Funktionsblock	Setzt ein generisches Plug-in-Modul zurück, Zurücksetzen der Hardware (ohne Speichermodul).
PLUGIN_WRITE auf seite 381	PLUGIN_WRITE	Eingabe/Ausgabe	Funktionsblock	Schreibt Daten auf ein generisches Plug-in-Modul (ohne Speichermodul).
POW auf seite 89	XPY	Arithmetisch	Funktion	Berechnet den Wert einer Realzahl potenziert mit dem Exponenten „Real“.
PWM auf seite 502	PWM	Prozesssteuerung	Funktionsblock	Schaltet den PWM (Impulsbreitenmodulation)-Ausgang für einen konfigurierten PWM-Kanal ON oder OFF.
R_TRIG auf seite 147	OSR	Boolesche Vorgänge	Funktionsblock	Ermittelt die steigende Flanke einer booleschen Variablen.
RAND auf seite 90	RAND	Arithmetisch	Funktion	RAND liefert nach dem Zufallsprinzip ausgewählte ganzzahlige Werte aus einem definierten Bereich.
RCP auf seite 383	RCP	Eingabe/Ausgabe	Funktionsblock	Liest/schreibt Rezeptdaten auf und von einer SD-Speicherkarte.
REPLACE auf seite 631	REPLACE	Zeichenfolgen-Manipulation	Funktion	Ersetzt Teile einer Zeichenfolge mit einer neuen Gruppe von Zeichen.
RHC auf seite 348	RHC	Eingabe/Ausgabe	Funktion	Liest den Hochgeschwindigkeits-Taktgeber.
RIGHT auf seite 629	RIGHT	Zeichenfolgen-Manipulation	Funktion	Extrahiert Zeichen aus der rechten Seite einer Zeichenfolge.
ROL auf seite 135	ROL	Binäre Operationen	Funktion	Dreht bei 32-Bit-Ganzzahlen die Ganzzahlenbits nach links.
ROR auf seite 137	ROR	Binäre Operationen	Funktion	Dreht bei 32-Bit-Ganzzahlen die Ganzzahlenbits nach rechts.
RPC auf seite 350	RPC	Eingabe/Ausgabe	Funktion	Liest die Prüfsumme des Benutzerprogramms.
RS auf seite 148	RS	Boolesche Vorgänge	Funktionsblock	Dominantes bistabiles Element zurücksetzen.
RTC_READ auf seite 385	RTC_READ	Eingabe/Ausgabe	Funktionsblock	Liest die Informationen des Echtzeituhr-Moduls (Real-Time Clock, RTC).
RTC_SET auf seite 387	RTC_SET	Eingabe/Ausgabe	Funktionsblock	Legt die Daten der Echtzeituhr (RTC) auf die RTC-Modulinformationen fest.
RTO auf seite 644	RTO	Zeit	Funktionsblock	Remanenzzeit. Erhöht einen internen Timer, wenn der Eingang aktiv ist, setzt aber den internen Timer nicht zurück, wenn der Eingang inaktiv wird.
SCALER auf seite 505	SCP	Prozesssteuerung	Funktionsblock	Skaliert den Eingabewert gemäß dem Ausgabebereich.
SCL auf seite 509	SCL	Prozesssteuerung	Funktionsblock	Rechnet einen nicht skalierten Eingabewert in einen Gleitkommawert in technischen Einheiten um.
SHL auf seite 139	SHL	Binäre Operationen	Funktion	Verschiebt 32-Bit-Ganzzahlen nach links und setzt 0 in das niederwertigste Bit.
SHR auf seite 141	SHR	Binäre Operationen	Funktion	Verschiebt bei 32-Bit-Ganzzahlen Ganzzahlen nach rechts und setzt 0 in das höchstwertige Bit.
SIN auf seite 92	SIN	Arithmetisch	Funktion	Berechnet den Sinus von einer Realzahl.
SIN_LREAL auf seite 94	SIN_LREAL	Arithmetisch	Funktion	Berechnet den Sinus einer langen Realzahl.
SOCKET_ACCEPT auf seite 566	SOCKET_ACCEPT	Übertragungen	Funktionsblock	Akzeptiert eine TCP-Verbindungsanforderung von einem Remote-Ziel und gibt eine Socket-Instanz zurück, die zum Senden und Empfangen von Daten in der neu erstellten Verbindung verwendet wird.
SOCKET_CREATE auf seite 570	SOCKET_CREATE	Übertragungen	Funktionsblock	Erstellt eine Instanz des Sockets und gibt eine Instanznummer zurück, die als Eingabe für nachfolgende Socket-Vorgänge verwendet wird.
SOCKET_DELETE auf seite 575	SOCKET_DELETE	Übertragungen	Funktionsblock	Löscht eine erstellte Socketinstanz.
SOCKET_DELETEALL auf seite 577	SOCKET_DELETEALL	Übertragungen	Funktionsblock	Löscht alle erstellten Socket-Instanzen.

Anweisung	Zugewiesene Anweisung (Logix-Thema)	Kategorie	Typ	Beschreibung
SOCKET_INFO auf seite 579	SOCKET_INFO	Übertragungen	Funktionsblock	Gibt Informationen zum Socket wie z. B. Fehlercodes und Ausführungsstatus zurück.
SOCKET_OPEN auf seite 584	SOCKET_OPEN	Übertragungen	Funktionsblock	Stellt die Verbindung für die angegebene Zieladresse für Transmission Control Protocol-Verbindungen (TCP) her. Für User Datagram Protocol-Verbindungen (UDP) wird dem Socket eine Ziel-IP-Adresse und eine Port-Nummer zugeordnet.
SOCKET_READ auf seite 588	SOCKET_READ	Übertragungen	Funktionsblock	Liest Daten von einem Socket.
SOCKET_WRITE auf seite 592	SOCKET_WRITE	Übertragungen	Funktionsblock	Sendet Daten an einen Socket.
SQRT auf seite 95	SQR	Arithmetisch	Funktion	Berechnet die Quadratwurzel einer reellen Zahl.
SR auf seite 152	SR	Boolesche Vorgänge	Funktionsblock	Dominantes bistabiles Element festlegen.
STACKINT auf seite 507	STACKINT	Prozesssteuerung	Funktionsblock	Verwaltet Stapel ganzzahliger Werte.
STIS auf seite 395	STS	Unterbrechen	Funktion	Startet den Timer für die ausgewählte geplante Benutzerunterbrechung (STI) mit dem Steuerungsprogramm und nicht automatisch.
SUS auf seite 521	SUS	Programmsteuerung	Funktionsblock	Versetzt den Micro800-Controller in den Ruhezustand.
SYS_INFO auf seite 389	SYS_INFO	Eingabe/Ausgabe	Funktionsblock	Liest den Statusdatenblock für den Controller Micro800.
TAN auf seite 98	TAN	Arithmetisch	Funktion	Berechnet den Tangens einer Realzahl.
TAN_LREAL auf seite 100	TAN_LREAL	Arithmetisch	Funktion	Berechnet den Tangens einer langen Realzahl.
TDF auf seite 648	TDF	Zeit	Funktion	Berechnet die Zeitdifferenz zwischen TimeA und TimeB.
TND auf seite 516	TND	Prozesssteuerung	Funktion	Hält den aktuellen Zyklus des Benutzerprogramm-Scans an.
TOF auf seite 635	TOF	Zeit	Funktionsblock	Zeitliche Steuerung der Ausschaltverzögerung. Setzt einen internen Timer um einen vorgegebenen Wert nach oben.
TON auf seite 637	TON	Zeit	Funktionsblock	Zeitliche Steuerung der Einschaltverzögerung. Setzt einen internen Timer um einen vorgegebenen Wert nach oben.
TONOFF auf seite 639	TONOFF	Zeit	Funktionsblock	Verzögert das Einschalten eines Ausgangs in einem Strompfad mit dem Wert TRUE und verzögert anschließend das Ausschalten des Ausgangs im Strompfad mit dem Wert FALSE.
TOW auf seite 650	TOW	Zeit	Funktion	Aktiviert eine Ausgabe, wenn der Wert der Echtzeituhr sich im Bereich der Wochenzeiteinstellung befindet.
TP auf seite 642	TP	Zeit	Funktionsblock	Zeitliche Impulssteuerung. Setzt einen internen Timer bei einer ansteigenden Flanke um einen gegebenen Wert nach oben.
TRIMPOT_READ auf seite 392	TRIMPOT_READ	Eingabe/Ausgabe	Funktionsblock	Liest den TrimPot-Wert aus einem bestimmten TrimPot.
TRUNC auf seite 101	TRN	Arithmetisch	Funktion	Schneidet Realzahlen ab und belässt nur eine Ganzzahl.
TTABLE auf seite 154	TTABLE	Boolesch	Funktion	Stellt den Wert der Ausgabe entsprechend einer Kombination von Eingangswerten bereit.
UIC auf seite 397	UIC	Unterbrechen	Funktion	Löscht das verlorene Bit für die ausgewählte Benutzerunterbrechung.
UID auf seite 398	UID	Unterbrechen	Funktion	Deaktiviert eine bestimmte Benutzerunterbrechung.
UIE auf seite 400	UIE	Unterbrechen	Funktion	Aktiviert eine bestimmte Benutzereingabe.
UIF auf seite 402	UIF	Unterbrechen	Funktion	Annulliert oder entfernt eine ausstehende Benutzereingabe.
XOR auf seite 151	XOR	Boolesche Vorgänge	Operator	Boolesches exklusives OR mit zwei Werten.
XOR_MASK auf seite 142	XOR_MASK	Binäre Operationen	Funktion	Ganzzahlige exklusive bitweise OR-Maske gibt invertierte Bitwerte zurück.

Sprache des Kontaktplans (KOP)

Ein Kontaktplan (KOP) ist die grafische Darstellung von booleschen Gleichungen, in der Kontakte (Eingabeargumente) mit Spulen (Ausgabeergebnissen) kombiniert werden. Mithilfe von grafischen Symbolen in einem Programmdiagramm (aufgebaut wie der Kontaktplan eines Relaisleiters) beschreibt die KOP-Sprache Tests und Änderungen boolescher Daten.

Grafische KOP-Symbole sind innerhalb des Diagramms als Plan für elektrische Kontakte angeordnet. Der englische Begriff „Ladder“ (Leiter) für Strompfad stammt aus dem Konzept von Stufen, die an beiden Enden mit vertikalen Stromschienen verbunden sind, wobei jede Stufe einen eigenen Kreislauf darstellt.

Connected Components Workbench enthält einen Kontaktplan-Sprach-Editor und unterstützt Elemente auf seite 33 und Befehle, die nur mit der Connected Components Workbench-Software verfügbar sind.

Kontaktplanprogramm (KOP)

Ein Kontaktplan (KOP) ist die grafische Darstellung von booleschen Gleichungen, in der Kontakte (Eingabeargumente) mit Spulen (Ausgabeergebnissen) kombiniert werden. Mithilfe von grafischen Symbolen in einem Programmdiagramm (aufgebaut wie der Kontaktplan eines Relaisleiters) beschreibt die KOP-Sprache Tests und Änderungen boolescher Daten.

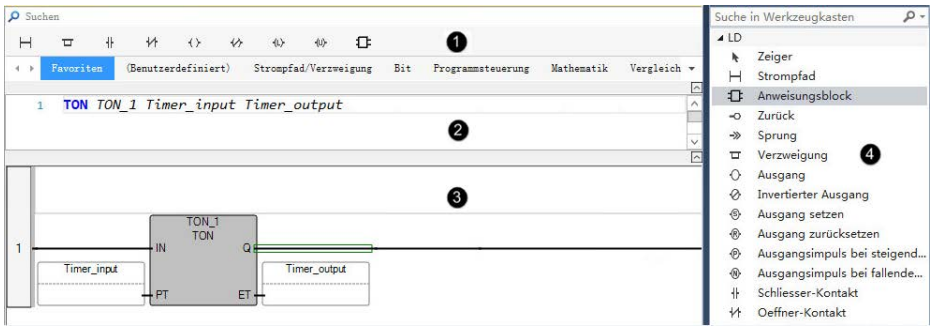
Die KOP-Sprache nutzt grafische Symbole in einem Programmdiagramm (aufgebaut wie der Kontaktplan eines Relaisleiters), um Tests und Änderungen boolescher Daten zu beschreiben.

Connected Components Workbench enthält einen Kontaktplan-Sprach-Editor auf seite 56, der Elemente und Befehle unterstützt, die nur mit der Connected Components Workbench -Software verfügbar sind.

Entwicklungsumgebung für KOP-Programm

Der Sprach-Editor für ein Kontaktplan-Programm (KOP-Programm), in dem eine KOP-Programmorganisationseinheit (POE) entwickelt wird.

Die folgende Abbildung zeigt die Hauptbereiche der Entwicklungsumgebung für das KOP-Programm.



Nr.	Name	Beschreibung
1	Anweisungs-Symbolleiste auf seite 30	Über die Symbolleiste können Sie Anweisungselemente schnell auswählen und im KOP-Grafikeditor platzieren oder per einfachem Klick zum KOP-Texteditor hinzufügen.
2	KOP-Texteditor	In diesem Texteditor können Sie mithilfe der ASCII-Anweisungsmnemonik die Logik bearbeiten.
3	KOP-Grafikeditor	In diesem Editor können Sie mithilfe grafischer Anweisungselemente die Logik bearbeiten.
4	KOP-Toolbox	Mit den Befehlen und Bausteinen können Sie Elemente zum KOP-Grafikeditor hinzufügen.

Anweisungs-Symbolleiste (KOP)

Als Befehlssymbolleiste wird der sekundäre Bereich im Sprach-Editor bezeichnet, der wie eine Symbolleiste funktioniert und dazu verwendet wird, dem Arbeitsbereich des Sprach-Editors Sprachelemente auf seite 30, wie z. B. Befehle, hinzuzufügen. Sie ergänzt die allgemeinen Workbench-Befehle und -Bausteine.

Element	Beschreibung
Reiter	Führt die Anweisungselemente nach Kategorie auf. Klicken Sie auf eine Kategorie, um sich die Anweisungen in dieser Kategorie anzeigen zu lassen. Auch die Pfeiltasten können zur Auswahl der verschiedenen Kategorien verwendet werden. Klicken Sie auf die Pfeile links von den Registerkarten, um zwischen den Kategorien zu wechseln. Klicken Sie auf den nach unten zeigenden Pfeil rechts von den Registerkarten, um eine Liste sämtlicher Kategorien aufzurufen.
Befehle	Führt die Anweisungen auf, die zu der ausgewählten Registerkarten-Kategorie gehören. Eine Kategorie kann auch grundlegende Anweisungselemente enthalten, die oben in der Liste fixiert sind. Klicken Sie auf die Pfeile links von den Anweisungen, um zwischen den Anweisungen zu wechseln.
Suche	Filtert die Anweisungen bei Eingabe von Namen oder beschreibenden Schlüsselwörtern sofort nach Kategorien. Klicken Sie auf einen Kategorienamen, um sich die gefilterten Anweisungen für diese Kategorie anzeigen zu lassen.
Favoriten	Passt eine Liste von Anweisungen an, die schnell gefunden und zum Programm hinzugefügt werden können.

Hinzufügen von Anweisungselementen aus der Anweisungs-Symbolleiste

In der Befehlssymbolleiste auf seite 30 können Sie Befehle im Arbeitsbereich des Sprach-Editors für das Kontaktplan(Ladder Diagram, LD)-Programm suchen, durchsuchen und hinzufügen.

So fügen Sie Befehlselemente aus der Befehls-Symbolleiste zum Sprach-Editor hinzu

1. Klicken Sie auf die Kategorie-Registerkarte, die die Anweisung enthält, die Sie hinzufügen möchten. Sie können mithilfe der Pfeiltasten durch die Registerkarte navigieren oder mithilfe der Drag-and-Drop-Methode eine Neuanordnung vornehmen.
2. Wählen Sie die Anweisung durch eine Drag-and-Drop-Operation oder durch Klicken auf die Anweisung aus.
3. (Optional) Wenn Sie Anweisungen schnell ausfindig machen möchten, klicken Sie in das Suchfeld und geben Sie einen Namen oder ein Stichwort ein, um nach dem Anweisungselement zu suchen. Mit der Taste **Esc** können Sie den Suchmodus beenden und die Navigation über die Pfeiltasten wieder aktivieren.
4. (Optional) Klicken Sie mit der rechten Maustaste auf die Anweisung und wählen Sie **Zu Favoriten hinzufügen** aus, um sie zur Registerkarte **Favoriten** hinzuzufügen, oder **Aus Favoriten entfernen**, um sie zu entfernen. Die Favoriteneinstellung wird automatisch gespeichert.

Elemente des Kontaktplans (KOP)

Kontaktplanelemente (KOP) sind die Komponenten, die Sie für die Erstellung eines Kontaktplan-Programms benötigen. Alle in der folgenden Tabelle aufgelisteten Elemente können zu einem Kontaktplan-Programm innerhalb von Connected Components Workbench hinzugefügt werden.

Element	Beschreibung
Strompfad auf seite 33	Stellt eine Gruppe von Schaltkreiselementen dar, die zur Aktivierung einer Spule führen.
Anweisungsblock (KOP) auf seite 40	Anweisungen umfassen Operatoren, Funktionen und Funktionsblöcke, einschließlich benutzerdefinierter Funktionsblöcke.
Verzweigung auf seite 37	Zwei oder mehr parallele Anweisungen.
Spule auf seite 40	Stellt die Zuweisung der Ausgänge oder internen Variablen dar. In einem KOP-Programm stellt eine Spule eine Aktion dar.
Kontakt auf seite 48	Stellt den Wert oder die Funktion eines Eingangs oder einer internen Variable dar.
Rücksprung auf seite 53	Stellt das bedingte Ende einer Diagrammausgabe dar.
Sprung auf seite 54	Stellen die bedingte und unbedingte Logik im LD-Programm dar, das die Ausführung von Plänen steuert.

Strompfad

Strompfade sind grafische Komponenten von KOP-Programmen. Sie stellen eine Gruppe von Stromkreiselementen dar, die zur Aktivierung einer Spule führen. Strompfade können mittels Bezeichnungen im Plan gekennzeichnet werden. Bezeichnungen steuern zusammen mit Sprüngen die Ausführung eines Plans. Kommentare sind frei formatierter Text, den Sie über den Strompfad hinzufügen können. Sie dienen Dokumentationszwecken.

Hinzufügen eines Strompfads zu einem KOP-Programm

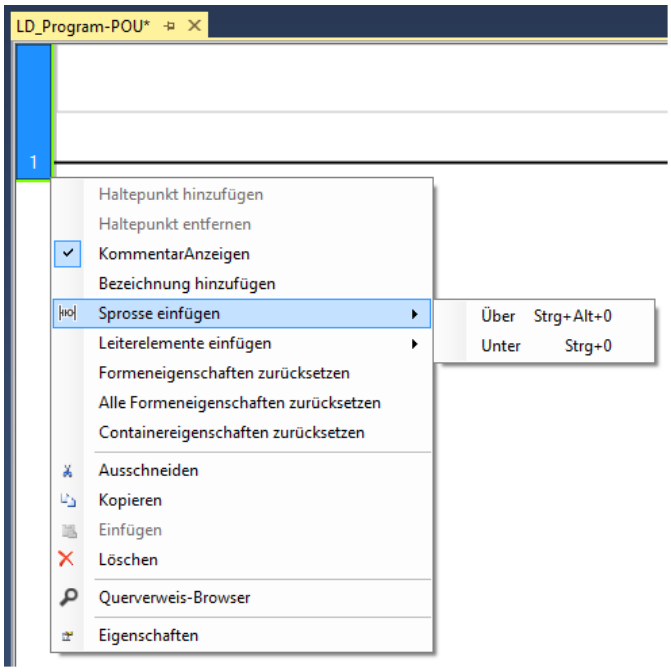
Strompfade auf seite 33 sind grafische Komponenten eines Kontaktplan-Programms (Ladder Diagram, LD). Sie stellen eine Gruppe von Stromkreiselementen dar, die zur Aktivierung einer Spule führen.

In Connected Components Workbench können Sie an folgenden Stellen einen Strompfad zu einem Kontaktplan-Programm (KOP – Ladder Diagram, LD) hinzufügen:

- KOP-Sprach-Editor
- Editor für mehrere Sprachen im Menü „Extras“
- KOP-Toolbox

So fügen Sie ein Strompfadelement zu einem Kontaktplan-Programm hinzu

1. Klicken Sie im KOP-Sprach-Editor mit der rechten Maustaste auf einen vorhandenen Strompfad, und führen Sie dann eine der folgenden Aktionen aus:
   - Klicken Sie auf **Kopieren**, und klicken Sie dann auf **Einfügen**, um eine Kopie des Strompfads in den Sprach-Editor einzufügen.
   - Klicken Sie auf **Strompfad einfügen**, und führen Sie dann eine der folgenden Aktionen aus:
     - Klicken Sie auf **Über**, um den Strompfad über dem ausgewählten Strompfad hinzuzufügen.
     - Klicken Sie auf **Unter**, um den Strompfad unter dem ausgewählten Strompfad hinzuzufügen.



- Wählen Sie einen Strompfad oder ein Element im KOP-Sprach-Editor aus, und drücken Sie dann entweder:
  - **STRG+ALT+O** – um den Strompfad über dem ausgewählten Strompfad hinzuzufügen.
  - **STRG+O** – um den Strompfad unter dem ausgewählten Strompfad hinzuzufügen.
- Wählen Sie einen Strompfad oder ein Element im KOP-Sprach-Editor aus, und drücken Sie dann entweder:
  - Klicken Sie auf **Extras** > **Editor für mehrere Sprachen** > **Strompfad einfügen unter**, um den Strompfad unter dem ausgewählten Strompfad hinzuzufügen.
  - Klicken Sie auf **Extras** > **Editor für mehrere Sprachen** > **Strompfad einfügen über**, um den Strompfad über dem ausgewählten Strompfad hinzuzufügen.

1. (optional) Öffnen Sie die KOP-Toolbox, um das Strompfadelement anzuzeigen.
   - Wenn Sie einen Strompfad unter einem vorhandenen Strompfad einfügen möchten, wählen Sie den Strompfad im KOP-Sprach-Editor aus, und doppelklicken Sie dann in der KOP-Toolbox auf **Strompfad**.

     Wenn vor dem Doppelklick auf **Strompfad** in der KOP-Toolbox kein Element ausgewählt ist, wird der Strompfad unterhalb des letzten Strompfads im KOP-Sprach-Editor eingefügt.
   - Wählen Sie **Strompfad** aus, und ziehen Sie das Element in den KOP-Sprach-Editor.

     Ein Pluszeichen (+) wird im Sprach-Editor für ein gültiges Ablageziel angezeigt. Lassen Sie die Maustaste los, um das Element hinzuzufügen.

     Tipp: Wenn der Kontaktplan mehr als 355 Strompfade enthält, verwenden Sie das nach unten weisende Dreieck statt der Bildlaufleiste, um weitere Strompfade anzuzeigen.

Strompfad-Bezeichnungen

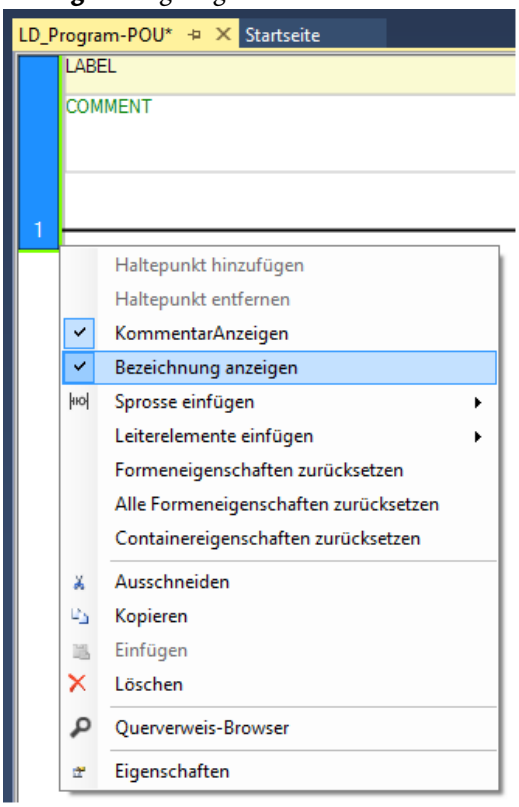
Bezeichnungen können für jeden Strompfad im KOP-Sprach-Editor optional hinzugefügt werden.

Bezeichnungen dürfen unbegrenzt viele Zeichen enthalten und müssen mit einem Buchstaben oder Unterstrich beginnen, gefolgt von Buchstaben, Zahlen und Unterstrichen. Bezeichnungen dürfen keine Leer- oder Sonderzeichen (beispielsweise '+', '-', oder '\') enthalten.

So fügen Sie eine Bezeichnung für einen Strompfad im KOP-Sprach-Editor hinzu

1. Klicken Sie zum Hinzufügen einer Bezeichnung zu einem Strompfad mit der rechten Maustaste auf den Strompfad, um das Menü des KOP-Sprache-Editors zu öffnen, und wählen Sie dann **Bezeichnung anzeigen**.

Der Strompfad wird aktualisiert und enthält nun die Bezeichnung. Im Menü des KOP-Sprach-Editors wird ein Häkchen neben **Bezeichnung anzeigen** eingefügt.



2. Wählen Sie die **Bezeichnung** aus, und geben Sie eine Beschreibung ein.
3. (optional) Um die Bezeichnung zu entfernen, klicken Sie im Menü des KOP-Sprach-Editors erneut auf **Bezeichnung anzeigen**.

Strompfadkommentare

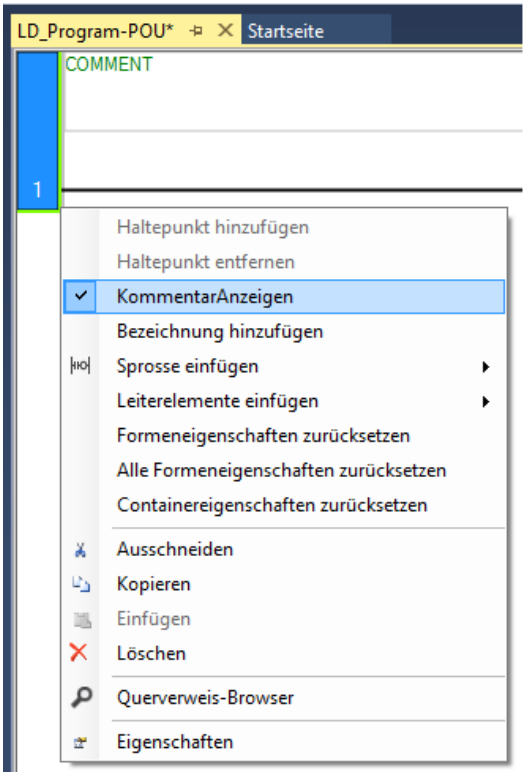
Kommentare können zu jedem Strompfad im KOP-Sprach-Editor optional hinzugefügt werden. Standardmäßig ist ein Kommentar enthalten, wenn Sie im Kontaktplan-Sprach-Editor ein Strompfadelement hinzufügen auf seite 33.

Kommentare werden

- in den Bereich über dem Strompfad eingegeben
- in frei formatiertem Textformat gespeichert
- im Controller gespeichert.

So fügen Sie einen Kommentar für einen Strompfad im KOP-Sprach-Editor hinzu bzw. entfernen ihn

1. Um einen Kommentar zu entfernen, klicken Sie auf den Strompfad, um das Menü des KOP-Sprach-Editors zu öffnen.

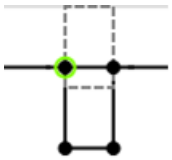


2. Klicken Sie im Menü des KOP-Sprach-Editors auf **Kommentar anzeigen**. Der Kommentar wird aus dem Strompfad gelöscht, und das Häkchen neben **Kommentar anzeigen** im Menü des KOP-Sprach-Editors wird entfernt.
3. (optional) Klicken Sie zum Hinzufügen eines Kommentars zu einem Strompfad im Menü des KOP-Sprach-Editors auf **Kommentar anzeigen**.

Verzweigung

Verzweigungen erstellen alternative Routen für Verbindungen. In einem Kontaktplan(Ladder Diagram, LD)-Sprach-Editor können Sie zu Elementen in einem Strompfad parallele Verzweigungen hinzufügen auf seite 38.

Verzweigungsbeispiel



Hinzufügen einer Verzweigung zu einem KOP-Programm

Verzweigungen auf seite 37 sind grafische Komponenten von Kontaktplan(Ladder Diagram, LD)-Programmen, die alternative Routen für Verbindungen erstellen und parallele Verzweigungen umfassen können.

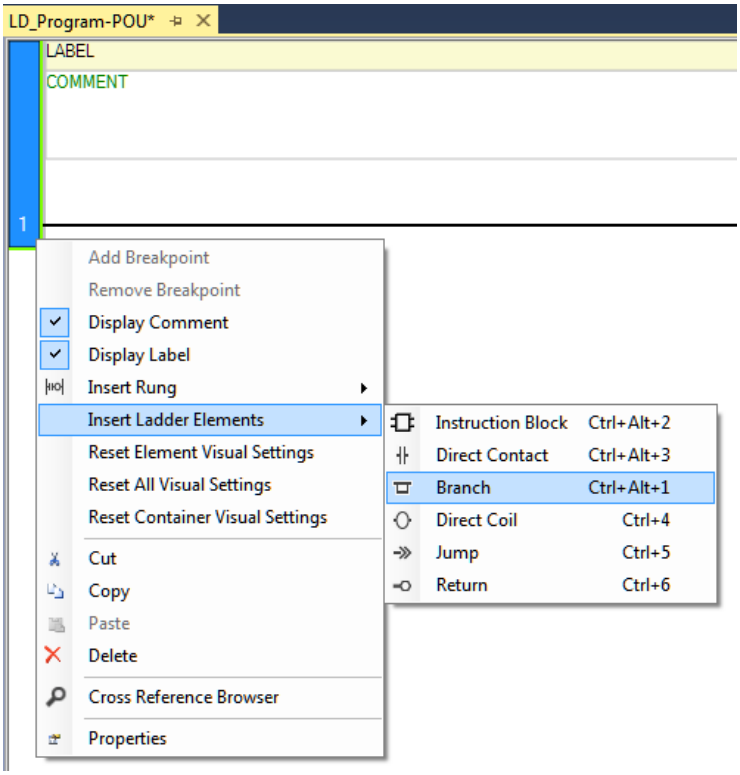
In Connected Components Workbench können Sie an folgenden Stellen eine Verzweigung zu einem Kontaktplan-Programm (KOP – Ladder Diagram, LD) hinzufügen:

- Kontaktplan (KOP)-Sprach-Editor
- Editor für mehrere Sprachen im Menü „Extras“
- KOP-Toolbox
- Anweisungs-Symbolleiste (KOP)

So fügen Sie eine Verzweigung zu einem Kontaktplan-Programm hinzu

1. Stellen Sie im KOP-Sprach-Editor sicher, dass das Kontaktplan-Programm über einen definierten Strompfad für die Verzweigung verfügt. Gehen Sie wahlweise wie folgt vor:
   - Klicken Sie mit der rechten Maustaste auf einen Strompfad oder ein Element im KOP-Sprach-Editor, wählen Sie **Kontaktplanelemente einfügen**, und klicken Sie dann auf **Verzweigung**.

     Wenn die **Variablen-Auswahl** geöffnet wird, wählen Sie eine Variable aus, oder klicken Sie ohne Auswahl einer Variable auf **OK**, um das Element **Verzweigung** hinzuzufügen.



- Wählen Sie einen Strompfad oder ein Element im KOP-Sprach-Editor aus, und drücken Sie dann entweder:

  STRG+ALT+1, um die Verzweigung auf der linken Seite des ausgewählten Elements oder des Strompfads hinzuzufügen.
  STRG+1, um die Verzweigung auf der rechten Seite des ausgewählten Elements hinzuzufügen.

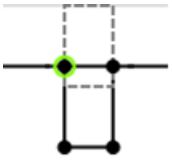
- Wählen Sie einen Strompfad oder ein Element im KOP-Sprach-Editor aus, und dann:

  Klicken Sie auf **Extras** > **Editor für mehrere Sprachen** > **Verzweigung einfügen vor**, um die Verzweigung auf der linken Seite des ausgewählten Elements einzufügen.
  Klicken Sie auf **Extras** > **Editor für mehrere Sprachen** > **Verzweigung einfügen nach**, um die Verzweigung auf der rechten Seite des ausgewählten Elements einzufügen.

- Öffnen Sie die KOP-Toolbox, um das Verzweigungselement anzuzeigen, und führen Sie dann eine der folgenden Aktionen aus:

  Doppelklicken Sie auf das Verzweigungselement, um es zum KOP-Sprach-Editor hinzuzufügen.
  Ziehen Sie das Verzweigungselement in den KOP-Sprach-Editor, und platzieren Sie es auf dem Strompfad.
  Ein Pluszeichen (+) wird im Sprach-Editor für ein gültiges Ablageziel angezeigt. Lassen Sie die Maustaste los, um das Element hinzuzufügen.

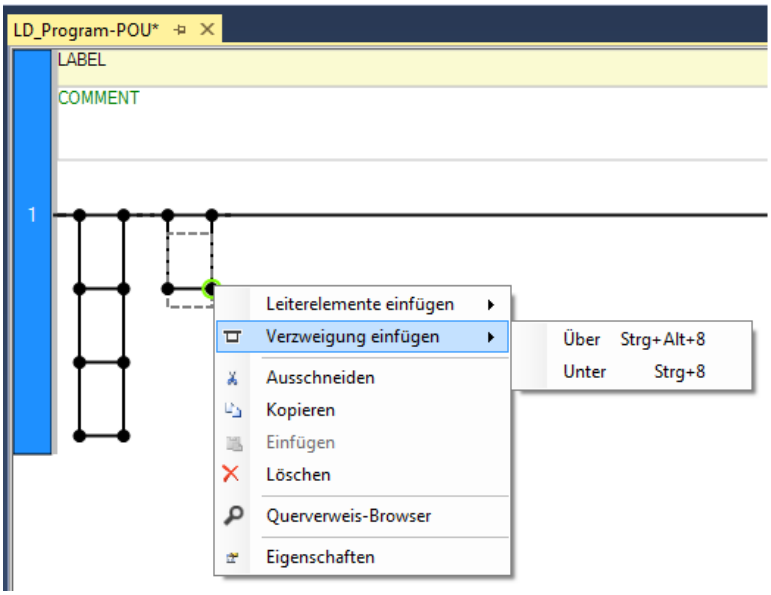
1. (optional) So fügen Sie eine parallele Verzweigung ein:
   a. Klicken Sie im KOP-Sprach-Editor mit der rechten Maustaste auf die Verzweigung, um das Menü des KOP-Sprach-Editors zu öffnen.



   b. Wählen Sie im Menü des KOP-Sprach-Editors **Verzweigung einfügen** aus, und klicken Sie dann entweder auf:

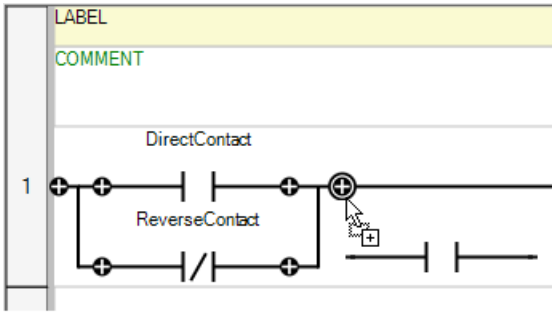
      Über, um die Verzweigung über der ausgewählten Verzweigung hinzuzufügen.

Unter, um die Verzweigung unter der ausgewählten Verzweigung hinzuzufügen.



2. (optional) Um ein Verzweigungselement an eine andere Position im KOP-Programm zu verschieben, wählen Sie das Element aus und ziehen es an die neue Position im KOP-Programm.

   Ein Pluszeichen (+) wird im Sprach-Editor für ein gültiges Ablageziel angezeigt. Lassen Sie die Maustaste los, um das Element an der Zielposition einzufügen.



Anweisungsblock (KOP)

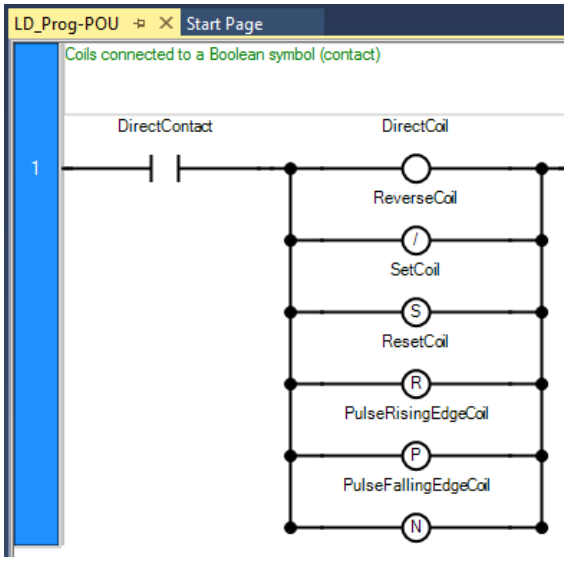
Ein Kontaktplan (KOP)-Anweisungsblock-Element ist ein IEC 61131-3-kompatibles Funktionselement in einem KOP-Programm. Es kann ein Funktionsblock, eine Funktion, ein benutzerdefinierter Baustein, eine benutzerdefinierte Funktion oder ein Operator sein.

Spule

Spulen sind grafische Komponenten von Kontaktplanprogrammen (Ladder Diagram, LD), die die Zuweisung eines Ausgangs oder einer internen Variablen darstellen. In KOP-Programmen stellt eine Spule eine Aktion dar. Die Spule muss links mit einem booleschen Symbol verbunden sein, z. B. ein Kontakt oder die boolesche Ausgabe eines Anweisungsblocks. Spulen können nur zu einem definierten Strompfad im KOP-Sprach-Editor hinzugefügt werden. Die Spulendefinition kann geändert werden, nachdem Sie die Spule zum Strompfad hinzugefügt haben.

Das folgende Beispiel zeigt die für Kontaktplan-Programme verfügbaren Spulenelementtypen.

Beispiel: Spulen



Hinzufügen von Spulenelementen

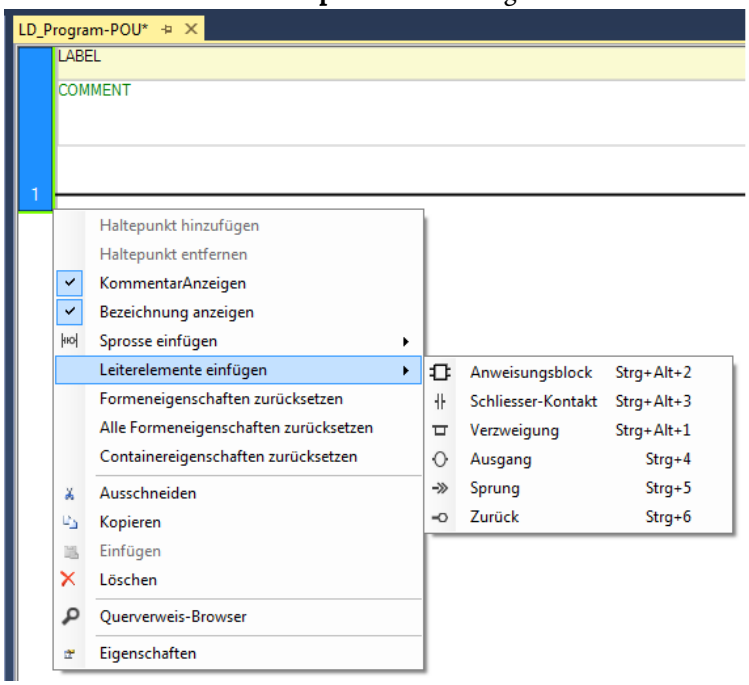
Eine Spule ist eine grafische Komponente eines Kontaktplan (KOP)-Programms, welche eine Ausgabeaktion oder eine interne Variable darstellt.

In Connected Components Workbench können Sie an folgenden Stellen eine Spule zu einem Kontaktplan (Ladder Diagram, LD)-Programm hinzufügen:

- Kontaktplan (KOP)-Sprach-Editor
- Editor für mehrere Sprachen im Menü „Extras“
- KOP-Toolbox
- Anweisungs-Symbolleiste (KOP)

1. So fügen Sie ein Spulenelement zu einem Kontaktplan-Programm hinzu: Vergewissern Sie sich im LD-Sprach-Editor, dass das LD-Programm über einen definierten Strompfad für die Spule verfügt. Gehen Sie wahlweise wie folgt vor:
   - Klicken Sie mit der rechten Maustaste auf einen Strompfad oder ein Element im KOP-Sprach-Editor, wählen Sie **Kontaktplanelemente einfügen**, und klicken Sie dann auf **Direkte Spule**.

Wenn die **Variablen-Auswahl** geöffnet wird, wählen Sie eine Variable aus, oder klicken Sie ohne Auswahl einer Variable auf **OK**, um das Element **Direkte Spule** hinzuzufügen.

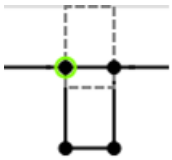


- Wählen Sie einen Strompfad oder ein Element im KOP-Sprach-Editor aus, und drücken Sie dann **STRG+4**, um die **direkte Spule** rechts neben dem Strompfad hinzuzufügen.
- Wählen Sie einen Strompfad oder ein Element im KOP-Sprach-Editor aus, und klicken Sie dann auf **Extras** > **Editor für mehrere Sprachen** > **Spule einfügen**, um die Spule rechts neben dem Strompfad hinzuzufügen.
- Öffnen Sie die KOP-Toolbox, um die Spulenelemente anzuzeigen (**Direkte Spule**, **Umkehrspule**, **Stellspule, Rückstellspule, Spule mit Impulsanstiegflanke**, **Spule mit Impulsabstiegflanke**), und dann:

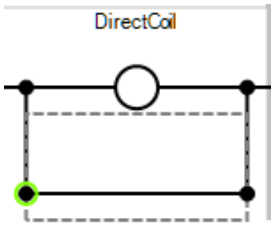
  Doppelklicken Sie auf das Spulenelement, um es zum KOP-Sprach-Editor hinzuzufügen.
  Ziehen Sie das Spulenelement in den KOP-Sprach-Editor, und platzieren Sie es auf dem Strompfad.
  Ein Pluszeichen (+) wird im Sprach-Editor für ein gültiges Ablageziel angezeigt. Lassen Sie die Maustaste los, um das Element hinzuzufügen.

1. (Optional) So fügen Sie eine parallele Spule ein

a. Stellen Sie im KOP-Sprach-Editor sicher, dass das KOP-Programm eine definierte Verzweigung hat, und klicken Sie dann mit der rechten Maustaste auf die oberste Ebene der Verzweigung auf seite 37, um das Menü des KOP-Sprach-Editors zu öffnen.

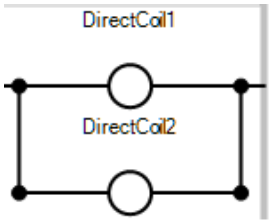


b. Wählen Sie im Menü des KOP-Sprach-Editors **Kontaktplanelemente einfügen** aus, und klicken Sie dann auf **Direkte Spule**. Das Element wird auf der obersten Ebene der Verzweigung eingefügt.



c. Klicken Sie mit der rechten Maustaste in die unterste Ebene der Verzweigung, um das Menü des KOP-Sprach-Editors zu öffnen.

d. Wählen Sie im Menü des KOP-Sprach-Editors **Kontaktplanelemente einfügen** aus, und klicken Sie dann auf **Direkte Spule**. Das Element wird auf der untersten Ebene der Verzweigung eingefügt.



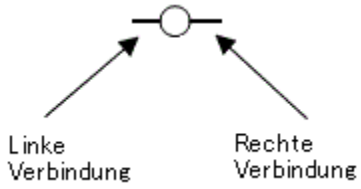
2. (optional) Um den Typ der Spule zu ändern, wählen Sie im KOP-Sprach-Editor die gewünschten Spule aus, und drücken Sie die **Leertaste**, bis im Sprach-Editor der gewünschte Spulentyp angezeigt wird.

   Mit jedem Betätigen der Leertaste ändert sich der Typ jeweils wie folgt: direkt – umgekehrt – Stellspule – Rückstellspule – Impulsanstiegflanke – Impulsabstiegflanke.

Direkte Spule

Spulen auf seite 40 sind grafische Komponenten von Kontaktplanprogrammen (Ladder Diagram, LD), die die Zuweisung eines Ausgangs oder einer internen Variablen darstellen.

Direkte Spulen unterstützen die boolesche Ausgabe einer Anschlussleitung im booleschen Status.



Die zugeordnete Variable wird mit dem booleschen Status der linken Verbindung zugewiesen. Der Status der linken Verbindung wird auf die rechte Verbindung übertragen. Die rechte Verbindung muss mit der rechten vertikalen Stromschiene verbunden sein (ausgenommen bei parallelen Spulen, bei denen nur die obere Spule mit der rechten vertikalen Stromschiene verbunden sein muss).

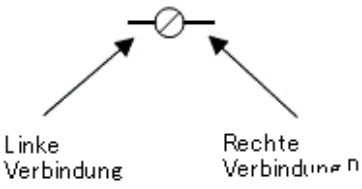
Die zugehörige boolesche Variable muss eine Ausgabe oder benutzerdefiniert sein.

Beispiel für eine direkte Spule

Invertierter Ausgang

Spulen auf seite 40 sind grafische Komponenten von Kontaktplanprogrammen (Ladder Diagram, LD), die die Zuweisung eines Ausgangs oder einer internen Variablen darstellen.

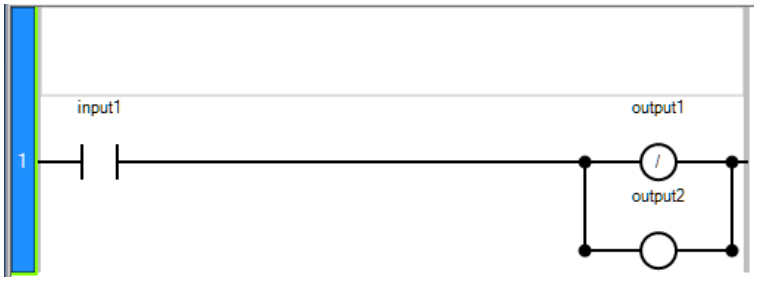
Umkehrspulen ermöglichen die boolesche Ausgabe durch boolesche Negation des Status einer Verbindungslinie.



Die zugeordnete Variable wird mit der booleschen Negation des Status der linken Verbindung zugewiesen. Der Status der linken Verbindung wird auf die rechte Verbindung übertragen. Die rechte Verbindung muss mit der rechten vertikalen Stromschiene verbunden sein (ausgenommen bei parallelen Spulen, bei denen nur die obere Spule mit der rechten vertikalen Stromschiene verbunden sein muss).

Die zugehörige boolesche Variable muss eine Ausgabe oder benutzerdefiniert sein.

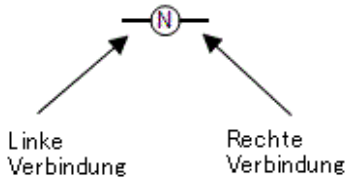
Beispiel für eine Umkehrspule



Spule mit Impulsabstiegflanke

Spulen auf seite 40 sind grafische Komponenten von Kontaktplanprogrammen (Ladder Diagram, LD), die die Zuweisung eines Ausgangs oder einer internen Variablen darstellen.

Impulsspulen mit fallender Flanke auf seite 45 (oder negativer Flanke) unterstützen einen booleschen Ausgang eines booleschen Zustands einer Verbindungsleitung.



Die zugeordnete Variable wird auf WAHR gesetzt, wenn der boolesche Status der linken Verbindung WAHR wird. In allen anderen Fällen wird die Ausgabevariable auf FALSCH zurückgesetzt. Der Status der linken Verbindung wird auf die rechte Verbindung übertragen. Die rechte Verbindung muss mit der rechten vertikalen Stromschiene verbunden sein (ausgenommen bei parallelen Spulen, bei denen nur die obere Spule mit der rechten vertikalen Stromschiene verbunden sein muss).

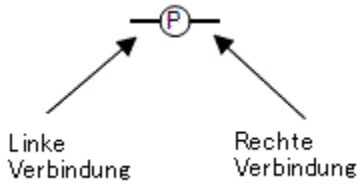
Die zugehörige boolesche Variable muss eine Ausgabe oder benutzerdefiniert sein.

Beispiel für Spule mit Impulsabstiegflanke

Spule mit Impulsanstiegflanke

Spulen auf seite 40 sind grafische Komponenten von Kontaktplanprogrammen (Ladder Diagram, LD), die die Zuweisung eines Ausgangs oder einer internen Variablen darstellen.

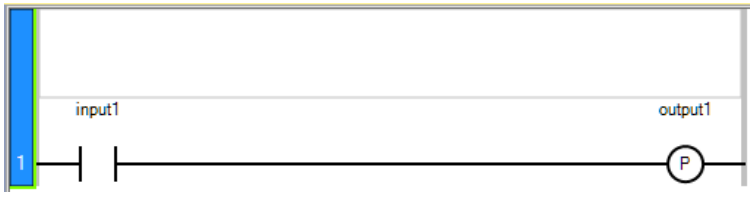
Impulsspulen mit ansteigender Flanke (oder positiv) unterstützen die boolesche Ausgabe einer Anschlussleitung im booleschen Status.



Die zugeordnete Variable wird auf WAHR gesetzt, wenn der boolesche Status der linken Verbindung WAHR wird. In allen anderen Fällen wird die Ausgabevariable auf FALSCH zurückgesetzt. Der Status der linken Verbindung wird auf die rechte Verbindung übertragen. Die rechte Verbindung muss mit der rechten vertikalen Stromschiene verbunden sein (ausgenommen bei parallelen Spulen, bei denen nur die obere Spule mit der rechten vertikalen Stromschiene verbunden sein muss).

Die zugehörige boolesche Variable muss eine Ausgabe oder benutzerdefiniert sein.

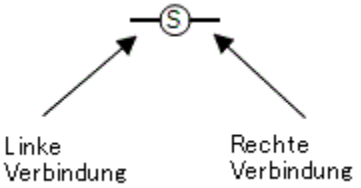
Beispiel für Spule mit Impulsanstiegflanke



Stellspule

Spulen sind grafische Komponenten von Kontaktplanprogrammen (Ladder Diagram, LD), die die Zuweisung eines Ausgangs oder einer internen Variablen darstellen. In KOP-Programmen stellt eine Spule eine Aktion dar.

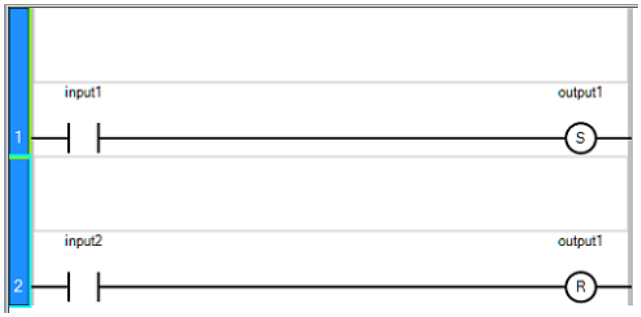
SET-Spulen ermöglichen die boolesche Ausgabe des booleschen Status einer Verbindungslinie.



Die zugeordnete Variable wird auf WAHR gesetzt, wenn der boolesche Status der linken Verbindung WAHR wird. Die Ausgabevariable behält diesen Wert bei, bis von einer Rückstellspule ein gegenteiliger Befehl erfolgt. Der Status der linken Verbindung wird auf die rechte Verbindung übertragen. Die rechte Verbindung muss mit der rechten vertikalen Stromschiene verbunden sein (ausgenommen bei parallelen Spulen, bei denen nur die obere Spule mit der rechten vertikalen Stromschiene verbunden sein muss).

Die zugehörige boolesche Variable muss eine Ausgabe oder benutzerdefiniert sein.

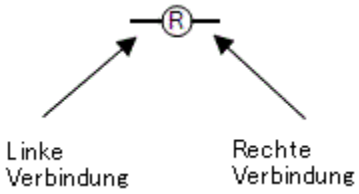
Beispiel für eine Stellspule



Rückstellspule

Spulen auf seite 40 sind grafische Komponenten von Kontaktplanprogrammen (Ladder Diagram, LD), die die Zuweisung eines Ausgangs oder einer internen Variablen darstellen.

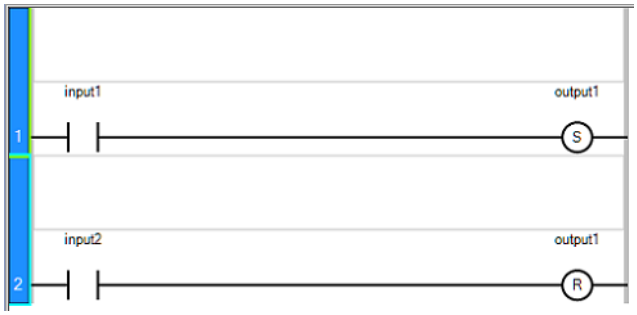
Rückstellspulen ermöglichen die boolesche Ausgabe des booleschen Status einer Verbindungslinie.



Die zugeordnete Variable wird auf FALSCH zurückgesetzt, wenn der boolesche Status der linken Verbindung WAHR wird. Die Ausgabevariable behält diesen Wert bei, bis von einer Stellspule ein gegenteiliger Befehl erfolgt. Der Status der linken Verbindung wird auf die rechte Verbindung übertragen. Die rechte Verbindung muss mit der rechten vertikalen Stromschiene verbunden sein (ausgenommen bei parallelen Spulen, bei denen nur die obere Spule mit der rechten vertikalen Stromschiene verbunden sein muss).

Die zugehörige boolesche Variable muss eine Ausgabe oder benutzerdefiniert sein.

Beispiel für Zurücksetzen einer Spule

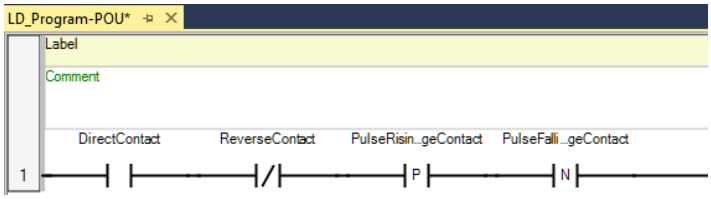


Kontakt

Kontakte sind grafische Komponenten von Kontaktplan-Programmen (Ladder Diagram, LD). Je nach Typ stellt ein Kontakt den Wert oder die Funktion einer Eingabe oder internen Variable dar. Kontakte können nur einem definierten Strompfad im KOP-Sprach-Editor hinzugefügt werden. Nach dem Hinzufügen des Kontakts kann seine Definition geändert werden.

Das folgende Beispiel zeigt die für Kontaktplan-Programme verfügbaren Kontaktelementtypen.

Beispiel: Kontakte



Hinzufügen eines Kontakts zu einem KOP-Programm

Kontakte sind grafische Komponenten eines Kontaktplans (KOP). Je nach Typ stellt ein Kontakt den Wert oder die Funktion einer Eingabe oder internen Variable dar. Kontakte können nur einem definierten Strompfad im KOP-Sprach-Editor hinzugefügt werden.

In Connected Components Workbench können Sie an folgenden Stellen einen Kontakt zu einem Kontaktplan-Programm (KOP – Ladder Diagram, LD) hinzufügen:

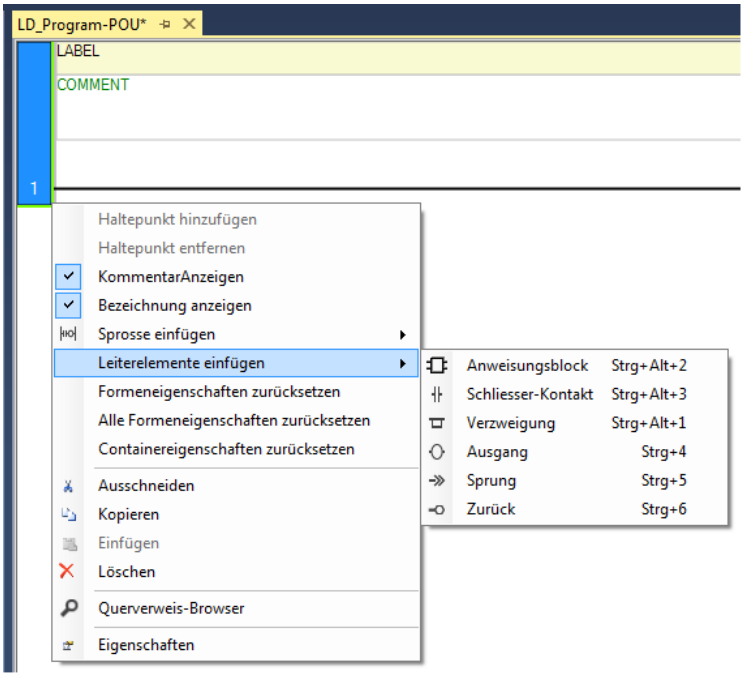
- Kontaktplan (KOP)-Sprach-Editor
- Editor für mehrere Sprachen im Menü „Extras“
- KOP-Toolbox
- Anweisungs-Symbolleiste (KOP)

So fügen Sie ein Kontaktelement zu einem Kontaktplan-Programm hinzu

1. Stellen Sie im KOP-Sprach-Editor sicher, dass das Kontaktplan-Programm über einen definierten Strompfad für den Kontakt verfügt. Gehen Sie wahlweise wie folgt vor:

- Klicken Sie mit der rechten Maustaste auf einen Strompfad oder ein Element im KOP-Sprach-Editor, wählen Sie **Kontaktplanelemente einfügen**, und klicken Sie dann auf **Direkter Kontakt**.

  Wenn die **Variablenauswahl** geöffnet wird, wählen Sie eine Variable aus, oder klicken Sie ohne Auswahl einer Variable auf **OK**, um das Element **Direkter Kontakt** hinzuzufügen.



- Wählen Sie einen Strompfad oder ein Element im KOP-Sprach-Editor aus, und drücken Sie dann entweder:

  STRG+ALT+3, um das direkte Kontaktelement auf der linken Seite des ausgewählten Elements oder des Strompfads hinzuzufügen.
  STRG+3, um das direkte Kontaktelement auf der rechten Seite des ausgewählten Elements oder des Strompfads hinzuzufügen.

- Wählen Sie einen Strompfad oder ein Element im KOP-Sprach-Editor aus, und dann:

  Klicken Sie auf **Extras** > **Editor für mehrere Sprachen** > **Kontakt einfügen vor**, um den Kontakt auf der linken Seite des ausgewählten Elements oder des Strompfads einzufügen.
  Klicken Sie auf **Extras** > **Editor für mehrere Sprachen** > **Kontakt einfügen nach**, um den Kontakt auf der rechten Seite des ausgewählten Elements oder des Strompfads einzufügen.

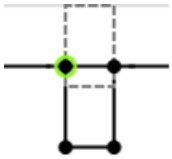
- Öffnen Sie die KOP-Toolbox, um die Kontaktelemente anzuzeigen (**Direkter Kontakt**, **Umkehrkontakt**, **Impulsanstiegflanken-Kontakt**, **Impulsabstiegflanken-Kontakt**), und dann:

  Doppelklicken Sie auf das Kontaktelement, um es zum KOP-Sprach-Editor hinzuzufügen.

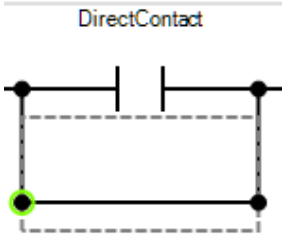
Ziehen Sie das Kontaktelement in den KOP-Sprach-Editor, und platzieren Sie es auf dem Strompfad.
Ein Pluszeichen (+) wird im Sprach-Editor für ein gültiges Ablageziel angezeigt. Lassen Sie die Maustaste los, um das Element hinzuzufügen.

1. (optional) So fügen Sie einen parallelen Kontakt ein:

   a. Stellen Sie im KOP-Sprach-Editor sicher, dass das KOP-Programm eine definierte Verzweigung auf seite 37 hat, und klicken Sie dann mit der rechten Maustaste auf die oberste Ebene der Verzweigung, um das Menü des KOP-Sprach-Editors zu öffnen.

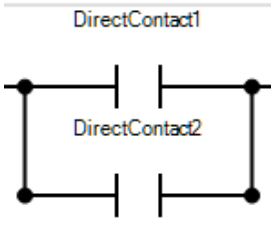


   b. Wählen Sie im Menü des KOP-Sprach-Editors **Kontaktplanelemente einfügen** aus, und klicken Sie dann auf **Direkter Kontakt**. Das Element wird auf der obersten Ebene der Verzweigung eingefügt.



   c. Klicken Sie mit der rechten Maustaste in die unterste Ebene der Verzweigung, um das Menü des KOP-Sprach-Editors zu öffnen.

   d. Wählen Sie im Menü des KOP-Sprach-Editors **Kontaktplanelemente einfügen** aus, und klicken Sie dann auf **Direkter Kontakt**. Das Element wird auf der untersten Ebene der Verzweigung eingefügt.



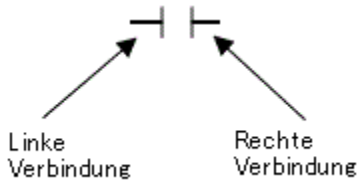
2. (optional) Um den Kontakttyp zu ändern, wählen Sie im Sprach-Editor den gewünschten Kontakt aus, und drücken Sie die **Leertaste**, bis im Sprach-Editor der gewünschte Kontakttyp angezeigt wird.

   Mit jedem Betätigen der Leertaste ändert sich der Kontakttyp jeweils wie folgt: direkt – umgekehrt – Impulsanstiegflanke – Impulsabstiegflanke.

Direkter Kontakt

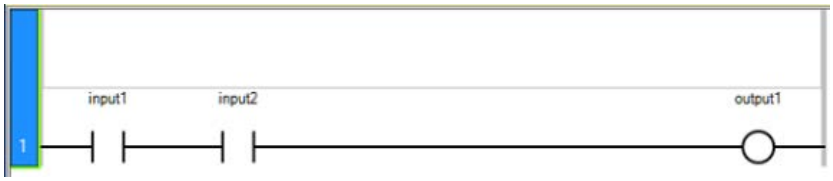
Kontakte auf seite 48 sind grafische Komponenten von Kontaktplan-Programmen (Ladder Diagram, LD).

Direkte Kontakte unterstützen eine boolesche Operation zwischen dem Status einer Anschlussleitung und einer booleschen Variablen.



Der Status der Verbindungslinie rechts vom Kontakt ist das logische AND zwischen dem Status der linken Verbindungslinie und dem Wert der dem Kontakt zugeordneten Variable.

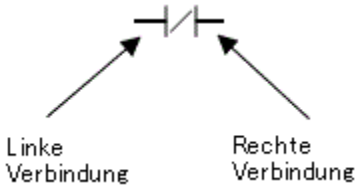
Beispiel für einen direkten Kontakt



Umkehrkontakt

Kontakte auf seite 48 sind grafische Komponenten von Kontaktplan-Programmen (Ladder Diagram, LD).

Umkehrkontakte ermöglichen einen booleschen Vorgang zwischen dem Status einer Verbindungslinie und der booleschen Negation einer booleschen Variablen.



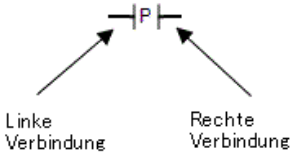
Der Status der Verbindungslinie rechts vom Kontakt ist das logische AND zwischen dem Status der linken Verbindungslinie und der booleschen Negation des Werts der Variable, die dem Kontakt zugeordnet ist.

Beispiel für einen Umkehrkontakt

Impuls-Kontakt bei steigender Flanke

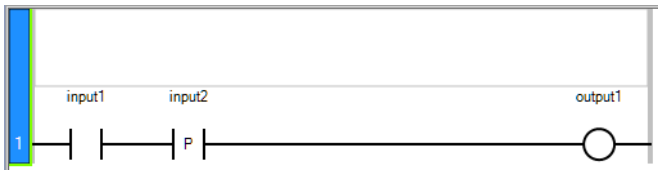
Kontakte auf seite 48 sind grafische Komponenten von Kontaktplan-Programmen (Ladder Diagram, LD).

Impuls-Kontakte bei steigender Flanke (bzw. positive Impuls-Kontakte) ermöglichen eine boolesche Operation zwischen einem Verbindungslinienstatus und der steigenden Flanke einer booleschen Variablen.



Der Status der Verbindungslinie rechts vom Kontakt wird auf WAHR gesetzt, wenn der Status der Verbindungslinie auf der linken Seite WAHR ist und der Status der zugeordneten Variable von FALSCH auf WAHR ansteigt. Der Status wird in allen anderen Fällen auf FALSCH zurückgesetzt.

Beispiel für Impuls-Kontakt bei steigender Flanke



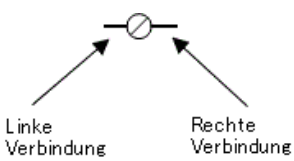
Empfehlung: Beschränken Sie die Verwendung von Ausgangsvariablen mit Flankenkontakten.

Wir empfehlen, keine Ausgänge oder Variablen mit Impulsanstiegflanken-Kontakt auf seite 52 (positiv) oder Impulsabstiegflanken-Kontakt auf seite 52 (negativ) zu verwenden. Diese Kontakte sind für physische Eingaben in einem Kontaktplan vorgesehen. Um die Flanke einer Variablen oder Ausgabe zu erkennen, empfehlen wir die Verwendung eines R_TRIG/F_TRIG-Funktionsblocks, der in jeder Sprache an jeder Stelle im Programm unterstützt wird.

Impuls-Kontakt bei fallender Flanke

Kontakte sind grafische Komponenten von Kontaktplan-Programmen (Ladder Diagram, LD).

Impuls-Kontakte bei fallender Flanke (bzw. negative Impuls-Kontakte) ermöglichen eine boolesche Operation zwischen einem Verbindungslinienstatus und der fallenden Flanke einer booleschen Variablen.



Der Status der Verbindungslinie rechts vom Kontakt wird auf WAHR gesetzt, wenn der Status der Verbindungslinie auf der linken Seite WAHR ist und der Status der zugeordneten Variable von WAHR auf FALSCH absteigt. Der Status wird in allen anderen Fällen auf FALSCH zurückgesetzt.

Beispiel für Impuls-Kontakt bei fallender Flanke

Empfehlung: Beschränken Sie die Verwendung von Ausgangsvariablen mit Flankenkontakten.

Wir empfehlen, keine Ausgänge oder Variablen mit Impulsanstiegflanken-Kontakt auf seite 52 (positiv) oder Impulsabstiegflanken-Kontakt auf seite 52 (negativ) zu verwenden. Diese Kontakte sind für physische Eingaben in einem Kontaktplan vorgesehen. Um die Flanke einer Variablen oder Ausgabe zu erkennen, empfehlen wir die Verwendung eines R_TRIG/F_TRIG-Funktionsblocks, der in jeder Sprache an jeder Stelle im Programm unterstützt wird.

Rücksprung

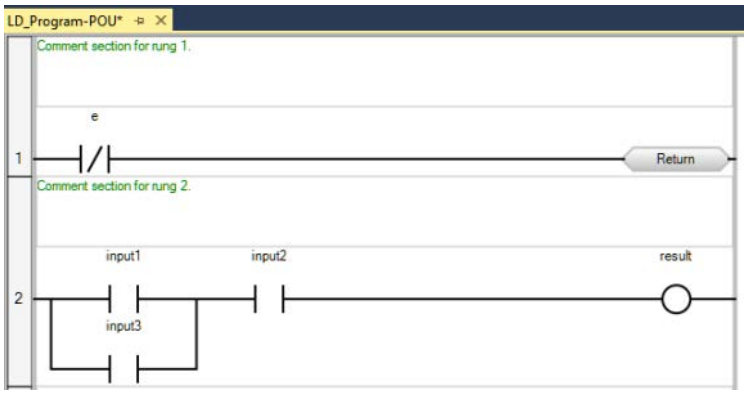
Rücksprünge sind Ausgaben zur Darstellung des bedingten Endes eines Kontaktplan-Programms.

Sie können Verbindungen nicht rechts neben einem Rücksprungelement platzieren.

Wenn der Zustand „Boolesch“ der linken Anschlussleitung den Wert TRUE hat, wird der Plan ohne Ausführung der Anweisungen in den nächsten Zeilen des Plans beendet.

Wenn es sich bei dem Kontaktplan um eine Funktion handelt, wird ihr Name mit einer Ausgabespule verknüpft, um den Rückgabewert festzulegen (Rückgabe an den aufrufenden Plan).

Beispiel eines Rücksprungs



So fügen Sie einen Rücksprung in ein Kontaktplan-Programm ein

Gehen Sie wahlweise wie folgt vor:

- Klicken Sie mit der rechten Maustaste auf einen Strompfad oder ein Element im KOP-Sprach-Editor, wählen Sie **Kontaktplanelemente einfügen**, und klicken Sie dann auf **Rücksprung**.
- Wählen Sie einen Strompfad oder ein Element im KOP-Sprach-Editor aus, und drücken Sie dann **STRG+6**.
- Wählen Sie einen Strompfad oder ein Element im KOP-Sprach-Editor aus, und klicken Sie dann auf **Extras** > **Editor für mehrere Sprachen** > **Rücksprung einfügen**.
- Öffnen Sie die KOP-Toolbox, um das Rücksprungelement anzuzeigen, und führen Sie dann eine der folgenden Aktionen aus:

  Doppelklicken Sie auf das Rücksprungelement, um es zum KOP-Sprach-Editor hinzuzufügen.
  Ziehen Sie das Rücksprungelement in den KOP-Sprach-Editor, und platzieren Sie es auf dem Strompfad.
  Ein Pluszeichen (+) wird im Sprach-Editor für ein gültiges Ablageziel angezeigt. Lassen Sie die Maustaste los, um das Element hinzuzufügen.

Sprung

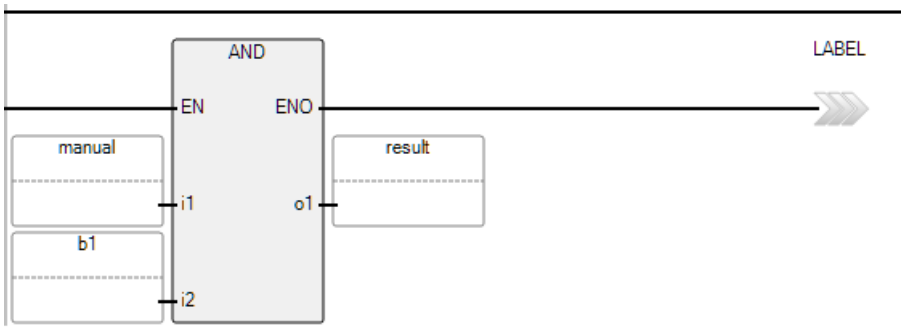
Sprünge sind bedingte oder unbedingte Elemente, welche die Ausführung eines Kontaktplanprogramms (KOP) steuern.

Sprungnotation

Die folgende Bezeichnung gibt einen Sprung zu einer Bezeichnung an:

>>LABEL – Sprung zu einer Bezeichnung mit dem Namen „LABEL“

Beispiel für einen Sprung



So fügen Sie einen Sprung ein

Gehen Sie wahlweise wie folgt vor:

- Klicken Sie mit der rechten Maustaste auf einen Strompfad oder ein Element im KOP-Sprach-Editor, wählen Sie **Kontaktplanelemente einfügen**, und klicken Sie dann auf **Sprung**.
- Wählen Sie einen Strompfad oder ein Element im KOP-Sprach-Editor aus, und drücken Sie dann **STRG+5**.
- Wählen Sie einen Strompfad oder ein Element im KOP-Sprach-Editor aus, und klicken Sie dann auf **Extras** > **Editor für mehrere Sprachen** > **Sprung einfügen**.
- Öffnen Sie die KOP-Toolbox, um das Sprungelement anzuzeigen, und führen Sie dann eine der folgenden Aktionen aus:

  Doppelklicken Sie auf das Sprungelement, um es zum KOP-Sprach-Editor hinzuzufügen.
  Ziehen Sie ein Kontaktelement in den KOP-Sprach-Editor, und platzieren Sie es auf dem Strompfad.
  Ein Pluszeichen (+) wird im Sprach-Editor für ein gültiges Ablageziel angezeigt. Lassen Sie die Maustaste los, um das Element hinzuzufügen.

Anweisungsblöcke in KOP-Programmen

Der Satz von Connected Components Workbench-Anweisungen umfasst IEC 61131-3-kompatible Anweisungsblöcke. Anweisungsblöcke wiederum bestehen aus Funktionsblöcken, Funktionen und Operatoren. Sie können die Eingaben und Ausgaben von Anweisungsblöcken mit Variablen, Kontakten, Spulen oder anderen Anweisungsblockeingaben und -ausgaben verbinden.

Benennungskonventionen für Anweisungsblöcke

Die IEC61131-3-Programmiersprachenspezifikation behandelt zahlreiche Aspekte programmierbarer Controller, wie z. B. die Ausführung des Betriebssystems, Datendefinitionen, Programmiersprachen und den Anweisungssatz. Daher bietet die IEC61131-3-Spezifikation ein Mindestmaß an Funktionalität, das im Hinblick auf die Anforderungen der Endbenutzerprogramme erweitert werden kann.

Anweisungsblocknamen

Funktionen und Funktionsblöcke werden durch ein Feld dargestellt, das den Namen der Anweisung und die Kurznamen der Parameter anzeigt. Für Funktionsblöcke wird der Name der Instanz über dem Funktionsblocknamen angezeigt.

Anweisungsblock-Rückgabeparameter

- Der Rückgabeparameter einer Funktion hat denselben Namen wie die Funktion. Der Rückgabeparameter ist die einzige Ausgabe.

- Die Rückgabeparameter eines Funktionsblocks können beliebige Namen haben. Mehrere Rückgabeparameter können mehrere Ausgaben bereitstellen.
- Sie können die Parameter eines Programms für mehrere Geräte definieren, indem Sie durch die Registerkarten der einzelnen Geräte, die in der **Parameteransicht** angezeigt werden, navigieren.

Arbeiten im KOP-Sprach-Editor

Wenn Sie Elemente zu einem Strompfad im Kontaktplan-Programm (KOP – Ladder Diagram, LD) hinzufügen, geschieht dies nach folgenden Kriterien:

- Das erste Element auf einem Strompfad wird an der im Kontaktplan ausgewählten Position eingefügt.
- Nachfolgende Elemente werden rechts neben dem ausgewählten Element in der Stufe eingefügt.
- Sie können keine Elemente rechts neben einem Spulenrücksprung oder -sprung einfügen.

Verschiedene Methoden, um ein Element zum Kontaktplan-Programm hinzuzufügen:

- **Kontaktplan-Editor**
  - Hinzufügen von Elementen, Löschen von Elementen, Kopieren und Einfügen von Elementen.
  - Mithilfe von Kontaktplan-Tastenkombinationen auf seite 58 können Sie Elemente hinzufügen.
- **KOP-Texteditor**
  - Fügen Sie Elemente hinzu und bearbeiten und löschen Sie sie.
  - Kopieren Sie Elemente von/in RSLogix 500 und <RSLX5000>.
- **Editor für mehrere Sprachen** im Menü **„Extras“**
  - Fügen Sie Elemente hinzu.
  - Exportieren Sie ein Bild des KOP-Programms.
  - Aktivieren oder deaktivieren Sie das automatische Öffnen der **Variablenauswahl** und des **Anweisungsblockselektors**.
- **KOP-Toolbox**
  - Fügen Sie Elemente hinzu.
- **Anweisungs-Symbolleiste**
  - Fügen Sie Elemente hinzu.

Eine zugewiesene Variable können Sie direkt im Sprach-Editor oder in der **Variablenauswahl** ersetzen.

So ändern Sie eine Variable im Sprach-Editor

1. Klicken Sie im Sprach-Editor auf den Variablennamen, um eine Dropdown-Liste mit globalen und lokalen Variablen anzuzeigen.

2. Gehen Sie wahlweise wie folgt vor:
   - Geben Sie einen neuen Variablennamen ein und doppelklicken Sie auf die Variable, um die **Variablen-Auswahl** zu öffnen. Drücken Sie die **Eingabetaste**, um die neue Variable zu bestätigen.
   - Wählen Sie in der Dropdown-Liste einen anderen Variablennamen aus.

So ändern Sie eine Variable in der Variablen-Auswahl

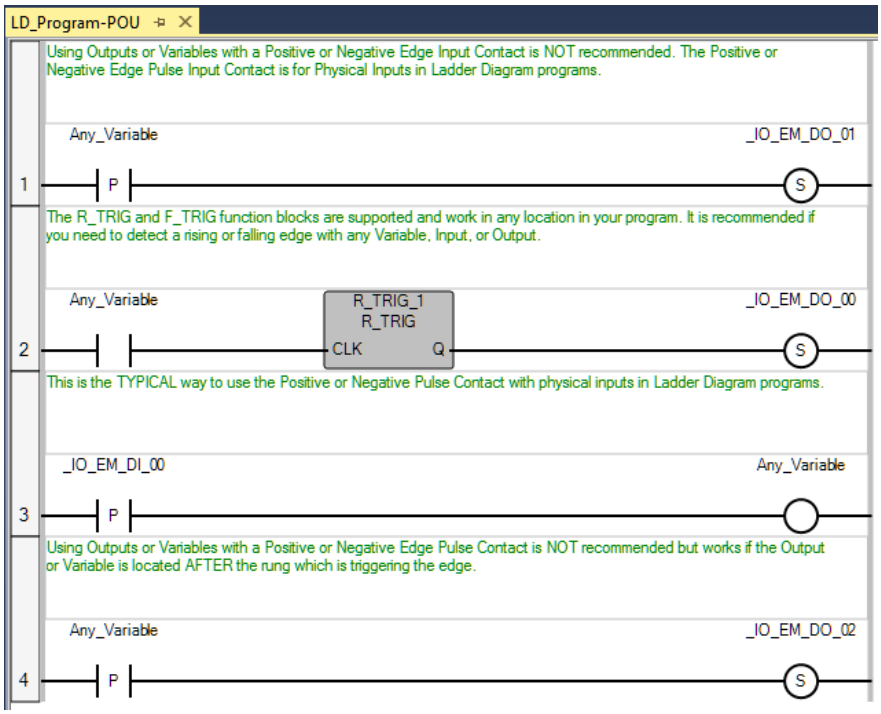
1. Doppelklicken Sie im Sprach-Editor auf die Variable, um den **Variablenselektor** zu öffnen.
2. Klicken Sie auf den Variablennamen, und wählen Sie aus der Dropdown-Liste mit globalen und lokalen Variablen eine andere Variable aus.
3. Klicken Sie auf eine vorhandene Variable, und geben Sie dann die konstanten Werte in das vorgesehene Textfeld ein.

Beispiele zum Kontaktplan (KOP)

Folgende Beispiele sind Kontaktplanprogramme (KOP).

Beispiel: R_TRIG, Funktionsblock

Das folgende Beispielprogramm zeigt die empfohlene Verwendung eines R_TRIG-Funktionsblocks zum Erkennen einer Flanke während der Verbindung zum Controller.

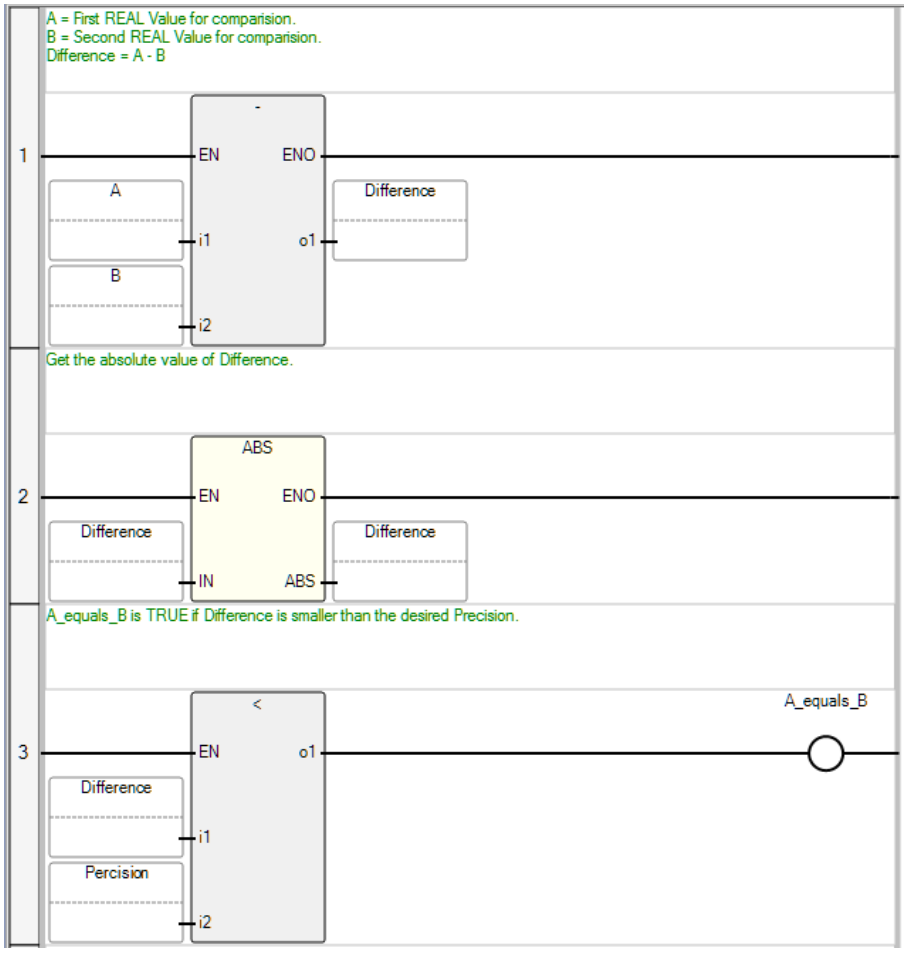


Beispiel: Vergleichen von Real-Werten mittels Subtraktion (-) ABS und „Kleiner als“ (<)

Der Datentyp „Real“ sollte zum Vergleichen von Werten auf Gleichheit nicht verwendet werden, da es Unterschiede bei der Auf- und Abrundung von Zahlen gibt. Zwei Ausgabewerte mögen in einer Connected Components Workbench-Anzeige gleich aussehen, werden aber mit „false“ ausgewertet.

Beispiel: Die Werte 23,500001 und 23,499999 werden in der Ansicht der Variablenausgabe beide mit 23,5 angezeigt, sind jedoch im Controller nicht gleich.

Sie prüfen, ob zwei Real-Datentypen gleich sind, indem Sie den Unterschied zwischen den Werten mit einer Subtraktionsanweisung ermitteln und anschließend feststellen, ob der Unterschied kleiner ist als ein festgesetzter Genauigkeitswert. Schauen Sie sich das folgende Beispiel eines LD-Programms an, in dem zwei Werte vom Datentyp „Real“ verglichen werden.



KOP-Tastenkombinationen

Folgende Tastenkombinationen stehen bei Verwendung der Kontaktplan-Sprache auf seite 29 zur Verfügung.

Tastenkombination	Beschreibung
Strg+0	Fügt eine Stufe nach einer ausgewählten Stufe ein.[1]

Tastenkombination	Beschreibung
Strg+Alt+0	Fügt eine Stufe vor einer ausgewählten Stufe ein.[1]
Strg+ 1	Fügt eine Verzweigung nach einem ausgewählten Element ein.
Strg+Alt+1	Fügt eine Verzweigung vor einem ausgewählten Element ein.
Strg+2	Fügt einen Anweisungsblock nach einem ausgewählten Element ein.[2]
Strg+Alt+2	Fügt einen Anweisungsblock vor einem ausgewählten Element ein.[2]
Strg+3	Fügt einen Kontakt nach einem ausgewählten Element ein.[2]
Strg+Alt+3	Fügt einen Kontakt vor einem ausgewählten Element ein.[2]
Strg+4	Fügt eine Spule nach einem ausgewählten Element ein.
Strg+5	Fügt einen Sprung nach einem ausgewählten Element ein.
Strg+Alt+5	Fügt einen Sprung nach einem ausgewählten Element ein.
Strg+6	Fügt einen Rücksprung nach einem ausgewählten Element ein.
Strg+8	Fügt eine Verzweigung über der ausgewählten Verzweigung ein.
Strg+Alt+8	Fügt eine Verzweigung unter der ausgewählten Verzweigung ein.
Löschen	Entfernt eine ausgewählte Stufe oder ein ausgewähltes Element.
Eingabetaste	Wenn ein Strompfad ausgewählt ist, wird durch Drücken der Eingabetaste das erste Element des Strompfad ausgewählt. Wenn kein Strompfadelement vorhanden ist, geschieht nichts.
Leertaste	Wenn eine Spule oder einen Kontakt ausgewählt ist, wird durch Drücken der Leertaste der Kontakt oder der Spulentyp geändert.
Umschalttaste+Eingabetaste	Fügt einen Zeilenumbruch ein.
Strg+Eingabetaste	Öffnet eine Zeile über der aktuellen Zeile.
Strg+Umschalttaste+Eingabetaste	Öffnet eine Zeile unter der aktuellen Zeile.
Strg+Umschalttaste+L	Entfernt die aktuelle Zeile.
Strg+Entf-Taste	Entfernt das nächste Wort in der aktuellen Zeile.
Rücktaste	Entfernt das Zeichen ganz links.
Strg+Rücktaste	Entfernt das vorherige Wort in der aktuellen Zeile.
Strg+C	Kopiert den ausgewählten Text in die Zwischenablage.
Strg+Einfg-Taste	Kopiert den ausgewählten Text in die Zwischenablage.
Strg+V	Fügt Text aus der Zwischenablage an der Einfügemarke ein.
Umschalttaste+Einfg-Taste	Fügt Text aus der Zwischenablage an der Einfügemarke ein.
Strg+Z	Macht den vorherigen Befehl rückgängig.
Strg+Y	Wiederholt den vorherigen Befehl.
Strg+Umschalttaste+Z	Wiederholt den vorherigen Befehl.
Strg+Links	Wechselt zur vorherigen Anweisung bzw. zum vorherigen Wort.
Strg+Rechts	Wechselt zur nächsten Anweisung bzw. zum nächsten Wort.
Startposition	Wechselt zum ersten Element des ausgewählten Strompfads. Wenn kein Strompfadelement vorhanden ist, geschieht nichts.
Ende	Wechselt zum letzten Element des ausgewählten Strompfads. Wenn kein Strompfadelement vorhanden ist, geschieht nichts.
Strg+Pos 1	Wechselt zum ersten Element des ersten Strompfads. Wenn kein Strompfadelement vorhanden ist, wird der erste Strompfad ausgewählt.
Strg+Ende	Wechselt zum letzten Element des letzten Strompfads. Wenn kein Strompfadelement vorhanden ist, wird der letzte Strompfad ausgewählt.
Bild auf	Wechselt zum Anfang des sichtbaren Codes.
Bild ab	Wechselt zum Ende des sichtbaren Codes.

Tastenkombination	Beschreibung
Strg+J	Wechselt zur übereinstimmenden Klammer.
Strg+Unten	Blättert nach unten.
Strg+Oben	Blättert nach oben.
Umschalttaste+Unten	Auswahl nach unten.
Umschalttaste+Mausklick links	Wählt mehrere Strompfade aus. Klicken Sie einzeln auf jede Stufe.
Umschalttaste+Oben	Auswahl nach oben.
Umschalttaste+Links	Auswahl nach links.
Umschalttaste+Rechts	Auswahl nach rechts.
Strg+Umschalttaste+Links	Wählt die vorherige Anweisung bzw. das vorherige Wort aus.
Strg+Umschalttaste+Rechts	Wählt die nächste Anweisung bzw. das nächste Wort aus.
Umschalttaste+Pos 1	Auswahl ab der Einfügemarke bis zum Zeilenanfang.
Umschalttaste+Ende	Auswahl ab der Einfügemarke bis zum Zeilenende.
Strg+Umschalttaste+Pos 1	Auswahl ab der Einfügemarke bis zum Dokumentanfang.
Strg+Umschalttaste+Ende	Auswahl ab der Einfügemarke bis zum Dokumentende.
Umschalttaste+Bild auf	Auswahl ab der Einfügemarke bis zum Anfang des sichtbaren Codes.
Umschalttaste+Bild ab	Auswahl ab der Einfügemarke bis zum Ende des sichtbaren Codes.
Strg+Umschalttaste+Bild auf	Auswahl ab der Einfügemarke bis zum Anfang des sichtbaren Codes.
Strg+Umschalttaste+Bild ab	Auswahl ab der Einfügemarke bis zum Ende des sichtbaren Codes.
Strg+A	Wählt das gesamte Dokument aus.
STRG+D	Wenn ein Strompfad oder ein Element des Strompfads ausgewählt, kann der Benutzer durch Drücken von STRG+D den Kommentar bearbeiten.
Strg+R	Aktivieren oder Deaktivieren des automatischen Selektoraufrufs. Standardmäßig wird der Anweisungsblockselektor oder das Variablenauswahl-Dialogfeld geöffnet, wenn ein Element zu einem Kontaktplan-Programm hinzugefügt wird.
Strg+Umschalttaste+W	Wählt das nächste Wort aus.
Strg+Umschalttaste+J	Auswahl zur übereinstimmenden Klammer.
Umschalttaste+Alt+Unten	Wählt die aktuelle und die nächste Zeile aus.
Umschalttaste+Alt+Oben	Wählt die aktuelle und die vorherige Zeile aus.
Umschalttaste+Alt+Links	Auswahl links der aktuellen Zeile.
Umschalttaste+Alt+Rechts	Auswahl rechts der aktuellen Zeile.
Strg+Umschalttaste+Alt+Links	Wählt die verfügbaren Spalten in Codezeilen von links nach rechts aus.
Strg+Umschalttaste+Alt+Rechts	Wählt die verfügbaren Spalten in Codezeilen von rechts nach links aus.
ESC	Hebt die Textauswahl auf.
Einfg-Taste	Wechselt zwischen dem Überschreibungs- und dem Einfügemodus.

[1] Wenn kein Strompfad ausgewählt ist, wird am Ende der Strompfadliste einer hinzugefügt.

[2] Wenn eine Verzweigung ausgewählt ist, wird am Ende der Verzweigung ein Element eingefügt.

Alarmanweisung

Verwenden Sie die Alarmanweisung, um Alarme auszugeben, sobald eine konfigurierte Ober- oder Untergrenze erreicht ist.

Funktionsblock	Beschreibung
LIM_ALRM auf seite 61	Hysterese bei einem „Real"-Wert für Ober- und Untergrenzen.

LIM_ALRM (Alarmgrenze)

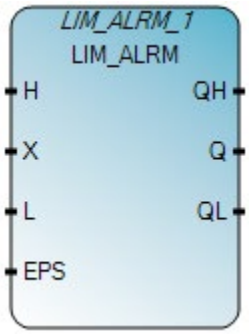
Dieser Alarm blockiert Hysterese bei einem „Real"-Wert für Ober- und Untergrenzen.

Eine Hysterese wird auf Ober- und Untergrenzen angewendet. Die Hysterese-Differenz für die Ober- oder Untergrenze beträgt die Hälfte des EPS-Parameters.

Ein Prozessalarm wird ausgelöst, wenn vom Controller ein Fehler empfangen und verarbeitet wird. Durch Prozessebenenalarme werden Sie informiert, wenn das Modul die konfigurierte Ober- oder Untergrenze eines Kanals überschritten hat.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

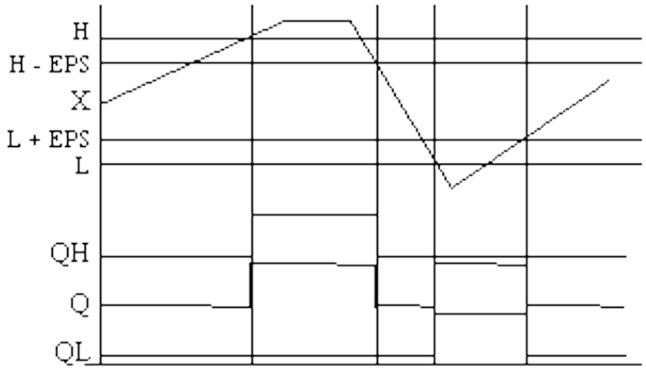


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

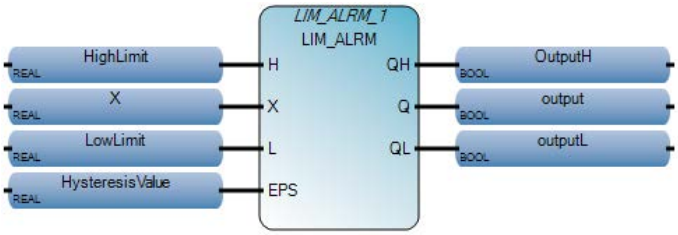
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Wenn TRUE, ist der Anweisungsblock aktiviert. TRUE – aktuelle LIM_ALRM-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
H	Eingang	REAL	Wert der Obergrenze.

Parameter	Parametertyp	Datentyp	Beschreibung
X	Eingang	REAL	Eingabe ist jeglicher Wert vom Typ „Real“.
L	Eingang	REAL	Wert der Untergrenze.
EPS	Eingang	REAL	Hysterese-Wert (muss über Null liegen).
QH	Ausgang	BOOL	Hoher Alarm: TRUE wenn X über Obergrenze H.
Q	Ausgang	BOOL	Alarmausgabe: TRUE wenn X außerhalb des zulässigen Bereichs.
QL	Ausgang	BOOL	Niedriger Alarm: TRUE wenn X unter Untergrenze L.
ENO	Ausgang	BOOL	Aktiviert die Ausgabe. Gilt nur für Kontaktplan-Programme.

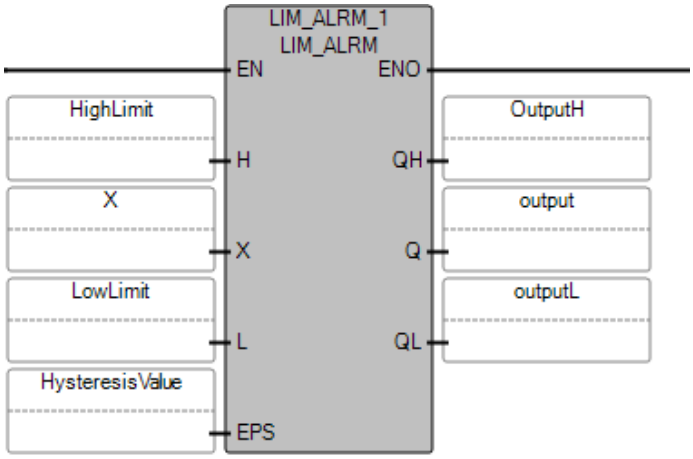
Beispiel für LIM_ALRM-Timing-Diagramm



Diagrammbeispiel für Funktionsblock LIM_ALRM



LIM_ALRM-Kontaktplan-Beispiel



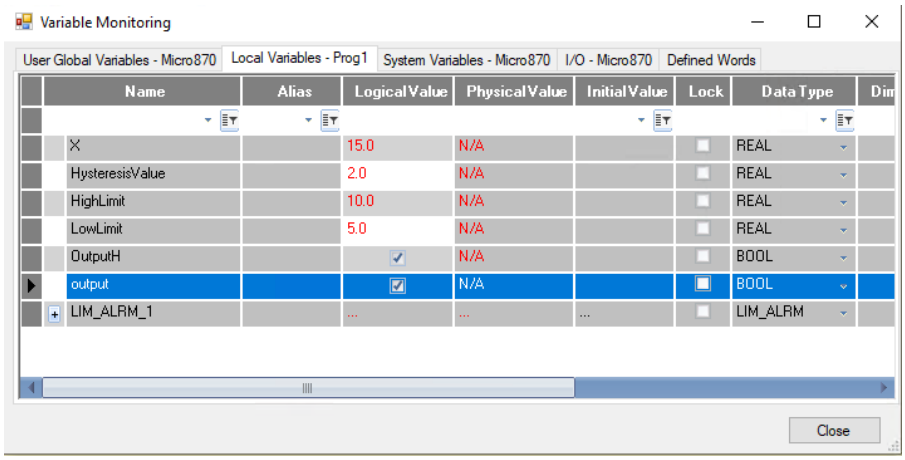
Beispiel für LIM_ALRM in strukturiertem Text

```
LIM_ALRM_1(
```

void **LIM_ALRM_1**(REAL **H**, REAL **X**, REAL **L**, REAL **EPS**)
Typ: LIM_ALRM, Alarm bei hohem/niedrigem Grenzwert mit Hysterese

```
1  HighLimit := 10.0;
2  X := 15.0;
3  LowLimit := 5.0;
4  HysteresisValue := 2.0;
5  LIM_ALRM_1(HighLimit, X, LowLimit, HysteresisValue);
6  OutputH := LIM_ALRM_1.QH;
7  OutputL := LIM_ALRM_1.QL;
8  output := LIM_ALRM_1.Q;
```

Ergebnisse



Arithmetische Anweisungen

Verwenden Sie die arithmetischen Anweisungen, um mathematische Berechnungen durchführen.

Funktion	Beschreibung
ABS auf seite 65	Liefert den absoluten Wert einer Realzahl.
ACOS auf seite 67	Berechnet den Arkuscosinus einer Realzahl.
ACOS_LREAL auf seite 68	Berechnet den Arkuscosinus einer langen Realzahl.
Addition auf seite 69	Addiert zwei oder mehr Werte vom Typ „Integer“, „Real“, „Time“ oder „String“.
ASIN auf seite 71	Berechnet den Arkussinus eines langen Real-Wertes.
ASIN_LREAL auf seite 72	Berechnet den Arkussinus einer langen Realzahl.
ATAN auf seite 74	Berechnet den Arkustangens einer Realzahl.
ATAN_LREAL auf seite 75	Berechnet den Arkustangens einer langen Realzahl.
COS auf seite 76	Berechnet den Cosinus einer Realzahl.
COS_LREAL auf seite 78	Berechnet den Cosinus einer langen Realzahl.
Division auf seite 79	Division zweier Werte der Typen „Integer“ und „Real“.
EXPT auf seite 81	Berechnet den Wert des Typs „Real“ einer Basiszahl potenziert mit dem Exponenten „Integer“.
LOG auf seite 82	Berechnet den Logarithmus (Basis 10) einer Realzahl.
MOD auf seite 84	Führt eine Modulberechnung für ganzzahlige Werte durch.
MOV auf seite 85	Kopiert einen Eingabewert in eine Ausgabe.
Multiplikation auf seite 86	Multipliziert zwei oder mehr Werte vom Typ „Integer“ oder „Real“.
Neg auf seite 88	Konvertiert einen Wert in einen negativen Wert.
POW auf seite 89	Berechnet den Wert einer Realzahl potenziert mit dem Exponenten „Real“.
RAND auf seite 90	RAND liefert nach dem Zufallsprinzip ausgewählte ganzzahlige Werte aus einem definierten Bereich.
SIN auf seite 92	Berechnet den Sinus von einer Realzahl.
SIN_LREAL auf seite 94	Berechnet den Sinus einer langen Realzahl.
SORT auf seite 95	Berechnet die Quadratwurzel einer reellen Zahl.
Subtraktion auf seite 97	Subtrahiert einen Wert vom Typ „Integer“, „Real“ oder „Time“ von einem anderen Wert vom Typ „Integer“, „Real“ oder „Time“.
TAN auf seite 98	Berechnet den Tangens einer Realzahl.
TAN_LREAL auf seite 100	Berechnet den Tangens einer langen Realzahl.
TRUNC auf seite 101	Schneidet Realzahlen ab und belässt nur eine Ganzzahl.

ABS (Absolutbetrag)

Gibt den (positiven) Absolutbetrag einer Realzahl zurück.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

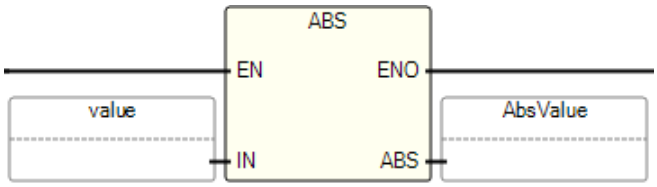
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

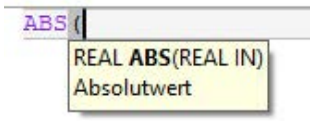
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführen der Berechnung des aktuellen Absolutwerts. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	REAL	Beliebiger realer Wert mit Vorzeichen.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.
ABS	Ausgang	REAL	Absoluter Wert (immer positiv).

Beispiel für ein ABS-Funktionsblockdiagramm

Beispiel für einen ABS Kontaktplan



Beispiel für strukturierten ABS-Text

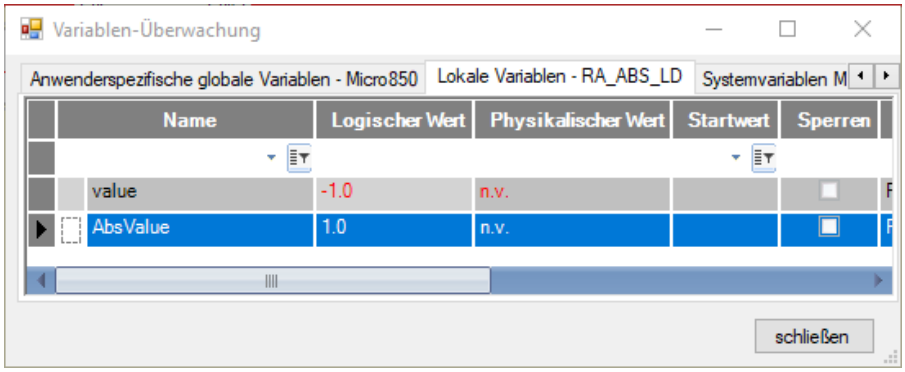



```
1  value := -1.0;
2  AbsValue := ABS(value);
```

(* ST-Äquivalenz: *)

over := (ABS (delta) > range);

Ergebnisse



ACOS (Arkuscosinus der Quelle)

Berechnet den Arkuscosinus einer Realzahl. Eingangs- und Ausgangswerte werden im Bogenmaß angegeben.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

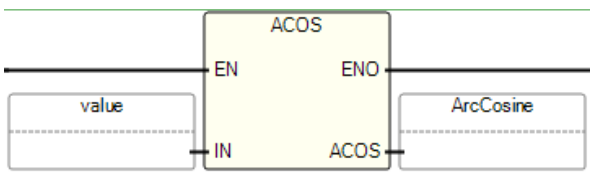
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführen der Berechnung des aktuellen Arkuscosinus. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	REAL	muss im Satz enthalten sein [-1.0 .. +1,0].
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.
ACOS	Ausgang	REAL	Arkuscosinus des Eingabewerts (in Satz [-p1/2..+p1/2])=0 für ungültige Eingabe.

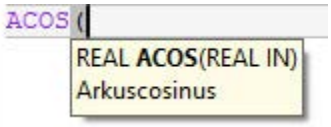
Beispiel für ein ACOS-Funktionsblockdiagramm

Beispiel für einen ACOS-Kontaktplan



Beispiel für strukturierten ACOS-Text

```
1  value := 0.5;
2  ArcCosine := ACOS(value);
```

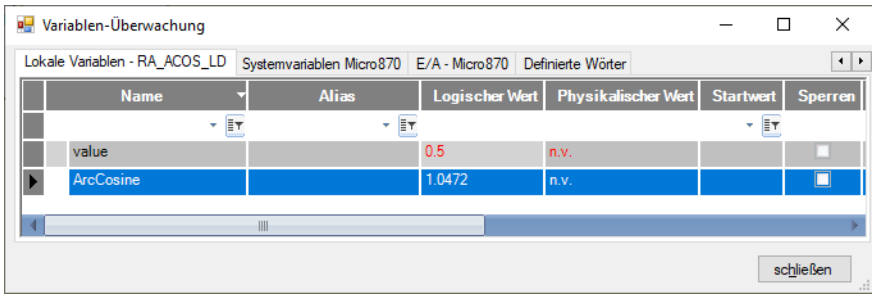


(* ST-Äquivalenz: *)

cosine := COS (angle);

result := ACOS (cosine); (* Ergebnis ist gleich Winkel *)

Ergebnisse

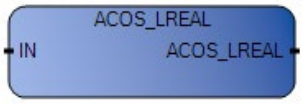


ACOS_LREAL (Arkuscosinus-Typ „Long Real“)

Berechnet den Arkuscosinus einer langen Realzahl.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

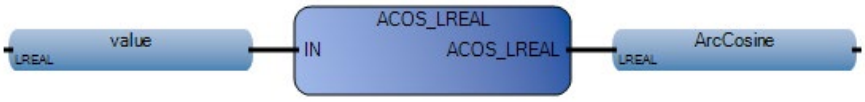
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



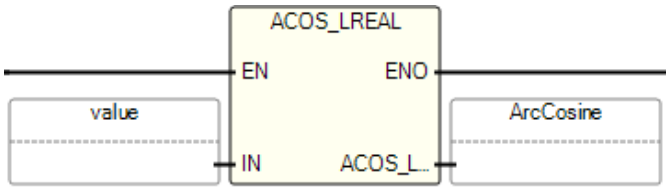
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Ausführen der aktuellen Berechnung. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	LREAL	muss im Satz enthalten sein [-1.0 .. +1,0].
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.
ACOS_LREAL	Ausgang	LREAL	Arkuscosinus des Eingangswertes (im Bereich [0,0 .. PI]) = 0,0 für ungültigen Eingang.

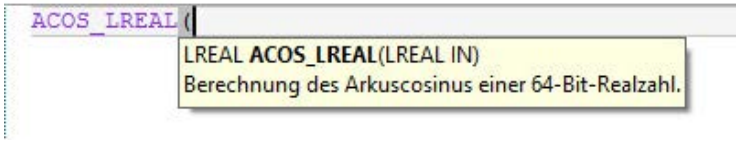
Beispiel für ein ACOS_LREAL-Funktionsblockdiagramm



Beispiel für einen ACOS_LREAL-Kontaktplan



Beispiel für strukturierten ACOS_LREAL-Text



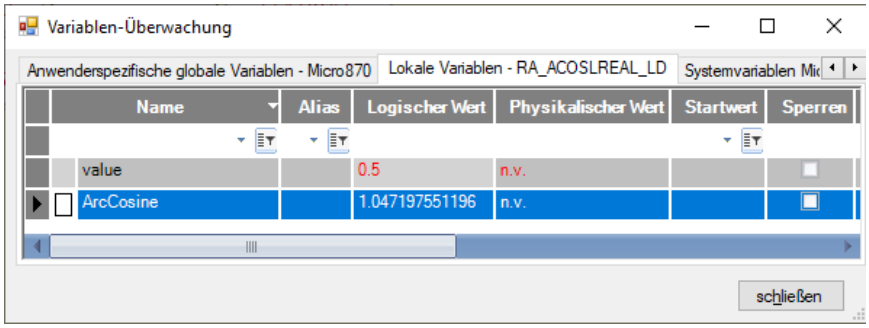

```
1  value := 0.5;
2  ArcCosine := ACOS_LREAL(value);
```

(* ST-Äquivalenz: *)

cosine := COS_LREAL (angle);

result := ACOS_LREAL (cosine); (* result is equal to angle *)

Ergebnisse



Addition

Addiert zwei oder mehr Werte vom Typ „Integer“, „Real“, „Time“ oder „String“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführen der aktuellen Addition. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME STRING	Summand in Datentyp „Real“, „Time“ oder „String“. Alle Eingaben müssen denselben Datentyp haben.
i2	Eingang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME STRING	Summand in Datentyp „Real“, „Time“ oder „String“. Alle Eingaben müssen denselben Datentyp haben.
o1	Ausgang	SINT USINT BYTE INT	Summe der Eingabewerte im Format „Real“, „Time“ oder „String“. Eingabe und Ausgabe müssen denselben Datentyp haben.

Parameter	Parametertyp	Datentyp	Beschreibung
		UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME STRING	
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten Additions-Text

(* ST-Äquivalenz: *)

```
ao10 := ai101 + ai102;
ao5 := (ai51 + ai52) + ai53;
```

ASIN (Arkussinus)

Berechnet den Arkussinus eines langen Real-Wertes. Eingangs- und Ausgangswerte werden im Bogenmaß angegeben.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

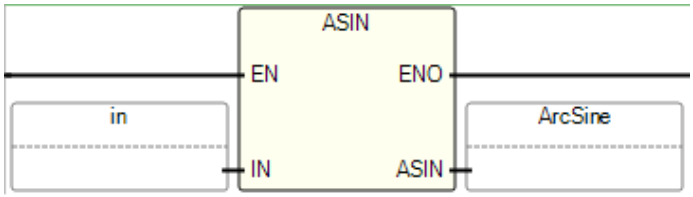
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Berechnung des aktuellen Arkussinus wird ausgeführt. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	REAL	muss im Satz enthalten sein [-1.0 .. +1,0].
ASIN	Ausgang	REAL	Arkussinus des Eingabewerts (in Satz [-p1/2..+p1/2])=0 für ungültigen Eingangswert.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

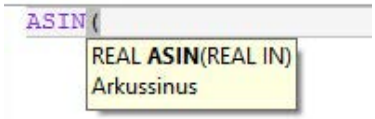
Beispiel für ein ASIN-Funktionsblockdiagramm

Beispiel für einen ASIN-Kontaktplan



Beispiel für strukturierten ASIN-Text

```
1  in := 0.5;
2  ArcSine := ASIN(in);
```

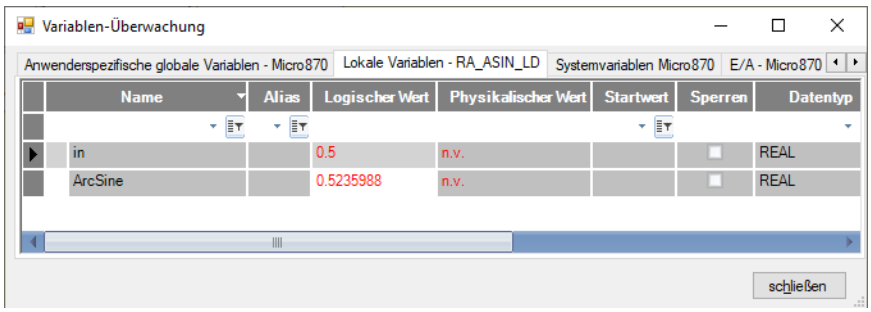


(* ST-Äquivalenz: *)

sine := SIN (angle);

result := ASIN (sine); (* Ergebnis ist gleich Winkel *)

Ergebnisse

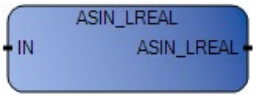


ASIN_LREAL (Arkussinus-Typ „Long Real“)

Berechnet den Arkussinus einer langen Realzahl.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

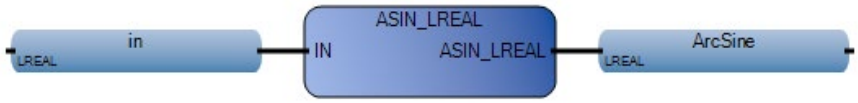
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



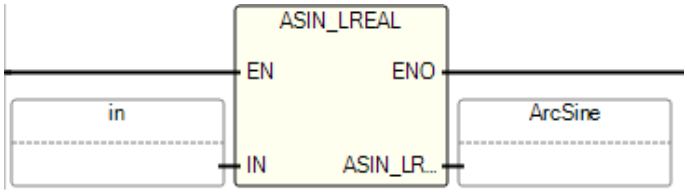
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Ausführen der aktuellen Berechnung. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	LREAL	muss im Satz enthalten sein [-1.0 .. +1,0].
ASIN_LREAL	Ausgang	LREAL	Arkussinus des Eingangswertes (im Bereich [-PI/2 .. +PI/2]) = 0,0 für ungültige Eingabe.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für ein ASIN_LREAL-Funktionsblockdiagramm

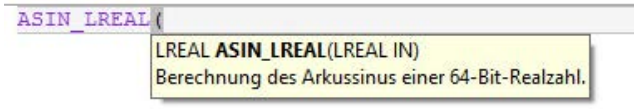


Beispiel für einen ASIN_LREAL-Kontaktplan



Beispiel für strukturierten ASIN_LREAL-Text

```
1  in := 0.5;
2  ArcSine := ASIN_LREAL(in);
```

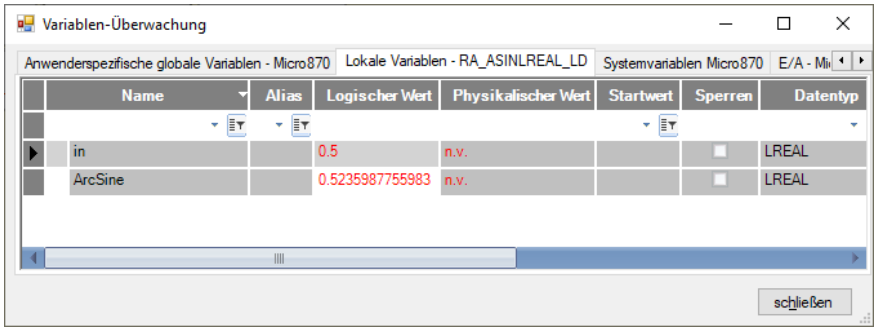


(* ST-Äquivalenz: *)

Sinus:= SIN_LREAL (Winkel);

Ergebnis:= ASIN (Sinus); (* Ergebnis ist gleich Winkel *)

Ergebnisse

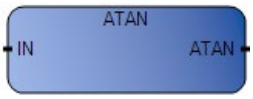


ATAN (Arkustangens)

Berechnet den Arkustangens einer Realzahl.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

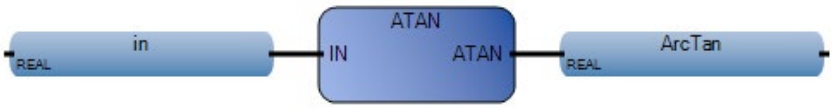
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



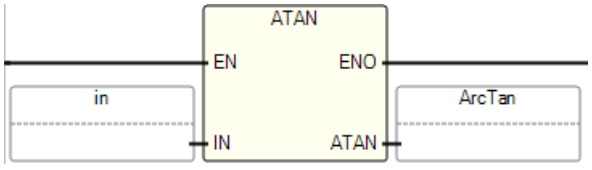
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Berechnung des aktuellen Arkustangens wird ausgeführt. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	REAL	Beliebiger realer Wert.
ATAN	Ausgang	REAL	Arkustangens des Eingangswertes (im Bereich [-PI/2 .. +PI/2]) = 0,0 für ungültige Eingabe.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Diagrammbeispiel für Funktionsblock ATAN



Beispiel für Kontaktplandiagramm ATAN



Beispiel für ATAN in strukturiertem Text

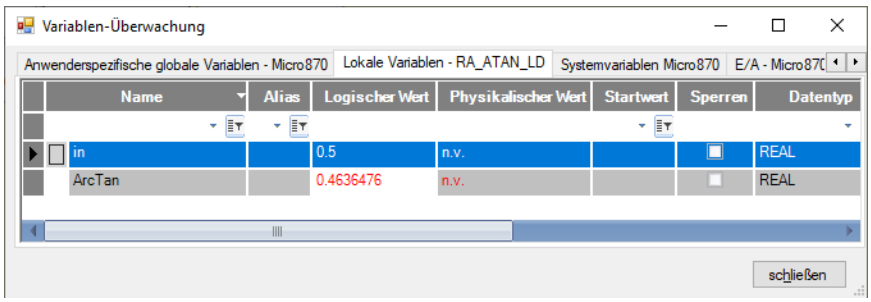
```
1  in := 0.5;
2  ArcTan := ATAN(in);
```

(* ST-Äquivalenz: *)

Tangens := TAN (Winkel);

Ergebnis := ATAN (Tangens); (* Ergebnis ist gleich Winkel *)

Ergebnisse

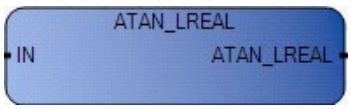


ATAN_LREAL (Arkustangens einer langen Realzahl)

Berechnet den Arkustangens einer langen Realzahl.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

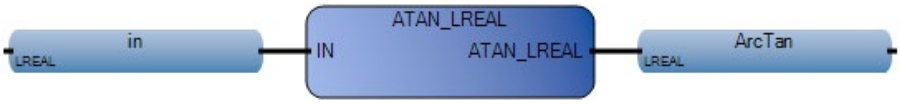
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



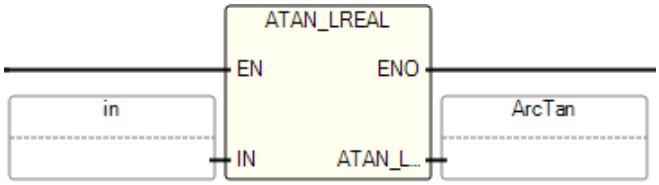
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Ausführen der aktuellen Berechnung. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	LREAL	Beliebige lange Realzahl.
ATAN_LREAL	Ausgang	LREAL	Arkustangens des Eingangswertes (im Bereich [-PI/2 .. +PI/2]) = 0,0 für ungültige Eingabe.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Diagrammbeispiel für Funktionsblock ATAN_LREAL

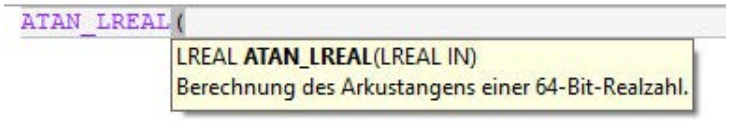


ATAN_LREAL-Kontaktplan-Beispiel



Beispiel für ATAN_LREAL in strukturiertem Text

```
1  in := 0.5;
2  ArcTan := ATAN_LREAL(in);
```

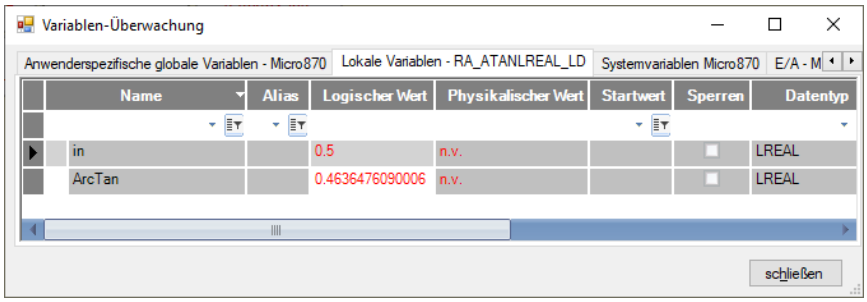


(* ST-Äquivalenz: *)

tangent := TAN_LREAL (angle);

result := ATAN_LREAL (tangent); (* Ergebnis ist gleich Winkel *)

Ergebnisse



COS (Cosinus)

Berechnet den Cosinus einer Realzahl.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

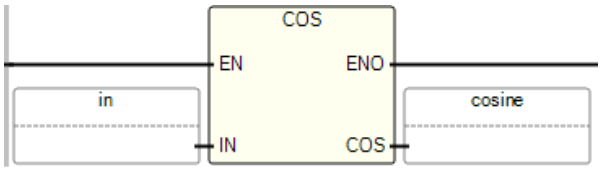
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

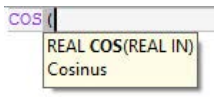
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Berechnung des aktuellen Cosinus wird ausgeführt. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	REAL	Beliebiger realer Wert.
COS	Ausgang	REAL	Cosinus des Eingangswertes (im Bereich [-1,0 .. +1,0]).
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Diagrammbeispiel für Funktionsblock COS

Beispiel für Kontaktplandiagramm COS



Beispiel für COS in strukturiertem Text



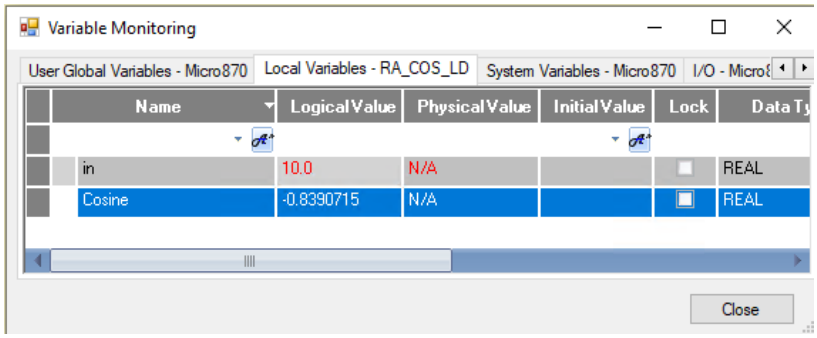

```
1  in := 10.0;
2  cosine := COS(in);
```

(* ST-Äquivalenz: *)

cosine := COS (angle);

result := ACOS (cosine); (* Ergebnis ist gleich Winkel *)

Ergebnisse

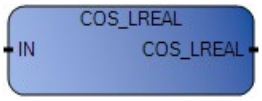


COS_LREAL (Cosinus lange Realzahl)

Berechnet den Cosinus einer langen Realzahl.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

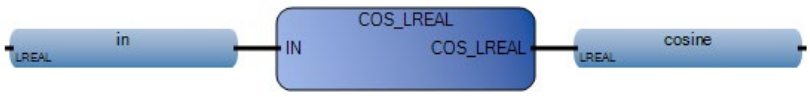
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



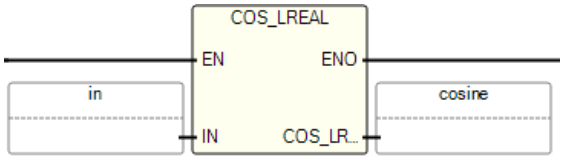
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Berechnung des aktuellen Cosinus wird ausgeführt. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	LREAL	Beliebige lange Realzahl.
COS_LREAL	Ausgang	LREAL	Cosinus des Eingangswertes (im Bereich [-1,0 .. +1,0]).
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

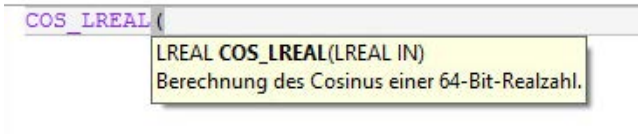
Diagrammbeispiel für Funktionsblock COS_LREAL



COS_LREAL-Kontaktplan-Beispiel



Beispiel für COS_LREAL in strukturiertem Text



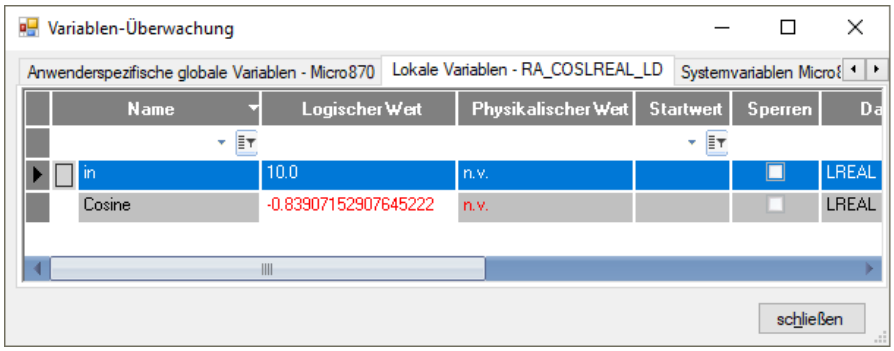

```
1  in := 10.0;
2  cosine := COS_LREAL(in);
```

(* ST-Äquivalenz: *)

cosine := COS_LREAL (angle);

result := ACOS_LREAL (cosine); (* result is equal to angle *)

Ergebnisse

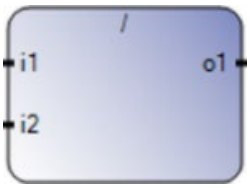


Division

Dividiert den ersten Eingabewert vom Typ „Integer“ oder „Real“ durch den zweiten Eingabewert vom Typ „Integer“ oder „Real“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - die aktuelle Divisionsberechnung wird ausgeführt. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.

Parameter	Parametertyp	Datentyp	Beschreibung
i1	Eingang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL	Dividend in Nicht-Null-Datentyp „Integer“ oder „Real“. Alle Eingaben müssen denselben Datentyp haben.
i2	Eingang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL	Divisor in Nicht-Null-Datentyp „Integer“ oder „Real“. Alle Eingaben müssen denselben Datentyp haben.
o1	Ausgang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL	Quotient der Eingaben in den Nicht-Null-Datentyp „Integer“ oder „Real“. Eingabe und Ausgabe müssen denselben Datentyp haben.
ENO	Ausgang	BOOL	Ausgabe aktivieren. Gilt nur für Kontaktplan-Programme.

Beispiel für Division in strukturiertem Text

(* ST-Äquivalenz: *)

```
ao10 := ai101 / ai102;
ao5 := (ai5 / 2) / ai53;
```

EXPT (Exponent)

Erhöht den Wert von IN (Basis) hoch EXP (Exponent), und gibt das Real-Ergebnis des Vorgangs aus.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

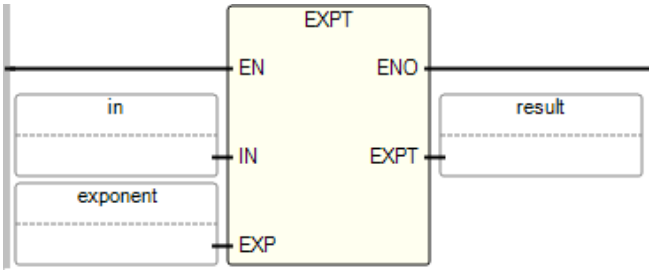
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführen der Berechnung des aktuellen Exponenten. FALSE – es findet keine Berechnung statt.
IN	Eingang	REAL	Beliebiger realer Wert mit Vorzeichen.
EXP	Eingang	DINT	Ganzzahliger Exponent.
EXPT	Ausgang	REAL	Der Real-Wert von IN hoch EXP.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

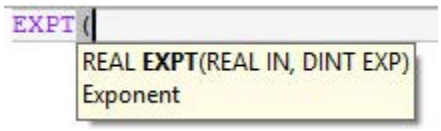
Diagrammbeispiel für Funktionsblock EXPT

Beispiel für Kontaktplandiagramm EXPT



Beispiel für EXPT in strukturiertem Text

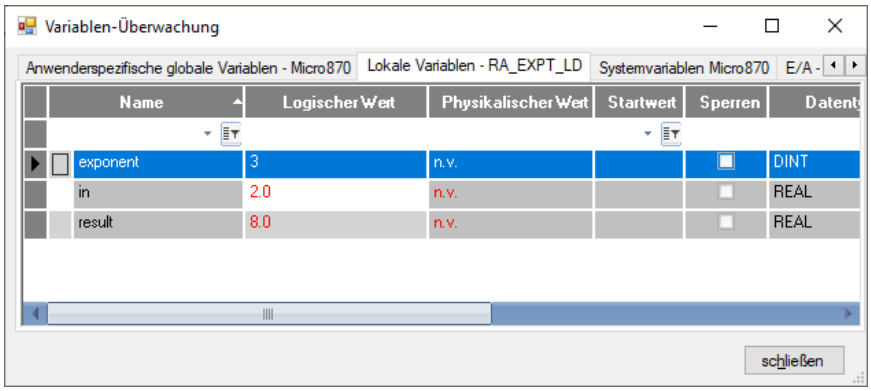
```
1  in := 2.0;
2  exponent := 3;
3  result := EXPT(in, exponent);
```



(* ST-Äquivalenz: *)

tb_size := ANY_TO_DINT (EXPT (2.0, range));

Ergebnisse



LOG (Logarithmusbasis 10)

Berechnet den Logarithmus (Basis 10) einer Realzahl.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

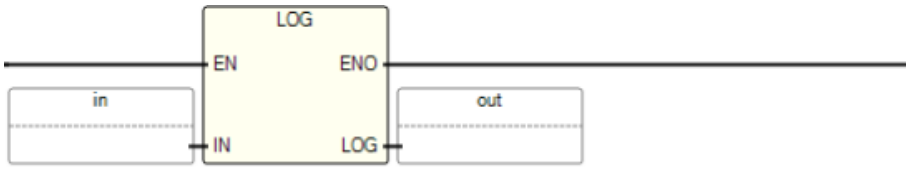
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – aktuelle Logarithmusberechnung ausführen. FALSE – es findet keine Berechnung statt.
IN	Eingang	REAL	Muss größer als Null sein.
LOG	Ausgang	REAL	Logarithmus (Basis 10) des Eingangswerts. Als Ergebnis wird -3,4E+38 für einen IN-Wert von null und einen negativen IN-Wert zurückgegeben.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe.

Parameter	Parametertyp	Datentyp	Beschreibung
			Gilt für Kontaktplan-Programme.

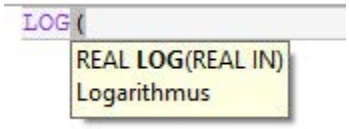
Beispiel für ein LOG-Funktionsblockdiagramm

Beispiel für Kontaktplandiagramm LOG



Beispiel für LOG in strukturiertem Text

```
1  in := 10.0;
2  output := LOG(in);
```

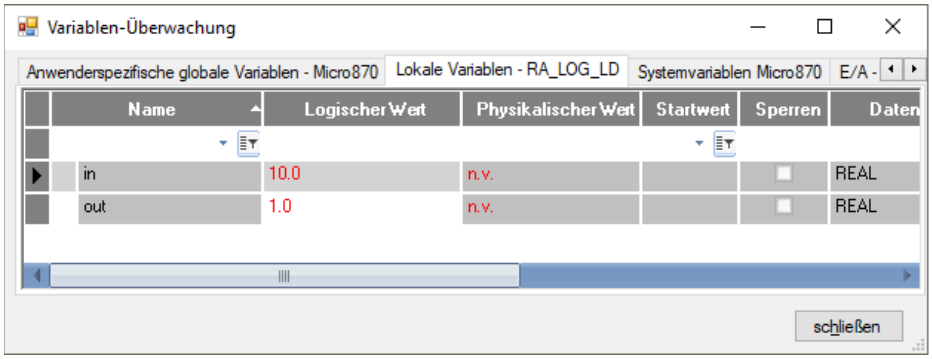


(* ST-Äquivalenz: *)

xpos := ABS (xval);

xlog := LOG (xpos);

Ergebnisse



MOD (Modulo)

Teilen Sie die IN-Eingabe durch die Basiseingabe und platzieren Sie den Rest in der MOD-Ausgabe.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

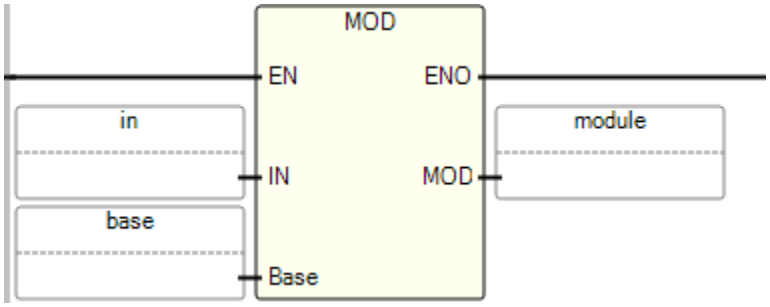
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

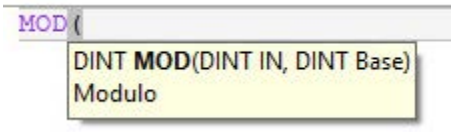
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Modulberechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	DINT	Beliebiger ganzzahliger Wert mit Vorzeichen.
Base	Eingang	DINT	Muss größer als Null sein.
MOD	Ausgang	DINT	Modulo-Berechnung (Eingabe MOD-Basis) / gibt -1 zurück, wenn Base <= 0.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Diagrammbeispiel für Funktionsblock MOD

Beispiel für Kontaktplandiagramm MOD



Beispiel für MOD in strukturiertem Text



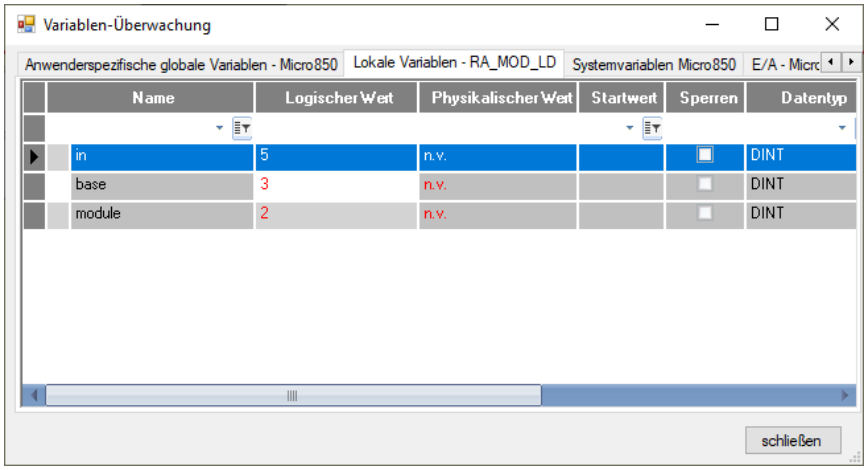

```
1  in := 5;
2  base := 3;
3  module := MOD(in, base);
```

(* ST-Äquivalenz: *)

division_result := (value / divider); (* ganzzahlige Division *)

rest_of_division := MOD (value, divider); (* Rest der Division *)

Ergebnisse



MOV (Verschieben)

Weist den Eingabewert (i1) der Ausgabe (o1) zu.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Verwenden Sie für strukturierte Textprogramme den „Gleich“ (=)-Operator statt MOV.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. ,TRUE – Ausführung der direkten Verknüpfung zu einer Ausgabeberechnung. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL DINT REAL TIME STRING SINT USINT INT UINT UDINT LINT ULINT DATE (Datum) LREAL BYTE WORD DWORD LWORD	Eingabe und Ausgabe müssen denselben Datentyp haben.
o1	Ausgang	BOOL DINT REAL TIME STRING SINT USINT INT UINT UDINT LINT ULINT DATE (Datum) LREAL BYTE WORD DWORD LWORD	Eingabe und Ausgabe müssen denselben Datentyp haben.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten Text

(* ST-Äquivalenz: *)

```
ao23 := ai10;
```

Multiplikation

Multipliziert zwei oder mehr Werte vom Typ „Integer“ oder „Real“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführung der aktuellen Multiplikation. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL	Faktor in Datentyp „Integer“ oder „Real“. Alle Eingaben müssen denselben Datentyp haben.
i2	Eingang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT LINT LWORD REAL LREAL	Faktor in Datentyp „Integer“ oder „Real“. Alle Eingaben müssen denselben Datentyp haben.
o1	Ausgang	SINT USINT BYTE INT	Produkt der Eingaben in den Datentyp „Integer“ oder „Real“. Eingabe und Ausgabe müssen denselben Datentyp haben.

Parameter	Parametertyp	Datentyp	Beschreibung
		UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL	
ENO	Ausgang	BOOL	Ausgabe aktivieren. Gilt nur für Kontaktplan-Programme.

Beispiel für Multiplikation in strukturiertem Text

(* ST-Äquivalenz *)

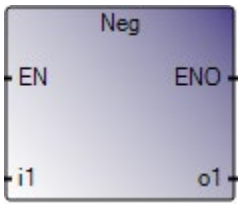
```
ao10 := ai101 * ai102;
ao5 := (ai51 * ai52) * ai53;
```

Neg (Negation)

Konvertiert einen Wert in einen negierten Wert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführung der aktuellen Konvertierung in Negativberechnung. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	SINT INT DINT LINT REAL LREAL	Eingabe und Ausgabe müssen denselben Datentyp haben.
o1	Ausgang	SINT INT	Eingabe und Ausgabe müssen denselben Datentyp haben.

Parameter	Parametertyp	Datentyp	Beschreibung
		DINT LINT REAL LREAL	
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für Neg in strukturiertem Text

(* ST-Äquivalenz: *)

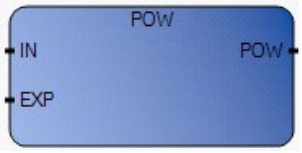
```
ao23 := - (ai10);
ro100 := - (ri1 + ri2);
```

POW (Potenzieren)

Wenn 'Basis' das erste und 'Exponent' das zweite Argument ist, wird das reellwertige Ergebnis von (Basis Exponent) berechnet.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

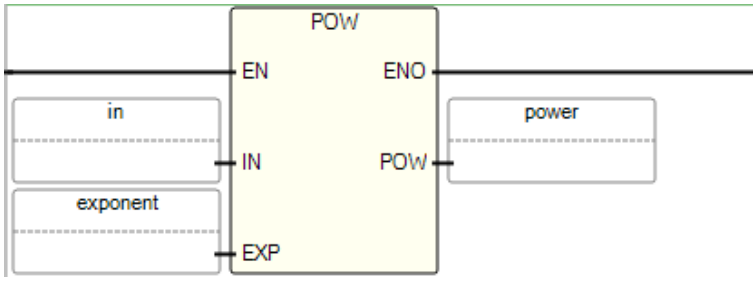


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

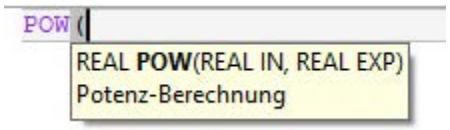
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführen der Berechnung des aktuellen Exponenten. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	REAL	Realzahl, mit der potenziert werden soll.
EXP	Eingang	REAL	Potenz (Exponent).
POW	Ausgang	REAL	(IN EXP) 1,0, wenn IN nicht gleich 0,0 und EXP gleich 0,0 0,0, wenn IN gleich 0,0 und EXP negativ 0,0, wenn sowohl IN als auch EXP gleich 0,0 0,0, wenn IN negativ und EXP nicht einer Ganzzahl entspricht.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für ein POW-Funktionsblockdiagramm

Beispiel Kontaktplandiagramm POW



Beispiel für den Strukturierten Text von POW

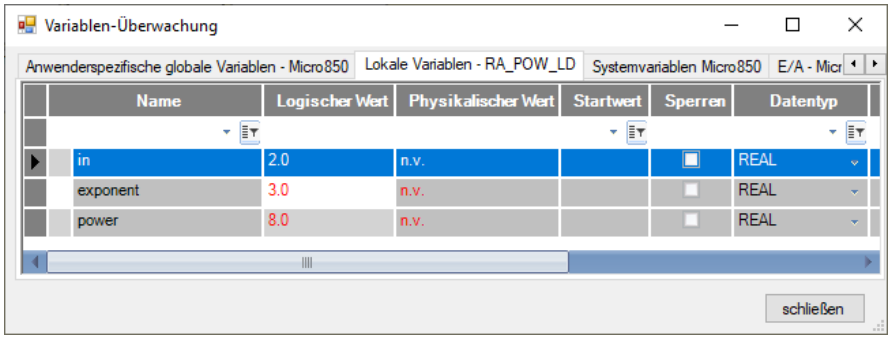



```
1  in := 2.0;
2  exponent := 3.0;
3  power := POW(in, exponent);
```

(* ST-Äquivalenz: *)

result := POW (xval, power);

Ergebnisse



RAND (Zufallswert)

RAND liefert nach dem Zufallsprinzip ausgewählte ganzzahlige Werte aus

einem definierten Bereich.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

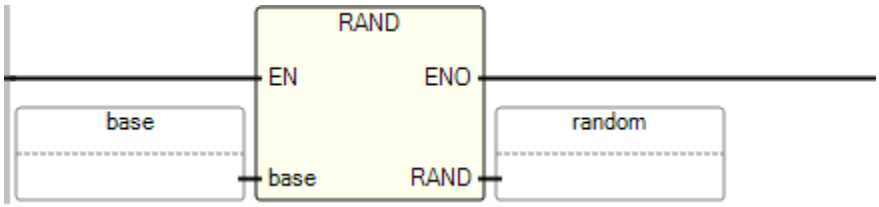
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführen der Berechnung ganzzahliger Zufallszahlen. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
base	Eingang	DINT	Definiert den unterstützten Zahlenbereich.
RAND	Ausgang	DINT	Zufälliger Wert im Bereich [0..Basis-1].
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für ein RAND-Funktionsblockdiagramm

Beispiel für Kontaktplandiagramm RAND



Beispiel für RAND in strukturiertem Text

```
1  base := 10;
2  random := RAND(base);
```

(* ST-Äquivalenz: *)

selected := MUX4 (RAND (4), 1, 4, 8, 16);

(*

Zufallsauswahl von 1 von 4 gegebenen Werten

Der durch den RAND-Aufruf ausgegebene Wert liegt im Bereich [0-3],

wenn also MUX4 'selected' ausgegeben wird, wird der folgende Wert 'zufällig' geholt

1 wenn 0 von RAND ausgegeben wird,

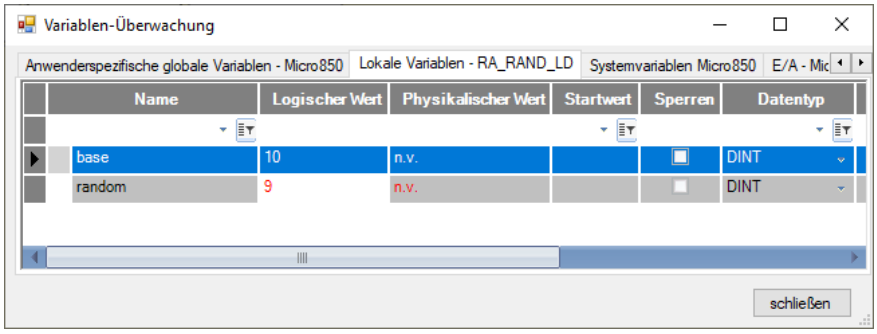
oder 4 wenn 1 von RAND ausgegeben wird,

oder 8 wenn 2 von RAND ausgegeben wird,

oder 16 wenn 3 von RAND ausgegeben wird,

*)

Ergebnisse



SIN (Sinus)

Berechnet den Sinus von einer Realzahl.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

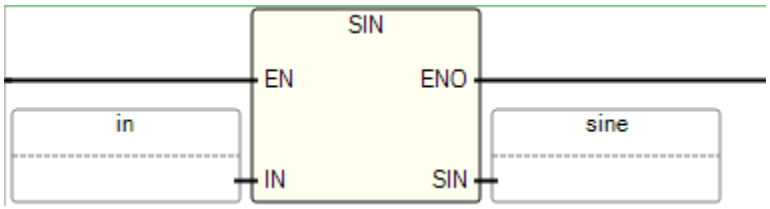
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

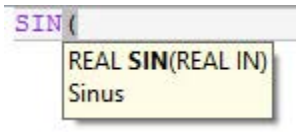
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Ausführen der Berechnung des aktuellen Sinus. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	REAL	Beliebiger realer Wert.
SIN	Ausgang	REAL	Sinus des Eingangswertes (im Bereich [-1,0 .. +1,0]).
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für ein SIN-Funktionsblockdiagramm

Beispiel für Kontaktplandiagramm SIN



Beispiel für SIN in strukturiertem Text



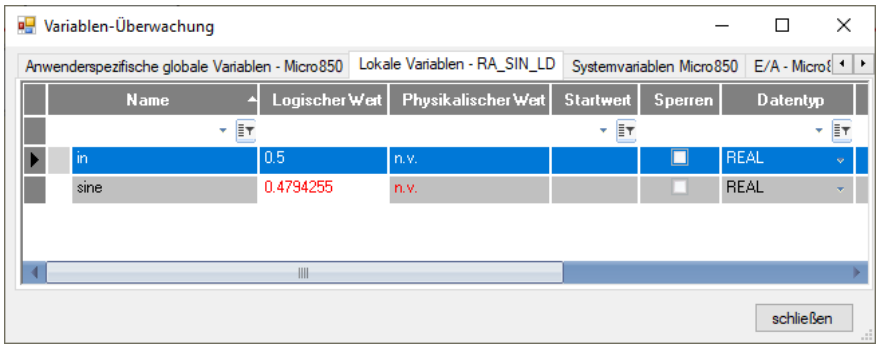

```
1  in := 0.5;
2  sine := SIN(in);
```

(* ST-Äquivalenz: *)

sine := SIN (angle);

```
result := ASIN (sine); (* Ergebnis ist gleich Winkel *)
```

Ergebnisse

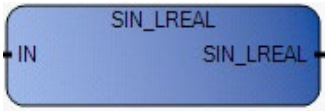


SIN_LREAL (Sinus langer Realzahl)

Berechnet den Sinus einer langen Realzahl.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

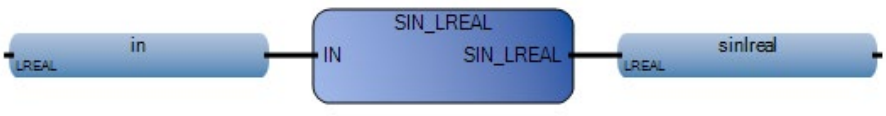
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



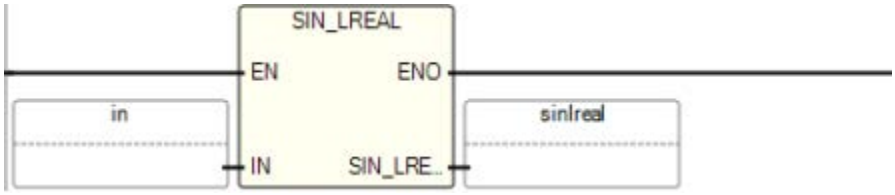
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Ausführen der aktuellen Berechnung. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	LREAL	Beliebige lange Realzahl.
SIN_LREAL	Ausgang	LREAL	Sinus des Eingangswertes (im Bereich [-1,0 .. +1,0]).
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

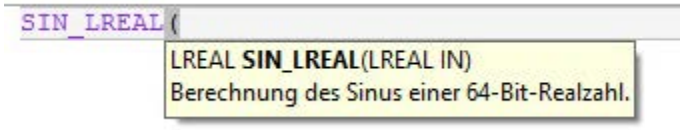
Beispiel für ein SIN_LREAL-Funktionsblock-Diagramm



Beispiel für einen SIN_LREAL-Kontaktplan



Beispiel für einen strukturierten SIN_LREAL-Text

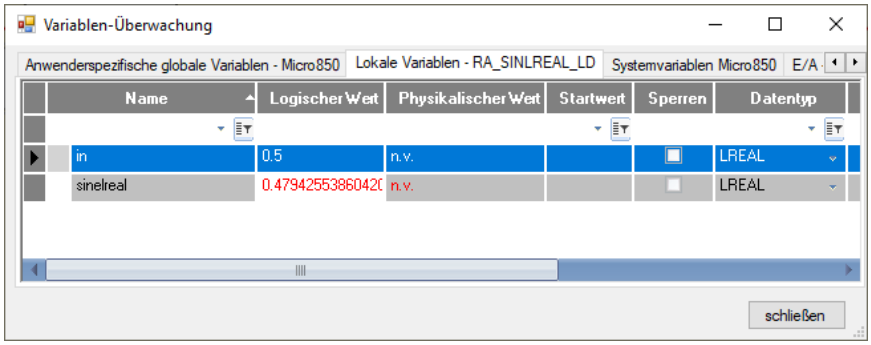



```
1  in := 0.5;
2  sinlreal := SIN_LREAL(in);
```

(* ST-Äquivalenz: *)

TESTOUTPUT1 := SIN_LREAL(TESTINPUT1) ;

Ergebnisse



SQRT (Quadratwurzel)

Berechnet die Quadratwurzel einer reellen Zahl.

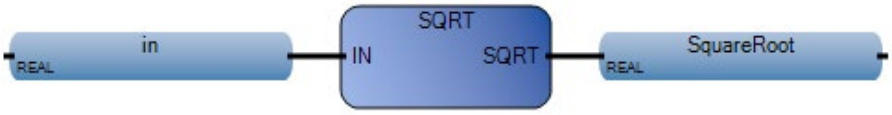
Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

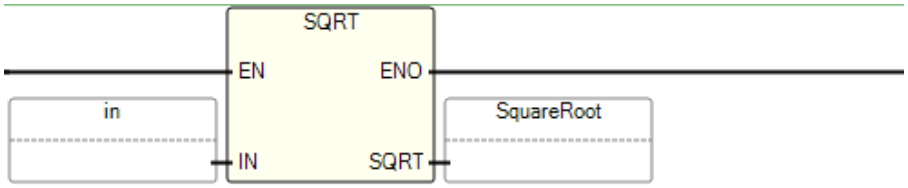
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführen der aktuellen Quadratwurzel-Berechnung. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	REAL	Muss größer oder gleich Null sein.
SQRT	Ausgang	REAL	Quadratwurzel des Eingangswertes. Bei negativen IN-Werten wird 0 als Ergebnis zurückgegeben.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für ein SQRT-Funktionsblockdiagramm

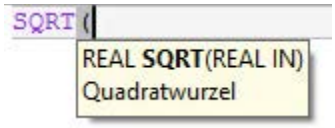


Beispiel für einen SQRT-Kontaktplan



Beispiel für SQRT in strukturiertem Text

```
1  in := 16.0;
2  SquareRoot := SQRT(in);
```

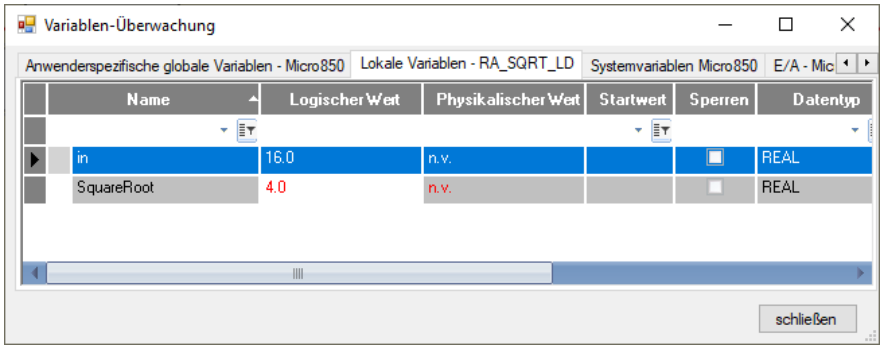


(* ST-Äquivalenz: *)

xpos := ABS (xval);

xroot := SQRT (xpos);

Ergebnisse

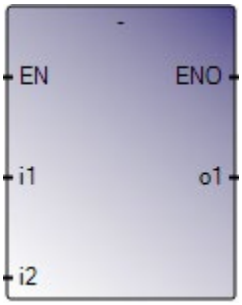


Subtraktion

Subtrahiert einen Wert vom Typ „Integer“, „Real“ oder „Time“ von einem anderen Wert vom Typ „Integer“, „Real“ oder „Time“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführen der aktuellen Addition. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME	Minuend in beliebigem Datentyp „Integer“, „Real“ oder „Time“. Alle Eingaben müssen denselben Datentyp haben.

Parameter	Parametertyp	Datentyp	Beschreibung
i2	Eingang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME	Subtrahend in beliebigem Datentyp „Integer", „Real" oder „Time". Alle Eingaben müssen denselben Datentyp haben.
o1	Ausgang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME	Differenz zwischen Minuend und Subtrahend in beliebigem Datentyp „Integer", „Real" oder „Time". Ausgaben müssen denselben Datentyp haben wie Eingaben.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten Subtraktions-Text

(* ST-Äquivalenz: *)

```
ao10 := ai101 - ai102;
ao5 := (ai51 - 1) - ai53;
```

TAN (Tangens)

Berechnet den Tangens einer Realzahl.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

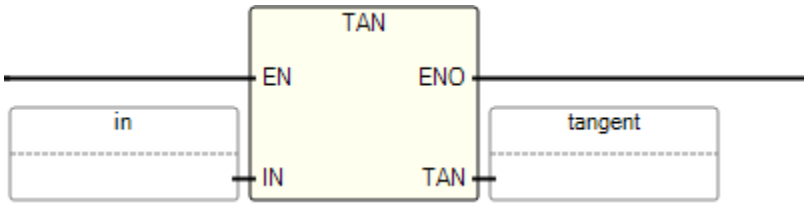
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführen der Berechnung des aktuellen Tangens. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	REAL	Kann nicht gleich PI/2 Modulo PI sein.
TAN	Ausgang	REAL	Tangens des Eingangswerts = 1E+38 für ungültigen Eingangswert.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für ein TAN-Funktionsblockdiagramm

Beispiel für Kontaktplandiagramm TAN



Beispiel für TAN in strukturiertem Text

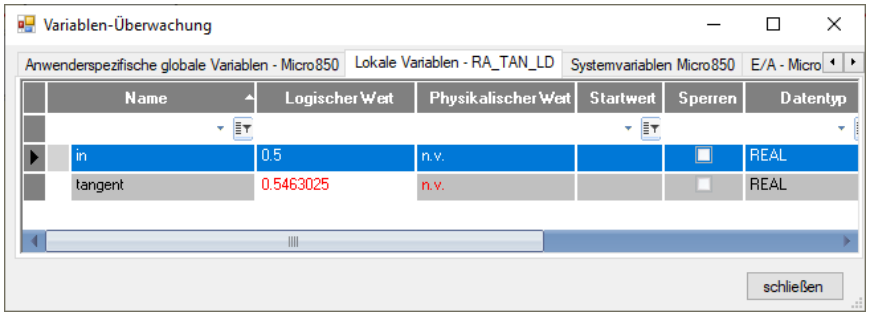
```
1  in := 0.5;
2  tangent := TAN(in);
```

(* ST-Äquivalenz: *)

Tangens := TAN (Winkel);

Ergebnis := ATAN (Tangens); (* Ergebnis ist gleich Winkel *)

Ergebnisse

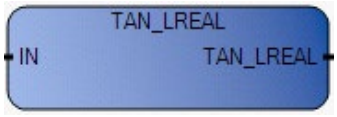


TAN_LREAL (Tangens-Typ "Long Real")

Berechnet den Tangens einer langen Realzahl.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

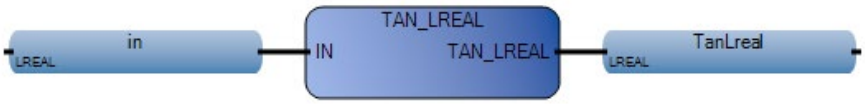
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



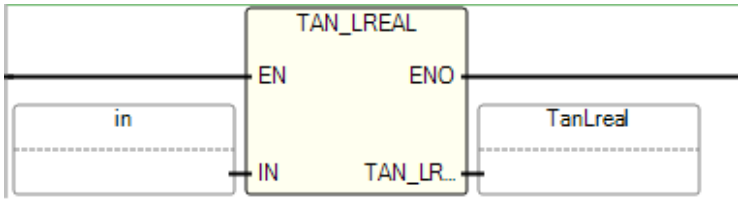
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Ausführen der aktuellen Berechnung. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	LREAL	Kann nicht gleich PI/2 Modulo PI sein.
TAN_LREAL	Ausgang	LREAL	Tangens des Eingangswerts = 1E+38 für ungültigen Eingangswert.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für ein TAN_LREAL-Funktionsblockdiagramm



Beispiel für einen TAN_LREAL-Kontaktplan



Beispiel für strukturierten TAN_LREAL-Text

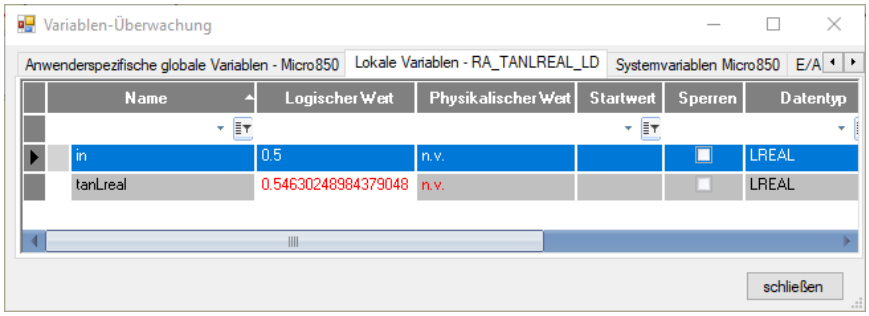
```
1  in := 0.5;
2  TanLreal := TAN_LREAL(in);
```

(* ST-Äquivalenz: *)

tangent := TAN_LREAL (angle);

result := ATAN_LREAL (tangent); (* Ergebnis ist gleich Winkel *)

Ergebnisse



TRUNC (Abschneiden)

Schneidet Realzahlen ab und belässt nur eine Ganzzahl.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

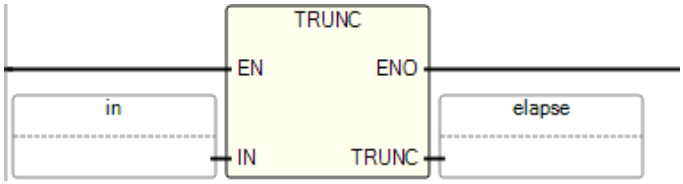
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Berechnung des Abschneidens von Realzahlen. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	REAL	Beliebiger realer Wert.
TRUNC	Ausgang	REAL	Wenn IN>0, ist die größte Ganzzahl kleiner oder gleich dem Eingangswert. Wenn IN<0, ist die kleinste Ganzzahl größer oder gleich dem Eingangswert.

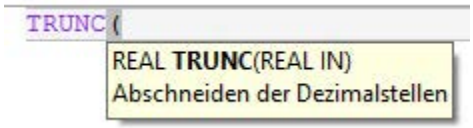
Parameter	Parametertyp	Datentyp	Beschreibung
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für ein TRUNC-Funktionsblockdiagramm

Beispiel für Kontaktplandiagramm TRUNC



Beispiel für TRUNC in strukturiertem Text



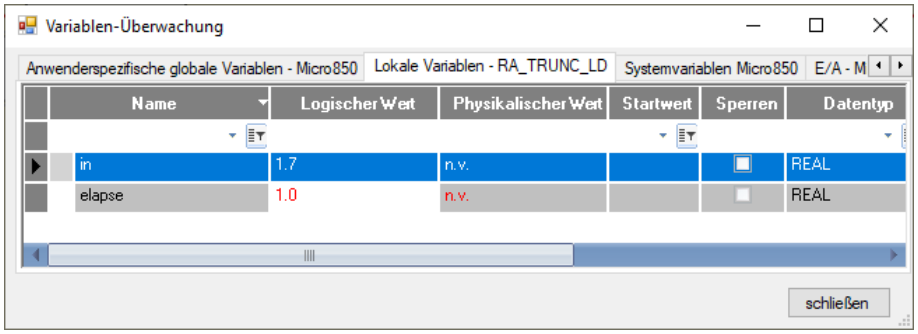

```
1  in := 1.7;
2  truncation := TRUNC(in);
```

(* ST-Äquivalenz: *)

result := TRUNC (+2.67) + TRUNC (-2.0891);

(* bedeutet: result := 2.0 + (-2.0) := 0.0; *)

Ergebnisse



ASCII-Anweisungen zu seriellen Ports

Verwenden Sie die ASCII-Anweisungen für den seriellen Anschluss, um den Kommunikationskanal für den Empfang oder das Senden von Daten zu verwenden oder zu ändern.

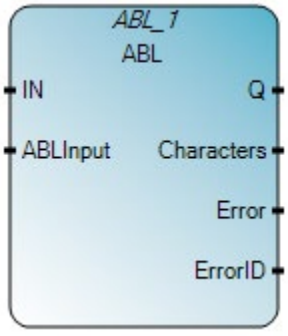
Funktionsblock	Beschreibung
ABL auf seite 103	Ermittelt die Anzahl von Zeichen im Puffer (einschließlich Zeilenendezeichen).
ACB auf seite 109	Ermittelt die Gesamtanzahl von Zeichen im Puffer.
ACL auf seite 105	Löscht die Empfangs- und Übertragungspuffer.
AHL auf seite 107	Legt Handshake-Leitungen des Modems fest, oder setzt sie zurück.
ARD auf seite 111	Liest Zeichen aus dem Eingabepuffer und legt diese Zeichen in einer Zeichenfolge ab.
ARL auf seite 114	Liest eine Zeile mit Zeichen aus dem Eingabepuffer und legt diese Zeichen in einer Zeichenfolge ab.
AWA auf seite 116	Schreibt eine Zeichenfolge mit zwei angefügten (benutzerkonfigurierten) Zeichen auf ein externes Gerät.
AWT auf seite 118	Schreibt Zeichen aus einer Quellzeichenfolge in ein externes Gerät.

ABL (ASCII-Test für Pufferzeile)

Zählt die Anzahl der ASCII-Zeichen im Eingabepuffer bis einschließlich des Zeilenendezeichens.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangszustand von Strompfad. TRUE – Wird eine steigende Flanke erkannt, starten Sie den Funktionsblock unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE - der Anweisungsblock befindet sich im Leerlauf.

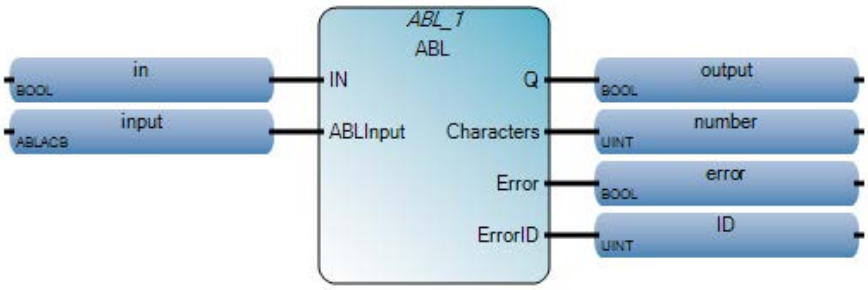
Parameter	Parametertyp	Datentyp	Beschreibung
ABLInput	Eingang	ABLACB	Der zu betreibende Kanal. Mit dem Datentyp ABLACB auf seite 121 können Sie die Parameter Kanal, TriggerType und Abbrechen für ABLInput definieren.
Q	Ausgang	BOOL	Gibt an, dass die Zeichenzählung noch läuft oder abgeschlossen ist. Die Ausgänge werden asynchron vom Programm-Scan aktualisiert. Die Anweisung kann von Ausgabe Q nicht neu ausgelöst werden, da IN von der Kante ausgelöst wird. TRUE - der Funktionsblock ist abgeschlossen. FALSE - der Funktionsblock ist nicht abgeschlossen.
Characters	Ausgang	UINT	Die Anzahl der Zeichen im Puffer. Das Pufferlimit beträgt 82 Zeichen.
Error	Ausgang	BOOL	Zeigt das Vorhandensein eine Fehlerbedingung an. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler für diese Anweisung sind in ABL-Fehlercodes definiert.

ABL-Fehlercodes

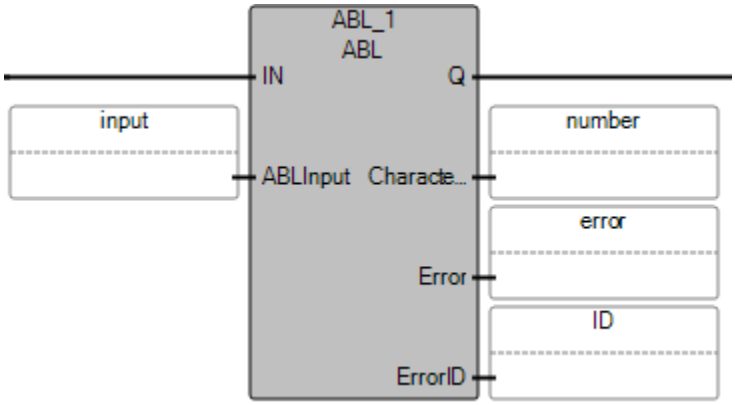
Verwenden Sie dieser Tabelle um die ABL-Fehlercodes und -Beschreibungen zu bestimmen.

Fehlercode	Fehlerbeschreibung
03	Übertragung kann nicht abgeschlossen werden, da das Clear-to-Send-Signal abgebrochen ist.
06	Unzulässiger Parameter erkannt.
07	Senden oder Empfangen von ASCII kann nicht abgeschlossen werden, da die Kanalkonfiguration über das Dialogfeld zur Kanalkonfiguration ausgeschaltet wurde.
08	ASCII-Schreibvorgang kann nicht abgeschlossen werden, da bereits eine ASCII-Übertragung durchgeführt wird.
09	Die angeforderte ASCII-Kommunikation wird von der aktuellen Kanalkonfiguration nicht unterstützt.
10	Der „Cancel"-Befehl wurde gesetzt; Ausführung der Anweisung wird gestoppt. Keine Aktion erforderlich.
11	Die angeforderte Länge für die Zeichenkette ist entweder ungültig, eine negative Zahl, größer als 82 oder 0. Gilt für die Funktionsblöcke ARD und ARL.
13	Die angeforderte (.LEN) im Steuerblock ist eine negative Zahl oder ein Wert, der die zusammen mit der Quellzeichenfolge gespeicherte Zeichenfolge übersteigt. Gilt für die Funktionsblöcke AWA und AWT.
14	Der ACL-Funktionsblock wurde abgebrochen.
16	Serielle Anschlüsse unterstützen keine RTS- oder CTS-Steuerleitungen.

Beispiel für ABL-Funktionsblockdiagramm



Beispiel für ABL-Kontaktplandiagramm



Beispiel für strukturierten ABL-Text

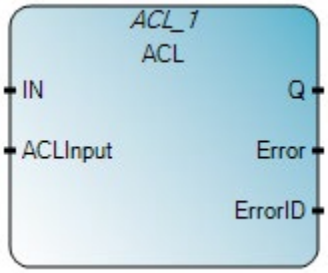
```
1  ABL_1(in, input);
2  output := ABL_1.Q;
3  number := ABL_1.Characters;
4  error := ABL_1.Error;
5  ID := ABL_1.ErrorID;
```

ACL (ASCII-Puffer löschen)

Löscht die Empfangs- und Übertragungspuffer und entfernt die Anweisungen aus der ASCII-Warteschlange.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangszustand von Strompfad. TRUE – Wird eine steigende Flanke erkannt, starten Sie den Funktionsblock unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE - der Anweisungsblock befindet sich im Leerlauf.

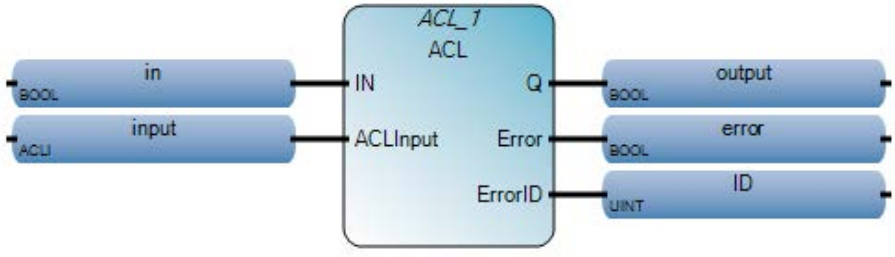
Parameter	Parametertyp	Datentyp	Beschreibung
ACLInput	Eingang	ACLI	Der zu betreibende Kanal und der Zustand der Übertragungs- und Empfangspuffer. Bei RXBuffer wird der Empfangspuffer gelöscht, und die Funktionsblöcke zum Empfang von ASCII (ARL und ARD) werden aus der ASCII-Warteschlange entfernt. Bei TXBuffer wird der Übertragungspuffer geleert, und die Funktionsblöcke zur Übertragung von ASCII (AWA und AWT) werden aus der ASCII-Warteschlange entfernt. Mit dem Datentyp ACLI auf seite 121 können Sie die Kanal-, RXBuffer- und TXBuffer-Parameter für ACLInput definieren.
Q	Ausgang	BOOL	Gibt an, wenn der Löschvorgang der ASCII-Warteschlange gerade ausgeführt wird oder abgeschlossen ist. TRUE - der Funktionsblock ist abgeschlossen. FALSE - der Funktionsblock ist nicht abgeschlossen.
Error	Ausgang	BOOL	Zeigt das Vorhandensein eine Fehlerbedingung an. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler für diese Anweisung sind in ABL-Fehlercodes definiert.

ABL-Fehlercodes

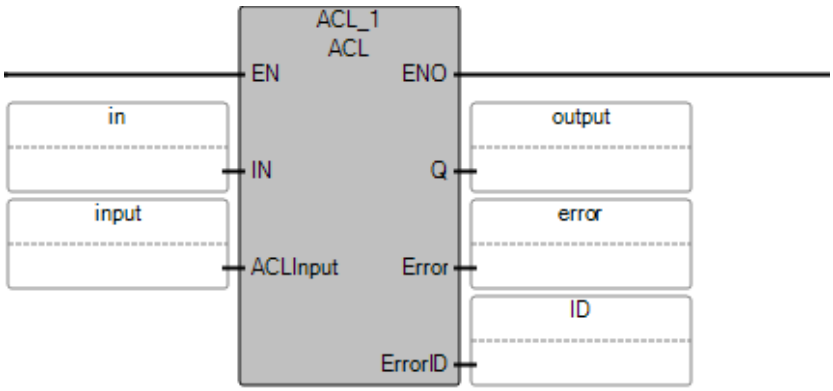
Verwenden Sie dieser Tabelle um die ABL-Fehlercodes und -Beschreibungen zu bestimmen.

Fehlercode	Fehlerbeschreibung
03	Übertragung kann nicht abgeschlossen werden, da das Clear-to-Send-Signal abgebrochen ist.
06	Unzulässiger Parameter erkannt.
07	Senden oder Empfangen von ASCII kann nicht abgeschlossen werden, da die Kanalkonfiguration über das Dialogfeld zur Kanalkonfiguration ausgeschaltet wurde.
08	ASCII-Schreibvorgang kann nicht abgeschlossen werden, da bereits eine ASCII-Übertragung durchgeführt wird.
09	Die angeforderte ASCII-Kommunikation wird von der aktuellen Kanalkonfiguration nicht unterstützt.
10	Der „Cancel"-Befehl wurde gesetzt; Ausführung der Anweisung wird gestoppt. Keine Aktion erforderlich.
11	Die angeforderte Länge für die Zeichenkette ist entweder ungültig, eine negative Zahl, größer als 82 oder 0. Gilt für die Funktionsblöcke ARD und ARL.
13	Die angeforderte (.LEN) im Steuerblock ist eine negative Zahl oder ein Wert, der die zusammen mit der Quellzeichenfolge gespeicherte Zeichenfolge übersteigt. Gilt für die Funktionsblöcke AWA und AWT.
14	Der ACL-Funktionsblock wurde abgebrochen.
16	Serielle Anschlüsse unterstützen keine RTS- oder CTS-Steuerleitungen.

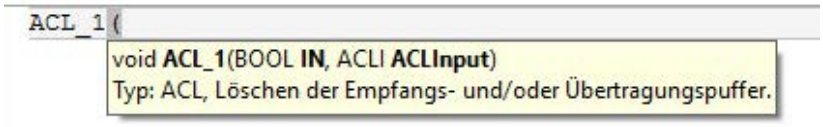
Beispiel für ACL-Funktionsblockdiagramm



Beispiel für ACL-Kontaktplandiagramm



Beispiel für strukturierten ACL-Text



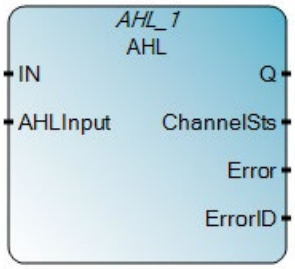

```
1  ACL_1(in, input);
2  output := ACL_1.Q;
3  error := ACL_1.Error;
4  ID := ACL_1.ErrorID;
```

AHL (ASCII-Handshake-Leitungen)

Legt die RS-232 Request to Send (RTS)-Handshake-Steuerleitungen für Ihr Modem fest oder setzt sie zurück.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangszustand von Strompfad. TRUE - bei erkannter Anstiegflanke wird der Anweisungsblock gestartet, vorausgesetzt, dass die letzte Operation abgeschlossen ist. FALSE - der Anweisungsblock befindet sich im Leerlauf.

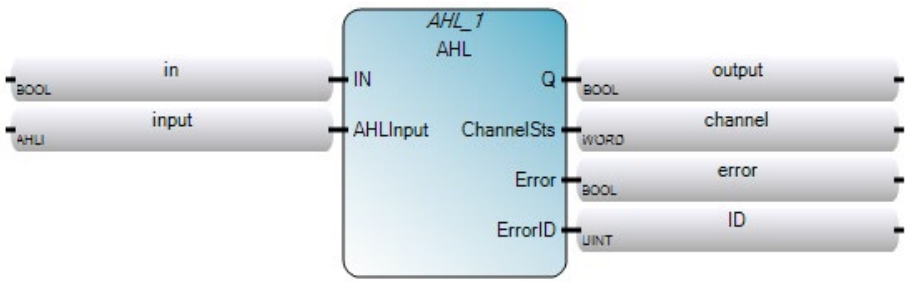
Parameter	Parametertyp	Datentyp	Beschreibung
AHLInput	Eingang	AHLI	Der zu betreibende Kanal und das Festlegen oder Zurücksetzen der RTS-Steuerleitung für das Modem. Mit dem Datentyp AHLI auf seite 122 können Sie die Parameter „Kanal", „SetRts", „ClrRts" und „Abbrechen" für AHLInput definieren.
Q	Ausgang	BOOL	Gibt an, wann das Festlegen oder Zurücksetzen abgeschlossen ist. Die Ausgaben dieser Anweisung werden asynchron vom Programm-Scan aktualisiert. Die Anweisung kann von Ausgabe Q nicht neu ausgelöst werden, da IN von der Kante ausgelöst wird. TRUE - der Funktionsblock ist abgeschlossen. FALSE - der Funktionsblock ist nicht abgeschlossen.
ChannelSts	Ausgang	WORD	Zeigt den aktuellen Status (0000 bis 001F) der Handshake-Leitungen für den angegebenen Kanal an.
Error	Ausgang	BOOL	Zeigt das Vorhandensein eine Fehlerbedingung an. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler für diese Anweisung sind in ABL-Fehlercodes definiert.

ABL-Fehlercodes

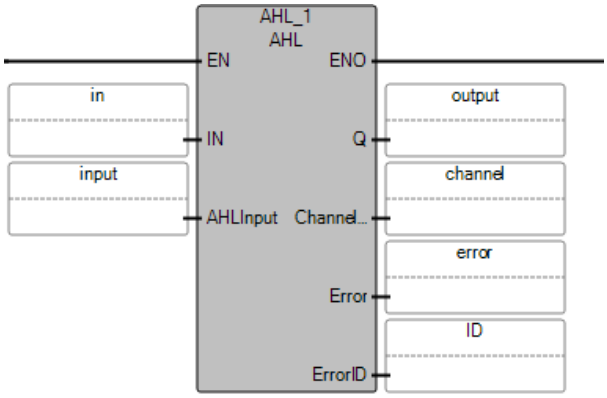
Verwenden Sie dieser Tabelle um die ABL-Fehlercodes und -Beschreibungen zu bestimmen.

Fehlercode	Fehlerbeschreibung
03	Übertragung kann nicht abgeschlossen werden, da das Clear-to-Send-Signal abgebrochen ist.
06	Unzulässiger Parameter erkannt.
07	Senden oder Empfangen von ASCII kann nicht abgeschlossen werden, da die Kanalkonfiguration über das Dialogfeld zur Kanalkonfiguration ausgeschaltet wurde.
08	ASCII-Schreibvorgang kann nicht abgeschlossen werden, da bereits eine ASCII-Übertragung durchgeführt wird.
09	Die angeforderte ASCII-Kommunikation wird von der aktuellen Kanalkonfiguration nicht unterstützt.
10	Der „Cancel"-Befehl wurde gesetzt; Ausführung der Anweisung wird gestoppt. Keine Aktion erforderlich.
11	Die angeforderte Länge für die Zeichenkette ist entweder ungültig, eine negative Zahl, größer als 82 oder 0. Gilt für die Funktionsblöcke ARD und ARL.
13	Die angeforderte (.LEN) im Steuerblock ist eine negative Zahl oder ein Wert, der die zusammen mit der Quellzeichenfolge gespeicherte Zeichenfolge übersteigt. Gilt für die Funktionsblöcke AWA und AWT.
14	Der ACL-Funktionsblock wurde abgebrochen.
16	Serielle Anschlüsse unterstützen keine RTS- oder CTS-Steuerleitungen.

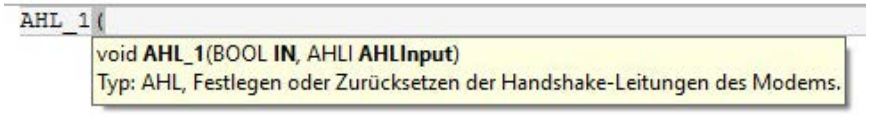
Beispiel für AHL-Funktionsblockdiagramm



Beispiel für AHL Kontaktplandiagramm



Beispiel für strukturierten AHL-Text



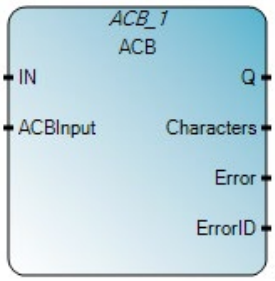

```
1  AHL_1(in, input);
2  output := AHL_1.Q;
3  channel := AHL_1.ChannelSts;
4  error := AHL_1.Error;
5  ID := AHL_1.ErrorID;
```

ACB (ASCII-Zeichen im Puffer)

Dieser Block zählt die Gesamtzahl der ASCII-Zeichen in der Zwischenablage, einschließlich Zeilenende.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangszustand von Strompfad. TRUE - bei erkannter Anstiegflanke wird der Anweisungsblock gestartet, vorausgesetzt, dass die letzte Operation abgeschlossen ist. FALSE - der Anweisungsblock befindet sich im Leerlauf.

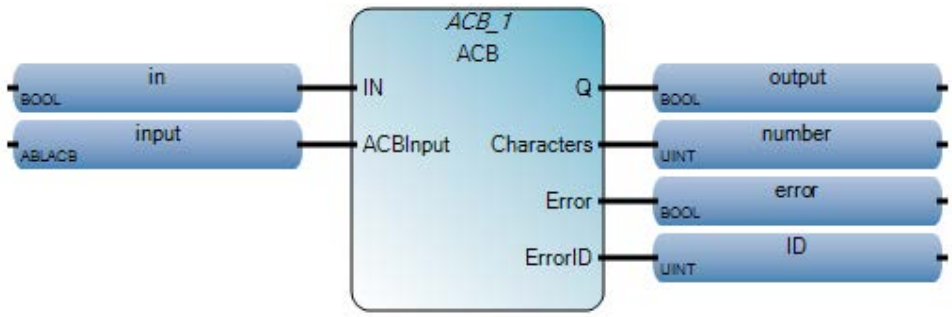
Parameter	Parametertyp	Datentyp	Beschreibung
ACBInput	Eingang	ABLACB auf seite 121	Der zu betreibende Kanal. Verwenden Sie den Datentyp „ABLACB", um die Kanal-, Auslösertyp- und Abbruchparameter für ACBInput zu definieren.
Q	Ausgang	BOOL	Gibt an, ob die Anzahl der Zeichen gerade ausgeführt wird oder abgeschlossen ist. Die Ausgaben dieses Funktionsblocks werden asynchron vom Programm-Scan aktualisiert. Die Anweisung kann von Ausgabe Q nicht neu ausgelöst werden, da IN von der Kante ausgelöst wird. TRUE – Die Zählung ist abgeschlossen. FALSE – Die Zählung wird gerade ausgeführt.
Characters	Ausgang	UINT	Die Anzahl der Zeichen im Puffer.
Error	Ausgang	BOOL	Zeigt das Vorhandensein eine Fehlerbedingung an. FALSE - Kein Fehler. WAHR – es wurde ein Fehler erkannt.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler für diese Anweisung sind in ABL-Fehlercodes definiert.

ACB-Fehlercodes

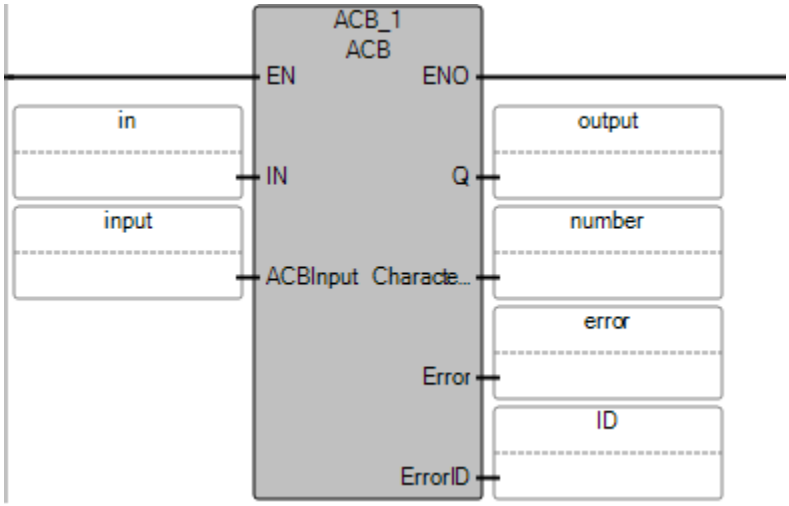
Verwenden Sie dieser Tabelle um die ABL-Fehlercodes und -Beschreibungen zu bestimmen.

Fehlercode	Fehlerbeschreibung
03	Übertragung kann nicht abgeschlossen werden, da das Clear-to-Send-Signal abgebrochen ist.
06	Unzulässiger Parameter erkannt.
07	Senden oder Empfangen von ASCII kann nicht abgeschlossen werden, da die Kanalkonfiguration über das Dialogfeld zur Kanalkonfiguration ausgeschaltet wurde.
08	ASCII-Schreibvorgang kann nicht abgeschlossen werden, da bereits eine ASCII-Übertragung durchgeführt wird.
09	Die angeforderte ASCII-Kommunikation wird von der aktuellen Kanalkonfiguration nicht unterstützt.
10	Der „Cancel"-Befehl wurde gesetzt; Ausführung der Anweisung wird gestoppt. Keine Aktion erforderlich.
11	Die angeforderte Länge für die Zeichenkette ist entweder ungültig, eine negative Zahl, größer als 82 oder 0. Gilt für die Funktionsblöcke ARD und ARL.
13	Die angeforderte (.LEN) im Steuerblock ist eine negative Zahl oder ein Wert, der die zusammen mit der Quellzeichenfolge gespeicherte Zeichenfolge übersteigt. Gilt für die Funktionsblöcke AWA und AWT.
14	Der ACL-Funktionsblock wurde abgebrochen.
16	Serielle Anschlüsse unterstützen keine RTS- oder CTS-Steuerleitungen.

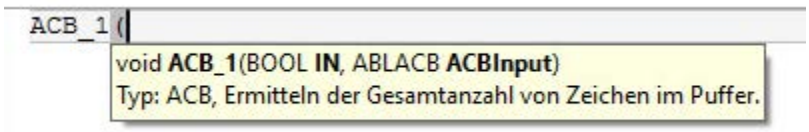
Beispiel für ACB-Funktionsblockdiagramm



Beispiel für ACB-Kontaktplandiagramm



Beispiel für strukturierten ACB-Text




```
1  ACB_1(in, input);
2  output := ACB_1.Q;
3  number := ACB_1.Characters;
4  error := ACB_1.Error;
5  ID := ACB_1.ErrorID;
```

ARD (ASCII lesen)

Liest ASCII-Zeichen aus dem Eingabepuffer und speichert sie in einer Zeichenfolge.

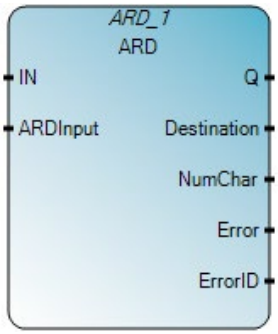
Details zum Betrieb:

- Die ARD-Anweisungen werden ausgeführt, bis alle Zeichen im ASCII-Puffer empfangen wurden. Wenn ein anderer ASCII-Befehl auf seite 103 ausgeführt wird, wird er in die Warteschleife gestellt, bis die ARD abgeschlossen wurde.
- Um den ARD-Befehl abzubrechen, führen Sie einen ACL-Befehl aus.
- Um zu verhindern, dass die ARD-Anweisung die ASCII-Warteschlange verzögert, während sie auf die erforderliche Anzahl von Zeichen wartet, verwenden Sie die Ergebnisse einer ACB-Anweisung, um die ARD-Anweisung auszulösen.
- Der Status der Anweisung kann aus dem Steuer-Bit der Anweisungsinstanz extrahiert werden (z. B. ARD_1.controlbit). Dies zeigt an, ob die Anweisung die ASCII-Anweisungsschlange blockiert und auf weitere Zeichen wartet:
  - 7. Bit = Anweisung ist aktiviert.
  - 6. Bit = Anweisung befindet sich in der Warteschlange.

- 5. Bit = Anweisung ist fertig.
- 3. Bit = Anweisung enthält einen Fehler.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangszustand von Strompfad. TRUE - bei erkannter Anstiegflanke wird der Anweisungsblock gestartet, vorausgesetzt, dass die letzte Operation abgeschlossen ist. FALSE - der Anweisungsblock befindet sich im Leerlauf.
ARDInput	Eingang	ARDARL auf seite 122	Liest Zeichen aus dem Puffer. Der Höchstwert sind 82 Zeichen. Verwenden Sie den Datentyp „ARDARL “, um die Kanal-, Länge- und Abbruchparameter für ARDInput zu definieren.
Q	Ausgang	BOOL	Gibt an, ob das Lesen des Puffers läuft oder abgeschlossen ist. Die Ausgaben dieser Anweisung werden asynchron vom Programm-Scan aktualisiert. Die Anweisung kann von Ausgabe Q nicht neu ausgelöst werden, da IN von der Kante ausgelöst wird. TRUE - der Funktionsblock ist abgeschlossen. FALSE - der Funktionsblock ist nicht abgeschlossen.
Destination	Ausgang	ASCIILOCADDR	Das Stringelement, in dem die Zeichen gespeichert werden.
NumChar	Ausgang	UINT	Die Anzahl von Zeichen.
Error	Ausgang	BOOL	Zeigt das Vorhandensein eine Fehlerbedingung an. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler für diese Anweisung sind in ABL-Fehlercodes definiert.

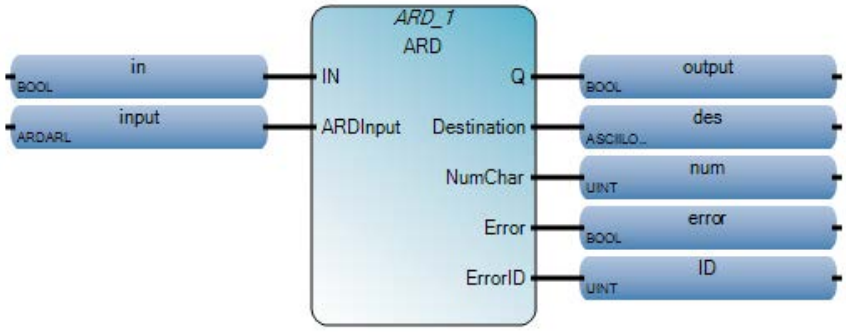
ABL-Fehlercodes

Verwenden Sie dieser Tabelle um die ABL-Fehlercodes und -Beschreibungen zu bestimmen.

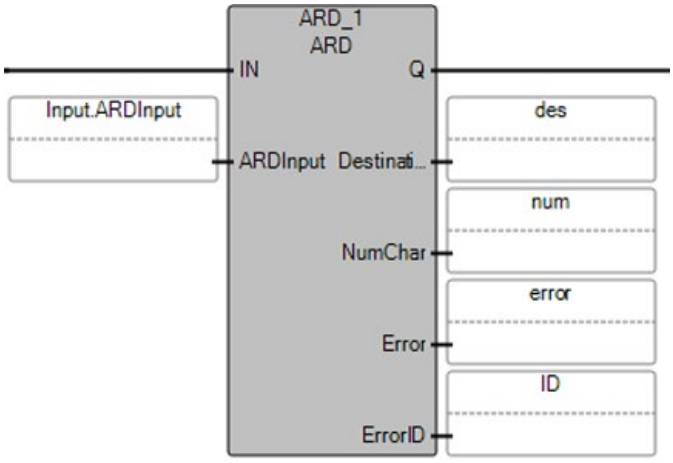
Fehlercode	Fehlerbeschreibung
03	Übertragung kann nicht abgeschlossen werden, da das Clear-to-Send-Signal abgebrochen ist.
06	Unzulässiger Parameter erkannt.

Fehlercode	Fehlerbeschreibung
07	Senden oder Empfangen von ASCII kann nicht abgeschlossen werden, da die Kanalkonfiguration über das Dialogfeld zur Kanalkonfiguration ausgeschaltet wurde.
08	ASCII-Schreibvorgang kann nicht abgeschlossen werden, da bereits eine ASCII-Übertragung durchgeführt wird.
09	Die angeforderte ASCII-Kommunikation wird von der aktuellen Kanalkonfiguration nicht unterstützt.
10	Der „Cancel"-Befehl wurde gesetzt; Ausführung der Anweisung wird gestoppt. Keine Aktion erforderlich.
11	Die angeforderte Länge für die Zeichenkette ist entweder ungültig, eine negative Zahl, größer als 82 oder 0. Gilt für die Funktionsblöcke ARD und ARL.
13	Die angeforderte (.LEN) im Steuerblock ist eine negative Zahl oder ein Wert, der die zusammen mit der Quellzeichenfolge gespeicherte Zeichenfolge übersteigt. Gilt für die Funktionsblöcke AWA und AWT.
14	Der ACL-Funktionsblock wurde abgebrochen.
16	Serielle Anschlüsse unterstützen keine RTS- oder CTS-Steuerleitungen.

Beispiel für ARD-Funktionsblockdiagramm



Beispiel für ARD-Kontaktplandiagramm



Beispiel für strukturierten ARD-Text

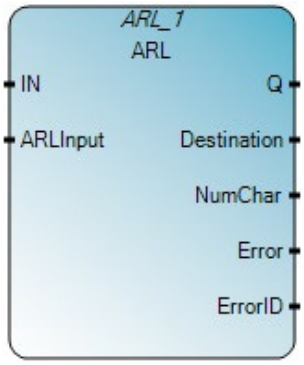
```
ARD_1(in, input);
output := ARD_1.Q;
des := ARD_1.Destination;
num := ARD_1.NumChar;
error := ARD_1.Error;
ID := ARD_1.ErrorID;
```

ARL (ASCII-Zeile lesen)

Liest eine Zeile von ASCII auf seite 103-Zeichen aus dem Puffer (bis einschließlich der Endzeichen) und speichert sie in einer Zeichenfolge.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangszustand von Strompfad. TRUE - bei erkannter Anstiegflanke wird der Anweisungsblock gestartet, vorausgesetzt, dass die letzte Operation abgeschlossen ist. FALSE - der Anweisungsblock befindet sich im Leerlauf.
ARLInput	Eingang	ARDARL auf seite 122	Liest eine Zeile von ASCII-Zeichen aus dem Puffer. Der Höchstwert sind 82 Zeichen. Verwenden Sie den Datentyp „ARDARL", um die Kanal-, Längen- und Abbruchparameter für ARDLInput zu definieren.
Q	Ausgang	BOOL	Gibt an, ob das Lesen einer Zeile aus dem Eingabepuffer läuft oder abgeschlossen ist. Die Ausgaben dieser Anweisung werden asynchron vom Programm-Scan aktualisiert. Die Anweisung kann von Ausgabe Q nicht neu ausgelöst werden, da IN von der Kante ausgelöst wird. TRUE - der Funktionsblock ist abgeschlossen. FALSE - der Funktionsblock ist nicht abgeschlossen.
Destination	Ausgang	ASCIILOCADDR	Das Stringelement, in dem die Zeichen gespeichert werden.
NumChar	Ausgang	UINT	Die Anzahl der Zeichen in der Zeile, einschließlich des Abschlusszeichens.

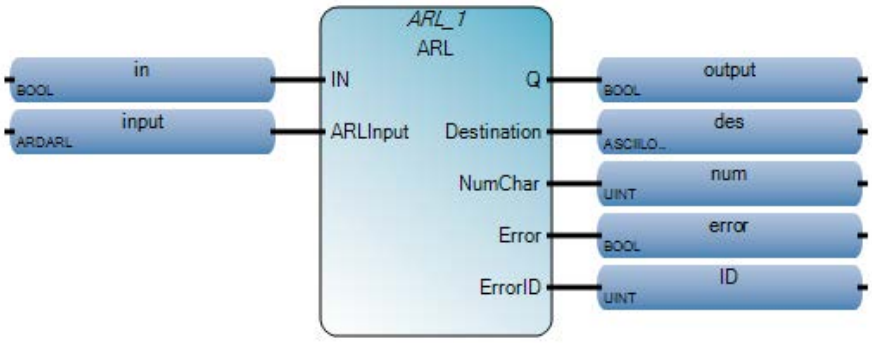
Parameter	Parametertyp	Datentyp	Beschreibung
Error	Ausgang	BOOL	Zeigt das Vorhandensein eine Fehlerbedingung an. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler für diese Anweisung sind in ABL-Fehlercodes definiert.

ABL-Fehlercodes

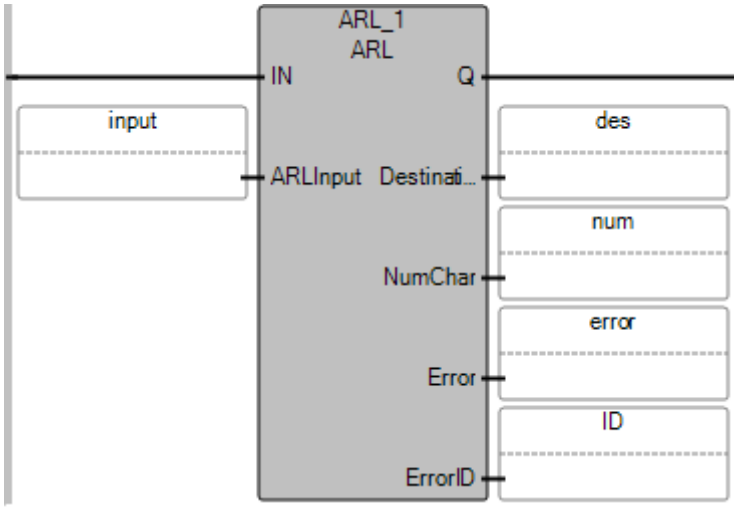
Verwenden Sie dieser Tabelle um die ABL-Fehlercodes und -Beschreibungen zu bestimmen.

Fehlercode	Fehlerbeschreibung
03	Übertragung kann nicht abgeschlossen werden, da das Clear-to-Send-Signal abgebrochen ist.
06	Unzulässiger Parameter erkannt.
07	Senden oder Empfangen von ASCII kann nicht abgeschlossen werden, da die Kanalkonfiguration über das Dialogfeld zur Kanalkonfiguration ausgeschaltet wurde.
08	ASCII-Schreibvorgang kann nicht abgeschlossen werden, da bereits eine ASCII-Übertragung durchgeführt wird.
09	Die angeforderte ASCII-Kommunikation wird von der aktuellen Kanalkonfiguration nicht unterstützt.
10	Der „Cancel"-Befehl wurde gesetzt; Ausführung der Anweisung wird gestoppt. Keine Aktion erforderlich.
11	Die angeforderte Länge für die Zeichenkette ist entweder ungültig, eine negative Zahl, größer als 82 oder 0. Gilt für die Funktionsblöcke ARD und ARL.
13	Die angeforderte (.LEN) im Steuerblock ist eine negative Zahl oder ein Wert, der die zusammen mit der Quellzeichenfolge gespeicherte Zeichenfolge übersteigt. Gilt für die Funktionsblöcke AWA und AWT.
14	Der ACL-Funktionsblock wurde abgebrochen.
16	Serielle Anschlüsse unterstützen keine RTS- oder CTS-Steuerleitungen.

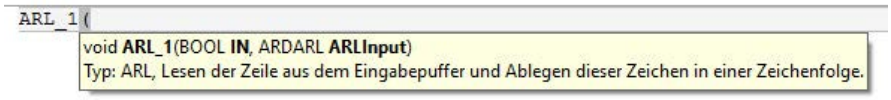
Beispiel für ARL-Funktionsblockdiagramm



Beispiel für den Kontaktplan von ARL



Beispiel für den Strukturierten Text von ARL



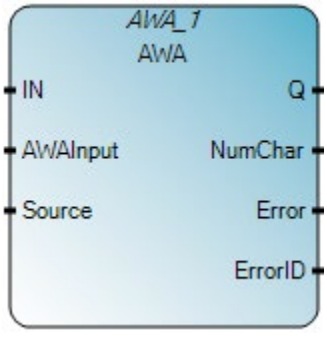

```
1  ARL_1(in, input);
2  output := ARL_1.Q;
3  des := ARL_1.Destination;
4  num := ARL_1.NumChar;
5  error := ARL_1.Error;
6  ID := ARL_1.ErrorID;
```

AWA (ASCII-Schreibanhang)

Schreibt eine Zeichenfolge mit zwei angefügten (benutzerkonfigurierten) Zeichen auf ein externes Gerät.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

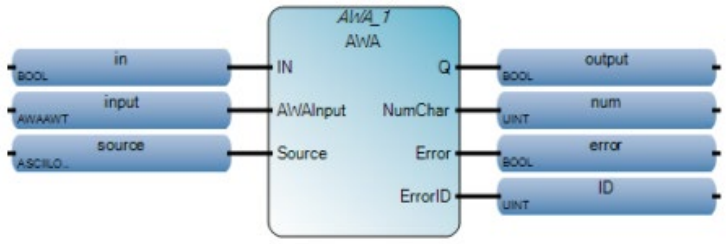
Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangszustand von Strompfad. TRUE - bei erkannter Anstiegflanke wird der Anweisungsblock gestartet, vorausgesetzt, dass die letzte Operation abgeschlossen ist. FALSE - der Anweisungsblock befindet sich im Leerlauf.
AWAInput	Eingang	AWAAWT	Der Kanal und die Anzahl (Länge) an Zeichen, die in den Puffer geschrieben werden. Das Maximum liegt bei 82. Mit dem Datentyp AWAAWT auf seite 122 können Sie die Parameter „Kanal", „Länge" und „Abbrechen" für AWAInput definieren.
Source	Eingang	ASCIILOCADDR	Die als Zeichenarray ausgegebene Quellzeichenfolge, entweder durch die ARD- oder die ARL-Anweisung.
Q	Ausgang	BOOL	Zeigt an, wenn der Schreibvorgang läuft oder abgeschlossen ist. Die Ausgaben dieser Anweisung werden asynchron vom Programm-Scan aktualisiert. Die Anweisung kann von Ausgabe Q nicht neu ausgelöst werden, da IN von der Kante ausgelöst wird. TRUE - der Funktionsblock ist abgeschlossen. FALSE - der Funktionsblock ist nicht abgeschlossen.
NumChar	Ausgang	UINT	Die Anzahl von Zeichen. „NumChar" kann die zur Übertragung angeforderte Länge („Length") unterschreiten, wenn die Länge der Quellzeichenfolge kürzer ist als die angeforderte Länge. Wird aktualisiert, wenn die Übertragung abgeschlossen und Q = TRUE ist.
Error	Ausgang	BOOL	Zeigt das Vorhandensein eine Fehlerbedingung an. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler für diese Anweisung sind in ABL-Fehlercodes definiert.

ABL-Fehlercodes

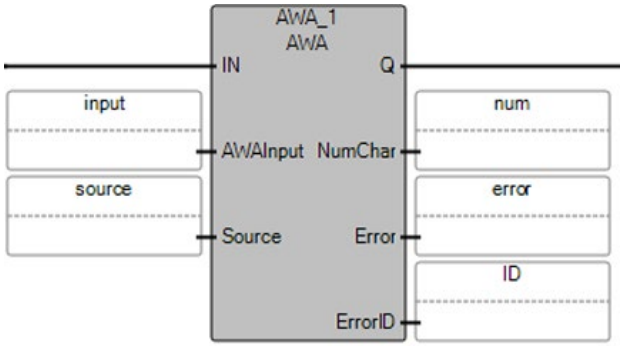
Verwenden Sie dieser Tabelle um die ABL-Fehlercodes und -Beschreibungen zu bestimmen.

Fehlercode	Fehlerbeschreibung
03	Übertragung kann nicht abgeschlossen werden, da das Clear-to-Send-Signal abgebrochen ist.
06	Unzulässiger Parameter erkannt.
07	Senden oder Empfangen von ASCII kann nicht abgeschlossen werden, da die Kanalkonfiguration über das Dialogfeld zur Kanalkonfiguration ausgeschaltet wurde.
08	ASCII-Schreibvorgang kann nicht abgeschlossen werden, da bereits eine ASCII-Übertragung durchgeführt wird.
09	Die angeforderte ASCII-Kommunikation wird von der aktuellen Kanalkonfiguration nicht unterstützt.
10	Der „Cancel"-Befehl wurde gesetzt; Ausführung der Anweisung wird gestoppt. Keine Aktion erforderlich.
11	Die angeforderte Länge für die Zeichenkette ist entweder ungültig, eine negative Zahl, größer als 82 oder 0. Gilt für die Funktionsblöcke ARD und ARL.
13	Die angeforderte (.LEN) im Steuerblock ist eine negative Zahl oder ein Wert, der die zusammen mit der Quellzeichenfolge gespeicherte Zeichenfolge übersteigt. Gilt für die Funktionsblöcke AWA und AWT.
14	Der ACL-Funktionsblock wurde abgebrochen.
16	Serielle Anschlüsse unterstützen keine RTS- oder CTS-Steuerleitungen.

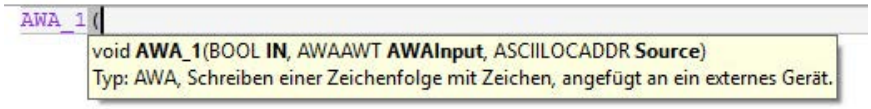
Beispiel für AWA-Funktionsblockdiagramm



Beispiel für AWA-Kontaktplandiagramm



Beispiel für strukturierten AWA-Text



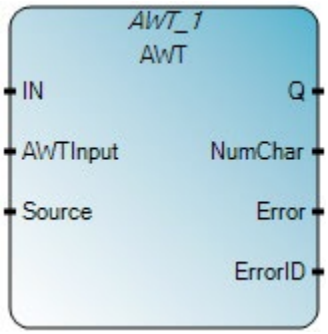

```
1  AWA_1(in, input, source);
2  output := AWA_1.Q;
3  num := AWA_1.NumChar;
4  error := AWA_1.Error;
5  ID := AWA_1.ErrorID;
```

AWT (ASCII-Schreibvorgang)

Schreibt ASCII-Zeichen aus einer Quellzeichenfolge in ein externes Gerät.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangszustand von Strompfad. TRUE - bei erkannter Anstiegflanke wird der Anweisungsblock gestartet, vorausgesetzt, dass die letzte Operation abgeschlossen ist. FALSE - der Anweisungsblock befindet sich im Leerlauf.
AWTInput	Eingang	AWAAWT	Der Kanal und die Anzahl (Länge) an Zeichen, die in den Puffer geschrieben werden. Das Maximum liegt bei 82. Mit dem Datentyp AWAAWT auf seite 122 können Sie die Parameter „Kanal", „Länge" und „Abbrechen" für AWTInput definieren.
Source	Eingang	ASCIILOCADDR	Die als Zeichenarray ausgegebene Quellzeichenfolge, entweder durch die ARD- oder die ARL-Anweisung.
Q	Ausgang	BOOL	Zeigt an, wenn der Schreibvorgang läuft oder abgeschlossen ist. Die Ausgaben dieser Anweisung werden asynchron vom Programm-Scan aktualisiert. Die Anweisung kann von Ausgabe Q nicht neu ausgelöst werden, da IN von der Kante ausgelöst wird. TRUE - der Funktionsblock ist abgeschlossen. FALSE - der Funktionsblock ist nicht abgeschlossen.
NumChar	Ausgang	UINT	Die Anzahl von Zeichen. „NumChar" kann die zur Übertragung angeforderte Länge („Length") unterschreiten, wenn die Länge der Quellzeichenfolge kürzer ist als die angeforderte Länge. Wird aktualisiert, wenn die Übertragung abgeschlossen und Q = TRUE ist.
Error	Ausgang	BOOL	Zeigt das Vorhandensein eine Fehlerbedingung an. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler für diese Anweisung sind in ABL-Fehlercodes definiert.

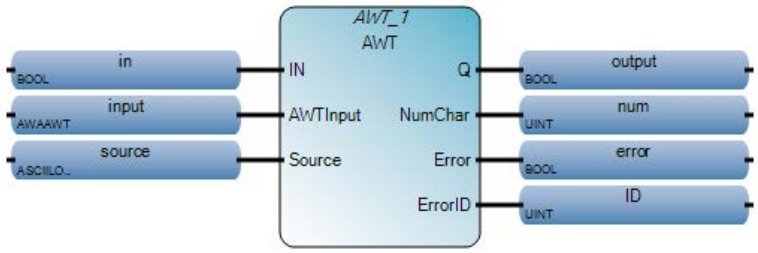
ABL-Fehlercodes

Verwenden Sie dieser Tabelle um die ABL-Fehlercodes und -Beschreibungen zu bestimmen.

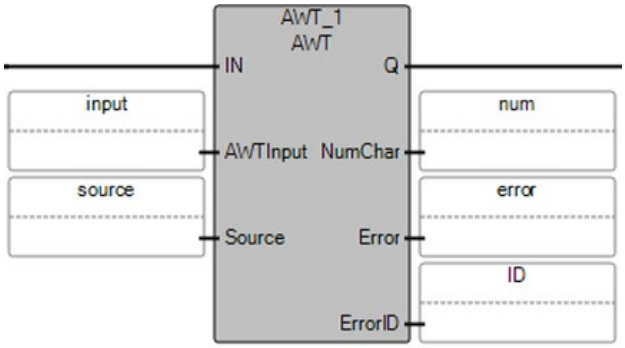
Fehlercode	Fehlerbeschreibung
03	Übertragung kann nicht abgeschlossen werden, da das Clear-to-Send-Signal abgebrochen ist.
06	Unzulässiger Parameter erkannt.
07	Senden oder Empfangen von ASCII kann nicht abgeschlossen werden, da die Kanalkonfiguration über das Dialogfeld zur Kanalkonfiguration ausgeschaltet wurde.

Fehlercode	Fehlerbeschreibung
08	ASCII-Schreibvorgang kann nicht abgeschlossen werden, da bereits eine ASCII-Übertragung durchgeführt wird.
09	Die angeforderte ASCII-Kommunikation wird von der aktuellen Kanalkonfiguration nicht unterstützt.
10	Der „Cancel"-Befehl wurde gesetzt; Ausführung der Anweisung wird gestoppt. Keine Aktion erforderlich.
11	Die angeforderte Länge für die Zeichenkette ist entweder ungültig, eine negative Zahl, größer als 82 oder 0. Gilt für die Funktionsblöcke ARD und ARL.
13	Die angeforderte (.LEN) im Steuerblock ist eine negative Zahl oder ein Wert, der die zusammen mit der Quellzeichenfolge gespeicherte Zeichenfolge übersteigt. Gilt für die Funktionsblöcke AWA und AWT.
14	Der ACL-Funktionsblock wurde abgebrochen.
16	Serielle Anschlüsse unterstützen keine RTS- oder CTS-Steuerleitungen.

Beispiel für AWT-Funktionsblockdiagramm



Beispiel für AWT-Kontaktplandiagramm



Beispiel für strukturierten AWT-Text

```
1  AWT_1(in, input, source);
2  output := AWT_1.Q;
3  num := AWT_1.NumChar;
4  error := AWT_1.Error;
5  ID := AWT_1.ErrorID;
```

ASCII-Parameterdetails

In den folgenden Themen werden weitere Details zu ASCII-Parametern und strukturierten Datentypen behandelt.

- Datentyp ABLACB
- Datentyp ACL
- Datentyp AHL ChannelSts
- Datentyp AHLI
- Datentyp ARDARL
- Datentyp AWAAWT

Datentyp ABLACB

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „ABLACB“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Channel	UINT	Nummer des seriellen Anschlusses: • 2 für den integrierten seriellen Anschluss oder • 5-9 für an den Schächten 1 bis 5 installierte Plug-Ins: • 5 für Schacht 1 • 6 für Schacht 2 • 7 für Schacht 3 • 8 für Schacht 4 • 9 für Schacht 5
TriggerType	USINT	Steht für eines der Folgenden: • 0: Meldung einmal ausgelöst (wenn IN von „False“ auf „True“ wechselt) • 1: Meldung kontinuierlich ausgelöst, wenn IN = True ist • Sonstiger Wert: Reserviert
Cancel	BOOL	Wenn diese Eingabe auf TRUE gesetzt ist, wird dieser Funktionsblock nicht ausgeführt.

Datentyp ACL

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „ABL“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Channel	UINT	Nummer des seriellen Anschlusses: • 2 für den integrierten seriellen Anschluss oder • 5-9 für an den Schächten 1 bis 5 installierte Plug-Ins: • 5 für Schacht 1 • 6 für Schacht 2 • 7 für Schacht 3 • 8 für Schacht 4 • 9 für Schacht 5
RXBuffer	BOOL	Bei TRUE wird der Empfangspuffer gelöscht und die Funktionsblöcke zum Empfang von ASCII (ARL und ARD) werden aus der ASCII-Warteschlange entfernt.
TXBuffer	BOOL	Bei TRUE wird der Übertragungspuffer geleert und die Funktionsblöcke zur Übertragung von ASCII (AWA und AWT) werden aus der ASCII-Warteschlange entfernt.

Datentyp AHL ChannelSts

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „AHL ChannelSts“ zu bestimmen.

Parameter	Datentyp	Beschreibung
DTRstatus	UINT	Wird für das DTR-Signal verwendet (reserviert)

Parameter	Datentyp	Beschreibung
DCDstatus	UINT	Wird für das DCD-Signal verwendet (Bit 3 des Wortes) 1 steht für aktiv
DSRstatus	UINT	Wird für das DSR-Signal verwendet (reserviert)
RTSstatus	UINT	Wird für das RTS-Signal verwendet (Bit 1 des Wortes) 1 steht für aktiv
CTSstatus	UINT	Wird für das CTS-Signal verwendet (Bit 0 des Wortes) 1 steht für aktiv

Datentyp AHLI

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „AHL“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Channel	UINT	Nummer des seriellen Anschlusses: • 2 für den integrierten seriellen Anschluss oder • 5-9 für an den Schächten 1 bis 5 installierte Plug-Ins: • 5 für Schacht 1 • 6 für Schacht 2 • 7 für Schacht 3 • 8 für Schacht 4 • 9 für Schacht 5
ClrRts	BOOL	Wird zum Zurücksetzen der RTS-Steuerleitung verwendet.
SetRts	BOOL	Wird zum Festlegen der RTS-Steuerleitung verwendet.
Cancel	BOOL	Wenn diese Eingabe auf TRUE gesetzt ist, wird dieser Funktionsblock nicht ausgeführt.

Datentyp ARDARL

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „ARDARL“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Channel	UINT	Nummer des seriellen Anschlusses: • 2 für den integrierten seriellen Anschluss oder • 5-9 für an den Schächten 1 bis 5 installierte Plug-Ins: • 5 für Schacht 1 • 6 für Schacht 2 • 7 für Schacht 3 • 8 für Schacht 4 • 9 für Schacht 5
Length	UINT	Die Anzahl der aus dem Puffer auszulesenden Zeichen (maximal 82).
Cancel	BOOL	Wenn diese Eingabe auf TRUE gesetzt ist, wird dieser Funktionsblock nicht ausgeführt. Bei bereits laufender Ausführung wird der Betrieb abgebrochen.

Datentyp AWAAWT

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „AWAAWT“ zu bestimmen.

Parameter	Datentyp	Beschreibung

Parameter	Datentyp	Beschreibung
Channel	UINT	Nummer des seriellen Anschlusses: • 2 für den integrierten seriellen Anschluss oder • 5-9 für an den Schächten 1 bis 5 installierte Plug-Ins: • 5 für Schacht 1 • 6 für Schacht 2 • 7 für Schacht 3 • 8 für Schacht 4 • 9 für Schacht 5
Length	UINT	Die Anzahl der in den Puffer zu schreibenden Zeichen (maximal 82). Wenn Sie die Länge („Length") auf 0 setzen, sendet „AWA" 0 Byte an Benutzerdaten und 2 Byte angehängter Zeichen an den Puffer.
Cancel	BOOL	Wenn TRUE, wird der Funktionsblock nicht ausgeführt. Bei bereits laufender Ausführung wird der Betrieb abgebrochen.

Binäre Anweisungen

Verwenden Sie die binären Anweisungen, um mathematische Operationen durchzuführen.

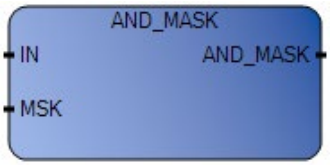
Operator	Beschreibung
AND_MASK auf seite 125	Führt eine bitweise AND-Operation zwischen zwei ganzzahligen Werten aus.
NOT_MASK auf seite 133	Ganzzahlige bitweise Negationsmaske; damit wird der Wert eines Parameters umgekehrt.
BSL auf seite 127	Verschiebt ein Bit in einem Array-Element nach links.
BSR auf seite 130	Verschiebt ein Bit in einem Array-Element nach rechts.
OR_MASK auf seite 134	Ganzzahlige bitweise OR-Maske, zur Aktivierung von Bits.
ROL auf seite 135	Dreht bei 32-Bit-Ganzzahlen die Ganzzahlenbits nach links.
ROR auf seite 137	Dreht bei 32-Bit-Ganzzahlen die Ganzzahlenbits nach links.
SHL auf seite 139	Verschiebt 32-Bit-Ganzzahlen nach links und setzt 0 in das niederwertigste Bit.
SHR auf seite 141	Verschiebt bei 32-Bit-Ganzzahlen Ganzzahlen nach rechts und setzt 0 in das höchstwertige Bit.
XOR_MASK auf seite 142	Ganzzahlige exklusive bitweise OR-Maske gibt invertierte Bitwerte zurück.

AND_MASK (AND-Maske)

Führt eine bitweise AND-Operation zwischen zwei ganzzahligen Werten aus.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

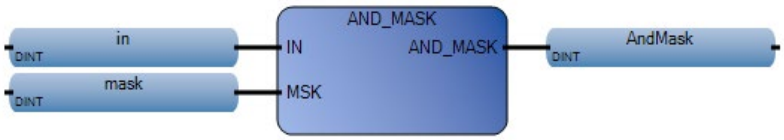


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

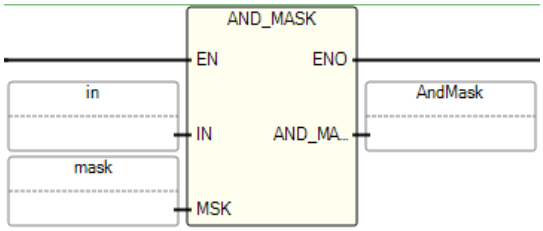
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Ausführung der Berechnung mit der ganzzahligen bitweisen AND-Maske. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	DINT	Muss als ganze Zahl vorliegen.
MSK	Eingang	DINT	Muss als ganze Zahl vorliegen.
AND_MASK	Ausgang	DINT	Bitweises logisches AND zwischen IN und MSK.

Parameter	Parametertyp	Datentyp	Beschreibung
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für ein AND_MASK-Funktionsblockdiagramm



Beispiel für einen AND_MASK-Kontaktplan



Beispiel für strukturierten AND_MASK-Text

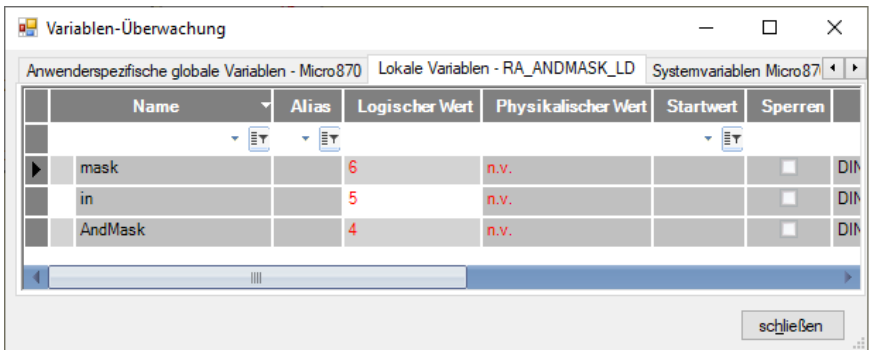
```
1  in := 5;
2  mask := 6;
3  AndMask := AND_MASK(in, mask);
```

(* ST-Äquivalenz: *)

Parität := AND_MASK (xvalue, 1); (* 1 wenn xvalue ungerade ist *)

Ergebnis := AND_MASK (16#abc, 16#fof); (* gleich 16#aoc *)

Ergebnisse



BSL (Bit-Verschiebung links)

Verschiebt ein Bit in einem Array-Element nach links.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Details zum Betrieb:

Die BSL-Anweisung ist ein sofortiger Prozess für FALSE-zu-TRUE-Strompfadwechsel und synchrone Ausgabe-Updates. Wenn der Ausführungsparameter TRUE ist, wird das Bit ganz links (adressiert durch Src + SrcOffset und Length) in das Unload-Bit kopiert und alle Bits im Array oder Nicht-Array werden um ein Bit nach links verschoben. Length und 16-Bit-Begrenzung werden berücksichtigt, abgesehen von BOOL-Datentypen. Das externe Bit wird anschließend zu Bit 0 (Src + SrcOffset) des ersten Elements verschoben.

Setzen Sie für Übertragungsvorgänge die Position von BitAddr auf die letzte Bit-Position oder das Unload-Bit. Ein Beispiel zur Nutzung der BSL-Anweisung ist die Nachverfolgung von Flaschen aus einer Abfüllanlage, dabei stellt jedes Bit eine Flasche dar.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Anweisung aktivieren. TRUE - wird eine Anstiegflanke erkannt, wird das Bit eine Position nach links verschoben. • Überprüfen Sie zunächst die Fehlerbedingungen. • Wenn Length = 0, wird das externe Bit in das Unload-Bit verschoben. Bei Scr erfolgt keine Bit-Verschiebung. Error- und ErrorID-Bits werden zurückgesetzt. Das Done-Bit wird gesetzt. • Wenn Length > 0 und Length ≤ 2048, werden die Error- und ErrorID-Bits zurückgesetzt. Nachdem die Bit-Verschiebung abgeschlossen ist, wird das Done-Bit gesetzt. • Wenn Length > 0 und Length ≤ 2048, wird das Bit ganz links (adressiert durch Src + SrcOffset und Length) in das Unload-Bit kopiert und alle Bits, die Teil des Datenfelds oder Nicht-Datenfelds sind, werden um ein Bit nach links verschoben (bis zum Length-Bit und zur 16-Bit-Begrenzung außer für BOOL). Das externe Bit wird zu Bit 0 (Src + SrcOffset) des ersten Elements verschoben.

Parameter	Parametertyp	Datentyp	Beschreibung
			FALSE - wird keine Anstiegflanke erkannt, wird die BSL-Operation nicht aktiviert.
Scr	Eingang	ANY_ELEMENTARY	Die Adresse des zu verschiebenden Src (Bit). Unterstützte Datentypen: BOOL, DWORD, INT, UINT, WORD, DINT und UDINT. • Arrays: Scr wird auf eine variablenbasierte Adresse gesetzt, beispielsweise: Source1, Source1[0] oder Source1[1] . • Nicht-Arrays: Scr wird auf eine Variablenadresse wie Source1 gesetzt.
SrcOffset	Eingang	UINT	Wenn SrcOffset 0 ist, wird beim ersten Element gestartet. • Arrays: SrcOffset wird auf 0 gesetzt. Wenn der Wert auf „Source1[0]" oder „Source1[1]" gesetzt wird, tritt folgender Fehler auf: 'Versatz der Quelle überschreitet die Arraygröße.' • Nicht-Arrays: SrcOffset muss auf 0 gesetzt werden, andernfalls tritt folgender Fehler auf: 'Versatz der Quelle überschreitet die Arraygröße.'
BitAddr	Eingang	BOOL	Position des in die Quelle Src verschobenen Bits.

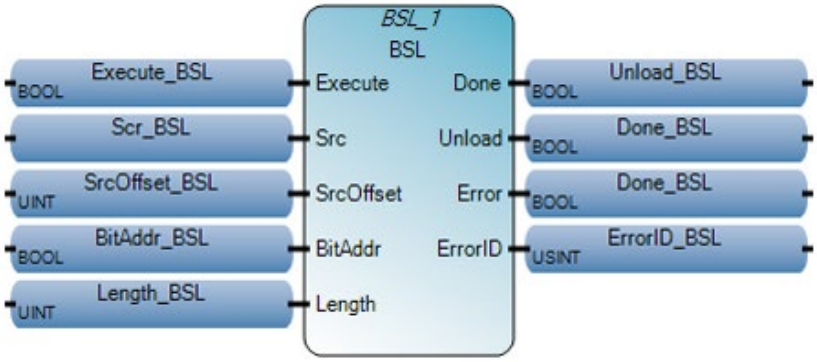
Length	Eingang	UINT	Length enthält die Anzahl von in Src zu verschiebenden Bits. Unterstützt Verschiebungen über Arrayelemente hinweg.
			• Für den Datentyp BOOL die Anzahl der im Array enthaltenen zu verschiebenden booleschen Datentypen.
			• Bei 16- und 32-Bit-Datentypen werden Bits in Vielfachen von 16 (z. B. 16, 32 und 64) verschoben. Wenn Length kein gerades Vielfaches von 16 ist, wird die Anzahl der verschobenen Bits an die nächste 16-Bit-Begrenzung gesendet.
			• Length basiert auf der Größe des Datentyps. Bei Längenüberschreitung des Bereichs tritt ein Fehler auf: 'Versatz der Quelle überschreitet die Arraygröße'. Length-Werte:
			  • BOOL: 1
			  • 16-Bit-Wort: 1-16
			  • 32-Bit-Wort: 1-32
			  • 64-Bit-Wort: 1-64

Done	Ausgang	BOOL	Wenn TRUE, wurde der Vorgang erfolgreich abgeschlossen. Wenn FALSE, ist bei dem Vorgang eine Fehlerbedingung aufgetreten.
Unload	Ausgang	BOOL	Aus der Src-Adresse verschobenes Bit.
Error	Ausgang	BOOL	Wenn ein Fehler auftritt, wird Error auf „True" gesetzt.
ErrorID	Ausgang	USINT	Wenn ein Fehler auftritt, enthält ErrorID den Fehlercode.

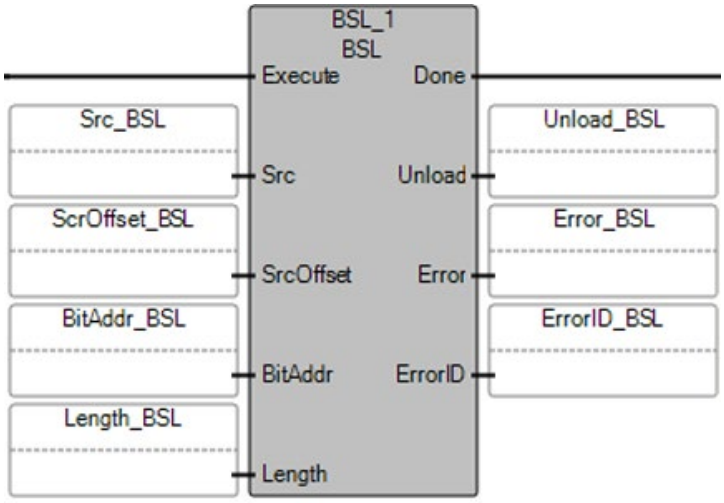
BSL-Fehlercodes

Fehlercode	Fehlerbeschreibung
01	Nicht unterstützte Dimension.
02	Datentyp nicht unterstützt.
03	Bit-Länge überschreitet 2048.
04	Quellenabstand überschreitet die Größe des Arrays.
05	Bit-Länge überschreitet die Größe des Arrays.
07	Ungültige Parameter.

Beispiel für BSL-Funktionsblockdiagramm



Beispiel für BSL-Kontaktplandiagramm



Beispiel für strukturierten BSL-Text

```
1  BSL_1(Execute,Src,SrcOffset,BitAddr,Length);
2  Done_BSL := BSL_1.Done;
3  Unload_BSL := BSL_1.Unload;
4  Error_BSL := BSL_1.Error;
5  ErrorID_BSL := BSL_1.ErrorID;
```

Ergebnisse

	Name	Alias	Logischer Wert	Physikalischer Wert	Startwert	Sperren	Datentyp	Dimension
+	BSL_1		...	...	...		BSL	
	Execute			n.v.			BOOL	
+	Src		...	...	...		BOOL	[1..4]
	SrcOffset		0	n.v.			UINT	
	BitAddr			n.v.			BOOL	
	Length		0	n.v.			UINT	
	Done_BSL			n.v.			BOOL	
	Upload_BSL			n.v.			BOOL	
	Error_BSL			n.v.			BOOL	
	ErrorID_BSL		0	n.v.			USINT	

BSR (Bit-Verschiebung rechts)

Verschiebt ein Bit in einem Array-Element nach rechts.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Details zum Betrieb:

Die BSR-Anweisung ist ein sofortiger Prozess für FALSE-zu-TRUE-Strompfadwechsel und synchrone Ausgabe-Updates. Wenn der Ausführungsparameter TRUE ist, wird das Bit ganz rechts (Bit 0 des durch Src + SrcOffset adressierten Elements) in das Unload-Bit kopiert und alle Bits im Array oder Nicht-Array werden um ein Bit nach rechts verschoben. Length und 16-Bit-Begrenzung werden berücksichtigt, abgesehen von BOOL-Datentypen. Das externe Bit wird anschließend zu Bit 0 (Src + SrcOffset) des ersten Elements verschoben.

Setzen Sie für Übertragungsvorgänge die Position von BitAddr auf die letzte Bit-Position oder das Unload-Bit. Ein Beispiel zur Nutzung der BSL-Anweisung ist die Nachverfolgung von Flaschen aus einer Abfüllanlage, dabei stellt jedes Bit eine Flasche dar.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Anweisung aktivieren. TRUE - wird eine Anstiegflanke erkannt, wird das Bit eine Position nach rechts verschoben. FALSE - wird keine Anstiegflanke erkannt, wird die BSR-Operation nicht aktiviert.
Scr	Eingang	ANY_ELEMENTARY	Die Adresse des zu verschiebenden Src (Bit). Unterstützte Datentypen: BOOL, DWORD, INT, UINT, WORD, DINT und UDINT. • Arrays: Scr wird auf eine variablenbasierte Adresse gesetzt, beispielsweise: Source1, Source1[0] oder Source1[1] . • Nicht-Arrays: Scr wird auf eine Variablenadresse wie Source1 gesetzt.
SrcOffset	Eingang	UINT	Wenn SrcOffset 0 ist, wird beim ersten Element gestartet. • Arrays: SrcOffset wird auf 0 gesetzt. Wenn der Wert auf „Source1[0]" oder „Source1[1]" gesetzt wird, tritt folgender Fehler auf: 'Versatz der Quelle überschreitet die Arraygröße.' • Nicht-Arrays: SrcOffset muss auf 0 gesetzt werden, andernfalls tritt folgender Fehler auf: 'Versatz der Quelle überschreitet die Arraygröße.'
BitAddr	Eingang	BOOL	Position des in die Quelle Src verschobenen Bits.

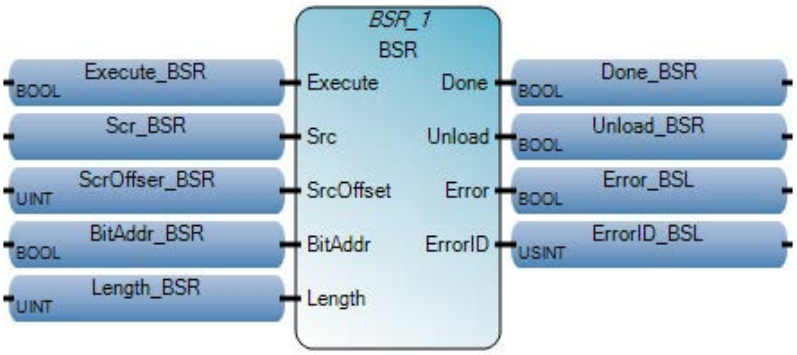
Length	Eingang	UINT	Length enthält die Anzahl von in Src zu verschiebenden Bits. Unterstützt Verschiebungen über Arrayelemente hinweg.
			• Für den Datentyp BOOL die Anzahl der im Array enthaltenen zu verschiebenden booleschen Datentypen.
			• Bei 16- und 32-Bit-Datentypen werden Bits in Vielfachen von 16 (z. B. 16, 32 und 64) verschoben. Wenn Length kein gerades Vielfaches von 16 ist, wird die Anzahl der verschobenen Bits an die nächste 16-Bit-Begrenzung gesendet.
			• Length basiert auf der Größe des Datentyps. Bei Längenüberschreitung des Bereichs tritt ein Fehler auf: 'Versatz der Quelle überschreitet die Arraygröße'. Length-Werte:
			  • BOOL: 1
			  • 16-Bit-Wort: 1-16
			  • 32-Bit-Wort: 1-32
			  • 64-Bit-Wort: 1-64

Done	Ausgang	BOOL	Wenn TRUE, wurde der Vorgang erfolgreich abgeschlossen. Wenn FALSE, ist bei dem Vorgang eine Fehlerbedingung aufgetreten.
Unload	Ausgang	BOOL	Aus der Src-Adresse verschobenes Bit.
Error	Ausgang	BOOL	Wenn ein Fehler auftritt, wird Error auf „True" gesetzt.
ErrorID	Ausgang	USINT	Wenn ein Fehler auftritt, enthält ErrorID den Fehlercode.

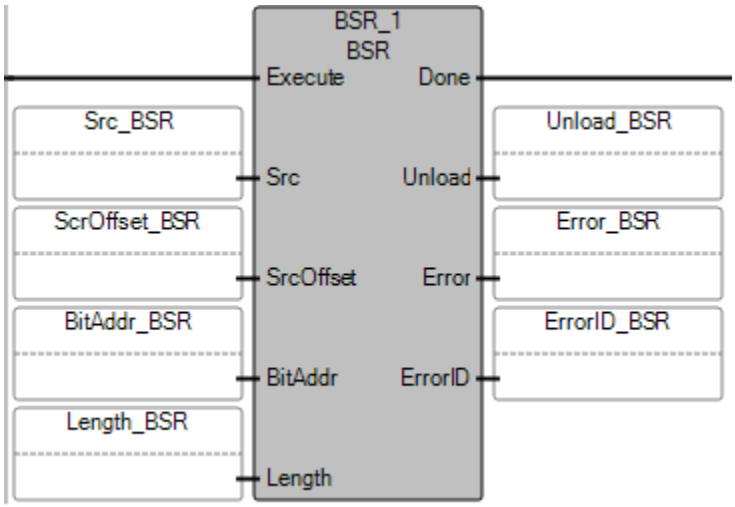
BSR-Fehlercodes

Fehlercode	Fehlerbeschreibung
01	Nicht unterstützte Dimension.
02	Datentyp nicht unterstützt.
03	Bit-Länge überschreitet 2048.
04	Quellenabstand überschreitet die Größe des Arrays.
05	Bit-Länge überschreitet die Größe des Arrays.
07	Ungültige Parameter.

Beispiel für BSR-Funktionsblockdiagramm



Beispiel für BSR-Kontaktplandiagramm



Beispiel für strukturierten BSR-Text

```
1  BSR_1(Execute,Scr,SrcOffset,BitAddr,Length);
2  Done_BSR := BSR_1.Done;
3  Unload_BSR := BSR_1.Unload;
4  Error_BSR := BSR_1.Error;
5  ErrorID_BSR := BSR_1.ErrorID;
```

Ergebnisse

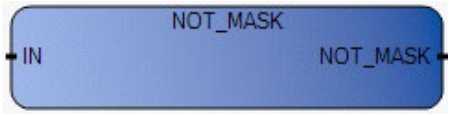
Name	Alias	Logischer Wert	Physikalischer Wert	Startwert	Sperren	Datentyp	Dimension
Prog1-VAR							
+ BSR_1		...	...	...		BSR	
Execute			n.v.			BOOL	
+ Scr		...	...	...		BOOL	[1..4]
SrcOffset		0	n.v.			UINT	
BitAddr			n.v.			BOOL	
Length		0	n.v.			UINT	
Done_BSR			n.v.			BOOL	
Unload_BSR			n.v.			BOOL	
Error_BSR			n.v.			BOOL	
ErrorID_BSR		0	n.v.			USINT	

NOT_MASK (Bit-zu-Bit-NOT-Maske)

Ganzzahlige bitweise Negationsmaske; damit wird der Wert eines Parameters umgekehrt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

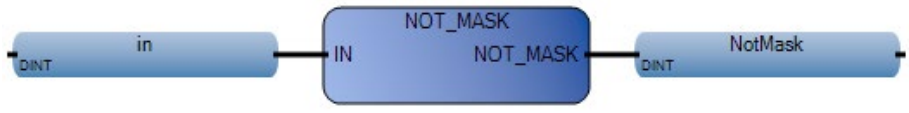
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



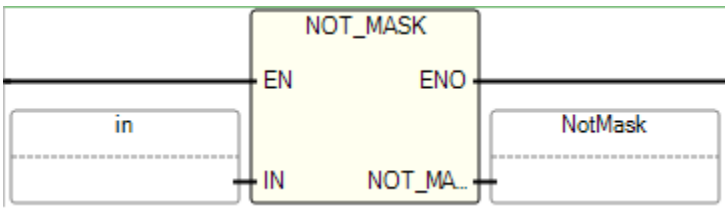
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführung der Berechnung mit der bitweisen Negationsmaske. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	DINT	Muss als ganze Zahl vorliegen.
NOT_MASK	Ausgang	DINT	Bitweise Negation an 32 Bits IN.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

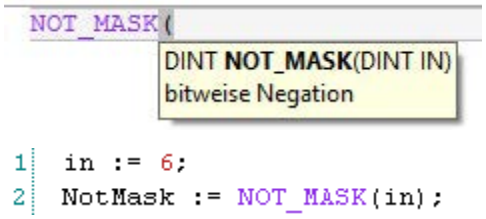
Diagrammbeispiel für Funktionsblock NOT_MASK



NOT_MASK-Kontaktplan-Beispiel



Beispiel für NOT_MASK in strukturiertem Text

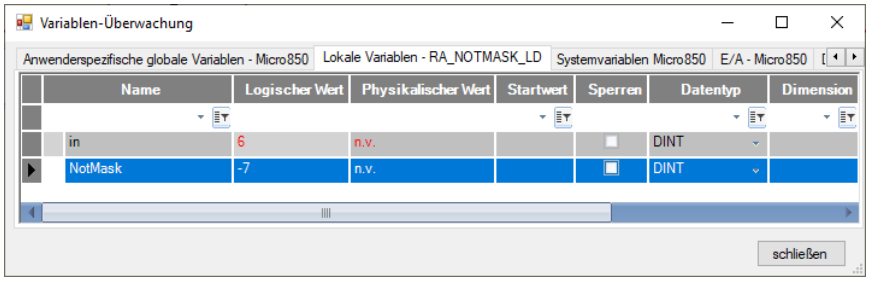


(*ST-Äquivalenz: *)

result := NOT_MASK (16#1234);

(* Ergebnis ist 16#FFFF_EDCB *)

Ergebnisse



OR_MASK (bitweise OR-Maske)

Ganzzahlige bitweise OR-Maske, zur Aktivierung von Bits.

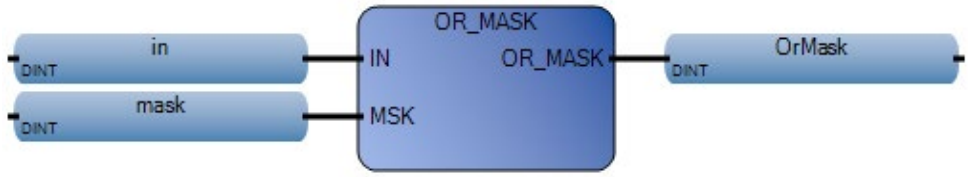
Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

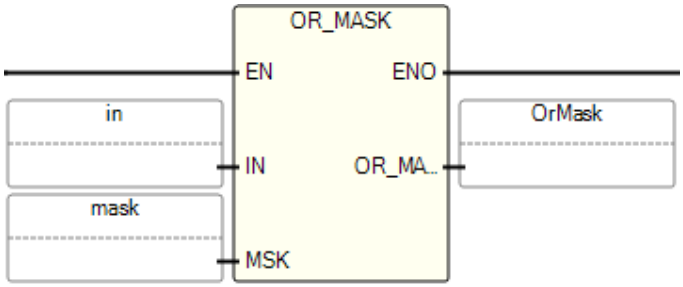
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführung der Berechnung mit der ganzzahligen bitweisen OR-Maske. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	DINT	Muss als ganze Zahl vorliegen.
MSK	Eingang	DINT	Muss als ganze Zahl vorliegen.
OR_MASK	Ausgang	DINT	Bitweises logisches OR zwischen IN und MSK.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

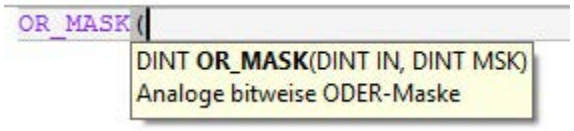
Beispiel für ein OR_MASK-Funktionsblockdiagramm



Beispiel für einen OR_MASK-Kontaktplan



Beispiele für strukturierten OR_MASK-Text



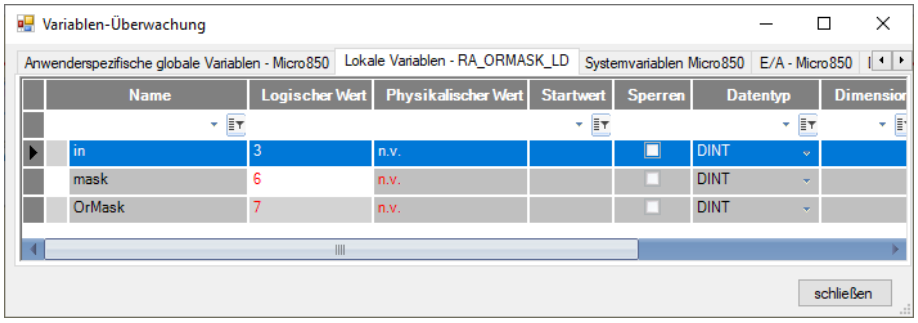

```
1  in := 3;
2  mask := 6;
3  OrMask := OR_MASK(in, mask);
```

(* ST-Äquivalenz: *)

parity := OR_MASK (xvalue, 1); (* ergibt immer einen ungeraden Wert *)

result := OR_MASK (16#abc, 16#fof); (* equals 16#fbf *)

Ergebnisse

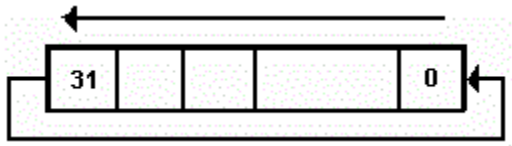


ROL (nach links drehen)

Dreht den Eingang vom Typ DINT kreisförmig um NbR-Bits nach links und füllt die Bits auf der rechten Seite mit den Bits, die gedreht werden.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

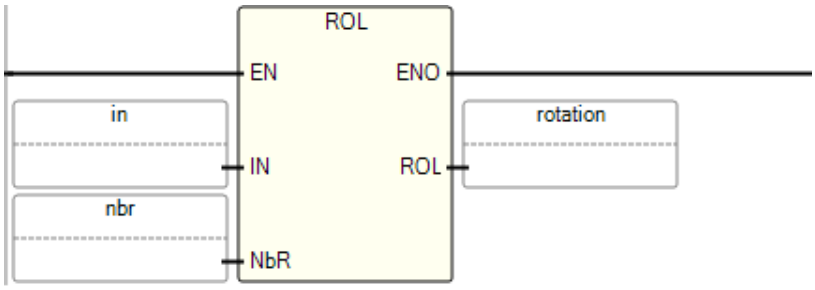


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

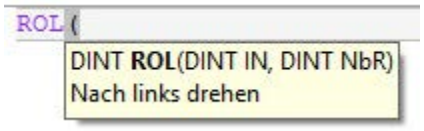
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Berechnung der Drehung der Bits einer Ganzzahl nach links. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	DINT	Ganzzahliger Wert.
NbR	Eingang	DINT	Anzahl der 1-Bit-Drehungen (im Bereich [1..31]).
ROL	Ausgang	DINT	Nach links gedrehter Wert. Wenn „NbR" <= 0, wird keine Änderung vorgenommen.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispieldiagramm mit einem ROL-Funktionsblock

Beispiel für Kontaktplandiagramm ROL



Beispiel für ROL in strukturiertem Text




```
1  in := 123;
2  nbr := 2;
3  rotation := ROL(in, nbr);
```

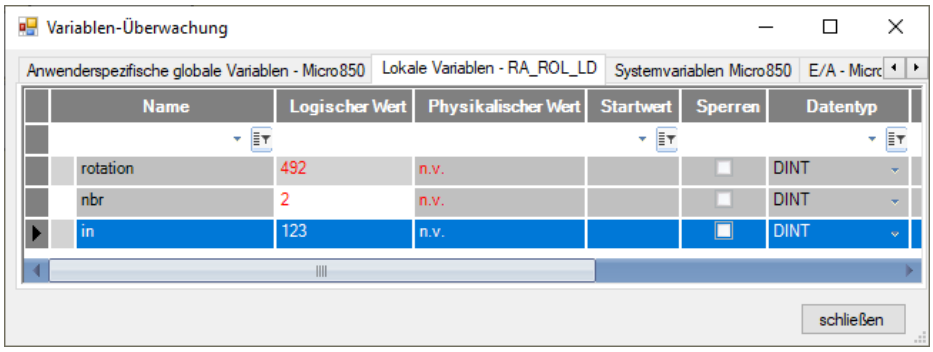
(* ST-Äquivalenz: *)

result := ROL (register, 1);

(* Register = 2#0100_1101_0011_0101 *)

(* Ergebnis = 2#1001_1010_0110_1010 *)

Ergebnisse

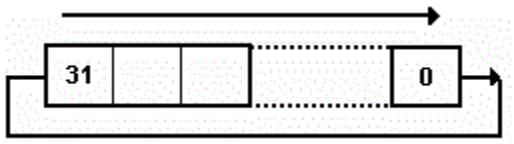


ROR (nach rechts drehen)

Dreht den Eingang vom Typ DINT kreisförmig um NbR-Bits nach rechts und füllt die Bits auf der linken Seite mit den Bits, die gedreht werden.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

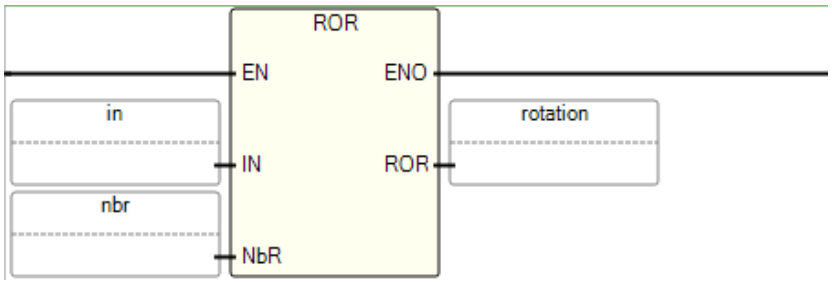


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

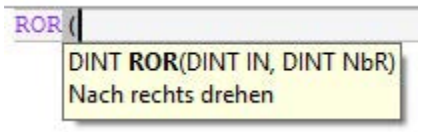
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Berechnung der Drehung der Bits einer Ganzzahl nach rechts. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	DINT	Beliebige Ganzzahl.
NbR	Eingang	DINT	Anzahl der 1-Bit-Drehungen (im Bereich [1..31]).
ROR	Ausgang	DINT	Nach rechts gedrehter Wert. Es wird keine Änderung vorgenommen, wenn „NbR" <= 0.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispieldiagramm mit einem ROR-Funktionsblock

Beispiel Kontaktplandiagramm ROR



Beispiel für ROR in strukturiertem Text




```
1  in := 123;
2  nbr := 2;
3  rotation := ROR(in, nbr);
```

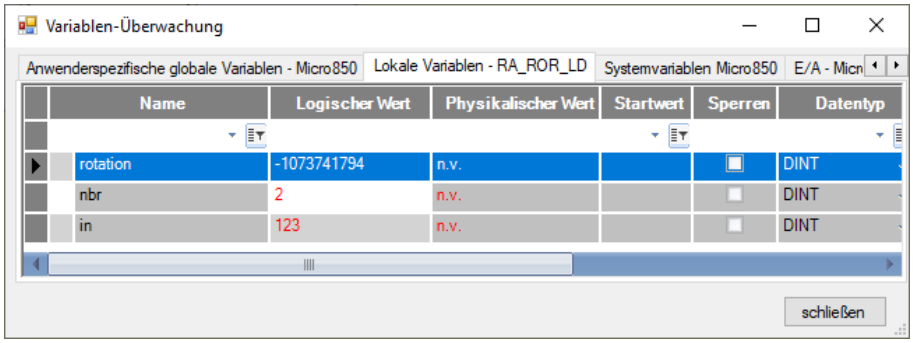
(* ST-Äquivalenz: *)

result := ROR (register, 1);

(* Register = 2#0100_1101_0011_0101 *)

(* Ergebnis = 2#1010_0110_1001_1010 *)

Ergebnisse

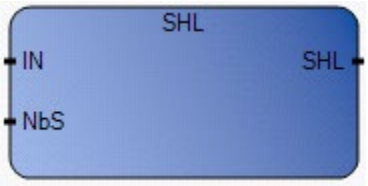


SHL (Verschiebung links)

Verschiebt 32-Bit-Ganzzahlen nach links und setzt 0 in das niederwertigste Bit.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

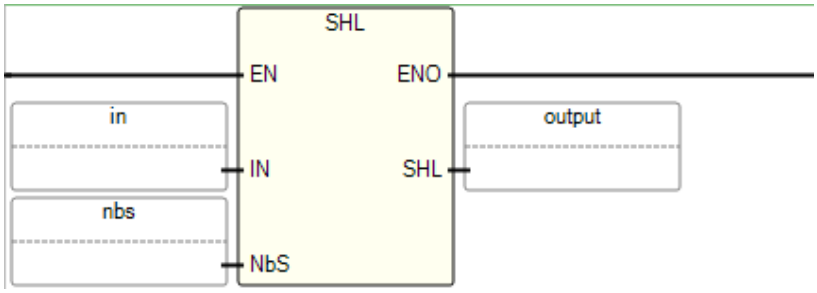


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Ganzzahlen nach lLinks zu verschieben. FALSE - Es findet keine Verschiebung von Ganzzahlen statt. Gilt für Kontaktplan-Programme.
IN	Eingang	DINT	Beliebige Ganzzahl.
NbS	Eingang	DINT	Anzahl der 1-Bit-Verschiebungen (im Bereich [1..31]).
SHL	Ausgang	DINT	Nach links verschobener Wert. Es wird keine Änderung vorgenommen, wenn NbS <= 0. Wenn ein Wert von 0 vorliegt, wird das niederwertigste Bit ersetzt.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für ein SHL-Funktionsblockdiagramm

Beispiel für Kontaktplandiagramm SHL



Beispiel für SHL in strukturiertem Text

```
1  in := 123;
2  nbs := 2;
3  output := SHL(in, nbs);
```

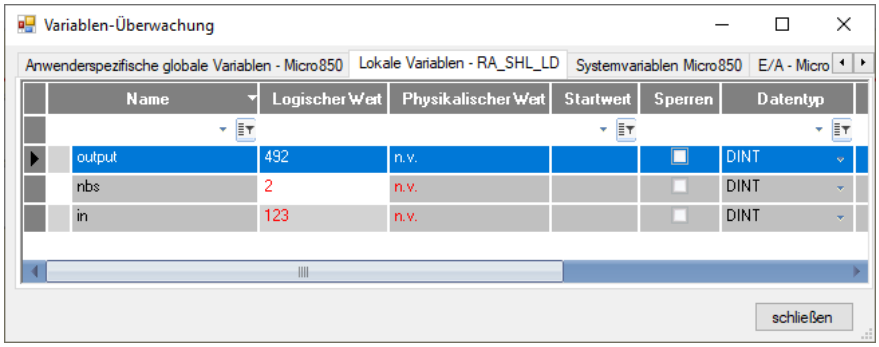
(* ST-Äquivalenz: *)

result := SHL (register,1);

(* Register = 2#0100_1101_0011_0101 *)

(* Ergebnis = 2#1001_1010_0110_1010 *)

Ergebnisse

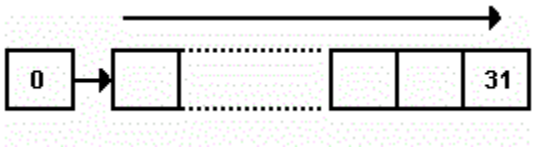


SHR (Verschiebung rechts)

Verschiebt die 32 Bits einer Ganzzahl nach rechts und bildet das Bit ganz links (wertigstes Bit) nach, um die freien Bits zu füllen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

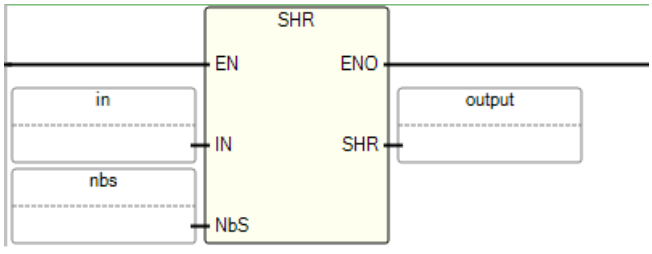


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Verschieben von Ganzzahlen nach rechts. FALSE - Es findet keine Verschiebung von Ganzzahlen statt. Gilt für Kontaktplan-Programme.
IN	Eingang	DINT	Beliebige Ganzzahl.
NbS	Eingang	DINT	Anzahl der 1-Bit-Verschiebungen (im Bereich [1..31]).
SHR	Ausgang	DINT	Nach rechts verschobener Wert. Es wird keine Änderung vorgenommen, wenn NbS <= 0. Wenn ein Wert von 0 vorliegt, wird das höchstwertige Bit ersetzt.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für ein SHR-Funktionsblockdiagramm

Beispiel für Kontaktplandiagramm SHR



Beispiel für SHR in strukturiertem Text

```
1  in := 123;
2  nbs := 2;
3  output := SHR(in, nbs);
```

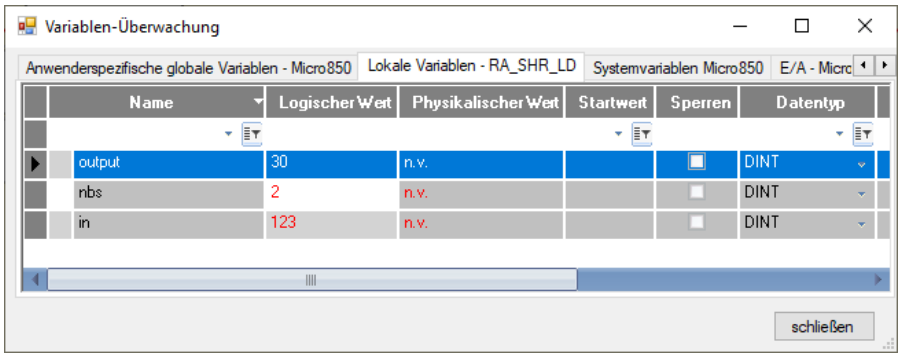
(* ST-Äquivalenz: *)

result := SHR (register,1);

(* Register = 2#1100_1101_0011_0101 *)

(* Ergebnis = 2#0110_0110_1001_1010 *)

Ergebnisse



XOR_MASK (exklusive OR-Maske)

Ganzzahlige exklusive bitweise OR-Maske gibt invertierte Bitwerte zurück.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

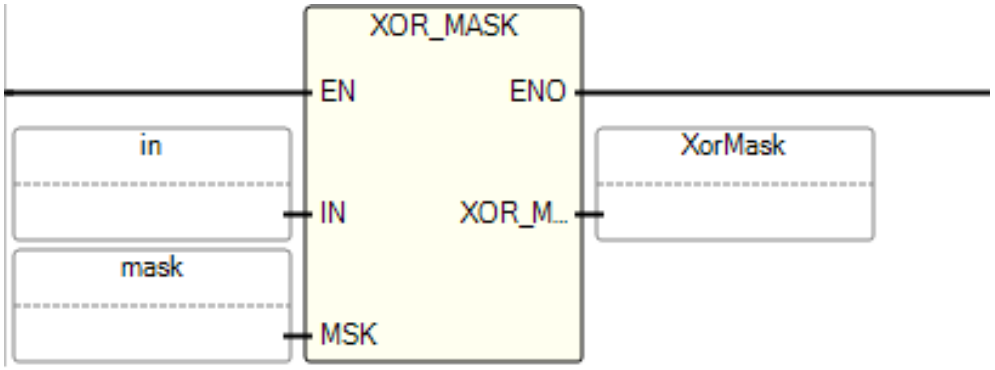
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführen der Berechnung mit der exklusiven bitweisen OR-Maske. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN	Eingang	DINT	Muss als ganze Zahl vorliegen.

Parameter	Parametertyp	Datentyp	Beschreibung
MSK	Eingang	DINT	Muss als ganze Zahl vorliegen.
XOR_MASK	Ausgang	DINT	Bitweises logisches exklusives OR zwischen IN und MSK.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für ein XOR_MASK-Funktionsblockdiagramm

Beispiel für einen XOR_MASK-Kontaktplan



Beispiele für strukturierten XOR_MASK-Text

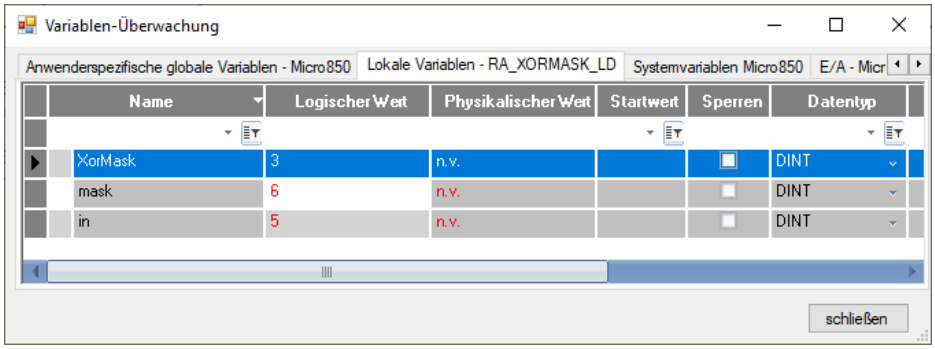
```
1  in := 5;
2  mask := 6;
3  XorMask := XOR_MASK(in, mask);
```

(* ST-Äquivalenz: *)

crc32 := XOR_MASK (prevcrc, nextc);

result := XOR_MASK (16#012, 16#011); (* gleich 16#003 *)

Ergebnisse


Variablen-Überwachung
Anwenderspezifische globale Variablen - Micro850
Lokale Variablen - RA_XORMASK_LD
Systemvariablen Micro850
E/A - Micr
Name
Logischer Wert
Physikalischer Wert
Startwert
Sperren
Datentyp
XorMask
3
n.v.
DINT
mask
6
n.v.
DINT
in
5
n.v.
DINT
schließen

Boolesche Anweisungen

Verwenden Sie boolesche Anweisungen, um einen Ausgangswert basierend auf einer logischen Berechnung aus den Eingaben zu bestimmen. Die Modulausgaben lassen sich direkt über das Programm oder unabhängig über das Modul. das die booleschen Anweisungen verwendet, steuern.

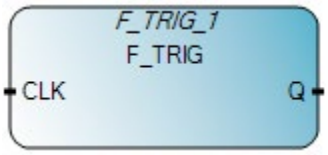
Funktion	Beschreibung
MUX4B auf seite 159	Multiplexer zwischen vier BOOL-Eingängen, gibt einen BOOL-Wert aus.
MUX8B auf seite 156	Multiplexer zwischen acht BOOL-Eingängen gibt einen BOOL-Wert aus.
TTABLE auf seite 154	Stellt den Wert der Ausgabe entsprechend einer Kombination von Eingangswerten bereit.
Funktionsblock	**Beschreibung**
F_TRIG auf seite 145	Ermittelt die fallende Flanke einer booleschen Variablen.
RS auf seite 148	Dominantes (höchste Priorität beim Bestimmen eines Anweisungsverhaltens) bistabiles Element zurücksetzen.
R_TRIG auf seite 147	Ermittelt die steigende Flanke einer booleschen Variablen.
SR auf seite 152	Dominantes bistabiles Element festlegen.
Operator	**Beschreibung**
AND auf seite 150	Führt eine boolesche AND-Operation zwischen zwei oder mehr Werten aus.
NOT auf seite 152	Konvertiert boolesche Werte in negierte Werte.
XOR auf seite 151	Boolesches exklusives OR mit zwei Werten.
OR auf seite 150	Boolesches OR mit mindestens zwei Werten.

F_TRIG (Erkennung der Abstiegsflanke)

Ermittelt die fallende Flanke einer booleschen Variablen. Der Block F_TRIG stellt den Ausgang Q für einen Zyklus ein, wenn der Eingang CLK von eingestellt auf gelöscht wechselt (d.h. eine fallende Flanke am Eingang CLK wird erkannt).

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

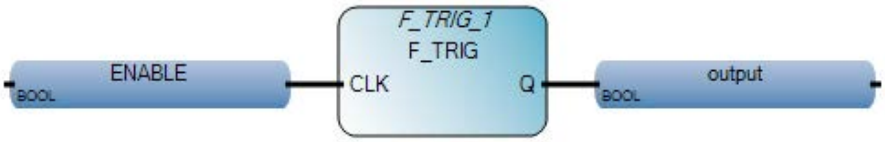
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



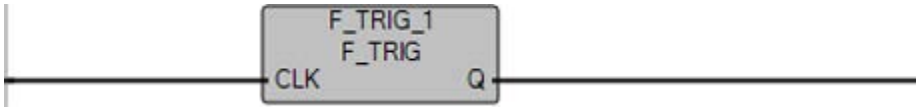
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
CLK	Eingang	BOOL	Überprüft die Eingabe für eine Abstiegsflanke. Jegliche Boolesche Variable. TRUE = keine Abstiegsflanke erkannt. FALSE = Abstiegsflanke an Eingabe CLK erkannt, Ausgabe Q auf TRUE gesetzt.
Q	Ausgang	BOOL	Gibt den Status für die Ausgabe Q an. TRUE = Abstiegsflanke entdeckt, Ausgabe Q für einen weiteren Zyklus festgelegt. FALSE = Keine Änderung an Ausgabe Q.

Diagrammbeispiel für Funktionsblock F_TRIG



F_TRIG-Kontaktplan-Beispiel



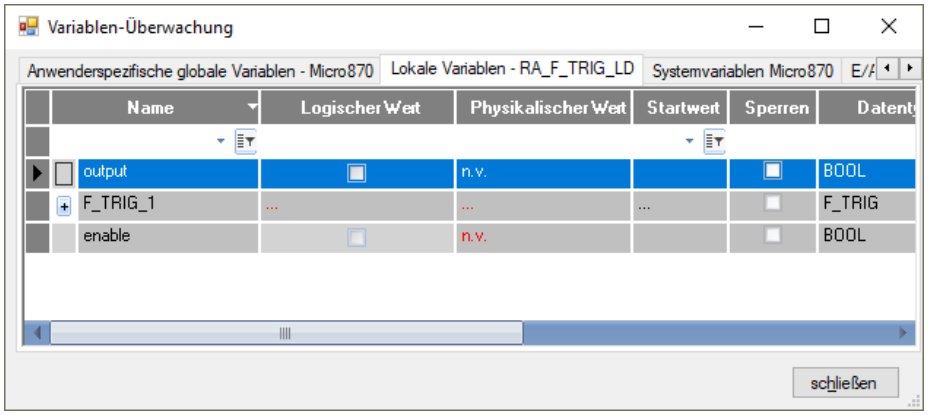
Beispiel für F_TRIG in strukturiertem Text

```
1  F_TRIG_1(ENABLE);
2  IF F_TRIG_1.Q THEN
3     output := TRUE;
4  END_IF;
```

(* ST-Äquivalenz: F_TRIG1 ist eine Instanz des Blocks F_TRIG*)

```
F_TRIG1(cmd);
nb_edge := ANY_TO_DINT(F_TRIG1.Q) + nb_edge;
```

Ergebnisse

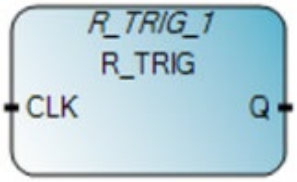


R_TRIG (Anstiegflankenerkennung)

Ermittelt die steigende Flanke einer booleschen Variablen. Der Block R_TRIG stellt den Ausgang Q für einen Zyklus ein, wenn der Eingang CLK von gelöscht auf eingestellt wechselt (d.h. eine Anstiegflanke am Eingang CLK wird erkannt).

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

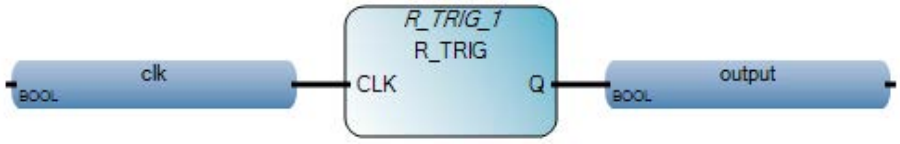
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
CLK	Eingang	BOOL	Jegliche Boolesche Variable. TRUE – Anstiegflanke erkannt, Q auf TRUE gesetzt. FALSE – keine Anstiegflanke erkannt, Q auf FALSE gesetzt.
Q	Ausgang	BOOL	TRUE – wenn CLK den Wert TRUE hat. FALSE – in allen anderen Fällen.

Beispiel für ein R_TRIG-Funktionsblockdiagramm



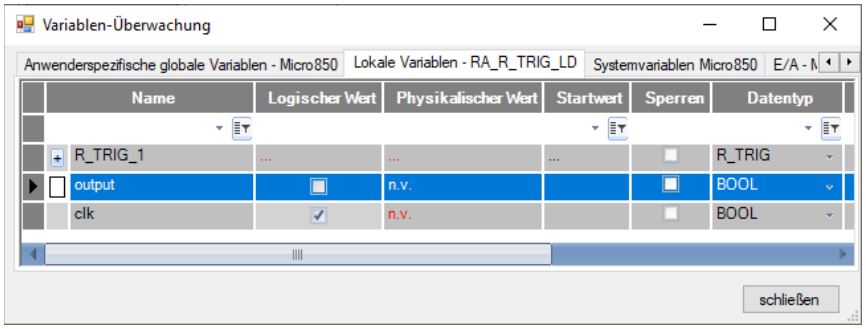
Beispiel für R_TRIG-Kontaktplandiagramm

Beispiel für strukturierten R_TRIG-Text

(* ST-Äquivalenz: R_TRIG1 ist eine Instanz eines R_TRIG-Blocks *)

```
R_TRIG1(cmd);
nb_edge := ANY_TO_DINT(R_TRIG1.Q) + nb_edge;
```

Ergebnisse



RS (Zurücksetzen/Setzen)

Setzt das dominante (höchste Priorität beim Bestimmen eines Anweisungsverhaltens) bistabile Element zurück bzw. legt es fest.

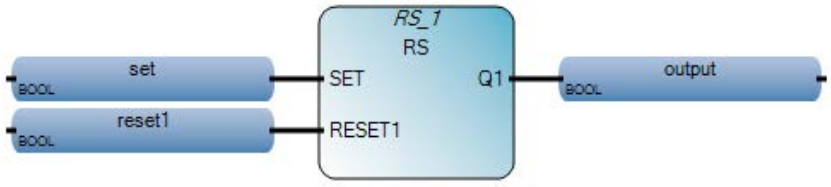
Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

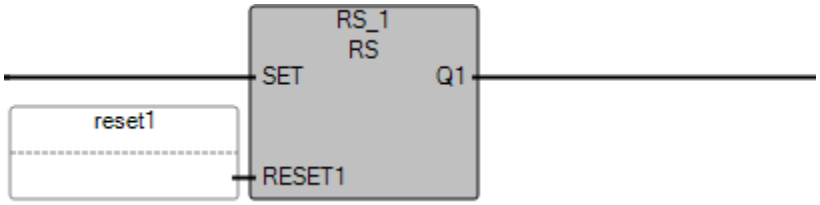
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
SET	Eingang	BOOL	TRUE – Setzt Q1 auf TRUE.
RESET1	Eingang	BOOL	TRUE – Setzt Q1 auf FALSE (dominant).
Q1	Ausgang	BOOL	Boolescher Speicherzustand.

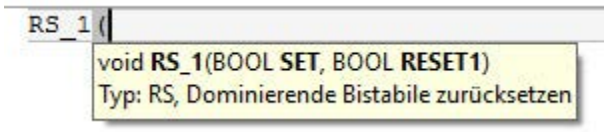
Beispiel für RS-Funktionsblockdiagramm



Beispiel für RS-Kontaktplandiagramm



Beispiel für strukturierten RS-Text

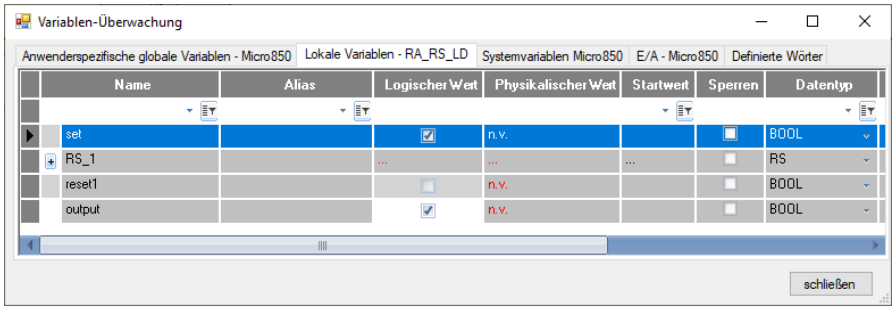



```
1  set := TRUE;
2  reset1 := FALSE;
3  RS_1(set, reset1);
4  output := RS_1.Q1;
```

(* ST-Äquivalenz: RS1 ist eine Instanz des Blocks RS*)

```
RS1(start_cmd, (stop_cmd OR alarm));
command := RS1.Q1;
```

Ergebnisse

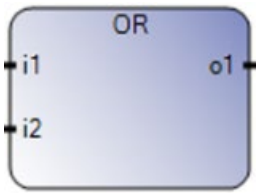


OR

Führt eine logische OR-Operation mit zwei oder mehr booleschen Werten aus, und gibt den booleschen Wert „True“ zurück, wenn einer der beiden Eingänge „True“ ist. Andernfalls gibt die Operation „False“ zurück.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
i1	Eingang	BOOL	
i2	Eingang	BOOL	
o1	Ausgang	BOOL	Boolesches OR der Eingabebegriffe. TRUE – Wenn mindestens eine Eingabe TRUE ist. FALSE – Wenn Eingaben FALSE sind.

Beispiel für strukturierten OR-Text

(* ST-Äquivalenz: *)

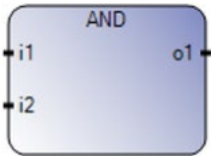
```
bo10 := bi101 OR NOT (bi102);
bo5 := (bi51 OR bi52) OR bi53;
```

AND

Führt eine boolesche AND-Operation zwischen zwei oder mehr Werten aus.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
i1	Eingang	BOOL	Wert als boolescher Datentyp.
i2	Eingang	BOOL	Wert als boolescher Datentyp.
o1	Ausgang	BOOL	Ergebnis der booleschen AND-Operation der Eingabewerte.

Beispiel für strukturierten AND-Text

(* ST-Äquivalenz: *)

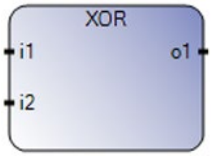
```
bo10 := bi101 AND NOT (bi102);
bo5 := (bi51 AND bi52) AND bi53;
```

XOR (exklusives OR)

Führt eine exklusive OR-Operation mit den beiden booleschen Werten durch.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
i1	Eingang	BOOL	
i2	Eingang	BOOL	
o1	Ausgang	BOOL	Boolesches Exclusive OR der beiden Eingabebegriffe. TRUE – Wenn einer der beiden Eingänge TRUE ist. FALSE – Wenn beide Eingänge FALSE sind oder beide Eingänge TRUE sind.

Beispiel für strukturierten XOR-Text

(* ST-Äquivalenz: *)

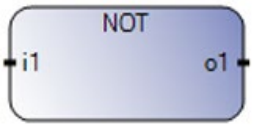
```
bo10 := bi101 XOR NOT (bi102);
bo5 := (bi51 XOR bi52) XOR bi53;
```

NOT

Konvertiert boolesche Werte in negierte Werte.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
i1	Eingang	BOOL	Jeder Wert vom Typ „Boolean“ oder jeder komplexe Ausdruck.
o1	Ausgang	BOOL	TRUE, wenn IN = FALSE. FALSE, wenn IN = TRUE.

Beispiel für strukturierten NOT-Text

(* ST-Äquivalenz: *)

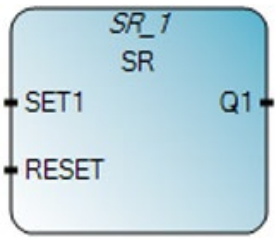
```
bo10 := NOT (bi101);
```

SR (setzen/zurücksetzen)

Dominantes bistabiles Element festlegen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

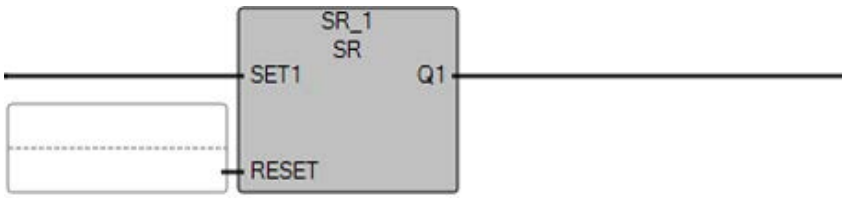
Parameter	Parametertyp	Datentyp	Beschreibung
SET1	Eingang	BOOL	TRUE – Setzt Q1 auf TRUE (dominant: höchste Priorität beim Bestimmen eines Anweisungsverhaltens).
RESET	Eingang	BOOL	TRUE – Setzt Q1 auf FALSE.
Q1	Ausgang	BOOL	Boolescher Speicherzustand. TRUE – Wenn SET1 den Wert TRUE hat. FALSE – Wenn RESET den Wert TRUE hat.

Beispiel für ein dominantes bistabiles Element

Set1	Reset	Q1	Result Q1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Beispiel für SR-Funktionsblockdiagramm

Beispiel für SR-Kontaktplandiagramm



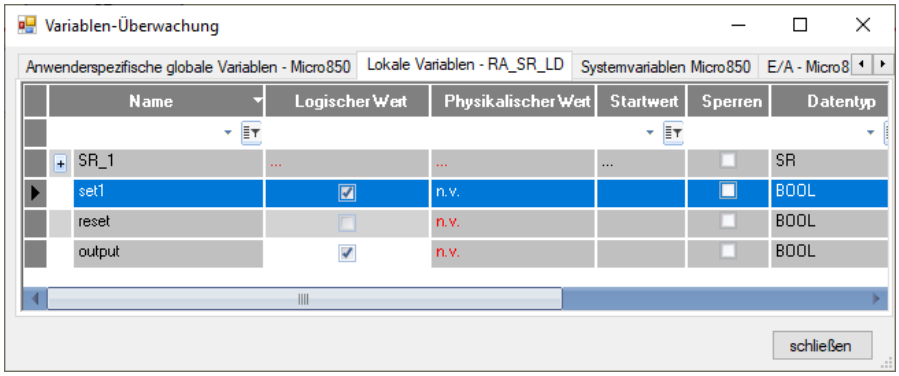
Beispiel für strukturierten SR-Text

```
1  set1 := TRUE;
2  reset := FALSE;
3  SR_1(set1, reset);
4  output := SR_1.Q1;
```

(* ST-Äquivalenz: SR1 ist eine Instanz des Blocks SR*)

```
SR1((auto_mode & start_cmd), stop_cmd);
command := SR1.Q1;
```

Ergebnisse



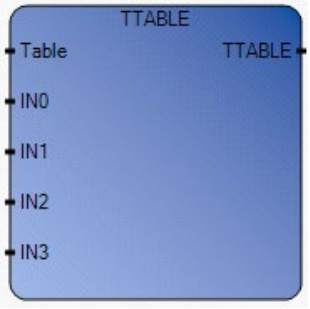
TTABLE (Wahrheitstabelle)

Stellt den Wert der Ausgabe entsprechend einer Kombination von Eingangswerten bereit.

Wenn der Wert mit 0xABCD festgelegt ist und In3 bis In0 der Zahl 7 entsprechen, gibt TTABLE den Wert von Bit 7 in der Tabelle zurück (also 1). Das niederwertigste Bit in der Tabelle ist Bit 0.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Table	Eingang	UINT	Wahrheitstabelle der BOOLESCHEN Funktion.
IN0	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert.
IN1	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert.
IN2	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert.
IN3	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert.
TTABLE	Ausgang	BOOL	Der Wert der Ausgabe entsprechend der Kombination von Eingangswerten.

Eingangskombinationen von TTABLE

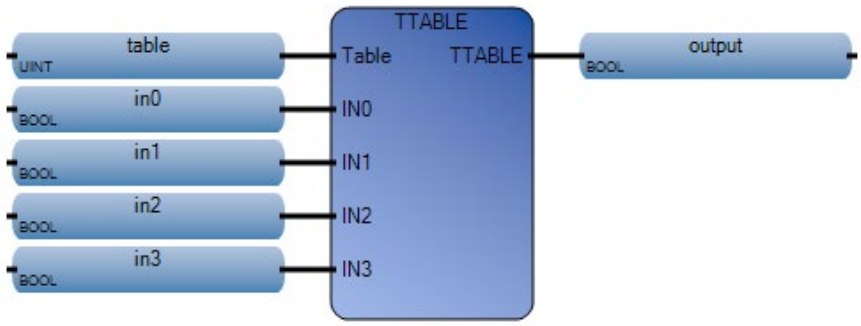
Die TTABLE-Anweisung besitzt vier Eingänge und somit 16 Kombinationen. Diese Kombinationen sind in einer Wahrheitstabelle enthalten; der Ausgabewert kann für jede Kombination angepasst werden. Die Anzahl der

konfigurierbaren Kombinationen hängt von der Anzahl der mit der Funktion verbundenen Eingänge ab.

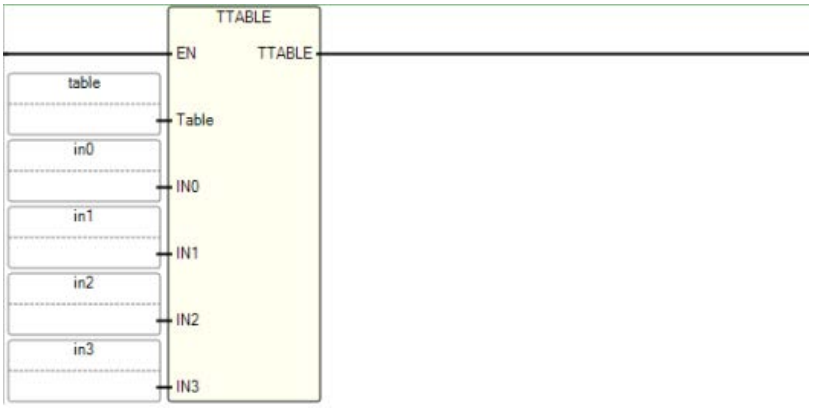
Beispiel für eine Wahrheitstabellen-Kombination.

Nummer	In3	In2	In1	In0
1	0	0	0	0
2	0	0	0	1
3	0	0	1	0
4	0	1	1	1
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

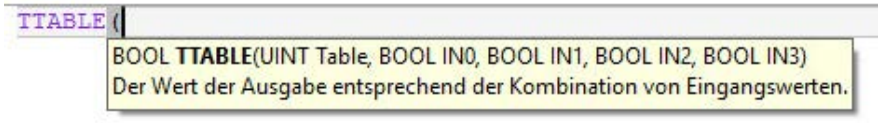
Beispiel für ein TTABLE-Funktionsblockdiagramm



Beispiel für TTABLE-Kontaktplandiagramm

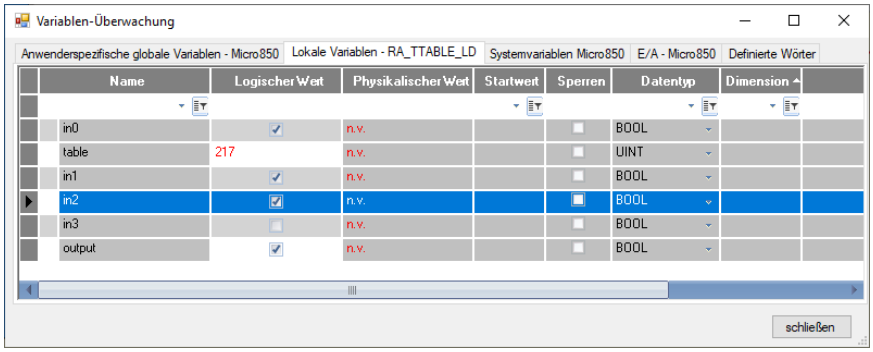


Beispiel für strukturierten TTABLE-Text




```
1  table := 217;
2  in0 := TRUE;
3  in1 := TRUE;
4  in2 := TRUE;
5  in3 := FALSE;
6  output := TTABLE(table, in0, in1, in2, in3);
```

Ergebnisse

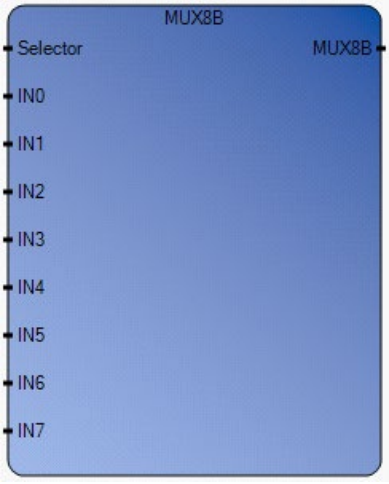


MUX8B (Multiplexer mit 8 BOOL-Eingängen)

Multiplexer zwischen acht BOOL-Eingängen gibt einen BOOL-Wert aus.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

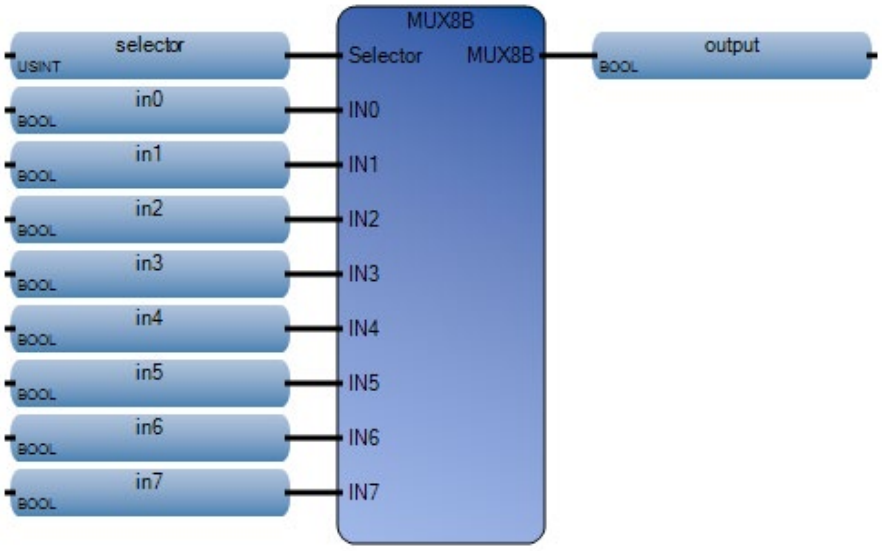
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



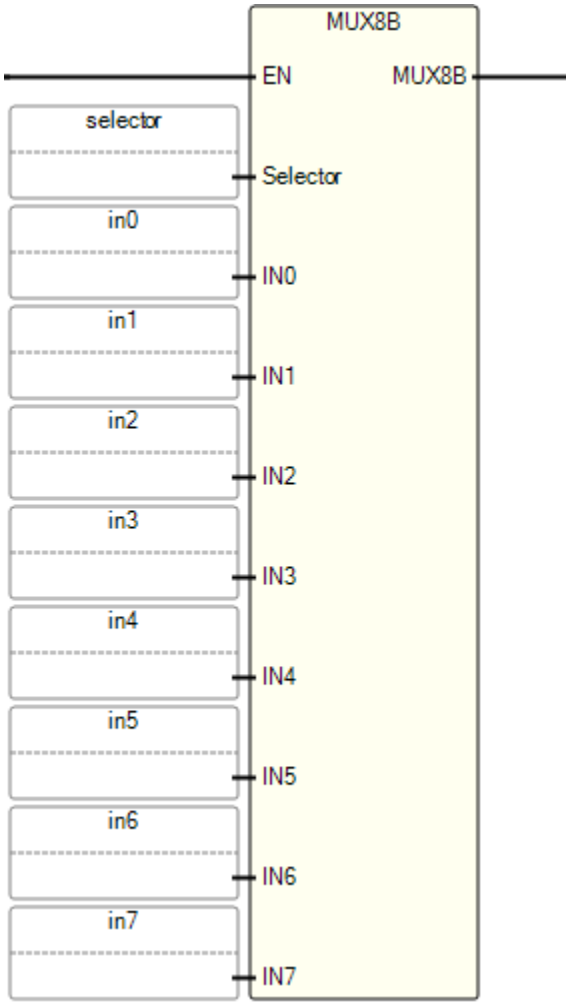
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Selector	Eingang	USINT	Ganzzahliger Auswahlwert muss im Satz [0...7] liegen.
IN0	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert. TRUE – Wenn die Auswahl 0 ist. FALSE - Wenn die Auswahl nicht 0 ist.
IN1	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert. TRUE – Wenn die Auswahl 1 ist. FALSE - Wenn die Auswahl nicht 1 ist.
IN2	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert. TRUE – Wenn die Auswahl 2 ist. FALSE - Wenn die Auswahl nicht 2 ist.
IN3	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert. TRUE – Wenn die Auswahl 3 ist. FALSE - Wenn die Auswahl nicht 3 ist.
IN4	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert. TRUE – Wenn die Auswahl 4 ist. FALSE - Wenn die Auswahl nicht 4 ist.
IN5	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert. TRUE – Wenn die Auswahl 5 ist. FALSE - Wenn die Auswahl nicht 5 ist.
IN6	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert. TRUE – Wenn die Auswahl 6 ist. FALSE - Wenn die Auswahl nicht 6 ist.
IN7	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert. TRUE – Wenn die Auswahl 7 ist. FALSE - Wenn die Auswahl nicht 7 ist.
MUX8B	Ausgang	BOOL	TRUE – wenn: • In0, wenn „Selector" = 0 • In1, wenn „Selector" = 1 • In2, wenn „Selector" = 2 • In3, wenn „Selector" = 3 • In4, wenn „Selector" = 4 • In5, wenn „Selector" = 5 • In6, wenn „Selector" = 6 • In7, wenn „Selector" = 7 FALSE – bei allen anderen Werten der Auswahl.

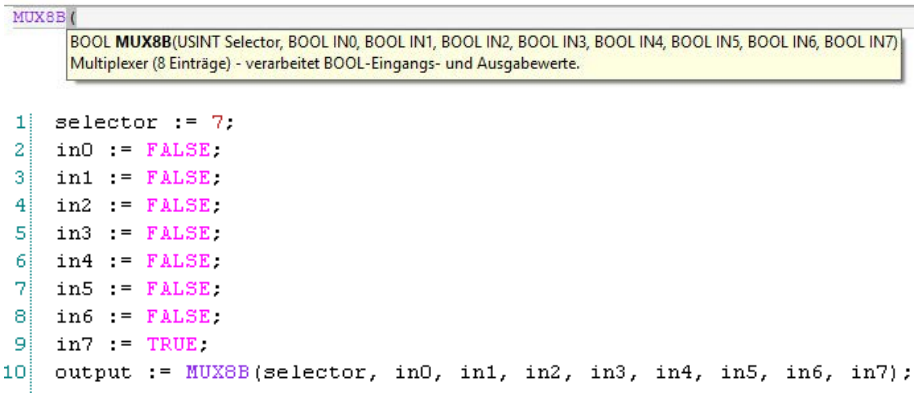
Beispiel für ein MUX8B-Funktionsblockdiagramm



Beispiel für Kontaktplandiagramm MUX8B



Beispiel für einen strukturierten MUX8B-Text

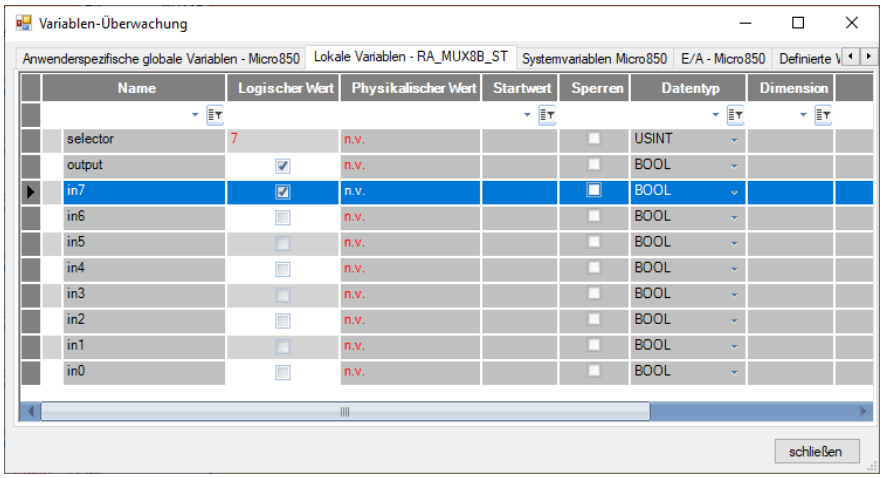


(* ST-Äquivalenz: *)

range := MUX8 (choice, 1, 5, 10, 50, 100, 500, 1000, 5000);

(* Auswahl aus 8 vordefinierten Bereichen; wenn beispielsweise 3 gewählt ist (choice), ist der Bereich 50 *)

Ergebnisse

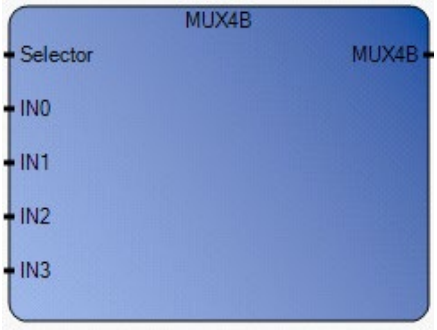


MUX4B (Multiplexer mit 4 BOOL-Eingängen)

Multiplexer zwischen vier BOOL-Eingängen, gibt einen BOOL-Wert aus.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

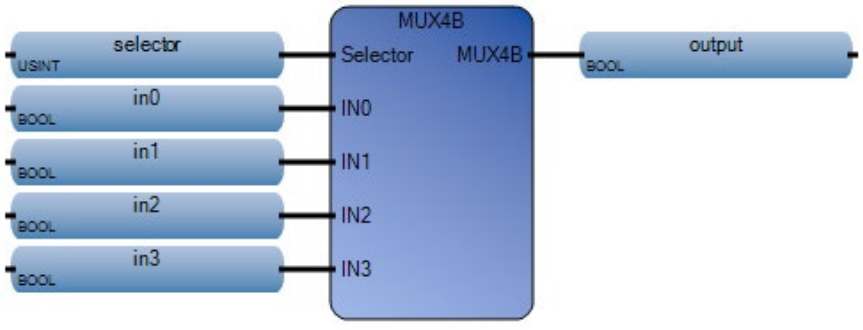
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



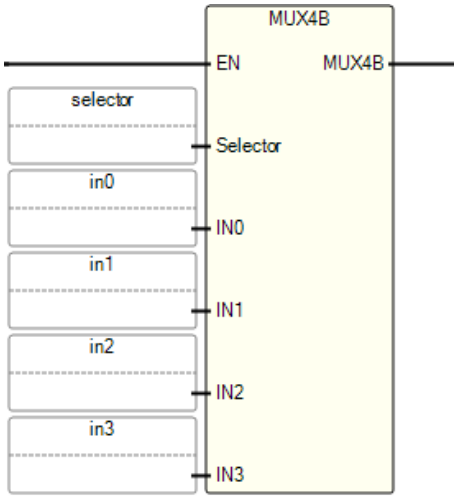
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Selector	Eingang	USINT	Ganzzahliger Auswahlwert muss im Satz [0...3] liegen.
IN0	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert. TRUE – Wenn die Auswahl 0 ist. FALSE - Wenn die Auswahl nicht 0 ist.
IN1	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert. TRUE – Wenn die Auswahl 1 ist. FALSE - Wenn die Auswahl nicht 1 ist.
IN2	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert. TRUE – Wenn die Auswahl 2 ist. FALSE - Wenn die Auswahl nicht 2 ist.
IN3	Eingang	BOOL	Beliebiger BOOLESCHER Eingabewert. TRUE – Wenn die Auswahl 3 ist. FALSE - Wenn die Auswahl nicht 3 ist.
MUX4B	Ausgang	BOOL	TRUE – wenn: • In0, wenn „Selector" = 0 • In1, wenn „Selector" = 1 • In2, wenn „Selector" = 2 • In3, wenn „Selector" = 3 FALSE – bei allen anderen Werten der Auswahl.

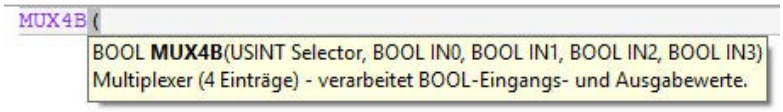
Beispiel für ein MUX4B-Funktionsblockdiagramm



Beispiel für Kontaktplandiagramm MUX4B



Beispiel für einen strukturierten MUX4B-Text



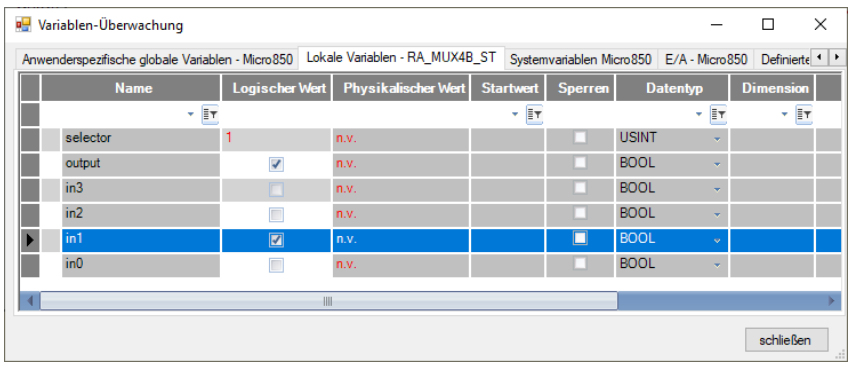

```
selector := 1;
in0 := FALSE;
in1 := TRUE;
in2 := FALSE;
in3 := FALSE;
output := MUX4B(selector, in0, in1, in2, in3);
```

(* ST-Äquivalenz: *)

range := MUX4 (choice, 1, 10, 100, 1000);

(* Auswahl aus 4 vordefinierten Bereichen; wenn beispielsweise 1 gewählt ist (choice), ist der Bereich 10 *)

Ergebnisse



Kommunikationsanweisungen

Verwenden Sie Kommunikationsanweisungen zum Lesen, Schreiben, Vergleichen und Konvertieren von Kommunikationszeichenfolgen.

Funktionsblock	Beschreibung
COM_IO_WDOG auf seite 163	Überwacht die Kommunikation mit dem Controller.
MSG_CIPGENERIC auf seite 166	Sendet eine generische explizite CIP-Nachricht.
MSG_CIPSYMBOLIC auf seite 173	Sendet eine symbolische explizite CIP-Nachricht.
MSG_MODBUS auf seite 177	Sendet eine Modbus-Nachricht.
MSG_MODBUS2 auf seite 183	Sendet eine MODBUS-/TCP-Nachricht über einen Ethernet-Kanal.
MSG_PCCC auf seite 189	Sendet eine PCCC-Nachricht über einen Ethernet-Kanal oder einen seriellen Anschluss.

COM_IO_WDOG

Überwacht den Austausch mit externen Geräten über die Ein- und Ausgänge der Steuerung. Wenn beispielsweise der CIP-Schreibbefehl für die Variable _IO_EM_DO_00 nicht innerhalb der konfigurierten Zeitüberschreitung über EtherNet/IP empfangen wird, läuft der Watch-Dog-Timer ab und alle Controller-Ausgänge werden zurückgesetzt.

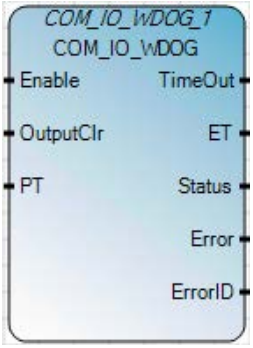
Es werden EtherNet/IP-, Modbus-TCP- und Modbus-RTU-Protokolle unterstützt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator. Nur mit Firmware-Version 12.00 und höher unterstützt.

Hinweis:

- Es können zwar mehrere Instanzen dieser Anweisung vorhanden sein, aber nur eine davon kann aktiviert sein. Andernfalls tritt ein Fehler auf, und die ErrorID wird auf 2 festgelegt.
- Es werden nur in die Steuerung integrierte, Plug-in- und digitale Erweiterungs-E/A-Module unterstützt.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

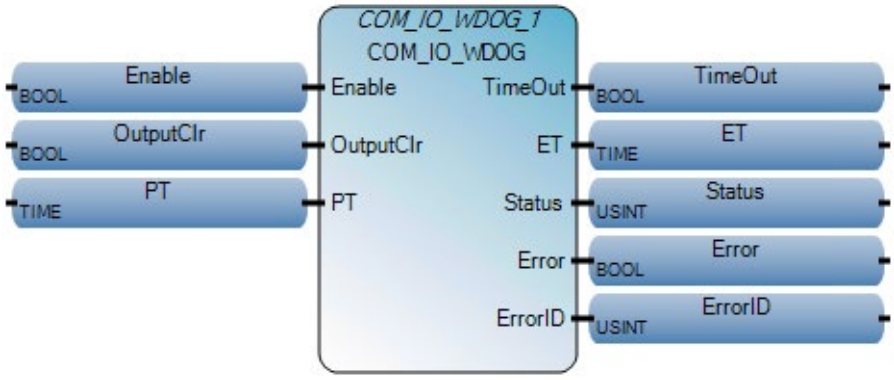
Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	TRUE - Dies sind die drei Fälle für den TRUE-Zustand. 1. Bei Anstiegflanke von TRUE beginnt der Funktionsblock mit der Ausführung. Der Status-Bit ist auf 0 eingestellt und der Rest der Bits wird gelöscht. Überprüfen Sie dann die folgenden Fehlerbedingungen. Wenn kein Fehler auftritt, fahren Sie mit dem nächsten Schritt fort. • Wenn PresetValue auf weniger als 1 Sekunde, Error auf TRUE und ErrorID auf 1 festgelegt sind, wird Status- Bit 4 festgelegt, TimeOut und ElapsedTime werden gelöscht. • Wenn der Timer bereits von der anderen Instanz abgerufen wurde, wird Error auf TRUE und ErrorID auf 2 festgelegt, Status- Bit 4 wird festgelegt, TimeOut und ElapsedTime werden gelöscht. 2. PresetTime > ElapsedTime. Wenn Enable auf TRUE festgelegt wurde und der Funktionsblock mit der Ausführung beginnt. Prüfen Sie, ob überhaupt Befehle erhalten werden. Wenn irgendwelche Befehle erhalten werden, setzen Sie den Timer sofort zurück. Legen Sie ElapsedTime auf 0 fest und legen Sie Statusbit 2. Der Rest der Bits wird gelöscht. Wenn keine Befehle erhalten werden, legen Sie Status- Bit auf 1 fest und der Rest der Bits wird gelöscht. 3. PresetTime = ElapsedTime. Wenn Enable auf TRUE festgelegt wurde und der Funktionsblock mit der Ausführung beginnt. Prüfen Sie, ob überhaupt Befehle erhalten werden. Werden Befehle erhalten, setzen Sie den Timer unverzüglich zurück. Legen Sie ElapsedTime auf 0 fest und legen Sie Statusbit 2. Der Rest der Bits wird gelöscht. Legen Sie andernfalls das TimeOut-Bit fest, legen Sie Status- Bit auf 3 fest und der Rest der Bits wird gelöscht. Setzen Sie die interne Variable fest, um die digitalen Ausgaben am Ende des Scans zu löschen, wenn die konfigurierte Aktion zum Löschen aller digitalen Ausgaben am Ende des Scans dient. FALSE – Die Anweisung wird nicht ausgeführt und die Funktionsblockausgänge werden gelöscht.
OutputClr	Eingang	BOOL	0 - Tun Sie nichts, wenn eine Zeitüberschreitung erfolgt. 1 - Löschen Sie alle digitalen Ausgaben am Ende des Scans (Integriert, EXIO und UPM), wenn eine Zeitüberschreitung erfolgt.
PT	Eingang	TIME	Wartezeit vor der Zeitüberschreitung. Der Wert für eine Zeitüberschreitung kann nicht weniger als eine Sekunde betragen, sonst wird ein Fehler angezeigt. Der maximale Wert für PresetTime kann der Maximalwert des Datentyp „TIME" sein.
TimeOut	Ausgang	BOOL	TRUE - ElapsedTime entspricht der PresetTime. FALSE - Enable ist auf FALSE festgelegt; der Timer ist nicht abgelaufen oder es ist ein Fehler aufgetreten.
ET	Ausgang	TIME	Die aktuell abgelaufene Zeit. Die möglichen Werte reichen von 0ms bis 1193h2m47s294ms.
Status	Ausgang	USINT	Status des Funktionsblocks. Bit 0 - Aktiviert Bit 1 - Timer wird ausgeführt, es wurde noch kein Aus- oder Eingang erhalten. Bit 2 - Der Ausgabe- oder Eingabebefehl wurde erhalten. Bit 3 - Die Zeitüberschreitung ist erfolgt. Kein Ausgabe- oder Eingabebefehl wurde erhalten. Bit 4 - Ein Fehler ist aufgetreten. Die anderen Bits sind reserviert.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist.
ErrorID	Ausgang	USINT	Wenn ein Fehler auftritt, enthält die ErrorID den Fehlercode.

COM_IO_WDOG-Fehlercode

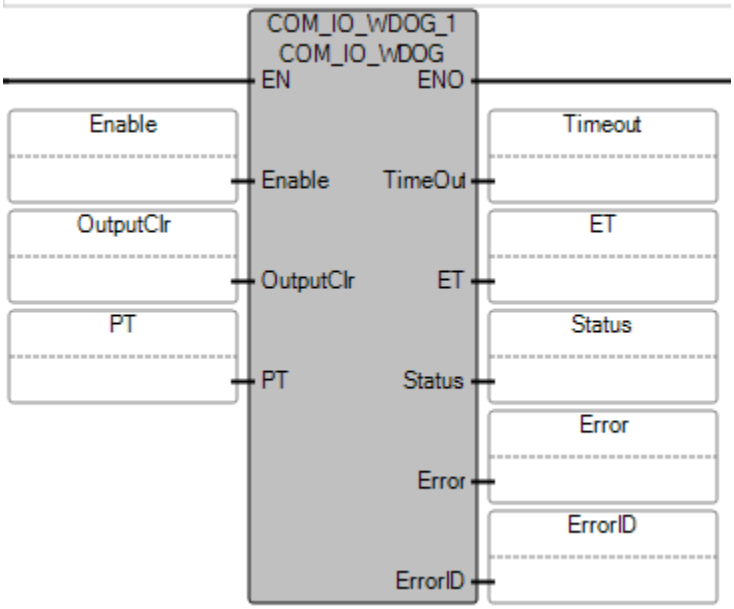
ErrorID-Code	Fehlerbeschreibung
1	Die PresetTime beträgt weniger als eine Sekunde.

ErrorID-Code	Fehlerbeschreibung
2	Es wird bereits eine andere Instanz des Funktionsblocks COM_IO_WDOG ausgeführt.

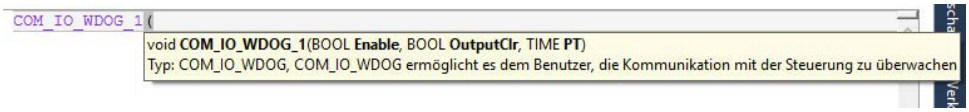
Diagrammbeispiel für Funktionsblock COM_IO_WDOG



COM_IO_WDOG-Kontaktplan-Beispiel



Beispiel für COM_IO_WDOG in strukturiertem Text




```
1  COM_IO_WDOG_1(Enable, OutputCrl, PT);
2   Timeout :=COM_IO_WDOG_1.Timeout;
3   ET :=COM_IO_WDOG_1.ET;
4   Status :=COM_IO_WDOG_1.Status;
5   Error :=COM_IO_WDOG_1.Error;
6   ErrorID :=COM_IO_WDOG_1.ErrorID;
```

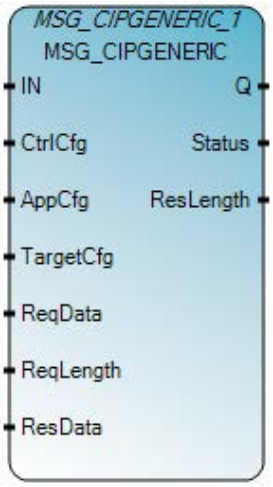
MSG_CIPGENERIC (Allgemeine CIP-Nachricht)

Sendet eine explizite CIP-Nachricht über einen Ethernet-Kanal oder einen seriellen Anschluss.

In einem Scanvorgang können maximal vier Nachrichtenanforderungen pro Kanal verarbeitet werden. Bei Kontaktplan-Programmen werden Nachrichtenanforderungen am Ende des Kontaktplanscans ausgeführt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

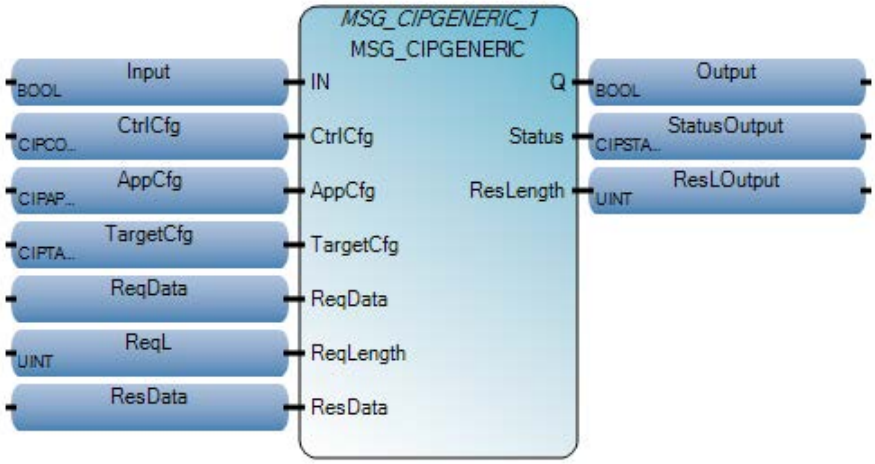


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

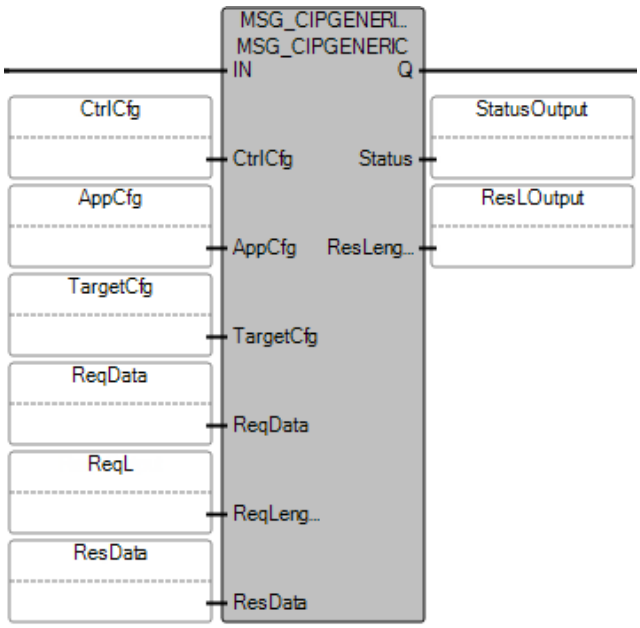
Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangszustand von Strompfad. TRUE – Anstiegsflanke erkannt. Startet den Anweisungsblock unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE – Anstiegflanke nicht erkannt, Leerlauf.
CtrlCfg	Eingang	CIPCONTROLCFG	Konfiguration der Ausführungssteuerung für Anweisungsblöcke Mit den Parametern des Datentyps CIPCONTROLCFG auf seite 168 können Sie „CtrlCfg“ definieren.
AppCfg	Eingang	CIPAPPCFG	Konfiguration des CIP-Service- und Anwendungspfads (EPATH). Mit den Parametern des Datentyps CIPAPPCFG auf seite 168, können Sie „AppCfg“ definieren.
TargetCfg	Eingang	CIPTARGETCFG	Konfiguration des Zielgeräts. Mit den Parametern des Datentyps CIPTARGETCFG auf seite 171 können Sie „TargetCfg“ definieren.
ReqData	Eingang	USINT[1..1]	CIP-Nachrichtenanforderungsdaten. Die Arraygröße muss größer als die „ReqLength“-Größe sein.
ReqLength	Eingang	UINT	Länge der CIP-Nachrichtenanforderungsdaten: • 0 - 490
ResData	Eingang	USINT[1..1]	CIP-Nachrichtenantwortdaten. Die Arraygröße muss größer als die „ReqLength“-Größe sein. Bei erstmaliger oder erneuter Auslösung eines MSG, werden die Daten im ResData-Array gelöscht.
Q	Ausgang	BOOL	Die Ausgaben dieser Anweisung werden asynchron vom Programm-Scan aktualisiert. Die Anweisung kann von Ausgabe Q nicht neu ausgelöst werden, da IN von der Kante ausgelöst wird. TRUE - MSG-Anweisung erfolgreich beendet.

Parameter	Parametertyp	Datentyp	Beschreibung
			FALSE - MSG-Anweisung ist nicht beendet.
Status	Ausgang	CIPSTATUS	Der Status des Anweisungsblocks. Wenn ein MSG erstmalig oder erneut ausgelöst wird, werden alle Elemente in „Status“ zurückgesetzt. Die Status-Ausgabe wird im Datentyp „CIPSTATUS“ auf seite 169 definiert.
ResLength	Ausgang	UINT	Länge der CIP-Nachrichtenantwortdaten: • 0 - 490 Wenn eine MSG ausgelöst oder erneut ausgelöst wird, wird ResLength auf 0 zurückgesetzt.

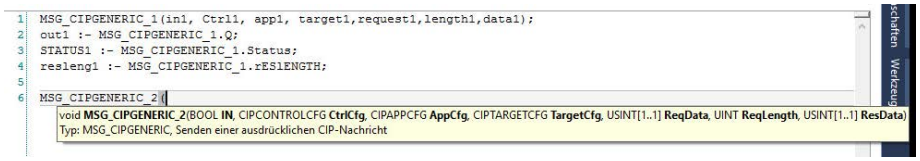
Beispiel für ein MSG_CIPGENERIC-Funktionsblockdiagramm



Beispiel für einen MSG_CIPGENERIC-Kontaktplan



Beispiel für einen strukturierten MSG_CIPGENERIC-Text



Datentyp „CIPAPPCFG“

Verwenden Sie diese Tabelle als Hilfestellung bei der Definition der Parameter für den Datentyp „CIPAPPCFG“.

Parameter	Datentyp	Beschreibung
Service	USINT	Dienstcode: 1 bis 127
Class	UINT	Wert der Klassen-ID des logischen Segments 1 bis 65535
Instance	UDINT	Wert der Instanz-ID des logischen Segments 0 bis 4294967295
Attribute	UINT	Wert der Attribut-ID des logischen Segments 1 bis 65535, 0 - Keine Attribut-ID verwendet
MemberCnt	USINT	Anzahl Mitglieder-IDs. Verwendete maximale Mitglieder-ID-Werte: 1 bis 3, 0 - Keine Mitglieder-ID verwendet
MemberId	UINT[3]	Mitglieder-ID-Werte: 0 bis 65535

Datentyp „CIPCONTROLCFG“

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „CIPCONTROLCFG“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Cancel	BOOL	TRUE - Die Ausführung des Funktionsblocks wird abgebrochen. Bit wird bei Aktivieren der Nachricht gelöscht. Wenn der Parameter „Abbrechen“ gesetzt und die Meldung aktiviert (EN-Bit gesetzt) und nicht fertig ist (DN-Bit nicht gesetzt), wird die Ausführung der Meldung abgebrochen und das ER-Bit gesetzt.
TriggerType	USINT	Steht für eines der Folgenden: • 0: Meldung einmal ausgelöst (wenn IN von „False“ auf „True“ wechselt) • 1 bis 65535: Zyklischer Auslösewert in Millisekunden. Meldung wird kontinuierlich ausgelöst, wenn IN = True ist. Legen Sie den Wert auf 1 fest, um die Meldung so schnell wie möglich auszulösen.
StrMode	USINT	Für die spätere Verwendung reserviert.

Auslösen einer CIP-Nachricht

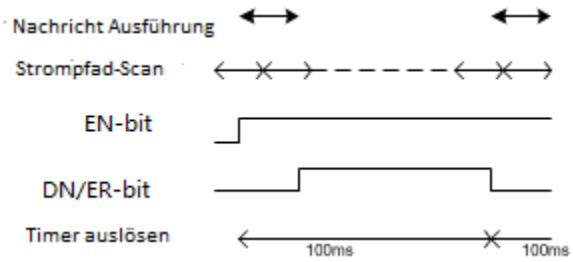
Eine CIP-Nachricht kann regelmäßig ausgelöst werden, indem dem Auslösertyp-Parameter ein von null verschiedener Wert zugewiesen wird.

Verwenden Sie diese Tabelle als Hilfestellung bei der Definition der Aktionen für den Parameter „Auslösertyp“.

Aktion	Ergebnisse
Nachricht ist aktiviert	Auslösertimer startet
Auslösertimer läuft ab, bevor die Nachricht abgeschlossen ist	Die Nachricht wird sofort im nächsten Kontaktplan-Scanzyklus ausgelöst.
Die Nachricht wird abgeschlossen, bevor die Auslöserzeit abläuft	Die Nachricht wird ausgelöst, wenn die Auslöserzeit abläuft.

Beispiel: Auslösen einer Nachricht

Im folgenden Beispiel wird der Auslösertyp-Wert auf 100 gesetzt.



Datentyp „CIPSTATUS“

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „CIPSTATUS“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Error	BOOL	Dieses Bit wird auf TRUE gesetzt, wenn bei der Ausführung des Funktionsblocks eine Fehlerbedingung auftritt.
ErrorID	UINT	Fehlercode-Wert. ErrorIDs sind in CIPSTATUS-Fehlercodes auf seite 170 definiert.
SubErrorID	UINT	Wert des Subfehlercodes. SubErrorIDs sind in CIPSTATUS-Fehlercodes definiert.
ExtErrorID	UINT	Code-Wert des Statusfehlers mit CIP-Erweiterung.
StatusBits	UINT	Dieser Parameter kann zur Überprüfung von Steuerbits: • Bit 0: EN – Aktivieren • Bit 1: EW – Warten aktivieren • Bit 2: ST – Start • Bit 3: ER – Fehler • Bit 4: DN – Fertig • Bit 5: CIPCONN - CIP-Verbindung trennen • Bit 6: EIPSESS - EIP-Sitzung schließen • Die anderen Bits sind reserviert StatusBits sind für CIPSTATUS-Statusbits auf seite 169 definiert.

CIPSTATUS-Statusbits

Die CISTATUS-Statusbits werden anhand des Status des Nachrichtenausführung, der Kommunikationspuffer und der Strompfadbedingungen festgelegt.

-	-	-	-	-	-	-	-	-	-	-	4	3	2	1	0

Bit	Name	Beschreibung	Verhalten
0	EN	Enable	Ist eingestellt, wenn der Strompfad auf „True“ wechselt und bleibt eingestellt, bis das DN-Bit oder das ER-Bit eingestellt wird oder der Strompfad auf False wechselt.
1	EW	Warten aktivieren	Ist aktiviert, wenn der Kommunikationspuffer für die Nachrichtenanforderung zugewiesen ist. Wird gelöscht, wenn das ST-Bit eingestellt wird.
2	ST	Start	Ist eingestellt, wenn die Nachricht übertragen wurde und auf eine Antwort wartet. Wird gelöscht, wenn das DN-Bit eingestellt wird.
3	ER	Fehler	Wird eingestellt, wenn die Nachricht nicht übertragen werden konnte. Es wird ein Fehlercode in die Fehler-ID eingetragen. Die Werte für das ER-Bit und den Fehlercode werden gelöscht, wenn der Strompfad beim nächsten Mal von False auf True wechselt.

Bit	Name	Beschreibung	Verhalten
4	DN	Fertig	Wird eingestellt, wenn die Nachricht erfolgreich übertragen wurde Die Werte für das DN-Bit werden gelöscht, wenn der Strompfad zum nächsten Mal von False auf True wechselt. Wenn das Done-Bit eingestellt ist, werden alle anderen Bits gelöscht, um anzuzeigen, dass die Nachricht erfolgreich abgeschlossen wurde. Wenn ein Fehler erkannt und das Fehler-Bit festgestellt wurde, werden die übrigen Statusbits (EN/EW/ST) nicht gelöscht.
5	CIPCONN	Fertig	Wird beim Trennen der CIP-Verbindung für die Kommunikation eingestellt. Das CIPCONN-Bit ist zutreffend, wenn ConnClose = True, in anderen Fällen ist das CIPCONN-Bit „False". Das CIPCONN-Bit wird auch für Seriell, Ethernet und USB verwendet.
6	EIPSESS	Fertig	Wird eingestellt, wenn die Kapselungs-CIP-Sitzung für die Kommunikation beendet wird. Das EIPSESS-Bit ist zutreffend, wenn ConnClose = True, in anderen Fällen ist das EIPSESS-Bit „False". Dieses Bit wird nur für Ethernet verwendet.

CIPSTATUS-Fehlercodes

Verwenden Sie diese Tabelle als Hilfestellung bei der Bestimmung der Parameterwerte für die Felder ErrorID und SubErrorID des Parameters CIPSTATUS, wenn das ER-Bit gesetzt ist.

ErrorID-Code	SubErrorID	Beschreibung des Fehlercodes
33	Fehler bei der Parameterkonfiguration	
	32	Kanalnummer fehlerhaft.
	36	CIP-Verbindungstyp wird nicht unterstützt.
	40	CIP-Symbolik-Datentyp wird nicht unterstützt.
	41	Ungültiger CIP-Symbolname.
	43	CIP-Klassenwert oder MemberID-Anzahl wird nicht unterstützt.
	48	Die Array-Größe für die Eingabedaten des Anweisungsblock ist unzureichend.
	49	Ungültiger Zielpfad.
	50	Dienstcode fehlerhaft.
	51	Das Array der Übertragungsdaten des Anweisungsblocks ist für die CIP-Kommunikation zu groß. Die maximale Länge der zu übertragenden Benutzerdaten ist je nach Nachrichtenkonfiguration unterschiedlich. Wenn die gesamte CIP-Nachrichtenlast (einschließliche Benutzerdaten und CIP-Nachrichtenüberlastung) mehr als 504 Byte beträgt, wird ein Fehler des Typs 0x21 (subError 0x33) angegeben.
	52	Ungültiger Wert für Segmenttyp.
	53	Ungültiger Wert für UCCM-Zeitüberschreitung. Wenn der Zeitüberschreitungswert der Kapselung kleiner als die Zeitüberschreitung für UCCM ist oder die Differenz zwischen Kapselungs- und UCCM-Zeitüberschreitung kleiner als oder gleich 1 Sekunde ist, wird ein Fehler 0x21 (SubError 0x35) gemeldet.
	54	Ungültiger Wert für Verbindungs-Zeitüberschreitung. Wenn der Zeitüberschreitungswert der Kapselung kleiner als der für die CONNECTED-Nachricht ist oder die Differenz zwischen Kapselungs- und Verbindungsnachrichten-Zeitüberschreitung <= 1 Sekunde ist, wird ein Fehler 0x21 (SubError 0x36) gemeldet.
55	Fehler bei der Zeitüberschreitung	
	112	Nachrichten-Zeitüberschreitung beim Warten in der Nachrichtenwarteschlange.
	113	Nachrichten-Zeitüberschreitung beim Warten auf eine Verbindung zur einzurichtenden Verbindungsschicht.
	114	Nachrichten-Zeitüberschreitung beim Warten auf Übertragung auf die Verbindungsschicht.
	115	Nachrichten-Zeitüberschreitung beim Warten auf eine Antwort von der Verbindungsschicht.
69	Fehlercodes in Verbindung mit Serverantwort-Formaten	
	65	Nachrichtenantwort stimmt nicht mit Anforderung überein.
	68	Der Nachrichtenantworttyp ist ungültig/wird nicht unterstützt. (MSG_CIPSYMBOLIC).

ErrorID-Code	SubErrorID	Beschreibung des Fehlercodes
208	Für das Netzwerk ist keine IP-Adresse konfiguriert.	
209	Maximale Anzahl der genutzten Verbindungen – keine Verbindungen verfügbar.	
210	Ungültige Internetadresse bzw. Knotenadresse.	
217	Ausführung der Nachricht wurde vom Benutzer abgebrochen. (Abbruchparameter wurde auf TRUE gesetzt).	
218	Kein Raum für Netzwerkpuffer verfügbar.	
222	Reserviert.	
223	Die Adresse der Verknüpfung ist nicht verfügbar. Eine Änderung der TCP/IP- oder Ethernet-Konfiguration wird ausgeführt.	
224	CIP-Antwort-Fehlercode. Mit SubErrorID wird der CIP-Status und mit ExtErrorID der erweiterte CIP-Statuswert festgelegt. Die Werte der möglichen Fehlercodes können Sie den CIP-Angaben entnehmen.	
255	Kanal ist heruntergefahren oder Neukonfiguration läuft. Der Fehlercode tritt unmittelbar nach dem Hochfahren auf, bis eine Verbindung eingerichtet wurde. Dieses Verhalten ist normal. Zudem kann er bei einer der folgenden Situationen auftreten: • Ein Ethernet-Kabel wird getrennt • Eine IP-Adresse wird nicht erkannt • Ein Plug-in des seriellen Anschlusses ist vorhanden, jedoch nicht konfiguriert	

Datentyp CIPTARGETCFG

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „CIPTARGETCFG“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Path	STRING[80]	Pfad für das Ziel. Es können höchstens zwei Hops angegeben werden. Die Pfadsyntax lautet: • {"<Port>,<Knoten/Einsteckplatzadresse>"}2
CipConnMode	USINT	CIP-Verbindungstyp. • 0 - Nicht verbunden (Standard) • 1 - Klasse-3-Verbindung
UcmmTimeout	UDINT	Getrennte Nachricht – Zeitüberschreitung (in Millisekunden). Die Wartezeit auf eine Antwort für getrennte Nachrichten einschließlich der Verbindungseinrichtung für verbundene Nachrichten. • Gültige Werte: 250 - 10.000 ms. • Setzen Sie ihn auf 0, um den Standardwert von 3.000 ms (3 Sekunden) zu verwenden. • Ein auf weniger als 250 ms gesetzter Wert wird auf 250 ms (Minimum) gesetzt. • Ein auf mehr als 10.000 ms gesetzter Wert wird auf 10.000 ms (Maximum) gesetzt.
ConnMsgTimeout	UDINT	Zeitüberschreitung bei Klasse-3-Verbindung (in Millisekunden). Wartedauer auf Antwort auf verbundene Nachrichten. Die Verbindung wird bei Ablauf der Zeitüberschreitung getrennt. • Gültige Werte: 800 - 10.000 ms. • Setzen Sie ihn auf 0, um den Standardwert von 10.000 ms (10 Sekunden) zu verwenden. • Ein auf weniger als 800 gesetzter Wert wird auf 800 ms (Minimum) gesetzt. • Ein auf mehr als 10.000 ms gesetzter Wert wird auf 10.000 ms (Maximum) gesetzt.
ConnClose	BOOL	Verhalten beim Trennen der Verbindung: • TRUE - Die Verbindung wird bei Abschluss der Nachricht getrennt. • FALSE - Verbindung bei abgeschlossener Nachricht nicht trennen (Standard).

Zielpfad für CIP-Messaging

Der Zielpfad für CIP-Messaging enthält Parameter, welche den Pfad und das

Ziel der CIP-Nachricht bestimmen.

Für den Zielpfad-Zeichenfolgenparameter wird folgende Syntax verwendet:

- „<Lokaler Port>, <1. Zieladresse>, [<Lokaler Port der 1. Zieladresse>, <2. Zieladresse>]“

Der erste Hop ist obligatorisch, der zweite optional.

Zeichenfolgenelement	Beschreibung
Lokaler Anschluss	Lokaler Anschluss zum Versenden der Nachricht. Bei dem Anschluss sollte es sich um einen aktiven EtherNet/IP- oder seriellen CIP-Anschluss handeln. USB-Anschlüsse werden nicht unterstützt.
Adresse des ersten Ziels	Zieladresse des ersten Hop • Geben Sie für EIP die IP-Adresse des Ziels an. Dabei muss es sich um eine Unicast-Adresse handeln, die nicht 0 und keine Multicast-, Broadcast-, lokale oder Loopback-Adresse (127.x.x.x) sein darf. • Geben Sie für CIP seriell die Knotenadresse des Ziels an. Der unterstützte Wert ist 1.
Lokaler Anschluss des ersten Ziels	Lokaler Anschluss zum Versenden der Nachricht.
Adresse des zweiten Ziels	Zieladresse des zweiten Hop

Zielpfad-Beispiel

In der folgenden Tabelle werden in einem Zielpfadstring verwendete Beispielwerte und die Ergebnisse für jeden String erläutert.

Zeichenfolgenelement	Beschreibung
„0,0“	Zielgerät ist das lokale Gerät.
„6,1“	Über Port 6 (serieller Micro830-UPM-Anschluss) wird der Knoten bei 1 erreicht.
„4,192.168.1.100“	Über Port 4 (integrierter Micro850-Ethernet-Anschluss) wird der Knoten bei 192.168.1.100 erreicht.
„4,192.168.1.100,1,0“	Über Port 4 (integrierter Micro850-Ethernet-Anschluss) wird der Knoten bei 192.168.0.100 erreicht (Logix ENET-Modul). Vom ENET-Modul über den Rückwandplatinen-Anschluss (Port 1) wird der Logix-Controller in Steckplatz 0 erreicht.

CIP/EIP-Nachrichtenverbindungen

Für die Client-Nachrichtenausführung werden maximal 16 CIP- (Klasse 3) und 16 EIP-Verbindungen unterstützt. In der folgenden Tabelle wird das Verhalten der CIP-/EIP-Verbindung beschrieben.

Szenario	Ergebnisse
Nachrichtenanforderung ist aktiviert und CipConnMode=1.	Wenn keine Verbindung zum Ziel besteht, wird eine CIP-Verbindung aufgebaut. Wenn bereits eine Verbindung mit dem Ziel besteht, wird die vorhandene CIP-Verbindung verwendet.
Nachrichtenverbindung ist aktiviert, CipConnMode=1 und der lokale Port der Nachricht ist Ethernet.	Wenn keine EIP-Verbindung zum Ziel vorhanden ist, wird vor dem Einrichten einer CIP-Verbindung eine EIP-Verbindung eingerichtet.
Nachrichtenverbindung ist aktiviert, CipConnMode=0 und der lokale Port der Nachricht ist Ethernet.	Wenn keine EIP-Verbindung mit dem Ziel besteht, wird eine EIP-Verbindung aufgebaut.

Szenario	Ergebnisse
Die Nachrichtenausführung wurde abgeschlossen und ConnClose wird auf „True“ gesetzt.	Es besteht nur eine Verbindung zum Ziel, die Verbindung wird geschlossen. Falls mehr als eine Verbindung zum Ziel besteht, wird die Verbindung getrennt, sobald die letzte Nachrichtenausführung abgeschlossen wurde. Beim Trennen der CIP-Verbindung werden sämtliche zugehörige EIP-Verbindungen ebenfalls geschlossen. Wenn mehrere CIP-Verbindungen auf die gleiche EIP-Verbindung zugreifen, wird die EIP-Verbindung getrennt, nachdem alle zugeordneten CIP-Verbindungen getrennt wurden.
Wenn ConnClose = True, werden die Nachrichtenverbindung und die EIP-Sitzung nach Abschluss der Nachrichtenausführung geschlossen.	Wenn mehr als eine Nachricht dieselbe Verbindung teilt, wird die Verbindung nach Abschluss der letzten Nachricht getrennt.
Eine CIP- oder EIP-Verbindung, die nicht mit einer aktiven Nachricht verknüpft ist, wird getrennt, wenn sie x Sekunden im Leerlauf ist. Dabei ist x ein konfigurierbarer Zeitüberschreitungswert bei Kapselungsinaktivität, der mit dem CIP-Dienst festgelegt werden kann.	Informationen bezüglich des CIP-Diensts finden Sie in den CIP-Spezifikationen, Ausgabe II für TCP/IP-Objekte.
Die Nachrichtenausführung wurde abgeschlossen und ConnClose wird auf False gesetzt.	Die Verbindung wurde nicht getrennt.
Die Verbindung ist mit keiner aktiven Nachricht verknüpft und bleibt für den Zeitraum, der im ConnTimeOut-Parameter angegeben ist, inaktiv.	Die Verbindung wurde getrennt.
Wechsel des Controllers von einem Ausführungsmodus (Betrieb, Remotelauf, Remotetestlauf, Remote-Kontaktplan) in einen Nicht-Ausführungsmodus.	Alle aktiven Verbindungen wurden getrennt.

Zeitüberschreitungs-Timer für CIP-Nachrichten

In der folgenden Tabelle wird beschrieben, wie sich Timer für CIPTARGETCFG-Zeitüberschreitungsparameter (UcmmTimeout und ConnMsgTimeout) basierend auf den Nachrichtenanforderungen und dem Status verhalten.

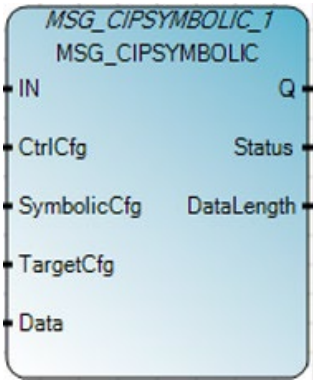
Aktion	Ergebnisse
Nachricht ist aktiviert	UcmmTimeout-Timer ist aktiviert
Verbindung wurde angefordert	ConnMsgTimeout-Timer ist aktiviert
ConnMsgTimeout-Timer ist aktiv	UcmmTimeout-Timer ist aktiviert.
Verbindungsanforderung ist abgeschlossen	UcmmTimeout-Timer ist reaktiviert

MSG_CIPSYMBOLIC (Symbolische CIP-Nachricht)

Sendet eine ausdrückliche CIP-Nachricht über einen Ethernet-Kanal oder einen seriellen Anschluss.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



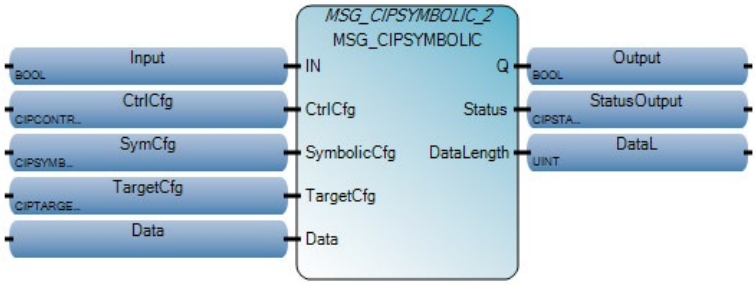
MSG_CIPSYMBOLIC-Vorgang

Wenn der Funktionsblock aktiviert ist, werden die Empfangspuffer für die Lesevorgänge an der Anstiegflanke von „Enable“ gelöscht.

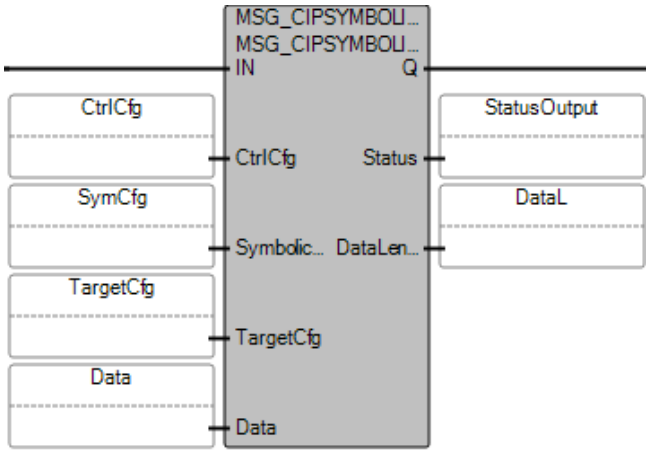
Argumente

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangszustand von Strompfad. TRUE – Anstiegsflanke erkannt. Startet den Anweisungsblock unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE – Anstiegflanke nicht erkannt, Leerlauf.
CtrlCfg	Eingang	CIPCONTROLCFG	Konfiguration der Ausführungssteuerung für Anweisungsblöcke Mit den Parametern des Datentyps CIPCONTROLCFG auf seite 168 können Sie „CtrlCfg“ definieren.
SymbolicCfg	Eingang	CIPSYMBOLICCFG auf seite 175	Hinweise zum Symbol für Lesen und Schreiben.
TargetCfg	Eingang	CIPTARGETCFG	Konfiguration des Zielgeräts. Mit den Parametern des Datentyps CIPTARGETCFG auf seite 171 können Sie „TargetCfg“ definieren.
Data	Eingang	USINT[490]	Der Lesebefehl speichert die vom Server zurückgegebenen Daten. Der Schreibbefehl puffert die an den Server zu sendenden Daten. Wenn eine MSG ausgelöst oder erneut ausgelöst wird, werden die Daten für den MSG-Lesebefehl gelöscht.
Q	Ausgang	BOOL	Die Ausgaben dieser Anweisung werden asynchron vom Programm-Scan aktualisiert. Die Anweisung kann von Ausgabe Q nicht neu ausgelöst werden, da IN von der Kante ausgelöst wird. TRUE - MSG-Anweisung erfolgreich beendet. FALSE - MSG-Anweisung ist nicht beendet.
Status	Ausgang	CIPSTATUS	Ausführungsstatus des Funktionsblocks Wenn eine MSG ausgelöst oder erneut ausgelöst wird, werden alle Elemente in „Status“ zurückgesetzt. Die Status-Ausgabe wird im Datentyp „CIPSTATUS“ auf seite 169 definiert.
DataLength	Ausgang	UDINT	Anzahl der gelesenen Datenbytes. Für geschriebene ist der Wert 0. Wenn eine MSG ausgelöst oder erneut ausgelöst wird, wird DataLength für den MSG-Lesebefehl auf 0 zurückgesetzt.

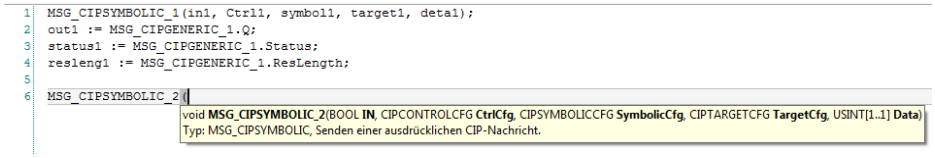
Beispiel für ein MSG_CIPSYMBOLIC-Funktionsblockdiagramm



Beispiel für einen MSG_CIPSYMBOLIC-Kontaktplan



Beispiel für einen strukturierten MSG_CIPSYMBOLIC-Text



Datentyp „CIPSYMBOLICCFG“

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „CIPSYMBOLICCFG“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Service	USINT	Dienstcode: • 0 - Lesen (Standard) • 1 - Schreiben
Symbol	STRING	Name der zu lesenden/schreibenden Variable. • Maximal 80 Zeichen. • Das Feld darf nicht leer sein. Die Symbolsyntax ist in symbolische Lese-/Schreib-Syntax auf seite 176 definiert.
Count	UINT	Anzahl der zu lesenden/schreibenden Variablenelemente. • Gültige Werte: 1 - 490 • 1 wird verwendet, wenn der Wert auf 0 gesetzt wird.
Type	Benutzerdefiniert	Benutzerdefinierter Datentyp für die Zielvariable. Der Typ ist in symbolischer Datentyp-Unterstützung definiert.

Parameter	Datentyp	Beschreibung
Offset	USINT	Für die spätere Verwendung reserviert. Ein Abstandsbyte der Lese-/Schreib-Variable zum Lesen/Schreiben einer großen Variable, die sich nicht einer Nachricht verarbeiten lässt. • 0 – 0xFF
Für die spätere Verwendung reserviert.		

Unterstützung symbolischer Datentypen

Verwenden Sie diese Tabelle, um zu ermitteln, welche Datentypen „MSG_CIPSYMBOLIC“ unterstützt.

Datentyp	Wert des Datentyps (hexadezimal)	Beschreibung
BOOL	193 (0xC1)	Logischer boolescher Wert mit den Werten WAHR (1) oder FALSCH (0).
SINT	194 (0xC2)	Wert einer 8-Bit-Ganzzahl mit Vorzeichen
INT	195 (0xC3)	Wert einer 16-Bit-Ganzzahl mit Vorzeichen
DINT	196 (0xC4)	Wert einer 32-Bit-Ganzzahl mit Vorzeichen
LINT	197 (0xC5)	Wert einer 64-Bit-Ganzzahl mit Vorzeichen
USINT	198 (0xC6)	Wert einer 8-Bit-Ganzzahl ohne Vorzeichen
UINT	199 (0xC7)	Wert einer 16-Bit-Ganzzahl ohne Vorzeichen
UDINT	200 (0xC8)	Wert einer 32-Bit-Ganzzahl ohne Vorzeichen
ULINT	201 (0xC9)	Wert einer 64-Bit-Ganzzahl ohne Vorzeichen
REAL	202 (0xCA)	32-Bit-Fließkommawert
LREAL	203 (0xCB)	64-Bit-Fließkommawert
STRING	218 (0xDA)	Zeichenfolge

Symbolische Syntax für Lesen/Schreiben

Die Syntax definiert die Kombinationen von Symbolen für einen gültige Lese-/Schreib-Anweisungsblock.

Gültige Symbolnamen

Jeder gültige Symbolname muss die nachstehenden Anforderungen erfüllen:

- Muss mit einem Buchstaben oder Unterstrich beginnen, der von einem Buchstaben, einer Zahl oder einem einzelnen Unterstrich gefolgt wird.
- Darf maximal 40 Zeichen enthalten.
- Darf keine zwei Unterstriche hintereinander enthalten.
- Verwenden Sonderzeichen [ ] . , als Trennzeichen.

Symbolsyntax

Verwenden Sie diese Tabelle, um die gültige Syntax für Symbole zu definieren. Es werden nur globale Variablen unterstützt.

Symbol	Syntax	Beispiel
Variable	PROGRAM:<Programmname>,<Symbolname>	PROGRAM:POU1.MyTag
Array	<Symbolname>[dim3, dim2, dim1] (Es werden maximal 3 Dimensionen unterstützt.)	MyTag1[0] MyTag2[3,6] MyTag3[1,0,4]
Struktur	<Symbolname>.<Symbolname des Strukturfelds>	MyTag4.time.year MyTag5.local.time[1].year

Unterstützte Datenpaketgröße für die serielle CIP-Funktion

Im Falle der Micro830-, Micro850- und Micro870-Controller kann die serielle CIP-Kommunikation sowohl von integrierten seriellen Anschlüssen als auch von seriellen Plug-in-Anschlüssen unterstützt werden. Das Datenpaket für die serielle CIP-Kommunikation umfasst Benutzerdaten und den CIP-Paket-Header.

In der Funktion eines seriellen CIP-Clients können die seriellen Micro830/Micro850-Anschlüsse maximal 490 Byte an gelesenen/geschriebenen Benutzerdaten unterstützen. Die Maximalspezifikation ist für serielle CIP-Daten mit einer Mindestgröße des Paket-Headers gültig. Ist der Paket-Header größer als die angegebene Mindestgröße, ist die vom CIP-Client unterstützte Maximalgröße an Benutzerdaten kleiner als 490 Byte. Falls das Datenpaket größer ist als die maximal vom CIP-Client unterstützte Datengröße meldet der Funktionsblock einen Fehler (0x21) und einen Teilfehler (0x33).

In der Funktion eines seriellen CIP-Servers können die seriellen Micro830-, Micro850- und Micro870-Anschlüsse mindestens 255 Byte an gelesenen/geschriebenen Benutzerdaten unterstützen. Die Spezifikation der Mindestgröße an Benutzerdaten ist für serielle CIP-Datenpakete mit maximaler Paket-Header-Größe gültig. Ist die Größe eines CIP-Paket-Headers kleiner als die angegebene Maximalgröße des Paket-Headers, kann der CIP-Client Datenpakete unterstützen, die größer sind als die Mindestspezifikation (das heißt, größer als 255 Byte). Ist jedoch die Größer der Benutzerdaten größer als die von der CIP-Serverfunktion unterstützte maximale Datengröße, kann es passieren, dass das CIP-Datenpaket gelöscht wird und es zu einer Zeitüberschreitung des Client kommt.

WICHTIG Bei der seriellen CIP-Serverfunktion wird empfohlen, nicht mehr als 255 Byte an Benutzerdaten in eine einzelne CIP-Nachricht zu lesen/schreiben.

MSG_MODBUS (Modbus-Nachricht)

Sendet eine Modbus-Nachricht über einen seriellen Anschluss.

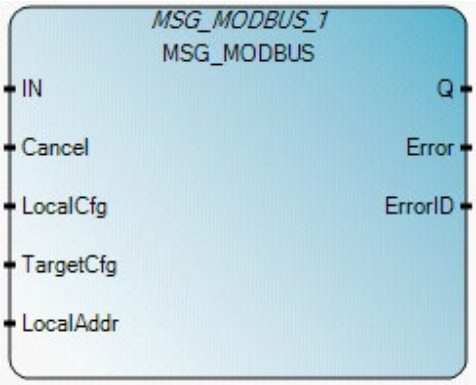
Details zum Betrieb:

- In einem Scanvorgang können maximal vier Nachrichtenanforderungen pro Kanal verarbeitet werden. Bei Kontaktplan-Programmen werden Nachrichtenanforderungen am Ende des Kontaktplanscans ausgeführt.

- Wenn ein Auslöser auf kontinuierliche Ausführung gesetzt ist, werden die Fehlercodes kontinuierlich gelöscht. Um Fehlercodes anzuzeigen, fügen Sie vor der MSG_MODBUS-Anweisung einen Strompfad ein.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

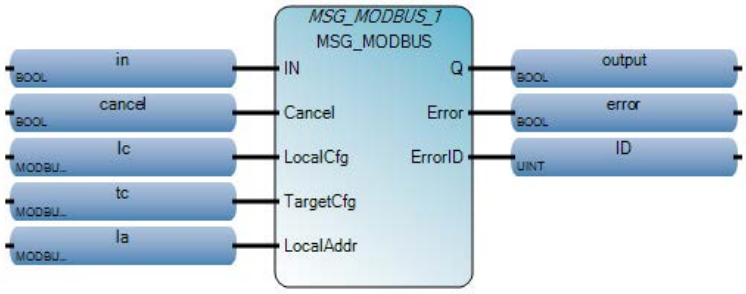
Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangszustand von Strompfad. TRUE – Anstiegsflanke erkannt. Startet den Anweisungsblock unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE – Anstiegflanke nicht erkannt, nicht gestartet.
Cancel	Eingang	BOOL	TRUE - Die Ausführung des Anweisungsblocks wird abgebrochen. FALSE – wenn IN = TRUE. Eingang abbrechen ist dominant.
LocalCfg	Eingang	MODBUSLOCPARA	Definition der Struktureingabe (lokales Gerät). Die Eingangsstruktur für das lokale Gerät können Sie mit dem Datentyp MODBUSLOCPARA auf seite 180 definieren.
TargetCfg	Eingang	MODBUSTARPARA	Definition der Strukturausgabe (Zielgerät). Die Eingangsstruktur für das Zielgerät können Sie mit dem Datentyp MODBUSTARPARA auf seite 182 definieren.
LocalAddr	Eingang	MODBUSLOCADDR	MODBUSLOCADDR ist ein 125-Wort-Array, der von Lesebefehlen zur Speicherung der Daten genutzt wird, die durch den Modbus-Slave zurückgegeben werden (1-125 Wörter); Schreibbefehlen dient er als Puffer für die Daten, die an den Modbus-Slave gesendet werden sollen (1-125 Wörter).
Q	Ausgang	BOOL	Die Ausgaben dieser Anweisung werden asynchron vom Programm-Scan aktualisiert. Die Anweisung kann von Ausgabe Q nicht neu ausgelöst werden, da IN von der Kante ausgelöst wird. TRUE - MSG-Anweisung erfolgreich beendet. FALSE - MSG-Anweisung ist nicht beendet.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler für diese Anweisung sind in MSG_MODBUS-Fehlercodes definiert.

MSG_MODBUS-Fehlercodes

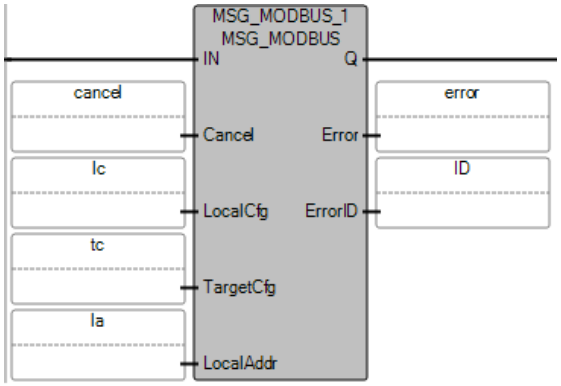
In dieser Tabelle werden die Fehlercodes für MSG_MODBUS beschrieben.

Fehlercode	Fehlerbeschreibung
3	Der Wert des Auslösertyps wurde von 2 zu 255 geändert.
20	Der lokale Kommunikationstreiber ist mit der MSG-Anweisung nicht kompatibel.
21	In einem Konfigurationsparameter eines lokalen Kanals liegt ein Fehler vor.
22	Die Adressierung für Target oder Local Bridge überschreitet den zulässigen Adressbereich für Knoten.
33	Eine MSG-Datei enthält einen unzulässigen Parameter.
54	Verlorenes Modem.
55	Die Meldung weist eine Zeitüberschreitung im lokalen Prozessor auf. Zeitüberschreitung in der Verbindungsschicht.
217	Die Meldung wurde durch den Benutzer abgebrochen.
129	Unzulässige Funktion.
130	Unzulässige Datenadressierung.
131	Unzulässiger Datenwert.
132	Fehler im Slave-Gerät.
133	Rückmeldung.
134	Das Slave-Gerät ist beschäftigt.
135	Negative Rückmeldung.
136	Speicherparitätsfehler.
137	Ungültige Antwort.
255	Der Kanal wurde geschlossen.

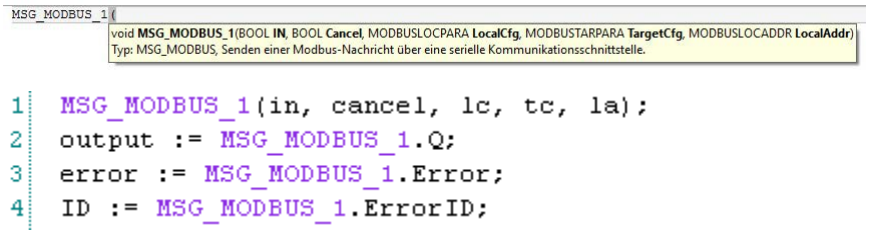
Beispiel für ein MSG_MODBUS-Funktionsblockdiagramm



Beispiel für einen MSG_MODBUS-Kontaktplan



Beispiel für einen strukturierten MSG_MODBUS-Text



Datentyp „MODBUSLOCPARA“

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp MODBUSLOCPARA zu bestimmen.

Parameter	Datentyp	Beschreibung
Channel	UINT	Nummer der seriellen Schnittstelle für Micro800 PLC: • 2 für den integrierten seriellen Anschluss oder • 5-9 für installierte Plug-Ins an den Schächten 1 bis: • 5 für Schacht 1 • 6 für Schacht 2 • 7 für Schacht 3 • 8 für Schacht 4 • 9 für Schacht 5
TriggerType	USINT	Steht für eines der Folgenden: • 0: Meldung einmal ausgelöst (wenn IN von „False“ auf „True“ wechselt) • 1: Meldung kontinuierlich ausgelöst, wenn IN = True ist • Sonstiger Wert: Reserviert
Cmd	USINT	Steht für eines der Folgenden: • 01: Read Coil Status (0xxxx) • 02: Read Input Status (1xxxx) • 03: Read Holding Registers (4xxxx) • 04: Read Input Registers (3xxxx) • 05: Write Single Coil (0xxxx) • 06: Write Single Register (4xxxx) • 15: Write Multiple Coils (0xxxx) • 16: Write Multiple Registers (4xxxx) • Andere Unterstützung von benutzerdefinierten Befehlen. Unterstützung von benutzerdefinierten MODBUSLOCPARA-Befehlen: Auch benutzerdefinierte Befehle im Bereich 0-255, die noch keinem Modbus-Befehl zugewiesen sind, werden unterstützt. Wenn ein benutzerdefinierter Befehl verwendet wird, enthält LocalCfg:ElementCnt die Anzahl der empfangenen Bytes. Die Antwort wird in den lokalen Adressdaten empfangen und überschreibt die Abfragedaten. • Beispiel für CMD=0x2B • Local Address Data 1:0x0E, READ_DEVICE_ID_MEI • Local Address Data 2:0x01, READ_DEV_ID_BASIC • Local Address Data 3:0x00, Read Vendor Object

Parameter	Datentyp	Beschreibung
ElementCnt	UINT	Grenzen • Für das Lesen von Spulen oder diskreten Eingängen (Read Coil/Discrete Inputs): 2000 Bits • Für das Lesen analoger Ausgänge (Read Register): 125 Wörter • Für das Schreiben digitaler Ausgänge (Write Coil): 1968 Bits • Für das Schreiben analoger Ausgänge (Write Register): 123 Wörter

Auslösen einer MSG_MODBUS-Nachricht

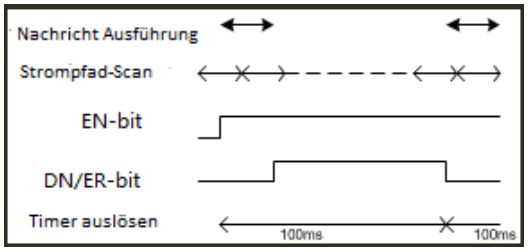
Eine Modbus-Nachricht kann regelmäßig ausgelöst werden, indem dem Auslösertyp-Parameter ein von null verschiedener Wert zugewiesen wird.

In dieser Tabelle wird das Verhalten des TriggerType-Parameters bei Verwendung mit dem MSG_MODBUS auf seite 177-Funktionsblock beschrieben.

Aktion	Ergebnisse
Nachricht ist aktiviert	Auslösertimer startet
Auslösertimer läuft ab, bevor die Nachricht abgeschlossen ist	Die Nachricht wird sofort im nächsten Kontaktplan-Scanzyklus ausgelöst.
Die Nachricht wird abgeschlossen, bevor die Auslöserzeit abläuft	Die Nachricht wird ausgelöst, wenn die Auslöserzeit abläuft.

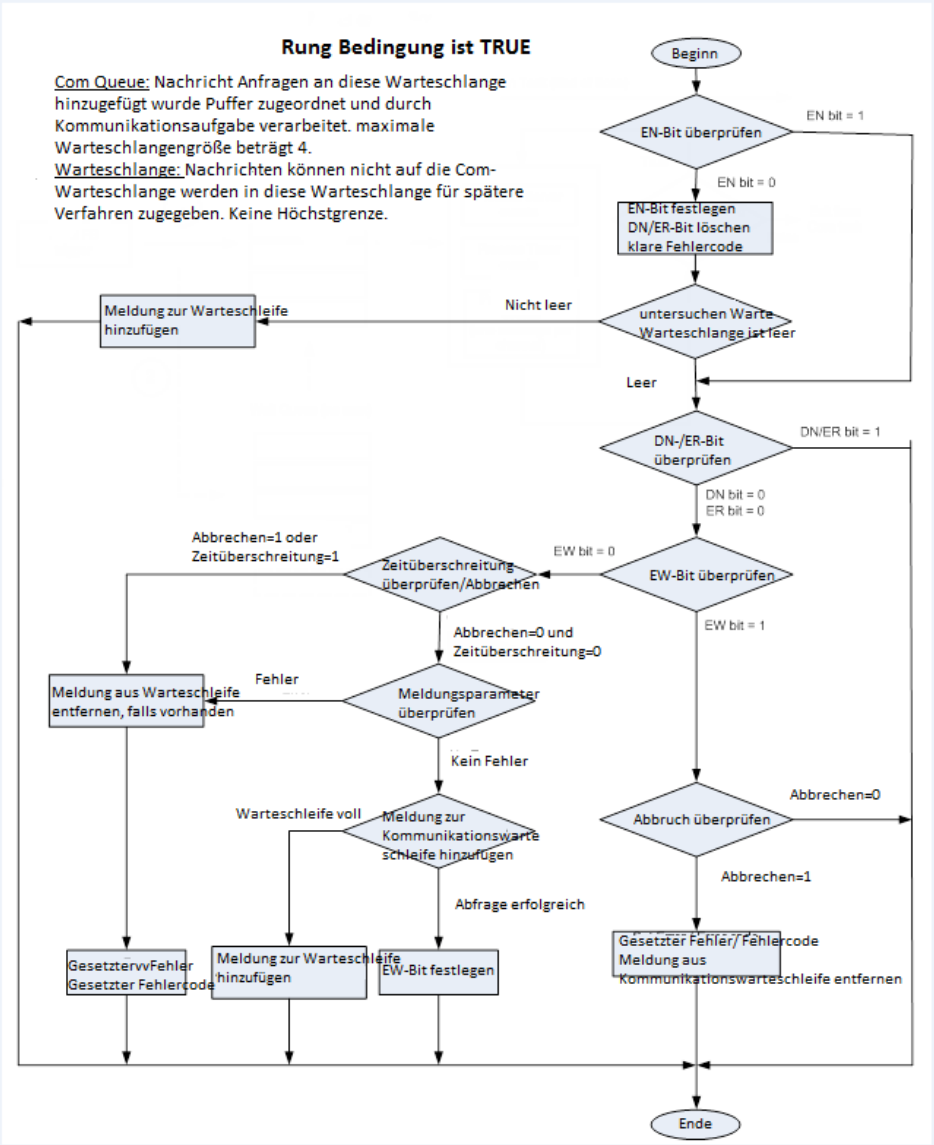
Beispiel: Auslösen einer Nachricht

Im folgenden Beispiel wird der Auslösertyp-Wert auf 100 gesetzt.



Nachrichtenausführungsprozess (Strompfad = TRUE)

In dem folgenden Prozessdiagramm werden die Nachrichtenanweisungsereignisse beschrieben, die auftreten, wenn der Strompfadzustand „True“ ist.



Com-Warteschlange: Nachrichtanforderungen, die zur Com-Warteschlange hinzugefügt wurden, wird ein Puffer zugewiesen. Sie werden von der Kommunikationsaufgabe verarbeitet. Die Größe der Warteschlange beträgt maximal 4.

Warteschlange: Nachrichten, die nicht zur Com-Warteschlange hinzugefügt werden können, werden zur Warteschlange hinzugefügt und zu einem späteren Zeitpunkt verarbeitet. Für die Warteschlange besteht keine Größenbeschränkung.

Datentyp „MODBUSTARPARA“

In der folgenden Tabelle wird der Datentyp „MODBUSTARPARA“ beschrieben:

Parameter	Datentyp	Beschreibung
Addr	UDINT	Zieldatenadresse (1-65536). Wird beim Senden um Eins verringert.

Parameter	Datentyp	Beschreibung
Node	USINT	Die Standard-Slave-Knotenadresse ist 1. Der Bereich liegt zwischen 1 und 247. Null ist die Modbus-Sendeadresse. Sie ist nur für Modbus-Schreibbefehle gültig (zum Beispiel 5, 6, 15 und 16).

MSG_MODBUS2 (MODBUS/TCP-Nachricht)

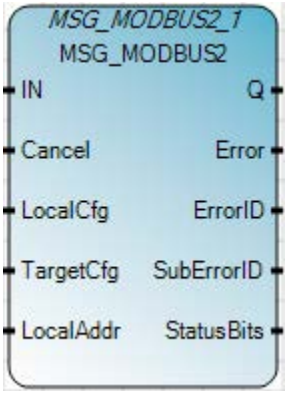
Sendet eine MODBUS-/TCP-Nachricht über einen Ethernet-Kanal.

Details zum Betrieb:

- In einem Scanvorgang können maximal vier Nachrichtenanforderungen pro Kanal verarbeitet werden. Bei Kontaktplan-Programmen werden Nachrichtenanforderungen am Ende des Kontaktplanscans ausgeführt.
- Wenn der MSG_MODBUS2 aktiviert ist, werden die Empfangspuffer für die Lesevorgänge bei Anstiegflanke von „Aktivieren“ gelöscht.
- Das Abbrechen der Ausführung der MSG_MODBUS2-Anweisung garantiert nicht, dass die ausgehende Nachrichtenanforderung abgebrochen wird, aber garantiert, dass die Antwort nicht verarbeitet wird.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro850 und Micro870. Für den Micro800-Simulator kann diese Anweisung zu einem Programm hinzugefügt werden. Sie dient aber lediglich als Platzhalter, um zu verhindern, dass die Anweisung bei der Controller-Änderung gelöscht wird.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangszustand von Strompfad. TRUE – Anstiegsflanke erkannt. Startet den Anweisungsblock unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE – Anstiegflanke nicht erkannt, Leerlauf.

Parameter	Parametertyp	Datentyp	Beschreibung
Cancel	Eingang	BOOL	TRUE - Die Ausführung des Anweisungsblocks wird abgebrochen. Das Abbrechen der Ausführung der MSG_MODBUS2-Anweisung garantiert nicht, dass die ausgehende Nachrichtenanforderung abgebrochen wird, aber garantiert, dass die Antwort nicht verarbeitet wird. FALSE – wenn IN = TRUE. Eingang abbrechen ist dominant.
LocalCfg	Eingang	MODBUS2LOCPARA	Definiert die Struktureingabe (lokales Gerät). Definieren der Eingangsstruktur für lokales Gerät mit dem Datentyp MODBUS2LOCPARA auf seite 186.
TargetCfg	Eingang	MODBUS2TARPARA	Definiert die Struktureingabe (Zielgerät). Definieren der Eingangsstruktur für Zielgerät mit dem Datentyp MODBUS2TARPARA auf seite 187.
LocalAddr	Eingang	MODBUSLOCADDR	Datentyp „MODBUSLOCADDR" ein 125-Wort-Array. Verwendeung von „LocalAddr": • Speichert für Lesen Befehle die durch den Modbus-Slave zurückgegebenen Daten (1-125 Wörter). • Zwichenspeichert für Schreibbefehle die an den Modbus-Slave zu sendenden Daten (1-125 Wörter).
Q	Ausgang	BOOL	Die Ausgaben dieser Anweisung werden asynchron vom Programm-Scan aktualisiert. Die Anweisung kann von Ausgabe Q nicht neu ausgelöst werden, da IN von der Kante ausgelöst wird. TRUE - MSG-Anweisung erfolgreich beendet. FALSE - MSG-Anweisung ist nicht beendet.
Error	Ausgang	BOOL	Zeigt an, dass ein Fehler erkannt wurde. TRUE - Ein Fehler ist aufgetreten. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler für diese Anweisung sind in Modbus2-Fehlercodes definiert.
SuberrorID	Ausgang	UINT	Wird zur Überprüfung von Statusbits verwendet • Bit 0: EN – Aktivieren • Bit 1: EW – Warten aktivieren • Bit 2: ST – Start • Bit 3: ER – Fehler • Bit 4: DN – Fertig Die anderen Bits sind reserviert.
StatusBits	Ausgang	UINT	SubError-Codewert, wenn der Fehler = TRUE. Bei erstmaliger oder erneuter Auslösung eines MSG, wird eine zuvor eingestellte SubErrorID gelöscht.

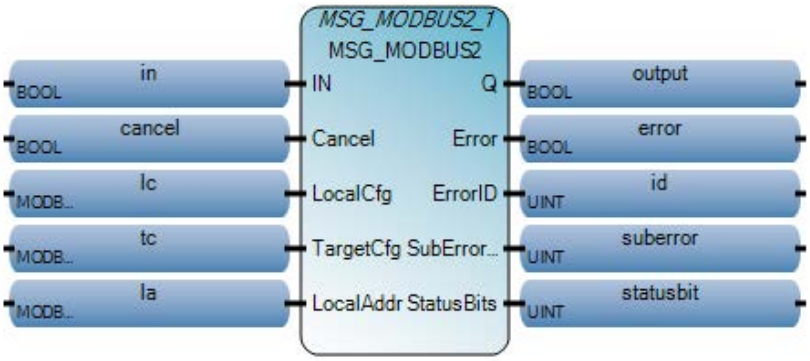
MSG_MODBUS2-Fehler und Teilfehlercodes

Wenn das ER-Bit gesetzt wird, zeigen die Felder „ErrorID" und „SubErrorID" die folgenden Fehlercodes an.

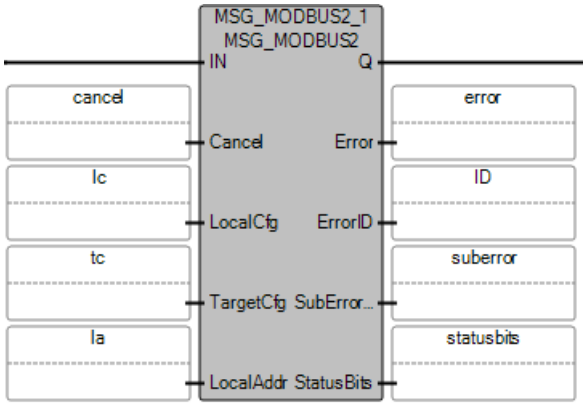
Fehler-ID	SubErrorID	Beschreibung
33	Fehler bei der Parameterkonfiguration	
	32	Kanalnummer fehlerhaft.
	37	Ungültige Elementanzahl.

Fehler-ID	SubErrorID	Beschreibung
	38	Unzulässige Datenadressierung.
55	**Fehler bei der Zeitüberschreitung**	
	112	Nachrichten-Zeitüberschreitung beim Warten in der Nachrichtenwarteschlange.
	113	Nachrichten-Zeitüberschreitung beim Warten auf eine Verbindung zur einzurichtenden Verbindungsschicht.
	114	Nachrichten-Zeitüberschreitung beim Warten auf Übertragung auf die Verbindungsschicht.
	115	Nachrichten-Zeitüberschreitung beim Warten auf eine Antwort von der Verbindungsschicht.
69	**Fehlercodes in Verbindung mit Serverantwort-Formaten.**	
208	Für das Netzwerk ist keine IP-Adresse konfiguriert.	
209	Maximale Anzahl der genutzten Verbindungen – keine Verbindungen verfügbar.	
210	Ungültige Internetadresse bzw. Knotenadresse.	
217	Ausführung der Nachricht wurde vom Benutzer abgebrochen. (Abbruchparameter wurde auf TRUE gesetzt).	
222	Netzwerkverbindung konnte vor der Zeitüberschreitung nicht hergestellt werden.	
255	Kanal ist heruntergefahren oder Neukonfiguration läuft. Der Fehlercode tritt unmittelbar nach dem Hochfahren auf, bis eine Verbindung eingerichtet wurde. Dieses Verhalten ist normal. Es kann auch dann auftreten, wenn ein Ethernet-Kabel getrennt wurde oder eine IP-Adresse nicht erkannt wird.	
	Slave-Antwort-Fehlercodes	
129	Unzulässige Funktion-Code	
130	Unzulässige Datenadressierung	
131	Ungültiger Datenwert	
132	Serverfehler	
133	Rückmeldung	
134	Negative Rückmeldung	
136	Speicherparitätsfehler	
137	Ungültige Antwort-Fehlercode. Den tatsächlichen Fehlercode finden Sie in der SubErrorID.	

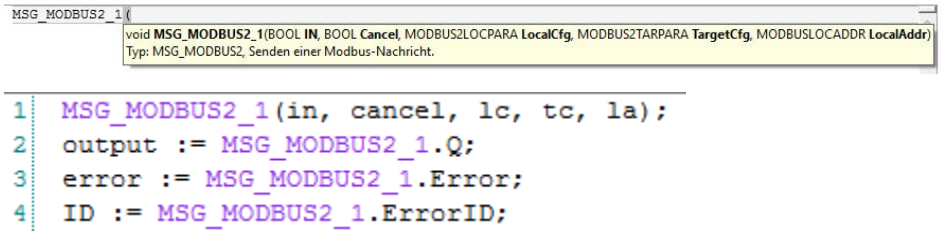
Beispiel für ein MSG_MODBUS2-Funktionsblockdiagramm



Beispiel für einen MSG_MODBUS2-Kontaktplan



Beispiel für einen strukturierten MSG_MODBUS2-Text



Datentyp MODBUS2LOCPARA

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp MODBUS2LOCPARA zu bestimmen.

Parameter	Datentyp	Beschreibung
Channel	UINT	Lokale Ethernet-Anschlussnummer: • 4 integrierte Ethernet-Anschlüsse für Micro850 und Micro820.
TriggerType	UDINT	Meldungsauslösetyp: • 0: Meldung einmal ausgelöst (wenn IN von „False“ auf „True“ wechselt) • 1 bis 65535 - Zyklischer Auslösewert in Millisekunden. Meldung wird regelmäßig ausgelöst, wenn IN „True“ ist und die Ausführung der vorherigen Meldung abgeschlossen ist. • Legen Sie hierfür den Wert 1 fest, wenn Meldungen so schnell wie möglich ausgelöst werden sollen. Siehe unten MSG_MODBUS2-Nachrichtenauslösung.
Cmd	USINT	Modbus-Befehl: • 01: Read Coil Status (0xxxx) • 02: Read Input Status (1xxxx) • 03: Read Holding Registers (4xxxx) • 04: Read Input Registers (3xxxx) • 05: Write Single Coil (0xxxx) • 06: Write Single Register (4xxxx) • 15: Write Multiple Coils (0xxxx) • 16: Write Multiple Registers (4xxxx) • Andere Unterstützung von benutzerdefinierten Befehlen Unterstützung von benutzerdefinierten MODBUS2LOCPARA-Befehlen: Auch benutzerdefinierte Befehle im Bereich 0-255, die noch keinem Modbus-Befehl zugewiesen sind, werden unterstützt. Wenn ein benutzerdefinierter Befehl verwendet wird, enthält LocalCfg:ElementCnt die Anzahl der empfangenen Bytes.

Parameter	Datentyp	Beschreibung
		Die Antwort wird in den lokalen Adressdaten empfangen und überschreibt die Abfragedaten. Beispiel für CMD=0x2B: • Local Address Data 1:0x0E, READ_DEVICE_ID_MEI • Local Address Data 2:0x01, READ_DEV_ID_BASIC • Local Address Data 3:0x00, Read Vendor Object
ElementCnt	UINT	Grenzen • Für das Lesen von Spulen oder diskreten Eingängen (Read Coil/Discrete Inputs): 2000 Bits • Für das Lesen analoger Ausgänge (Read Register): 125 Wörter • Für das Schreiben digitaler Ausgänge (Write Coil): 1968 Bits • Für das Schreiben analoger Ausgänge (Write Register): 123 Wörter

MSG_MODBUS2-Nachrichtenauslösung

Eine Modbus-Nachricht kann regelmäßig ausgelöst werden, indem dem Auslösertyp-Parameter ein von null verschiedener Wert zugewiesen wird.

In dieser Tabelle wird beschrieben, was geschieht, wenn ein TriggerType-Parameter mit dem MSG_MODBUS2 auf seite 183-Funktionsblock verwendet wird.

Aktion	Ergebnisse
Nachricht ist aktiviert.	Auslösertimer startet.
Auslösertimer läuft ab, bevor die Nachricht abgeschlossen ist.	Die Nachricht wird sofort im nächsten Kontaktplan-Scanzyklus ausgelöst.
Die Nachricht wird abgeschlossen, bevor die Auslöserzeit abläuft.	Die Nachricht wird ausgelöst, wenn die Auslöserzeit abläuft.

Datentyp MODBUS2TARPARA

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp MODBUS2TARPARA zu bestimmen.

Parameter	Datentyp	Beschreibung
Addr	UDINT	Modbus-Datenadresse des Zielgeräts: • 1 - 65536 • Wird beim Senden um Eins verringert. • Firmware setzt eine Low-Word-Adresse ein, wenn der Adresswert größer als 65536 ist.
NodeAddress[4]	USINT	IP-Adresse des Zielgeräts. Die IP-Adresse muss eine gültige Unicastadresse und darf nicht 0 sein, Multicast, Sendung, lokale Adresse oder Loop-back-Adresse (127.x.x.x). So geben Sie beispielsweise 192.168.2.100 an: • NodeAddress[0]=192 • NodeAddress[1]=168 • NodeAddress[2]=2 • NodeAddress[3]=100
Port	UINT	Ziel-TCP-Portnummer. Standard-Modbus/TCP-Port lautet 502. 1 - 65535 Setzen Sie ihn auf 0, um den Standardwert 502 zu verwenden.

Parameter	Datentyp	Beschreibung
UnitId	USINT	Einheit-Bezeichner. Wird eingesetzt, um mit Slave-Geräten über eine Modbus-Bridge zu kommunizieren. Weitere Einzelheiten finden Sie in den Angaben zum Modbus. Beachten Sie, dass der Micro800 nicht versuchen soll, diesen Wert zu prüfen. 0 - 255 Setzen Sie den Wert auf 255, wenn das Zielgerät keine Bridge ist.
MsgTimeOut	UDINT	Zeitüberschreitung der Nachricht (in Millisekunden). Wartedauer zum Antworten auf einen initiierten Befehl. • 250-10.000 • Setzen Sie ihn auf 0, um den Standardwert 3.000 zu verwenden. • Ein Wert, der kleiner ist als 250 (Minimum), wird auf 250 gesetzt. • Ein Wert, der größer ist als 10.000 (Maximum), wird auf 10.000 gesetzt. Siehe Modbus-/TCP-Timer zur Zeitüberschreitung der Nachricht.
ConnTimeOut	UDINT	Zeitüberschreitung des TIP-Verbindungsaufbaus (in Millisekunden). Wartedauer zum Aufbauen einer erfolgreichen TCP-Verbindung mit dem Zielgerät. • 250-10.000 • Setzen Sie ihn auf 0, um den Standardwert 5.000 zu verwenden. • Ein Wert, der kleiner ist als 250 (Minimum), wird auf 250 gesetzt. • Ein Wert, der größer ist als 10.000 (Maximum), wird auf 10.000 gesetzt. Siehe Modbus-/TCP-Timer zur Zeitüberschreitung der Nachricht.
ConnClose	BOOL	Verhalten der TCP-Verbindungstrennung. • True – Die TCP-Verbindung nach Abschluss der Nachricht trennen. • False – Die TCP-Verbindung nach Abschluss der Nachricht nicht trennen [Standard]. Siehe Modbus-/TCP-Nachrichtenverbindungen.

Modbus-/TCP-Timer zur Zeitüberschreitung der Nachricht

In dieser Tabelle wird das Verhalten von MsgTimeOut und ConnTimeOut basierend auf Nachrichtenanforderungen und Status beschrieben.

Aktion	Ergebnisse
Nachricht ist aktiviert.	Aktiviert den MsgTimeOut-Timer.
TCP-Verbindung wird angefordert.	Aktiviert den ConnTimeOut-Timer.
ConnTimeOut-Timer ist aktiv.	Deaktiviert den MsgTimeOut-Timer.
Verbindungsanforderung ist abgeschlossen.	Reaktiviert den MsgTimeOut-Timer.

Modbus-/TCP-Nachrichtenverbindungen

Modbus-TCP-Client unterstützt maximal 16 Verbindungen. In dieser Tabelle wird das Verhalten der Modbus-/TCP-Verbindung beschrieben.

Szenario	Ergebnisse
Nachrichtenanforderung ist aktiviert und es besteht keine Verbindung zum Ziel.	Wenn keine Verbindung mit dem Ziel besteht, wird eine neue Verbindung aufgebaut. Wenn bereits eine Verbindung mit dem Ziel besteht, wird die vorhandene Verbindung verwendet.
Die Nachrichtenausführung wurde abgeschlossen und ConnClose wird auf „True“ gesetzt.	Es besteht nur eine Verbindung zum Ziel, die Verbindung wird geschlossen. Falls mehr als eine Verbindung zum Ziel besteht, wird die Verbindung getrennt, sobald die letzte Nachrichtenausführung abgeschlossen wurde.
Die Nachrichtenausführung wurde abgeschlossen und ConnClose wird auf False gesetzt.	Die Verbindung wurde nicht getrennt.

Szenario	Ergebnisse
Die Verbindung ist mit keiner aktiven Nachricht verknüpft und bleibt für den Zeitraum, der im ConnTimeOut-Parameter angegeben ist, inaktiv.	Die Verbindung wurde getrennt.
Wechsel des Controllers von einem Ausführungsmodus (Betrieb, Remotelauf, Remotetestlauf, Remote-Kontaktplan) in einen Nicht-Ausführungsmodus.	Alle aktiven Verbindungen wurden getrennt.

MSG_PCCC (Nachricht für programmierbare Controller-Kommunikationsbefehle)

Sendet eine PCCC-Nachricht über einen Ethernet-Kanal oder einen seriellen Anschluss.

Die unterstützten Befehle sind: SLC-geschriebenes Read and Write

Details zum Betrieb:

- Wenn die PCCC-Nachricht über einen Ethernet-Kanal gesendet wird und sich mehr als vier Funktionsblockanforderungen in der Warteschlange für CIP_GENERIC_FB, CIP_SYMBOLIC oder PCCC_MSG_FB befinden, können bis zu vier Nachrichten pro Kanal in einer Abfrage verarbeitet werden.

  Bei Kontaktplan-Programmen werden Nachrichtenanforderungen am Ende des Kontaktplanscans ausgeführt.
- Wenn die PCCC-Nachricht über einen seriellen Anschluss gesendet wird und sich mehr als acht Funktionsblockanforderungen in der Warteschlange für die serielle Kommunikation befinden, können in einer Abfrage insgesamt acht Nachrichten auf allen Kanälen verarbeitet werden.

  Bei Kontaktplan-Programmen werden Nachrichtenanforderungen am Ende des Kontaktplanscans ausgeführt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

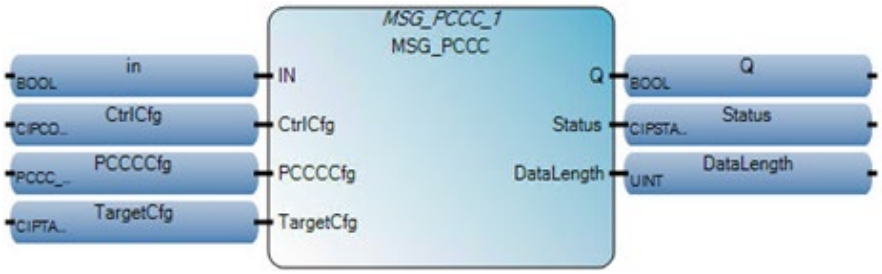
Diese Anweisung gilt für Micro870 L70E-Controller.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

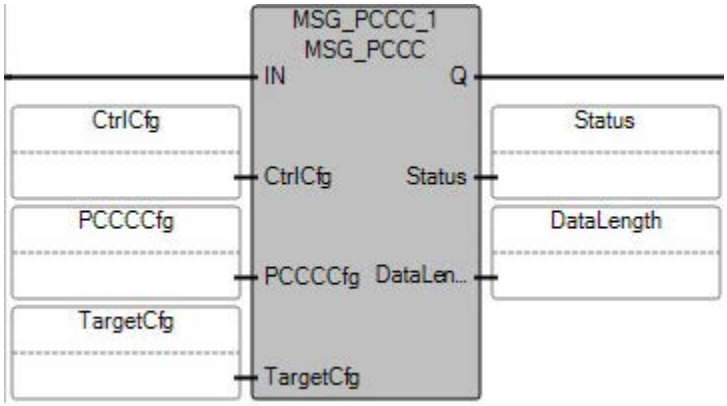
Parameter	Parametertyp	Datentyp	Beschreibung

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangszustand von Strompfad. TRUE – Anstiegflanke erkannt. Startet den Funktionsblock unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE – Anstiegflanke nicht erkannt, nicht gestartet.
CtrlCfg	Eingang	CIPCONTROLCFG	Konfiguration der Ausführungssteuerung für Funktionsblöcke. Verwenden Sie den Datentyps CIPCONTROLCFG auf seite 191, um „CtrlCfg" zu definieren.
PCCCCfg	Eingang	PCCC_CFG	Die PCCC-Befehle und -Adresse. Definieren Sie den Datentyp PCCCCFG auf seite 192, um PCCCCfg zu definieren.
TargetCfg	Eingang	CIPTARGETCFG	Konfiguration des Zielgeräts. Verwenden Sie den Datentyp TARGETCFG auf seite 193, um „TargetCfg" zu definieren.
Q	Ausgang	BOOL	True – Die MSG-Anweisung ist abgeschlossen. False – Die MSG-Anweisung ist nicht abgeschlossen.
Status	Ausgang	CIPSTATUS	Der Ausführungsstatus des Funktionsblocks. Wenn eine MSG-Anweisung ausgelöst oder erneut ausgelöst wird, werden alle Elemente in „Status" zurückgesetzt. Die Status-Ausgabe wird im Datentyp „CIPSTATUS" auf seite 195 definiert.
DataLength	Ausgang	UINT	Die Anzahl der Datenbytes für die CIPPCCC-Nachrichtenantwort. Wenn eine MSG-Anweisung ausgelöst oder erneut ausgelöst wird, wird DataLength auf 0 zurückgesetzt. DataLength wird unterstützt, wenn eine PCCC-Nachricht über einen Ethernet-Kanal gesendet wird.

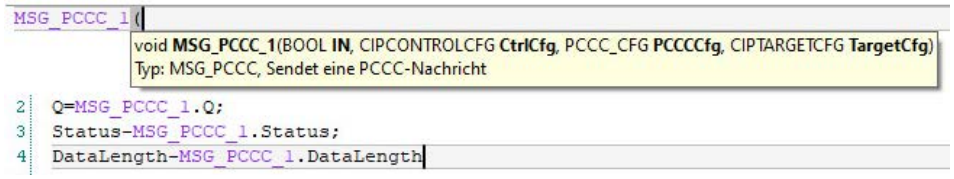
Beispiel für ein MSG_PCCC-Funktionsblockdiagramm



Beispiel für einen MSG_PCCC-Kontaktplan



Beispiel für den strukturierten MSG_PCCC-Text



Datentyp „CIPCONTROLCFG“

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „CIPCONTROLCFG“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Cancel	BOOL	TRUE - Die Ausführung des Funktionsblocks wird abgebrochen. Bit wird bei Aktivieren der Nachricht gelöscht.
TriggerType	UDINT	Steht für eines der Folgenden: • 0: Meldung einmal ausgelöst (wenn IN von „False“ auf „True“ wechselt) • 1 bis 65535: Zyklischer Auslösewert in Millisekunden. Meldung wird kontinuierlich ausgelöst, wenn IN = True ist. Legen Sie den Wert auf 1 fest, um die Meldung so schnell wie möglich auszulösen.
StrMode	USINT	Für die spätere Verwendung reserviert.

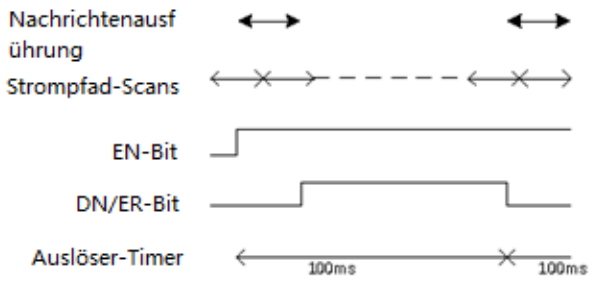
PCCC-Nachrichtenauslösung

Eine Nachricht wird regelmäßig ausgelöst, indem dem TriggerType-Parameter ein Nicht-Nullwert zugewiesen wird.

Aktion	Ergebnisse
Nachricht ist aktiviert	Auslösertimer startet.
Auslösertimer läuft ab, bevor die Nachricht abgeschlossen ist	Die Nachricht wird sofort im nächsten Kontaktplan-Scanzyklus ausgelöst.
Die Nachricht wird abgeschlossen, bevor die Auslöserzeit abläuft	Die Nachricht wird ausgelöst, wenn die Auslöserzeit abläuft.

Beispiel: Auslösen einer Nachricht

Im folgenden Beispiel wird der Auslösertyp-Wert auf 100 gesetzt.



Datentyp PCCCCFG

Verwenden Sie diese Tabelle bei der Definition der Parameter für den Datentyp „PCCCCFG“.

Parameter	Datentyp	Beschreibung
Service	USINT	Dienstcode: • 0 - Lesen (Standard) • 1 - Schreiben
SrcAdr	STRING	Quelladresse: • Quelldateiname, wenn der Wert von „Source“ 0 ist Syntax: *<Dateityp><Dateinummer>:<Elementnummer>* Beispiel: N7:1 Die unterstützten Dateitypen sind: • N (Ganzzahl) • F (Fließkomma) • B (Bool) • L (Lang) • ST (Zeichenfolge) • Tag-Variablenname, wenn der Wert von „Service“ 1 ist • Beispiel eines 1-D-Arrays: testarrayVariable1*[x]*, wobei x die Position des Arrays im Array-Bereich ist. Eindimensionale Arrays werden unterstützt. • Beispiel einer normalen Variable oder eines 1-D-Arrays, dessen Index bei 0 beginnt: VariableName1.
NumOfElements	UINT	Die Anzahl der Elemente: • Wenn „Service“ 0 ist, darf die maximale Anzahl an Elementen (Elementwert * von einzelnem Element belegte Bytes) 244 nicht überschreiten. • Wenn „Service“ 1 ist, darf die maximale Anzahl an Elementen (Elementwert * von einzelnem Element belegte Bytes) 232 nicht überschreiten.
DstAdr	STRING	Zieladresse: • Quelldateiname, wenn der Wert von „Service“ 1 ist Syntax: *<Dateityp><Dateinummer>:<Elementnummer>* Beispiel: N7:1 Die unterstützten Dateitypen sind: • N (Ganzzahl) • F (Fließkomma) • B (Bool) • L (Lang)

Parameter	Datentyp	Beschreibung
		• ST (Zeichenfolge) • Tag-Variablenname, wenn der Wert von „Service“ 0 ist • Beispiel eines 1-D-Arrays: testarrayVariable2*[x]* wobei x die Position des Arrays im Array-Bereich ist. Eindimensionale Arrays werden unterstützt. • Beispiel einer normalen Variable oder eines 1-D-Arrays, dessen Index bei 0 beginnt: VariableName2.

Datentyp TARGETCFG

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „TARGETCFG“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Pfad	STRING[80]	Pfad des mithilfe der Syntax *{"<Port>,<Knoten-/Steckplatzadresse>"}* definierten Ziels Definition der Portnummer: • 2 für den integrierten seriellen Port oder • 4 für den integrierten Ethernet-Port • 5 – 7 für die in den Steckplätzen 1 – 3 installierten SERIALISOL-Plug-in-Module der Serie 2080
CipConnMode	USINT	CIP-Verbindungstyp. • 1 - Klasse-3-Verbindung Der Parameter muss für Ethernet- und serielle PCCC-Nachrichten auf "1" gesetzt werden.
Ucmm-Zeitüberschreitung	UDINT	Getrennte Nachricht – Zeitüberschreitung (in Millisekunden). Die Wartezeit auf eine Antwort für getrennte Nachrichten einschließlich der Verbindungseinrichtung für verbundene Nachrichten. • Gültige Werte: 250 - 10.000 ms. • Setzen Sie ihn auf 0, um den Standardwert von 3.000 ms (3 Sekunden) zu verwenden. • Ein auf weniger als 250 ms gesetzter Wert wird auf 250 ms (Minimum) gesetzt. • Ein auf mehr als 10.000 ms gesetzter Wert wird auf 10.000 ms (Maximum) gesetzt. Der Parameter gilt für EtherNet/IP.

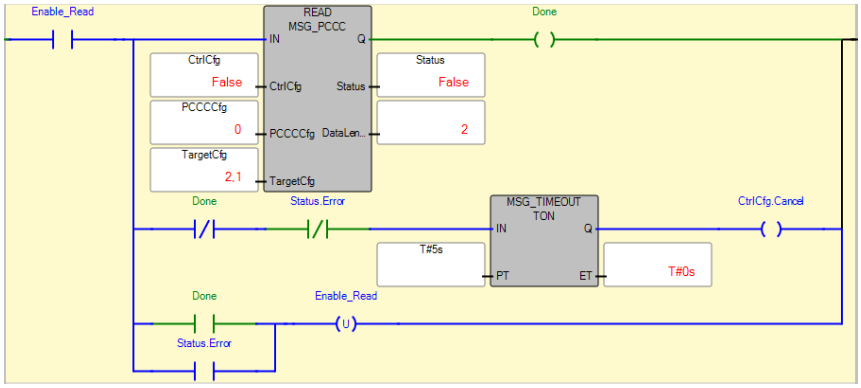
Parameter	Datentyp	Beschreibung
ConnMsgTimeout	UDINT	Zeitüberschreitung bei Klasse-3-Verbindung (in Millisekunden). Wartedauer auf Antwort auf verbundene Nachrichten. Die Verbindung wird bei Ablauf der Zeitüberschreitung getrennt. • Gültige Werte: 800 - 10.000 ms. • Setzen Sie ihn auf 0, um den Standardwert von 10.000 ms (10 Sekunden) zu verwenden. • Ein auf weniger als 800 gesetzter Wert wird auf 800 ms (Minimum) gesetzt. • Ein auf mehr als 10.000 ms gesetzter Wert wird auf 10.000 ms (Maximum) gesetzt. Der Parameter gilt für EtherNet/IP und seriell.
ConnClose	BOOL	Verhalten beim Trennen der Verbindung: • TRUE - Die Verbindung wird bei Abschluss der Nachricht getrennt. • FALSE - Verbindung bei abgeschlossener Nachricht nicht trennen (Standard). Der Parameter gilt für EtherNet/IP.

Zielpfad-Beispiel

In der folgenden Tabelle werden in einem Zielpfadstring verwendete Beispielwerte und die Ergebnisse für jeden String erläutert.

Zeichenfolgenelement	Beschreibung
„6,1“	Über das in Steckplatz 2 am Knoten an der Adresse 1 installierte Plug-in-Modul für den seriellen Anschluss. Beachten Sie, dass die Plug-in-Steckplätze 1 – 3 den Ports 5 – 7 entsprechen.
„4,192.168.1.100“	Über Port 4, integrierter Micro870-Ethernet-Anschluss, wird der Knoten bei 192.168.1.100 erreicht.
Lokaler Anschluss	Lokaler Anschluss zum Versenden der Nachricht. Der Anschluss sollte ein aktiver EtherNet/IP- oder serieller CIP-Anschluss sein. USB-Anschlüsse werden nicht unterstützt.
Zieladresse	• Bei EIP gibt die Zieladresse die IP-Adresse des Ziels an. Bei der IP-Adresse muss es sich um eine Unicast-Adresse handeln. Sie darf nicht 0 oder keine Multicast-, Broadcast-, lokale oder Loopback-Adresse (127.x.x.x) sein. • Beim seriellen CIP-Anschluss gibt die Zieladresse die Knotenadresse des Ziels an. Der unterstützte Wertebereich geht von 0 bis 254.

WICHTIG	Wenn eine MSG_PCCC-Anweisung an einem seriellen Port ausgelöst wird, muss der Benutzer insbesondere bei der Verwendung von Funkmodems einen Timer für die Zeitüberschreitung von Nachrichten integrieren, um einen Fehler der Nachrichtenanweisung auszulösen, falls eine fehlerfreie Antwort nicht innerhalb einer angemessenen Zeit empfangen wurde. Die Timer-Sollzeit sollte der Benutzer entsprechend seiner Anwendung festlegen. Nachstehend finden Sie ein Beispiel.



Datentyp „CIPSTATUS“

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „CIPSTATUS“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Error	BOOL	Dieses Bit wird auf TRUE gesetzt, wenn bei der Ausführung des Funktionsblocks eine Fehlerbedingung auftritt.
ErrorID	UINT	Fehlercode-Wert. Weitere Informationen finden Sie in der Fehlercodetabelle auf seite 197
SubErrorID	UINT	Wert des Subfehlercodes.
ExtErrorID	UINT	Code-Wert des Statusfehlers mit CIP-Erweiterung.
StatusBits	UINT	Dieser Parameter kann zur Überprüfung von Steuerbits: • Bit 0: EN – Aktivieren • Bit 1: EW – Warten aktivieren • Bit 2: ST – Start • Bit 3: ER – Fehler • Bit 4: DN – Fertig • Bit 5: CIPCONN - CIP-Verbindung trennen • Bit 6: EIPSESS - EIP-Sitzung schließen Bit 5 und Bit 6 sind für Ethernet/IP reserviert, und andere Bits sind reserviert. StatusBits sind für CIPSTATUS-Statusbits auf seite 195 definiert.

CIPSTATUS-Statusbits

Die CISTATUS-Statusbits werden anhand des Status des Nachrichtenausführung, der Kommunikationspuffer und der Strompfadbedingungen festgelegt.

Bit-Name	Beschreibung	Verhalten
EN	Aktivieren	Ist eingestellt, wenn der Strompfad auf „True“ wechselt und bleibt eingestellt, bis das DN-Bit oder das ER-Bit eingestellt wird oder der Strompfad auf False wechselt.
EW	Warten aktivieren	Ist aktiviert, wenn der Kommunikationspuffer für die Nachrichtenanforderung zugewiesen ist. Wird gelöscht, wenn das ST-Bit eingestellt wird.
ST	Start	Ist eingestellt, wenn die Nachricht übertragen wurde und auf eine Antwort wartet. Wird gelöscht, wenn das DN-Bit eingestellt wird.

Bit-Name	Beschreibung	Verhalten
ER	Fehler	Wird eingestellt, wenn die Nachricht nicht übertragen werden konnte. Es wird ein Fehlercode in die Fehler-ID eingetragen. Die Werte für das ER-Bit und den Fehlercode werden gelöscht, wenn der Strompfad beim nächsten Mal von False auf True wechselt.
DN	Fertig	Wird eingestellt, wenn die Nachricht erfolgreich übertragen wurde. Die Werte für das DN-Bit werden gelöscht, wenn der Strompfad zum nächsten Mal von False auf True wechselt.
CIPCONN	Fertig	Wird beim Trennen der CIP-Verbindung für die Kommunikation eingestellt. Das CIPCONN-Bit ist zutreffend, wenn ConnClose = True, in anderen Fällen ist das CIPCONN-Bit „False". Das Bit wird nur für Ethernet verwendet.
EIPSESS	Fertig	Wird eingestellt, wenn die Kapselungs-CIP-Sitzung für die Kommunikation beendet wird. Das EIPSESS-Bit ist zutreffend, wenn ConnClose = True, in anderen Fällen ist das EIPSESS-Bit „False". Das Bit wird nur für Ethernet verwendet.

Tipp: Wenn die Nachricht aktiviert (EN-Bit gesetzt) und nicht abgeschlossen ist (DN-Bit nicht gesetzt), wird die Ausführung der Nachricht beendet und das ER-Bit gesetzt, wenn der Abbrechen-Parameter gesetzt ist.

CIP/EIP-Nachrichtenverbindungen

Für die Client-Nachrichtenausführung werden maximal 16 CIP- (Klasse 3) und 16 EIP-Verbindungen unterstützt. In der folgenden Tabelle wird das Verhalten der CIP-/EIP-Verbindung beschrieben. Alle folgenden Szenarien gelten für Ethernet.

Szenario	Ergebnisse
Nachrichtenanforderung ist aktiviert und CipConnMode=1.	Wenn keine Verbindung zum Ziel besteht, wird eine CIP-Verbindung aufgebaut. Wenn der lokale Port der Nachricht Ethernet ist und keine EIP-Verbindung zum Ziel vorhanden ist, wird vor dem Einrichten einer CIP-Verbindung eine EIP-Verbindung eingerichtet.
Nachrichtenverbindung ist aktiviert, CipConnMode=0 und der lokale Port der Nachricht ist Ethernet.	Wenn keine EIP-Verbindung mit dem Ziel besteht, wird eine EIP-Verbindung aufgebaut. Wenn mehrere Nachrichten für das gleiche Ziel bestimmt sind, müssen die Nachrichten dieselbe Verbindung verwenden.
Wenn ConnClose auf True steht, werden die Nachrichtenverbindung und die EIP-Sitzung nach Abschluss der Nachrichtenausführung geschlossen.	Wenn mehrere Nachrichten dieselbe Verbindung nutzen, wird die Verbindung nach Abschluss der letzten Nachricht getrennt.
Die Nachrichtenausführung wurde abgeschlossen und ConnClose wird auf False gesetzt.	Die Verbindung wird nach Abschluss der Nachrichtenausführung nicht getrennt.

Szenario	Ergebnisse
Eine CIP- oder EIP-Verbindung, die nicht mit einer aktiven Nachricht verknüpft ist, wird getrennt, wenn sie für x Sekunden im Leerlauf ist, wobei x ein konfigurierbarer Wert für die Zeitüberschreitung bei Kapselungsinaktivität ist, der mit dem CIP-Dienst festgelegt werden kann.	Informationen bezüglich des CIP-Diensts finden Sie in den CIP-Spezifikationen, Ausgabe II für TCP/IP-Objekte.
Eine CIP-Verbindung wird geschlossen, und alle EIP-Verbindungen werden ebenfalls geschlossen.	Wenn mehrere CIP-Verbindungen auf die gleiche EIP-Verbindung zugreifen, wird die EIP-Verbindung geschlossen, nachdem alle zugeordneten CIP-Verbindungen geschlossen wurden.
Wechsel des Controllers von einem Ausführungsmodus (Betrieb, Remotelauf, Remotetestlauf, Remote-Kontaktplan) in einen Nicht-Ausführungsmodus.	Alle aktiven Verbindungen wurden getrennt.

Nachricht-Zeitüberschreitung

In der folgenden Tabelle wird beschrieben, wie sich die Timer für die Parameter inUcmmTimeout und ConnMsgTimeout in Abhängigkeit von Nachrichtenanforderungen und Status verhalten.

Aktion	Ergebnisse
Nachricht ist aktiviert	UcmmTimeout-Timer ist aktiviert
CIP-Verbindung wird angefordert	ConnMsgTimeout-Timer ist aktiviert
ConnMsgTimeout-Timer ist aktiv	UcmmTimeout-Timer ist deaktiviert Gilt für Ethernet/IP
Verbindungsanforderung ist abgeschlossen	UcmmTimeout-Timer ist reaktiviert Gilt für Ethernet/IP

MSG_PCCC-Fehlercodes

Verwenden Sie diese Tabelle bei der Bestimmung der Parameterwerte für die Felder ErrorID und SubErrorID, wenn das ER-Bit gesetzt ist.

ErrorID-Code	SubErrorID-Code	Beschreibung des Fehlercodes
30		SubErrorID steht für den Remote-Fehler von Status (STS) dar. Weitere Informationen finden Sie in Kapitel 8 des DF1 Protocol and Command Set Reference Manual.
31		SubErrorID steht für den Remote-Fehler von erweiterten Status (EXT STS) dar. Weitere Informationen finden Sie in Kapitel 8 des DF1 Protocol and Command Set Reference Manual.
32	PCCC-Konfigurationsfehler	
	1	Nicht unterstützter Dienstcode Die unterstützten Dienstcodes sind: • 0 - Lesen (Standard) • 1 - Schreiben
	2	Ungültige Quelladresse Korrekte Quelladressen finden Sie unter „SrcAdr" in der Tabelle des Datentyps PCCCCFG auf seite 192. Für den Datentyp „String" ist die unterstützte feste Zeichenfolgenlänge 82.
	3	Ungültige Zieladresse Korrekte Quelladressen finden Sie unter „DstAdr" in der Tabelle des Datentyps PCCCCFG auf seite 192. Für den Datentyp „String" ist die unterstützte feste Zeichenfolgenlänge 82.

ErrorID-Code	SubErrorID-Code	Beschreibung des Fehlercodes
	4	Ungültiger Quelldatentyp Die unterstützten Dateitypen sind: • N (Ganzzahl) • F (Fließkomma) • B (Bool) • L (Lang) • ST (Zeichenfolge)
	5	Ungültiger Zieldatentyp Die unterstützten Dateitypen sind: • N (Ganzzahl) • F (Fließkomma) • B (Bool) • L (Lang) • ST (Zeichenfolge)
	6	Der Dateityp und der Symbol-Datentyp sind nicht kompatibel. Der Fließkomma-Datentyp sendet Nachrichten nur zu oder empfängt sie von Fließkomma-Datentypen.
	41	Ungültiger CIP-Symbolname
	48	Die Array-Größe für die Eingabedaten des Funktionsblocks ist unzureichend. Stellen Sie sicher, dass die Datenpuffergröße die angeforderte Datenlänge überschreitet.
33	**Fehler bei der Parameterkonfiguration**	
	32	Kanalnummer fehlerhaft.
	36	CIP-Verbindungstyp wird nicht unterstützt. Vergewissern Sie sich, dass für PCCC-Nachrichten der Parameter "CipConnMode" auf "1" gesetzt ist.
	48	Ungültige Elementanzahl Der Bereich für die Elementanzahl liegt zwischen 1 und dem Höchstwert. Weitere Informationen finden Sie in der Tabelle „Elementanzahl“.
	49	Ungültiger Zielpfad.
	51	Unzulässige Nutzlast oder die Größe des übertragenden Datenarrays des Funktionsblocks ist für die CIP-Kommunikation zu groß.
	52	Ungültiger Wert für Segmenttyp
	53	Ungültiger Wert für UCCM-Zeitüberschreitung Wenn der Zeitüberschreitungswert der Kapselung kleiner als die Zeitüberschreitung für UCCM ist oder die Differenz zwischen Kapselungs- und UCCM-Zeitüberschreitung kleiner oder gleich 1 Sekunde ist, wird der Fehler gemeldet. Dieser Code gilt für Ethernet/IP.
	54	Ungültiger Wert für Verbindungs-Zeitüberschreitung Wenn der Zeitüberschreitungswert der Kapselung kleiner als der für die CONNECTED-Nachricht ist oder die Differenz zwischen Kapselungs- und CONNECTED-Nachrichten-Zeitüberschreitung <= 1 Sekunde ist, wird der Fehler gemeldet. Dieser Code gilt für EtherNet/IP.
	Fehler bei der Zeitüberschreitung	
55	112	Nachrichten-Zeitüberschreitung beim Warten in der Nachrichtenwarteschlange. Zur Vermeidung dieses Problems prüfen Sie, ob die Anzahl der ausführenden Funktionsblöcke für jeden Port gleich oder kleiner als vier ist.
	113	Nachrichten-Zeitüberschreitung beim Warten auf eine Verbindung zur einzurichtenden Verbindungsschicht. Zur Vermeidung dieses Problems müssen Sie Folgendes sicherstellen: • Das Kabel funktioniert und das Ziel ist vorhanden. • Der Controller unterstützt die Verarbeitung von maximal acht CIP-Client-Nachrichten.

ErrorID-Code	SubErrorID-Code	Beschreibung des Fehlercodes
	114	Nachrichten-Zeitüberschreitung beim Warten auf Übertragung auf die Verbindungsschicht. Zur Vermeidung dieses Problems stellen Sie sicher, dass der Controller die Verarbeitung von bis zu acht CIP-Client-Nachrichten unterstützt.
	115	Nachrichten-Zeitüberschreitung beim Warten auf eine Antwort von der Verbindungsschicht. Zur Vermeidung dieses Problems stellen Sie sicher, dass das Ziel vorhanden ist und die Kabel funktionieren.
	Fehlercodes in Verbindung mit Serverantwort-Formaten	
69	65	Nachrichtenantwort stimmt nicht mit Anforderung überein.
	68	Der Datentyp der Nachrichtenantwort ist ungültig oder wird nicht unterstützt.
208		Für das Netzwerk ist keine IP-Adresse konfiguriert. Um das Problem zu lösen, konfigurieren Sie eine gültige IP-Adresse. Der Code gilt für EtherNet/IP.
209		Maximale Anzahl der genutzten Verbindungen – keine Verbindungen verfügbar. Zur Vermeidung dieses Problems reduzieren Sie die Anzahl der verwendeten Verbindungen. Der Code gilt für EtherNet/IP.
210		Ungültige Internetadresse bzw. Knotenadresse Zur Vermeidung dieses Problems verwenden Sie nicht 0, Broadcast, Multicast oder Loopback für die Ziel-IP-Adresse. Der Code gilt für EtherNet/IP.
217		Die Ausführung der Nachricht wurde vom Benutzer abgebrochen (Abbruchparameter wurde auf TRUE gesetzt). Zur Vermeidung dieses Problems setzen Sie das Abbruchbit nicht oder führen Sie die RMD-Operation nicht durch.
218		Kein Raum für Netzwerkpuffer verfügbar Zur Vermeidung dieses Problems stellen Sie sicher, dass keine unangemessenen Kontinuierlichen Kommunikationsaktivitäten stattfinden. Der Code gilt für EtherNet/IP.
222		Reserviert.
223		Die Adresse der Verknüpfung ist nicht verfügbar. Eine Änderung der TCP/IP- oder Ethernet-Konfiguration wird ausgeführt. Dieser Code gilt für EtherNet/IP.
224		CIP-Antwort-Fehlercode Mit SubErrorID wird der CIP-Status und mit ExtErrorID der erweiterte CIP-Statuswert festgelegt. Die Werte der möglichen Fehlercodes können Sie den CIP-Angaben entnehmen. Der Code gilt für EtherNet/IP.
255		Kanal ist heruntergefahren oder Neukonfiguration läuft. Dieser Fehler kann bei der Neukonfiguration auftreten, oder der Port ist nicht verfügbar, wenn die MSG_CIPPCCC-Anweisungen ausgeführt werden. Zur Vermeidung dieses Problems warten Sie, bis die Konfiguration abgeschlossen ist, bevor Sie die Anweisungen ausführen. Der Code gilt für EtherNet/IP.

Tabelle der Elementanzahl

Quell-/Zieldatentyp	Maximale Elementanzahl für Dienstcode von 0 – Lese-Dienst	Maximale Elementanzahl für Dienstcode 1 – Schreib-Dienst
BIT	122	116
INT	122	116
FLOAT	61	58
STRING	2	2

Quell-/Zieldatentyp	Maximale Elementanzahl für Dienstcode von 0 – Lese-Dienst	Maximale Elementanzahl für Dienstcode 1 – Schreib-Dienst
DOUBLE	61	58

Informationen zum Erstellen eines MSG_PCCC-Programms finden Sie unter dem Beispiel: Erstellung eines Programms für den MSG_PCCC-Nachrichtenaustausch, das einen Wert auf eine Adresse in MicroLogix schreibt und von dieser Adresse liest auf seite 200.

Beispiel: Erstellen Sie ein Programm für MSG_PCCC-Nachrichtenaustausch, um einen Wert auf eine Adresse in MicroLogix zu schreiben und von dieser zu lesen

In diesem Beispiel erfahren Sie, wie Sie ein Programm für Meldungen erstellen, um über Ethernet den Wert einer globalen Variable von Controller B in eine Altadresse von Controller A zu schreiben und den Wert einer Altadresse von Controller A zu lesen und in einer globalen Variable von Controller B zu speichern.

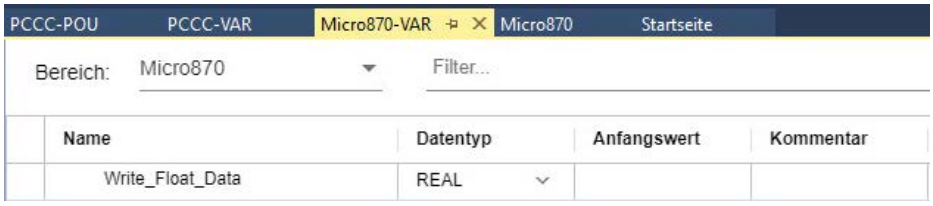
Beispiel für das PCCC-Schreiben: Hinzufügen eines MSG_PCCC-Funktionsblocks, Variablen und eines Kontakts

Um ein Programm für den MSG_PCCC-Nachrichtenaustausch zu erstellen, das den Wert eines Controllers der Serie Micro870E auf eine Altadresse in MicroLogix 1400 schreibt, führen Sie die folgenden Aufgaben durch.

Um für eine Quellvariable eine globale Variable zu erstellen

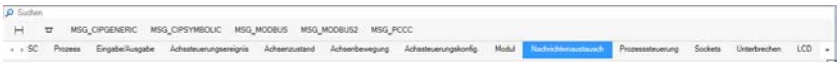
1. Doppelklicken Sie im **Projekt-Organizer** auf **Globale Variablen**, um den Wähler **Globale Variablen** anzeigen zu lassen.
2. Geben Sie in die Spalte **Name Write_Float_Data** ein.
3. Überprüfen Sie, ob der Datentyp REAL ist.

Der Wähler **Globale Variablen** sollte ähnlich der folgenden Abbildung angezeigt werden.

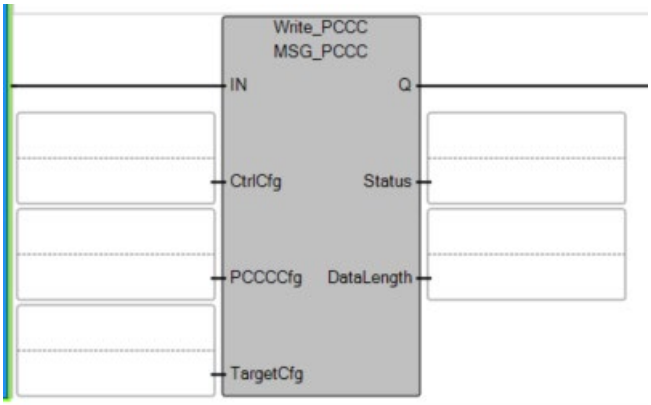


Um einen Funktionsblock und Variablen hinzuzufügen

1. Wählen Sie in der **Befehlsymbolleiste – Nachrichtenaustausch** die Option **MSG_PCCC** aus und ziehen Sie das Element auf den Strompfad.

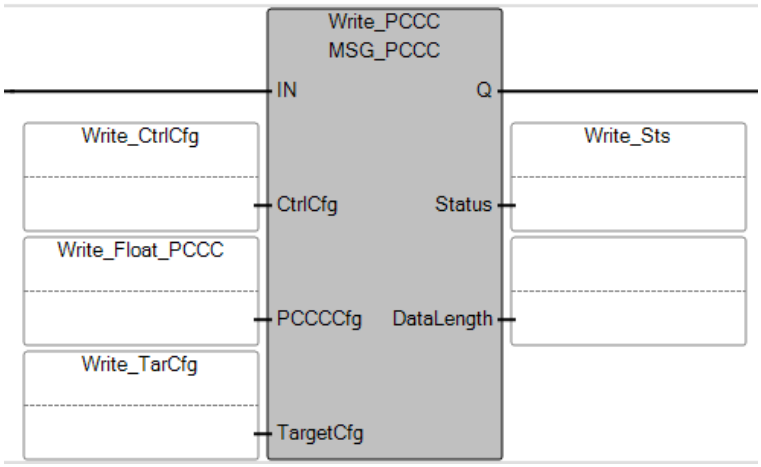


2. (Optional) Doppelklicken Sie auf **MSG_PCCC**, um die Instanz des Funktionsblocks auf „Write_PCCC“ zu ändern.



3. Variablennamen zuweisen.
   a. Doppelklicken Sie im **Projekt-Organizer** auf die Option **Lokale Variablen**, um die Seite **Lokale Variablen** anzeigen zu lassen.
   b. Fügen Sie im **Variablenwähler** die in der Tabelle aufgelisteten Variablen und Datentypen hinzu.
   c. Weisen Sie den Variablen die Befehlsein- und -ausgabeparameter zu.

Parameter	Variablenname	Datentyp
CtrlCfg	Write_CtrlCfg	CIPCONTROLCFG
PCCCCfg	Write_PCCC	PCCC_CFG
TargetCfg	Write_TarCfg	CIPTARGETCFG
Status	Write_Sts	CIPSTATUS



4. Konfigurieren Sie die Anfangswerte für die lokalen Variablen und fügung Sie die Anfangswerte für CtrlCfg hinzu.
   a. Klappen Sie auf der Seite **Lokale Variablen** den Parameter „CtrlCfg“ auf, um zusätzliche Parameter anzeigen zu lassen.
   b. Geben Sie in der Spalte **Anfangswert** für jeden Parameter die folgenden Werte ein.

Parameter	Anfangswert
Write_CtrlCfg.Cancel	Leer lassen
Write_CtrlCfg.TriggerType	50
Write_CtrlCfg.StrMode	Leer lassen

Tipp: In diesem Beispiel wird der Wert **50** eingegeben, um ein zyklisches Auslöseintervall von 50 ms zu definieren.

5. Fügen Sie die Startwerte für PCCCCfg hinzu.
   a. Klappen Sie auf der Seite **Lokale Variablen** den Parameter „PCCCCfg“ auf, um zusätzliche Parameter anzeigen zu lassen.
   b. Geben Sie in der Spalte **Anfangswert** für jeden Parameter die folgenden Werte ein.

Parameter	Anfangswert
Write_Float_PCCC.Service	1
Write_Float_PCCC.SrcAdr	„Write_Float_Data“
Write_Float_PCCC.NumOfElements	1
Write_Float_PCCC.DstAdr	„F8:0“

Tipp: Als Zieladresse ist im MicroLogix-Controller **F8:0** eingegeben.

Der Wähler „Globale Variablen“ sollte ähnlich der folgenden Abbildung angezeigt werden.

Name	Anfangswert	Datentyp	Kommentar
Write_CtrlCfg	...	CIPCONTR...	
Write_CtrlCfg.Cancel		BOOL	Ausführung der...
Write_CtrlCfg.TriggerType	50	UDINT	0 – Einmal ausl...
Write_CtrlCfg.StrMode		USINT	Reservierter Pa...
Write_Float_PCCC	...	PCCC_CFG	
Write_Float_PCCC.Service	1	USINT	0 – Lesen, 1 – ...
Write_Float_PCCC.SrcAdr	""Write_Float_D...	STRING	Quelladresse: I...
Write_Float_PCCC.NumOfElements	1	USINT	Anzahl der Ele...
Write_Float_PCCC.DstAdr	'F8:0'	STRING	Zieladresse: Im...

6. Konfigurieren Sie die Anfangswerte für TarCfg.
   a. Klappen Sie auf der Seite **Lokale Variablen** den Parameter „TarCfg“ auf, um zusätzliche Parameter anzeigen zu lassen.
   b. Geben Sie in der Spalte **Anfangswert** für jeden Parameter die folgenden Werte ein.

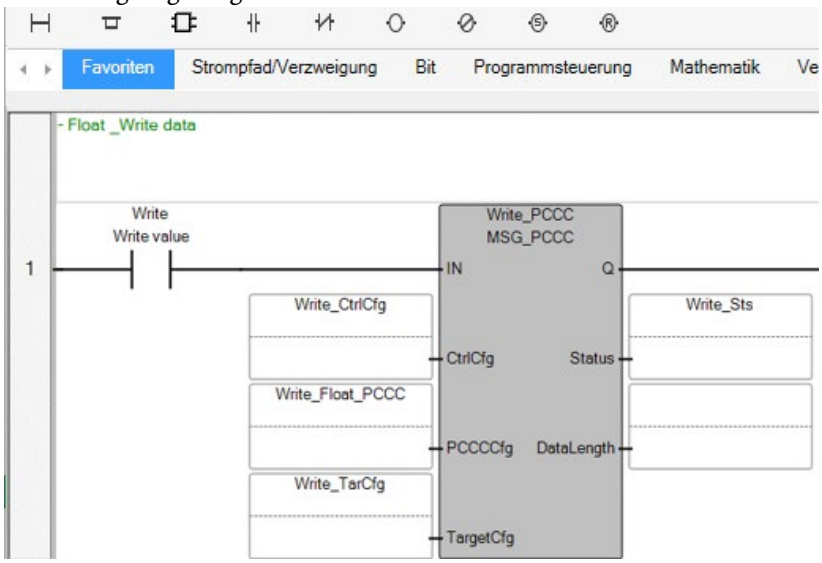
Parameter	Anfangswert
Write_TarCfg.Path	„4,192.168.3.33“
Write_TarCfg.CipConnMode	1
Write_TarCfg.UcmmTimeout	0
Write_TarCfg.ConnMsgTimeout	0
Write_TarCfg.ConnClose	FALSE

Der Wähler „Globale Variablen“ sollte ähnlich der folgenden Abbildung angezeigt werden.

Name	Anfangswert	Datentyp	Kommentar
Write_TarCfg	...	CIPTARGE...	
Write_TarCfg.Path	'4,192.168.3.33'	STRING	CIP-Zielpfad
Write_TarCfg.CipConnMode	1	USINT	0 – Nicht verbu...
Write_TarCfg.UcmmTimeout	0	UDINT	Zeitüberschreit...
Write_TarCfg.ConnMsgTimeout	0	UDINT	Zeitüberschreit...
Write_TarCfg.ConnClose	FALSE	BOOL	TRUE: CIP-Ver...

7. Fügen Sie einen Kontakt hinzu.
   a. Wählen Sie in der **Befehlsymbolleiste – Favorit** die Option **Direkter Kontakt** aus und ziehen Sie das Element links neben den MSG_PCCC-Funktionsblockeingang des ersten Strompfads.
   b. Klicken Sie auf den Kontakt und weisen Sie ihm die Variable **Schreiben** zu.

   Der Strompfad Ihres Kontaktplan-Programms für den MSG_PCCC-Nachrichtenaustausch sollte ähnlich der folgenden Abbildung angezeigt werden.



8. Laden Sie das Programm herunter.

   Nach dem erfolgreichen Herunterladen geben Sie den in MicroLogix in die Variable **Write_Float_Data** im Controller Micro800 zu schreibenden Wert ein und lösen dann den **Schreiben**-Tag aus, um einen Schreibwert von Write_Float_Data in den MicroLogix-Controller an der Adresse F8:0 durchzuführen. Beachten Sie, dass die MicroLogix-Adresse auf den Wert aktualisiert wird.

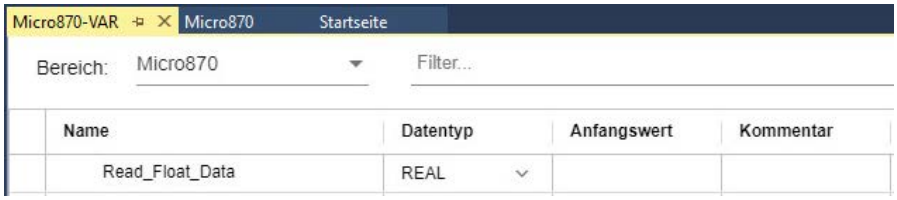
Beispiel für PCCC-Lesen Hinzufügen eines PCCC_MSG-Funktionsblocks , Variablen und eines Kontakts

Um ein Programm für den MSG_PCCC-Nachrichtenaustausch zu erstellen, das einen Wert aus einer Altadresse in MicroLogix 1400 an einem Controller der Produktreihe Micro870E-Controller ausliest, führen Sie die folgenden Aufgaben durch.

Um eine globale Variable für die Speicherung der Ergebnisse zu erstellen

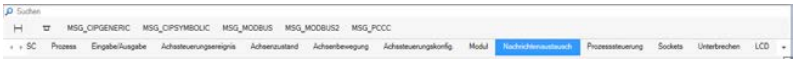
1. Doppelklicken Sie im **Projekt-Organizer** auf **Globale Variablen**, um den Wähler **Globale Variablen** anzeigen zu lassen.
2. Geben Sie in die Spalte **Name Read_Float_Data** ein.
3. Überprüfen Sie, ob der Datentyp REAL ist.

Der Wähler **Globale Variablen** sollte ähnlich der folgenden Abbildung angezeigt werden.

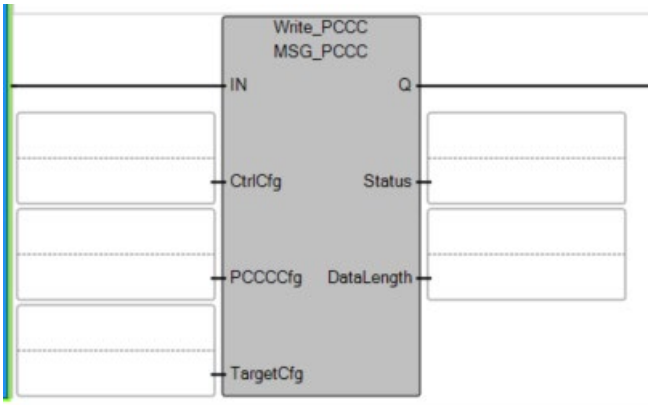


Um einen Funktionsblock und Variablen hinzuzufügen

1. Wählen Sie in der **Befehlsymbolleiste – Nachrichtenaustausch** die Option **MSG_PCCC** aus und ziehen Sie das Element auf den Strompfad.



2. (Optional) Doppelklicken Sie auf **MSG_PCCC**, um die Instanz des Funktionsblocks auf „Read_PCCC“ zu ändern.

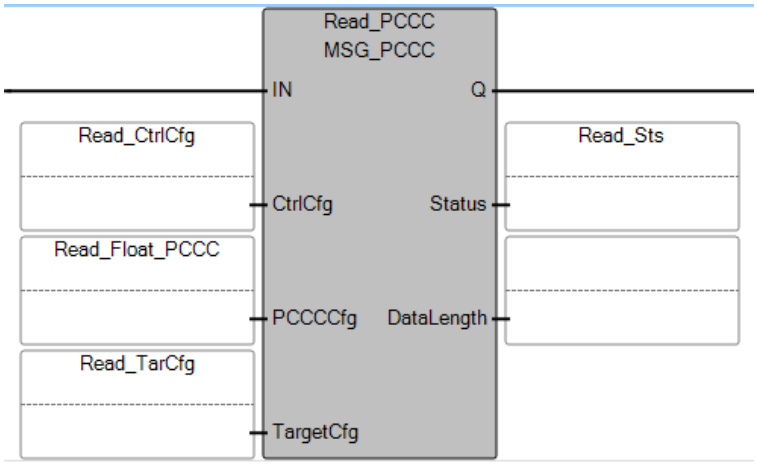


3. Variablennamen zuweisen.
   a. Doppelklicken Sie im **Projekt-Organizer** auf die Option **Lokale Variablen**, um die Seite **Lokale Variablen** anzeigen zu lassen.

b. Fügen Sie im **Variablenwähler** die in der Tabelle aufgelisteten Variablen und Datentypen hinzu.

c. Weisen Sie den Variablen die Befehlsein- und -ausgabeparameter zu.

Parameter	Variablenname	Datentyp
CtrlCfg	Read_CtrlCfg	CIPCONTROLCFG
PCCCCfg	Read_PCCC	PCCC_CFG
TargetCfg	Read_TarCfg	CIPTARGETCFG
Status	Read_Sts	CIPSTATUS



4. Konfigurieren Sie die Anfangswerte für die lokalen Variablen und fügung Sie die Anfangswerte für CtrlCfg hinzu.

   a. Klappen Sie auf der Seite **Lokale Variablen** den Parameter „CtrlCfg“ auf, um zusätzliche Parameter anzeigen zu lassen.

   b. Geben Sie in der Spalte **Anfangswert** für jeden Parameter die folgenden Werte ein.

Parameter	Anfangswert
Read_CtrlCfg.cancel	Leer lassen
Read_CtrlCfg.TriggerType	50
Read_CtrlCfg.StrMode	Leer lassen

Tipp: In diesem Beispiel wird der Wert **50** eingegeben, um ein zyklisches Auslöseintervall von 50 ms zu definieren.

5. Fügen Sie die Startwerte für PCCCCfg hinzu.

   a. Klappen Sie auf der Seite **Lokale Variablen** den Parameter „PCCCCfg“ auf, um zusätzliche Parameter anzeigen zu lassen.

   b. Geben Sie in der Spalte **Anfangswert** für jeden Parameter die folgenden Werte ein.

Parameter	Anfangswert
Read_Float_PCCC.Service	1
Read_Float_PCCC.SrcAdr	„F8:0“
Read_Float_PCCC.NumOfElements	1

Parameter	Anfangswert
Read_Float_PCCC.DstAdr	„Read_Float_Data“

Tipp: Als Quelladresse ist im MicroLogix-Controller **F8:0** eingegeben.

Der Wähler „Globale Variablen“ sollte ähnlich der folgenden Abbildung angezeigt werden.

Name	Anfangswert	Datentyp	Kommentar
Read_Ctrlcfg	...	CIPCONTR...	
Read_Ctrlcfg.Cancel		BOOL	Ausführung der...
Read_Ctrlcfg.TriggerType	50	UDINT	0 – Einmal ausl...
Read_Ctrlcfg.StrMode		USINT	Reservierter Pa...
Read_Float_PCCC	...	PCCC_CFG	
Read_Float_PCCC.Service	1	USINT	0 – Lesen, 1 – ...
Read_Float_PCCC.SrcAdr	'F8:0'	STRING	Quelladresse: I...
Read_Float_PCCC.NumOfElements	1	USINT	Anzahl der Ele...
Read_Float_PCCC.DstAdr	'Read_Float_D...	STRING	Zieladresse: Im...

6. Konfigurieren Sie die Anfangswerte für TarCfg.
   a. Klappen Sie auf der Seite **Lokale Variablen** den Parameter „TarCfg“ auf, um zusätzliche Parameter anzeigen zu lassen.
   b. Geben Sie in der Spalte **Anfangswert** für jeden Parameter die folgenden Werte ein.

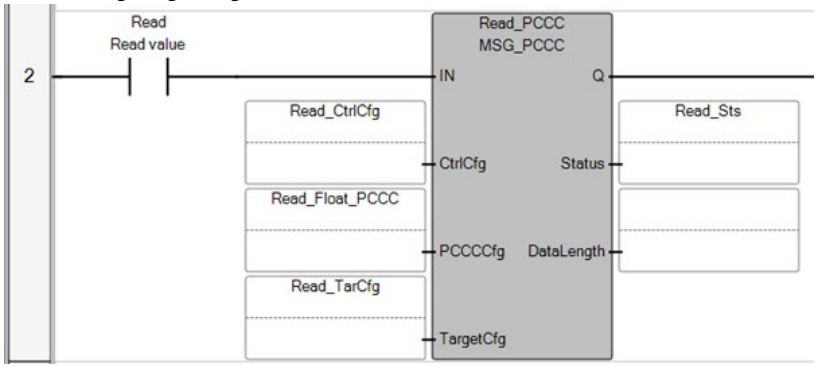
Parameter	Anfangswert
Read_TarCfg.Path	„4,192.168.3.33“
Read_TarCfg.CipConnMode	1
Read_TarCfg.UcmmTimeout	0
Read_TarCfg.ConnMsgTimeout	0
Read_TarCfg.ConnClose	FALSE

Der Wähler „Globale Variablen“ sollte ähnlich der folgenden Abbildung angezeigt werden.

Name	Anfangswert	Datentyp	Kommentar
Read_TarCfg	...	CIPTARGE...	
Read_TarCfg.Path	'4,192.168.3.33'	STRING	CIP-Zielpfad
Read_TarCfg.CipConnMode	1	USINT	0 – Nicht verbu...
Read_TarCfg.UcmmTimeout	0	UDINT	Zeitüberschreit...
Read_TarCfg.ConnMsgTimeout	0	UDINT	Zeitüberschreit...
Read_TarCfg.ConnClose	FALSE	BOOL	TRUE: CIP-Ver...

7. Fügen Sie einen Kontakt hinzu.
   a. Wählen Sie in der **Befehlsymbolleiste – Favorit** die Option **Direkter Kontakt** aus und ziehen Sie das Element links neben den MSG_PCCC-Funktionsblockeingang des zweiten Strompfads.
   b. Klicken Sie auf den Kontakt und weisen Sie ihm die Variable **Lesen** zu.

Der Strompfad Ihres Kontaktplan-Programms für den MSG_PCCC-Nachrichtenaustausch sollte ähnlich der folgenden Abbildung angezeigt werden.



8. Laden Sie das Programm herunter.

   Nach erfolgreichem Herunterladen geben Sie den Wert in den MicroLogix-Controller an der Adresse F8:0 ein und lösen anschließend im Micro800-Controller den Lesen-Tag aus, um das Lesen eines Werts aus der MicroLogix-Adresse F8:0 durchzuführen und beobachten Sie, wie der Wert in Read_Float_Data des Micro800-Controllers geschrieben wird.

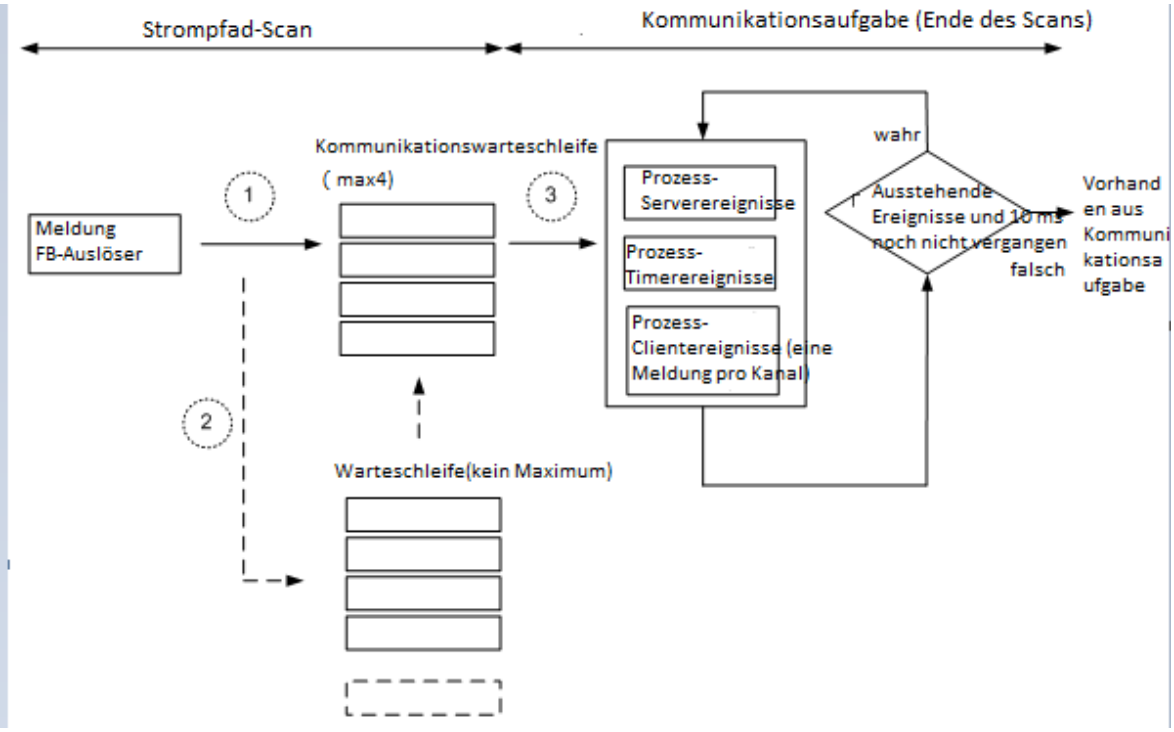
Nachrichtenausführungsprozesse und Timing-Diagramme

Die folgenden Themen beschreiben, wie und wann MSG_CIPGENERIC auf seite 166, MSG_CIPSYMBOLIC auf seite 173, MSG_MODBUS2 auf seite 183 und MSG_PCCC auf seite 189-Nachrichtenanweisungen basierend auf ihren Bit- und Strompfadzuständen ausgeführt werden.

- Nachrichtenausführungsprozess (allgemein)
- Nachrichtenausführungsprozess (Strompfad = TRUE)
- Nachrichtenausführungs-Timing-Diagramm (Strompfad = True)
- Nachrichtenausführungsprozess (Strompfad = FALSE)
- Nachrichtenausführung-Timing-Diagramm (Strompfad = FALSE)
- Nachrichtenausführungsprozess (Fehler)
- Nachrichtenausführungs-Timing-Diagramm (Fehler bei der PCCC-Konfiguration)
- Nachrichtenausführungs-Timing-Diagramm (Nachricht wird abgebrochen)

Nachrichtenausführungsprozess (allgemein)

Das folgende Diagramm zeigt auf, wie und wann Nachrichten basierend auf dem Status der Com Queue ausgeführt werden.

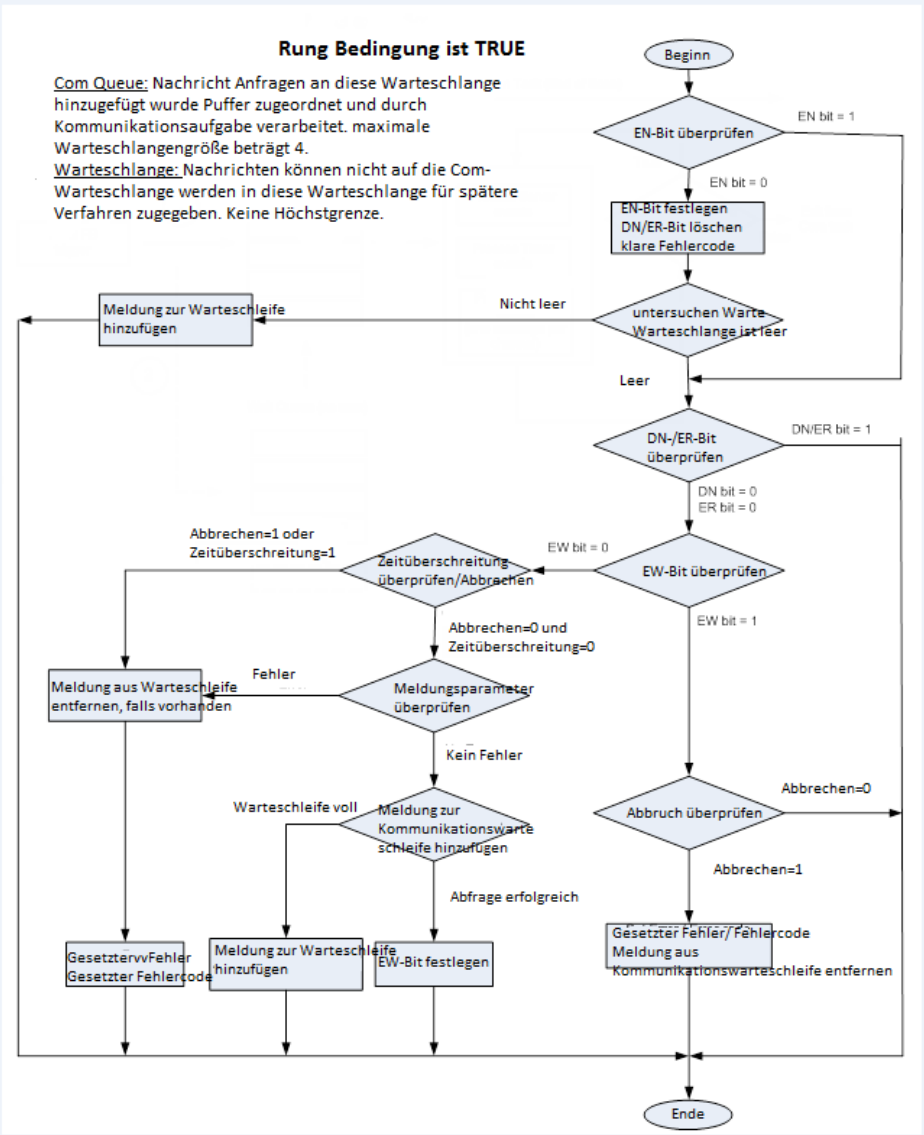


In der folgenden Tabelle wird die Abfolge der Ereignisse beschrieben, die im vorherigen Diagramm ermittelt wurden.

Nr.	Beschreibung der Ereignisse
1	Die Nachricht ist aktiviert. Falls die Com-Warteschlange leer ist, wird der Nachricht ein Puffer zugewiesen und sie wird zur Übertragung der Com-Warteschlange hinzugefügt. Die Größe der Com-Warteschlange beträgt 4 und jeder Kanal besitzt eine eigene Warteschlange.
2	Wenn die Com-Warteschlange voll ist, wird die Nachricht der allgemeinen Warteschlange hinzugefügt. Wenn die Com-Warteschlange leer ist, wird die Nachricht von der allgemeinen Warteschlange in die Com-Warteschlange überführt. Es gibt keine Größenbeschränkung für die allgemeine Warteschlange, und jeder Kanal besitzt eine separate Warteschlange.
3	Der Kommunikationstask gibt die in der Com-Warteschlange befindlichen Nachrichten nach Abschluss jedes Scans zur Übertragung weiter. Die Warteschlangen der Kanäle werden nacheinander per Ringverteilung verarbeitet. Von jedem Kanal wird eine Nachricht ausgeführt und der Prozess fährt fort, bis alle Nachrichten ausgeführt wurden oder der Kommunikationszeitplan (10 ms) abgelaufen ist. Der Kanal neben dem zuletzt verarbeiteten Kanal wird so geplant, dass er bei dem nächsten Scanende zuerst erscheint.

Nachrichtenausführungsprozess (Strompfad = TRUE)

In dem folgenden Prozessdiagramm werden die Nachrichtenanweisungsereignisse beschrieben, die auftreten, wenn der Strompfadzustand „True“ ist.



Com-Warteschlange: Nachrichtanforderungen, die zur Com-Warteschlange hinzugefügt wurden, wird ein Puffer zugewiesen. Sie werden von der Kommunikationsaufgabe verarbeitet. Die Größe der Warteschlange beträgt maximal 4.

Warteschlange: Nachrichten, die nicht zur Com-Warteschlange hinzugefügt werden können, werden zur Warteschlange hinzugefügt und zu einem späteren Zeitpunkt verarbeitet. Für die Warteschlange besteht keine Größenbeschränkung.

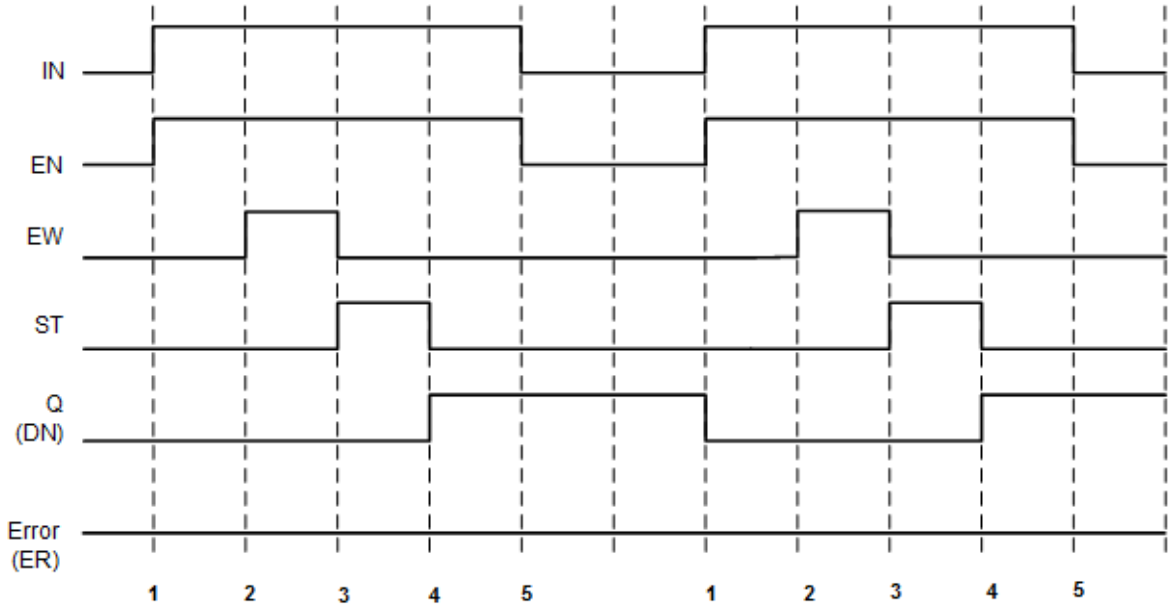
Nachrichtenausführungs-Timing-Diagramm (Strompfad = TRUE)

In der folgenden Tabelle werden die Nachrichtenzustände und der Bitstatus für jeden Ausführungsschritt im Timing-Diagramm ermittelt, dabei bleibt der Strompfadzustand „True“.

Schritt	Nachrichtenbeschreibung	Bitstatus

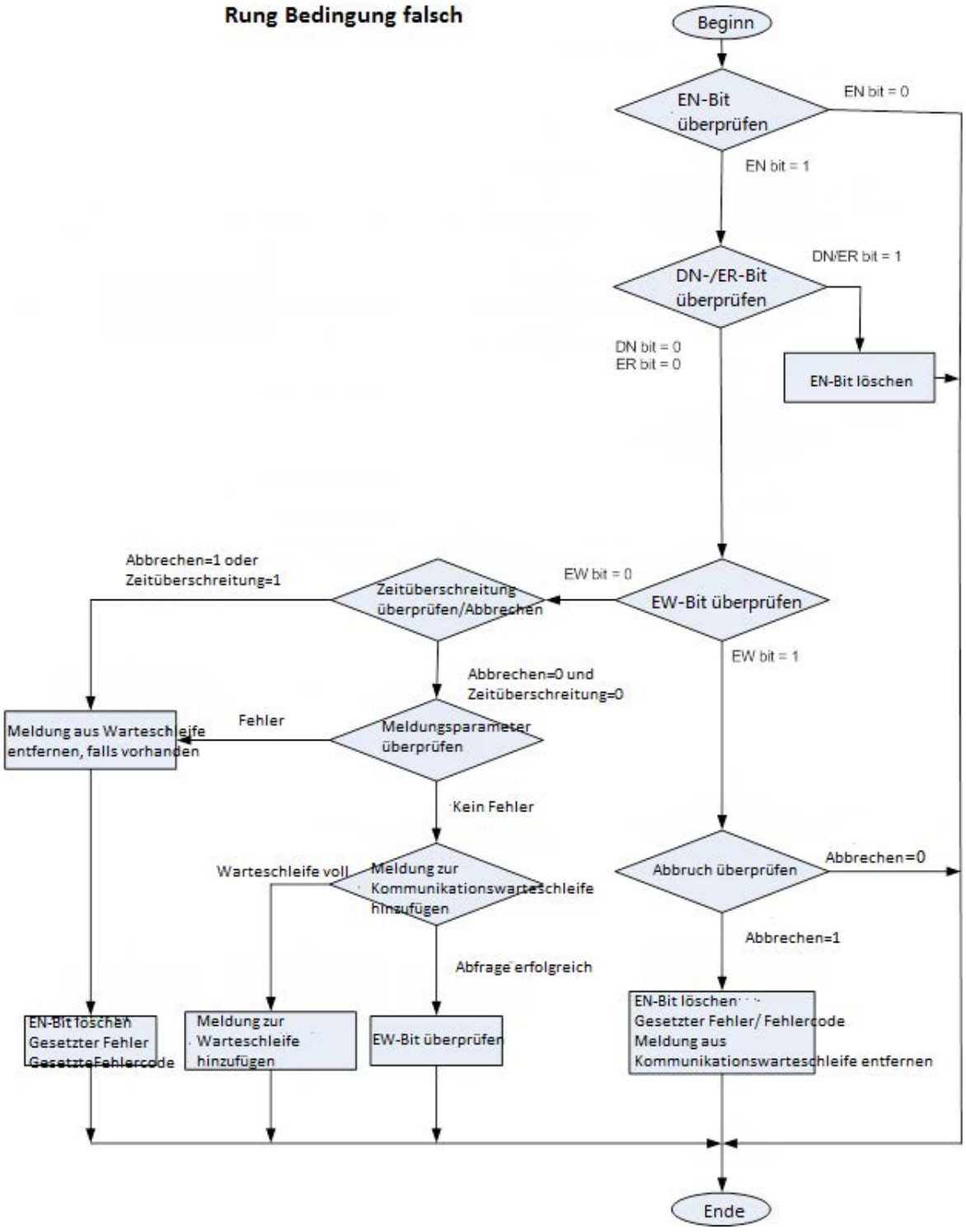
Schritt	Nachrichtenbeschreibung	Bitstatus
1	Strompfadzustand wird TRUE. Nachrichtenausführung ist aktiviert.	EN-Bit wurde festgelegt. Alle anderen Bits wurden gelöscht.
2	Der Nachrichtensteuerungspuffer wird abgerufen. Zu diesem Zeitpunkt werden die Eingabedaten (d.h. Datenparameter zum Schreiben von Nachrichten) zur Übertragung kopiert. Später an den Eingabedaten vorgenommene Änderungen werden nicht in der übertragenen Nachricht reflektiert.	EN-Bit wurde festgelegt.
3	Nachrichtenübertragung beginnt.	EW-Bit wurde gelöscht. ST-Bit wurde festgelegt.
4	Nachrichtenantwort wurde empfangen.	ST-Bit wurde gelöscht. DN-Bit wurde festgelegt.
5	Strompfadzustand wird FALSE.	EN-Bit wurde gelöscht.

Timing-Diagramm für (Strompfad = TRUE)



Nachrichtenausführungsprozess (Strompfad = FALSE)

In dem folgenden Prozessdiagramm werden die Nachrichtenanweisungsereignisse beschrieben, die auftreten, wenn der Strompfadzustand „False“ ist.



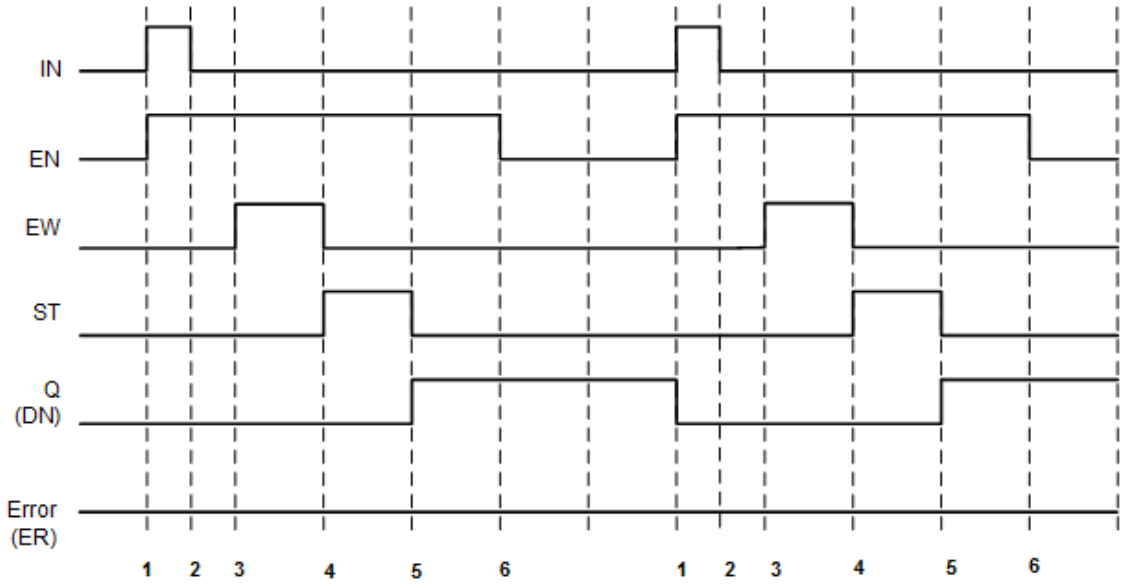
Nachrichtenausführung-Timing-Diagramm (Strompfad = FALSE)

In der folgenden Tabelle werden die Nachrichtenzustände und der Bitstatus für jeden Ausführungsschritt im Timing-Diagramm ermittelt, wenn der Strompfadzustand während der Ausführung zu FALSE wechselt.

Schritt	Nachrichtenbeschreibung	Bitstatus

Schritt	Nachrichtenbeschreibung	Bitstatus
1	Strompfadzustand wird TRUE. Nachrichtenausführung ist aktiviert.	EN-Bit wurde festgelegt. Alle anderen Bits wurden gelöscht.
2	Strompfadzustand wird FALSE. Nachrichtenausführung wird fortgesetzt.	
3	Der Nachrichtenpuffer wird abgerufen. Zu diesem Zeitpunkt werden die Eingabedaten (d. h. Datenparameter zum Schreiben von Nachrichten) zur Übertragung kopiert. Später an den Eingabedaten vorgenommene Änderungen werden nicht in der übertragenen Nachricht reflektiert.	EN-Bit wurde festgelegt.
4	Nachrichtenübertragung beginnt.	EW-Bit wurde gelöscht. ST-Bit wurde festgelegt.
5	Nachrichtenantwort wurde empfangen.	ST-Bit wurde gelöscht. DN-Bit wurde festgelegt.
6	Nachricht wird nach Schritt 5 erneut gescannt.	EN-Bit wurde gelöscht.

Timer-Diagramm für (Strompfad = FALSE)



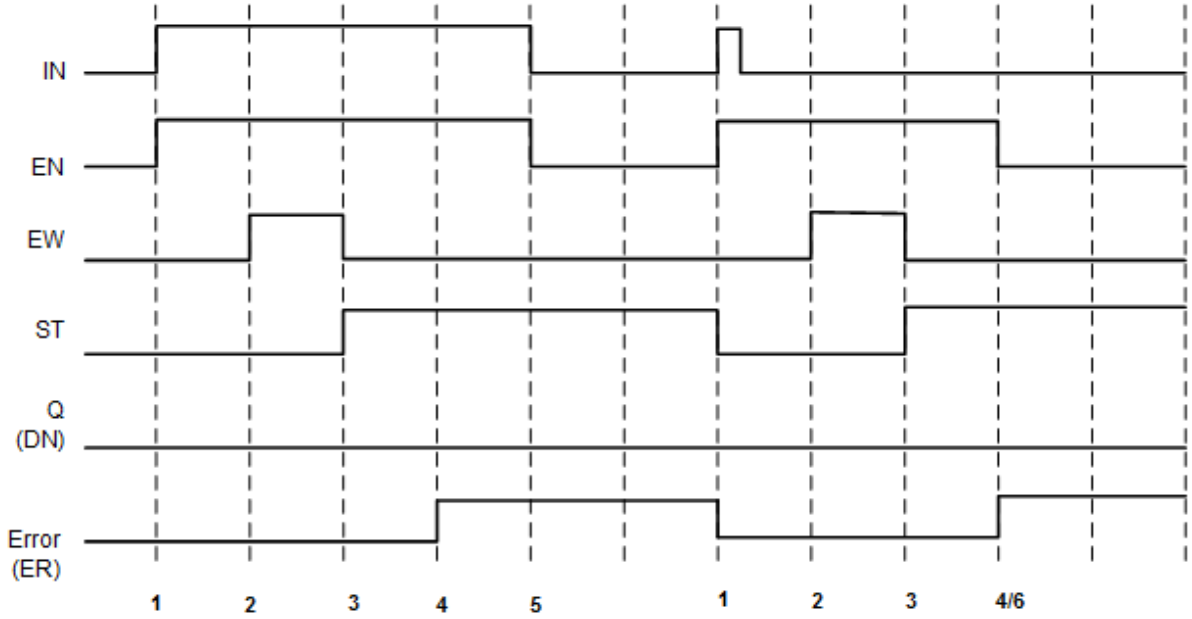
Nachrichtenausführungsprozess (Fehler)

In der folgenden Tabelle werden die Nachrichtenzustände und der Bitstatus für jeden Ausführungsschritt beschrieben, die im Timing-Diagramm ermittelt werden, wenn während der Ausführung ein Fehler auftritt.

Schritt	Nachrichtenbeschreibung	Bitstatus
1	Strompfadzustand wird TRUE. Nachrichtenausführung ist aktiviert.	EN-Bit wurde festgelegt. Alle anderen Bits wurden gelöscht.
2	Nachrichtenpuffer wurde abgerufen.	EN-Bit wurde festgelegt.
3	Nachrichtenübertragung beginnt.	EW-Bit wurde gelöscht. ST-Bit wurde festgelegt.
4	Zeitüberschreitung bei der Nachrichtenübertragung.	EW- und ST-Bits verändern sich nicht.
4-6	Wenn Strompfadzustand FALSE wird.	EN-Bit wurde gelöscht. ER-Bit wurde festgelegt.

Nachrichtenausführungs-Timing-Diagramm (Fehler)

In dem folgenden Timing-Diagramm wird ein typisches Muster eines Fehlers während der Ausführung dargestellt.

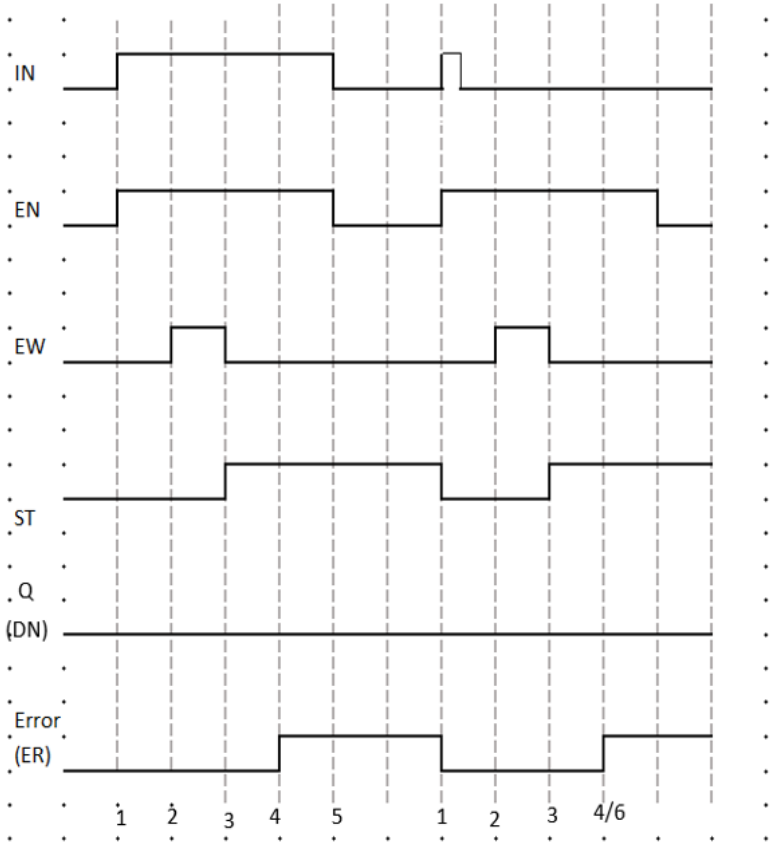


Nachrichtenausführungs-Timing-Diagramm (Fehler bei der PCCC-Konfiguration)

In der folgenden Tabelle werden die Nachrichtenzustände und der Bitstatus für jeden Ausführungsschritt beschrieben, die im Timing-Diagramm ermittelt werden, wenn während der Ausführung ein PCCC-Konfigurationsfehler auftritt.

Schritt	Nachrichtenbeschreibung	Bitstatus
1	Strompfadzustand wird TRUE. Nachrichtenausführung ist aktiviert.	EN-Bit wurde festgelegt.
2	Nachrichtenpuffer wurde abgerufen.	EW-Bit wurde festgelegt.
3	In PCCCCfg sind ungültige Elemente vorhanden.	ER-Bit wird auf 1 gesetzt, bevor ST-Bit auf 1 gesetzt wird. EW- und ST-Bits verändern sich nicht.
4	Wenn Strompfadzustand FALSE wird.	EN-, EW- und ER-Bits werden zurückgesetzt.

Das folgende Diagramm zeigt die typische Fehlerbedingung, wenn die Anzahl der Elemente in PCCCCfg in eine ungültige Zahl 17 geändert wird.

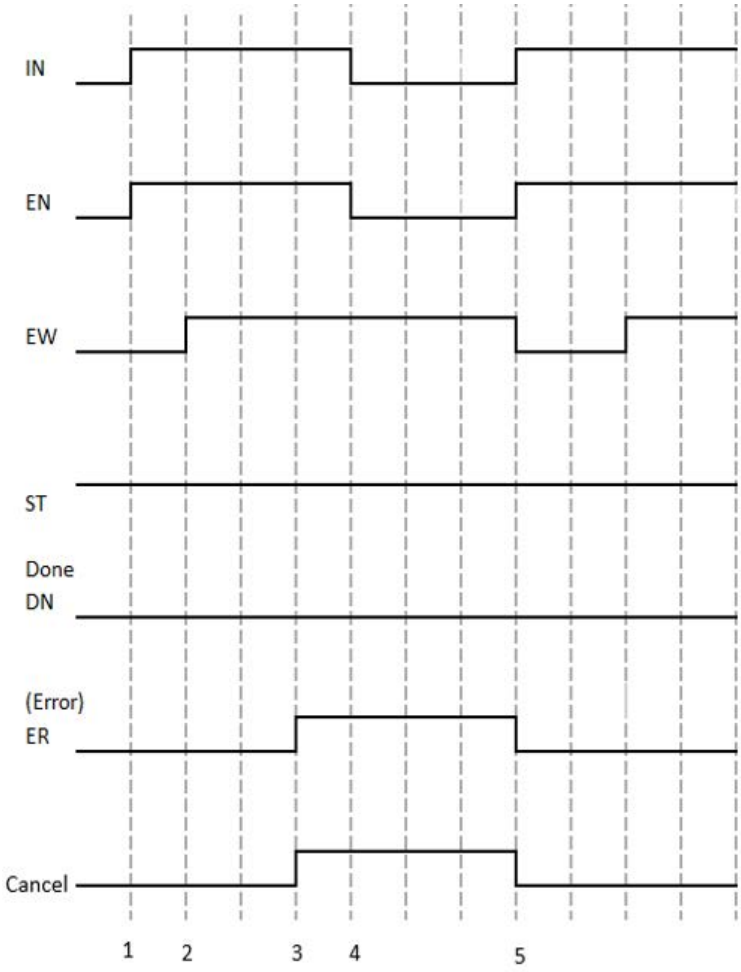


Nachrichtenausführungs-Timing-Diagramm (Nachricht wird abgebrochen)

In der folgenden Tabelle werden die Nachrichtenzustände und der Bitstatus für jeden Ausführungsschritt beschrieben, die im Timing-Diagramm ermittelt werden, wenn eine Nachricht abgebrochen wird.

Schritt	Nachrichtenbeschreibung	Bitstatus
1	Strompfadzustand wird TRUE. Nachrichtenausführung ist aktiviert.	EN-Bit wurde festgelegt. Alle anderen Bits werden gelöscht.
2	Nachrichtenpuffer wurde abgerufen.	EW-Bit wurde festgelegt.
3	Ausführung der Nachricht wird abgebrochen.	ER-Bit wurde festgelegt. EW- und ST-Bits verändern sich nicht.
4	Strompfadzustand wird FALSE.	EN-Bit wurde gelöscht.
5	Strompfadzustand wird TRUE.	Der Abbrechen-Eingang wird gelöscht. Der Funktionsblock beginnt mit der Ausführung.

Timing-Diagramm, wenn eine Nachricht abgebrochen wird



Verwenden der Kommunikationsfunktionsblöcke (Nachricht)

In diesem Abschnitt werden bestimmte Informationen und Beispiele zur Verwendung von Kommunikationsbefehlen auf seite 163 in Logik-Programmen beschrieben. In den folgenden Themen finden Sie Informationen und Beispiele zur Verwendung der MSG_CIPGENERIC auf seite 217- und MSG_CIPSYMBOLIC-Funktionsblöcke, um Programme zu erstellen auf seite 225.

Konfigurieren von Objektdatenwerten für die explizite Nachrichtenvermittlung (MSG_CIPGENERIC)

Damit Sie den MSG_CIPGENERIC-Funktionsblock auf seite 166 für explizite Nachrichtenvermittlung einsetzen können, müssen Sie die AppCfg-Parameter mit den korrekten Werten konfigurieren.

Zusätzliche Informationen über die Nachrichtenkommunikation

Es gibt verschiedene Informationsquellen zur Implementierung und Verwendung der Nachrichtenkommunikation, darunter die Hilfe zu Connected Components Workbench, Benutzerhandbücher und die Rockwell Automation-Literaturbibliothek.

In der folgenden Tabelle werden zusätzliche Informationsquellen zur Nachrichtenkommunikation aufgeführt.

Informationsquelle	Beschreibung	So finden Sie die Informationen
Benutzerhandbuch für bestimmte Kommunikationsgeräte	Enthält wichtige Informationen über Messaging und zur Konfiguration von Nachrichtenfunktionsblöcke.	Connected Components Workbench Hilfe-Menü
EtherNet-/IP-Adapter 22-COMM-E FRN 1.xxx, Anhang C	Stellt Informationen zu EtherNet-/IP-Objekten bereit, auf die Mithilfe von Explicit Messages zugegriffen werden kann.	Connected Components Workbench Hilfe-Menü
EtherNet-/IP-Angaben	Definiert die einzuschließenden Objekte für jedes CIP-Gerät: Bezeichnet das Objekt, das Nachrichtenrouterobjekt und das Netzwerkobjekt.	ODVA-Website (http://www.odva.org)
Programmierbare Micro800-Controller: Erste Schritte mit dem CIP-Client-Messaging	Enthält eine Schnellstartanleitung für den Einsatz von CIP GENERIC- und CIP Symbolic-Messaging in speicherprogrammierbaren Micro830- und Micro850-Steuerungen (SPS).	Rockwell Automation Literaturbibliothek

So rufen Sie Benutzerhandbücher und Schnellstartanleitungen auf:

1. Um auf Schnellstartanleitungen zuzugreifen, klicken Sie im Menü **Hilfe** auf **Hilfe anzeigen**.
2. Doppelklicken Sie auf **Connected Components Workbench**.
3. Doppelklicken Sie auf **Erste Schritte in Connected Components Workbench**.
4. Um auf Antriebsanleitungen zuzugreifen, klicken Sie im Menü **Hilfe** auf **Benutzerhandbücher**, um das Dialogfeld „Handbücher“ anzuzeigen.
5. Klicken Sie auf das (+)-Zeichen neben den Antrieben, um die Kategorie zu erweitern und anschließend auf die Klasse, bis Sie das Handbuch gefunden haben.
6. Doppelklicken Sie auf den Namen des Handbuchs, um die PDF-Datei zu öffnen.
7. Wenn Sie auf das Ethernet/IP-Handbuch zugreifen möchten, klicken Sie im Menü **Hilfe** auf **Benutzerhandbücher**, um das Dialogfeld „Handbücher“ anzuzeigen.
8. Klicken Sie auf das (+)-Zeichen neben den **Antrieben**, um die Kategorie zu erweitern und anschließend auf die Zusatzgeräte der Klasse PowerFlex 4 zuzugreifen.
9. Doppelklicken Sie auf das 22-COMM-E-EtherNet-/IP-Adapter-Benutzerhandbuch, um die PDF-Datei zu öffnen.

So rufen Sie Handbücher aus der Rockwell Automation Literaturbibliothek auf:

1. Wechseln Sie zu http://literature.rockwellautomation.com.

2. Um auf das nicht englische Sprachversionen der Benutzerhandbücher zuzugreifen, wählen Sie die Sprache im Dropdown-Feld „Publikationssprache“ (rechts) aus.

3. Geben Sie in das Feld **Suche** die Gerätekatalognummer vollständig oder teilweise ein. Geben Sie z. B. „2080-LC30“ ein, um Micro830-Benutzerhandbücher anzuzeigen.
4. Geben Sie im Feld **Suche** die Katalognummer vollständig oder teilweise ein. Geben Sie beispielsweise "2080-LC30" ein, wenn Sie Micro830-Benutzerhandbücher anzeigen möchten.

CIP Register-Objektdaten

MSG_CIPGENERIC-Funktionsblöcke verwenden die CIP Register-Objektdaten im AppCfg-Parameter. Die Objektdaten beinhalten Folgendes:

- Klassencode
- Instanz
- Instanzattribut
- Dienst

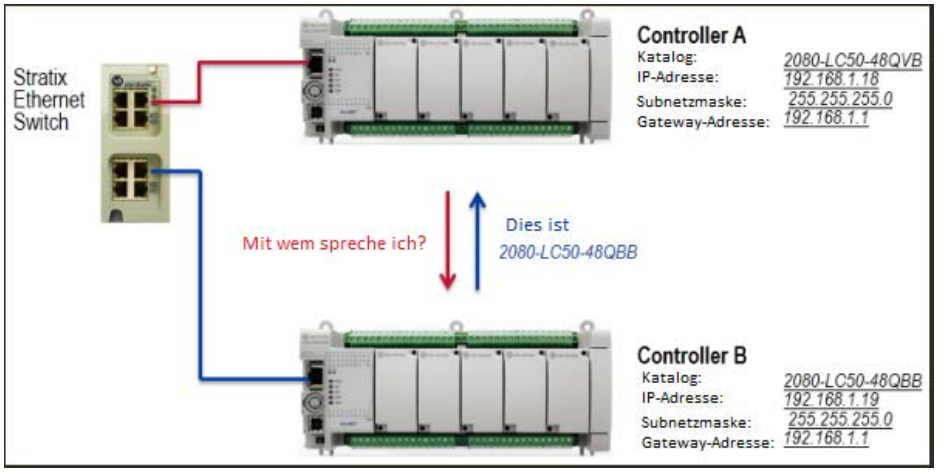
Werte für den MSG_CIPGENERIC AppCfg-Parameter

Verwenden Sie die Werte aus dem CIP Register-Objekt in Ihren Eingabevariablen, um die Parameter des MSG_CIPGENERIC-Funktionsblocks zu konfigurieren. Die folgende Abbildung zeigt die in den MSG_CIPGENERIC-Funktionsblockparametern verwendeten CIP Register-Objektdatenwerte.

Name	Alias	Datentyp	Dimension	Projektwert	Startwert
MSG_ReadDrive		MSG_CIPGENE		...	...
MyAppCfg		CIPAPPCFG		...	...
MyAppCfg.Service		USINT			14
MyAppCfg.Class		UINT			7
MyAppCfg.Instance		UDINT			4
MyAppCfg.Attribute		UINT			4
MyAppCfg.MemberCnt		USINT			

Beispiel: So erstellen Sie ein MSG_CIPGENERIC-Nachrichtenprogramm zum Lesen von Daten von einem Controller

In diesem Beispiel wird veranschaulicht, wie Sie ein Nachrichtenprogramm erstellen können, um mithilfe eines MSG_CIPGENERIC-Funktionsblocks und eines COP-Funktionsblocks Kataloginformationen auf Controller B von Controller A abzurufen.



Führen Sie die folgenden Aufgaben durch, um ein MSG_CIPGENERIC-Nachrichtenprogramm zu erstellen, das Informationen von einem anderen Controller liest.

Nein	Aufgabe
1	Ermitteln von Anfangswerten für die Eingabevariablen (MSG_CIPGENERIC) auf seite 218
2	Hinzufügen eines MSG_CIPGENERIC-Funktionsblocks und -Variablen auf seite 219
3	Konfigurieren von Anfangswerten für Variablen auf seite 220
4	Hinzufügen eines Kontakts und einer Spule auf seite 222
5	Hinzufügen eines COP-Funktionsblocks, von Variablen und eines Kontakts (MSG_CIPGENERIC) auf seite 223
6	Verifizieren der korrekten IP-Konfiguration auf Controller B auf seite 225

Ermitteln von Anfangswerten für die Eingangsvariablen, MSG_CIPGENERIC

Führen Sie die folgenden allgemeinen Schritte aus, um Eingabevariablen und Anfangswerte hinzuzufügen, und rufen Sie die Werte des Identitätsobjekts ab, um die Anfangswerte für den AppCfg-Parameter zu konfigurieren.

So fügen Sie Eingabevariablen und Anfangswerte hinzu:

1. Klicken Sie im Menü **Hilfe** auf **Benutzerhandbücher**.
2. Erweitern Sie die Auswahl „Antriebe“ und suchen Sie das Benutzerhandbuch für den Typ des Kommunikationsadapters, den Sie verwenden (Benutzerhandbuch für EtherNet/IP-Adapter).
3. Doppelklicken Sie auf das Benutzerhandbuch, um es zu öffnen.
4. Überprüfen Sie die Überschriften des Appendix, um den Abschnitt zu finden, der die Informationen zu den EtherNet-/IP-Objekten enthält, auf die mit „Explizite Nachrichten“ (Appendix C) zugegriffen werden kann.
5. Gehen Sie zum Abschnitt des Appendix und ermitteln Sie den mit Ihrer „Expliziten Nachricht“ („Identitätsobjekt“) verwandten Objekttyp.
6. Identifizieren Sie auf der Grundlage der dann abgerufenen Information über den Typ die Anfangswerte für die Parameter „AppCfg“.

Beispiel für Ethernet-/IP-Objektdaten und „AppFCG“-Parameter

In der folgenden Tabelle werden die Ethernet-/IP-Objektdaten bezeichnet, die eingesetzt werden, um Kataloginformationen von einem Controller auszulesen.

Beispiel einer Eingabevariable	AppCfg-Parameter	Ethernet-/IP-Objektdatenoption	Beschreibung	Anfangswert
MyAppCfg.Service	Dienst	Dienstcode	Für Klasse = Ja implementieren Für Instanz = Ja implementieren Attribut „Single“ abrufen	14 (0x0E hexadezimal)
MyAppCfg.Class	Klasse	Klassencode	EtherNet-/IP-Objektklasse = Identitätsobjekt	01
MyAppCfg.Instance	Instanz	Instanzen	22-COMM-E	01
MyAppCfg.Attribute	Attribut	Instanzattribut	Produktname und -bewertung als SHORT STRING abrufen	07

Hinzufügen eines MSG_CIPGENERIC-Funktionsblocks und -Variablen

Um einen MSG_CIPGENERIC-Funktionsblock zu einem Kontaktplan-Programm und dann Eingangsvariablen zu dem Funktionsblock hinzuzufügen, befolgen Sie die nachstehenden Schritte.

So fügen Sie einen MSG_CIPGENERIC-Funktionsblock hinzu:

1. Hinzufügen eines Controllers:
   - Erweitern Sie den Ordner **Controller** und den Ordner „Micro850“, um alle Micro850-Controller anzuzeigen.
   - Doppelklicken Sie auf einen Controller (2080-LC50-48QVB), um ihn zum **Projekt-Organizer** hinzuzufügen.
2. Hinzufügen eines Kontaktplan-Programms:
   - Klicken Sie im **„Projekt-Organizer“** mit der rechten Maustaste auf **Programme**, klicken Sie auf **Hinzufügen** und dann auf **Neuer KOP: Kontaktplan**.
   - Klicken Sie im **Projekt-Organizer** mit der rechten Maustaste auf das Symbol des Kontaktplans, klicken Sie auf **Umbenennen** und geben Sie „CIPExplicitMessage“ ein.
   - Doppelklicken Sie im **Projekt-Organizer** auf das Kontaktplan-Programm, um die Programmorganisationseinheit (POE) der Kontaktplanelemente (KOP) im Sprach-Editor anzuzeigen.
3. Hinzufügen des MSG_CIPGENERIC-Funktionsblocks:
   - Um den **Anweisungsblockauswahl** anzuzeigen, wählen Sie in der **Toolbox Anweisungsblock** aus und ziehen ihn per Drag-and-Drop auf den Kontaktplan.
   - Geben Sie im Feld „Suche“ **MSG** ein, um die Nachrichtenfunktionsblöcke anzuzeigen.
   - Geben Sie **MSG_ReadDrive** in das Feld **Instanz** ein.

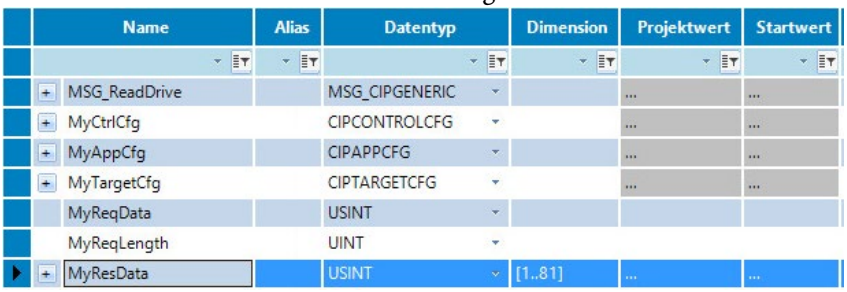
- Doppelklicken Sie auf **MSG_CIPGENERIC**, um eine Instanz des Funktionsblocks zum Kontaktplan hinzuzufügen.

4. Hinzufügen lokaler MSG_CIPGENERIC-Eingangsvariablen:
   - Doppelklicken Sie im **Projekt-Organizer** auf **Lokale Variablen**, um die Seite **Lokale Variablen** anzuzeigen.
   - Fügen Sie auf der Seite **Variablen** die in der Tabelle aufgeführten Variablen und Datentypen hinzu.

Parameter	Variablenname	Datentyp
CtrlCfg	MyCtrlCfg	CIPCONTROLCFG
AppCfg	MyAppCfg	CIPAPPCFG
TargetCfg	MyTargetCfg	CIPTARGETCFG
ReqData	MyReqData	USINT
ReqLength	MyReqLength	UINT
ResData	MyResData	USINT (Array)

5. Für die „MyResData“-Variable doppelklicken Sie auf **Dimensionen** und ändern die Arraygröße auf [1..81].

   Die Seite **Variablen** sollte wie in dem folgenden Bild aussehen.



Name	Alias	Datentyp	Dimension	Projektwert	Startwert
MSG_ReadDrive		MSG_CIPGENERIC		...	...
MyCtrlCfg		CIPCONTROLCFG		...	...
MyAppCfg		CIPAPPCFG		...	...
MyTargetCfg		CIPTARGETCFG		...	...
MyReqData		USINT			
MyReqLength		UINT			
MyResData		USINT	[1..81]	...	...

Konfigurieren von Anfangswerten für Variablen

Befolgen Sie diese Schritte, um Anfangswerte zu Eingabevariablen hinzuzufügen, die Sie zuvor erstellt haben; weisen Sie die Variablen anschließend dem richtigen Eingabeparameter des MSG_CIPGENERIC-Funktionsblocks zu.

So weisen Sie Variablen zu MSG_CIPGENERIC zu:

1. So konfigurieren Sie Anfangswerte für die Eingangsvariable „MyCtrlCfg“:
   - Erweitern Sie auf der Seite **Lokale Variablen** die Option MyCtrlCfg, um die entsprechenden Parameter anzuzeigen.
   - Geben Sie in der Spalte **Anfangswert** für jeden Parameter die folgenden Werte ein.

Parameter	Anfangswert	Kommentare
MyCtrlCfg.Cancel	Leer lassen	Nicht benötigt.
MyCtrlCfg.TriggerType	0	Wir müssen die Katalognummer nur einmal abrufen.
MyCtrlcfg.StrMode	Leer lassen	Nicht benötigt.

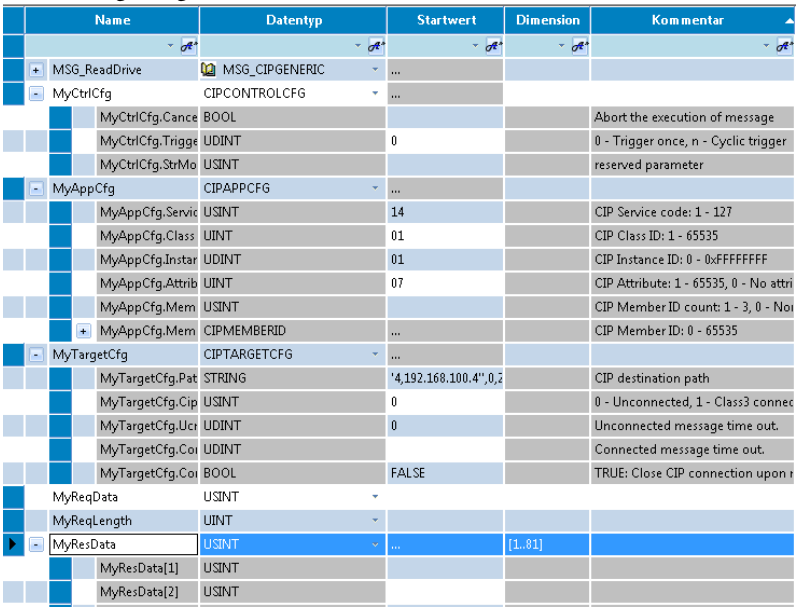
2. So konfigurieren Sie Anfangswerte für die Eingabevariable „MyAppCfg“
   - Erweitern Sie auf der Seite **Lokale Variablen** die Option MyAppCfg, um die entsprechenden Parameter anzuzeigen.
   - Geben Sie in der Spalte **Anfangswert** für jeden Parameter die folgenden Werte ein.

Parameter	Anfangswert
MyAppCfg.Service	14
MyAppCfg.Class	01
MyAppCfg.Instance	01
MyAppCfg.Attribute	07

3. So konfigurieren Sie Anfangswerte für die Eingabevariable „MyTargetCfg“
   - Erweitern Sie auf der Seite **Lokale Variablen** die Option MyTargetCfg, um die entsprechenden Parameter anzuzeigen.
   - Geben Sie in der Spalte **Anfangswert** für jeden Parameter die folgenden Werte ein.

Parameter	Anfangswert	Kommentare
MyTargetCfg.Path	'4,192.168.100.4'	Die erste „4“ zeigt die Nachricht an, die über den integrierten Ethernet-Anschluss versandt wird. 192.168.100.4 ist die IP-Adresse der Ethernet-Schnittstelle des Antriebs.
MyTargetCfg.CipConnMode	0	Bei CIP-Nachrichten wird „Nicht verknüpft“ bevorzugt.
MyTargetCfg.UcmmTimeout	Leer	Nicht verknüpfte Nachrichten besitzen eine standardmäßige Zeitüberschreitung von 3000 Millisekunden, wenn ihre „Anfangswerte“ leer sind.
MyTargetCfg.ConnMsgTimeout	Leer	Verknüpfte Nachrichten besitzen eine standardmäßige Zeitüberschreitung von 3000 Millisekunden, wenn ihre „Anfangswerte“ leer sind.
MyTargetCfg.ConnClose	FALSE	Für eine verknüpfte Nachrichtenübertragung könnte die CIP-Verbindung sofort nach Abschluss der Nachrichtenanweisung durch Einstellen des „Anfangswert“ auf „TRUE“ geschlossen werden.

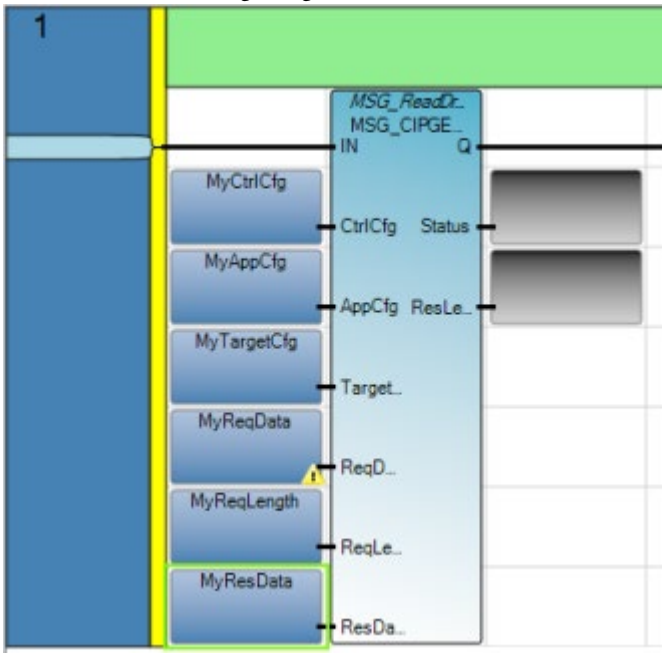
Die Parameter auf der Seite **Variablen** sollten ähnlich wie in der Abbildung dargestellt aussehen.



4. So weisen Sie die Variablen den Parametern zu
   - Klicken Sie im Kontaktplan zu Ihrer Programmorganisationseinheit (POE) auf den oberen Teil des Variableneingabeblocks, um die Dropdown-Liste der Variablen anzuzeigen.
   - Aus jeder Liste weisen Sie jeden Eingabeparameter gemäß folgender Tabelle seiner richtigen Eingabevariable zu.

Parameter	Eingabevariable	Kommentare
CtrlCfg	MyCtrlCfg	Die Katalognummer darf nur einmal abgerufen werden, damit der Anfangswert „MyCtrlCfg.TriggerType“ auf 0 gesetzt wird.
AppCfg	MyAppCfg	Die Anfangswerte werden durch Nachschlagen der Objektdatenwerte für „Dienst“, „Klasse“, „Instanz“ und „Attribut“ bestimmt.
Ziel	MyTargetCfg	Die Anfangswerte sind für die Zielgerätekonfiguration.
ReqData	MyReqData	Da dies eine „Gelesen“-Nachricht ist, gibt es keine Anforderungsdaten und die Parameter „ReqData“ werden nicht verwendet.
ReqLength	MyReqLength	Da dies eine „Gelesen“-Nachricht ist, gibt es keine Anforderungsdaten und die Parameter „ReqLength“ werden nicht verwendet.
ResData	MyResData	Die Zeichenfolge der Katalognummer wird im Array im kurzen ODVA-„String“-Format gespeichert. Das erste Arrayelement definiert die Länge der Stärke und der Rest speichert den hexadezimalen Wert des Stringzeichens. Die maximale Anzahl der Zeichen beträgt 80, plus Längenelement, sodass „MyResData“ definiert ist als Array mit 1 Dimension mit 81 Elementen.

Die Instanz des MSG_CIPGENERIC-Funktionsblocks sollte ähnlich wie in der Abbildung dargestellt aussehen.



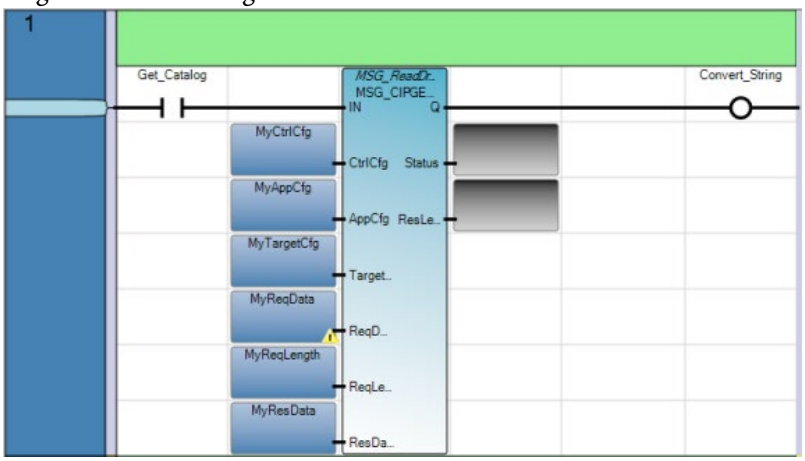
Hinzufügen eines Kontakts und einer Spule

Verwenden Sie die folgenden Schritte, um eine Spule und einen Kontakt zur MSG_CIPGENERIC-Anweisung hinzuzufügen, die Kataloginformationen in einen von Menschen lesbaren String konvertiert.

So fügen Sie eine Spule zu MSG_CIPGENERIC hinzu:

1. Wählen Sie in der **Toolbox** die Option **Direkter Kontakt** aus und ziehen Sie das Element per Drag-and-Drop links neben den MSG_CIPGENERIC-Funktionsblockeingang auf dem ersten Kontaktplan.
2. Geben Sie in der „Variablenauswahl“ im Feld „Name“ des Kontakts **Get_Catalog** ein.
3. Wählen Sie in der **Toolbox** die Option **Direkte Spule** aus und ziehen Sie das Element per Drag-and-Drop rechts neben den MSG_CIPGENERIC-Funktionsblockausgang auf dem ersten Kontaktplan.
4. Geben Sie in der „Variablenauswahl“ im Feld „Name“ der Spule **Convert_String** ein.

   Der erste Strompfad Ihres Kontaktplan-Programms für MSG_CIPGENERIC-Nachrichtenübertragung sollte dem in der folgenden Abbildung ähneln.



Hinzufügen eines COP-Funktionsblocks, von Variablen und eines Kontakts (MSG_CIPGENERIC)

Befolgen Sie diese Schritte, um einen COP-Funktionsblock, Variablen und einen Kontakt hinzuzufügen. Die COP-Anweisung dient der Konvertierung von Daten vom Quelldatentyp (zum Beispiel DINT oder REAL) in einen Zieldatentyp. In diesem Beispiel wird die Kataloginformation in einen von Menschen lesbaren String konvertiert.

So fügen Sie einen COP-Funktionsblock hinzu:

1. Wählen Sie in der **Toolbox** die Option **Strompfad**, und kopieren Sie ihn unter den ersten Kontaktplan, um einen zweiten Kontaktplan hinzuzufügen.
2. Hinzufügen des COP-Funktionsblocks:
   - Wählen Sie in der **Toolbox** den **Block** aus, und ziehen Sie das Element per Drag-and-Drop auf den zweiten Kontaktplan, um den **Anweisungsblockselektor** anzuzeigen.

- Doppelklicken Sie auf **COP**, um eine Instanz des Funktionsblocks zum Kontaktplan hinzuzufügen.

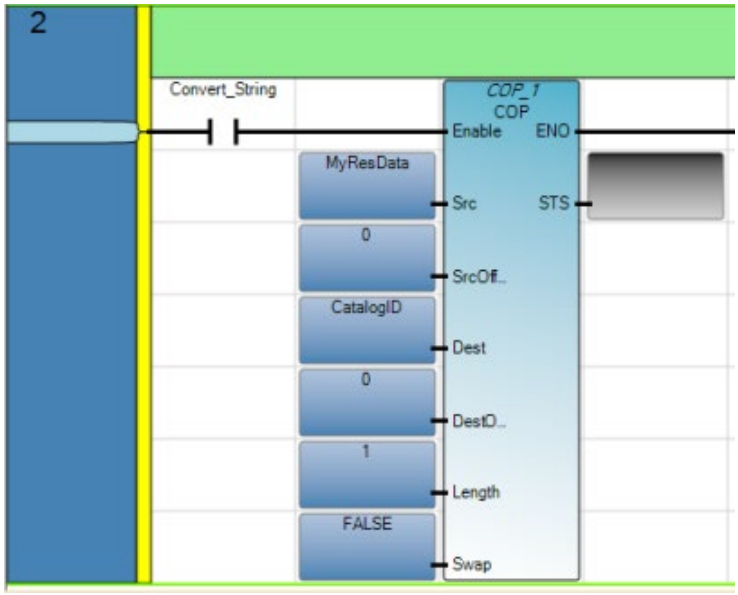
3. Fügen Sie lokale Eingabevariablen für COP hinzu:
   - Doppelklicken Sie im **Projekt-Organizer** auf **Lokale Variablen**, um die Seite **Lokale Variablen** anzuzeigen.
   - Fügen Sie auf der Seite **Lokale Variablen** die in der folgenden Tabelle aufgeführten Variablen und Datentypen hinzu.

Parameter	Variablenname	Datentyp
Src	MyResData	Array USINT
SrcOffset	0	UINT
Dest	CatalogID	Array STRING
DestOffset	0	UINT
Length	1	UINT
Swap	FALSE	BOOLEAN

4. Für die „CatalogID“-Variable doppelklicken Sie auf **Dimension** und ändern die Arraygröße auf [1..1].
5. Hinzufügen eines Kontakts:
   - Wählen Sie in der **Toolbox** die Option **Direkter Kontakt** aus, und ziehen Sie das Element per Drag-and-Drop links neben den COP-Funktionsblockeingang auf dem zweiten Kontaktplan.
   - Wählen Sie in der „Variablenauswahl“ die Variable **Convert_String** für den Kontakt.

Ergebnis

Der zweite Strompfad Ihres Kontaktplan-Programms für MSG_CIPGENERIC-Nachrichtenübertragung sollte dem in der folgenden Abbildung ähneln.



Verifizieren der korrekten IP-Konfiguration auf Controller B

Befolgen Sie diese Schritte, um Richtigkeit der IP-Adresseinstellungen auf Controller B zu überprüfen.

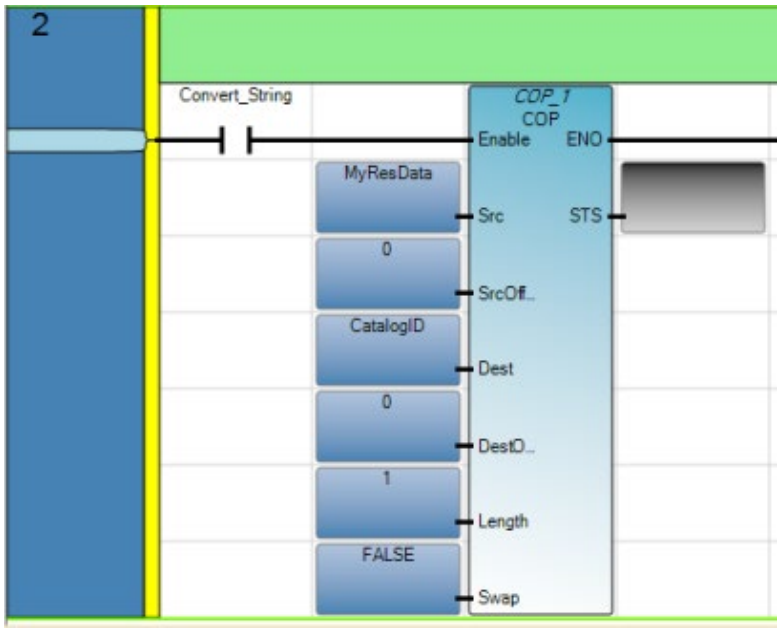
So überprüfen Sie die IP-Adresse:

1. Öffnen des Anwendungsarbeitsbereichs für den Controller:
2. Doppelklicken Sie im **Projekt-Organizer** auf den Controller, um den Controller-Arbeitsbereich zu öffnen.
3. Erweitern Sie **Ethernet** in der **Controller**-Baumstruktur des Controller-Arbeitsbereichs und klicken dann auf „Internet-Protokoll“, um die Konfigurationsseite „Controller“ anzuzeigen.
4. Überprüfen Sie, ob die IP-Adresseinstellungen korrekt sind und den Angaben in der folgender Tabelle entsprechen.

IP-Konfigurationsoption	Wert
IP-Adresse	192.168.1.19
Subnetzmaske	255.255.255.0
Gateway-Adresse	192.168.1.1

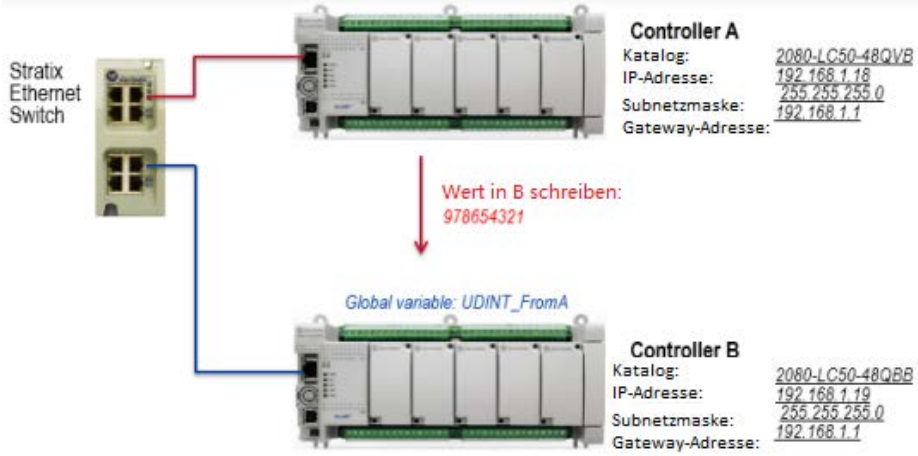
Ergebnisse

Die „Internet-Protokoll“-Optionen auf Ihrer Konfigurationsseite „Controller“ sollten denen in der folgenden Abbildung ähneln.



Beispiel: So erstellen Sie ein MSG_CIPSYMBOLIC-Nachrichtenprogramm zum Schreiben eines Wertes in eine Variable

In diesem Beispiel erfahren Sie, wie Sie ein Nachrichtenprogramm erstellen, um einen Wert aus Controller A in eine globale Variable für Controller B zu schreiben.



Um ein MSG_CIPSYMBOLIC-Messaging-Programm zu erstellen, das eingesetzt werden kann, um einen Wert in eine globale Variable eines anderen Controllers zu schreiben, führen Sie folgende Aktionen durch:

Nein	Aufgabe
1	Hinzufügen eines COP-Funktionsblocks, Variablen und eines Kontakts (MSG_CIPSYMBOLIC) auf seite 226
2	Hinzufügen eines „Gleich“-Operators und einer Spule auf seite 228
3	Hinzufügen eines MSG_CIPSYMBOLIC-Funktionsblocks, der Variablen und eines Kontakts auf seite 229
4	Verifizieren der korrekten IP-Konfiguration auf Controller B auf seite 225
5	Erzeugen einer globalen Variable für Controller B auf seite 232
6	Überprüfen der vollständigen Programmergebnisse auf seite 233

Hinzufügen eines COP-Funktionsblocks, von Variablen und eines Kontakts (MSG_CIPSYMBOLIC)

Die COP-Anweisung dient dazu, die von Ihnen eingegebenen Daten in den Zieldatentyp zu konvertieren, damit die Daten mit der Controller-Variable kompatibel sind.

So fügen Sie einen COP-Funktionsblock hinzu:

1. Hinzufügen eines Controllers:
   - Erweitern Sie den Ordner **Controller** und den Ordner „Micro850“, um alle Micro850-Controller anzuzeigen.
   - Doppelklicken Sie auf einen Controller (2080-LC50-48QVB), um ihn zum **Projekt-Organizer** hinzuzufügen.
2. Hinzufügen eines Kontaktplan-Programms:
   - Klicken Sie im **„Projekt-Organizer“** mit der rechten Maustaste auf **Programme**, klicken Sie auf **Hinzufügen** und dann auf **Neuer KOP: Kontaktplan**.
   - Klicken Sie im „Projekt-Organizer“ mit der rechten Maustaste auf **Projekt-Organizer**, klicken Sie auf **Umbenennen** und geben Sie **CIPSymbolicMessage** ein.
   - Doppelklicken Sie im **Projekt-Organizer** auf das Kontaktplan-Programm, um die Programmorganisationseinheit

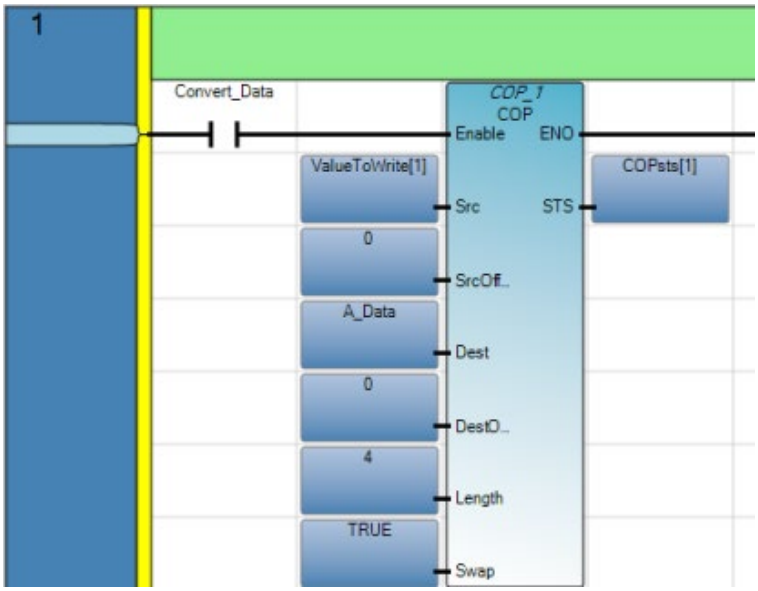
(POE) der Kontaktplanelemente (KOP) im Sprach-Editor anzuzeigen.

3. Hinzufügen eines COP-Funktionsblocks:
   - Wählen Sie in der **Toolbox** den **Anweisungsblock** aus, und ziehen Sie ihn per Drag-and-Drop auf den ersten Kontaktplan, um den **Anweisungsblockselektor** anzuzeigen.
   - Geben Sie in „Suche“ **COP** ein, und doppelklicken Sie auf **COP**, um eine Instanz des Funktionsblocks zum Kontaktplan hinzuzufügen.
4. Hinzufügen von COP-Variablen und Anfangswerten:
   - Doppelklicken Sie in der Programmorganisationseinheit (POE) des Kontaktplans auf **Lokale Variablen**, um die Seite **Lokale Variablen** anzuzeigen.
   - Fügen Sie auf der Seite **Variablen** die in der folgenden Tabelle aufgeführten Variablen und Datentypen hinzu.
5. Erstellen von Arrays:
   - Für „ValueToWrite“ doppelklicken Sie auf **Dimensionen** und ändern Sie die Arraygröße auf [1 - 1].
   - Für „A_Data“ doppelklicken Sie auf **Dimensionen** und ändern die Arraygröße auf [1 - 4].
6. Geben Sie die Daten aus der Spalte **Wert** der folgenden Tabelle für jede Variable in das Feld **Anfangswert** ein.
7. Hinzufügen eines Kontakts:
   - Wählen Sie in der „Toolbox“ die Option **Direkter Kontakt** aus, und ziehen Sie das Element per Drag-and-Drop links neben den COP-Funktionsblockeingang auf dem ersten Kontaktplan.
   - Weisen Sie dem Kontakt im „Variablenselektor“ eine Variable mit dem Namen **Convert_Data** zu.

Verwenden Sie die Variablen, die in der Tabelle für die COP-Funktionsblock definiert sind.

Parameter	Variablenname	Datentyp
Src	ValueToWrite	Array UDINT Anfangswert: 987654321
SrcOffset	0	UINT
Dest	A_Data	Array USINT
DestOffset	0	UINT
Length	4	UINT
Swap	TRUE	BOOLEAN
STS	COPsts	Array UINT

Der erste Strompfad Ihres Kontaktplan-Programms für MSG_CIPSYMBOLIC-Nachrichtenübertragung sollte dem in der folgenden Abbildung ähneln.



Hinzufügen eines „Gleich“-Operators und einer Spule

Die Anweisung „Gleich“ wird verwendet, um das Schreiben eines Werts auszulösen, wenn die Datentypkonversion erfolgreich war. Um einen „Gleich“ (=)-Operator, Variablen und eine Spule hinzuzufügen, befolgen Sie die nachstehenden Schritte.

So fügen Sie einen „Gleich“-Operator hinzu:

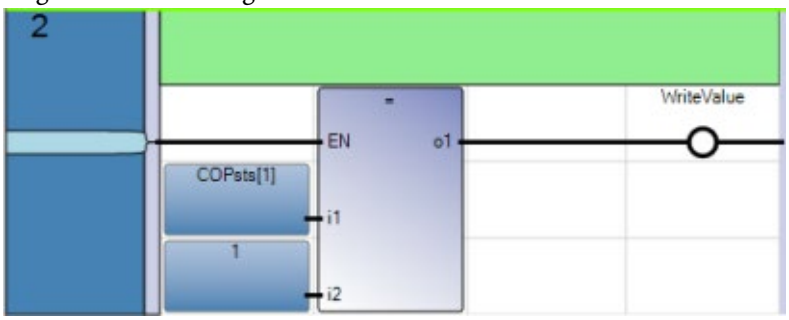
1. Wählen Sie in der **Toolbox** die Option **Strompfad**, und kopieren Sie ihn unter den ersten Kontaktplan, um einen zweiten Kontaktplan hinzuzufügen.
2. So fügen Sie einen „Gleich“-Operator hinzu:
   - Wählen Sie in der **Toolbox** den **Anweisungsblock** aus, und ziehen Sie das Element per Drag-and-Drop auf den zweiten Kontaktplan, um den **Anweisungsblockselektor** anzuzeigen.
   - Geben Sie in „Suche“ das Zeichen „=“ ein, und doppelklicken Sie auf **„=“**, um eine Instanz des Operators zum Kontaktplan hinzuzufügen.
3. So fügen Sie „Gleich“-Variablen hinzu:
   - Doppelklicken Sie im Kontaktplan zu Ihrer Programmorganisationseinheit (POE) auf eine Variable, um die **Variablenauswahl** anzuzeigen.
   - Weisen Sie in der **Variablenauswahl** Variablennamen gemäß der Auflistung in folgender Tabelle zu.

Parameter	Variablenname
i1	COPsts
i2	1

4. So fügen Sie eine Spule zum „Gleich“-Operator hinzu:

   Wählen Sie in der **Toolbox** die Option **Direkte Spule** aus, und kopieren Sie sie auf die rechte Seite des „Gleich“-Operatorausgangs auf dem zweiten Kontaktplan.
   Geben Sie in der **Variablenauswahl** im Feld „Name“ der Spule **WriteValue** ein.

   Der zweite Strompfad Ihres Kontaktplan-Programms für MSG_CIPGENERIC-Nachrichtenübertragung sollte dem in der folgenden Abbildung ähneln.



Hinzufügen eines MSG_CIPSYMBOLIC-Funktionsblocks, der Variablen und eines Kontakts

Um einen MSG_CIPSYMBOLIC-Funktionsblock, Eingangsvariablen und einen Kontakt zu einem Kontaktplan hinzuzufügen, befolgen Sie die nachstehenden Schritte.

So fügen Sie einen Funktionsblock und Variablen hinzu:

1. Wählen Sie in der **Toolbox Strompfad** und kopieren Sie ihn direkt unter den zweiten Kontaktplan, um einen dritten Strompfad hinzuzufügen.
2. Hinzufügen des MSG_CIPSYMBOLIC-Funktionsblocks:
   - Um den **Anweisungsblockauswahl** anzuzeigen, wählen Sie in der **Toolbox Anweisungsblock** aus und ziehen ihn per Drag-and-Drop auf den Kontaktplan.
   - Geben Sie im Feld „Suche“ **MSG** ein, um die Nachrichtenfunktionsblöcke anzuzeigen.
   - Geben Sie **MSG_WriteVariable** in das Feld **Instanz** ein.
   - Doppelklicken Sie auf **MSG_CIPSYMBOLIC**, um eine Instanz des Funktionsblocks mit dem Namen „MSG_WriteVariable“ zum Kontaktplan hinzuzufügen.
3. Zuweisen von Variablennamen:
   - Doppelklicken Sie im Kontaktplan zu Ihrer Programmorganisationseinheit (POE) auf eine Variable, um die **Variablenauswahl** anzuzeigen.
   - Weisen Sie in der **Variablenauswahl** Variablennamen gemäß der Auflistung in folgender Tabelle zu.
4. Für die „MyResData“-Variable doppelklicken Sie auf **Dimension** und ändern die Arraygröße auf [1...4].

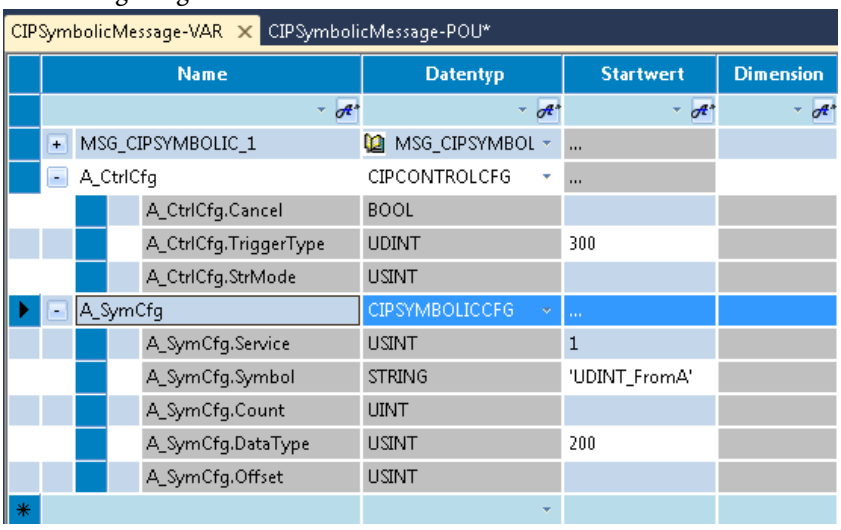
5. Konfigurieren von Anfangswerten für die lokalen Variablen und Hinzufügen von „CtrlCfg“-Anfangswerten:
   - Erweitern Sie auf der Seite **Lokale Variablen** den Parameter „CtrlCfg“, um zusätzliche Parameter anzuzeigen.
   - Geben Sie in der Spalte „Anfangswert“ für jeden Parameter die folgenden Werte ein.

Parameter	Anfangswert
A_CtrlCfg.Cancel	Leer lassen
A_CtrlCfg.TriggerType	300
A_Ctrlcfg.StrMode	Leer lassen

6. Hinzufügen von „SymCfg“-Anfangswerten:
   - Erweitern Sie auf der Seite **Lokale Variablen** den Parameter „SymCfg“, um zusätzliche Parameter anzuzeigen.
   - Geben Sie in der Spalte „Anfangswert“ für jeden Parameter die folgenden Werte ein.

Parameter	Anfangswert
A_SymCfg.Service	1
A_SymCfg.Symbol	'UDINT_FromA'
A_SymCfg.Count	Leer lassen
A_SymCfg.DataType	200
A_SymCfg.Offset	Leer lassen

Die Seite „Lokale Variablen“ sollte ähnlich wie in der folgenden Abbildung dargestellt aussehen.

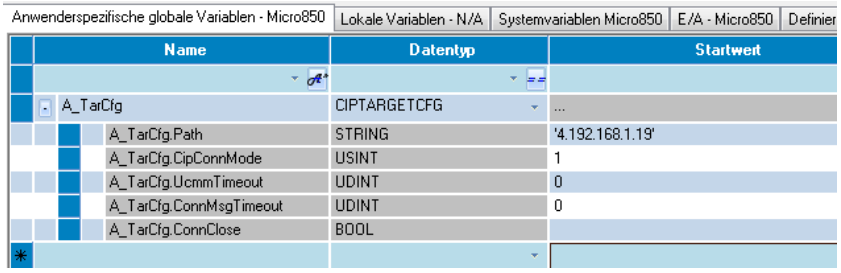


7. Konfigurieren von Anfangswerten für „TargetCfg“
   - Doppelklicken Sie im Kontaktplan zu Ihrer Programmorganisationseinheit (POE) auf die Variable „TargetCfg“, um den globalen „Variablenselektor“ zu öffnen.
   - Erweitern Sie den Parameter „TargetCfg“, um zusätzliche Parameter anzuzeigen.
   - Geben Sie in der Spalte „Anfangswert“ für jeden Parameter die folgenden Werte ein.

Parameter	Anfangswert

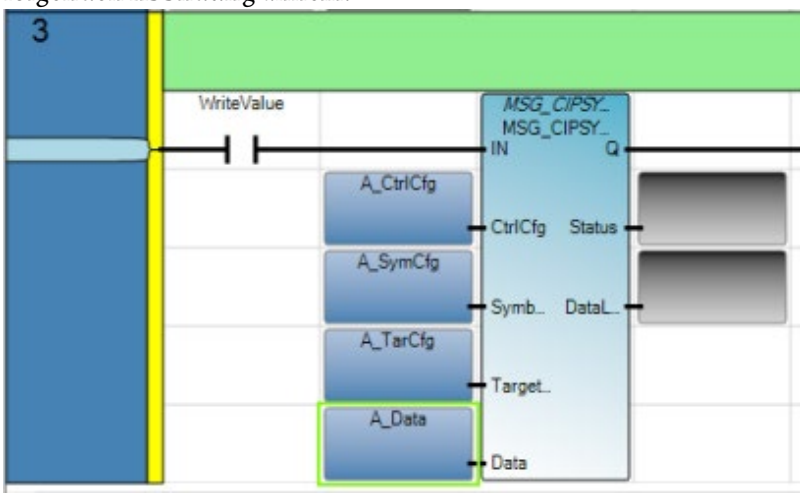
Parameter	Anfangswert
A_TarCfg.Path	'4,192.168.1.19'
A_TarCfg.CipConnMode	1
A_TarCfg.UcmmTimeout	0
A_TarCfg.ConnMsgTimeout	0
A_TarCfg.ConnClose	Leer lassen

„Globalen Variablenselektor verwenden“ sollte ähnlich wie in der folgenden Abbildung dargestellt aussehen.



8. Hinzufügen eines Kontakts:
   - Wählen Sie in der „Toolbox“ die Option **Direkter Kontakt** aus, und ziehen Sie das Element per Drag-and-Drop links neben den MSG_CIPSYMBOLIC-Funktionsblockeingang auf dem dritten Kontaktplan.
   - Weisen Sie dem Kontakt in der „Variablenauswahl“ die Variable **WriteValue** zu.

   Der dritte Strompfad Ihres Kontaktplan-Programms für MSG_CIPSYMBOLIC-Nachrichtenübertragung sollte dem in der folgenden Abbildung ähneln.



Verifizieren der korrekten IP-Konfiguration auf Controller B

Befolgen Sie diese Schritte, um Richtigkeit der IP-Adresseinstellungen auf Controller B zu überprüfen.

So überprüfen Sie die IP-Adresse:

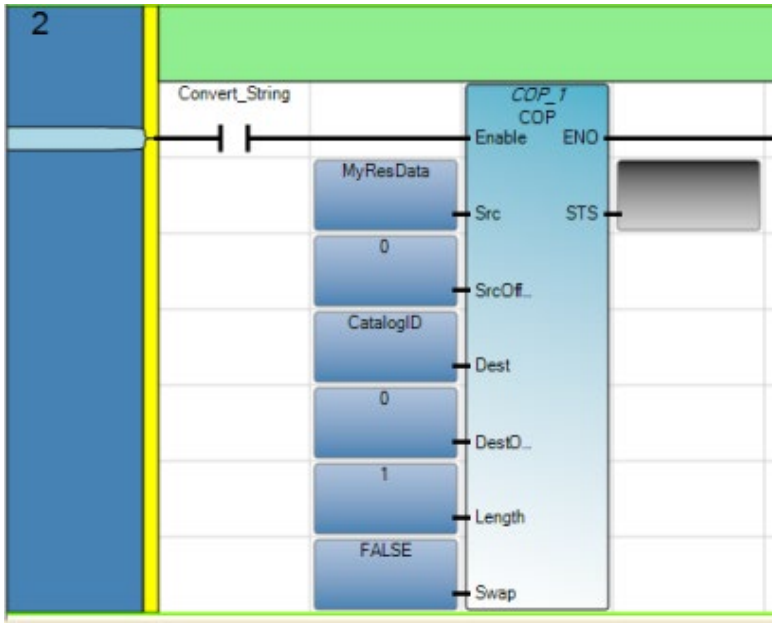
1. Öffnen des Anwendungsarbeitsbereichs für den Controller:

2. Doppelklicken Sie im **Projekt-Organizer** auf den Controller, um den Controller-Arbeitsbereich zu öffnen.
3. Erweitern Sie **Ethernet** in der **Controller**-Baumstruktur des Controller-Arbeitsbereichs und klicken dann auf „Internet-Protokoll“, um die Konfigurationsseite „Controller“ anzuzeigen.
4. Überprüfen Sie, ob die IP-Adresseinstellungen korrekt sind und den Angaben in der folgender Tabelle entsprechen.

IP-Konfigurationsoption	Wert
IP-Adresse	192.168.1.19
Subnetzmaske	255.255.255.0
Gateway-Adresse	192.168.1.1

Ergebnisse

Die „Internet-Protokoll“-Optionen auf Ihrer Konfigurationsseite „Controller“ sollten denen in der folgenden Abbildung ähneln.



Erzeugen einer globalen Variable für Controller B

Befolgen Sie diese Schritte, um für Controller B eine globale Variable zu erstellen.

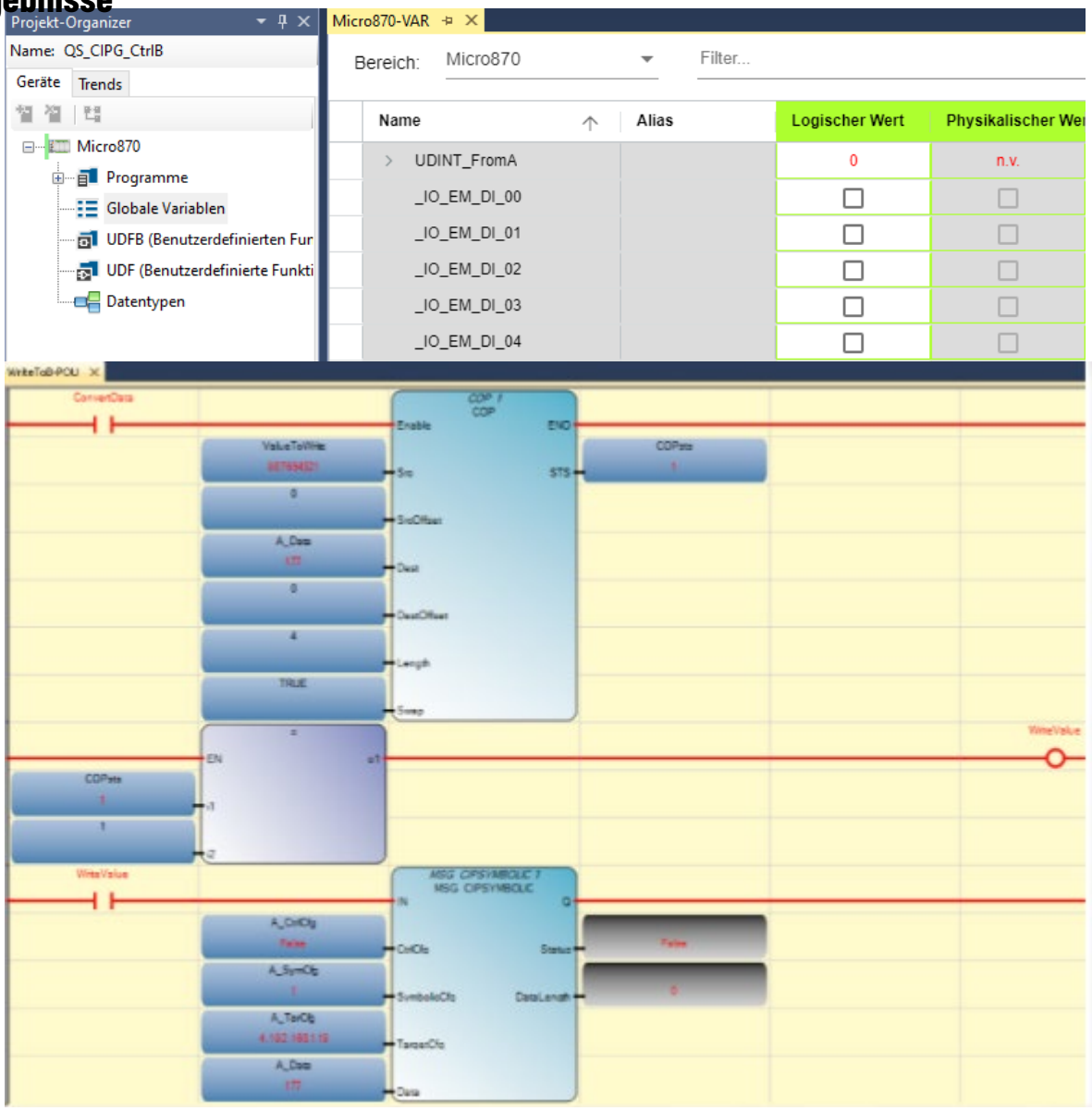
So erstellen Sie eine globale Variable:

1. Doppelklicken Sie im **Projekt-Organizer** auf **Globale Variablen**, um den Selektor **Globale Variablen** anzuzeigen.
2. Geben Sie in die Spalte **Name** UDINT_FromA ein.
3. Konfigurieren der verbleibenden Parameter:
   - Überprüfen Sie, ob der Datentyp „UDINT“ ist.
   - Überprüfen Sie, ob das **Lese-/Schreib**-Attribut ausgewählt ist.

Der Selektor **Globale Variablen** sollte ähnlich wie in der folgenden Abbildung dargestellt aussehen.

Überprüfen der vollständigen Programmergebnisse

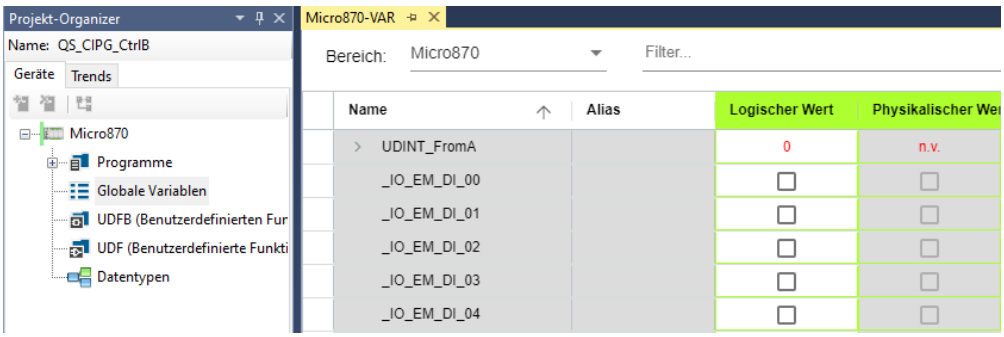
Im folgenden Beispiel wird das vollständige MSG_CIPSYMBOLIC-Nachrichtenprogramm angezeigt, nachdem es ausgeführt wurde.



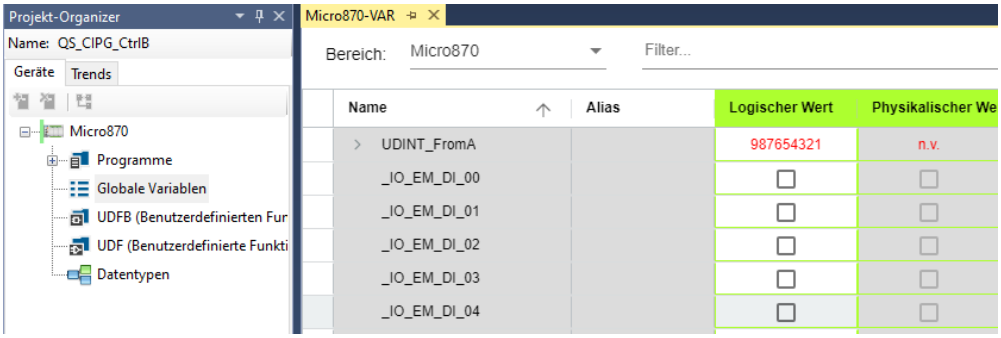
Ergebnisse von Controller B

Die beiden folgenden Abbildungen zeigen die Werte für Controller B zum Zeitpunkt vor und nach der Ausführung des Nachrichtenprogramms angezeigt.

Vor Ausführung des Programms



Nach Ausführung des Programms

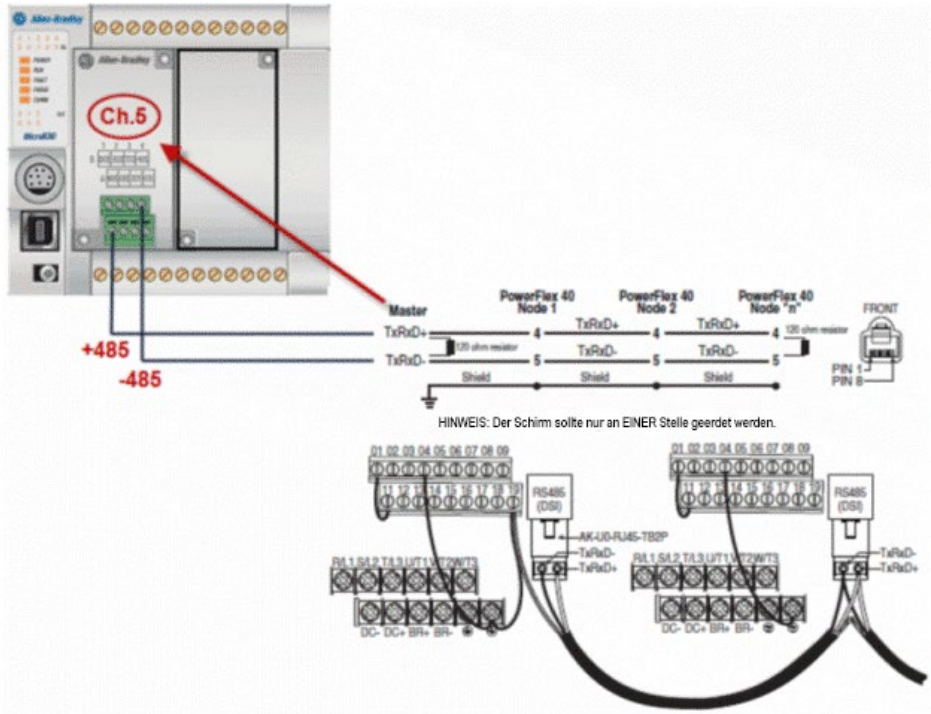


Beispiel: So konfigurieren Sie die Modbus-Kommunikation zum Lesen von und Schreiben auf ein Laufwerk

In diesen Beispielen wird beschrieben, wie die Modbus-Kommunikation konfiguriert werden muss, damit Statusdaten mithilfe des MSG_MODBUS auf seite 177-Befehls von einem PowerFlex 4-Antrieb ausgelesen und Steuerungsdaten in diesen Antrieb geschrieben werden können.

Verkabelung des Micro830

Dieses Beispiel geht von einem Micro830-Controller mit einem SERIALISOL-Modul aus, das über den ersten Steckplatz angeschlossen ist (Kanal 5). Ein einziger PowerFlex 40 ist verbunden; das folgende Diagramm gibt jedoch eine Anleitung für eine Mehrpunkt-Verkabelung. Weiterführende Informationen zum Verkabeln von Geräten entnehmen Sie dem Benutzerhandbuch.



AK-U0-RJ45-TB2P ist ein RJ45-Stecker mit 2 Klemmen für die RS485-Kommunikation

Beispiel für Modbus-Leseoperation

Mit der folgenden MSG_MODBUS-Anweisung können Statusdaten aus dem PowerFlex 40-Frequenzumrichter gelesen werden.

Frequenzumrichterstatus

„1807“ bedeutet, der Frequenzumrichter

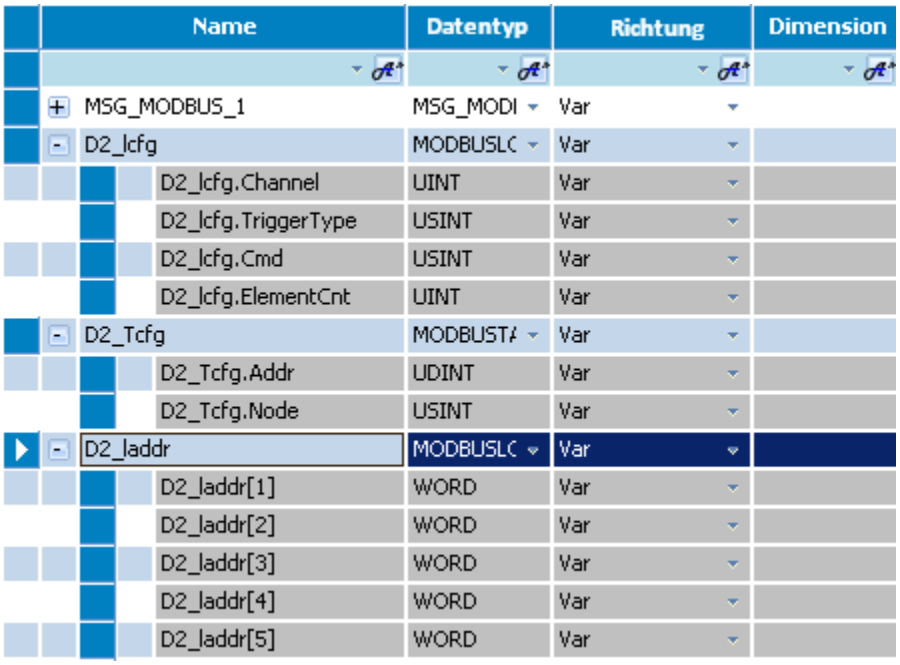
- ist bereit (Ready, Bit 0 ON)
- ist aktiv (Active, Bit 1 ON)
- hat den Startbefehl erhalten (Commanded Forward, Bit 2 ON)
- dreht vorwärts (Rotating Forward, Bit 3 ON)
- Status einiger Digitaleingänge des Frequenzumrichters

„278“ steht für 27,8 Hz.

Weitere Informationen über die Bits von Logikstatusworten, Fehlercode-Beschreibungen, Drehzahlsoll- und -istwerte sowie andere Statuscodes entnehmen Sie dem PowerFlex-Benutzerhandbuch.

Konfiguration einer MSG_MODBUS-Leseoperation

In der folgenden Abbildung werden die Variablenoptionen für den MSG_MODBUS-Anweisungsblock verwendet, um Statusdaten aus einem PowerFlex 40-Antrieb zu lesen.



Variablen der MSG_MODBUS-Leseoperation

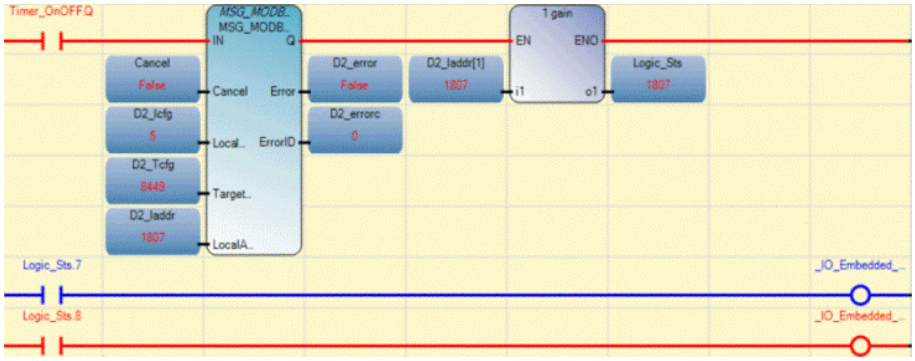
In der folgenden Tabelle werden die Variablen und Werte identifiziert, mit denen die MSG_MODBUS-Anweisung konfiguriert wird, um Statusdaten aus einem PowerFlex 4-Frequenzumrichter zu lesen.

Variable	Wert	Beschreibung
*.Channel	5	Kanal 5 - Anschlussort des SERIALISOL-Moduls
*.TriggerType	0	Auslösung beim Übergang von FALSE auf TRUE
*.Cmd	3	Modbus-Funktionscode “03” - Read Holding Registers
*.ElementCnt	4	Länge
*.Addr	8449	Adresse des PowerFlex-Logikstatusworts + 1

Variable	Wert	Beschreibung
*.Node	2	Adresse des PowerFlex-Knotens
*_laddr[1]	{data}	PowerFlex-Logikstatuswort
*_laddr[2]	{data}	PowerFlex-Fehlercode
*_laddr[3]	{data}	PowerFlex Soll-Drehzahl (Drehzahlsollwert)
*_laddr[4]	{data}	PowerFlex Drehzahlrückführung (Drehzahlistwert)

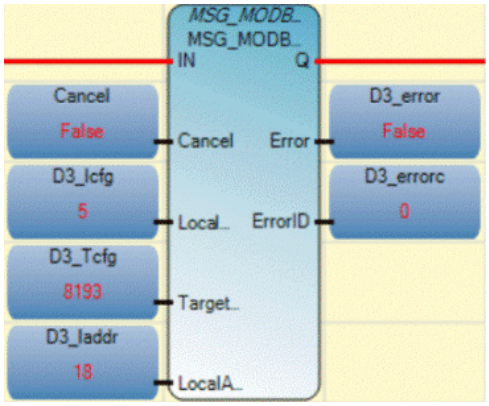
Beispiel einer MOV-Anweisung

In den folgenden Abbildungen sehen Sie ein Beispiel, wie Sie mithilfe der MOV-Anweisung den *_I[1]-Arraywert in ein Wort verschieben können, um den direkten Zugriff auf einzelne Bits zu ermöglichen.



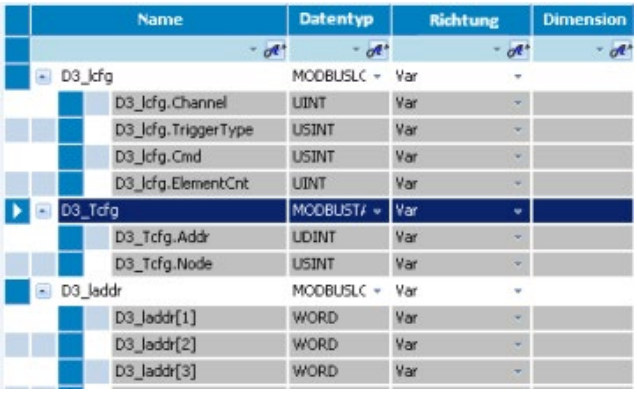
Beispiel für Modbus-Schreiboperation

Mit der folgenden MSG_MODBUS-Anweisung werden Steuerungsdaten an einen PowerFlex 40 geschrieben.



Konfiguration einer MSG_MODBUS-Schreiboperation

In der folgenden Tabelle werden die Variablen und Werte identifiziert, mit denen die MSG_MODBUS-Anweisung konfiguriert wird, um Steuerungsdaten an einen PowerFlex 4-Frequenzumrichter zu schreiben.



Variablen einer MSG_MODBUS-Schreiboperation

Die folgende Tabelle enthält eine Auflistung der Variablen mit ihren Werten und einer Beschreibung des jeweiligen Verwendungszwecks.

Variable	Wert	Beschreibung
*.Channel	5	Kanal 5 - Anschlussort des SERIALISOL-Moduls
*.TriggerType	0	Auslösung beim Übergang von FALSE auf TRUE
*.Cmd	16	Modbus-Funktionscode "16" - Write Holding Registers
*.ElementCnt	2	Länge
*.Addr	8193	Adresse des PowerFlex-Logikstatusworts + 1
*.Node	2	Adresse des PowerFlex-Knotens
*_Iaddr[1]	{data}	PowerFlex-Logikbefehlwort
*_Iaddr[2]	{data}	PowerFlex-Drehzahlreferenzwort

Kommunikationsprotokollunterstützung

Die MSG_CIP-Funktionsblöcke unterstützen verschiedene Kommunikationsprotokolle, wie in diesem Abschnitt beschrieben.

Funktionsblöcke unterstützen Nachrichtenkommunikationsprotokolle

Nachrichtenprotokoll	Kommunikationsmedien	Diesen Funktionsblock verwenden
Modbus/RTU-Client und -Server	Über einen seriellen Anschluss als Modbus-RTU konfiguriert	MSG_MODBUS auf seite 177
Modbus/TCP-Client und -Server	Über das Ethernet statt über einen seriellen Anschluss	MSG_MODBUS2 auf seite 183
Ethernet IP-Client und -Server	Über einen integrierten Ethernet-Kanal	MSG_CIPSYMBOLIC auf seite 173 MSG_CIPGENERIC auf seite 166
CIP seriell-Client und -Server	Ethernet-Kabel oder serielles Kabel	MSG_CIPSYMBOLIC auf seite 173
ASCII	Über einen mit einem ASCII-Treiber konfigurierten RS-232-Anschluss	Siehe unter ASCII-Befehle für seriellen Anschluss auf seite 103

Modbus-RTU

Modbus ist ein Master-Slave-Kommunikationsprotokoll im Halbduplexverfahren, das einem einzelnen Master die Kommunikation mit bis zu 247-Slave-Geräten ermöglicht. Der Modbus-Netzwerkmaster liest und schreibt Bits und Register. Modbus-RTU wird durch eine entsprechende Konfiguration des seriellen Anschlusses unterstützt.

Weitere Informationen zum Modbus-Protokoll finden Sie in den Modbus-Protokollspezifikationen (auf http://www.modbus.org).

Modbus/TCP

Das Modbus/TCP Server-Kommunikationsprotokoll verwendet dieselben Modbus-Mapping-Funktionen wie Modbus-RTU, wird jedoch über das Ethernet und nicht über einen seriellen Anschluss unterstützt.

Der Micro850-Controller unterstützt bis zu 16 gleichzeitige Modbus TCP Server-Verbindungen. Anstelle einer Protokollkonfiguration ist lediglich eine Konfiguration der Modbus-Mapping-Tabelle erforderlich.

EtherNet/IP

Die Controller Micro820 und Micro850 unterstützen bis zu 16 gleichzeitige EtherNet/IP-Serververbindungen über einen integrierten Ethernet-Kommunikationskanal auf seite 240. Mithilfe des Kanals kann ein Micro850-Controller über ein lokales Netzwerk bei einer Transferrate von 10 Mbps/100 Mbps mit verschiedenen Geräten verbunden werden.

Common Industrial Protocol (CIP)

Das CIP-Protokoll ist ein Transport- und Anwendungsschicht-Protokoll zum Versenden von Nachrichten über Ethernet/IP-, ControlNet- und DeviceNet-Netzwerke, das eine einfache Verbindung von HMIs mit einem Micro830- oder einem Micro850-Controller ermöglicht.

CIP-Explicit-Messaging

Das CIP-Protokoll wird für Explicit Messaging verwendet. Explicit Messaging (Konfiguration, Datensammlung und Diagnose) wird verwendet, um Daten zu übertragen, die keine kontinuierlichen Updates erfordern. Explicit Messaging wird durch den Einsatz von CIP Generic-Client-Messaging über den MSG_CIPGENERIC-Funktionsblock und durch den Einsatz von CIP Symbolic-Client-Messaging über den MSG_CIPSYMBOLIC-Funktionsblock unterstützt.

CIP seriell

CIP seriell verwendet das DF1 Full Duplex-Protokoll und ermöglicht Punkt-zu-Punkt-Verbindungen zwischen zwei Geräten. Es kombiniert Datentransparenz (American National Standards Institute ANSI - X3.28-1976-Spezifikation Unterkategorie D1) und eine gleichzeitige Übertragung in zwei Richtungen mit integrierten Antworten (Unterkategorie F1)

Die Controller Micro830 und Micro850 unterstützen CIP seriell über eine RS-232-Verbindung zu externen Geräten, z. B. zu Computern, auf denen RSLinx Classic-Software ausgeführt wird, PanelView Component-Terminals (ab Firmware-Version 1.70) oder sonstigen Controllern, die CIP seriell über DF1 Vollduplex unterstützen, wie z. B. ControlLogix- und CompactLogix-Controller mit integrierten seriellen Anschlüssen.

Der Treiber für serielle Anschlüsse kann über die Gerätekonfigurationsstruktur als CIP seriell, Modbus-RTU, ASCII oder Herunterfahren konfiguriert werden.

Integrierte Kommunikationskanäle

In die Micro820-, Micro830- und Micro850-Controller sind die folgenden zusätzlichen Kommunikationskanäle integriert:

- Ein nicht isolierter RS-232/485-Combo-Anschluss
- Ein nicht isolierter USB-Programmieranschluss (nur Micro830 und Micro850)
- Ein RJ-45-Ethernet-Anschluss (nur Micro820 und Micro850)

Vergleichsanweisungen

Verwenden Sie Vergleichsanweisungen, um Werte des Typs „Integer“, „Real“, „Time“, „Date“ und „String“ mithilfe eines Ausdrucks oder einer spezifischen Vergleichsanweisung zu vergleichen.

Anweisung	Beschreibung
(=) Gleich auf seite 241	Vergleicht den ersten Eingang mit dem zweiten Eingang, um zu prüfen, ob sie gleich sind. Für die Datentypen „Integer“, „Real“, „Time“, „Date“ und „String“.
(>) Größer als auf seite 244	Vergleicht Eingangswerte, um zu ermitteln, ob der erste größer als der zweite ist.
(>=) Größer als oder gleich auf seite 245	Vergleicht Eingangswerte, um zu ermitteln, ob der erste größer als der zweite ist oder identisch mit ihm.
(<) Kleiner als auf seite 246	Vergleicht Eingangswerte, um zu ermitteln, ob der erste kleiner als der zweite ist.
(<=) Kleiner als oder gleich auf seite 248	Vergleicht Eingangswerte, um zu ermitteln, ob der erste kleiner als der zweite ist oder identisch mit ihm.
(<>) Nicht gleich auf seite 249	Vergleicht Eingangswerte, um zu ermitteln, ob der erste sich vom zweiten unterscheidet.

Gleich

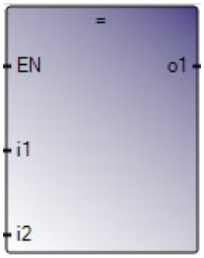
Vergleicht die erste Eingabe mit der zweiten Eingabe, um die Gleichheit bei den Datentypen „Integer“, „Real“, „Time“, „Date“ und „String“ festzustellen.

Details zum Betrieb:

- Bei TON-, TP- und TOF-Funktionsblöcken wird die Gleichheitsprüfung der „Time“-Werte nicht empfohlen.
- Der Datentyp „Real“ wird für den Vergleich der Werte auf Gleichheit nicht empfohlen, da Zahlen in der mathematischen Operation anders gerundet werden als die, die in der Anzeige der Variablenausgabe stehen. Daher können zwei Ausgabewerte in der Anzeige gleich aussehen, werden aber dennoch mit „false“ ausgewertet. Beispiel: Die Werte 23,500001 und 23,499999 werden in der Ansicht der Variablenausgabe beide mit 23,5 angezeigt, sind jedoch im Controller nicht gleich.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Funktion aktivieren. TRUE - der Gleichheitsvergleich wird ausgeführt. FALSE – es findet kein Vergleich statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	Alle Eingaben müssen denselben Datentyp haben. Die „Time"-Eingabe gilt für strukturierten Text, Kontaktpläne oder Funktionsblockdiagramme. Der Datentyp „Real" wird nicht empfohlen.
i2	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME	

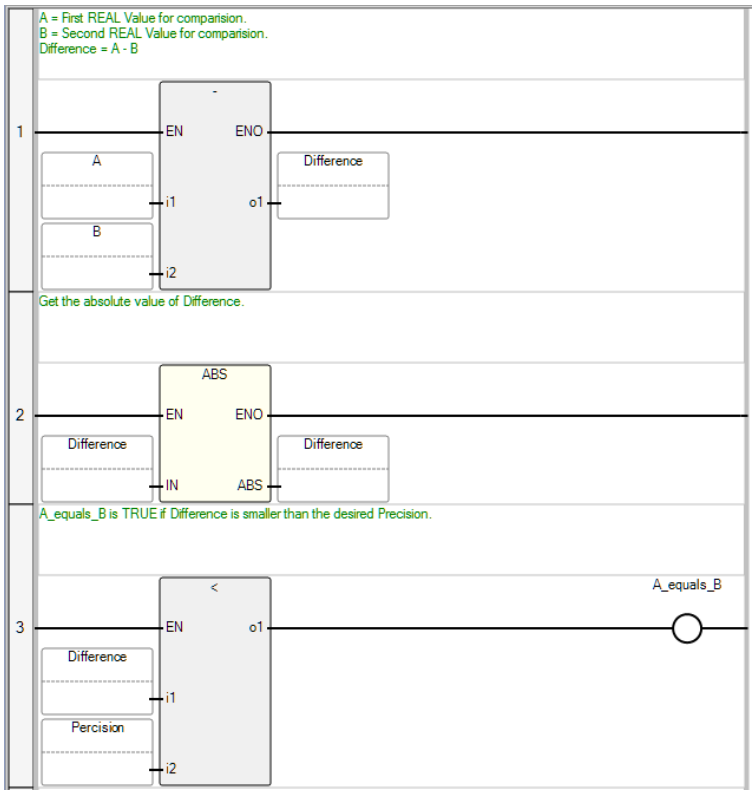
Parameter	Parametertyp	Datentyp	Beschreibung
		DATE (Datum) STRING	
o1	Ausgang	BOOL	TRUE, wenn i1 = i2.

Beispiel für den Vergleich von Real-Werten mittels Subtraktion (-) ABS und „Kleiner als“ (<)

Der Datentyp „Real“ sollte zum Vergleichen von Werten auf Gleichheit nicht verwendet werden, da es Unterschiede bei der Auf- und Abrundung von Zahlen gibt. Zwei Ausgabewerte mögen in einer Connected Components Workbench-Anzeige gleich aussehen, werden aber mit „false“ ausgewertet.

Beispiel: Die Werte 23,500001 und 23,499999 werden in der Ansicht der Variablenausgabe beide mit 23,5 angezeigt, sind jedoch im Controller nicht gleich.

Sie prüfen, ob zwei Real-Datentypen gleich sind, indem Sie den Unterschied zwischen den Werten mit einer Subtraktionsanweisung ermitteln und anschließend feststellen, ob der Unterschied kleiner ist als ein festgesetzter Genauigkeitswert. Schauen Sie sich das folgende Beispiel eines LD-Programms an, in dem zwei Werte vom Datentyp „Real“ verglichen werden.



Beispiel für „Gleich“ (=) in strukturiertem Text

(* ST-Äquivalenz: *)

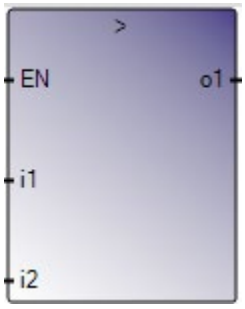
```
aresult := (10 = 25); (* aresult ist FALSE *)
mresult := ('ab' = 'ab'); (* mresult ist TRUE *)
```

Größer als

Vergleicht die Eingabewerte „Integer“, „Real“, „Time“, „Date“ und „String“, um zu ermitteln, ob der erste Wert größer ist als der zweite Wert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Eingabevergleich ausführen. FALSE – es findet kein Vergleich statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	Alle Eingaben müssen denselben Datentyp haben.
i2	Eingang	SINT USINT - BYTE INT	

Parameter	Parametertyp	Datentyp	Beschreibung
		UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	
o1	Ausgang	BOOL	TRUE, wenn i1 > i2.

Beispiel für „Größer als“ (>) in strukturiertem Text

(* ST-Äquivalenz: *)

```
aresult := (10 > 25); (* aresult ist FALSE *)
mresult := ('ab' > 'a'); (* mresult ist TRUE *)
```

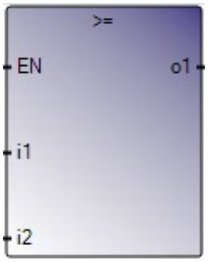
Größer als oder gleich

Vergleicht die Eingabewerte „Integer“, „Real“, „Time“, „Date“ und „String“, um zu ermitteln, ob der erste Wert größer ist als der zweite Wert oder gleich groß.

Bei TON, TP und TOF wird die Gleichheitsprüfung der „Time“-Werte nicht empfohlen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Eingabevergleich ausführen. FALSE – es findet kein Vergleich statt. Gilt nur für Kontaktplan-Programme.

Parameter	Parametertyp	Datentyp	Beschreibung
i1	Eingang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	Alle Eingaben müssen denselben Datentyp haben. Die „Time“-Eingabe gilt für strukturierten Text, Kontaktpläne oder Funktionsblockdiagramme.
i2	Eingang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	
o1	Ausgang	BOOL	TRUE, wenn i1 >= i2.

Beispiel für „Größer als oder gleich“ (>=) in strukturiertem Text

(* ST-Äquivalenz: *)

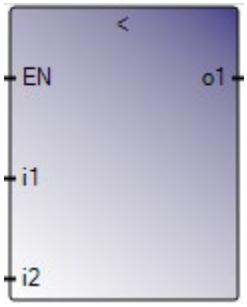
```
aresult := (10 >= 25); (* aresult ist FALSE *)
mresult := ('ab' >= 'ab'); (* mresult ist TRUE *)
```

Kleiner als

Vergleicht die Eingabewerte „Integer“, „Real“, „Time“, „Date“ und „String“, um zu ermitteln, ob der erste Wert kleiner ist als der zweite Wert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Eingabevergleich ausführen. FALSE – es findet kein Vergleich statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	Alle Eingaben müssen denselben Datentyp haben.
i2	Eingang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum)	

Parameter	Parametertyp	Datentyp	Beschreibung
		STRING	
o1	Ausgang	BOOL	TRUE, wenn i1 < i2.

Beispiel für „Kleiner als“ (<) in strukturiertem Text

(* ST-Äquivalenz: *)

```
aresult := (10 < 25); (* aresult ist TRUE *)
mresult := ('z' < 'B'); (* mresult ist FALSE *)
```

(* IL-Äquivalenz: *)

KOP	10
LT	25
ST	aresult
KOP	'z'
LT	'B'
ST	mresult

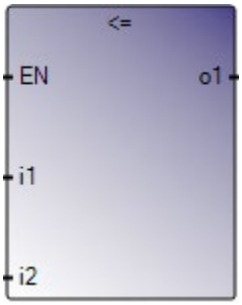
Kleiner als oder gleich

Vergleicht die Eingabewerte „Integer“, „Real“, „Time“, „Date“ und „String“, um zu ermitteln, ob der erste Wert kleiner ist als der zweite Wert oder gleich groß.

Bei TON, TP und TOF wird die Gleichheitsprüfung der „Time“-Werte nicht empfohlen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Eingabevergleich ausführen. FALSE – es findet kein Vergleich statt. Gilt nur für Kontaktplan-Programme.

Parameter	Parametertyp	Datentyp	Beschreibung
i1	Eingang	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	Alle Eingaben müssen denselben Datentyp haben. Die „Time"-Eingabe gilt für strukturierten Text, Kontaktpläne oder Funktionsblockdiagramme.
i2	Eingang	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	
o1	Ausgang	BOOL	TRUE, wenn i1 <= i2.

Beispiel für „Kleiner als oder gleich" (<=) in strukturiertem Text

(* ST-Äquivalenz: *)

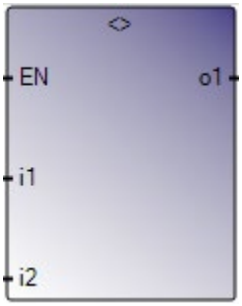
```
aresult := (10 <= 25); (* aresult ist TRUE *)
mresult := ('ab' <= 'ab'); (* mresult ist TRUE *)
```

Nicht gleich

Vergleicht die Eingabewerte vom Typ „Integer", „Real", „Time", „Date" und „String", um zu ermitteln, ob der erste Wert nicht gleich dem zweiten Wert ist.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Ausführung der aktuelle Vergleichsberechnung. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.

Parameter	Parametertyp	Datentyp	Beschreibung
i1	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	Alle Eingaben müssen denselben Datentyp haben.
i2	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	
o1	Ausgang	BOOL	TRUE – wenn erster Wert <> zweiter Wert.

Beispiel für „Nicht gleich“ (<>) in strukturiertem Text

(* ST-Äquivalenz: *)

```
aresult := (10 <> 25); (* aresult ist TRUE *)
mresult := ('ab' <> 'ab'); (* mresult ist FALSE *)
```

Zähleranweisungen

Zähleranweisungen werden zur Steuerung von Vorgängen auf der Grundlage der Ereignisanzahl verwendet.

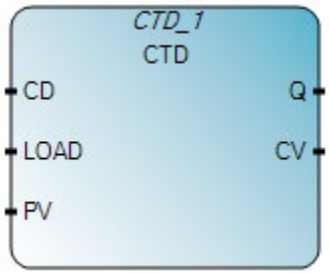
Anweisung	Beschreibung
CTD auf seite 251	Zählt Ganzzahlen abwärts von einem gegebenen Wert bis 0, in Schritten von 1.
CTU auf seite 253	Dieser Block zählt Ganzzahlen aufwärts von 0 bis zu einem gegebenen Wert, in Schritten von 1.
CTUD auf seite 255	Dieser Block zählt Ganzzahlen aufwärts von 0 bis zu einem gegebenen Wert in Schritten von 1, oder abwärts von einem gegebenen Wert bis 0 in Schritten von 1.

CTD (Herunterzählen)

Zählt Ganzzahlen abwärts von einem gegebenen Wert bis 0, in Schritten von 1.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

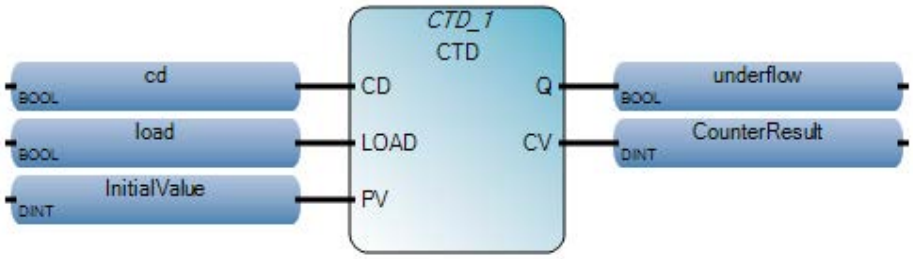


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

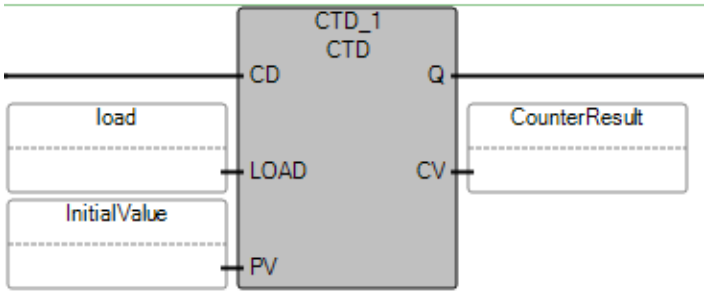
Parameter	Parametertyp	Datentyp	Beschreibung
CD	Eingang	BOOL	Zählt herunter. TRUE – wird eine Anstiegflanke erkannt, wird in Einerschritten heruntergezählt. FALSE - wird eine fallende Flanke erkannt, bleibt der Zählerwert unverändert.
LOAD	Eingang	BOOL	„Laden" verifiziert den PV-Wert gegen den Wert für „Herunterzählen". TRUE - legen Sie CV=PV fest. FALSE - es wird weiter in Einerschritten heruntergezählt.
PV	Eingang	DINT	Programmierter Maximalwert des Zählers.
Q	Ausgang	BOOL	Gibt an, ob die Herunterzählen-Anweisung zu einer Zahl geführt hat, die kleiner oder gleich dem Maximalwert des Zählers ist.

Parameter	Parametertyp	Datentyp	Beschreibung
			TRUE - Zählerergebnis <=0 (Unterlaufzustand). FALSE - Zählerergebnis >0.
CV	Ausgang	DINT	Aktueller Zählerwert.

Diagrammbeispiel für Funktionsblock CTD



Beispiel für CTD-Kontaktplandiagramm



Beispiel für CTD in strukturiertem Text

```
1  InitialValue := 10;
2  CTD_1(cd, load, InitialValue);
```

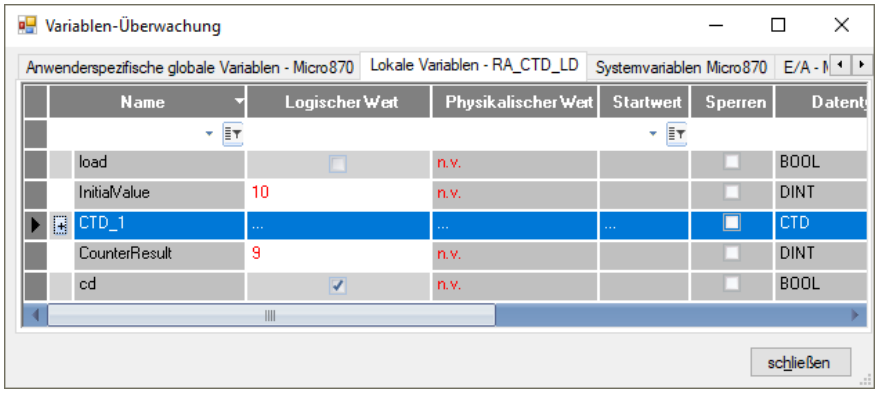
(*ST-Äquivalenz: CTD1 ist eine Instanz des Blocks *)

CTD1(trigger,load_cmd,100);

Unterlauf := CTD1.Q;

result := CTD1.CV;

Ergebnisse

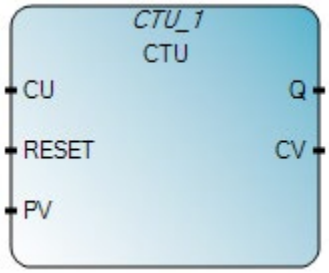


CTU (Hochzählen)

CTU zählt (Ganzzahlen) aufwärts von 0 bis zu einem gegebenen Wert, in Schritten von 1.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

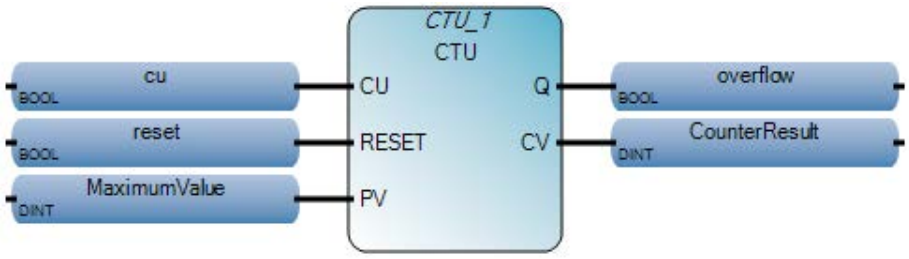
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



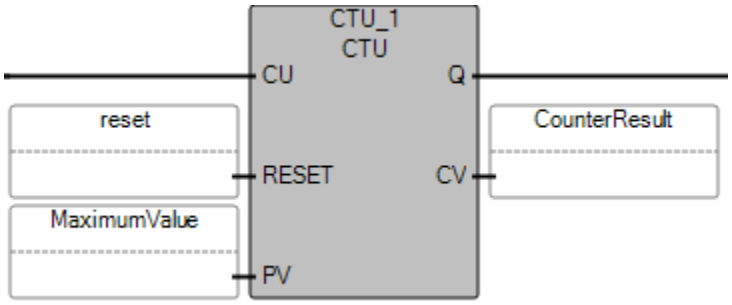
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
CU	Eingang	BOOL	Zählt hoch. TRUE – wird eine Anstiegflanke erkannt, wird in Einerschritten hochgezählt. FALSE - wird eine fallende Flanke erkannt, bleibt der Zählerwert unverändert.
RESET	Eingang	BOOL	„Reset" verifiziert den PV-Wert gegen den Wert für „Hochzählen". TRUE - setzt den CV-Wert auf null. FALSE - es wird weiter in Einerschritten hochgezählt.
PV	Eingang	DINT	Programmierter Maximalwert des Zählers.
Q	Ausgang	BOOL	Gibt an, ob die Hochzählen-Anweisung zu einer Zahl geführt hat, die größer oder gleich dem Maximalwert des Zählers ist. TRUE - Zählerergebnis => PV (Überlaufzustand). FALSE - Zählerergebnis < PV
CV	Ausgang	DINT	Aktuelles Zählerergebnis.

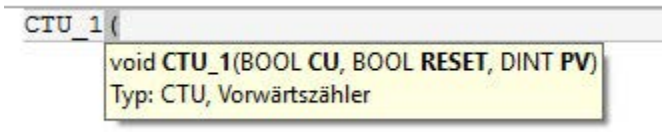
Diagrammbeispiel für Funktionsblock CTU



Beispiel für CTU-Kontaktplandiagramm



Beispiel für CTU in strukturiertem Text

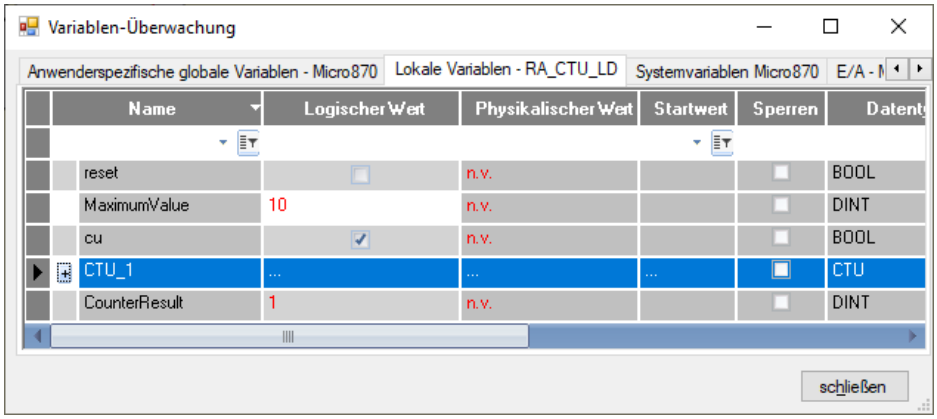



```
1  MaximumValue := 10;
2  CTU_1(cu, reset, MaximumValue);
```

(* ST-Äquivalenz: CTU1 ist eine Instanz des Blocks CTU*)

```
CTU1(trigger,NOT(auto_mode),100);
overflow := CTU1.Q;
result := CTU1.CV;
```

Ergebnisse

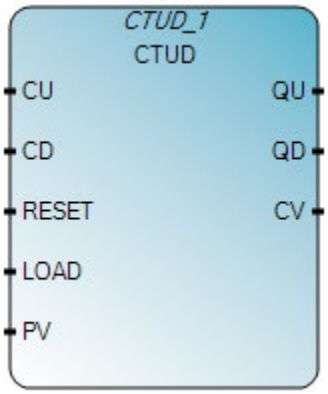


CTUD (Hoch-/Herunterzählen)

Dieser Block zählt Ganzzahlen aufwärts von 0 bis zu einem gegebenen Wert in Schritten von 1, oder abwärts von einem gegebenen Wert bis 0 in Schritten von 1.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

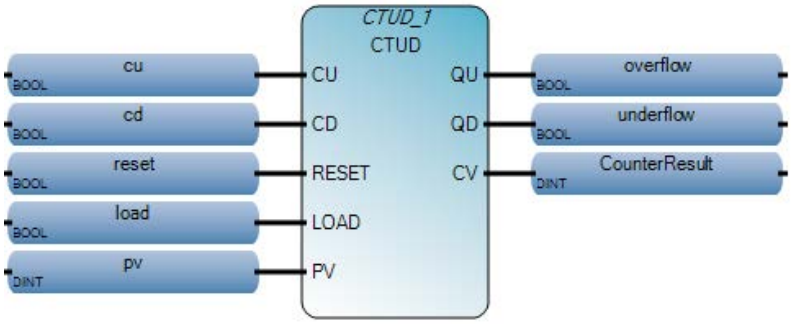
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



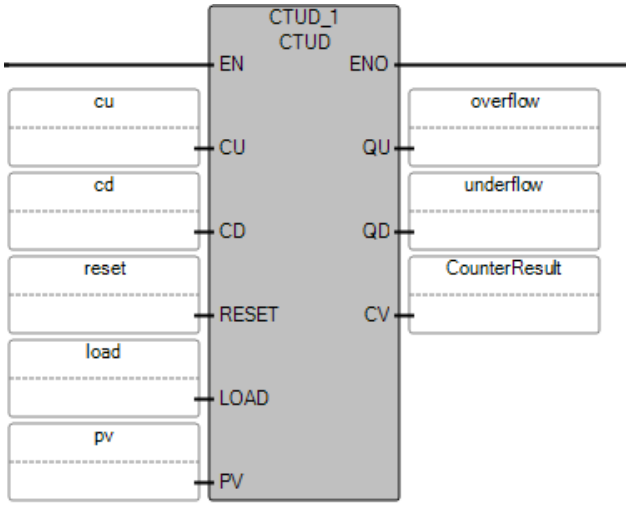
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
CU	Eingang	BOOL	TRUE - wird eine Anstiegflanke erkannt, wird hochgezählt.
CD	Eingang	BOOL	TRUE - wird eine Anstiegflanke erkannt, wird heruntergezählt.
RESET	Eingang	BOOL	Zurücksetzen eines dominanten (höchste Priorität beim Bestimmen eines Anweisungsverhaltens) Befehls. (CV = 0, wenn RESET = TRUE).
LOAD	Eingang	BOOL	Ladebefehl. TRUE - setzt CV = PV.
PV	Eingang	DINT	Programmierter Maximalwert.
QU	Ausgang	BOOL	Überlauf. TRUE - wenn CV >= PV.
QD	Ausgang	BOOL	Unterlauf. TRUE - wenn CV <= 0.
CV	Ausgang	DINT	Zählerergebnis.

Beispiel für ein CTUD-Funktionsblockdiagramm



CTUD-Kontaktplan-Beispiel



Beispiel für CTUD in strukturiertem Text

```
1  cu := TRUE;
2  cd := TRUE;
3  reset := FALSE;
4  load := FALSE;
5  pv := 10;
6  CTUD_1(cu, cd, reset, load, pv);
```

```
CTUD_1(
    void CTUD_1(BOOL CU, BOOL CD, BOOL RESET, BOOL LOAD, DINT PV)
    Typ: CTUD, Vorwärts-/Rückwärtszähler
```

(* ST-Äquivalenz: Wir nehmen an, dass CTUD1 eine Instanz des Blocks ist*)

```
CTUD1(trigger1, trigger2, reset_cmd, load_cmd,100);
full := CTUD1.QU;
empty := CTUD1.QD;
nb_elt := CTUD1.CV;
```

Anweisungen zur Datenumwandlung

Verwenden Sie Anweisungen zur Datenumwandlung, um den Datentyp einer Variable in einen anderen Datentyp umzuwandeln.

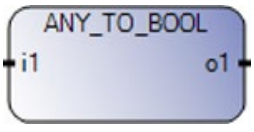
Anweisung	Beschreibung
ANY_TO_BOOL auf seite 257	Konvertiert einen nicht booleschen Wert in einen booleschen Wert.
ANY_TO_BYTE auf seite 258	Konvertiert einen Wert in ein Byte.
ANY_TO_DATE auf seite 259	Konvertiert die Datentypen „String“, „Integer“, „Real“ oder „Time“ in den Datentyp „Date“.
ANY_TO_DINT auf seite 261	Konvertiert einen Wert in einen Wert vom Typ „Double Integer“.
ANY_TO_DWORD auf seite 262	Konvertiert einen Wert in einen Wert vom Typ „Double Word“.
ANY_TO_INT auf seite 263	Konvertiert einen Wert in einen Wert vom Typ „Integer“.
ANY_TO_LINT auf seite 264	Konvertiert einen Wert in einen Long-Integer-Wert.
ANY_TO_LREAL auf seite 265	Konvertiert einen Wert in einen Wert vom Typ „Long Real“.
ANY_TO_LWORD auf seite 266	Konvertiert einen Wert in einen Wert vom Typ „Long Word“.
ANY_TO_REAL auf seite 267	Konvertiert einen Wert in einen Wert vom Typ „Real“.
ANY_TO_SINT auf seite 268	Konvertiert einen Wert in einen Short-Integer-Wert.
ANY_TO_STRING auf seite 269	Konvertiert einen Wert in einen Wert vom Typ „String“.
ANY_TO_TIME auf seite 270	Konvertiert einen Wert in den Datentyp „Time“.
ANY_TO_UDINT auf seite 271	Konvertiert einen Wert in einen Wert vom Typ „Double Integer“ ohne Vorzeichen.
ANY_TO_UINT auf seite 272	Konvertiert einen Wert in einen Wert vom Typ „Integer“ ohne Vorzeichen.
ANY_TO_ULINT auf seite 273	Konvertiert einen Wert in einen Long-Integer-Wert ohne Vorzeichen.
ANY_TO_USINT auf seite 274	Konvertiert einen Wert in einen Short-Integer-Wert ohne Vorzeichen.
ANY_TO_WORD auf seite 275	Konvertiert einen Wert in einen Wert vom Typ „Word“.

ANY_TO_BOOL

Konvertiert einen nicht booleschen Wert in einen booleschen Wert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in einen booleschen Wert ausführen.

Parameter	Parametertyp	Datentyp	Beschreibung
			FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
i1	Eingang	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	Jeder nicht boolesche Wert.
o1	Ausgang	BOOL	Boolescher Wert.

Beispiel für strukturierten ANY_TO_BOOL-Text

(* ST-Äquivalenz: *)

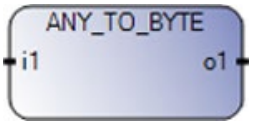
```
ares := ANY_TO_BOOL (10);          (* ares ist TRUE *)
tres := ANY_TO_BOOL (t#0s);        (* tres ist FALSE *)
mres := ANY_TO_BOOL ('FALSE');     (* mres ist FALSE *)
```

ANY_TO_BYTE

Konvertiert einen Wert in einen 8-Bit-Byte-Wert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in die „Byte"-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.

Parameter	Parametertyp	Datentyp	Beschreibung
i1	Eingang	BOOL SINT USINT INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	Jeder Wert, der kein Byte-Wert ist.
o1	Ausgang	BYTE	Ein 8-Bit-Wert des Typs „Byte“.
ENO	Ausgang	BOOL	Ausgabe aktivieren. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_BYTE-Text

(* ST-Äquivalenz: *)

```
bres := ANY_TO_BYTE (true);          (* bres ist 1 *)
tres := ANY_TO_BYTE (t#0s46ms);      (* tres ist 46 *)
mres := ANY_TO_BYTE ('0198');        (* mres ist 198 *)
```

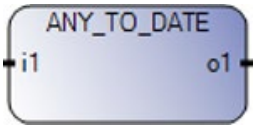
ANY_TO_DATE

Konvertiert die Datentypen „String“, „Integer“, „Real“ oder „Time“ in den Datentyp „Date“.

Die Datentypen Boolean, Byte und Word werden unterstützt, jedoch nicht empfohlen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



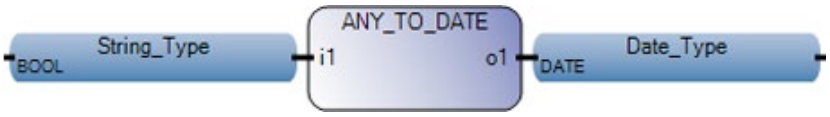
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in die „Date“-Berechnung ausführen.

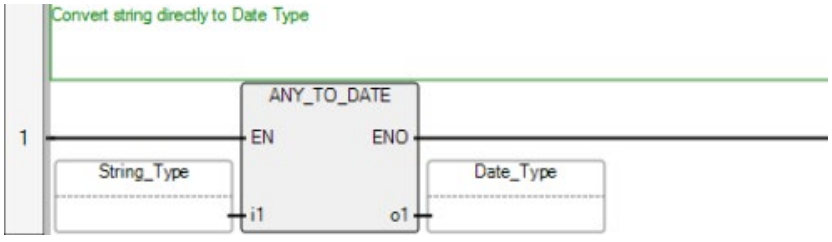
Parameter	Parametertyp	Datentyp	Beschreibung
			FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME STRING	• Strings werden direkt in den Datentyp DATE konvertiert und müssen das Format YYYY-MM-DD, DATE#YYYY-MM-DD oder D#YYYY-MM-DD aufweisen. • Die Datentypen Integer und Real (müssen positiv sein) können in Sekunden angegeben und zum Datum 1970-01-01 hinzugefügt werden. • Time wird zum Datum 1970-01-01 hinzugefügt.
o1	Ausgang	DATE (Datum)	Konvertierter Date-Wert. Fehler während der Konvertierung (z. B. ein negativer Date-Wert) führen zu einem INVALID-Ergebnis.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für ein ANY_TO_DATE-Funktionsblockdiagramm

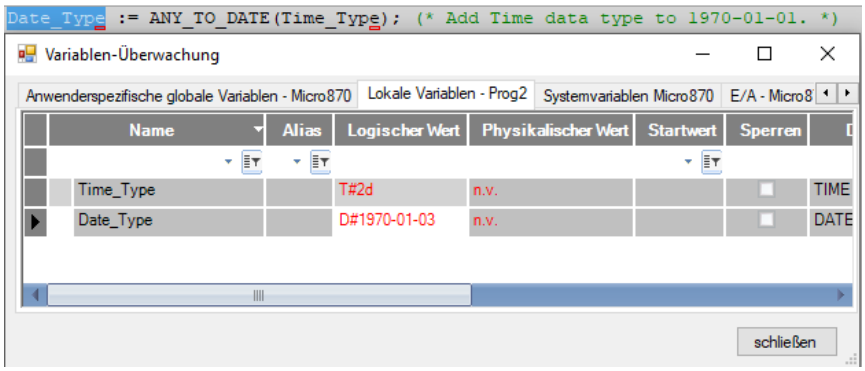
ANY_TO_DATE



Beispiel für einen ANY_TO_DATE-Kontaktplan



Beispiel für strukturierten ANY_TO_DATE-Text

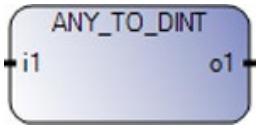


ANY_TO_DINT

Konvertiert einen Wert in einen 32-Bit-Wert vom Typ „Double Integer“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in den 32-Bit-Wert des Typs „Double Integer“ ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE INT UINT WORD UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	Jeder Wert mit Ausnahme von „Double Integer“.
o1	Ausgang	DINT	Ein 32-Bit-Wert vom Typ „Double Integer“.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe.

Parameter	Parametertyp	Datentyp	Beschreibung
			Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_DINT-Text

(* ST-Äquivalenz: *)

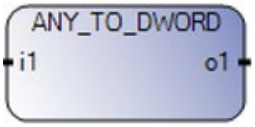
```
bres := ANY_TO_DINT (true);          (* bres ist 1 *)
tres := ANY_TO_DINT (t#1s46ms);      (* tres ist 1046 *)
mres := ANY_TO_DINT ('0198');        (* mres ist 198 *)
```

ANY_TO_DWORD

Konvertiert einen Wert in einen 32-Bit-Wert vom Typ „Double Word“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in den 32-Bit-Wert des Typs „Double Word“ ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT LINT ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	Jeder Wert mit Ausnahme von „Double Word“.
o1	Ausgang	DWORD	Ein 32-Bit-Wert vom Typ „Double Word“.

Parameter	Parametertyp	Datentyp	Beschreibung
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_DWORD-Text

(* ST-Äquivalenz: *)

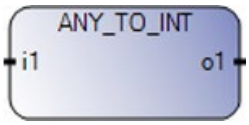
```
bres := ANY_TO_DWORD (true);          (* bres ist 1 *)
tres := ANY_TO_DWORD (t#1s46ms);      (* tres ist 1046 *)
mres := ANY_TO_DWORD ('0198');        (* mres ist 198 *)
```

ANY_TO_INT

Konvertiert einen Wert in einen 16-Bit-Integer-Wert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in den 16-Bit-Wert des Typs „Integer“ ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	Jeder Wert, der kein 16-Bit-Wert vom Typ „Integer“ ist.
o1	Ausgang	INT	Ein 16-Bit-Wert des Typs „Integer“.

Parameter	Parametertyp	Datentyp	Beschreibung
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_INT-Text

(* ST-Äquivalenz: *)

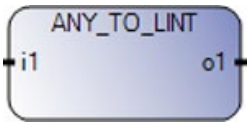
```
bres := ANY_TO_INT (true);          (* bres ist 1 *)
tres := ANY_TO_INT (t#0s46ms);      (* tres ist 46 *)
mres := ANY_TO_INT ('0198');        (* mres ist 198 *)
```

ANY_TO_LINT

Konvertiert einen Wert in einen 64-Bit-Wert vom Typ „Long Integer“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in den 64-Bit-Wert des Typs „Long Integer“ ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	Jeder Wert mit Ausnahme von „Long Integer“.
o1	Ausgang	LINT	Ein 64-Bit-Wert des Typs „Long Integer“.

Parameter	Parametertyp	Datentyp	Beschreibung
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_LINT-Text

(* ST-Äquivalenz: *)

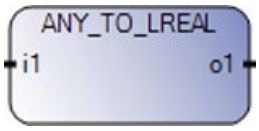
```
bres := ANY_TO_LINT (true);          (* bres ist 1 *)
tres := ANY_TO_LINT (t#0s46ms);      (* tres ist 46 *)
mres := ANY_TO_LINT ('0198');        (* mres ist 198 *)
```

ANY_TO_LREAL

Konvertiert jeden beliebigen Wert in einen Wert vom Typ „Long Real“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in die „Long Real“-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL TIME DATE (Datum) STRING	Jeder Wert mit Ausnahme von „Long Real“.
o1	Ausgang	LREAL	Ein Wert vom Typ „Long Real“.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_LREAL-Text

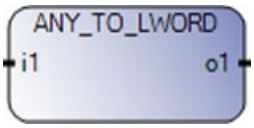
```
(* ST-Äquivalenz: *)

bres := ANY_TO_LREAL (true);          (* bres ist 1,0 *)
tres := ANY_TO_LREAL (t#1s46ms);      (* tres ist 1046,0 *)
ares := ANY_TO_LREAL (198);           (* ares ist 198.0 *)
```

ANY_TO_LWORD

Konvertiert einen Wert in einen 64-Bit-Wert vom Typ „Long Word“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in den 64-Bit-Wert des Typs „Long Word“ ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT REAL LREAL IME DATE (Datum) STRING	Jeder Wert mit Ausnahme von „Long Word“.
o1	Ausgang	LWORD	Ein 64-Bit-Wert des Typs „Long Word“.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_LWORD-Text

(* ST-Äquivalenz: *)

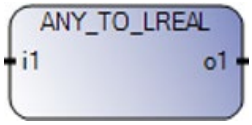
```
bres := ANY_TO_LWORD (true);          (* bres ist 1 *)
tres := ANY_TO_LWORD (t#0s46ms);      (* tres ist 46 *)
mres := ANY_TO_LWORD ('0198');        (* mres ist 198 *)
```

ANY_TO_REAL

Konvertiert einen Wert in einen Wert vom Typ „Real“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in die „Real“-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD LREAL TIME DATE STRING	Jeder Wert mit Ausnahme von „Real“.
o1	Ausgang	REAL	Ein Wert vom Typ „Real“.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_REAL-Text

(* ST-Äquivalenz: *)

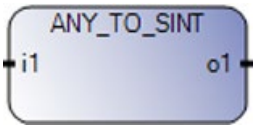
```
bres := ANY_TO_REAL (true);          (* bres ist 1,0 *)
tres := ANY_TO_REAL (t#1s46ms);      (* tres ist 1046,0 *)
ares := ANY_TO_REAL (198);           (* ares ist 198.0 *)
```

ANY_TO_SINT

Konvertiert einen Wert in einen Short-Integer-Wert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in einen 8-Bit-Short-Integer-Wert ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	Jeder Wert, der kein Short-Integer-Wert ist.
o1	Ausgang	SINT	Ein Short-Integer-Wert.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_SINT-Text

(* ST-Äquivalenz: *)

```
bres := ANY_TO_SINT (true);          (* bres ist 1 *)
tres := ANY_TO_SINT (t#0s46ms);      (* tres ist 46 *)
```

```
mres := ANY_TO_SINT ('0198');                    (* mres ist 198 *)
```

ANY_TO_STRING

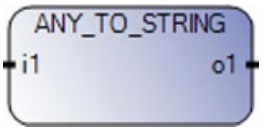
Konvertiert einen Wert in einen Wert vom Typ „String“.

Details zum Betrieb:

- Wenn Sie einen Datentyp REAL in eine Zeichenfolge konvertieren, verwendet die ANY_TO_STRING-Anweisung das IEEE 754-Format.
  - ANY_TO_STRING konvertiert 125.0 zu 1.25000E+02
- Wenn die Ziellänge der Zeichenfolge 5 Zeichen ist:
  - ANY_TO_STRING konvertiert 125.0 zu 1.25000E+02 und gibt 1.250 als Zielzeichenfolge aus.
  - Erstellen Sie ggf. einen benutzerdefinierten Baustein zur Umrechnung von der Exponentialschreibweise in Zahlen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in die „String“-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE (Datum)	Jeder Wert mit Ausnahme von „String“.

Parameter	Parametertyp	Datentyp	Beschreibung

o1	Ausgang	STRING	Wenn IN das Format „Boolean“ hat: „FALSE“ oder „TRUE“.
			Wenn IN das Format „Integer“ oder „Real“ hat: eine dezimale Darstellung.
			Wenn IN das Format „TIME“ hat, ist Folgendes möglich:
			TIME time1
			STRING s1
			time1 :=13 ms;
			s1:=ANY_TO_STRING(time1);
			(* s1 = '0s13' *).

ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_STRING-Text

(* ST-Äquivalenz: *)

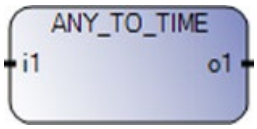
```
bres := ANY_TO_STRING (TRUE);          (* bres ist 'TRUE' *)
ares := ANY_TO_STRING (125);           (* ares ist '125' *)
```

ANY_TO_TIME

Konvertiert einen Nicht-Time- oder Nicht-Date-Wert in einen Wert vom Typ „Time“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in die „Time“-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT	Jeder positive Wert mit Ausnahme des Datentyps „Time“ oder „Date“. IN (bzw. der Vorkommateil, wenn IN das Format „Real“ hat) ist die Anzahl der Millisekunden. STRING (Anzahl der Millisekunden, der Wert 300032 steht beispielsweise für 5 Minuten und 32 Millisekunden).

Parameter	Parametertyp	Datentyp	Beschreibung
		ULINT LWORD REAL LREAL STRING	
o1	Ausgang	TIME	Wert vom Typ „Time“, dargestellt durch IN. Der Wert 1193h2m47s295ms weist auf eine ungültige Uhrzeit hin.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_TIME-Text

(* ST-Äquivalenz: *)

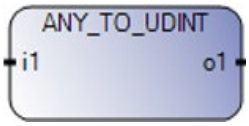
```
ares := ANY_TO_TIME (1256);          (* ares := t#1s256ms *)
rres := ANY_TO_TIME (1256.3);        (* rres := t#1s256ms *)
```

ANY_TO_UDINT

Konvertiert einen Wert in einen 32-Bit-Wert vom Typ „Double Integer“ ohne Vorzeichen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in den 32-Bit-Wert des Typs „Double Integer“ ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT DWORD LINT ULINT LWORD	Jeder Wert mit Ausnahme von „Double Integer“ ohne Vorzeichen.

Parameter	Parametertyp	Datentyp	Beschreibung
		REAL LREAL TIME DATE (Datum) STRING	
o1	Ausgang	UDINT	Ein 32-Bit-Wert vom Typ „Double Integer“ ohne Vorzeichen.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_UDINT-Text

(* ST-Äquivalenz: *)

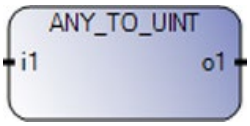
```
bres := ANY_TO_UDINT (true);          (* bres ist 1 *)
tres := ANY_TO_UDINT (t#1s46ms);      (* tres ist 1046 *)
mres := ANY_TO_UDINT ('0198');        (* mres ist 198 *)
```

ANY_TO_UINT

Konvertiert einen Wert in einen Wert vom Typ „Integer“ ohne Vorzeichen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in einen 16-Bit-Wert des Typs „Integer“ ohne Vorzeichen ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE INT WORD DINT UDINT DWORD LINT ULINT	Jeder Wert vom Typ „Integer“ ohne Vorzeichen.

Parameter	Parametertyp	Datentyp	Beschreibung
		LWORD REAL LREAL TIME DATE (Datum) STRING	
o1	Ausgang	UINT	Ein Wert vom Typ „Integer“ ohne Vorzeichen.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_UINT-Text

(* ST-Äquivalenz: *)

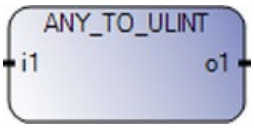
```
bres := ANY_TO_UINT (true);          (* bres ist 1 *)
tres := ANY_TO_UINT (t#0s46ms);      (* tres ist 46 *)
mres := ANY_TO_UINT ('0198');        (* mres ist 198 *)
```

ANY_TO_ULINT

Konvertiert einen Wert in einen 64-Bit-Wert vom Typ „Long Integer“ ohne Vorzeichen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in den 64-Bit-Wert des Typs „Long Integer“ ohne Vorzeichen ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD	Jeder Wert mit Ausnahme von „Long Integer“ ohne Vorzeichen.

Parameter	Parametertyp	Datentyp	Beschreibung
		LINT LWORD REAL LREAL TIME DATE (Datum) STRING	
o1	Ausgang	ULINT	Ein 64-Bit-Wert des Typs „Long Integer" ohne Vorzeichen.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_ULINT-Text

(* ST-Äquivalenz: *)

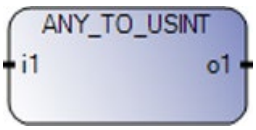
```
bres := ANY_TO_ULINT (true);          (* bres ist 1 *)
tres := ANY_TO_ULINT (t#0s46ms);      (* tres ist 46 *)
mres := ANY_TO_ULINT ('0198');        (* mres ist 198 *)
```

ANY_TO_USINT

Konvertiert einen Wert in einen Short-Integer-Wert ohne Vorzeichen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in den 8-Bit-Wert des Typs „Short Integer" ohne Vorzeichen ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT BYTE INT UINT WORD DINT UDINT DWORD LINT	Jeder Wert, der kein Short-Integer-Wert ist.

Parameter	Parametertyp	Datentyp	Beschreibung
		ULINT LWORD REAL LREAL TIME DATE (Datum) STRING	
o1	Ausgang	USINT	Ein Short-Integer-Wert ohne Vorzeichen.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_USINT-Text

(* ST-Äquivalenz: *)

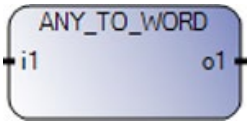
```
bres := ANY_TO_USINT (true);          (* bres ist 1 *)
tres := ANY_TO_USINT (t#0s46ms);      (* tres ist 46 *)
mres := ANY_TO_USINT ('0198');        (* mres ist 198 *)
```

ANY_TO_WORD

Konvertiert einen Wert in einen 16-Bit-Wert vom Typ „Word“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Konvertierung in den 16-Bit-Wert des Typs „Word“ ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
i1	Eingang	BOOL SINT USINT BYTE INT DINT UDINT DWORD LINT ULINT LWORD	Jeder Wert mit Ausnahme von „WORD“.

Parameter	Parametertyp	Datentyp	Beschreibung
		REAL LREAL TIME DATE (Datum) STRING	
o1	Ausgang	WORD	Ein Wert vom Typ „Word“.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für strukturierten ANY_TO_WORD-Text

(* ST-Äquivalenz: *)

```
bres := ANY_TO_WORD (true);          (* bres ist 1 *)
tres := ANY_TO_WORD (t#0s46ms);      (* tres ist 46 *)
mres := ANY_TO_WORD ('0198');        (* mres ist 198 *)
```

Anweisungen zur Datenbearbeitung

Verwenden Sie Anweisungen zur Datenbearbeitung zum Ändern der Ausgabedaten, um den Status zu ändern, ohne dabei das Programm zu ändern.

Anweisung	Beschreibung
AVERAGE auf seite 277	Berechnet einen Ausführungsdurchschnitt einer Zahl von einem bestimmten Wert.
COP auf seite 279	Kopiert Binärdaten des Quellelements in das Zielelement.
MAX auf seite 285	Berechnet das Maximum zweier ganzzahliger Werte.
MIN auf seite 284	Berechnet das Minimum zweier ganzzahliger Werte.

AVERAGE

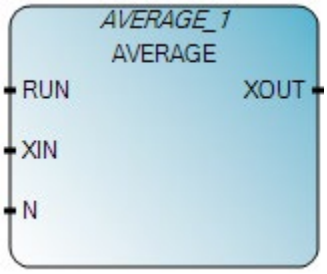
Berechnet einen gleitenden Mittelwert für eine Anzahl definierter Muster und speichert den Wert nach jedem Zyklus.

Details zum Betrieb:

- Die definierte Anzahl an Mustern (N) darf 127 nicht überschreiten.
- Wenn Sie den Wert N einrichten oder ändern, müssen Sie RUN auf FALSE und anschließend wieder auf TRUE setzen.
- Falls der Befehl RUN die Variable FALSE enthält (Zurücksetzungs-Modus), ist der Ausgabewert gleich dem Eingangswert.
- Wenn die maximal mögliche Anzahl gespeicherter Werte erreicht ist, wird der älteste gespeicherte Wert durch den jüngsten überschrieben.
- Die Verwendung von Fließkomma-Datenpunkten kann aufgrund der Rundungsbeschränkungen, die für Fließkomma-Berechnungen gelten, zu fehlerhaften Berechnungen führen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

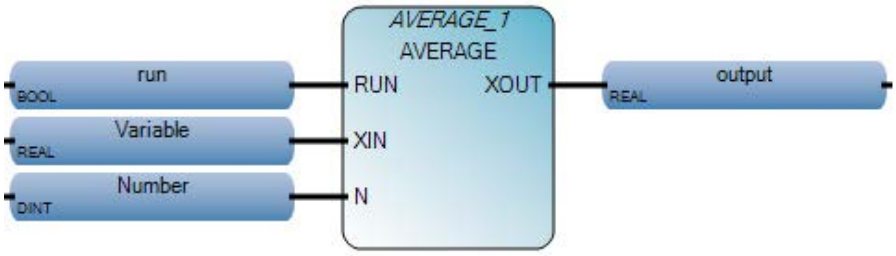
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



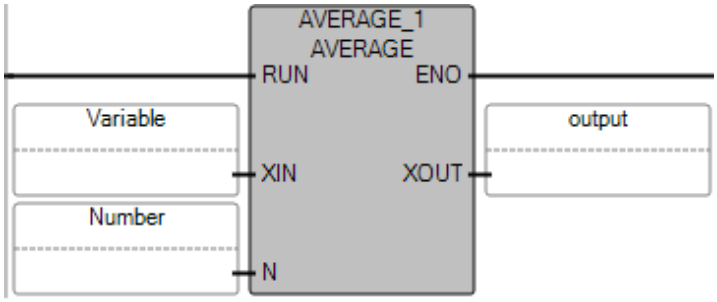
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
RUN	Eingang	BOOL	TRUE = Ausführen FALSE = Zurücksetzen
XIN	Eingang	REAL	Beliebige reelle Variable.
N	Eingang	DINT	Für die Anwendung definierte Anzahl der gespeicherten Werte.
XOUT	Ausgang	REAL	Ausführungsdurchschnitt des Werts XIN.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Diagrammbeispiel für Funktionsblock AVERAGE



AVERAGE-Kontaktplan-Beispiel



Beispiel für AVERAGE in strukturiertem Text

```
1  AVERAGE_1(run, Variable, Number);
2  output := AVERAGE_1.XOUT;
```

(* ST-Äquivalenz: AVERAGE1 eine Instanz eines AVERAGE-Blocks *)

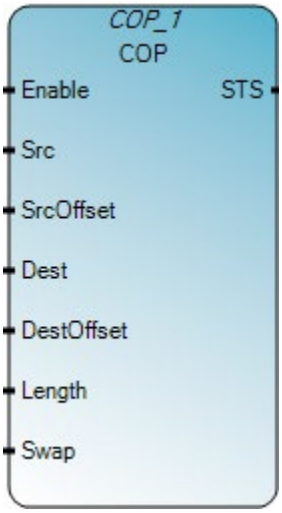
```
AVERAGE1((auto_mode & store_cmd), sensor_value, 100);
ave_value := AVERAGE1.XOUT;
```

COP

Kopiert Binärdaten des Quellelements in das Zielelement. Das Quellelement bleibt unverändert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp		Beschreibung
Enable	Eingang	BOOL		Anweisungsblock aktivieren. COP wird pegelgetriggert. TRUE - Kopiervorgang wird ausgeführt. FALSE – der Funktionsblock befindet sich im Leerlauf.
Src	Eingang	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT	DWORD REAL TIME DATE (Datum) STRING LWORD ULINT LINT LREAL	Anfangselement zum Kopieren. Wenn die Quelle vom Datentyp STRING ist, muss das Ziel auch vom Datentyp STRING oder vom Datentyp USINT (UCHAR und BYTE) sein. Andernfalls wird eine Nichtübereinstimmung des Datentyps gemeldet.
SrcOffset	Eingang	UINT		Der Quellelementversatz wird mit Array-Datentypen verwendet, um die Position im Quell-Array zu ermitteln, von der die Daten kopiert werden sollen. Setzen Sie den Abstand in folgenden Fällen auf 0: • Wenn er nicht vom Datentyp „Array" ist oder • Zum Kopieren vom ersten Element für einen Datentyp „Array".
Dest	Eingang	BOOL SINT USINT BYTE INT UINT	DWORD REAL TIME DATE (Datum) STRING LWORD	Durch die Quelle zu überschreibendes Anfangselement. Wenn das Ziel vom Datentyp STRING ist, muss die Quelle vom Datentyp STRING oder vom Datentyp USINT (UCHAR und BYTE) sein. Andernfalls wird eine Nichtübereinstimmung des Datentyps gemeldet.

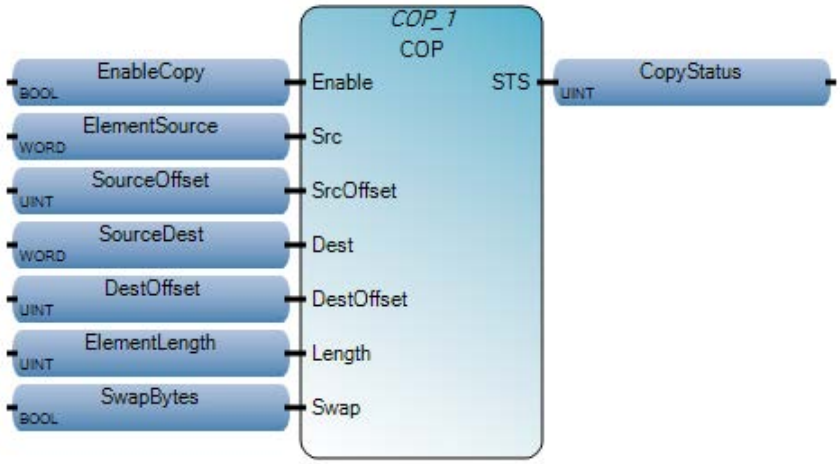
Parameter	Parametertyp	Datentyp		Beschreibung
		WORD DINT UDINT	ULINT LINT	
DestOffset	Eingang	UINT		Der Zielelementversatz wird mit Array-Datentypen verwendet, um die Position im Ziel-Array zu ermitteln, an die die Daten kopiert werden sollen. Setzen Sie den Abstand in folgenden Fällen auf 0: • Wenn er nicht vom Datentyp „Array" ist oder • Zum Kopieren vom ersten Element für einen Datentyp „Array".
Length	Eingang	UINT		Anzahl der zu kopierenden Zielelemente. Wenn das Ziel vom Datentyp STRING ist, wird die Anzahl der zu kopierenden Zeichenfolgen angezeigt.
Swap	Eingang	BOOL		Wird verwendet, um die Daten aus den Quell- und Zielelementen auszutauschen, sodass die Zieldaten die Quelldaten ersetzen und die Quelldaten die Zieldaten ersetzen. TRUE - Bytes werden gemäß Datentyp ausgetauscht. In folgenden Fällen wird nicht ausgetauscht: • Der Quell- oder Zieldatentyp ist STRING, oder • Wenn die Länge der Quell- und der Zieldaten jeweils 1 Byte beträgt.
Sts	Ausgang	UINT		Status des Kopiervorgangs. Die Definitionen für den Parameter „Sts" sind in COP-Statuscodes definiert.
ENO	Ausgang	BOOL		Ausgang wird aktiviert. Gilt nur für Kontaktplan-Programme.

COP-Statuscodes (Sts)

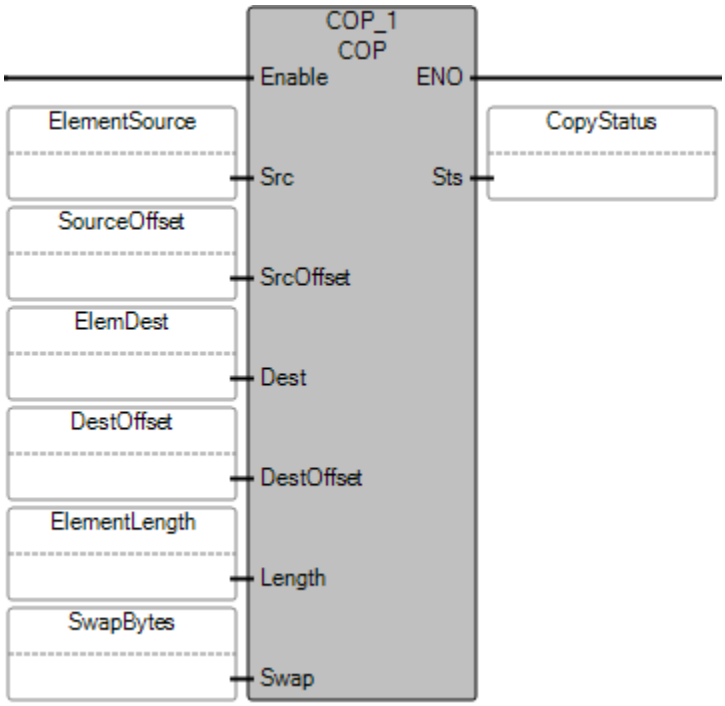
In der folgenden Tabelle werden die COP-Statuscodes beschrieben.

COP-Statuscode	Statusbeschreibung
0x00	Keine Aktion ergriffen (nicht aktiviert).
0x01	COP-Funktionsblock erfolgreich.
0x02	Der Zielort verfügt beim Kopieren aus „String" über Ersatzbytes.
0x03	Quellortdaten sind abgeschnitten.
0x04	Länge der Kopie ist ungültig.
0x05	Beim Datentyp „String" als entweder Quell- oder Zielort gibt es eine Nichtübereinstimmung des Datentyps.
0x06	Die Quellortdatengröße ist für die Kopie zu klein.
0x07	Die Zielortdatengröße ist für die Kopie zu klein.
0x08	Versatz der „Quellortdaten" ist ungültig.
0x09	Versatz der „Zielortdaten" ist ungültig.
0x0A	Die Daten sind entweder am Quell- oder am Zielort ungültig.

Diagrammbeispiel für Funktionsblock COP



COP-Kontaktplan Beispiel



Beispiel für COP in strukturiertem Text

```
1  COP_1(EnableCopy, ElementSource, SourceOffset, SourceDest,
2    DestOffset, ElementLength, SwapBytes);
3  output :=COP_1.STS;
```

Kopieren zu einem anderen Datentyp

Beim Kopieren in oder aus dem Datentyp „String“ wird das kurze ODVA-„String“-Format für die Daten im USINT-Array verwendet.

Wenn COP auf seite 279 zwischen einem anderen Paar von Datentypen

verwendet wird, ist der Kopiervorgang gültig, auch wenn der Datentyp in der Quelle nicht mit dem Datentyp im Ziel übereinstimmt und selbst wenn sie kein gültiges Format aufweisen. Die Logik muss auf Anwendungsebene validiert werden.

Von einem USINT-Array zu einem String-Array

Um einen USINT-Array zu einem String-Array zu kopieren, müssen die Daten im USINT-Array im folgenden Format sein:

- Byte1: Länge des ersten Strings
- Byte2: Erstes Byte-Zeichen
- Byte3: Zweites Byte-Zeichen
- Byte n: Letztes Byte-Zeichen
- Byte (n+1): Länge des zweiten Strings
- Byte (n+2): Erstes Byte-Zeichen für den zweiten String

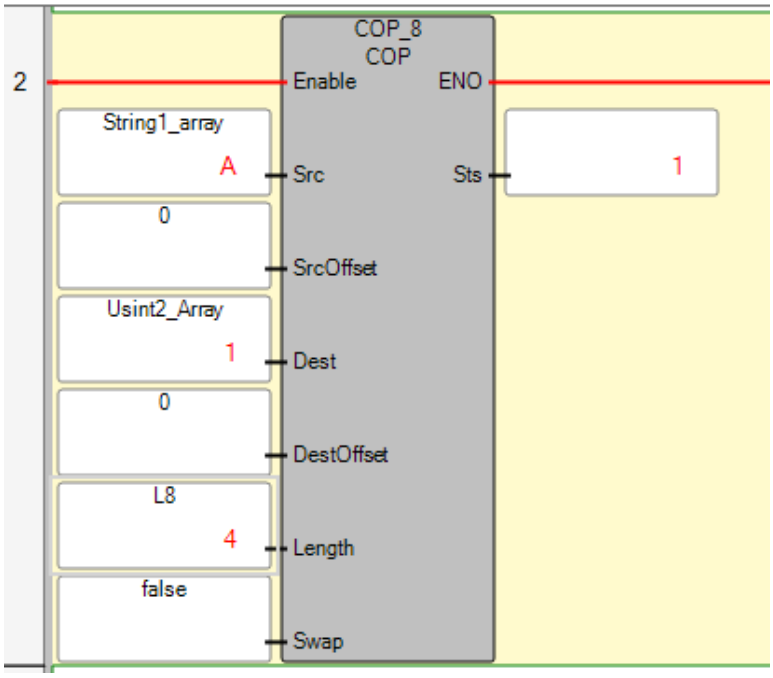
Beispiel für ein COP-String-Array

Das folgende Beispiel zeigt einen COP auf seite 279-Befehl, der ein Zeichenfolgen- Array in ein Usint-Array kopiert. Die COP-Anweisung überspringt alle Elemente im Quell-Array mit Länge Null.

Für dieses Beispiel gilt:

- Die Länge wird als 4 Bytes angegeben.
- Die Anzahl der zu kopierenden Zielelemente beträgt 4 Bytes.
- Alle Array-Elemente mit Länge Null (leer) werden übersprungen.
- Die COP-Anweisung findet ein Element mit Nicht-Null-Länge im String-Array [1], das zu einem Ziel-USINT-Array [1] mit 1 kopiert wird. Eins ist die Länge des Strings im String-Array [1], und Usint-Array [2] ist 65, der ASCII-Code für "A".
- Die COP-Anweisung findet ein Element mit Nicht-Null-Länge im String-Array [10], nämlich "a", das zu einem Ziel-USINT-Array [3] mit 1 kopiert wird. Eins ist die Länge des Strings im String-Array [10], und Usint-Array [4] ist 97, der ASCII-Code für "a".

Kontaktplan



Array-Elemente mit logischen Werten

Name	Alias	Logischer Wert
String_String_1.String1_array		...
String_String_1.String1_array[1]		A
String_String_1.String1_array[2]		
String_String_1.String1_array[3]		
String_String_1.String1_array[4]		
String_String_1.String1_array[5]		
String_String_1.String1_array[6]		
String_String_1.String1_array[7]		
String_String_1.String1_array[8]		
String_String_1.String1_array[9]		
String_String_1.String1_array[10]		a
String_String_1.USint2_Array		...
String_String_1.USint2_Array[1]		1
String_String_1.USint2_Array[2]		65
String_String_1.USint2_Array[3]		1
String_String_1.USint2_Array[4]		97
String_String_1.USint2_Array[5]		0
String_String_1.USint2_Array[6]		0
String_String_1.USint2_Array[7]		0
String_String_1.USint2_Array[8]		0
String_String_1.USint2_Array[9]		0
String_String_1.USint2_Array[10]		0
String_String_1.USint2_Array[11]		0
String_String_1.USint2_Array[12]		0
String_String_1.USint2_Array[13]		0
String_String_1.L8		4
String_String_1.COP_8		...

MIN (Minimum)

Berechnet das Minimum zweier ganzzahliger Werte.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

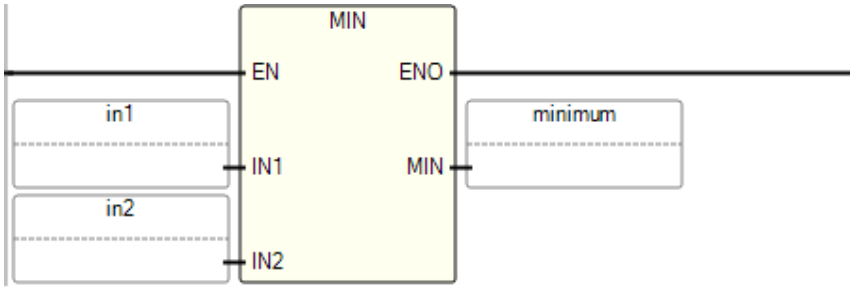
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Berechnung des Minimums ganzzahliger Werte ausführen. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN1	Eingang	DINT	Beliebiger ganzzahliger Wert mit Vorzeichen.
IN2	Eingang	DINT	Kann nicht real sein.
MIN	Ausgang	DINT	Minimum der zwei eingegebenen Werte.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Diagrammbeispiel für Funktionsblock MIN

Beispiel für MIN-Kontaktplandiagramm



Beispiel für strukturierten MIN-Text

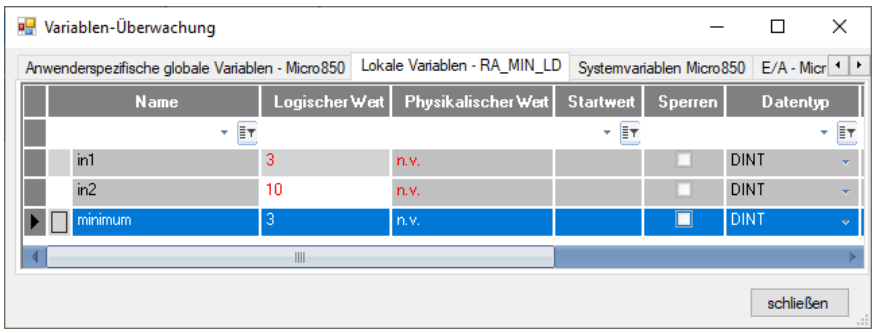
```
in1 := 3;
in2 := 10;
minimum := MIN(in1, in2);
```

(* ST-Äquivalenz: *)

new_value := MAX (MIN (max_value, value), min_value);

(* beschränkt den Wert auf den gegebenen Bereich [min_value..max_value] *)

Ergebnisse

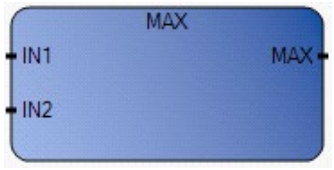


MAX (Maximum)

Berechnet das Maximum zweier ganzzahliger Werte.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



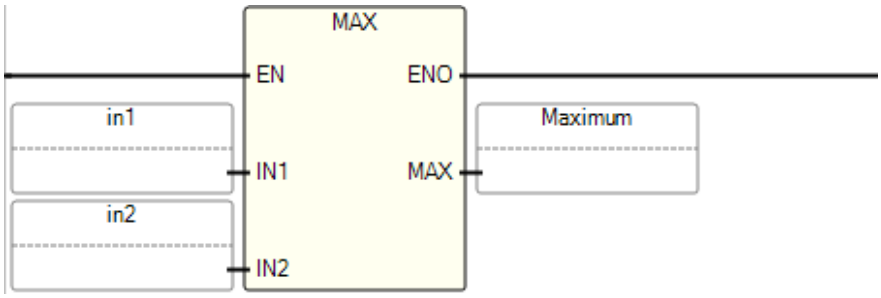
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Berechnung des Maximums ganzzahliger Werte ausführen. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
IN1	Eingang	DINT	Beliebiger ganzzahliger Wert mit Vorzeichen.

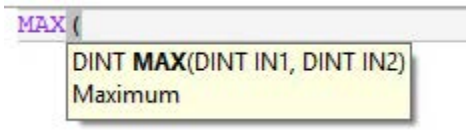
Parameter	Parametertyp	Datentyp	Beschreibung
IN2	Eingang	DINT	Kann nicht real sein.
MAX	Ausgang	DINT	Maximum der zwei eingegebenen Werte.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für MAX-Funktionsblockdiagramm

Beispiel für MAX-Kontaktplandiagramm



Beispiel für strukturierten MAX-Text



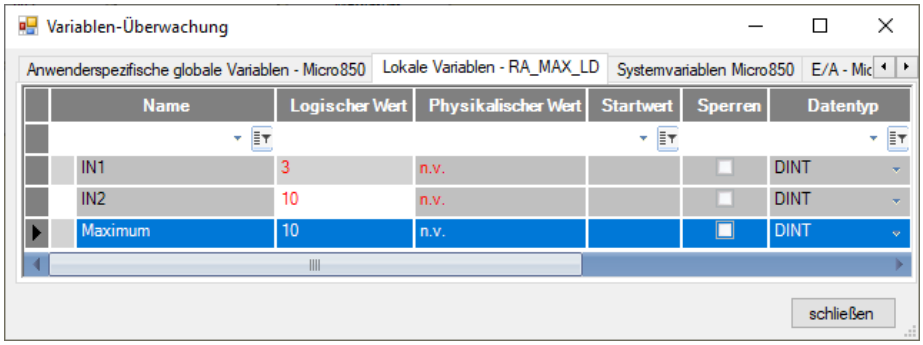

```
1  IN1 := 3;
2  IN2 := 10;
3  Maximum := MAX(IN1, IN2);
```

(* ST-Äquivalenz: *)

new_value := MAX (MIN (max_value, value), min_value);

(* beschränkt den Wert auf den gegebenen Bereich [min_value..max_value] *)

Ergebnisse


Variablen-Überwachung
Anwenderspezifische globale Variablen - Micro850
Lokale Variablen - RA_MAX_LD
Systemvariablen Micro850
E/A - Mic
Name
Logischer Wert
Physikalischer Wert
Startwert
Sperren
Datentyp
IN1
3
n.v.
DINT
IN2
10
n.v.
DINT
Maximum
10
n.v.
DINT
schließen

Hochgeschwindigkeitszähler-Anweisungen (HSC)

Verwenden Sie Hochgeschwindigkeitszähler-Anweisungen, um den Hochgeschwindigkeitszähler zu überwachen und zu steuern.

Anweisung	Beschreibung
HSC auf seite 290	HSC wendet hohe und niedrige Voreinstellungen und Ausgabequellenwerte auf den Hochgeschwindigkeitszähler an.
HSC_SET_STS auf seite 306	HSC_SET_STS legt den HSC-Zählstatus manuell fest oder setzt ihn zurück.

Was ist ein Hochgeschwindigkeitszähler?

Ein Hochgeschwindigkeitszähler erkennt und zählt enge (schnelle) Impulse und gibt spezielle Anweisungen aus, um Steuervorgänge einzuleiten, wenn die ermittelten Zahlen die voreingestellten Werte erreicht haben. Steuervorgänge umfassen die automatische und sofortige Ausführung der Hochgeschwindigkeitszähler-Unterbrechungsroutine und die sofortige Aktualisierung der Ausgabe basierend auf der konfigurierten Quelle und dem Maskenmuster.

Funktionen des Hochgeschwindigkeitszählers

Da die HSC-Anweisungen Hochleistungsanforderungen erfordern, wird ihr Vorgang von kundenspezifischen Schaltungen durchgeführt, die parallel zum Hauptsystemprozessor geschaltet sind. Die verbesserten Funktionen des Hochgeschwindigkeitszählers (HSC) umfassen:

- 100 kHz Hochleistungsbetrieb mit direkter Steuerung der Ausgänge
- 32-Bit-Ganzzahldaten mit Vorzeichen (Zählbereich ± 2.147.483.647)
- Programmierbare hohe und niedrige Voreinstellungen
- Überlauf- und Unterlauf-Sollwert
- Automatische Unterbrechungsverarbeitung basierend auf der kumulierten Zählung
- Bearbeitbare Laufzeit-Parameter (vom Benutzersteuerprogramm); HSC-Anweisungsbetrieb

Unterstützung des Micro800-Controllers für HSC

Alle Micro830-, Micro850- und Micro870-Controller, mit Ausnahme des 2080-LCxx-AWB, unterstützen bis zu sechs HSC-Eingänge. Die

HSC-Funktionalität ist in Micro800-Controller integriert, sie setzt Hochgeschwindigkeitszähler-Hardware (integrierte Ausgaben im Controller) und die HSC-Anweisung in der Anwendung ein. Der HSC-Befehl auf seite 289 konfiguriert die Hardware des Hochgeschwindigkeitszählers und aktualisiert den Bildakkumulator.

WICHTIG Die HSC-Funktion kann nur mit der integrierten E/A des Controllers eingesetzt werden. Sie kann nicht mit E/A-Erweiterungsmodulen verwendet werden.

HSC (Hochgeschwindigkeitszähler)

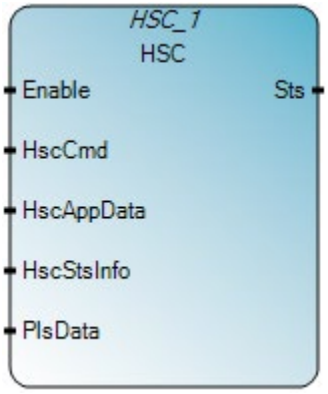
HSC wendet hohe und niedrige Voreinstellungen und Ausgabequellenwerte auf den Hochgeschwindigkeitszähler an.

Details zum Betrieb:

- Der programmierbare Endschalter (PLS) wird aktiviert, indem Sie den Parameter „HSCAppData.PLSEnable“ auf „True“ setzen.
- Der Parameter „PLSPosition“ wird nach Abschluss des gesamten Zyklus und Erreichen des HSCSTS.HP-Werts zurückgesetzt.

Diese Anweisung gilt für Micro830-, Micro850- und Micro870-Controller.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

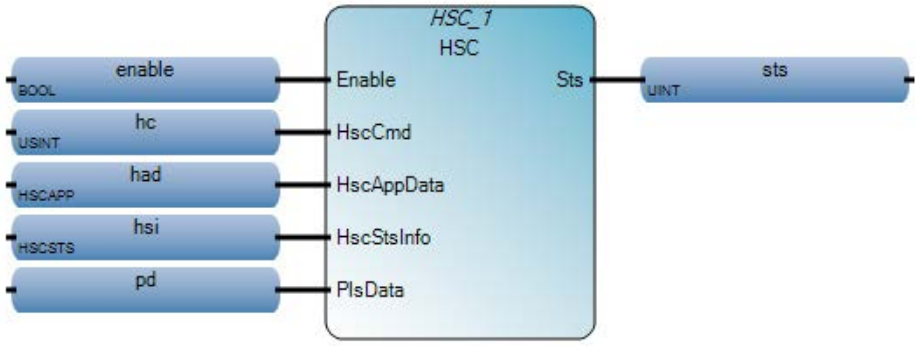


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

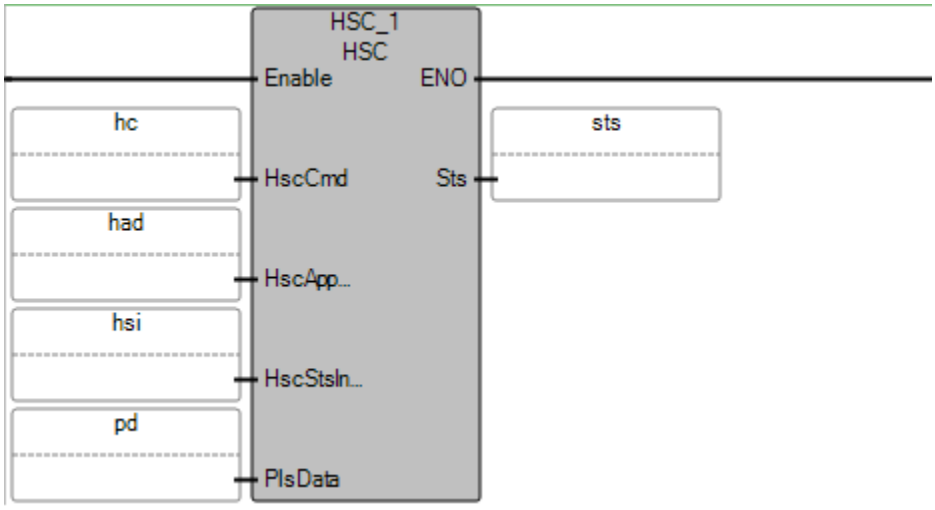
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Strompfad für Anweisungsblock. TRUE – Timer startet Erhöhung. FALSE – der Funktionsblock befindet sich im Leerlauf. Wir raten davon ab, den EN-Parameter mit dem HSC-Funktionsblock zu verwenden, da der Zeitgeber die Erhöhungsschritte fortsetzt, wenn EN auf FALSE gesetzt wird. Gilt nur für Kontaktplan-Programme.
Enable	Eingang	BOOL	Aktivieren Sie den Anweisungsblock. TRUE – der im HSC-Befehlsparameter festgelegte HSC-Vorgang wird ausgeführt. FALSE – es werden keine HSC-Befehle ausgegeben.
HscCmd	Eingang	USINT	Gibt Befehle an HSC.
HSCAppData	Eingang	HSCAPP auf seite 292	Konfiguration der HSC-Anwendung, die üblicherweise nur einmal benötigt wird.

Parameter	Parametertyp	Datentyp	Beschreibung
HSCStsInfo	Eingang	HSCSTS auf seite 296	HSC-Dynamikstatus, der während des HSC-Zählvorgangs kontinuierlich aktualisiert wird.
PlsData	Eingang	DINT UDINT	Datenstruktur des programmierbaren Endschalters (Programmable Limit Switch, PLS).
Sts	Ausgang	UINT	HSC-Ausführungsstatus. HSC-Statuscodes: • 0x00 – Keine Maßnahme ergriffen (nicht aktiviert). • 0x01 – HSC-Ausführung erfolgreich. • 0x02 – HSC-Befehl ungültig. • 0x03 – HSC-ID außerhalb des Gültigkeitsbereichs. • 0x04 – HSC-Konfigurationsfehler.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

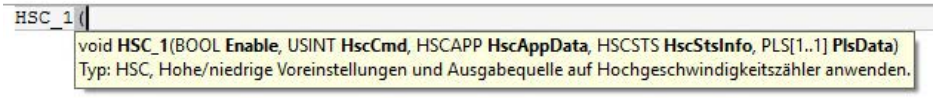
Diagrammbeispiel für Funktionsblock HSC



HSC-Kontaktplan-Beispiel



Beispiel für HSC in strukturiertem Text




```
1  HSC_1(enable, hc, had, hsi, pd);
2  sts := HSC_1.Sts;
```

HSCCmd-Werte

In der folgenden Tabelle werden die HSC-Befehle für jeden HSC-Befehlswert beschrieben.

HSC-Befehl	Beschreibung des Befehls
0x01	HSC RUN • HSC starten (wenn sich der HSC im Leerlauf-Modus befindet und der Kontaktplan aktiviert ist). • Nur HSC-Statusinformationen aktualisieren (wenn sich der HSC im Betriebsmodus befindet und der Kontaktplan aktiviert ist).
0x02	HSC Stop: Zählvorgang eines HSC stoppen (wenn sich der HSC im Betriebsmodus befindet und der Strompfad aktiviert ist).
0x03	HSC Load/Set: HSC-Anwendungsdaten für 6 Elemente neu laden (wenn der Kontaktplan aktiviert ist): HPSetting, LPSetting, HPOutput, LPOutput, OFSetting und UFSetting. **Hinweis:** Durch diesen Befehl wird das folgende Eingabeelement nicht erneut geladen: HSC-Akkumulator.
0x04	Zurücksetzen des HSC-Akkumulators (wenn der Strompfad aktiviert ist).

HSC-Befehlsergebnisse

Befehlswert	Ergebnis	Zustände
HscCmd =1	Startet den HSC-Mechanismus und der HSC wechselt in den Betriebsmodus.	Im Betriebsmodus bewirkt ein Setzen des Eingangsparameters „Enable" auf „False" kein Beenden des Zählvorgangs. HscCmd =2 muss ausgegeben werden, um das Zählen zu stoppen.
	Der HSC-Mechanismus aktualisiert automatisch die Werte.	Der „HSC AppData.Accumalator" wird durch den „HSC Sts.Accumulator" aktualisiert
HscCmd =4 (Reset)	Legt für den Wert „HSC Acc" den Wert „HSC AppData.Accumalator" fest.	Der Befehl „HscCmd =4" führt zu keinem Stoppen der HSC-Zählung. Wenn der HSC bei Auslösung des Befehls „HscCmd =4" zählt, können einige Zählwerte verlorengehen. Um HSC Acc beim Zählen einen bestimmten Wert zuzuweisen, schreiben Sie den Wert unmittelbar vor der Auslösung des Befehls „HscCmd =4" in HSC AppData.Accumalator.

Datentyp HSCAPP

Verwenden Sie den Datentyp HSCAPP, um den HSCAppData-Parameter in der HSC-Anweisung zu definieren. Die Parameter des Datentyps HSCApp werden zur Definition von HSC-Konfigurationsdaten eingesetzt.

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp HSCAPP zu bestimmen.

Parameter	Datentyp	Datenformat	Zugriff auf Benutzerprogramm	Beschreibung
PLSEnable	BOOL	Bit	Read/Write	Aktivieren bzw. Deaktivieren des programmierbaren Endschalters (PLS) für Hochgeschwindigkeitszähler.
HSCID	UINT	Wort	Read/Write	Definiert den HSC.
HSCMode	UINT	Wort	Read/Write	Definiert den HSC-Modus.

Parameter	Datentyp	Datenformat	Zugriff auf Benutzerprogramm	Beschreibung
Accumulator	DINT	Long Word	Read/Write	Akkumulator-Anfangswert. HSCApp.Accumulator legt den Anfangswert des Akkumulators fest, wenn der Hochgeschwindigkeitszähler startet. Wenn sich der HSC im Zählmodus befindet, wird der Akkumulator automatisch vom HSC-Untersystem aktualisiert, um den tatsächlichen HSC-Akkumulatorwert auszugeben.
HPSetting	DINT	Long Word	Read/Write	Hoher Voreinstellungswert. Der Parameter HSCApp.HPSetting legt den oberen Sollwert fest (bei Zählungen), der definiert, wann ein HSC-Untersystem eine Unterbrechung generiert. Die in die hohe Voreinstellung geladenen Daten müssen kleiner oder gleich den Daten im Überlauf-Parameter (HSCAPP.OFSetting) sein, ansonsten wird ein HSC-Fehler ausgegeben.
LPSetting	DINT	Long Word	Read/Write	Niedriger Voreinstellungswert. HSCApp.LPSetting legt den unteren Sollwert fest (bei Zählungen), der definiert, wann ein HSC-Untersystem eine Unterbrechung generiert. Die in die niedrige Voreinstellung geladenen Daten müssen größer oder gleich den Daten im Unterlauf-Parameter (HSCAPP.UFSetting) sein, ansonsten wird ein HSC-Fehler ausgegeben. Wenn der Unterlauf und die niedrigen voreingestellten Werte negative Zahlen sind, muss die niedrige Voreinstellung eine Zahl mit einem absoluten Wert sein, der kleiner ist als der Unterlauf.
OFSetting	DINT	Long Word	Read/Write	Überlauf-Einstellung. Die Überlauf-Einstellung HSCApp.OFSetting definiert die obere Zählgrenze des Zählers. Wenn der kumulierte Wert des Zählers über den in der OFSetting angegebenen Wert steigt, wird eine Überlaufunterbrechung generiert. Nachdem die Überlaufunterbrechung generiert wurde, setzt das HSC-Untersystem den kumulierten Wert auf den Unterlaufwert zurück und der Zähler fährt mit der Zählung beim Unterlaufwert (Zahlen gehen bei dem Wechsel nicht verloren). OFSetting-Werte müssen folgendermaßen aussehen: • Zwischen -2.147.483.648 und 2.147.483.647. • Größer als der Unterlaufwert. • Größer als oder gleich den Daten in der hohen Voreinstellung (HSCAPP.HPSetting), ansonsten wird ein HSC-Fehler ausgegeben.
UFSetting	DINT	Long Word	Read/Write	Unterlauf-Einstellung. Die Unterlauf-Einstellung HSCApp.UFSetting definiert die untere Zählgrenze des Zählers. Wenn der kumulierte Wert des Zählers unter den in der UFSetting angegebenen Wert fällt, wird eine Unterlaufunterbrechung generiert. Nachdem die Unterlaufunterbrechung generiert wurde, setzt das HSC-Untersystem den kumulierten Wert auf den Unterlaufwert zurück und der Zähler beginnt die Zählung beim Überlaufwert (Zahlen gehen bei dem Wechsel nicht verloren). UFSetting-Werte müssen folgendermaßen aussehen: • Zwischen -2.147.483.648 und 2.147.483.647. • Niedriger als der Überlaufwert. • Kleiner als oder gleich den Daten in der niedrigen Voreinstellung (HSCAPP.LPSetting), ansonsten wird ein HSC-Fehler ausgegeben.

Parameter	Datentyp	Datenformat	Zugriff auf Benutzerprogramm	Beschreibung
OutputMask	UDINT	Wort	Read/Write	Ausgabemaske für die Ausgabe. HSCApp.OutputMask definiert die integrierten Ausgaben am Controller, die den Hochgeschwindigkeitszähler direkt steuern. Das HSC-Untersystem kann die Ausgaben ohne Steuerprogramm-Interaktion basierend auf den hohen und niedrigen Voreinstellungen des HSC-Akkumulators EIN oder AUS schalten. Das in der HSCApp.OutputMask gespeicherte Bitmuster definiert, welche Ausgaben von dem HSC gesteuert werden und welche nicht vom HSC gesteuert werden. Das Bit-Muster HSCAPP.OutputMask entspricht den Ausgabe-Bits am Controller und kann nur während der Einrichtung konfiguriert werden. Bits, für die (1) festgelegt wurde, können vom HSC-Untersystem aktiviert und an- oder ausgeschaltet werden. Bits, für die (0) festgelegt wurde, können nicht vom HSC-Untersystem an- oder ausgeschaltet werden. Um den HSC dafür einzusetzen, die Ausgaben 0, 1, 3 zu steuern, müssen folgende Werte zugewiesen werden: • HscAppData.OutputMask = 2#1011 oder • HscAppData.OutputMask = 11
HPOutput	UDINT	Long Word	Read/Write	32-Bit-Ausgabeeinstellung für das Erreichen des hohen Voreinstellungswerts. HSCApp.HPOutput definiert den Zustand (1 = EIN oder 0 = AUS) der Ausgaben am Controller, wenn die obere Voreinstellung erreicht wird. Weitere Informationen zum direkten Ein- oder Ausschalten von Ausgaben basierend auf der hohen Voreinstellung. Das hohe Ausgabe-Bitmuster können Sie während der Ersteinrichtung konfigurieren, oder Sie verwenden den HSC-Funktionsblock, um die neuen Parameter zu laden, während der Controller in Betrieb ist.
LPOutput	UDINT	Long Word	Read/Write	32-Bit-Ausgabeeinstellung für das Erreichen des niedrigen Voreinstellungswerts. HSCApp.LPOutput definiert den Zustand (1 = EIN, 0 = AUS) der Ausgaben am Controller, wenn die untere Voreinstellung erreicht wurde. Weitere Informationen zum direkten Ein- oder Ausschalten von Ausgaben basierend auf der unteren Voreinstellung. Das niedrige Ausgabe-Bitmuster können Sie während der Ersteinrichtung konfigurieren, oder Sie verwenden den HSC-Funktionsblock, um die neuen Parameter zu laden, während der Controller in Betrieb ist.

HSCApp-Einstellungen im Vergleich zu PLSData-Einstellungen

Bei aktivierter PLS-Funktion werden relevante HSCApp-Einstellungen gemäß der nachstehenden Tabelle durch die entsprechenden PLSData-Einstellungen abgelöst.

HSCApp-Einstellung	PLSData-Einstellung
HSCAPP.HpSetting	HSCHP
HSCAPP.LpSetting	HSCLP
HSCAPP.HPOutput	HSCHPOutput
HSCAPP.LPOutput	HSCLPOutput

HSCApp.HSCID

Der Parameter HSCApp.HSCID identifiziert den Hochgeschwindigkeitszähler.

In der folgenden Tabelle werden die Werte für HSCID beschrieben:

Auswahl der Ausgabe	Bit	Beschreibung
Erstes Wort der HSC-Funktionsdaten	15-13	Modultyp des HSC: • 0x00 - Integriert. • 0x01 - Erweiterung. • 0x02 - Einsteck-Anschluss.
	12-8	Schacht-ID des Moduls: • 0x00 - Integriert. • 0x01-0x1F - ID des Erweiterungsmoduls. • 0x01-0x05 - ID des Einsteck-Anschlusses.
	7-0	HSC-ID innerhalb des Moduls: • 0x00-0x0F - Integriert. • 0x00-0x07 - ID des HSC zur Erweiterung. • 0x00-0x07 - ID des HSC für den Einsteck-Anschluss. Für die erste Version von Connected Components Workbench werden nur die IDs 0x00-0x05 unterstützt.

HSCApp.HSCMode

Der Parameter HSCApp.HSCMode weist den Hochgeschwindigkeitszähler einem der zehn Zählungsmodi zu. Der Moduswert wird durch die Programmierung des Geräts konfiguriert und ist über das Steuerprogramm zugänglich.

Weitere Informationen zu HSC-Betriebsmodi und Eingangszuweisungen finden Sie im Abschnitt zu HSC-Eingängen und Verdrahtungszuordnungen des *Benutzerhandbuchs für programmierbare Micro830- und Micro850-Controller*.

Bei HSC-Betriebsmodi unterstützen der Haupt-HSC und Unter-HSC verschiedene Modi.

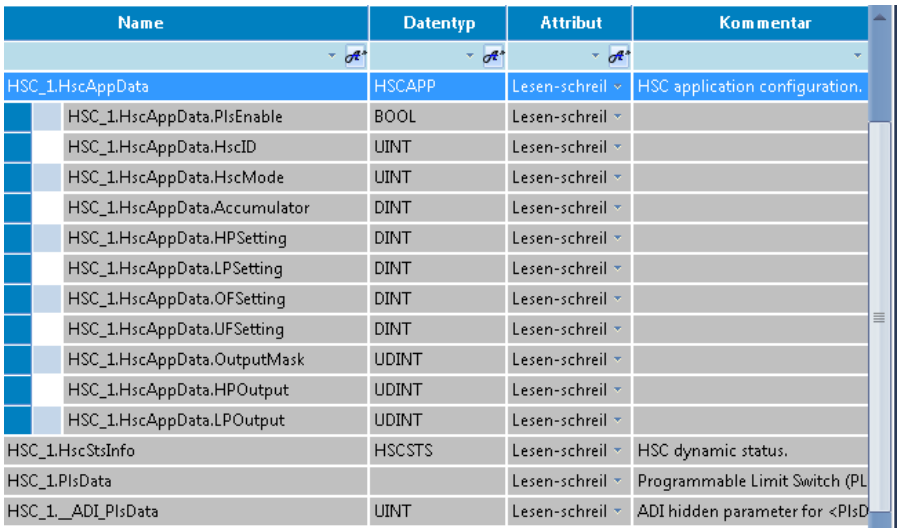
- Die Haupthochgeschwindigkeitszähler unterstützen zehn Betriebsmodi.
- Unterhochgeschwindigkeitszähler unterstützen fünf Betriebsmodi (Modus 0, 2, 4, 6, 8).
- Wenn der Haupthochgeschwindigkeitszähler sich im Modus 1, 3, 5, 7 oder 9 befindet, wird der Unterhochgeschwindigkeitszähler deaktiviert.

HSCMode	Zählmodus
0	Aufwaertszähler. Der Akkumulator wird sofort gelöscht (0), sobald er den hohen Voreinstellungswert erreicht. Eine niedrige Voreinstellung kann in diesem Modus nicht definiert werden.
1	Aufwärtszähler mit externer Zurücksetz- und Haltefunktion. Der Akkumulator wird sofort gelöscht (0), sobald er den hohen Voreinstellungswert erreicht. Eine niedrige Voreinstellung kann in diesem Modus nicht definiert werden.
2	Zähler mit externer Richtung.
3	Zähler mit externer Richtung mit Zurücksetz- und Haltefunktion.

HSCMode	Zählmodus
4	Zwei Eingangszähler (hoch und runter).
5	Zwei Eingangszähler (hoch und runter) mit externer Zurücksetz- und Haltefunktion.
6	Quadratur-Zähler (phasenverschobene Eingänge A und B).
7	Quadratur-Zähler (phasenverschobene Eingänge A und B) mit externer Zurücksetz und Haltefunktion.
8	Quadratur-X4-Zähler (phasenverschobene Eingänge A und B).
9	Quadratur-X4-Zähler (phasenverschobene Eingänge A und B) mit externer Zurücksetz und Haltefunktion.

Beispiel für HSCAppData-Parameter

Das folgende Bild veranschaulicht die HSCAppData-Parameter im **Variablenselektor**.



Name	Datentyp	Attribut	Kommentar
HSC_1.HscAppData	HSCAPP	Lesen-schreil	HSC application configuration.
HSC_1.HscAppData.PlsEnable	BOOL	Lesen-schreil	
HSC_1.HscAppData.HscID	UINT	Lesen-schreil	
HSC_1.HscAppData.HscMode	UINT	Lesen-schreil	
HSC_1.HscAppData.Accumulator	DINT	Lesen-schreil	
HSC_1.HscAppData.HPSetting	DINT	Lesen-schreil	
HSC_1.HscAppData.LPSetting	DINT	Lesen-schreil	
HSC_1.HscAppData.OFSetting	DINT	Lesen-schreil	
HSC_1.HscAppData.UFSetting	DINT	Lesen-schreil	
HSC_1.HscAppData.OutputMask	UDINT	Lesen-schreil	
HSC_1.HscAppData.HPOutput	UDINT	Lesen-schreil	
HSC_1.HscAppData.LPOutput	UDINT	Lesen-schreil	
HSC_1.HscStsInfo	HSCSTS	Lesen-schreil	HSC dynamic status.
HSC_1.PlsData		Lesen-schreil	Programmable Limit Switch (PL
HSC_1.__ADI_PlsData	UINT	Lesen-schreil	ADI hidden parameter for <PlsD

Datentyp HSCSTS

HSCSTSInfo (Datentyp HSCSTS) zeigt den Status des Hochgeschwindigkeitszählers an.

HSCSTSInfo-Statusaktionen

Während der HSC-Zählung treten die nachstehenden HSC-Statusaktionen auf.

- Wenn der HSC-Funktionsblock mit dem Befehl 0x01 zählt, wird der HSC-Status kontinuierlich aktualisiert.
- Wenn ein Fehler auftritt, wird das Kennzeichen „Error_Detected“ aktiviert und ein Fehlercode eingestellt.

HSCSTSInfo-Parameter

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm	Beschreibung
CountEnable	BOOL	0.. 9	Schreibgeschützt	Zählung aktiviert.
ErrorDetected	BOOL	0.. 9	Read/Write	Ein anderer Wert als null bedeutet, dass ein Fehler erkannt wurde.
CountUpFlag	BOOL	0.. 9	Schreibgeschützt	Kennzeichen zum Hochzählen.

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm	Beschreibung
CountDwnFlag	BOOL	2.. 9	Schreibgeschützt	Kennzeichen zum Herunterzählen.
Mode1Done	BOOL	0 oder 1	Read/Write	Der HSC befindet sich in Modus 1A oder Modus 1B und der Akkumulator zählt hoch bis zum HP-Wert.
OVF	BOOL	0.. 9	Read/Write	Es wurde ein Überlauf erkannt.
UNF	BOOL	0.. 9	Read/Write	Es wurde ein Unterlauf erkannt.
CountDir	BOOL	0.. 9	Schreibgeschützt	1: hochzählen; 0: herunterzählen
HPReached	BOOL	2.. 9	Read/Write	Hoher Voreinstellungswert erreicht.
LPReached	BOOL	2.. 9	Read/Write	Niedriger Voreinstellungswert erreicht.
OFCauseInter	BOOL	0.. 9	Read/Write	Durch einen Überlauf wurde eine HSC-Unterbrechung ausgelöst.
UFCauseInter	BOOL	2.. 9	Read/Write	Durch eine HSC.-Unterbrechung ausgelöster Unterlauf.
HPCauseInter	BOOL	0.. 9	Read/Write	Hoher Voreinstellungswert erreicht, was zu einer HSC-Unterbrechung führt.
LPCauseInter	BOOL	2.. 9	Read/Write	Niedriger Voreinstellungswert erreicht, was zu einer HSC-Unterbrechung führt.
PlsPosition	UINT	0.. 9	Schreibgeschützt	Position des programmierbaren Endschalters (Programmable Limit Switch, PLS). Der Parameter „PLSPosition" wird nach Abschluss des gesamten Zyklus und Erreichen des HP-Werts zurückgesetzt.
ErrorCode	UINT	0.. 9	Read/Write	Zeigt die vom HSC-Untersystem ermittelten Fehlercodes.
Accumulator	DINT		Read/Write	Istwert des Akkumulators.
HP	DINT		Schreibgeschützt	Letzter hoher Voreinstellungswert.
LP	DINT		Schreibgeschützt	Letzter niedriger Voreinstellungswert.
HPOutput	UDINT		Read/Write	Letzter hoher voreingestellter Ausgabewert.
LPOutput	UDINT		Read/Write	Letzter niedriger voreingestellter Ausgabewert.

HSCSTSInfo-Parameterdetails

HSCSTSInfo-Parameter (Datentyp HSCSTS) zeigen den Status des Hochgeschwindigkeitszählers an.

CountEnable

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.CountEnable	BOOL	0.. 9	Schreibgeschützt

Gibt den Status des Hochgeschwindigkeitszählers an, also ob die Zählung aktiviert (1) oder deaktiviert ist (0, Standard).

ErrorDetected

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.ErrorDetected	BOOL	0.. 9	Read/Write

Ermittelt, ob im HSC-Untersystem ein Fehler vorliegt. Konfigurationsfehler sind die am häufigsten auftretenden Fehlertypen und werden als „ErrorDetectedr" ausgegeben. Wenn für das Bit der Wert (1) festgelegt wurde, müssen Sie sich den in Parameter HSCSTS.ErrorCode angegebenen Fehlercode ansehen, der vom Controller verwaltet wird. Sie können das ErrorDetected-Bit gegebenenfalls löschen.

CountUpFlag

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.CountUpFlag	BOOL	0.. 9	Schreibgeschützt

werden mit allen HSCs eingesetzt (Modi 0...9). Wenn das HSCSTS.CountEnable-Bit festgelegt wurde, wird das Hochzähl-Bit (1) festgelegt. Wenn das HSCSTS.CountEnable-Bit gelöscht wurde, wird das Hochzähl-Bit (0) gelöscht.

CountDownFlag

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.CountDownFlag	BOOL	2.. 9	Schreibgeschützt

wird mit bidirektionalen Zählern eingesetzt (Modi 2...9). Wenn das HSCSTS.CountEnable-Bit festgelegt wurde, wird das Herunterzähl-Bit (1) festgelegt. Wenn das HSCSTS.CountEnable-Bit festgelegt wurde, wird das Herunterzähl-Bit (0) festgelegt.

Mode1Done

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.Mode1Done	BOOL	0 oder 1	Read/Write

Das HSC-Untersystem setzt das HSCSTS.Mode1Done-Statuskennzeichen auf (1), wenn der HSC für das Verhalten von Modus 0 oder 1 konfiguriert ist und der Akkumulator bis zum hohen Voreinstellungswert zählt.

OVF

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.OVF	BOOL	0.. 9	Read/Write

Das HSC-Untersystem setzt das HSCSTS.OVF-Statuskennzeichen auf (1), wenn der kumulierte Wert (HSCSTS.Accumulator) bis zur Überlauf variable (HSCAPP.OFSetting) gezählt hat. Dieses Bit ist vorläufig und wird vom HSC-Untersystem festgelegt. Das Steuerprogramm kann einen Überlaufzustand einsetzen, nachverfolgen und löschen (0).

Überlaufzustände generieren keinen Fehlerstatus des Controllers.

UNF

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.UNF	BOOL	0.. 9	Read/Write

Das HSC-Untersystem setzt das HSCSTS.UNF-Statuskennzeichen auf (1), wenn der kumulierte Wert (HSCSTS.Accumulator) bis zur Unterlaufvariable (HSCAPP.UFSetting) gezählt hat. Dieses Bit ist vorläufig und wird vom HSC-Untersystem festgelegt. Das Steuerprogramm kann einen Unterlaufzustand einsetzen, nachverfolgen und löschen (0).

Unterlaufzustände generieren keinen Fehlerstatus des Controllers.

CountDir

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.CountDir	BOOL	0.. 9	Schreibgeschützt

Das HSC-Untersystem steuert das HSCSTS.CountDir-Statuskennzeichen. Wenn der HSC-Akkumulator hoch zählt, wird das Richtungskennzeichen auf (1) gesetzt. Wenn der HSC-Akkumulator runter zählt, wird das Richtungskennzeichen gelöscht (0).

Wenn der kumulierte Wert gestoppt wird, behält das Richtungsbit seinen Wert bei. Das Richtungskennzeichen ändert sich nur, wenn der kumulierte Zählwert umgekehrt wird.

Dieses Bit wird vom HSC-Untersystem kontinuierlich aktualisiert, wenn sich der Controller in einem Betriebsmodus befindet.

HPReached

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.HPReached	BOOL	2.. 9	Read/Write

Das HSC-Untersystem setzt das HSCSTS.HPReached-Statuskennzeichen auf (1), wenn der kumulierte Wert (HSCSTS.Accumulator) größer als oder gleich der hohen voreingestellten Variable (HSCAPP.HPSetting) ist.

Dieses Bit wird vom HSC-Untersystem kontinuierlich aktualisiert, wenn sich der Controller in einem Ausführungsmodus befindet. Das Schreiben in dieses Element wird nicht empfohlen.

LPReached

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.LPReached	BOOL	2.. 9	Schreibgeschützt

Das HSC-Untersystem setzt das HSCSTS.LPReached-Statuskennzeichen auf (1), wenn der kumulierte Wert (HSCSTS.Accumulator) kleiner als oder gleich der niedrigen voreingestellten Variable (HSCAPP.LPSetting) ist.

Dieses Bit wird vom HSC-Untersystem kontinuierlich aktualisiert, wenn sich der Controller in einem Ausführungsmodus befindet. Das Schreiben in dieses Element wird nicht empfohlen.

OFCauseInter

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.OFCauseInter	BOOL	0.. 9	Read/Write

Das Überlaufunterbrechungs-Statusbit wird auf (1) gesetzt, wenn der HSC-Akkumulator bis zum Überlaufwert zählt und die HSC-Unterbrechung ausgelöst wird. Dieses Bit kann im Steuerprogramm eingesetzt werden, um zu ermitteln, ob die Überlaufvariable die HSC-Unterbrechung verursacht hat. Falls das Steuerprogramm eine bestimmte Steuerungsaktion basierend auf dem Überlauf durchführen muss, wird dieses Bit als Bedingungslogik eingesetzt.

Dieses Bit kann vom Steuerprogramm gelöscht werden (0) und wird vom HSC-Untersystem gelöscht, wenn diese Zustände erkannt werden:

- Niedrige voreingestellte Unterbrechung wird ausgeführt
- Hohe voreingestellte Unterbrechung wird ausgeführt
- Unterlauf-Unterbrechung wird durchgeführt

UFCauseInter

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.UFCauseInter	BOOL	2.. 9	Read/Write

Das Unterlaufunterbrechungs-Statusbit wird auf (1) gesetzt, wenn der HSC-Akkumulator bis zum Unterlaufwert zählt und die HSC-Unterbrechung ausgelöst wird. Dieses Bit kann im Steuerprogramm eingesetzt werden, um zu ermitteln, ob die Unterlaufvariable die HSC-Unterbrechung verursacht hat. Falls das Steuerprogramm eine bestimmte Steuerungsaktion basierend auf dem Unterlauf durchführen muss, wird dieses Bit als Bedingungslogik eingesetzt.

Dieses Bit kann vom Steuerprogramm gelöscht werden (0) und wird vom HSC-Untersystem gelöscht, wenn diese Zustände erkannt werden:

- Niedrige voreingestellte Unterbrechung tritt auf
- Hohe voreingestellte Unterbrechung tritt auf
- Überlauf-Unterbrechung tritt auf

HPCauseInter

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.HPCauseInter	BOOL	0.. 9	Read/Write

Das hohe voreingestellte Unterbrechungs-Statusbit wird auf (1) gesetzt, wenn der HSC-Akkumulator den hohen Voreinstellungswert erreicht hat und die HSC-Unterbrechung ausgelöst wird. Dieses Bit kann im Steuerprogramm eingesetzt werden, um zu ermitteln, ob der hohe Voreinstellungszustand die HSC-Unterbrechung verursacht hat. Falls das Steuerprogramm eine bestimmte Steuerungsaktion basierend auf der hohen Voreinstellung durchführen muss, wird dieses Bit als Bedingungslogik eingesetzt.

Dieses Bit kann vom Steuerprogramm gelöscht werden (0) und wird vom HSC-Untersystem gelöscht, wenn diese Zustände erkannt werden:

- Niedrige voreingestellte Unterbrechung tritt auf
- Unterlauf-Unterbrechung tritt auf
- Überlauf-Unterbrechung tritt auf

LPCauseInter

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.LPCauseInter	BOOL	2.. 9	Read/Write

Das niedrige voreingestellte Unterbrechungs-Statusbit wird auf (1) gesetzt, wenn der HSC-Akkumulator den niedrigen Voreinstellungswert erreicht hat und die HSC-Unterbrechung ausgelöst wird. Dieses Bit kann im Steuerprogramm eingesetzt werden, um zu ermitteln, ob der niedrige Voreinstellungszustand die HSC-Unterbrechung verursacht hat. Falls das Steuerprogramm eine bestimmte Steuerungsaktion basierend auf der niedrigen Voreinstellung durchführen muss, wird dieses Bit als Bedingungslogik eingesetzt.

Dieses Bit kann vom Steuerprogramm gelöscht werden (0) und wird vom HSC-Untersystem gelöscht, wenn diese Zustände erkannt werden:

- Hohe voreingestellte Unterbrechung tritt auf
- Unterlauf-Unterbrechung tritt auf
- Überlauf-Unterbrechung tritt auf

PlsPosition

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.PLSPosition	UINT	0.. 9	Schreibgeschützt

Wenn sich der HCS im Zählungsmodus befindet und PLS aktiviert ist, gibt dieser Parameter an, welches PLS-Element für die aktuelle HSC-Konfiguration verwendet wird.

ErrorCode

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.ErrorCode	BOOL	0.. 9	Schreibgeschützt

Zeigt die vom HSC-Untersystem ermittelten Fehlercodes.

Unterelement des Fehlercodes	Fehlercode beim HSC-Zählvorgang	Zugriff auf Benutzerprogramm
Bit 15-8 (hohes Byte)	0-255	Der Nicht-Null-Wert für das hohe Byte gibt an, dass der HSC-Fehler durch die PLS-Dateneinstellung verursacht wurde. Der Wert des hohen Byte gibt an, durch welches Element der PLS-Daten der Fehler ausgelöst wird.
Bit 7-0 (niedriges Byte)	0x00	Es tritt kein Fehler auf.
	0x01	Ungültiger HSC-Zählmodus.
	0x02	Ungültiger hoher Voreinstellungswert.
	0x03	Ungültiger Überlauf.
	0x04	Ungültiger Unterlauf.
	0x05	Keine PLS-Daten.

Accumulator

Parameter	Datentyp	Zugriff auf Benutzerprogramm
HSCApp.Accumulator	DINT	Read/Write

Legt den Anfangswert des Akkumulators fest, wenn der Hochgeschwindigkeitszähler startet. Wenn sich der HSC im Zählmodus befindet, wird der Akkumulator automatisch vom HSC-Untersystem aktualisiert, um den tatsächlichen HSC-Akkumulatorwert auszugeben.

HP

Parameter	Datentyp	Zugriff auf Benutzerprogramm
HSCSTS.HP	DINT	Schreibgeschützt

HSCSTS.HP ist der obere Sollwert (bei Zählungen), der definiert, wann ein HSC-Untersystem eine Unterbrechung generiert.

Die in die hohe Voreinstellung geladenen Daten müssen kleiner oder gleich den Daten im Überlauf-Parameter (HSCAPP.OFSetting) sein, ansonsten wird ein HSC-Fehler ausgegeben.

Das ist der letzte hohe Voreinstellungswert, der möglicherweise vom PLS-Datenblock durch die PLS-Funktion aktualisiert wird.

LP

Parameter	Datentyp	HSC-Modus	Zugriff auf Benutzerprogramm
HSCSTS.LP	DINT		Schreibgeschützt

HSCSTS.LP ist der untere Sollwert (bei Zählungen), der definiert, wann ein HSC-Untersystem eine Unterbrechung generiert.

Die in die niedrige Voreinstellung geladenen Daten müssen größer oder gleich den Daten im Unterlauf-Parameter (HSCAPP.UFSetting) sein, ansonsten wird ein HSC-Fehler ausgegeben. Wenn der Unterlauf und die niedrigen voreingestellten Werte negative Zahlen sind, muss die niedrige Voreinstellung eine Zahl mit einem kleineren absoluten Wert sein.

Das ist der letzte niedrige Voreinstellungswert, der möglicherweise vom PLS-Datenblock durch die PLS-Funktion aktualisiert wird.

HPOutput

Parameter	Datentyp	Zugriff auf Benutzerprogramm
HSCApp.HPOutput	UDINT	Read/Write

Definiert den Zustand (1 = EIN oder 0 = AUS) der Ausgaben am Controller, wenn eine hohe Voreinstellung erreicht wird. Weitere Informationen zum direkten Ein- oder Ausschalten von Ausgaben basierend auf der hohen Voreinstellung.

Das hohe Ausgabe-Bitmuster können Sie während der Einrichtung konfigurieren oder Sie verwenden den HSC-Funktionsblock, um die neuen Parameter zu laden, während der Controller in Betrieb ist.

LPOutput

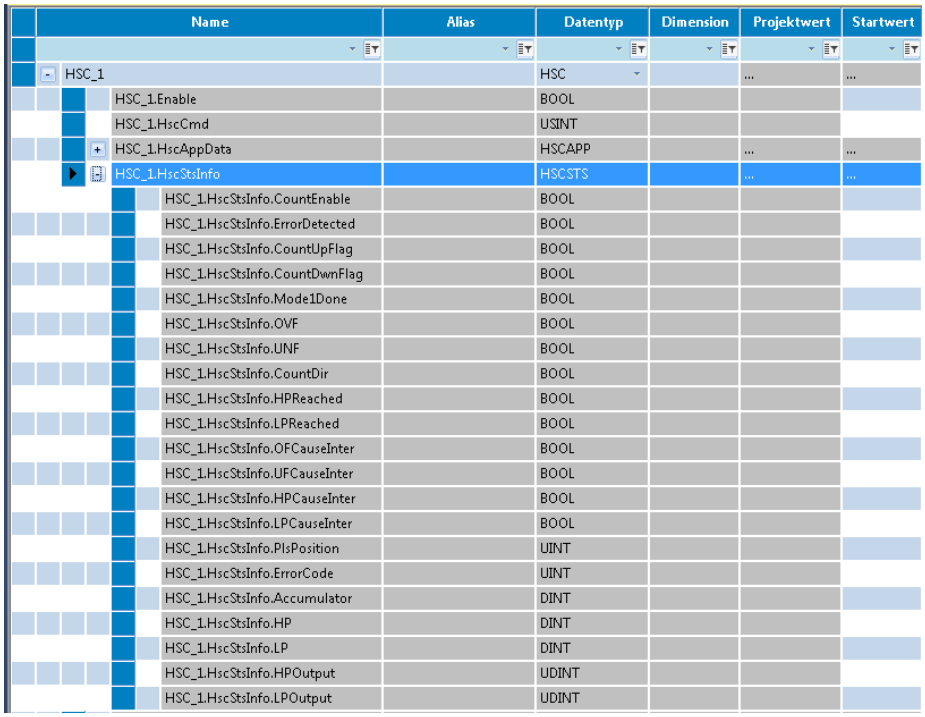
Parameter	Datentyp	Zugriff auf Benutzerprogramm
HSCApp.LPOutput	UDINT	Read/Write

LPOutput (HSCApp.LPOutput) definiert den Zustand (1 = "an", 0 = "aus ") der Ausgaben am Controller, wenn die niedrige Voreinstellung erreicht wurde. Weitere Informationen zum direkten Ein- oder Ausschalten von Ausgaben basierend auf der unteren Voreinstellung.

Das niedrige Ausgabe-Bitmuster können Sie während der Einrichtung konfigurieren oder Sie verwenden den HSC-Funktionsblock, um die neuen Parameter zu laden, während der Controller in Betrieb ist.

Beispiel für HSCSTSInfo-Parameter

Das nachstehende Bild zeigt die HSCStsInfo-Parameter im **Variablensektor**.



Name	Alias	Datentyp	Dimension	Projektwert	Startwert
HSC_1		HSC		...	...
HSC_1.Enable		BOOL			
HSC_1.HscCmd		USINT			
HSC_1.HscAppData		HSCAPP		...	...
HSC_1.HscStsInfo		HSCSTS		...	...
HSC_1.HscStsInfo.CountEnable		BOOL			
HSC_1.HscStsInfo.ErrorDetected		BOOL			
HSC_1.HscStsInfo.CountUpFlag		BOOL			
HSC_1.HscStsInfo.CountDwnFlag		BOOL			
HSC_1.HscStsInfo.Mode1Done		BOOL			
HSC_1.HscStsInfo.OVF		BOOL			
HSC_1.HscStsInfo.UNF		BOOL			
HSC_1.HscStsInfo.CountDir		BOOL			
HSC_1.HscStsInfo.HPReached		BOOL			
HSC_1.HscStsInfo.LPReached		BOOL			
HSC_1.HscStsInfo.OFCauseInter		BOOL			
HSC_1.HscStsInfo.UFCauseInter		BOOL			
HSC_1.HscStsInfo.HPCauseInter		BOOL			
HSC_1.HscStsInfo.LPCauseInter		BOOL			
HSC_1.HscStsInfo.PlsPosition		UINT			
HSC_1.HscStsInfo.ErrorCode		UINT			
HSC_1.HscStsInfo.Accumulator		DINT			
HSC_1.HscStsInfo.HP		DINT			
HSC_1.HscStsInfo.LP		DINT			
HSC_1.HscStsInfo.HPOutput		UDINT			
HSC_1.HscStsInfo.LPOutput		UDINT			

Datentyp PLS

PLSData (Datentyp PLS) wird verwendet, um den programmierbaren Endschalter zu konfigurieren auf seite 313.

PLSData-Strukturelemente

Die PLS-Datenstruktur ist ein flexibles Array mit folgenden Elementen.

Element	Element-Reihenfolge	Datentyp	Beschreibung des Elements
HSCHP	Wort 0...1	DINT	Obere Voreinstellung
HSCLP	Wort 2...3	DINT	Untere Voreinstellung
HSCHPOutput	Wort 4...5	UDINT	Ausgabe hoher Datenwerte
HSCLPOutput	Wort 6...7	UDINT	Ausgabe niedriger Datenwerte

Die Gesamtzahl der Elemente für eine PLS-Datenstruktur sollte 255 nicht übersteigen.

PLSData-Parameter

In der folgenden Tabelle sind die PLSData-Parameterdetails aufgeführt.

Element	Datentyp	Datenformat	HSC-Modus	Zugriff auf Benutzerprogramm	Beschreibung
HSCHP	DINT	32-Bit-Ganzzahl mit Vorzeichen	0	Read/Write	Obere Voreinstellung
HSCLP	DINT	32-Bit-Ganzzahl mit Vorzeichen	0	Read/Write	Untere Voreinstellung
HSCHPOutput	UDINT	32-Bit-Binärzahl	0	Read/Write	Ausgabe hoher Datenwerte
HSCLPOutput	UDINT	32-Bit-Binärzahl	0	Read/Write	Ausgabe niedriger Datenwerte

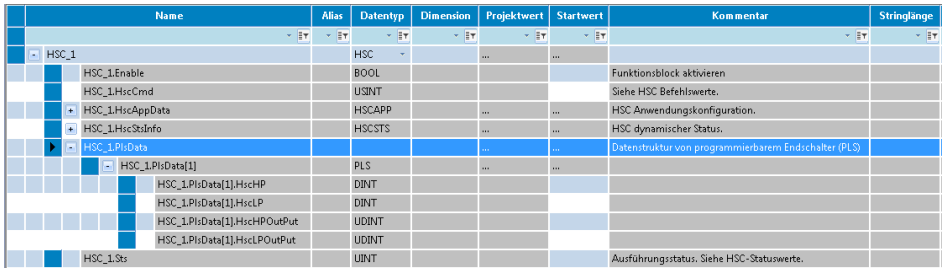
HSCApp-Einstellungen im Vergleich zu PLSData-Einstellungen

Bei aktivierter PLS-Funktion werden relevante HSCApp-Einstellungen gemäß der nachstehenden Tabelle durch die entsprechenden PLSData-Einstellungen abgelöst.

HSCApp-Einstellung	PLSData-Einstellung
HSCAPP.HpSetting	HSCHP
HSCAPP.LpSetting	HSCLP
HSCAPP.HPOutput	HSCHPOutput
HSCAPP.LPOutput	HSCLPOutput

Beispiel für PLSData-Parameter

Das nachstehende Bild zeigt die PLSData-Parameter im **Variablensektor**.



Datentyp HSCE_CHANNEL

Der Datentyp HSCE_CHANNEL wird wie folgt beschrieben:

```
@typedef struct struct_HSCE_channel
{

USINT    ModuleType;
USINT    SlotID;
USINT    HSCID;
} HSCE_CHANNEL;
```

Es folgt die Beschreibung:

Byte	Beschreibung
Modultyp	0x00: Integriert 0x01: Erweiterung (Scheibenmodul) 0x02: Universeller Anschluss
Schacht-ID	0x00: Integriert 0x01-0x1F: ID des Erweiterungsmoduls (Scheibenmodul)

Byte	Beschreibung
	0x01-0x05: ID des universellen Anschlusses
HSCID	0x00-0x0F: Integriert 0x00-0x07: ID des HSC auf seite 290 zur Erweiterung 0x00-0x01: ID des HDC für den universellen Anschluss

Datentyp HSCE_STS

Es folgt die Beschreibung des Datentyps HSCE_STS:

Parameter	Datentyp	HSC-Modus	Beschreibung
CountEnable	BOOL	0.. 13	Zählung aktiviert.
ErrorDetected	BOOL	0.. 13	Ein anderer Wert als null bedeutet, dass ein Fehler erkannt wurde.
CountUpFlag	BOOL	0.. 13	Kennzeichen zum Hochzählen.
CountDwnFlag	BOOL	2.. 13	Kennzeichen zum Herunterzählen.
Mode1Done	BOOL	0 oder 1	Der HSC befindet sich in Modus 1A oder Modus 1B und der Akkumulator zählt hoch bis zum HP-Wert.
OVF	BOOL	0.. 13	Es wurde ein Überlauf erkannt.
UNF	BOOL	0.. 13	Es wurde ein Unterlauf erkannt.
CountDir	BOOL	0.. 13	1: hochzählen; 0: herunterzählen
HPReached	BOOL	2.. 13	Hoher Voreinstellungswert erreicht.
LPReached	BOOL	2.. 13	Niedriger Voreinstellungswert erreicht.
OFCauseInter	BOOL	0.. 13	Durch einen Überlauf wurde eine HSC auf seite 290-Unterbrechung ausgelöst.
UFCauseInter	BOOL	2.. 13	Durch eine HSC-Unterbrechung ausgelöster Unterlauf.
HPCauseInter	BOOL	0.. 13	Hoher Voreinstellungswert erreicht, was zu einer HSC-Unterbrechung führt.
LPCauseInter	BOOL	2.. 13	Niedriger Voreinstellungswert erreicht, was zu einer HSC-Unterbrechung führt.
StateInfo	USINT		Informationen zum Zustandsautomaten des HSCE-Zählers
PlsPosition	UINT	0.. 9	Position des programmierbaren Endschalters (Programmable Limit Switch, PLS). Der Parameter „PLSPosition" wird nach Abschluss des gesamten Zyklus und Erreichen des HP-Werts zurückgesetzt.
ErrorCode	UINT	0.. 13	Zeigt die vom HSC-Untersystem ermittelten Fehlercodes.
Accumulator	DINT		Istwert des Akkumulators.
HP	DINT		Letzter hoher Voreinstellungswert.
LP	DINT		Letzter niedriger Voreinstellungswert.
HPOutput	UDINT		Letzter hoher voreingestellter Ausgabewert.
LPOutput	UDINT		Letzter niedriger voreingestellter Ausgabewert.

Datentyp PLS_HSCE

Beim programmierbaren Endschalter handelt es sich um einen zusätzlichen Satz an Betriebsmodi für den Hochgeschwindigkeitszähler. Bei Betrieb in diesen Modi werden die voreingestellten Datenwerte wie auch die Ausgabe-Datenwerte bei jedem Erreichen einer der Voreinstellungen mittels benutzerseitig bereitgestellter Daten aktualisiert. Sie können diese Modi programmieren, indem Sie eine PLS-Datei mit den zu verwendenden Datensätzen bereitstellen. Die PLS_HSCE-Datenstruktur ist ein flexibles Datenfeld, dessen Elemente wie folgt definiert sind:

Element	Datentyp	Beschreibung des Elements
HighPreset	LINT	Hoher Voreinstellungswert
LowPreset	LINT	Niedriger Voreinstellungswert

Element	Datentyp	Beschreibung des Elements
HiPresetOutput	UDINT	Ausgangsdaten mit hohem Voreinstellungswert
LoPresetOutput	UDINT	Ausgangsdaten mit niedrigem Voreinstellungswert

```
// Struktur des PLS-Elements
typedef struct PLS_HSCE_EleStruct
{
LINT HighPreset;     // Hoher Voreinstellungswert für HSC
LINT LowPreset;      // Niedriger Voreinstellungswert für
HSC
UDINT HiPresetOutput; // Hoher
Ausgangs-Voreinstellungswert für HSC
UDINT LoPresetOutput; // Niedriger
Ausgangs-Voreinstellungswert für HSC
} PLS_HSCE_EleStruct;
```

Die Gesamtzahl der Elemente einer PLS_HSCE-Datenstruktur sollte bei einem HSC-Plug-in 24 nicht übersteigen.

HSC_SET_STS (Hochgeschwindigkeitszählerstatus festlegen)

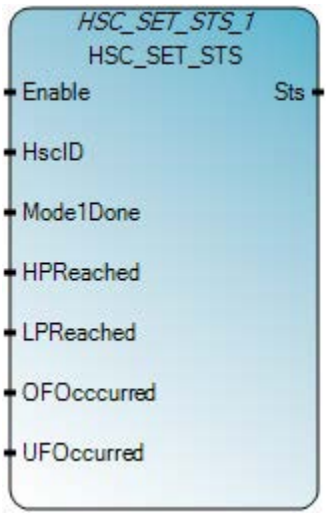
HSC_SET_STS legt den HSC-Zählstatus manuell fest oder setzt ihn zurück.

Details zum Betrieb:

- Der HSC-Funktionsblock muss für den HSC_SET_STC-Funktionsblock gestoppt werden (zählt nicht), damit der HTS-Status festgelegt oder zurückgesetzt werden kann. Wenn der HSC-Funktionsblock nicht gestoppt wird, werden die Eingabeparameter weiterhin aktualisiert, und mithilfe von HSC_SET_STS vorgenommene Änderungen werden ignoriert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für Micro830-, Micro850- und Micro870-Controller.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung

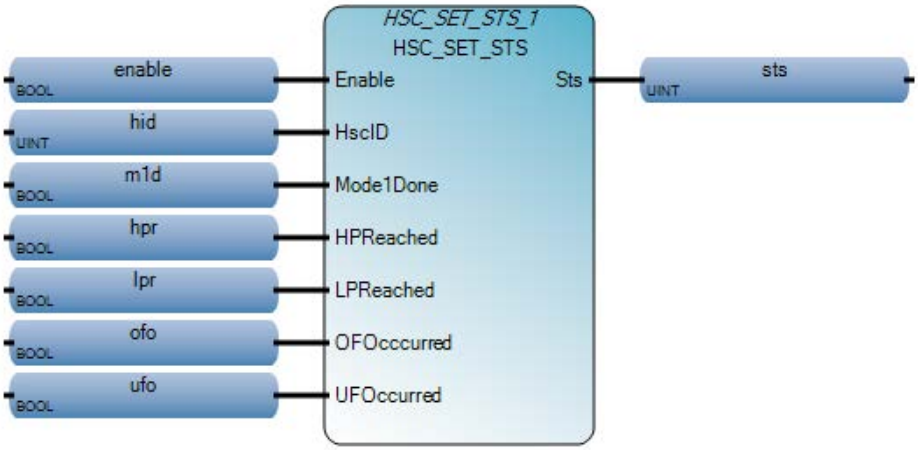
Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Festlegen/Zurücksetzen des HSC-Status. FALSE – es ist keine HSC-Statusänderung vorhanden.
HscID	Eingang	UINT	Legt den HSC-Status manuell fest oder setzt ihn zurück.
Mode1Done	Eingang	BOOL	Modus 1A oder 1B Zählung wird durchgeführt. Dieses Bit kann eingestellt oder zurückgesetzt werden, wenn HSC nicht zählt.
HPReached	Eingang	BOOL	Hoher Voreinstellungswert erreicht. Dieses Bit kann eingestellt oder zurückgesetzt werden, wenn HSC nicht zählt.
LPReached	Eingang	BOOL	Niedriger Voreinstellungswert erreicht. Dieses Bit kann eingestellt oder zurückgesetzt werden, wenn HSC nicht zählt.
OFOccurred	Eingang	BOOL	Ein Überlauf ist aufgetreten. Dieses Bit kann eingestellt oder zurückgesetzt werden, wenn HSC nicht zählt.
UFOccurred	Eingang	BOOL	Ein Unterlauf ist aufgetreten. Dieses Bit kann eingestellt oder zurückgesetzt werden, wenn HSC nicht zählt.
Sts	Ausgang	UINT	Statuscodes werden in den HSC-Statuscodes (Sts) definiert.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

HSC-Statuscodes (STS)

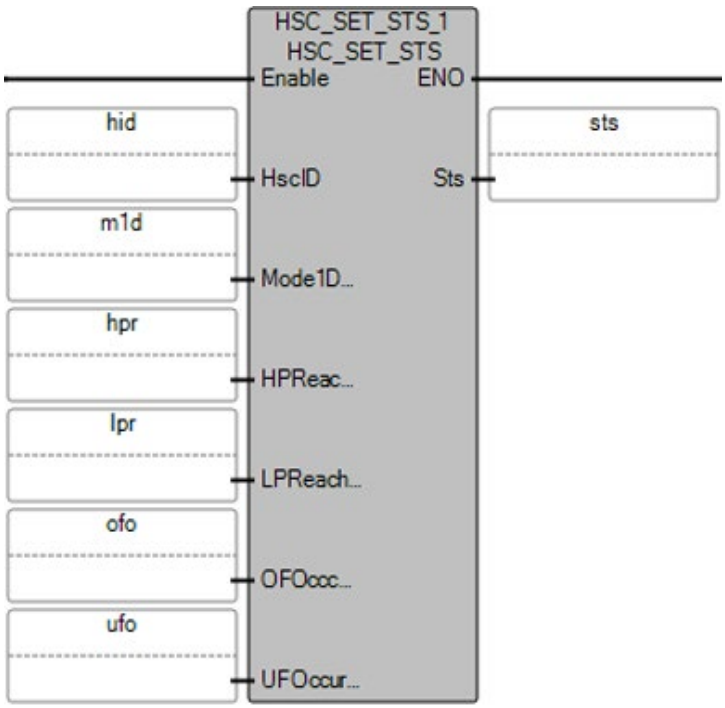
In der folgenden Tabelle werden die Statuscodes für den HSC-Funktionsblock beschrieben.

Statuscode	Statusbeschreibung
0x00	Keine Aktion ergriffen (nicht aktiviert).
0x01	HSC-Ausführung erfolgreich.
0x02	HSC-Befehl ungültig.
0x03	HSC-ID außerhalb des Gültigkeitsbereichs.
0x04	HSC-Konfigurationsfehler.

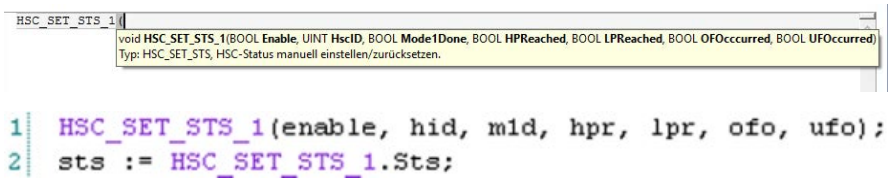
Diagrammbeispiel für Funktionsblock HSC_SET_STS



HSC_SET_STS-Kontaktplan-Beispiel



Beispiel für HSC_SET_STS in strukturiertem Text



Verwenden der Hochgeschwindigkeitszähler-Anweisungen

Dieser Abschnitt stellt bestimmte Details und Beispiele zum Einsatz von Hochgeschwindigkeitszähler-Anweisungen in Logik-Programmen bereit, einschließlich folgender Funktionen:

Aktualisieren der HSC-Anwendungsdaten auf seite 308

Konfigurieren einer Hochgeschwindigkeitszähler-Benutzerunterbrechung (HSC) auf seite 309

Konfigurieren eines programmierbaren Endschalters (PLS) auf seite 313

Beispiel: So erstellen Sie einen Hochgeschwindigkeitszähler (HSC) auf seite 315

Beispiel: So können Sie eine Funktion für den programmierbaren Endschalter (PLS) hinzufügen auf seite 323

Beispiel: Programmierbarer Endschalter (PLS) aktiviert auf seite 324

Aktualisieren der HSC-Anwendungsdaten

Die HSC-Konfiguration wird in den HSC-Anwendungsdaten definiert und erfolgt normalerweise nur einmal vor der Programmierung der HSC-Anweisung. An den HSC-Anwendungsdaten (HSCAppData-Parameter)

vorgenommene Änderungen werden ignoriert, während der HSC zählt.

So aktualisieren Sie die HSC-Konfiguration

1. Aktualisieren Sie die HSCAppData.
2. Rufen Sie den HSC-Befehl auf seite 290 mit dem Befehl 0x03 (festlegen / erneut laden) auf.

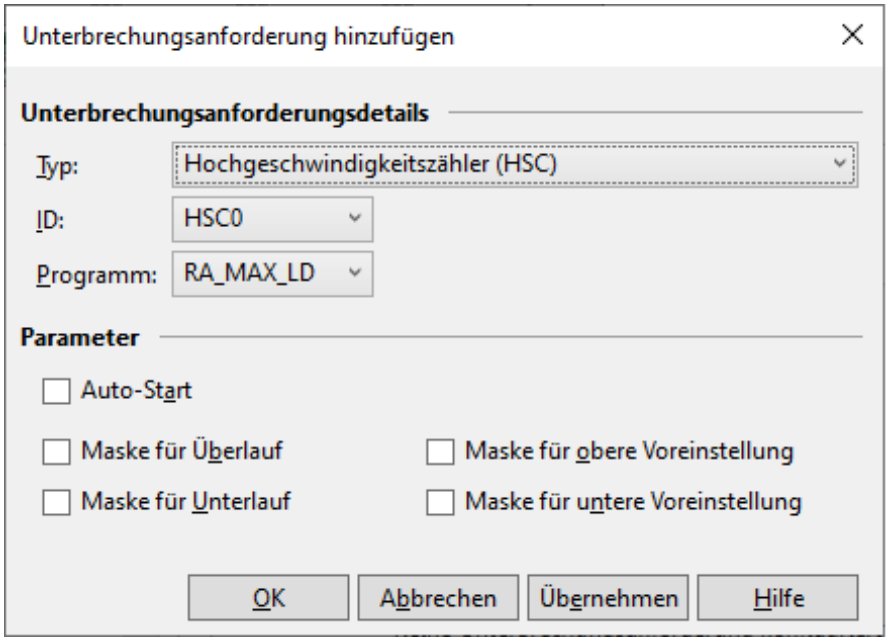
Dialogfeld zur Benutzerunterbrechung des Hochgeschwindigkeitszählers

Wo finde ich das Dialogfeld zur Benutzerunterbrechung des Hochgeschwindigkeitszählers?

Wählen Sie bei „Unterbrechungstyp“ die **Hochgeschwindigkeitszähler-Benutzerunterbrechung (HSC)**.

Verwenden Sie das Dialogfeld HSC-Unterbrechung zur:

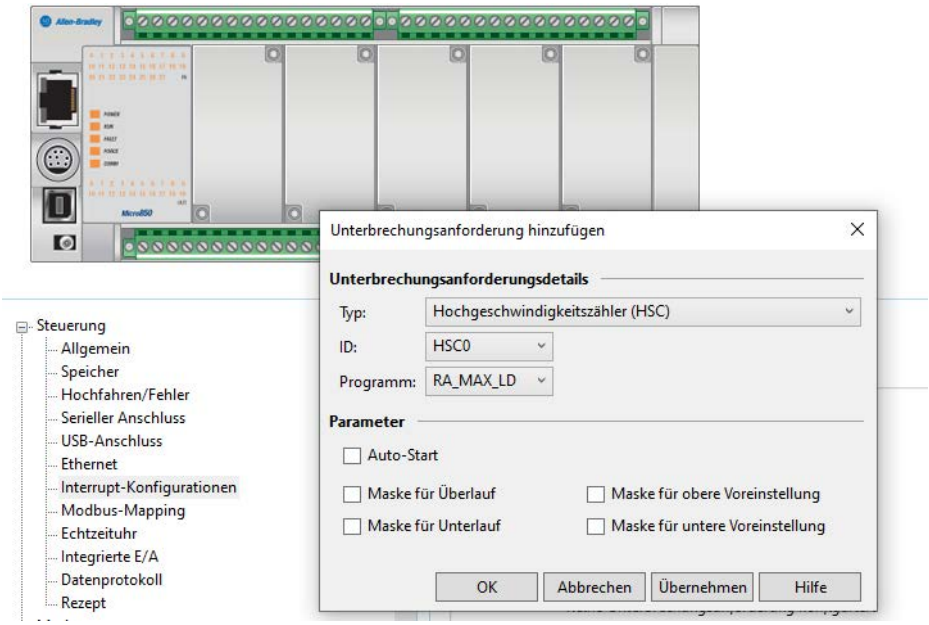
- Konfigurieren der Unterbrechungseigenschaften auf seite 309 wie z. B. der ID und des Programms, in dem sie verwendet werden soll.
- Konfiguration der Unterbrechungsparameter.



Konfigurieren einer Hochgeschwindigkeitszähler-Benutzerunterbrechung (HSC)

Eine Benutzerunterbrechung führt dazu, dass der Controller den Task, den er gerade durchführt, unterbricht, und anschließend zum unterbrochenen Task zurückkehrt.

Micro830-, Micro850- und Micro870-Controller unterstützen bis zu sechs HSC-Benutzerunterbrechungen, die eingesetzt werden können, um eine ausgewählte Benutzer-Logik zu einem vorkonfigurierten Ereignis auszuführen.



Hinzufügen und Konfigurieren einer Hochgeschwindigkeitszähler-Benutzerunterbrechung (HSC)

Um im Arbeitsbereich der Controller-Konfiguration eine HSC-Unterbrechung auf seite 309 hinzuzufügen und zu konfigurieren, gehen Sie folgendermaßen vor.

So fügen Sie eine HSC-Unterbrechung hinzu:

1. Doppelklicken Sie im **Projekt-Organizer** auf den Controller, um den Controller-Arbeitsbereich zu öffnen.
2. Klicken Sie in der **Controller**-Baumstruktur auf **Interrupts**, um die Konfigurationsseite **Interrupt** anzuzeigen.
3. Klicken Sie mit der rechten Maustaste in eine leere Zeile und klicken Sie auf **Hinzufügen**, um das **Interrupt**-Eigenschaftendialogfeld anzuzeigen.
4. So konfigurieren Sie eine HSC-Unterbrechung:
   - Wählen Sie bei „Unterbrechungstyp“ die **Hochgeschwindigkeitszähler-Benutzerunterbrechung (HSC)**.
   - Wählen Sie die HSC-Unterbrechungseigenschaften auf seite 310 aus.
   - Wählen Sie die HSC-Unterbrechungsparameter auf seite 312 aus.
5. Schließen Sie das **Interrupt**-Eigenschaftendialogfeld.

HSC-Unterbrechungseigenschaften

Die Statusbits der HSC-Unterbrechungseigenschaften zeigen den Status aktiviert/deaktiviert oder den Ausführungsstatus an und geben an, ob ein Unterbrechungszustand verloren gegangen ist.

Benutzerunterbrechungsaktivierung (HSC0.Enabled)

Parameter	Datenformat	HSC-Modi	Zugriff auf Benutzerprogramm
HSC0.Enabled	Bit	0.. 9	Schreibgeschützt

Das aktivierte Bit wird eingesetzt, um den Aktivierungs- oder Deaktivierungsstatus der HSC-Unterbrechung anzugeben.

Benutzerunterbrechungsausführung (HSC0.EX)

Parameter	Datenformat	HSC-Modi	Zugriff auf Benutzerprogramm
HSC0.Ex	Bit	0.. 9	Schreibgeschützt

Das EX (Benutzerunterbrechungsausführung)sbit wird auf (1) gesetzt, wenn ein HSC-Untersystem durch einen der folgenden Zustände mit der Verarbeitung der HSC-Subroutine beginnt:

- Niedriger Voreinstellungswert erreicht
- Hoch-Voreinstellung erreicht
- Überlaufzustand - zählt mithilfe des Überlaufwerts hoch
- Unterlaufzustand - zählt mithilfe des Unterlaufwerts runter

Das HSC EX-Bit kann im Steuerprogramm als Bedingungslogik eingesetzt werden, um zu ermitteln, ob eine HSC-Unterbrechung ausgeführt wird.

Das HSC-Untersystem löscht die (0) des EX-Bit, wenn der Controller die Verarbeitung der HSC-Subroutine abschließt.

Benutzerunterbrechung ausstehend (HSC0.PE)

Parameter	Datenformat	HSC-Modi	Zugriff auf Benutzerprogramm
HSC0.PE	Bit	0.. 9	Schreibgeschützt

Das PE-Statuskennzeichen (Benutzerunterbrechung ausstehend) gibt an, dass eine Unterbrechung aussteht. Das PE-Statusbit kann überwacht werden oder für Logikzwecke im Steuerprogramm eingesetzt werden, wenn Sie ermitteln müssen, wann eine Subroutine nicht direkt ausgeführt werden kann. Das PE-Bit wird vom Controller verwaltet und automatisch festgelegt und gelöscht.

Benutzerunterbrechung verloren (HSC0.LS)

Parameter	Datenformat	HSC-Modi	Zugriff auf Benutzerprogramm
HSC0.LS	Bit	0.. 9	Schreibgeschützt

Die LS (Benutzerunterbrechung verloren) ist ein Statuskennzeichen, das angibt, dass eine Unterbrechung verloren gegangen ist. Der Controller kann einen aktiven Benutzerunterbrechungszustand verarbeiten und einen ausstehenden Benutzerunterbrechungszustand erhalten, bevor es das verlorene Bit festlegt.

Das LS-Bit wird vom Controller festgelegt. Das Steuerprogramm kann einen verlorenen Zustand einsetzen oder überwachen.

HSC-Unterbrechungsparameter

Die HSC-Unterbrechungsparameter werden eingesetzt, um die Start- und Maskenoptionen zu konfigurieren.

Auto-Start (HSC0.AS)

Parameter	Datenformat	HSC-Modi	Zugriff auf Benutzerprogramm
HSC0.AS	Bit	0.. 9	Schreibgeschützt

Auto-Start wird mit der Programmierung des Geräts konfiguriert und als Teil des Benutzerprogramms gespeichert. Das Auto-Start-Bit definiert, ob die HSC-Unterbrechungsfunktion automatisch starten soll, wenn ein Controller einen Betriebs- oder Testmodus startet.

Überlaufmaske (HSC0.MV)

Das Steuer-Bit **MV** (Überlaufmaske) wird dafür verwendet, das Auftreten einer Überlaufunterbrechung zu aktivieren (zu ermöglichen) oder zu deaktivieren (nicht zu ermöglichen). Wenn das Bit gelöscht ist (0) und ein Zustand **Überlauf erreicht** vom **HSC** erkannt wird, wird die **HSC**-Benutzerunterbrechung nicht ausgeführt.

Das **MV**-Bit wird vom Benutzerprogramm gesteuert und erlangt seinen Wert über einen Einschaltzyklus zurück. Das Benutzerprogramm muss das **MV**-Bit einstellen und löschen.

Parameter	Datenformat	HSC-Modi	Zugriff auf Benutzerprogramm
HSC0.MV	Bit	0.. 9	Schreibgeschützt

Unterlaufmaske (HSC0.MN)

Parameter	Datenformat	HSC-Modi	Zugriff auf Benutzerprogramm
HSC0.MN	Bit	2.. 9	Schreibgeschützt

Das Steuer-Bit MN (Unterlaufmaske) wird verwendet, um das Auftreten einer Unterlaufunterbrechung zu aktivieren (zu ermöglichen) oder zu deaktivieren (nicht zu ermöglichen). Wenn das Bit gelöscht ist (0) und ein Zustand Unterlauf erreicht vom HSC erkannt wird, wird die HSC-Benutzerunterbrechung nicht ausgeführt.

Das MN-Bit wird vom Benutzerprogramm gesteuert und erlangt seinen Wert über einen Einschaltzyklus zurück. Das Benutzerprogramm muss das MN-Bit einstellen und löschen.

Maske mit hohem Voreinstellungswert (HSC0.MH)

Parameter	Datenformat	HSC-Modi	Zugriff auf Benutzerprogramm
HSC0.MH	Bit	0.. 9	Schreibgeschützt

Das Steuer-Bit MH (Maske mit oberer Voreinstellung) wird dafür verwendet, das Auftreten einer Unterbrechung mit oberer Voreinstellung zu aktivieren (zu ermöglichen) oder zu deaktivieren (nicht zu ermöglichen). Wenn das Bit gelöscht ist (0) und ein Zustand Obere Voreinstellung erreicht vom HSC erkannt wird, wird die HSC-Benutzerunterbrechung nicht ausgeführt.

Das MH-Bit wird vom Benutzerprogramm gesteuert und erlangt seinen Wert über einen Einschaltzyklus zurück. Das Benutzerprogramm muss das MH-Bit einstellen und löschen.

Maske mit niedrigem Voreinstellungswert (HSC0.ML)

Parameter	Datenformat	HSC-Modi	Zugriff auf Benutzerprogramm
HSC0.ML	Bit	2.. 9	Schreibgeschützt

Das Steuer-Bit ML (Maske mit unterer Voreinstellung) wird dafür verwendet, das Auftreten einer Unterbrechung mit unterer Voreinstellung zu aktivieren (zu ermöglichen) oder zu deaktivieren (nicht zu ermöglichen). Wenn das Bit gelöscht ist (0) und ein Zustand Untere Voreinstellung erreicht vom HSC erkannt wird, wird die HSC-Benutzerunterbrechung nicht ausgeführt.

Das ML-Bit wird vom Benutzerprogramm gesteuert und erlangt seinen Wert über einen Einschaltzyklus zurück. Das Benutzerprogramm muss das ML-Bit einstellen und löschen.

Konfigurieren eines programmierbaren Endschalters (PLS)

Der Hochgeschwindigkeitszähler auf seite 290 verfügt über zusätzliche Betriebsmodi zur Implementierung eines programmierbaren Endschalters (Programmable Limit Switch, PLS). Die PLS-Funktion wird zur Konfiguration des Hochgeschwindigkeitszählers als PLS oder als Drehnockenschalter verwendet. Die PLS-Funktion unterstützt bis zu 255 Paare aus unterer und oberer Voreinstellung und kann verwendet werden, wenn Sie mehr als ein Paar solcher Werte benötigen.

Aktivierung des PLS in der HSC

Der PLS-Modus funktioniert nur zusammen mit dem HSC des Micro800-Controllers und muss in der HSC-Anweisung aktiviert werden, indem der Parameter „HSCAppData.PLSEnable“ auf „True“ eingestellt wird.

Der Parameter „PLSPosition“ wird nach Abschluss des gesamten Zyklus und Erreichen des HSCSTS.HP-Werts zurückgesetzt. Durch Zurücksetzen des HSC-Anweisung oder Verschieben von 0 auf den PLSPosition-Parameter wird PLSPosition nicht zurückgesetzt.

HSC-Operation mit aktivierter PLS-Funktion

Die PLS-Funktion ist mit allen anderen HSC-Funktionen kompatibel, auch mit der Fähigkeit, zu wählen, welche HSC-Ereignisse eine Benutzerunterbrechung erzeugen.

Wenn die PLS-Funktion aktiviert ist auf seite 324 und sich der Controller im Betriebsmodus befindet, zählt der HSC eingehende Impulse und es treten die folgenden Ereignisse ein.

- Wenn der Zähler den ersten in den PLS-Daten definierten Voreinstellungswert (HSCHP oder HSCLP) erreicht, werden die Ausgabequellendaten (HSCHPOutput oder HSCLPOutput) durch die HSC-Maske (HSCAPP.OutputMask) geschrieben.
- An diesem Punkt werden die nächsten in den PLS-Daten definierten Voreinstellungswerte (HSCHP und HSCLP) aktiv.
- Wenn der HSC bis zum neuen Voreinstellungswert zählt, werden die neuen Ausgabedaten durch die HSC-Maske geschrieben.
- Dieser Prozess wird fortgesetzt, bis das letzte Element innerhalb des PLS-Datenblocks geladen wird.
- An diesem Punkt wird das aktive Element innerhalb des PLS-Datenblocks auf null zurückgesetzt.
- Dieses Verhalten wird als Kreisbetrieb bezeichnet.

Die Differenz der PLS-Voreinstellung zwischen dem integrierten HSC- und Plug-in-HSC-Modul

Das Verhalten von PLS HSCHP- und HSCLP-Voreinstellung zwischen dem integrierten HSC-Modul und Plug-in-HSC-Modul weichen voneinander ab. Das Bit der oberen Voreinstellung des integrierten HSC wird nur eingestellt, wenn das letzte PLS ausgeführt wird, während das Bit der oberen Voreinstellung des Plug-in-HSC-Moduls beim ersten Ausführen des PLS eingestellt wird. Beispiel,

- Integriertes HSC PLS0-PLS23:

  Hoher Voreinstellungswert wird eingestellt, wenn PLS23LP=Akkumulatorwert
  Untere Voreinstellung wird eingestellt, wenn PLS23LP=Akkumulatorwert

- Plug-in-HSC PLS0-PLS23:

  Obere Voreinstellung wird eingestellt, wenn PLS0HP=Akkumulatorwert
  Untere Voreinstellung wird eingestellt, wenn PLS0LP=Akkumulatorwert

Beispiel: So erstellen Sie ein Hochgeschwindigkeitsprogramm (HSC)

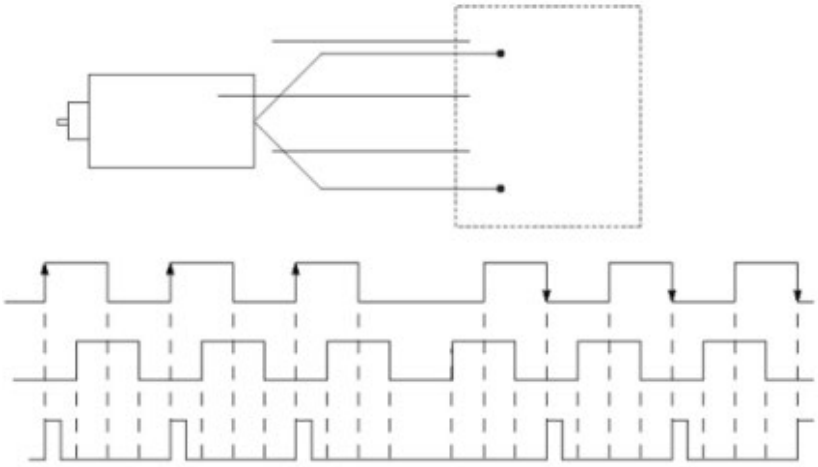
Dieses Beispiel zeigt, wie ein Hochgeschwindigkeitszähler (High-Speed Counter, HSC) auf seite 290-Programm erstellt wird, das einen Quadratur-Encoder verwendet und über eine Funktion für programmierbare Endschalter (Programmable Limit Switch, PLS) verfügt.

In diesem Beispiel verwendeter Quadratur-Codierer

Das Hochgeschwindigkeitszähler-Programmbeispiel setzt einen HSC-Funktionsblock und einen Quadratur-Zähler mit den phasenverschobenen Eingängen A und B ein. Der Quadratur-Codierer bestimmt die Rotationsrichtung und die Position von rotierenden Maschinen, wie die der Drehbank. Der bidirektionale Zähler zählt die Rotation des Quadratur-Codierers.

Der folgende Quadratur-Codierer ist mit den Eingängen 0 und 1 verbunden. Die Zählrichtung wird durch die Phasenverschiebung zwischen A und B bestimmt.

- Wenn A vor B läuft, zählt der Zähler hoch.
- Wenn B vor A läuft, zählt der Zähler runter.



Erstellen eines Hochgeschwindigkeitszähler (HSC)-Programms

Um ein HSC-Programm zu erstellen, aufzubauen und zu testen und danach eine PLS-Funktion hinzuzufügen, führen Sie die folgenden Aufgaben durch.

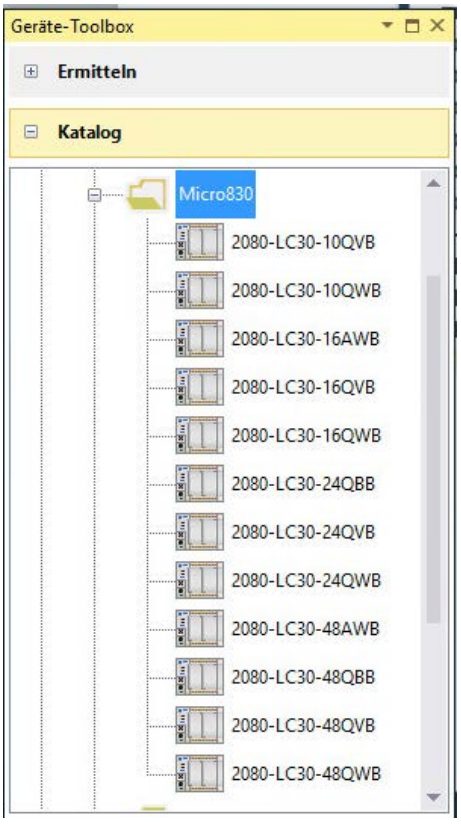
Schritt	Aufgabe
1	Erstellen Sie ein Kontaktplan-Diagramm und fügen Sie Variablen hinzu auf seite 316
2	Den HSC-Variablen Werte zuweisen auf seite 318
3	Zuweisen von Variablen und Erstellen des Programms auf seite 319
4	Testen des Programms und Ausführen des Hochgeschwindigkeitszählers auf seite 321
5	Hinzufügen einer Funktion für den programmierbaren Endschalter (PLS) auf seite 323

Erstellen Sie ein Kontaktplan-Diagramm und fügen Sie Variablen hinzu

Erstellen Sie ein Kontaktplan-Diagramm und fügen Sie anschließend lokale Variablen zum Strompfad hinzu. Dieses Beispielprogramm verwendet einen 2080-LC50-24QVB-Controller. Der HSC auf seite 290 wird auf allen Micro830- und Micro850-Controllern außer 2080-LCxx-xxAWB-Controllertypen unterstützt.

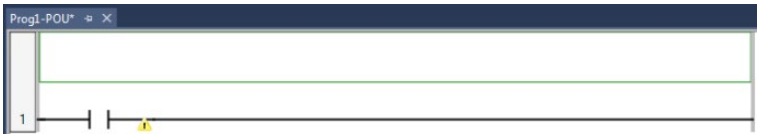
So erstellen Sie ein Kontaktplan-Diagramm und fügen Variablen hinzu:

1. Erweitern Sie in der **Geräte-Toolbox** die Registerkarte **Katalog**, um die Geräteordner anzuzeigen.
2. Erweitern Sie den Ordner „Controller“ und den Ordner „Micro830“, um alle Micro830-Controller anzuzeigen. Doppelklicken Sie auf einen Controller (2080-LC50-24QVB), um ihn zum **Projekt-Organizer** hinzuzufügen.

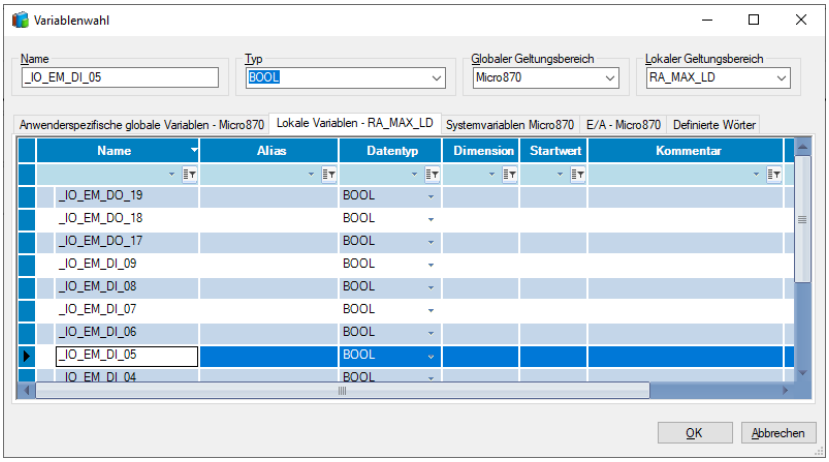


3. Klicken Sie mit der rechten Maustaste im **Projekt-Organizer** auf **Programme**, klicken Sie auf **Hinzufügen** und dann auf **Neuer KOP: Kontaktplan** zum Hinzufügen eines neuen Kontakt-Logikprogramms.
4. Klicken Sie mit der rechten Maustaste auf **UntitledLD** und wählen Sie **Öffnen**.
5. Im Dialogfeld **Toolbox**.
   - Doppelklicken Sie auf **Direkter Kontakt**, um ihn zum Kontaktplan hinzuzufügen oder

- ziehen Sie einen **Direkten Kontakt** per Drag und Drop in den Kontaktplan.

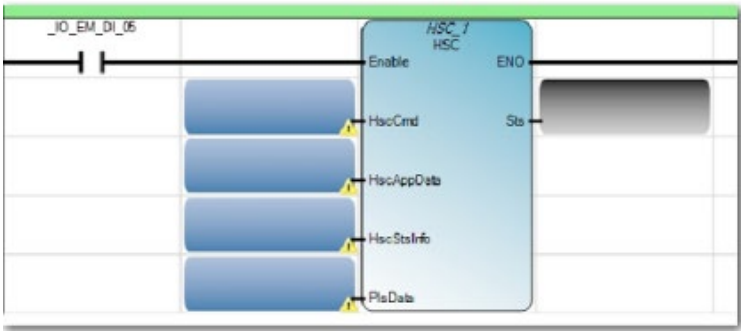


6. Weisen Sie dem direkten Kontakt eine Variable zu:
   - Doppelklicken Sie auf den direkten Kontakt, um den **Variablenselektor** anzuzeigen, und klicken Sie anschließend auf die Registerkarte **E/A - Micro830**.
   - Klicken Sie auf **_IO_EM_DI_05** und anschließend auf **OK**, um den direkten Kontakt Eingang 5 zuzuweisen.



7. Wählen Sie im Dialogfeld **Toolbox** einen Funktionsblock aus und ziehen Sie ihn rechts neben den direkten Kontakt, siehe folgendes Bild.

8. Doppelklicken Sie auf den Funktionsblock, um den **Anweisungsblockselektor** zu öffnen.
9. Wählen Sie im **Anweisungsblockselektor** die Option **HSC** aus, und klicken Sie anschließend auf **OK**.
10. Überprüfen Sie, ob der Kontaktplan wie in der folgenden Abbildung aussieht.

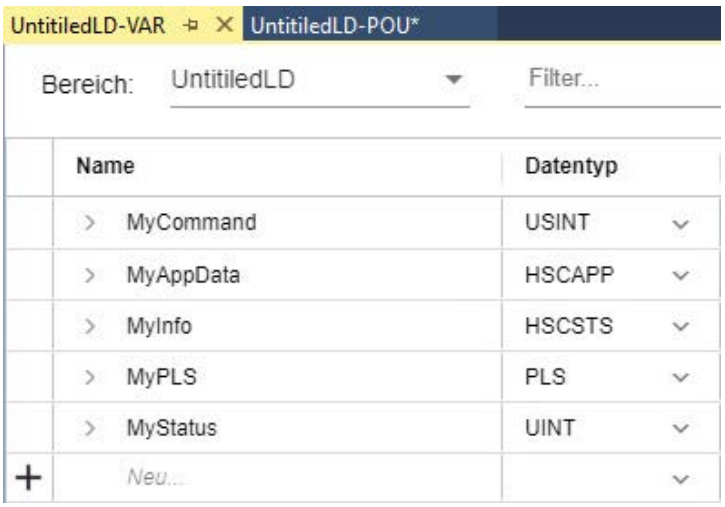


11. Doppelklicken Sie im **Projekt-Organizer** auf **Lokale Variablen**, um die Seite **Variablen** anzuzeigen.
12. Fügen Sie auf der Seite **Variablen** folgende Variablen und Datentypen hinzu.

Variablenname	Datentyp
MyCommand	USINT
MyAppData	HSCAPP
MyInfo	HSCSTS
MyPLS	PLS
MyStatus	UINT

Ergebnis

Die Seite **Variablen** sollte wie in der folgenden Abbildung aussehen:



Den HSC-Variablen Werte zuweisen

Befolgen Sie diese Schritte, nachdem Sie Variablen hinzugefügt haben, um den Variablen mithilfe der Spalte „Anfangswert“ im **Variablenselektor** Werte hinzuzufügen. Ein Standardprogramm verwendet normalerweise eine Routine, um den Variablen Werte zuzuweisen.

So weisen Sie den HSC-Variablen Werte zu:

1. Erweitern Sie **MyAppData**, um alle Variablen anzuzeigen.
2. Weisen Sie dem HSC-Modus einen Wert zu:
   - Geben Sie für die MyAppData.HSCMode-Variable 6 in das Feld **Anfangswert** ein.
   - Weitere Informationen zu jedem Wert finden Sie unter „HSCMode“ im Datentyp HSCAPP auf seite 292.
3. Weisen Sie den MyAppData-Variablen die verbliebenen zu, siehe folgende Abbildung.
   - Geben Sie den Wert in das Feld **Anfangswert** ein.

- Weitere Informationen zu jedem Wert finden Sie unter Datentyp HSCAPP auf seite 292.

Name	Alias	Datentyp	Dimension	Projektwert	Startwert
MyAppData		HSCAPP		...	...
MyAppData.PlsEnable		BOOL			FALSE
MyAppData.HscID		UINT			0
MyAppData.HscMode		UINT			6
MyAppData.Accumulator		DINT			
MyAppData.HPSetting		DINT			40
MyAppData.LPSetting		DINT			-40
MyAppData.OFSetting		DINT			50
MyAppData.UFSetting		DINT			-50
MyAppData.OutputMask		UDINT			3
MyAppData.HPOutput		UDINT			1
MyAppData.LPOutput		UDINT			2

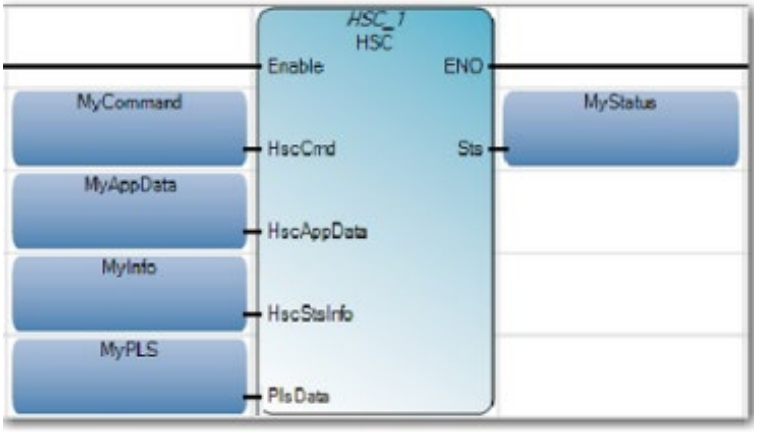
4. Weisen Sie dem HSC-Befehl einen Wert zu:
   - Geben Sie für die MyCommand-Variable 1 in das Feld **Anfangswert** ein.
   - Weitere Informationen zu Befehlswerten finden Sie unter HSCCmd-Werte auf seite 292.

Zuweisen von Variablen und Erstellen des Programms

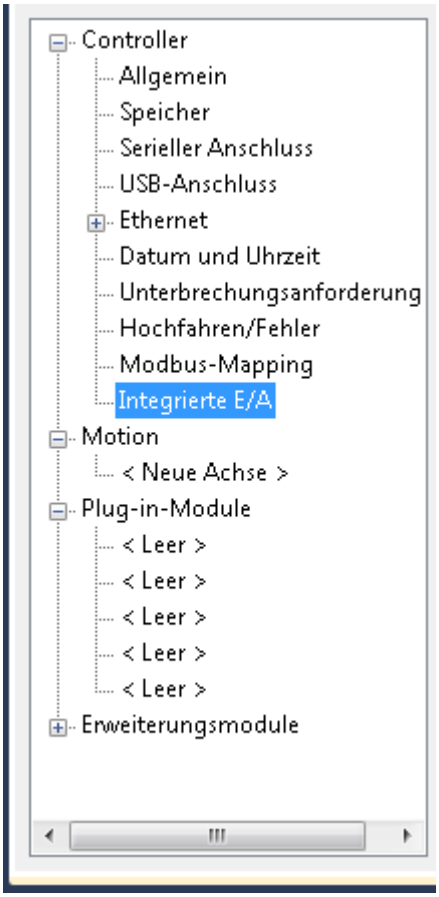
Nachdem Sie Werte in die HSC-Variablen eingegeben haben, befolgen Sie diese Schritte, um dem Funktionsblock Variablen zuzuweisen und das Programm zu erstellen.

So weisen Sie Variablen zu und erstellen das Programm:

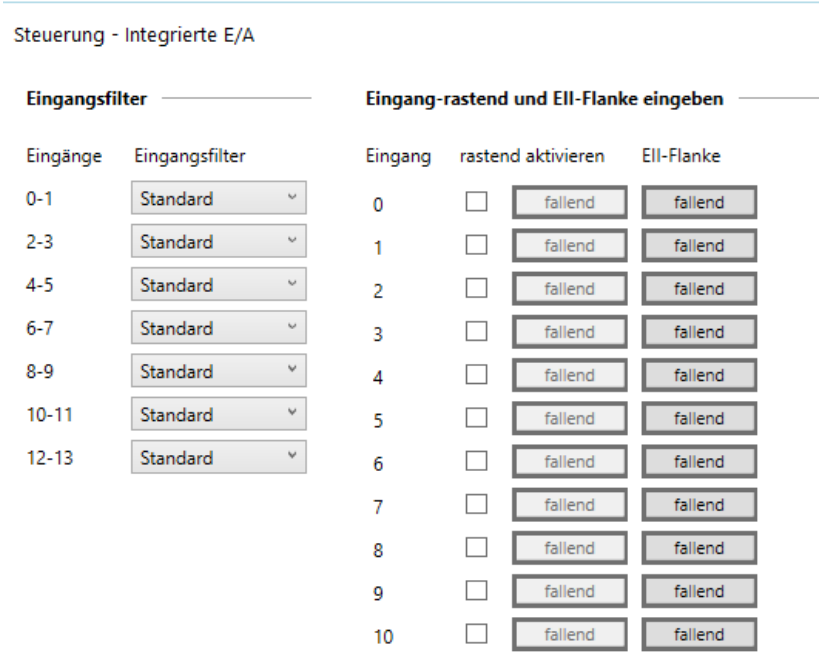
1. Weisen Sie im Kontaktplan-Editor jede Variable dem entsprechenden HSC-Funktionsblockelement zu, wie abgebildet.



2. Klicken Sie im **Projekt-Organizer** auf den Controller, um die **Controller**-Baumstruktur anzuzeigen.



3. Klicken Sie in der **Controller**-Baumstruktur auf **Integrierte E/A** und wählen Sie die Eingangsfilter für den Codierer.



4. Prüfen Sie, ob der Codierer mit dem Micro830-Controller verbunden ist.

5. Starten Sie den Micro830-Controller und verbinden Sie ihn mit Ihrem Computer.
6. Erstellen Sie das Programm und laden Sie es auf den Controller herunter.

Testen des Programms und Ausführen des Hochgeschwindigkeitszählers

Nachdem Sie das HSC-Programm auf den Controller heruntergeladen haben können Sie es testen und anschließend den Hochgeschwindigkeitszähler ausführen.

So testen Sie das Programm:

1. Stellen Sie eine Verbindung zum Controller her.
2. Doppelklicken Sie im **Projekt-Organizer** auf das HSC-Programm und anschließend auf **Lokale Variablen**.

   Die Werte der beiden HSC-Ausgänge werden angezeigt: STS (MyStatus) und HSCSTS (MyInfo).
3. Doppelklicken Sie auf den direkten Kontakt _IO_EM_DI_05, um das **Variablenselektor-**Fenster anzuzeigen.
4. Klicken Sie auf die Registerkarte **E/A Micro830** und anschließend auf die Zeile **_IO_EM_DI_05**.
5. Wählen Sie **Sperren** und dann **Logischer Wert**, um die **EIN**-Position des Eingangs zu erzwingen.

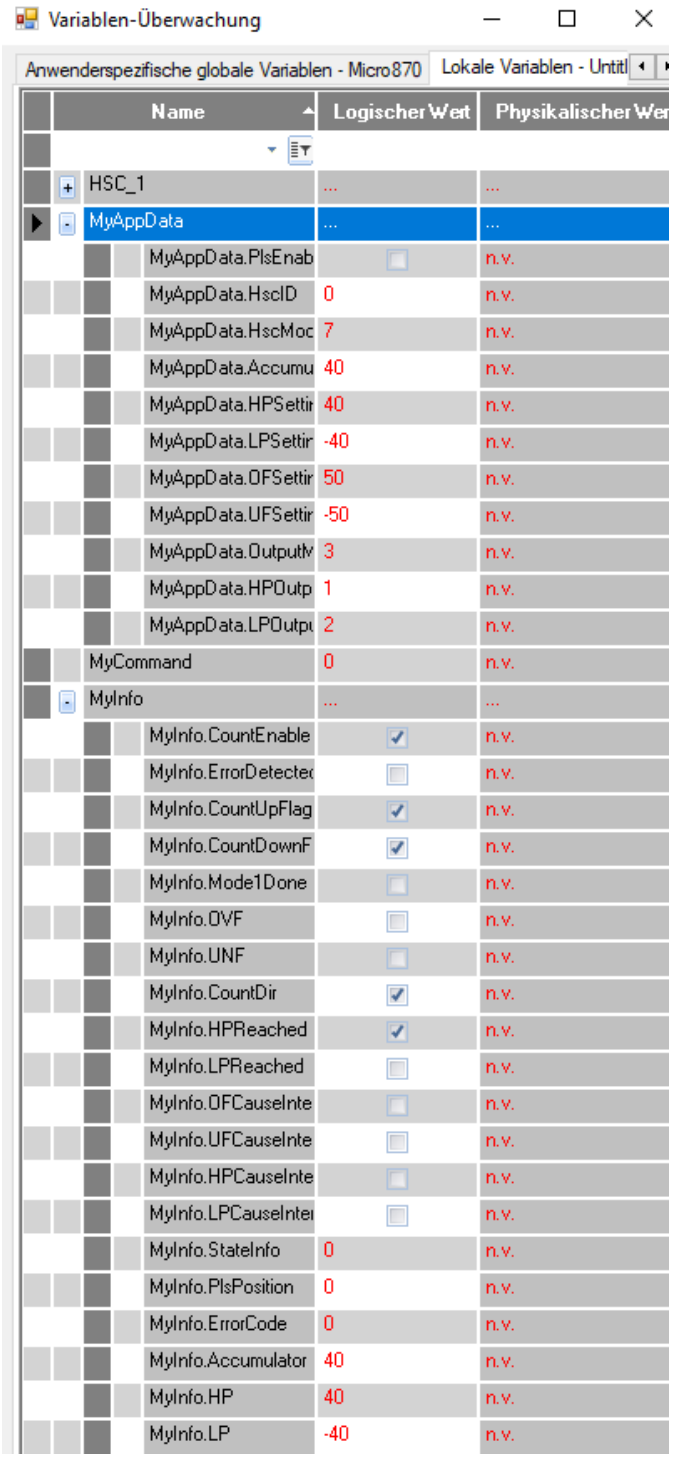
Name	Alias	Logischer Wert	Physikalischer Wert	Startwert	Sperren	Datentyp	Dimension
_IO_EM_DO_00						BOOL	
_IO_EM_DO_01						BOOL	
_IO_EM_DO_02						BOOL	
_IO_EM_DO_03						BOOL	
_IO_EM_DO_04						BOOL	
_IO_EM_DO_05						BOOL	
_IO_EM_DO_06						BOOL	
_IO_EM_DO_07						BOOL	
_IO_EM_DO_08						BOOL	
_IO_EM_DO_09						BOOL	
_IO_EM_DI_00						BOOL	
_IO_EM_DI_01						BOOL	
_IO_EM_DI_02						BOOL	
_IO_EM_DI_03						BOOL	
_IO_EM_DI_04						BOOL	
_IO_EM_DI_05						BOOL	
_IO_EM_DI_06						BOOL	
_IO_EM_DI_07						BOOL	
_IO_EM_DI_08						BOOL	
_IO_EM_DI_09						BOOL	

6. Um die Ergebnisse anzuzeigen, klicken Sie auf die Registerkarte **Lokale Variablen**, um Variablenänderungen anzuzeigen.
7. Erweitern Sie die **MyAppData**- und die **MyInfo**-Variablenliste.
8. Schalten Sie den Codierer ein, damit Sie sich ansehen können, wie der Zähler hoch/runter zählt. Wenn der Codierer beispielsweise an einer Antriebswelle befestigt ist, schalten Sie den Motor ein, um die HSC-Zählung einzuleiten.
9. Überprüfen Sie, ob der logische Wert der MyStatus-Variable 1 ist, was angibt, dass der HSC ausgeführt wird.
10. Der Zählerwert wird unter MyInfo.Accumulator angezeigt.

Unter Hochgeschwindigkeitszähler (high-speed counter, HSC) auf seite 290 finden Sie eine vollständige Liste der Statuscodes.

Ergebnisse

Hat MyInfo.Accumulator in diesem Beispiel eine obere Voreinstellung von 40 erreicht, schalten sich Ausgang 0 und das HPReached-Kennzeichen ein. Wenn MyInfo.Accumulator die untere Voreinstellung von -40 erreicht, schalten sich Ausgang 1 und das LPReached-Kennzeichen ein.



Add a Programmable Limit Switch (PLS) function

In diesem Beispiel wird erläutert, wie Sie dem HSC-Programm eine Funktion für den programmierbaren Endschalter (PLS) hinzufügen können.

Variablenwerte für die Zählereinstellungen:

- **MyAppData.PlsEnable** wird verwendet, um die PLS-Einstellungen zu aktivieren oder zu deaktivieren. Wenn die Variable MyAppData verwendet wird, sollte der Wert FALSE (deaktiviert) eingestellt werden.
- **MyAppData.HscID** wird verwendet um anzugeben, welche integrierten Eingänge auf Basis von Modus und Anwendungstyp verwendet werden. In den Abschnitten über HSC-Eingänge und Verdrahtungszuordnungen finden Sie Informationen zu den unterschiedlichen IDs, die verwendet werden können, sowie zu den integrierten Eingängen und deren Eigenschaften.
- Wenn ID 0 verwendet wird, kann ID 1 nicht auf demselben Controller verwendet werden, da die Eingänge von der Zurücksetz- und Haltefunktion verwendet werden.
- **MyAppData.HscMode** wird zur Angabe des Betriebsmodus verwendet, den HSC zum Zählen verwendet. Siehe HSC-Modus (HSCAPP.HSCMode) auf seite 292.

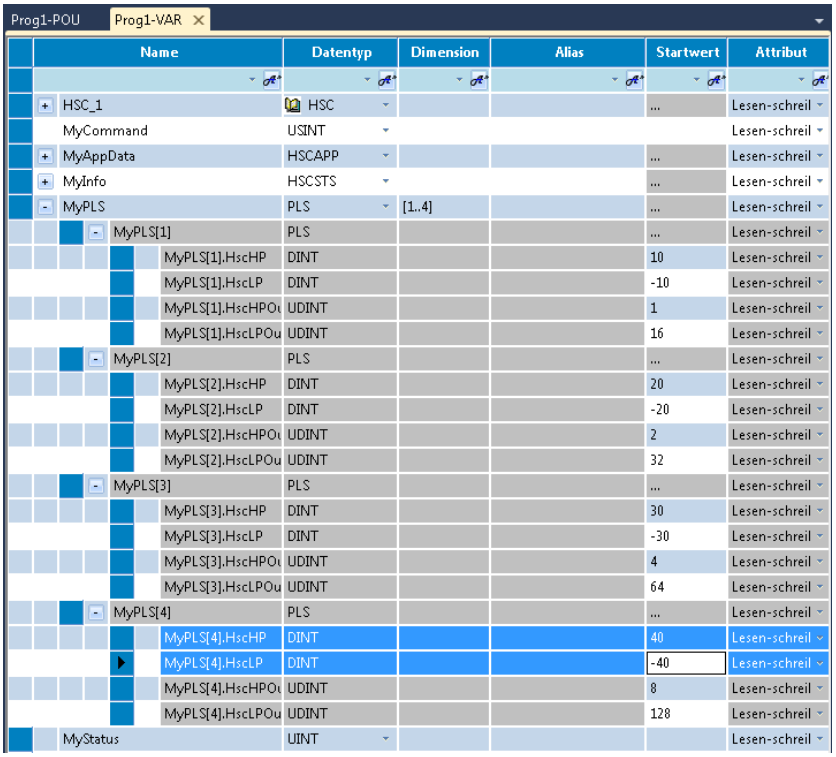
So aktivieren Sie PLS:

1. Doppelklicken Sie im **Projekt-Organizer** auf **Lokale Variablen**, um die Seite **Variablen** anzuzeigen.
2. So aktivieren Sie die PLS-Funktion:
   - Aktivieren Sie in dem Feld „Anfangswert“ für die MyAppData-Variable die Variable und wählen Sie TRUE.
3. So konfigurieren Sie die Unterlauf- und Überlaufeinstellungen:
   - Geben Sie in das Feld „Anfangswert“ für MyAppData.OFSetting 50 ein.
   - Geben Sie in das Feld „Anfangswert“ für MyAppData.UFSetting -50 ein.
4. (Optional) Konfigurieren Sie die Ausgangsmaske, falls ein Ausgang verwendet werden soll.

   Ergebnisse für dieses Beispiel:

   - Die PLS-Variable hat eine Größenordnung von [1..4]. Das heißt, der HSC kann vier Paare mit unterer und oberer Voreinstellung aufweisen.
   - Die obere Voreinstellung sollte niedriger sein als OFSetting und die untere Voreinstellung sollte größer als UFSetting sein.

- Die HscHPOutPut- und HscLPOutPut-Werte bestimmen, welche Ausgänge angeschaltet werden, wenn eine obere oder untere Voreinstellung erreicht wird.



Beispiel: Programmierbarer Endschalter (PLS) aktiviert

In diesem Beispiel werden die Ergebnisse beschrieben, wenn PLS mithilfe von spezifischen HSC auf seite 290- und PLSData-Parameterwerten aktiviert wurde.

HSC-Parameterwerte

Es wird angenommen, dass die HSC-Parameter auf die folgenden Werte festgelegt werden:

- HSCApp.OutputMask = 31
- HSCApp.HSCMode = 0
- HSC steuert integrierten Ausgang (nur 0-4)

PLSData-Parameterwerte

In diesem Beispiel wird angenommen, dass die PLSData-Parameter für die Variable (HSC_PLS) wie folgt konfiguriert sind.

Name	Datentyp	Dimension	Startwert
HSC_1	HSC		...
HSC_PLS	PLS	[1..4]	...
HSC_PLS[1]	PLS		...
HSC_PLS[1].HscHP	DINT		250
HSC_PLS[1].HscLP	DINT		-2
HSC_PLS[1].HscHPOutPut	UDINT		3
HSC_PLS[1].HscLPOutPut	UDINT		0
HSC_PLS[2]	PLS		...
HSC_PLS[2].HscHP	DINT		500
HSC_PLS[2].HscLP	DINT		-2
HSC_PLS[2].HscHPOutPut	UDINT		7
HSC_PLS[2].HscLPOutPut	UDINT		0
HSC_PLS[3]	PLS		...
HSC_PLS[3].HscHP	DINT		250
HSC_PLS[3].HscLP	DINT		-2
HSC_PLS[3].HscHPOutPut	UDINT		15
HSC_PLS[3].HscLPOutPut	UDINT		0
HSC_PLS[4]	PLS		...
HSC_PLS[4].HscHP	DINT		1000
HSC_PLS[4].HscLP	DINT		-2
HSC_PLS[4].HscHPOutPut	UDINT		31
HSC_PLS[4].HscLPOutPut	UDINT		0

Ergebnisse aktivierter PLS

Für dieses Beispiel treten folgende Ereignisse ein.

- Bei der ersten Ausführung der Kontaktplanlogik: HSCSTS.Accumulator = 1, alle Ausgaben sind deaktiviert.
- Wenn HSCSTS.Accumulator = 250, wird HSC_PLS[1].HSCHPOutput durch die HSCAPP.OutputMask gesendet, und die Ausgänge 0 und 1 werden mit Strom versorgt.
- Das Senden der Ausgabe der oberen Voreinstellung durch die Ausgabemaske wird wiederholt, wenn der HSCSTS.Accumulator die Werte 500, 750 und 1000 erreicht und der Controller die Ausgänge 0-2, 0-3 bzw. 0-4 mit Strom versorgt.
- Nach Abschluss des Vorgangs wird der Zyklus zurückgesetzt und wiederholt sich bei HSCSTS.HP = 250.
- Wenn der vollständige Zyklus abgeschlossen ist und der HSCSTS. HP-Wert erreicht ist, wird der PLSPosition-Parameter zurückgesetzt.

HSCE-Anweisungen

Verwenden Sie HSCE-Anweisungen, um den Hochgeschwindigkeitszähler zu überwachen und zu steuern.

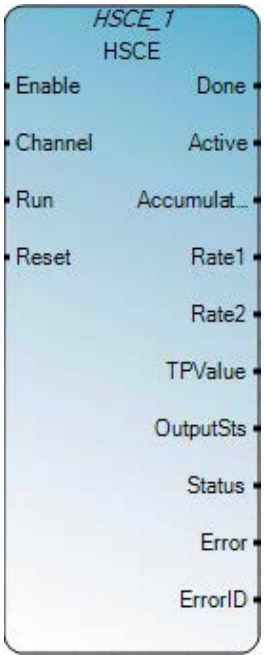
Anweisung	Beschreibung
HSCE auf seite 327	Akkumulatorwert starten, stoppen, lesen des HSCE.
HSCE_CFG auf seite 330	HSCE_CFG ist die Konfiguration für Hochgeschwindigkeitszähler.
HSCE_CFG_PLS auf seite 332	HSCE_CFG_PLS ist die PLS-Konfiguration für Hochgeschwindigkeitszähler.
HSCE_READ_STS auf seite 334	HSCE_READ_STS liest den Status des Hochgeschwindigkeitszählers aus.
HSCE_SET_STS auf seite 327	HSCE_SET_STS stellt/setzt den Status des Hochgeschwindigkeitszählers manuell ein/zurück

HSCE

HSCE dient zum Steuern und Auslesen des HSC-Zählers.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.

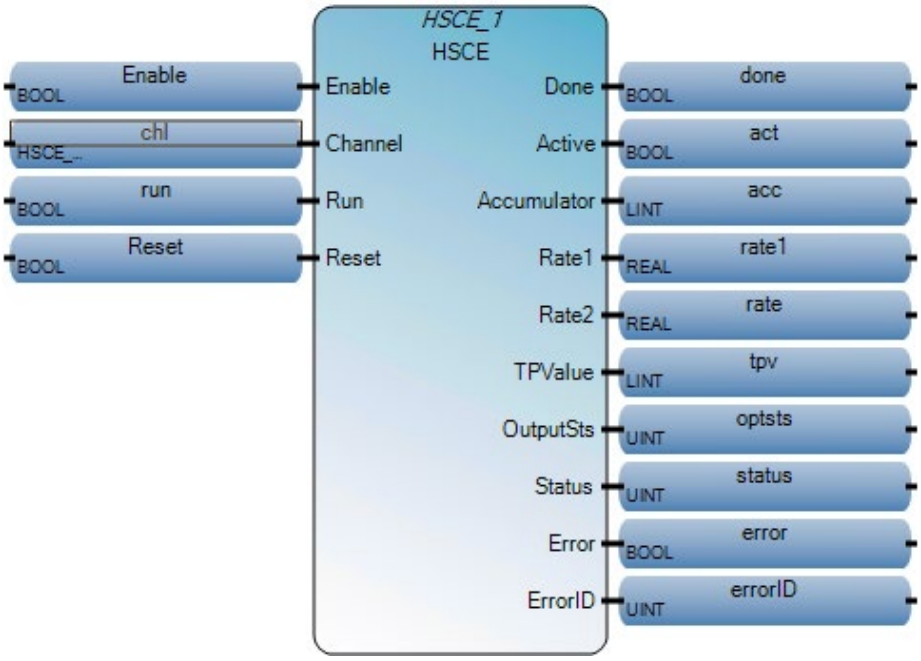


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

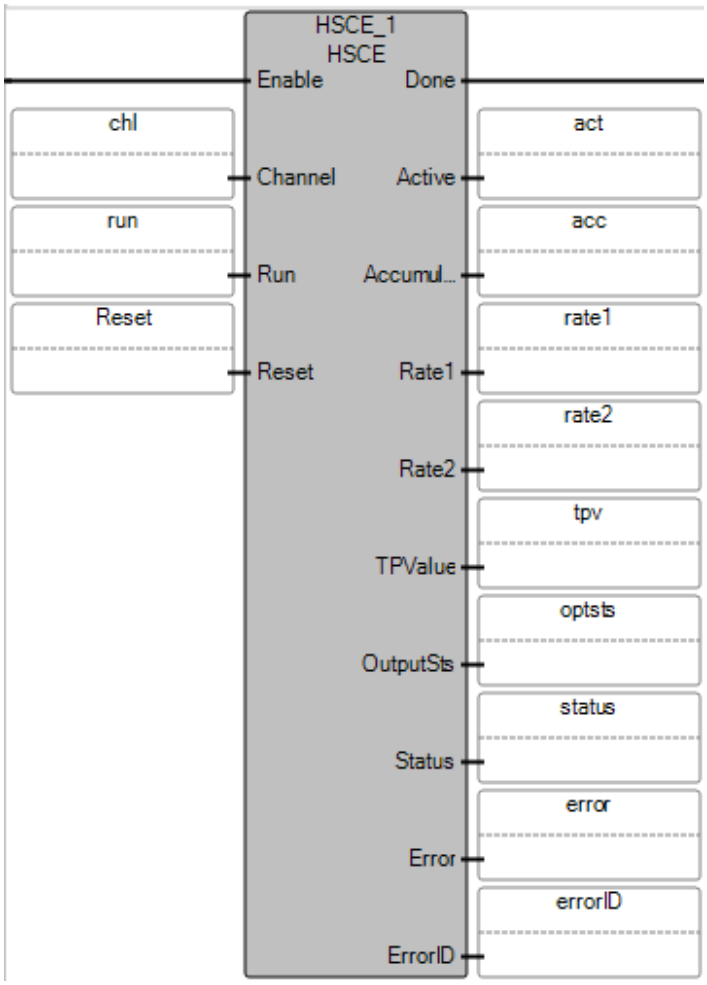
Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	TRUE - der HSCE initiiert den Funktionsblock und das HSC-Plug-In-Modul. Akkumulator = InitAcc. Status des HSCE wird zurückgesetzt. Rate1 = 0, Rate2 = 0.

Parameter	Parametertyp	Datentyp	Beschreibung
			FALSE - der Akkumulator wird durch Auslesen des Plug-In-Moduls aktualisiert. Fertig, Aktiv, Fehler = FALSE, wobei ErrorID = 0, Rate1, Rate2 = 0.
Channel	Eingang	HSCE_CHANNEL	Der HSCE-Kanal.
Run	Eingang	BOOL	Für HSCE zum Zählen des Betriebsstatus. TRUE - HSCE zählt die Pulse. FALSE - HSCE stoppt das Zählen.
Reset	Eingang	BOOL	True - alle Ausgänge werden gelöscht und Rate1 und Rate2 des Akkumulators werden ebenfalls auf 0 zurückgesetzt. HSC-Plug-In-Modulstatus löschen. Die Priorität des Reset-Eingangs ist höher als die des Eingangs des Laufs
Done	Ausgang	BOOL	True - wenn HSCE Enable True lautet und kein Fehler erkannt wurde False - wenn HSCE Enable True lautet, Lauf jedoch False lautet.
Active	Ausgang	BOOL	False - wenn HSCE Enable True lautet, Fertig True und Lauf False lauten. False - wenn HSCE Enable True lautet, Lauf jedoch False lautet.
Rate1	Ausgang	REAL	Aktuelle Pulsfrequenz in Verbrauchereinheit pro Sekunde (Pro-Impuls-Verfahren)
Rate2	Ausgang	REAL	Aktuelle Pulsfrequenz in Benutzereinheit pro Sekunde (Zyklisches Verfahren)
TPValue	Ausgang	LINT	Erfassung des Akkumulatorwerts, wenn der Messtaster ausgelöst wird.
OutputSts	Ausgang	UINT	Physikalischer und virtueller Ausgangsstatus des HSC-Plug-Ins gelten nur für das HSC-Zähler 0. Bit 0: Ausgang 0 (Physikalischer Ausgangsstatus) Bit 1 bis 15: Ausgang 1 bis Ausgang 15 (Virtueller Ausgangsstatus)
Status	Ausgang	UINT	HSC-Statusinformationen.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist.
ErrorID	Ausgang	UINT	Wenn ein Fehler auftritt, enthält die ErrorID den Fehlercode.

Beispiel für ein HSCE-Funktionsblockdiagramm



Beispiel für einen HSCE-Kontaktplan



Beispiele für eine HSCE-Textstruktur

```
HSCE_1(
```

void **HSCE_1**(BOOL **Enable**, HSCE_CHANNEL **Channel**, BOOL **Run**, BOOL **Reset**)
Typ: HSCE, Akkumulatorwert starten, stoppen und lesen

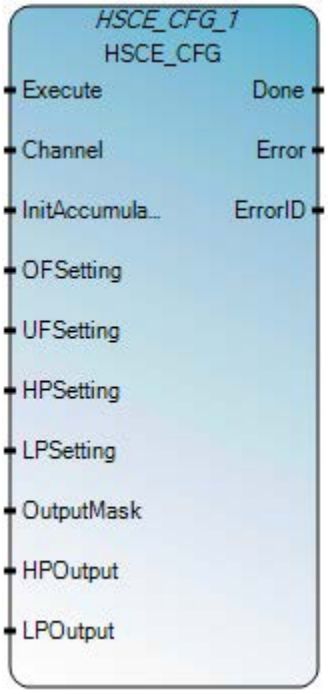
```
1  HSCE_1(Enable,chl,Run,Reset);
2  Done_HSCE := HSCE_1.Done;
3  Active_HSCE := HSCE_1.Active;
4  Accumulator_HSCE := HSCE_1.Accumulator;
5  Rate1_HSCE := HSCE_1.Rate1;
6  Rate2_HSCE := HSCE_1.Rate2;
7  TPValue_HSCE := HSCE_1.TPValue;
8  Outputsts_HSCE := HSCE_1.OutputSts;
9  Status_HSCE := HSCE_1.Status;
10 Error_HSCE := HSCE_1.Error;
11 ErrorID_HSCE := HSCE_1.ErrorID;
```

HSCE_CFG

HSCE_CFG dient zur Konfiguration für Hochgeschwindigkeitszähler.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

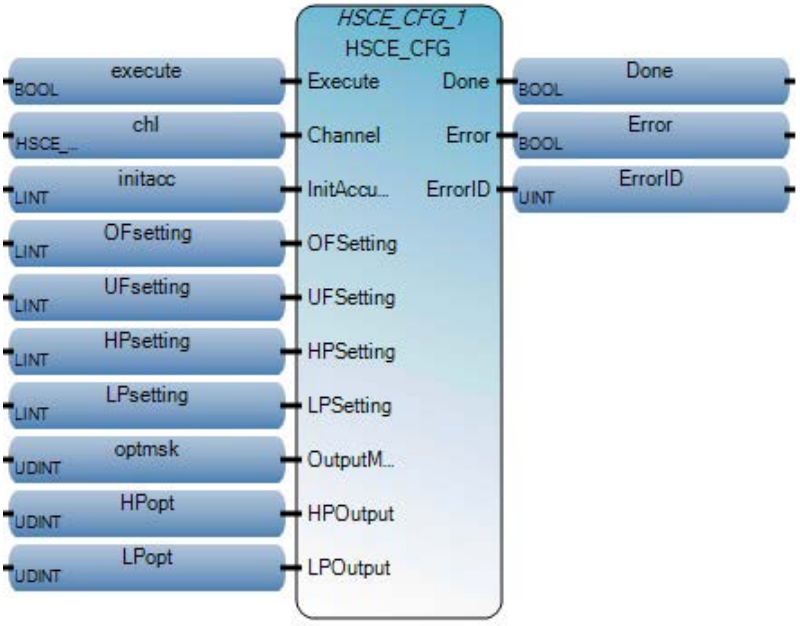
Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



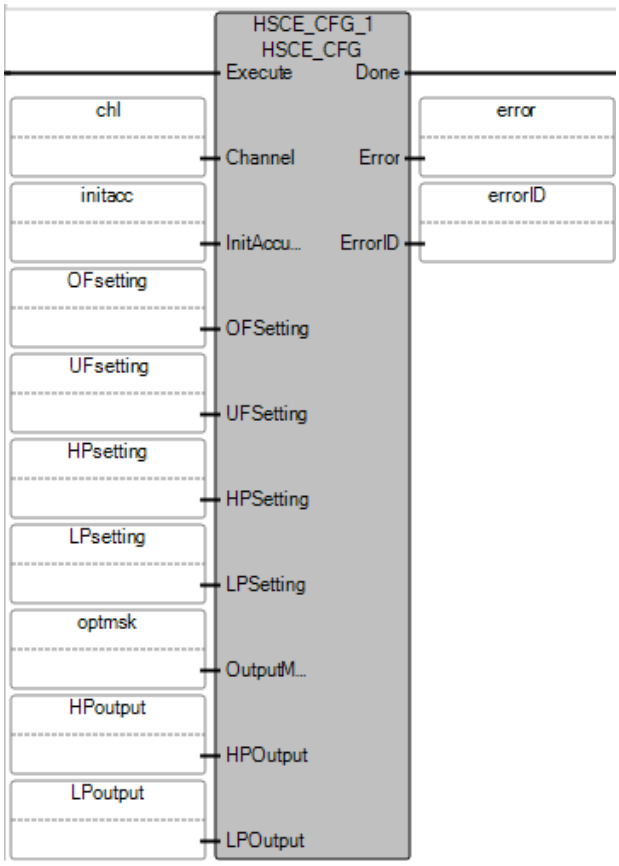
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Die steigende Flanke initiiert die HSC-Konfiguration. (HSCE Enable sollte FALSE entsprechen) Die Abstiegflanke löscht alle Ausgangswerte.
Channel	Eingang	HSCE_CHANNEL	Der HSCE-Kanal.
InitAccumulator	Eingang	LINT	Anfangswert des Akkumulators.
OFSetting	Eingang	LINT	Grenzwert des Zählerüberlaufs.
UFSetting	Eingang	LINT	Grenzwert des Zählerunterlaufs.
HPSetting	Eingang	LINT	Obere Voreinstellung (HP) des HSCE.
LPSetting	Eingang	LINT	Untere Voreinstellung (LP) des HSCE.
PLS_Offset	Eingang	USINT	Versatz zum Starten im PLS-Datenarray.
OutputMask	Eingang	USINT	Ausgabemaske für PLS-Funktionalität.
HPOutput	Eingang	UDINT	Status der Ausgänge mit oberer Voreinstellung.
LPOutput	Eingang	UDINT	Status der Ausgänge mit unterer Voreinstellung.
Done	Ausgang	BOOL	HSC-Konfigurationsaktion (initiiert durch diese Anweisung) wurde erfolgreich ausgeführt.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist.
ErrorID	Ausgang	UINT	Wenn ein Fehler auftritt, enthält die ErrorID den Fehlercode.

Beispiel für ein HSCE_CGF-Funktionsblockdiagramm



Beispiel für einen HSCE_CGF-Kontaktplan



Beispiele für eine HSCE_CGF-Textstruktur

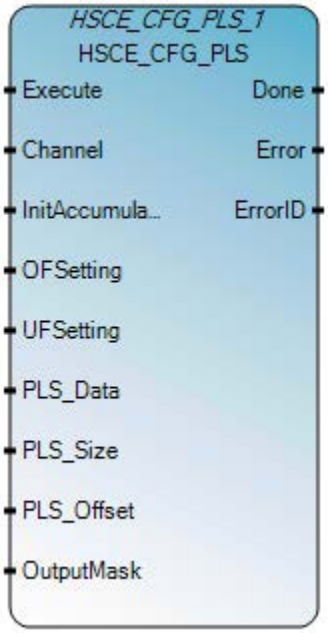
```
1  HSCE_CFG_1(execute,chl,initacc,OFsetting,UFsetting,HPsetting,LPsetting,optmsk,HPopt,LPopt);
2  Done_HSCE_CFG :=HSCE_CFG_1.Done;
3  Error_HSCE_CFG :=HSCE_CFG_1.Error;
4  ErrorID_HSCE_CFG := HSCE_CFG_1.ErrorID;
```

HSCE_CFG_PLS

Diese Anweisung dient zur HSC-Konfiguration mit einem programmierbaren Endschalter (PLS). Bei dieser Funktion handelt es sich um einen zusätzlichen Satz an Betriebsmodi für den Hochgeschwindigkeitszähler. Bei Betrieb in diesen Modi werden die voreingestellten Datenwerte wie auch die Ausgabe-Datenwerte bei jedem Erreichen einer der Voreinstellungen mittels benutzerseitig bereitgestellter Daten aktualisiert. Sie können diese Modi programmieren, indem Sie eine PLS-Datei mit den zu verwendenden Datensätzen bereitstellen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.

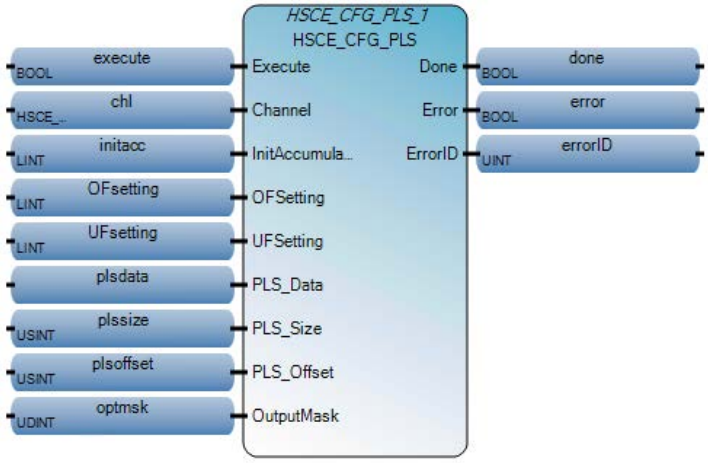


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

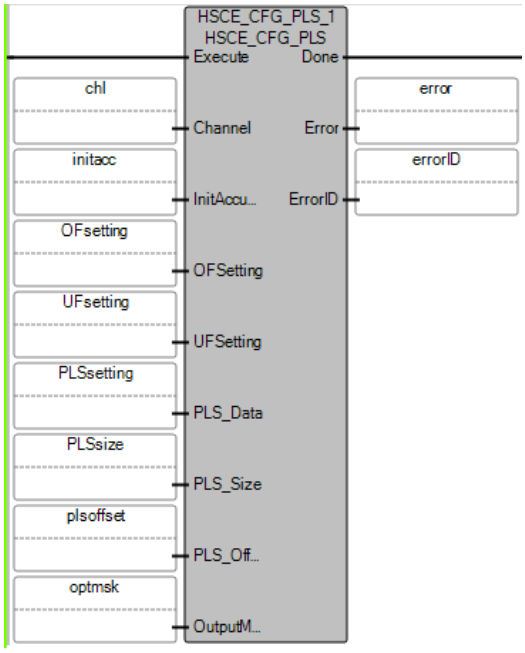
Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Die steigende Flanke initiiert die HSC-Konfiguration. (HSCE Enable sollte FALSE entsprechen) Die Abstiegflanke löscht alle Ausgangswerte.
Channel	Eingang	HSCE_CHANNEL	Der HSCE-Kanal.
InitAccumulator	Eingang	LINT	Anfangswert des Akkumulators.
OFSetting	Eingang	LINT	Grenzwert des Zählerüberlaufs.
UFSetting	Eingang	LINT	Grenzwert des Zählerunterlaufs.

Parameter	Parametertyp	Datentyp	Beschreibung
PLS_Data	Eingang	PLS2	Array des PLS (PLS_64)
PLS_Size	Eingang	USINT	PLS-Datengröße und der Höchstwert für das Plug-In lautet 24.
PLS_Offset	Eingang	USINT	Versatz zum Starten im PLS-Datenarray.
OutputMask	Eingang	USINT	Ausgabemaske für PLS-Funktionalität.
Done	Ausgang	BOOL	HSC-Konfigurationsaktion (initiiert durch diese Anweisung) wurde erfolgreich ausgeführt.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist.
ErrorID	Ausgang	UINT	Wenn ein Fehler auftritt, enthält die ErrorID den Fehlercode.

Beispiel für ein HSCE_CGF_PLS-Funktionsblockdiagramm



Beispiel für einen HSCE_CGF_PLS-Kontaktplan



Beispiele für eine HSCE_CGF_PLS-Textstruktur

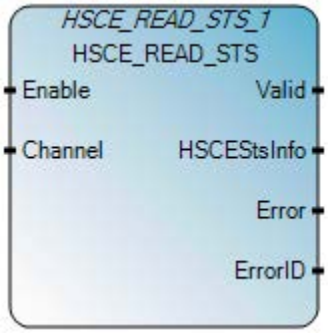
```
HSCE_CFG_PLS_1(Execute,chl,initacc,OFsetting,UFsetting,PLS_data,PLS_Size,PLS_Offset,optmsk);
Done_HSCE_CFG_PLS :=HSCE_CFG_PLS_1.Done;
Error_HSCE_CFG_PLS :=HSCE_CFG_PLS_1.Error;
ErrorID_HSCE_CFG_PLS := HSCE_CFG_PLS_1.ErrorID;
```

HSCE_READ_STS

Die Anweisung dient zum Auslesen des aktuellen HSC-Status.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

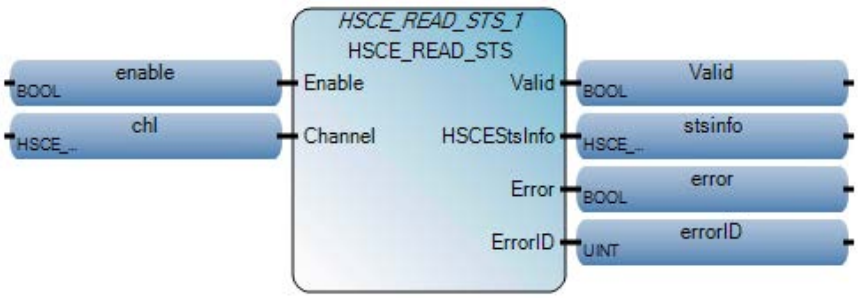
Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



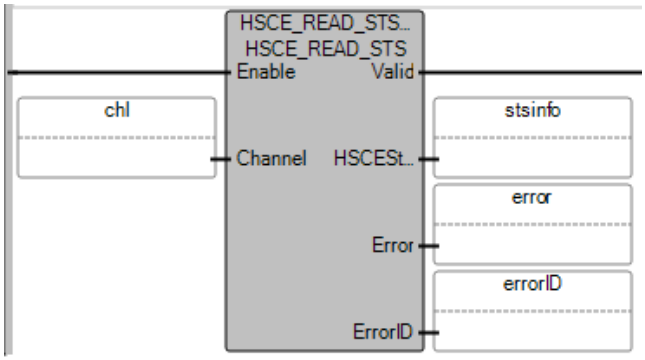
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Wenn die Option Enable True ist, wird HSC2StsInfo aktualisiert.
Channel	Eingang	HSCE_CHANNEL	Der HSCE-Kanal.
Valid	Ausgang	BOOL	HSC2StsInfo ist gültig bei TRUE.
HSCEStsInfo	Eingang	LINT	Grenzwert des Zählerüberlaufs.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist.
ErrorID	Ausgang	UINT	Wenn ein Fehler auftritt, enthält die ErrorID den Fehlercode.

Beispiel für ein HSCE_READ_STS-Funktionsblockdiagramm



Beispiel für einen HSCE_READ_STS-Kontaktplan



Beispiele für eine HSCE_READ_STS-Textstruktur

```
HSCE_READ_STS_1(
```

void **HSCE_READ_STS_1**(BOOL **Enable**, HSCE_CHANNEL **Channel**)
Typ: HSCE_READ_STS, Status des Hochgeschwindigkeitszählers lesen

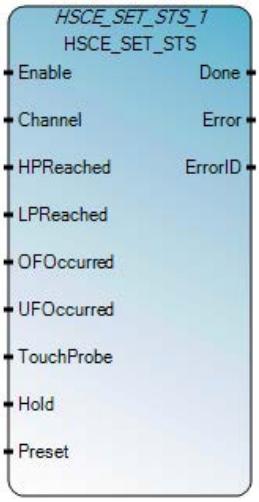
```
1  HSCE_READ_STS_1(Enable,Channel);
2  Valid_HSCE_READ_STS :=HSCE_READ_STS_1.Valid;
3  HSCEstsInfo_HSCE_READ_STS := HSCE_READ_STS_1.HSCEStsInfo;
4  Error_HSCE_READ_STS := HSCE_READ_STS_1.Error;
5  ErrorID HSCE READ STS := HSCE READ STS 1.ErrorID;
```

HSCE_SET_STS

Die Anweisung legt den HSC-Zählstatus manuell fest oder setzt ihn zurück. Der HSC-Funktionsblock muss für den HSC_SET_STC-Funktionsblock gestoppt werden, damit der HTS-Status festgelegt oder zurückgesetzt werden kann. Wenn der HSC-Funktionsblock nicht gestoppt wird oder der HSC-Kanal nicht konfiguriert ist , verursacht der HSC2_SET_STS-Funktionsblock einen Fehler.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

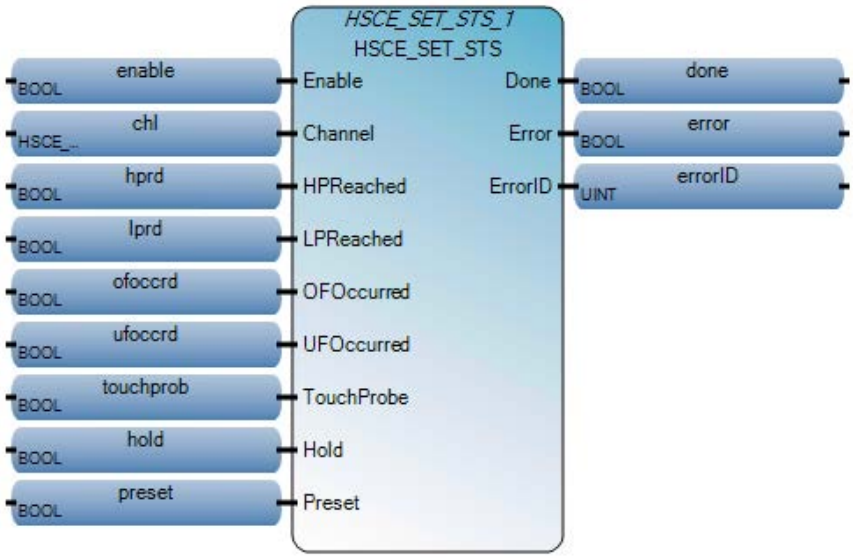
Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



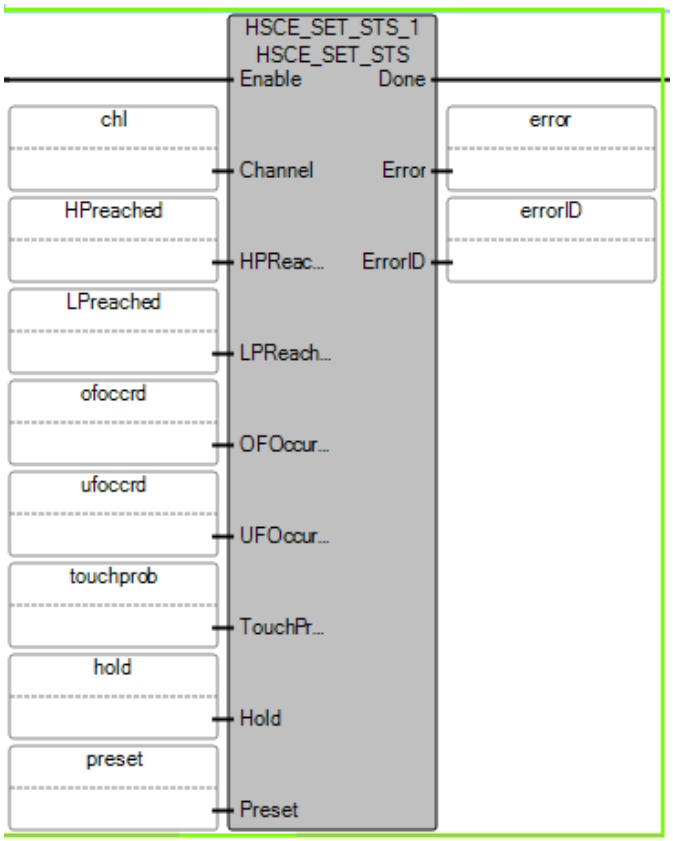
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Funktionsblock aktivieren. TRUE – Festlegen/Zurücksetzen des HSC-Status. FALSE – es ist keine HSC-Statusänderung vorhanden.
Channel	Eingang	HSCE_CHANNEL	Der HSCE-Kanal.
HPReached	Ausgang	BOOL	Wenn eine obere Voreinstellung erreicht wurde, wird dieses Bit durch das Plug-in-Modul auf TRUE gesetzt. Legen Sie dieses Bit mit dem HSCE_SET_STS-Funktionsblock fest oder setzen Sie es damit zurück.
LPReached	Eingang	BOOL	Wenn eine untere Voreinstellung erreicht wurde, wird dieses Bit durch das Plug-in-Modul auf TRUE gesetzt. Legen Sie dieses Bit mit dem HSCE_SET_STS-Funktionsblock fest oder setzen Sie es damit zurück.
OFOccurred	Eingang	BOOL	Wenn ein Überlauf auftritt, wird dieses Bit durch das Plug-in-Modul auf TRUE gesetzt. Legen Sie dieses Bit mit dem HSCE_SET_STS-Funktionsblock fest oder setzen Sie es damit zurück.
UFOccurred	Eingang	BOOL	Wenn ein Unterlauf aufgetreten ist, wird dieses Bit durch das Plug-in-Modul auf TRUE gesetzt. Legen Sie dieses Bit mit dem HSCE_SET_STS-Funktionsblock fest oder setzen Sie es damit zurück.
TouchProbe	Eingang	BOOL	Wenn ein Unterlauf aufgetreten ist, wird dieses Bit durch das Plug-in-Modul auf TRUE gesetzt.
Hold	Eingang	BOOL	Wenn ein HSC-Halten auftritt, wird dieses Bit durch das Plug-in-Modul auf TRUE gesetzt. Legen Sie dieses Bit mit dem HSCE_SET_STS-Funktionsblock fest oder setzen Sie es damit zurück. Dieser Eingabeparameter ist nur wirksam für Zähler 0. Für Zähler 1 wird er immer zurückgesetzt
Preset	Eingang	BOOL	Wenn ein Z ACC-Reset ausgelöst wird, wird dieses Bit durch das Plug-in-Modul auf TRUE gesetzt. Legen Sie dieses Bit mit dem HSCE_SET_STS-Funktionsblock fest oder setzen Sie es damit zurück. Dieser Eingabeparameter ist nur wirksam für Zähler 0. Für Zähler 1 wird er immer zurückgesetzt.
Done	Ausgang	BOOL	HSC-Konfigurationsaktion (initiiert durch diese Anweisung) wurde erfolgreich ausgeführt.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist.
ErrorID	Ausgang	UINT	Wenn ein Fehler auftritt, enthält die ErrorID den Fehlercode.

Beispiel für ein HSCE_SET_STS-Funktionsblockdiagramm



Beispiel für einen HSCE_SET_STS-Kontaktplan



Beispiele für eine HSCE_SET_STS-Textstruktur

```
1  HSCE_SET_STS_1(Enable, chl, hprd, ofocccrd, ufoccrd, touchprob, hold, preset);
2  Done_HSCE_SET_STS := HSCE_SET_STS_1.Done;
3  Error_HSCE_SET_STS := HSCE_SET_STS_1.Error;
4  ErrorID_HSCE_SET_STS := HSCE_SET_STS_1.ErrorID;
```

HSCE-Fehlercodes

In der folgenden Tabelle werden die Statusfehlercodes für HSCE-Befehle auf seite 327 beschrieben:

ErrorID-Code	Fehlerbeschreibung	Maßnahme
0	Die Anweisung wurde erfolgreich abgeschlossen.	
1	Ungültige HSC-Konfigurationsdateien.	Wenden Sie sich an den technischen Support von Rockwell Automation. Kontaktinformationen erhalten Sie hier: http://www.rockwellautomation.com/support
2	Ungültiger HSC-Modultyp.	Korrigieren Sie den Modultyp. Wählen Sie beispielsweise „Plug-in“ als Modultyp aus.
3	Ungültige HSC-Schacht-ID.	Korrigieren Sie die Schacht-ID im Kanaleingang des Funktionsblocks.
4	Ungültige HSC-ID.	Korrigieren Sie die HSC-ID. Stellen Sie beispielsweise „0“ für den Zähler 0 ein.
5	Ungültiger Modus für den Kanal.	Wenden Sie sich an den technischen Support von Rockwell Automation.
6	Ungültige PLS-Größe.	Die PLS-Größe sollte ≤ 24 und die Größe des PLS-Datenfelds ≥ der PLS-Größe sein.
7	Ungültiger PLS-Offset.	Die Summe von PLS-Offset und PLS-Größe sollte die Größe des PLS-Datenfelds nicht überschreiten.
8	Ungültiger Wert für InitAccumulator.	Korrigieren Sie den Wert für InitAccumulator. Möglicherweise befindet er sich außerhalb des Bereichs der LP- und HP-Grenzwerte (LP ≤ InitACC ≤ HP).
9	Ungültiger Wert für LP.	Korrigieren Sie den Wert für LP. Möglicherweise befindet er sich außerhalb des Bereichs.
10	Ungültiger Wert für OF.	Korrigieren Sie den Wert für OF.
11	Ungültiger Wert für UF.	Korrigieren Sie den Wert für UF.
12	Ungültiger Wert für HP.	Korrigieren Sie den Wert für HP.
13	Es ist keine Konfiguration für HSCE vorhanden.	Wenden Sie sich an den technischen Support von Rockwell Automation.
14	Ungültiger HSCE-Zustand.	Überprüfen Sie den Funktionsblock für HSCE auf seinen Zustand.
15	Ungültiges Plug-in-Modul	Überprüfen Sie das Plug-in-Modul, um sicherzustellen, dass es sich um ein HSC-Modul handelt.
16	HSCE wird ausgeführt.	Wenn eine HSC-Zählung durchgeführt wird, ist weder eine Konfiguration noch eine Einstellung des Status möglich. HSCE_CFG auf seite 330 und HSCE_CFG_PLS auf seite 332 werden ausgeführt, während HSCE läuft.
17	Fehler beim Anhalten des HSC-Plug-ins.	Wenden Sie sich an den technischen Support von Rockwell Automation.
18	Fehler beim Schreiben der Aktualisierungsrate.	Wenden Sie sich an den technischen Support von Rockwell Automation.
19	Fehler beim Schreiben der Plus-Anzahl.	Wenden Sie sich an den technischen Support von Rockwell Automation.
20	Fehler beim Schreiben der Anwendung des ACC.	Wenden Sie sich an den technischen Support von Rockwell Automation.
21	Fehler beim Schreiben des ACC.	Wenden Sie sich an den technischen Support von Rockwell Automation.
22	Fehler beim Schreiben der Anwendung.	Wenden Sie sich an den technischen Support von Rockwell Automation.
23	Fehler beim Schreiben der Anzahl an PLS.	Wenden Sie sich an den technischen Support von Rockwell Automation.
24	Fehler beim Schreiben des Unterlaufs.	Wenden Sie sich an den technischen Support von Rockwell Automation.
25	Fehler beim Schreiben des Überlaufs.	Wenden Sie sich an den technischen Support von Rockwell Automation.
26	Fehler beim Schreiben der niedrigen Voreinstellung.	Wenden Sie sich an den technischen Support von Rockwell Automation.
27	Fehler beim Schreiben der hohen Voreinstellung.	Wenden Sie sich an den technischen Support von Rockwell Automation.
28	Fehler beim Schreiben der niedrigen Voreinstellung für den Ausgang.	Wenden Sie sich an den technischen Support von Rockwell Automation.

ErrorID-Code	Fehlerbeschreibung	Maßnahme
29	Fehler beim Schreiben der hohen Voreinstellung für den Ausgang.	Wenden Sie sich an den technischen Support von Rockwell Automation.
30	Fehler beim Schreiben der Ausgabemaske.	Wenden Sie sich an den technischen Support von Rockwell Automation.
31	Fehler beim Schreiben der niedrigen Voreinstellung für den Ausgang des PLS.	Wenden Sie sich an den technischen Support von Rockwell Automation.
32	Fehler beim Schreiben der hohen Voreinstellung für den Ausgang des PLS.	Wenden Sie sich an den technischen Support von Rockwell Automation.
33	Fehler beim Schreiben der niedrigen Voreinstellung des PLS.	Wenden Sie sich an den technischen Support von Rockwell Automation.
34	Fehler beim Schreiben der hohen Voreinstellung für den Ausgang des PLS.	Wenden Sie sich an den technischen Support von Rockwell Automation.
35	Fehler beim Schreiben des PLS-Offsets.	Wenden Sie sich an den technischen Support von Rockwell Automation.
36	Fehler beim Schreiben der PLS-Anzahl.	Wenden Sie sich an den technischen Support von Rockwell Automation.
37	Fehler beim Lesen des Status.	Wenden Sie sich an den technischen Support von Rockwell Automation.
38	Fehler beim Lesen der hohen Voreinstellung.	Wenden Sie sich an den technischen Support von Rockwell Automation.
39	Fehler beim Lesen der niedrigen Voreinstellung.	Wenden Sie sich an den technischen Support von Rockwell Automation.
40	Fehler beim Lesen des ACC.	Wenden Sie sich an den technischen Support von Rockwell Automation.
41	Fehler beim Lesen des Zahlenwerts der Impulsbreite.	Wenden Sie sich an den technischen Support von Rockwell Automation.
42	Fehler beim Lesen der Impulsbreite.	Wenden Sie sich an den technischen Support von Rockwell Automation.
43	Fehler beim Lesen der Anzahl der Impulse.	Wenden Sie sich an den technischen Support von Rockwell Automation.
44	Fehler beim Lesen der Aktualisierungsrate.	Wenden Sie sich an den technischen Support von Rockwell Automation.
45	Fehler beim Schreiben des Status.	Wenden Sie sich an den technischen Support von Rockwell Automation.
46	Fehler beim Lesen der niedrigen Voreinstellung für den Ausgang.	Wenden Sie sich an den technischen Support von Rockwell Automation.
47	Fehler beim Lesen der hohen Voreinstellung für den Ausgang.	Wenden Sie sich an den technischen Support von Rockwell Automation.
48	Fehler beim Lesen der PLS-Anzahl.	Wenden Sie sich an den technischen Support von Rockwell Automation.
49	n. z.	Wenden Sie sich an den technischen Support von Rockwell Automation.
50	Fehler beim Schreiben des Anwendungsstatus.	Wenden Sie sich an den technischen Support von Rockwell Automation.
51	Ungültiger Kanaleingang.	Wenden Sie sich an den technischen Support von Rockwell Automation.
52	Fehler beim Lesen des Messtasters	Wenden Sie sich an den technischen Support von Rockwell Automation.
53	Fehler beim Schreiben des Zurücksetzens des ACC.	Wenden Sie sich an den technischen Support von Rockwell Automation.
54	Fehler beim Start des HSC.	Wenden Sie sich an den technischen Support von Rockwell Automation.
55	Zähler ist deaktiviert.	Der HSC-Zähler ist deaktiviert. Finden Sie durch Prüfen der Kanalkonfiguration heraus, ob der HSC-Zähler aktiviert ist.
56	Ungültiger Wert der Ausgabemaske.	Überprüfen Sie, ob sich die Ausgabemaske im gültigen Bereich befindet. Für das Plug-in-HSC-Modul liegt der Bereich zwischen 0 und 65535.
57	Ungültiger hoher voreingestellter Ausgangswert.	Überprüfen Sie, ob sich der HP-Wert für den Ausgang im gültigen Bereich befindet. Für das Plug-in-HSC-Modul liegt der Bereich zwischen 0 und 65535.
58	Ungültiger niedriger voreingestellter Ausgangswert.	Überprüfen Sie, ob sich der LP-Wert für den Ausgang im gültigen Bereich befindet. Für das Plug-in-HSC-Modul liegt der Bereich zwischen 0 und 65535.
59	Die UPM-Revision wird nicht unterstützt.	Überprüfen Sie die Revisionskonfiguration des HSC-Plug-in-Moduls.
60	Kein HSC-Modul hinzugefügt.	Überprüfen Sie, ob es sich bei dem verwendeten Plug-in-Modul um ein HSC-Modul handelt.
61	HSC-Plug-in-Modul ist nicht konfiguriert.	Wenden Sie sich an den technischen Support von Rockwell Automation.

ErrorID-Code	Fehlerbeschreibung	Maßnahme
62	Ein UPM-Schreibfehler tritt auf, wenn das HSC-Plug-in in den Ruhemodus versetzt wird.	Wenden Sie sich an den technischen Support von Rockwell Automation.
63	Ein UPM-Schreibfehler tritt auf, wenn das HSC-Plug-in in den Betriebsmodus versetzt wird.	Wenden Sie sich an den technischen Support von Rockwell Automation.
64	Ein UPM-Lesefehler tritt auf, wenn der Ausgansstatus gelesen wird.	Wenden Sie sich an den technischen Support von Rockwell Automation.

Eingabe-/Ausgabeanweisungen

Verwenden Sie Eingabe-/Ausgabeanweisungen zum Lesen oder Schreiben von Daten mithilfe von Signalen, die an ein Gerät gesendet werden, das physisch mit einem speicherprogrammierbaren logischen Controller verbunden ist, in oder aus einem Controller oder einem Modul. Eingangsrelais-Transfersignale an interne Relais und Ausgangsrelais übertragen Signale an externe Ausgabegeräte.

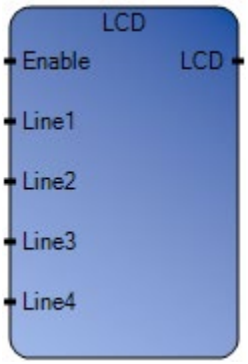
Anweisung	Beschreibung
LCD auf seite 341	Nur Micro810. Zeigt eine Zeichenfolge oder eine Zahl auf einem LCD-Modul an.
LCD_BKLT_REM auf seite 343	Legt die Parameter der Remote-LCD-Hintergrundbeleuchtung in einem Benutzerprogramm fest.
LCD_REM auf seite 345	Zeigt benutzerdefinierte Nachrichten für das Remote-LCD-Modul an.
RHC auf seite 348	Liest den Hochgeschwindigkeits-Taktgeberwert im Micro800-Controller.
RPC auf seite 350	Liest die Prüfsumme des Benutzerprogramms, entweder aus dem Controller oder dem Speichermodul.
DLG auf seite 351	Schreibt Variablenwerte aus der Laufzeit-Engine in eine Datenprotokollierungsdatei auf einer SD-Karte.
IIM auf seite 353	Aktualisiert die Eingaben vor normalem Ausgabescan.
IOM auf seite 355	Aktualisiert die Ausgaben vor normalem Ausgabescan.
KEY_READ auf seite 357	Nur Micro810. Liest den Tastenstatus auf dem optionalen LCD-Modul, wenn die Benutzeranzeige aktiv ist.
KEY_READ_REM auf seite 359	Nur Micro820. Liest den Tastenstatus auf dem optionalen Remote-LCD-Modul, wenn die Benutzeranzeige aktiv ist.
MM_INFO auf seite 361	Liest die Header-Informationen des Speichermoduls.
MODULE_INFO auf seite 364	Liest die Modulinformationen von einem Plug-in- oder Erweiterungsmoduls mit Ausnahme des 2080-MEMBAK-RTC-Speichermoduls.
PLUGIN_INFO auf seite 375	Liest die Modulinformationen von einem generischen Plug-in- oder Erweiterungsmodul mit Ausnahme des 2080-MEMBAK-RTC-Speichermoduls.
PLUGIN_READ auf seite 378	Liest die Daten von einem generischen Plug-in-Modul mit Ausnahme des 2080-MEMBAK-RTC-Speichermoduls.
PLUGIN_RESET auf seite 379	Setzt die Hardware von einem generischen Plug-in-Modul mit Ausnahme des 2080-MEMBAK-RTC-Speichermoduls zurück.
PLUGIN_WRITE auf seite 381	Schreibt Daten in ein generisches Plug-in-Modul mit Ausnahme des 2080-MEMBAK-RTC-Speichermoduls.
RCP auf seite 383	Liest/schreibt Rezeptdaten auf und von einer SD-Speicherkarte.
RTC_READ auf seite 385	Liest die Informationen des Echtzeituhr-Moduls (Real-Time Clock, RTC).
RTC_SET auf seite 387	Stellt die Daten der Echtzeituhr auf die RTC-Modulinformationen ein.
SYS_INFO auf seite 389	Liest den Statusdatenblock für den Controller Micro800.
TRIMPOT_READ auf seite 392	Liest den TrimPot-Wert aus einem bestimmten TrimPot.

LCD

Zeigt eine Zeichenfolge oder eine Zahl auf dem optionalen LCD-Modul an.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

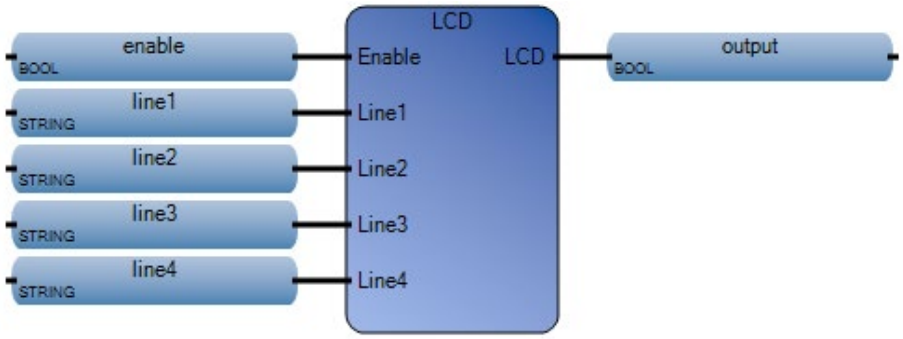
Diese Anweisung gilt für Micro810-Controller.



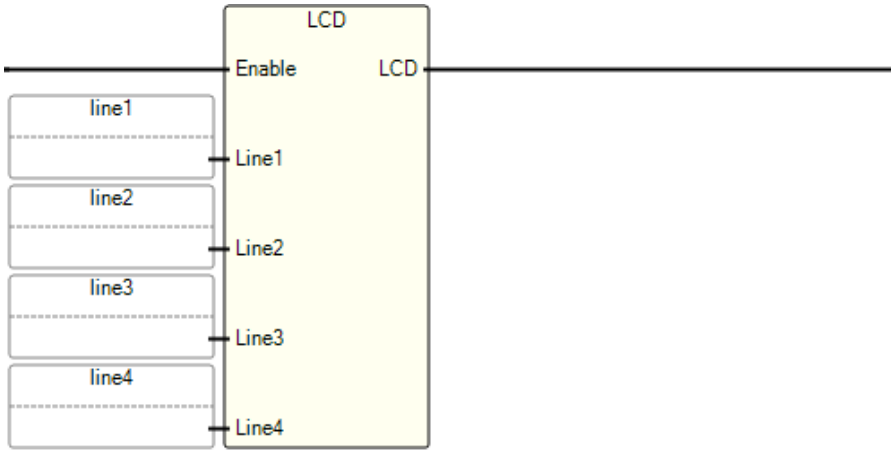
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisung aktivieren. TRUE – die LCD wechselt statt der Anzeige des E/A-Status zum benutzerdefinierten Bildschirm (Zeichenfolgen werden auf dem LCD-Modul angezeigt). FALSE – das LCD-Modul zeigt den Inhalt der E/A-Statusanzeige an.
Line1	Eingang	STRING	Zeichenfolge, die in Zeile 1 des LCD-Moduls angezeigt wird.
Line2	Eingang	STRING	Zeichenfolge, die in Zeile 2 des LCD-Moduls angezeigt wird.
Line3	Eingang	STRING	Zeichenfolge, die in Zeile 3 des LCD-Moduls angezeigt wird.
Line4	Eingang	STRING	Zeichenfolge, die in Zeile 4 des LCD-Moduls angezeigt wird.
LCD	Ausgang	BOOL	TRUE – Funktion ist aktiviert.

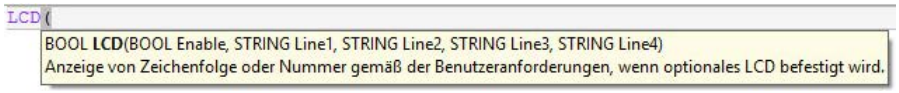
Diagrammbeispiel für Funktionsblock LCD



Beispiel für LCD-Kontaktplandiagramm



Beispiel für LCD in strukturiertem Text

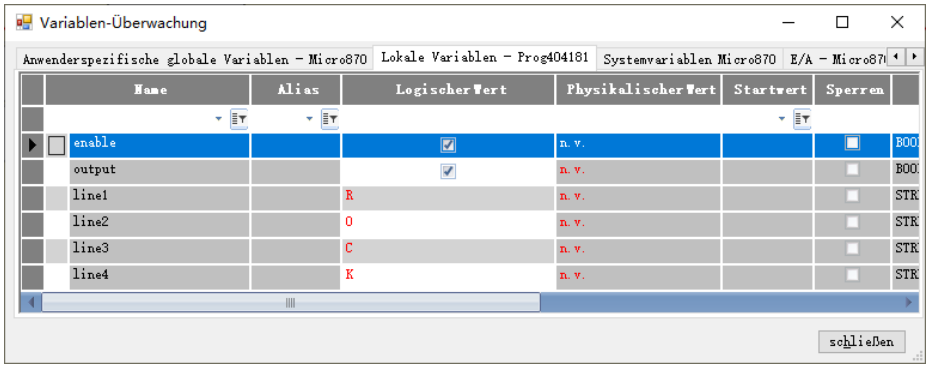



```
1  enable := TRUE;
2  line1 := 'R';
3  line2 := 'O';
4  line3 := 'C';
5  line4 := 'K';
6  output := LCD(enable, line1, line2, line3, line4);
```

(* ST-Äquivalenz: *)

TESTOUTPUT := LCD(LCDENABLE, LINE1, LINE2, LINE3, LINE4) ;

Ergebnisse



LCD_BKLT_REM (Remote-LCD-Hintergrundbeleuchtung)

Legt die Parameter der Remote-LCD-Hintergrundbeleuchtung in einem Benutzerprogramm fest.

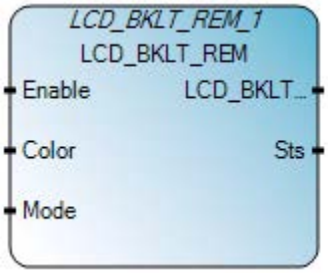
Details zum Betrieb:

Die in LCD_BKLT_REM definierten Hintergrundbeleuchtungseinstellungen werden verwendet, wenn das Remote-LCD-Display:

- eine benutzerdefinierte, mithilfe von LCD_REM definierte Anzeige ist.
- die standardmäßige E/A-Statusanzeige ist.
- Bei allen anderen Anzeigen werden die über das Menü auf dem Remote-LCD definierten Hintergrundbeleuchtungseinstellungen verwendet.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für Micro820-Controller.

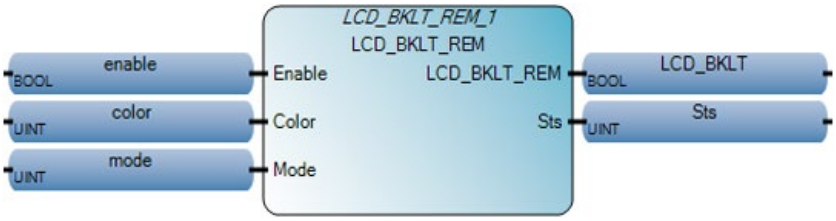


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

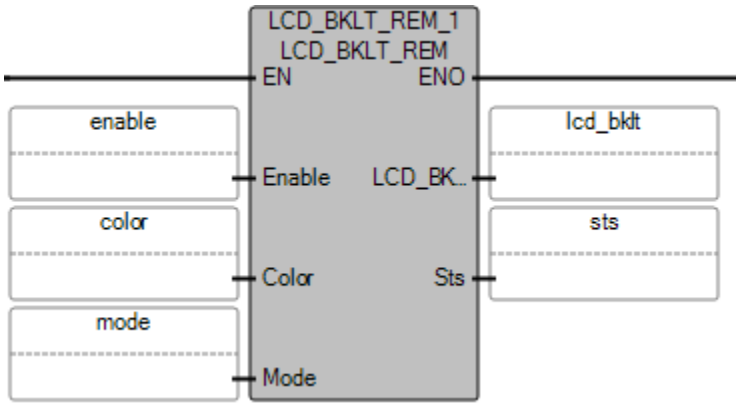
Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisung aktivieren. TRUE – REM_LCD_BKLT ausführen, Überschreiben aller aktuellen Hintergrundbeleuchtungseinstellungen. FALSE – REM_LCD_BKLT wird deaktiviert, und die Remote-LCD-Menüeinstellungen sind wirksam.
Color	Eingang	UINT	Farbcode der Hintergrundbeleuchtung • 0: Weiß • 1: Blau • 2: Rot • 3: Grün • 4-65535: Reserviert
Mode	Eingang	UINT	• 0: Dauerhaft AUS • 1: Dauerhaft EIN • 2: Blinken (Intervall von 1 s) • 3-65535: Reserviert
LCD_BKLT_REM	Ausgang	BOOL	TRUE – Anweisung erfolgreich ausgeführt. FALSE – Fehler bei Ausführung der Anweisung aufgetreten.
Sts	Ausgang	UINT	Status des Remote-LCD-Vorgangs. LCD_BKLT_REM-Statuscodes: • 0 – Aktivierungseingabe ist falsch. • 1 – erfolgreich. • 2 – Remote-LCD nicht erkannt. Kann auftreten wenn: • Remote-LCD ist nicht physisch am Controller angeschlossen oder die Verdrahtung ist fehlerhaft. • Einstellungen des seriellen Anschlusses sind anders als von der Remote-LCD gefordert.

Parameter	Parametertyp	Datentyp	Beschreibung
			• 3 – Verbindungsfehler. Kann auftreten, wenn es einen Fehler mit dem internen Status der Maschine gibt, z. B. Inkompatibilität zwischen Controller-FW-Version und RLCD-FW-Version. • 4 – Ungültiger Farbcode. • 5 – Ungültiger Modus. • 6-65535 – Reserviert.

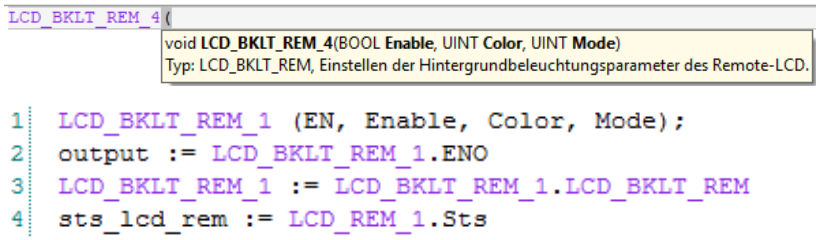
Diagrammbeispiel für Funktionsblock LCD_BKLT_REM



LCD_BKLT_REM-Kontaktplan-Beispiel



Beispiel für LCD_BKLT_REM in strukturiertem Text



LCD_REM (Remote-LCD)

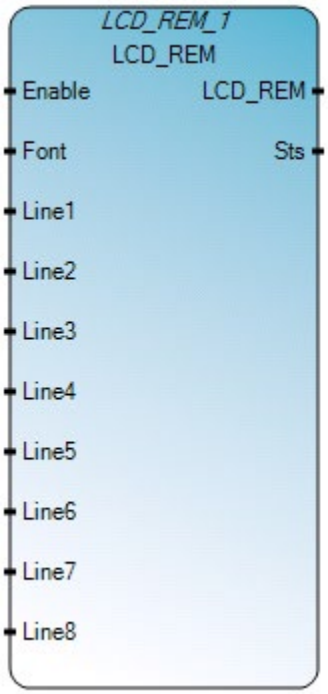
Zeigt benutzerdefinierte Nachrichten für das Remote-LCD-Modul an.

Details zum Betrieb:

- Für Zeile 1 bis Zeile 8 beträgt die maximale Länge der Zeichenfolge 24 Zeichen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für Micro820-Controller.

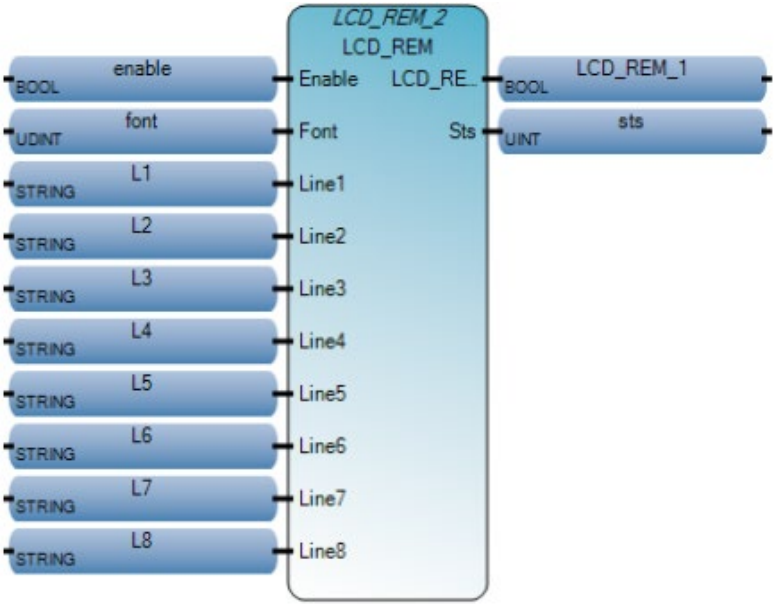


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

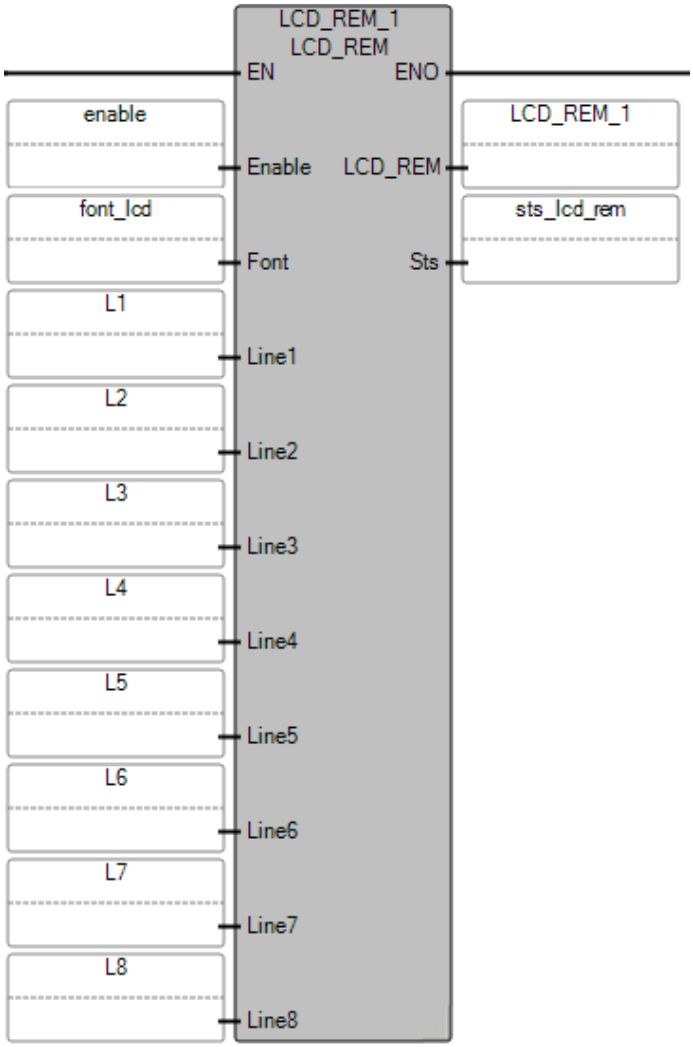
Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Aktivieren Sie den Anweisungsblock. TRUE – Remote-LCD schaltet von der E/A-Statusanzeige in die benutzerdefinierte Anzeige. FALSE – Remote-LCD schaltet zurück zur E/A-Statusanzeige.
Font	Eingang	UDINT	Schriftgröße für die Startmeldung: • 0: Standard (Groß – 8x16) • 1: Klein (8x8) • 2: Groß (8x16) • 3: Extragroß (16x16) • 4 und Folgende: Reserviert Die Remote-LCD-Größe ist 192x64 Pixel. Sobald die Schriftgröße für Startmeldung eine kleine Schriftart ist, zeigt das Remote-LCD Folgendes an: • Zeichenfolgen in Zeile 1 bis Zeile 8. • Maximal 24 Zeichen pro Zeile. Sobald die Schriftgröße für Startmeldung groß oder extra groß ist, zeigt das Remote-LCD Folgendes an: • Zeichenfolgen in Zeile 1 bis Zeile 4. • Zeichenfolgen in Zeile 5 bis Zeile 8 werden ignoriert. • Maximal 12 Zeichen pro Zeile.
Line1	Eingang	Zeichenkette	Zeichenfolge, die in Zeile 1 des LCD-Moduls angezeigt werden soll. Für Zeile 1 bis Zeile 8 beträgt die maximale Länge der Zeichenfolge 24 Zeichen.
Line2	Eingang	Zeichenkette	Zeichenfolge, die in Zeile 2 des LCD-Moduls angezeigt wird.

Parameter	Parametertyp	Datentyp	Beschreibung
Line3	Eingang	Zeichenkette	Zeichenfolge, die in Zeile 3 des LCD-Moduls angezeigt wird.
Line4	Eingang	Zeichenkette	Zeichenfolge, die in Zeile 4 des LCD-Moduls angezeigt wird.
Line5	Eingang	Zeichenkette	Zeichenfolge, die in Zeile 5 des LCD-Moduls angezeigt wird.
Line6	Eingang	Zeichenkette	Zeichenfolge, die in Zeile 6 des LCD-Moduls angezeigt wird.
Line7	Eingang	Zeichenkette	Zeichenfolge, die in Zeile 7 des LCD-Moduls angezeigt wird.
Line8	Eingang	Zeichenkette	Zeichenfolge, die in Zeile 8 des LCD-Moduls angezeigt wird.
LCD_REM	Ausgang	BOOL	Funktionsblock aktivieren. Wenn Enable = TRUE, Benutzeranzeige ist aktiv. Wenn Enable = FALSE, E/A-Status/Menü-Anzeige ist aktiv.
Sts	Ausgang	UINT	Status des Remote-LCD-Vorgangs. LCD_REM-Statuscodes: • 0 – Aktivierungseingabe ist falsch. • 1 – Benutzermeldung erfolgreich angezeigt. • 2 – Remote-LCD nicht erkannt. • 3 – Verbindungsfehler. Kann auftreten wenn: • Remote-LCD ist nicht physisch am Controller angeschlossen (oder die Verdrahtung ist fehlerhaft). • Einstellungen des seriellen Anschlusses sind anders als von der Remote-LCD gefordert. • 4 – Ungültiger Schriftartcode. • 5 – 5-65535 – Reserviert.

Diagrammbeispiel für Funktionsblock LCD_REM



LCD_REM-Kontaktplan-Beispiel



Beispiel für LCD_REM in strukturiertem Text

```
LCD_REM_1(
```

void **LCD_REM_1**(BOOL **Enable**, UDINT **Font**, STRING **Line1**, STRING **Line2**, STRING **Line3**, STRING **Line4**, STRING **Line5**, STRING **Line6**, STRING **Line7**, STRING **Line8**)
Typ: LCD_REM, Anzeige der Benutzerzeichenfolgen auf der Remote-LCD, wenn diese angeschlossen wird.

```
1  LCD_REM_1 (EN, Enable, Font, Line1, Line2, Line3, Line4, Line5, Line6, Line7, Line8);
2  output := LCD_REM_1.ENO
3  LCD_REM_1 := LCD_REM_1.LCD_REM
4  sts_lcd_rem := LCD_REM_1.Sts
```

RHC (Hochgeschwindigkeits-Taktgeber lesen)

Liest den Hochgeschwindigkeits-Taktgeberwert im Micro800-Controller.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

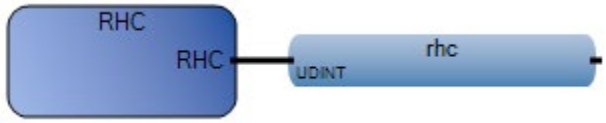
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Hochgeschwindigkeitstaktgeber lesen. FALSE – Keine Aktion. Gilt für Kontaktplan-Programme.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.
RHC	Ausgang	UDINT	Der Wert des Hochgeschwindigkeits-Taktgebers

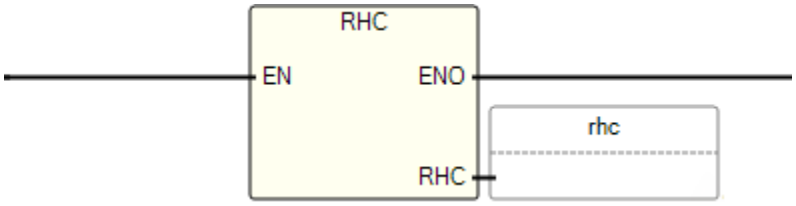
Auflösung des Hochgeschwindigkeits-Taktgebers

Controller-Typ	Inkremente	Timebase	Auflösung
Micro810	4 alle 40 Mikrosekunden	10 Mikrosekunden	40 Mikrosekunden
Micro820 Micro830 Micro850	1 alle 10 Mikrosekunden	10 Mikrosekunden	10 Mikrosekunden

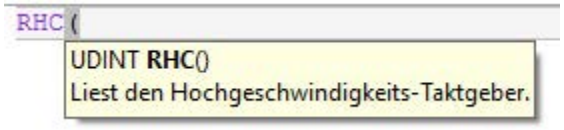
Beispieldiagramm mit einem RHS-Funktionsblock



Beispiel für RHS-Kontaktplandiagramm



Beispiel eines strukturierten RHS-Texts




```
1  rhc := RHC();
```

(* ST-Äquivalenz: *)

TESTOUTPUT2 := RHC() ;

RPC (Prüfsumme des Benutzerprogramms lesen)

Liest die Prüfsumme des Benutzerprogramms, entweder aus dem Controller oder dem Speichermodul.

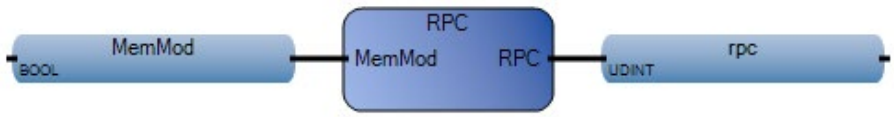
Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

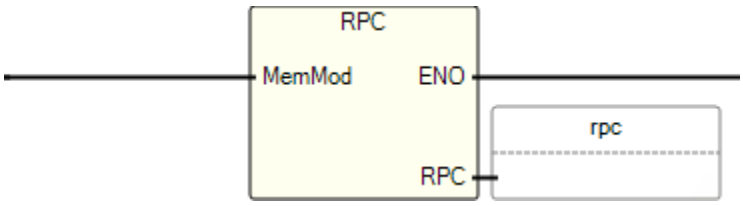
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
MemMod	Eingang	BOOL	TRUE – Wert wird aus dem Speichermodul übernommen. FALSE – Wert wird aus dem Micro800-Controller genommen.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.
RPC	Ausgang	UDINT	Der Prüfsummenwert des angegebenen Benutzerprogramms.

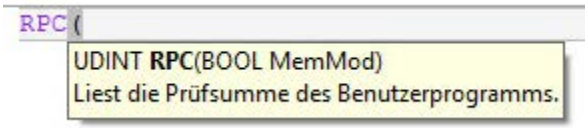
Beispiel für ein RPC-Funktionsblockdiagramm



Beispiel eines RPC-Kontaktplans



Beispiel eines strukturierten RPC-Texts




```
1  MemMod := TRUE;
2  rpc := RPC(MemMod);
```

(* ST-Äquivalenz: *)

TESTOUTPUT2 := RPC(TESTINPUT) ;

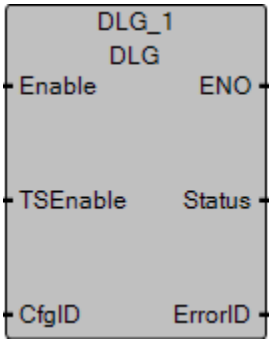
DLG (Datenerfassung)

Schreibt Variablenwerte aus der Laufzeit-Engine in eine Datenprotokollierungsdatei auf einer SD-Karte.

Beim Schreiben in eine Datenerfassung sind maximal 50 Gruppenordner pro Tag zulässig. Jeder Gruppenordner darf maximal 50 Dateien mit einer Dateigröße von 4k-8k haben.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

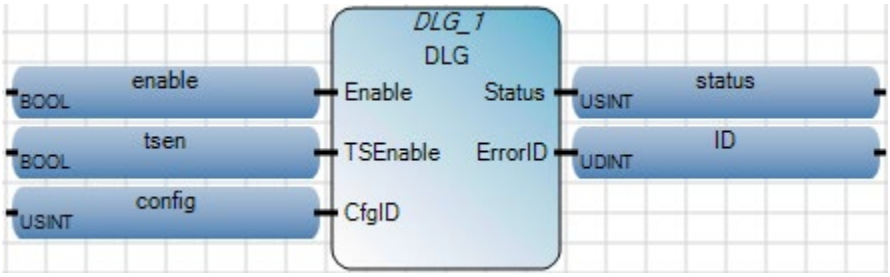
Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Schreiben des Datenprotokolls aktiviert. TRUE - wird eine Anstiegflanke erkannt, wird die Datenerfassungsoperation gestartet, sofern die vorherigen Anweisungsoperationen abgeschlossen sind. FALSE – keine Anstiegflanke erkannt.
TSEnable	Eingang	BOOL	TRUE - Aktivierungskennzeichen für Protokollierung von Datum/Uhrzeit-Stempel.
CfgID	Eingang	USINT	Konfigurations-VA ID-Nummer zur Datenprotokollierung von 1-10.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.
Status	Ausgang	USINT	Aktueller Status der Anweisung. Statuscodes der Datenerfassung: • 0 - Leerlauf

Parameter	Parametertyp	Datentyp	Beschreibung
			• 1 - Läuft • 2 - Erfolg; zeigt an, dass die Datenerfassung abgeschlossen wurde. • 3 - Fehler; zeigt an, dass die Datenerfassung mit einem Fehler abgeschlossen wurde.
ErrorID	Ausgang	UDINT	Ein eindeutiger numerischer Fehlercode für DLG.

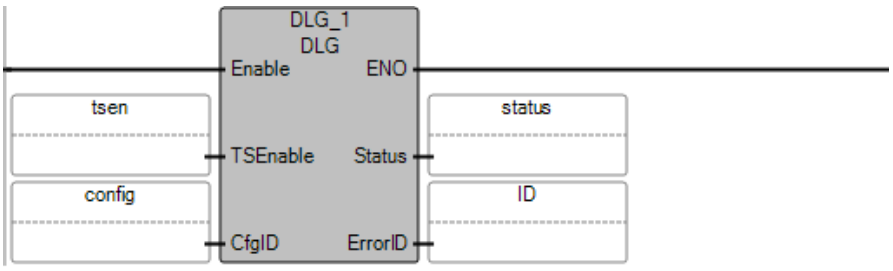
DLG-Fehlercodes

Fehlercode	Fehlername	Kommentare
0	DLG_ERR_NONE	Kein Fehler.
1	DLG_ERR_NO_SDCARD	Keine SD-Karte vorhanden.
2	DLG_ERR_RESERVED	Reserviert.
3	DLG_ERR_DATAFILE_ACCESS	Fehler beim Zugriff auf die Datenprotokollierungsdatei.
4	DLG_ERR_CFG_ABSENT	Keine Datenprotokollierungs-Konfigurationsdatei vorhanden.
5	DLG_ERR_CFG_ID	Keine Konfigurations-ID in der Datenprotokollierungs-Konfigurationsdatei vorhanden
6	DLG_ERR_RESOURCE_BUSY	Der mit dieser Datenprotokollierungs-ID verknüpfte Datenprotokollierungsvorgang wird von einem anderen FB-Vorgang verwendet.
7	DLG_ERR_CFG_FORMAT	Das Format der Datenprotokollierungs-Konfigurationsdatei ist ungültig.
8	DLG_ERR_RTC	Echtzeituhr ist ungültig.
9	DLG_ERR_UNKNOWN	Nicht spezifizierter Fehler ist aufgetreten.

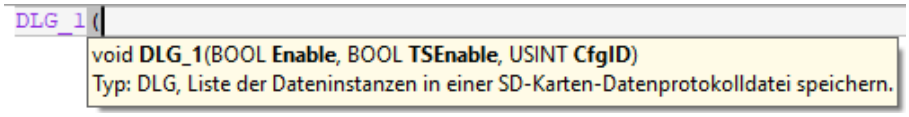
Diagrammbeispiel für Funktionsblock DLG



DLG-Kontaktplan-Beispiel



Beispiel für DLG in strukturiertem Text




```
1  DLG_1 (EN, Enable, TSEnable, CfgID)
2  output := DLG_1.ENO
3  status := DLG_1.Status
4  ID := DLG_1.ErrorID
```

IIM (sofortige Eingabe)

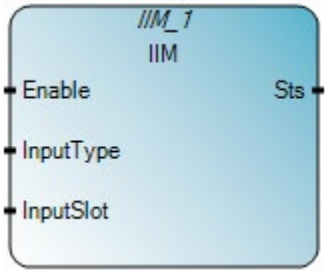
Aktualisieren der Eingaben vor normalem Ausgabescan.

Details zum Betrieb:

- In der Regel zu Beginn eines Interrupt-Programms verwendet, um Eingänge, die umgehend gescannt werden, um die aktuellen Eingänge abzurufen, auszuwählen oder zu maskieren.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für Micro830-, Micro850- und Micro870-Controller.

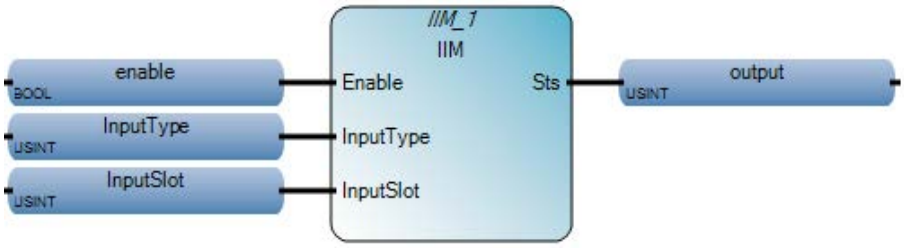


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

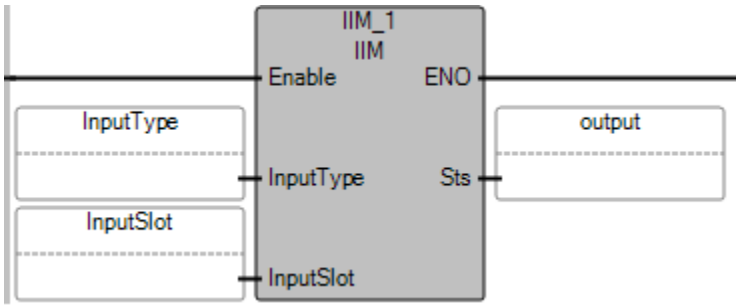
Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Anweisungsblock ausführen. FALSE – nicht ausführen.
InputType	Eingang	USINT	Der Eingangstyp: 0 - Integrierter Eingang. 1 - Einsteck-Modul.
InputSlot	Eingang	USINT	Identifiziert den Eingangs-Steckplatz. 0 – Integrierter Eingang 1,2,3,4,5 – Plug-in-Steckplatznummer. (Steckplätze werden von links nach rechts, beginnend mit Nummer 1, nummeriert.) Bei integrierten Eingängen: immer 0. Für Einsteck-Module werden die Steckplätze 1, 2, 3, 4, 5 verwendet (Steckplatz-Nummer des Moduls, wobei die Nummerierung von links bei 1 beginnt).
Sts	Ausgang	USINT	Sofortiger Eingangsscanstatus. IIM-Statuscodes (Sts): • 0x00 – Nicht aktiviert (es wird keine Aktion ausgeführt). • 0x01– Eingangs-/Ausgangsscan erfolgreich durchgeführt. • 0x02 – Eingangs-/Ausgangstyp ungültig.

Parameter	Parametertyp	Datentyp	Beschreibung
			• 0x03 – Steckplatz für Eingang/Ausgang ungültig.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. TRUE – Eingang aktualisiert. FALSE – Eingang nicht aktualisiert. Gilt nur für Kontaktplan-Programme.

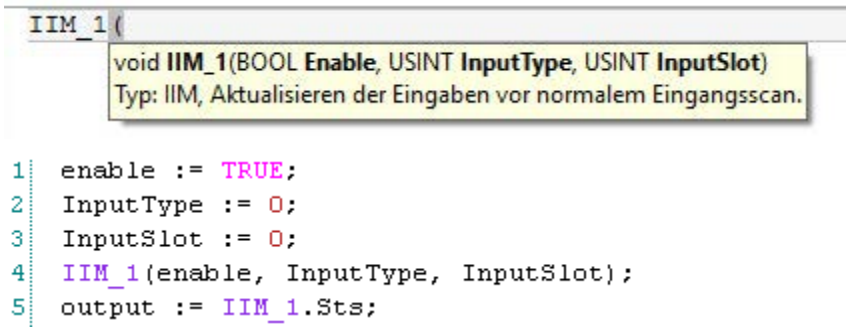
Diagrammbeispiel für Funktionsblock IIM



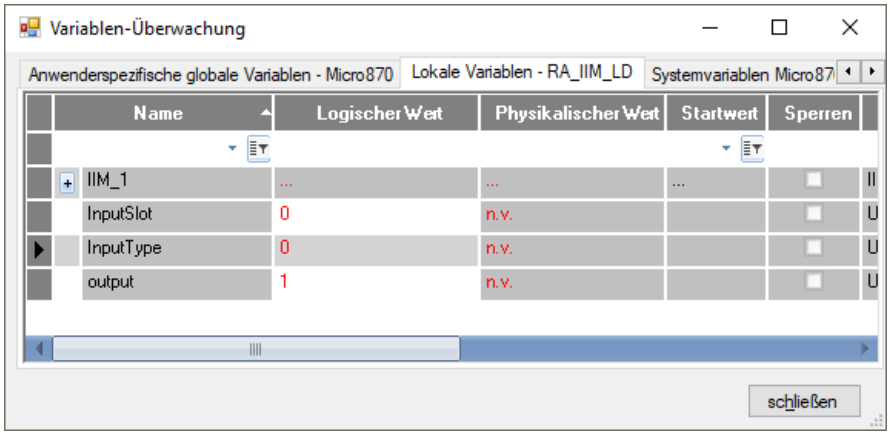
IIM-Kontaktplan-Beispiel



Beispiel für IIM in strukturiertem Text



Ergebnisse



IOM (sofortige Ausgabe)

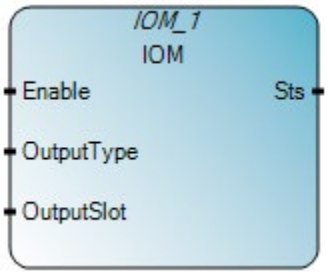
Aktualisieren der Ausgaben vor normalem Ausgabescan.

Details zum Betrieb:

- In der Regel zum Ende eines Interrupt-Programms verwendet, um Ausgänge, die umgehend gescannt und aktualisiert werden, auszuwählen oder zu maskieren.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für Micro830-, Micro850- und Micro870-Controller.

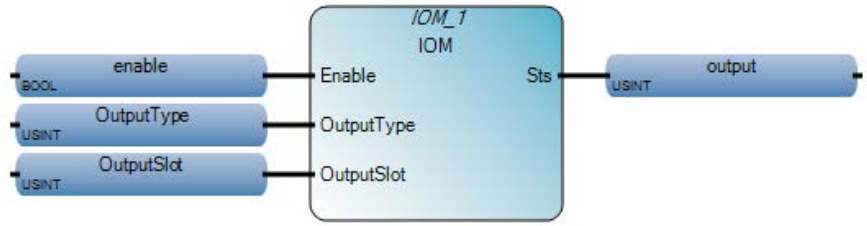


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

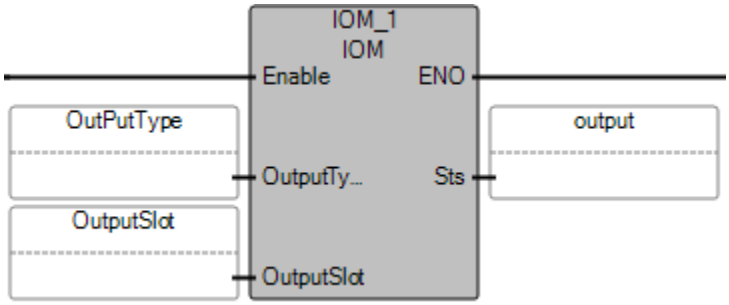
Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Anweisung ausführen. FALSE – nicht ausführen.
OutputType	Eingang	USINT	Identifiziert den Typ der Ausgabe: 0 - Integrierter Ausgang. 1 - Einsteck-Ausgang.
OutputSlot	Eingang	USINT	Identifiziert den Ausgangs-Steckplatz: 0 – Integrierter Ausgang 1,2,3,4,5 – Plug-in-Steckplatznummer. (Steckplätze werden von links nach rechts, beginnend mit Nummer 1, nummeriert.) Bei integrierten Ausgängen: immer 0. Für Einsteck-Module Einsteckplätze 1, 2, 3, 4, 5 verwendet (Steckplatz-Nummer des Moduls, wobei die Nummerierung von links bei 1 beginnt).

Parameter	Parametertyp	Datentyp	Beschreibung
Sts	Ausgang	USINT	Sofortiger Ausgangsscanstatus. IOM-Statuscodes (Sts): • 0x00 – Nicht aktiviert (es wird keine Aktion ausgeführt). • 0x01– Eingangs-/Ausgangsscan erfolgreich durchgeführt. • 0x02 – Eingangs-/Ausgangstyp ungültig. • 0x03 – Steckplatz für Eingang/Ausgang ungültig.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. TRUE – Ausgang aktualisiert. FALSE – Ausgang nicht aktualisiert. Gilt nur für Kontaktplan-Programme.

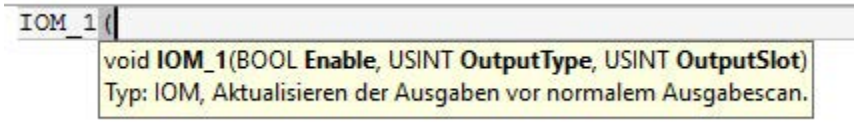
Diagrammbeispiel für Funktionsblock IOM



IOM-Kontaktplan-Beispiel

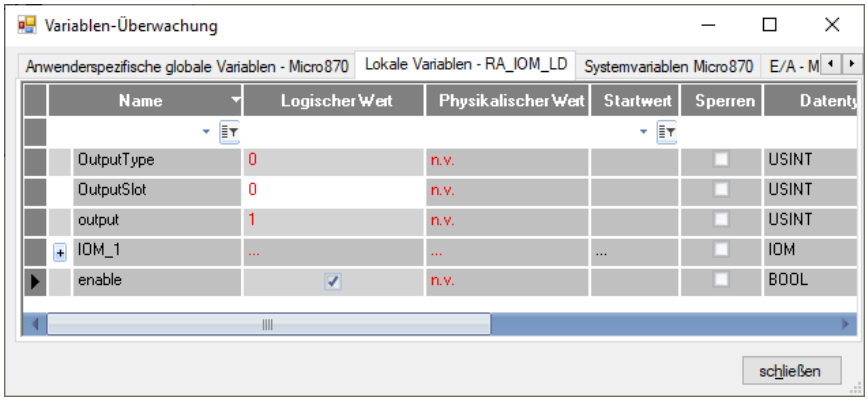


Beispiel für IOM in strukturiertem Text




```
1  enable := TRUE;
2  OutputType := 0;
3  OutputSlot := 0;
4  IOM_1(enable, OutputType, OutputSlot);
5  output := IOM_1.Sts;
```

Ergebnisse

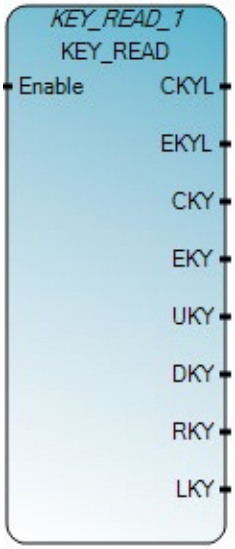


KEY_READ (Lesen der Tasten auf LCD-Modul)

Liest den Tastenstatus auf dem optionalen LCD-Modul, wenn die Benutzeranzeige aktiv ist.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

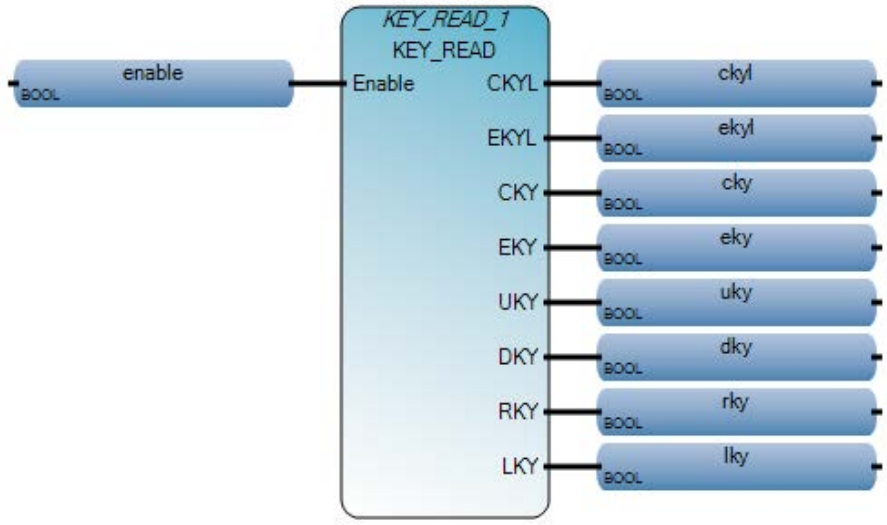
Diese Anweisung gilt für Micro810-Controller.



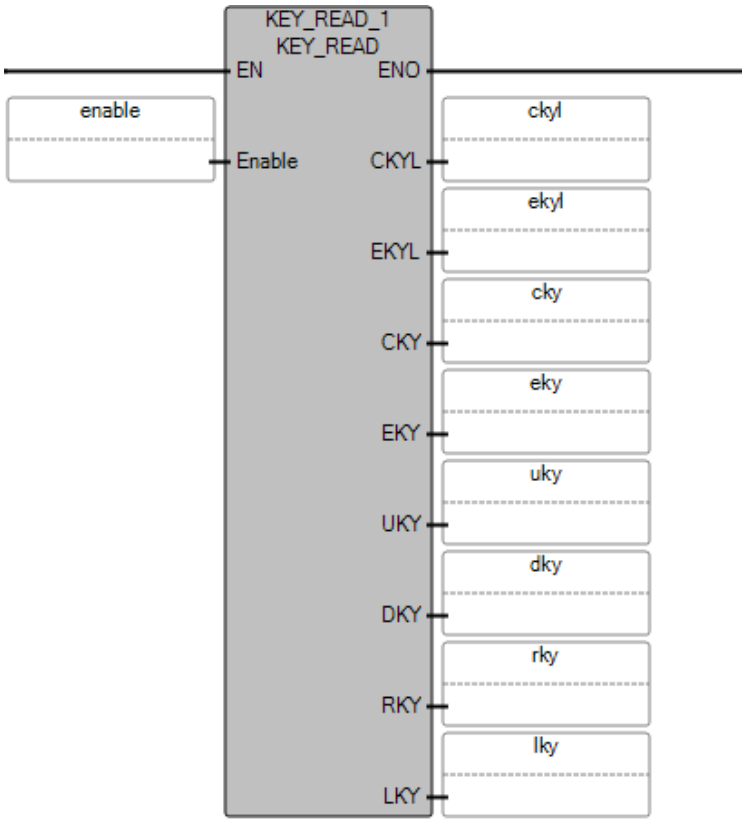
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Aktiviert den Anweisungsblock. TRUE – aktiviert das Lesen der Tasten auf der Remote-LCD-Tastatur. FALSE – deaktiviert das Lesen der Tasten auf der Remote-LCD-Tastatur.
CKYL	Ausgang	BOOL	TRUE: ESC-Taste für länger als 2 Sekunden gedrückt.
EKYL	Ausgang	BOOL	TRUE: OK-Taste für länger als 2 Sekunden gedrückt.
CKY	Ausgang	BOOL	TRUE: ESC-Taste gedrückt.
EKY	Ausgang	BOOL	TRUE: OK-Taste gedrückt.
UKY	Ausgang	BOOL	TRUE: Nach oben-Taste gedrückt.
DKY	Ausgang	BOOL	TRUE: Nach unten-Taste gedrückt.
LKY	Ausgang	BOOL	TRUE: Nach links-Taste gedrückt.
RKY	Ausgang	BOOL	TRUE: Nach rechts-Taste gedrückt.

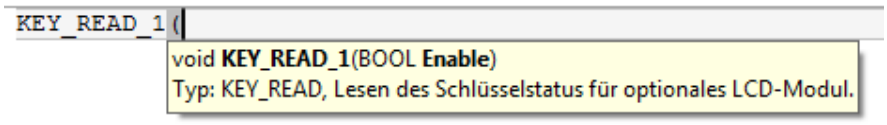
Diagrammbeispiel für Funktionsblock KEY_READ



KEY_READ-Kontaktplan-Beispiel



Beispiel für KEY_READ in strukturiertem Text




```
KEY_READ_1(enable);
ckyl := KEY_READ_1.CKYL;
ekyl := KEY_READ_1.EKYL;
cky := KEY_READ_1.CKY;
eky := KEY_READ_1.EKY;
uky := KEY_READ_1.UKY;
dky := KEY_READ_1.DKY;
lky := KEY_READ_1.LKY;
rky := KEY_READ_1.RKY;
```

(* ST-Äquivalenz: *)

```
KEY_READ_1(KEYENABLE) ;
KEY_EKYL := KEY_READ_1.EKYL ;
KEY_CKY := KEY_READ_1.CKY ;
KEY_EKY := KEY_READ_1.EKY ;
KEY_UKY := KEY_READ_1.UKY ;
KEY_DKY := KEY_READ_1.DKY ;
KEY_RKY := KEY_READ_1.RKY ;
KEY_LKY := KEY_READ_1.LKY ;
```

KEY_READ_REM (Tasten für Remote-LCD lesen)

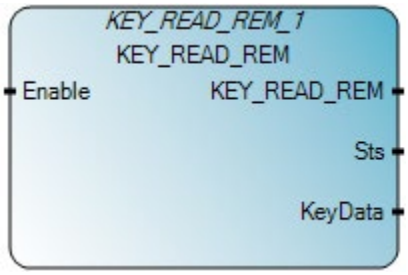
Liest den Tastenstatus auf dem optionalen Remote-LCD-Modul, wenn die Benutzeranzeige aktiv ist.

Details zum Betrieb:

- Verwenden Sie die LCD_REM-Anweisung, um die Benutzeranzeige auf dem Remote-LCD-Modul zu aktivieren. Wenn die Benutzeranzeige nicht aktiv ist, tritt während der Ausführung von KEY_READ_REM ein Fehler auf.
- P-BUTTON-Eigenschaft in LCD-Funktionsdatei wird aktiviert; andernfalls ergibt jeder Tastenstatus „FALSE“.
- Für die KEY_READ_REM-Anweisung werden nur einzelne Tastenbetätigungen unterstützt; Tastenkombinationen werden nicht unterstützt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für Micro820-Controller.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Aktivieren FALSE – Deaktivieren
KEY_READ_REM	Ausgang	BOOL	TRUE – Remote-LCD-Tastendaten erfolgreich gelesen. FALSE – Enable = False, Fehler beim Lesen. Remote-LCD-Tastendaten oder Benutzeranzeige nicht aktiv.
Sts	Ausgang	UINT	Status des KEY_READ_REM-Vorgangs. KEY_READ_REM-Statuscodes: • 0 – Aktivierungseingabe ist falsch. • 1 – Lesen der Tastaturdaten erfolgreich. • 2 – Remote-LCD nicht erkannt. Kann auftreten wenn: • Remote-LCD ist nicht physisch am Controller angeschlossen (oder die Verdrahtung ist fehlerhaft). • Einstellungen des seriellen Anschlusses sind anders als von der Remote-LCD gefordert. • 3 – Verbindungsfehler. Kann auftreten, wenn es einen Fehler mit dem internen Status der Maschine gibt. Wird möglicherweise durch eine Inkompatibilität zwischen Controller-FW-Version und RLCD-FW-Version verursacht. • 4 – Benutzeranzeige ist nicht aktiv. • 5-65535 – Reserviert.
KeyData	Ausgang	UDINT	Remote-LCD-KeyPad-Daten. KeyData-Definitionen sind in der KeyData-Bitfelder-Tabelle definiert.

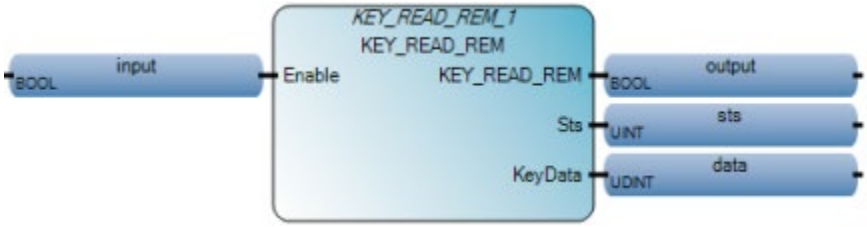
KeyData-Bitfelder-Tabelle

Verwenden Sie diese Tabelle, um die KeyData-Bitfelder für KEY_READ_REM zu ermitteln.

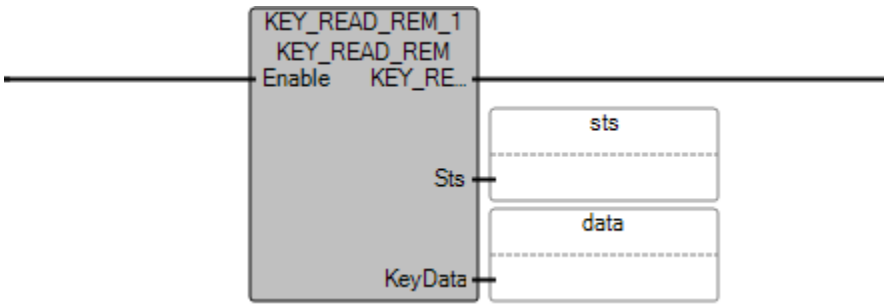
Bit-Nr. in KeyData	Name	Parameterbeschreibung
0	UKY	TRUE = Nach-oben-Taste gedrückt.
1	DKY	TRUE = Nach-unten-Taste gedrückt.
2	LKY	TRUE = Nach-links-Taste gedrückt.
3	RKY	TRUE = Nach-rechts-Taste gedrückt.
4	F1KY	TRUE = F1-Taste gedrückt.
5	F2KY	TRUE = F2-Taste gedrückt.

Bit-Nr. in KeyData	Name	Parameterbeschreibung
6	F3KY	TRUE = F3-Taste gedrückt.
7	F4KY	TRUE = F4-Taste gedrückt.
8	F5KY	TRUE = F5-Taste gedrückt.
9	F6KY	TRUE = F6-Taste gedrückt.
10	EKY	TRUE = Eingabetaste gedrückt.
11	CKY	TRUE = Abbruchtaste gedrückt.
12	EKYL	TRUE = ESC-Taste länger als 2 Sekunden gedrückt.
13	CKYL	TRUE = Abbruchtaste länger als 2 Sekunden gedrückt.
14-31	--	Reserviert.

Diagrammbeispiel für Funktionsblock KEY_READ_REM



KEY_READ_REM-Kontaktplan-Beispiel



Beispiel für KEY_READ_REM in strukturiertem Text

```
KEY_READ_REM_2(
```

void **KEY_READ_REM_2**(BOOL **Enable**)
Typ: KEY_READ_REM, Überprüfen des Schlüsselstatus für Remote-LCD.

```
1  KEY_READ_REM (Enable);
2  output := KEY_READ_REM.ENO
3  sts := KEY_READ_REM.Sts
4  data := KEY_READ_REM.KeyData
```

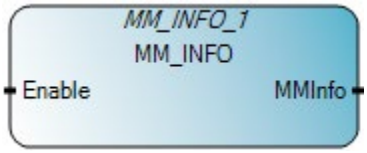
MM_INFO (Speichermodulinformationen)

Liest die Header-Informationen des Speichermoduls. Wenn kein Speichermodul vorhanden ist, werden alle Werte auf null (0) zurückgesetzt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Die Ausgänge werden jedes Mal zurückgesetzt, wenn die Anweisung auf die simulierte Steuerung (2080-LC50-48QWB-SIM) angewendet wird.

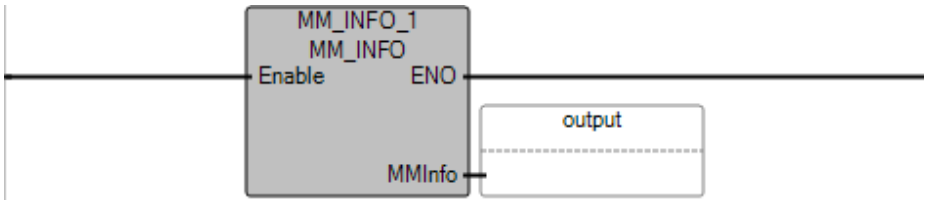


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

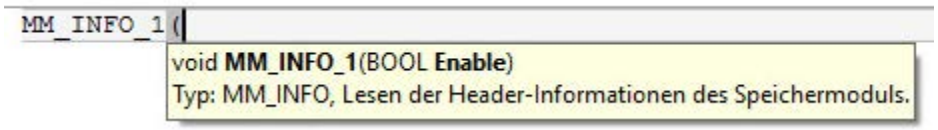
Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Header-Informationen des Speichermoduls lesen. FALSE – es findet kein Lesevorgang statt, und die ausgegebenen Speichermodulinformationen sind ungültig.
MMInfo	Ausgang	MMINFO	Informationen zum Speichermodul werden im Datentyp MMINFO auf seite 363 definiert.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Diagrammbeispiel für Funktionsblock MM_INFO

Beispiel für MM_INFO-Kontaktplan



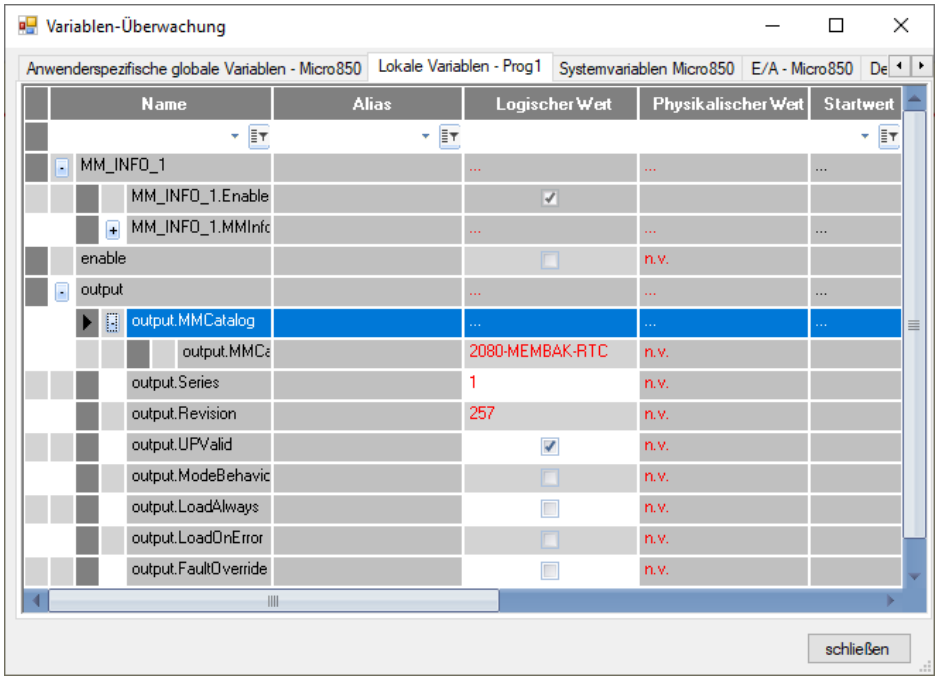
Beispiel für strukturierten MM_INFO-Text



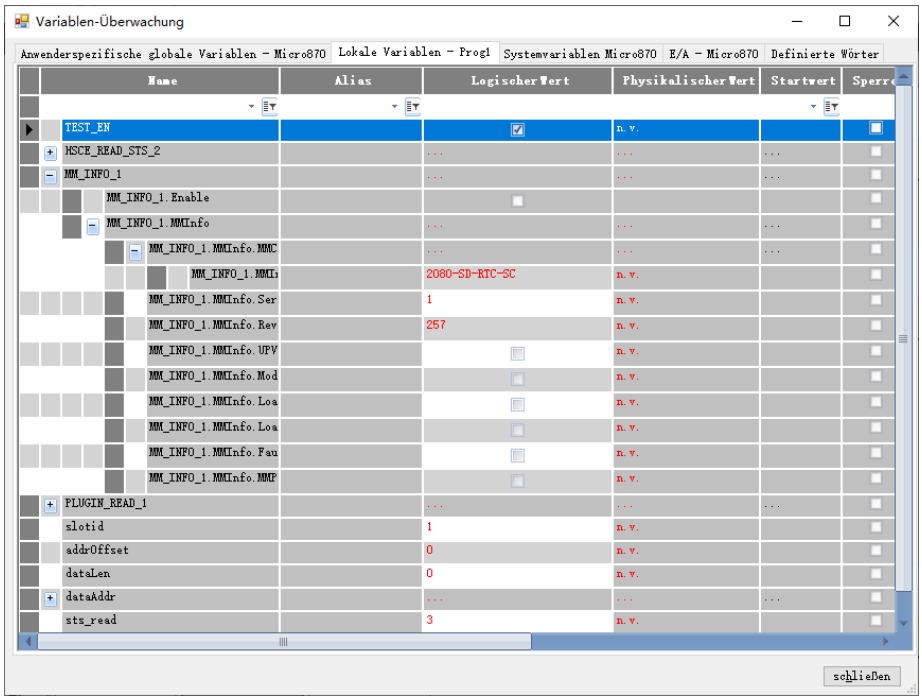

```
1  MM_INFO_1(enable);
2  output := MM_INFO_1.MMInfo;
```

Ergebnisse

Bei Controllern, die 2080-MEMBAK-RTC verwenden:



Bei Controllern, die eine SD-Karte verwenden:



Datentyp MMINFO

In der folgenden Tabelle werden die Parameter des Datentyps „MMINFO“ beschrieben.

Parameter	Datentyp	Beschreibung
MMCatalog	MMCATNUM	Die Katalognummer des Speichermoduls. Wenn bei Controllern mit SD-Karte der MM_INFO auf seite 361-Befehl verwendet wird, dann ist der MMCatalog „SD CARD“.

Parameter	Datentyp	Beschreibung
Series	UINT	Die Serie des Speichermoduls. Wenn bei Controllern mit SD-Karte die MM_INFO-Anweisung verwendet wird, dann ist die Serie 1.
Revision	UINT	Die Revision des Speichermoduls. Wenn bei Controllern mit SD-Karte die MM_INFO-Anweisung verwendet wird, dann ist die Revision 257.
UPValid	BOOL	Nutzprogramm ist vorhanden (TRUE: Gültiges Projekt möglicherweise gefunden) Hinweis: Selbst wenn TRUE, besteht noch immer die Möglichkeit, dass das Projekt während des Herunterladens erkannt oder als ungültig wiederhergestellt wird, wenn einzelne Dateien fehlen oder beschädigt sind.
ModeBehavior	BOOL	Modusverhalten (TRUE: Beim Einschalten in den Betriebsmodus wechseln).
LoadAlways	BOOL	Speichermodul beim Einschalten immer auf Controller zurücksetzen.
LoadOnError	BOOL	Speichermodul beim Einschalten mit Fehler auf Controller zurücksetzen.
FaultOverride	BOOL	Fehler beim Einschalten nicht berücksichtigen.
MMPresent	BOOL	Speichermodul ist vorhanden.

MODULE_INFO

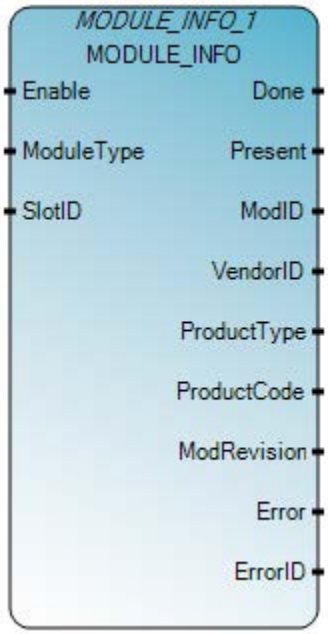
Liest Modulinformationen aus einem Plug-in- oder Erweiterungsmodul.

Details zum Betrieb:

- Plug-in-Modulinformationen werden zur Laufzeit gelesen.
- Das 2080-MEMBAK-RTC Plug-in-Speichermodul wird nicht unterstützt.
- Erweiterungsmodulinformationen werden gelesen, wenn das Modul eingeschaltet wird.
- Wenn ein Plug-in- oder Erweiterungsmodul nicht mit einer Modul-ID, einem Produkt-Typ oder einem Produktcode definiert wird, gibt der Vorgang MODULE_INFO 0 für den jeweiligen Ausgabeparameter zurück.
- Die Identifizierungsinformationen für Plug-in- und Erweiterungsmodule wird durch Allen-Bradley definiert und wird nachstehend als Teil der Beschreibung MODULE_INFO bereitgestellt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870. Erweiterungsmodule werden nur für Micro850- und Micro870-Controller unterstützt.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE - führt MODULE_INFO-Lesevorgang aus. FALSE - führt den Lesevorgang nicht aus. Alle Ausgangsdatenwerte werden auf 0 zurückgesetzt.
ModuleType	Eingang	USINT	Identifiziert den Modultyp: • 1 – 2085-Erweiterungsmodul. • 2 – 2080-Plug-in-Modul.
SlotID	Eingang	USINT	Die Steckplatznummer, an der sich das Plug-in- oder Erweiterungsmodul befindet. Schacht-IDs sind: 1, 2, 3, 4, 5 Schacht 1 befindet sich ganz links.
Done	Ausgang	BOOL	TRUE – Vorgang wurde erfolgreich abgeschlossen. FALSE – Vorgang kann nicht ausgeführt werden, oder ein Fehler ist aufgetreten.
Present	Ausgang	BOOL	Erkennt das Plug-in- oder Erweiterungsmodul im Steuerungssteckplatz. TRUE – Modul ist physisch vorhanden. FALSE – Modul ist nicht physisch vorhanden.
ModID	Ausgang	UINT	Die Kennung für das Modul im Controller-Steckplatz. • Plug-in-Module werden mit einer eindeutigen Modul-ID definiert. • Erweiterungsmodule werden nicht mit einer eindeutigen Modul-ID definiert, ModID gibt 0 zurück.
VendorID	Ausgang	UINT	Die Hersteller-ID des Plug-in- oder Erweiterungsmoduls. Für Allen-Bradley-Produkte ist die Lieferanten-ID 1.
ProductType	Ausgang	UINT	Der Produkt-Typ des Plug-in- oder Erweiterungsmoduls.
ProductCode	Ausgang	UINT	Der Produktcode des Plug-in- oder Erweiterungsmoduls.
ModRevision	Ausgang	UINT	Die Revisionsinformationen des Plug-in- oder Erweiterungsmoduls.

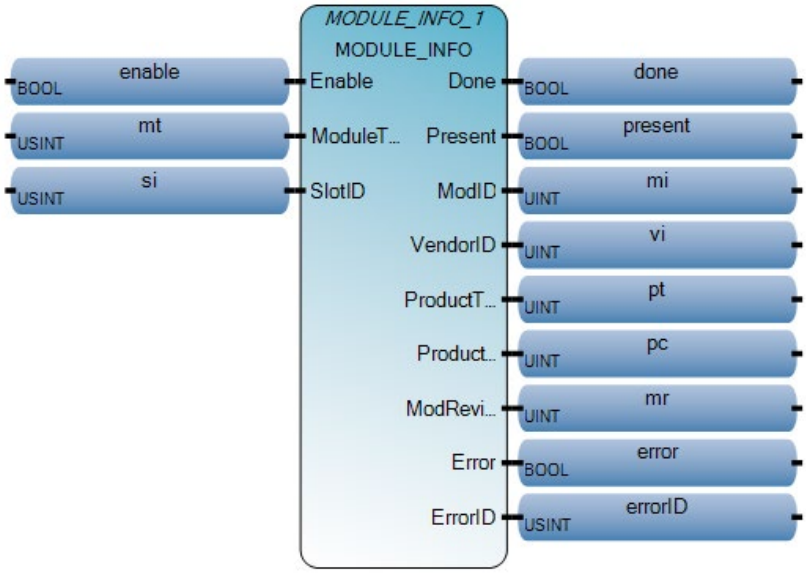
Parameter	Parametertyp	Datentyp	Beschreibung
Error	Ausgang	BOOL	Zeigt das Vorhandensein eine Fehlerbedingung an. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	USINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler sind in den MODULE_INFO-Fehlercodes definiert.

MODULE_INFO-Fehlercodes

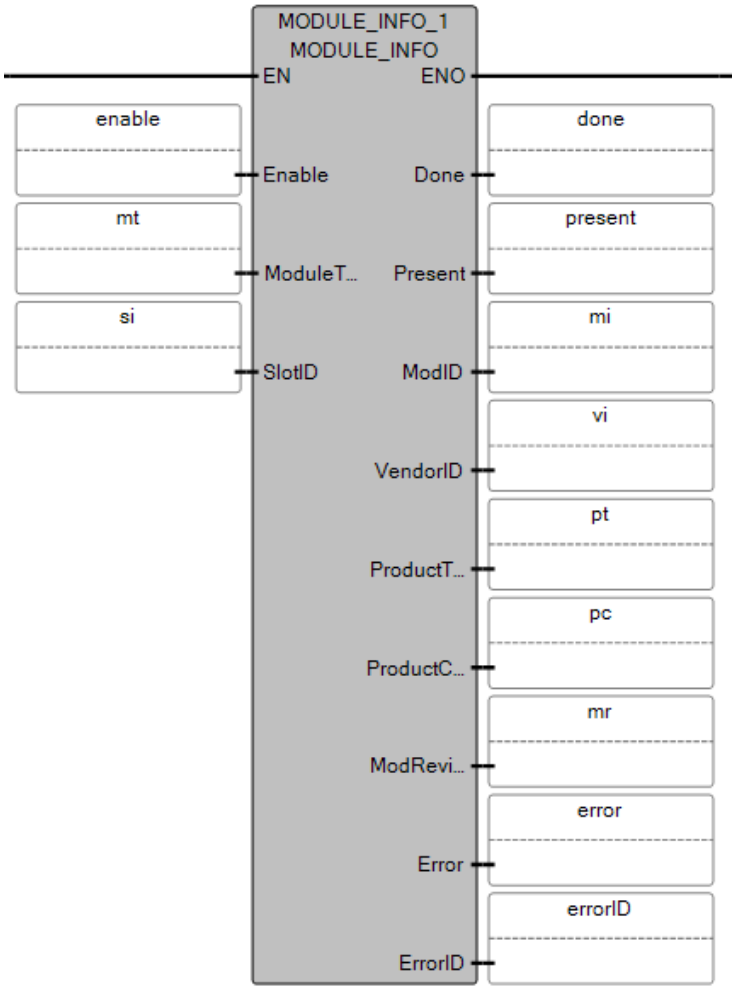
Verwenden Sie diese Tabelle, um die MODULE_INFO-Fehlercodes und deren Beschreibung zu bestimmen.

Fehlercode	Fehlerbeschreibung
1	Ungültiger Modultyp. Ändern Sie diesen in einen gültigen Modultyp. Die gültigen Modultypen sind: 1 – 2085-Erweiterungsmodul. Nur auf Micro850-Controllern unterstützt. 2 – 2080-Plug-in-Modul. Unterstützt auf Micro820-, Micro830-, Micro850-Controllern.
2	Ungültige Steckplatznummer. Ändern Sie diese in eine gültige Steckplatznummer.
3	Ungültiger Erweiterungsmodultyp.
4	Schwerwiegender Fehler für Erweiterungsmodul.
5	Informationen lesen wird für Plug-in-Modul nicht unterstützt.
6	Lesefehler des Plug-in-Moduls beim Lesen der Modul-ID aufgetreten.
7	Lesefehler des Plug-in-Moduls beim Lesen der Lieferanten-ID aufgetreten.
8	Lesefehler des Plug-in-Moduls beim Lesen des Produkt-Typs aufgetreten.
9	Lesefehler des Plug-in-Moduls beim Lesen des Produktcodes aufgetreten.
10	Lesefehler des Plug-in-Moduls beim Lesen der Modulrevision aufgetreten.

Beispiel für ein MODULE_INFO-Funktionsblockdiagramm



Beispiel für einen MODULE_INFO-Kontaktplan



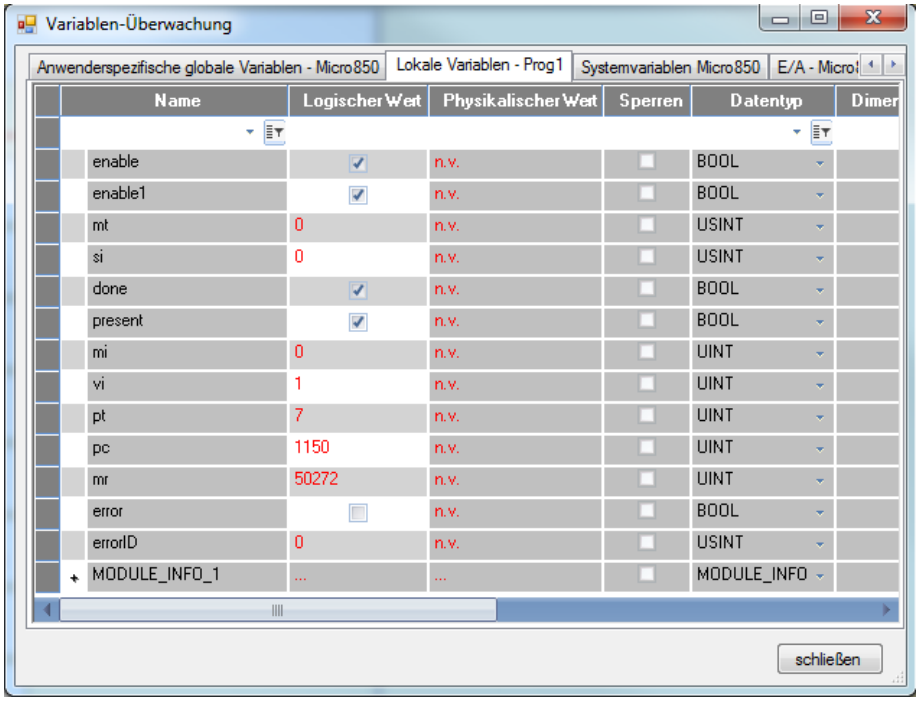
Beispiel für strukturierten MODULE_INFO-Text

```
1  MODULE_INFO_1(enable, mt, si);
2  done := MODULE_INFO_1.Done;
3  present :=MODULE_INFO_1.Present;
4  mi :=MODULE_INFO_1.ModID;
5  vi :=MODULE_INFO_1.VendorID;
6  pt :=MODULE_INFO_1.ProductType;
7  pc :=MODULE_INFO_1.ProductCode;
8  mr :=MODULE_INFO_1.ModRevision;
9  error :=MODULE_INFO_1.Error;
10 errorID :=MODULE_INFO_1.ErrorID;
```

```
MODULE_INFO_1(
```

void **MODULE_INFO_1**(BOOL **Enable**, USINT **ModuleType**, USINT **SlotID**)
Typ: MODULE_INFO, Modulinformationen von Plug-In und Erweiterungs-E/A abrufen

Ergebnisse



MODULE_INFO - Informationen zum Plug-in- und Erweiterungsmodul

Die folgenden Informationen enthalten den Typ des Plug-in- und Erweiterungsmoduls, die Modul-ID, die Hersteller-ID, Produktcodes und die Revisionswortbeschreibungen für Erweiterungsmodule, wie von Allen-Bradley definiert.

Informationen zu Plug-in-Modulen

Verwenden Sie diese Tabelle zur Bestimmung der von Allen-Bradley definierten Plug-in-Modulinformationen.

Plug-In-Modul	Plug-in-Typ	Modul-ID	Hersteller-ID	Produkttyp	Produktcode
2080-IF2	Analog	96	1	10	32
2080-IF4	Analog	98	1	10	33
2080-OF2	Analog	100	1	10	34
2080-TC2	Analog	102	1	10	35
2080-RTD2	Analog	104	1	10	36
2080-DNET20	Kommunikation	34	1	12	249
2080-SERIALISOL	Kommunikation	32	1	-	-
2080-IQ4	Digital	192	1	7	192
2080-OB4	Digital	193	1	7	193
2080-OV4	Digital	194	1	7	194
2080-IQ4OB4	Digital	195	1	7	195
2080-IQ4OV4	Digital	196	1	7	196
2080-OW4I	Digital	197	1	7	197
2080-MOT-HSC	Spezialität	48	1	43	48
2080-TRIMPOT6	Spezialität	72	1	-	-

Informationen zu Erweiterungsmodulen

Verwenden Sie diese Tabelle zur Bestimmung der von Allen-Bradley definierten Erweiterungsmodulinformationen.

Modellnummer	Erweiterungsmodultyp	Modul-ID	Hersteller-ID	Produkttyp	Produktcode
2085-IF4	Analog	-	1	10	208
2085-IF8	Analog	-	1	10	206
2085-IRT4	Analog	-	1	10	213
2085-OF4	Analog	-	1	10	214
2085-IA8	Digital	-	1	7	1148
2085-IM8	Digital	-	1	7	1152
2085-IQ16	Digital	-	1	7	1144
2085-IQ32T	Digital	-	1	7	1145
2085-OA8	Digital	-	1	7	1149
2085-OB16	Digital	-	1	7	1146
2085-OV16	Digital	-	1	7	1147
2085-OW16	Digital	-	1	7	1151
2085-OW8	Digital	-	1	7	1150

Revisions-Wortbeschreibungen für Erweiterungsmodul

Verwenden Sie diese Tabelle zur Bestimmung der Revisionswortinformationen für Allen-Bradley-Erweiterungsmodule.

Bit	Name	Beschreibung
15 14	Max. Baudrate (1:0)	Diese Bits beschreiben die maximale Frequenz für die maximale Baudrate (1:0). 00 (Bin) ist 2 Mbit/s 01 (Bin) ist 4 Mbit/s 10 (Bin) ist 8 Mbit/s 11 (Bin) ist 16 Mbit/s
13 12 11 10	Kleine Revision (3:0)	Kleine Revision (3:0) ist die Bezeichnung für eine kleine Produktrevision. Dieses Feld zeigt die kleine Revision der Katalognummer unter Angabe von Hersteller-ID, Produkt-Typ, Produktcode, Serie und Hauptrevision an. Die kleine Revision reicht von 0 bis 15.
9 8 7 6 5	Hauptrevision (4:0)	Hauptrevision (4:0) ist die Bezeichnung für eine umfassende Produktrevision. Dieses Feld zeigt die Hauptrevision der Katalognummer unter Angabe von Hersteller-ID, Produkt-Typ, Produktcode und Serie an. Die Hauptrevision reicht von 0 bis 31.

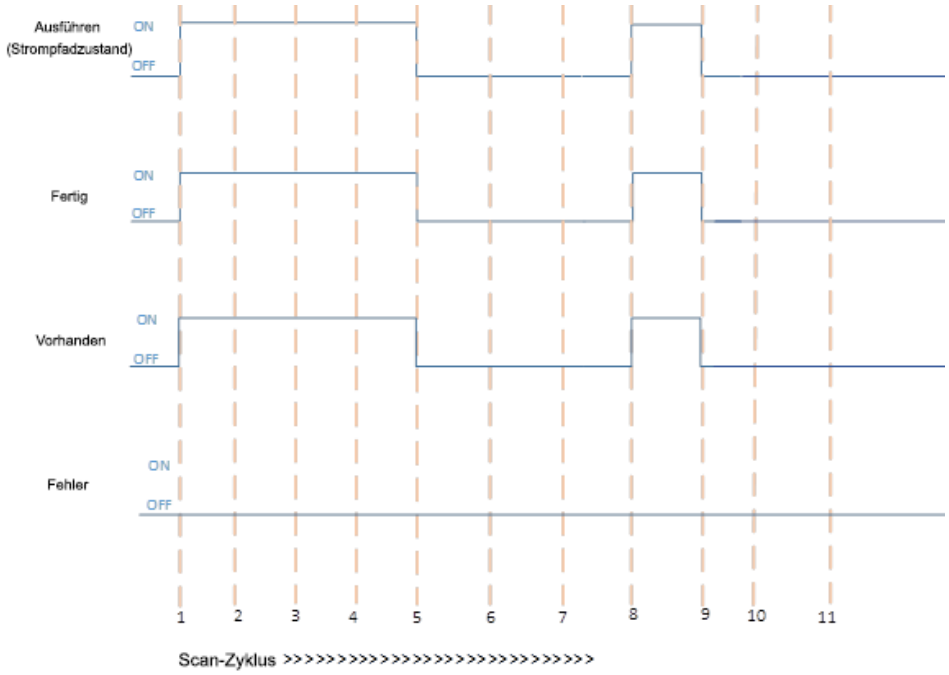
4	Serie (4:0)	SERIES (4:0) ist die Benennung der Produktserie. Dieses Feld zeigt die Serienbuchstaben der Katalognummer unter Angabe von Hersteller-ID, Produkt-Typ und Produktcode an.

Serie (4:0)	Serie	Serie (4:0)	Serie	Serie (4:0)	Serie	Serie (4:0)	Serie
0	A	8	I	16	Q	24	Y
1	B	9	J	17	R	25	Z
2	C	10	K	18	S	26	AA
3	D	11	L	19	T	27	AB
4	E	12	M	20	U	28	AC
5	F	13	N	21	V	29	AD
6	G	14	O	22	W	30	AE
7	H	15	P	23	X	31	>AE

MODULE_INFO-Anweisung für Timing-Diagramme

In den folgenden Beispielen für ein Zeitmessungsdiagramm werden Ausführungsszenarien für den MODULE_INFO-Befehl auf seite 364 beschrieben.

Wird erfolgreich ausgeführt, wenn ein Modul physisch vorhanden ist.

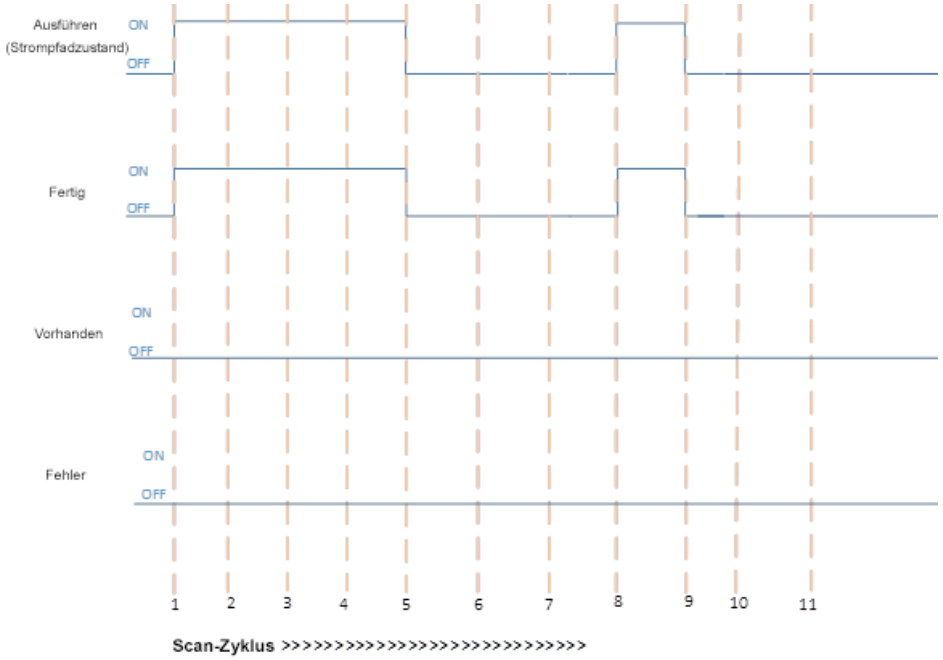


Verwenden Sie diese Tabelle, um die MODULE_INFO-Parameterwerte für jeden Scan-Zyklus zu ermitteln.

Scan-Zyklus	Beschreibung

Scan-Zyklus	Beschreibung
1, 8	Strompfadbedingung wird TRUE, wenn gilt: • Aktivierungseingabebit ist TRUE. • ModuleType und SlotID sind gültig. Ein physisches Modul ist vorhanden. • Die Ausgabebits Fertig und Vorhanden sind TRUE. • Fehlerausgabebit ist FALSE. • Aktualisieren Sie modulbezogene Informationen für Modul-ID, Hersteller-ID, Produkttyp, Produktcode und Modulrevision entsprechend.
2,3,4	Keine Änderung in Strompfadbedingung.
5. 9	Die Strompfadbedingung wird FALSE, wenn das Aktivierungsbit FALSE ist. Alle Ausgabeparameter werden gelöscht.
6, 7, 10, 11	Keine Änderung in Strompfadbedingung. Aktivierungsbit ist FALSE. Alle Ausgabeparameter werden gelöscht.

Wird erfolgreich ausgeführt, wenn das Modul nicht physisch vorhanden ist.

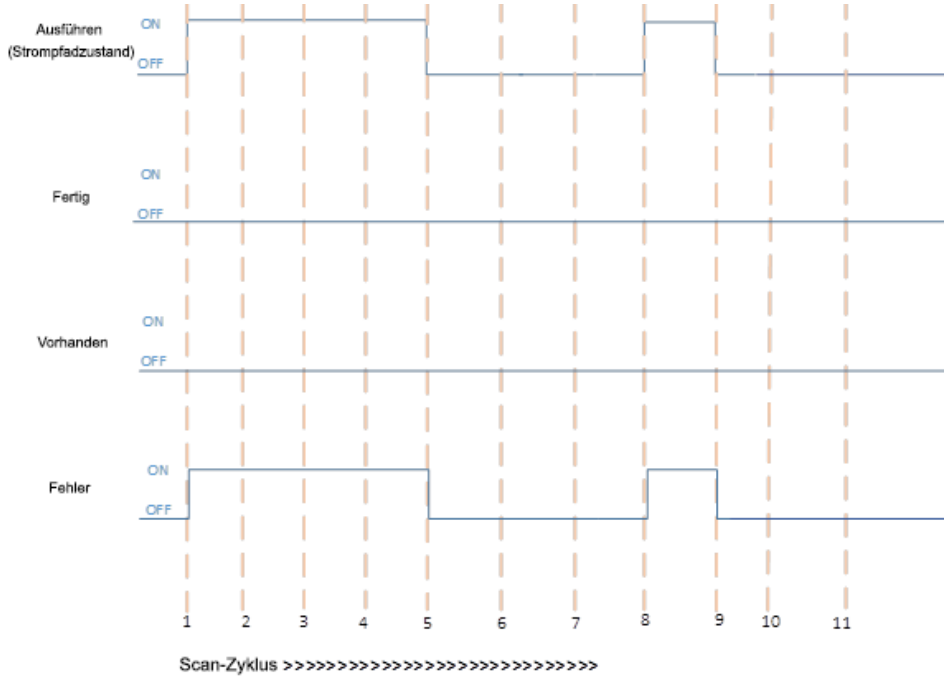


Verwenden Sie diese Tabelle, um die MODULE_INFO-Parameterwerte für jeden Scan-Zyklus zu ermitteln.

Scan-Zyklus	Beschreibung
1, 8	Strompfadbedingung wird TRUE, wenn gilt: • Aktivierungseingabebit ist TRUE. • ModuleType und SlotID sind gültig. Es ist kein physisches Modul vorhanden. • Done-Ausgangsbit ist TRUE. • Die Ausgabebits Fehler und Vorhanden sind FALSE. • Aktualisieren Sie modulbezogene Informationen für Modul-ID, Hersteller-ID, Produkttyp, Produktcode und Modulrevision entsprechend.

Scan-Zyklus	Beschreibung
2,3,4	Keine Änderung in Strompfadbedingung. • Aktivierungseingabebit ist TRUE. • Eingangsparameter sind gültig, und es ist kein physisches Modul vorhanden. • Die Ausgabeparameter werden entsprechend aktualisiert.
5. 9	Strompfadbedingung wird FALSE, wenn gilt: • Aktivierungsbit ist FALSE. • Alle Ausgabeparameter werden gelöscht.
6, 7, 10, 11	Keine Änderung in Strompfadbedingung. Aktivierungsbit ist FALSE. Alle Ausgabeparameter werden gelöscht.

MODULE_INFO-Ausführung mit Fehler

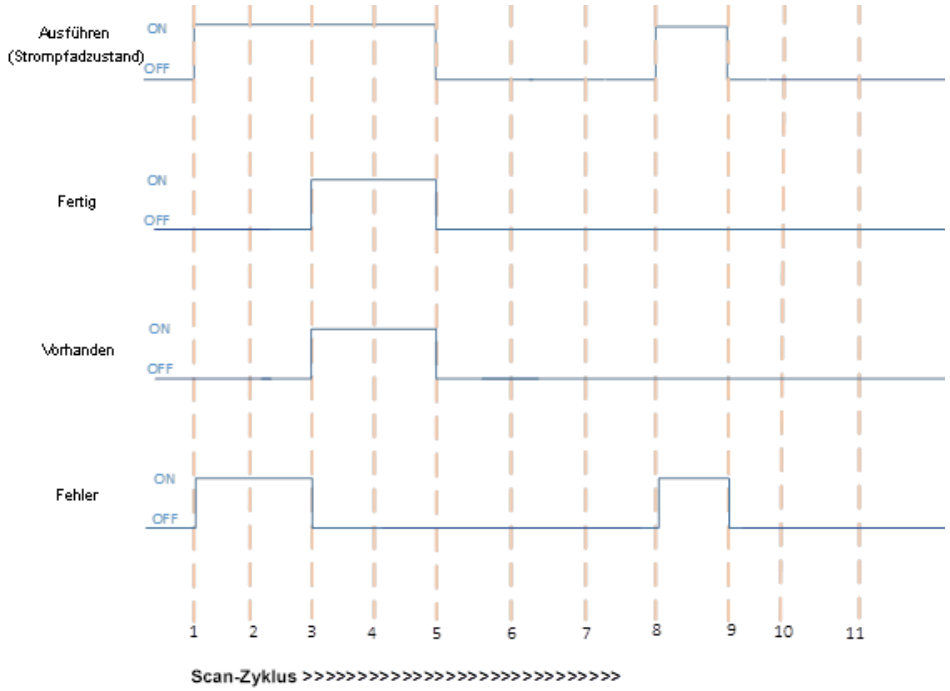


Verwenden Sie diese Tabelle, um die MODULE_INFO-Parameterwerte für jeden Scan-Zyklus zu ermitteln.

Scan-Zyklus	Beschreibung
1, 8	Strompfadbedingung wird TRUE, wenn gilt: • Aktivierungseingabebit ist TRUE. • ModuleType und SlotID sind gültig. Es ist kein physisches Modul vorhanden. • Die Ausgabebits Fertig und Vorhanden sind TRUE. • Ausgangsbit Fehler ist TRUE. • Modulbezogene Informationen für Modul-ID, Hersteller-ID, Produkttyp, Produktcode und Modulrevision werden gelöscht.
2,3,4	Keine Änderung in Strompfadbedingung. • Aktivierungseingabebit ist TRUE. • ModuleType, SlotID oder beides ist ungültig. • Die Ausgabeparameter werden entsprechend aktualisiert.

Scan-Zyklus	Beschreibung
5. 9	Strompfadbedingung wird FALSE, wenn gilt: • Aktivierungsbit ist FALSE. • Alle Ausgabeparameter werden gelöscht.
6, 7, 10, 11	Keine Änderung in Strompfadbedingung. Aktivierungsbit ist FALSE. Alle Ausgabeparameter werden gelöscht.

MODULE_INFO erfolgreiche Ausführung mit Fehler - kein physisches Modul

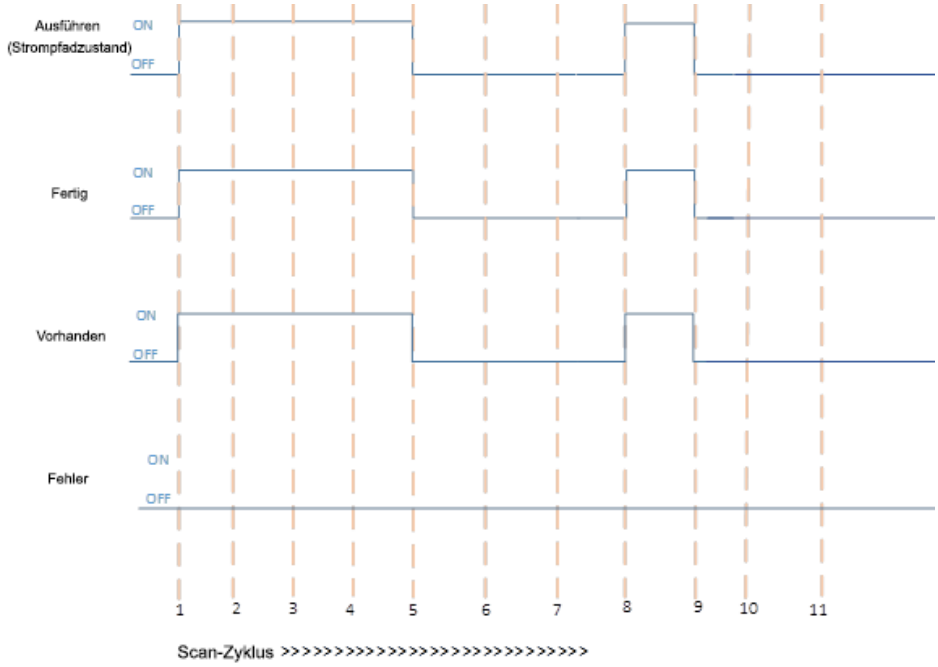


Verwenden Sie diese Tabelle, um die MODULE_INFO-Parameterwerte für jeden Scan-Zyklus zu ermitteln.

Scan-Zyklus	Beschreibung
1, 8	Strompfadbedingung wird TRUE, wenn gilt: • Aktivierungseingabebit ist TRUE. • ModuleType oder SlotID oder beides ist ungültig. Es ist kein physisches Modul vorhanden. • Die Ausgabebits Fertig und Vorhanden sind FALSE. • Ausgangsbit Fehler ist TRUE. • Modulbezogene Informationen für Modul-ID, Hersteller-ID, Produkttyp, Produktcode und Modulrevision werden gelöscht.
2	Keine Änderung in Strompfadbedingung. • Aktivierungseingabebit ist TRUE. • ModuleType oder SlotID oder beides ist ungültig. • Die Ausgabeparameter werden entsprechend aktualisiert.

Scan-Zyklus	Beschreibung
3, 4	Keine Änderung in Strompfadbedingung. • Aktivierungseingabebit ist TRUE. • ModuleType und SlotID sind beide gültig. Modul ist physisch vorhanden. • Die Ausgabebits Fertig und Vorhanden sind TRUE. • Fehlerausgabebit ist FALSE. • Aktualisieren Sie Modulinformationen für Modul-ID, Hersteller-ID, Produkttyp, Produktcode und Modulrevision entsprechend.
5. 9	Strompfadbedingung wird FALSE, wenn gilt: • Aktivierungsbit ist FALSE. • Alle Ausgabeparameter werden gelöscht.
6, 7, 10, 11	Keine Änderung in Strompfadbedingung. Aktivierungsbit ist FALSE. Alle Ausgabeparameter werden gelöscht.

MODULE_INFO erfolgreiche Ausführung mit Fehler, wenn physisches Modul vorhanden ist



Verwenden Sie diese Tabelle, um die MODULE_INFO-Parameterwerte für jeden Scan-Zyklus zu ermitteln.

Scan-Zyklus	Beschreibung
1, 8	Strompfadbedingung wird TRUE, wenn gilt: • Aktivierungseingabebit ist TRUE. • ModuleType oder SlotID oder beides ist gültig. Modul ist physisch vorhanden. • Die Ausgabebits Fertig und Vorhanden sind TRUE. • Fehlerausgabebit ist FALSE • Aktualisieren Sie modulbezogene Informationen für Modul-ID, Hersteller-ID, Produkttyp, Produktcode und Modulrevision entsprechend.

Scan-Zyklus	Beschreibung
2	Keine Änderung in Strompfadbedingung. • Aktivierungseingabebit ist TRUE. • Moduleingangsparameter sind gültig, und das Modul ist physisch vorhanden. • Die Ausgabeparameter werden entsprechend aktualisiert.
3, 4	Keine Änderung in Strompfadbedingung. • Aktivierungseingabebit ist TRUE. • ModuleType und SlotID sind beide ungültig. Modul ist physisch vorhanden. • Die Ausgabebits Fertig und Vorhanden sind FALSE. • Ausgangsbit Fehler ist TRUE und wird gelöscht. • Aktualisieren Sie Modulinformationen für Modul-ID, Hersteller-ID, Produkttyp, Produktcode und Modulrevision entsprechend.
5. 9	Strompfadbedingung wird FALSE, wenn gilt: • Aktivierungsbit ist FALSE. • Alle Ausgabeparameter werden gelöscht.
6, 7, 10, 11	Keine Änderung in Strompfadbedingung. Aktivierungsbit ist FALSE. Alle Ausgabeparameter werden gelöscht.

PLUGIN_INFO (Plug-in-Informationen)

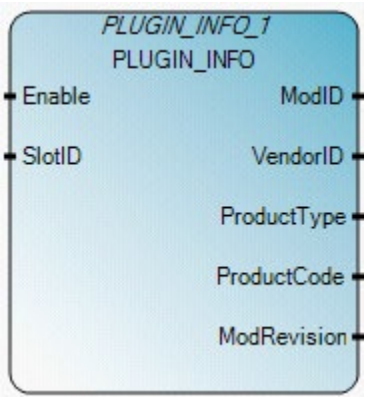
Liest Modulinformationen aus einem generischen Plug-in- oder Erweiterungsmodul.

Details zum Betrieb:

- In Connected Components Workbench 10 oder höher kann die PLUGIN_INFO-Anweisung alle generischen Plug-in- oder Erweiterungsmodulinformationen lesen, mit Ausnahme des 2080-MEMBAK-RTC-Speichermoduls.
- Wenn kein generisches Plug-in- oder Erweiterungsmodul vorhanden ist, werden alle Werte auf null (0) zurückgesetzt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

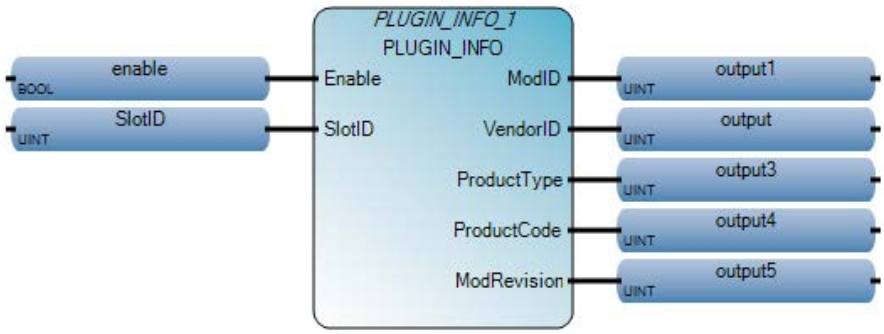
Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



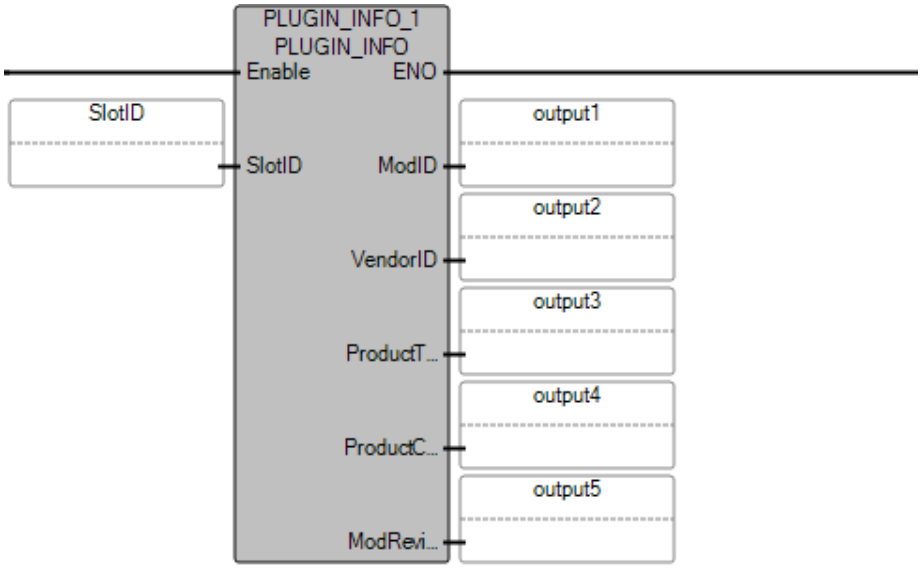
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Plug-in- oder Erweiterungsmodulinformationen werden gelesen und ausgeführt. FALSE – der Anweisungsblock wird nicht ausgeführt. Alle Ausgangsdatenwerte werden auf 0 zurückgesetzt.
SlotID	Eingang	UINT	Nummer des Einsteckplatzes: Schacht-ID = 1,2,3,4,5 (Beginnend mit dem ersten Steckplatz von links = 1). Nummer des Erweiterungsplatzes: Schacht-ID = 101, 102, 103, 104 (Beginnend mit dem ersten Steckplatz von links = 101).
ModID	Ausgang	UINT	Physische ID des generischen Plug-in-Moduls. • Falls das Erweiterungsmodul nicht unterstützt wird, ModID = 0xFFF • Wenn kein Plug-in- oder Erweiterungsmodul vorhanden ist, ModID = 0x0000
VendorID	Ausgang	UINT	Hersteller-ID des generischen Plug-in- oder Erweiterungsmoduls. Bei Allen Bradley-Produkten lautet die Lieferanten-ID 1. Wenn kein Plug-in- oder Erweiterungsmodul vorhanden ist, VendorID = 0x0000
ProductType	Ausgang	UINT	Produkttyp des generischen Plug-in- oder Erweiterungsmoduls. Wenn kein Plug-in- oder Erweiterungsmodul vorhanden ist, ProductType = 0x0000
ProductCode	Ausgang	UINT	Produktcode des generischen Plug-in- oder Erweiterungsmoduls. Wenn kein Plug-in- oder Erweiterungsmodul vorhanden ist, ProductCode = 0x0000
ModRevision	Ausgang	UINT	Revisionsinformationen des generischen Plug-in- oder Erweiterungsmoduls. Wenn kein Plug-in- oder Erweiterungsmodul vorhanden ist, ModRevision = 0x0000
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme. Die ENO-Ausgabe wird automatisch im Kontaktplan hinzugefügt.

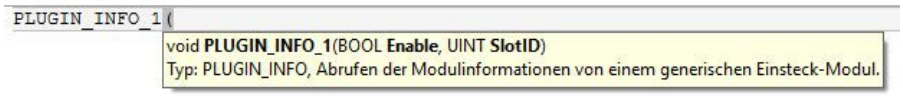
Beispiel für ein PLUGIN_INFO-Funktionsblockdiagramm



Beispiel für einen PLUGIN_INFO-Kontaktplan

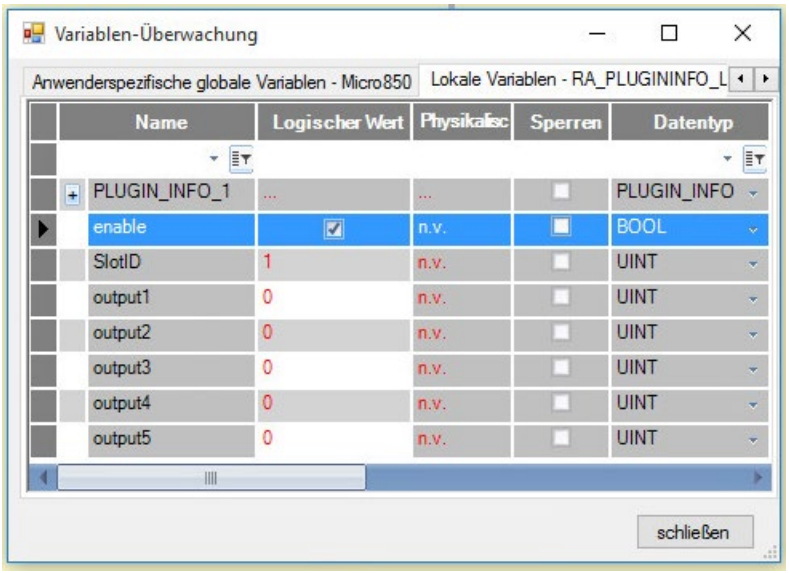


Beispiel für strukturierten PLUGIN_INFO-Text




```
1  SlotID := 1;
2  PLUGIN_INFO_1(enable, SlotID);
3  output1 := PLUGIN_INFO_1.ModID;
4  output2 := PLUGIN_INFO_1.VendorID;
5  output3 := PLUGIN_INFO_1.ProductType;
6  output4 := PLUGIN_INFO_1.ProductCode;
7  output5 := PLUGIN_INFO_1.ModRevision;
```

Ergebnisse



PLUGIN_READ (schreibgeschütztes Plug-in)

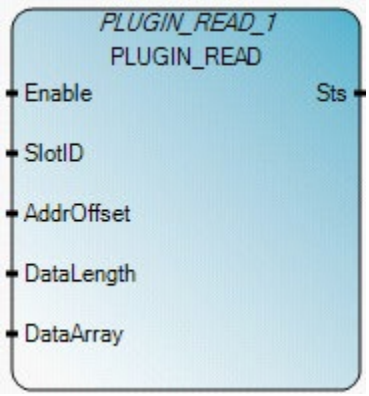
Liest Daten aus einem generischen Plug-in-Modul.

Details zum Betrieb:

- Alle Plug-in-Module außer 2080-MEMBAK-RTC-Module.
- Wenn kein generisches Plug-in-Modul vorhanden ist, werden alle Werte auf null (0) zurückgesetzt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

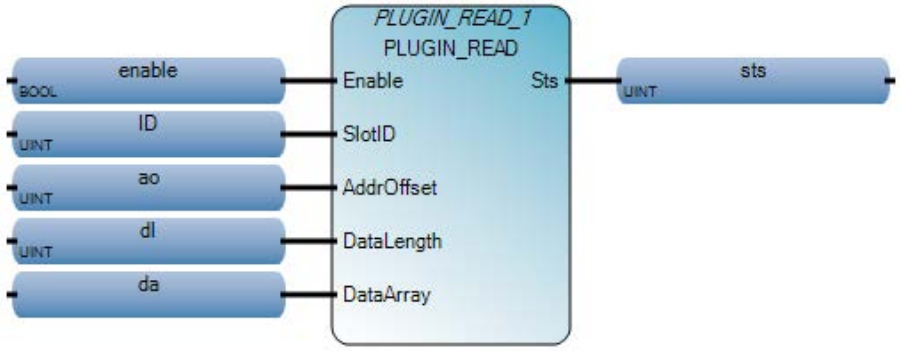
Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



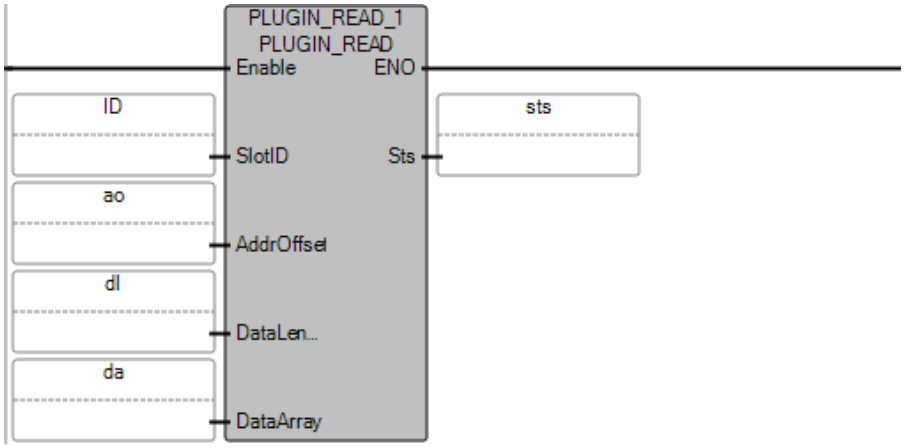
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – UPM-Lesevorgang wird ausgeführt. FALSE – es wird kein Lesevorgang ausgeführt, und die Daten im Datenarray sind ungültig.
SlotID	Eingang	UINT	Nummer des Einsteckplatzes. Schacht-ID = 1,2,3,4,5 (beginnend mit dem äußersten linken Steckplatz = 1).
Offset	Eingang	UINT	Adressabstand der ersten auszulesenden Daten; wird vom ersten Byte des generischen Plug-in-Moduls aus berechnet.
DataLength	Eingang	UINT	Die Anzahl der zu lesenden Byte.
DataArray	Eingang	USINT	Ein Array, das zum Speichern der aus dem generischen Plug-in-Modul gelesenen Daten verwendet wird.
Sts	Ausgang	UINT	Statuscodes für PLUGIN_READ. Statuscodes (Sts): • 0x00 – Funktionsblock nicht aktiviert (kein Vorgang). • 0x01 – Plug-in-Vorgang erfolgreich. • 0x02 – Plug-in-Vorgang schlägt aufgrund ungültiger Schacht-ID fehl. • 0x03 – Plug-in-Vorgang schlägt fehl, da es sich nicht um ein gültiges generisches Plug-in-Modul handelt. • 0x04 – Plug-in-Vorgang schlägt fehl, da sich die zum Betrieb verwendeten Daten außerhalb des Bereichs befinden. • 0x05 – Plug-in-Vorgang schlägt aufgrund eines Paritätsfehlers beim Datenzugriff fehl.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für ein PLUGIN_READ-Funktionsblockdiagramm



Beispiel für einen PLUGIN_READ-Kontaktplan



Beispiel für strukturierten PLUGIN_READ-Text

```
PLUGIN_READ_1(
```

void **PLUGIN_READ_1**(BOOL **Enable**, UINT **SlotID**, UINT **AddrOffset**, UINT **DataLength**, USINT[1..1] **DataArray**)
Typ: PLUGIN_READ, Lesen von Daten aus einem generischen Einsteck-Modul.

```
1  PLUGIN_READ_1(enable, ID, ao, dl, da);
2  sts := PLUGIN_READ_1.Sts;
```

PLUGIN_RESET (Plug-in zurücksetzen)

Setzt jegliche generische Plug-in-Modul-Hardware außer 2080-MEMBAK-RTC-Module zurück. Nach dem Zurücksetzen der Hardware, ist das generische Plug-in-Modul bereit zur Konfiguration und zum Betrieb.

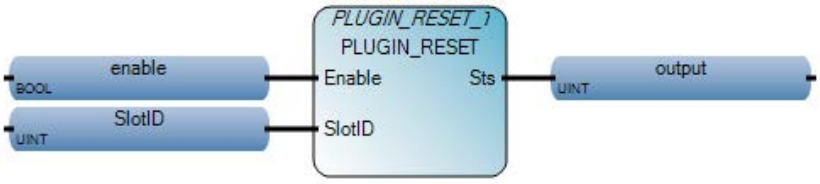
Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.

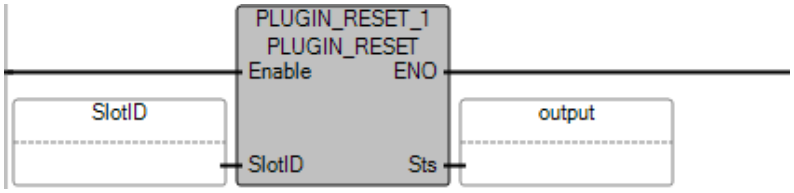
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Plug-in-Reset wird ausgeführt. FALSE – es findet kein Zurücksetzen statt.
SlotID	Eingang	UINT	Nummer des Einsteckplatzes. Schacht-ID = 1,2,3,4,5 (beginnend mit dem äußersten linken Steckplatz = 1).
Sts	Ausgang	UINT	PLUGIN_RESET-Statuscodes. Statuscodes (Sts): • 0x00 – Funktionsblock nicht aktiviert (kein Vorgang). • 0x01 – Plug-in-Vorgang erfolgreich. • 0x02 – Plug-in-Vorgang schlägt aufgrund ungültiger Schacht-ID fehl. • 0x03 – Plug-in-Vorgang schlägt fehl, da es sich nicht um ein gültiges generisches Plug-in-Modul handelt. -2080-MOT-HSC-Modulkonfiguration im High Hochgeschwindigkeitszähler-Anweisungsmodus. • 0x04 – Plug-in-Vorgang schlägt fehl, da sich die zum Betrieb verwendeten Daten außerhalb des Bereichs befinden. • 0x05 – Plug-in-Vorgang schlägt aufgrund eines Paritätsfehlers beim Datenzugriff fehl.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für ein PLUGIN_RESET-Funktionsblockdiagramm



Beispiel für einen PLUGIN_RESET-Kontaktplan



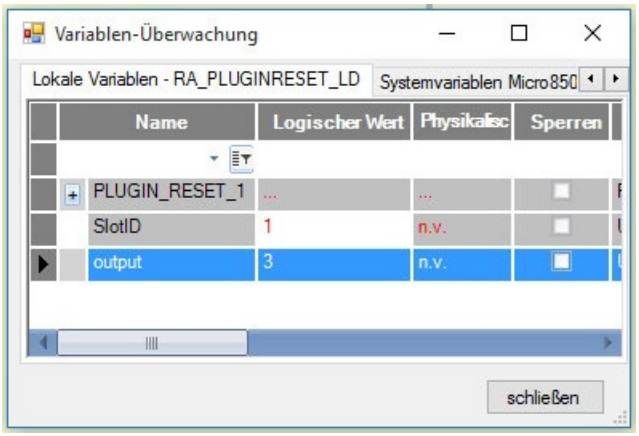
Beispiel für strukturierten PLUGIN_RESET-Text

```
PLUGIN_RESET_1(
```

void **PLUGIN_RESET_1**(BOOL **Enable**, UINT **SlotID**)
Typ: PLUGIN_RESET, Zurücksetzen eines generischen EINSTECK-Moduls (Hardware-Reset).

```
1  SlotID := 1;
2  PLUGIN_RESET_1(enable, SlotID);
3  output := PLUGIN_RESET_1.Sts;
```

Ergebnisse



PLUGIN_WRITE (Plugin schreiben)

Schreibt einen Block von Daten in jegliche generische Plug-in-Modul-Hardware außer 2080-MEMBAK-RTC-Module.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

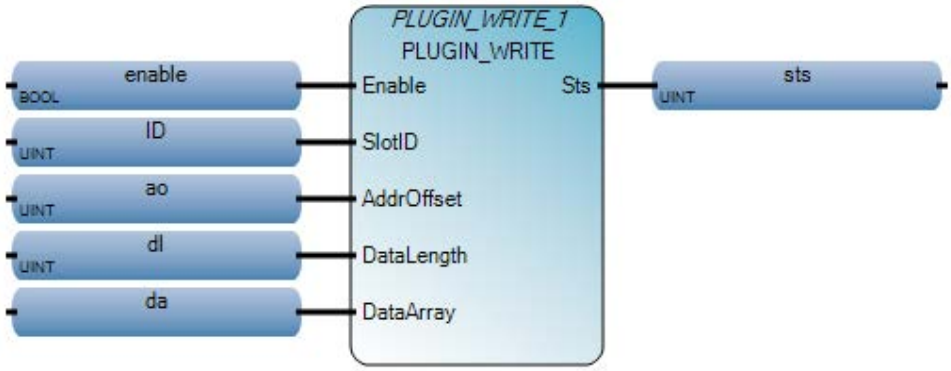
Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

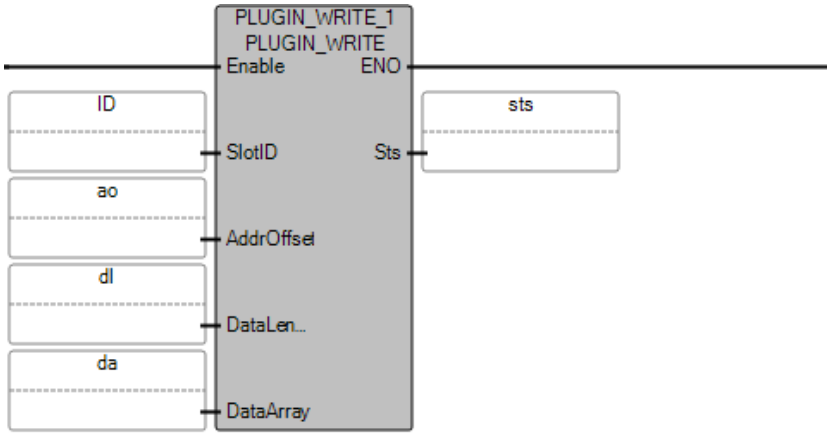
Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Plug-in-Schreibvorgang wird ausgeführt. FALSE – es findet kein Schreibvorgang für Daten statt.
SlotID	Eingang	UINT	Nummer des Einsteckplatzes. Schacht-ID = 1,2,3,4,5 (beginnend mit dem äußersten linken Steckplatz = 1).

Parameter	Parametertyp	Datentyp	Beschreibung
AddrOffset	Eingang	UINT	Adressabstand der ersten zu schreibenden Daten; wird vom ersten Byte des generischen Plug-in-Moduls aus berechnet.
DataLength	Eingang	UINT	Die Anzahl der zu schreibenden Byte.
DataArray	Eingang	USINT	In das generische Plug-in-Modul zu schreibende Daten.
Sts	Ausgang	UINT	PLUGIN_WRITE-Statuscodes. Statuscodes (Sts): • 0x00 – Funktionsblock nicht aktiviert (kein Vorgang). • 0x01 – Plug-in-Vorgang erfolgreich. • 0x02 – Plug-in-Vorgang schlägt aufgrund ungültiger Schacht-ID fehl. • 0x03 – Plug-in-Vorgang schlägt fehl, da es sich nicht um ein gültiges generisches Plug-in-Modul handelt. • 0x04 – Plug-in-Vorgang schlägt fehl, da sich die zum Betrieb verwendeten Daten außerhalb des Bereichs befinden. • 0x05 – Plug-in-Vorgang schlägt aufgrund eines Paritätsfehlers beim Datenzugriff fehl.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für ein PLUGIN_WRITE-Funktionsblockdiagramm



Beispiel für einen PLUGIN_WRITE-Kontaktplan



Beispiel für strukturierten PLUGIN_WRITE-Text

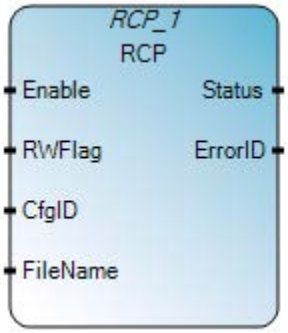
```
1  PLUGIN_WRITE_1(enable, ID, ao, dl, da);
2  sts := PLUGIN_WRITE_1.Sts;
```

RCP (Rezept)

Liest den Wert einer Variable aus der Rezeptdatendatei, die in dem Ordner für Rezeptdatendateien auf der SD-Karte gespeichert ist, und aktualisiert den Wert auf der Laufzeit-Engine. Schreibt den Variablenwert mit der Laufzeit-Engine in die Rezeptdatendatei auf der SD-Karte.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



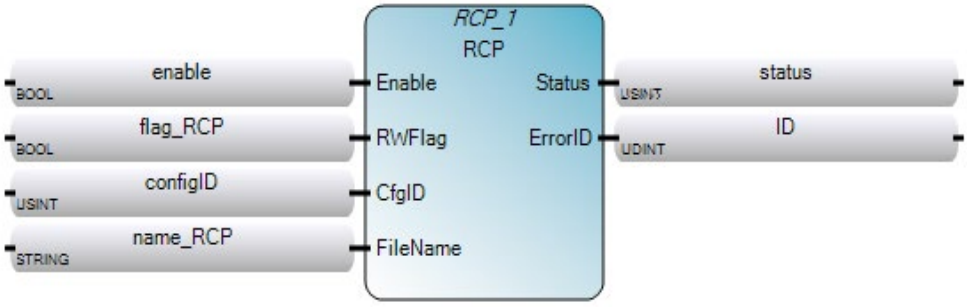
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock für Rezept lesen/schreiben wird aktiviert. TRUE – wenn Anstiegflanke erkannt wird, Rezeptanweisung ausführen, sobald vorheriger Vorgang abgeschlossen ist. FALSE – wenn keine Anstiegflanke erkannt wird, Rezeptanweisung nicht ausführen.
RWFlag	Eingang	BOOL	TRUE – RWFlag (Schreibvorgang). Das Rezept schreibt die Variablenwerte mit der Laufzeit-Engine in eine Rezeptdatendatei auf der SD-Karte. FALSE – RWFlag (Lesevorgang). Das Rezept liest die Variablenwerte von der SD-Karte und aktualisiert die zugehörigen Variablenwerte der Laufzeit-Engine.
CfgID	Eingang	USINT	Rezeptkonfigurations-VA ID-Nummer 1-10.
FileName	Eingang	STRING	Name der Rezeptdatendatei (maximal 30 Zeichen lang).
Status	Ausgang	USINT	Aktueller Status des Rezeptanweisungsblocks. RCP-Statuscodes: • 0 Leerlauf • 1 Ausführung • 2 Erfolg; ohne Fehler abgeschlossen. • 3 Fehler; mit Fehler abgeschlossen.
ErrorID	Ausgang	UDINT	Der numerische RCP-Fehlercode. Die Definitionen sind in RCP-Fehlercodes definiert.

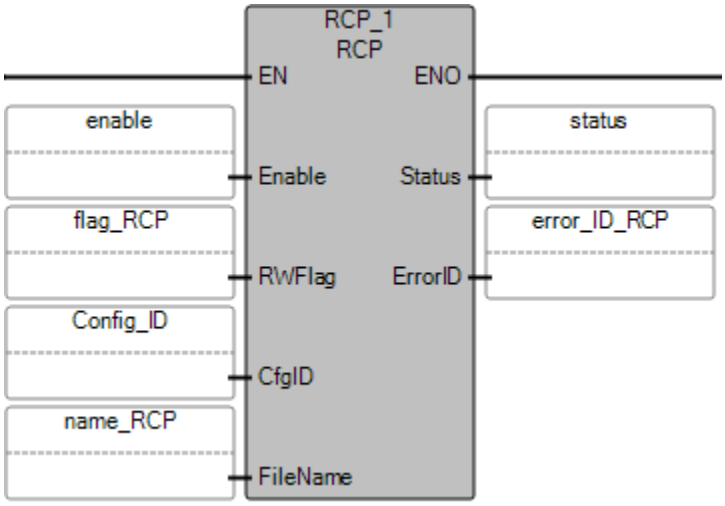
RCP-Fehlercodes

Fehlercode	Fehlername
0	RCP_ERR_NONE
1	RCP_ERR_NO_SDCARD
2	RCP_ERR_DATAFILE_FULL
3	RCP_ERR_DATAFILE_ACCESS SD-Karte wurde identifiziert als: • beschädigt. • voll. • schreibgeschützt.
4	RCP_ERR_CFG_ABSENT
5	RCP_ERR_CFG_ID
6	RCP_ERR_RESOURCE_BUSY
7	RCP_ERR_CFG_FORMAT
8	RCP_ERR_RESERVED Reserviert für künftige mögliche Erweiterungen.
9	RCP_ERR_UNKNOWN
10	RCP_ERR_DATAFILE_NAME
11	RCP_ERR_DATAFOLDER_INVALID
12	RCP_ERR_DATAFILE_ABSENT
13	RCP_ERR_DATAFILE_FORMAT
14	RCP_ERR_DATAFILE_SIZE Rezeptdatendatei ist zu groß (>4 kB).

Beispiel für ein RCP-Funktionsblockdiagramm



Beispiel für ein RCP-Kontaktplan



Beispiel **für strukturierten RCP-Text**

```
RCP_1(
```

void **RCP_1**(BOOL **Enable**, BOOL **RWFlag**, USINT **CfgID**, STRING **FileName**)
Typ: RCP, Speichern/Wiederherstellen der Datenliste in/aus einer SD-Karten-Rezeptdatei.

```
1  RCP_1 (EN, Enable, RWFlag, CfgID, FileName);
2  output := RCP_1.ENO
3  status := RCP_1.Status
4  error_ID_RCP := RCP_1.ErrorID
```

RTC_READ (Lesen der Echtzeituhr)

Liest die Informationen des Echtzeituhr-Moduls (Real-Time Clock, RTC).

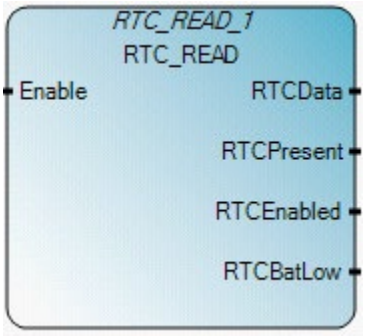
Details zum Betrieb:

- Micro810- oder Micro820-Controller mit integrierter RTC:
  - RTCBatLow ist immer auf null (0) gesetzt.
  - RTCEnabled ist immer auf eins (1) gesetzt.
- Wenn der integrierte RTC seine Ladung/seinen Speicherinhalt aufgrund eines Stromausfalls verloren hat:
  - RTCData wird auf 2000/1/1/0/0/0 gesetzt.
  - RTCEnabled wird auf eins (1) gesetzt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Die Ausgänge werden jedes Mal zurückgesetzt, wenn die Anweisung auf die simulierte Steuerung (2080-LC50-48QWB-SIM) angewendet wird.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

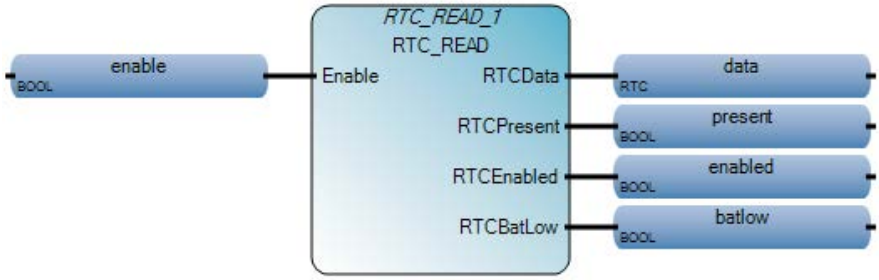
Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – RTC-Informationen lesen. FALSE – Es findet kein Lesevorgang statt, und die ausgegebenen RTC-Daten sind ungültig.
RTCData	Ausgang	RTC	Informationen zu RTC-Daten: JJ/MM/TT, hh/mm/ss, Woche. RTCData-Ausgabe wird mit dem Datentyp „RTC“ definiert.
RTCPresent	Ausgang	BOOL	TRUE - Freilaufende Uhr wird verwendet oder RTC-Hardware ist angeschlossen. FALSE - Freilaufende Uhr wird nicht verwendet oder RTC-Hardware ist nicht angeschlossen.
RTCEnabled	Ausgang	BOOL	TRUE - Freilaufende Uhr wird verwendet oder RTC-Hardware ist aktiviert (Zeitgebung). FALSE - Freilaufende Uhr wird nicht verwendet, RTC-Hardware ist nicht deaktiviert (keine Zeitgebung).
RTCBatLow	Ausgang	BOOL	TRUE - Ladestand der RTC-Batterie niedrig. FALSE - Ladestand der RTC-Batterie nicht niedrig.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Datentyp RTC

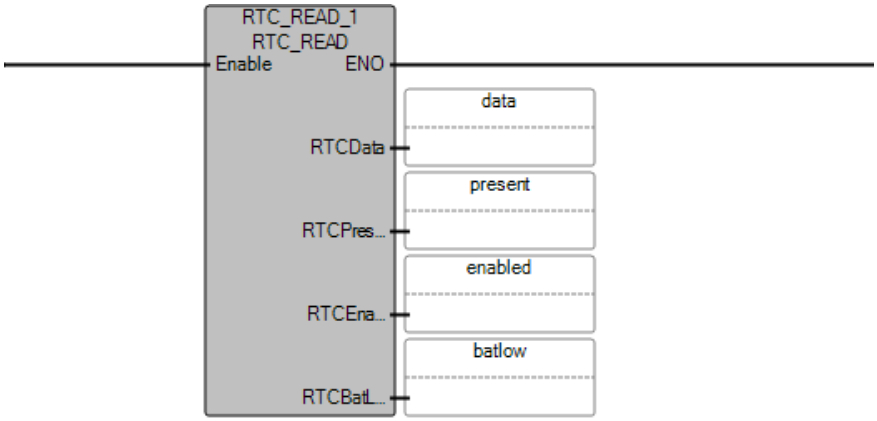
Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „RTC“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Year	UINT	Die Jahreseinstellung für den RTC. 16-Bit-Wert, der gültige Bereich reicht von 2000 (01. Januar, 00:00:00) bis 2098 (31. Dezember, 23:59:59)
Month	UINT	Die Monatseinstellung für den RTC.
Day	UINT	Die Tageseinstellung für den RTC.
Hour	UINT	Die Stundeneinstellung für den RTC.
Minute	UINT	Die Minuteneinstellung für den RTC.
Second	UINT	Die Sekundeneinstellung für den RTC.
DayOfWeek	UINT	Die Wochentagseinstellung für den RTC. Dieser Parameter wird für „RTC_SET“ ignoriert.

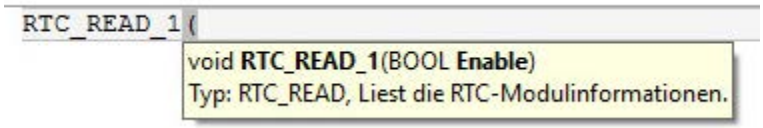
Beispiel für ein RTC_READ-Funktionsblockdiagramm



Beispiel für einen RTC_READ-Kontaktplan



Beispiel für einen strukturierten RTC_READ-Text




```
1  RTC_READ_1(enable);
2  data := RTC_READ_1.RTCData;
3  present := RTC_READ_1.RTCPresent;
4  enabled := RTC_READ_1.RTCEnabled;
5  batlow := RTC_READ_1.RTCBatLow;
```

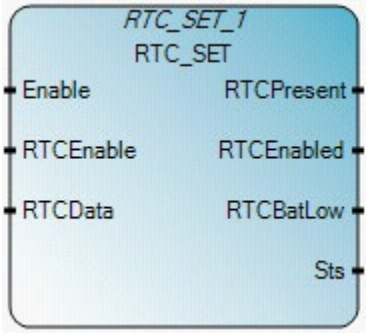
RTC_SET (Echtzeituhr einstellen)

Stellen Sie die Echtzeituhr (RTC) auf die RTC-Modul-Informationen ein.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Die Ausgänge werden jedes Mal zurückgesetzt, wenn die Anweisung auf die simulierte Steuerung (2080-LC50-48QWB-SIM) angewendet wird.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – RTC_SET mit den RTC-Informationen aus der Eingabe ausführen. Typischerweise nur für 1 Programm-Scan beim Update von RTC ausführen. FALSE – RTC_SET nicht ausführen. Auf FALSE einstellen, um RTC normal auszuführen.
RTCEnable	Eingang	BOOL	TRUE - Um RTC mit den angegebenen RTC-Daten zu aktivieren. FALSE - Um RTC zu deaktivieren.
RTCData	Eingang	RTC	RTC-Dateninformationen: JJ/mm/tt, Hh/mm/ss, Woche wie im Datentyp „RTC“ definiert. RTCData wird ignoriert, wenn RTCEnable = 0.
RTCPresent	Ausgang	BOOL	TRUE - Freilaufende Uhr wird verwendet oder RTC-Hardware ist angeschlossen. FALSE - Freilaufende Uhr wird nicht verwendet oder RTC-Hardware ist nicht angeschlossen.
RTCEnabled	Ausgang	BOOL	TRUE - Freilaufende Uhr wird verwendet oder RTC-Hardware ist aktiviert (Zeitgebung). FALSE - Freilaufende Uhr wird nicht verwendet oder RTC-Hardware ist nicht deaktiviert (keine Zeitgebung).
RTCBatLow	Ausgang	BOOL	TRUE - Ladestand der RTC-Batterie niedrig. FALSE - Ladestand der RTC-Batterie nicht niedrig.
Sts	Ausgang	USINT	Der Status des Lesevorgangs. RTC_Set-Statuswerte (Sts): • 0x00 – Funktionsblock nicht aktiviert (kein Vorgang). • 0 x 01 – RTC-Einstellvorgang erfolgreich. • 0 x 02 – RTC-Einstellvorgang nicht erfolgreich.

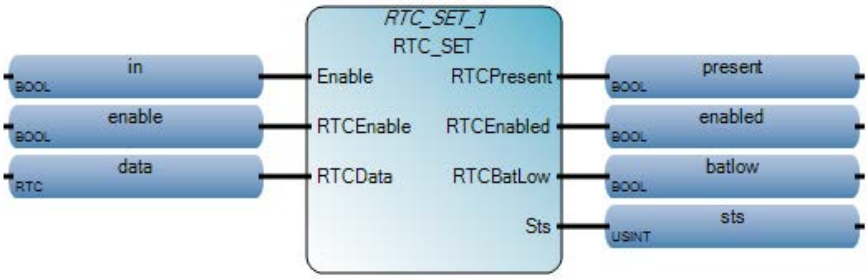
Datentyp RTC

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „RTC“ zu bestimmen.

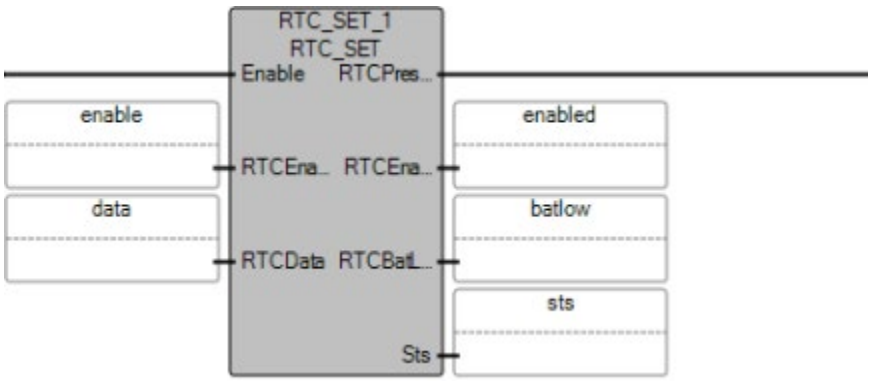
Parameter	Datentyp	Beschreibung
Year	UINT	Die Jahreseinstellung für den RTC. 16-Bit-Wert, der gültige Bereich reicht von 2000 (01. Januar, 00:00:00) bis 2098 (31. Dezember, 23:59:59)
Month	UINT	Die Monatseinstellung für den RTC.

Parameter	Datentyp	Beschreibung
Day	UINT	Die Tageseinstellung für den RTC.
Hour	UINT	Die Stundeneinstellung für den RTC.
Minute	UINT	Die Minuteneinstellung für den RTC.
Second	UINT	Die Sekundeneinstellung für den RTC.
DayOfWeek	UINT	Die Wochentagseinstellung für den RTC. Dieser Parameter wird für „RTC_SET“ ignoriert.

Beispiel für ein RTC_SET-Funktionsblockdiagramm



Beispiel für einen RTC_SET-Kontaktplan



Beispiel für einen strukturierten RTC_SET-Text

```
RTC_SET_1(
```

void **RTC_SET_1**(BOOL **Enable**, BOOL **RTCEnable**, RTC **RTCData**)
Typ: RTC_SET, Einstellen der RTC-Daten im RTC-Modul.

```
1  RTC_SET_1(in, enable, data);
2  present := RTC_SET_1.RTCPresent;
3  enabled := RTC_SET_1.RTCEnabled;
4  batlow := RTC_SET_1.RTCBatLow;
5  sts := RTC_SET_1.Sts;
```

SYS_INFO

Liest den Statusdatenblock für den Controller Micro800.

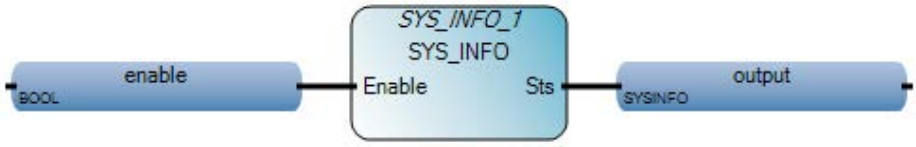
Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

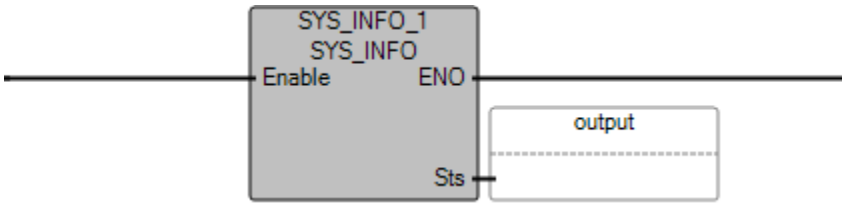
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Lesevorgang ausführen. FALSE – Funktion nicht ausführen.
Sts	Ausgang	SYSINFO	Datenblock mit dem Systemstatus. Der Sts-Ausgang wird im Datentyp SYS_INFO auf seite 391 definiert.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für ein SYS_INFO-Funktionsblockdiagramm



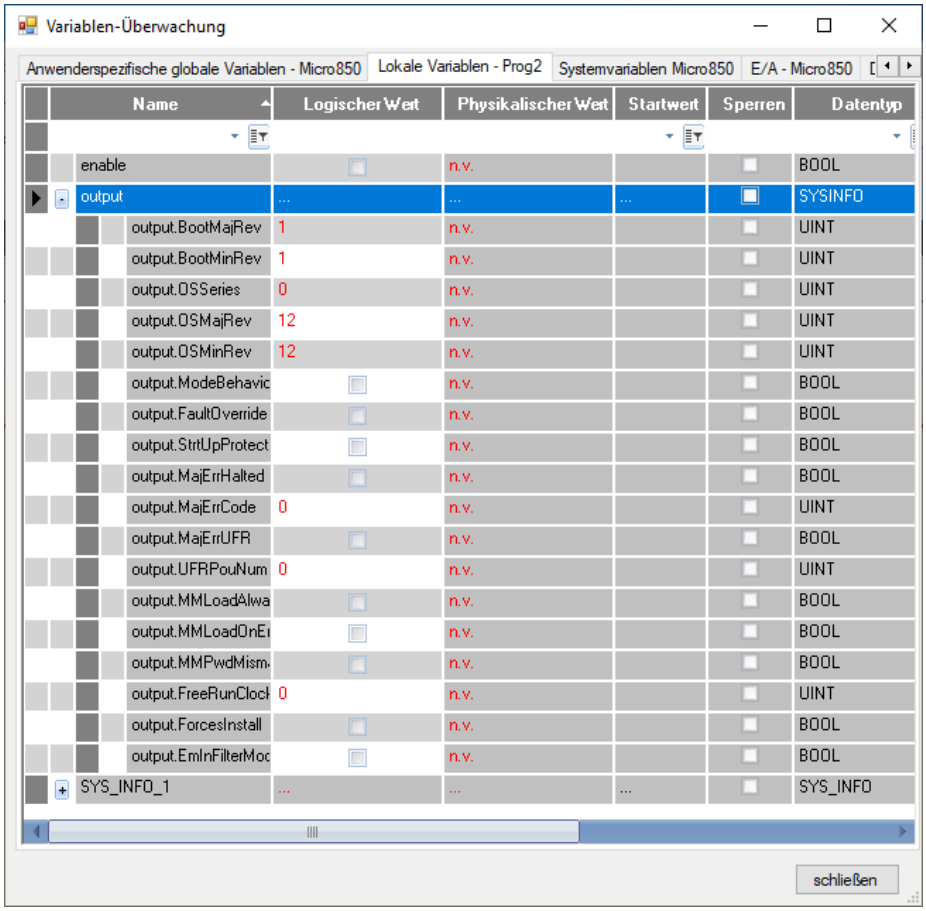
Beispiel für einen SYS_INFO-Kontaktplan



Beispiel für einen strukturierten SYS_INFO-Text

```
1  SYS_INFO_1(enable);
2  output := SYS_INFO_1.Sts;
```

Ergebnisse



Datentyp SYS_INFO

In der folgenden Tabelle wird der Datentyp „SYSINFO“ beschrieben.

Parameter	Datentyp	Beschreibung
BootMajRev	UINT	Große Revision booten.
BootMinRev	UINT	Kleine Revision booten.
Operating System Series	UINT	Betriebssystemserie: 0 kennzeichnet ein Gerät der Serie A. 1 kennzeichnet ein Gerät der Serie B.
OSMajRev	UINT	BS – Große Revision.
OSMinRev	UINT	BS – Kleine Revision.
ModeBehaviour	BOOL	Modusverhalten (TRUE: Beim Einschalten in den Betriebsmodus wechseln).
FaultOverride	BOOL	Fehler-Freifahren (TRUE: Fehler beim Einschalten nicht berücksichtigen).
StrtUpProtect	BOOL	Schutz beim Hochfahren (TRUE: Den Schutz beim Hochfahren beim Einschalten ausführen). Für zukünftige Versionen.
MajErrHalted	BOOL	Schwerwiegender Fehler gestoppt (TRUE: Schwerwiegender Fehler gestoppt).
MajErrCode	UINT	Hauptfehler-Code.
MajErrUFR	BOOL	Schwerer Fehler während Benutzerfehlerroutine. Für zukünftige Versionen.
UFRPouNum	UINT	Programmnummer der Benutzerfehlerroutine.
MMLoadAlways	BOOL	Speichermodul beim Einschalten immer auf Controller zurücksetzen (TRUE: Wiederherstellen).

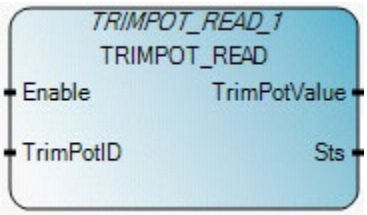
Parameter	Datentyp	Beschreibung
MMLoadOnError	BOOL	Speichermodul beim Einschalten mit Fehler auf Controller zurücksetzen (TRUE: Wiederherstellen).
MMPwdMismatch	BOOL	Nicht übereinstimmendes Kennwort für Speichermodul (TRUE: Kennwörter für Controller und Speichermodul stimmen nicht überein).
FreeRunClock	UINT	Freilaufende Uhr, die sich in Schritten von 100 Mikrosekunden bis 65535 erhöht und dann auf 0 zurückgesetzt wird. Sie können die Uhr einsetzt, die global zugänglich ist, falls Sie mehr Auflösung als den Standardtimer von 1 Millisekunde benötigen. Optionen für Micro830- und Micro850-Controller. Der Wert für <Micro810>-Controller bleibt 0.
ForcesInstall	BOOL	Erzwingt aktiviert (TRUE: Aktiviert).
EMINFilterMod	BOOL	integrierter Filter modifiziert (TRUE: Modifiziert).

TRIMPOT_READ (read trimpot)

Liest den TrimPot-Wert aus einem bestimmten TrimPot.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

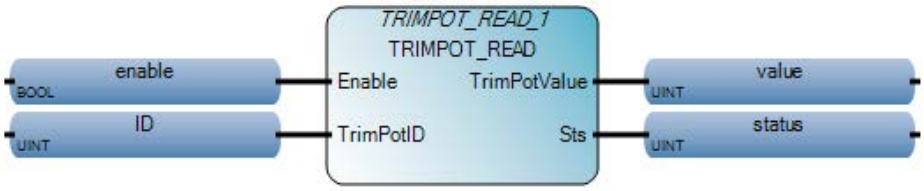
Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850 und Micro870.



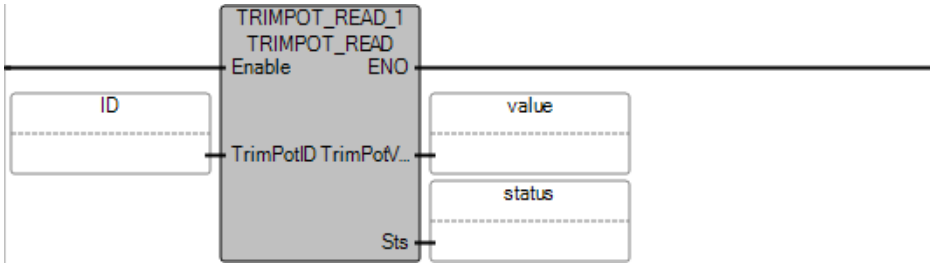
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Ausführen des Trimpot-Lesevorgangs. FALSE – Es findet kein Lesevorgang statt, und der ausgegebene Trimpot-Wert ist ungültig.
TrimPotID	Eingang	UINT	Die ID des zu lesenden TrimPot. TrimPotID ist in den Trimpot-ID-Definitionen auf seite 393 definiert.
TrimPotValue	Ausgang	UINT	Aktueller TrimPot-Wert.
Sts	Ausgang	UINT	Der Status des Trimpot-Lesevorgangs. TRIMPOT-Statuscodes (Sts): • 0x00 – Funktionsblock nicht aktiviert (kein Lese/Schreib-Vorgang). • 0 x 01 – Lese-/Schreib-Vorgang erfolgreich. • 0x02 – Lese/Schreib-Vorgang aufgrund einer ungültigen Trimpot-ID nicht erfolgreich. • 0x03 – Schreibvorgang aufgrund eines Werts außerhalb des Gültigkeitsbereichs nicht erfolgreich.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Beispiel für TRIMPOT-Funktionsblockdiagramm



Beispiel für TRIMPOT-Kontaktplan



Beispiel für strukturierten TRIMPOT-Text

```
TRIMPOT_READ_1(
```

void **TRIMPOT_READ_1**(BOOL **Enable**, UINT **TrimPotID**)
Typ: TRIMPOT_READ, Lesen des Trimpot-Wertes aus einem bestimmten Trimpot.

```
1  TRIMPOT_READ_1(enable, ID);
2  value := TRIMPOT_READ_1.TrimPotValue;
3  status := TRIMPOT_READ_1.Sts;
```

Definition der Trimpot-ID

In der folgenden Tabelle wird die Trimpot-ID-Definition beschrieben, die im TRIMPOT_read-Befehl auf seite 392 verwendet wird.

Auswahl der Ausgabe	Bit	Beschreibung
Definition der Trimpot-ID	15 - 13	Modultyp des Trimpot: • 0x00 - Integriert. • 0x01 - Erweiterung. • 0x02 - Einsteck-Anschluss.
	12 - 8	Schacht-ID des Moduls: • 0x00 - Integriert. • 0x01-0x1F - ID des Erweiterungsmoduls. • 0x01-0x05 - ID des Einsteck-Anschlusses.
	7 - 4	Trimpot-Typ: • 0x00 - Reserviert. • 0x01 - Digitaler Trimpot-Typ 1 (LCD-Modul 1). • 0x02 - Mechanisches Trimpot-Modul 1.

Auswahl der Ausgabe	Bit	Beschreibung
	3 - 0	Trimpot-ID innerhalb des Moduls: • 0x00-0x0F - Integriert. • 0x00-0x07 - ID des Trimpot zur Erweiterung. • 0x00-0x07 - ID des Trimpot für den Einsteck-Anschluss. Die Trimpot-ID beginnt bei 0.

Unterbrechungsanweisungen

Verwenden Sie Unterbrechungsanweisungen, um dem Prozessor zu signalisieren, dass ein Ereignis behandelt werden muss. Gewöhnlich wird das Unterbrechungssignal für Bedingungen mit hoher Priorität verwendet, die eine Unterbrechung des vom Prozessor ausgeführten aktuellen Knotens erfordert.

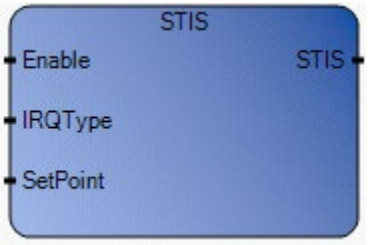
Funktion	Beschreibung
STIS auf seite 395	Startet den Timer für die ausgewählte geplante Benutzerunterbrechung (STI) mit dem Steuerungsprogramm und nicht automatisch.
UIC auf seite 397	Löscht das verlorene Bit für die ausgewählte Benutzerunterbrechung.
UID auf seite 398	Deaktiviert eine bestimmte Benutzerunterbrechung.
UIE auf seite 400	Aktiviert eine bestimmte Benutzereingabe.
UIF auf seite 402	Annulliert oder entfernt eine ausstehende Benutzereingabe.

STIS (Zeitlich festgelegten Start auswählen)

Startet den STI-Timer mit dem Steuerungsprogramm und nicht automatisch.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

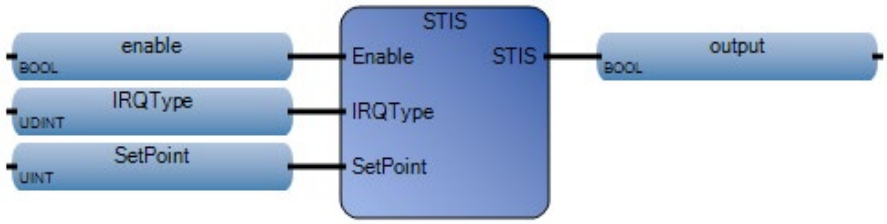


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

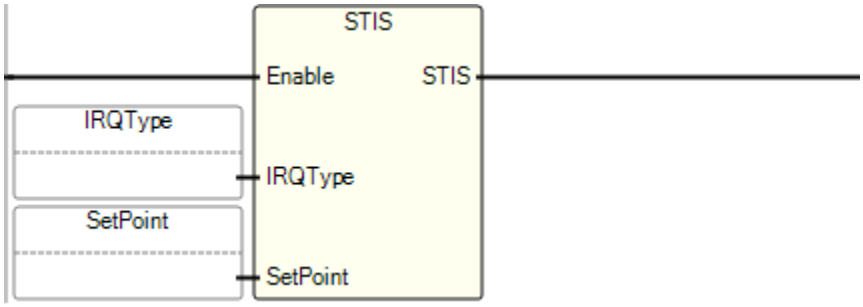
Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisung aktivieren. TRUE – Starten des STI-Timers über das Steuerprogramm. FALSE – Funktion nicht ausführen.
IRQType	Eingang	UDINT	Verwenden Sie die STI-vordefinierten Datenworte. - IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3

Parameter	Parametertyp	Datentyp	Beschreibung
SetPoint	Eingang	UINT	Die Zeitdauer (in ms), die vor der Ausführung des auswählbaren, zeitlich festgelegten Interrupts verstreichen muss. Der Wert 0 deaktiviert die STIS-Funktion. Ein Wert zwischen 1 und 65535 aktiviert die STIS-Funktion.
STIS	Ausgang	BOOL	Strompfadstatus (entspricht „Enable“).

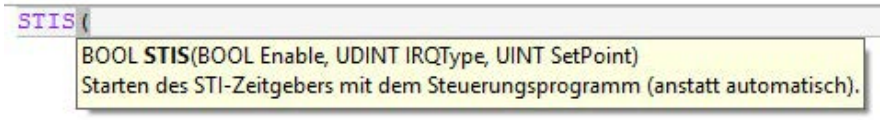
Beispiel für ein STIS-Funktionsblockdiagramm



Beispiel für einen STIS-Kontaktplan



Beispiel für strukturierten STIS-Text

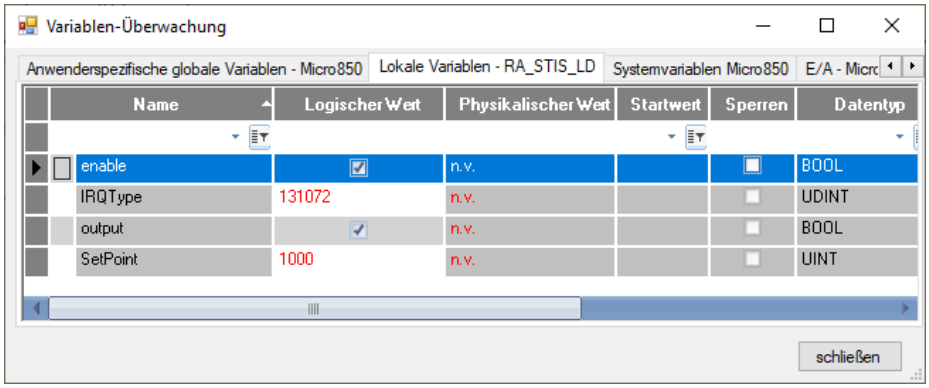



```
1  enable := TRUE;
2  IRQType := IRQ_STI2;
3  SetPoint := 1000;
4  output := STIS(enable, IRQType, SetPoint);
```

(* ST-Äquivalenz: *)

TESTOUTPUT := STIS(TESTENABLE, 2, 1000) ;

Ergebnisse

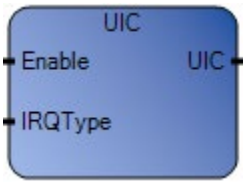


UIC (Interrupt-Lost-Bit löschen)

Löscht das Bit für „Interrupt verloren“ für die ausgewählten Benutzerunterbrechungen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

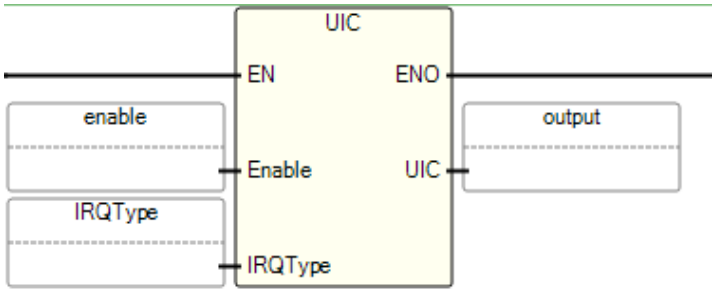


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung	
Enable	Eingang	BOOL	Anweisung aktivieren. TRUE – Startet den-Bitlöschvorgang FALSE – Funktion nicht ausführen.	
IRQType	Eingang	UDINT	Verwenden Sie die STI-vordefinierten Datenworte. - IRQ_EII0 - IRQ_EII1 - IRQ_EII2 - IRQ_EII3 - IRQ_EII4 - IRQ_EII5 - IRQ_EII6 - IRQ_EII7 - IRQ_HSC0 - IRQ_HSC1 - IRQ_HSC2	- IRQ_HSC3 - IRQ_HSC4 - IRQ_HSC5 - IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3 - IRQ_UFR - IRQ_UPM0 - IRQ_UPM1 - IRQ_UPM2 - IRQ_UPM3 - IRQ_UPM4
UIC	Ausgang	BOOL	Strompfadstatus (entspricht „Enable“).	

Beispiel für ein UIC-Funktionsblockdiagramm

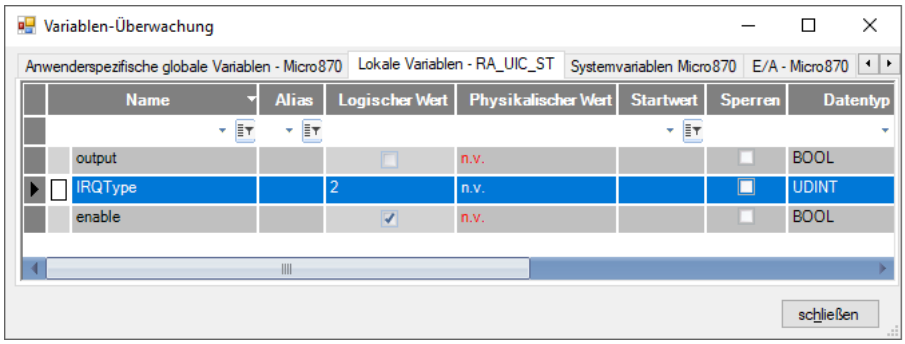
Beispiel für einen UIC-Kontaktplan



Beispiel für einen strukturierten UIC-Text

```
1  enable := TRUE;
2  IRQType := 2;
3  output := UIC (enable, IRQType);
```

Ergebnisse



UID (Interrupt deaktivieren)

Deaktiviert die ausgewählten Benutzerunterbrechungen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

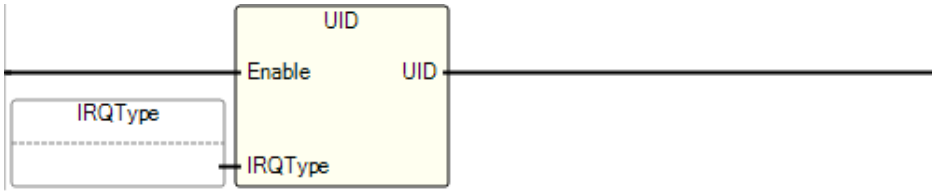
Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

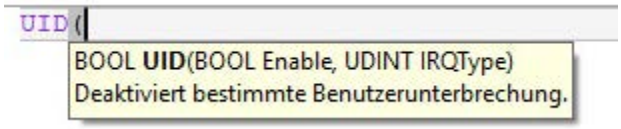
Parameter	Parametertyp	Datentyp	Beschreibung	
Enable	Eingang	BOOL	Anweisung aktivieren. TRUE – Deaktivierungsvorgang starten. FALSE – Funktion nicht ausführen.	
IRQType	Eingang	UDINT	Verwenden Sie die STI-vordefinierten Datenworte. - IRQ_EII0 - IRQ_EII1 - IRQ_EII2 - IRQ_EII3 - IRQ_EII4 - IRQ_EII5 - IRQ_EII6 - IRQ_EII7 - IRQ_HSC0 - IRQ_HSC1 - IRQ_HSC2	- IRQ_HSC3 - IRQ_HSC4 - IRQ_HSC5 - IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3 - IRQ_UFR - IRQ_UPM0 - IRQ_UPM1 - IRQ_UPM2 - IRQ_UPM3 - IRQ_UPM4
UID	Ausgang	BOOL	Strompfadstatus (entspricht „Enable“).	

Beispiel für ein UID-Funktionsblockdiagramm

Beispiel für einen UID-Kontaktplan



Beispiel für einen strukturierten UID-Text

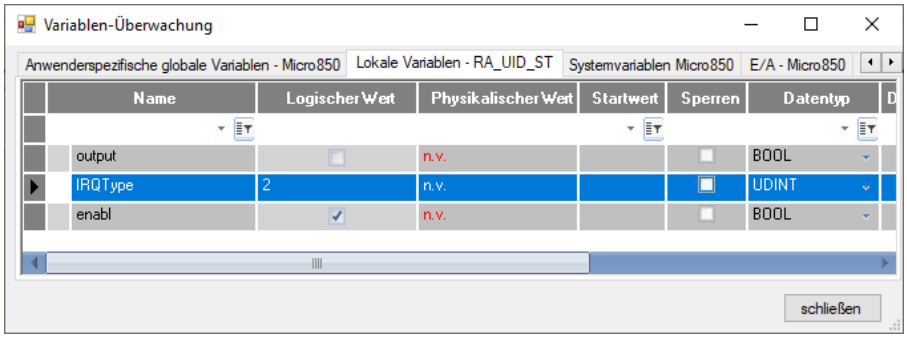



```
1  enable := TRUE;
2  IRQType := 2;
3  output := UID(enable, IRQType);
```

(* ST-Äquivalenz: *)

TESTOUTPUT := UID(TESTENABLE, 2) ;

Ergebnisse



UIE (Interrupt aktivieren)

Aktiviert eine Benutzerunterbrechung.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

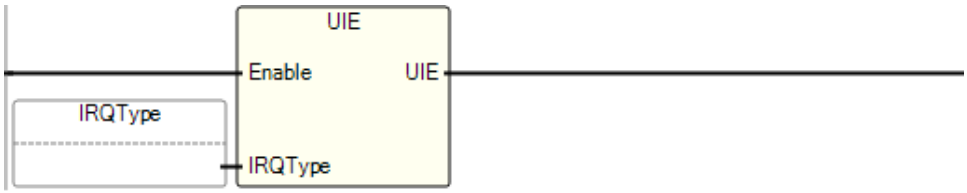
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung	
Enable	Eingang	BOOL	Anweisung aktivieren. TRUE – Aktivierungsvorgang starten. FALSE – Funktion nicht ausführen.	
IRQType	Eingang	UDINT	Verwenden Sie die STI-vordefinierten Datenworte. - IRQ_EII0	- IRQ_HSC3 - IRQ_HSC4 - IRQ_HSC5

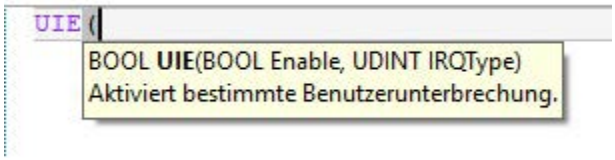
Parameter	Parametertyp	Datentyp	Beschreibung	
			- IRQ_EII1 - IRQ_EII2 - IRQ_EII3 - IRQ_EII4 - IRQ_EII5 - IRQ_EII6 - IRQ_EII7 - IRQ_HSC0 - IRQ_HSC1 - IRQ_HSC2	- IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3 - IRQ_UFR - IRQ_UPM0 - IRQ_UPM1 - IRQ_UPM2 - IRQ_UPM3 - IRQ_UPM4
UIE	Ausgang	BOOL	Strompfadstatus (entspricht „Enable“).	

Beispiel für ein UIE-Funktionsblockdiagramm

Beispiel für einen UIE-Kontaktplan



Beispiel für einen strukturierten UIE-Text

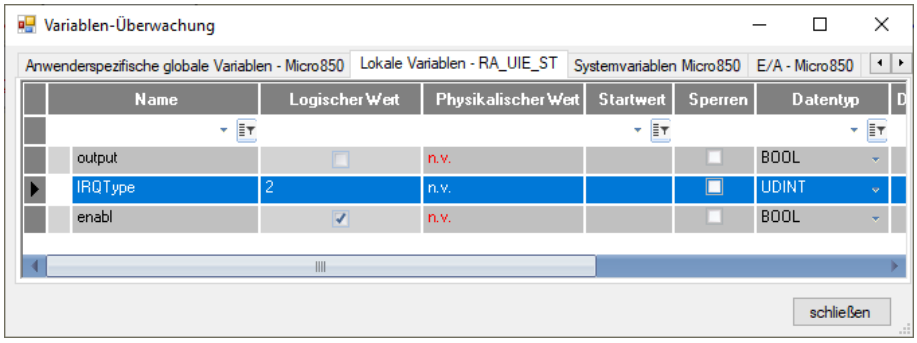



```
1  enable := TRUE;
2  IRQType := 2;
3  output := UIE(enable, IRQType);
```

(* ST-Äquivalenz: *)

TESTOUTPUT := UIE(TESTENABLE, 2) ;

Ergebnisse



UIF (Ausstehenden Interrupt annullieren)

Leert oder entfernt eine ausstehende Benutzerunterbrechung.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

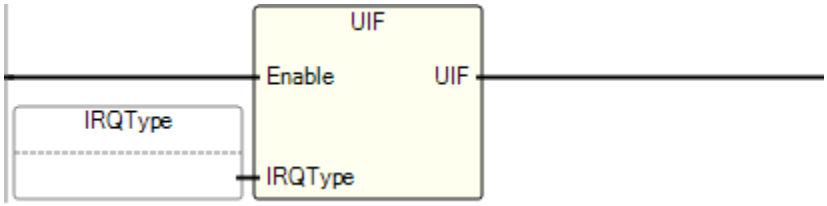
Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

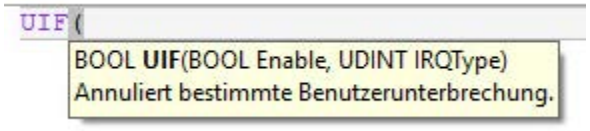
Parameter	Parametertyp	Datentyp	Beschreibung	
Enable	Eingang	BOOL	Anweisung aktivieren. TRUE – UIF-Vorgang starten. FALSE – Funktion nicht ausführen.	
IRQType	Eingang	UDINT	Verwenden Sie die STI-vordefinierten Datenworte. - IRQ_EII0 - IRQ_EII1 - IRQ_EII2 - IRQ_EII3 - IRQ_EII4 - IRQ_EII5 - IRQ_EII6 - IRQ_EII7 - IRQ_HSC0 - IRQ_HSC1 - IRQ_HSC2	- IRQ_HSC3 - IRQ_HSC4 - IRQ_HSC5 - IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3 - IRQ_UFR - IRQ_UPM0 - IRQ_UPM1 - IRQ_UPM2 - IRQ_UPM3 - IRQ_UPM4
UIF	Ausgang	BOOL	Strompfadstatus (entspricht „Enable“).	

Beispiel für ein UIF-Funktionsblockdiagramm

Beispiel für einen UIF-Kontaktplan



Beispiel für einen strukturierten UIF-Text

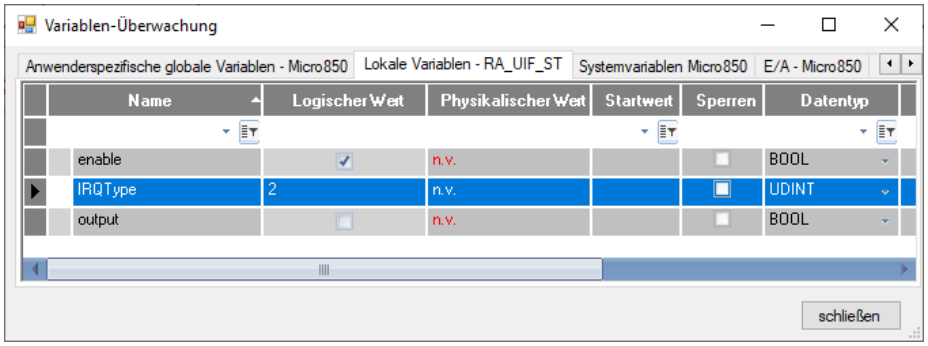



```
1  enable := TRUE;
2  IRQType := 2;
3  output := UIF(enable, IRQType);
```

(* ST-Äquivalenz: *)

TESTOUTPUT := UIF(TESTENABLE, 2) ;

Ergebnisse



Anweisungen für Bewegungssteuerung

Verwenden Sie die Anweisungen für Bewegungssteuerung zum Programmieren und Entwerfen der Bewegung einer bestimmten Achse. Bewegungssteuerung erfordert Connected Components Workbench 2.0 oder höher.

Tipp:

- Administrative Anweisungen unterstützen PTO- und Feedback Bewegung.
- Die Anweisungen für Bewegungssteuerung, die eine FB-Achse unterstützen, sind: MC_ReadActualPosition und MC_ReadActualVelocity.

Anweisung	Beschreibung
Administrativ	
MC_AbortTrigger auf seite 416	Bricht Bewegungsfunktionsblöcke ab, die mit Auslöseereignissen verbunden sind.
MC_Power auf seite 435	Steuert die Leistungsstufe (EIN oder AUS).
MC_ReadAxisError auf seite 445	Liest die Achsfehler, die in keinem Zusammenhang mit den Bewegungssteuerungs-Funktionsblöcken stehen.
MC_ReadBoolParameter auf seite 449	Gibt den Wert eines anbieterspezifischen Parameters vom Typ BOOL zurück.
MC_ReadParameter auf seite 451	Gibt den Wert eines herstellerspezifischen Parameters vom Typ REAL zurück.
MC_ReadStatus auf seite 453	Gibt den Achsstatus hinsichtlich der aktuell erfolgenden Bewegung zurück.
MC_Reset auf seite 457	Setzt den Achszustand von „ErrorStop“ auf „StandStill“. Dies geschieht durch das Zurücksetzen aller internen achsbezogenen Fehler.
MC_SetPosition auf seite 460	Verschiebt das Koordinatensystem einer Achse durch Bearbeiten der tatsächlichen Position.
MC_TouchProbe auf seite 465	Zeichnet eine Achsenposition bei einem Auslöseereignis auf.
MC_WriteBoolParameter auf seite 469	Ändert den Wert eines anbieterspezifischen Parameters vom Typ BOOL.
MC_WriteParameter auf seite 472	Ändert den Wert eines anbieterspezifischen Parameters vom Typ REAL.
Bewegung	
MC_Halt auf seite 418	Befiehlt das Anhalten einer gesteuerten Bewegung unter normalen Betriebsbedingungen.
MC_Home auf seite 421	Befiehlt der Achse, die Sequenz <*search home*> auszuführen.
MC_MoveAbsolute auf seite 424	Befiehlt eine gesteuerte Bewegung zu einer angegebenen absoluten Position.
MC_MoveRelative auf seite 428	Befiehlt eine gesteuerte Bewegung eines angegebenen Abstands im Verhältnis zur tatsächlichen Position zum Zeitpunkt der Ausführung.
MC_MoveVelocity auf seite 431	Befiehlt eine gesteuerte Endlosbewegung mit einer angegebenen Geschwindigkeit.
MC_ReadActualPosition auf seite 439	Gibt die tatsächliche Position der Feedbackachse zurück. Erfordert Connected Components Workbench 8 oder höher.
MC_ReadActualVelocity auf seite 442	Gibt die tatsächliche Geschwindigkeit der Feedbackachse zurück. Erfordert Connected Components Workbench 8 oder höher.
MC_Stop auf seite 462	Befiehlt einen kontrollierten Bewegungsstopp und ändert den Zustand der Achse in „Stoppen“.

Allgemeine Regeln für Bewegungssteuerungs-Funktionsblöcke

Die allgemeinen Regeln für die Achssteuerungs-Funktionsblöcke auf seite 405 des Micro800-Controllers folgen den Spezifikationen der PLCopen-Achssteuerung. Die nachfolgende Tabelle enthält allgemeine Regeln zur Schnittstelle der Bewegungssteuerungs-Funktionsblöcke.

Regel gültig für	Regel
Eingabeparameter	**Mit Ausführung**: Die Parameter werden mit der Anstiegsflanke der Ausführungseingabe verwendet. Ändern Sie zum Modifizieren eines Parameters den Eingabeparameter und lösen Sie die Bewegung erneut aus. Wenn eine Instanz eines Funktionsblocks einen neuen Ausführungsbefehl empfängt, bevor sie abgeschlossen wird (als eine Reihe von Befehlen in derselben Instanz), wird der neue Ausführungsbefehl ignoriert und die zuvor ausgegebene Anweisung setzt die Ausführung fort. **Mit Aktivierung**: Die Parameter werden mit der Anstiegsflanke der Aktivierungseingabe verwendet und können fortlaufend geändert werden.
Fehlende Eingabeparameter	Fehlende Eingaben werden während der Kompilierung der Benutzeranwendung erfasst. Auf Controller-Ebene werden keine Fehler bezüglich fehlender Eingaben behandelt.
Überschreiten von Anwendungsgrenzen durch Eingaben	Wenn ein Funktionsblock mit Parametern angefordert wird und dies zu einer Verletzung der Anwendungsgrenzen führt, generiert die Instanz des Funktionsblocks einen Fehler. In diesem Fall wird die Fehlerausgabe mit „On" (Ein) gekennzeichnet und die Ausgabefehler-ID gibt die entsprechenden Fehlerinformationen an. In den meisten Fällen bleibt der Controller im Betriebsmodus, und es wird kein Bewegungsfehler als schwerwiegender Controller-Fehler gemeldet.
Vorzeichenregeln für Eingaben	Die Eingaben „Beschleunigung" (Acceleration), „Verzögerung" (Deceleration) und „Stoß" (Jerk) haben immer positive Werte. Die Eingaben „Geschwindigkeit" (Velocity), „Position" und „Abstand" (Distance) können sowohl positive als auch negative Werte haben.
Position im Vergleich zu Abstand	Die Position ist ein in einem Koordinatensystem definierter Wert. Der Abstand ist ein relativer Messwert im Zusammenhang mit technischen Einheiten. Der Abstand ist die Differenz zwischen zwei Positionen.
Positions-/Abstandseingabe	In Micro800-Controllern werden nur lineare Bewegungen unterstützt. Beim Funktionsblock „MC_MoveAbsolute" ist die Positionseingabe die absolute Position, die an die Achse ausgegeben wird. Bei MC_MoveRelative ist die Abstandseingabe die relative Position (eine aktuelle Achsenposition von 0 vorausgesetzt) zur aktuellen Position.
Geschwindigkeitseingabe	„Geschwindigkeit" kann ein Wert mit Vorzeichen sein, aber auch die Richtungseingabe dazu verwenden, um auch das Vorzeichen der Geschwindigkeit zu definieren (negative Geschwindigkeit x negative Richtung = positive Geschwindigkeit). Der E-Parameter „Direction" (Richtung) bezieht sich aus Kompatibilitätsgründen auf die Ein- und Ausgabe der Geschwindigkeit.
Richtungseingabe	Bei Abstandsbewegungen (Position) mit definierter Zielposition (absolut oder relativ) ist die Bewegungsrichtung eindeutig. Die Richtungseingabe für die Abstandsbewegung wird ignoriert. Bei Geschwindigkeitsbewegungen kann der Richtungseingabewert 1 (positive Richtung), 0 (aktuelle Richtung) oder -1 (negative Richtung) sein. Bei anderen Werten wird nur das jeweilige Vorzeichen berücksichtigt. Beispiel: -3 kennzeichnet eine negative Richtung, +2 kennzeichnet eine positive Richtung usw. Bei Geschwindigkeitsbewegungen (MC_MoveVelocity) bestimmt das Vorzeichen von (Geschwindigkeit x Richtung) die tatsächliche Bewegungsrichtung, wenn der Wert nicht 0 ist. Beispiel: Wenn Geschwindigkeit x Richtung = +300, ist die Richtung positiv.
Beschleunigungs-, Verzögerungs- und Stoßeingaben	• Verzögerungs- und Beschleunigungseingaben müssen einen positiven Wert haben. Wenn die Verzögerung oder Beschleunigung einen nicht-positiven Wert hat, meldet der Funktionsblock einen Fehler (Fehler-ID: MC_FB_ERR_RANGE). • Die Stoßeingabe muss einen nicht-negativen Wert haben. Wenn der Stoß einen negativen Wert hat, meldet der Funktionsblock einen Fehler (Fehler-ID: MC_FB_ERR_RANGE). • Wenn „Maximaler Stoß" mit null konfiguriert wurde, müssen alle Stoß-Parameter für den Bewegungssteuerungs-Funktionsblock einschließlich der Stoßeinstellung für „MC_Stop" mit 0 konfiguriert sein. Andernfalls meldet der Funktionsblock einen Fehler (Fehler-ID: MC_FB_ERR_RANGE). • Wenn „Stoß" auf einen Nicht-Nullwert gesetzt wurde, wird das S-Kurvenprofil generiert. Ist „Stoß" auf 0 gesetzt, wird das trapezförmige Profil generiert. • Die Konfiguration für die Stoßausgangsstellung ist nicht auf die Konfiguration für maximalen Stoß beschränkt. • Falls der Bewegungsmotor das von den dynamischen Eingabeparametern vorgeschriebene Bewegungsprofil nicht generieren kann, meldet der Funktionsblock einen Fehler (Fehler-ID: MC_FB_ERR_PROFILE).

Regel gültig für	Regel
Ausgabeausschließlichkeit	**Mit Ausführung**: Wenn der Ausführungsparameter TRUE ist, muss eine der Ausgaben „Beschäftigt" (Busy), „Fertig" (Done), „Fehler" (Error) oder „Befehl abgebrochen" (CommandAborted) ebenfalls TRUE sein. In einem Funktionsblock kann jeweils nur eine Ausgabe den Wert TRUE haben. Es wird jeweils nur eine der Ausgaben „Aktiv" (Active), „Fehler" (Error), „Fertig" (Done) und „Befehl abgebrochen" (CommandAborted) festgelegt. **Mit Aktivierung**: In einem Funktionsblock mit gültigen oder fehlerhaften Ausgaben kann jeweils nur eine Ausgabe den Wert TRUE haben.
Ausgabezustand	**Mit Ausführung**: Die Ausgaben „Fertig" (Done), „Fehler" (Error), „Fehler-ID" (ErrorID) und „Befehl abgebrochen" (CommandAborted) werden mit der Abstiegsflanke der Anweisung „Ausführen" zurückgesetzt. Die Abstiegsflanke von „Ausführen" stoppt jedoch nicht die Ausführung des eigentlichen Funktionsblock oder wirkt sich darauf aus. Selbst bei Zurücksetzen von „Ausführen" vor Abschluss des Funktionsblocks sind die entsprechenden Ausgaben für mindestens einen Zyklus festgelegt. Wenn eine Instanz eines Funktionsblocks einen neuen Ausführungsbefehl empfängt, bevor sie abgeschlossen wird (als eine Reihe von Befehlen in derselben Instanz), wird der neue Ausführungsbefehl ignoriert und die zuvor ausgegebene Anweisung setzt die Ausführung fort. **Mit Aktivierung**: Die Ausgaben „Gültig" (Valid), „Aktiviert" (Enabled), „Beschäftigt" (Busy), „Fehler" (Error) und „Fehler-ID" (ErrorID) werden so schnell wie möglich mit der Abstiegsflanke von „Aktivieren" (Enable) zurückgesetzt.
Verhalten der Ausgabe „Fertig" (Done)	Die Ausgabe „Fertig" (Done) wird gesetzt, wenn die angeforderte Aktion erfolgreich abgeschlossen wurde. Wenn mehrere Funktionsblöcke nacheinander in derselben Achse ausgeführt werden, gilt Folgendes: • Wenn eine Bewegung einer Achse durch eine andere Bewegung derselben Achse unterbrochen wird, ohne das finale Ziel erreicht zu haben, wird im ersten Funktionsblock nicht die Ausgabe „Fertig" (Done) gesetzt.
Verhalten der Ausgabe „Beschäftigt" (Busy)	Jeder Funktionsblock kann die Ausgabe „Beschäftigt" (Busy) haben. Sie zeigt an, dass der Funktionsblock entweder noch nicht beendet ist (bei Funktionsblöcken mit Ausführungseingabe) oder nicht ausgeführt wird und neue Ausgabewerte erwartet werden können (bei Aktivierungseingabe). „Beschäftigt" wird an die Anstiegsflanke von „Ausführen" gesetzt und zurückgesetzt, wenn eine der Ausgaben „Fertig" (Done), „Abgebrochen" (Aborted) oder „Fehler" (Error) gesetzt wird. Dieser Funktionsblock sollte mindestens so lange in der aktiven Schleife des Anwendungsprogramms belassen werden, wie für „Beschäftigt" der Wert TRUE zutrifft, da sich die Ausgaben noch ändern können. Beschäftigte Funktionsblöcke mit derselben Instanz können erst ausgeführt werden, wenn diese nicht mehr beschäftigt ist. Funktionsblöcke mit unterschiedlichen Instanzen können den Funktionsblock überschreiben, der derzeit ausgeführt wird.
Verhalten der Ausgabe „Befehl abgebrochen" (CommandAborted)	Die Ausgabe „Befehl abgebrochen" (CommandAborted) wird gesetzt, wenn eine angeforderte Bewegung durch einen anderen Bewegungsbefehl unterbrochen wird. Das Rücksetzverhalten von Befehl abgebrochen (CommandAborted) ist vergleichbar mit dem der Ausgabe „Fertig" (Done). Wenn Befehl abgebrochen (CommandAborted) auftritt, werden andere Ausgabesignale wie z.B. „In Umlaufgeschwindigkeit" (InVelocity) zurückgesetzt.
Ausgabe „Aktiv" (Active)	Die Ausgabe „Aktiv" (Active) ist bei gepufferten Funktionsblöcken erforderlich und wird gesetzt, sobald der Funktionsblock die Bewegungssteuerung der entsprechenden Achse übernimmt. Im ungepufferten Modus können die Ausgaben „Aktiv" (Active) und „Beschäftigt" (Busy) denselben Wert haben.
Status „Aktivieren" (Enable) und „Gültig" (Valid)	Die Eingabe „Aktivieren" (Enable) ist mit der Ausgabe „Gültig" (Valid) verbunden. „Aktivieren" (Enable) bezieht sich auf die jeweilige Ebene und „Gültig" (Valid) zeigt an, dass eine gültige Reihe von Ausgaben im Funktionsblock zur Verfügung steht. Die Ausgabe „Gültig" (Valid) hat den Wert TRUE, solange ein gültiger Ausgabewert zur Verfügung steht und die Eingabe „Aktivieren" (Enable) ebenfalls den Wert TRUE hat. Der relevante Ausgabewert kann aktualisiert werden, solange die Eingabe „Aktivieren" (Enable) den Wert TRUE hat. Liegt ein Funktionsblockfehler vor, ist die Ausgabe nicht gültig („Gültig" wird auf FALSE gesetzt). Wenn die Fehlerbedingung verschwindet, werden die Werte wieder angezeigt und die Ausgabe „Gültig" (Valid) wird erneut gesetzt.
Behandlung von Ausgabefehlern	**Ausgaben zum Definieren von Fehlern** Alle Blöcke haben folgende zwei Ausgaben zu Fehlern, die während der Ausführung auftreten können: • Fehler – Anstiegsflanke von „Fehler" meldet, dass während der Ausführung des Funktionsblocks ein Fehler aufgetreten ist. • Fehler-ID – Fehlernummer.

Regel gültig für	Regel
	Die Ausgaben „Fertig" (Done) und „In Umlaufgeschwindigkeit" (InVelocity) betreffen einen erfolgreichen Abschluss und schließen somit die Ausgabe „Fehler" (Error) aus. Instanzfehler führen nicht immer zu einem Achsenfehler (löst Fehlerstopp der Achse aus).
	So werden Fehlerausgaben zurückgesetzt • Die Fehlerausgaben der relevanten Funktionsblöcke werden mit der fallenden Flanke von „Ausführen" und „Aktivieren" zurückgesetzt. • Die Fehlerausgaben eines Funktionsblocks mit „Aktiveren" können während des Betriebs zurückgesetzt werden (ohne Zurücksetzen von „Aktivieren").
	Fehlertypen • Funktionsblocklogik (beispielsweise Parameter außerhalb des Bereichs, Zustand „Versuchte Maschinenverletzung" usw.) • HW- oder SW-Grenze • Mechanismus/Motor • Laufwerk
Benennungskonventionen ENUM-Typen	Aufgrund der Benennungseinschränkungen im IEC-Standard zur Eindeutigkeit von Variablennamen wird für die ENUMs der „mc"-Verweis zum PLCopen-Bewegungssteuerungs-Namespace verwendet. Auf diese Weise wird ein Konflikt durch die Verwendung der ENUM-Typen 'positiv' und 'negativ' beispielsweise für Variablen mit diesem Namen im restlichen Verlauf des Projekts vermeiden, da sie „mcPositive" und „mcNegative" benannt werden.

Parameterdetails von Bewegungssteuerungs-Funktionsblock

Die nachfolgenden Themen stellen detaillierte Informationen zu Bewegungssteuerungsparametern bereit, die für alle Funktionsblöcke der Bewegungssteuerung relevant sind.

Bewegungssteuerungs-Achsenstatus auf seite 408

Parameternummern von Bewegungssteuerungs-Funktionsblock auf seite 410

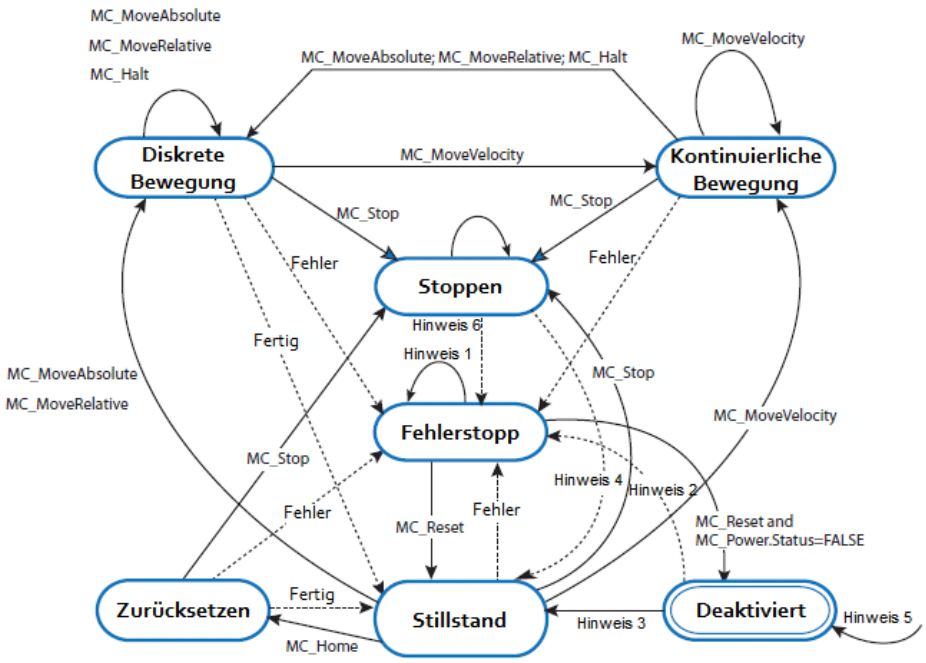
Fehler-IDs von Bewegungssteuerungs-Funktionsblock auf seite 411

Bewegungssteuerungs-Achsenstatus

Grundsätzlich wird das Verhalten der Achse auf einer höheren Ebene und bei Aktivierung mehrerer Bewegungssteuerungs-Funktionsblöcke dadurch bestimmt, dass Bewegungsbefehle immer nacheinander angenommen werden, selbst wenn der Controller zu einer echten Parallelverarbeitung fähig ist. Jeder Bewegungsbefehl ist ein Wechsel, der den Zustand der Achse und als Folge die Berechnung der aktuellen Bewegung ändert.

Bewegungssteuerungs-Achsenstatusdiagramm

Die Achse befindet sich immer in einem der definierten Zustände, die im folgenden Diagramm dargestellt sind.



Bewegungssteuerungs-Achsenstatusverhalten

Nein	Hinweis
1	In den Zuständen „Fehler-Stopp“ und „Stoppen“ können alle Funktionsblöcke (ausgenommen MC_Reset) aufgerufen werden, obwohl sie nicht ausgeführt werden. „MC_Reset“ generiert einen Wechsel in den Stillstand-Zustand (Standstill). Tritt ein Fehler auf, während sich die Maschine im Zustand „Stoppen“ befindet, wird ein Wechsel in den Zustand „Fehler-Stopp“ generiert.
2	Power.Enable = TRUE und Fehler in der Achse.
3	Power.Enable = TRUE und kein Fehler in der Achse.
4	MC_Stop.Done AND NOT MC_Stop.Execute.
5	Wenn „MC_Power“ mit Enable = False aufgerufen wird, geht die Achse bei jedem Zustand einschließlich „Fehler-Stopp“ in den Zustand „Deaktiviert“.
6	Tritt ein Fehler auf, während sich die Maschine im Zustand „Stoppen“ befindet, wird ein Wechsel in den Zustand „Fehler-Stopp“ generiert.

Bewegungssteuerungs-Achsenstatuscodewerte

Sie können den Achsenstatus mithilfe der Achsenmonitorfunktion überwachen. In der folgenden Tabelle sind die Werte angegeben, die zur Definition der jeweiligen vordefinierten Achsenstatus verwendet werden.

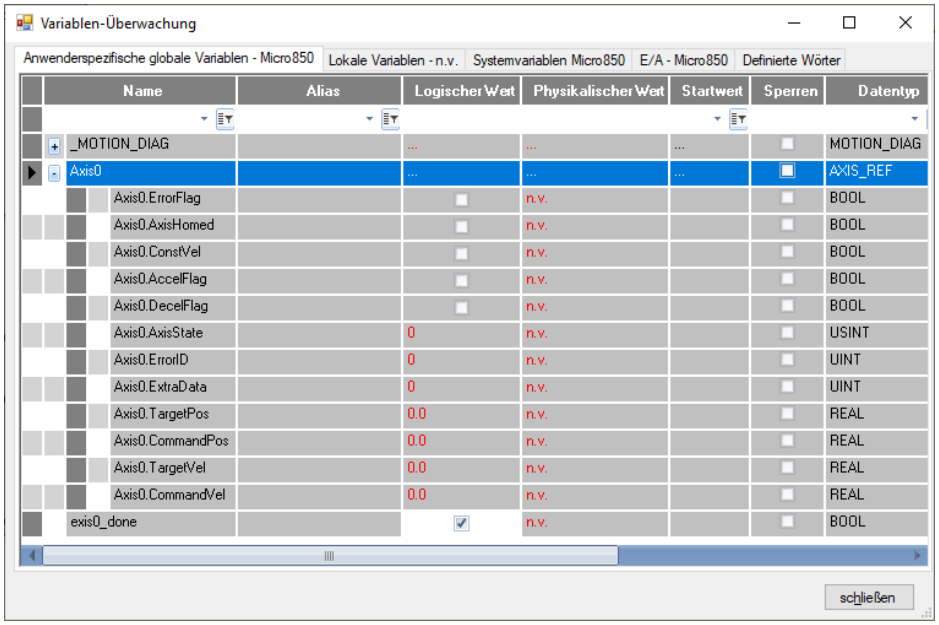
Zustandswert	Zustandsname
0x00	Deaktiviert

Zustandswert	Zustandsname
0x01	Stillstand
0x02	Diskrete Bewegung
0x03	Kontinuierliche Bewegung
0x04	Homing
0x06	Stoppen
0x07	Fehler-Stopp

Achsenstatus-Aktualisierungen

Bei der Bewegungsausführung erfolgt die Aktualisierung des Achsenstatus in Abhängigkeit davon, ob der relevante Bewegungsfunktionsblock vom POU-Scan abgerufen wird. Dies ist der Fall, auch wenn das Bewegungsprofil vom Bewegungsmotor als Hintergrundtask gesteuert wird und unabhängig vom POU-Scan erfolgt.

Beispiel: auf einer sich bewegenden Achse eines Kontaktplan-POU (Status des Kontaktplans = true) wird ein MC_MoveRelative-Funktionsblock im Kontaktplan gescannt und die Achse beginnt sich zu bewegen. Bevor MC_MoveRelative beendet wird, wird der Zustand des Kontaktplans „False“ und MC_MoveRelative wird nicht mehr gescannt. In diesem Fall kann der Zustand der Achse nicht von diskreter Bewegung zu „StandStill“ (Stillstand) wechseln, auch dann nicht, wenn die Achse vollständig angehalten wird und die Geschwindigkeit o beträgt.



Parameternummern von Bewegungssteuerungs-Funktionsblock

Die nachfolgenden Funktionsblöcke setzen bei der Programmierung von Funktionsblöcken bestimmte Parameternummern ein.

- MC_ReadParameter
- MC_ReadBoolParameter
- MC_WriteParameter
- MC_WriteBoolParameter

Ermittlung der Parameternummern

Die Parameternummern von 0 bis 999 sind für Standardparameter reserviert. Erweiterungen durch einen Lieferanten oder Benutzer sind ebenfalls zulässig, wenngleich dies die Portabilität zwischen unterschiedlichen Plattformen beeinträchtigen kann. Ist die Parameternummer größer als 999, handelt es sich um einen lieferantenspezifischen Parameter.

Parameternummer	Parametername	Datentyp	R/W	Beschreibung
1	Commanded Position	REAL	R	Angeforderte Position.
2	SWLimitPos	REAL	R/W	Position des positiven Software-Endschalters.
3	SWLimitNeg	REAL	R/W	Position des negativen Software-Endschalters.
4	EnableLimitPos	BOOL	R/W	Positiven Software-Endschalter aktivieren.
5	EnableLimitNeg	BOOL	R/W	Negativen Software-Endschalter aktivieren.
8	MaxVelocitySystem	REAL	R	Maximal zulässige Geschwindigkeit der Achse im Bewegungssystem.
9	MaxVelocityAppl	REAL	R/W	Maximal zulässige Geschwindigkeit der Achse in der Anwendung.
11	CommandedVelocity	REAL	R	Angeforderte Geschwindigkeit.
12	MaxAccelerationSystem	REAL	R	Maximal zulässige Beschleunigung der Achse im Bewegungssystem.
13	MaxAccelerationAppl	REAL	R/W	Maximal zulässige Beschleunigung der Achse in der Anwendung.
14	MaxDecelerationSystem	REAL	R	Maximal zulässige Verzögerung der Achse im System.
15	MaxDecelerationAppl	REAL	R/W	Maximal zulässige Verzögerung der Achse in der Anwendung.
16	MaxJerk	REAL	R/W	Maximal zulässige Erschütterung (Stoß) der Achse.
1001	TargetPosition	REAL	R	Finale Zielposition für den derzeit aktiven Bewegungsfunktionsblock
1002	TargetVelocity1	REAL	R	Finale Zielgeschwindigkeit für den derzeit aktiven Bewegungsfunktionsblock
1005	Duty Cycle	REAL	R/W	Der Impulsarbeitszyklus für einen Impuls. Der gültige Wert beträgt 0 – 100, und gibt 0 % - 100 % an. (Die PWM-Funktion kann durch Anpassen dieses Werts realisiert werden). Dieser Parameter kann nur mithilfe dieses Funktionsblocks konfiguriert werden. Der Controller legt den Standardwert auf 50,0. **Hinweis**: Für den Arbeitszyklus wird der Wert von der Standardeinstellung 50,0 überschrieben, wenn der Controller vom Betriebsmodus in den Programm-Modus und dann wieder in den Betriebsmodus umgeschaltet wird oder wenn der Controller einen Einschaltzyklus durchläuft.
1006	PulsePerRevolution	REAL	R	Die vom Benutzer in der CCW GUI eingegebene Einstellung „Impulse pro Umdrehung“.
1007	TravelPerRevolution	REAL	R	Die vom Benutzer in der CCW GUI eingegebene Einstellung „Distanz pro Umdrehung“.

Fehler-IDs von Bewegungssteuerungs-Funktionsblock

Wenn ein Bewegungssteuerungs-Funktionsblock mit einem Fehler endet und sich die Achse im Zustand „ErrorStop“ befindet, kann die Achse mit dem Funktionsblock „MC_Reset“ oder „MC_Power Off/On“ und „MC_Reset“ wiederhergestellt werden. Die Achse kann in den normalen Bewegungsbetrieb zurückgesetzt werden, ohne den Betrieb des Controllers zu beenden.

Verwenden Sie diese Tabelle, um die Fehler für die Bewegungssteuerungs-Funktionsblöcke zu bestimmen.

Wert	MAKRO-ID	Beschreibung
00	MC_FB_ERR_ NO	Der Funktionsblock wird erfolgreich ausgeführt.
01	MC_FB_ERR_ WRONG_STATE	Der Funktionsblock kann nicht ausgeführt werden, da sich die Achse nicht im richtigen Zustand befindet. Überprüfen Sie den Zustand der Achse auf seite 408.
02	MC_FB_ERR_ RANGE	Der Funktionsblock kann aufgrund von im Funktionsblock gesetzten ungültigen achsendynamischen Parametern (Geschwindigkeit, Beschleunigung, Verzögerung oder Stoß) nicht ausgeführt werden. Korrigieren Sie die Einstellung für die dynamischen Parameter im Funktionsblock gemäß den Einstellungen auf der Konfigurationsseite für die Achsendynamik.
03	MC_FB_ERR_ PARAM	Der Funktionsblock kann aufgrund eines im Funktionsblock gesetzten anderen ungültigen Parameters (nicht Geschwindigkeit, Beschleunigung, Verzögerung oder Stoß) nicht ausgeführt werden. Korrigieren Sie die Einstellung für die Parameter (z. B. Modus oder Position) für den Funktionsblock.
04	MC_FB_ERR_ AXISNUM	Der Funktionsblock kann nicht ausgeführt werden, da die Achse nicht vorhanden ist, die Daten der Achsenkonfiguration beschädigt sind oder die Achse falsch konfiguriert wurde.
05	MC_FB_ERR_ MECHAN	Der Funktionsblock kann nicht ausgeführt werden, da in dieser Achse ein Fehler aufgrund von Antriebs- oder mechanischen Problemen aufgetreten ist. Prüfen Sie die Verbindung zwischen Antrieb und Controller (Signale für „Antrieb bereit“ und „In Position"), und stellen Sie sicher, dass der Antrieb normal arbeitet.
06	MC_FB_ERR_ NOPOWER	Der Funktionsblock kann nicht ausgeführt werden, da diese Achse nicht eingeschaltet ist. Schalten Sie die Achse mit dem Funktionsblock „MC_Power“ ein.
07	MC_FB_ERR_ RESOURCE	Der Funktionsblock kann nicht ausgeführt werden, da die für den Funktionsblock erforderliche Ressource von einem anderen Funktionsblock gesteuert wird oder nicht zur Verfügung steht. Stellen Sie sicher, dass die vom Funktionsblock benötigte Ressource zur Nutzung verfügbar ist. Beispiele: • „MC_Power“ versucht, dieselbe Achse zu steuern. • „MC_Stop“ wird gleichzeitig für dieselbe Achse ausgeführt. • „MC_TouchProbe“ wird gleichzeitig für dieselbe Achse ausgeführt. • „MC_TouchProbe“ wird ausgeführt, aber in der Bewegungskonfiguration ist keine Fühlerberührung aktiviert.
08	MC_FB_ERR_ PROFILE	Der Funktionsblock kann nicht ausgeführt werden, da das im Funktionsblock definierte Bewegungsprofil nicht abgerufen werden kann. Korrigieren Sie das Profil im Funktionsblock.
09	MC_FB_ERR_ VELOCITY	Der Funktionsblock kann nicht ausgeführt werden, da das im Funktionsblock angeforderte Bewegungsprofil aufgrund der aktuellen Achsengeschwindigkeit nicht abgerufen werden kann. Beispiele: • Der Funktionsblock fordert während der Bewegung der Achse eine Richtungsumkehr. • Das erforderliche Bewegungsprofil kann nicht erzielt werden, weil die aktuelle Geschwindigkeit zu niedrig oder zu hoch ist. Prüfen Sie die Einstellungen des Bewegungsprofils im Funktionsblock und korrigieren Sie das Profil oder führen Sie den Funktionsblock erneut aus, wenn die Achsengeschwindigkeit mit dem angeforderten Bewegungsprofil kompatibel ist.
0A	MC_FB_ERR_ SOFT_LIMIT	Dieser Funktionsblock kann nicht ausgeführt werden, da er das Software-Limit überschreitet, oder der Funktionsblock wird abgebrochen, sobald das Software-Limit erreicht wird. Prüfen Sie die Geschwindigkeits- oder Zielpositionseinstellungen für den Funktionsblock oder passen Sie die Einstellung der Weichen Grenze an.
0B	MC_FB_ERR_ HARD_LIMIT	Dieser Funktionsblock wird abgebrochen, da während der Achsenbewegung der aktive Zustand des Hardware-Limit-Schalters festgestellt wurde, oder der Funktionsblock wird abgebrochen, wenn vor Beginn der Achsenbewegung der aktive Zustand des Hardware-Limit-Schalters festgestellt wird. Bewegen Sie die Achse in umgekehrter Richtung von der harten Grenze weg.

Wert	MAKRO-ID	Beschreibung
0C	MC_FB_ERR_ LOG_LIMIT	Dieser Funktionsblock kann nicht ausgeführt werden, da er den logischen Grenzwert für den PTO-Akkumulator überschreitet, oder der Funktionsblock wird abgebrochen, wenn der logische Grenzwert für den PTO-Akkumulator erreicht wird. Prüfen Sie die Geschwindigkeits- oder Zielpositionseinstellungen für den Funktionsblock. Passen Sie alternativ mit dem Funktionsblock „MC_SetPosition" das Achsenkoordinatensystem an.
0D	MC_FB_ERR_ ERR_ENGINE	A Während der Ausführung dieses Funktionsblocks wurde ein Ausführungsfehler des Bewegungsmotors festgestellt. Durchlaufen Sie den Einschaltzyklus der gesamten Bewegungseinrichtung (einschließlich Controller, Antriebe und Aktoren), und laden Sie die Benutzeranwendung erneut herunter. Bleibt der Fehler weiterhin bestehen, wenden Sie sich an den technischen Support.
10	MC_FB_ERR_ NOT_HOMED	Der Funktionsblock kann nicht ausgeführt werden, da die Achse zuerst zurückgesetzt werden muss. Setzen Sie die Achse mit dem Funktionsblock „MC_Home" zurück.
80	MC_FB_ERR_ PARAM_MODIFIED	Warnung: Die erforderliche Geschwindigkeit wurde auf einen niedrigeren Wert gesetzt. Der Funktionsblock wird erfolgreich mit niedrigerer Geschwindigkeit ausgeführt.

Achsfehler-Szenarios

Wenn die an einer Achse ausgeführte Anweisung eines Bewegungsfunktionsblocks zu einem Funktionsblockfehler führt, wird in den meisten Fällen auch die Achse mit dem Zustand „Fehler" gekennzeichnet, und das zugehörige ErrorID-Element wird auf die AXIS_REF-Daten für die Achse gesetzt.

In den folgenden Situationen wird bei einem Achsenfehler allerdings keine Fehlerkennzeichnung vorgenommen, sodass die Benutzeranwendung nach dem Ändern des Achsenzustands den Bewegungsfunktionsblock für die Achse immer noch erfolgreich ausführen kann.

Szenario	Beispiel
Ein Bewegungsfunktionsblock gibt Anweisungen an eine Achse aus, aber die Achse ist in einem Zustand, in dem der Funktionsblock nicht ordnungsgemäß ausgeführt werden kann.	Die Achse hat keinen Strom, oder die Achse ist in einer Homing-Sequenz oder einem Nothalt-Zustand.
Ein Bewegungsfunktionsblock gibt Anweisungen an eine Achse aus, aber diese Achse wird noch von einem anderen Bewegungsfunktionsblock gesteuert. Die Achse kann nicht erlauben, dass die Bewegung ohne vollständiges Anhalten von dem neuen Funktionsblock gesteuert wird.	Der neue Funktionsblock gibt den Befehl zum Ändern der Bewegungsrichtung an die Achse aus.
Wenn ein Bewegungsfunktionsblock versucht, eine Achse zu steuern, die noch von einem anderen Bewegungsfunktionsblock gesteuert wird, und das neu definierte Bewegungsprofil vom Controller nicht umgesetzt werden kann.	Benutzeranwendung gibt einen MC_MoveAbsolute-Funktionsblock mit S-Kurvenprofil an eine Achse aus, wobei der gegebene Abstand für die Bewegung der Achse zu kurz ist.

Datentyp AXIS_REF

Der Datentyp "AXIS_REF" ist eine Datenstruktur, die Informationen für eine Bewegungsachse enthält und in allen Servosteuerungs-Funktionsblöcken als Eingangs- und Ausgangsvariable verwendet wird. Eine Instanz eines AXIS_REF-Datentyps wird automatisch erstellt, wenn Sie der Konfiguration eine Bewegungsachse hinzufügen.

Parameter	Datentyp	Beschreibung
Axis_ID	AXIS_REF	Die logische Achsen-ID, die automatisch von Connected Components Workbench zugewiesen wurde. Sie kann vom Benutzer weder bearbeitet noch angezeigt werden.

Parameter	Datentyp	Beschreibung
Error Flag	BOOL	Zeigt an, ob in der Achse ein Fehler vorliegt. Nachdem die Achse mit einem Fehler gekennzeichnet wurde und die Fehler-ID nicht 0 ist, muss die Achse mithilfe von „MC_Reset" zurückgesetzt werden, bevor ein anderer Bewegungsfunktionsblock ausgegeben werden kann.
AxisHomed	BOOL	Zeigt an, ob die Achse erfolgreich zurückgesetzt wurde oder nicht. Wenn der Benutzer versucht, das Zurücksetzen einer Achse zu wiederholen, wenn „AxisHomed" bereits gesetzt ist (Zurücksetzen erfolgreich ausgeführt), und das Ergebnis ist nicht erfolgreich, wird der Status „AxisHomed" gelöscht.
ConstVel	BOOL	Zeigt an, ob sich die Achse in der Bewegung „Konstante Geschwindigkeit" befindet oder nicht. Feststehende Achsen werden in „Konstante Geschwindigkeit" nicht berücksichtigt.
AccFlag	BOOL	Zeigt an, ob sich die Achse in der Bewegung „Beschleunigen" befindet oder nicht.
DecelFlag	BOOL	Zeigt an, ob sich die Achse in der Bewegung „Verzögern" befindet oder nicht.
AxisState	USINT	Zeigt den aktuellen Zustand der Achse an.
ErrorID	UINT	Zeigt die Ursache des Achsenfehlers an, wenn ein Fehler durch „ErrorFlag" angezeigt wird. Dieser Fehler resultiert in der Regel aus einem Fehler bei der Ausführung des Bewegungssteuerungs-Funktionsblocks.
ExtraData	UINT	Reserviert.
TargetPos	REAL	Zeigt die finale Zielposition der Achse für die Funktionsblöcke „MoveAbsolute" und „MoveRelative" an. Bei den Funktionsblöcken „MoveVelocity", „Stop" und „Halt" hat „TargetPos" den Wert 0 – ausgenommen, die durch vorherige Positionsfunktionsblöcke gesetzte „TargetPos" wird nicht gelöscht.
CommandPos	REAL	Während der Bewegung ist dies die aktuelle Position, die der Controller der Achse anweist. Zwischen der tatsächlichen Achsenposition und dieser Befehlsposition kann es zu einer leichten Verzögerung kommen.
TargetVel	REAL	Die maximale Zielgeschwindigkeit, die der Achse für einen bewegenden Funktionsblock angewiesen wird. Der Wert von „TargetVel" im aktuellen Funktionsblock oder ein kleinerer Wert, abhängig von den anderen Parametern im selben Funktionsblock.
CommandVel	REAL	Während der Bewegung zeigt dieses Element die aktuelle Geschwindigkeit an, die der Controller der Achse anweist. Zwischen der tatsächlichen Geschwindigkeit der Achse und „CommandVel" kann es aufgrund von Antriebsverzögerungen oder einem Überschwingen bei der Antriebsanpassung zu leichten Abweichungen kommen.

Datentyp FB_AXIS_REF

Der Datentyp FB_AXIS_REF ist eine Datenstruktur, die Informationen über eine Bewegungsfeedbackachse enthält. Sie wird als Eingabe- und Ausgabevariable in Bewegungssteuerungs-Funktionsblöcken verwendet. Beim Hinzufügen eines HSC-Moduls wird automatisch eine Instanz des Datentyps FB_AXIS_REF erstellt, wenn als Modus Feedbackachse konfiguriert ist.

WICHTIG Nachdem die Feedbackachse mit einem Fehler gekennzeichnet wurde und die Fehler-ID nicht null ist, muss die FBAxis mithilfe von MC_Reset zurückgesetzt werden, bevor ein anderer Bewegungsfunktionsblock gestartet wird.

Parameter	Datentyp	Beschreibung
Axis_ID	FB_AXIS_REF	Die Logikachsen-ID wird automatisch von der Connected Components Workbench zugewiesen. Sie kann vom Benutzer weder bearbeitet noch angezeigt werden.

Parameter	Datentyp	Beschreibung
ErrorFlag	BOOL	Zeigt an, ob in der Feedbackachse ein Fehler vorliegt.
ConstVel	BOOL	Zeigt an, ob sich die Feedbackachse in fortlaufender Bewegung „Konstante Geschwindigkeit" befindet oder nicht. Feststehende Achsen werden in „Konstante Geschwindigkeit" nicht berücksichtigt.
AccelFlag	BOOL	Zeigt an, ob sich die Feedbackachse in der Bewegung „Beschleunigen" befindet oder nicht.
DecelFlag	BOOL	Zeigt an, ob sich die Feedbackachse in der Bewegung „Verzögern" befindet oder nicht.
AxisState	USINT	Zeigt den aktuellen Zustand der Feedbackachse an.
ErrorID	UINT	Zeigt die Ursache des Achsenfehlers an, wenn ein Fehler durch „ErrorFlag" angezeigt wird. Dieser Fehler resultiert in der Regel aus einem Fehler bei der Ausführung des Bewegungssteuerungs-Funktionsblocks.
ExtraData	UINT	Reserviert.
ActualPos	REAL	Tatsächliche mechanische Position, die aus dem Bewegungs-Feedbackkanal (HSC) gelesen wird.
ActualVel	REAL	Tatsächliche mechanische Geschwindigkeit, die aus dem Bewegungs-Feedbackkanal (HSC) gelesen wird.

Achsenvariablen

Achsenvariablen werden verwendet, um die Position, Geschwindigkeit, Beschleunigung und Fehler für eine bestimmte Bewegungssteuerungsachse zu kontrollieren.

So weisen Sie eine Variable einem Achsenausgabeparameter zu:

- In einem Funktionsblockdiagramm

  Sie können den Achsenausgabeparameter eines Bewegungssteuerungs-Funktionsblocks grafisch mit dem Achseneingabeparameter eines anderen Bewegungssteuerungs-Funktionsblocks verbinden. Beispiel: Der Achsenausgabeparameter MC_POWER kann mit dem Achseneingabeparameter MC_HOME verbunden werden.

- In einem Kontaktplan

  Sie können dem Achsenausgabeparameter eines Bewegungssteuerung-Funktionsblocks keine Variable zuweisen, da sie schreibgeschützt ist.

Überwachen einer AXIS_REF-Variable

Sie können eine AXIS_REF- oder FBAXIS_REF-Variable bei angeschlossenem Controller über die Software, wenn der Bewegungsmotor aktiv ist, oder als Teil der Benutzerlogik in der Benutzeranwendung überwachen. Außerdem können Sie die AXIS_REF- oder FBAXIS_REF-Variable remote über verschiedene Kommunikationskanäle überwachen.

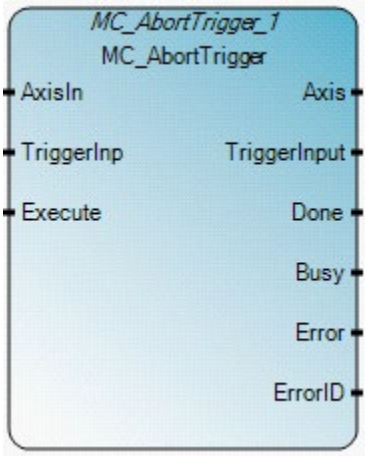
MC_AbortTrigger (Bewegungssteuerungs-Abbruchauslöser)

Bricht Bewegungsfunktionsblöcke ab, die mit Auslöseereignissen verbunden sind.

MC_AbortTrigger wird nur ausgeführt, wenn er einer Achse zugewiesen ist, die von MC_TouchProbe gesteuert wird.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

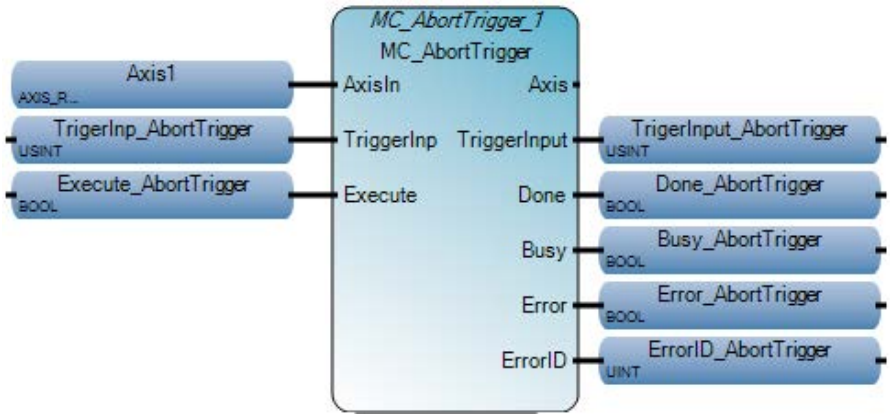
Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung



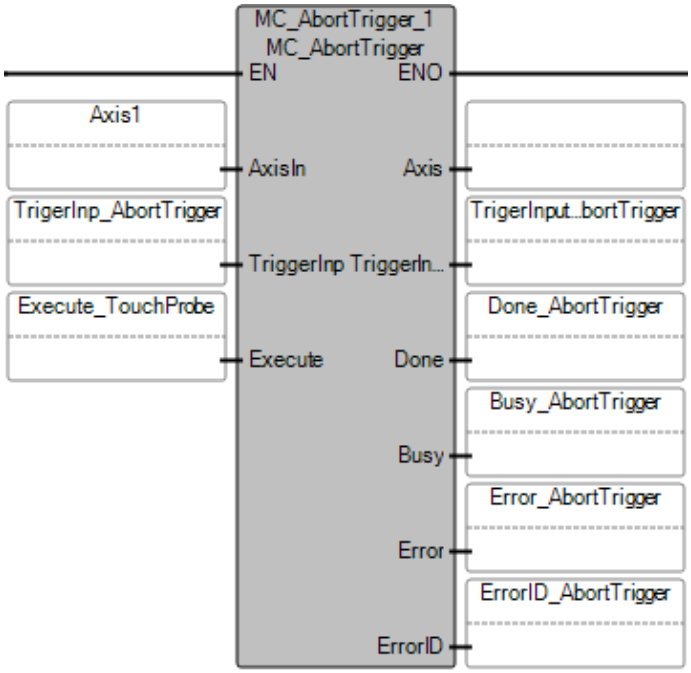
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parameter	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_AbortTrigger-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF auf seite 413 FB_AXIS_REF auf seite 414	Verwenden Sie den Datentyp AXIS_REF, um den AxisIn-Parameter zu definieren. Wenn die Achse eine FB_Axis ist (Feedbackachse), verwenden Sie den Datentyp FB_AXIS_REF zum Definieren der Parameter für AxisIn.
TriggerInp	Eingang	USINT	Dieser Parameter wird ignoriert.
Execute	Eingang	BOOL	Wenn TRUE, wird „Ereignis auslösen“ an der Anstiegflanke abgebrochen.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.
TriggerInput	Ausgang	USINT	Dieser Parameter wird ignoriert.
Done	Ausgang	BOOL	TRUE – „Ereignis auslösen“ wird abgebrochen.
Busy	Ausgang	BOOL	TRUE – der Funktionsblock wird nicht beendet.
Error	Ausgang	BOOL	WAHR – es wurde ein Fehler erkannt.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks definiert.

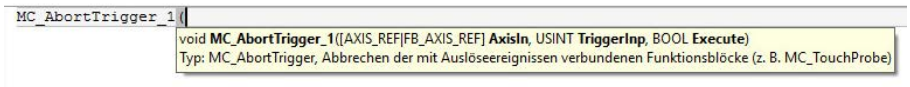
Diagrammbeispiel für Funktionsblock MC_AbortTrigger



MC_AbortTrigger-Kontaktplan-Beispiel

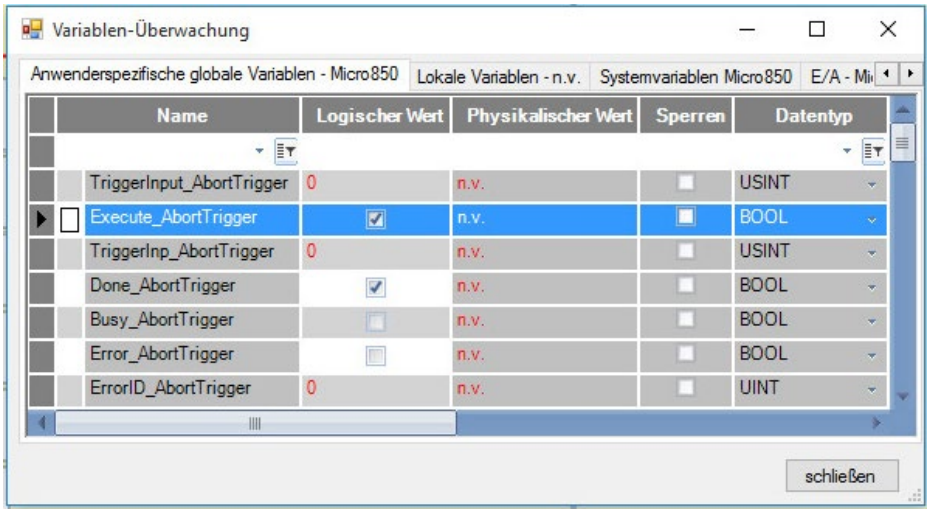


Beispiel für MC_AbortTrigger in strukturiertem Text




```
MC_AbortTrigger_1(Axis1,TrigerInp_AbortTrigger,Execute_AbortTrigger);
Done_AbortTrigger := MC_AbortTrigger_1.Done;
Busy_AbortTrigger := MC_AbortTrigger_1.Busy;
Error_AbortTrigger := MC_AbortTrigger_1.Error;
ErrorID_AbortTrigger := MC_AbortTrigger_1.ErrorID;
```

Ergebnisse



MC_Halt (Bewegungssteuerungs-Halt)

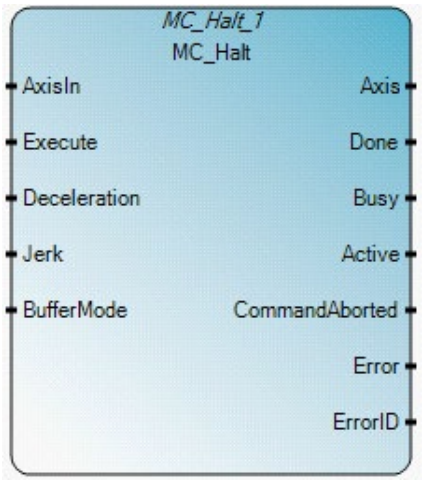
Befiehlt das Anhalten einer gesteuerten Bewegung unter normalen Betriebsbedingungen.

Details zum Betrieb;

- Der Achsenstatus ändert sich in „DiscreteMotion“, bis die Geschwindigkeit null beträgt. Wenn die Geschwindigkeit null erreicht, wird „Done“ auf „True“ gesetzt und der Achsenstatus ändert sich zu „StandStill“.
  - Während der Verzögerung der Achse kann ein anderer Bewegungsbefehl ausgeführt werden, wodurch MC_Halt überschrieben wird.
  - Wenn MC_Halt beim Achsenstatus „Homing“ ausgegeben wird, meldet der Anweisungsblock einen Fehler, und der Homing-Prozess wird nicht unterbrochen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

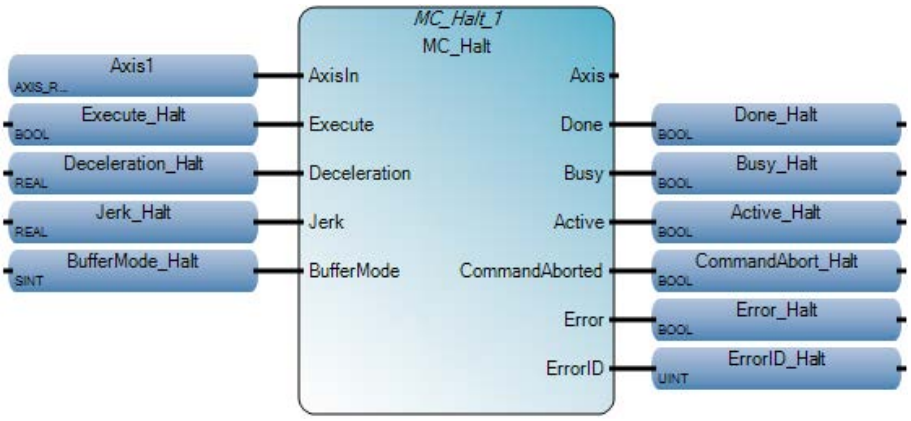
Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung



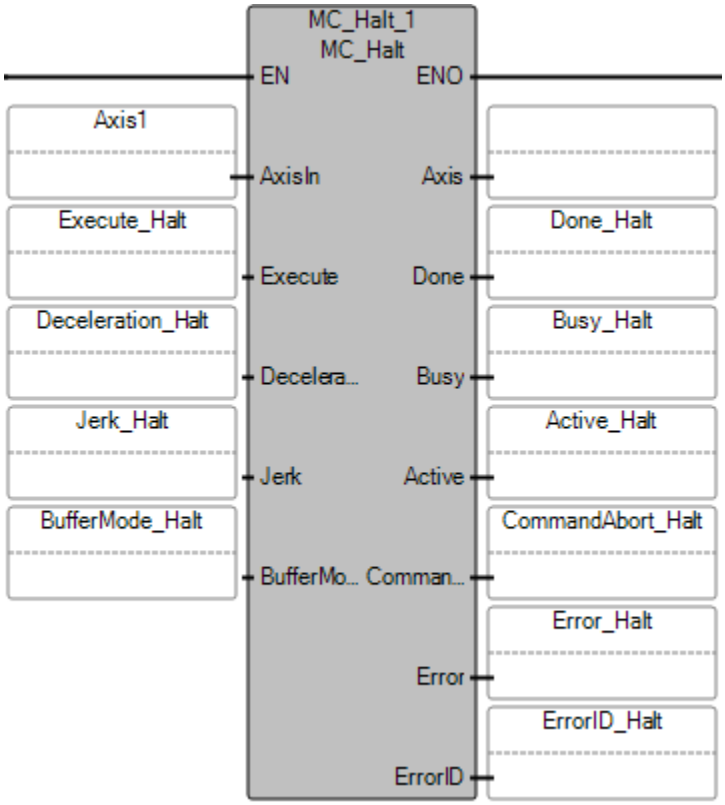
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_Halt-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF	Definieren Sie AxisIn mithilfe der Parameter des Datentyps AXIS_REF.
Execute	Eingang	BOOL	Gibt an, wann die Bewegung beginnen soll. TRUE – Bewegung an der Anstiegsflanke beginnen. FALSE – Bewegung nicht beginnen. Wird „MC_Halt" während des Homing-Prozesses ausgeführt, wird „MC_Halt" auf MC_FB_ERR_STATE gesetzt, und der Homing-Prozess wird fortgesetzt.
Deceleration	Eingang	REAL	Wert der Verzögerung (immer positiv) (Senkung der Energie zum Motor.) Wenn Verzögerung <= 0 und der Achsenstatus ist nicht „Standstill", wird „MC_Halt" auf MC_FB_ERR_RANGE gesetzt.
Jerk	Eingang	REAL	Wert für den Stoß (immer positiv). Wenn Stoß < 0 und der Achsenstatus ist „Standstill", wird „MC_Halt" auf MC_FB_ERR_RANGE gesetzt.
BufferMode	Eingang	SINT	Nicht verwendet. Der Modus ist immer „MC_Aborting".
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF auf seite 413	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.
Done	Ausgang	BOOL	Geschwindigkeit null erreicht.
Busy	Ausgang	BOOL	Dieser Anweisungsblock ist nicht abgeschlossen.
Active	Ausgang	BOOL	Gibt an, dass der Anweisungsblock die Steuerung über die Achse besitzt.
CommandAborted	Ausgang	BOOL	Der Befehl wurde durch einen anderen Befehl oder ein Anhalten aufgrund eines Fehlers überschrieben.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.

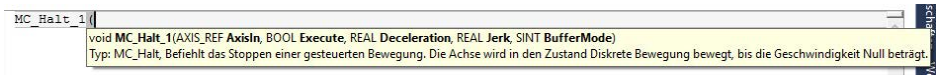
Diagrammbeispiel für Funktionsblock MC_Halt



MC_Halt-Kontaktplan-Beispiel

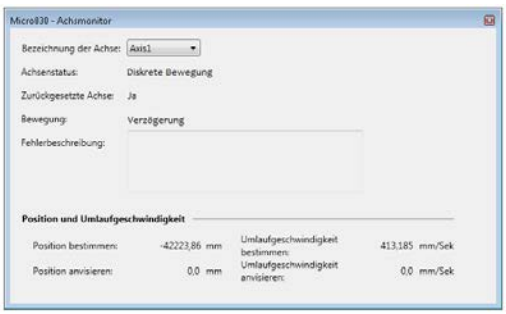


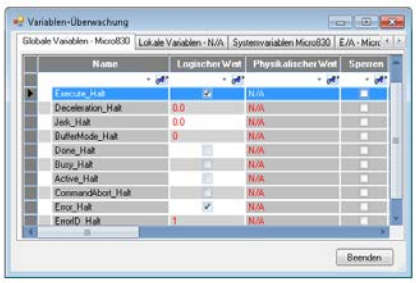
Beispiel für MC_Halt in strukturiertem Text




```
Deceleration_Halt := 10.0;
Jerk_Halt := 10.0;
MC_Halt_1(Axis1,Execute_Halt,Deceleration_Halt,Jerk_Halt,BufferMode_Halt);
Done_Halt := MC_Halt_1.Done;
Busy_Halt := MC_Halt_1.Busy;
Active_Halt := MC_Halt_1.Active;
CommandAbort_Halt := MC_Halt_1.CommandAborted;
Error_Halt := MC_Halt_1.Error;
ErrorID_Halt := MC_Halt_1.ErrorID;
```

Ergebnisse





MC_Home (Bewegungssteuerung – Ausgangsstellung)

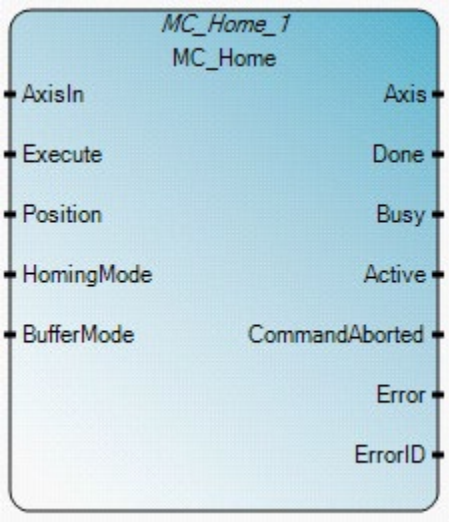
Befiehlt der Achse, die Sequenz <*search home*> auszuführen. Die Details dieser Sequenz sind herstellerabhängig und können über die Achsenparameter festgelegt werden. Mit der Eingabe „Position“ wird die absolute Position festgelegt, wenn ein Referenzsignal erkannt und der konfigurierte Versatz von der Ausgangsstellung erreicht wurde.

Details zum Betrieb:

- Nachdem „MC_Power“ ausgegeben wurde, wird der Achsenstatus „Homed“ (Zurückgesetzt) auf 0 (nicht zurückgesetzt) zurückgesetzt. In den meisten Fällen muss der Funktionsblock „MC_Home“ nach Einschalten der Achse ausgeführt werden, um die Achsenposition und den Bezugswert für die Ausgangsstellung zu kalibrieren.
- Der Funktionsblock „MC_Home“ kann nur mithilfe der Funktionsblöcke „MC_Stop“ oder „MC_Power“ abgebrochen werden. Wenn er vor seiner vollständigen Ausführung abgebrochen wird, wird die zuvor gefundene Referenzposition als ungültig betrachtet und der Achsenstatus „Referenzposition“ zurückgesetzt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_Home-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF	Definieren Sie AxisIn mithilfe der Parameter des Datentyps AXIS_REF.

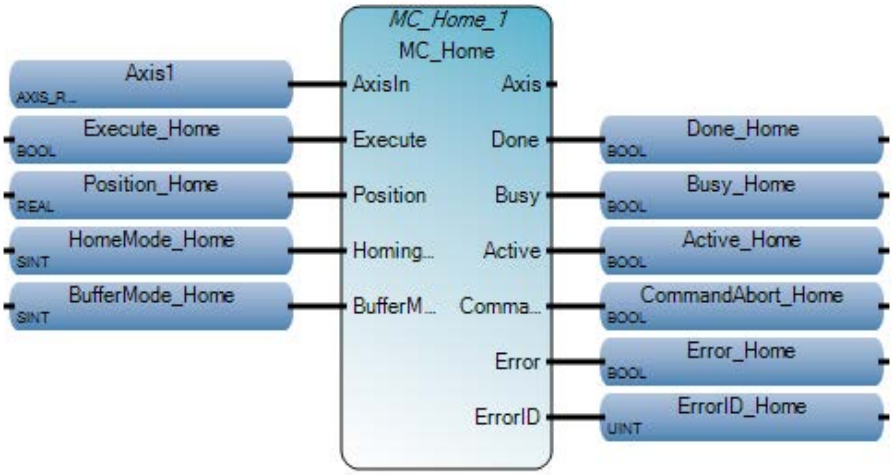
Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Gibt an, wann die Bewegung beginnen soll. TRUE – Bewegung an der Anstiegsflanke beginnen. FALSE – Bewegung nicht beginnen.
Position	Eingang	REAL	Die absolute Position wird gesetzt, wenn das Referenzsignal erkannt und der konfigurierte Versatz zur Ausgangsstelle erreicht wurde. Der Wertbereich für diese Eingabe reicht von -0x40000000 bis 0x40000000 physischen Impulsen nach Umwandlung der Position von der Einheit der Benutzerposition in PTO-Pulse. Legen Sie den Wert „Position" innerhalb der weichen Grenze fest. Ein ungültiger Eingabewert führt zu einem Fehler. Fehler-ID = **MC_FB_ERR_PARAM**.
HomingMode	Eingang	SINT	Zahleneingabe für den Homing-Modus (Zurücksetzen-Modus).
BufferMode	Eingang	SINT	Nicht verwendet. Dieser Modus ist immer „mcAborting".
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF auf seite 413	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.
Done	Ausgang	BOOL	TRUE – der Homing-Vorgang wurde erfolgreich abgeschlossen, und der Zustand der Achse wurde auf StandStill gesetzt. FALSE – der Homing-Vorgang läuft oder wurde nicht abgeschlossen.
Busy	Ausgang	BOOL	TRUE – der Anweisungsblock ist nicht abgeschlossen. FALSE – der Anweisungsblock ist abgeschlossen.
Active	Ausgang	BOOL	TRUE – gibt an, dass der Anweisungsblock die Steuerung über die Achse besitzt.
CommandAborted	Ausgang	BOOL	TRUE – Befehl wurde durch einen anderen Befehl oder durch ein Anhalten aufgrund eines Fehlers überschrieben.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UNIT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.

HomingModes

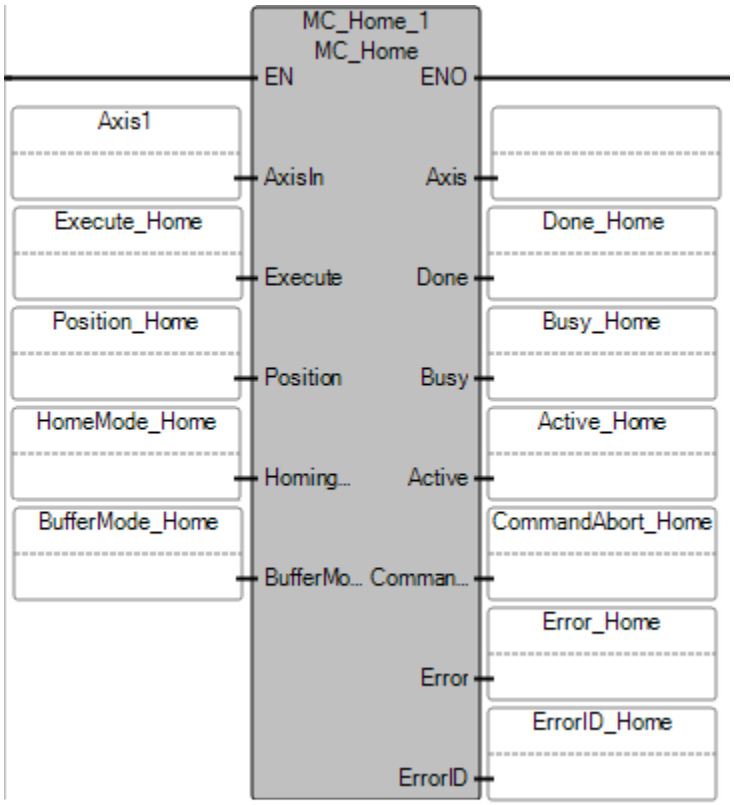
Mithilfe dieser Tabelle können Sie die Werte für den Parameter „HomingMode" im Befehl für die Achssteuerung auf seite 405 MC_Home ermitteln.

Wert	Name	Beschreibung
0x00	MC_HOME_ABS_SWITCH	Homing-Vorgang durch das Suchen des Schalters „Home Absolute" (absolute Ausgangsstellung)
0x01	MC_HOME_LIMIT_SWITCH	Zurücksetzvorgang durch das Suchen des Endschalters
0x02	MC_HOME_REF_WITH_ABS	Zurücksetz-Vorgang durch das Suchen des Schalters „Home Absolute" sowie durch Verwenden des Kodierer-Referenzimpulses
0x03	MC_HOME_REF_PULSE	Zurücksetz-Vorgang durch das Suchen des Endschalters sowie durch das Verwenden des Kodierer-Referenzimpulses
0x04	MC_HOME_DIRECT	Statischer Zurücksetz-Vorgang mit direkter Erzwingung einer Ausgangsstellung von der Benutzerreferenz aus. Der Funktionsblock legt die aktuelle Position, in der sich der Mechanismus befindet, als Ausgangsstellung fest. Dabei wird die Position durch den Eingabeparameter „Position" ermittelt.

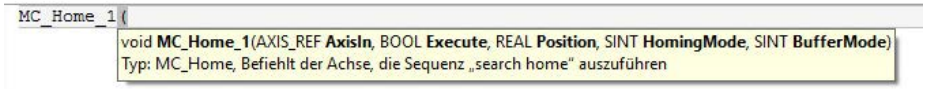
Diagrammbeispiel für Funktionsblock MC_Home



MC_Home-Kontaktplan-Beispiel

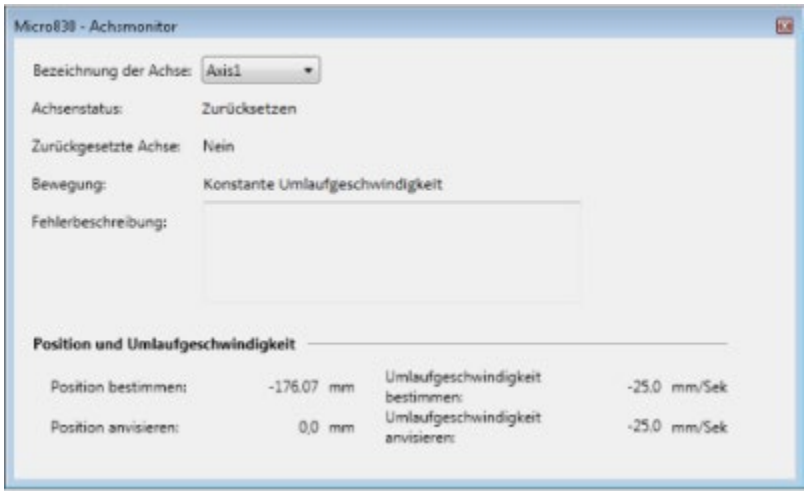


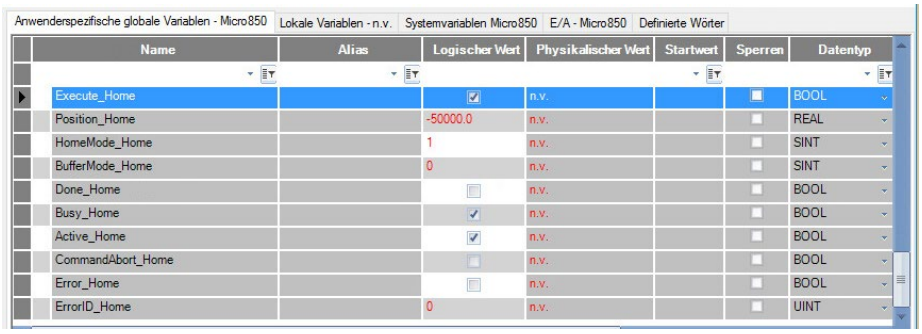
Beispiel für MC_Home in strukturiertem Text



```
Position_Home := -50000.0;
HomeMode_Home := 4;  (*1*)
MC_Home_1(Axis1,Execute_Home,Position_Home,HomeMode_Home,BufferMode_Home);
Done_Home := MC_Home_1.Done;
Busy_Home := MC_Home_1.Busy;
Active_Home := MC_Home_1.Active;
CommandAbort_Home := MC_Home_1.CommandAborted;
Error_Home := MC_Home_1.Error;
ErrorID_Home := MC_Home_1.ErrorID;
```

Ergebnisse





MC_MoveAbsolute (Bewegungssteuerung absolute Bewegung)

Befiehlt eine gesteuerte Bewegung zu einer angegebenen absoluten Position.

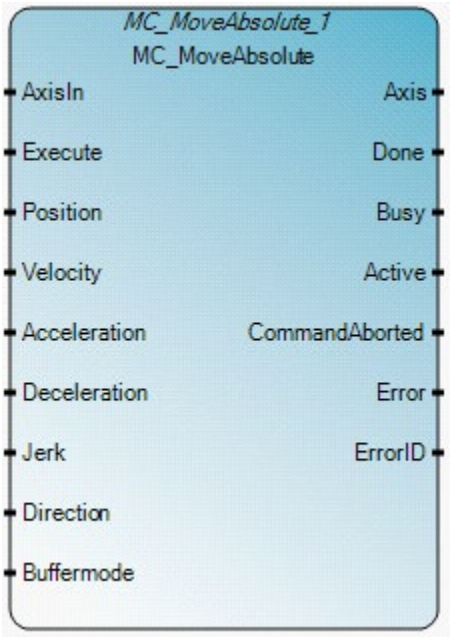
Details zum Betrieb:

- Für den Micro800-Controller:
  - Wird das Vorzeichen der Geschwindigkeit für einen „MC_MoveAbsolute“-Funktionsblock ignoriert, da die Bewegungsrichtung durch die aktuelle Position und die Zielposition bestimmt wird.
  - Wird die Richtungseingabe für einen „MC_MoveAbsolute“-Funktionsblock ignoriert, da die Zielposition nur mit einer mathematischen Lösung erreicht werden kann.
- Wird der Funktionsblock „MC_MoveAbsolute“ beim Achsenstatus „Stillstand“ (Standstill) des Micro800-Controllers und bei einem relativen Abstand von Null zur Bewegung ausgegeben, wird die Ausführung des Funktionsblocks umgehend mit „Fertig“ (Done) gemeldet.

- Wenn ein „MC_MoveAbsolute“-Funktionsblock an eine Achse ausgegeben wird, die sich nicht in zurückgesetzter Position befindet, meldet der Funktionsblock einen Fehler.
- Der Funktionsblock „MoveAbsolute“ wird mit Geschwindigkeit Null beendet, wenn er nicht durch einen anderen Funktionsblock überschrieben wird.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung

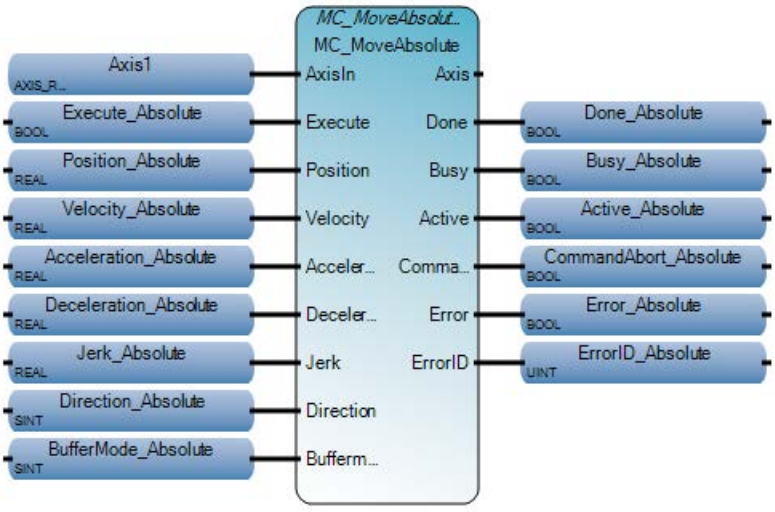


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

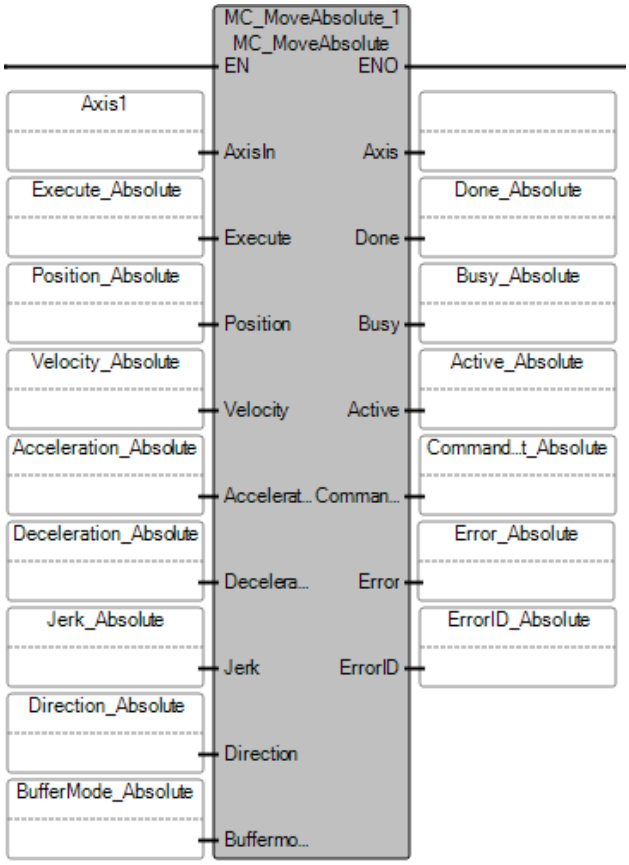
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_MoveAbsolute-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF	Definieren Sie AxisIn mithilfe der Parameter des Datentyps AXIS_REF.
Execute	Eingang	BOOL	Gibt an, wann die Bewegung beginnen soll. TRUE – Bewegung an der Anstiegsflanke beginnen. FALSE – Bewegung nicht beginnen. Die Achse sollte sich bei Ausgabe dieses Ausführungsbefehls in der Ausgangsstellung befinden, andernfalls kommt es zum Fehler „MC_FB_ERR_NOT_HOMED“.
Position	Eingang	REAL	Zielposition für die Bewegung in technischer Einheit (negativ oder positiv). Die technische Einheit wird auf der Konfigurationsseite **Bewegung - Allgemein** für die jeweilige Achse definiert.

Parameter	Parametertyp	Datentyp	Beschreibung
Velocity	Eingang	REAL	Wert der Höchstgeschwindigkeit. Wenn „Stoß" = 0, kann die Höchstgeschwindigkeit nicht erreicht werden. Das Vorzeichen von „Velocity" wird ignoriert; die Ermittlung der Bewegungsrichtung erfolgt über die Eingabe „Position".
Acceleration	Eingang	REAL	Wert der Beschleunigung (immer positiv - Anstieg der Energie zum Motor.) Benutzereinheit/s^2
Deceleration	Eingang	REAL	Wert der Verzögerung (immer positiv - Senkung der Energie zum Motor.) u/s^2
Jerk	Eingang	REAL	Wert für den Stoß (immer positiv). u/s^3 Wenn der Wert der Eingabe „Jerk" 0 beträgt, wird das Profil „Trapezoid" durch den Bewegungsmotor berechnet. Wenn „Stoß" größer ist als 0, wird das Profil „S-Curve" berechnet.
Direction	Eingang	SINT	Dieser Parameter wird nicht verwendet.
BufferMode	Eingang	SINT	Dieser Parameter wird nicht verwendet.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF auf seite 413	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.
Done	Ausgang	BOOL	Wenn TRUE, wurde die angewiesene Position erreicht. Wenn die Eingabe „In-Position" für diese Achse als „Enabled" konfiguriert ist, muss der Frequenzumrichter das Eingangssignal „In-Position" aktiv schalten, bevor dieses „Done"-Bit zu „True" wechselt. Wenn diese Aktion nicht abgebrochen wird, wird sie mit Geschwindigkeit Null abgeschlossen.
Busy	Ausgang	BOOL	Wenn TRUE, wurde der Funktionsblock nicht beendet.
Active	Ausgang	BOOL	Wenn TRUE, besitzt der Funktionsblock die Steuerung über die Achse.
CommandAborted	Ausgang	BOOL	Wenn TRUE, wurde der Befehl durch einen anderen Befehl oder durch ein Anhalten aufgrund eines Fehlers überschrieben.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.

Diagrammbeispiel für Funktionsblock MC_MoveAbsolute



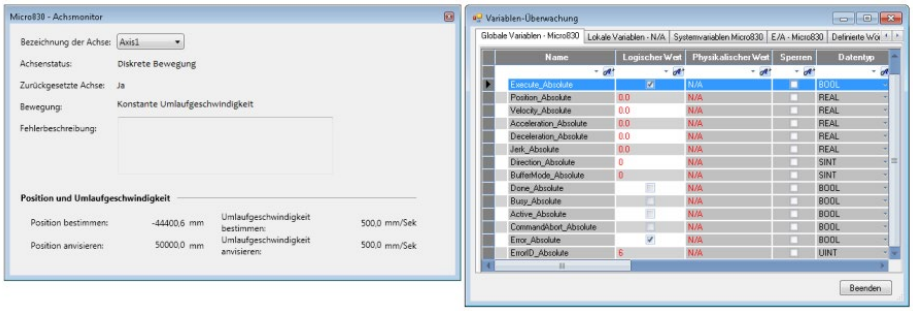
MC_MoveAbsolute-Kontaktplan-Beispiel



Beispiel für MC_MoveAbsolute in strukturiertem Text

```
Position_Absolute := 50000.0;
Velocity_Absolute := 500.0;
Acceleration_Absolute := 1000.0;
Deceleration_Absolute := 1000.0;
Jerk_Absolute := 10.0;
MC_MoveAbsolute_1(Axis1,Execute_Absolute,Position_Absolute,Velocity_Absolute,
Acceleration_Absolute,Deceleration_Absolute,Jerk_Absolute,Direction_Absolute,BufferMode_Absolute);
Done_Absolute := MC_MoveAbsolute_1.Done;
Busy_Absolute := MC_MoveAbsolute_1.Busy;
Active_Absolute := MC_MoveAbsolute_1.Active;
CommandAbort_Absolute := MC_MoveAbsolute_1.CommandAborted;
Error_Absolute := MC_MoveAbsolute_1.Error;
ErrorID_Absolute := MC_MoveAbsolute_1.ErrorID;
```

Ergebnisse



MC_MoveRelative (Bewegungssteuerung relative Bewegung)

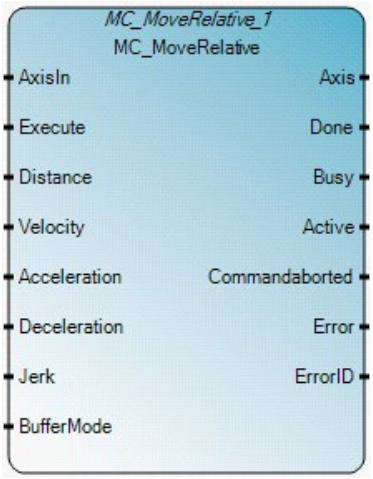
Befiehlt eine gesteuerte Bewegung eines angegebenen Abstands im Verhältnis zur tatsächlichen Position zum Zeitpunkt der Ausführung.

Details zum Betrieb:

- Da die Bewegungsrichtung für MC_MoveRelative durch die aktuelle Position und die Zielposition bestimmt wird, wird das Vorzeichen der Geschwindigkeit ignoriert.
- MoveRelative wird mit Geschwindigkeit Null beendet, wenn der Vorgang nicht durch einen anderen Funktionsblock überschrieben wird.
- Wird MC_MoveRelative beim Achsenstatus „Stillstand“ des Micro800-Controllers und bei einem relativen Abstand von Null zur Bewegung ausgegeben, wird die Ausführung des Funktionsblocks umgehend mit „Fertig“ gemeldet.
- Bei einem Micro800-Controller wird das Vorzeichen der Eingangsgeschwindigkeit für MC_MoveRelative ignoriert, da die Bewegungsrichtung durch die aktuelle Position und die Zielposition bestimmt wird.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung



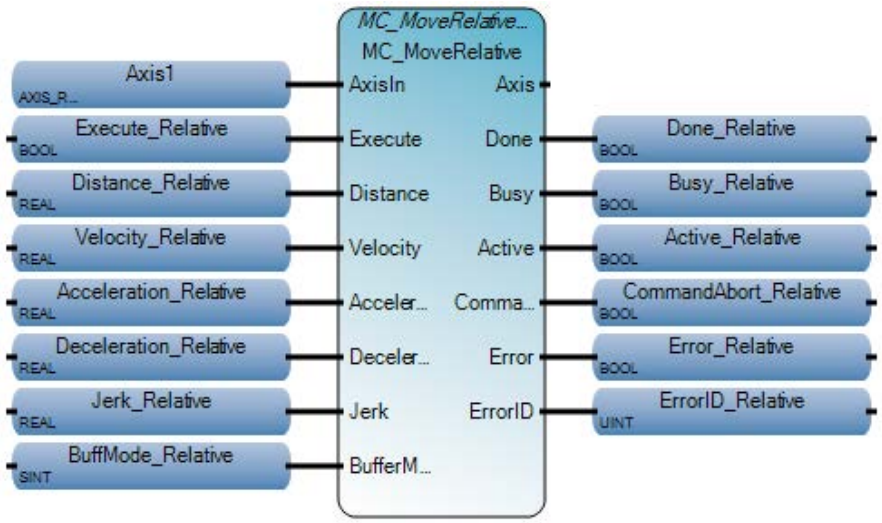
MC_MoveRelative-Vorgang

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

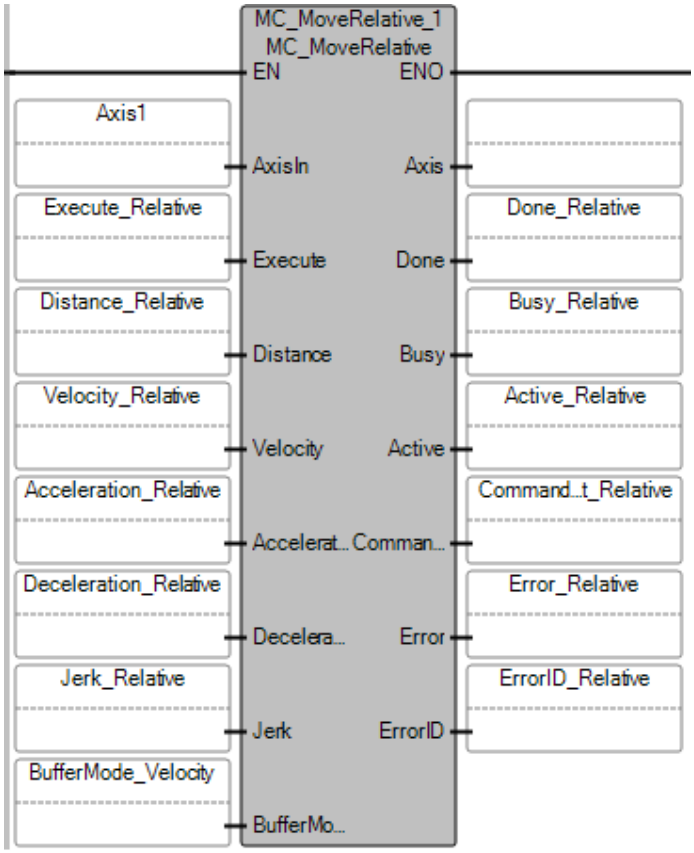
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_MoveRelative-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.

Parameter	Parametertyp	Datentyp	Beschreibung
AxisIn	Eingang	AXIS_REF auf seite 413	Definieren Sie AxisIn mithilfe der Parameter des Datentyps AXIS_REF.
Execute	Eingang	BOOL	Gibt an, wann die Bewegung beginnen soll. TRUE – Bewegung an der Anstiegsflanke beginnen. FALSE – Bewegung nicht beginnen.
Distance	Eingang	REAL	Relative Entfernung für die Bewegung (in technischer Einheit [u]).
Velocity	Eingang	REAL	Wert der Höchstgeschwindigkeit (wird nicht unbedingt erreicht) [u/s]. Da die Ermittlung der Bewegungsrichtung durch die Eingabe „Position" erfolgt, wird das Vorzeichen von „Velocity" (Geschwindigkeit) vom Funktionsblock ignoriert. Wenn „Stoß" = 0, kann die Höchstgeschwindigkeit nicht erreicht werden.
Acceleration	Eingang	REAL	Wert der Beschleunigung (zunehmende Energie des Motors) [u/s2]
Deceleration	Eingang	REAL	Wert der Verzögerung (abnehmende Energie des Motors) [u/s2]
Jerk	Eingang	REAL	Wert des Stoßes [u/s^3]
BufferMode	Eingang	SINT	Dieser Parameter wird nicht verwendet.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.
Done	Ausgang	BOOL	TRUE – angewiesene Entfernung erreicht. Wenn die Eingabe „In-Position" für eine Achse aktiviert ist, muss das Signal „In-Position" aktiv geschaltet werden, bevor „Done" zu „True" wechselt.
Busy	Ausgang	BOOL	TRUE – der Anweisungsblock ist nicht abgeschlossen. FALSE – der Anweisungsblock ist abgeschlossen.
Active	Ausgang	BOOL	TRUE – gibt an, dass der Anweisungsblock die Steuerung über die Achse besitzt.
CommandAborted	Ausgang	BOOL	TRUE – Befehl wurde durch einen anderen Befehl oder durch ein Anhalten aufgrund eines Fehlers überschrieben.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.

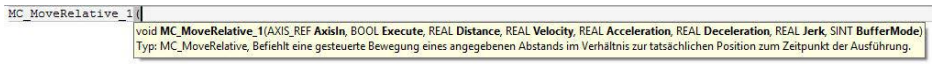
Diagrammbeispiel für Funktionsblock MC_MoveRelative



MC_MoveRelative-Kontaktplan-Beispiel

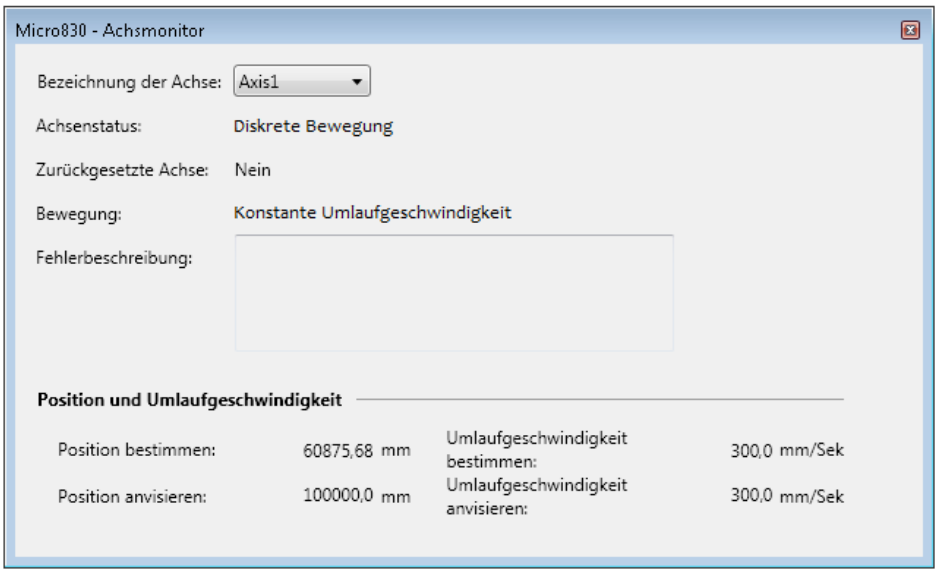


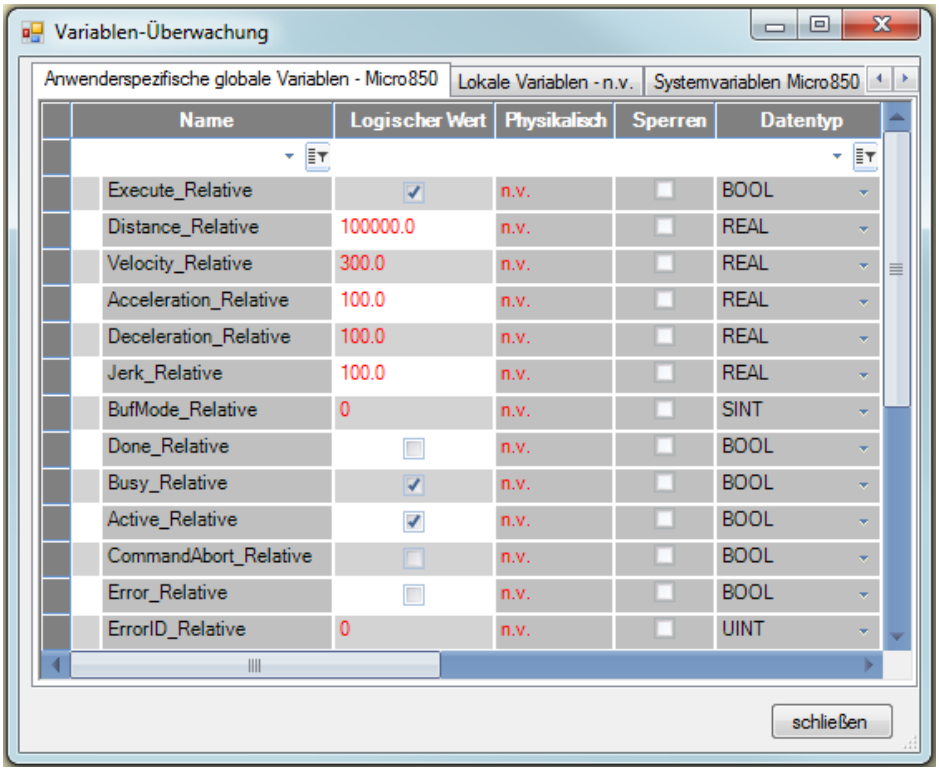
Beispiel für MC_MoveRelative in strukturiertem Text




```
Distance_Relative := 100000.0;
Velocity_Relative := 300.0;
Acceleration_Relative := 100.0;
Deceleration_Relative := 100.0;
Jerk_Relative := 100.0;
MC_MoveRelative_1(Axis1,Execute_Relative,Distance_Relative,
Velocity_Relative,Acceleration_Relative,Deceleration_Relative,Jerk_Relative,BuffMode_Relative);
Done_Relative := MC_MoveRelative_1.Done;
Busy_Relative := MC_MoveRelative_1.Busy;
Active_Relative := MC_MoveRelative_1.Active;
CommandAbort_Relative := MC_MoveRelative_1.Commandaborted;
Error_Relative := MC_MoveRelative_1.Error;
ErrorID_Relative := MC_MoveRelative_1.ErrorID;
```

Ergebnisse





MC_MoveVelocity (Bewegungssteuerung – Bewegungsgeschwindigkeit)

Befiehlt eine gesteuerte Endlosbewegung mit einer angegebenen Geschwindigkeit.

Details zum Betrieb:

- Wenn die Eingabe „DirectionIn“ für MC_MoveVelocity gleich 0 ist und:
  - sich die Achse bewegt, muss das Vorzeichen der Geschwindigkeitseingabe ignoriert werden. Die Achse bewegt sich weiter in ihrer aktuellen Richtung, und neue dynamische Parameter werden angewendet.

- die Achse keinen Achsenzustand besitzt, meldet MC_MoveVolecity einen Fehler.

- Wird beim Ausführen von MC_MoveVelocity der PTO-Impulsgrenzwert erreicht, wird der PTO-Akkumulatorwert auf o gesetzt (oder auf das entgegengesetzte Softwarelimit, wenn das Limit aktiviert ist), und die Ausführung wird fortgesetzt.
- Wenn sich die Achse bewegt und MC_MoveVelocity eine Bewegung ausgibt, in der die Richtung (Vorzeichen von Geschwindigkeit * Richtung) entgegen der aktuellen Bewegungsrichtung verläuft, meldet MC_MoveVelocity einen Fehler.
- Sobald das Signal „InVelocity“ gesetzt ist, zeigt es an, dass MC_MoveVelocity beendet wurde. Jedes nachfolgende Achssteuerungsereignis hat keine Auswirkung auf die Ausgaben von MC_MoveVelocity, ausgenommen das Signal „InVelocity“.
- Der „InVelocity“-Ausgang von MC_MoveVelocity bleibt „Wahr“, sobald die Geschwindigkeit der Achse die befohlene Geschwindigkeit erreicht und solang bis MC_MoveVelocity angehalten wird.
- Das Vorzeichen von (Geschwindigkeit * Richtung) legt die Bewegungsrichtung für MC_MoveVelocity fest. Wenn Geschwindigkeit und Richtung dasselbe Vorzeichen haben, wird eine positive Richtung ausgegeben. Wenn Geschwindigkeit und Richtung unterschiedliche Vorzeichen haben, wird eine negative Richtung ausgegeben.
- Das Signal „InVelocity“ wird an der Abstiegflanke von „Ausführen“ zurückgesetzt oder wenn MC_MoveVelocity von einem anderen Funktionsblock bzw. einem anderen Achssteuerungsereignis überschrieben wird.
- Um die von MC_MoveVelocity initiierte Bewegung anzuhalten bzw. zu ändern, muss der Befehlsblock von einem anderen Befehlsblock unterbrochen oder überschrieben werden. Dazu gehört auch die erneute Ausführung von MC_MoveVelocity mit anderen Parametern.
- Wenn MC_MoveVelocity mit dem Achsenstatus „StandStill“ (nicht gesteuert durch einen anderen Funktionsblock) ausgegeben wird und ein Funktionsblockfehler auftritt, wechselt die Achse in den Zustand „ErrorStop“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

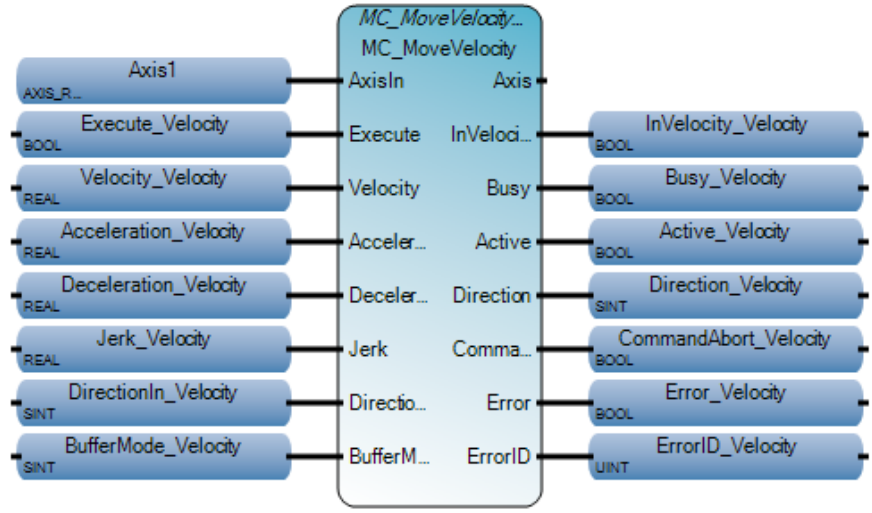
Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

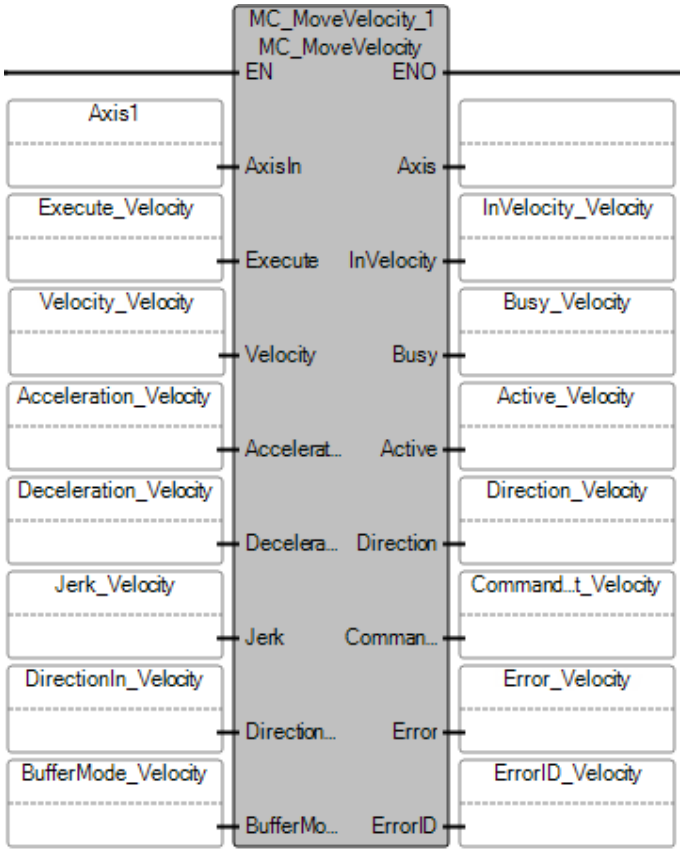
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_MoveVelocity-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF	Definieren Sie AxisIn mithilfe der Parameter des Datentyps AXIS_REF.
Execute	Eingang	BOOL	Gibt an, wann die Bewegung beginnen soll. TRUE – Bewegung an der Anstiegsflanke beginnen. FALSE – Bewegung nicht beginnen.
Velocity	Eingang	REAL	Wert der Höchstgeschwindigkeit [u/s].
Acceleration	Eingang	REAL	Wert der Beschleunigung (zunehmende Energie des Motors) [u/s^2]
Deceleration	Eingang	REAL	Wert der Verzögerung (abnehmende Energie des Motors) [u/s^2]
Jerk	Eingang	REAL	Wert des Stoßes [u/s^3]
DirectionIn	Eingang	SINT	Gültige Werte sind: -1, 0, 1.
BufferMode	Eingang	SINT	Dieser Parameter wird nicht verwendet.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF auf seite 413	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.
InVelocity	Ausgang	BOOL	TRUE – die Geschwindigkeitsvorgabe wurde erreicht (erstmalig).
Busy	Ausgang	BOOL	TRUE – der Anweisungsblock ist nicht abgeschlossen. FALSE – der Anweisungsblock ist abgeschlossen.
Active	Ausgang	BOOL	TRUE – gibt an, dass der Funktionsblock die Steuerung über die Achse besitzt.
Direction	Ausgang	SINT	Gültige Werte sind: -1, 0, 1.
CommandAborted	Ausgang	BOOL	TRUE – Befehl wurde durch einen anderen Befehl oder durch ein Anhalten aufgrund eines Fehlers überschrieben.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.

Parameter	Parametertyp	Datentyp	Beschreibung
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.

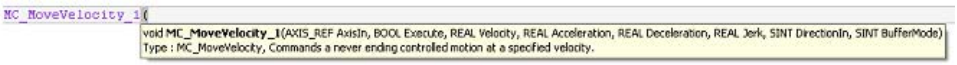
Diagrammbeispiel für Funktionsblock MC_MoveVelocity



MC_MoveVelocity-Kontaktplan-Beispiel

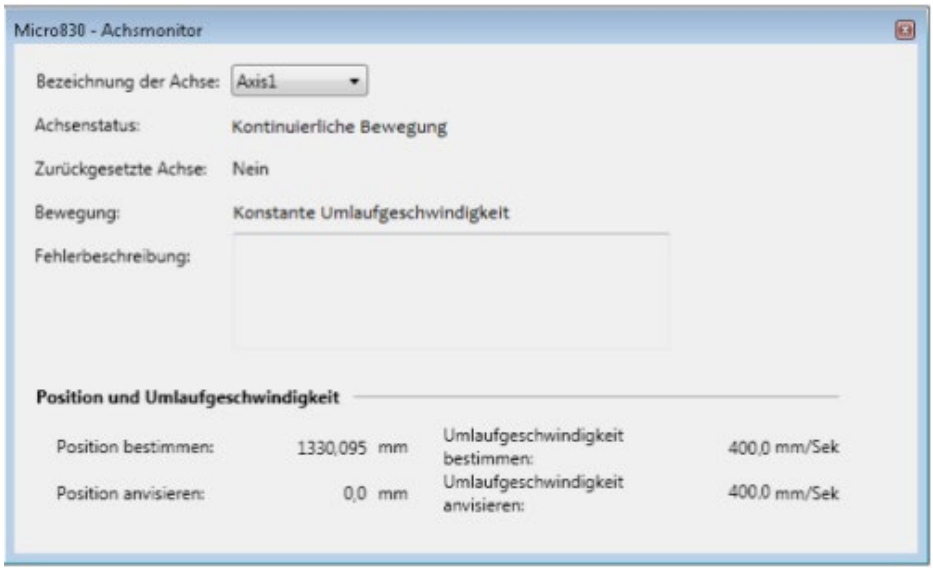


Beispiel für MC_MoveVelocity in strukturiertem Text




```
Velocity_Velocity := 400.0;
Acceleration_Velocity := 100.0;
Deceleration_Velocity := 100.0;
Jerk_Velocity := 100.0;
DirectionIn_Velocity := 1;
MC_MoveVelocity_1(Axis1,Execute_Velocity,Velocity_Velocity,
Acceleration_Velocity,Deceleration_Velocity,Jerk_Velocity,DirectionIn_Velocity,BufferMode_Velocity);
InVelocity_Velocity := MC_MoveVelocity_1.InVelocity;
Busy_Velocity := MC_MoveVelocity_1.Busy;
Active_Velocity := MC_MoveVelocity_1.Active;
Direction_Velocity := MC_MoveVelocity_1.Direction;
CommandAbort_Velocity := MC_MoveVelocity_1.CommandAborted;
Error_Velocity := MC_MoveVelocity_1.Error;
ErrorID_Velocity := MC_MoveVelocity_1.ErrorID;
```

Ergebnisse



Name	Alias	Logischer Wert	Physikalischer Wert	Startwert	Sperren	Datentyp	Dime
Execute_Velocity		☑	n.v.		☐	BOOL	
Velocity_Velocity		400.0	n.v.		☐	REAL	
Acceleration_Velocity		100.0	n.v.		☐	REAL	
Deceleration_Velocity		100.0	n.v.		☐	REAL	
Jerk_Velocity		100.0	n.v.		☐	REAL	
DirectionIn_Velocity		1	n.v.		☐	SINT	
BufferMode_Velocity		0	n.v.		☐	SINT	
InVelocity_Velocity		☑	n.v.		☐	BOOL	
Busy_Velocity		☐	n.v.		☐	BOOL	
Active_Velocity		☐	n.v.		☐	BOOL	
Direction_Velocity		☐	n.v.		☐	BOOL	
CommandAbort_Velocity		☐	n.v.		☐	BOOL	
Error_Velocity		☐	n.v.		☐	BOOL	
ErrorID_Velocity		0	n.v.		☐	UINT	

MC_Power (Bewegungssteuerung – Leistung)

Steuert die Leistungsstufe (EIN oder AUS).

Details zum Betrieb:

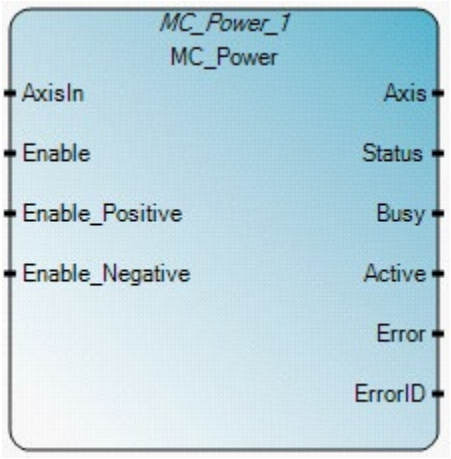
- Wenn Sie ein in CCW7 erstelltes Projekt in CCW8 importieren, wird der neue Mc_Power-Eingabeparameter __DTI_AxisIn angezeigt.

Wählen Sie im Falle eines Fehlers bei der Erstellung die Anweisung neu, und wiederholen Sie die Erstellung.

- Nachdem die Achse eingeschaltet wurde, wird der Achsenstatus „Homed“ (Zurückgesetzt) auf 0 (nicht zurückgesetzt) zurückgesetzt.
- Die Eingaben „Enable_Positive“ und „Enable_Negative“ für „MC_Power“ sind pegelausgelöst; sie werden geprüft, wenn die Aktivierungseingabe von OFF zu ON wechselt. Der fliegende Wechsel für die Eingaben „Enable_Positive“ und „Enable_Negative“ ohne Umschalten der Aktivierungseingabe wird nicht geprüft.
- Bei einem Stromausfall während des Betriebs (wenn Servoanschluss erkannt wurde) wechselt der Achsenstatus zu „ErrorStop“ (Fehler-Stopp).
- Der MC_Power-Befehl hat einen Zeitüberschreitungswert von 2 Minuten. MC_Power gibt einen Fehler zurück, wenn die Zeitüberschreitung abläuft und der Antriebsbereitschaftseingang FALSE ist.
- Wenn der Funktionsblock „MC_Power“ mit auf „True“ gesetzter Aktivierungseingabe aufgerufen wird, während sich die Achse im Zustand „Disabled“ (Deaktiviert) befindet, wechselt die Achse in den Zustand „StandStill“ (Stillstand), außer wenn ein Fehler vorliegt, dann wechselt die Achse in den Zustand „ErrorStop“ (Fehler-Stopp).
- Es darf nur ein Funktionsblock „MC_Power“ pro Achse ausgegeben werden. Die Verwendung eines anderen „MC_ Power“-Funktionsblocks zum simultanen Steuern derselben Achse wird vom Bewegungsmotor zurückgewiesen.
- Wenn ein Ein/Aus-Schalter für eine Achse vorhanden ist, wird die absolute Achsenposition nicht zurückgesetzt.
- Wenn der Funktionsblock „MC_Power“ mit Enable = False aufgerufen wird, wechselt die Achse bei jedem Zustand einschließlich „ErrorStop“ (Fehler-Stopp) in den Zustand „Disabled“ (Deaktiviert).
- Der MC_Power-Funktionsblock kann die Achse einschalten, wenn „Enable = True“, oder ausschalten, wenn „Enable = False“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

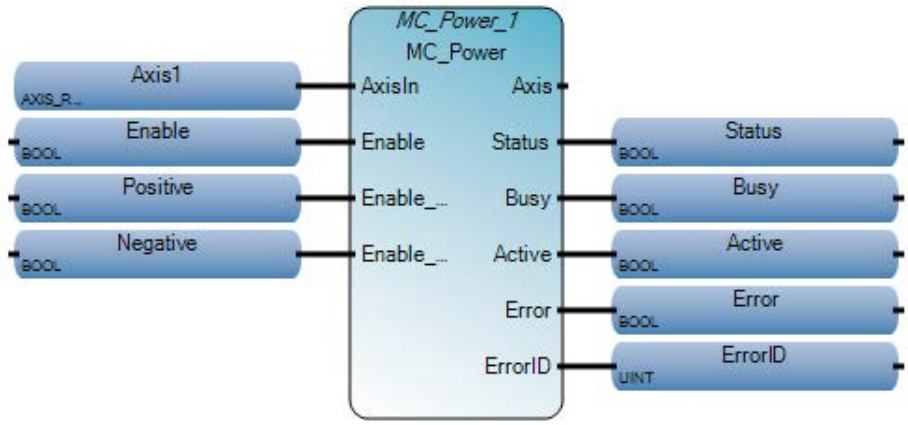
Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung



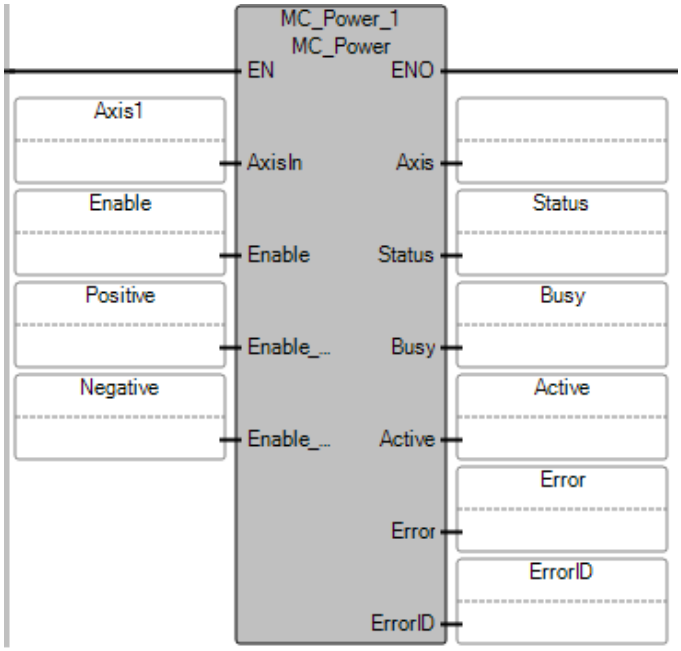
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_Power-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF FB_AXIS_REF	Definieren Sie AxisIn mithilfe der Parameter des Datentyps AXIS_REF auf seite 413. Bei FB_Axis (Feedbackachse) verwenden Sie den Datentyp FB_AXIS_REF auf seite 414 zum Definieren der AxisIn-Parameter.
Enable	Eingang	BOOL	TRUE – der Strom ist eingeschaltet. FALSE – der Strom ist ausgeschaltet.
Enable_Positive	Eingang	BOOL	TRUE - Bewegung in die positive Richtung ist zulässig.
Enable_Negative	Eingang	BOOL	TRUE - Bewegung in die negative Richtung ist zulässig.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt. Datentyp AXIS_REF.
Status	Ausgang	BOOL	Status der Leistungsstufe: • TRUE – das Einschalten des Antriebs ist abgeschlossen. • FALSE – das Einschalten des Antriebs ist nicht abgeschlossen.
Busy	Ausgang	BOOL	TRUE – der Anweisungsblock ist nicht abgeschlossen. FALSE – der Anweisungsblock ist abgeschlossen.
Active	Ausgang	BOOL	TRUE – gibt an, dass der Funktionsblock die Steuerung über die Achse besitzt.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.

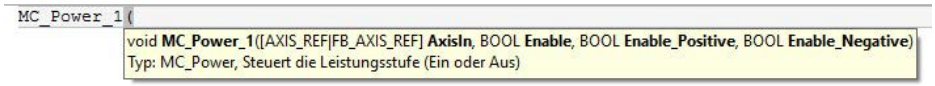
Diagrammbeispiel für Funktionsblock MC_Power



MC_Power-Kontaktplan-Beispiel

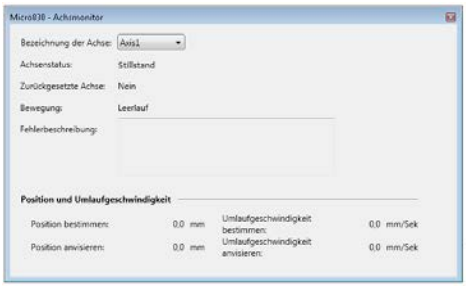


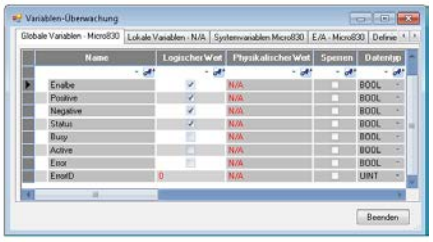
Beispiel für MC_Power in strukturiertem Text




```
Positive := True;
Negative := True;
MC_Power_1(Axis1,Enable,Positive,Negative);
Status := MC_Power_1.Status;
Busy := MC_Power_1.Busy;
Active := MC_Power_1.Active;
Error := MC_Power_1.Error;
ErrorID := MC_Power_1.ErrorID;
```

Ergebnisse





MC_ReadActualPosition (Servosteuerung liest tatsächliche Position)

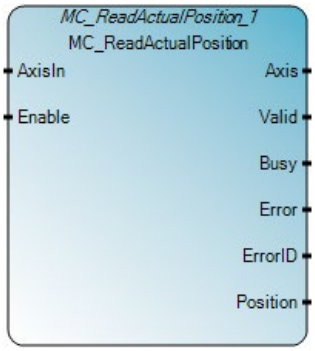
Gibt die tatsächliche Position der Feedbackachse zurück. MC_ReadActualPosition gilt nur für Feedback-Bewegungen.

Details zum Betrieb:

- Überprüfen Sie vor der Ausführung von MC_ReadActualPosition, ob sich die Achse in einem der folgenden Zustände befindet:
  - Deaktiviert
  - Stillstand
  - Diskrete Bewegung
  - Fehler-Stopp
- Die tatsächliche Position für eine Feedback-Achse wird nach dem Download nicht auf 0 zurückgesetzt. Zum Zurücksetzen oder Deaktivieren der Position für die Feedback-Achse verwenden Sie die Anweisung „MC_Home“ oder „MC_SetPosition“.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung

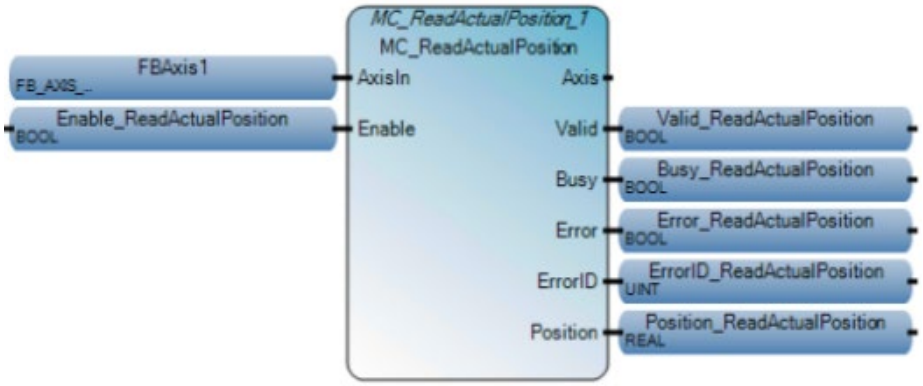


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

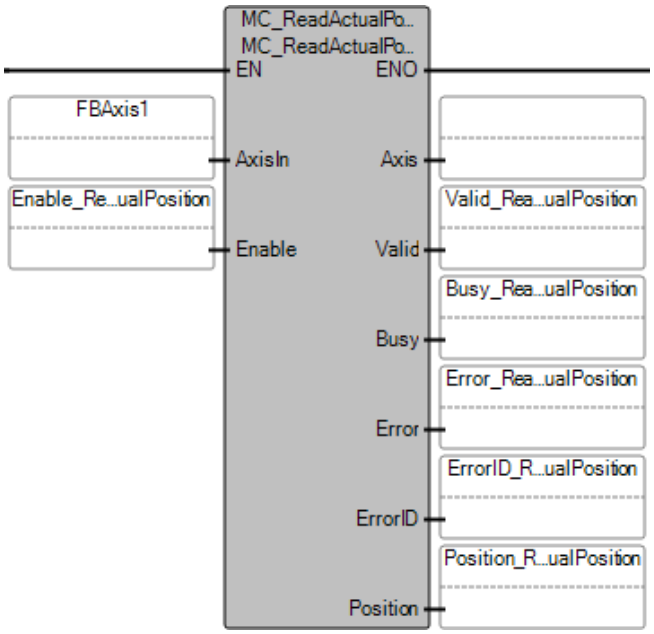
Parameter	Parametertyp	Datentyp	Beschreibung
AxisIn	Eingang	FB_AXIS_REF	Bei FB_Axis (Feedbackachse) verwenden Sie den Datentyp FB_AXIS_REF auf seite 414 zum Definieren der AxisIn-Parameter.
Enable	Eingang	BOOL	TRUE – ruft bei Aktivierung den Wert des Parameters fortlaufend ab. FALSE – im Leerlauf.

Parameter	Parametertyp	Datentyp	Beschreibung
Axis	Ausgang	FB_AXIS_REF	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt. Die Ausgangsparameter der Achse werden im Datentyp FB_AXIS_REF definiert.
Valid	Ausgang	BOOL	TRUE – der Anweisungsblock ist aktiv, und es werden neue Ausgabewerte erwartet. FALSE – der Anweisungsblock ist inaktiv.
Busy	Ausgang	BOOL	TRUE – der Anweisungsblock ist nicht abgeschlossen. FALSE – der Anweisungsblock ist abgeschlossen.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UNIT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.
Position	Ausgang	REAL	Der Wert der tatsächlichen absoluten Position für die Achse der Feedback-Bewegungen. (in Achseneinheit [u])

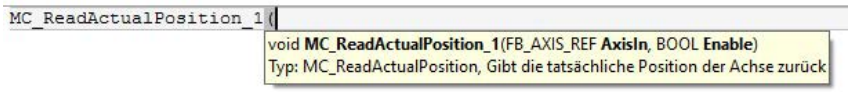
Diagrammbeispiel für Funktionsblock MC_ReadActualPosition



Beispiel für MC_ReadActualPosition-Kontaktplan

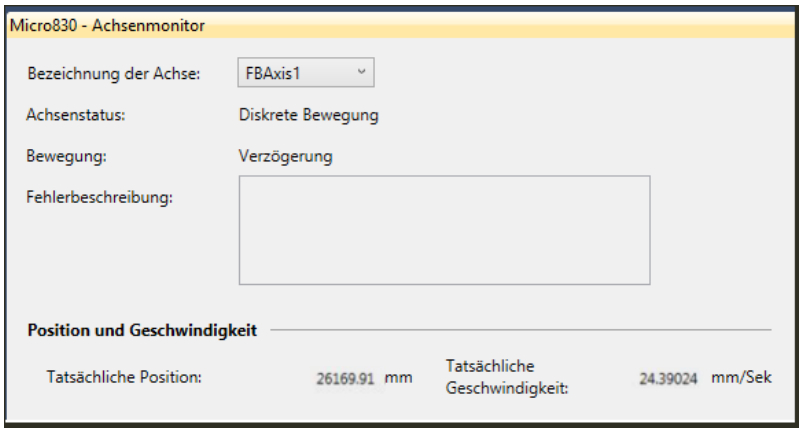


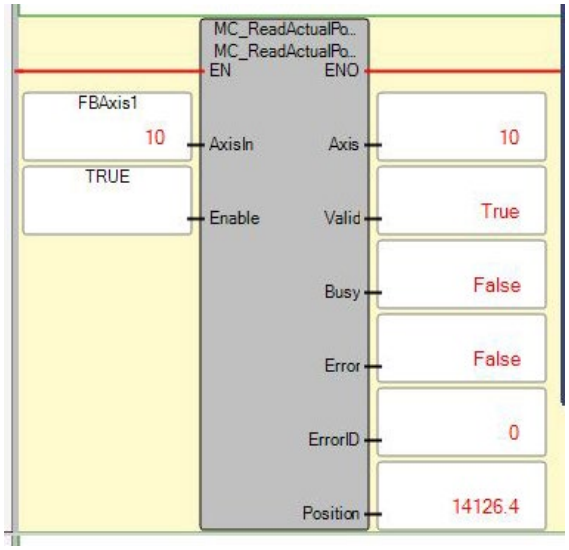
Beispiel für strukturierten MC_ReadActualPosition-Text




```
MC_ReadActualPosition_1(FBAxis, Enable_ReadActualPosition)
Valid_ReadActuralPosition := MC_ReadActualPosition_1.Valid
Busy_ReadActuralPosition := MC_ReadActualPosition_1.Busy
Error_Valid_ReadActuralPosition := MC_ReadActualPosition_1.Error
ErrorID_Valid_ReadActuralPosition := MC_ReadActualPosition_1.ErrorID
Position_Valid_ReadActuralPosition := MC_ReadActualPosition_1.Position
```

Ergebnisse





	Name	Alias	Logischer Wert
-	MC_ReadActualPosition_2		...
+	MC_ReadActualPosition_2.AxisIn		...
	MC_ReadActualPosition_2.Enable		☑
+	MC_ReadActualPosition_2.Axis		...
	MC_ReadActualPosition_2.Valid		☑
	MC_ReadActualPosition_2.Busy		☐
	MC_ReadActualPosition_2.Error		☐
	MC_ReadActualPosition_2.ErrorID		0
	MC_ReadActualPosition_2.Position		14126.39

MC_ReadActualVelocity (Servosteuerung liest tatsächliche Geschwindigkeit)

Gibt den Wert für die tatsächliche Geschwindigkeit der Feedbackachse zurück.

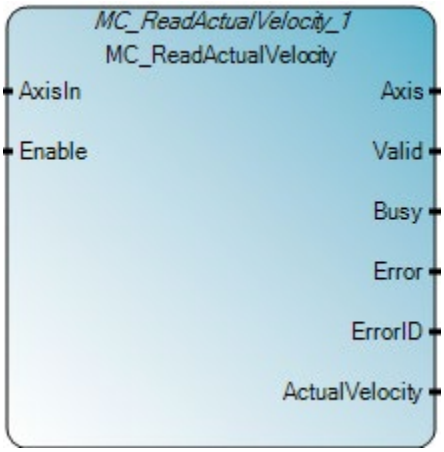
MC_ReadActualVelocity gilt nur für Feedback-Bewegungen.

Überprüfen Sie vor der Ausführung von MC_ReadActualVelocity, ob sich die Achse in einem der folgenden Zustände befindet:

- Stillstand
- Diskrete Bewegung
- Fehler-Stopp

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung

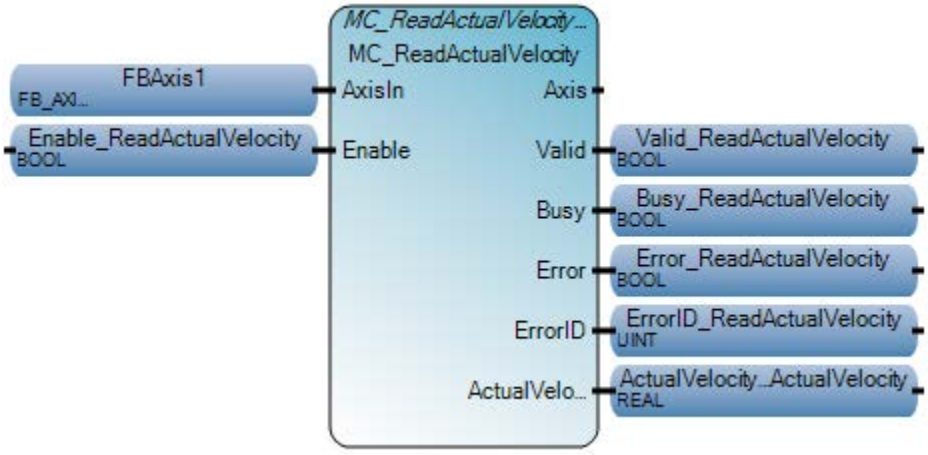


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

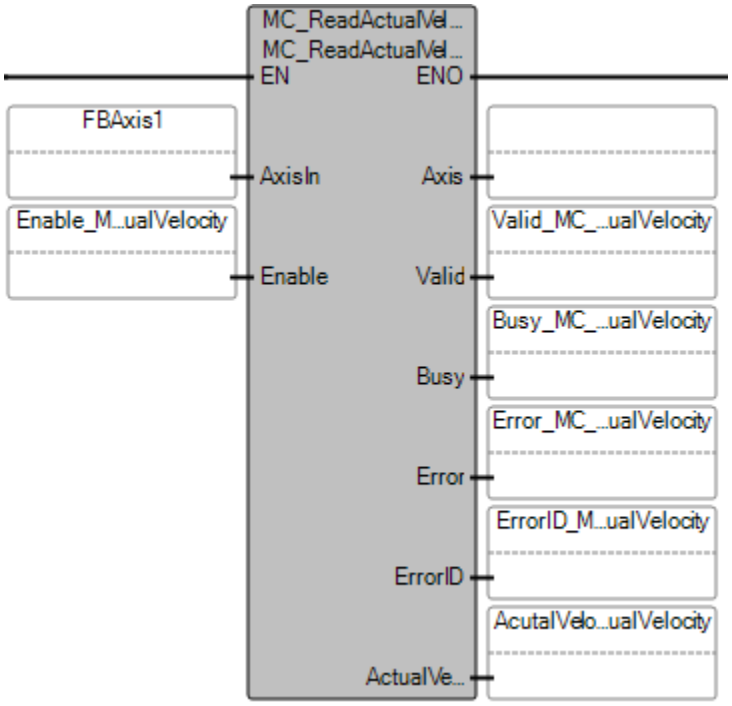
Parameter	Parametertyp	Datentyp	Beschreibung
AxisIn	Eingang	FB_AXIS_REF	Bei FB_Axis (Feedbackachse) verwenden Sie den Datentyp FB_AXIS_REF zum Definieren der AxisIn-Parameter.
Enable	Eingang	BOOL	TRUE – den Wert für die tatsächliche Geschwindigkeit kontinuierlich abrufen. FALSE – die Daten sind nicht mehr gültig, alle Ausgaben werden auf 0 zurückgesetzt, und „Gültig" wird auf „False" gesetzt.
Axis	Ausgang	FB_AXIS_REF	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt. Die Ausgangsparameter der Achse werden im Datentyp FB_AXIS_REF auf seite 414 definiert.
Valid	Ausgang	BOOL	TRUE – der Funktionsblock ist aktiv, und es werden neue Ausgabewerte erwartet. FALSE – die Funktion erwartet keine neuen Ausgabewerte.
Busy	Ausgang	BOOL	TRUE – der Funktionsblock wird nicht beendet. FALSE – die Funktion befindet sich im Leerlauf.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UNIT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.

Parameter	Parametertyp	Datentyp	Beschreibung
ActualVelocity	Ausgang	REAL	Der Wert für die tatsächliche Geschwindigkeit der Feedback-Bewegungsachse (in Achseneinheit [u/s]). ActualVelocity ist ein Wert mit Vorzeichen, der Richtungsinformationen enthält.

Diagrammbeispiel für Funktionsblock MC_ReadActualVelocity



Beispiel für MC_ReadActualVelocity-Kontaktplan



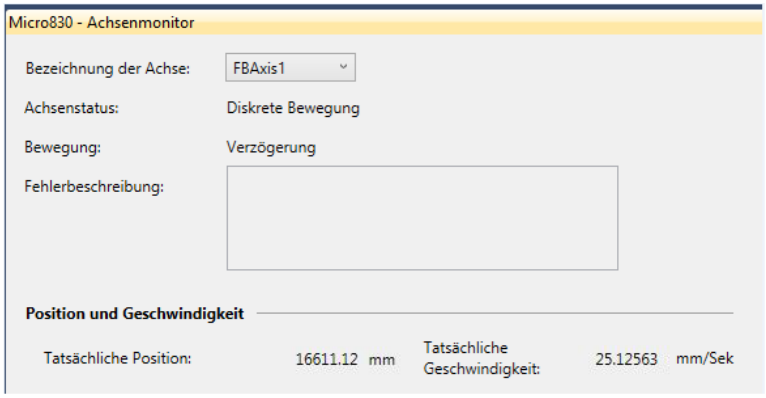
Beispiel für strukturierten MC_ReadActualVelocity-Text

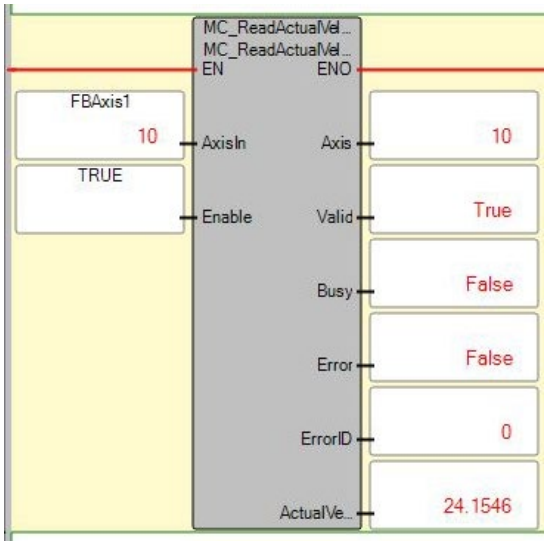
```
MC_ReadActualVelocity_1(
```

void **MC_ReadActualVelocity_1**(FB_AXIS_REF **AxisIn**, BOOL **Enable**)
Typ: MC_ReadActualVelocity, Gibt die tatsächliche Position der Achse zurück

```
MC_ReadActualVelocity_1(FBAxis1,Enable_ReadActualVelocity)
Valid_ReadActualVelocity := MC_ReadActualVelocity_1.Valid
Busy_ReadActualVelocity := MC_ReadActualVelocity_1.Busy
Error_ReadActualVelocity := MC_ReadActualVelocity_1.Error
ErrorID_ReadActualVelocity := MC_ReadActualVelocity_1.ErrorID
Position_ReadActualVelocity := MC_ReadActualVelocity_1.ErrorID
```

Ergebnisse





	Name	Alias	Logischer Wert
▶ -	MC_ReadActualVelocity_1		...
+	MC_ReadActualVelocity_1.AxisIn		...
	MC_ReadActualVelocity_1.Enable		☑
+	MC_ReadActualVelocity_1.Axis		...
	MC_ReadActualVelocity_1.Valid		☑
	MC_ReadActualVelocity_1.Busy		☐
	MC_ReadActualVelocity_1.Error		☐
	MC_ReadActualVelocity_1.ErrorID		0
	MC_ReadActualVelocity_1.ActualVelocity		24.1545887

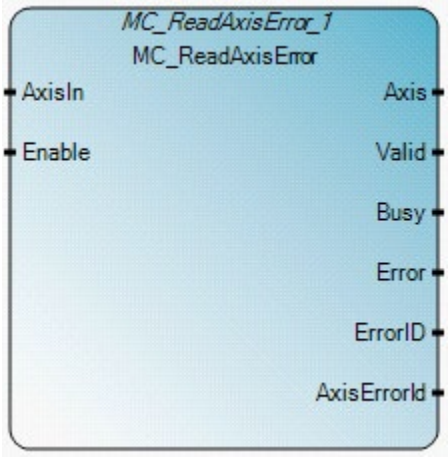
MC_ReadAxisError (Servosteuerung liest Achsenfehler)

Liest die Achsfehler, die in keinem Zusammenhang mit den Bewegungssteuerungs-Funktionsblöcken stehen.

Wenn sich eine Achse im Zustand „Disabled“ (Deaktiviert) befindet, kann MC_ReadAxisError eine Fehler-ID ungleich null für die Achse erhalten oder nicht, da eine deaktivierte Achse Fehler enthalten oder fehlerfrei sein kann.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung

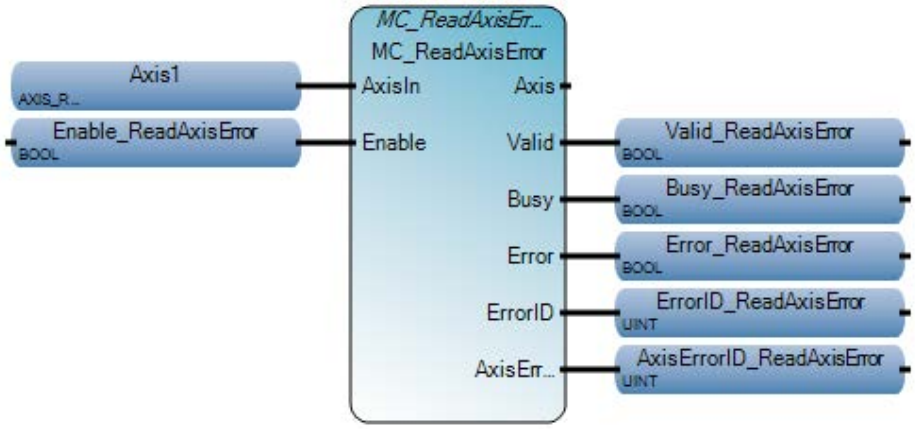


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

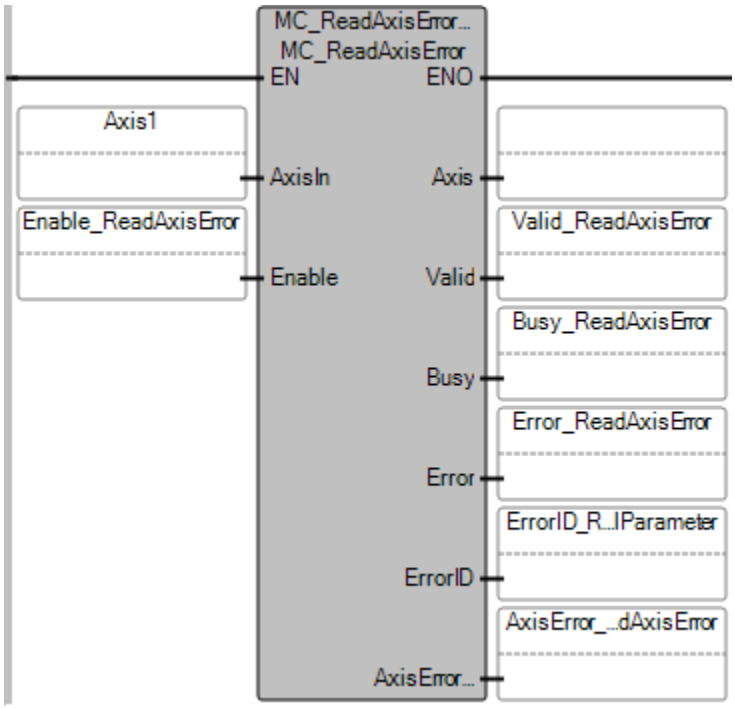
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_ReadAxisError-Berechnung ausführen. FALSE – die Werte Error, ErrorID und AxisErrorID werden auf „False“ (oder 0) zurückgesetzt. Gilt nur für KOP-Programme.
AxisIn	Eingang	AXIS_REF FB_AXIS_REF	Definieren Sie AxisIn mithilfe der Parameter des Datentyps AXIS_REF auf seite 413. Bei FB_Axis (Feedbackachse) verwenden Sie den Datentyp FB_AXIS_REF zum Definieren der AxisIn-Parameter.
Enable	Eingang	BOOL	TRUE – ruft bei Aktivierung den Wert des Parameters fortlaufend ab. FALSE – die Ausgaben Error, ErrorID und AxisErrorID werden auf 0 zurückgesetzt.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt. Die Ausgangsparameter der Achse werden im Datentyp FB_AXIS_REF auf seite 414 definiert.
Valid	Ausgang	BOOL	TRUE – der Anweisungsblock ist aktiv, und es werden neue Ausgabewerte erwartet. FALSE – der Anweisungsblock ist inaktiv.
Busy	Ausgang	BOOL	TRUE – der Anweisungsblock ist nicht abgeschlossen. FALSE – der Anweisungsblock ist abgeschlossen.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.

Parameter	Parametertyp	Datentyp	Beschreibung
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks definiert.
AxisErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Achsenfehler angibt. Die Fehler bei diesem Befehl sind in AxisErrorID-Fehlercodes auf seite 447 definiert.

Beispiel für ein MC_ReadAxisError-Funktionsblockdiagramm



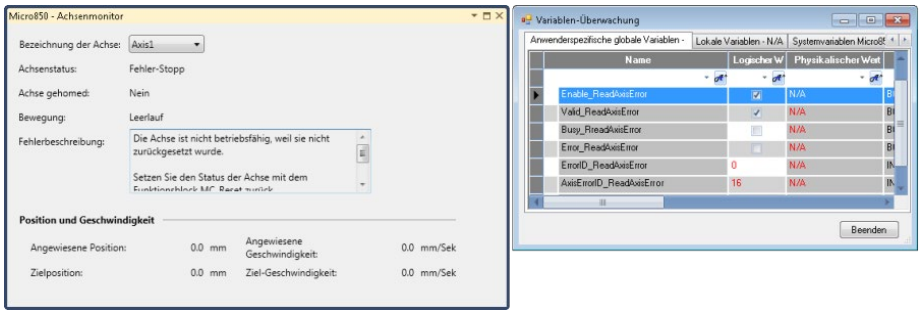
Beispiel für einen MC_ReadAxisError-Kontaktplan



Beispiel für einen strukturierten MC_ReadAxisError-Text

```
MC_ReadAxisError_1(Axis1,Enable_ReadAxisError);
Valid_ReadAxisError := MC_ReadAxisError_1.Valid;
Busy_ReadAxisError := MC_ReadAxisError_1.Busy;
Error_ReadAxisError := MC_ReadAxisError_1.Error;
ErrorID_ReadAxisError := MC_ReadAxisError_1.ErrorID;
AxisErrorID_ReadAxisError := MC_ReadAxisError_1.AxisErrorId;
```

Ergebnisse



AxisErrorID-Fehlercodes

In der folgenden Tabelle werden die Achsenfehler-Codes beschrieben, die im Parameter „AxisErrorID“ für MC_ReadAxisError auf seite 445 angegeben werden.

Wert	MAKRO-ID	Beschreibung
00	MC_FB_ERR_NO	Die Achse befindet sich in einem betriebsbereiten Zustand (es gibt nichts anzuzeigen).
01	MC_FB_ERR_WRONG_STATE	Die Achse ist aufgrund eines während der Ausführung eines Funktionsblocks erkannten ungültigen Achsenstatus nicht betriebsbereit. Setzen Sie den Status der Achse mit den Funktionsblöcken MC_Power und MC_Reset zurück.
02	MC_FB_ERR_RANGE	Die Achse ist aufgrund eines im Funktionsblock als ungültig festgelegten achsendynamischen Parameters (Geschwindigkeit, Beschleunigung, Verzögerung oder Stoß) nicht betriebsbereit. Setzen Sie den Status der Achse mit den Funktionsblöcken MC_Power und MC_Reset zurück. Korrigieren Sie die Einstellungen für die dynamischen Parameter im Funktionsblock, die mit den Einstellungen auf der Konfigurationsseite für die Achsendynamik in Konflikt stehen.
03	MC_FB_ERR_PARAM	Die Achse ist aufgrund eines im Funktionsblock als ungültig festgelegten Parameters (außer Geschwindigkeit, Beschleunigung, Verzögerung oder Stoß) nicht betriebsbereit. Setzen Sie den Status der Achse mit den Funktionsblöcken MC_Power und MC_Reset zurück. Korrigieren Sie die Einstellung für die Parameter (z. B. Modus oder Position) im Funktionsblock.
04	MC_FB_ERR_AXISNUM	Interner Bewegungsfehler, Fehler-ID = 0x04. Wenden Sie sich an Ihren zuständigen technischen Supportmitarbeiter von Rockwell Automation. Kontaktinformationen erhalten Sie hier: http://www.rockwellautomation.com/support

Wert	MAKRO-ID	Beschreibung
05	MC_FB_ERR_ MECHAN	Die Achse ist aufgrund von Antriebsproblemen oder mechanischen Problemen nicht betriebsfähig. Prüfen Sie die Verbindung zwischen Antrieb und Controller (Signale für „Antrieb bereit" und „In Position"), und stellen Sie sicher, dass der Antrieb normal arbeitet. Setzen Sie den Status der Achse mit den Funktionsblöcken MC_Power und MC_Reset zurück.
06	MC_FB_ERR_ NOPOWER	Die Achse ist nicht eingeschaltet. Schalten Sie die Achse mit dem Funktionsblock „MC_Power" ein. Setzen Sie den Status der Achse mit dem Funktionsblock „MC_Reset" zurück.
07	MC_FB_ERR_ RESOURCE	Die Achse ist nicht betriebsbereit, da sie oder ihre von einem Funktionsblock benötigten Ressourcen von einem anderen Funktionsblock gesteuert werden oder nicht verfügbar sind. Stellen Sie sicher, dass die Achse oder ihre vom Funktionsblock benötigten Ressourcen zur Nutzung verfügbar sind. Setzen Sie den Status der Achse mit den Funktionsblöcken MC_Power und MC_Reset zurück. Überprüfen und korrigieren Sie die Anwendung, wenn mehrere Instanzen des gleichen Funktionsblocks zur gleichen Zeit versuchen, die Achse zu steuern.
08	MC_FB_ERR_ PROFILE	Die Achse ist nicht betriebsbereit, da das in einem Funktionsblock definierte Bewegungsprofil ungültig ist. Setzen Sie den Status der Achse mit den Funktionsblöcken MC_Power und MC_Reset zurück. Korrigieren Sie das Profil im Funktionsblock.
09	MC_FB_ERR_ VELOCITY	Die Achse ist nicht betriebsbereit, da das im Funktionsblock angeforderte Bewegungsprofil mit der aktuellen Achsengeschwindigkeit in Konflikt steht. Mögliche Ursachen: • Der Funktionsblock fordert während der Bewegung der Achse eine Richtungsumkehr. • Die aktuelle Geschwindigkeit ist für das angeforderte Bewegungsprofil zu niedrig oder zu hoch. Setzen Sie den Status der Achse mit den Funktionsblöcken MC_Power und MC_Reset zurück. Korrigieren Sie das Bewegungsprofil im Funktionsblock, oder führen Sie den Funktionsblock erneut aus, wenn die Achsengeschwindigkeit mit dem angeforderten Bewegungsprofil kompatibel ist.
10	MC_FB_ERR_ SOFT_LIMIT	Die Achse ist nicht betriebsbereit, da ein Fehler aufgrund einer weichen Grenze aufgetreten ist oder das Ausführen des Funktionsblocks zu einem Fehler aufgrund einer weichen Grenze führen würde. Setzen Sie den Status der Achse mit den Funktionsblöcken MC_Power und MC_Reset zurück. Prüfen Sie die Geschwindigkeits- oder Zielpositionseinstellungen für den Funktionsblock, oder passen Sie die Einstellung des weichen Grenzwerts an.
11	MC_FB_ERR_ HARD_LIMIT	Die Achse ist aufgrund eines Fehlers im Zusammenhang mit einer Hardware-Grenze nicht betriebsfähig. Setzen Sie den Status der Achse mit dem Funktionsblock „MC_Reset" zurück, und bewegen Sie die Achse dann in umgekehrter Richtung von der harten Grenze weg.
12	MC_FB_ERR_ LOG_LIMIT	Die Achse ist nicht betriebsbereit, da ein Fehler im Zusammenhang mit einem Grenzwert für die PTO-Akkumulator-Logik erkannt wurde oder das Ausführen des Funktionsblocks zu einem solchen Fehler führen würde. Setzen Sie den Status der Achse mit den Funktionsblöcken MC_Power und MC_Reset zurück. Prüfen Sie die Geschwindigkeits- oder Zielpositionseinstellungen für den Funktionsblock. Passen Sie das Koordinatensystem der Achse mit dem Funktionsblock „MC_SetPosition" an.
13	MC_FB_ERR_ ERR_ENGINE	Die Achse ist aufgrund einer Störung des Bewegungsmotors nicht betriebsfähig. Schalten Sie die gesamte Maschine aus und wieder ein, und laden Sie die Benutzeranwendung für den Controller erneut herunter. Bleibt der Fehler weiterhin bestehen, wenden Sie sich an Ihren zuständigen technischen Supportmitarbeiter von Rockwell Automation. Kontaktinformationen erhalten Sie hier: http://www.rockwellautomation.com/support.
16	MC_FB_ERR_ NOT_HOMED	Die Achse ist nicht betriebsfähig, weil sie nicht zurückgesetzt wurde. Setzen Sie den Status der Achse mit den Funktionsblöcken MC_Power und MC_Reset zurück. Setzen Sie die Achse mit dem Funktionsblock „MC_Home" zurück.

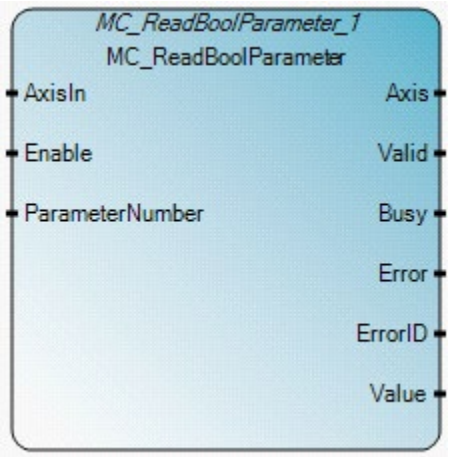
Wert	MAKRO-ID	Beschreibung
128	MC_FB_ERR_ PARAM_MODIFIED	Interne Bewegungswarnung, Warnungs-ID = 0x80. Wenden Sie sich an Ihren zuständigen technischen Supportmitarbeiter von Rockwell Automation. Kontaktinformationen erhalten Sie hier: http://www.rockwellautomation.com/support.

MC_ReadBoolParameter (Servosteuerung liest BOOL-Parameter)

Gibt den Wert eines anbieterspezifischen Parameters vom Typ BOOL zurück.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung

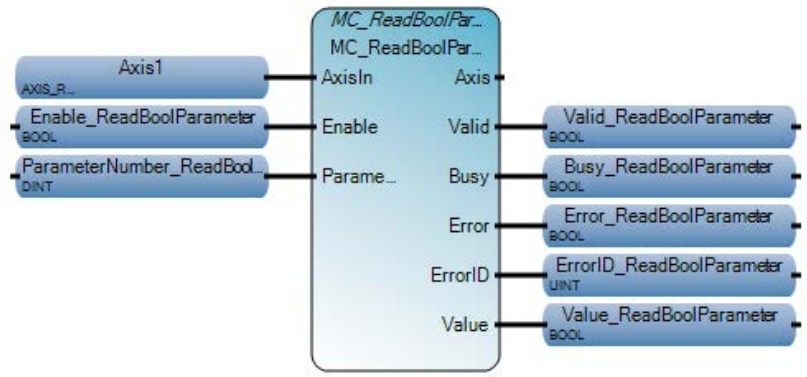


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

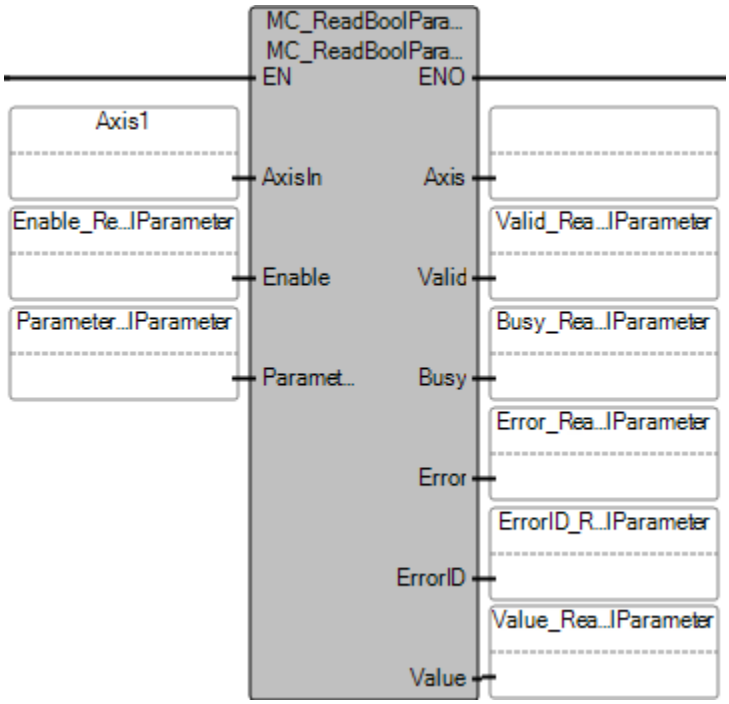
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Aktivieren des Befehlsblocks. TRUE – aktuelle MC_ReadBoolParameter-Berechnung ausführen. FALSE – Wertausgabe auf 0 zurücksetzen. Gilt nur für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF FB_AXIS_REF	Definieren Sie AxisIn mithilfe des Datentyps AXIS_REF auf seite 413. Bei FB_Axis (Feedbackachse) verwenden Sie den Datentyp FB_AXIS_REF auf seite 414 zum Definieren der AxisIn-Parameter.
Enable	Eingang	BOOL	TRUE – ruft bei Aktivierung den Wert des Parameters fortlaufend ab. FALSE – die Wertausgabe wird auf 0 zurückgesetzt.
ParameterNumber	Eingang	DINT	Parameterkennung. Definitionen für Parameternummern werden in den Parameternummern für den Servosteuerungs-Funktionsblock festgelegt.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.
Valid	Ausgang	BOOL	TRUE – der Wert des Parameters ist verfügbar. FALSE – der Parameterwert ist nicht verfügbar.

Parameter	Parametertyp	Datentyp	Beschreibung
Busy	Ausgang	BOOL	TRUE – der Funktionsblock funktioniert, und es werden neue Ausgabewerte erwartet. FALSE – die Funktion befindet sich im Leerlauf.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.
Value	Ausgang	BOOL	Wert des angegebenen Parameters im Datentyp, wie vom Lieferanten angegeben.

Beispiel für ein MC_ReadBoolParameter-Funktionsblockdiagramm



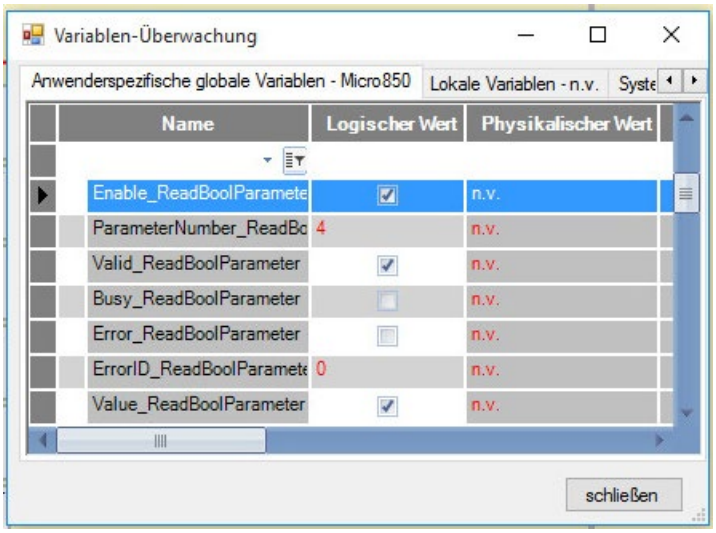
Beispiel für einen MC_ReadBoolParameter-Kontaktplan



Beispiel für einen strukturierten MC_ReadBoolParameter-Text

```
ParameterNumber_ReadBoolParameter := 4;
MC_ReadBoolParameter_1(Axis1,Enable_ReadBoolParameter,ParameterNumber_ReadBoolParameter);
Valid_ReadBoolParameter := MC_ReadBoolParameter_1.Valid;
Busy_ReadBoolParameter := MC_ReadBoolParameter_1.Busy;
Error_ReadBoolParameter := MC_ReadBoolParameter_1.Error;
ErrorID_ReadBoolParameter := MC_ReadBoolParameter_1.ErrorID;
Value_ReadBoolParameter := MC_ReadBoolParameter_1.Value;
```

Ergebnisse

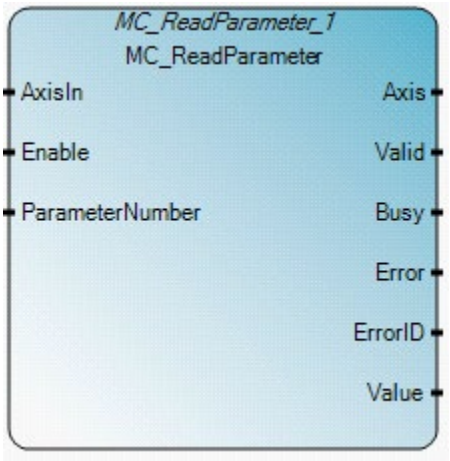


MC_ReadParameter (Servosteuerung liest Parameter)

Gibt den Wert eines anbieterspezifischen Parameters vom Typ „Real“ zurück.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

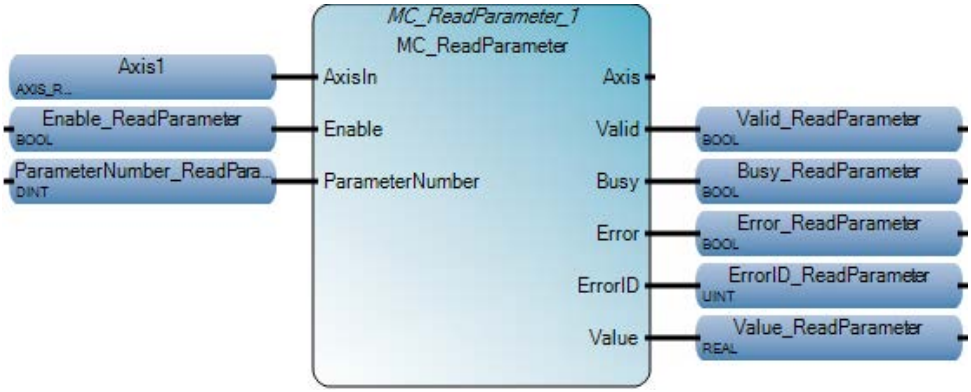
Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung



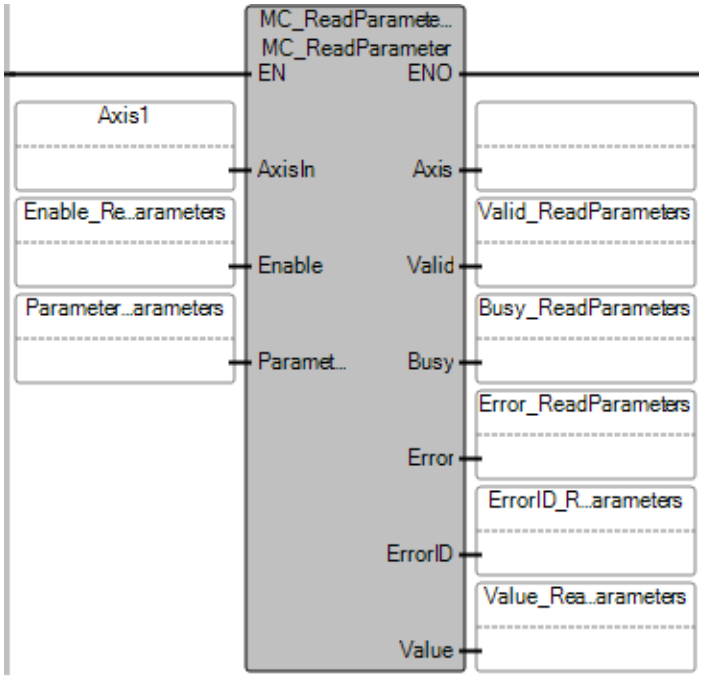
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_ReadParameter-Berechnung ausführen. FALSE – die Wertausgabe wird auf 0 zurückgesetzt. Gilt nur für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF FB_AXIS_REF	Definieren Sie AxisIn mithilfe des Datentyps AXIS_REF auf seite 413. Bei FB_Axis (Feedbackachse) verwenden Sie den Datentyp FB_AXIS_REF auf seite 414 zum Definieren der AxisIn-Parameter.
Enable	Eingang	BOOL	TRUE – ruft bei Aktivierung den Wert der Parameternummer fortlaufend ab. FALSE – die Wertausgabe wird auf 0 zurückgesetzt.
ParameterNumber	Eingang	DINT	Parameterkennung. Definitionen für Parameternummern werden in den Parameternummern für den Servosteuerungs-Funktionsblock festgelegt.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.
Valid	Ausgang	BOOL	TRUE – es sind gültige Ausgaben vorhanden. FALSE – es sind keine gültigen Ausgaben vorhanden.
Busy	Ausgang	BOOL	TRUE – der Funktionsblock funktioniert, und es werden neue Ausgabewerte erwartet. FALSE – der Funktionsblock befindet sich im Leerlauf.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.
Value	Ausgang	REAL	Wert des angegebenen Parameters im Datentyp, wie vom Lieferanten angegeben.

Diagrammbeispiel für Funktionsblock MC_ReadParameter



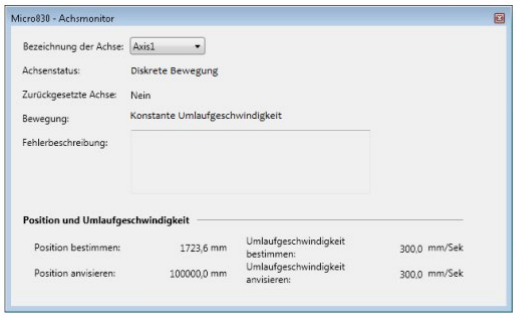
Beispiel für MC_ReadParameter-Kontaktplan

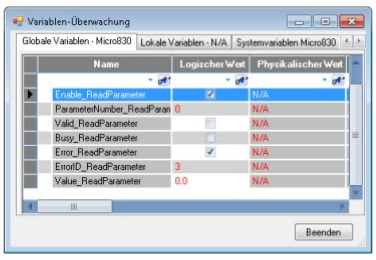


Beispiel für strukturierten MC_ReadParameter-Text

```
ParameterNumber_ReadParameter := 11;
MC_ReadParameter_1(Axis1,Enable_ReadParameter,ParameterNumber_ReadParameter);
Valid_ReadParameter := MC_ReadParameter_1.Valid;
Busy_ReadParameter := MC_ReadParameter_1.Busy;
Error_ReadParameter := MC_ReadParameter_1.Error;
ErrorID_ReadParameter := MC_ReadParameter_1.ErrorID;
Value_ReadParameter := MC_ReadParameter_1.Value;
```

Ergebnisse





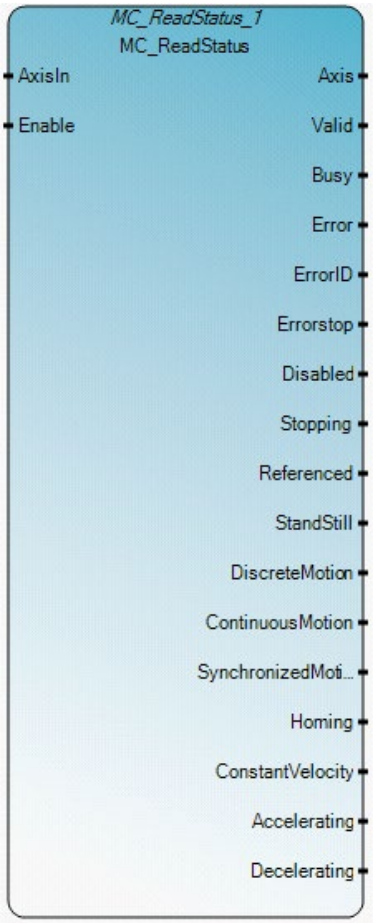
MC_ReadStatus (Servosteuerung liest Status)

Gibt den Achsstatus hinsichtlich der aktuell erfolgenden Bewegung zurück.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro830, Micro850 und

Micro870-Controller mit Unterstützung der Bewegungssteuerung

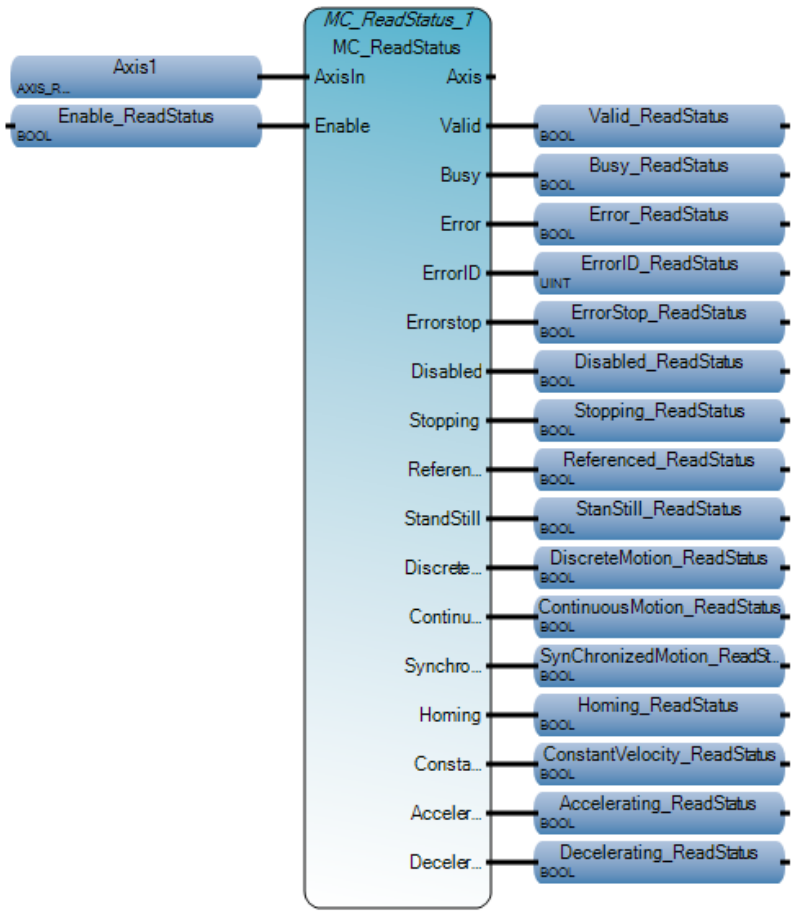


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

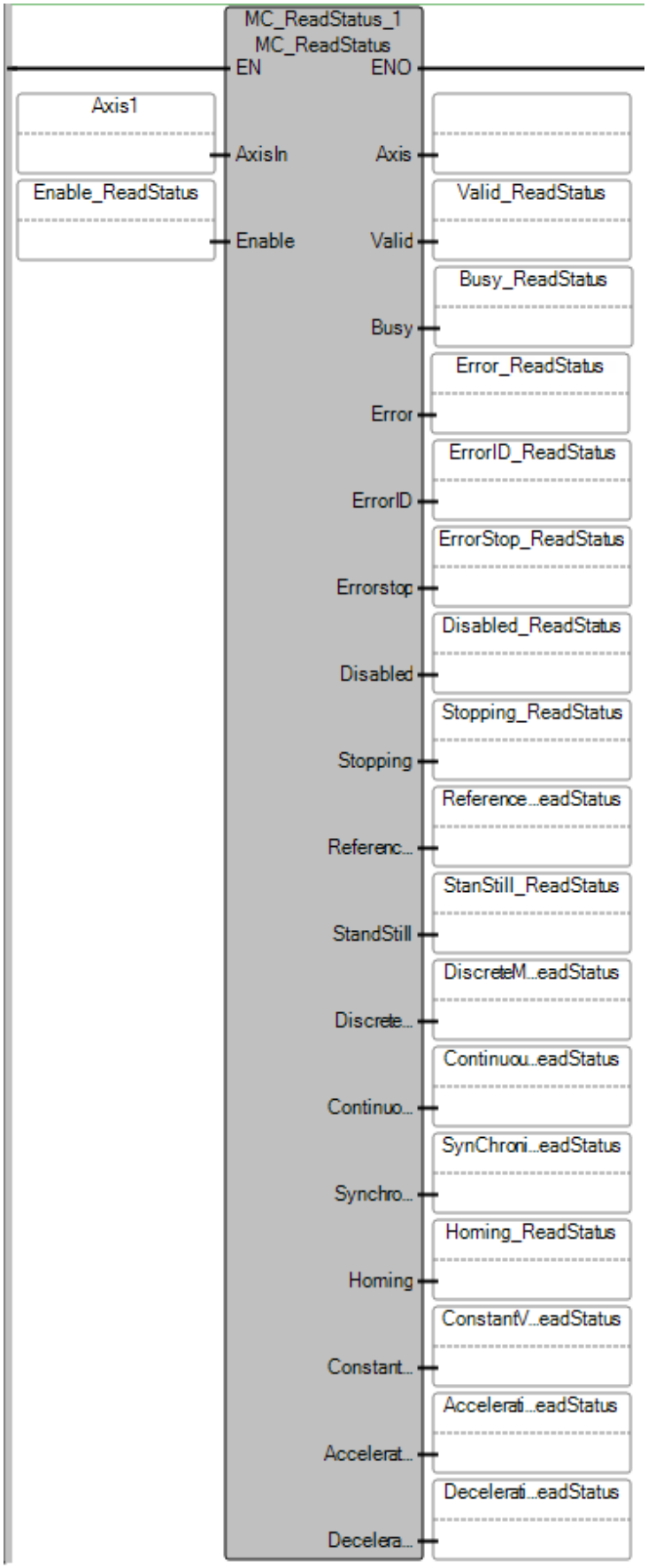
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_ReadStatus-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF FB_AXIS_REF	Definieren Sie AxisIn mithilfe der Parameter des Datentyps AXIS_REF auf seite 413. Bei FB_Axis (Feedbackachse) verwenden Sie den Datentyp FB_AXIS_REF auf seite 414 zum Definieren der AxisIn-Parameter.
Enable	Eingang	BOOL	TRUE – der Achsenstatus wird kontinuierlich abgerufen. FALSE – alle Statusausgaben werden auf 0 zurückgesetzt.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.
Valid	Ausgang	BOOL	TRUE – es sind gültige Ausgaben vorhanden. FALSE – Ausgaben nicht verfügbar.
Busy	Ausgang	BOOL	TRUE – der Funktionsblock funktioniert, und es werden neue Ausgabewerte erwartet. FALSE – der Funktionsblock befindet sich im Leerlauf.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.

Parameter	Parametertyp	Datentyp	Beschreibung
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks definiert.
ErrorStop	Ausgang	BOOL	TRUE – der Achsenstatus lautet „ErrorStop". Achsenstatus werden in den Werten und Namen für den Servosteuerungs-Achsenstatus definiert.
Disabled	Ausgang	BOOL	TRUE – der Achsenstatus lautet „Disabled".
Stopping	Ausgang	BOOL	TRUE – der Achsenstatus lautet „Stopping".
Referenced	Ausgang	BOOL	TRUE – der Achsenstatus wird zurückgesetzt, die absolute Referenzposition für die Achse ist bekannt.
StandStill	Ausgang	BOOL	TRUE – der Achsenstatus lautet „StandStill".
DiscreteMotion	Ausgang	BOOL	TRUE – der Achsenstatus lautet „DiscreteMotion".
ContinuousMotion	Ausgang	BOOL	TRUE – der Achsenstatus lautet „ContinuousMotion".
SynchronizedMotion	Ausgang	BOOL	Synchronisierte Bewegung wird von Micro800-Controllern nicht unterstützt. TRUE – niemals „true". FALSE – immer „false".
Homing	Ausgang	BOOL	TRUE – der Achsenstatus lautet „Homing".
ConstantVelocity	Ausgang	BOOL	TRUE – der Motor läuft mit konstanter Geschwindigkeit.
Accelerating	Ausgang	BOOL	TRUE – Achse wird beschleunigt, Energiezufuhr zum Motor wird erhöht.
Decelerating	Ausgang	BOOL	TRUE – Achse wird verlangsamt, Energiezufuhr zum Motor wird reduziert.

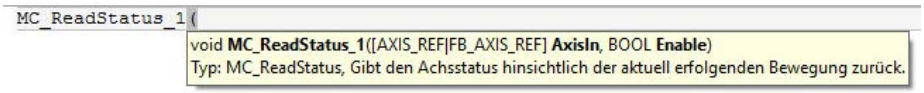
Diagrammbeispiel für Funktionsblock MC_ReadStatus



Beispiel für MC_ReadStatus-Kontaktplan

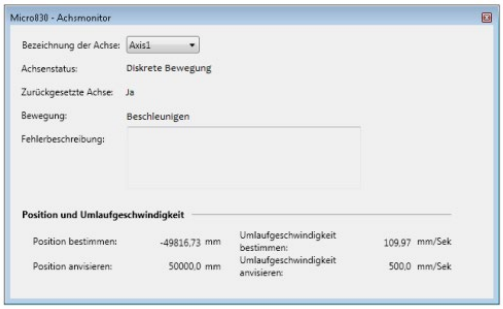


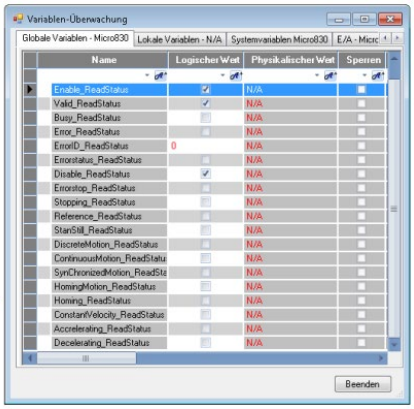
Beispiel für strukturierten MC_ReadStatus-Text




```
MC_ReadStatus_1(Axis1,Enable_ReadStatus);
Valid_ReadStatus := MC_ReadStatus_1.Valid;
Busy_ReadStatus := MC_ReadStatus_1.Busy;
Error_ReadStatus := MC_ReadStatus_1.Error;
ErrorID_ReadStatus := MC_ReadStatus_1.ErrorID;
ErrorStop_ReadStatus := MC_ReadStatus_1.Errorstop;
Disabled_ReadStatus := MC_ReadStatus_1.Disabled;
Stopping_ReadStatus := MC_ReadStatus_1.Stopping;
Referenced_ReadStatus := MC_ReadStatus_1.Referenced;
StanStill_ReadStatus := MC_ReadStatus_1.StandStill;
DiscreteMotion_ReadStatus := MC_ReadStatus_1.DiscreteMotion;
ContinuousMotion_ReadStatus := MC_ReadStatus_1.ContinuousMotion;
SynChronizedMotion_ReadStatus := MC_ReadStatus_1.SynchronizedMotion;
Homing_ReadStatus := MC_ReadStatus_1.Homing;
ConstantVelocity_ReadStatus := MC_ReadStatus_1.ConstantVelocity;
Accelerating_ReadStatus := MC_ReadStatus_1.Accelerating;
Decelerating_ReadStatus := MC_ReadStatus_1.Decelerating;
```

Ergebnisse





MC_Reset (Zurücksetzen der Servosteuerung)

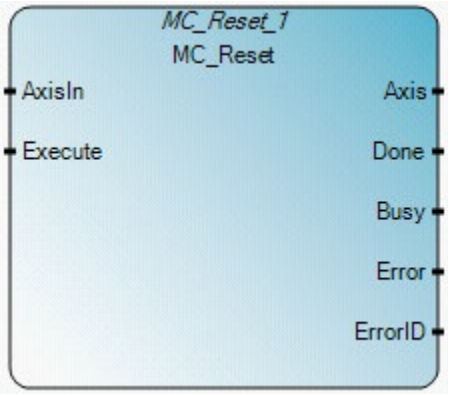
Setzt den Achszustand von „ErrorStop“ auf „StandStill“. Dies geschieht durch das Zurücksetzen aller internen achsbezogenen Fehler. Die Ausgaben der Funktionsblock-Instanzen werden nicht geändert.

Details zum Betrieb:

- Wenn der Achsenalarmstatus nach dem Ausführen von MC_Reset unverändert ist, führen Sie MC_Power und dann MC_Reset aus.
- MC_Reset setzt nur den Achsenstatus von „ErrorStop“ nach „StandStill“ zurück. Die Ausführung von MC_Reset in anderen Achsenzuständen, einschließlich „Disabled“, führt zu einem Fehler und hat keine Auswirkung auf die laufende Bewegung oder den Zustand der Achse.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

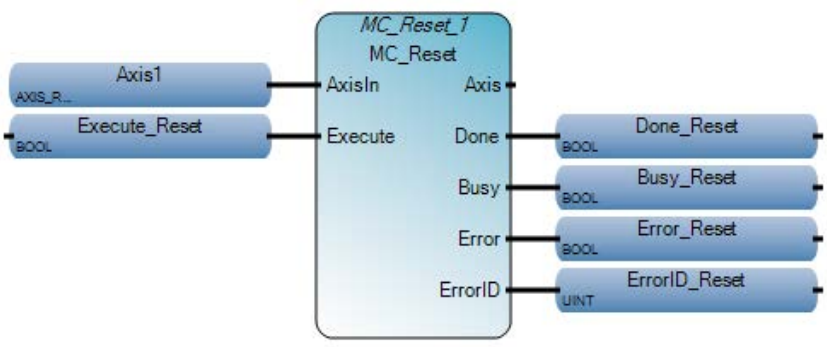
Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung



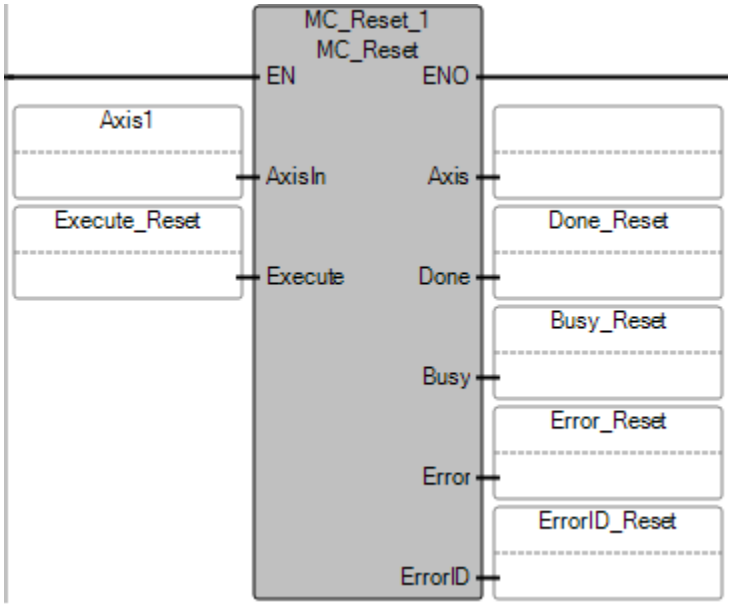
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_Rest-Berechnung ausführen. FALSE, es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF FB_AXIS_REF	Definieren Sie AxisIn mithilfe des Datentyps AXIS_REF. Bei FB_Axis (Feedbackachse) verwenden Sie den Datentyp FB_AXIS_REF auf seite 414 zum Definieren der AxisIn-Parameter.
Execute	Eingang	BOOL	TRUE – die Achse wird zur Anstiegsflanke zurückgesetzt.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.
Done	Ausgang	BOOL	TRUE – Achsenstatus lautet „StandStill" oder „Disabled".
Busy	Ausgang	BOOL	TRUE – der Funktionsblock wird nicht beendet.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.

Diagrammbeispiel für Funktionsblock MC_Reset



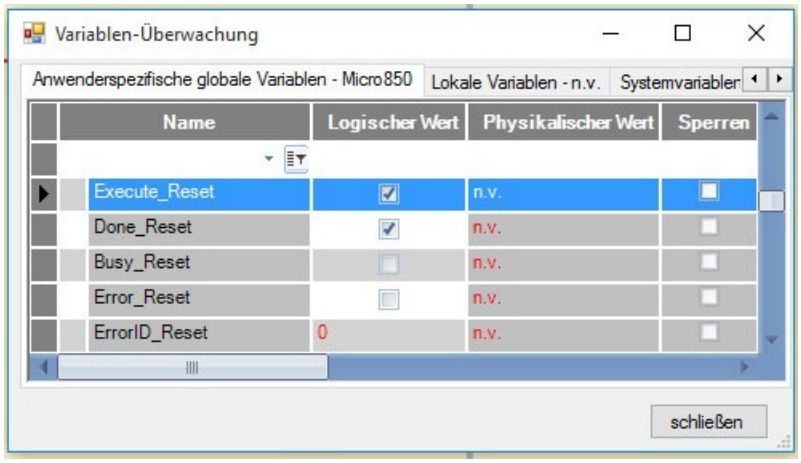
Beispiel für den Kontaktplan von MC_Reset



Beispiel für den Strukturierten Text von MC_Reset

```
MC_Reset_1(Axis1,Execute_Reset);
Done_Reset := MC_Reset_1.Done;
Busy_Reset := MC_Reset_1.Busy;
Error_Reset := MC_Reset_1.Error;
ErrorID_Reset := MC_Reset_1.ErrorID;
```

Ergebnisse



MC_SetPosition (Servosteuerung legt Position fest)

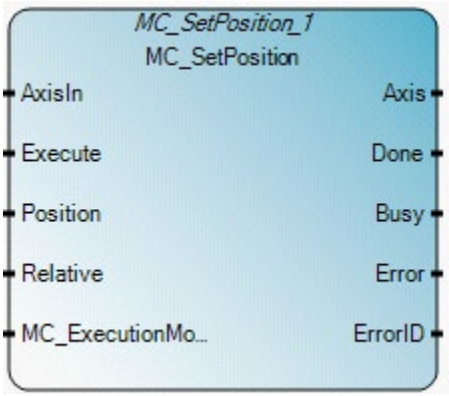
Verschiebt das Koordinatensystem einer Achse durch Ändern der tatsächlichen Achsposition mit demselben Wert, ohne dabei irgendeine Bewegung zu verursachen.

Details zum Betrieb:

- MC_SetPosition kann nur im Achsenstatus „StandStill“ oder „Kontinuierliche Bewegung“ (MC_ExecutionMode = 0) erfolgreich beendet werden oder wenn die laufende Bewegung abgeschlossen und im Zustand „StandStill“ (MC_ExecutionMode = 1) beendet wird.
- MC_SetPosition funktioniert genau wie MC_Home, wenn HomingMode = MC_HOME_DIRECT (0x04), ausgenommen, der Funktionsblock „MC_Home“ legt den Zustand „Zurückgesetzte Achse“ fest.
- Wenn MC_ExecutionMode = 0 (mcImmediately), wird beim Ausführen von Funktionsblock „MC_SetPosition“ ein Fehler gemeldet, wenn die Achse eine laufende nicht kontinuierliche Bewegung hat.
- Wenn MC_ExecutionMode = 1 (mcQueued), findet die tatsächliche Positionsfestlegung nur statt, wenn alle vorherigen laufenden Bewegungen gestoppt werden. Mindestens eine der Ausgaben „Done“ (Fertig), „Aborted“ (Abgebrochen) oder „Error“ (Fehler) jedes vorherigen Funktionsblocks muss also „True“ sein.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_SetPosition-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF FB_AXIS_REF	Definieren Sie AxisIn mithilfe des Datentyps AXIS_REF. Bei FB_Axis (Feedbackachse) verwenden Sie den Datentyp FB_AXIS_REF

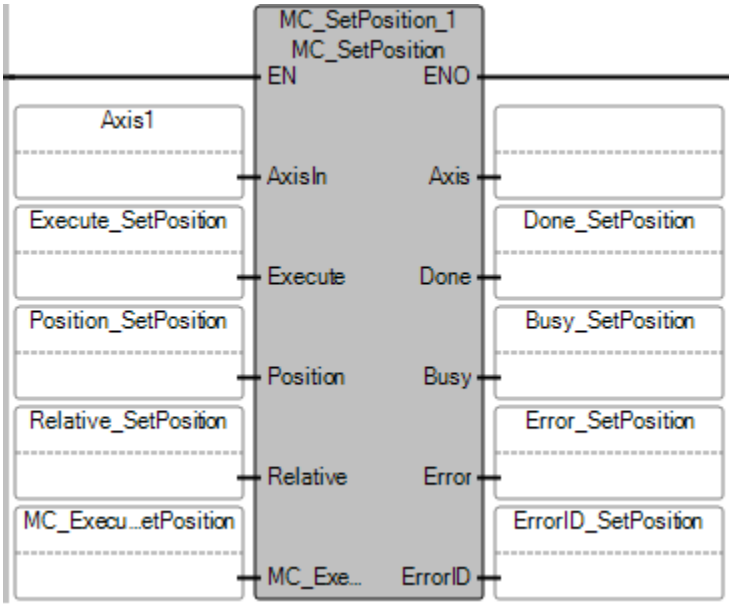
Parameter	Parametertyp	Datentyp	Beschreibung
			auf seite 414 zum Definieren der AxisIn-Parameter.
Execute	Eingang	BOOL	TRUE – das Setzen der Achsposition wird gestartet.
Position	Eingang	REAL	Die für die Achse einzustellende absolute Position oder relative Entfernung.
Relative	Eingang	BOOL	TRUE – die relative Entfernung für die Achse wird festgelegt. FALSE – die absolute Position für die Achse wird festgelegt.

MC_ExecutionMode	Eingang	SINT	Die möglichen Werte lauten:
			• 0 (*mcImmediately*) – Die Funktionalität ist sofort gültig.
			• 1 (*mc0ueued*) - Die neue Funktionalität wird gültig, wenn:
			  • alle vorherigen Bewegungsbefehle auf einen der folgenden Ausgabeparameter eingestellt sind: „Done" (Fertig), „Aborted" (Abgebrochen) oder „Error" (Fehler).
			  • die Achse keinen Bewegungszustand besitzt.
			Bei (MC_ExecutionMode = 0) wird dieser Funktionsblock im Achsenstatus „Disabled" (Deaktiviert) oder „StandStill" (Stillstand) erfolgreich beendet. Beim Ausführen dieses Funktionsblocks wird ein Fehler gemeldet, wenn die Achse eine laufende, nicht kontinuierliche Bewegung in diesem Modus hat.
			Bei (MC_ExecutionMode = 1) wird dieser Funktionsblock im Achsenstatus „Disabled" (Deaktiviert) oder „StandStill" (Stillstand) erfolgreich beendet, bzw. die laufende Bewegung kann abgeschlossen werden und endet im Zustand „Standstill" (Stillstand).
			Andere Eingabewerte sind derzeit reserviert und werden als ungültige Parameter betrachtet.

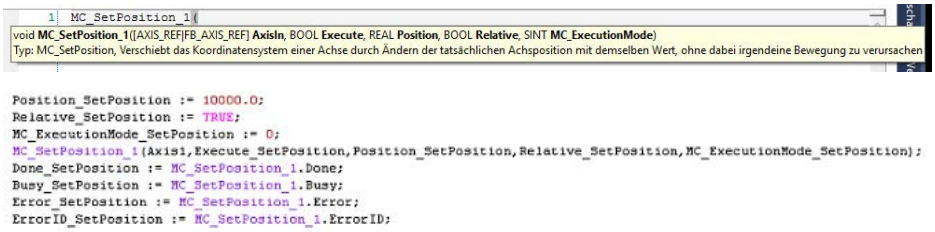
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.
Done	Ausgang	BOOL	TRUE – die Position hat einen neuen Wert.
Busy	Ausgang	BOOL	TRUE – der Funktionsblock wird nicht beendet.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.

Beispiel für ein MC_SetPosition-Funktionsblockdiagramm

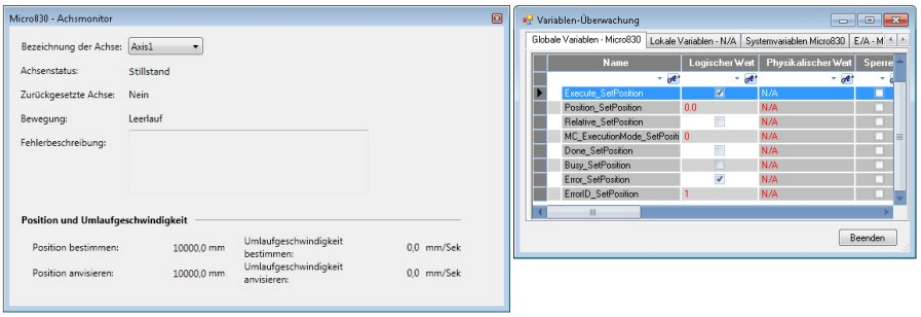
Beispiel für einen MC_SetPosition-Kontaktplan



Beispiel für einen strukturierten MC_SetPosition-Text



Ergebnisse



MC_Stop (Servosteuerungsstopp)

Befiehlt einen kontrollierten Bewegungsstopp und ändert den Zustand der Achse in „Stoppen“. Alle zurzeit ausgeführten Funktionsblöcke werden abgebrochen. Sämtliche Bewegungsbefehle eines Funktionsblocks werden ignoriert, bis der Achsenstatus zu „StandStill“ wechselt.

Details zum Betrieb:

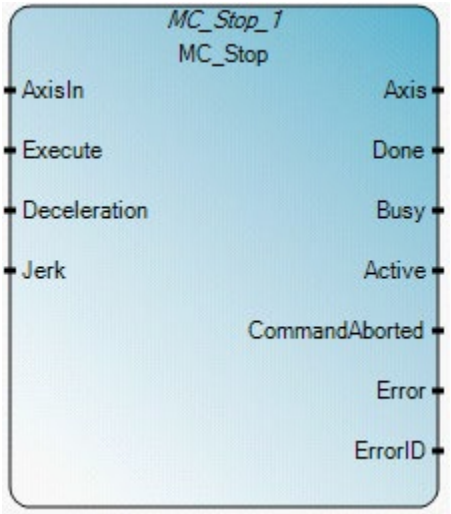
- Solange die Eingabe „Ausführen“ hoch ist, bleibt die Achse im Zustand „Stoppen“. Wenn sich die Achse im Zustand „Stopping“ (Stoppen)

befindet, kann kein anderer Bewegungsfunktionsblock eine Bewegung auf dieser Achse ausführen.

- Wenn die Verzögerung gleich null ist, werden die Parameter für MC_Stop durch die Achsenkonfigurationseinstellung für Nothalt bestimmt, einschließlich Nothalt-Typ, Nothalt-Verzögerung und Nothalt-Jerk.
- Wenn während der Stoppsequenz keine Fehler erkannt werden, wechselt der Achsenstatus nach „Done“ (Fertig) zu „StandStill (Stillstand), und die Eingabe „Ausführen“ wechselt zu „False“.
- Verwenden Sie MC_Stop für Nothalt-Funktionen oder Ausnahmesituationen. Verwenden Sie MC_Halt für normale Bewegungsstopps.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung

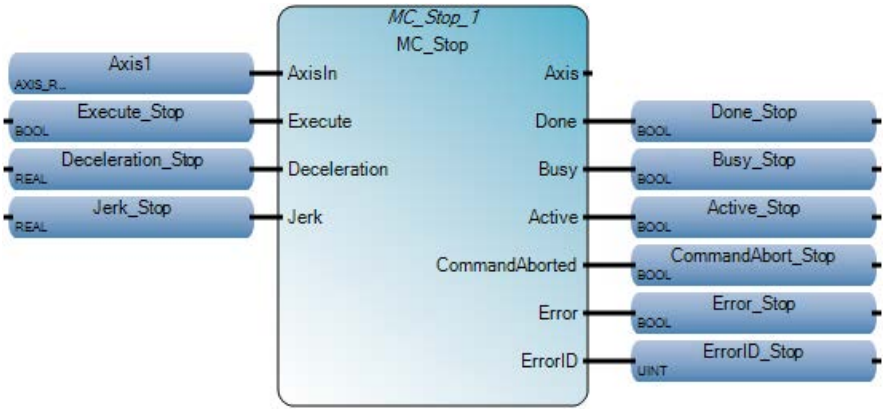


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

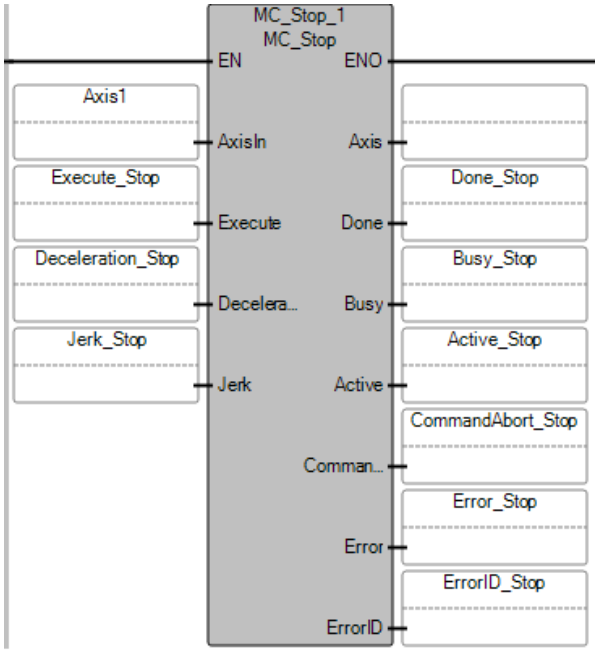
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_Stop-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF	Definieren Sie die Parameter für AxisIn mithilfe des Datentyps AXIS_REF.
Execute	Eingang	BOOL	TRUE – die Aktion beginnt an der Anstiegsflanke. FALSE – wird nicht ausgeführt.
Deceleration	Eingang	REAL	Wert der Verzögerung [u/s^2].
Jerk	Eingang	REAL	Wert des Stoßes [u/s^3].
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Parameter	Parametertyp	Datentyp	Beschreibung
Axis	Ausgang	AXIS_REF auf seite 413	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.
Done	Ausgang	BOOL	TRUE – Geschwindigkeit null wurde erreicht, ohne dass während der Stoppsequenz ein Fehler aufgetreten ist.
Busy	Ausgang	BOOL	TRUE – der Funktionsblock wird nicht beendet.
Active	Ausgang	BOOL	TRUE – gibt an, dass der Funktionsblock die Steuerung über die Achse besitzt.
CommandAborted	Ausgang	BOOL	TRUE – Befehl wurde durch den Funktionsblock MC_Power(OFF) oder durch ErrorStop überschrieben.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.

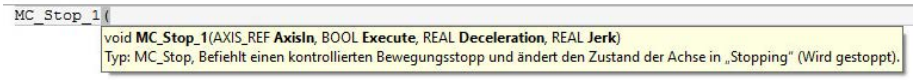
Diagrammbeispiel für Funktionsblock MC_Stop



Beispiel für MC_Stop-Kontaktplan

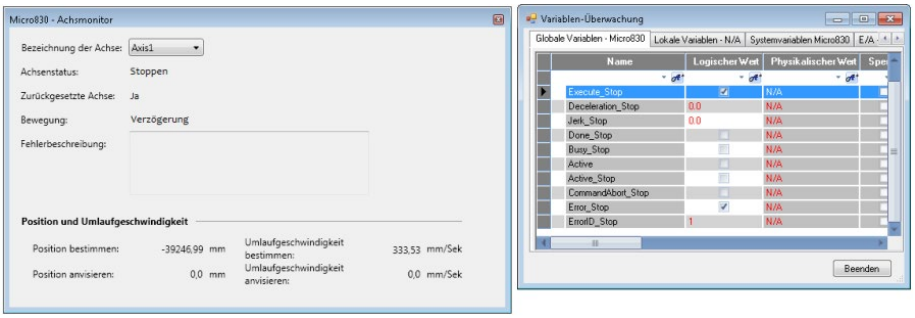


Beispiel für strukturierten MC_Stop-Text




```
Deceleration_Stop := 10.0;
Jerk_Stop := 10.0;
MC_Stop_1(Axis1,Execute_Stop,Deceleration_Stop,Jerk_Stop);
Done_Stop := MC_Stop_1.Done;
Busy_Stop := MC_Stop_1.Busy;
Active_Stop := MC_Stop_1.Active;
CommandAbort_Stop := MC_Stop_1.CommandAborted;
Error_Stop := MC_Stop_1.Error;
ErrorID_Stop := MC_Stop_1.ErrorID;
```

Ergebnisse



MC_TouchProbe (Servosteuerungs-Berührungssensor)

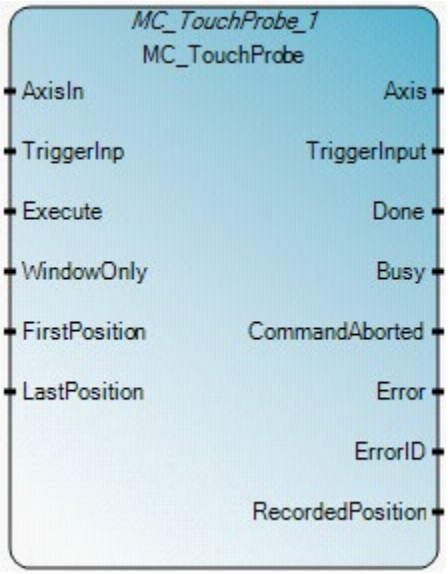
Zeichnet eine Achsenposition bei einem Auslöseereignis auf.

Details zum Betrieb:

- Wenn die Fensterrichtung (erste Position --> letzte Position) entgegengesetzt der Bewegungsrichtung ist, wird das Fenster „Fühlerberührung" nicht aktiviert.
- Bei ungültiger Fenstereinstellung (FirstPosition oder LastPosition) meldet der Funktionsblock „MC_TouchProbe" einen Fehler.
- Wenn eine zweite Instanz des Funktionsblocks „MC_TouchProbe" auf derselben Achse ausgegeben wird und die erste Funktionsblockinstanz im Zustand „Busy" (Beschäftigt) ist, meldet die zweite Funktionsblockinstanz einen Fehler.
- Es darf nur eine Instanz des Funktionsblocks „MC_TouchProbe" auf derselben Achse ausgegeben werden.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

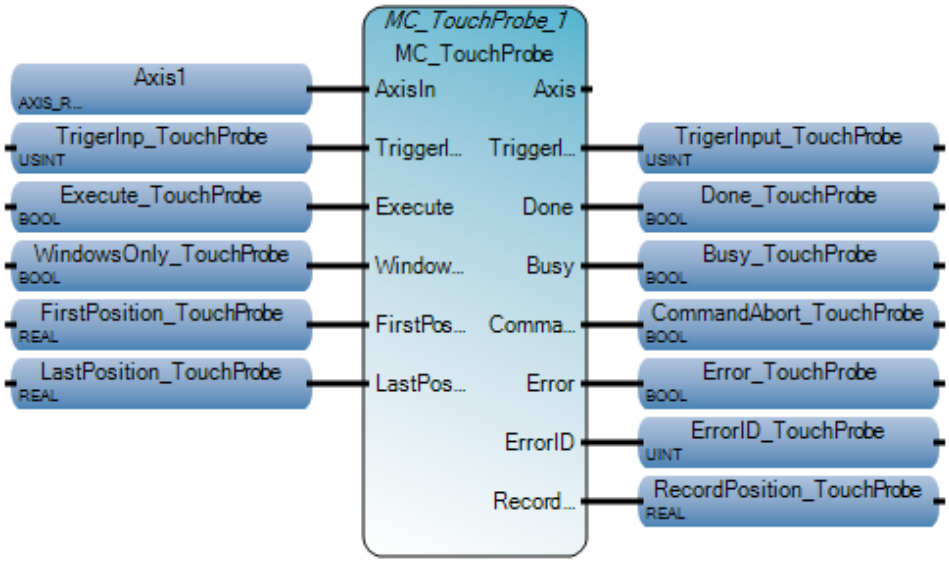
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_TouchProbe-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF FB_AXIS_REF	Definieren Sie die Parameter für AxisIn mithilfe des Datentyps AXIS_REF. Bei einer Feedbackachse verwenden Sie den Datentyp FB_AXIS_REF auf seite 414 zum Definieren der AxisIn-Parameter.
TriggerInp	Eingang	USINT	Wird zurzeit nicht unterstützt. In der Achsenkonfiguration einen Auslösepunkt konfigurieren.
Execute	Eingang	BOOL	TRUE – Aufzeichnung des Berührungssensors beginnt an der Anstiegsflanke. FALSE – wird nicht ausgeführt.
WindowOnly	Eingang	BOOL	TRUE – nur das Fenster zum Akzeptieren von Auslöseereignissen wird verwendet. Die Bewegungsauflösung ist auf das durch den Benutzer konfigurierte Bewegungsmotor-Intervall begrenzt. Bei der „WindowOnly"-Funktionalität des Messfühlers besteht eine maximale Verzögerung der Reaktionszeit, die dem Bewegungsmotor-Intervall für sowohl die „FirstPosition"- als auch die „LastPosition"-Aktivierung entspricht. Die maximal mögliche Verzögerung beim Auslösen der Position (sowohl „FirstPosition" (erste Position) als auch „LastPosition" (letzte Position) kann über die Formel (Bewegungsmotor-Intervall * Bewegungsgeschwindigkeit) berechnet werden.
FirstPosition	Eingang	REAL	Startposition des Fensters, über das Auslöseereignisse angenommen werden (in technischen Einheiten [u]). Wert im Fenster enthalten.
LastPosition	Eingang	REAL	Stoppposition des Fensters, über das Auslöseereignisse angenommen werden (in technischen Einheiten [u]). Wert im Fenster enthalten.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF FB_AXIS_REF	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.

Parameter	Parametertyp	Datentyp	Beschreibung
TriggerInput	Ausgang	USINT	Wird zurzeit nicht unterstützt.
Done	Ausgang	BOOL	TRUE – das Auslöseereignis wurde aufgezeichnet.
Busy	Ausgang	BOOL	TRUE – der Funktionsblock wird nicht beendet.
CommandAborted	Ausgang	BOOL	TRUE – Der Befehl wurde durch den Funktionsblock „MC_Power(OFF)" oder durch den Funktionsblock „Anhalten aufgrund eines Fehlers" überschrieben.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.
RecordedPosition	Ausgang	REAL	Position, an der das Auslöseereignis aufgetreten ist (in technischen Einheiten [u]) Bei der Bewegung handelt es sich um eine Leerlaufbewegung. Die Achsenposition zu dem Zeitpunkt, an dem das Auslöseereignis auftritt. Wenn die Achsenbewegung eine Leerlaufbewegung ist, handelt es sich um die Positionsvorgabe (nicht die tatsächliche Position) zum Zeitpunkt des Auftretens des Auslöseereignisses, wenn es keine Bewegungsverzögerung zwischen Antrieb und Motor gibt.

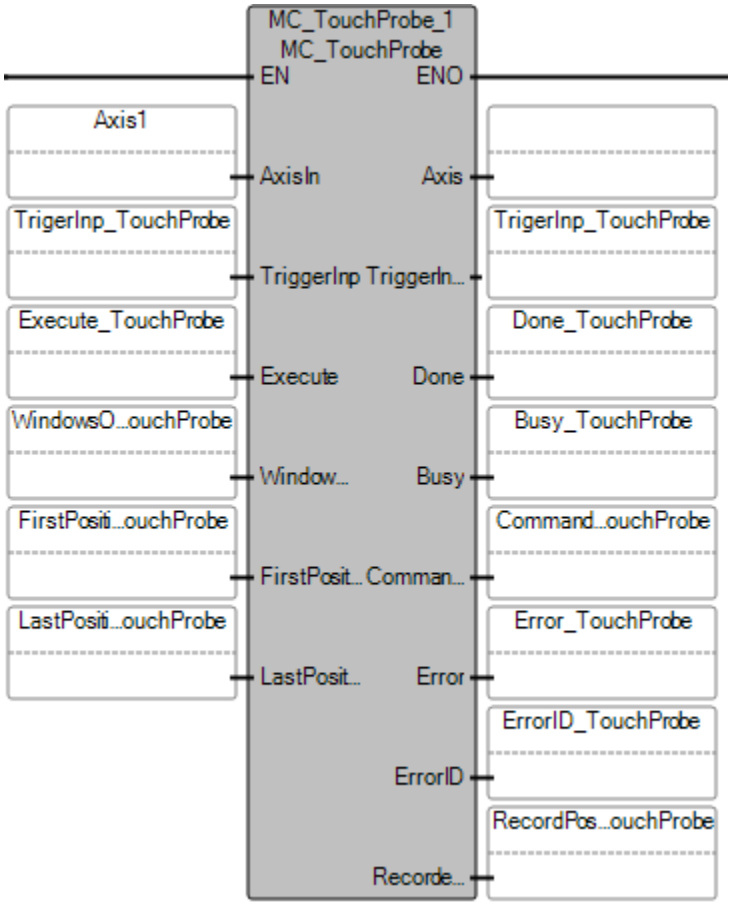
Bewegungsfestgelegte Eingabe/Ausgabe

Bewegungssignale	PT00	PT01	PT02
PTO-Impuls	Output_0	Output_1	Output2
PTO-Richtung	Output_3	Output_4	Output_5
Unterer (negativer) Endschalter	Input_0	Input_4	Input_8
Oberer (positiver) Endschalter	Input_1	Input_5	Input_9
Schalter „Home Absolute"	Input_2	Input_6	Input_10
Schalter Fühlerberührungseingabe	Input_3	Input_7	Input_11

Diagrammbeispiel für Funktionsblock MC_TouchProbe



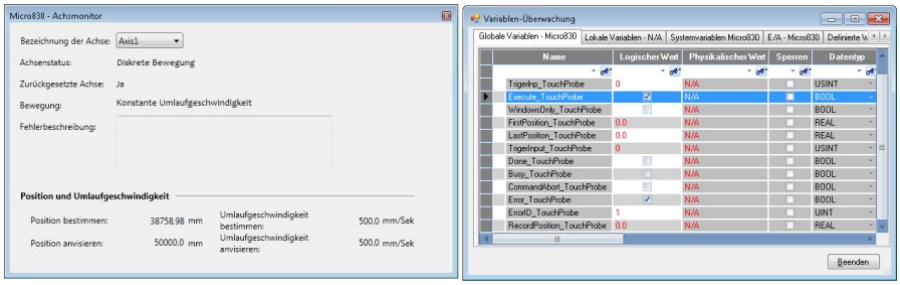
Beispiel für MC_TouchProbe-Kontaktplan



Beispiel für strukturierten MC_TouchProbe-Text

```
FirstPosition_TouchProbe := 10000.0;
LastPosition_TouchProbe := 50000.0;
MC_TouchProbe_1(Axis1,TrigerInp_TouchProbe,Execute_TouchProbe,WindowsOnly_TouchProbe,
FirstPosition_TouchProbe,LastPosition_TouchProbe);
Done_TouchProbe := MC_TouchProbe_1.Done;
Busy_TouchProbe := MC_TouchProbe_1.Busy;
Error_TouchProbe := MC_TouchProbe_1.Error;
ErrorID_TouchProbe := MC_TouchProbe_1.ErrorID;
RecordPosition_TouchProbe := MC_TouchProbe_1.RecordedPosition;
```

Ergebnisse



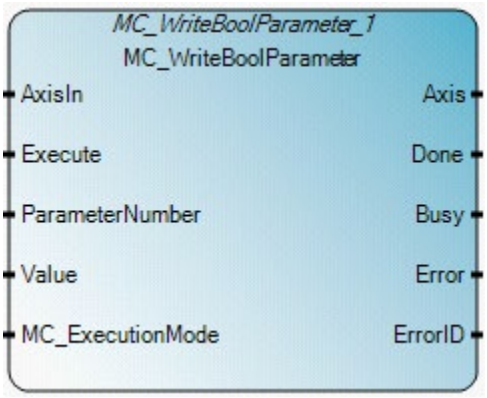
MC_WriteBoolParameter (Servosteuerung schreibt BOOL-Parameter)

Ändert den Wert eines anbieterspezifischen Parameters vom Typ BOOL.

Die vom Funktionsblock „MC_WriteBoolParameter“ gesetzten Parameter werden nur vorübergehend auf die Anwendung angewendet. Sie werden durch permanente Einstellungen überschrieben, die vom Benutzer in der Connected Components Workbench-Bewegungskonfiguration konfiguriert werden, wenn der Controller vom Modus PRG in den Modus RUN wechselt oder aus- und wieder eingeschaltet wird.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

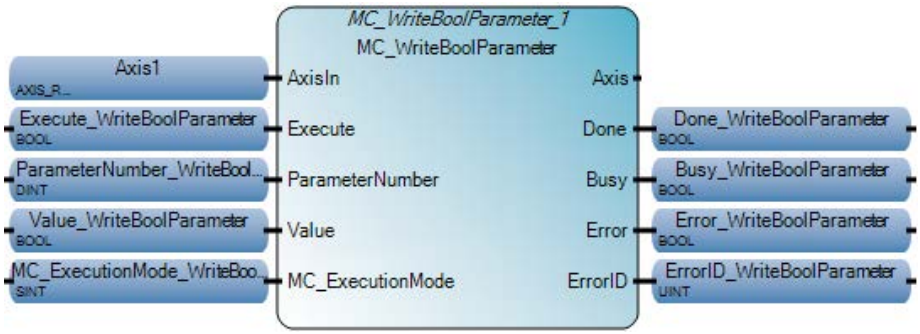
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_WriteBoolParameter-Berechnung ausführen. FALSE – die Wertausgabe wird auf 0 zurückgesetzt. Gilt nur für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF FB_AXIS_REF	Definieren Sie die Parameter für AxisIn mithilfe des Datentyps AXIS_REF. Bei einer Feedbackachse verwenden Sie den Datentyp FB_AXIS_REF zum Definieren der AxisIn-Parameter.
Execute	Eingang	BOOL	TRUE – der Parameterwert wird an der Anstiegsflanke geschrieben.
ParameterNumber	Eingang	DINT	Parameterkennung. Die Parameternummer wird in den Parameterinformationen für den Achssteuerungs-Funktionsblock definiert auf seite 408.
Value	Eingang	BOOL	TRUE – der angegebene Parameter hat einen neuen Wert.

Parameter	Parametertyp	Datentyp	Beschreibung

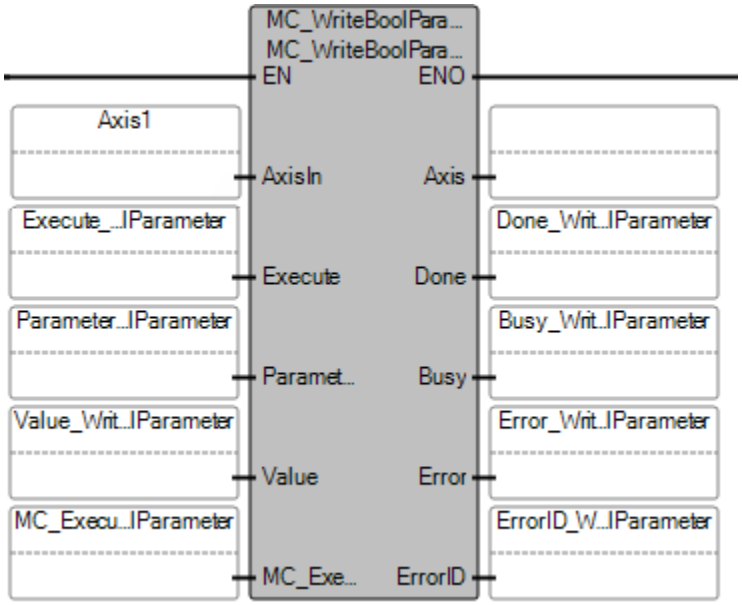
MC_ExecutionMode	Eingang	SINT	Die möglichen Werte lauten:
			• 0 (*mcImmediately*) – Die Funktionalität ist sofort gültig.
			• 1 (*mcQueued*) - Die neue Funktionalität wird gültig, wenn:
			  • alle vorherigen Bewegungsbefehle, einen der folgenden Ausgabeparameter gesetzt haben: „Done" (Fertig), „Aborted" (Abgebrochen) oder „Error" (Fehler)
			  • die Achse keinen Bewegungszustand besitzt.
			Bei (MC_ExecutionMode = 0) für alle Parameter außer dem Arbeitszyklus (1005) kann dieser FB nur abgeschlossen werden, wenn der Achsenstatus „Disabled" (Deaktiviert) oder „StandStill" (Stillstand) lautet.
			Bei (MC_ExecutionMode = 0) für den Parameter des Arbeitszyklus (1005) kann dieser FB abgeschlossen werden, sofern der Achsenstatus nicht „Homing" (Zurücksetzen) oder „ErrorStop" (Fehler-Stopp) lautet.
			Bei (MC_ExecutionMode = 1) kann dieser Funktionsblock nur im Achsenstatus „Disabled" (Deaktiviert) oder „StandStill" (Stillstand) erfolgreich beendet werden, bzw. die laufende Bewegung kann abgeschlossen werden und endet im Zustand „Standstill" (Stillstand).
			Andere Eingabewerte sind derzeit reserviert und werden als ungültige Parameter betrachtet.

ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.
Done	Ausgang	BOOL	TRUE – der Parameter wurde erfolgreich geschrieben.
Busy	Ausgang	BOOL	TRUE – der Funktionsblock wird nicht beendet.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.

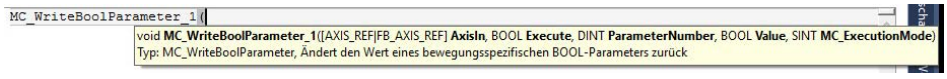
Diagrammbeispiel für Funktionsblock MC_WriteBoolParameter



Beispiel für Kontaktplan MC_WriteBoolParameter

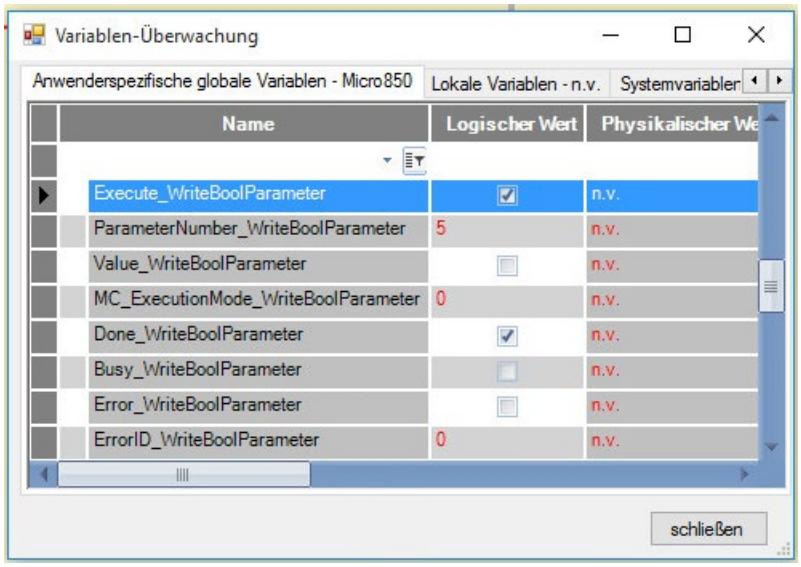


Beispiel für strukturierten MC-WriteBoolParameter-Text




```
ParameterNumber_WriteBoolParameter := 5;
MC_WriteBoolParameter_1(Axis1,Execute_WriteBoolParameter,ParameterNumber_WriteBoolParameter,
Value_WriteBoolParameter,MC_ExecutionMode_WriteBoolParameter);
Done_WriteBoolParameter := MC_WriteBoolParameter_1.Done;
Busy_WriteBoolParameter := MC_WriteBoolParameter_1.Busy;
Error_WriteBoolParameter := MC_WriteBoolParameter_1.Error;
ErrorID_WriteBoolParameter := MC_WriteBoolParameter_1.ErrorID;
```

Ergebnisse



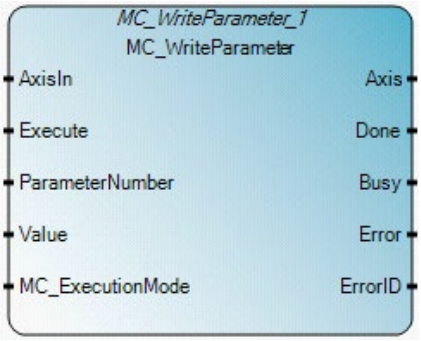
MC_WriteParameter (Servosteuerung schreibt Parameter)

Ändert den Wert eines anbieterspezifischen Parameters vom Typ REAL.

Die vom Funktionsblock „MC_WriteParameter“ gesetzten Parameter werden nur vorübergehend auf die Anwendung angewendet. Sie werden durch permanente Einstellungen überschrieben, die vom Benutzer in der Connected Components Workbench-Bewegungskonfiguration konfiguriert werden, wenn der Controller vom Modus PRG in den Modus RUN wechselt oder aus- und wieder eingeschaltet wird.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro830, Micro850 und Micro870-Controller mit Unterstützung der Bewegungssteuerung



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

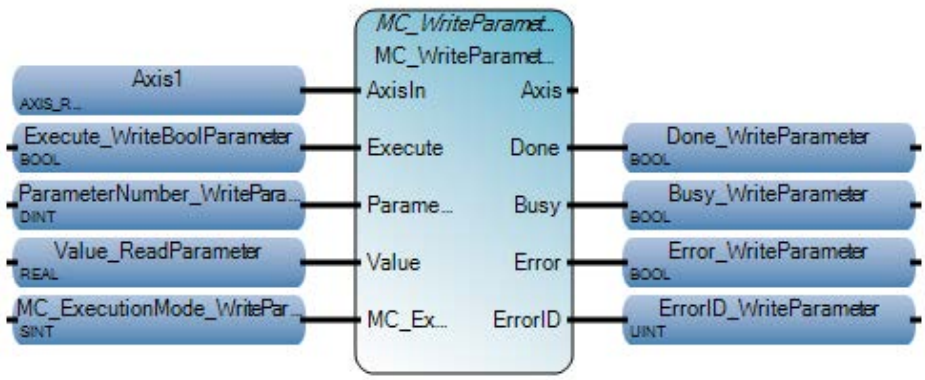
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – aktuelle MC_WriteParameter-Berechnung ausführen. FALSE – es findet keine Berechnung statt. Gilt nur für Kontaktplan-Programme.
AxisIn	Eingang	AXIS_REF FB_AXIS_REF	Definieren Sie die Parameter für AxisIn mithilfe des Datentyps AXIS_REF auf seite 413. Bei einer Feedbackachse verwenden Sie den Datentyp FB_AXIS_REF auf seite 414 zum Definieren der AxisIn-Parameter.
Execute	Eingang	BOOL	TRUE – der Parameterwert wird an der Anstiegsflanke geschrieben.
ParameterNumber	Eingang	DINT	Parameterkennung. Die Parameternummer wird in den Parameterinformationen für den Achssteuerungs-Funktionsblock definiert auf seite 408.
Value	Eingang	REAL	Neuer Wert des angegebenen Parameters.

Parameter	Parametertyp	Datentyp	Beschreibung

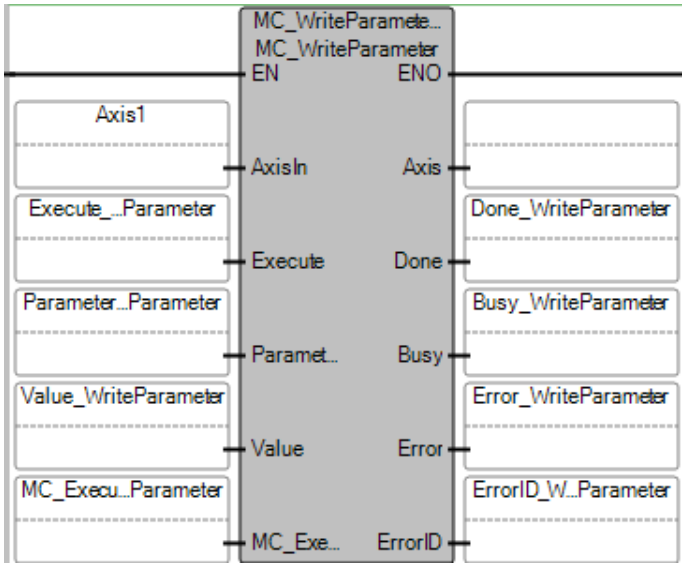
MC_ExecutionMode	Eingang	SINT	Die möglichen Werte lauten:
			• 0 (*mcImmediately*) – Die Funktionalität ist sofort gültig.
			• 1 (*mcQueued*) - Die neue Funktionalität wird gültig, wenn:
			  • alle vorherigen Bewegungsbefehle, einen der folgenden Ausgabeparameter gesetzt haben: „Done" (Fertig), „Aborted" (Abgebrochen) oder „Error" (Fehler)
			  • die Achse keinen Bewegungszustand besitzt
			  • angegeben wird, dass der Ausgabeparameter „Busy" (Beschäftigt) auf FALSE gesetzt ist.
			Bei (MC_ExecutionMode = 0) für alle Parameter außer dem Arbeitszyklus (1005) kann dieser FB nur abgeschlossen werden, wenn der Achsenstatus „Disabled" (Deaktiviert) oder „StandStill" (Stillstand) lautet.
			Bei (MC_ExecutionMode = 0) für den Parameter des Arbeitszyklus (1005) kann dieser FB abgeschlossen werden, sofern der Achsenstatus nicht „Homing" (Zurücksetzen) oder „ErrorStop" (Fehler-Stopp) lautet.
			
			Bei (MC_ExecutionMode = 1) kann dieser Funktionsblock nur im Achsenzustand „Deaktiviert" oder „Stillstand" erfolgreich beendet werden, bzw. die laufende Bewegung kann abgeschlossen werden und endet im Zustand „Stillstand".
			Andere Eingabewerte sind derzeit reserviert und werden als ungültige Parameter betrachtet.

ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Axis	Ausgang	AXIS_REF	In Kontaktplan-Programmen ist die Achsenausgabe schreibgeschützt.
Done	Ausgang	BOOL	TRUE – der Parameter wurde erfolgreich geschrieben.
Busy	Ausgang	BOOL	TRUE – gibt an, dass der Funktionsblock die Steuerung über die Achse besitzt.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
ErrorID	Ausgang	UINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler bei diesem Befehl sind in den Fehler-IDs des Achssteuerungs-Funktionsblocks auf seite 411 definiert.

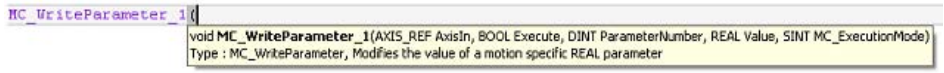
Beispiel für das Funktionsblockdiagramm MC_WriteParameter



Beispiel für Kontaktplan MC_WriteParameter

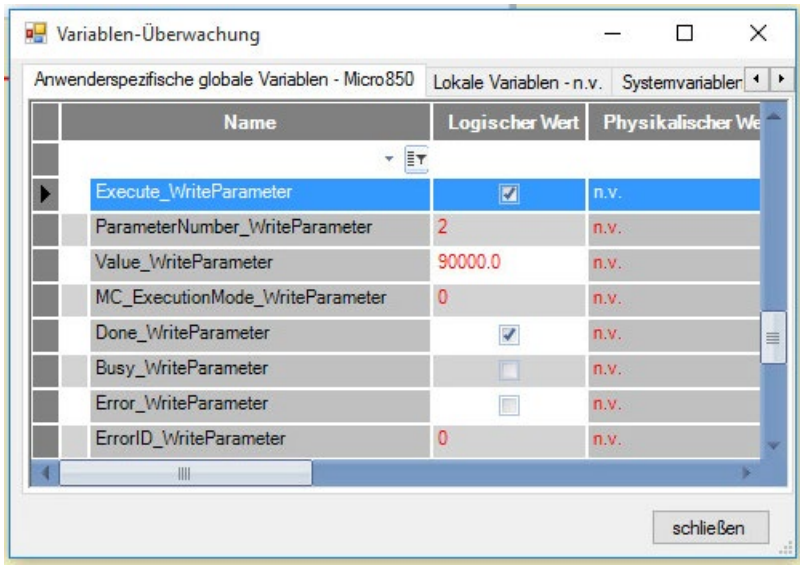


Beispiel für strukturierten MC_WriteParameter-Text




```
ParameterNumber_WriteParameter := 2;
Value_WriteParameter := 90000.0;
MC_WriteParameter_1(Axis1,Execute_WriteParameter,ParameterNumber_WriteParameter,
Value_WriteParameter,MC_ExecutionMode_WriteParameter);
Done_WriteParameter := MC_WriteParameter_1.Done;
Busy_WriteParameter := MC_WriteParameter_1.Busy;
Error_WriteParameter := MC_WriteParameter_1.Error;
ErrorID_WriteParameter := MC_WriteParameter_1.ErrorID;
```

Ergebnisse



Anweisungen für die Prozesssteuerung

Verwenden Sie Anweisungen für die Prozesssteuerung zur Überwachung und Einhaltung der Prozessschleifen für Größen wie Druck, Temperatur, Durchflussrate und Füllstand. Prozesssteuerungen regeln den Lauf, indem Sie ein Ausgabesignal an das Steuerventil senden.

Anweisung	Beschreibung
DERIVATE auf seite 475	Differenziert einen Real-Wert über eine definierte Zykluszeit.
FFL auf seite 477	Lädt 8-Bit-, 16-Bit-, 32-Bit- oder 64-Bit-Daten in ein vom Benutzer erzeugtes Array, einen so genannten FIFO-Stapel.
FFU auf seite 486	Entlädt 8-Bit-, 16-Bit-, 32-Bit- oder 64-Bit-Daten aus einem vom Benutzer erzeugten Array, einen so genannten FIFO-Stapel. Die Daten werden in der gleichen Reihenfolge entladen, wie sie mit der FFL-Anweisung geladen wurden.
HYSTER auf seite 490	Führt eineboolesche Hysterese bei Unterschieden von reellen Werten durch.
INTEGRAL auf seite 492	Integriert einen Real-Wert während der definierten Zykluszeit.
LFL (LIFO-Laden) auf seite 498	Lädt 8-Bit-, 16-Bit-, 32-Bit- oder 64-Bit-Daten in ein vom Benutzer erzeugtes Array, einen so genannten LIFO-Stapel.
LFU (LIFO-Entladen) auf seite 500	Entlädt 8-Bit-, 16-Bit-, 32-Bit- oder 64-Bit-Daten aus einem vom Benutzer erzeugten Array, einen so genannten LIFO-Stapel. Die Daten werden in der gleichen Reihenfolge entladen, wie sie mit der LFL-Anweisung geladen wurden.
PWM auf seite 502	Schaltet die Impulsbreitenmodulation (PWM)-Ausgabe für einen konfigurierten PWM-Kanal ein oder aus.
SCALER auf seite 505	Skaliert den Eingabewert entsprechend dem Ausgabebereich.
STACKINT auf seite 507	Verwaltet einen Stapel ganzzahliger Werte.
LIMIT auf seite 517	Beschränkt ganzzahlige Werte auf ein gegebenes Intervall.
TND auf seite 516	Hält den aktuellen Zyklus des Benutzerprogramm-Scans an.

DERIVATE

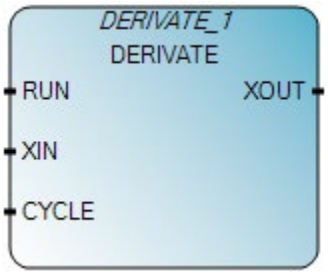
Differenzierung eines Real-Wertes über eine definierte Zykluszeit.

Details zum Betrieb:

- Wenn der Parameterwert „CYCLE“ geringer ist als die Zykluszeit für die Ausführung des Geräts, wird diese Zyklus-Zeitgebung für die Abtastperiode erzwungen.
- Die Ableitung erfolgt mit einer Zeitbasis von Millisekunden.Das heißt, das Ableiten einer Eingabe von 1000, die sich im Laufe eines Zeitraums von 1 Sekunde zu 2000 verändert, führt zu einem Wert von 1. Um die Ausgabe der Anweisung in die Einheit Sekunden umzuwandeln, muss die Ausgabe mit 1000 multipliziert werden.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

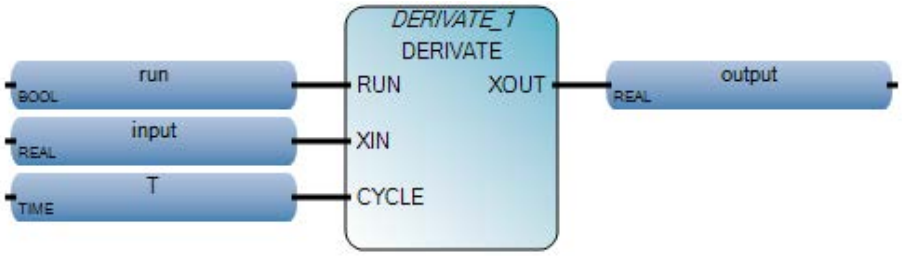
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



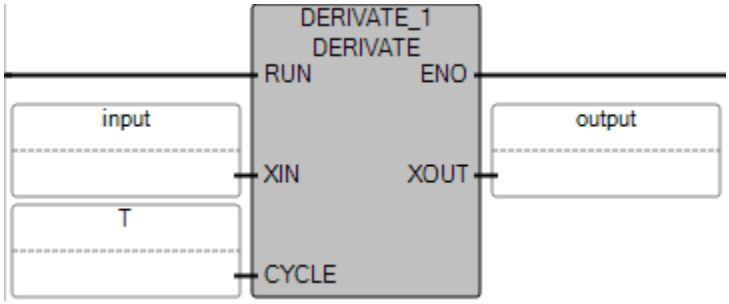
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
RUN	Eingang	BOOL	Zeigt den Betriebsmodus der Anweisung an. TRUE - normal (Berechnung durchführen) FALSE - Reset
XIN	Eingang	REAL	Definiert den Wert, für den die Ableitungsberechnung durchgeführt wird. Der Wert muss ein REAL-Wert sein.
CYCLE	Eingang	TIME	Definiert die Abtastperiode, während der Werte gesammelt werden. Die möglichen Zeitraumwerte liegen zwischen 0 ms und 49d17h2m47s294ms.
XOUT	Ausgang	REAL	Differenzierte Ausgabe.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

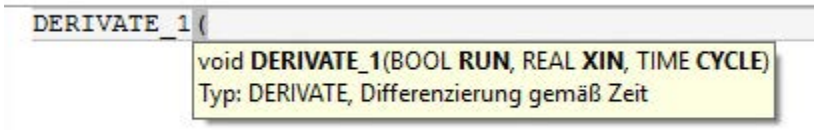
Diagrammbeispiel für Funktionsblock DERIVATE



DERIVATE-Kontaktplan-Beispiel



Beispiel für DERIVATE in strukturiertem Text




```
1  DERIVATE_1(run, input, T);
2  output := DERIVATE_1.XOUT;
```

(* ST-Äquivalenz: DERIVATE1 ist eine Instanz eines DERIVATE-Blocks *)

```
DERIVATE1(manual_mode, sensor_value, t#100ms);
derivated_value := DERIVATE1.XOUT;
```

FFL (FIFO laden)

Lädt 8-Bit-, 16-Bit-, 32-Bit- oder 64-Bit-Daten in ein vom Benutzer erzeugtes Array, einen so genannten FIFO-Stapel.

Details zum Betrieb:

- FFL-Anweisung - kein Ausführungsmodus bis Ausführungsmodus
  - Wenn Ausführung von FALSE auf TRUE wechselt:
    - Fehlerbedingung wird überprüft.
    - Der Inhalt von Src wird in den FIFO-Stapel in der verfügbaren Position geladen, und die Position erhöht sich um 1, wenn die Position kleiner oder gleich Null ist und weniger als die Länge beträgt.
    - Full ist auf TRUE eingestellt, wenn die Länge der Position entspricht. Full ist FALSE, wenn die Position geringer als oder gleich Null und kleiner als die Länge ist.
    - Done ist eingestellt, wenn die Anweisung erfolgreich ausgeführt wird.
  - Wenn Ausführung von TRUE auf FALSE wechselt:
    - Error, Done und ErrorID sind auf FALSE eingestellt.
    - Empty ist auf TRUE eingestellt, wenn die Position Null entspricht.
    - Full ist auf TRUE eingestellt, wenn die Länge der Position entspricht. Full ist FALSE, wenn die Position geringer als oder gleich Null und kleiner als die Länge ist.
    - FFL-Fehlerbedingungen werden nicht überprüft.
  - Wenn Ausführung von FALSE auf TRUE wechselt:
    - Es wird kein Ladevorgang ausgeführt.
    - Empty ist auf TRUE eingestellt, wenn die Position Null entspricht.
    - Full ist auf FALSE eingestellt, wenn die Position geringer als oder gleich Null und kleiner als die Länge ist. Full ist auf TRUE eingestellt, wenn die Länge der Position entspricht.
    - FFL-Fehlerbedingungen werden nicht überprüft.

- Wenn Ausführung von FALSE auf FALSE wechselt:
  - Die Bits Error, Done und ErrorID sind auf FALSE eingestellt.
  - Die Bits Full und Empty behalten Werte von vorherigen Ausführungszuständen bei.
- Empty ist auf TRUE eingestellt, wenn die Position Null entspricht.
  - Full ist auf TRUE eingestellt, wenn die Länge der Position entspricht. Full ist FALSE, wenn die Position geringer als oder gleich Null und kleiner als die Länge ist.
  - FFL-Fehlerbedingungen werden nicht überprüft.
- FFL-Anweisung - Ausführungsmodus bis kein Ausführungsmodus:
  - Error, ErrorID, Done, Empty und Full behalten den Ausführungsmoduszustand bei.

- Zum Erstellen eines einzelnen Elements für den FIFO-Parameter:
  - Nicht-Array:
    - Variablenbasierte Adresse wie z. B. Fifo1 ist für FIFO zulässig.
    - Länge sollte als 1 konfiguriert sein.
  - Array:
    - Variablenbasierte Adresse wie z. B. Fifo1 oder Fifo1[0] ist für FIFO zulässig.
    - Länge sollte als 1 konfiguriert sein.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Anweisungsblock aktivieren. TRUE - Wenn die Anstiegsflanke erkannt wird, starten Sie den FFL-Vorgang. FALSE – Anstiegflanke wird nicht erkannt.
Src	Eingang	ANY_ELEMENTARY	Der Src-Operand ist die Adresse des Werts, der zum Füllen der aktuell verfügbaren Position im

Parameter	Parametertyp	Datentyp	Beschreibung
			FIFO-Stapel verwendet wird. Elementare Datentypen, die für Scr unterstützt werden: • DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT. • String wird nicht unterstützt! • Array-Elemente wie Array[1] oder Array[Index] werden unterstützt.
FIFO	Eingang	ANY_ELEMENTARY	Die Startadresse des Stapels. FIFO muss für die Anweisungen FFL und FFU gleich konfiguriert sein. Elementare Datentypen, die für FIFO unterstützt werden: DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT. • String wird nicht unterstützt! • Nur eindimensionale Arrays werden für FIFO unterstützt.
FIFOCon	Eingang	FF_LF_CON	FIFO-Konfiguration und -Steuerung. Für FFL- und FFU-Anweisungen muss dieselbe Konfiguration konfiguriert werden. Verwenden Sie den Datentyp FF_LF_CON, um Position und Länge zu konfigurieren.
Done	Ausgang	BOOL	Gibt an, wenn der FFU-Vorgang abgeschlossen ist. TRUE – Vorgang wurde erfolgreich abgeschlossen. FALSE – Beim Vorgang ist eine Fehlerbedingung aufgetreten oder die FFU-Anweisung kann nicht ausgeführt werden.
Empty	Ausgang	BOOL	Gibt an, dass der FIFO-Stapel leer ist. TRUE – wenn die Position gleich 0 ist. FALSE – wenn die Position nicht gleich 0 ist.
Full	Ausgang	BOOL	Gibt an, dass der FIFO-Stapel voll ist. TRUE – wenn die Länge der Position entspricht. FALSE – wenn die Position größer als oder gleich Null und kleiner als die Länge ist.
Error	Ausgang	BOOL	Zeigt das Vorhandensein eine Fehlerbedingung an. TRUE – beim Vorgang ist ein Fehler aufgetreten. FALSCH – Vorgang wurde erfolgreich abgeschlossen oder der Befehl wird nicht ausgeführt.
ErrorID	Ausgang	USINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler sind in FFL-Fehlercodes definiert.

FF_LF_CON-Datentyp

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „FF_LF_CON“ zu bestimmen.

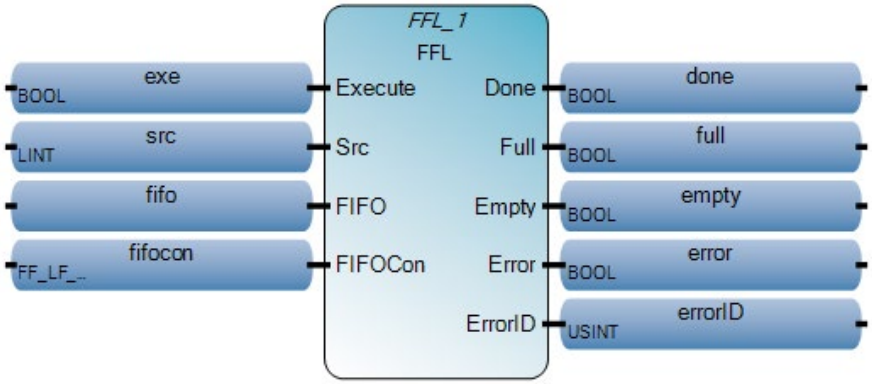
Parameter	Datentyp	Beschreibung
Length	UINT	Anzahl der Elemente, die für den FIFO-Vorgang verwendet werden. Maximal 1024.
Position	USINT	Bestimmt die nächste verfügbare Position im FIFO für den Src-Eintrag oder für das Entfernen. Die Position ist der Abstand des Arrays. Beispiel 1: • Benutzer konfiguriert ein Array, arr[0..5]. Zunächst ist die Position als 1 konfiguriert. Daten werden in arr[1] übertragen und die Position erhöht sich inkrementell auf Position + 1. Beispiel 2: • Benutzer konfiguriert ein Array als arr[1..5]. Zunächst ist die Position als 1 konfiguriert. Daten werden in arr[1] übertragen und die Position erhöht sich inkrementell auf Position + 1.

FFL-Fehlercodes

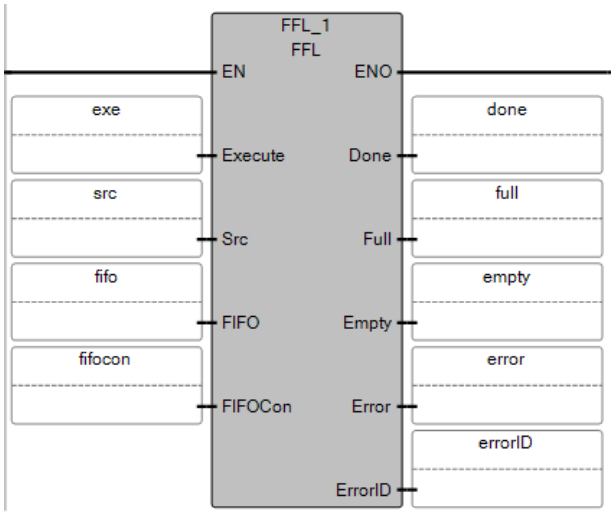
Verwenden Sie diese Tabelle, um die FFL- und FFU-Fehlercodes und deren Beschreibung zu bestimmen.

Fehlercode	Fehlerbeschreibung
0	Kein Fehler.
1	FFL-Src-Datentyp wird nicht unterstützt.
2	FFU-Dest-Datentyp wird nicht unterstützt.
3	FIFO-Datentyp wird nicht unterstützt.
4	Src- und Dest-Datentyp stimmen nicht mit dem FIFO-Datentyp überein. Maßnahme: Datentyp des FFL Src-Parameters und des FFU Dest-Parameters müssen mit dem FIFO Array-Datentyp übereinstimmen.
5	FIFO – Array-Dimension wird nicht unterstützt. Maßnahme: FIFO unterstützt nur eindimensionale Arrays.
6	FIFOCon-Steuerlementlänge überschreitet FIFO-Arraygröße. Maßnahme: FIFOCon-Steuerlementlänge darf FIFO-Arraygröße nicht überschreiten.
7	FIFOCon-Länge überschreitet die maximale Länge.
8	FIFOCon-Länge ist 0.
9	FIFOCon-Position überschreitet die FIFOCon-Länge.
10	FFL-Steuerelementlänge ist gleich der Position.
11	FFU-Steuerelementposition ist 0.
12	FFL- oder FIFO-Array-Dimension wird nicht unterstützt. Maßnahme: FFL und FFU unterstützen nur eindimensionale Arrays.
13	FFL- oder FFU-DestOffset überschreitet Dest-Array-Größe.

Beispiel für ein FFL-Funktionsblockdiagramm



Beispiel für einen FFL-Kontaktplan

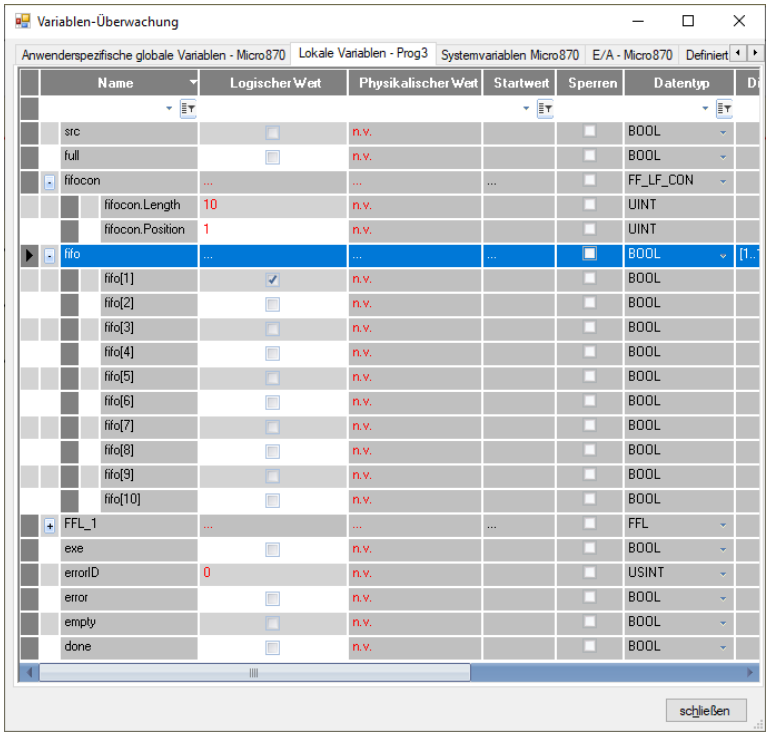


Beispiel für strukturierten FFL-Text

```
1  FFL_1(exe, src, fifo, fifocon);
2  done := FFL_1.Done;
3  full := FFL_1.Full;
4  empty := FFL_1.Empty;
5  error := FFL_1.Error;
6  errorID := FFL_1.ErrorID;
```

```
FFL_1(
void FFL_1(BOOL Execute, ANY_ELEMENTARY Src, ANY_ELEMENTARY[1..1] FIFO, FF_LF_CON FIFOCon)
Typ: FFL, FIFO-Last
```

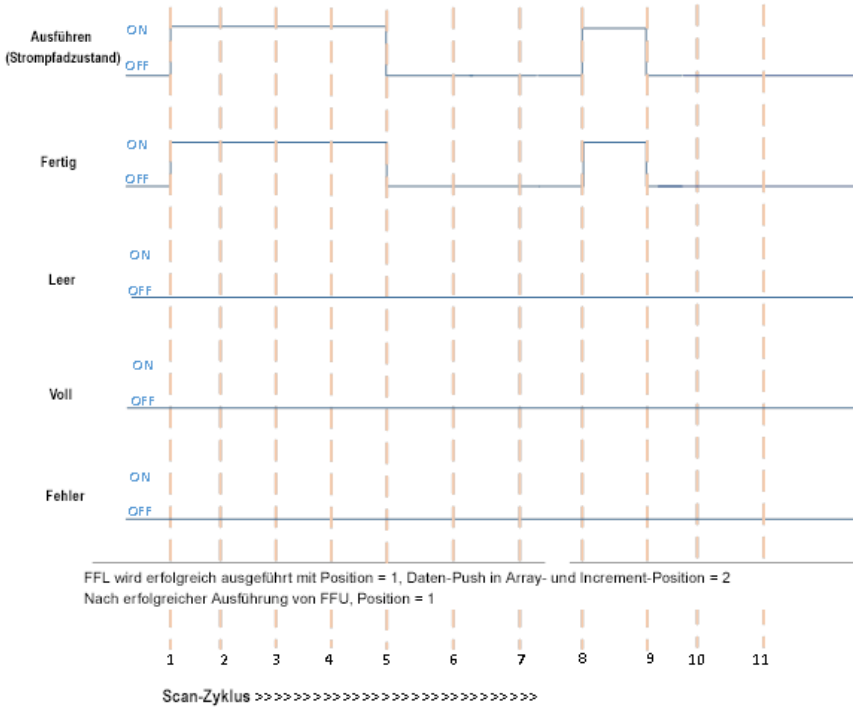
Ergebnisse



FFL- und FFU-Anweisung für Timing-Diagramme

In den folgenden Beispielen zu Zeitdiagrammen werden Ausführungsszenarien für die Befehle FFL auf seite 477 (FIFO-Laden) und FFU auf seite 486 (FIFO-Entladen) beschrieben.

Erfolgreiche FFL-Ausführung gefolgt von erfolgreicher FFU-Ausführung

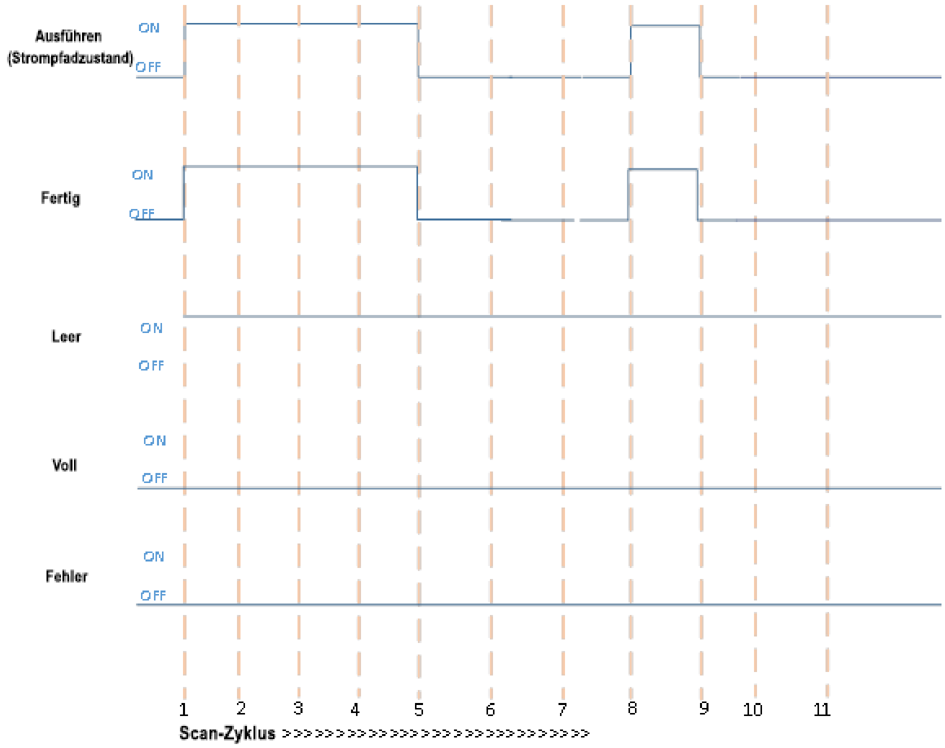


Verwenden Sie diese Tabelle, um die Parameterwerte für jeden Scan-Zyklus zu ermitteln.

Scan-Zyklus	Beschreibung
1	Strompfadbedingung wird TRUE, wenn gilt: • Execute-Eingangsbit ist TRUE. • Daten in FIFO-Stapel laden. • Done-Ausgangsbit ist TRUE.
2,3,4	Keine Änderung in Strompfadbedingung.
5	Strompfadbedingung wird FALSE, wenn gilt: • Execute-Bit ist FALSE. • Done-Ausgangsbit ist FALSE.
6, 7	Keine Änderung in Strompfadbedingung. • Execute-Bit ist FALSE. • Done-Ausgangsbit ist FALSE.
8	Strompfad wechselt zu TRUE, wenn gilt: • Execute-Eingangsbit ist TRUE. • Daten aus FIFO-Stapel entladen. • Done-Ausgangsbit ist TRUE.
9	Keine Änderung in Strompfadbedingung. • Execute-Bit ist FALSE. • Done-Ausgangsbit ist FALSE.

Scan-Zyklus	Beschreibung
10, 11	Keine Änderung in Strompfadbedingung.

Erfolgreich ausgeführt, wenn das Empty-Bit TRUE ist

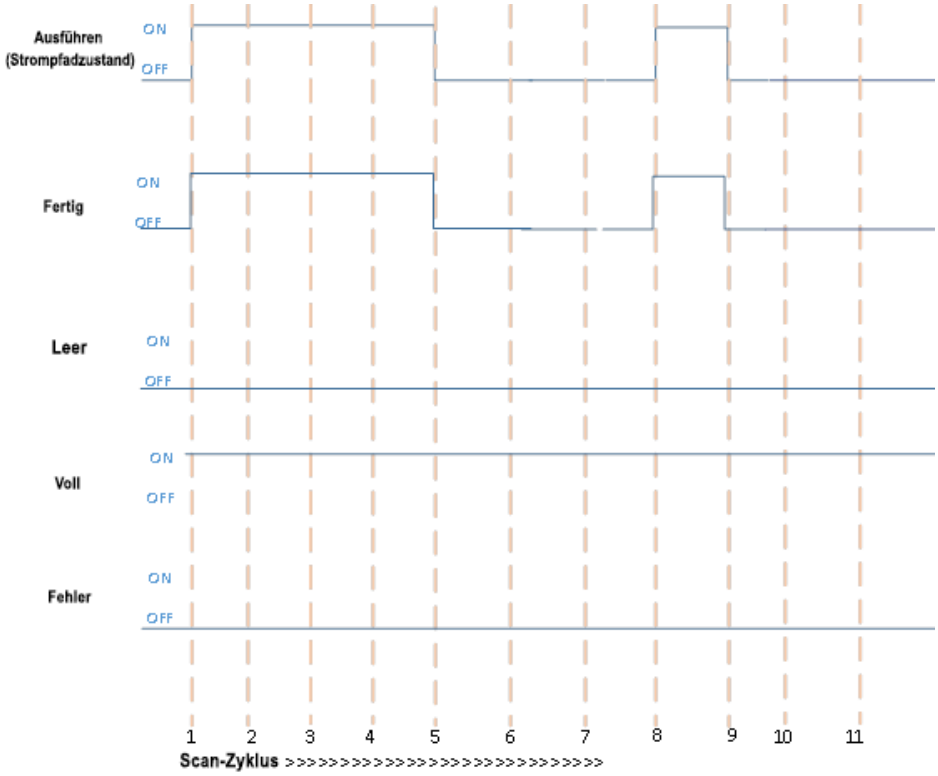


Verwenden Sie diese Tabelle, um die Parameterwerte für jeden Scan-Zyklus zu ermitteln.

Scan-Zyklus	Beschreibung
1	Strompfadbedingung wird TRUE, wenn gilt: • Execute-Eingangsbit ist TRUE. Die Ausführung beginnt. • Position ist Null. Empty-Bit ist TRUE. • Done-Ausgangsbit ist TRUE.
2,3,4	Keine Änderung in Strompfadbedingung.
5	Strompfadbedingung wird FALSE, wenn gilt: • Execute-Bit ist FALSE. • Empty-Bit ist TRUE. • Done-Ausgangsbit ist FALSE.
6, 7	Keine Änderung in Strompfadbedingung.
8	Strompfad wechselt zu TRUE, wenn gilt: • Execute-Eingangsbit ist TRUE. Die Ausführung beginnt. • Empty-Bit ist TRUE. • Done-Ausgangsbit ist TRUE.
9	Strompfadbedingung wird FALSE, wenn gilt: • Execute-Bit ist FALSE. • Empty-Bit ist TRUE. • Done-Ausgangsbit ist FALSE.

Scan-Zyklus	Beschreibung
10, 11	Keine Änderung in Strompfadbedingung.

Erfolgreich ausgeführt, wenn das Empty-Bit TRUE ist

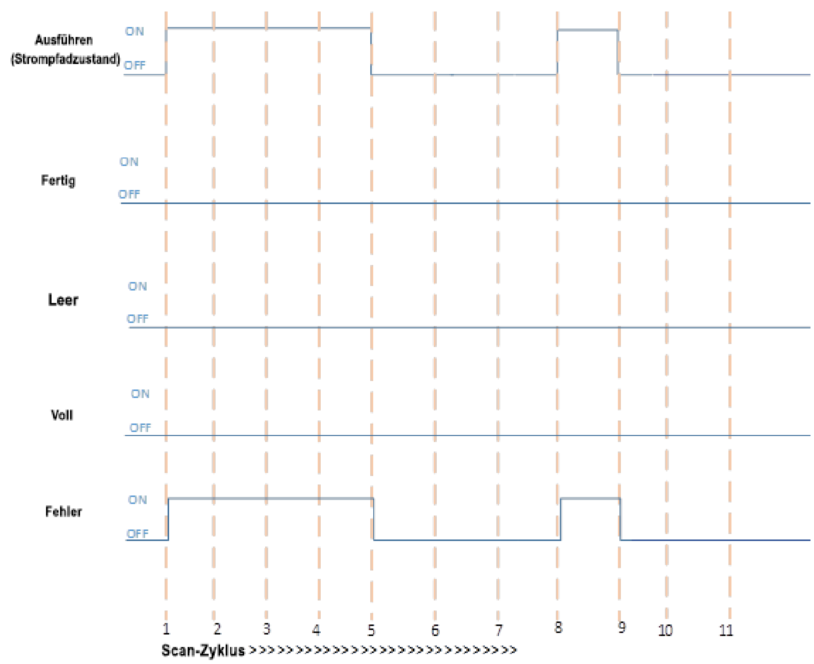


Verwenden Sie diese Tabelle, um die Parameterwerte für jeden Scan-Zyklus zu ermitteln.

Scan-Zyklus	Beschreibung
1	Strompfadbedingung wird TRUE, wenn gilt: • Execute-Eingangsbit ist TRUE. Die Ausführung beginnt. • Position ist gleich Length, Full-Bit ist TRUE. • Done-Ausgangsbit ist TRUE.
2,3,4	Keine Änderung in Strompfadbedingung.
5	Strompfadbedingung wird FALSE, wenn gilt: • Execute-Bit ist FALSE. • Full-Bit ist TRUE. • Done-Ausgangsbit ist FALSE.
6, 7	Keine Änderung in Strompfadbedingung.
8	Strompfad wechselt zu TRUE, wenn gilt: • Execute-Eingangsbit ist TRUE. Die Ausführung beginnt. • Full-Bit ist TRUE. • Done-Ausgangsbit ist TRUE.

Scan-Zyklus	Beschreibung
9	Strompfadbedingung wird FALSE, wenn gilt: • Execute-Bit ist FALSE. • Full-Bit ist TRUE. • Done-Ausgangsbit ist FALSE.
10, 11	Keine Änderung in Strompfadbedingung.

Fehler bei der Ausführung von FFL und FFU



Verwenden Sie diese Tabelle, um die Parameterwerte für jeden Scan-Zyklus zu ermitteln.

Scan-Zyklus	Beschreibung
	Strompfadbedingung wird TRUE, wenn gilt: • Execute-Eingangsbit ist TRUE. Die Ausführung beginnt. • Fehlerbit ist TRUE.
2,3,4	Keine Änderung in Strompfadbedingung.
5	Strompfadbedingung wird FALSE, wenn gilt: • Execute-Bit ist FALSE. • Error- und ErrorID-Bits sind FALSE.
6, 7	Keine Änderung in Strompfadbedingung.
8	Strompfad wechselt zu TRUE, wenn gilt: • Execute-Eingangsbit ist TRUE. Die Ausführung beginnt. • Fehlerbit ist TRUE.
9	Strompfadbedingung wird FALSE, wenn gilt: • Execute-Bit ist FALSE. • Error- und ErrorID-Bits sind FALSE.
10, 11	Keine Änderung in Strompfadbedingung.

FFU (FIFO entladen)

Entlädt 8-Bit-, 16-Bit-, 32-Bit- oder 64-Bit-Daten aus einem vom Benutzer erzeugten Array, einem so genannten FIFO-Stapel (First-in-First-Out), in der gleichen Reihenfolge, in der die Daten mithilfe der FFL-Anweisung geladen wurden.

Details zum Betrieb:

- FFU-Anweisung – vom Nicht-Ausführungsmodus zum Ausführungsmodus:
  - Wenn Ausführung von FALSE auf TRUE wechselt:
    - FFU-Fehlerbedingungen werden überprüft.
    - Entlädt den Inhalt des FIFO-Stapels an der Null-Position, wenn Position größer als Null und kleiner als oder gleich Length ist.
    - Verbleibende Elemente werden um eine Position in Richtung Null verschoben, das höchste Element des FIFO-Stapels wird auf Null gesetzt, Position wird um 1 verringert.
    - Empty ist auf TRUE eingestellt, wenn die Position Null entspricht.
    - Done ist eingestellt, wenn die Anweisung erfolgreich ausgeführt wird.
  - Wenn Ausführung von TRUE auf FALSE wechselt:
    - Error, Done und ErrorID sind auf FALSE eingestellt.
    - Empty ist auf TRUE eingestellt, wenn die Position Null entspricht.
    - Full ist auf TRUE eingestellt, wenn die Länge der Position entspricht. Full ist auf FALSE eingestellt, wenn die Position geringer als oder gleich Null und kleiner als die Länge ist.
    - FFU-Fehlerbedingungen werden nicht überprüft.
  - Wenn Ausführung von FALSE auf TRUE wechselt:
    - Es wird kein Entladevorgang ausgeführt.
    - Empty ist auf TRUE eingestellt, wenn die Position Null entspricht.
    - Full ist auf TRUE eingestellt, wenn die Länge der Position entspricht.
    - FFU-Fehlerbedingungen werden nicht überprüft.
  - Wenn Ausführung von FALSE auf FALSE wechselt:
    - Error, Done und ErrorID sind auf FALSE eingestellt.
    - Empty ist auf TRUE eingestellt, wenn die Position Null entspricht.
    - Full ist auf TRUE eingestellt, wenn die Länge der Position entspricht. Full ist auf FALSE eingestellt, wenn die Position geringer als oder gleich Null und kleiner als die Länge ist.
    - FLU-Fehlerbedingungen werden nicht überprüft.

- FFU-Anweisung – vom Ausführungsmodus zum Nicht-Ausführungsmodus:
  - Error, ErrorID, Done, Empty und Full behalten den Ausführungsmoduszustand bei.
- Zum Erstellen eines einzelnen Elements für den FIFO-Parameter:
  - Nicht-Array:
    - Variablenbasierte Adresse wie z. B. Fifo1 ist für FIFO zulässig.
    - Länge sollte als 1 konfiguriert sein.
  - Array:
    - Variablenbasierte Adresse wie z. B. Fifo1 oder Fifo1[0] ist für FIFO zulässig.
    - Länge sollte als 1 konfiguriert sein.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – wenn Anstiegflanke erkannt wird, FFU-Vorgang starten. FALSE – Anstiegflanke wird nicht erkannt.
Dest	Eingang	ANY_ELEMENTARY	Enthält den Wert, der im FIFO-Stapel vorhanden ist. Elementare Datentypen, die für Dest unterstützt werden: • DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT. • String wird nicht unterstützt! • Unterstützt nur eindimensionale Arrays wie Array[1] oder Array[Index].
DestOffset	Eingang	UINT	Abstand des Zielelements. Elementversatz wenn Zieladresse ein Array-Datentyp ist, ansonsten sollte der Versatz auf 0 eingestellt werden. Zum Entladen zum ersten Element für Array-Datentyp sollte der Versatz auf 0 eingestellt werden.

Parameter	Parametertyp	Datentyp	Beschreibung
FIFO	Eingang	ANY_ELEMENTARY	Die Startadresse des Stapels. FIFO muss für die FFL- und FFU-Anweisungen gleich konfiguriert sein. • Elementare Datentypen, die für FIFO unterstützt werden: DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT. • String wird nicht unterstützt! • Es wird nur eine Dimension für FIFO unterstützt.
FIFOCon	Eingang	FF_LF_CON	FIFO-Konfiguration und -Steuerung. Für FFL- und FFU-Anweisungen muss dieselbe Konfiguration konfiguriert werden. Verwenden Sie den Datentyp FF_LF_CON, um Position und Länge zu konfigurieren.
Done	Ausgang	BOOL	Gibt an, wenn der FFU-Vorgang abgeschlossen ist. TRUE – Vorgang wurde erfolgreich abgeschlossen. FALSE – Beim Vorgang ist eine Fehlerbedingung aufgetreten oder die FFU-Anweisung kann nicht ausgeführt werden.
Full	Ausgang	BOOL	Gibt an, dass der FIFO-Stapel voll ist. TRUE – wenn die Länge der Position entspricht. FALSE – wenn die Position größer als Null und kleiner als die Länge ist.
Empty	Ausgang	BOOL	Gibt an, dass der FIFO-Stapel leer ist. TRUE – wenn die Position gleich 0 ist. FALSE – wenn die Position nicht gleich 0 ist.
Error	Ausgang	BOOL	Zeigt das Vorhandensein eine Fehlerbedingung an. TRUE – beim Vorgang ist ein Fehler aufgetreten. FALSCH – Vorgang wurde erfolgreich abgeschlossen oder der Befehl wird nicht ausgeführt.
ErrorID	Ausgang	USINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler sind in FFU-Fehlercodes definiert.

FF_LF_CON-Datentyp

Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „FF_LF_CON“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Length	UINT	Anzahl der Elemente, die für den FIFO-Vorgang verwendet werden. Maximal 1024.
Position	USINT	Bestimmt die nächste verfügbare Position im FIFO für den Src-Eintrag oder für das Entfernen. Die Position ist der Abstand des Arrays. Beispiel 1: • Benutzer konfiguriert ein Array, arr[0..5]. Zunächst ist die Position als 1 konfiguriert. Daten werden in arr[1] übertragen und die Position erhöht sich inkrementell auf Position + 1. Beispiel 2: • Benutzer konfiguriert ein Array als arr[1..5]. Zunächst ist die Position als 1 konfiguriert. Daten werden in arr[1] übertragen und die Position erhöht sich inkrementell auf Position + 1.

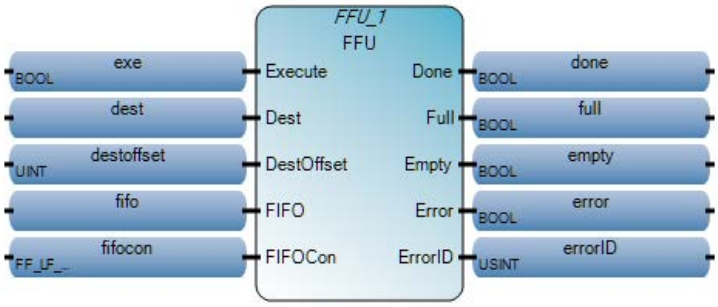
FFU-Fehlercodes

Verwenden Sie diese Tabelle, um die FFL- und FFU-Fehlercodes und deren Beschreibung zu bestimmen.

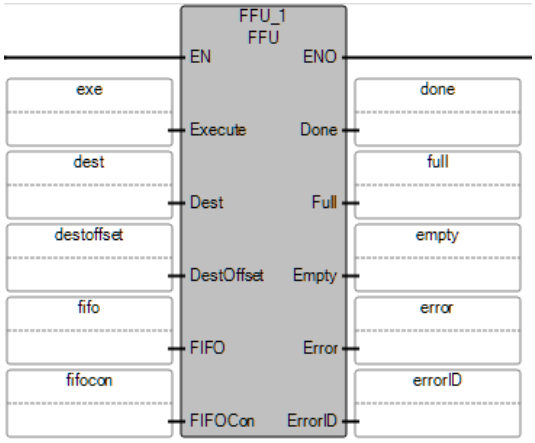
Fehlercode	Fehlerbeschreibung
0	Kein Fehler.

Fehlercode	Fehlerbeschreibung
1	FFL-Src-Datentyp wird nicht unterstützt.
2	FFU-Dest-Datentyp wird nicht unterstützt.
3	FIFO-Datentyp wird nicht unterstützt.
4	Src- und Dest-Datentyp stimmen nicht mit dem FIFO-Datentyp überein. Maßnahme: Datentyp des FFL Src-Parameters und des FFU Dest-Parameters müssen mit dem FIFO Array-Datentyp übereinstimmen.
5	FIFO – Array-Dimension wird nicht unterstützt. Maßnahme: FIFO unterstützt nur eindimensionale Arrays.
6	FIFOCon-Steuerlementlänge überschreitet FIFO-Arraygröße. Maßnahme: FIFOCon-Steuerlementlänge darf FIFO-Arraygröße nicht überschreiten.
7	FIFOCon-Länge überschreitet die maximale Länge.
8	FIFOCon-Länge ist 0.
9	FIFOCon-Position überschreitet die FIFOCon-Länge.
10	FFL-Steuerelementlänge ist gleich der Position.
11	FFU-Steuerelementposition ist 0.
12	FFL- oder FIFO-Array-Dimension wird nicht unterstützt. Maßnahme: FFL und FFU unterstützen nur eindimensionale Arrays.
13	FFL- oder FFU-DestOffset überschreitet Dest-Array-Größe.

Beispiel für ein FFU-Funktionsblockdiagramm



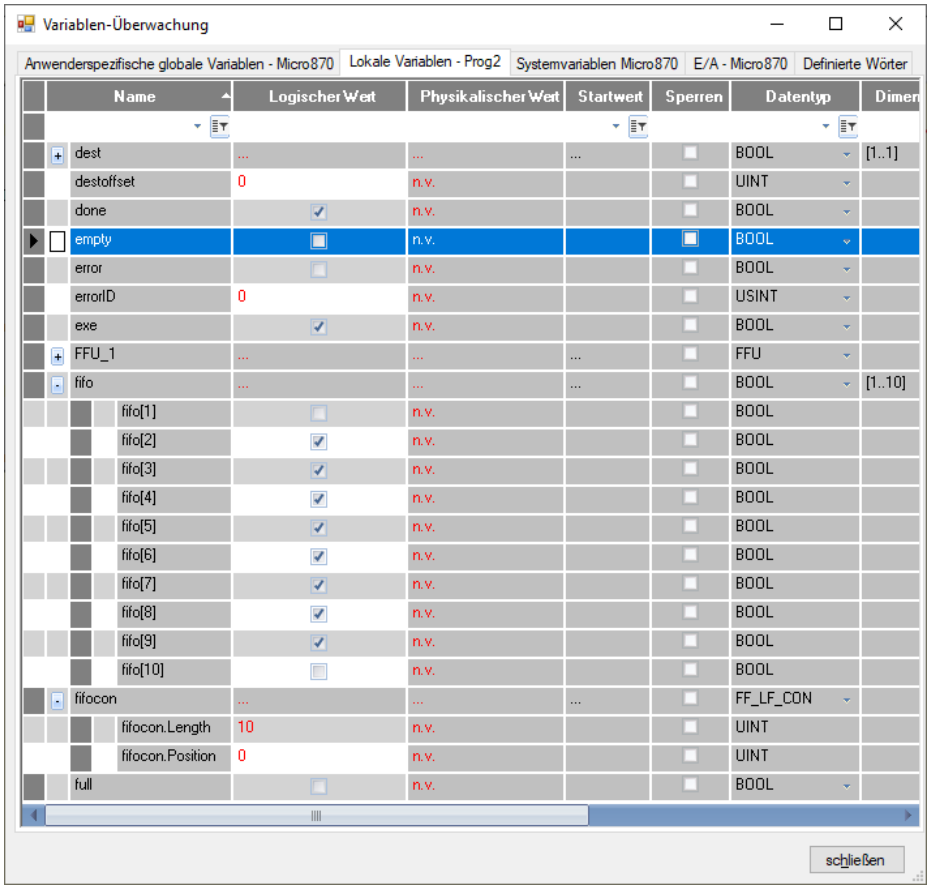
Beispiel für einen FFU-Kontaktplan



Beispiel für strukturierten FFU-Text

```
1  FFU_1(exe, dest, destoffset, fifo, fifocon);
2  done := FFU_1.Done;
3  full := FFU_1.Full;
4  empty := FFU_1.Empty;
5  error := FFU_1.Error;
6  errorID := FFU_1.ErrorID;
```

Ergebnisse

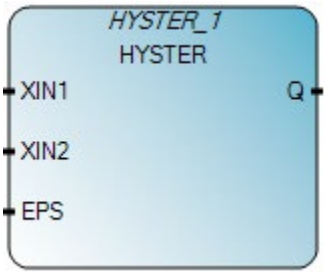


HYSTER (Hysterese)

Boolesche Hysterese bei Unterschieden von reellen Werten Vergleicht den aktuellen Wert einer Eingabe mit der Obergrenze, die durch Hinzufügen der historischen Verzögerung, gemessen durch Hysterese, festgelegt wurde, mit dem erwarteten Wert für eine Eingabe und ermittelt, ob der aktuelle Wert den Grenzwert überschreitet.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

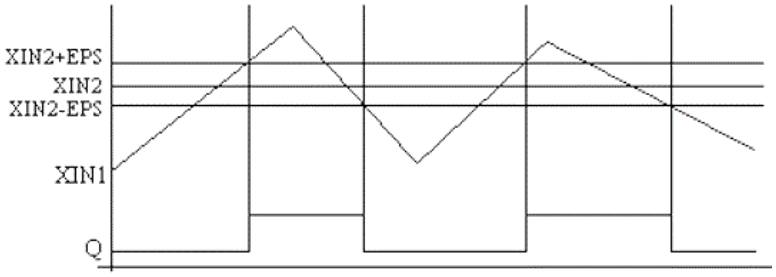


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

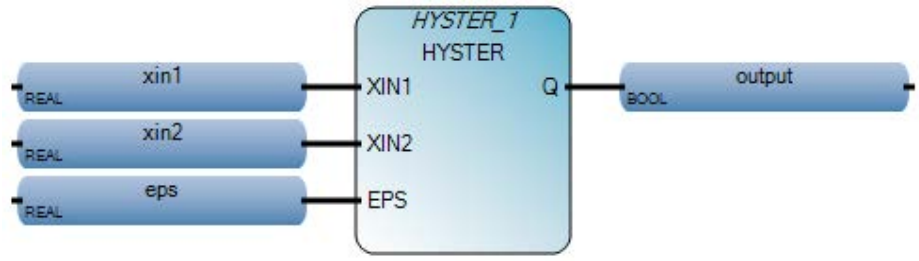
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Anweisungsblock ausführen. FALSE – Anweisungsblock nicht ausführen. Gilt nur für Kontaktplan-Programme.
XIN1	Eingang	REAL	Beliebiger realer Wert.
XIN2	Eingang	REAL	Zum Testen, ob der Eingabewert XIN1 die Obergrenze überschritten hat, die für diesen Eingabewert XIN2 + EPS definiert wurde.
EPS	Eingang	REAL	Hysterese-Wert (muss über Null liegen).
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Q	Ausgang	BOOL	Das Ergebnis der HYSTER-Anweisung. TRUE – Der Eingang überschritt die obere Grenze, liegt jedoch nicht unter der unteren Grenze. FALSE – Die Eingabe überschritt die obere Grenze nicht.

Beispiel eines HYSTER-Zeitgebungs-Diagramms

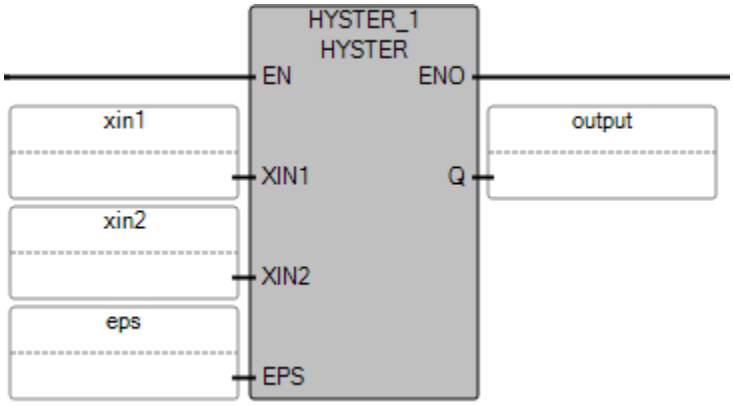
Im folgenden Diagramm wird HYSTER verwendet, um die Leistungsverzögerung aufgrund von Motorreibung über einen Zeitraums von 5 Sekunden zu ermitteln. Die Anweisung wird alle 10 Millisekunden ausgeführt. Während des Starts war der Betrieb des Motors weniger effizient.



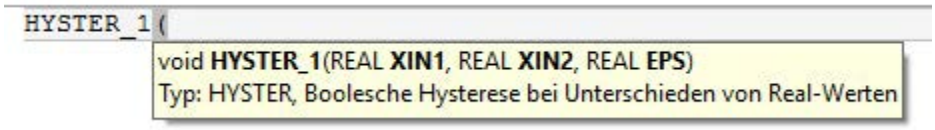
Diagrammbeispiel für Funktionsblock HYSTER



HYSTER-Kontaktplan-Beispiel



Beispiel für HYSTER in strukturiertem Text




```
1  xin1 := 10.0;
2  xin2 := 1.0;
3  eps := 1.0;
4  HYSTER_1(xin1, xin2, eps);
5  output := HYSTER_1.Q;
```

INTEGRAL

Integriert einen Real-Wert während der definierten Zykluszeit.

Details zum Betrieb:

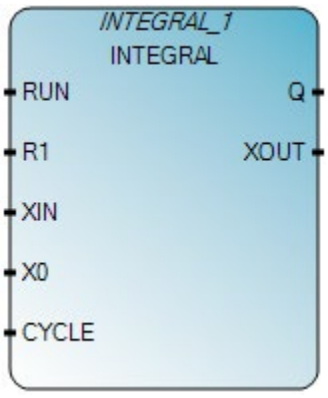
- Wenn der Funktionsblock „INTEGRAL“ zuerst initialisiert wird, werden seine anfänglichen Werte nicht berücksichtigt. Verwenden Sie den Parameter „R1“, um die Anfangswerte für eine Berechnung festzulegen.
- Um einen Verlust des integrierten Werts zu verhindern, wird der Integrationswert nicht automatisch gelöscht, wenn der Controller von „PROGRAM“ zu „RUN“ wechselt oder der Parameter „RUN“ sich von FALSE auf TRUE verändert. Verwenden Sie den Parameter „R1“, um den integralen Wert beim ersten Wechseln des Controllers aus dem

„PROGRAM“- in den „RUN“-Modus sowie beim Beginnen einer neuen Integration zu löschen.

- Es ist nicht ratsam, die optionalen EN- oder ENO-Parameter mit diesem Funktionsblock zu verwenden, da die Berechnung der Zykluszeit unterbrochen wird, wenn EN den Wert FALSE hat. Hierdurch käme es zu einer fehlerhaften Integration. Wenn die EN- oder ENO-Parameter verwendet werden, schalten Sie den R1-Parameter mit EN = TRUE um, um das aktuelle Ergebnis zu löschen und eine korrekte Integration sicherzustellen.
- Die Integration erfolgt mit einer Zeitbasis von Millisekunden (das heißt, das Integrieren einer Eingabe von 1 mit einem Anfangswert von 0 mit der Dauer von 1 Sekunde führt zu einem Wert von 1000). Um die Ausgabe der Anweisung in die Einheit Sekunden umzuwandeln, muss die Ausgabe durch 1000 dividiert werden.
- Wenn der Parameterwert „CYCLE“ geringer ist als die Zykluszeit für die Ausführung des Geräts, wird diese Zyklus-Zeitgebung für die Abtastperiode erzwungen.
- „XIN“-Abtastung und Funktionsblock werden bei jeder Zykluszeit + Scan-Zeitschwankung ausgeführt.
- Für ein bestimmtes Benutzerprogramm variiert die Scan-Zeitschwankung von Controller zu Controller.
- Die Zykluszeit bestimmt die Empfindlichkeit des Funktionsblocks „Integral“. Änderungen während XIN zwischen zwei Abtastungen (oder innerhalb der Zykluszeit) werden beim Berechnen des Integral-Werts XOUT nicht berücksichtigt.
- Die Zykluszeit und die Scan-Zeitschwankung tragen beide zur Ungenauigkeit der Integral-Ausgabe bei, wie im Beispiel von „XIN“ synchron mit Funktionsblockausführung und im Beispiel von „XIN“ nicht synchron mit Funktionsblockausführung gezeigt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

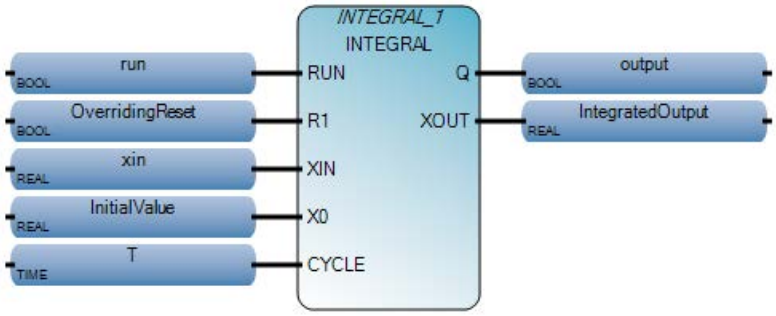
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



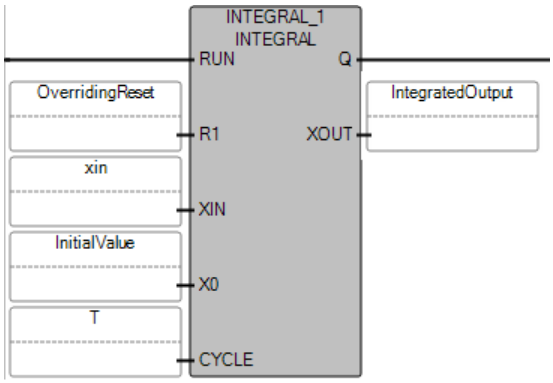
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
RUN	Eingang	BOOL	Modus: TRUE = integrieren / FALSE = halten.
R1	Eingang	BOOL	Zurücksetzen wird außer Kraft gesetzt.
XIN	Eingang	REAL	Eingabe: jeglicher Wert vom Typ „Real“.
X0	Eingang	REAL	Anfangswert.
CYCLE	Eingang	TIME	Abtastperiode. Mögliche Werte reichen von 0ms bis 49d17h2m47s294ms.
Q	Ausgang	BOOL	Nicht R1.
XOUT	Ausgang	REAL	Integrierte Ausgabe.

Diagrammbeispiel für Funktionsblock INTEGRAL



INTEGRAL-Kontaktplan-Beispiel



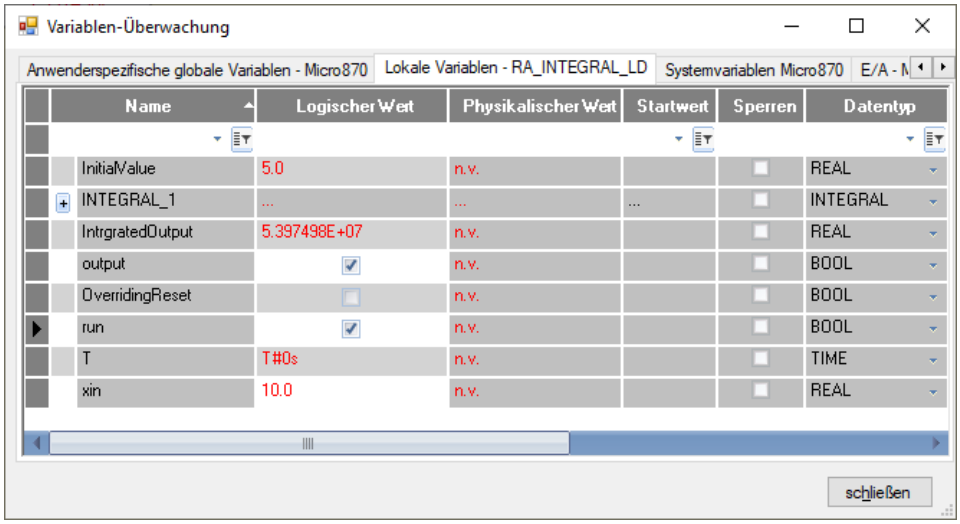
Beispiel für INTEGRAL in strukturiertem Text

```
1  run := TRUE;
2  xin := 10.0;
3  InitialValue := 5.0;
4  T := T#10s;
5  INTEGRAL_1(run, OverridingReset, xin, InitialValue, T);
6  output := NOT OverridingReset;
7  IntegratedOutput := INTEGRAL_1.XOUT;
```

(* ST-Äquivalenz: INTEGRAL1 ist eine Instanz eines INTEGRAL-Blocks *)

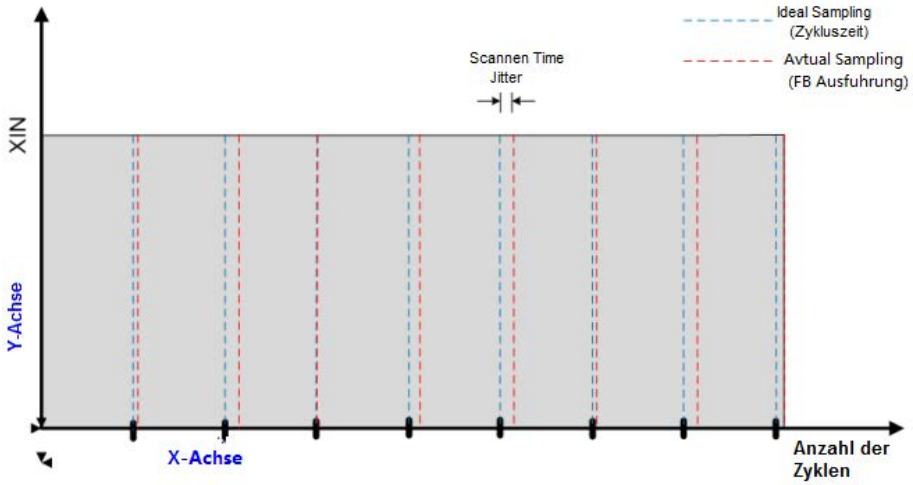
```
INTEGRAL1(manual_mode, NOT(manual_mode), sensor_value,
init_value, t#100ms);
controlled_value := INTEGRAL1.XOUT;
```

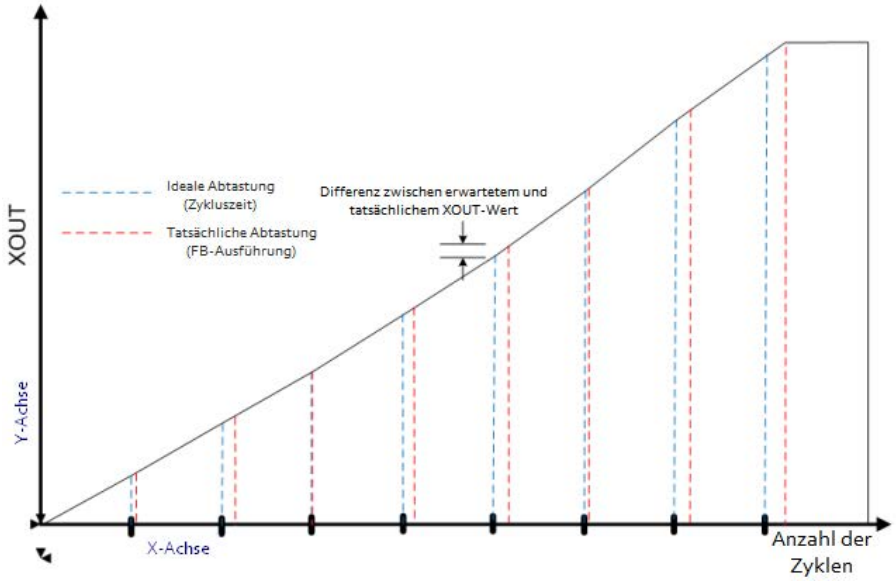
Ergebnisse



Beispiel: „XIN“ ist synchron mit Funktionsblockausführung

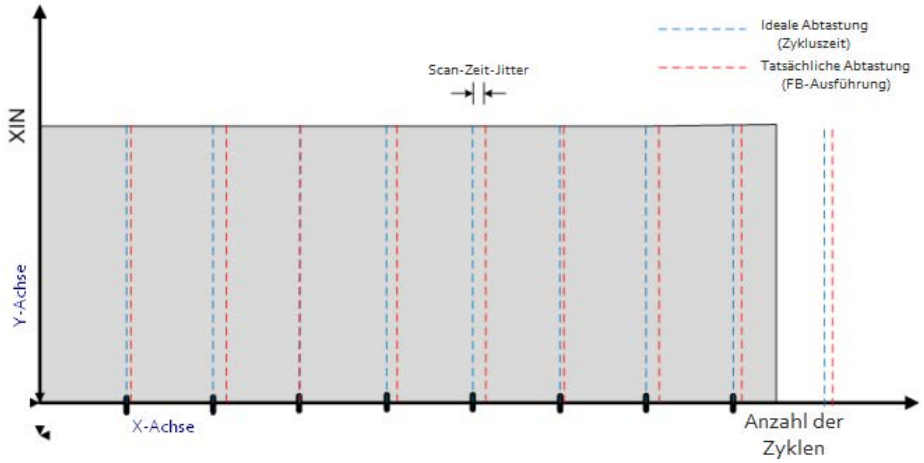
Die nachfolgenden Bilder zeigen die Auswirkung der Scan-Zeitschwankung auf den XOUT-Wert:

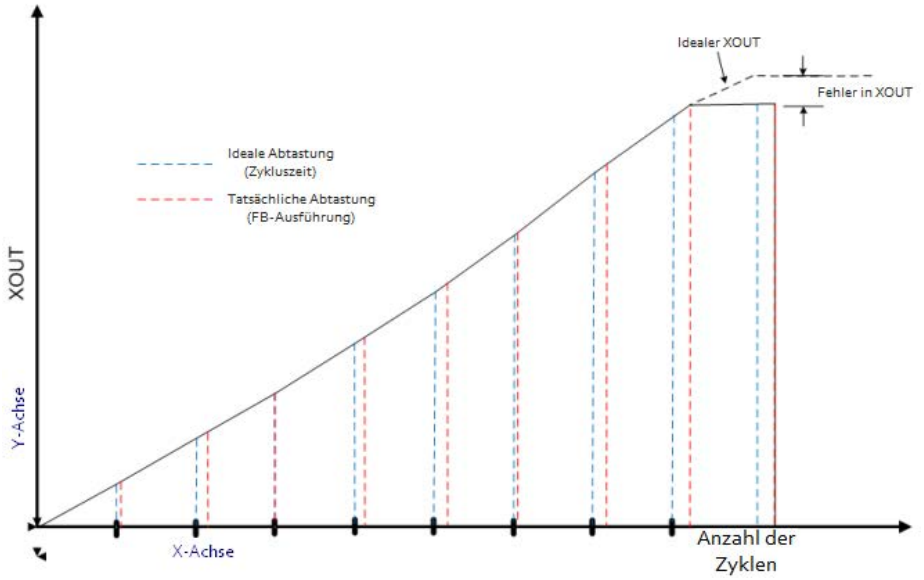




XIN nicht synchron mit Funktionsblockausführung

Die nachfolgenden Bilder zeigen ein Beispiel, in dem ein Fehler in den XOUT-Wert eines „Integral“-Funktionsblocks eingeführt wird:



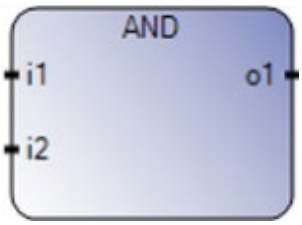


AND

Führt eine boolesche AND-Operation zwischen zwei oder mehr Werten aus.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
i1	Eingang	BOOL	Wert als boolescher Datentyp.
i2	Eingang	BOOL	Wert als boolescher Datentyp.
o1	Ausgang	BOOL	Ergebnis der booleschen AND-Operation der Eingabewerte.

Beispiel für strukturierten AND-Text

(* ST-Äquivalenz: *)

```
bo10 := bi101 AND NOT (bi102);
bo5 := (bi51 AND bi52) AND bi53;
```

LFL(LIFO-Last)

Mithilfe der LFL-Anweisung werden die Daten (8-Bit, 16-Bit, 32-Bit oder 64-Bit) in ein vom Benutzer erzeugtes Array, einen so genannten LIFO-Stapel geladen. Die Befehle LFL und LFU auf seite 500 werden in Paaren verwendet.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

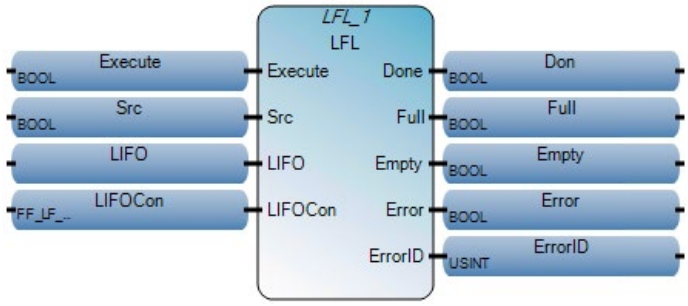
Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisungen zu ermitteln.

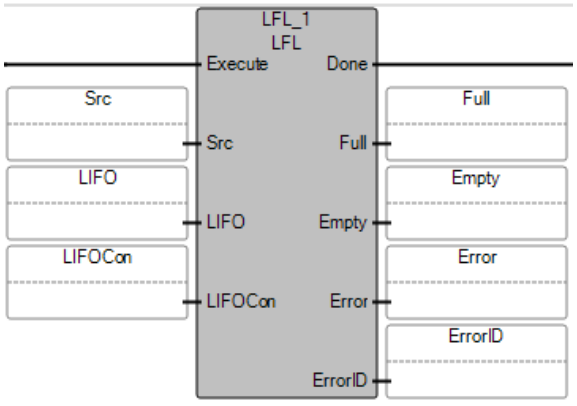
Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Anweisungsblock aktivieren. TRUE - Wenn die Anstiegsflanke erkannt wird, starten Sie den LFL-Vorgang. FALSE – Anstiegflanke wird nicht erkannt.
Src	Eingang	ANY_ELEMENTARY	Der Src-Operand ist die Adresse des Wertes, der zum Füllen der aktuell verfügbaren Position im LIFO-Stapel verwendet wird. Unterstützte elementare Datentypen: • DWORD, REAL, TIME, DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT.\ • Array-Element, wie z. B. Array [1] oder Array [Index]. • String wird nicht unterstützt
LIFO	Eingang	ANY_ELEMENTARY	Die Startadresse des Stapels. LIFO muss für die LFL- und LFU-Anweisungen genauso konfiguriert werden. Unterstützte elementare Datentypen: • DWORD, REAL, TIME, DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL,

Parameter	Parametertyp	Datentyp	Beschreibung
			SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT. • Nur einzelne Dimension wird unterstützt. • String wird nicht unterstützt!
LIFOCon	Eingang	FF_LF_CON	LIFO-Konfiguration und -Steuerung. Dieselbe Konfiguration muss für LFL und LFU-Anweisungen konfiguriert werden. Verwenden Sie den Datentyp FF_LF_CON, um Position und Länge zu konfigurieren.
Full	Ausgang	BOOL	Gibt an, wenn der LIFO Stapel vollständig ist. TRUE – wenn die Länge der Position entspricht. FALSE – wenn die Position größer als oder gleich Null und kleiner als die Länge ist.
Empty	Ausgang	BOOL	Gibt an, wenn der LIFO Stapel leer ist. TRUE - Wenn Position gleich Null. FALSE – Wenn Position nicht gleich Null ist.
Error	Ausgang	BOOL	Zeigen das Vorhandensein eine Fehlerbedingung an. TRUE – beim Vorgang ist ein Fehler aufgetreten. FALSE - Vorgang erfolgreich abgeschlossen oder Anweisung kann nicht ausgeführt werden.
ErrorID	Ausgang	USINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler sind in den LFL-Fehlercodes definiert.
Done	Ausgang	BOOL	Gibt an, wenn der Vorgang abgeschlossen ist. TRUE - Vorgang erfolgreich abgeschlossen. FALSE - Beim Vorgang ist eine Fehlerbedingung aufgetreten oder die LFL-Anweisung kann nicht ausgeführt werden.

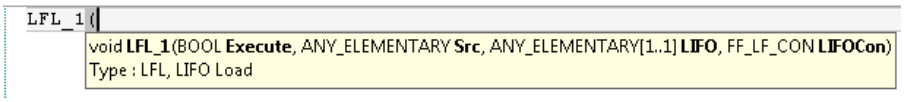
Beispiel für ein LFL-Funktionsblockdiagramm



Beispiel für einen LFL-Kontaktplan



Beispiel für strukturierten LFL-Text




```
1  LFL_1(exe, Src, fifo, fifocon);
2  Done:=LFL_1.Done;
3  Full:=LFL_1.Full;
4  Empty:=LFL_1.Empty;
5  Error:=LFL_1.Error;
6  ErrorID:=LFL_1.ErrorID;
```

LFU(LIFO-Entlastung)

Mithilfe der LFU-Anweisung werden Daten (8-Bit, 18-Bit, 32-Bit, 64-Bit) von einem Benutzer entladen – Array namens LIFO-Stapel erstellt. Die Befehle LFU und LFL auf seite 498 werden in Paaren verwendet.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

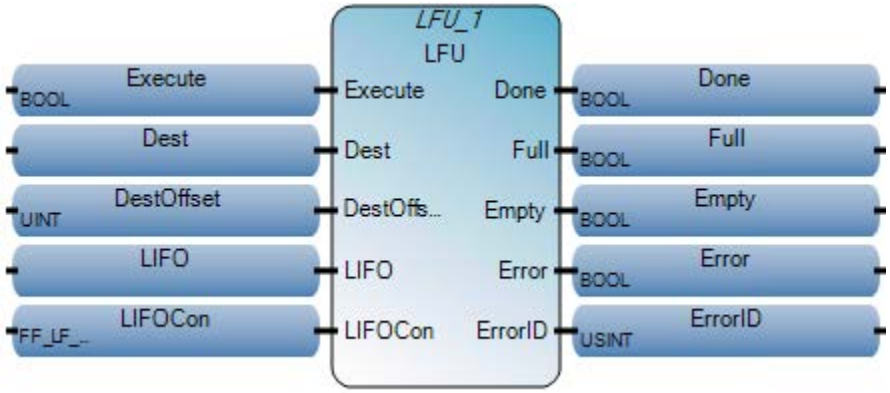
Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisungen zu ermitteln.

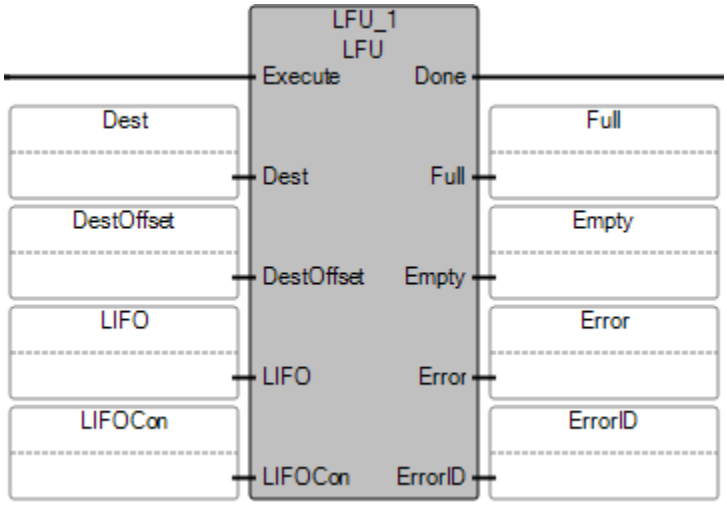
Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Wenn Anstiegflanke erkannt wurde, starten Sie den LFU-Vorgang. FALSE – Anstiegflanke wird nicht erkannt.
Dest	Eingang	ANY_ELEMENTARY	Enthält den Wert, der im LIFO-Stapel vorhanden ist. Elementare Datentypen, die für Dest unterstützt werden: • DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT. • String wird nicht unterstützt! • Unterstützt nur eindimensionale Arrays wie Array[1] oder Array[Index].
DestOffset	Eingang	UINT	Abstand des Zielelements. Elementversatz wenn Zieladresse ein Array-Datentyp ist, ansonsten sollte der Versatz auf 0 eingestellt werden.

Parameter	Parametertyp	Datentyp	Beschreibung
			Zum Entladen zum ersten Element für Array-Datentyp sollte der Versatz auf 0 eingestellt werden.
LIFO	Eingang	ANY_ELEMENTARY	Die Startadresse des Stapels. LIFO muss für die LFL- und LFU-Anweisungen genauso konfiguriert werden. Unterstützte elementare Datentypen: • DWORD, REAL, TIME, DWORD, REAL, TIME, DATE, LWORD, ULINT, LINT, LREAL, BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT. • Nur einzelne Dimension wird unterstützt. • String wird nicht unterstützt!
LIFOCon	Eingang	FF_LF_CON	LIFO-Konfiguration und -Steuerung. Dieselbe Konfiguration muss für LFL und LFU-Anweisungen konfiguriert werden. Verwenden Sie den Datentyp FF_LF_CON, um Position und Länge zu konfigurieren.
Full	Ausgang	BOOL	Gibt an, wenn der LIFO Stapel vollständig ist. TRUE – wenn die Länge der Position entspricht. FALSE – wenn die Position größer als oder gleich Null und kleiner als die Länge ist.
Error	Ausgang	BOOL	Zeigen das Vorhandensein eine Fehlerbedingung an. TRUE – beim Vorgang ist ein Fehler aufgetreten. FALSE - Vorgang erfolgreich abgeschlossen oder Anweisung kann nicht ausgeführt werden.
ErrorID	Ausgang	USINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler sind in den LFU-Fehlercodes definiert.
Done	Ausgang	BOOL	Gibt an, wenn der Vorgang abgeschlossen ist. TRUE – Vorgang erfolgreich abgeschlossen. FALSE – Beim Vorgang ist ein Fehler aufgetreten oder die LFU-Anweisung wird nicht ausgeführt.

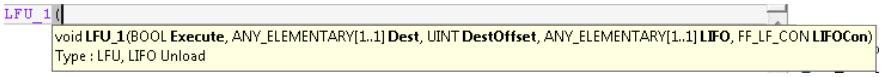
Beispiel für ein LFU-Funktionsblockdiagramm



Beispiel für ein LFU-Kontaktplandiagramm



Beispiel für einen strukturierten Text




```
1  LFU_1(exe, Dest, destoffset, lifo, lifocon);
2  Done:=LFU_1.Done;
3  Full:=LFU_1.Full;
4  Empty:=LFU_1.Empty;
5  Error:=LFU_1.Error;
6  ErrorID:=LFU_1.ErrorID;
```

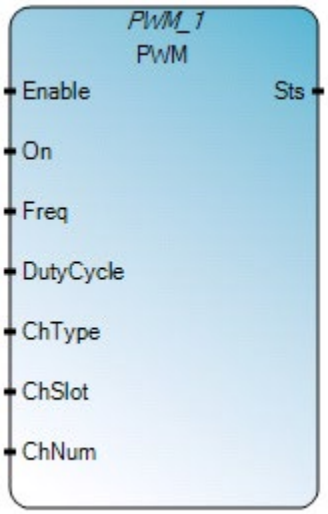
PWM (Impulsbreitenmodulation)

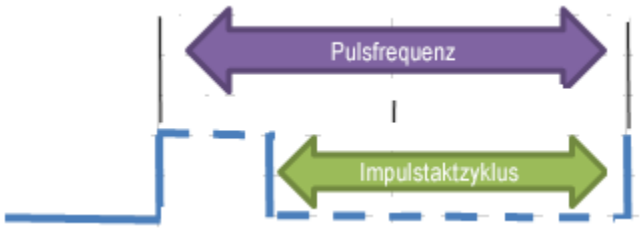
Schaltet den PWM (Impulsbreitenmodulation)-Ausgang für einen konfigurierten PWM-Kanal ON oder OFF.

Dieser Anweisungsblock wird bei Micro820 2080-LC20-20QBB-Controllern verwendet und unterstützt einen PWM-Kanal (über den integrierten Ausgangskanal 6).

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Dieser Befehl gilt nur für Micro820-Controller.



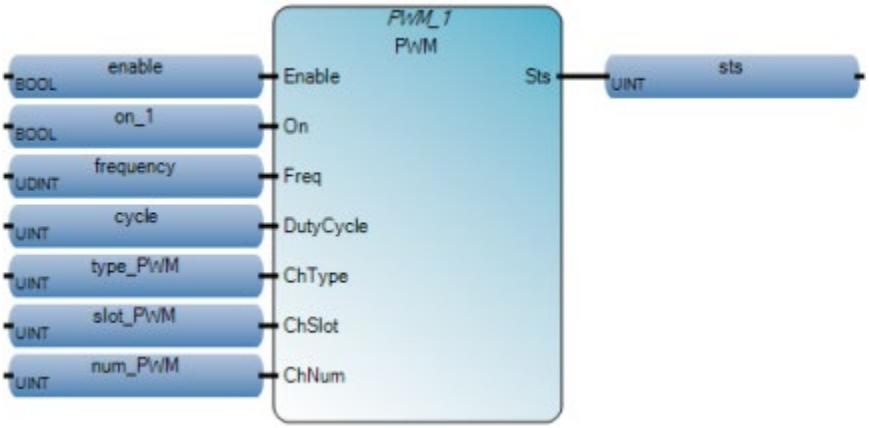


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

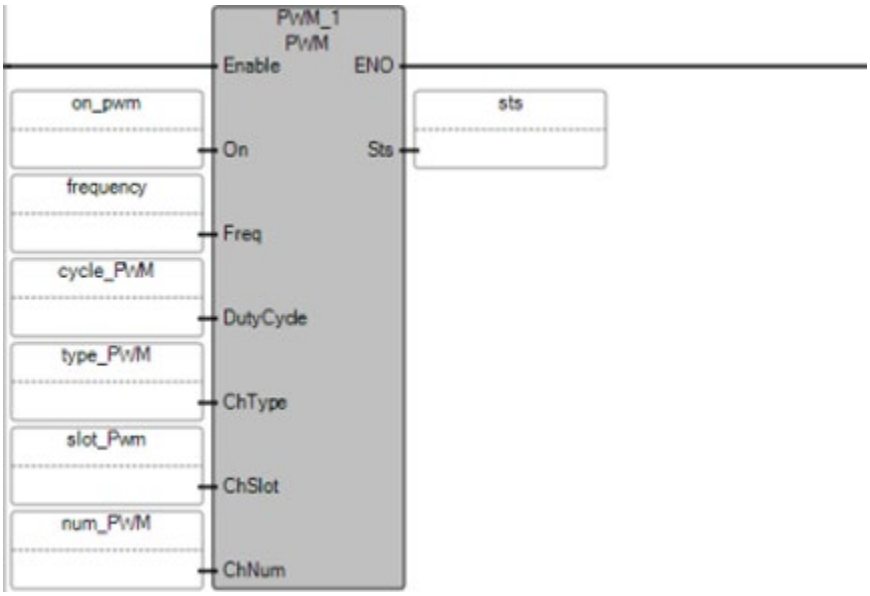
Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. Diese Ebene wird durch einen Anweisungsblock ausgelöst. TRUE – Status aktualisieren. PWM wird je nach On-Eingangsparameter und gültiger Konfiguration aktiv oder inaktiv. FALSE – Nur Status wird aktualisiert. Der PWM-Zustand (aktiv oder inaktiv) ist nicht betroffen.
On	Eingang	BOOL	Schaltet den PWM-Ausgang ON bzw. auf „Aktiv" oder OFF bzw. auf „Inaktiv". TRUE = Der PWM-Ausgang wird aktiviert oder ist mit der letzten gültigen Konfiguration weiterhin aktiv. Die Ausgangs-LED ist eingeschaltet, wenn PWM aktiv ist, auch wenn der Arbeitszyklus auf 0 % eingestellt wurde. FALSE = Der PWM-Ausgang ist inaktiv, wenn die Konfiguration ebenfalls gültig ist.
Freq	Eingang	UDINT	Impuls-Frequenz. • 1 bis 100000 Hz
DutyCycle	Eingang	UINT	Impuls-Arbeitszyklus. • 0-1000 (0-100 %)
ChType	Eingang	UINT	Kanaltyp • 0 – Integriert • 1 – Plug-in • 2 – Erweiterung
ChSlot	Eingang	UINT	Kanalsteckplatz • 0 – Integriert
ChNum	Eingang	UINT	Kanalnummer • 0 – PWM CH0

Parameter	Parametertyp	Datentyp	Beschreibung
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.
Sts	Ausgang	UINT	PWM-Statuscodes: • 00 – Funktionsblock nicht aktiviert (kein Betrieb). • 01 – PWM-Konfiguration erfolgreich. • 02 – Ungültiger Arbeitszyklus. • 03 – Ungültige Frequenz. • 04 – Ungültiger Kanaltyp. • 05 – Ungültiger Kanalsteckplatz. • 06 – Ungültige Kanalnummer. • 07 – Ungültiger Katalog. Die PWM-Funktion wird im verwendeten Katalog nicht unterstützt.

Beispiel für ein PWM-Funktionsblockdiagramm



Beispiel für einen PWM-Kontaktplan



Beispiel für strukturierten PWM-Text

```
PWM_1(
```

void **PWM_1**(BOOL **Enable**, BOOL **On**, UDINT **Freq**, UINT **DutyCycle**, UINT **ChType**, UINT **ChSlot**, UINT **ChNum**)
Typ: PWM, Aktivieren des PWM-Ausgangs.

```
1  PWM (EN, Enable, On, Freq, DutyCycle, ChType, ChSlot, ChNum);
2  output := PWM.ENO
3  sts := PWM.Sts
```

SCALER (Skalieren)

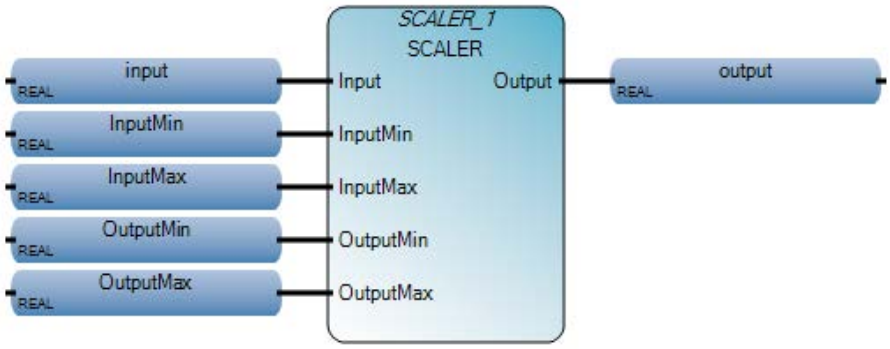
Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.,

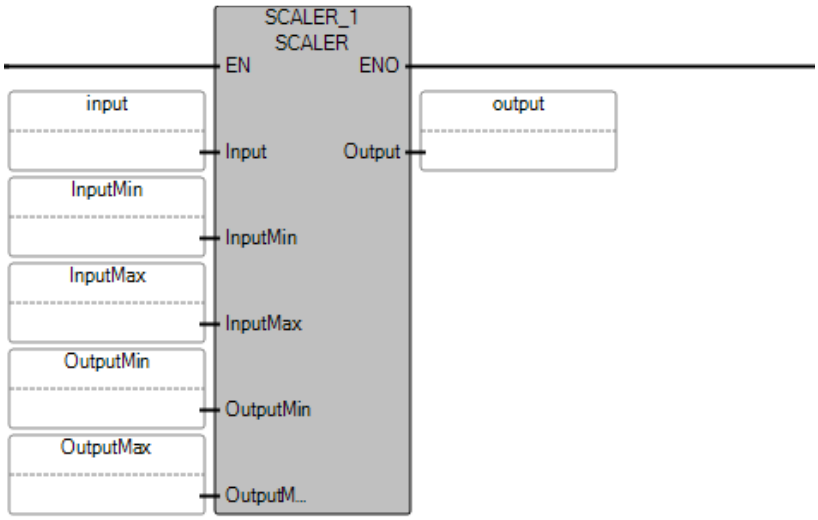
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisungsblock aktivieren. TRUE - Ausführung der Skalierungsgleichung. FALSE - Es ist keine Skalierungsgleichung vorhanden. Gilt nur für Kontaktplan-Programme.
Input	Eingang	REAL	Eingangssignal. Die Eingabe ist nicht durch InputMin und InputMax beschränkt. Um die Eingabe zu beschränken ist eine LIM_ALRM-Anweisung erforderlich, damit die Eingabe angepasst ist, bevor er in die SCALER-Anweisung eingegeben wird.
InputMin	Eingang	REAL	Neigung und Offset-Wert ermitteln.
InputMax	Eingang	REAL	Neigung und Offset-Wert ermitteln.
OutputMin	Eingang	REAL	Neigung und Offset-Wert ermitteln.
OutputMax	Eingang	REAL	Neigung und Offset-Wert ermitteln.
Output	Ausgang	REAL	Skalierter Ausgang. Der Ausgang ist nicht zwischen OutputMin und OutputMax festgelegt.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

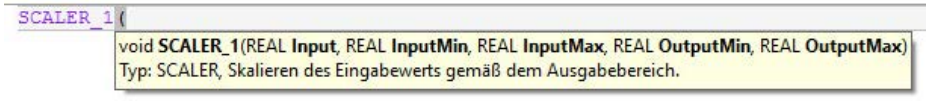
Beispiel für ein SCALER-Funktionsblockdiagramm



Beispiel für einen SCALER-Kontaktplan



Beispiel für einen strukturierten SCALER-Text

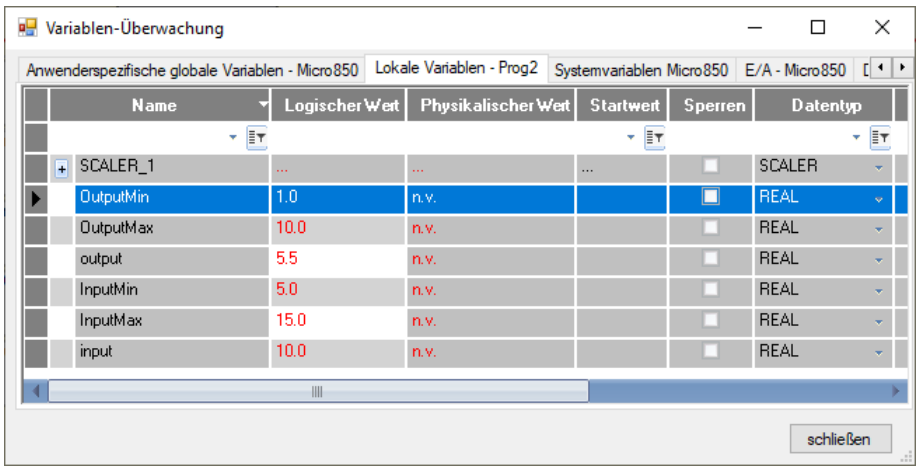



```
1  input := 10.0;
2  InputMin := 5.0;
3  InputMax := 15.0;
4  OutputMin := 1.0;
5  OutputMax := 10.0;
6  SCALER_1(input, InputMin, InputMax, OutputMin, OutputMax);
7  output := SCALER_1.Output;
```

(* ST-Äquivalenz: SCALER1 ist eine Instanz des Blocks SCALER *)

```
SCALER1(Signal_In, 4.0, 20.0 , 0.0 , 150.0 ) ;
Out_Temp := SCALER1.Output ;
```

Ergebnisse



STACKINT (Ganzzahl-Stack)

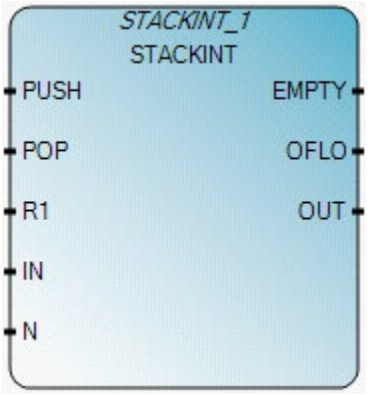
Verwaltet einen Stapel ganzzahliger Werte.

Details zum Betrieb:

- STACKINT enthält eine Erkennungsfunktion für ansteigende Flanken bei PUSH- und POP-Befehlen. Die Maximalgröße des Stapels ist 128. Der OFLO-Wert ist nur nach einem Zurücksetzen gültig (R1 wurde mindestens ein Mal auf TRUE und wieder zurück auf FALSE gesetzt).
- Die in der Anwendung definierte Stapelgröße (N) kann nicht unter 1 oder über 128 liegen.
  - Wenn N > 1, nimmt STACKINT den Wert 1 an.
  - Wenn N > 128, nimmt STACKINT den Wert 128 an.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

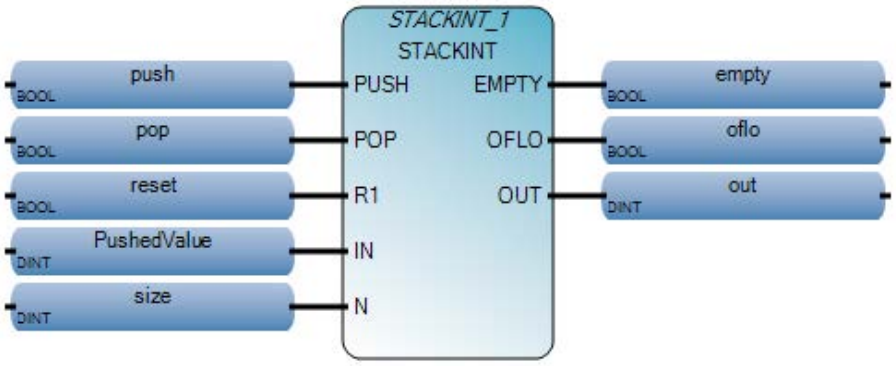
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



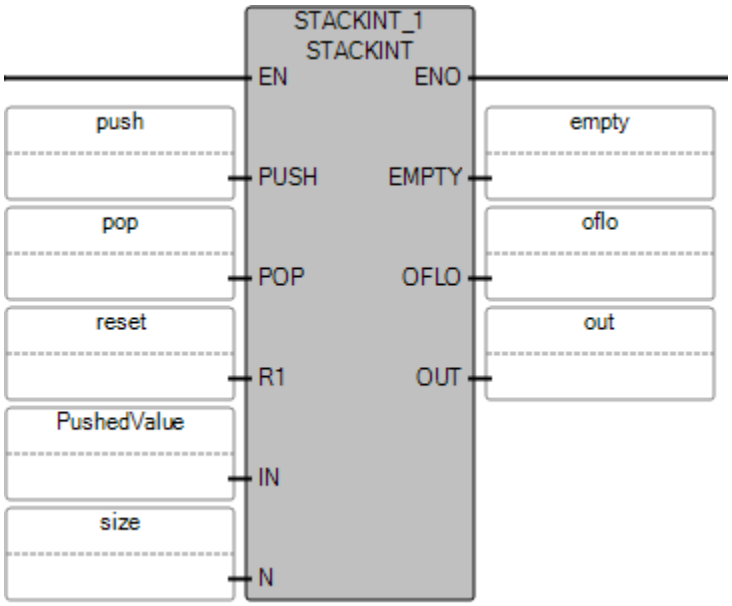
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
PUSH	Eingang	BOOL	TRUE – Anstiegflanke entdeckt bei PUSH-Befehl. Fügt den Wert IN oben auf den Stapel hinzu. FALSE – Anstiegflanke nicht erkannt bei PUSH-Befehl
POP	Eingang	BOOL	TRUE – Anstiegflanke entdeckt bei PUSH-Befehl. Löscht den letzten Wert, der oben auf dem Stapel hinzugefügt wurde. FALSE – Anstiegflanke nicht erkannt bei POP-Befehl.
R1	Eingang	BOOL	TRUE – Setzt den Stapel auf seinen Leerzustand zurück. FALSE – Kein Zurücksetzen.
IN	Eingang	DINT	Per Push-Befehl gespeicherter Wert.
N	Eingang	DINT	Anwendungsdefinierte Stapelgröße. Die Maximalgröße des Stapels ist 128
EMPTY	Ausgang	BOOL	TRUE – Der Stapel ist leer. FALSE – Der Stapel enthält Werte.
OFLO	Ausgang	BOOL	TRUE – Stapelüberlauf. Der Stapel ist voll, und R1 wurde mindestens einmal auf TRUE und wieder zurück auf FALSE gesetzt. FALSE – Die Stapelgröße ist 128 oder weniger. Kein Überlauf.
OUT	Ausgang	DINT	Oberster Wert auf dem Stapel. OUT ist gleich 0, wenn für OFLO der Wert TRUE zurückgegeben wird.

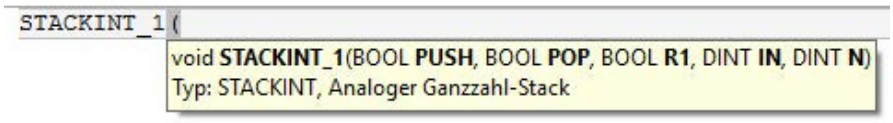
Beispiel für ein STACKINT Funktionsblockdiagramm



Beispiel für einen STACKINT-Kontaktplan



Beispiel für einen strukturierten STACKINT-Text



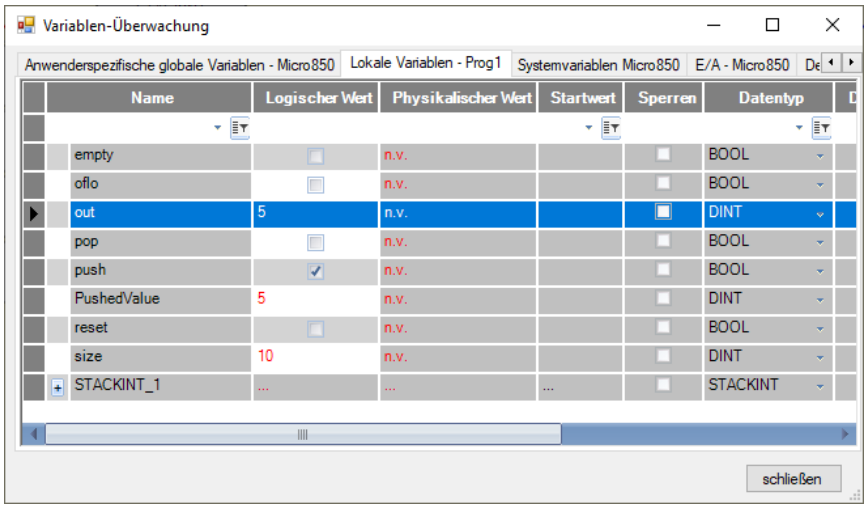

```
1  PushedValue := 5;
2  size := 10;
3  STACKINT_1(push, pop, reset, PushedValue, size);
4  empty := STACKINT_1.EMPTY;
5  oflo := STACKINT_1.OFLO;
6  out := STACKINT_1.OUT;
```

(* ST-Äquivalenz: STACKINT1 ist eine Instanz eines STACKINT-Blocks *)

```
STACKINT1(err_detect, acknowledge, manual_mode,
err_code, max_err);
```

```
appli_alarm := auto_mode AND NOT(STACKINT1.EMPTY);
err_alarm := STACKINT1.OFLO;
last_error := STACKINT1.OUT;
```

Ergebnisse

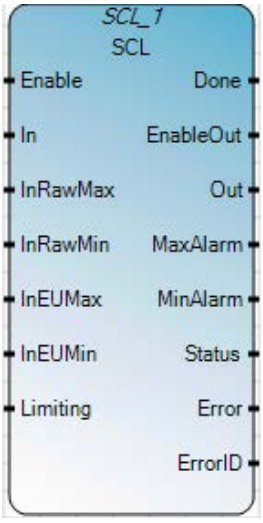


SCL

Rechnet einen nicht skalierten REAL-Eingabewert in einen REAL-Gleitkommawert in technischen Einheiten und um bezieht Alarmmeldung und Begrenzung der Ausgabe ein.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	TRUE - Anstiegflanke erkannt. • Wenn InRawMin >= InRawMax, werden Status.0- und Status.1- Bits auf 1 festgelegt.

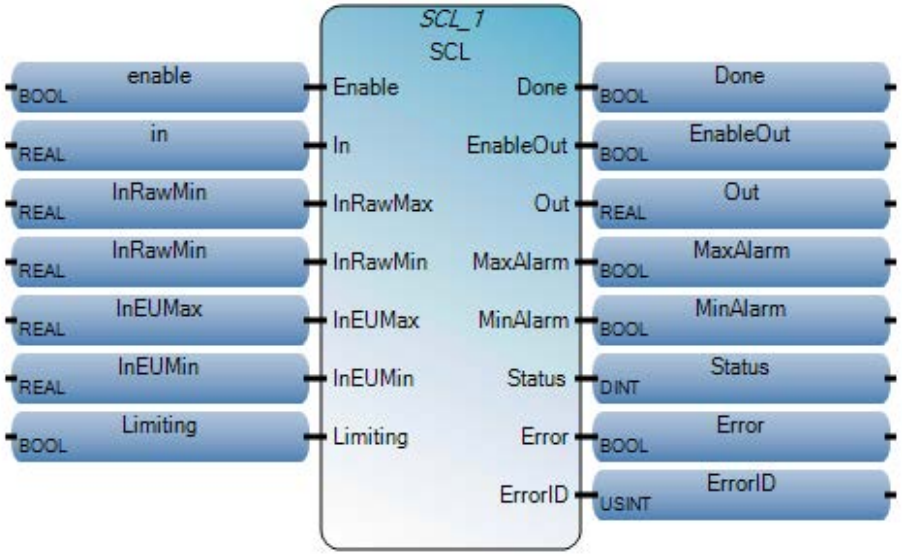
Parameter	Parametertyp	Datentyp	Beschreibung
			Done, MaxAlarm und MinAlarm werden gelöscht. Error wird auf TRUE festgelegt und ErrorID wird auf 1 festgelegt. • Else Out wird als Erstes berechnet. Dann werden die Alarmbedingungen überprüft. Wenn MaxAlarm festgelegt wurde, wird MinAlarm gelöscht und umgekehrt. Wenn die Begrenzung festgelegt wurde, befindet sich Out im Bereich von InEUMin und InEUMax. Dann wird das Done-Bit auf TRUE gesetzt. Die Status-Bits werden auf 0 festgelegt. Error und ErrorID werden auf 0 festgelegt. • Der berechnete Out-Wert wird mit NAN (Not a Number) verglichen. Entspricht NAN dem Out-Wert, wird EnableOut gelöscht. Done wird gelöscht, wenn der Fehler-Zustand festgesetzt wurde. FALSE – keine Anstiegflanke erkannt. • Die Anweisung wird nicht ausgeführt. • Die Ausgänge werden nicht aktualisiert. Es werden nur ErrorID, EnableOut und Done auf 0 festgelegt.
In	Eingang	REAL	Die analoge Signaleingabe. Gültig = jeder Gleitkommawert Standard = 0,0
InRawMax	Eingang	REAL	Der maximal zu erreichende Wert durch die Eingabe bei der Anweisung. Ist InRawMax <= InRawMin, legt die Anweisung das entsprechende Bit im Status und Error fest. ErrorID werden aktualisiert. Die Out-Aktualisierung wird abgebrochen. Gültig = InRawMax > InRawMin Standard = 0,0
InRawMin	Eingang	REAL	Der minimal zu erreichende Wert durch die Eingabe bei der Anweisung. Ist InRawMin >= InRawMax, legt die Anweisung das entsprechende Bit im Status und Error fest. ErrorID werden aktualisiert. Die Out-Aktualisierung wird abgebrochen. Gültig = InRawMin < InRawMax Standard = 0,0
InEUMax	Eingang	REAL	Der skalierte Wert der Eingabe entsprechend InRawMax. Gültig = jeder reale Wert Standard = 0,0
InEUMin	Eingang	REAL	Der skalierte Wert der Eingabe entsprechend InRawMin. Gültig = jeder reale Wert Standard = 0,0
Limiting	Eingang	BOOL	Begrenzungsauswahl. TRUE – Out wird auf zwischen InEUMin und InEUMax beschränkt.
Done	Ausgang	BOOL	Gibt an, wenn der Vorgang abgeschlossen ist. TRUE - Der Vorgang wurde erfolgreich abgeschlossen. FALSE – Beim Vorgang ist ein Fehler aufgetreten oder Enable wurde auf FALSE festgelegt.
EnableOut	Ausgang	BOOL	Gibt an, wenn die Anweisung aktiviert ist. Wird auf FALSE festgelegt, wenn Out überläuft.
Out	Ausgang	REAL	Steht für den skalierten Wert der Analogeingabe.
MaxAlarm	Ausgang	BOOL	Die maximale Eingabealarmanzeige. Dieser Wert wird auf TRUE gesetzt, wenn Eingabe > InRawMax.
MinAlarm	Ausgang	BOOL	Die minimale Eingabealarmanzeige. Dieser Wert wird auf TRUE gesetzt, wenn Eingabe < InRawMin.
Status	Ausgang	DINT	Status des Funktionsblocks. InstructFault (Status.0) Wird festgelegt, wenn Ausführungsfehler der Anweisung erkannt werden. Dabei handelt es sich nicht um den geringfügigen oder schweren Controllerfehler. Überprüfen Sie die verbleibenden Status-Bits, um festzustellen, was passiert ist. InRawRangeInv (Status.1)

Parameter	Parametertyp	Datentyp	Beschreibung
			InRawMin >= InRawMax Status.3 bis Status.31 sind für den zukünftigen Gebrauch vorbehalten und ihr Wert beträgt 0.
Error	Ausgang	BOOL	Zeigt das Vorhandensein eine Fehlerbedingung an. TRUE – Beim Vorgang ist ein Fehler aufgetreten. FALSCH – Vorgang wurde erfolgreich abgeschlossen oder der Befehl wird nicht ausgeführt.
ErrorID	Ausgang	BOOL	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler sind in den Fehlercodes definiert.

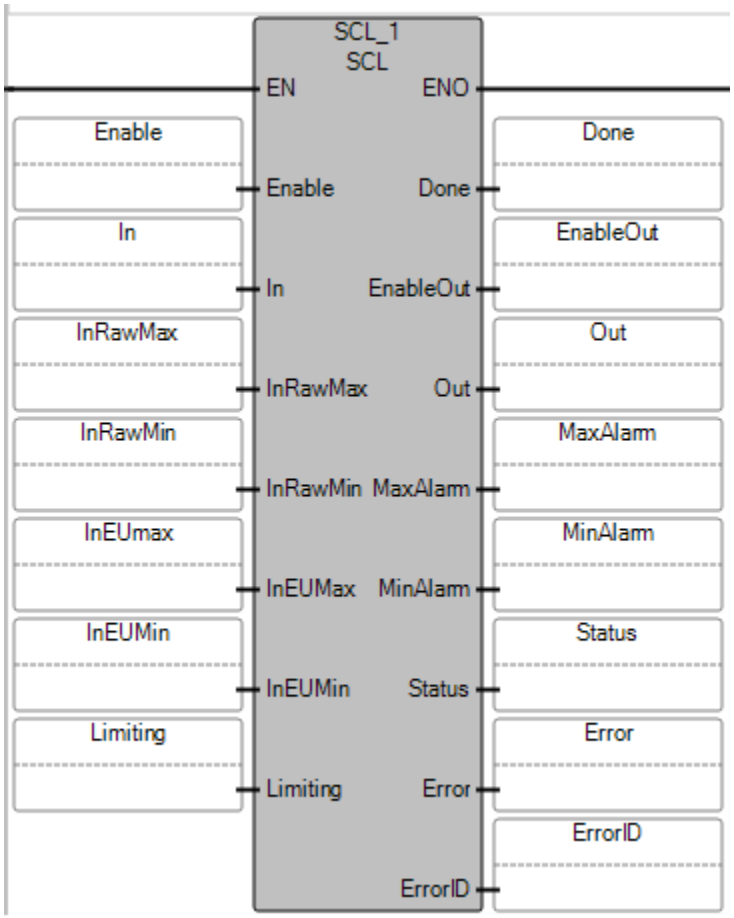
Fehlercode

ErrorID-Code	Fehlerbeschreibung
1	InRawMax<= InRawMin

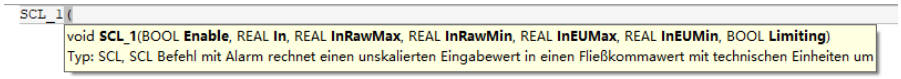
Beispiel für das SCL-Funktionsblockdiagramm



Beispiel für SCL-Kontaktpläne



Beispiel für SCL in strukturiertem Text

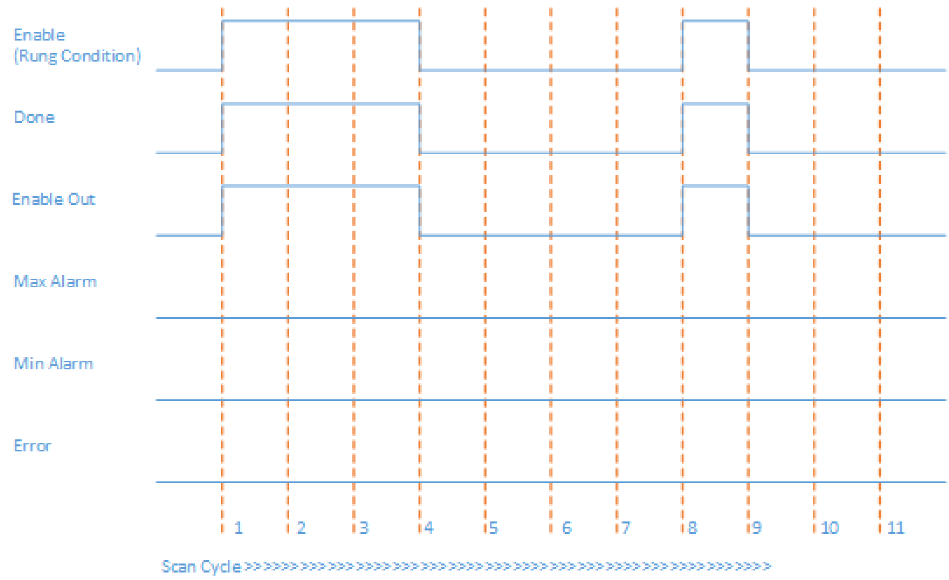



```
1  SCL_1(Enable, In, InRawMax, InRawMin, InEUMax, InEUMin, Limiting);
2  Done :=SCL_1.Done;
3  EnableOut :=SCL_1.EnableOut;
4  Out :=SCL_1.Out;
5  MaxAlarm :=SCL_1.MaxAlarm;
6  MinAlarm :=SCL_1.MinAlarm;
7  Status :=SCL_1.Status;
8  Error :=SCL_1.Error;
9  ErrorID :=SCL_1.ErrorID;
```

Beispiele für Timingdiagramme von SCL-Anweisungen

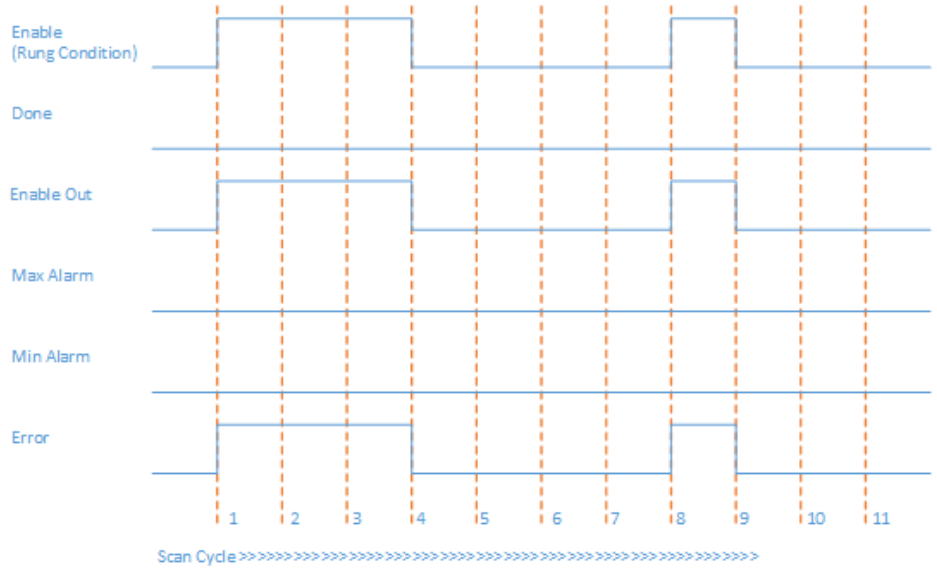
In den folgenden Beispielen zu Zeitdiagrammen werden Ausführungsszenarien für den SCL auf seite 509-Befehl beschrieben.

Erfolgreiche SCL-Ausführung



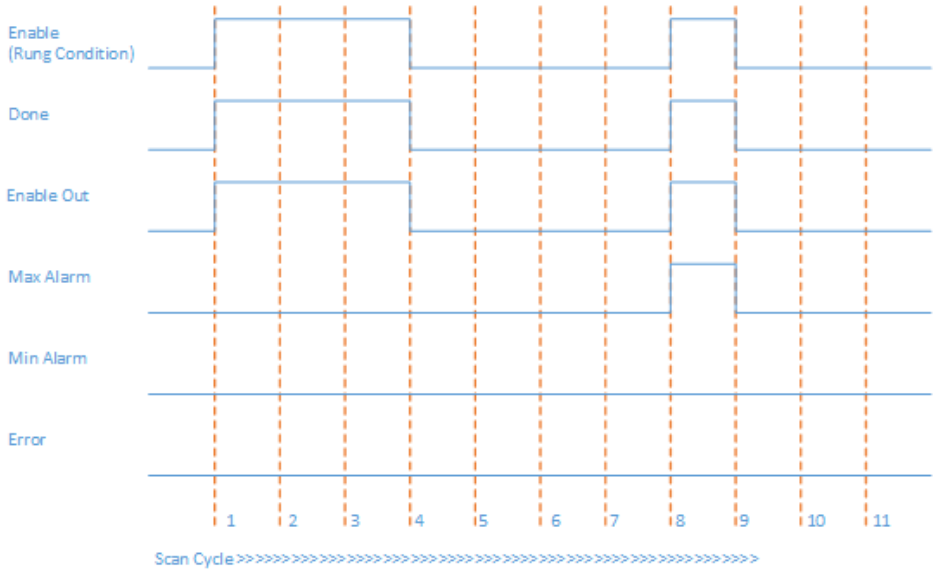
Scan-Zyklus	Beschreibung
1	Wenn Enable auf TRUE festgelegt ist und die Eingangsparameter gültig sind und sich innerhalb des Bereichs befinden, beginnt der Funktionsblock mit der Ausführung. • Done und EnableOut werden auf TRUE festgelegt. • Out wird entsprechend der vorhandenen Eingaben berechnet. • MaxAlarm, MinAlarm und Error werden auf FALSE festgelegt. • Status und ErrorID werden auf 0 festgelegt, da kein Fehler generiert wird.
2, 3	Keine Änderung in Strompfadbedingung.
4	Wenn Enable auf FALSE festgelegt ist, wird die Ausführung des Funktionsblocks gestoppt. • Done, EnableOut und Error werden auf FALSE festgelegt. • ErrorID wird auf 0 festgelegt. • Out, MaxAlarm, MinAlarm und Status behalten ihren letzten Wert.
5, 6, 7	Keine Änderung in Strompfadbedingung.
8	• Wenn Enable auf TRUE festgelegt ist und die Eingangsparameter gültig sind und sich innerhalb des Bereichs befinden, beginnt der Funktionsblock mit der Ausführung. • Done und EnableOut werden auf TRUE festgelegt. • Out wird entsprechend der vorhandenen Eingaben berechnet. • MaxAlarm, MinAlarm und Error werden auf FALSE festgelegt. • ErrorID und Status werden auf 0 festgelegt, da kein Fehler generiert wird.
9	Wenn Enable auf FALSE festgelegt ist, wird die Ausführung des Funktionsblocks gestoppt. • Done, EnableOut und Error werden auf FALSE festgelegt. • ErrorID wird auf 0 festgelegt. • Out, MaxAlarm, MinAlarm und Status behalten ihren letzten Wert.
10, 11	Keine Änderung in Strompfadbedingung.

Fehlgeschlagene SCL-Ausführung



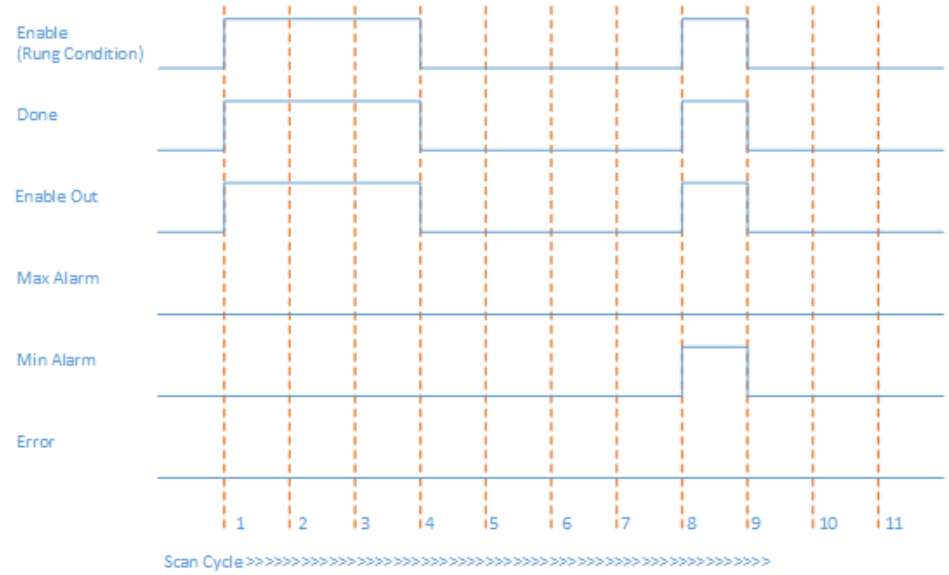
In diesem Beispiel sind alle Eingangsparameter gültig und innerhalb des Bereichs, aber InRawMin >= InRawMax. In den Scanzyklen 1 und 8 wird Error auf True festgelegt und ErrorID auf 1, wenn Enable auf TRUE festgelegt wurde und der Funktionsblock ausgeführt wird.

Generierung von MaxAlarm



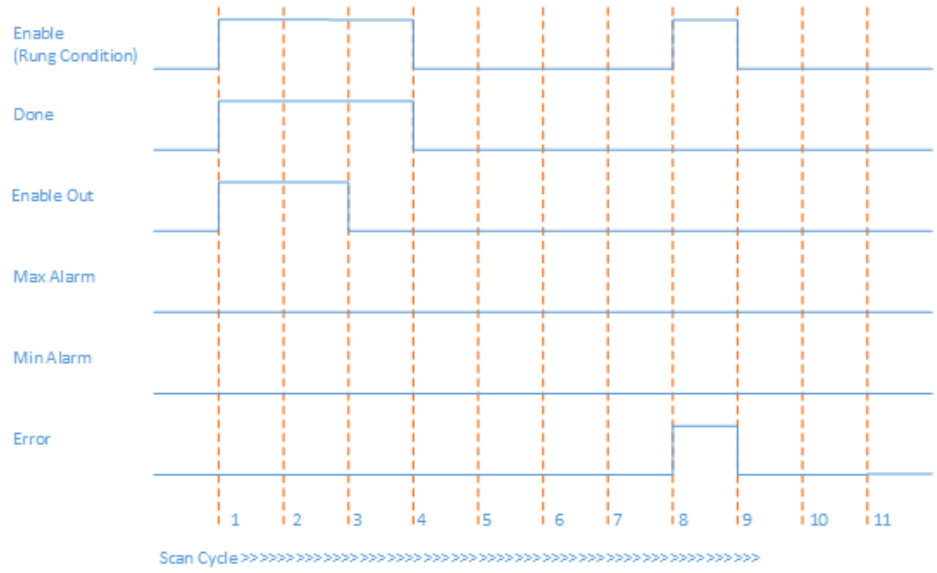
In Scanzyklus 8 in diesem Beispiel sind alle Eingangsparameter gültig und innerhalb des Bereichs, aber In > InRawMax. Wenn Enable auf TRUE festgelegt wurde und der Funktionsblock mit der Ausführung beginnt, wird MaxAlarm auf TRUE festgelegt.

Generierung von MinAlarm



In Scanzyklus 8 in diesem Beispiel sind alle Eingangsparameter gültig und innerhalb des Bereichs, aber In < InRawMin. Wenn Enable auf TRUE festgelegt wurde und der Funktionsblock mit der Ausführung beginnt, wird MinAlarm auf TRUE festgelegt.

Ausgabeüberlauf-Bedingung und Fehler bei der Eingangskonfiguration



In Scanzyklus 3 in diesem Beispiel sind die Eingangsparameter für den Funktionsblock gültig und innerhalb des Bereichs, aber Out wird aufgrund der Eingabeparameter überlaufen. EnableOut wird auf FALSE festgelegt. Der Out-Wert ist gültig.

In Scanzyklus 8 in diesem Beispiel sind die Eingangsparameter für den Funktionsblock gültig und innerhalb des Bereichs, aber Out wird aufgrund der Eingabeparameter überlaufen und InRawMin >= InRawMax, Error wurde auf TRUE festgelegt. Error wird auf 1 und Status auf 3 festgelegt.

TND (aktuelles Programm wird angehalten)

Hält den aktuellen Zyklus des Benutzerprogramm-Scans an. Nach Abschluss von Ausgangsscan, Eingangsscan und Variablenverwaltung wird das Benutzerprogramm ab dem Anfang der ersten Routine erneut ausgeführt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

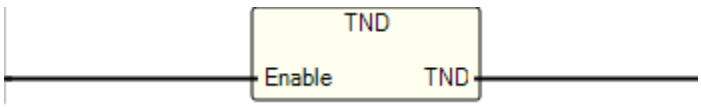
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

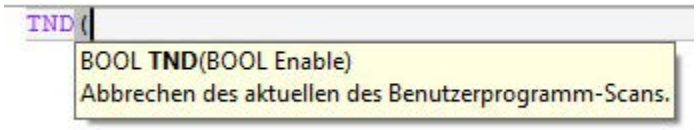
Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Funktion aktivieren. Wenn Enable = TRUE, Funktion ausführen. Wenn Enable = FALSE, Funktion nicht ausführen.
TND	Ausgang	BOOL	Wenn TRUE, Funktion ausgeführt. Wenn die Variablenüberwachung aktiviert ist, wird der Wert der Überwachungsvariable dem Ausgang der Anweisung zugewiesen. Wenn die Variablenüberwachung deaktiviert ist, wird der Wert der Ausgangsvariable dem Ausgang der Anweisung zugewiesen.

Beispiel für ein TND-Funktionsblockdiagramm

Beispiel für einen TND-Kontaktplan



Beispiel für strukturierten TND-Text

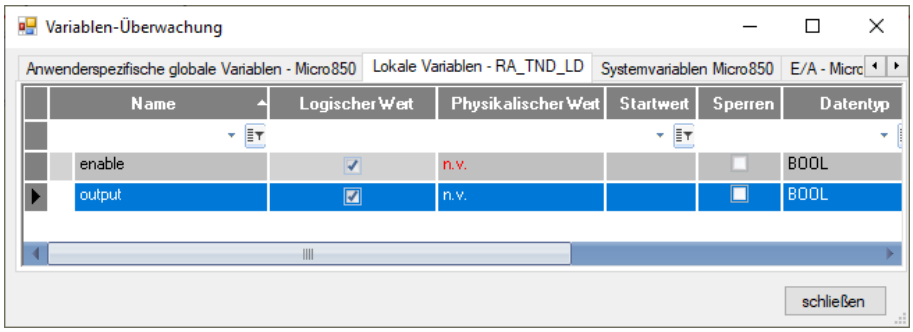



```
1  enable := TRUE;
2  output := TND(enable);
```

(* ST-Äquivalenz: *)

TESTOUTPUT := TND(TESTENABLE) ;

Ergebnisse



LIMIT (Grenzwerttest)

Beschränkt ganzzahlige Werte auf ein gegebenes Intervall. Ganzzahlige Werte zwischen Minimal- und Maximalwert bleiben unverändert. Ganzzahlige Werte über dem Maximalwert werden durch den Maximalwert ersetzt. Ganzzahlige Werte unter dem Minimalwert werden durch den Minimalwert ersetzt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

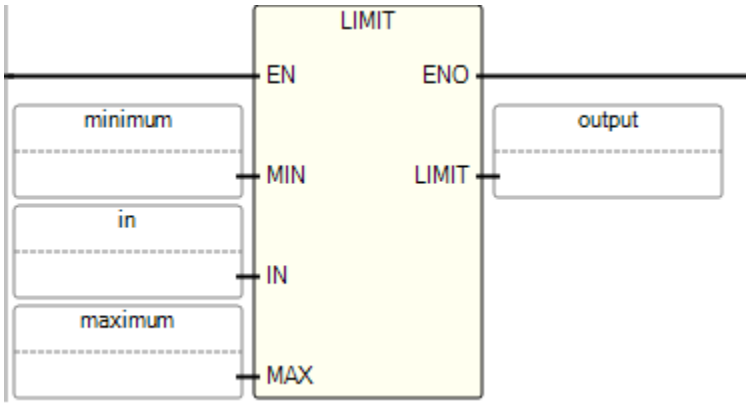
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Funktion aktivieren. TRUE – aktuelle LIMIT-Berechnung ausführen. FALSE – es findet keine Berechnung statt.

Parameter	Parametertyp	Datentyp	Beschreibung
			Gilt für Kontaktplan-Programme.
MIN	Eingang	DINT	Unterstützter Mindestwert.
IN	Eingang	DINT	Beliebiger ganzzahliger Wert mit Vorzeichen.
MAX	Eingang	DINT	Unterstützter Maximalwert.
LIMIT	Ausgang	DINT	Eingangswert, beschränkt auf den unterstützten Bereich.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Diagrammbeispiel für Funktionsblock LIMIT

LIMIT-Kontaktplan-Beispiel



Beispiel für LIMIT in strukturiertem Text

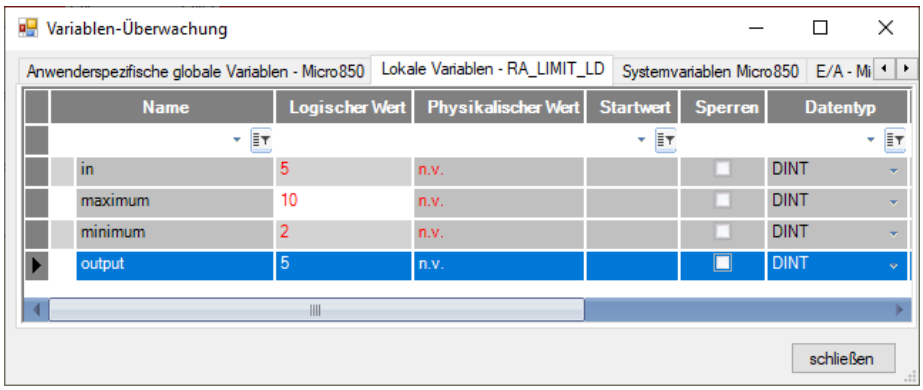
```
1  minimum := 2;
2  in := 5;
3  maximum := 10;
4  output := LIMIT(minimum, in, maximum);
```

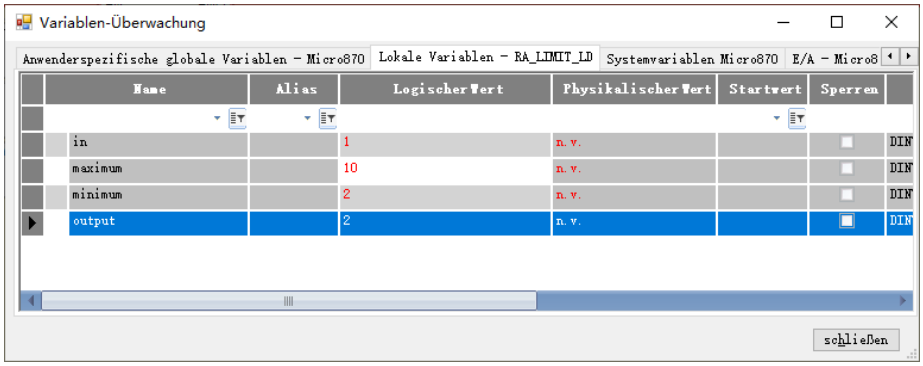
(* ST-Äquivalenz: *)

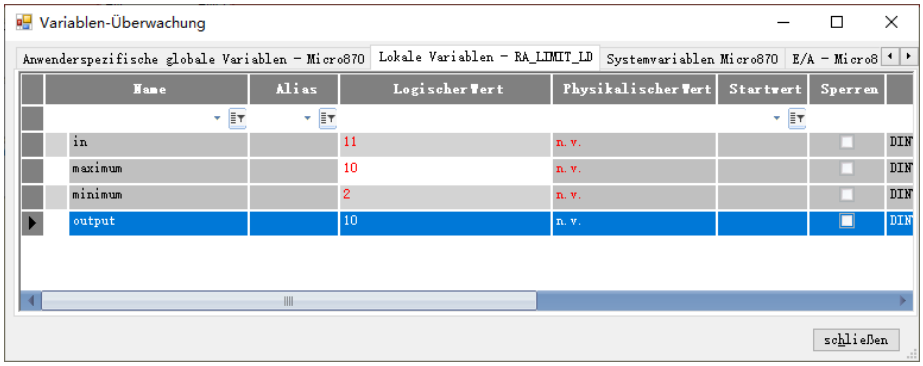
new_value := LIMIT (min_value, value, max_value);

(* beschränkt den Wert auf den gegebenen Bereich [min_value..max_value] *)

Ergebnisse







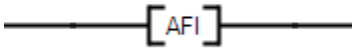
Anweisungen zur Programmsteuerung

Verwenden Sie die Anweisung zur Programmsteuerung, um Anweisungen gleichzeitig aus einem Benutzerprogramm und über ein Bedienerschnittstellengerät zu steuern.

Anweisung	Beschreibung
AFI auf seite 521	Deaktiviert einen Strompfad.
NOP auf seite 521	Fungiert als Platzhalter.
SUS auf seite 521	Versetzt den Micro800-Controller in den Ruhezustand.

AFI (Always False)

Verwenden Sie die AFI-Anweisung am Anfangspunkt eines Strompfads, um diesen beim Debuggen vorübergehend zu deaktivieren. So muss der Strompfad nicht aus dem Programm gelöscht werden. Der Ausgang für diese Anweisung ist immer FALSE.

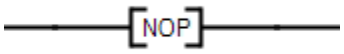


Unterstützte Sprache: Kontaktplan.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

NOP (No Operation)

Die NOP-Anweisung fungiert als Platzhalter. Sie können die NOP-Anweisung überall auf einem Strompfad platzieren.



Unterstützte Sprache: Kontaktplan.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

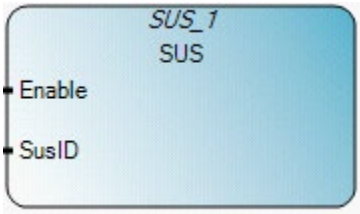
SUS (Ruhezustand)

Versetzt den Micro800-Controller in den Ruhezustand. Der Controller bleibt im Betriebsmodus, die Ausführung wird jedoch für unbestimmte Zeit ausgesetzt. Im Ruhezustand werden Fehler im Benutzerprogramm erfasst, und der Ruhezustand hilft beim Überwachen des Benutzerprogramms. Platzieren Sie die SUS-Anweisung im Benutzerprogramm in Abschnitten, für die Sie ungewöhnliche Zustände erfassen möchten. Im Ruhezustandsmodus

ist „RUN LED“ auf „OFF“ gesetzt, um anzugeben, dass sich der Programm-Scan im Leerlauf befindet.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

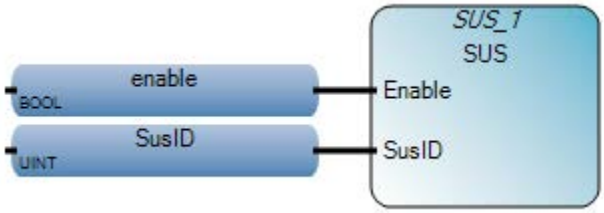
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



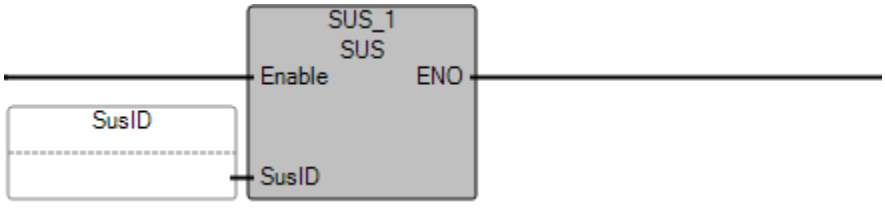
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Funktion ausführen. FALSE – Funktion nicht ausführen.
SusID	Eingang	UINT	Suspensions-ID.
ENO	Ausgang	BOOL	Ausgabe aktivieren. Gilt nur für Kontaktplan-Programme.

Beispiel für ein SUS-Funktionsblockdiagramm



Beispiel für einen SUS-Kontaktplan



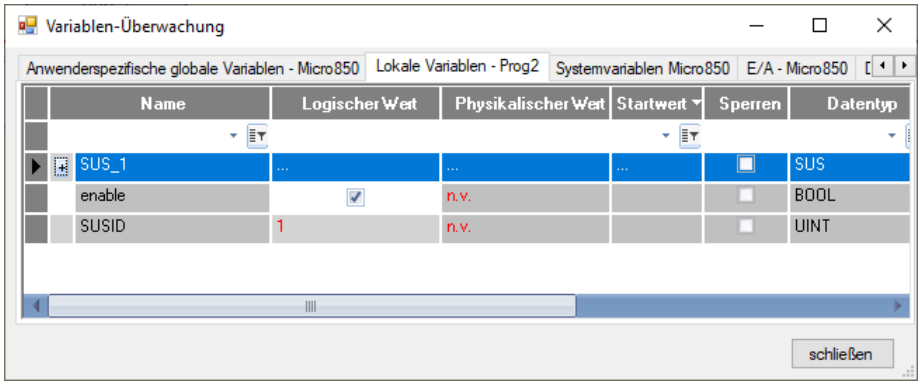
Beispiel für einen strukturierten SUS-Text

```
SUS_1(
```

void **SUS_1**(BOOL **Enable**, UINT **SusID**)
Typ: SUS, Versetzen der Anwendungsausführung in den Ruhezustand.

```
1  SusID := 1;
2  SUS_1(enable, SusID);
```

Ergebnisse



Anweisung „Proportional-Integral-Differential“ (PID)

Verwenden Sie die Anweisung „Proportional-Integral-Differential“ (PID) zur genaueren Steuerung des Prozesses mithilfe einer PID-Funktion.

Anweisung	Beschreibung
IPIDCONTROLLER auf seite 525	Konfiguriert und steuert die Eingänge und Ausgänge für die Proportional-Integral-Differential (PID)-Logik.
PID auf seite 550	Konfiguriert und steuert die Ausgänge, welche die physischen Eigenschaften, wie etwa Temperatur, Druck, Füllstand oder Durchflussrate, mithilfe von Prozessschleifen steuern.

Was ist eine „Proportional-Integral-Differential“-Steuerung (PID)?

Eine „Proportional-Integral-Differential“-Steuerung (PID) ermöglicht der Prozesssteuerung die präzise Einhaltung des Sollwerts, indem Steuerungsausgaben angepasst werden. Ein „PID“-Funktionsblock kombiniert alle erforderlichen Logiken, um eine proportionale/integrale/differentielle Steuerung (PID) auszuführen.

IPIDCONTROLLER (proportionaler/integraler/differentieller Controller)

Konfigurieren und steuern Sie die Eingänge und Ausgänge für die proportionale/integrale/differentielle Logik (PID). Die PID-Logik wird zur Steuerung des physischen Eigenschaften wie Temperatur, Druck, Flüssigkeit, Ebene oder Durchflussrate mithilfe von Prozessschleifen verwendet, die einen Fehlerwert als Differenz zwischen einem gewünschten Sollwert und einer gemessenen Prozessvariablen berechnen. Der Controller versucht, den Fehler mit der Zeit durch Anpassung einer Steuervariablen zu minimieren. Die Berechnung umfasst proportionale (P), integrale (I) und differentielle (D) Faktoren, die wie folgt verwendet werden:

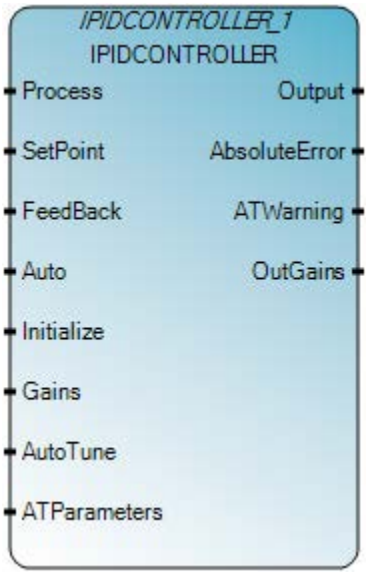
- P – vorhandene Werte des Fehlers.

- I – vergangene Werte des Fehlers.

- D – mögliche zukünftige Werte des Fehlers, basierend auf der aktuellen Änderungsrate, die physische Eigenschaften wie Temperatur, Druck, Füllstand oder Durchflussrate mithilfe von Prozessschleifen steuert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

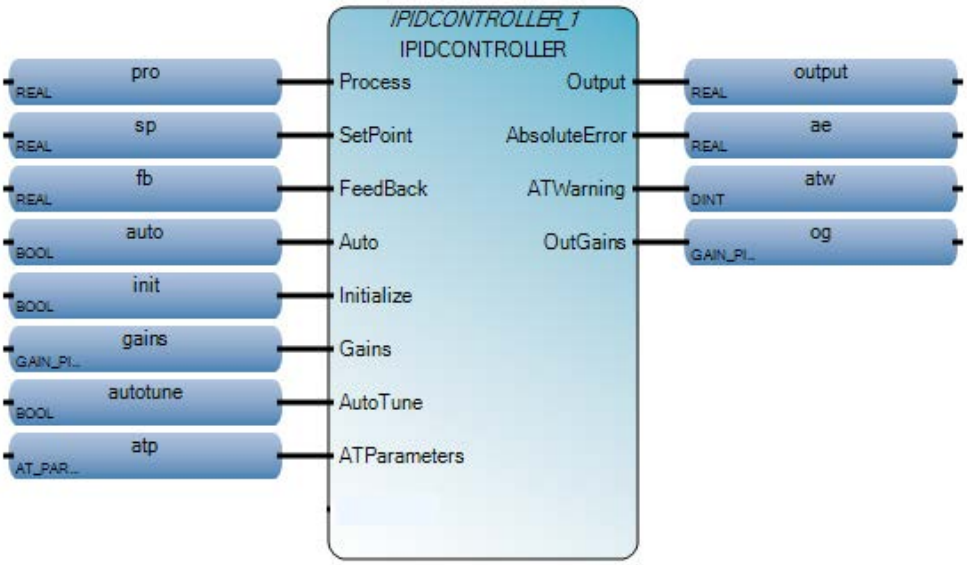


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

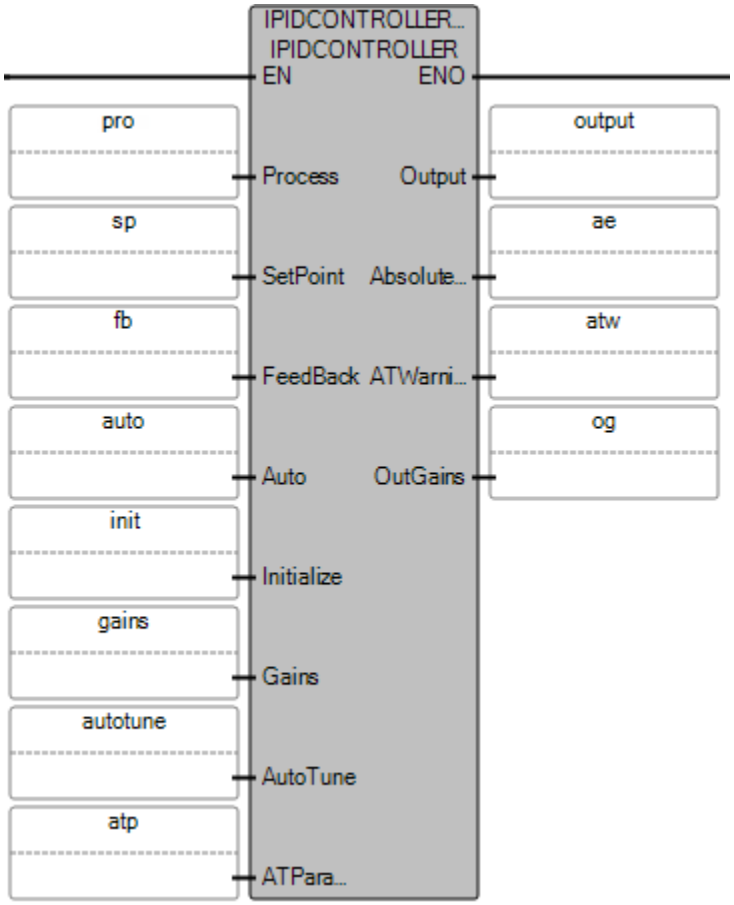
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Wenn TRUE, ist der Anweisungsblock aktiviert. TRUE – PID-Berechnung ausführen. FALSE - der Anweisungsblock befindet sich im Leerlauf. Gilt für Kontaktplan-Programme.
Process	Eingang	REAL	Prozesswert, d. h. der Wert, der an der Prozessausgabe gemessen wird.
SetPoint	Eingang	REAL	Sollwert.
Feedback	Eingang	REAL	Feedbacksignal, d. h. der Wert der auf den Prozess angewandten Steuervariablen. Bei dem Feedback kann es sich beispielsweise um die Ausgabe von IPIDCONTROLLER handeln.
Auto	Eingang	BOOL	Der Betriebsmodus des PID-Controllers: • TRUE - der Controller wird im Normalmodus ausgeführt. • FALSE - der Controller verursacht ein Zurücksetzen von R zur Verfolgung (F-GE).
Initialize	Eingang	BOOL	Eine Wertänderung (TRUE zu FALSE oder FALSE zu TRUE) bewirkt, dass die Steuerung jeden proportionalen Anstieg während des Zyklus beseitigt. Initialisiert außerdem AutoTune-Sequenzen.
Gains	Eingang	GAIN_PID	PID des Anstiegs für IPIDController. Mit dem Datentyp GAIN_PID auf seite 529 können Sie die Parameter für den Anstiegseingang definieren.
AutoTune	Eingang	BOOL	TRUE – Wenn der Wert für AutoTune auf TRUE und für Auto und Initialize auf FALSE eingestellt ist, wird die AutoTune-Sequenz gestartet. FALSE –AutoTune wird nicht gestartet.
ATParameters	Eingang	AT_Param	AutoTune-Parameter. Mit dem Datentyp AT_Param auf seite 530 können Sie die Parameter für den ATParameters-Eingang definieren.
Output	Ausgang	REAL	Ausgabewert vom Controller.
AbsoluteError	Ausgang	REAL	Absoluter Fehler („Process“ – „SetPoint“) vom Controller.

Parameter	Parametertyp	Datentyp	Beschreibung
ATWarnings	Ausgang	DINT	(ATWarning) Warnung für die AutoTune-Sequenz. Mögliche Werte sind: • 0 - es erfolgt keine automatische Abstimmung. • 1 – im Selbsteinstellungsmodus. • 2 - es erfolgt eine Selbsteinstellung. • -1 - FEHLER 1 Eingabe automatisch auf TRUE gesetzt, keine Selbsteinstellung möglich. • -2 - FEHLER 2 Selbsteinstellungsfehler; „ATDynaSet“ abgelaufen.
OutGains	Ausgang	GAIN_PID	Berechneter Anstieg nach AutoTune-Sequenz. Verwenden Sie den Datentyp GAIN_PID, um die OutGains-Ausgabe zu definieren.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

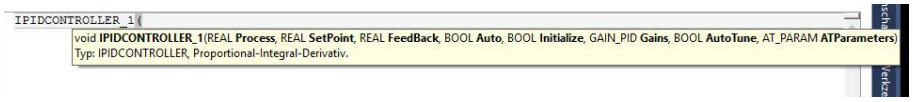
Diagrammbeispiel für Funktionsblock IPIDCONTROLLER



IPIDCONTROLLER-Kontaktplan-Beispiel



Beispiel für IPIDCONTROLLER in strukturiertem Text

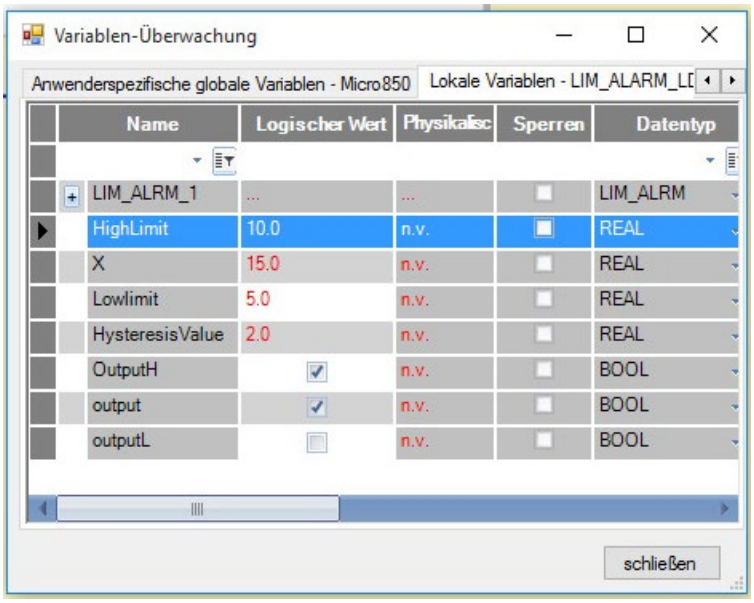



```
1  IPIDCONTROLLER_1(pro, sp, fb, auto, init, gains, autotune, atp, em);
2  output := IPIDCONTROLLER_1.Output;
3  ae := IPIDCONTROLLER_1.AbsoluteError;
4  atw := IPIDCONTROLLER_1.ATWarning;
5  og := IPIDCONTROLLER_1.OutGains;
```

```
(* ST-Äquivalenz: IPIDController1 ist eine Instanz des
Blocks IPIDController. *)

IPIDController1(Proc,
SP,
 FBK,
 Auto,
 Init,
 G_In,
 A_Tune,
 A_TunePar,
 Err );
 Out_process := IPIDController1.Output ;
A_Tune_Warn := IPIDController1.ATWarning ;
```

```
Gain_Out := IPIDController1.OutGains ;
```

Ergebnisse



Datentyp GAIN_PID

In der folgenden Tabelle wird der Datentyp „GAIN_PID“ für den Befehl IPIDCONTROLLER auf seite 525 beschrieben.

Parameter	Datentyp	Beschreibung
DirectActing	BOOL	Die Art der Aktion: • TRUE - direkte Aktion (Ausgabe bewegt sich in die gleiche Richtung wie der Fehler). Das heißt, der tatsächliche Prozesswert ist höher als der „SetPoint“ (Sollwert) und die entsprechende Aktion des Controllers besteht im Erhöhen der Ausgabe (z. B. Kühlanwendungen). • FALSE - umgekehrte Aktion (Ausgabe bewegt sich entgegengesetzt zum Fehler). Das heißt, der tatsächliche Prozesswert ist höher als der „SetPoint“ (Sollwert) und die entsprechende Aktion des Controllers besteht im Verringern der Ausgabe (z. B. Heizung).
ProportionalGain	REAL	Proportionaler Anstieg für PID (>= 0,0001). Wenn ProportionalGain (< 0,0001) ist, dann ist ProportionalGain = 0,0001 **Proportionaler Anstieg für PID (P_Gain)** Ein höherer proportionaler Anstieg führt zu größeren Änderungen in der Ausgabe auf Basis des Unterschieds zwischen dem PV (gemessener Prozesswert) und dem SV (Sollwert). Je höher der Anstieg, desto schneller verringert sich der Fehler. Dadurch können jedoch Instabilitäten wie Oszillationen entstehen. Je geringer der Anstieg, desto langsamer verringert sich der Fehler. Das System ist dann jedoch stabiler und weniger anfällig für große Fehler. Der P_Gain ist normalerweise der wichtigste Anstieg und der erste Anstieg, der während der Abstimmung angepasst wird.
TimeIntegral	REAL	Der Zeitintegralwert für PDI in Sekunden (>= 0,0001). Wenn TimeIntegral (< 0,0001) ist, dann ist TimeIntegral = 0,0001 **Zeitintegralwert für PID** Eine niedrigere integrale Zeitkonstante führt zu schnelleren Änderungen in der Ausgabe auf Basis des Unterschieds zwischen dem über diesen Zeitraum integrierten PV (gemessener Prozesswert) und SV (Sollwert). Eine niedrigere integrale Zeitkonstante verringert den Steady-State-Fehler (Fehler, der auftritt, wenn der SV nicht geändert wird). Gleichzeitig erhöht sie aber die Wahrscheinlichkeit von Instabilitäten wie Oszillationen. Eine höhere integrale Zeitkonstante verlangsamt die Reaktionsfähigkeit des Systems und macht es stabiler. Der PV nähert sich dem SV dann jedoch langsamer an.
TimeDerivative	REAL	Der abgeleitete Zeitwert für PDI in Sekunden (> 0,0). Wenn TimeDerivative (< 0,0) ist, dann ist TimeDerivative = 0,0 Wenn von TimeDerivative = 0, agiert IPID als PI.

Parameter	Datentyp	Beschreibung
		Abgeleiteter Zeitwert für PID (Td) Ein niedrigerer abgeleiteter Zeitwert führt zu schnelleren Änderungen in der Ausgabe auf Basis des Unterschieds zwischen dem PV (gemessener Prozesswert) und dem SV (Sollwert). Bei einem niedrigeren abgeleiteten Zeitwert kann ein System schneller auf plötzliche Fehleränderungen reagieren (SV ändert sich). Gleichzeitig steigt jedoch die Wahrscheinlichkeit von Instabilitäten wie Oszillationen. Bei einem höheren abgeleiteten Zeitwert reagiert ein System weniger schnell auf plötzliche Fehleränderungen und ist weniger anfällig für Störungen und Schrittwechsel im PV. Der Wert „TimeDerivative“ (TD) steht in Beziehung zum abgeleiteten Anstieg, erlaubt aber die Abstimmung des Ableitungsbeitrags zu PID anhand der Zeit. Daher muss die Abtastzeit berücksichtigt werden.
DerivativeGain	REAL	Abgeleiteter Anstieg für PID (> 0,0). Wenn DerivativeGain (< 0,0) ist, dann ist DerivativeGain = 0,1 **Abgeleiteter Anstieg für PID (D_Gain)** Ein höherer abgeleiteter Anstieg führt zu größeren Änderungen in der Ausgabe auf Basis der Änderungsrate des Unterschieds zwischen dem PV (gemessener Prozesswert) und dem SV (Sollwert). Bei einem höheren Anstieg kann ein System schneller auf plötzliche Fehleränderungen reagieren. Gleichzeitig steigt jedoch die Wahrscheinlichkeit von Instabilitäten wie Oszillationen. Bei einem niedrigeren Anstieg reagiert ein System weniger schnell auf plötzliche Fehleränderungen und ist weniger anfällig für Störungen und Schrittwechsel im PV.

Datentyp AT_Param

In der folgenden Tabelle wird der Datentyp „AT_Param“ beschrieben.

Parameter	Datentyp	Beschreibung
Load	REAL	Ladeparameter für die automatische Abstimmung. Dies ist der Ausgabewert beim Starten von AutoTune.
Deviation	REAL	Differenz für die automatische Abstimmung. Dies ist die Standarddifferenz, die zum Evaluieren des für AutoTune benötigten Rauschbands verwendet wird.
Step	REAL	Schrittwert für AutoTune. Muss größer sein als das Rauschband und kleiner als ½ Last („Load“).
ATDynamSet	REAL	Wartezeit in Sekunden, bevor die automatische Abstimmung verlassen wird.
ATReset	BOOL	Die Angabe, ob der Ausgabewert nach einer AutoTune-Sequenz auf Null zurückgesetzt wird: • TRUE - setzt die Ausgabe auf Null zurück. • FALSE - belässt den Ausgang auf dem „Laden“-Wert.

So implementiert der Funktionsblock IPIDController die PID-Steuerung

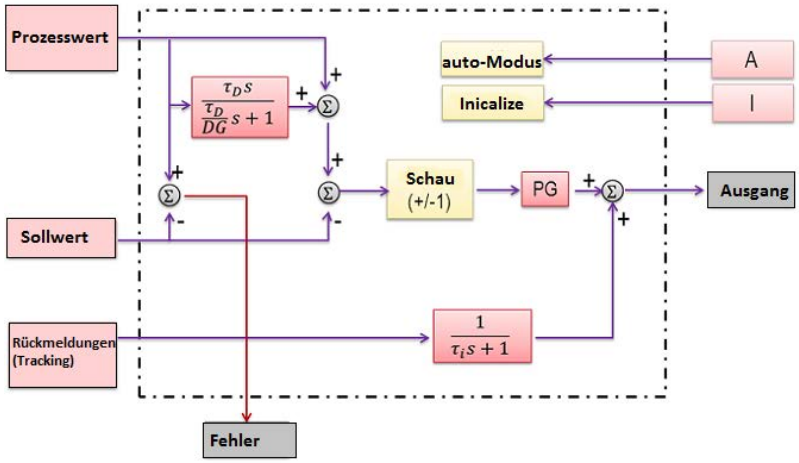
Der in der Anweisungsliste Connected Components Workbench verfügbare Funktionsblock IPIDController basiert auf der PID-Steuerungstheorie und kombiniert alle erforderlichen Logiken, um eine Verarbeitung des analogen Eingangskanals und eine proportionale/integrale/differentielle Steuerung (PID) auszuführen. Im HMI steht der IPID-Bildschirm für die Verwendung mit dem Funktionsblock IPIDController zur Verfügung.

Beschreibung des IPIDController-Funktionsblocks

Der IPIDController basiert auf folgenden Funktionsblockkomponenten:

- A: Aktion (+/- 1)
- PG: Proportionaler Anstieg (Proportional Gain)
- DG: Abgeleiteter Filteranstieg (Derivative Gain)
- td: ãD

- ti: ãI



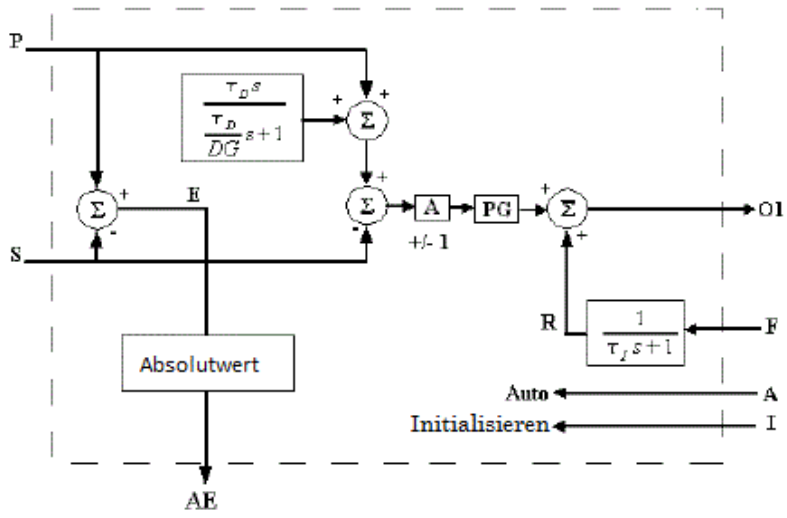
Vorbeugen vor einem inneren Aufschaukeln

Wenn die Differenz zwischen Sollwert und Prozesswert groß ist, erhöht sich der Ausgabewert erheblich, und während des Zeitraums, den er zum Absenken benötigt, kann der Prozess nicht gesteuert werden. Der „IPIDController“-Funktionsblock verfolgt interaktiv die Rückmeldung und beugt einem inneren Aufschaukeln vor. Wenn die Ausgabe gesättigt ist, wird die ganzzahlige Bedingung im Controller neu berechnet, sodass der dann neue Wert eine Ausgabe an der Sättigungsgrenze ermöglicht.

Vorgang des Funktionsblocks IPIDController

Wenn **Input Auto** den Wert **TRUE** besitzt, wird der IPIDController im normalen automatischen Modus ausgeführt.

Wenn **Input Auto** den Wert **FALSE** besitzt, verursacht dies ein Zurücksetzen von R zur Verfolgung (F-GE). Hierdurch wird die Ausgabe von IPIDController gezwungen, das Feedback innerhalb der IPIDController-Grenzwerte zu verfolgen, und der Controller schaltet ohne Erhöhung der Ausgabe zurück auf Automatik.



Wenn AutoTune den Wert FALSE besitzt, führt ein Wechsel von FALSE zu TRUE oder von TRUE zu FALSE beim Initialisieren der Eingabe („Input Initialize“) dazu, dass der IPIDController jegliche Aktion eines proportionalen Anstiegs während dieses Zyklus eliminiert (z. B. „Initialize“). Sie können diesen Prozess nutzen, um ein Anstoßen der Ausgabe zu vermeiden, wenn mittels eines Schalt-Funktionsblocks Änderungen am Sollwert vorgenommen werden.

So führen Sie eine AutoTune-Sequenz aus:

Zum Ausführen einer AutoTune-Sequenz müssen die ATParameters für die Eingabe festgelegt sein. Die Parameter „Input Gain“ und „DirectActing“ müssen entsprechend dem Prozess und dem festgelegten „DerivativeGain“ eingestellt sein (in der Regel 0,1). Die AutoTune-Sequenz wird mit der folgenden Abfolge eingeleitet:

1. Setzen Sie die Eingabe „Initialize“ auf TRUE.
2. Setzen Sie die Eingabe „Autotune“ auf TRUE.
3. Ändern Sie die Eingabe „Initialize“ auf FALSE.
4. Warten Sie, bis die Ausgabe „ATWarning“ sich auf 2 ändert.
5. Übertragen Sie die Werte für die Ausgabe „OutGains“ auf die Eingabe „Gains“.

Je nach Prozessen und Bedarf ist zum Abschließen der Abstimmung möglicherweise etwas Feinabstimmung erforderlich. Beim Einstellen von „TimeDerivative“ auf 0,0 zwingt der IPIDController „DerivativeGain“ auf 1,0 und fungiert anschließend als ein PI-Regler.

Verwenden der Anweisung „Proportional-Integral-Differential“ (PID)

In diesem Abschnitt finden sich spezifische Einzelheiten und Beispiele für die Verwendung einer proportionalen/integralen/differentiellen Anweisung, einschließlich der Folgenden:

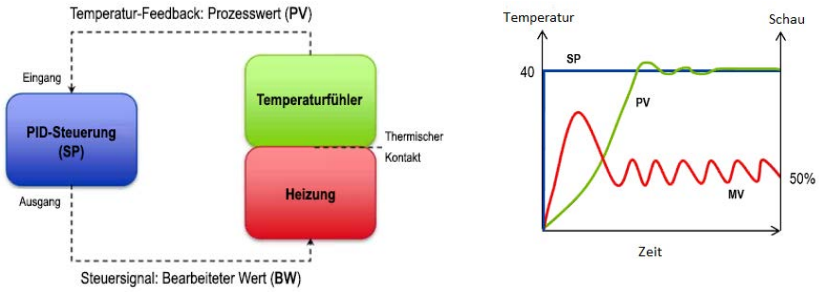
Beispiel: So erzeugen Sie eine Rückmeldungsschleife für den bearbeiteten Wert

Durch Hinzufügen einer Rückmeldungsschleife für den bearbeiteten Wert, beugt einem übermäßigen Aufschaukeln durch Bereitstellung eines minimalen und maximalen Werts für den BW vor.

Beispiel einer Temperatur-Rückmeldungsschleife

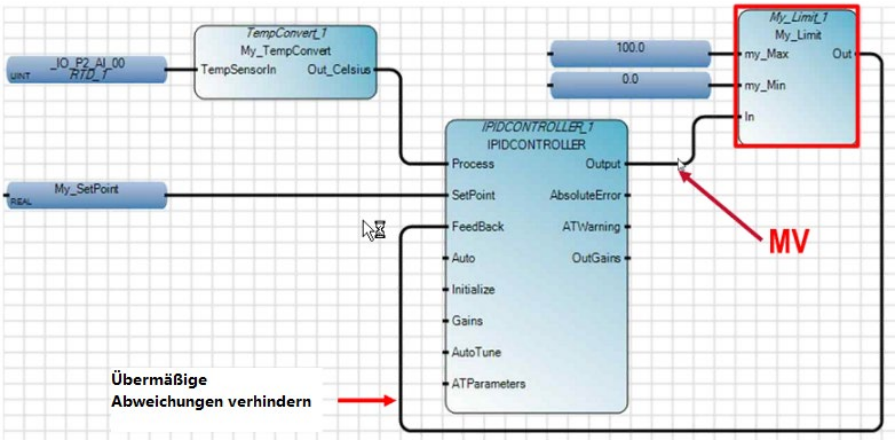
Zu Beginn des Temperatursteuerungsprozesses ist die Differenz zwischen dem Prozesswert (PW) und dem Sollwert (SW) groß, wie in der folgenden Darstellung angezeigt. In diesem Beispiel einer Temperatur-Rückmeldungsschleife startet der PW bei 0 Grad Celsius und bewegt sich in Richtung des SW-Wertes von 40 Grad Celsius. Beachten Sie

auch, dass die Fluktuation zwischen dem hohen und niedrigen bearbeiteten Wert (BW) geringer wird und sich mit der Zeit stabilisiert. Das Verhalten des BW hängt von den in jedem der P-, I- und D-Parameter verwendeten Werte ab.



IPIDController mit einer Rückmeldungsschleife

Das folgende Funktionsblockdiagramm beinhaltet eine Rückmeldungsschleife für den bearbeiteten Wert, der einem übermäßigen Aufschaukeln durch Bereitstellung eines minimalen und maximalen Werts für den BW vorbeugt.



Beispiel: So implementieren Sie die Selbsteinstellung in einen IPIDController-Funktionsblock

Sie können den AutoTune-Parameter des IPIDController-Funktionsblocks verwenden, um die Selbsteinstellung in das Steuerungsprogramm zu implementieren.

Anforderungen und Empfehlungen zur Selbsteinstellung

Im Folgenden finden Sie eine Übersicht über die Anforderungen und Empfehlungen zur erfolgreichen Implementierung einer Selbsteinstellung.

- Durch die Selbsteinstellung muss die Ausgabe der Rückmeldungsschleife oszillieren; dies bedeutet, dass der IPIDController häufig genug aufgefordert werden muss, die Oszillation entsprechend abzutasten.
- Der Anweisungsblock des IPIDController muss mit einem relativ konstanten Zeitintervall ausgeführt werden.
- Konfigurieren Sie die Scan-Uhrzeit des Programms so, dass sie der Hälfte des Oszillationszeitraums entspricht.
- Ziehen Sie die Verwendung eines Anweisungsblocks zur strukturierten Textunterbrechung (STI) in Betracht, um den IPIDController-Anweisungsblock zu steuern.

Beispiel: So fügen Sie einen UDFB zu einem PID-Programm hinzu

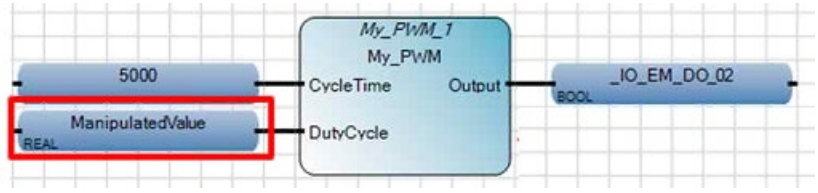
Sie können UDFBs außerhalb des Hauptprogramms hinzufügen, um spezielle Funktionen auszuführen, wie etwa die Konvertierung von Maßeinheiten oder die Übertragung von Werten.

Übertragen des Selbsteinstellungs-Anstiegswerts

Dieses UDFB überträgt den Wert der Selbsteinstellungssteigerung an „My_GainTransfer“, damit er vom Controller verwendet werden kann.

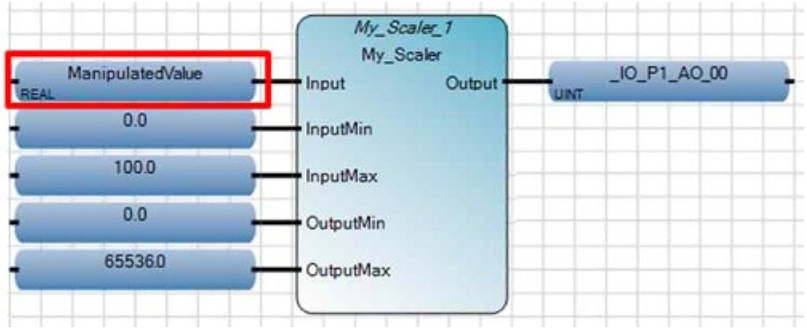
Konvertieren eines bearbeiteten Werts in eine digitale Ausgabe

Dieser UDFB konvertiert einen bearbeiteten Wert (BW) in einen digitalen Ausgang (DA), damit dieser zur Steuerung eines digitalen Eingangs (DE) verwendet werden kann.



Konvertieren eines bearbeiteten Werts in einem analogen Ausgang

Dieser UDFB konvertiert einen bearbeiteten Wert (BW) in einen analogen Ausgang (AA), damit dieser zur Steuerung eines analogen Eingangs (AE) verwendet werden kann.



Verwenden der Selbsteinstellung mit dem IPIDController-Funktionsblock

Sie können den AutoTune-Parameter des IPIDController-Funktionsblocks verwenden, um die Selbsteinstellung in das Steuerungsprogramm zu implementieren.

Anforderungen und Empfehlungen zur Selbsteinstellung

Im Folgenden finden Sie eine Übersicht über die Anforderungen und Empfehlungen zur erfolgreichen Implementierung einer Selbsteinstellung.

- Durch die Selbsteinstellung muss die Ausgabe der Rückmeldungsschleife oszillieren; dies bedeutet, dass der IPIDController häufig genug aufgefordert werden muss, die Oszillation entsprechend abzutasten.
- Der Anweisungsblock des IPIDController muss mit einem relativ konstanten Zeitintervall ausgeführt werden.
- Konfigurieren Sie die Scan-Uhrzeit des Programms so, dass sie der Hälfte des Oszillationszeitraums entspricht.
- Erwägen Sie die Verwendung eines Anweisungsblock für auswählbare, zeitlich festgelegte Unterbrechung (STI), um den IPIDController-Anweisungsblock zu kontrollieren.

Selbsteinstellung in Systemen erster und zweiter Ordnung

Die Selbsteinstellung kann in einem System erster Ordnung eingesetzt werden, das ein einzelnes Element verwendet, oder in einem System zweiter Ordnung, das zwei unabhängige Elemente verwendet.

Ein System erster Ordnung verwendet ein einzelnes unabhängiges Element zur Energiespeicherung. Zum Beispiel:

- Die Kühlung eines Flüssigkeitsbehälters, mit Wärmeenergie als Speichereinheit.
- Der Durchfluss von einem Flüssigkeitsbehälter in einen anderen, mit potentieller Energie als Speichereinheit.
- Ein Motor mit konstantem Drehmoment und Schwungscheibe, mit kinetischer Rotationsenergie als Speichereinheit.

- Ein elektrisches Netz mit Widerstands-Kapazitäts-Schaltung, mit kapazitiver Speicherenergie als Speichereinheit.

In einem System erster Ordnung kann die Funktion in einer Standardform geschrieben werden, wie etwa $f(t) = \tau\ dy/dt + y(t)$.

Dabei gilt:

Variable	Beschreibung	Beispiel: Die Kühlung eines Flüssigkeitsbehälters, mit Wärmeenergie als Speichereinheit
t	Systemzeitkonstante	Ist gleich RC Dabei gilt R = thermischer Widerstand der Wänden des Tanks C = thermische Kapazität der Flüssigkeit
f	Zwangsfunktion	Ist die Umgebungstemperatur
y	Variable des Systemzustands	Ist die Flüssigkeitstemperatur

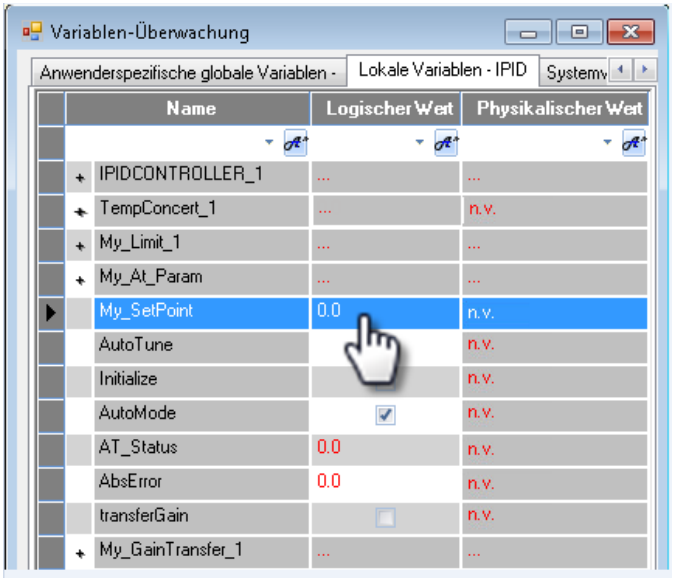
Ein Systems zweiter Ordnung verwendet zwei unabhängige Elemente, die gespeicherte Energie austauschen. Zum Beispiel:

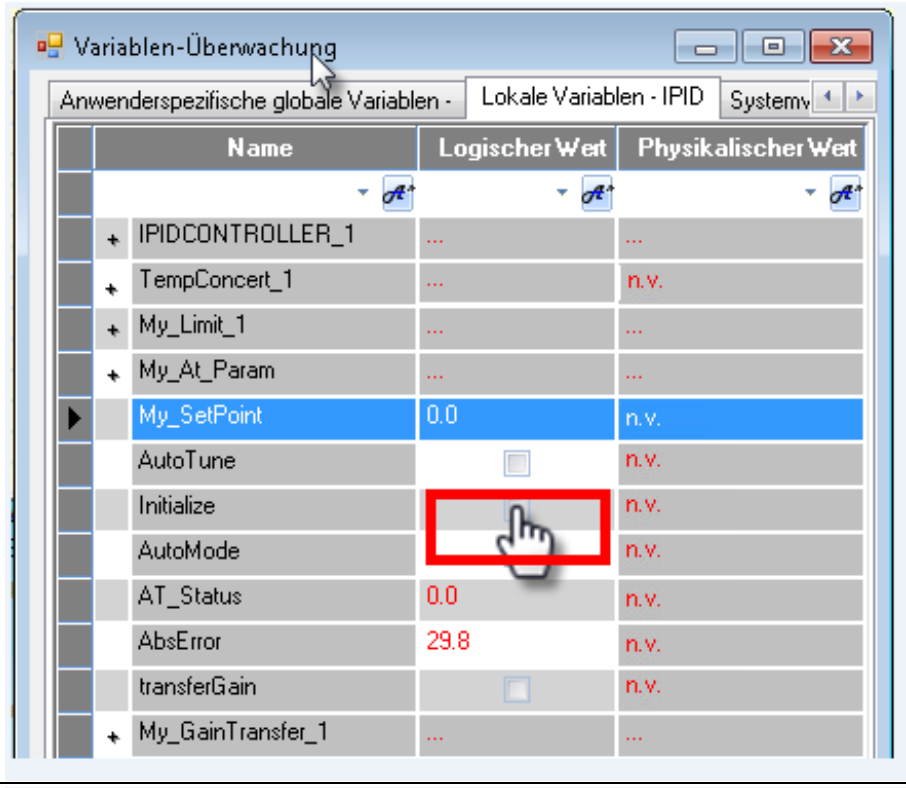
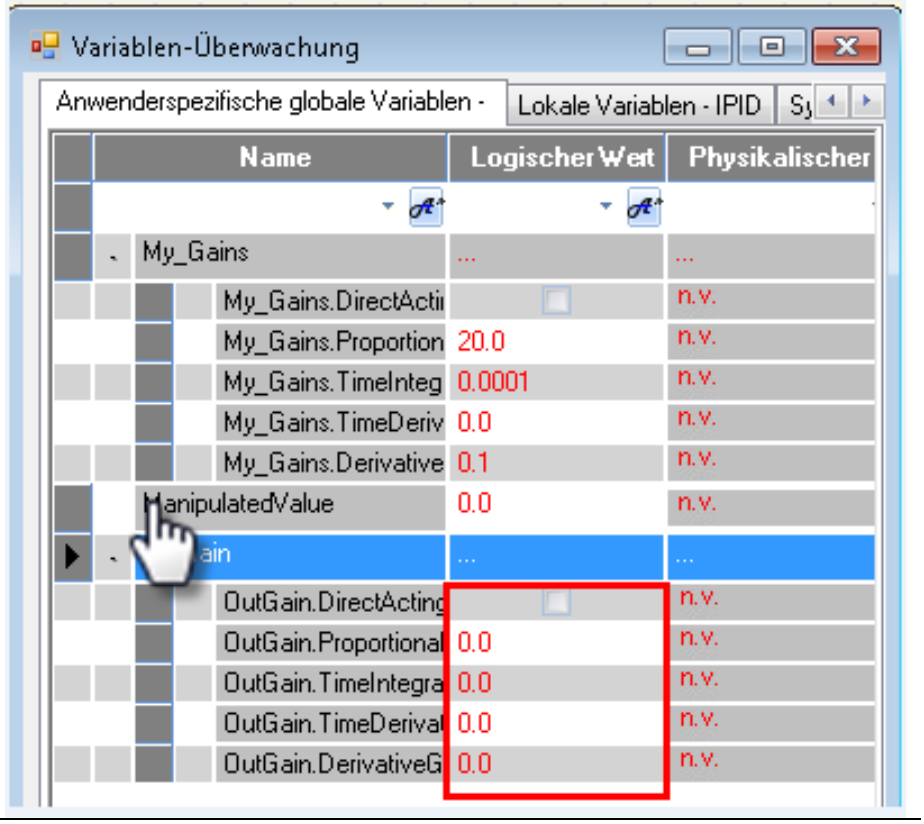
- Ein Motor, der ein Schwungrad antreibt, wobei der Motor mit dem Schwungrad über eine Welle mit einer Torsionssteifigkeit gekoppelt ist. Hier sind die kinetische Rotationsenergie und die Torsionsfederenergie die Speichereinheiten.
- Ein elektrischer Schaltkreis, bestehend aus Stromquelle, die einen seriellen Niedrigohmwiderstand (Drosselspule und Widerstand) mit einem Shunt C (Kondensator) betreibt. Die induktive Energie und die kapazitive Energie sind hier die Speichereinheiten.

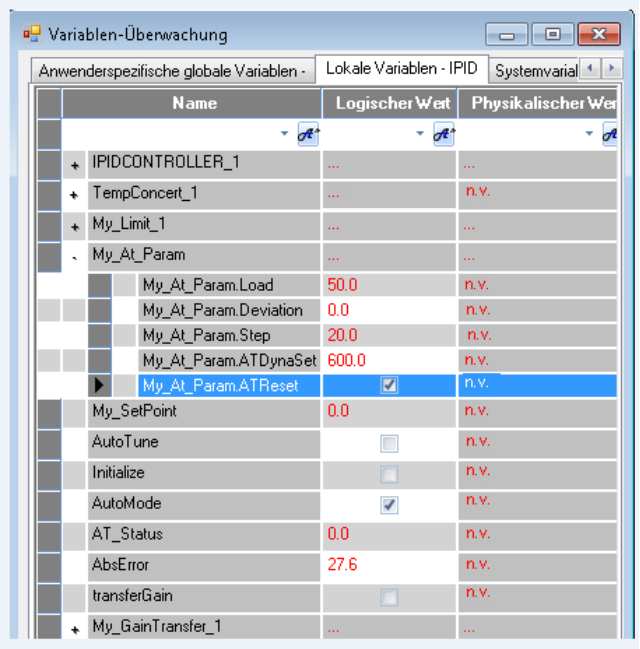
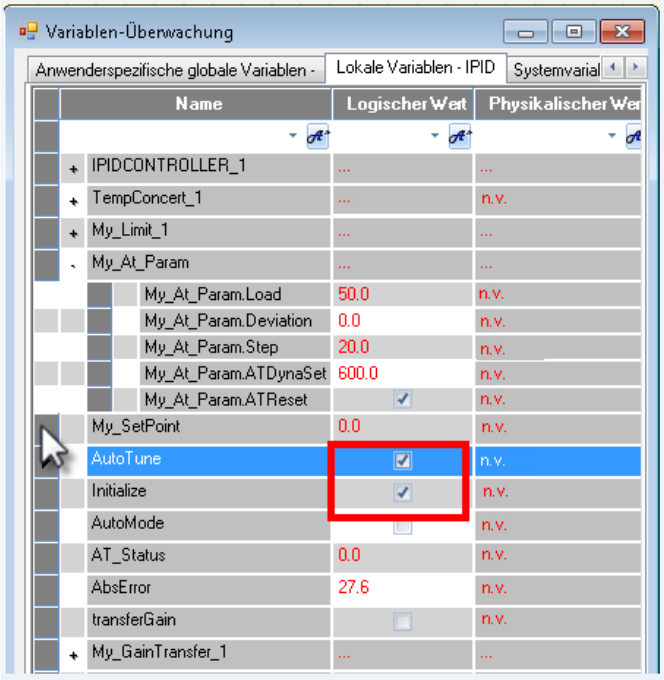
Motorbetriebene Systeme und Heizsysteme können typischerweise von einem Schaltkreis mit Niedrigohmwiderstand und Parallelkondensator erstellt werden.

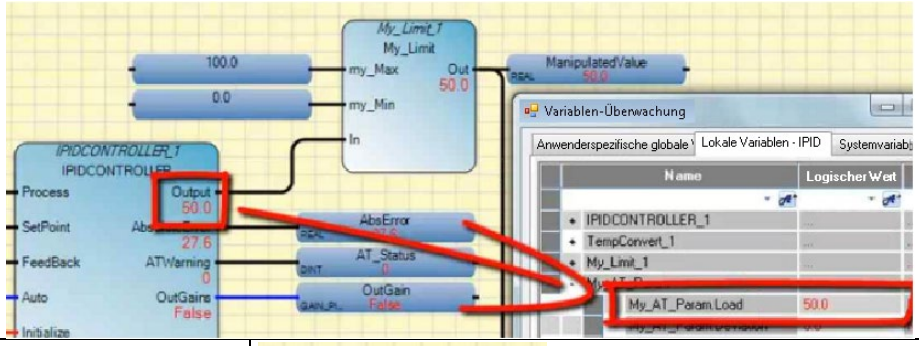
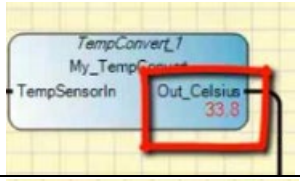
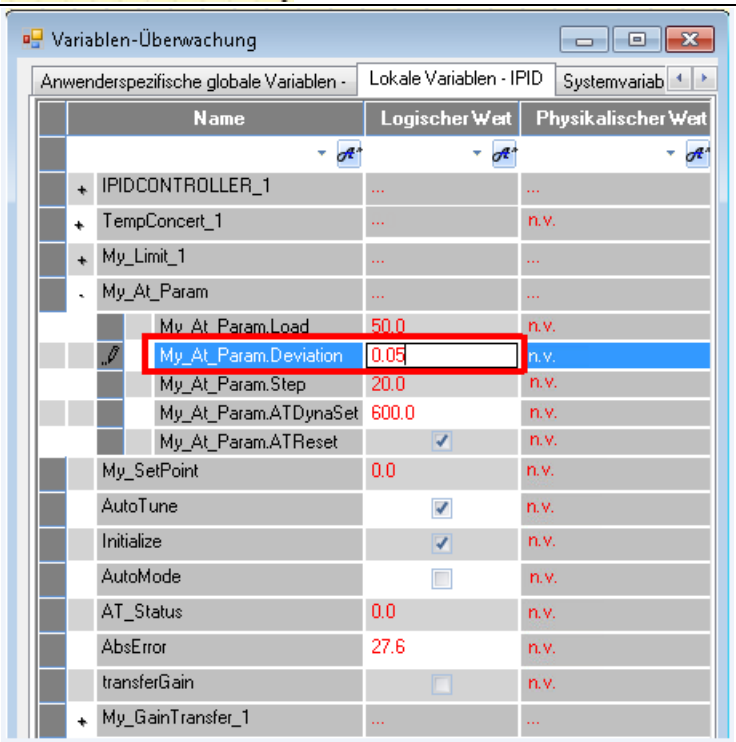
Konfigurieren der Selbsteinstellung

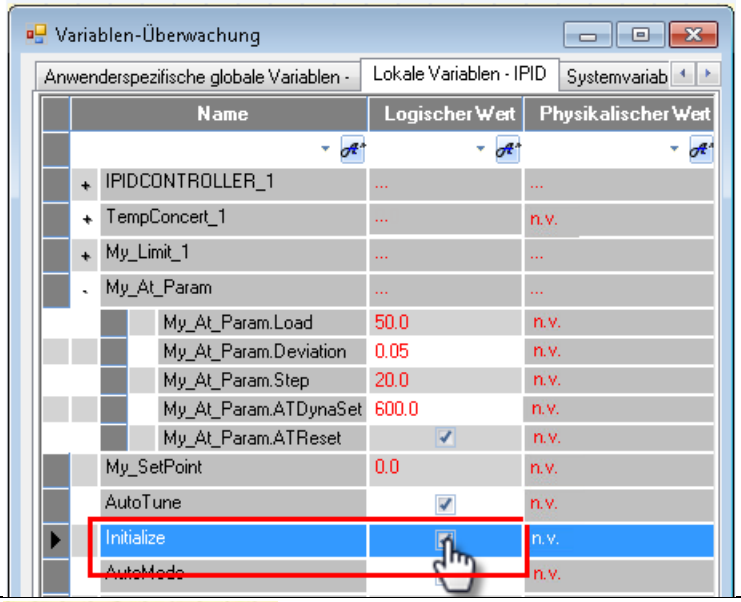
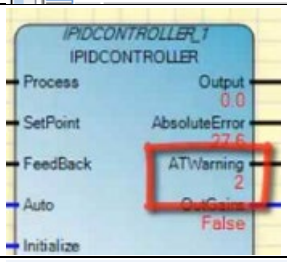
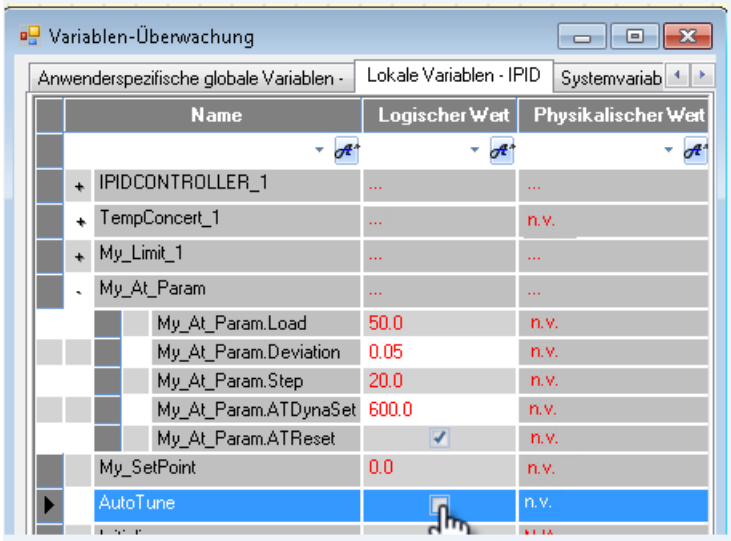
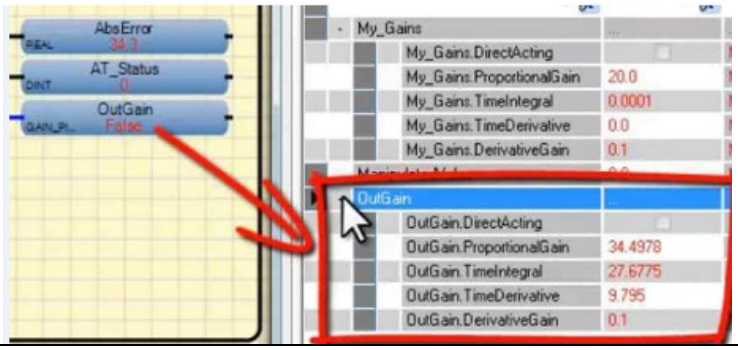
Führen Sie die folgenden allgemeinen Schritte aus, wenn Sie mithilfe einer IPIDController-Funktion eine automatische Optimierung implementieren auf seite 535.

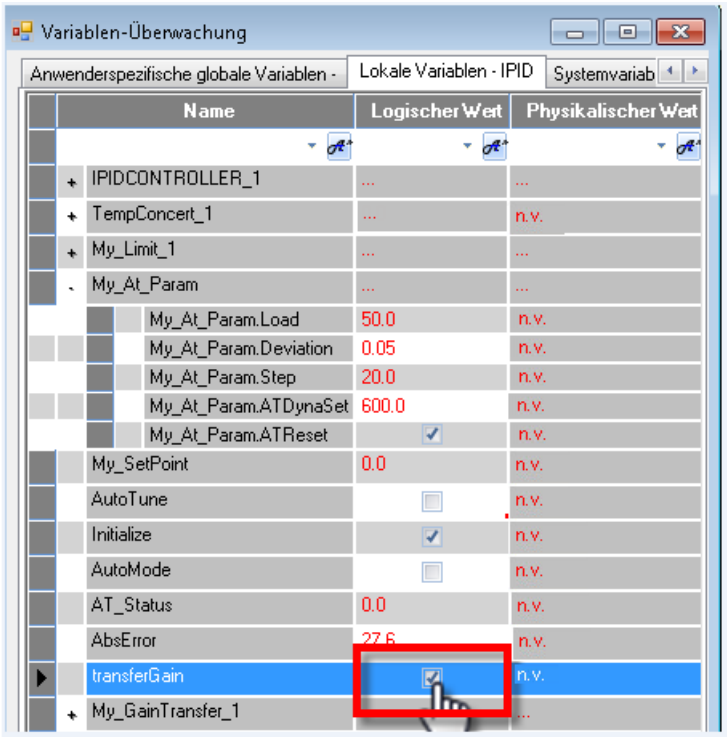
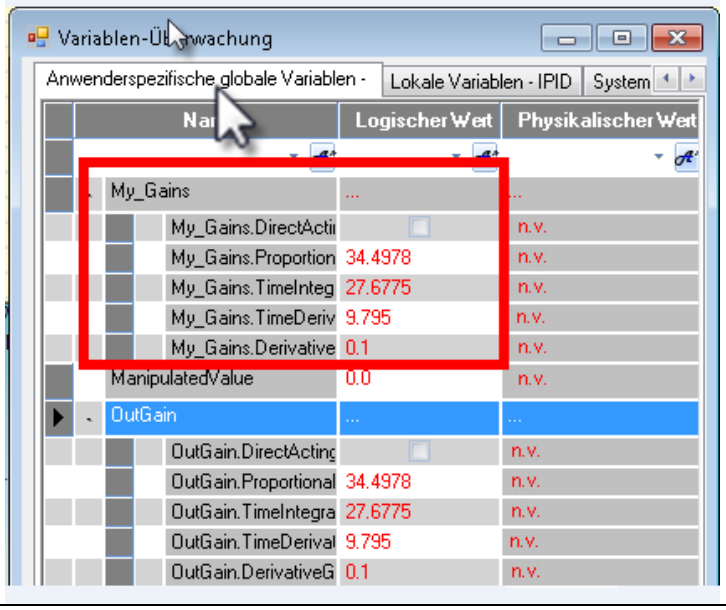
Nr.	Schritt	Beispiel
1	Sollwert auf Null zurückstellen.	

Nr.	Schritt	Beispiel
2	Auto-Modus auf „False“ ändern.	
3	Anstiegsparameter einstellen.	

Nr.	Schritt	Beispiel
4	Selbsteinstellungs-Parameter einstellen.	Stellen Sie die Selbsteinstellungs-Parameter einschließlich eines Anfangslastwerts ein, ändern Sie ihn schrittweise für die Ausgabe, eine geschätzte Zeit bis zum Abschluss der Selbsteinstellung und das Zurücksetzen der Selbsteinstellung. 
5	Setzen Sie „Initialize“ und „AutoTune“ auf True.	

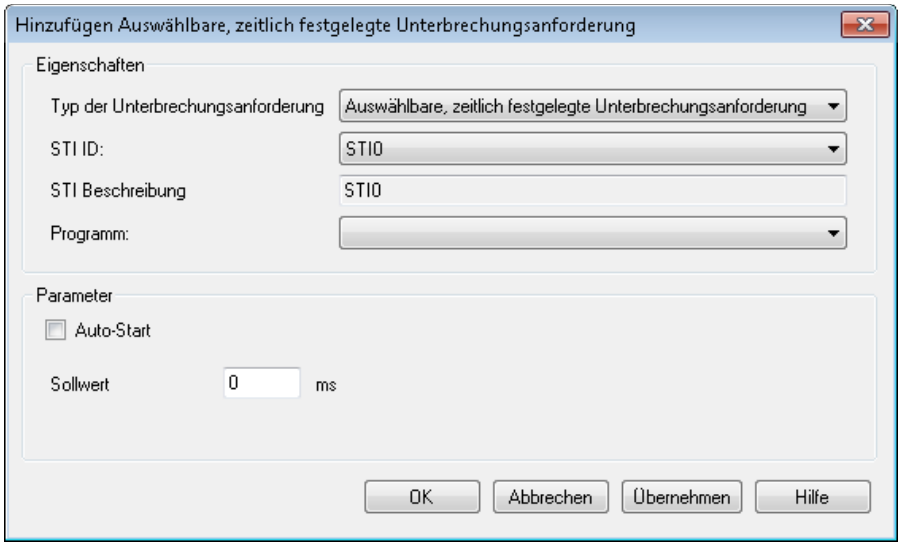
Nr.	Schritt	Beispiel
6	Achten Sie auf die Änderungen der Ausgabe auf den Wert von „Last“, wenn Sie „AutoTune“ auf True stellen.	
7	Achten Sie auf den schnellen Anstieg des Prozesswerts, bis er sich dem Sättigungspunkt nähert.	
8	Achten Sie auf die Stabilisierung des Prozesswerts und seiner Fluktuation.	
9	Stellen Sie die Abweichung ein.	

Nr.	Schritt	Beispiel
10	Setzen Sie „Initialize“ auf False.	
11	Der Controller startet die Selbsteinstellung. Warten Sie, bis „ATWarning“ sich auf 2 ändert.	
12	Setzen Sie „AutoTune“ auf False.	
13	Achten Sie auf die eingestellten Werte, die in „OutGains“ angezeigt werden.	

Nr.	Schritt	Beispiel
14	Übertragen Sie den Parameter von „OutGains“ auf „My_Gains“.	
15	Achten Sie auf die Aktualisierung des Controllers mit dem eingestellten Anstiegsparameter.	

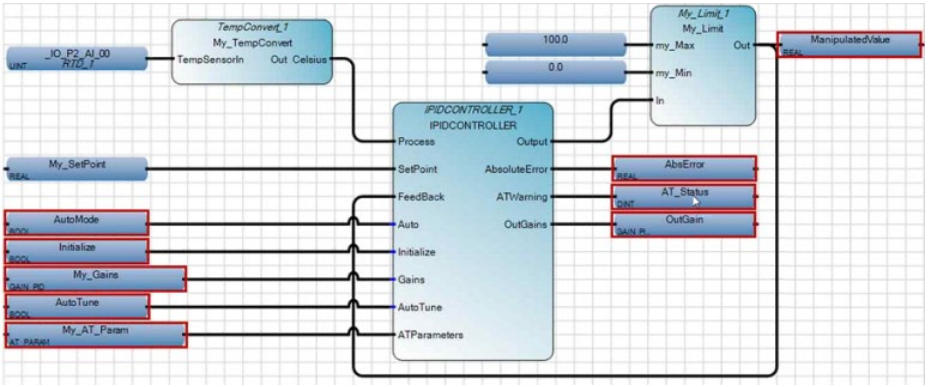
Verwenden Sie die auswählbare, zeitlich festgelegte Unterbrechung (STI) mit Selbsteinstellung

Obwohl eine PID-Anweisung funktioniert, wenn sie nicht von einer auswählbaren, zeitlich festgelegten Unterbrechung (STI) gesteuert wird, erhöht sich bei Verwendung einer STI die Erfolgsrate der Selbsteinstellung, da die Selbsteinstellung in einem festen Zyklus abläuft.



Beispiel: IPIDController mit Selbsteinstellung

Im folgenden Beispielprogramm werden die Variablen angezeigt, die zum Konfigurieren der Parameter für die Selbsteinstellung verwendet werden.



Selbsteinstellungs-Parameter

In der folgenden Tabelle sind die Variablen beschrieben, die mit jedem Parameter im Beispiel zur Konfiguration der Selbsteinstellung verwendet werden.

Eingabeparameter		
Variable	**Parameter**	**Beschreibung**
AutoMode	Auto	Der Betriebsmodus des PID-Controllers: TRUE - der Controller wird im Normalmodus ausgeführt. FALSE – ein abgeleiteter Begriff wird ignoriert und zwingt die Controller-Ausgabe, die Rückmeldung innerhalb der Grenzen des Controllers zu verfolgen; damit kann der Controller wieder auf „Auto“ zurückschalten, ohne mit der Ausgabe zu kollidieren.
Initialize	Initialize	Initialisiert die „AutoTune“-Sequenz. Eine Wertveränderung von TRUE zu FALSE oder FALSE zu TRUE veranlasst den Controller, jeglichen proportionalen Anstieg während des Zyklus zu eliminieren.
My_Gains	Gains	Etabliert „Steigungen PID“ für den IPIDController.

My_Gains.DirectActing	DirectActing	Definiert die Art der Aktion für die Ausgabe. TRUE - direkte Aktion, bei der sich die Ausgabe in die gleiche Richtung bewegt wie der Fehler. Das heißt, der tatsächliche Prozesswert ist höher als der „SetPoint“ (Sollwert) und die entsprechende Aktion des Controllers besteht im Erhöhen der Ausgabe. Beispiel: Kühlanwendungen. FALSE - umgekehrte Aktion, bei der sich die Ausgabe in die entgegengesetzte Richtung bewegt wie der Fehler. Das heißt, der tatsächliche Prozesswert ist höher als der „SetPoint“ (Sollwert) und die entsprechende Aktion des Controllers besteht im Absenken der Ausgabe. Beispiel: Heizanwendungen.
My_Gains.ProportionalGain	ProportionalGain	Proportionaler Anstieg für PID (>= 0,0001).
My_Gains.TimeIntegral	TimeIntegral	Zeitintegralwert für PID (>= 0,0001). Diese Tendenz zu einer Oszillation erhöht sich mit dem Absenken bei ti.
My_Gains.TimeDerivative	TimeDerivative	Abgeleiteter Zeitwert für PID (> 0,0). Die Dämpfung erhöht sich mit der Erhöhung der abgeleiteten Zeit, sinkt jedoch ab, wenn der abgeleitete Zeitwert zu groß ist.
My_Gains.DerivativeGain	Abgeleiteter Anstieg für PID (> 0,0).	
AutoTune	Wenn hier TRUE und für Auto und Initialize FALSE eingestellt ist, wird die AutoTune-Sequenz gestartet.	
ATParameters		
Load	• Anfangsausgabewert während der Selbsteinstellung. • Ermöglicht eine Stabilisierung des Prozesswerts an der Last	
Deviation	• Die Standardabweichung für eine Reihe stabilisierter Prozesswerte. Zum Beispiel, wenn der Prozesswert zwischen 31,4 und 32,0 stabilisiert ist, wäre der Wert der Standardabweichung (32,0–31,4)/2 = 0,3. • Einige Prozesswerte, wie etwa Temperatur, benötigen einige Zeit, um sich zu stabilisieren.	
Schritt	• Der Prozess zur Selbsteinstellung beachtet nur, wie der Prozesswert auf die Änderungen des Schrittwerts reagiert und leitet die Anstiegsparameter ab.	
ATDynaSet	• Zugewiesene Zeit für die Fertigstellung der Selbsteinstellung. Sie muss größer sein als die für den Selbsteinstellungsprozess benötigte Zeit. • Ein allgemeiner Wert ist für viele System 600 Sekunden allerdings können einige Systeme mehr Zeit benötigen.	
ATReset	• Falls TRUE, wird die Ausgabe nach Abschluss der Selbsteinstellung auf „0“ zurückgesetzt. • Falls FALSE, bleibt der Ausgangswert nach Abschluss der Selbsteinstellung auf dem „Load“-Wert.	
Ausgangsparameter		
Parameter	**Beschreibung**	
AbsoluteError	Absoluter Fehler („Process“ – „SetPoint“) vom Controller.	
ATWarning	Warnung für die Selbsteinstellungssequenz. Mögliche Werte sind: • 0 - es erfolgt keine automatische Abstimmung. • 1 – im Selbsteinstellungsmodus. • 2 - es erfolgt eine Selbsteinstellung. • -1 - FEHLER 1 Eingabe automatisch auf TRUE gesetzt, keine Selbsteinstellung möglich. • -2 - FEHLER 2 Selbsteinstellungsfehler; „ATDynaSet“ abgelaufen	
OutGains	Berechneter Anstieg nach AutoTune-Sequenz.	

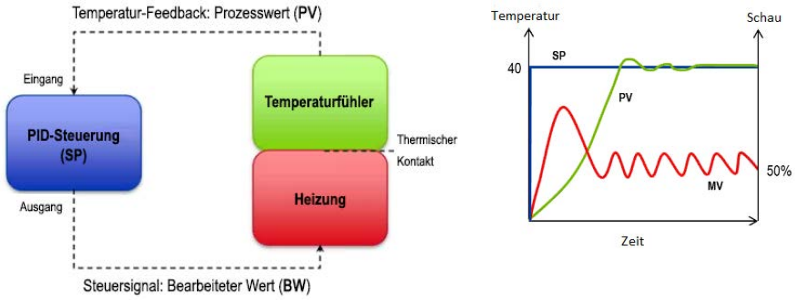
Beispiel: So erzeugen Sie eine Rückmeldungsschleife für den bearbeiteten Wert

Durch Hinzufügen einer Rückmeldungsschleife für den bearbeiteten Wert, beugt einem übermäßigen Aufschaukeln durch Bereitstellung eines minimalen und maximalen Werts für den BW vor.

Beispiel einer Temperatur-Rückmeldungsschleife

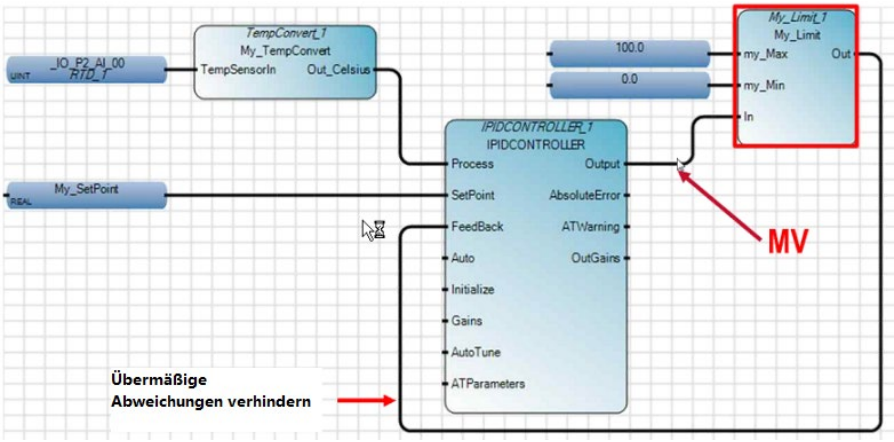
Zu Beginn des Temperatursteuerungsprozesses ist die Differenz zwischen dem Prozesswert (PW) und dem Sollwert (SW) groß, wie in der folgenden Darstellung angezeigt. In diesem Beispiel einer

Temperatur-Rückmeldungsschleife startet der PW bei 0 Grad Celsius und bewegt sich in Richtung des SW-Wertes von 40 Grad Celsius. Beachten Sie auch, dass die Fluktuation zwischen dem hohen und niedrigen bearbeiteten Wert (BW) geringer wird und sich mit der Zeit stabilisiert. Das Verhalten des BW hängt von den in jedem der P-, I- und D-Parameter verwendeten Werte ab.



IPIDController mit einer Rückmeldungsschleife

Das folgende Funktionsblockdiagramm beinhaltet eine Rückmeldungsschleife für den bearbeiteten Wert, der einem übermäßigen Aufschaukeln durch Bereitstellung eines minimalen und maximalen Werts für den BW vorbeugt.



Beispiel: So fügen Sie einen UDFB zu einem PID-Programm hinzu

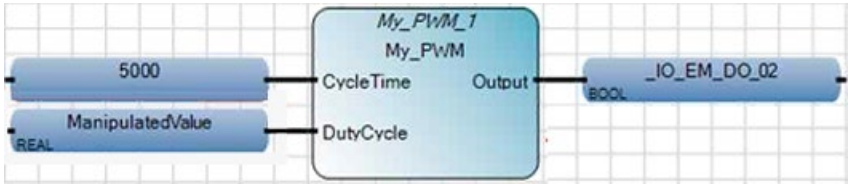
Sie können für PID-Programme UDFBs außerhalb des Hauptprogramms hinzufügen, um spezielle Funktionen auszuführen, wie etwa die Konvertierung von Maßeinheiten oder die Übertragung von Werten. Beispiele zu UDFBs:

Übertragen des Selbsteinstellungs-Anstiegswerts

Dieses UDFB überträgt den Wert der Selbsteinstellungssteigerung an „My_GainTransfer“, damit er vom Controller verwendet werden kann.

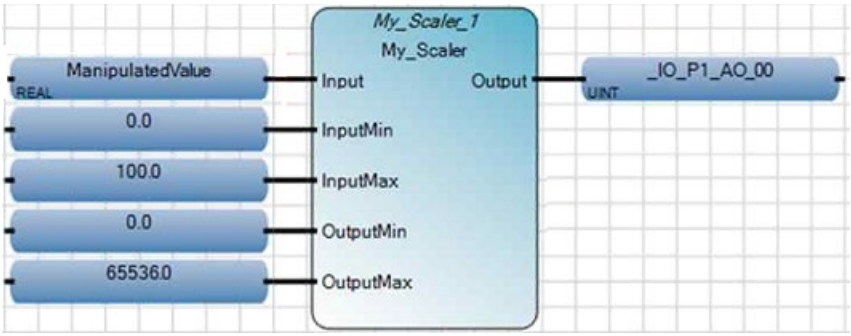
Konvertieren eines bearbeiteten Werts in eine digitale Ausgabe

Dieser UDFB konvertiert einen bearbeiteten Wert (BW) in einen digitalen Ausgang (DA), damit dieser zur Steuerung eines digitalen Eingangs (DE) verwendet werden kann.



Konvertieren eines bearbeiteten Werts in einem analogen Ausgang

Dieser UDFB konvertiert einen bearbeiteten Wert (BW) in einen analogen Ausgang (AA), damit dieser zur Steuerung eines analogen Eingangs (AE) verwendet werden kann.



Beispiel: So erstellen Sie ein IPIDController-Programm zur Temperatursteuerung

Das Temperatursteuerungsprogramm hält die Temperatur innerhalb der Steuerungszone.

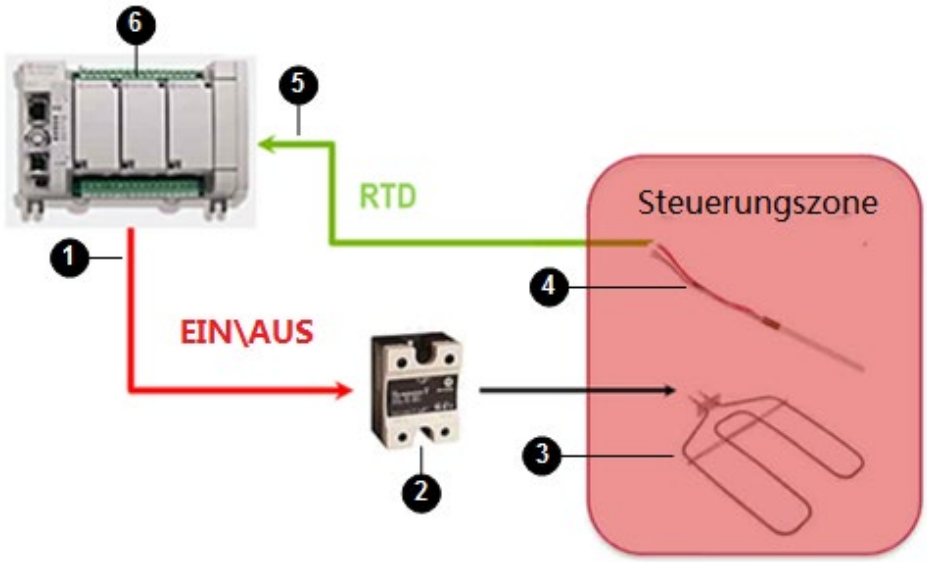
Sollwert, Prozess- und bearbeitete Werte

In der folgenden Tabelle ist definiert, wie die Werte SW, PW und BW im Temperatursteuerungsprogramm verwendet werden.

Element	Beschreibung
Sollwert (SW)	Misst die Temperatur in Grad Celsius, was die Temperatur für die Steuerungszone definiert.
Prozesswert (PW)	Muss in die gleiche Einheit wie der SW konvertiert werden, also in Grad Celsius.
Bearbeiteter Wert (BW)	Muss in einen analogen Wert konvertiert werten, sodass er die Ausgabe zum PWM zur Steuerung des Heizelements sein kann.

Temperatursteuerungssystem

Das folgende Diagramm und Tabelle definieren die Komponenten des Temperatursteuerungssystems, die durch das Temperatursteuerungsprogramm gesteuert werden, und die Ereignisse, die auftreten, wenn das Steuerprogramm ausgeführt wird.



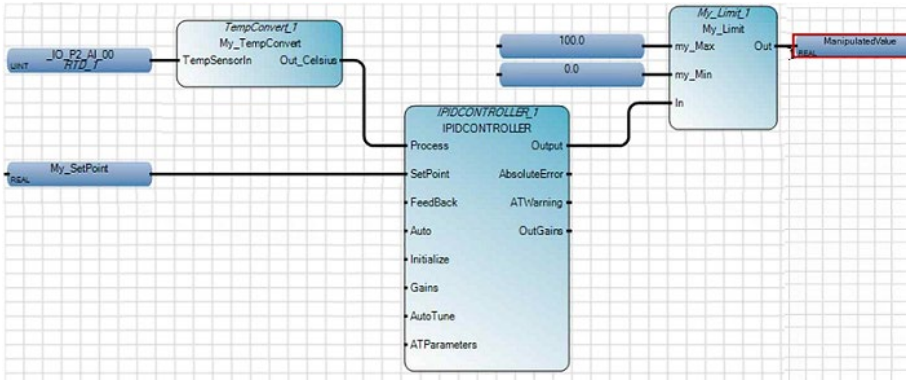
Sequenz von Ereignissen für das Temperatursteuerungsprogramm

In der folgenden Tabelle werden die Komponenten des Temperatursteuerungssystems angezeigt. Die Tabelle beschreibt sequenziell die Ereignisse, die im System auftreten, wenn das Temperatursteuerungsprogramm ausgeführt wird.

Nein	Element	Beschreibung
1	Controller-Ausgang	Sendet BW an PWM (Ein/Aus).
2	Impulsbreitenmodulation (PWM-Temperatur-Controller	Halbleiterrelais, das das Heizelement steuert.
3	Heizelement	Erhöht Temperatur in der Steuerungszone.
4	Widerstands-Temperaturdetektor (RTD)	Misst die Temperatur in der Steuerungszone und sendet den PW (RTD-Signal) an den Controller-Eingang.
5	Controller-Eingang	Empfängt den PW (RTD-Signal).
6	PLC-Programm	Konvertiert den PW (RTD-Signal) in die gleiche Einheit um wie die des SW (Grad Celsius) und bestimmt die Differenz zwischen dem PW und dem SW und stellt den BW gemäß der in den P-, I- und D-Parametern definierten Parameterwerte ein.

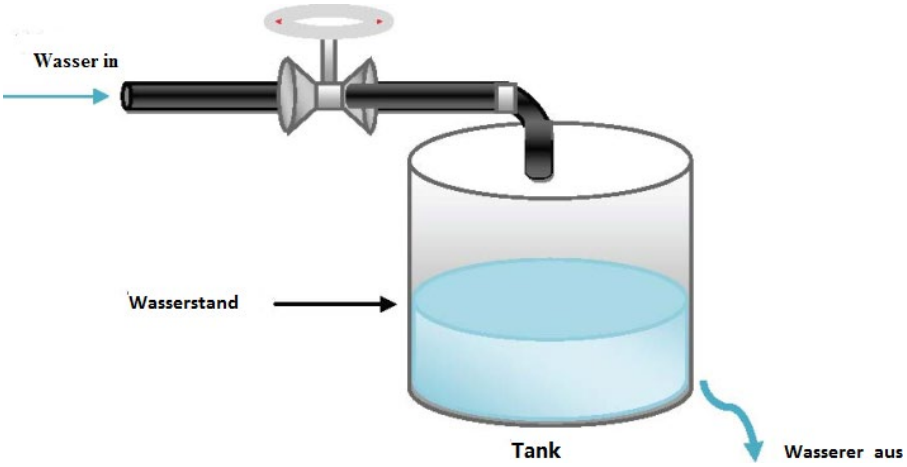
Beispiel: Funktionsblockdiagramm zur Temperatursteuerung

Dieses Funktionsblockdiagramm zeigt die vordefinierten und benutzerdefinierten Funktionsblöcke an, die in der Anwendung zur Temperatursteuerung in einer Steuerungszone verwendet werden.



Beispiel: So erstellen Sie ein IPIDController-Programm zur Steuerung des Wasserversorgungspegels

Das Beispiel eines Programms zur Steuerung des Wasserversorgungspegels hält ausreichend Wasser im Wasserversorgungstank bereit, der über einen Abfluss verfügt. Ein Magnetventil steuert das eintretende Wasser und füllt den Tank bis zu einer festgelegten Marke auf; das abfließende Wasser wird ebenfalls auf eine zuvor festgelegte Abflussrate gehalten.



Informationen zu einem Programmbeispiel

Das Beispiel des Programm zur Steuerung des Wasserversorgungspegels beinhaltet folgende Funktionen.

- Die Sequenz der Ereignisse, die im Steuerungsprozess auftreten
- Art der Verwendung von Sollwert, Prozess- und bearbeitete Werte im Steuerungsprogramm
- Ein Beispiel eines Funktionsblockdiagramms, das den IPIDController und andere Anweisungsblöcke anzeigt

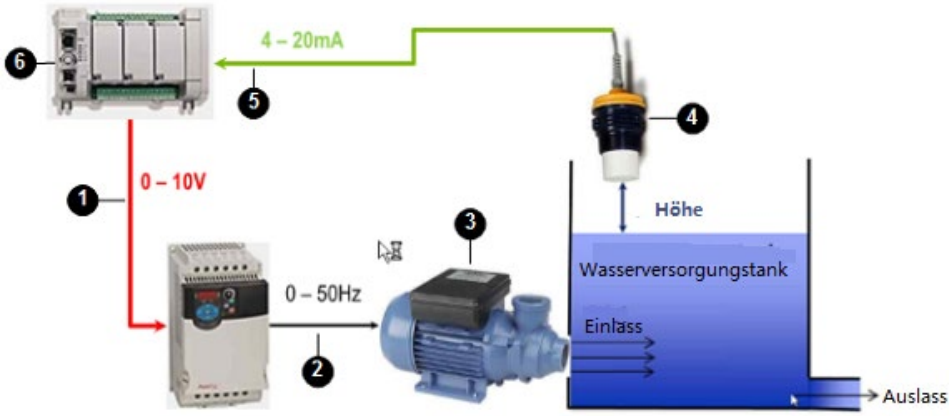
Sollwert, Prozess- und bearbeitete Werte

In der folgenden Tabelle ist definiert, wie die Werte SW, PW und BW im Programm zur Steuerung des Wasserversorgungspegels verwendet werden.

Element	Beschreibung
Sollwert (SW)	Misst die Höhe, die den gewünschten Wasserversorgungspegel definiert.
Prozesswert (PW)	4-20 mA muss in die gleiche Einheit wie der SW konvertiert werden, also in eine Maßeinheit der Höhe.
Bearbeiteter Wert (BW)	Muss in einen analogen Wert konvertiert werten, sodass er die Ausgabe zum Antrieb der Pumpe sein kann.

System für Wasserversorgungspegel

Im folgenden Diagramm werden die Komponenten des Systems für den Wasserversorgungspegel angezeigt, das vom Programm zur Steuerung des Wasserversorgungspegels gesteuert wird. Die Tabelle nach dem Diagramm beschreibt die Ereignisse, die auftreten, wenn das Steuerungsprogramm ausgeführt wird.



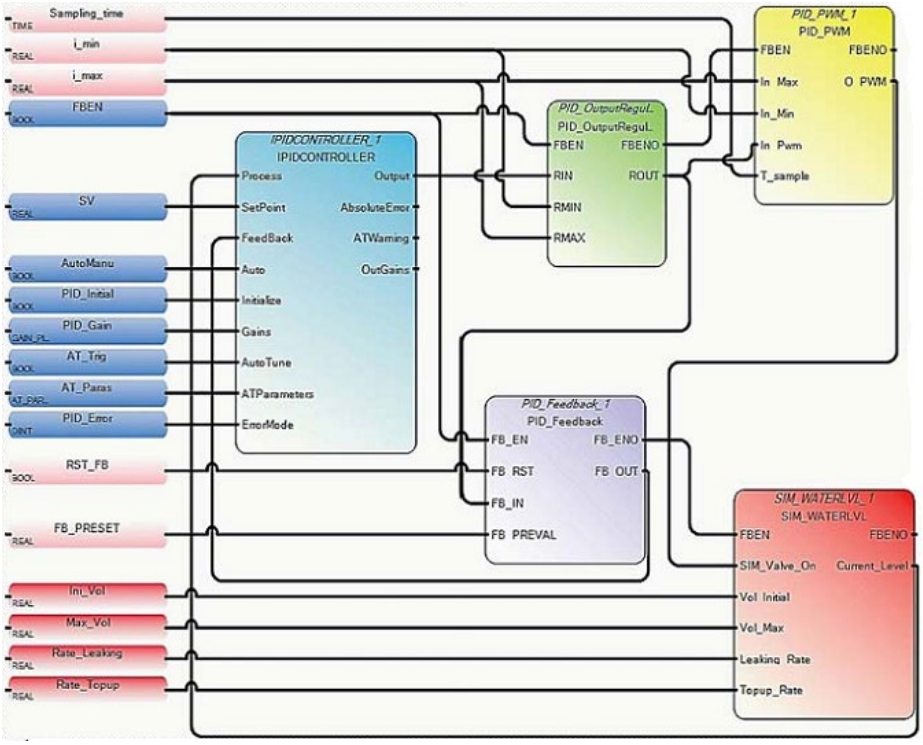
Sequenz der Ereignisse in einem System für den Wasserversorgungspegel

In der folgenden Tabelle werden die Komponenten des Systems für den Wasserversorgungspegel angezeigt. Die Tabelle beschreibt sequenziell die Ereignisse, die im System auftreten, wenn das Programm zur Steuerung des Wasserversorgungspegels ausgeführt wird.

Nein	Element	Beschreibung
1	Controller-Ausgang	Sendet den BW an den PowerFlex-Antrieb (0-10 V).
2	PowerFlex-Antrieb	Steuert die Wasserpumpe (0-50 Hz).
3	Wasserpumpe	Steuert den Wasserpegel im Versorgungstank.
4	Ausgabetransfergerät	Misst die Höhe des Wasserversorgungspegels (4-20 mA) und sendet den PW an den Controller.
5	Controller-Eingang	Empfängt dem PW (Wasserversorgungspegel von 4-20 mA).
6	PLC-Programm	Konvertiert den PW (RTD-Signal) in die gleiche Einheit um wie die des SW (Maßeinheit der Höhe) und bestimmt die Differenz zwischen dem PW und dem SW und stellt den BW gemäß der in den P-, I- und D-Parametern definierten Parameterwerte ein.

Beispiel: Funktionsblockdiagramm zur Steuerung des Wasserversorgungspegels

Das folgende Funktionsblockdiagramm zeigt die vordefinierten und benutzerdefinierten Funktionsblöcke für das Programm an, um den Wasserversorgungspegel zu steuern.



Im FBD des Wasserpegels verwendete Funktionsblöcke und UDFBs

Diese in der Sprache des Funktionsblockdiagramms (FBD) entwickelte Anwendung verwendet die in der folgenden Tabelle beschriebenen Anweisungen.

Funktionsblock	Beschreibung
IPIDController-Funktionsblock	Bereitstellen einer eine PID-Prozesssteuerung.
PID_OutputRegulator UDFB	Regelt die Ausgabe von IPIDCONTROLLER innerhalb eines sicheren Bereichs, um sicherzustellen, dass die im Prozess verwendete Hardware nicht beschädigt wird. **Beispielcode:** IF RMIN ≤ RIN ≤ RMAX, then ROUT = RIN, IF RIN < RMIN, then ROUT = RMIN, IF RIN > RMAX, then ROUT = RMAX
PID_Feedback UDFB	Agiert als Multiplexer. **Beispielcode:** IF "FB_RST" is false, FB_OUT=FB_IN; If "FB_RST" is true, then FB_OUT=FB_PREVAL.
PID_PWM UDFB	Stellt eine PWM-Funktion bereit, die einen realen Wert in einen zeitbezogenen EIN/AUS-Ausgang konvertiert.
SIM_WATERLVL UDFB	Simuliert den Prozess im Anwendungsbeispiel.

PID (Proportional-Integral-Differential)

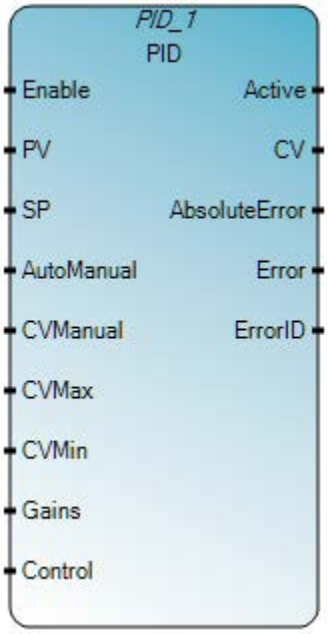
Ausgangsanweisung, die die physischen Eigenschaften, wie etwa Temperatur, Druck, Füllstand oder Durchflussrate, mithilfe von Prozessschleifen steuert.

Details zum Betrieb:

- Wenn aktiviert, steuert PID den Prozess mithilfe der Eingangsparameter einschließlich SW und Anstieg des PID-Controllers.
- Beim Moduswechsel von Ausführung zu Programm wird die PID-Anweisung deaktiviert, Parameterwerte werden beibehalten.
- Beim Moduswechsel von Programm zu Ausführung bleibt die PID-Anweisung deaktiviert, bis ein Benutzer „Enable“ auf TRUE setzt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Aktivierungsanweisung. TRUE – Ausführung mit den aktuellen Eingabeparameter starten. FALSE – PID wird nicht ausgeführt. CV auf 0 setzen, AbsoluteError berechnen.
PV	Eingang	REAL	Prozesswert. Dieser Wert wird in der Regel aus einem analogen Eingang gelesen. Die SI-Einheit muss dem Sollwert entsprechen.
SP	Eingang	REAL	Der Sollwert für den Prozess.
AutoManual	Eingang	BOOL	Automatische oder manuelle Modusauswahl. TRUE – CV wird von PID gesteuert. FALSE – PID wird ausgeführt und CV wird von CVManual-Eingang gesteuert.

Parameter	Parametertyp	Datentyp	Beschreibung
CVManual	Eingang	REAL	Definierter Steuerwerteingang für den manuellen Modus. Der gültige Bereich für CVManual ist: CVMin < CVManual < CVMax
CVMin	Eingang	REAL	Minimaler Steuerwert. Wenn CV < CVMin, dann gilt CV = CVMin. Wenn CVMin > CVMax, tritt ein Fehler auf.
CVMax	Eingang	REAL	Maximaler Steuerwert. Wenn CV > CVMax, dann gilt CV = CVMax. Wenn CVMax < CVMin, tritt ein Fehler auf.
Gains	Eingang	PID_GAINS	Anstieg von PID für Controller. Verwenden Sie den Datentyp PID_GAINS, um den Anstiegsparameter zu konfigurieren.
Control	Eingang	BOOL	Regelrichtung des Prozesses: TRUE – direkte Aktion, wie z. B. Kühlung. FALSE – umgekehrte Aktion, wie z. B. Heizung.
Active	Ausgang	BOOL	Status des PID-Controllers. TRUE – PID ist aktiv. FALSE – PID wurde angehalten.
CV	Ausgang	REAL	Die Ausgabe des Steuerwerts. Wenn ein Fehler aufgetreten ist, ist CV 0.
AbsoluteError	Ausgang	REAL	Der Absolutfehler ist die Differenz zwischen dem Prozesswert (PV) und dem Sollwert (SP).
Error	Ausgang	BOOL	Zeigt das Vorhandensein eine Fehlerbedingung an. TRUE – beim Vorgang ist ein Fehler aufgetreten. FALSCH – Vorgang wurde erfolgreich abgeschlossen oder der Befehl wird nicht ausgeführt.
ErrorID	Ausgang	USINT	Ein eindeutiger numerischer Wert, der den Fehler angibt. Die Fehler sind in PID-Fehlercodes definiert.

PID_GAINS-Datentyp

In der folgenden Tabelle wird der Datentyp „PID_GAINS“ für den PID-Befehl auf seite 525 beschrieben.

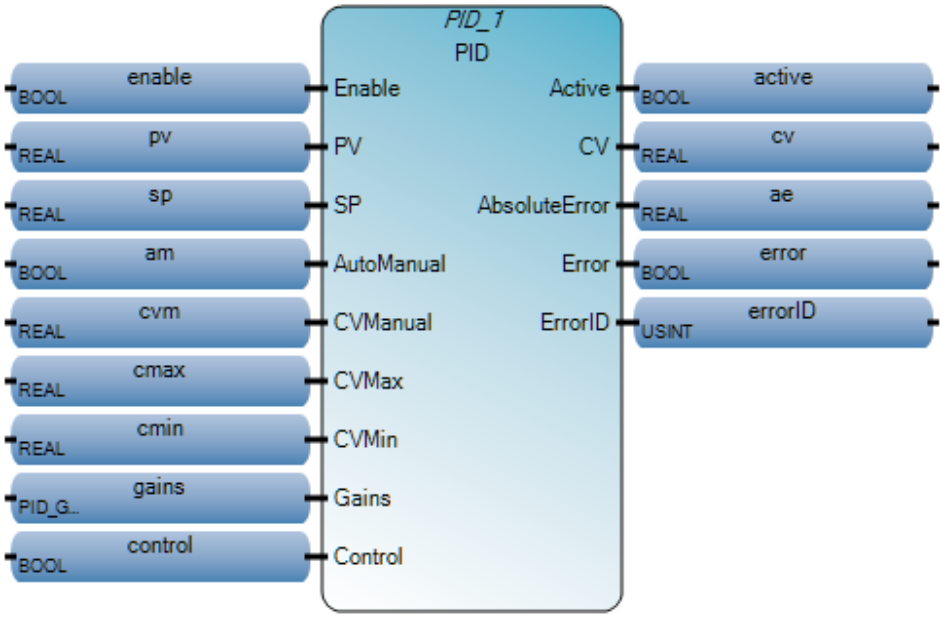
Parameter	Parametertyp	Datentyp	Beschreibung
Kc	Eingang	REAL	Controller-Anstieg für PID. Proportional und Integral sind von diesem Anstieg abhängig. (>= 0,0001). Durch die Erhöhung von Kc wird die Reaktionszeit verkürzt, jedoch auch das Überschwingen und Oszillieren des PID erhöht. Wenn Kc ungültig ist, tritt ein Fehler auf.
Ti	Eingang	REAL	Integrale Zeitkonstante in Sekunden (>= 0,0001). Durch Erhöhen des Ti wird das Überschwingen und Oszillieren des PID reduziert. Wenn Ti ungültig ist, tritt ein Fehler auf.
Td	Eingang	REAL	Abgeleitete Zeitkonstante in Sekunden (>= 0,0). Wenn Td gleich 0 ist, ist kein differenzierendes Verhalten vorhanden, und das PID wird zu einer PI-Steuerung. Ein Erhöhen des Td reduziert das Überschwingen und beseitigt das Oszillieren der PID-Steuerung. Wenn Td ungültig ist, tritt ein Fehler auf.
FC	Eingang	REAL	Filterkonstante (>= 0,0). Der empfohlene Bereich für FC ist 0 bis 20. Durch Erhöhen des FC glättet die Antwort der PID-Steuerung. Wenn FC ungültig ist, tritt ein Fehler auf.

PID-Fehlercodes

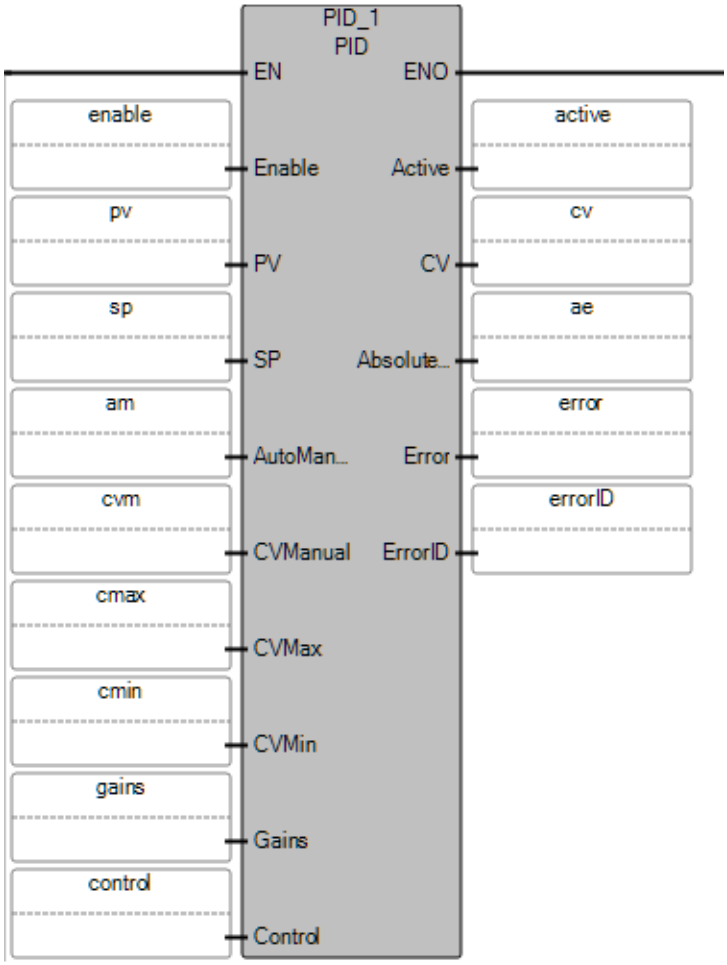
Verwenden Sie diese Tabelle um die PID-Fehlercodes und deren Beschreibung zu bestimmen.

Fehlercode	Fehlerbeschreibung
0	PID funktioniert normal.
1	Kc ist ungültig.
2	Ti ist ungültig.
3	Td ist ungültig.
4	FC ist ungültig.
5	CVMin > CVMax oder CVMax < CVMin
6	CVManual < CVMin CVManual ist ungültig.
7	CVManual > CVMax CVManual ist ungültig.

Beispiel für ein PID-Funktionsblockdiagramm



Beispiel für einen PID-Kontaktplan

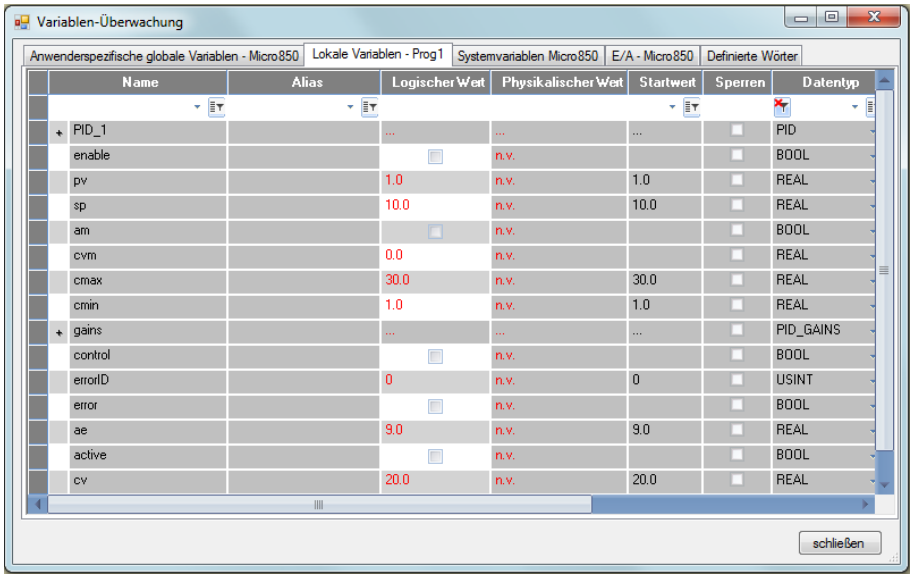


Beispiel für strukturierten PID-Text

```
1  PID_1(enable, pv, sp, am, cvm, cmax, cmin, gains, control);
2  active := PID_1.Active;
3  cv := PID_1.CV;
4  ae := PID_1.AbsoluteError;
5  error := PID_1.Error;
6  errorID := PID_1.ErrorID;
```

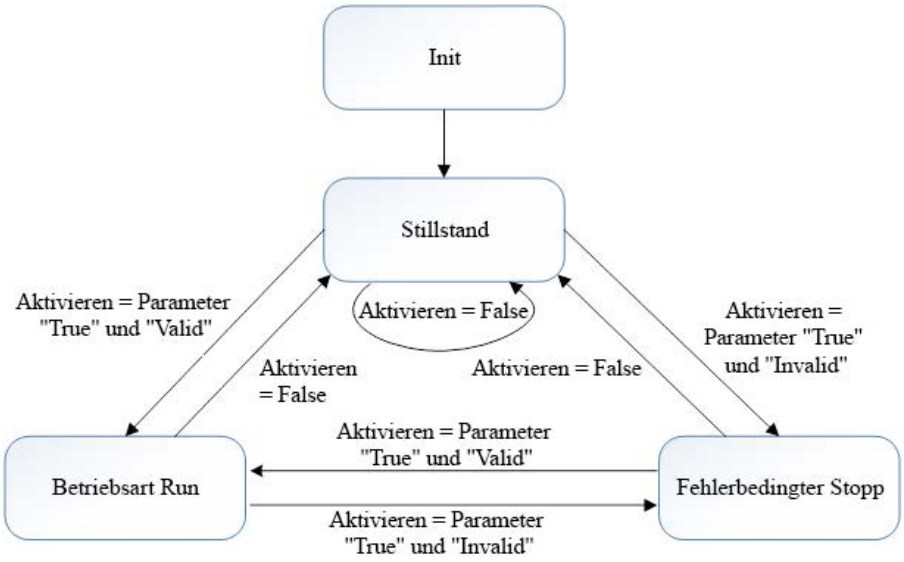
```
PID_1(
void PID_1(BOOL Enable, REAL PV, REAL SP, BOOL AutoManual, REAL CVManual, REAL CVMax, REAL CVMin, PID_GAINS Gains, BOOL Control)
Typ: PID, Proportional-Integral-Derivativ
```

PID-Ergebnisse



PID-Anweisung für Maschinenstatus

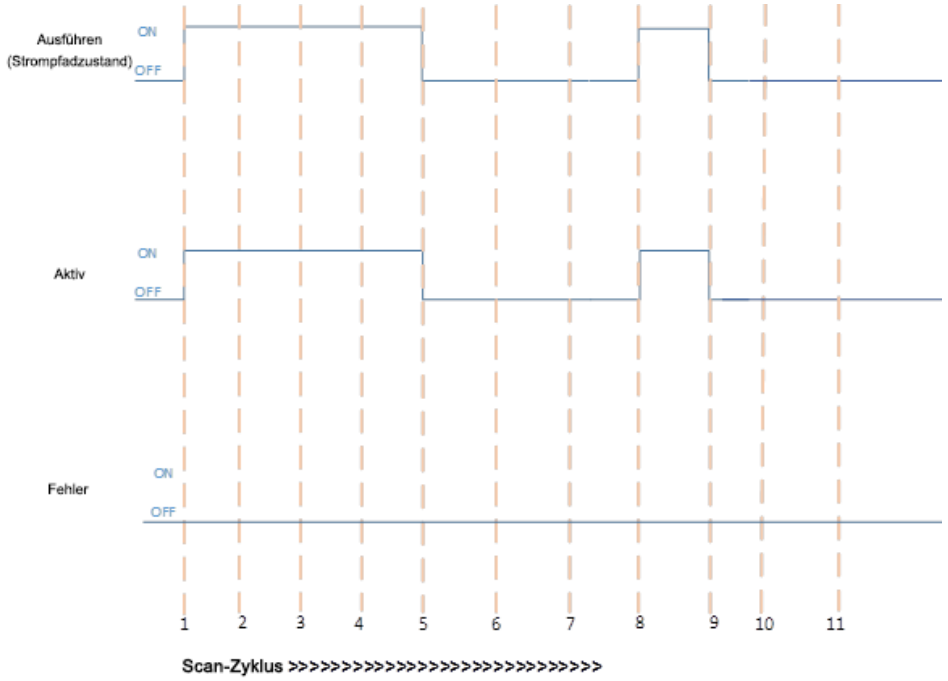
Im Diagramm zum PID-Maschinenzustand werden die Verarbeitungszustände für den PID-Befehl auf seite 550 beschrieben.



PID-Anweisung für Timing-Diagramme

In den folgenden Beispielen für ein Zeitmessungsdiagramm werden Ausführungsszenarien für den PID-Befehl (Proportional-Integral-Differential, proportional-integral-derivative) auf seite 550 beschrieben.

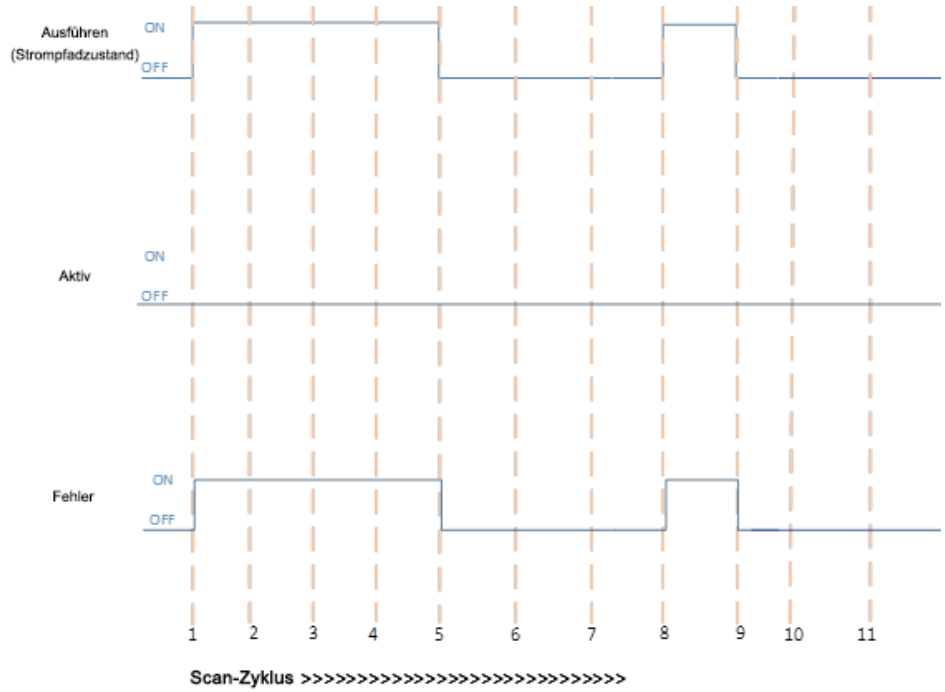
Erfolgreiche PID-Ausführung



Verwenden Sie diese Tabelle, um die Parameterwerte für jeden Scan-Zyklus zu ermitteln.

Scan-Zyklus	Beschreibung
1, 8	Strompfadbedingung wird TRUE, wenn gilt: • Aktivierungseingabebit ist TRUE. • Eingabeparameter sind gültig. • Aktives Bit ist TRUE. • Fehlerbit ist FALSE.
2,3,4	Keine Änderung in Strompfadbedingung. • Aktivierungseingabebit ist TRUE. • Eingabeparameter sind gültig. • PID-Ausgabeparameter werden aktualisiert.
5, 9	Strompfadbedingung wird FALSE, wenn gilt: • Aktivierungsbit ist FALSE. • Löscht die PID-Ausgabeparameter mit Ausnahme von AbsoluteError. • AbsoluteError berechnet basierend auf PV- und SP-Eingabewerten.
6, 7, 10, 11	Keine Änderung in Strompfadbedingung. • Aktivierungsbit ist FALSE. • Löscht die PID-Ausgabeparameter mit Ausnahme von AbsoluteError. • AbsoluteError berechnet basierend auf PV- und SP-Eingabewerten.

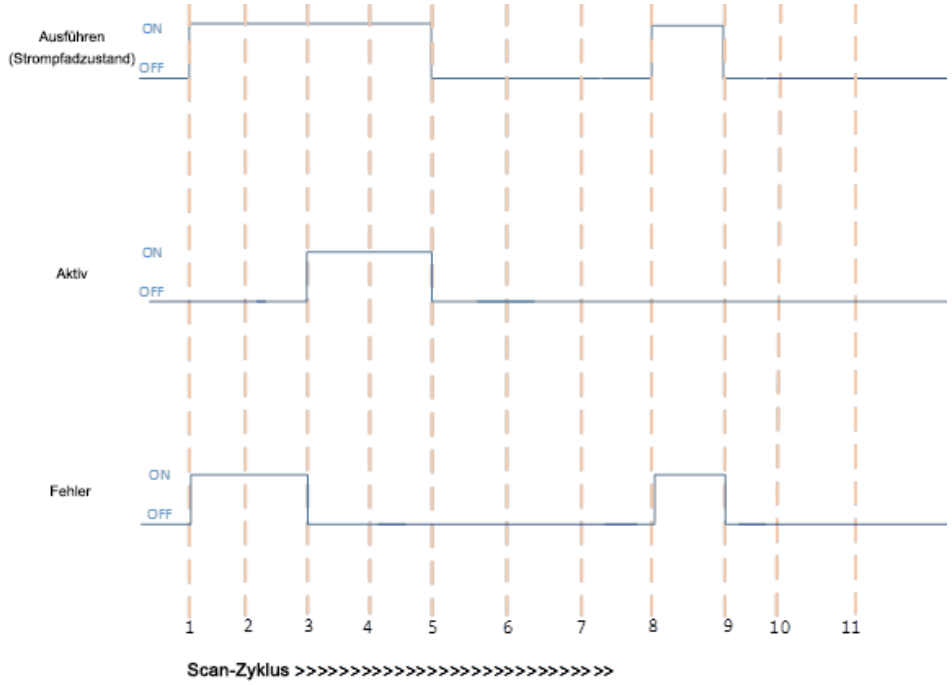
PID-Ausführung mit Fehler



Verwenden Sie diese Tabelle, um die Parameterwerte für jeden Scan-Zyklus zu ermitteln.

Scan-Zyklus	Beschreibung
1, 8	Strompfadbedingung wird TRUE, wenn gilt: • Aktivierungseingabebit ist TRUE. • Eingabeparameter sind ungültig. • Aktives Bit ist FALSE. • Fehlerbit ist TRUE. ErrorID-Ausgabe ist eingestellt. • CV-Ausgabe ist auf 0 eingestellt. • AbsoluteError berechnet basierend auf PV- und SP-Eingabewerten.
2,3,4	Keine Änderung in Strompfadbedingung. • Aktivierungseingabebit ist TRUE. • Eingabeparameter sind ungültig. • PID-Ausgabeparameter werden aktualisiert.
5, 9	Strompfadbedingung wird FALSE, wenn gilt: • Aktivierungsbit ist FALSE. • Löscht die PID-Ausgabeparameter mit Ausnahme von AbsoluteError. • AbsoluteError berechnet basierend auf PV- und SP-Eingabewerten.
6, 7, 10, 11	Keine Änderung in Strompfadbedingung. • Aktivierungsbit ist FALSE. • Löscht die PID-Ausgabeparameter mit Ausnahme von AbsoluteError. • AbsoluteError berechnet basierend auf PV- und SP-Eingabewerten.

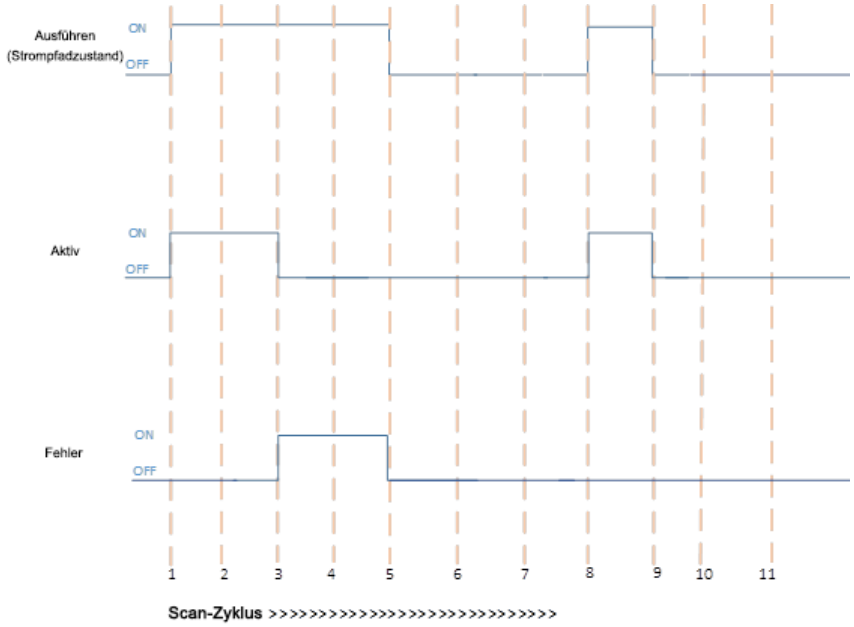
PID-Ausführung mit Fehler, dann erfolgreiche Ausführung



Verwenden Sie diese Tabelle, um die Parameterwerte für jeden Scan-Zyklus zu ermitteln.

Scan-Zyklus	Beschreibung
1, 8	Strompfadbedingung wird TRUE, wenn gilt: • Aktivierungseingabebit ist TRUE. • Eingabeparameter sind ungültig. • Aktives Bit ist FALSE. • Fehlerbit ist TRUE. ErrorID-Ausgabe ist eingestellt. • CV-Ausgabe ist auf 0 eingestellt. • AbsoluteError berechnet basierend auf PV- und SP-Eingabewerten.
2	Keine Änderung in Strompfadbedingung. • Aktivierungseingabebit ist TRUE. • Eingabeparameter sind ungültig. • PID-Ausgabeparameter werden aktualisiert.
3, 4	Keine Änderung in Strompfadbedingung. • Aktivierungseingabebit ist TRUE. • Eingabeparameter sind gültig. • Aktives Bit ist TRUE. • Fehlerbit ist FALSE. • PID-Ausgabeparameter werden aktualisiert.
5, 9	Strompfadbedingung wird FALSE, wenn gilt: • Aktivierungsbit ist FALSE. • Löscht die PID-Ausgabeparameter mit Ausnahme von AbsoluteError. • AbsoluteError berechnet basierend auf PV- und SP-Eingabewerten.
6, 7, 10, 11	Keine Änderung in Strompfadbedingung. • Aktivierungsbit ist FALSE. • Löscht die PID-Ausgabeparameter mit Ausnahme von AbsoluteError. • AbsoluteError berechnet basierend auf PV- und SP-Eingabewerten.

Erfolgreiche PID-Ausführung und Fehler



Verwenden Sie diese Tabelle, um die Parameterwerte für jeden Scan-Zyklus zu ermitteln.

Scan-Zyklus	Beschreibung
1, 8	Strompfadbedingung wird TRUE, wenn gilt: • Aktivierungseingabebit ist TRUE. • Eingabeparameter sind gültig. • Aktives Bit ist TRUE. • Fehlerbit ist FALSE. • PID-Ausgabeparameter werden aktualisiert.
2	Keine Änderung in Strompfadbedingung. • Aktivierungseingabebit ist TRUE. • Eingabeparameter sind gültig. • PID-Ausgabeparameter werden aktualisiert.
3, 4	Keine Änderung in Strompfadbedingung. • Aktivierungseingabebit ist TRUE. • Eingabeparameter sind ungültig. • Fehlerbit ist TRUE. ErrorID-Ausgabe ist eingestellt. • CV-Ausgabe ist auf 0 eingestellt. • AbsoluteError berechnet basierend auf PV- und SP-Eingabewerten.
5, 9	Strompfadbedingung wird FALSE, wenn gilt: • Aktivierungsbit ist FALSE. • Löscht die PID-Ausgabeparameter mit Ausnahme von AbsoluteError. • AbsoluteError berechnet basierend auf PV- und SP-Eingabewerten.
6, 7, 10, 11	Keine Änderung in Strompfadbedingung. • Aktivierungsbit ist FALSE. • Löscht die PID-Ausgabeparameter mit Ausnahme von AbsoluteError. • AbsoluteError berechnet basierend auf PV- und SP-Eingabewerten.

Anweisungen für Echtzeituhr (RTC)

Befolgen Sie die Anweisungen für die Echtzeituhr, den Kalender und die Uhr zu konfigurieren.

Anweisung	Beschreibung
RTC_READ auf seite 385	Liest die Informationen des Echtzeituhr-Moduls (Real-Time Clock, RTC).
RTC_SET auf seite 387	Stellt die Daten der Echtzeituhr auf die RTC-Modul-Informationen ein.

RTC_READ (Lesen der Echtzeituhr)

Liest die Informationen des Echtzeituhr-Moduls (Real-Time Clock, RTC).

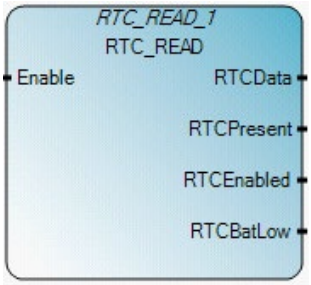
Details zum Betrieb:

- Micro810- oder Micro820-Controller mit integrierter RTC:
  - RTCBatLow ist immer auf null (0) gesetzt.
  - RTCEnabled ist immer auf eins (1) gesetzt.
- Wenn der integrierte RTC seine Ladung/seinen Speicherinhalt aufgrund eines Stromausfalls verloren hat:
  - RTCData wird auf 2000/1/1/0/0/0 gesetzt.
  - RTCEnabled wird auf eins (1) gesetzt.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Die Ausgänge werden jedes Mal zurückgesetzt, wenn die Anweisung auf die simulierte Steuerung (2080-LC50-48QWB-SIM) angewendet wird.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – RTC-Informationen lesen.

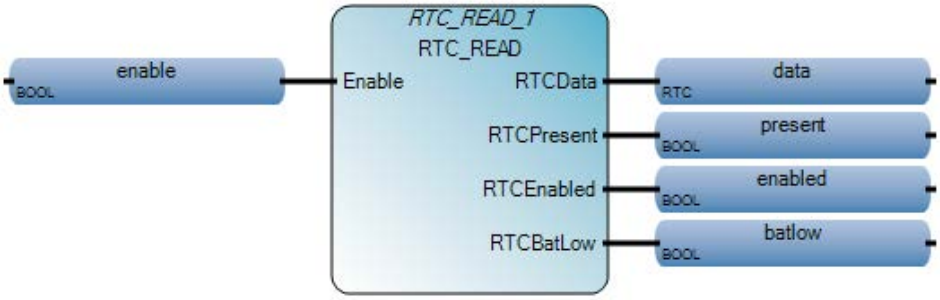
Parameter	Parametertyp	Datentyp	Beschreibung
			FALSE – Es findet kein Lesevorgang statt, und die ausgegebenen RTC-Daten sind ungültig.
RTCData	Ausgang	RTC	Informationen zu RTC-Daten: JJ/MM/TT, hh/mm/ss, Woche. RTCData-Ausgabe wird mit dem Datentyp „RTC“ definiert.
RTCPresent	Ausgang	BOOL	TRUE - Freilaufende Uhr wird verwendet oder RTC-Hardware ist angeschlossen. FALSE - Freilaufende Uhr wird nicht verwendet oder RTC-Hardware ist nicht angeschlossen.
RTCEnabled	Ausgang	BOOL	TRUE - Freilaufende Uhr wird verwendet oder RTC-Hardware ist aktiviert (Zeitgebung). FALSE - Freilaufende Uhr wird nicht verwendet, RTC-Hardware ist nicht deaktiviert (keine Zeitgebung).
RTCBatLow	Ausgang	BOOL	TRUE - Ladestand der RTC-Batterie niedrig. FALSE - Ladestand der RTC-Batterie nicht niedrig.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt nur für Kontaktplan-Programme.

Datentyp RTC

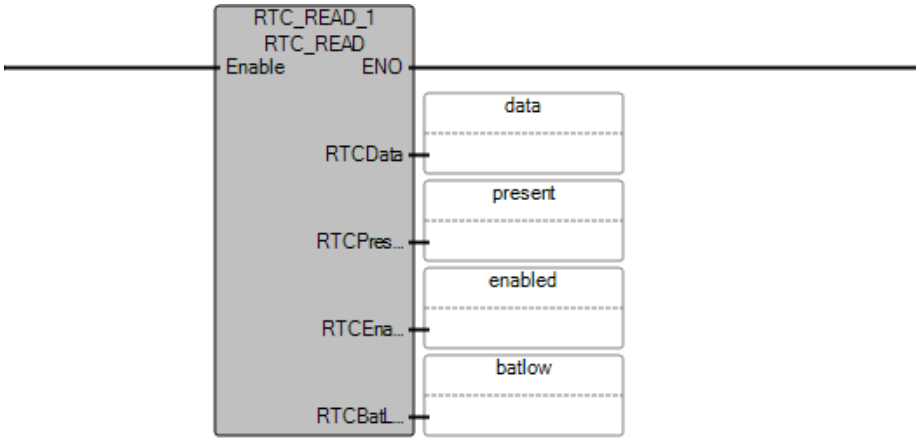
Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „RTC“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Year	UINT	Die Jahreseinstellung für den RTC. 16-Bit-Wert, der gültige Bereich reicht von 2000 (01. Januar, 00:00:00) bis 2098 (31. Dezember, 23:59:59)
Month	UINT	Die Monatseinstellung für den RTC.
Day	UINT	Die Tageseinstellung für den RTC.
Hour	UINT	Die Stundeneinstellung für den RTC.
Minute	UINT	Die Minuteneinstellung für den RTC.
Second	UINT	Die Sekundeneinstellung für den RTC.
DayOfWeek	UINT	Die Wochentagseinstellung für den RTC. Dieser Parameter wird für „RTC_SET“ ignoriert.

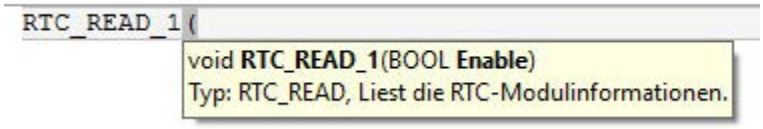
Beispiel für ein RTC_READ-Funktionsblockdiagramm



Beispiel für einen RTC_READ-Kontaktplan



Beispiel für einen strukturierten RTC_READ-Text




```
1  RTC_READ_1(enable);
2  data := RTC_READ_1.RTCData;
3  present := RTC_READ_1.RTCPresent;
4  enabled := RTC_READ_1.RTCEnabled;
5  batlow := RTC_READ_1.RTCBatLow;
```

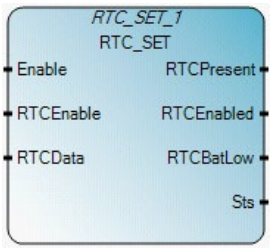
RTC_SET (Echtzeituhr einstellen)

Stellen Sie die Echtzeituhr (RTC) auf die RTC-Modul-Informationen ein.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Die Ausgänge werden jedes Mal zurückgesetzt, wenn die Anweisung auf die simulierte Steuerung (2080-LC50-48QWB-SIM) angewendet wird.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Enable	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – RTC_SET mit den RTC-Informationen aus der Eingabe ausführen.

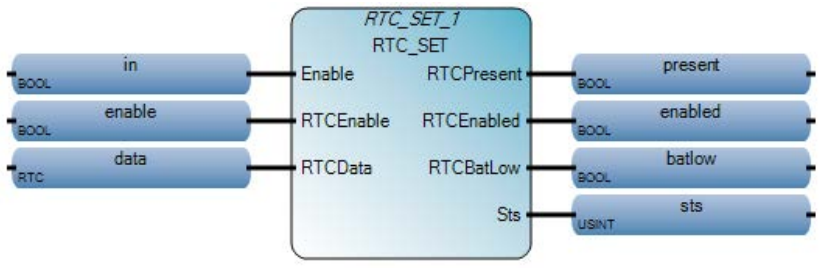
Parameter	Parametertyp	Datentyp	Beschreibung
			Typischerweise nur für 1 Programm-Scan beim Update von RTC ausführen. FALSE – RTC_SET nicht ausführen. Auf FALSE einstellen, um RTC normal auszuführen.
RTCEnable	Eingang	BOOL	TRUE - Um RTC mit den angegebenen RTC-Daten zu aktivieren. FALSE - Um RTC zu deaktivieren.
RTCData	Eingang	RTC	RTC-Dateninformationen: JJ/mm/tt, Hh/mm/ss, Woche wie im Datentyp „RTC“ definiert. RTCData wird ignoriert, wenn RTCEnable = 0.
RTCPresent	Ausgang	BOOL	TRUE - Freilaufende Uhr wird verwendet oder RTC-Hardware ist angeschlossen. FALSE - Freilaufende Uhr wird nicht verwendet oder RTC-Hardware ist nicht angeschlossen.
RTCEnabled	Ausgang	BOOL	TRUE - Freilaufende Uhr wird verwendet oder RTC-Hardware ist aktiviert (Zeitgebung). FALSE - Freilaufende Uhr wird nicht verwendet oder RTC-Hardware ist nicht deaktiviert (keine Zeitgebung).
RTCBatLow	Ausgang	BOOL	TRUE - Ladestand der RTC-Batterie niedrig. FALSE - Ladestand der RTC-Batterie nicht niedrig.
Sts	Ausgang	USINT	Der Status des Lesevorgangs. RTC_Set-Statuswerte (Sts): • 0x00 – Funktionsblock nicht aktiviert (kein Vorgang). • 0 x 01 – RTC-Einstellvorgang erfolgreich. • 0 x 02 – RTC-Einstellvorgang nicht erfolgreich.

Datentyp RTC

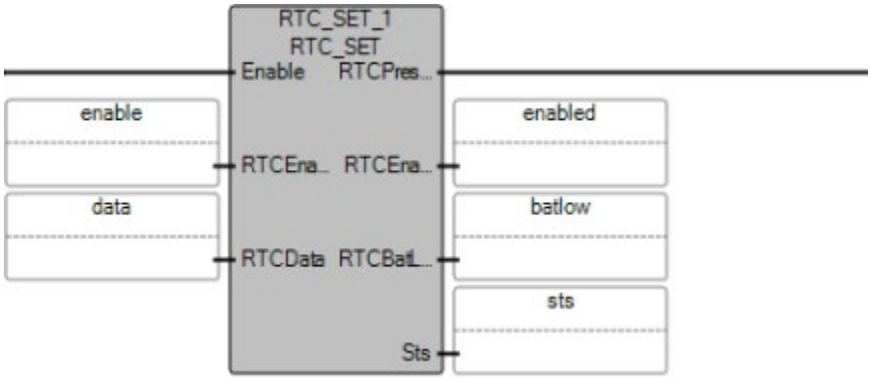
Verwenden Sie diese Tabelle, um die Parameterwerte für den Datentyp „RTC“ zu bestimmen.

Parameter	Datentyp	Beschreibung
Year	UINT	Die Jahreseinstellung für den RTC. 16-Bit-Wert, der gültige Bereich reicht von 2000 (01. Januar, 00:00:00) bis 2098 (31. Dezember, 23:59:59)
Month	UINT	Die Monatseinstellung für den RTC.
Day	UINT	Die Tageseinstellung für den RTC.
Hour	UINT	Die Stundeneinstellung für den RTC.
Minute	UINT	Die Minuteneinstellung für den RTC.
Second	UINT	Die Sekundeneinstellung für den RTC.
DayOfWeek	UINT	Die Wochentagseinstellung für den RTC. Dieser Parameter wird für „RTC_SET“ ignoriert.

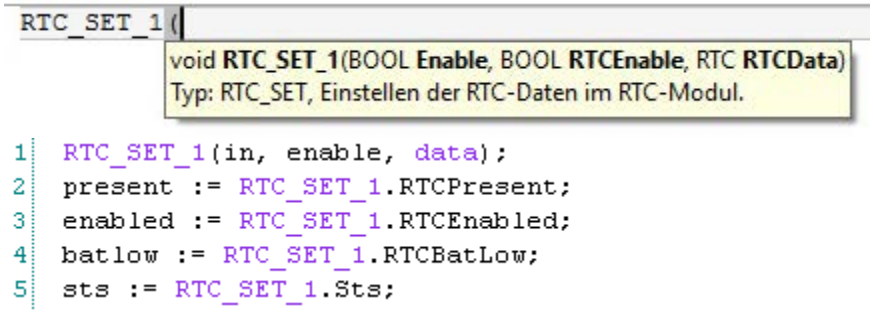
Beispiel für ein RTC_SET-Funktionsblockdiagramm



Beispiel für einen RTC_SET-Kontaktplan



Beispiel für einen strukturierten RTC_SET-Text



Socket-Anweisungen

Verwenden Sie das Sockets-Protokoll für die Ethernet-Kommunikation mit Geräten, die Modbus-TCP und EtherNet/IP nicht unterstützen. Sockets unterstützen Client, Server, TCP (Transmission Control Protocol) und UDP (User Datagram Protocol). Typische Anwendungen umfassen die Kommunikation mit Druckern, Strichcode-Lesegeräten und PCs.

Socket-Anweisungsverhalten bei **Änderungen im Betriebsmodus**:

- Wird ein Löschvorgang bei einer SOCKET_OPEN-, SOCKET_ACCEPT-, SOCKET_READ- oder SOCKET_WRITE-Anweisung im **Änderung im Betriebsmodus** ausgeführt, wird die Socket Instanz gelöscht.
- Im **Änderung im Betriebsmodus** bewirkt das Ändern einer SOCKET_READ-Eingabe, während Sie sich in der Zustand BUSY befinden, einen Fehler, und das empfangene Paket wird verworfen. SOCKET_READ-Eingabeparameter sind: Length, Offset, Data Array Size, Data Array Variable.
- SOCKET_READ ist die einzige SOCKET-Anweisung, das Hinzufügen und Ändern im **Änderung im Betriebsmodus** unterstützt.
- Wenn die Ethernet-IP-Einstellungen mit einer **Änderung im Betriebsmodus** geändert werden, werden alle erstellten Socket-Instanzen gelöscht, ähnlich wie bei SOCKET_DELETEALL.

Anweisungsverarbeitung und Ausgabeaktualisierungen für Socket-Anweisungen

- **Asynchron**: Entsprechende Anweisungen, laut denen alle Ausgaben asynchron mit Benutzerprogramm-Scans aktualisiert werden, z. B. ein Strompfad-Scan. Die asynchrone Ausgabe kann nicht für Flanken-Triggererkennung verwendet werden. Asynchrone Ausgabeparameter sind nicht gesperrt und können nach Abschluss der entsprechenden Socket-Anweisungen aktualisiert werden.
- **Synchron**: Entsprechende Anweisungen, laut denen alle Ausgaben synchron mit Benutzerprogramm-Scans aktualisiert werden. Synchrone Ausgabeparameter sind gesperrt und können nach Abschluss der entsprechenden Socket-Anweisungen nicht mehr geändert werden.
- **Kombination**: Entsprechende Anweisungen, laut denen einige Ausgaben synchron mit Benutzerprogramm-Scans aktualisiert werden. Verbleibende Ausgaben werden asynchron mit Benutzerprogramm-Scans aktualisiert.

- **Sofortige Ausführungsanweisung**: Anweisung zum Abschluss der gewünschten Funktion vor dem Wechsel zur nächsten Anweisung.
- **Nicht sofortige Ausführungsanweisung**: Anweisung braucht mehr als einen Programm-Scan, um die gewünschte Funktion durchzuführen. Anweisungen fertigen einen Schnappschuss der Eingabeparameter an, wenn sie einen Wechsel **False > True** erkennen.

Verwenden Sie diese Tabelle, um die Anwendung von Socket-Anweisungen zu bestimmen.

Anweisung	Beschreibung	TCP-Client	TCP-Server	UDP mit Öffnen	UDP ohne Öffnen	Anweisungsverarbeitung	Anweisungsausgabe-Aktualisierung
SOCKET_ACCEPT auf seite 566	Akzeptiert eine TCP-Verbindungsanforderung von einem Remote-Ziel und gibt eine Socket-Instanz zurück, die zum Senden und Empfangen von Daten in der neu erstellten Verbindung verwendet wird.	NO	Ja	NO	NO	Nicht sofort	Kombination
SOCKET_CREATE auf seite 570	Erstellt eine Instanz des Sockets und gibt eine Instanznummer zurück, die nachfolgende Socket-Vorgänge verwendet.	Ja	Ja	Ja	Ja	Sofort	Synchron
SOCKET_DELETE auf seite 575	Löscht eine erstellte Socketinstanz. TCP-Verbindungen werden vor dem Löschen geschlossen.	Ja	Ja	Ja	Ja	Nicht sofort	Synchron
SOCKET_DELETEALL auf seite 577	Löscht alle erstellten Socket-Instanzen.	Ja	Ja	Ja	Ja	Nicht sofort	Synchron
SOCKET_INFO auf seite 579	Gibt Informationen zum Socket wie z. B. Fehlercodes und Ausführungsstatus zurück.	Ja	Ja	Ja	Ja	Sofort	Synchron
SOCKET_OPEN auf seite 584	TCP-Verbindungen werden mit der angegebenen Zieladresse geöffnet. UDP-Verbindungen ordnen dem angegebenen Socket eine Ziel-IP-Adresse und Port-Nummer zu.	Ja	NO	Ja	NO	Nicht sofort	Synchron
SOCKET_READ auf seite 588	Liest Daten von einem Socket. Versucht, die angegebene Anzahl der Bytes zu empfangen, und gibt die Anzahl der empfangenen Bytes zurück.	Ja	Ja	Ja	Ja	Nicht sofort	Kombination
SOCKET_WRITE auf seite 592	Sendet Daten an einen Socket. Versucht, die angeforderte Anzahl der Bytes zu senden, und gibt die Anzahl der gesendeten Bytes zurück.	Ja	Ja	Ja	Ja	Nicht sofort	Kombination

SOCKET_ACCEPT

Nur für Transmission Control Protocol-Verbindungen (TCP). Akzeptiert eine TCP-Verbindungsanforderung von einem Remote-Ziel und gibt eine Socket-Instanz zurück, die zum Senden und Empfangen von Daten in der neu erstellten Verbindung verwendet wird.

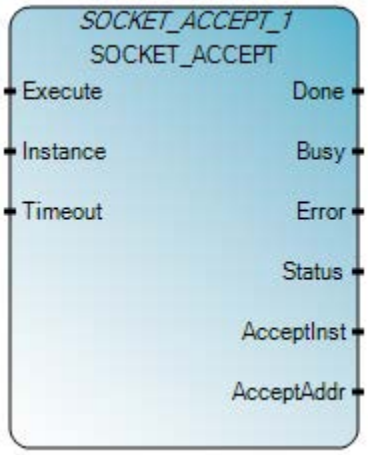
Details zum Betrieb:

- Bevor Sie „Socket_Accept“ ausführen, müssen Sie SOCKET_CREATE ausführen und die lokale Portnummer angeben, um die Verbindung zu akzeptieren.
- Ausgaben werden synchron vom Programm-Scan aktualisiert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro850 und Micro870. Für den Micro800-Simulator kann diese Anweisung zu einem Programm hinzugefügt werden. Sie dient aber lediglich als Platzhalter, um zu verhindern, dass die Anweisung bei der Controller-Änderung gelöscht wird.

Die Ausgänge werden jedes Mal zurückgesetzt, wenn die Anweisung auf die simulierte Steuerung (2080-LC50-48QWB-SIM) angewendet wird.

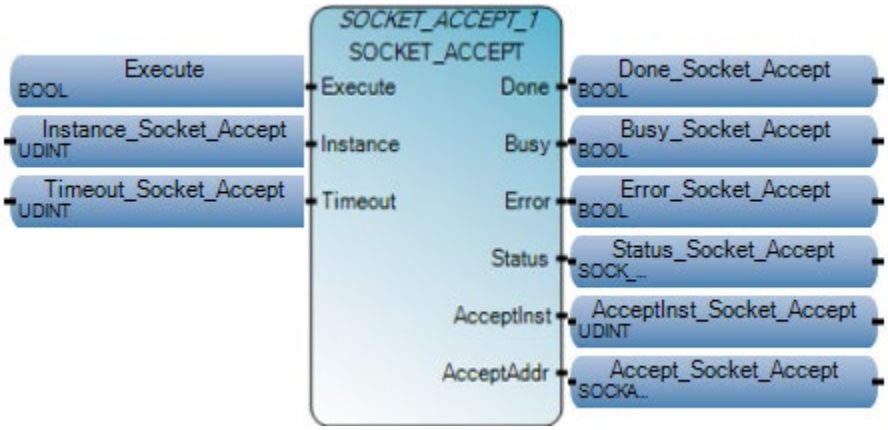


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

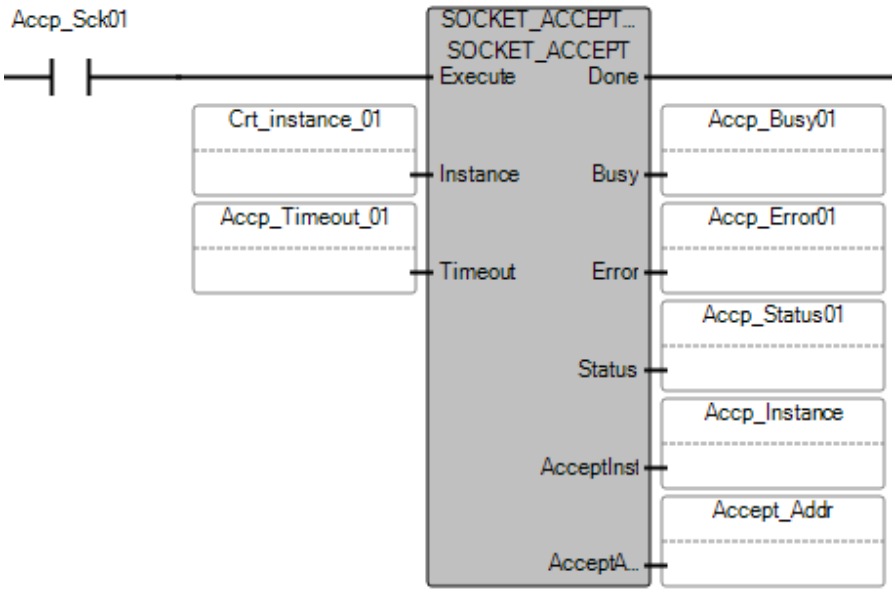
Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Anstiegsflanke erkannt. Startet den Anweisungsblock unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE – Keine Anstiegflanke erkannt; Befehlsblock nicht gestartet.
Instance	Eingang	UDINT	Identifiziert die Socket-Instanz. Kopieren vom zurückgegebenen Socket-Handler aus einer **SOCKET_CREATE**-Anweisung.
Timeout	Eingang	UDINT	Zeitüberschreitung für **SOCKET_ACCEPT**-Anweisungsblockinstanzen. Der Funktionsblock gibt einen Fehler zurück, wenn der Zeitüberschreitungswert geringer als der Mindestwert ist. Zeitüberschreitungsbereich: 1000- 86400000 Millisekunden Einstellen von Zeitüberschreitung auf 0 zur Verwendung des Standardwerts 10000 (10 Sekunden).
Done	Ausgang	BOOL	Gibt an, wenn der Vorgang abgeschlossen ist. TRUE – Vorgang erfolgreich abgeschlossen. FALSE – Vorgang wird ausgeführt oder eine Fehlerbedingung ist aufgetreten. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Busy	Ausgang	BOOL	TRUE – Vorgang ist nicht abgeschlossen. FALSE – Vorgang ist abgeschlossen. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.

Parameter	Parametertyp	Datentyp	Beschreibung
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Status	Ausgang	SOCK_STATUS	Der Status wird mit dem Datentyp SOCK_STATUS auf seite 596 definiert, der ErrorID auf seite 597-, SubErrorID- und StatusBits auf seite 601-Informationen enthält. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
AcceptInst	Ausgang	UDINT	Enthält die Annahmeinstanz für diese Socket-Instanz. Verwenden Sie diese einzigartige Annahmeinstanznummer mit aufeinander folgendem **SOCKET_READ** und **SOCKET_WRITE** für diese Verbindung. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
AcceptAddr	Ausgang	SOCKADDR_CFG	Eine Datenstruktur, die die Annahmeadresse für den Socket enthält. Weitere Informationen finden Sie unter Datentyp SOCKADDR_CFG auf seite 596. Zur Angabe einer IP-Adresse von 192.168.2.100 und Port = 12000: • AcceptAddr.IPAddress[0]=192 • AcceptAddr.IPAddress[1]=168 • AcceptAddr.IPAddress[2]=2 • AcceptAddr.IPAddress[3]=100 • AcceptAddr.Port = 12000 Die Ausgabe wird synchron vom Programm-Scan aktualisiert.

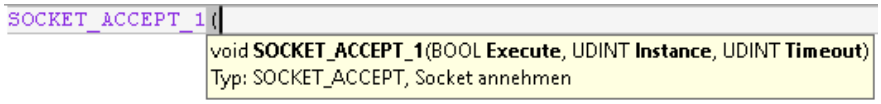
Beispiel für ein SOCKET_ACCEPT-Funktionsblockdiagramm



Beispiel für einen SOCKET_ACCEPT-Kontaktplan



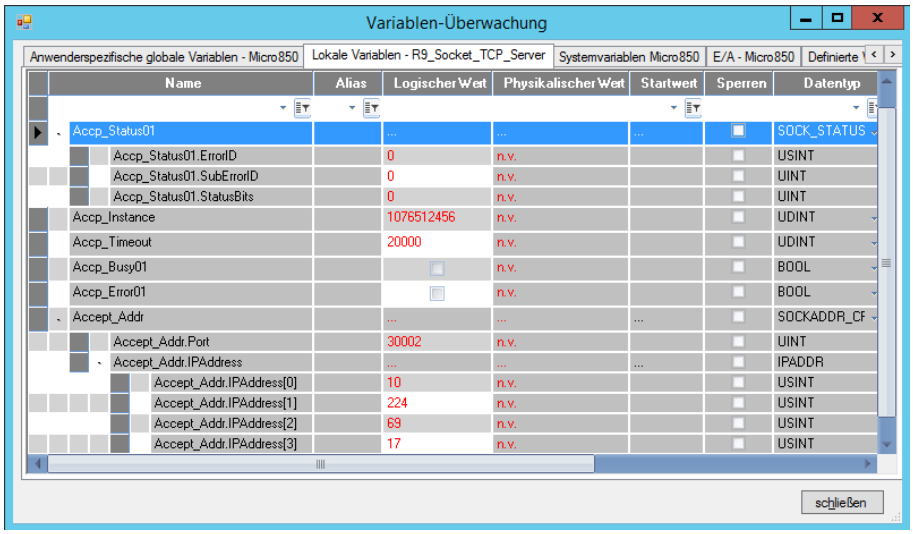
Beispiel für einen strukturierten SOCKET_ACCEPT-Text




```
SOCKET_ACCEPT_1(Execute_SOCKET_ACCEPT, Instance _SOCKET_ACCEPT, Timout_SOCKET_ACCEPT);
Done_SOCKET_ACCEPT :- SOCKET_ACCEPT. Done;
Busy_SOCKET_ACCEPT :- SOCKET_ACCEPT. Busy;
Error_SOCKET_ACCEPT :- SOCKET_ACCEPT. Error;
Status_SOCKET_ACCEPT :- SOCKET_ACCEPT. Status;
AcceptInst_SOCKET_ACCEPT :- SOCKET_ACCEPT. AcceptInst;
AcceptAddr_SOCKET_ACCEPT :- SOCKET_ACCEPT. AccptAddr;
```

Ergebnisse

TCP-Server



SOCKET_CREATE

Erstellt eine Instanz des Sockets und gibt eine Instanznummer zurück, die als Eingabe für nachfolgende Socket-Vorgänge verwendet wird.

Details zum Betrieb:

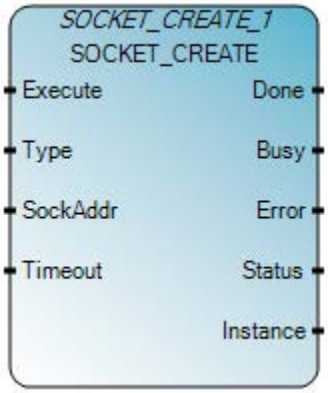
- Socket-Befehle auf seite 565 unterstützen Vollduplex-Kommunikation mit Remote-Geräten.
- User Datagram Protocol (UDP)-Verbindungen unterstützen maximal acht UDP-Datagram-Pakete in der Warteschlange. Die Warteschlange enthält die aktuellsten Pakete.
- Micro820- und Micro850-Controller mit Revision 9 oder höher unterstützen bis zu acht Socket-Instanzen. Die Socket-Instanzen unterstützen UDP- und TCP-Sockets:
  - Verwenden Sie alle acht Instanzen für Client-seitige Transmission Control Protocol (TCP)-Verbindungen.
  - Verwenden Sie alle acht Instanzen, um eine Überwachung auf eingehende TCP-Verbindungen vorzunehmen und akzeptieren Sie anschließend acht Verbindungen von anderen Geräten.
  - Führen Sie TCP-Client- und Server-Vorgänge durch.
  - Führen Sie TCP- und UDP-Vorgänge durch.
- Um eingehende TCP-Verbindungen zu demselben Port zu akzeptieren, müssen Sie eine neue Socket-Instanz erstellen.
- TCP-Verbindungsunterbrechung:
  - Das Benutzer-Anwendungsprogramm sollte die Unterbrechung der TCP-Verbindungen erkennen und das Ereignis verarbeiten. Je nach Benutzeranwendung kann die Option einen Controller-Fehler bewirken.
    - Fehler des Controllers
    - Versuchen Sie, die Verbindung erneut herzustellen.
  - So stellen Sie die Kommunikation mit einem anderen Gerät wieder her:
    - Löschen Sie die Socket-Instanz für die unterbrochene Verbindung.
    - Wenn es sich bei der Verbindung um einen TCP-Client handelt, erstellen Sie mithilfe von SOCKET_CREATE eine neue Socket-Instanz und führen Sie SOCKET_OPEN auf dem Zielgerät aus.
    - Wenn es sich bei der Verbindung um einen TCP-Server handelt, erstellen Sie mithilfe von SOCKET_CREATE eine neue Socket-Instanz und führen Sie SOCKET_ACCEPT aus, um auf eine weitere Verbindung vom Remote-Gerät zu warten.
  - Anwendungsmeldungen für TCP-Verbindungen:
    - Eine TCP-Verbindung ist ein Byte-Datenstrom zwischen zwei Anwendungen. Das Anwendungsprotokoll ermittelt die Meldungsformate.

- Meldungen können eine feste oder variable Größe aufweisen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro850 und Micro870. Für den Micro800-Simulator kann diese Anweisung zu einem Programm hinzugefügt werden. Sie dient aber lediglich als Platzhalter, um zu verhindern, dass die Anweisung bei der Controller-Änderung gelöscht wird.

Die Ausgänge werden jedes Mal zurückgesetzt, wenn die Anweisung auf die simulierte Steuerung (2080-LC50-48QWB-SIM) angewendet wird.

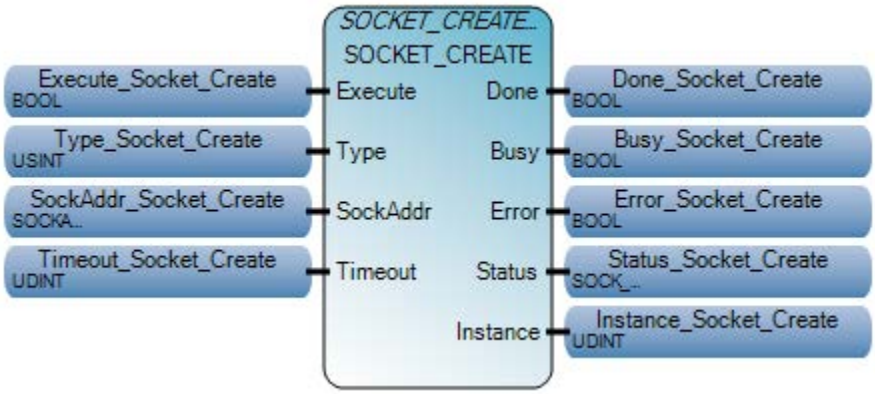


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

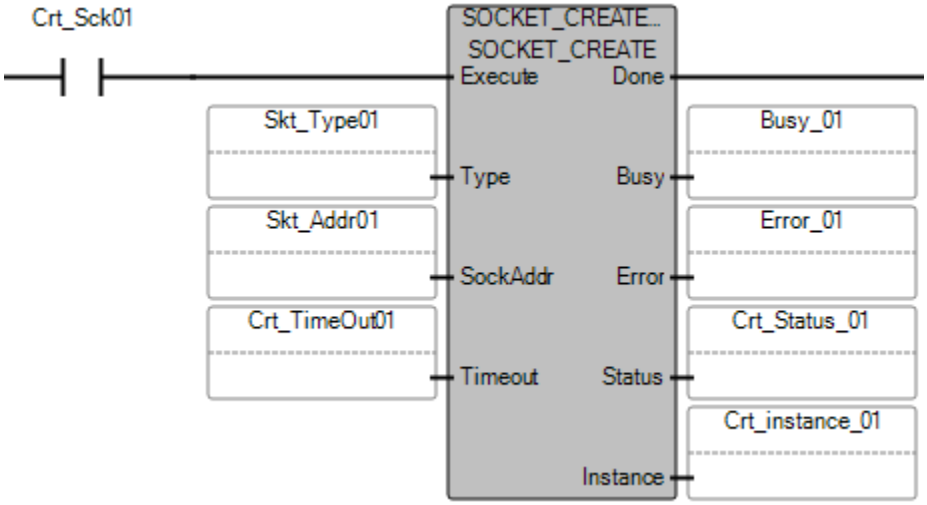
Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Anstiegsflanke erkannt. Startet den Anweisungsblock unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE – Keine Anstiegsflanke erkannt.
Type	Eingang	USINT	Angeben des Socket-Typs: • Transmission Control Protocol (TCP) • User Datagram Protocol (UDP)
SockAddr	Eingang	SOCKADDR_CFG	Legt die Adresskonfiguration des Socket fest. Das EtherNet/IP-Modul, um die lokale Port-Nummer auswählen, „SockAddr" wird auf 0 gesetzt. Geben Sie die lokale Port-Nummer an, bei der eine Anwendung überwacht und empfängt, oder:. • Alle Array-Elemente müssen Null sein • Für TCP-Client-Vorgänge 0 angeben, sofern keine spezifische lokale Portnummer gewünscht ist. • Für die TCP-Serverkommunikation die Portnummer angeben, um eingehende Verbindungsanfragen zu akzeptieren. • Für UDP eine lokale Portnummer festlegen, um Datagramme an einem bestimmten Port zu empfangen. Lokaler Portbereich: 1 bis 65535. Ein Fehler tritt auf, wenn die angegebene lokale Portnummer bereits vom Micro820- oder Micro850-Controller verwendet wird. Die Steuerung nutzt die folgenden Portnummern: TCP-Ports:

Parameter	Parametertyp	Datentyp	Beschreibung
			• EtherNet/IP: 44818 • ModbusTCP: 502 • DHCP-Server: 67 • DHCP-Client: 68 UDP-Ports: • EtherNet/IP: 2222 • DHCP-Server: 67 • DHCP-Client: 68 Siehe unter Datentyp SOCKADDR_CFG auf seite 596.
Timeout	Eingang	UDINT	Geben Sie das Timeout für die Socket-Inaktivität an. Wenn eine Socket-Instanz innerhalb einer bestimmten Inaktivitätszeitüberschreitung keine Anforderungen erhält, wird die Socket-Instanz gelöscht. Wenn nach dem Löschen der Socket-Instanz eine Anforderung gesendet wird, wird der Fehler „Socket-Instanz nicht unterstützt“ zurückgegeben. Der Anweisungsblock gibt einen Fehler zurück, wenn der Zeitüberschreitungswert geringer als der Mindestwert ist. Setzen Sie die Zeitüberschreitung so fest, dass sie länger als das längste Intervall zwischen den Socket-Vorgängen ist. Wenn die Inaktivitätszeitüberschreitung zu kurz ist, tritt ggf. eine Zeitüberschreitung der Socket-Instanzen auf. Zeitüberschreitungsbereich: 1.000 - 86.400.000 Millisekunden Setzen Sie die **Zeitüberschreitung** auf 0 zur Verwendung des Standardwerts 300000 (5 Minuten).
Done	Ausgang	BOOL	Gibt an, wenn der Vorgang abgeschlossen ist. TRUE – Vorgang erfolgreich abgeschlossen. FALSE – Vorgang wird ausgeführt oder eine Fehlerbedingung ist aufgetreten.
Busy	Ausgang	BOOL	TRUE – Vorgang ist nicht vollständig abgeschlossen. FALSE – Vorgang ist vollständig abgeschlossen.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler.
Status	Ausgang	SOCK_STATUS	Der Status wird mit dem Datentyp **SOCK_STATUS** definiert, der ErrorID-, SubErrorID- und StatusBits-Informationen enthält. Siehe unter Datentyp SOCK_STATUS auf seite 596, Socket-Befehlsstatusbits auf seite 601 und Socket-Fehlercodes auf seite 597.
Instance	Ausgang	UDINT	Enthält den Socket-Handler. Den Instanzparameter für nachfolgende Socket-Anweisungen verwenden.

Beispiel für ein SOCKET_CREATE-Funktionsblockdiagramm



Beispiel für einen SOCKET_CREATE-Kontaktplan



Beispiel für einen strukturierten SOCKET_CREATE-Text

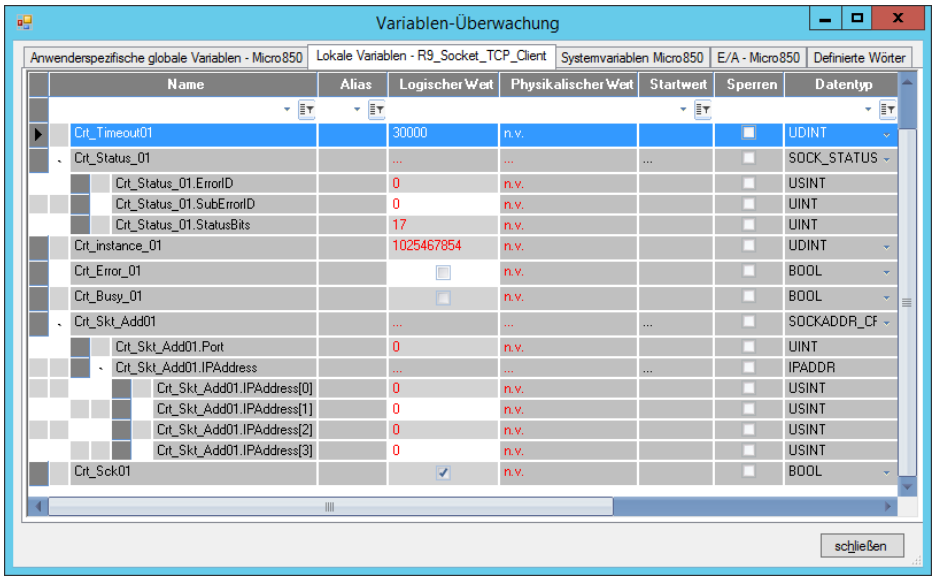
```
SOCKET_CREATE_1(
```

void **SOCKET_CREATE_1**(BOOL **Execute**, USINT **Type**, SOCKADDR_CFG **SockAddr**, UDINT **Timeout**)
Typ: SOCKET_CREATE, Socket erstellen

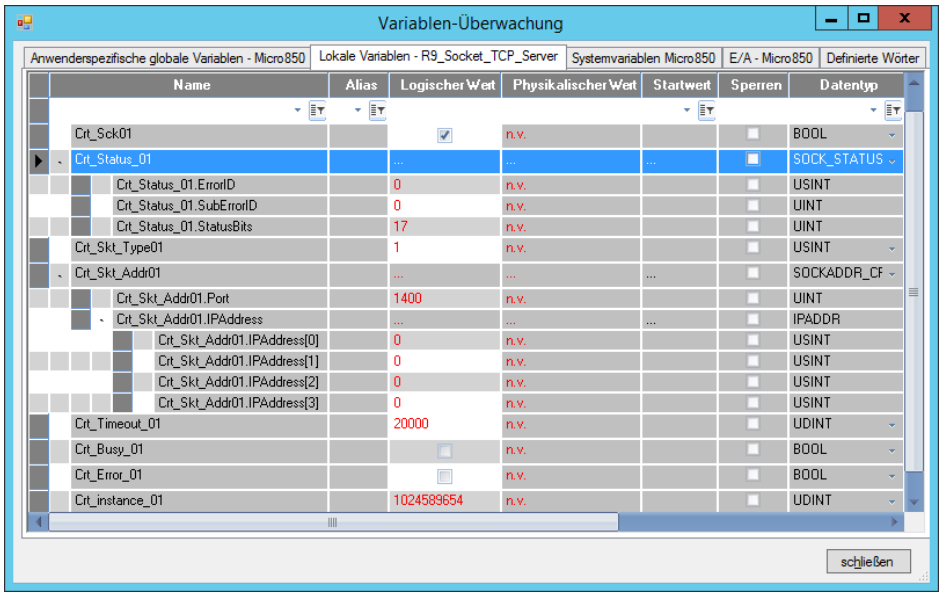
```
SOCKET_CREATE_1 (Execute_SOCKET_CREATE, Type_SOCKET_CREATE, SockAddr_SOCKET_CREATE, Timeout_SOCKET_CREATE);
Done_SOCKET_CREATE := SOCKET_CREATE.Done;
Busy_SOCKET_CREATE := SOCKET_CREATE.Busy;
Error_SOCKET_CREATE := SOCKET_CREATE.Error;
Status_SOCKET_CREATE := SOCKET_CREATE.Status;
Instance_SOCKET_CREATE := SOCKET_CREATE.Instance;
```

Ergebnisse

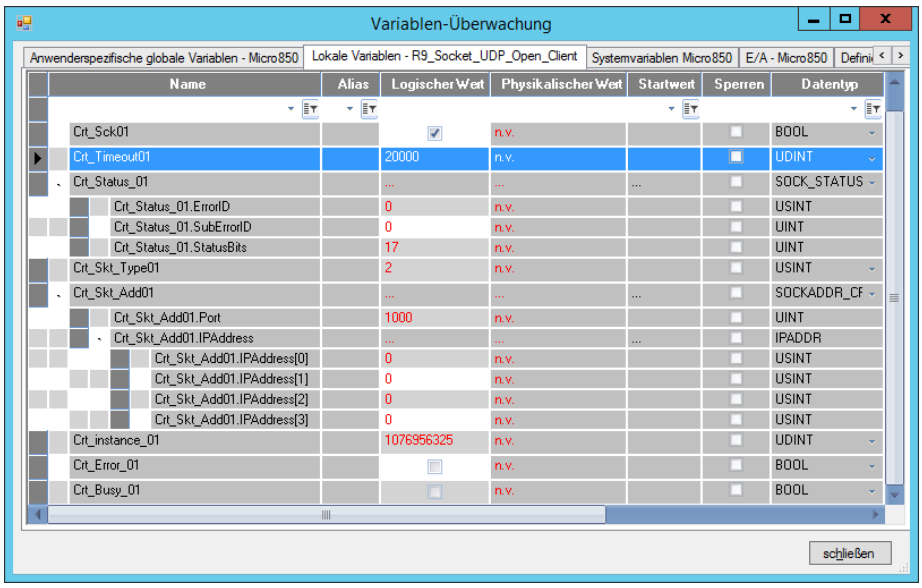
TCP-Client-Beispiel



TCP-Server-Beispiel



UDP-Beispiel



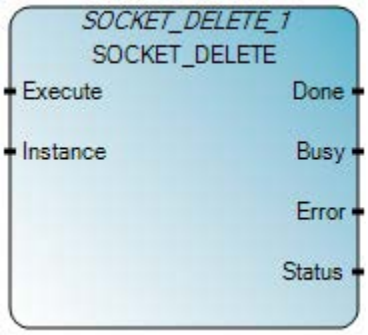
SOCKET_DELETE

Löscht eine erstellte Socketinstanz. Für Transmission Control Protocol-Verbindungen (TCP) trennt (passives Trennen) **SOCKET_DELETE** ebenfalls die Verbindung, bevor die Instanz gelöscht wird. Ausgaben werden synchron vom Programm-Scan aktualisiert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro850 und Micro870. Für den Micro800-Simulator kann diese Anweisung zu einem Programm hinzugefügt werden. Sie dient aber lediglich als Platzhalter, um zu verhindern, dass die Anweisung bei der Controller-Änderung gelöscht wird.

Die Ausgänge werden jedes Mal zurückgesetzt, wenn die Anweisung auf die simulierte Steuerung (2080-LC50-48QWB-SIM) angewendet wird.

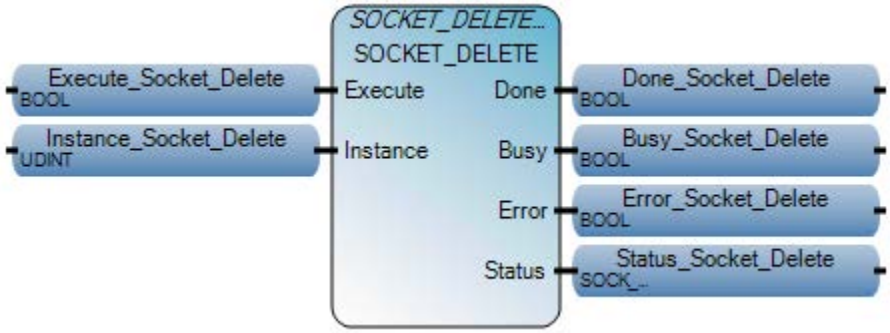


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

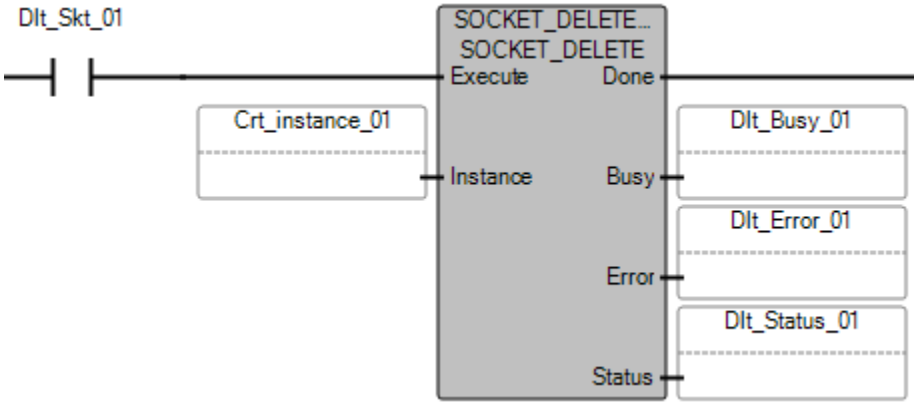
Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Anstiegsflanke erkannt. Startet den Anweisungsblock unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE – Keine Anstiegsflanke erkannt.

Parameter	Parametertyp	Datentyp	Beschreibung
Instance	Eingang	UDINT	Kopieren Sie den zurückgegebenen Socket-Handler einer SOCKET_CREATE- oder SOCKET_ACCEPT-Anweisung, um den entsprechenden Socket zu löschen. • Kopieren Sie für die Socket-Typen von UDP- und TCP-Client Socket den zurückgegebenen Socket-Handler aus einer SOCKET_CREATE-Anweisung. • Kopieren Sie für den Socket-Typ des TCP-Servers vom zurückgegebenen Socket-Handler aus einer SOCKET_ACCEPT-Anweisung.
Done	Ausgang	BOOL	Gibt an, wenn der Vorgang abgeschlossen ist. TRUE – Vorgang erfolgreich abgeschlossen. FALSE – Vorgang wird ausgeführt oder eine Fehlerbedingung ist aufgetreten. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Busy	Ausgang	BOOL	TRUE – Vorgang ist nicht abgeschlossen. FALSE – Vorgang ist abgeschlossen. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Status	Ausgang	SOCK_STATUS	Der Status wird mit dem Datentyp SOCK_STATUS auf seite 596 definiert, der ErrorID auf seite 597-, SubErrorID- und StatusBits auf seite 601-Informationen enthält. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.

Beispiel für ein SOCKET_DELETE-Funktionsblockdiagramm BSR



Beispiel für einen SOCKET_DELETE-Kontaktplan



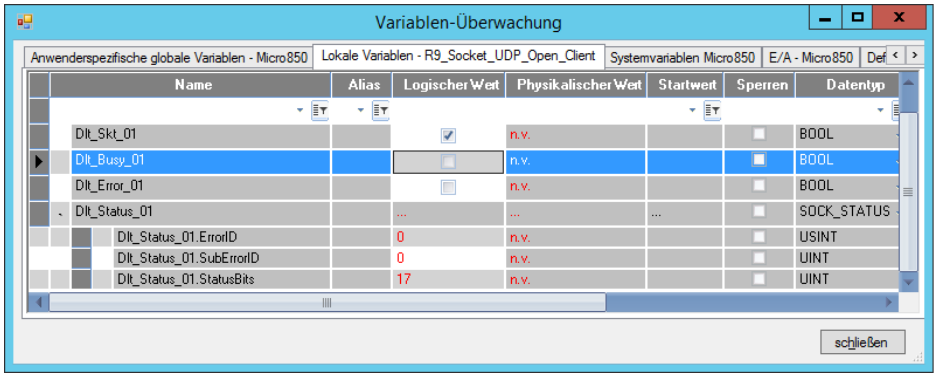
Beispiel für einen strukturierten SOCKET_DELETE-Text

```
SOCKET_DELETE_1(
```

void **SOCKET_DELETE_1**(BOOL **Execute**, UDINT **Instance**)
Typ: SOCKET_DELETE, Socket löschen

```
SOCKET_DELETE_1 (Execute_SOCKET_DELETE, Instance_SOCKET_DELETE);
Done_SOCKET_DELETE := SOCKET_DELETE. Done;
Busy_SOCKET_DELETE := SOCKET_DELETE. Busy;
Error_SOCKET_DELETE := SOCKET_DELETE. Error;
Status_SOCKET_DELETE := SOCKET_DELETE. Status;
```

Ergebnisse



SOCKET_DELETEALL

Löscht alle erstellten Socket-Instanzen.

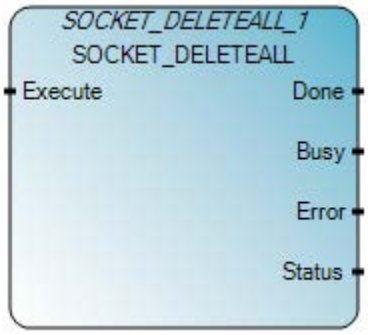
Details zum Betrieb:

- Wenn das Ethernet-Kabel vom Controller abgezogen oder die Controller-IP-Adresse geändert wird, können Sie SOCKET_DELETEALL ausführen, um alle zuvor erstellten Socket-Instanzen zu löschen.
- Ausgaben werden synchron vom Programm-Scan aktualisiert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro850 und Micro870. Für den Micro800-Simulator kann diese Anweisung zu einem Programm hinzugefügt werden. Sie dient aber lediglich als Platzhalter, um zu verhindern, dass die Anweisung bei der Controller-Änderung gelöscht wird.

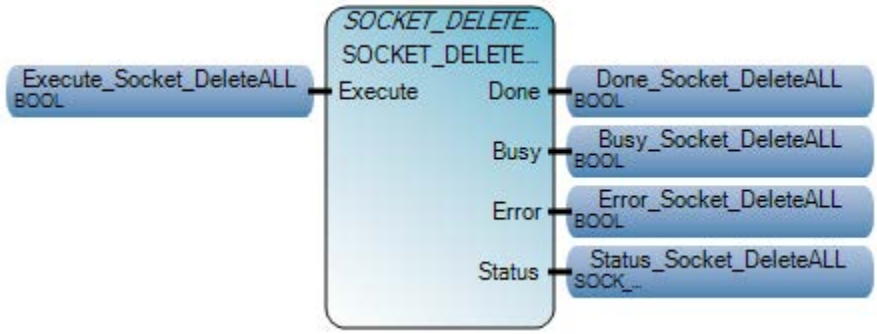
Die Ausgänge werden jedes Mal zurückgesetzt, wenn die Anweisung auf die simulierte Steuerung (2080-LC50-48QWB-SIM) angewendet wird.



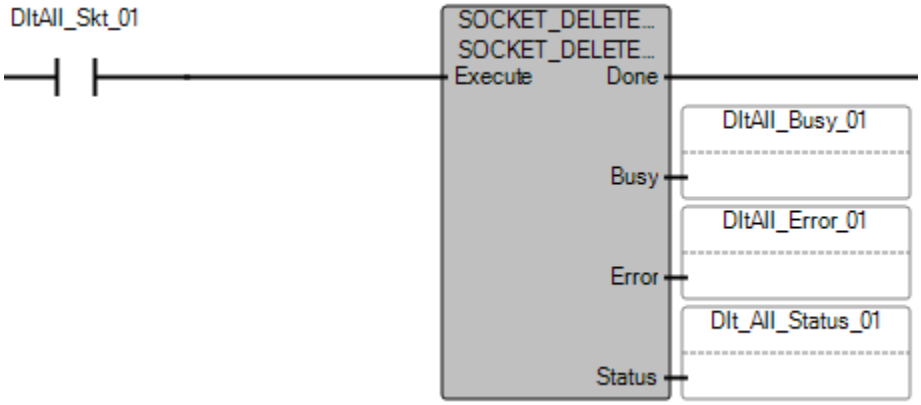
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Anstiegsflanke erkannt. Startet den Anweisungsblock unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE – Keine Anstiegsflanke erkannt.
Done	Ausgang	BOOL	Gibt an, wenn der Vorgang abgeschlossen ist. TRUE – Vorgang erfolgreich abgeschlossen. FALSE – Vorgang wird ausgeführt oder eine Fehlerbedingung ist aufgetreten. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Busy	Ausgang	BOOL	TRUE – Vorgang ist nicht abgeschlossen. FALSE – Vorgang ist abgeschlossen. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Error	Ausgang	BOOL	Dieses Feld wird auf TRUE gesetzt, wenn bei der Ausführung des Funktionsblocks eine Fehlerbedingung auftritt. Weitere Informationen finden Sie unter Socket-Fehlercodes auf seite 597. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Status	Ausgang	SOCK_STATUS	Der Status wird mit dem Datentyp SOCK_STATUS auf seite 596 definiert, der ErrorID auf seite 597-, SubErrorID- und StatusBits auf seite 601-Informationen enthält. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.

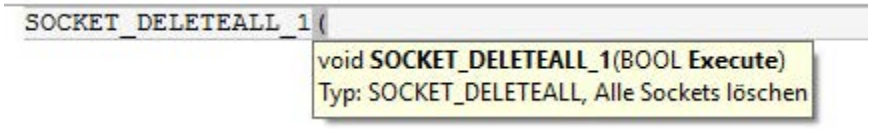
Beispiel für ein SOCKET_DELETEALL-Funktionsblockdiagramm



Beispiel für einen SOCKET_DELETEALL-Kontaktplan

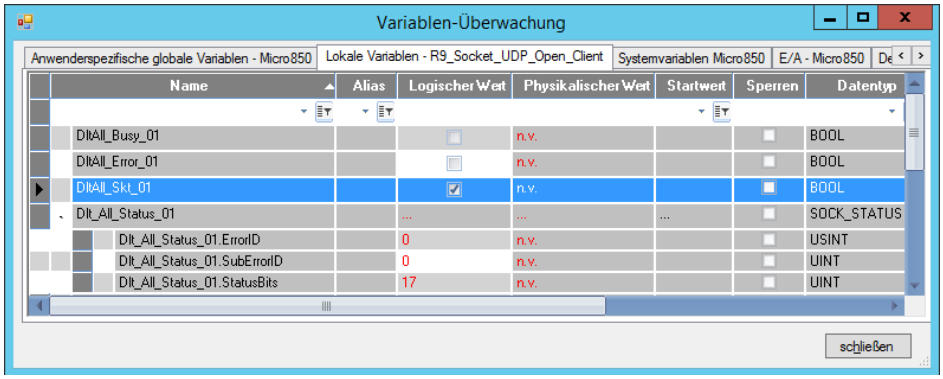


Beispiel für einen strukturierten SOCKET_DELETEALL-Text




```
SOCKET_DELETEALL (Execute_SOCKET_DELETEALL);
Done_Socket_DeleteAll := SOCKET_DELETE. Done;
Busy_Socket_DeleteAll := SOCKET_DELETE. Busy;
Error_Socket_DeleteAll := SOCKET_DELETE. Error;
Status_Socket_DeleteAll := SOCKET_DELETE. Status;
```

Ergebnisse



SOCKET_INFO

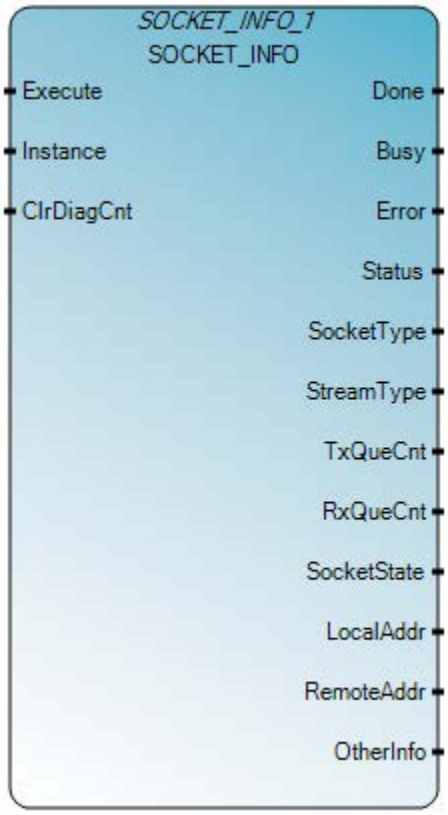
Gibt Informationen für eine Socket-Instanz zurück, wie z. B. Fehlercodes und den Ausführungsstatus. Die Ausgänge werden synchron vom Programm-Scan aktualisiert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro850 und Micro870. Für den Micro800-Simulator kann diese Anweisung zu einem Programm

hinzugefügt werden. Sie dient aber lediglich als Platzhalter, um zu verhindern, dass die Anweisung bei der Controller-Änderung gelöscht wird.

Die Ausgänge werden jedes Mal zurückgesetzt, wenn die Anweisung auf die simulierte Steuerung (2080-LC50-48QWB-SIM) angewendet wird.



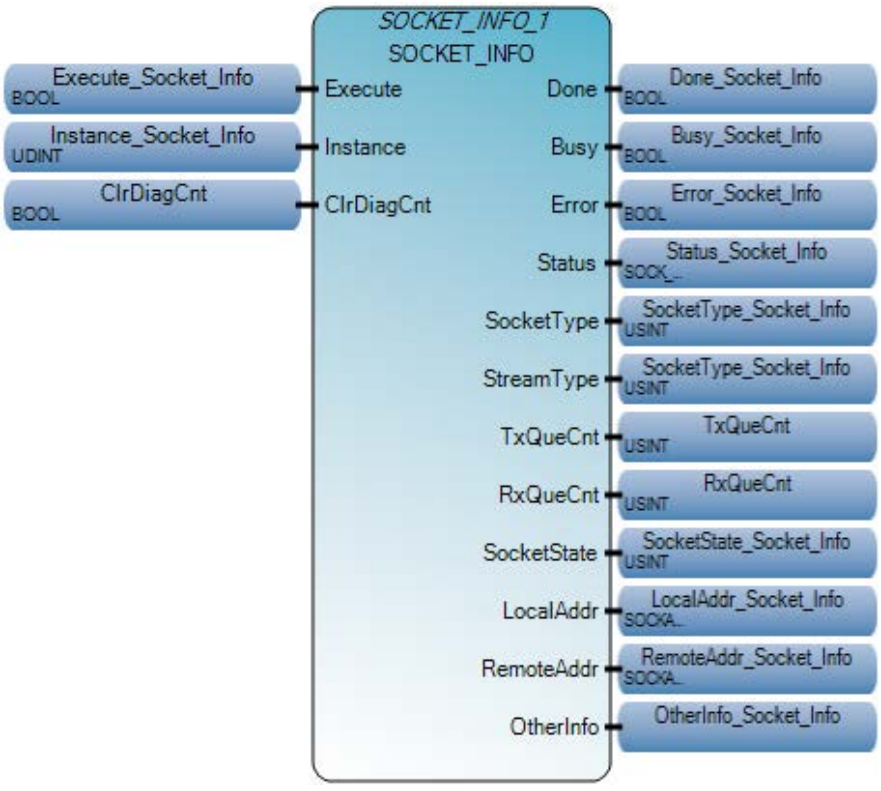
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Anstiegsflanke erkannt. Startet den Anweisungsblock unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE – Keine Anstiegsflanke erkannt.
Instance	Eingang	UDINT	Kopieren Sie den zurückgegebenen Socket-Handler einer **SOCKET_CREATE**- oder **SOCKET_ACCEPT**-Anweisung, um den entsprechenden Socket zu löschen. • Kopieren Sie für die Socket-Typen von UDP- und TCP-Client Socket den zurückgegebenen Socket-Handler aus einer **SOCKET_CREATE**-Anweisung. • Kopieren Sie für den Socket-Typ des TCP-Servers vom zurückgegebenen Socket-Handler aus einer **SOCKET_ACCEPT**-Anweisung. Wenn Instance = 0 ist, wird eine Zusammenfassung aller Socket-Instanzen zurückgegeben.
ClrDiagCnt	Eingang	BOOL	TRUE – Zählerinformationen der Diagnose des Transmission Control Protocol (TCP) oder User Datagram Protocol (UDP) werden gelöscht. FALSE – Kein Löschen der TCP / UDP-Zähler-Informationen. Beispiel: TCP und UDP, **OtherInfo** Array-Index 1-6. Wenn Instance = 0 ist, wird **OtherInfo** Array-Index 7-14 gelöscht.
Done	Ausgang	BOOL	Gibt an, wenn der Vorgang abgeschlossen ist. TRUE – Vorgang erfolgreich abgeschlossen. FALSE – Vorgang wird ausgeführt oder eine Fehlerbedingung ist aufgetreten.

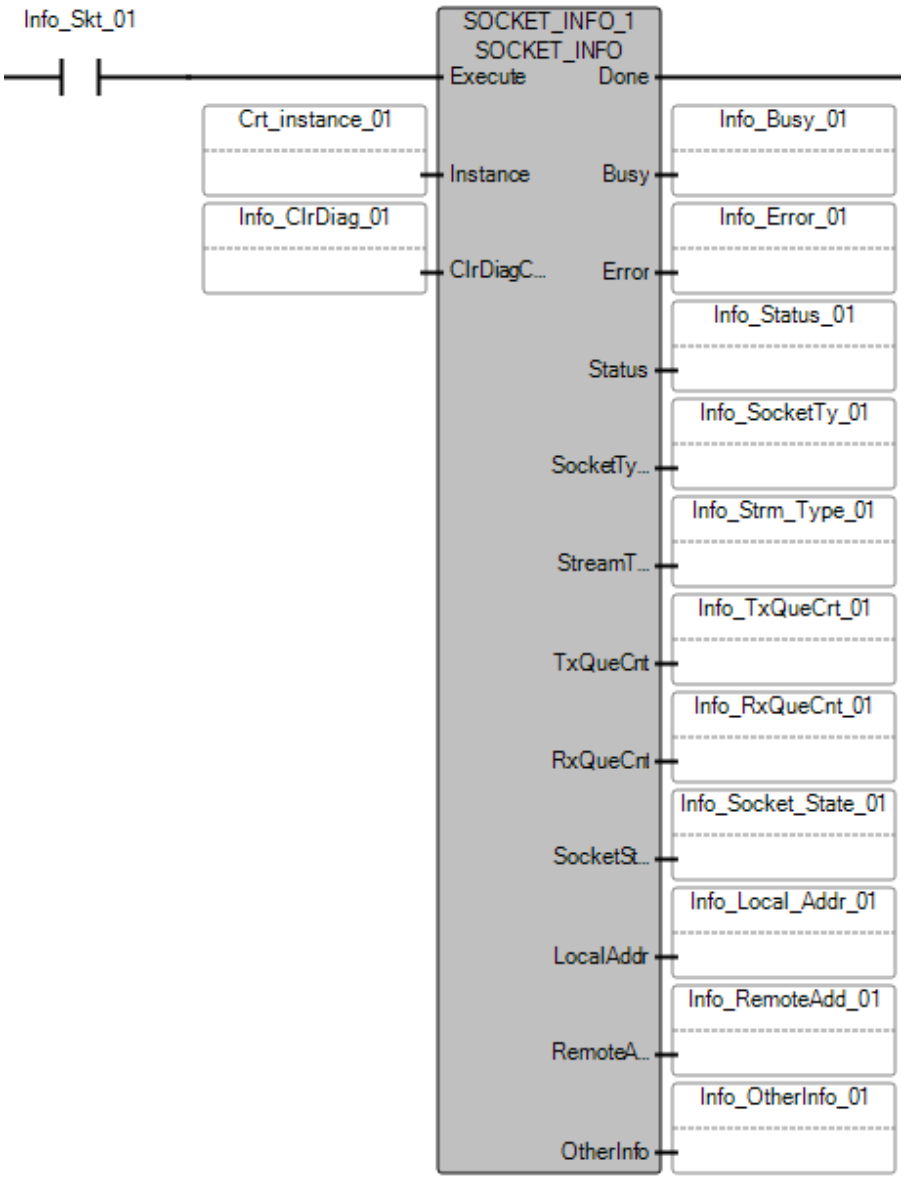
Parameter	Parametertyp	Datentyp	Beschreibung
			Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Busy	Ausgang	BOOL	TRUE – Vorgang ist nicht abgeschlossen. FALSE – Vorgang ist abgeschlossen. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Status	Ausgang	SOCK_STATUS	Der Status wird mit dem Datentyp SOCK_STATUS auf seite 596 definiert, der ErrorID auf seite 597-, SubErrorID- und StatusBits auf seite 601-Informationen enthält. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
SocketType	Ausgang	USINT	Socket-Instanztyp: • 0 - Nicht verwendet • 1 - TCP • 2 - UDP Wenn Socket_Info **Instance** = 0 ist, wird **SocketType** als 0 angezeigt.
StreamType	Ausgang	USINT	Socket-Streamtyp: • 0 - Kein • 1 - TCP-Server • 2 - TCP-Client Wenn Socket_Info **Instance** = 0, wird **StreamType** als 0 angezeigt.
TxQueCnt	Ausgang	USINT	Anzahl von sich aktuell in der Warteschlange befindlichen Tx-Nachrichten. Wenn Socket_Info **Instance** = 0 ist, wird **TxQueCnt** als 0 angezeigt.
RxQueCnt	Ausgang	USINT	Anzahl von sich aktuell in der Warteschlange befindlichen Rx-Nachrichten. Wenn Socket_Info **Instance** = 0 ist, wird **RxQueCnt** als 0 angezeigt.
SocketState	Ausgang	USINT	Socket-Anweisungsstatusinformationen. Weitere Informationen siehe Socket-Maschinenstatus. Wenn Socket_Info **Instance** = 0 ist, wird **SocketState** als 0 angezeigt.
LocalAddr	Ausgang	SOCKETADDR_CFG	Lokale Adresse des Sockets. Weitere Informationen finden Sie unter Datentyp SOCKADDR_CFG auf seite 596. Wenn Socket_Info **Instance** = 0 ist, wird **LocalAddr** als 0 angezeigt.
RemoteAddr	Ausgang	SOCKETADDR_CFG	Remote-Adresse des Sockets. Weitere Informationen finden Sie unter Datentyp SOCKADDR_CFG auf seite 596. **RemoteAddr** wird in folgenden Fällen als 0 angezeigt: • Socket_Info **Instance** ist 0 • User Datagram Protocol-Verbindungen (UDP) ohne **SOCKET_OPEN** • UDP mit **SOCKET_OPEN** und deaktiviertem **RxFilter**.
OtherInfo		UDINT[1..15]	Socket-**Instance** als TCP konfiguriert, Array-Indexbeschreibung lautet: • 1 - Gesendete Pakete: Gesamtanzahl von über einen Socket gesendete TCP-Pakete. • 2 - Gesendete Byte: Gesamtanzahl von über einen Socket gesendete TCP-Byte. • 3 - Empfangene Pakete: Gesamtanzahl von über einen Socket empfangene TCP-Pakete. • 4 - Empfangene Byte: Gesamtanzahl von über einen Socket empfangene TCP-Byte. • 5 - Erneut übermittelte Pakete: Gesamtanzahl von erneut übermittelten TCP-Paketen. • 6 - Prüfsummenfehler: Gesamtanzahl von TCP-Paketen mit Prüfsummenfehlern auf einem Socket. • 7 - TCP-Status: Aktueller Status eines Sockets. • (8-11) – OtherInfo wird für TCP nicht unterstützt, es wird als 0 angezeigt. • 12 ,13,14,15 - werden als 0 angezeigt. Socket-Instanz als UDP konfiguriert, Array-Indexbeschreibung lautet: • 1 - Gesendete Pakete: Gesamtanzahl von über einen Socket gesendete UDP-Pakete.

Parameter	Parametertyp	Datentyp	Beschreibung
			• 2 - Gesendete Byte: Gesamtanzahl von über einen Socket gesendete UDP-Byte. • 3 - Empfangene Pakete: Gesamtanzahl von über einen Socket empfangene UDP-Pakete. • 4 - Empfangene Byte: Gesamtanzahl von über einen Socket empfangene UDP-Byte. • 5 - Gelöschte Pakete: Gesamtanzahl von empfangenen UDP-Pakten, die auf einem Socket aufgrund einer Überschreitung der maximalen Warteschlangengröße von 8 gelöscht wurden. • 6 - Prüfsummenfehler: Gesamtanzahl von UDP-Paketen mit Prüfsummenfehlern auf dem Socket. • 7-15 - werden als 0 angezeigt. Socket-Instanz als 0 konfiguriert, Array-Indexbeschreibung lautet: • 1 - Anzahl verfügbarer Socket-Instanzen. Maximale Anzahl unterstützter Sockets. • 2 - Anzahl verwendeter Socket-Instanzen. Anzahl erfolgreich erstellter Sockets. • 3 - Anzahl von als TCP erstellter Socket-Instanzen. • 4 - Anzahl von als TCP-Client erstellter Socket-Instanzen. • 5 - Anzahl von als TCP-Server erstellter Socket-Instanzen. • 6 - Anzahl von als UDP erstellter Socket-Instanzen. • 7 - **SOCKET_READ** Erfolgszählung, wenn Socket-Instanz als TCP konfiguriert ist. • 8 - **SOCKET_WRITE** Erfolgszählung, wenn Socket-Instanz als TCP konfiguriert ist. • 9 - **SOCKET_READ** Fehlerzählung, wenn Socket-Instanz als TCP konfiguriert ist. • 10 - **SOCKET_WRITE** Fehlerzählung, wenn Socket-Instanz als TCP konfiguriert ist. • 11 - **SOCKET_READ** Erfolgszählung, wenn Socket-Instanz als UDP konfiguriert ist. • 12 - **SOCKET_WRITE** Erfolgszählung, wenn Socket-Instanz als UDP konfiguriert ist. • 13 - **SOCKET_READ** Fehlerzählung, wenn Socket-Instanz als UDP konfiguriert ist. • 14 - **SOCKET_WRITE** Fehlerzählung, wenn Socket-Instanz als UDP konfiguriert ist. • 15 - werden als 0 angezeigt.

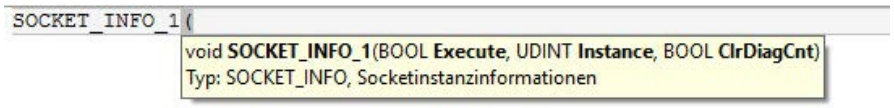
Beispiel für ein SOCKET_INFO-Funktionsblockdiagramm



Beispiel für einen SOCKET_INFO-Kontaktplan

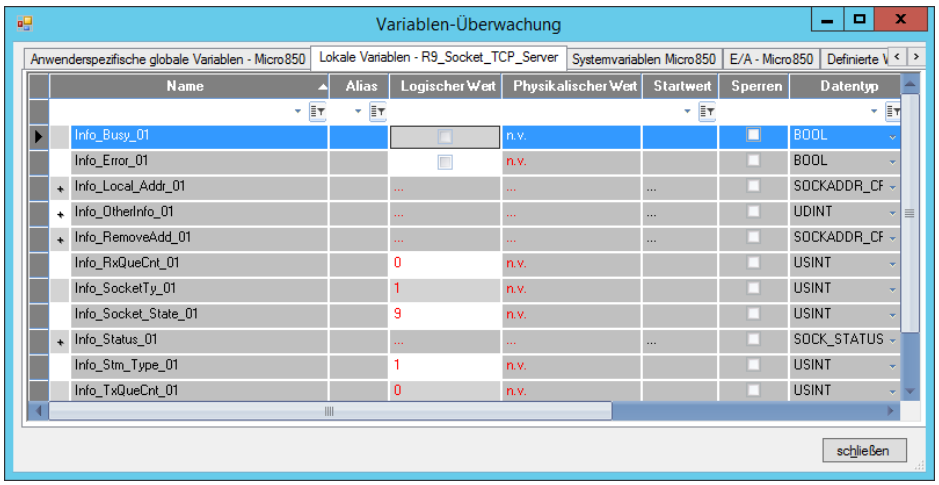


Beispiel für einen strukturierten SOCKET_INFO-Text




```
SOCKET_INFO_1 (Execute_SOCKET_INFO, Instance_SOCKET_INFO, ClrDiagC_SOCKET_INFO);
Done_SOCKET_INFO := SOCKET_INFO. Done;
Busy_SOCKET_INFO := SOCKET_INFO. Busy;
Error_SOCKET_INFO := SOCKET_INFO. Error;
Status_SOCKET_INFO := SOCKET_INFO. Status;
SocketType_SOCKET_INFO := SOCKET_INFO. SocketTy;
StreamType_SOCKET_INFO := SOCKET_INFO. StreamType;
TxQueCnt_SOCKET_INFO := SOCKET_INFO. TxQueCnt;
RxQueCnt_SOCKET_INFO := SOCKET_INFO. RxQueCnt;
SocketState_SOCKET_INFO := SOCKET_INFO. SocketState;
LocalAddr_SOCKET_INFO := SOCKET_INFO. LocalAddr;
RemoteAddr_SOCKET_INFO := SOCKET_INFO. RemoteAddr;
OtherInfo SOCKET INFO := SOCKET INFO. OtherInfo;
```

Ergebnisse



SOCKET_OPEN

Stellt die Verbindung für die angegebene Zieladresse für Transmission Control Protocol-Verbindungen (TCP) her. Für User Datagram Protocol-Verbindungen (UDP) wird dem Socket eine Ziel-IP-Adresse und eine Port-Nummer zugeordnet.

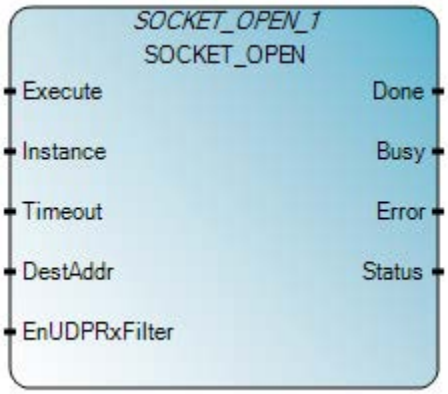
Details zum Betrieb:

- Für User Datagram Protocol (UDP)-Verbindungen mittels **SOCKET_OPEN** sind IP-Adresse und Port-Nummer nicht bei jedem Senden der Daten erforderlich.
- Für UDP-Verbindungen, bei denen **SOCKET_OPEN** nicht verwendet wird, muss die Zieladresse bei jeder Verwendung von **SOCKET_WRITE** zum Senden von Daten angegeben werden.
- Bei Verwendung von **SOCKET_READ** wird neben dem Empfang von Daten die Adresse des Absenders zurückgegeben. Die Adresse des Absenders kann zum Senden einer Antwort mittels **SOCKET_WRITE** verwendet werden.
- Ein **SOCKET_OPEN**-Vorgang wird ggf. vor dem Zeitüberschreitungszeitraum zurückgegeben, ohne dass eine Transmission Control Protocol (TCP)-Verbindung erstellt wird. Dies kann auftreten, wenn das Zielgerät in Betrieb ist, jedoch Verbindungen an der angegebenen Portnummer nicht beachtet; in diesem Fall gibt **SOCKET_OPEN** vor dem Zeitüberschreitungszeitraum einen Fehler zurück.
- Die Ausgänge werden synchron vom Programm-Scan aktualisiert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro850 und Micro870. Für den Micro800-Simulator kann diese Anweisung zu einem Programm hinzugefügt werden. Sie dient aber lediglich als Platzhalter, um zu verhindern, dass die Anweisung bei der Controller-Änderung gelöscht wird.

Die Ausgänge werden jedes Mal zurückgesetzt, wenn die Anweisung auf die simulierte Steuerung (2080-LC50-48QWB-SIM) angewendet wird.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

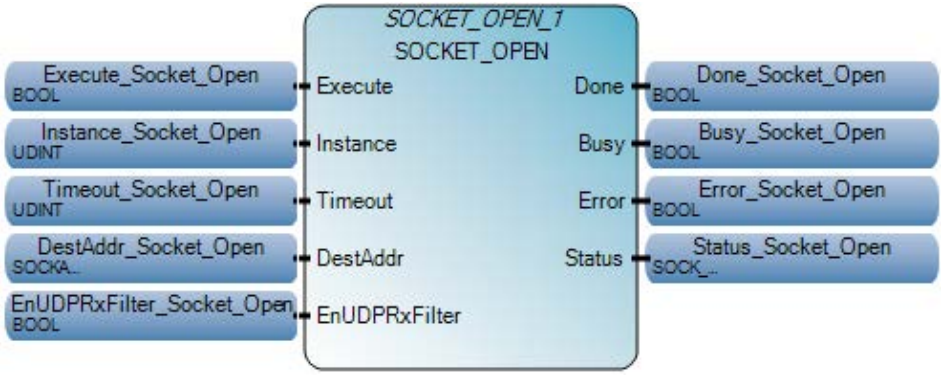
Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Anstiegsflanke erkannt. Startet den Anweisungsblock unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE – Keine Anstiegsflanke erkannt.
Instance	Eingang	UDINT	Kopieren vom zurückgegebenen Socket-Handler aus einem **SOCKET_CREATE**-Funktionsblock.
Timeout	Eingang	UDINT	Zeitüberschreitung für SOCKET_OPEN-Funktionsblock. Der Funktionsblock gibt einen Fehler zurück, wenn der Zeitüberschreitungswert geringer als der Mindestwert ist. Zeitüberschreitungsbereich: 1.000 - 1800000 Millisekunden Einstellen von Zeitüberschreitung auf 0 zur Verwendung des Standardwerts 10000 (10 Sekunden).

DestAddr	Eingang	SOCKADDR_CFG auf seite 596	Die Adresse der Zielverbindung.
			Eine Verbindung zwischen der IP-Adresse und der Portnummer des Remote-Host ist erforderlich.
			Die folgenden IP-Adressen werden nicht für DestAddr unterstützt:
			• Eigene IP-Adresse
			• Loop-back-Adresse
			• 0.0.0.0
			• Sendeadresse (Nur unterstützt für TCP), Ausnahmeadressen
			  • Multicast-Adresse der Klasse D (224.x.x.x)
			  • Lokale Link-Adresse (169.254.x.x)
			Beispiel für IP-Adresse von 192.168.2.100 und Port 12000:
			• DestAddr.IPAddress[0]=192
			• DestAddr.IPAddress[1]=168
			• DestAddr.IPAddress[2]=2
			• DestAddr.IPAddress[3]=100
			• DestAddr.Port = 12000

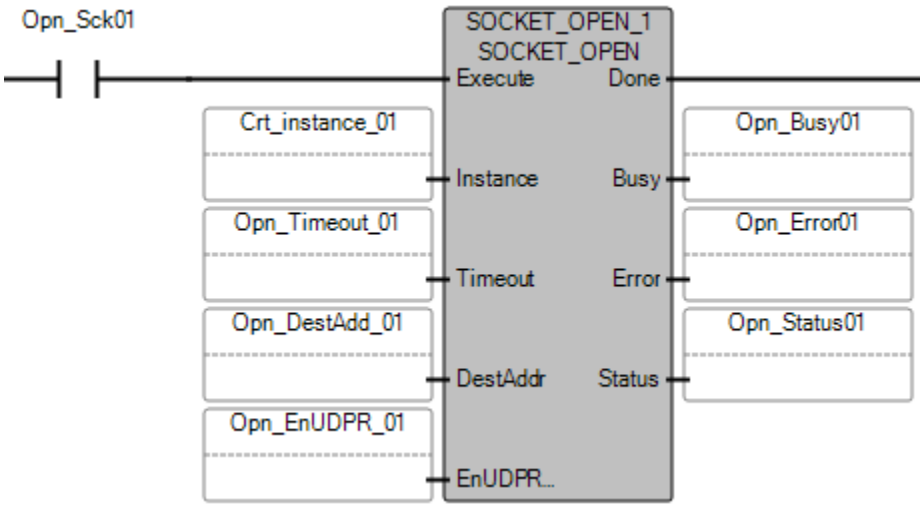
EnUDPRxFilter	Eingang	BOOL	Wenn für UDP-Socket SOCKET_OPEN und EnUDPRxFilter aktivieren verwendet werden, wird kein Paketfilter für eine spezifische IP-Adresse und Portnummer bei jedem Lesen der Daten benötigt. • Eine UDP-Socket mit offenem wird erstellt. DestAddr als IP 192.168.1.157/Port 161.

Parameter	Parametertyp	Datentyp	Beschreibung
			EnUDPRxFilter aktivieren, Socket_Read durchführen. • Beim Empfangen von Daten der Steuerung von DestAddr (192.168.1.157 / 161) schließt Socket_Read den Vorgang erfolgreich ab. Wenn der Controller Daten von einer anderen IP-Adresse oder einem anderen Port empfängt, dann ignoriert Socket_Read das Paket und wartet auf das DestAddr-Paket. EnUDPRxFilter deaktiviert: • SOCKET_READ durchführen. Wenn die Steuerung bei der Konfiguration des UDP-Ports empfängt, schließt SOCKET_READ den Vorgang erfolgreich ab. • Die Anwendung überprüft, ob das eingehende Paket vom erwarteten Gerät eingeht oder nicht. • Die Anwendung führt die Filterung basierend auf dem SOCKET_READ-Ausgangsparameter UDPAddr durch.
Done	Ausgang	BOOL	Gibt an, wenn der Vorgang abgeschlossen ist. TRUE – Vorgang erfolgreich abgeschlossen. FALSE – Vorgang wird ausgeführt oder eine Fehlerbedingung ist aufgetreten. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Busy	Ausgang	BOOL	TRUE – Vorgang ist nicht abgeschlossen. FALSE – Vorgang ist abgeschlossen. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Status	Ausgang	SOCK_STATUS	Der Status wird mit dem Datentyp SOCK_STATUS auf seite 596 definiert, der ErrorID auf seite 597-, SubErrorID- und StatusBits auf seite 601-Informationen enthält. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.

Beispiel für ein SOCKET_OPEN-Funktionsblockdiagramm



Beispiel für einen SOCKET_OPEN-Kontaktplan



Beispiel für einen strukturierten SOCKET_OPEN-Text

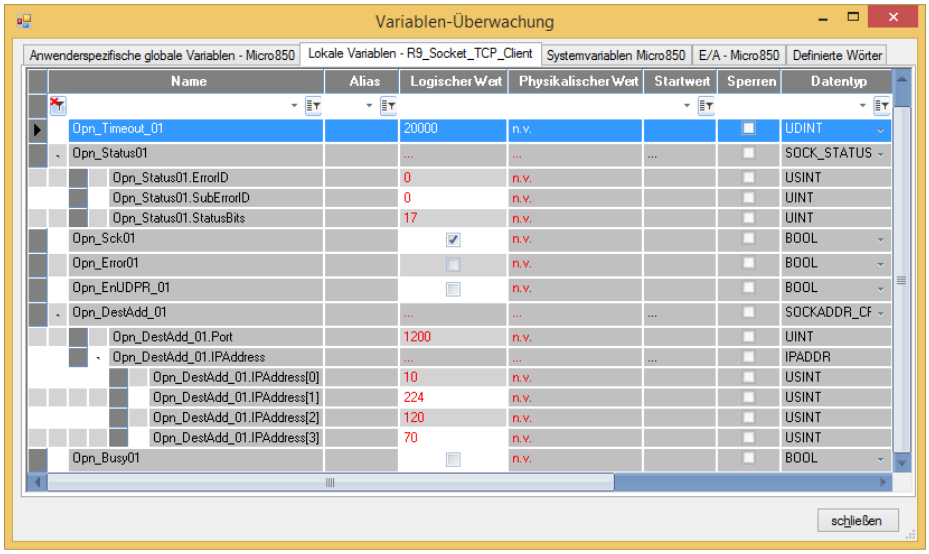
```
SOCKET_OPEN_1 (
```

void **SOCKET_OPEN_1**(BOOL **Execute**, UDINT **Instance**, UDINT **Timeout**, SOCKADDR_CFG **DestAddr**, BOOL **EnUDPRxFilter**)
Typ: SOCKET_OPEN, Socket öffnen

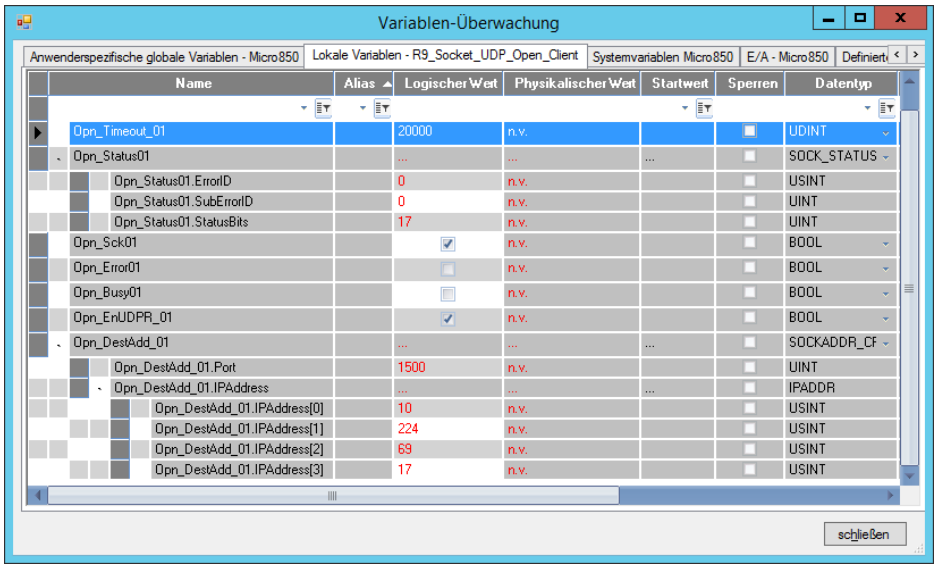
```
SOCKET_OPEN_1 (Execute_SOCKET_OPEN. Instance_SOCKET_OPEN, Timeout_SOCKET_OPEN, BestAddr_SOCKET_OPEN, EnVDPR_SOCKET_OPEN)
Done_SOCKET_OPEN := _SOCKET_OPEN. Done:
Busy_SOCKET_OPEN := _SOCKET_OPEN. Busy:
Error_SOCKET_OPEN := _SOCKET_OPEN. Error:
Status_SOCKET_OPEN := _SOCKET_OPEN. Status:
```

Ergebnisse

TCP-Beispiel



UDP-Beispiel



SOCKET_READ

Liest Daten an einen Socket, und gibt die angegebene Anzahl der Bytes zurück. Wird für Transmission Control Protocol (TCP) beim Empfang von Daten zurückgegeben, bis die angeforderte Anzahl Byte erreicht wurde. Wird für Protokoll UDP (User Datagram) abgeschlossen, wenn ein Datagram zur Verfügung steht.

Details zum Betrieb:

Das folgende Verhalten von **SOCKET_READ** kann Auswirkungen auf vorhandene Kommunikation einschließlich Nicht-Socket-Kommunikation haben.

- Wenn der **SOCKET_READ**-Vorgang nicht synchron mit dem Remote-Gerät ausgeführt wird, hält der Controller das Remote-Gerät-Empfangspaket zurück, bis eine der folgenden Situationen auftritt:
  - **SOCKET_READ** wird ausgeführt
  - Socket-Zeitüberschreitung läuft ab
  - RST wird von einem Remote-Gerät empfangen
  - **SOCKET_DELETE** oder **SOCKET_DELETEALL** wird ausgeführt
  - Der Benutzer führt einen **Ausführungsmodus ändern**-Vorgang aus, durch den alle erstellten Socket-Instanzen gelöscht werden.
  - Der Controller wechselt vom Ausführungsmodus in den Programmmodus, sodass alle erstellten Socket-Instanzen gelöscht werden.
  - Der Controller wechselt vom Ausführungsmodus in den Programmmodus, sodass Zählerinformationen der Socket-Diagnose und einzelne Socket-Zählerinformationen gelöscht werden.

- Wenn der Length- oder Offset-Parameterwert geändert wird, während der SOCKET_READ-Vorgang ausgeführt wird (BUSY = True), tritt ein Fehler auf, und das Empfangspaket wird verworfen.
- Die **SOCKET_READ**-Anweisung gibt möglicherweise weniger Bytes als angefordert zurück. RxLength enthält die Anzahl empfangener Datenbyte. Schreiben Sie Programme zur Überprüfung von RxLength und geben Sie anschließend weitere Leseanforderungen aus, um eine komplette Anwendungsmeldung zu erhalten.
- Im **Änderung im Betriebsmodus** bewirkt das Ändern einer **SOCKET_READ**-Eingabe, während Sie sich in der Zustand BUSY befinden, zu einem Fehler, und das empfangene Paket wird verworfen
- Die Ausgänge werden synchron vom Programm-Scan aktualisiert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro850 und Micro870. Für den Micro800-Simulator kann diese Anweisung zu einem Programm hinzugefügt werden. Sie dient aber lediglich als Platzhalter, um zu verhindern, dass die Anweisung bei der Controller-Änderung gelöscht wird.

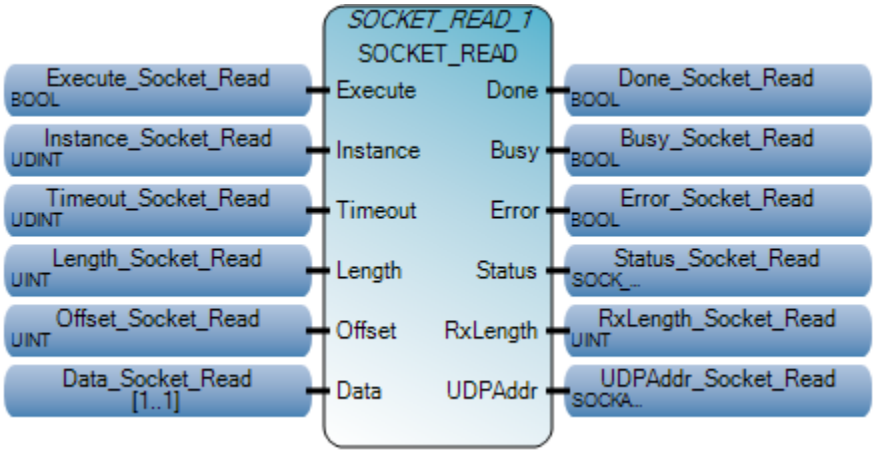
Die Ausgänge werden jedes Mal zurückgesetzt, wenn die Anweisung auf die simulierte Steuerung (2080-LC50-48QWB-SIM) angewendet wird.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

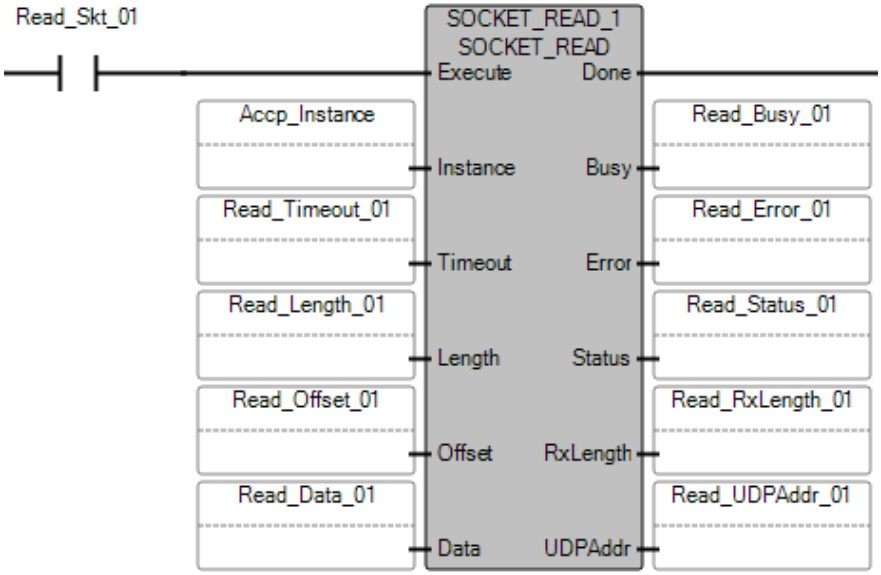
Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Anweisung aktivieren. TRUE – Anstiegflanke erkannt, startet die Anweisung unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE – Startet keine Anweisung.
Instance	Eingang	UDINT	Kopieren vom zurückgegebenen Socket-Handler aus einer SOCKET_CREATE- oder SOCKET_ACCEPT-Anweisung. • Kopieren Sie für die Socket-Typen von UDP- und TCP-Client Socket den zurückgegebenen Socket-Handler aus einer SOCKET_CREATE-Anweisung. • Kopieren Sie für den Socket-Typ des TCP-Servers vom zurückgegebenen Socket-Handler aus einer SOCKET_ACCEPT-Anweisung.

Parameter	Parametertyp	Datentyp	Beschreibung
Timeout	Eingang	UDINT	Zeitüberschreitung für **SOCKET_READ**. Der Anweisungsblock gibt einen Fehler zurück, wenn der Zeitüberschreitungswert geringer als der Mindestwert ist. Zeitüberschreitungsbereich: 1000- 86400000 Millisekunden Einstellen von **Zeitüberschreitung** auf 0 zur Verwendung des Standardwerts 10000 (10 Sekunden).
Length	Eingang	UINT	Definiert die Anzahl der zu lesenden Bytes. **RxLength** auf die tatsächliche Anzahl der gelesenen Bytes überprüfen. **SOCKET_READ** kann weniger Bytes als angefordert zurückgeben. Unterstützt bis zu 256 Bytes.
Offset	Eingang	UNIT	Versatz im **Datenarray**. Beginn des Lesens der Daten an diesem Ort.
Data	Ausgang	USINT[1..1]	Ein Array, das zum Speichern der aus **SOCKET_READ** gelesenen Daten verwendet wird. • Die Datenarray-Größe muss >= (**Versatz** + **Länge**) sein. • Das Datenarray kann größer als die Länge des Socket-Lesevorgangs sein. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Done	Ausgang	BOOL	Gibt an, wenn der Vorgang abgeschlossen ist. TRUE – Vorgang erfolgreich abgeschlossen. FALSE – Vorgang wird ausgeführt oder eine Fehlerbedingung ist aufgetreten. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Busy	Ausgang	BOOL	TRUE – Vorgang ist nicht abgeschlossen. FALSE – Vorgang ist abgeschlossen. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Status	Ausgang	SOCK_STATUS	Der Status wird mit dem Datentyp SOCK_STATUS auf seite 596 definiert, der ErrorID auf seite 597-, SubErrorID- und StatusBits auf seite 601-Informationen enthält. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
RxLength	Ausgang	UNIT	Enthält die Anzahl der empfangenen Datenbytes.
UDPAddr	Ausgang	SOCKADDR_CFG auf seite 596	Die Adresse des Geräts, das die User Datagram Protocol (UDP)-Daten sendet. Beispiel setzt UDPAddr auf 192.168.2.100 und Port auf 12000: UDPAddr.IPAddress[0]=192 UDPAddr.IPAddress[1]=168 UDPAddr.IPAddress[2]=2 UDPAddr.IPAddress[3]=100 UDPAddr.Port = 12000 Für das Transmission Control Protocol (TCP) wird diese Struktur nicht verwendet und enthält alle Nullen. Die TCP-Verbindung übermittelt die Remote-Adressinformationen.

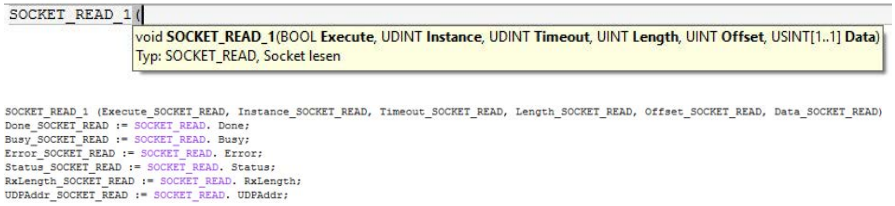
Bbeispiel für ein SOCKET_READ-Funktionsblockdiagramm



Beispiel für einen SOCKET_READ-Kontaktplan

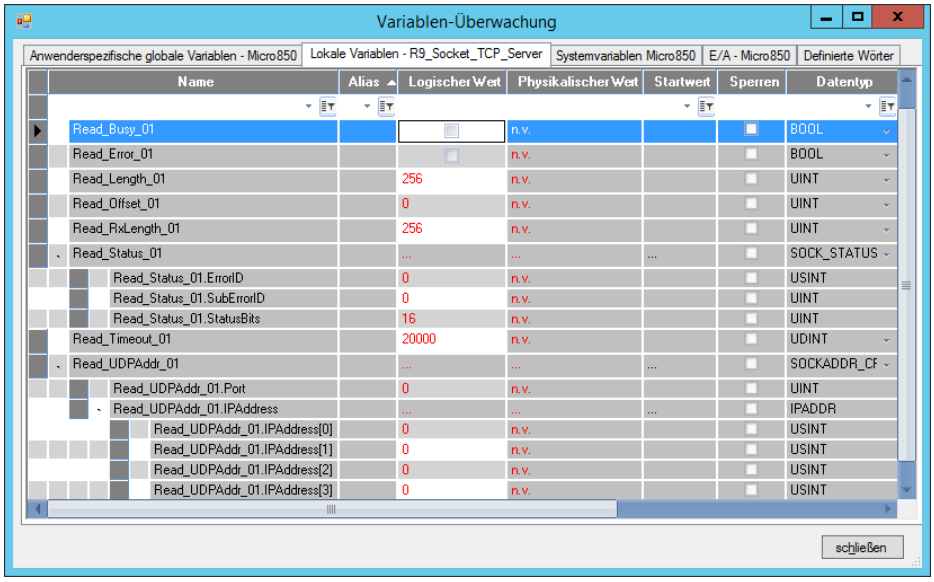


Beispiel für einen strukturierten SOCKET_READ-Text

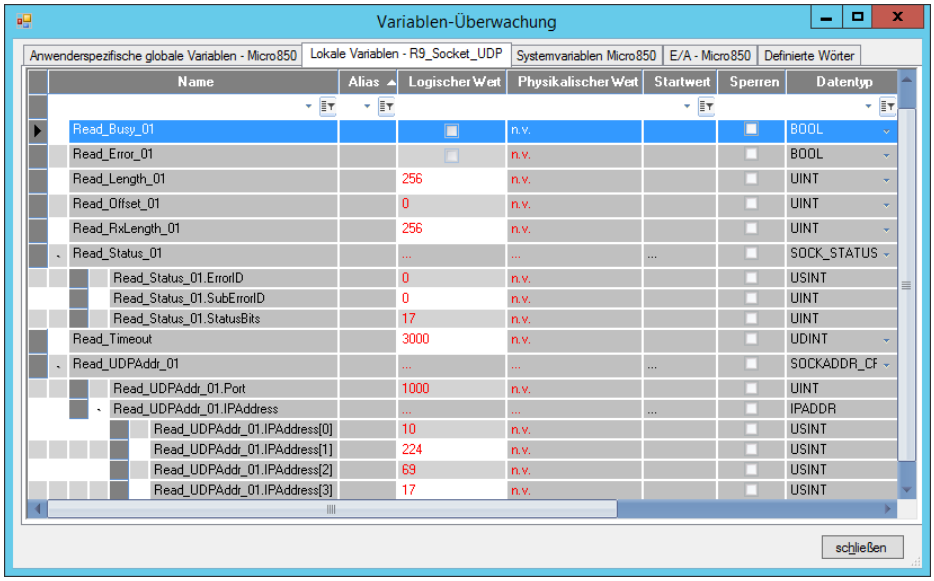


Ergebnisse

TCP-Beispiel:



UDP-Beispiel:



SOCKET_WRITE

Sendet Daten an einen Socket.

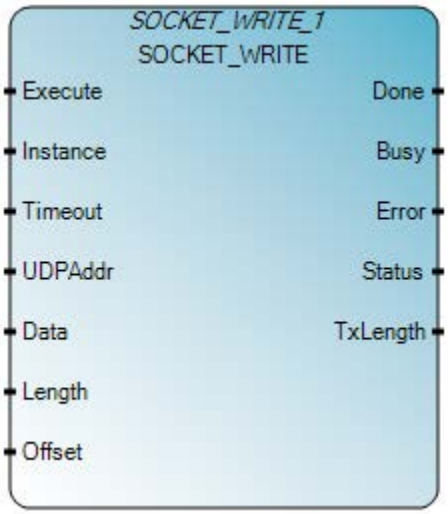
Details zum Betrieb:

- **SOCKET_WRITE** versucht, die angeforderte Anzahl von im Length-Parameter angegebenen Datenbyte zu senden. Wenn der Sendevorgang abgeschlossen ist, gibt **SOCKET_WRITE** die Anzahl von in den TxLength-Parameter geschriebenen Datenbyte zurück.
- Die Ausgabe wird asynchron vom Programm-Scan aktualisiert.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro820, Micro850 und Micro870. Für den Micro800-Simulator kann diese Anweisung zu einem Programm hinzugefügt werden. Sie dient aber lediglich als Platzhalter, um zu verhindern, dass die Anweisung bei der Controller-Änderung gelöscht wird.

Die Ausgänge werden jedes Mal zurückgesetzt, wenn die Anweisung auf die simulierte Steuerung (2080-LC50-48QWB-SIM) angewendet wird.

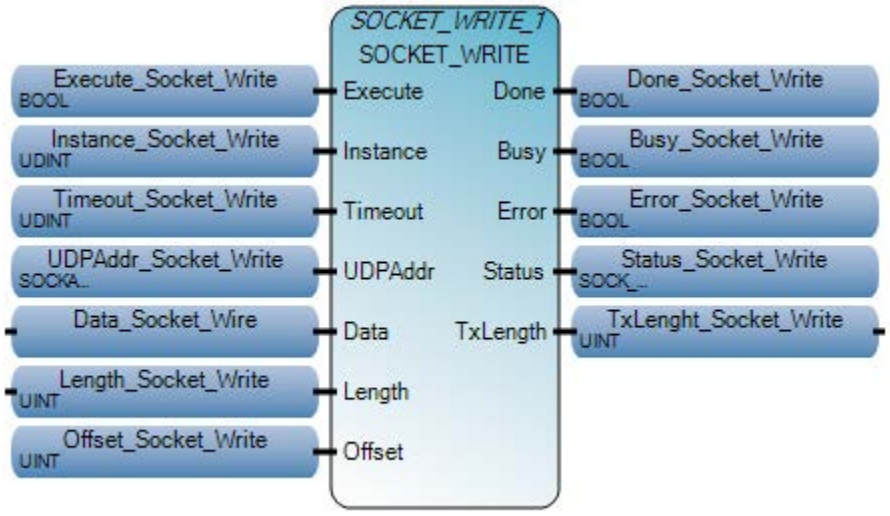


Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

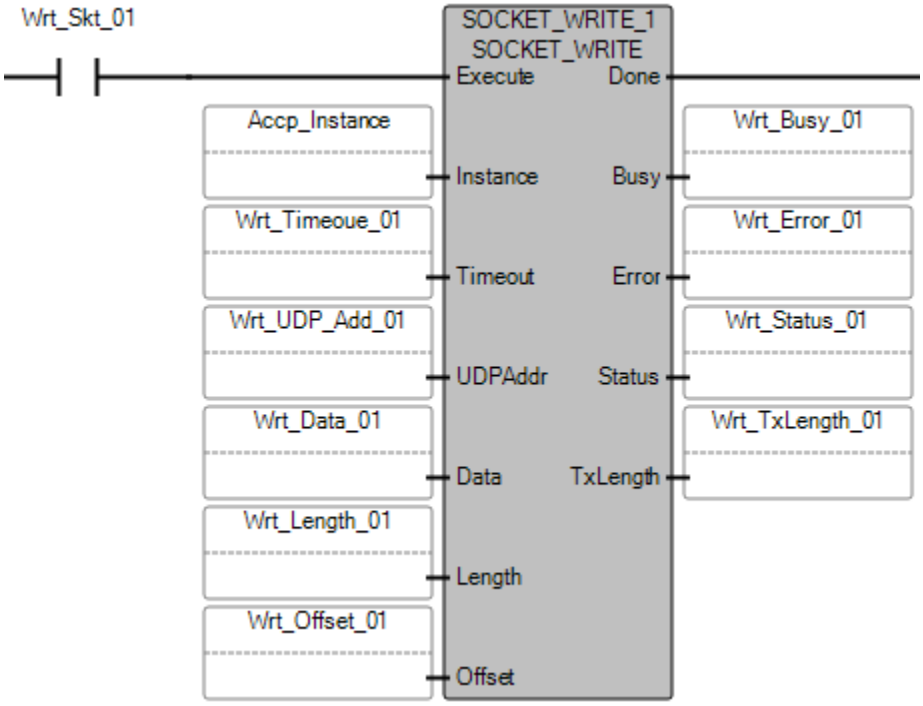
Parameter	Parametertyp	Datentyp	Beschreibung
Execute	Eingang	BOOL	Anweisungsblock aktivieren. TRUE – Anstiegsflanke erkannt. Startet den Anweisungsblock unter der Voraussetzung, dass der letzte Vorgang abgeschlossen ist. FALSE – Keine Anstiegsflanke erkannt.
Instance	Eingang	UDINT	Kopieren vom zurückgegebenen Socket-Handler aus einer **SOCKET_CREATE** oder **SOCKET_ACCEPT**-Anweisung. • Kopieren Sie für die Socket-Typen von UDP- und TCP-Client Socket den zurückgegebenen Socket-Handler aus einer **SOCKET_CREATE**-Anweisung. • Kopieren Sie für den Socket-Typ des TCP-Servers vom zurückgegebenen Socket-Handler aus einer **SOCKET_ACCEPT**-Anweisung.
Timeout	Eingang	UDINT	Zeitüberschreitung für **SOCKET_WRITE**-Instanzen. Die Anweisung gibt einen Fehler zurück, wenn der Zeitüberschreitungswert geringer als der Mindestwert ist. Zeitüberschreitungsbereich: 1000 bis 1800000 Millisekunden Einstellen von Zeitüberschreitung auf 0 zur Verwendung des Standardwerts 10000 (10 Sekunden).
UDPAddr	Eingang	SOCKADDR_CFG auf seite 596	Die UDP-Zieladresse, in die die Daten geschrieben werden, wenn eine **SOCKET_OPEN**-Anweisung seit der Erstellung des Socket nicht ausgeführt wurde. Für TCP oder wenn eine **SOCKET_OPEN**-Anweisung für UDP ausgeführt wurde, wird diese Struktur nicht verwendet und sollte Nullen enthalten. Die TCP-Verbindung und die **SOCKET_OPEN**-Anweisung für UDP übermitteln alle Remote-Adresseninformationen. Beispiel für UDPAddr von 192.168.2.100 und Port 12000: UDPAddr.IPAddress[0]=192 UDPAddr.IPAddress[1]=168 UDPAddr.IPAddress[2]=2 UDPAddr.IPAddress[3]=100 UDPAddr.Port = 12000

Parameter	Parametertyp	Datentyp	Beschreibung
			Verwenden Sie den Datentyp **SOCKADDR_CFG**, um UDPAddr zu definieren.
Data	Eingang	USINT[1..1]	Ein Array zum Speichern des Datenschreibenvorgangs zur jeweiligen Socket-Instanz mithilfe der **SOCKET_WRITE**-Anweisung. • Die Datenarray-Größe muss >= (**Versatz** + **Länge**) sein. • Das Datenarray kann größer als die **SOCKET_WRITE**-Länge sein.
Length	Eingang	UINT	Die Anzahl der zu schreibenden Datenbytes. Maximal 256 Bytes.
Offset	Eingang	UNIT	Versatz im Datenarray. Der Datenschreibvorgang aus SOCKET_WRITE beginnt an dieser Position.
Done	Ausgang	BOOL	Gibt an, wenn der Vorgang abgeschlossen ist. TRUE – Vorgang erfolgreich abgeschlossen. FALSE – Vorgang wird ausgeführt oder eine Fehlerbedingung ist aufgetreten. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Busy	Ausgang	BOOL	TRUE – Vorgang ist nicht abgeschlossen. FALSE – Vorgang ist abgeschlossen. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Error	Ausgang	BOOL	Gibt an, dass ein Fehler aufgetreten ist. WAHR – es wurde ein Fehler erkannt. FALSE - Kein Fehler. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
Status	Ausgang	SOCK_STATUS	Der Status wird mit dem Datentyp SOCK_STATUS auf seite 596 definiert, der ErrorID auf seite 597-, SubErrorID- und StatusBits auf seite 601-Informationen enthält. Die Ausgabe wird synchron vom Programm-Scan aktualisiert.
TxLength	Ausgang	UNIT	Die Anzahl der Bytes der geschriebenen Daten.

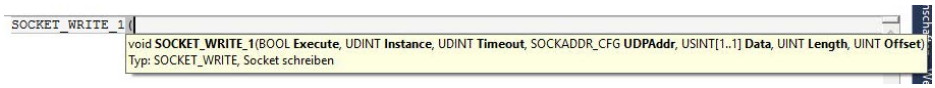
Beispiel für ein SOCKET_WRITE-Funktionsblockdiagramm



Beispiel für einen SOCKET_WRITE-Kontaktplan



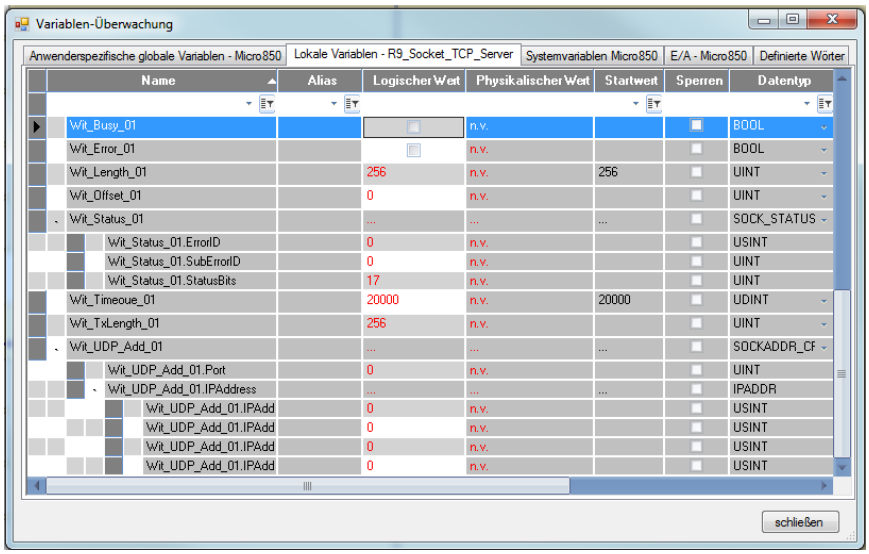
Beispiel für einen strukturierten SOCKET_WRITE-Text



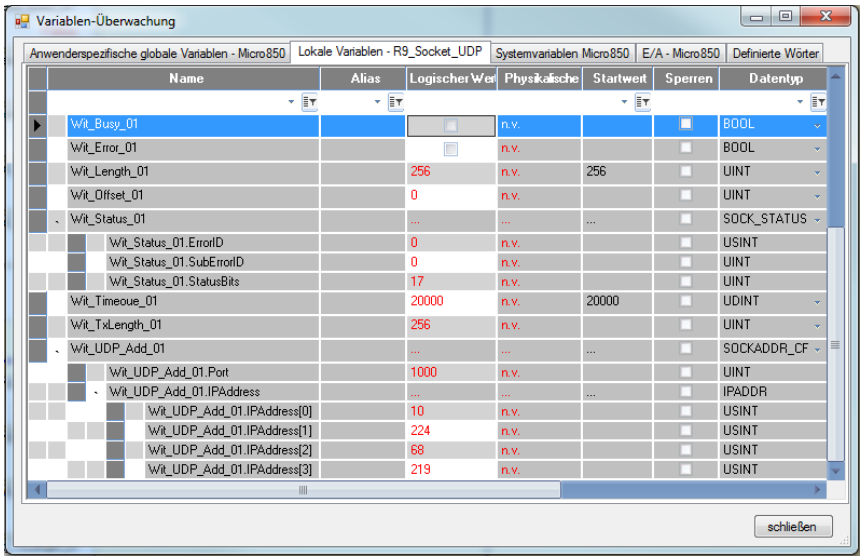

```
SOCKET_WRITE_1(Execute_SOCKET_WRITE, Instance_SOCKET_WRITE, Timeout_SOCKET_WRITE, UDPAddr_SOCKET_WRITE, Length_SOCKET_WRITE, Offset_SOCKET_WRITE)
Done_SOCKET_WRITE := SOCKET_WRITE.Done;
Busy_SOCKET_WRITE := SOCKET_WRITE.Busy;
Error_SOCKET_WRITE := SOCKET_WRITE.Error;
Status_SOCKET_WRITE := SOCKET_WRITE.Status;
TxLength_SOCKET_WRITE := SOCKET_WRITE.TxLength;
```

Ergebnisse

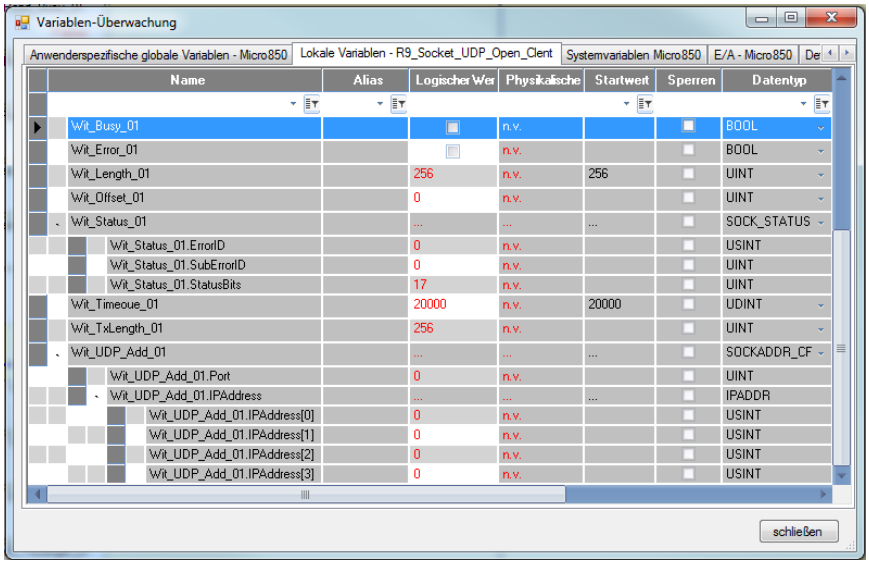
TCP-Beispiel:



Beispiel für eine UDP-Verbindung ohne Öffnen



Beispiel für eine UDP-Verbindung mit Öffnen



Datentyp SOCKADDR_CFG

In der folgenden Tabelle wird der Datentyp **SOCKADDR_CFG** beschrieben.

Parameter	Datentyp	Beschreibung
Port	UINT	Geben Sie eine lokale Port-Nummer an, bei der eine Anwendung überwacht und empfängt.
IP Address[4]	USINT	Geben Sie eine IP-Adresse an. Beispiel für die IP-Adresse 192.168.2.100: • `IPAddress[0]=192` • `IPAddress[1]=168` • `IPAddress[2]=2` • `IPAddress[3]=100`

Datentyp SOCK_STATUS

In der folgenden Tabelle wird der Datentyp **SOCK_STATUS** beschrieben.

Parameter	Datentyp	Beschreibung
ErrorID	USINT	Fehlercode auf seite 597-Wert.
SubErrorID	UINT	Wert des Subfehlercodes.
StatusBits	UINT	Ausführungs-Statusbits für Socket-Befehle auf seite 601.

Socket-Fehlercodes

In der folgenden Tabelle werden die Statusfehlercodes für Socket-Anweisungen beschrieben.

ErrorID-Code	SubErrorID-Code	Fehlerbeschreibung	Maßnahme
0		Die Socket-Anweisung wurde erfolgreich abgeschlossen.	
1		Die Socket-Anweisung steht aus.	
2		Die Socket-Instanz ist nicht verfügbar.	Vergewissern Sie sich, dass die Socket-Instanz nicht gelöscht wurde oder das Zeitlimit überschritten wurde.
3		Der **SOCKET_DELETEALL**-Vorgang wird ausgeführt.	Warten Sie, bis der ausstehende **SOCKET_DELETEALL**-Vorgang abgeschlossen ist.
4	1	Unzulässiger Parameter, der Kanal ist ungültig.	Keine Aktion. Für die spätere Verwendung reserviert.

4	2	Unzulässiger Parameter, ungültige Socket-IP-Adresse
		• Es tritt ein Fehler auf. **SOCKET_CREATE** enthält eine **SockAddr** mit einem Wert ungleich Null.
		• **SOCKET_OPEN** oder **SOCKET_WRITE** wird mit einer ungültigen Ziel-IP-Adresse ausgeführt. Ungültige Ziel-IP-Adressen sind:
		  • Eigene IP-Adresse
		  • 0.0.0.0
		  • Loop-Back-Adresse (127.x.x.x)
		  • Multicast-Adresse der Klasse D (224.x.x.x)
		  • Lokale Link-Adresse (169.254.x.x)
		  • Broadcast-Adresse (gilt nur für TCP-Socket-Instanz)
			Ändern Sie diese in eine gültige IP-Adresse.

4	3	Ungültiger Parameter, ungültige Socket-Port-Adresse.
		Fehler tritt auf, wenn die folgenden Ports in **SOCKET_CREATE**, **SOCKET_OPEN** oder **SOCKET_WRITE** angegeben sind:
		• TCP-Ports
		  • 44818 - EtherNet/IP
		  • 502 - ModbusTCP
		  • 67 - DHCP-Server
		  • 68 - DHCP-Client
		  • 0 - Ungültiger Port
		• UDP-Ports
		  • 2222 – EtherNet/IP
		  • 67 – DHCP-Server
		  • 68 – DHCP-Client
		  • 0 – Ungültiger Port
			Ändern Sie die Portadresse.

4	4	Unzulässiger Parameter, ungültiger Socket-Typ. Gültige Socket-Typen: • TCP - 1 • UDP - 2	Ändern Sie den Socket-Typ.
4	5	Unzulässiger Parameter, ungültiger	Ändern Sie den Zeitüberschreitungswert in einen gültigen

ErrorID-Code	SubErrorID-Code	Fehlerbeschreibung	Maßnahme
		Socket-Zeitüberschreitungswert. Gültige Zeitüberschreitungswerte sind: • SOCKET_CREATE, SOCKET_ACCEPT und SOCKET_READ: • 0 (Standard) oder ein beliebiger Wert zwischen 1.000 und 86.400.000 ms. • SOCKET_OPEN und SOCKET_WRITE: • 0 (Standard) oder ein beliebiger Wert zwischen 1.000 und 86.400.000 ms.	Wert.
4	6	Unzulässiger Parameter, ungültige Socket-Instanz. • Socket-Typ für TCP-Server: • Verwenden Sie die Socket-Instanz, die in SOCKET_ACCEPT für SOCKET_WRITE, SOCKET_READ, SOCKET_DELETE und SOCKET_INFO zurückgegeben wird. • Socket-Typ für UDP- und TCP-Client: • Verwenden Sie die Socket-Instanz, die in SOCKET_CREATE für SOCKET_OPEN, SOCKET_WRITE, SOCKET_READ, SOCKET_DELETE und SOCKET_INFO zurückgegeben wird.	Verwenden Sie die Instanzanzahl, die in SOCKET_CREATE- und SOCKET_ACCEPT-Anweisungen nach erfolgreicher Ausführung zurückgegeben wird.
4	7	Ungültiger Parameter, ungültige Array-Länge.	Erhöhen Sie die Größe des Arrays, der die Lese- und Schreibdaten des Sockets enthält.
4	8	Unzulässiger Parameter, ungültige Array-Dimension.	Verwenden Sie einen eindimensionalen Array, der die Lese- und Schreibdaten des Sockets enthält.
5		Socket-Anforderung, um den Vorgang abzubrechen. Fehler tritt auf, wenn: • Ethernet-Verbindung wurde deaktiviert oder Ethernet-Kabel wurde getrennt • SOCKET_DELETE-Vorgang wird ausgeführt, während SOCKET_OPEN, SOCKET_ACCEPT, SOCKET_READ oder SOCKET_WRITE -Vorgang noch läuft. • Änderung im Betriebsmodus wird ausgeführt, während SOCKET_OPEN, SOCKET_ACCEPT, SOCKET_READ oder SOCKET_WRITE-Vorgang noch läuft. • IP-Adressenüberschneidung festgestellt, während SOCKET_OPEN, SOCKET_ACCEPT, SOCKET_READ oder SOCKET_WRITE-Vorgang noch läuft.	Starten Sie den Socket-Vorgang basierend auf dem Socket-Typ neu. Informationen zum Neustart des Socket-Vorgangs finden Sie unter dem entsprechenden Maschinenstatus.
6	1	Unzulässige Socket-Sequenz, Socket-Öffnungsvorgang läuft bzw. ist verbunden. Fehler tritt auf, wenn: • SOCKET_OPEN wird durchgeführt (Zustand BUSY) und Benutzer führt eine weitere SOCKET_OPEN-Instanz mit derselben Socket-Instanz aus. • SOCKET_OPEN wird durchgeführt (Zustand BUSY) und Benutzer führt SOCKET_ACCEPT mit derselben Socket-Instanz aus.	Verwenden Sie eine einzige SOCKET_OPEN-Ausführung für die entsprechende Socket-Instanz. Führen Sie keinen SOCKET_ACCEPT-Vorgang für dieselbe Socket-Instanz aus.
6	2	Unzulässige Socket-Sequenz, Socket-Akzeptierungsvorgang läuft bzw. ist verbunden. Fehler tritt auf, wenn: • SOCKET_ACCEPT wird durchgeführt (Zustand BUSY) und Benutzer führt eine weitere SOCKET_ACCEPT-Instanz mit derselben Socket-Instanz aus. • SOCKET_ACCEPT wird durchgeführt (Zustand BUSY) und Benutzer führt SOCKET_OPEN mit derselben Socket-Instanz aus.	Verwenden Sie eine einzige SOCKET_ACCEPT-Ausführung für die entsprechende Socket-Instanz. Führen Sie keinen SOCKET_OPEN-Vorgang für dieselbe Socket-Instanz aus.
6	3	Unzulässige Socket-Sequenz, Socket als TCP-Client konfiguriert.	

ErrorID-Code	SubErrorID-Code	Fehlerbeschreibung	Maßnahme
6	4	Unzulässige Socket-Sequenz, Socket als TCP-Server konfiguriert.	
6	5	Unzulässige Socket-Sequenz, Socket ist verbunden.	Stellen Sie sicher, dass der Socket nicht bereits mit einem Remote-Gerät verbunden ist, bevor Sie **SOCKET_OPEN** oder **SOCKET_ACCEPT** erneut ausführen.
6	6	Unzulässige Socket-Sequenz, Sockel ist als UDP konfiguriert.	Stellen Sie sicher, dass UDP-Socket-Instanzen nicht mit **SOCKET_ACCEPT** verwendet werden.
6	7	Unzulässige Socket-Sequenz, Socket ist nicht verbunden.	Stellen Sie sicher, dass der Socket mit dem Ziel verbunden ist.
7		Socket-Instanz-Zeitüberschreitung.	Konfigurieren Sie die Zeitüberschreitungswerte für **SOCKET_ACCEPT**, **SOCKET_OPEN**, **SOCKET_READ** und **SOCKET_WRITE** entsprechend.
8		Das Socket-Modul ist nicht initialisiert.	Vergewissern Sie sich, dass die Ethernet-Verbindung aktiviert ist oder das Ethernet-Kabel des Controllers mit dem Netzwerk verbunden ist, oder beheben Sie den IP-Adressenkonflikt des Controllers.
9	1	Schwerwiegender Fehler in Socket-Anweisung, Socket-Instanz fehlt.	
9	2	Schwerwiegender Fehler in Socket-Anweisung, Socket-Instanz ungültig.	
9	3	Schwerwiegender Fehler in Socket-Anweisung, Sperrung der Socket-Instanz ungültig.	
9	4	Schwerwiegender Fehler in Socket-Anweisung, Socket-Typ ungültig.	
9	5	Schwerwiegender Fehler in Socket-Anweisung, Abbruch-Handler fehlt.	
10	1	Fehler in Socket-Hintergrundverarbeitung, Adresse wird verwendet.	
10	2	Fehler in Socket-Hintergrundverarbeitung, UDP hat großes Paket empfangen und das Paket ist größer als die Socket-Leselänge.	Stellen Sie sicher, dass die **SOCKET_READ**-Länge größer als die empfangene Paketgröße ist. Maximale Lesegröße wird empfohlen. Maximale Größe für **SOCKET_READ** ist 256.
10	3	Fehler in Socket-Hintergrundverarbeitung, TCP hat großes Paket empfangen und das Paket ist größer als die Socket-Leselänge.	Stellen Sie sicher, dass die **SOCKET_READ**-Länge größer als die empfangene Paketgröße ist. Da Clients bis zur Empfangs-Fenstergröße von 256 senden können, wird 256 empfohlen.
10	4	Fehler in Socket-Hintergrundverarbeitung, RST empfangen oder Verbindung mit Remote-Gerät getrennt.	Starten Sie den TCP-Client oder Server des Controllers wie im Statusdiagramm dargestellt neu. Überprüfen Sie das Zielgerät, das RST gesendet hat, und stellen Sie sicher, dass das Ziel mit dem richtigen Status neu gestartet wird.
10	5	Fehler in Socket-Hintergrundverarbeitung, UDP-Paket von anderem Gerät erhalten, Paket verworfen. Für die spätere Verwendung reserviert.	
10	6	Fehler in Socket-Hintergrundverarbeitung, Warteschlange ist voll. Fehler tritt auf, wenn: • **SOCKET_READ** oder **SOCKET_WRITE** führt vier Anweisungen für die gleiche Socket-Instanz im selben Scan aus. • Führen Sie **SOCKET_READ** aus, wenn vier **SOCKET_READ**-Anweisungen den Zustand BUSY für die gleiche Socket-Instanz haben.	Warten Sie auf die Verfügbarkeit der **SOCKET_READ**- und **SOCKET_WRITE**-Warteschlange für die entsprechende Socket-Instanz.
10	7	Fehler in Socket-Hintergrundverarbeitung, **SOCKET_READ**-Parameteränderung (Länge, Abstand, Daten-Arraygröße und Daten-Arrayvariable).	Wenn der Zustand für **SOCKET_READ BUSY** lautet, dürfen die Eingabeparameter nicht geändert werden.
11	1	Schwerwiegender Fehler in Socket-Hintergrundverarbeitung,	

ErrorID-Code	SubErrorID-Code	Fehlerbeschreibung	Maßnahme
		fehlender Datenzeiger.	
11	2	Schwerwiegender Fehler in Socket-Hintergrundverarbeitung, fehlender Sitzungszeiger.	
11	3	Schwerwiegender Fehler in Socket-Hintergrundverarbeitung, ungültiger TCP- oder UDP-Socket-Zeiger.	
11	4	Schwerwiegender Fehler in Socket-Hintergrundverarbeitung, Socket-Instanztyp ungültig.	
11	5	Schwerwiegender Fehler in Socket-Hintergrundverarbeitung, Socket-Instanz fehlt.	
11	6	Schwerwiegender Fehler in Socket-Hintergrundverarbeitung, Socket-Instanz ungültig.	
11	7	Schwerwiegender Fehler in Socket-Hintergrundverarbeitung, Socket-Zustand ungültig.	
11	8	Schwerwiegender Fehler in Socket-Hintergrundverarbeitung, Socket-Typ ungültig.	
11	9	Schwerwiegender Fehler in Socket-Hintergrundverarbeitung, TCP-Löschfehler.	
11	10	Schwerwiegender Fehler in Socket-Hintergrundverarbeitung, UDP-Löschfehler.	
128	1	Kein Paket für Trennmeldung verfügbar.	
128	2	Nicht genügend Platz zur Voranstellung des TCP-Headers.	
128	3	Paket-Anhangszeiger ist ungültig.	
128	7	Ungültiger Socket-Zeiger.	
128	10	Ungültige Option für Diensttyp, Fragment oder Time-to-live.	
128	17	Ungültiger Aufrufer für diesen Dienst.	
128	18	Ungültiges Paket.	
128	20	Diese Komponente wurde nicht aktiviert.	
128	21	Diese Komponente wurde bereits aktiviert.	
128	22	Überwachung ist für den angegebenen Port nicht aktiviert.	
128	26	Angeforderte Unterbrechung wurde abgebrochen.	
128	33	Server-IP-Adresse ist ungültig.	
128	34	Dieser Socket ist an einen anderen Port gebunden.	
128	35	Port ist an einen anderen Socket gebunden.	
128	36	Socket ist nicht gebunden.	
128	38	Die Socket-Bindung wurde während des Wartens auf Paketempfang aufgehoben.	
128	39	Socket wurde nicht erstellt.	
128	51	Keine zusätzliche Überwachungs-Anforderungsstrukturen verfügbar.	
128	52	Es ist bereits eine aktive Listenanforderung für diesen Port vorhanden.	
128	53	Socket befindet sich nicht im geschlossenen Zustand.	
128	54	Der angegebene Server-Socket befindet sich nicht im Überwachungszustand.	
128	55	Keine Wartezeit angegeben, Verbindungsversuch läuft.	
128	56	Verbindung fehlgeschlagen.	
128	57	Anforderung ist größer als die vom Empfänger angegebene zusätzliche Fenstergröße in Bytes.	
128	64	Ein anderer Thread ist unterbrochen. Es ist nur ein Thread zulässig.	

ErrorID-Code	SubErrorID-Code	Fehlerbeschreibung	Maßnahme
128	65	Trennung konnte nicht innerhalb des Timeouts abgeschlossen werden.	
128	66	Socket ist gebunden.	
128	69	Kein verfügbarer Port.	
128	70	Ungültiger Port.	
128	71	Es ist bereits ein gültiger Socket-Zeiger für diesen Port vorhanden, oder der angegebene Port verfügt nicht über eine aktive Überwachungsanforderung.	
128	72	Identisch mit **NX_SUCCESS**, außer eine Verbindungsanforderung in der Warteschlange wurde während dieses Aufrufs verarbeitet.	
128	73	Fehler durch Überschreiten der TCP-Übertragungs-Warteschlange.	

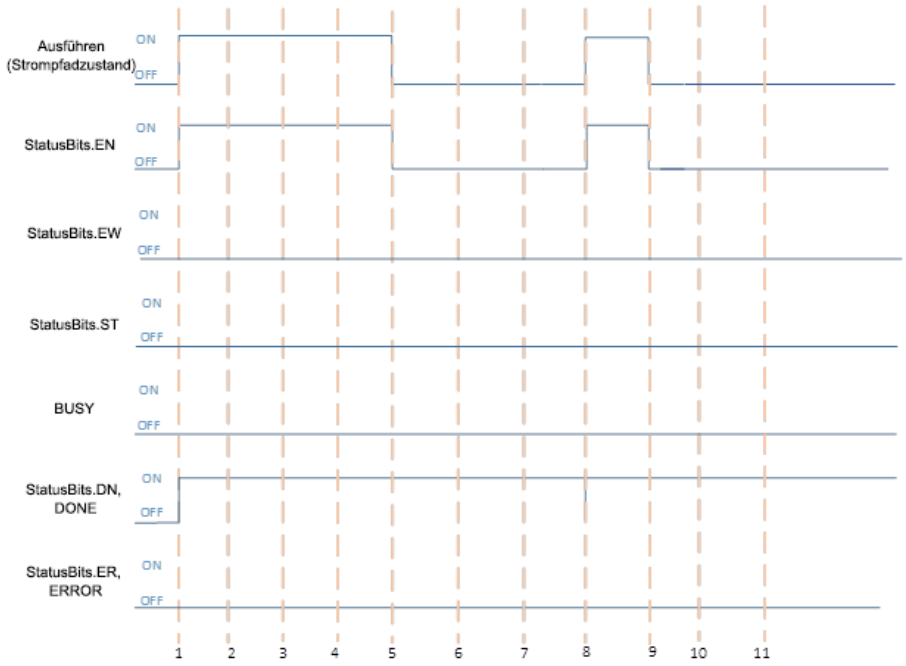
Statusbits für Socket-Anweisungen

In der folgenden Tabelle werden die Ausführungs-Statusbits für Socket-Befehle auf seite 565 beschrieben.

Bit-Zahl	Name	Beschreibung
0	EN - Aktivierungs-Bit	Das EN-Bit wird gesetzt, wenn die Anweisung aufgrund eines Wechsels von FALSE zu TRUE aktiviert ist, aber noch nicht abgeschlossen oder fehlerhaft ist.
1	EW - Aktivierungs-Bit der Warteschlange	Das EW-Bit wird gesetzt, wenn der Controller erkennt, dass eine Socket-Anweisungsanforderung in die Warteschlange eingegeben wurde. Der Controller setzt das EW-Bit zurück, wenn das ST-Bit gesetzt wurde.
2	ST - Start-Bit	Das ST-Bit wird gesetzt, wenn die Anweisung in der Warteschlange ausgeführt wird. Das ST-Bit wird zurückgesetzt, wenn das DN- oder ER-Bit gesetzt wird.
3	ER - Fehler-Bit	Gibt an, dass während der Ausführung der Anweisung ein Fehler aufgetreten ist. Das ER-Bit wird zurückgesetzt, wenn die Ausführungsbedingung das nächste Mal von False auf True wechselt.
4	DN - Fertig-Bit	Das DN-Bit wird gesetzt, wenn die Socket-Anweisung erfolgreich abgeschlossen wurde. Das DN-Bit wird zurückgesetzt, wenn die Ausführungsbedingung das nächste Mal von False auf True wechselt.

Socket-Anweisung für Timing-Diagramme

Erfolgreiche Ausführung für Socket-Befehle auf seite 565, wenn der Prozess unmittelbar erfolgt



Bedingung A: Strompfadbedingung ist TRUE während der Ausführung der Anweisungen

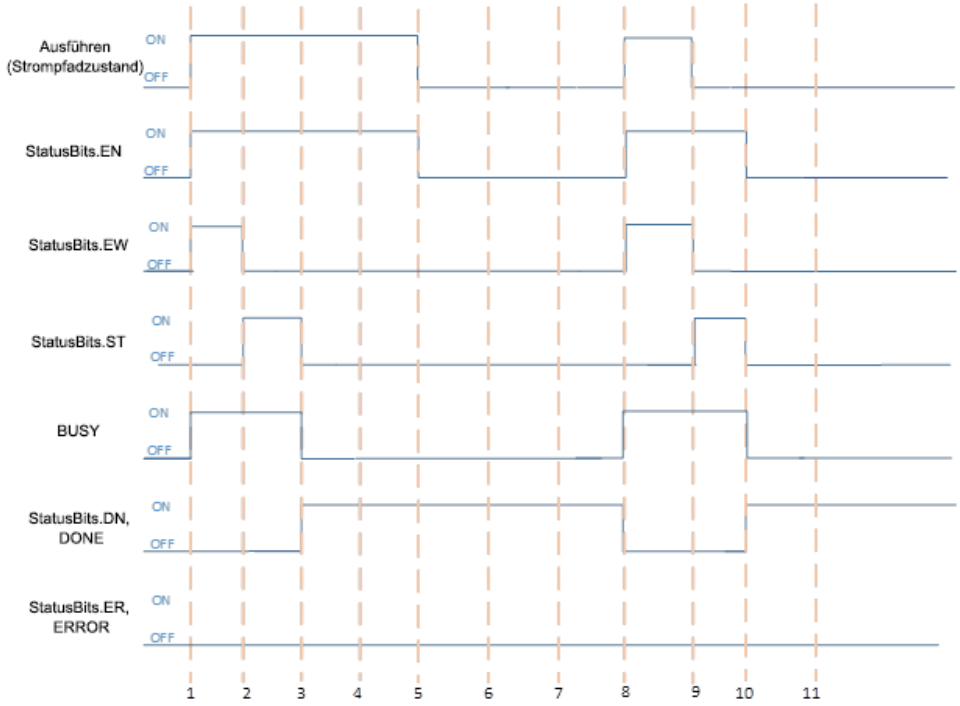
Strompfadbedingung	Beschreibung
1	Strompfadbedingung wird TRUE, wenn gilt: • Socket-Anweisungsausführung ist aktiviert. • EN-Bit wurde festgelegt. Alle anderen Bits wurden gelöscht. • Socket-Anweisung wurde erfolgreich ausgeführt. • Ausgabe für DN-Bit und DONE ist festgelegt.
2,3,4	Keine Änderung in Strompfadbedingung.
5	Strompfadbedingung wird FALSE, wenn EN-Bit gelöscht wird.
6,7	Keine Änderung in Strompfadbedingung.

Bedingung B: Strompfadbedingung ist FALSE während der Ausführung der Anweisungen

Strompfadbedingung	Beschreibung

Strompfadbedingung	Beschreibung
8	Strompfadbedingung wird TRUE, wenn gilt: • Socket-Anweisungsausführung ist aktiviert. • Socket-Anweisung wurde erfolgreich ausgeführt. • EN-Bit wurde festgelegt. Alle anderen Bits wurden gelöscht. • Ausgabe für DN-Bit und DONE ist festgelegt.
9	Strompfadbedingung wird FALSE, wenn gilt: • EN-Bit wurde gelöscht.
10.11	Keine Änderung in Strompfadbedingung.

Erfolgreiche Ausführung für Socket-Anweisungen, wenn der Prozess nicht unmittelbar erfolgt



Bedingung A: Strompfadbedingung ist TRUE während der Ausführung der Anweisungen

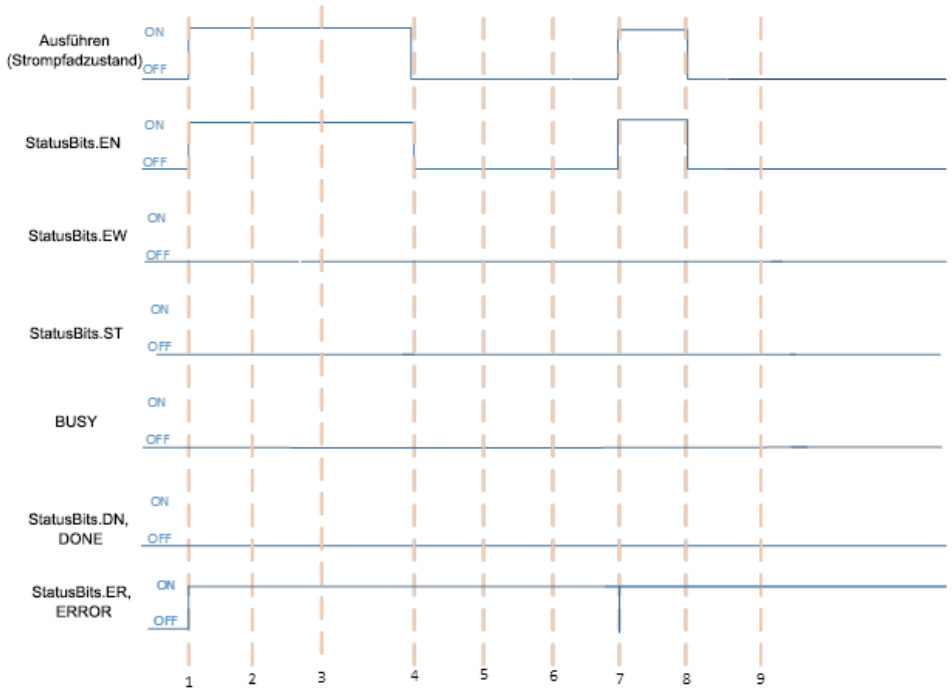
Strompfadbedingung	Beschreibung
1	Strompfadbedingung wird TRUE, wenn gilt: • Socket-Anweisungsausführung ist aktiviert. • EN-Bit wurde festgelegt. Alle anderen Bits wurden gelöscht. • Socket-Anweisungen führen Hintergrundverarbeitung aus. • Ausgabe für EW-Bit und BUSY ist festgelegt. • Gesperrte Socket-Anweisungs-Eingabeparameter für Hintergrundverarbeitung.

Strompfadbedingung	Beschreibung
2	Ausführung der Socket-Anweisungen wird gestartet, wenn gilt: • EW-Bit wurde gelöscht. • ST-Bit wurde festgelegt.
3	Ausführung der Socket-Anweisungen ist erfolgreich, wenn gilt: • Ausgabe für ST-Bit und BUSY wurde entfernt. • Ausgabe für DN-Bit und DONE ist festgelegt.
4	Keine Änderung in Strompfadbedingung.
5	Strompfadbedingung wird FALSE, wenn gilt: • EN-Bit wurde gelöscht.
6,7	Keine Änderung in Strompfadbedingung.

Bedingung B: Strompfadbedingung wechselt zu FALSE während der Ausführung der Anweisungen

Strompfadbedingung	Beschreibung
8	Strompfadbedingung wird TRUE, wenn gilt: • Socket-Anweisungsausführung ist aktiviert. • EN-Bit wurde festgelegt. Alle anderen Bits wurden gelöscht. • Socket-Anweisungen zur Hintergrundverarbeitung gesendet. • Ausgabe für EW-Bit und BUSY ist festgelegt. • Gesperrte Socket-Anweisungs-Eingabeparameter für Hintergrundverarbeitung.
9	Ausführung der Socket-Anweisungen wird gestartet, wenn gilt: • EW-Bit wurde gelöscht. • ST-Bit wurde festgelegt. • Strompfadbedingung wird FALSE.
10	Ausführung der Socket-Anweisungen ist erfolgreich, wenn gilt: • Ausgabe für ST-Bit und BUSY wurde entfernt. • Ausgabe für DN-Bit und DONE ist festgelegt. • Strompfadbedingung ist FALSE. • EN-Bit wurde gelöscht.

Socket-Anweisungen schlagen fehl, wenn EN = TRUE und EW und ST = FALSE sind



Bedingung A: Strompfadbedingung ist TRUE während der Ausführung der Anweisungen

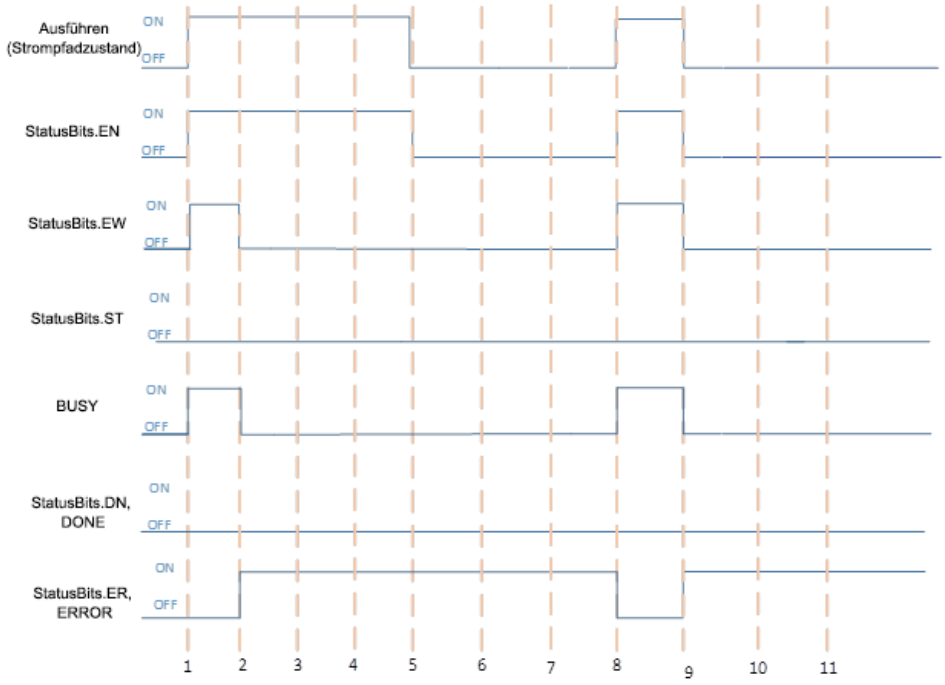
Strompfadbedingung	Beschreibung
1	Strompfadbedingung wird TRUE, wenn gilt: • Socket-Anweisungsausführung ist aktiviert. • EN-Bit wurde festgelegt. Alle anderen Bits wurden gelöscht. • Ausführung der Socket-Anweisungen ist fehlerhaft. • Ausgabe für ER-Bit und Fehler ist festgelegt.
2,3	Keine Änderung in Strompfadbedingung.
4	Strompfadbedingung wird FALSE, wenn gilt: • EN-Bit wurde gelöscht.
5,6	Keine Änderung in Strompfadbedingung.

Bedingung B: Strompfadbedingung wechselt zu FALSE während der Ausführung der Anweisungen

Strompfadbedingung	Beschreibung

Strompfadbedingung	Beschreibung
7	Strompfadbedingung wird TRUE, wenn gilt: • Socket-Anweisungsausführung ist aktiviert. • EN-Bit wurde festgelegt. Alle anderen Bits wurden gelöscht. • Ausführung der Socket-Anweisungen ist fehlerhaft. • Ausgabe für ER-Bit und Fehler ist festgelegt.
8	Strompfadbedingung wird FALSE, wenn gilt: • EN-Bit wurde gelöscht.
9	Keine Änderung in Strompfadbedingung.

Socket-Anweisungen schlagen fehl, wenn EW TRUE ist und die Anweisungsverarbeitung nicht unmittelbar erfolgt



Bedingung A: Strompfadbedingung ist TRUE während der Ausführung der Anweisungen

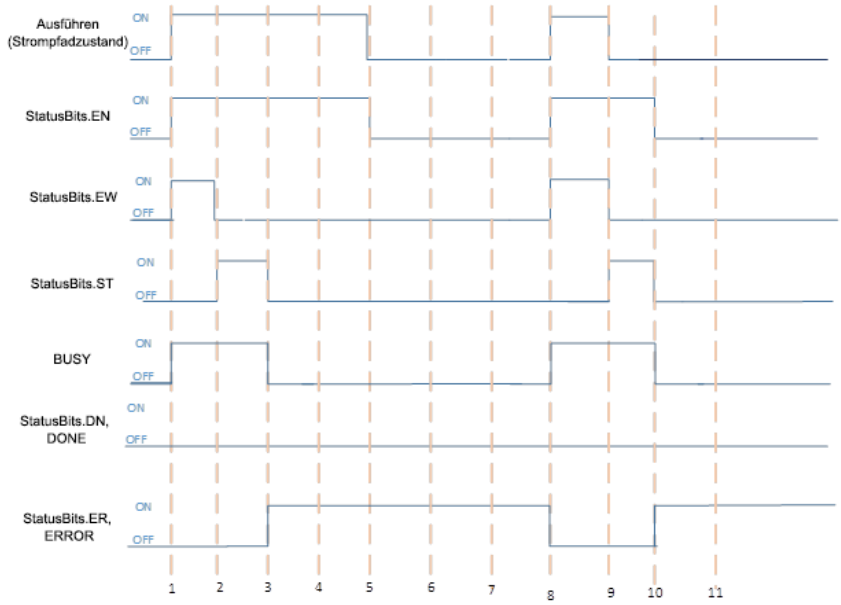
Strompfadbedingung	Beschreibung
1	Strompfadbedingung wird TRUE, wenn gilt: • Socket-Anweisungsausführung ist aktiviert. • EN-Bit wurde festgelegt. Alle anderen Bits wurden gelöscht. • Socket-Anweisungen zur Hintergrundverarbeitung gesendet. • Ausgabe für EW-Bit und BUSY ist festgelegt. • Gesperrte Socket-Anweisungs-Eingabeparameter für Hintergrundverarbeitung.

Strompfadbedingung	Beschreibung
2	Ausführung der Socket-Anweisungen ist fehlerhaft, wenn gilt: • Ausgabe für EW-Bit und BUSY wurde entfernt. • Ausgabe für ER-Bit und Fehler ist festgelegt.
3,4	Keine Änderung in Strompfadbedingung.
5	Strompfadbedingung wird FALSE, wenn gilt: • EN-Bit wurde gelöscht.
6,7	Keine Änderung in Strompfadbedingung.

Bedingung B: Strompfadbedingung wechselt zu FALSE während der Ausführung der Anweisungen

Strompfadbedingung	Beschreibung
8	Strompfadbedingung wird TRUE, wenn gilt: • Socket-Anweisungsausführung ist aktiviert. • EN-Bit wurde festgelegt. Alle anderen Bits wurden gelöscht. • Socket-Anweisungen zur Hintergrundverarbeitung gesendet. • Ausgabe für EW-Bit und BUSY ist festgelegt. • Gesperrte Socket-Anweisungs-Eingabeparameter für Hintergrundverarbeitung.
9	Ausführung der Socket-Anweisungen ist fehlerhaft, wenn gilt: • Ausgabe für EW-Bit und BUSY wurde entfernt. • Ausgabe für ER-Bit und Fehler ist festgelegt. • Strompfadbedingung wird FALSE. • EN-Bit wurde gelöscht.
10.11	Keine Änderung in Strompfadbedingung.

Socket-Anweisungen schlagen fehl, wenn ST TRUE ist und die Anweisungsverarbeitung nicht unmittelbar erfolgt



Bedingung A: Strompfadbedingung ist TRUE während der Ausführung der Anweisungen

Strompfadbedingung	Beschreibung
1	Strompfadbedingung wird TRUE, wenn gilt: • Socket-Anweisungsausführung ist aktiviert. • EN-Bit wurde festgelegt. Alle anderen Bits wurden gelöscht. • Socket-Anweisungen zur Hintergrundverarbeitung gesendet. • Ausgabe für EW-Bit und BUSY ist festgelegt. • Gesperrte Socket-Anweisungs-Eingabeparameter für Hintergrundverarbeitung.
2	Ausführung der Socket-Anweisungen wird gestartet, wenn gilt: • EW-Bit wurde gelöscht • ST-Bit wurde festgelegt.
3	Ausführung der Socket-Anweisungen ist fehlerhaft, wenn gilt: • Ausgabe für ST-Bit und BUSY wurde entfernt. • Ausgabe für ER-Bit und Fehler ist festgelegt.
4	Keine Änderung in Strompfadbedingung.
5	Strompfadbedingung wird FALSE, wenn gilt: • EN-Bit wurde gelöscht.
6,7	Keine Änderung in Strompfadbedingung.

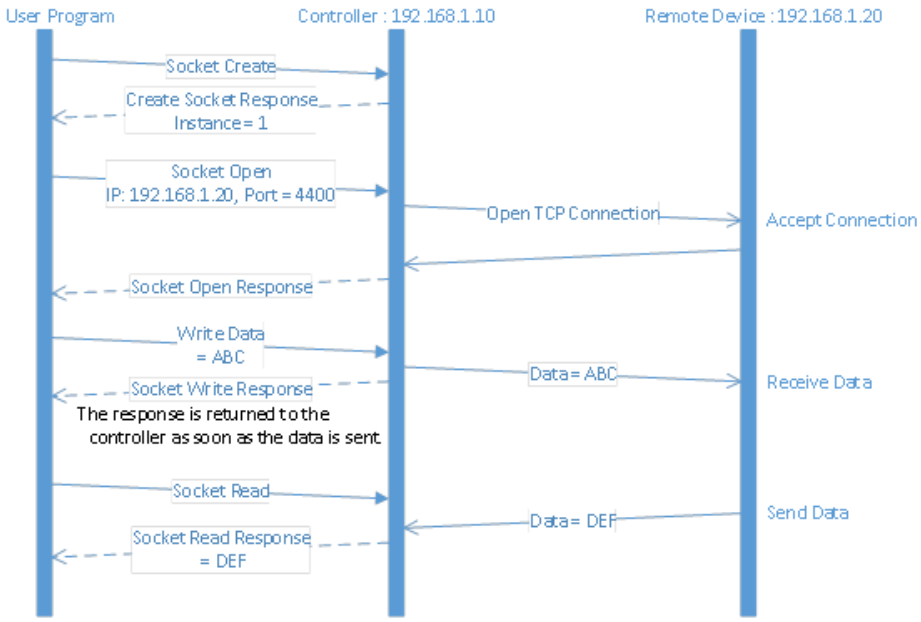
Bedingung B: Strompfadbedingung wechselt zu FALSE während der Ausführung der Anweisungen

Strompfadbedingung	Beschreibung
8	Strompfadbedingung wird TRUE, wenn gilt: • Socket-Anweisungsausführung ist aktiviert. • EN-Bit wurde festgelegt. Alle anderen Bits wurden gelöscht. • Socket-Anweisungen zur Hintergrundverarbeitung gesendet. • Ausgabe für EW-Bit und BUSY ist festgelegt. • Gesperrte Socket-Anweisungs-Eingabeparameter für Hintergrundverarbeitung.
9	Ausführung der Socket-Anweisungen wird gestartet, wenn gilt: • EW-Bit wurde gelöscht und ST-Bit wurde festgelegt.
10	Ausführung der Socket-Anweisungen ist fehlerhaft, wenn gilt: • Ausgabe für ST-Bit und BUSY wurde entfernt. • Ausgabe für ER-Bit und Fehler ist festgelegt. • Strompfadbedingung wird FALSE. • EN-Bit wurde gelöscht.
11	Keine Änderung in Strompfadbedingung.

Socket-Anweisung für Transaktionsdiagramme

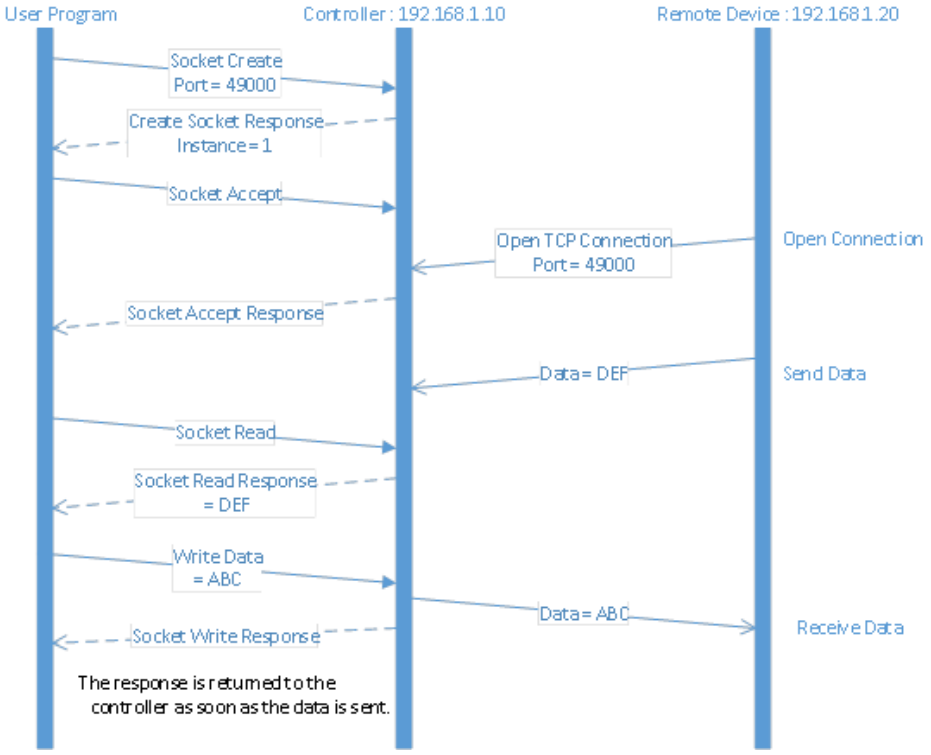
Transaktionen für TCP-Client

Das folgende Diagramm zeigt eine typische Sequenz von Socket-Schnittstellentransaktionen, wenn der Controller als TCP-Client agiert.



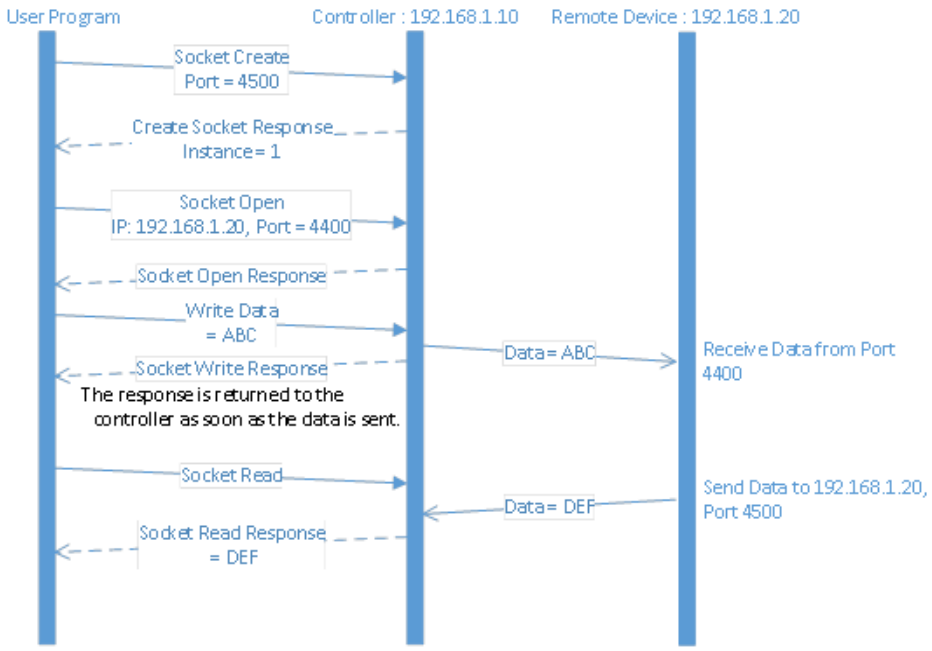
Transaktionen für TCP-Server

Das folgende Diagramm zeigt eine typische Sequenz von Socket-Schnittstellentransaktionen, wenn der Controller als TCP-Server agiert.



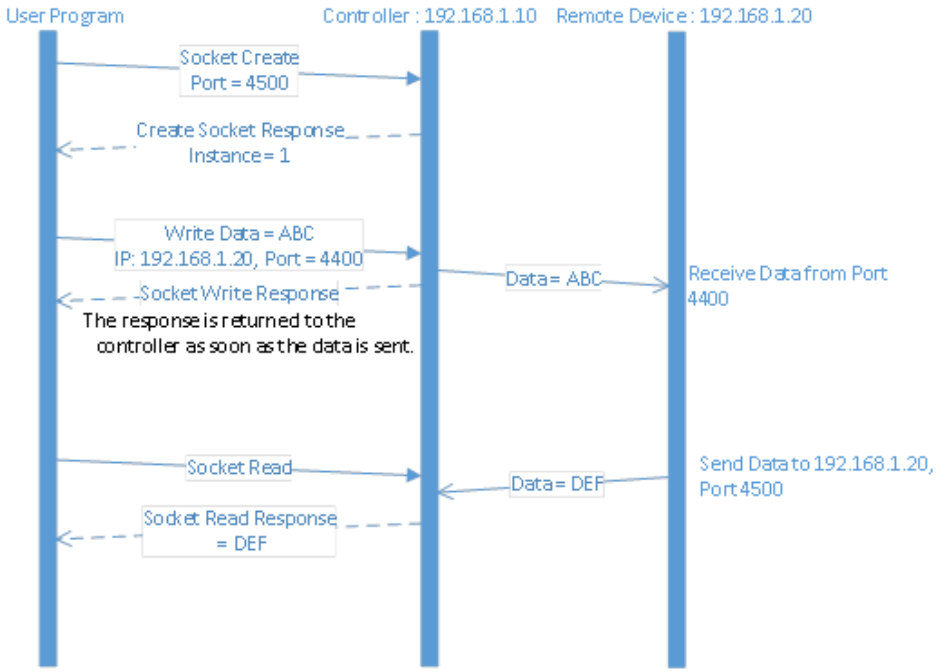
Transaktionen für UDP mit geöffneter Verbindung

Das folgende Diagramm zeigt eine typische Sequenz von Socket-Schnittstellentransaktionen für die UDP-Kommunikation, wenn ein Dienst mit geöffneter Verbindung verwendet wird, um die Zieladresse festzulegen.



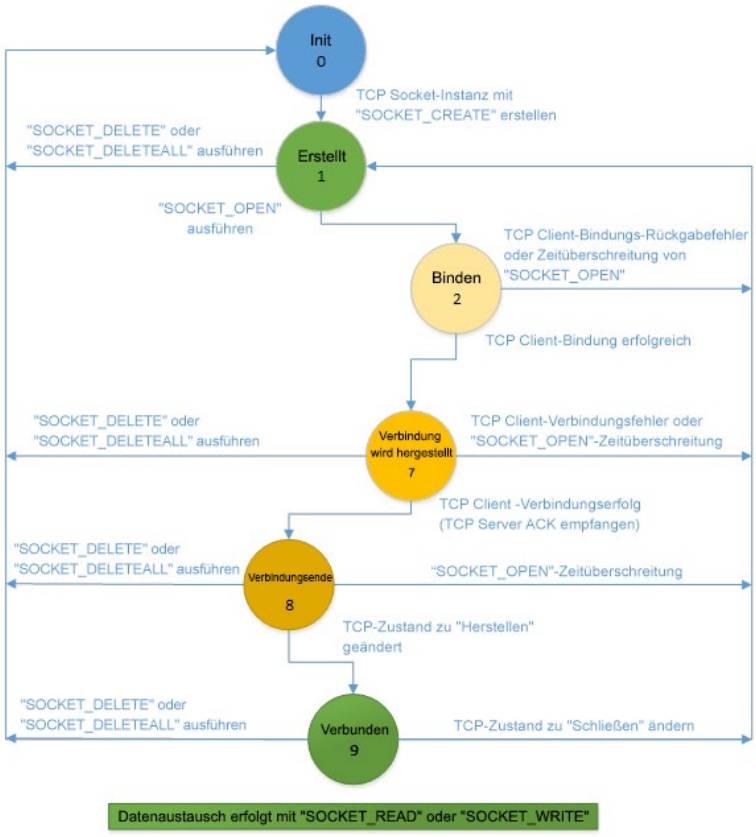
Transaktionen für UDP ohne geöffnete Verbindungen

Das folgende Diagramm zeigt eine typische Sequenz von Socket-Schnittstellentransaktionen für die UDP-Kommunikation, wenn kein Dienst mit geöffneter Verbindung verwendet wird, um die Zieladresse festzulegen. In diesem Fall gibt der Controller das Ziel für jedes Datagramm an und empfängt die Absenderadresse zusammen mit jedem erhaltenen Datagramm.

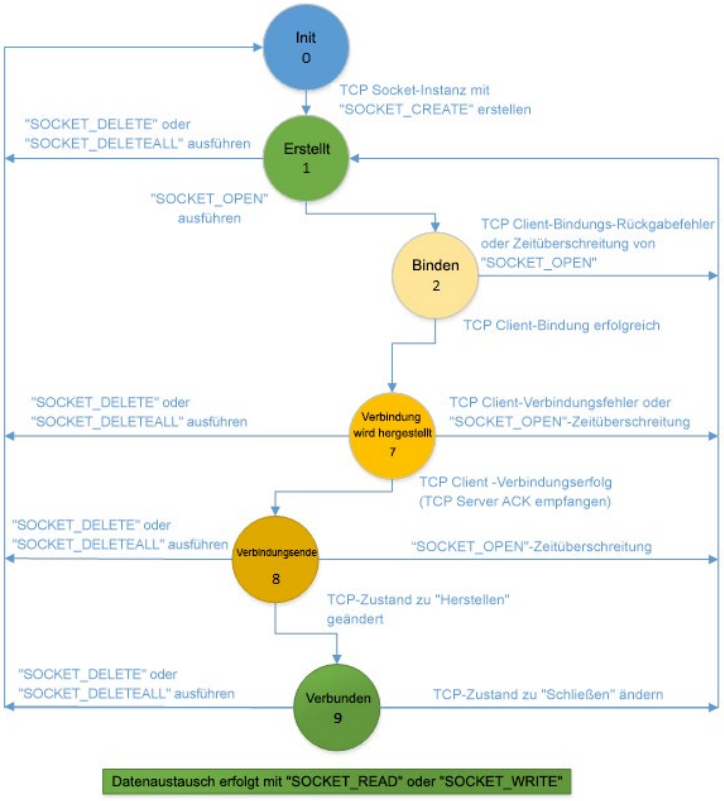


Maschinendiagramme für TCP

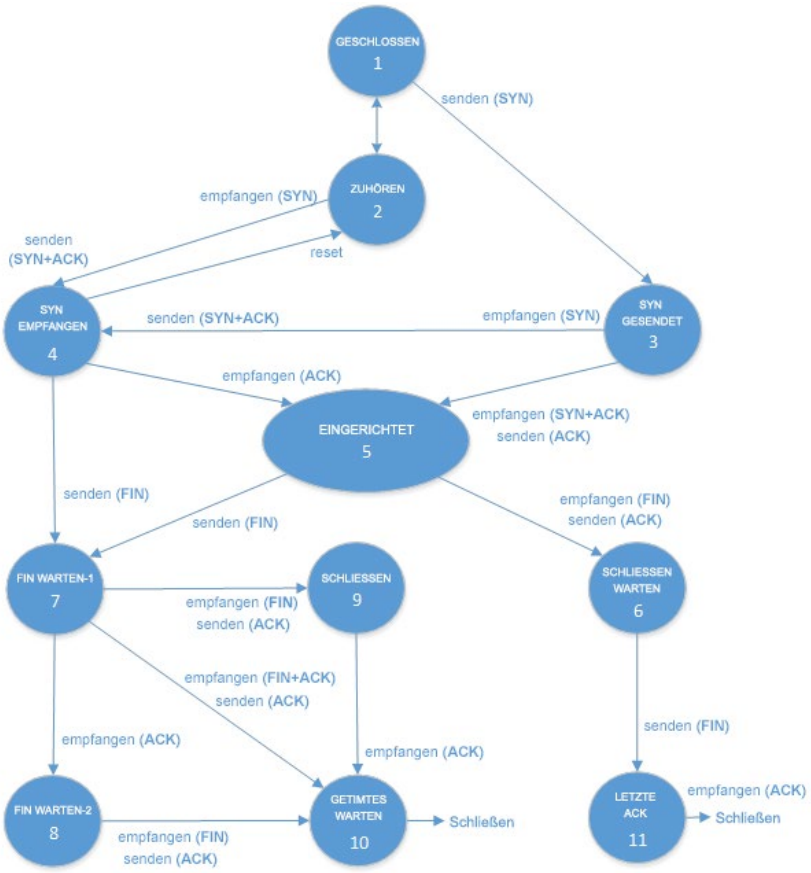
TCP-Client mit



TCP-Server mit Socket-Anweisungen

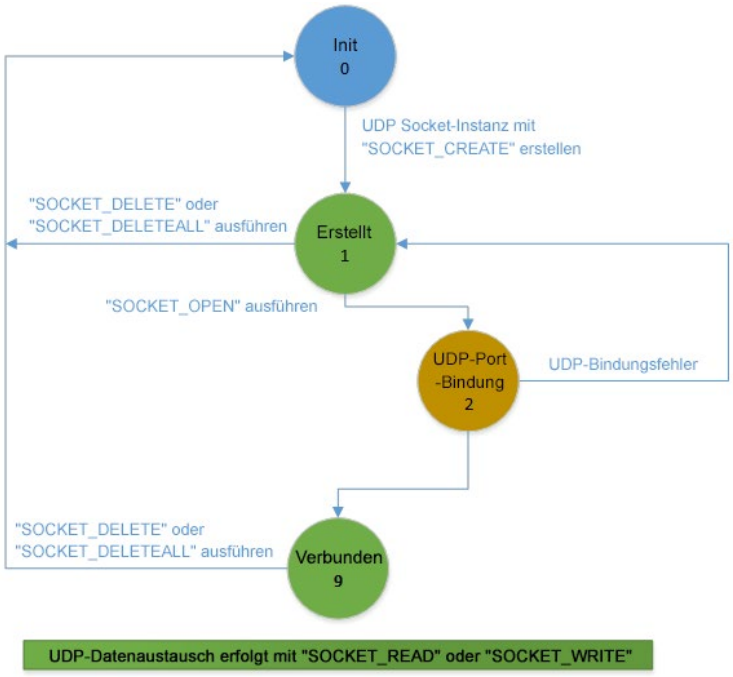


TCP-Zustandsdiagramm

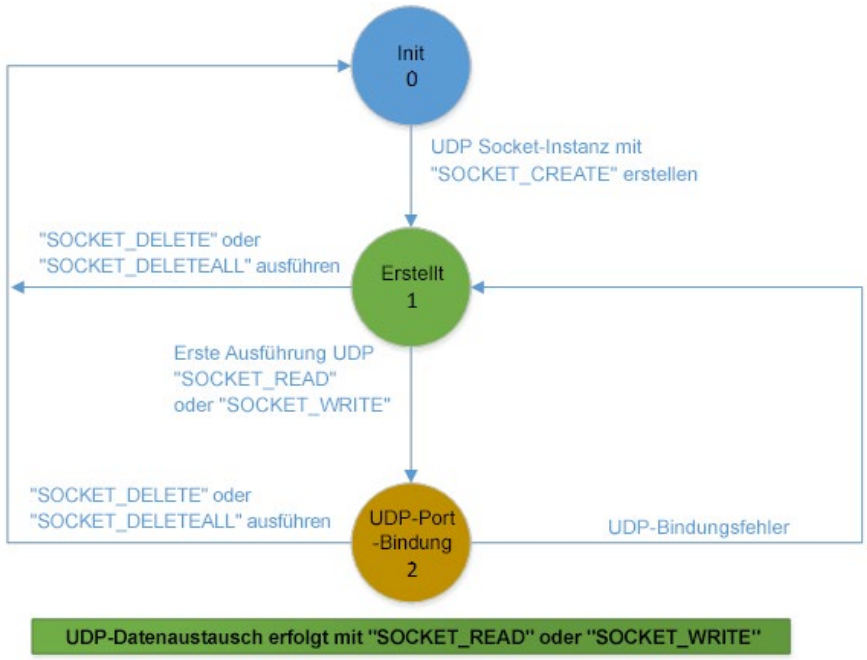


Maschinendiagramme für UDP

UDP-Datagramm mit SOCKET_OPEN-Anweisung



UDP-Datagramm ohne SOCKET_OPEN-Anweisung



Anweisungen zur Zeichenfolgenbearbeitung

Mithilfe von Anweisungen zur Zeichenfolgenbearbeitung können Sie eine Sequenz von Symbolen ändern, die aus einer Liste oder einem Alphabet ausgewählt wurden, um den Ausgangsstatus zu ändern. Geben Sie zum korrekten Lesen von Eingangszeichenfolgen mit Sonderzeichen die Zeichen der Zeichenfolge ein, nachdem das Programm mit der Funktionsblockinstanz online ist.

Anweisung	Beschreibung
ASCII auf seite 615	Gibt den ASCII-Code für Zeichen in Zeichenfolgen zurück. Zeichen -> ASCII-Code
CHAR auf seite 617	Gibt eine Zeichenfolge mit einem Zeichen für einen ASCII-Code zurück. ASCII-Code -> Zeichen
DELETE auf seite 618	Löscht Zeichen aus einer Zeichenfolge.
FIND auf seite 620	Sucht die Position von Teilzeichenfolgen innerhalb von Zeichenfolgen und stellt diese bereit.
INSERT auf seite 622	Fügt Teilzeichenfolgen auf benutzerdefinierten Positionen innerhalb von Zeichenfolgen ein.
LEFT auf seite 624	Extrahiert Zeichen von der linken Seite einer Zeichenfolge.
MID auf seite 626	Extrahiert Zeichen aus der Mitte einer Zeichenfolge.
MLEN auf seite 628	Berechnet die Länge der Zeichenfolge.
REPLACE auf seite 631	Ersetzt Teile einer Zeichenfolge mit einer neuen Gruppe von Zeichen.
RIGHT auf seite 629	Extrahiert Zeichen aus der rechten Seite einer Zeichenfolge.

ASCII

Gibt den ASCII-Code für Zeichen in Zeichenfolgen zurück.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

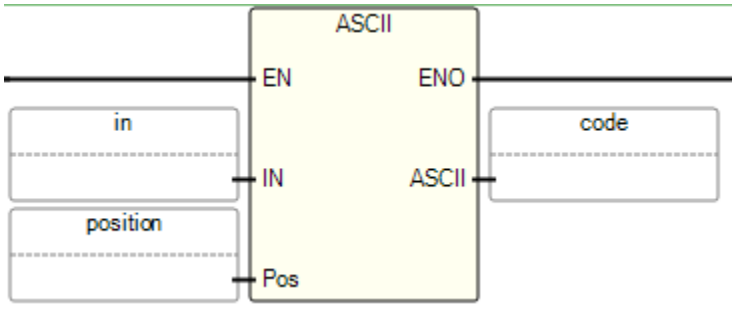
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Anzeige des ASCII-Codes für Zeichen. FALSE - keine Anzeige. Gilt für Kontaktplan-Programme.

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	STRING	Beliebige nicht-leere Zeichenfolge.
Pos	Eingang	DINT	Position des ausgewählten Zeichens im Bereich [1.. len] („len" ist die Länge der IN-Zeichenfolge).
ASCII	Ausgang	DINT	Code des ausgewählten Zeichens (im Bereich [0 .. 255]). Bei 0 ist die Position außerhalb der Zeichenfolge.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für ein ASCII-Funktionsblockdiagramm

Beispiel für einen ASCII-Kontaktplan



Beispiel für strukturierten ASCII-Text

```
1  position := 1;
2  code := ASCII(in, position);
```

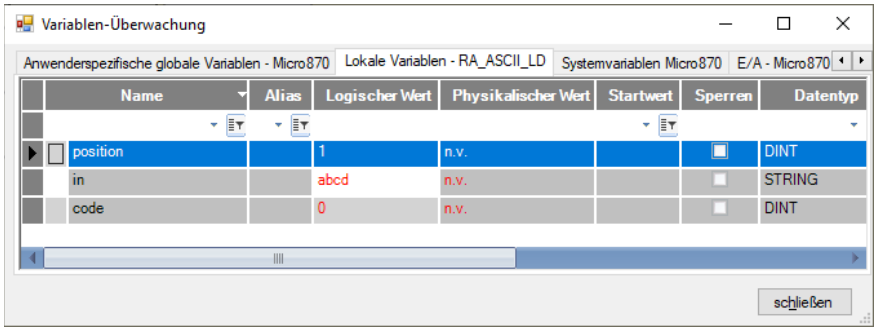
```
ASCII(
    DINT ASCII(STRING IN, DINT Pos)
    Zeichen -> ASCII-Code
```

(* ST-Äquivalenz: *)

FirstChr := ASCII (message, 1);

(* FirstChr ist der ASCII-Code des ersten Zeichens der Zeichenfolge *)

Ergebnisse



CHAR (ASCII-Code zu Zeichenfolge)

Gibt eine Zeichenfolge mit einem Zeichen für einen ASCII-Code zurück. ASCII-Code -> Zeichen

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

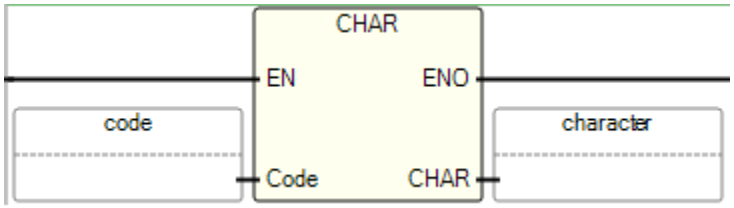
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - es wird eine Zeichenfolge mit einem Zeichen ausgegeben. FALSE – Keine Aktion. Gilt für Kontaktplan-Programme.
Code	Eingang	DINT	ASCII-Code im Bereich [0 .. 255].
CHAR	Ausgang	STRING	Zeichenfolge (String) bestehend aus einem Zeichen. Der ASCII-Code des Zeichens wird als Eingangscode angegeben.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

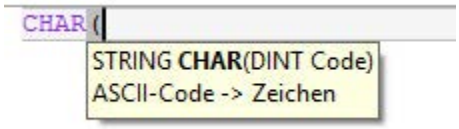
Diagrammbeispiel für Funktionsblock CHAR

CHAR-Kontaktplan-Beispiel



Beispiel für CHAR in strukturiertem Text

```
1  code := 97;
2  character := CHAR(code);
```



(* ST-Äquivalenz: *)

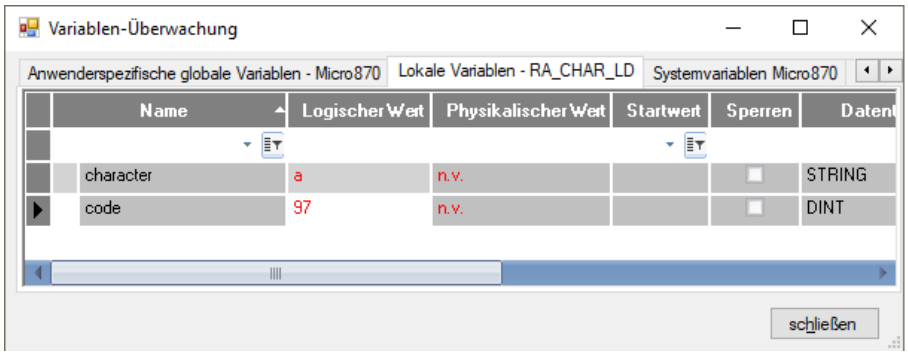
Display := CHAR (value + 48);

(* Wert liegt im Bereich [0..9] *)

(* 48 ist der ASCII-Code für '0' *)

(* Ergebnis ist eine Zeichenfolge (String) mit einem Zeichen von '0' bis '9' *)

Ergebnisse



DELETE (Löschen einer Teilzeichenfolge)

Löscht Zeichen aus einer Zeichenfolge.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

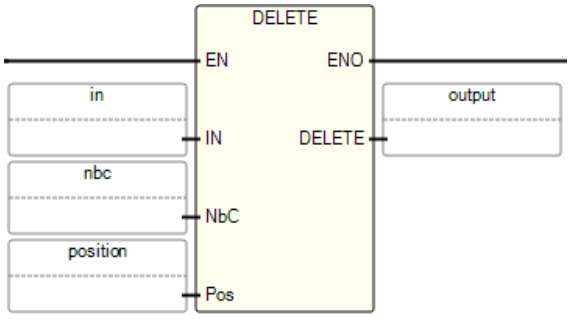
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

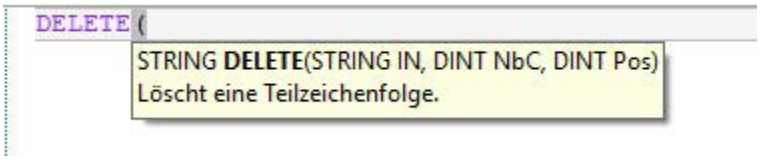
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - der angegebene Teil der Zeichenfolge wird gelöscht. FALSE – Keine Aktion. Gilt für Kontaktplan-Programme.
IN	Eingang	STRING	Beliebige nicht-leere Zeichenfolge.
NbC	Eingang	DINT	Anzahl der zu löschenden Zeichen.
Pos	Eingang	DINT	Position des ersten gelöschten Zeichens (das erste Zeichen der Zeichenfolge hat Position 1).
DELETE	Ausgang	STRING	Ausgabe ist: • eine geänderte Zeichenfolge. • eine leere Zeichenfolge (wenn „Pos" < 1). • die Anfangszeichenfolge (wenn „Pos" > Länge der IN-Zeichenfolge). • die Anfangszeichenfolge (wenn „NbC" <= 0).
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Diagrammbeispiel für Funktionsblock DELETE

DELETE-Kontaktplan-Beispiel



Beispiel für DELETE in strukturiertem Text



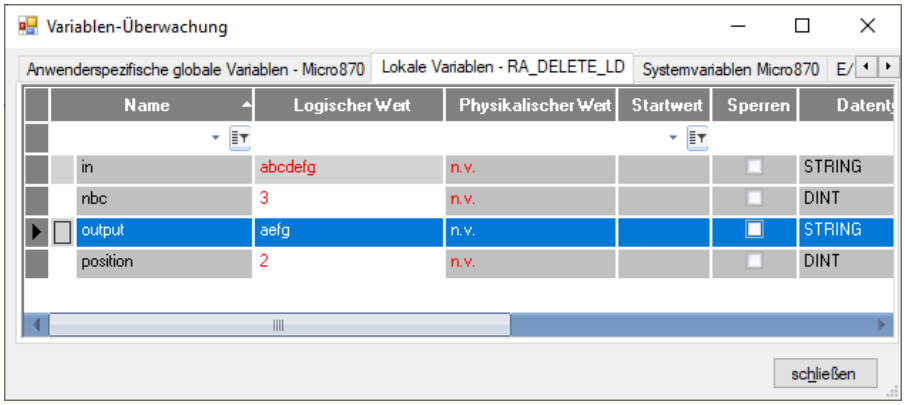

```
1  nbc := 3;
2  position := 2;
3  output := DELETE(in, nbc, position);
```

(* ST-Äquivalenz: *)

complete_string := INSERT ('ABCD ', 'EFGH', 5); (* complete_string ist 'ABCDEFGH ' *)

sub_string := DELETE (complete_string, 4, 3); (* sub_string is 'ABGH '*)

Ergebnisse



FIND (Teilzeichenfolge suchen)

Sucht die Position von Teilzeichenfolgen innerhalb von Zeichenfolgen und stellt diese bereit.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

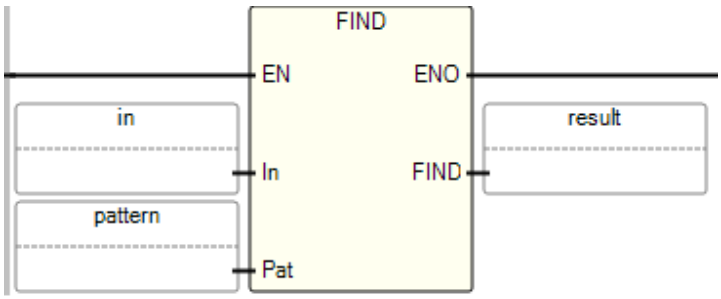
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

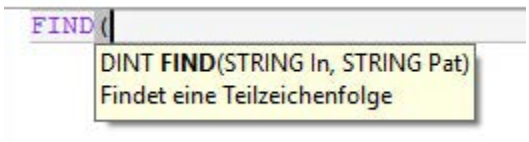
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - es wird eine Position innerhalb von Zeichenfolgen gesucht. FALSE - es wird keine Position gesucht. Gilt für Kontaktplan-Programme.
In	Eingang	STRING	Beliebige nicht-leere Zeichenfolge.
Pat	Eingang	STRING	Beliebige nicht-leere Zeichenfolge (Muster).
FIND	Ausgang	DINT	Ausgabe ist: • 0, falls die Teilzeichenfolge „Pat" nicht gefunden wurde. • Position des ersten Zeichens des ersten Vorkommens der Teilzeichenfolge „Pat" (erste Position ist 1) Diese Anweisung berücksichtigt Groß- und Kleinschreibung.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Diagrammbeispiel für Funktionsblock FIND

FIND-Kontaktplan-Beispiel



Beispiel für FIND in strukturiertem Text

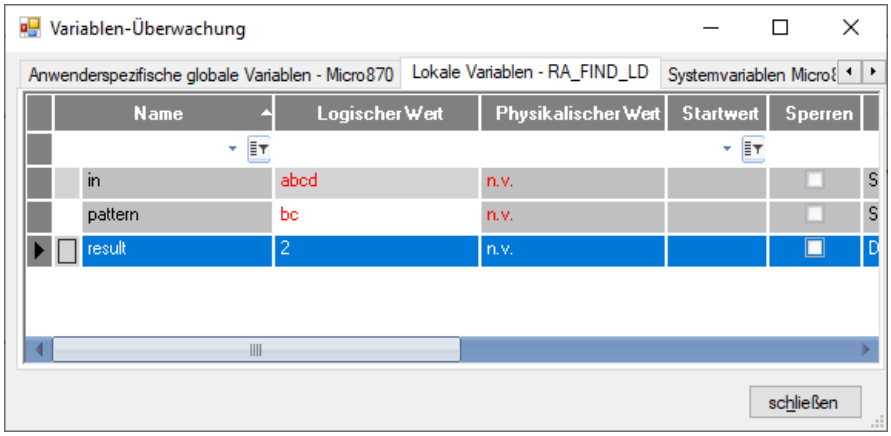



```
1  result := FIND(in, pattern);
```

(* ST-Äquivalenz: *)

```
complete_string := 'ABCD' + 'EFGH'; (* complete_string ist 'ABCDEFGH ' *)
found := FIND (complete_string, 'CDEF'); (* found ist 3 *)
```

Ergebnisse



INSERT (Zeichenfolge einfügen)

Fügt Teilzeichenfolgen auf benutzerdefinierten Positionen innerhalb von Zeichenfolgen ein.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

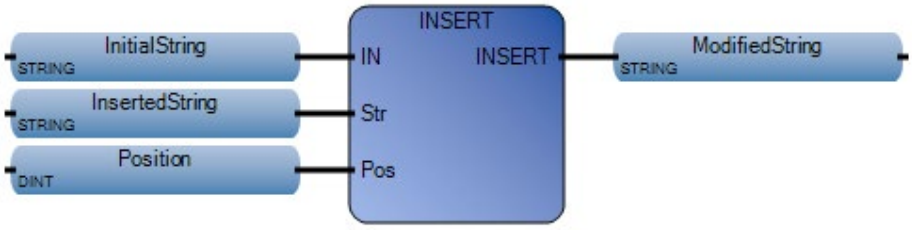
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

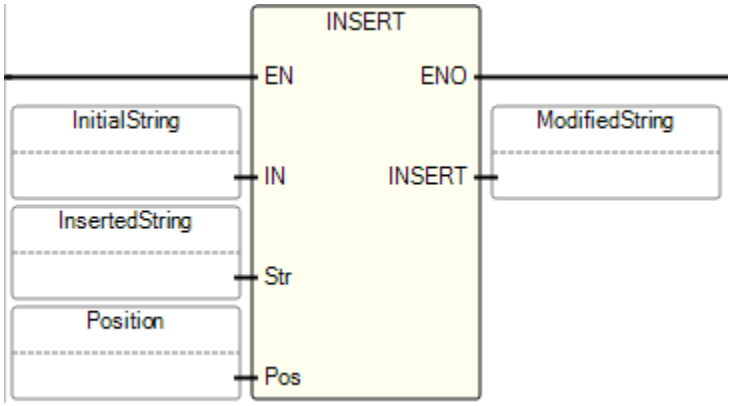
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Einfügen einer Teilzeichenfolge in eine Zeichenfolge. FALSE – Keine Aktion. Gilt für Kontaktplan-Programme.
IN	Eingang	STRING	Anfangszeichenfolge.
Str	Eingang	STRING	Einzufügende Zeichenfolge.
Pos	Eingang	DINT	Einfügeposition Dabei wird die Zeichenfolge vor der angegebenen Position eingefügt (Die erste gültige Position ist 1).
INSERT	Ausgang	STRING	Geänderte Zeichenfolge. Mögliche Werte: • leere Zeichenfolge, wenn „Pos" < 0 • Verknüpfung beider Zeichenfolgen, wenn die angegebene Position die Länge der IN-Zeichenfolge überschreitet

Parameter	Parametertyp	Datentyp	Beschreibung
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

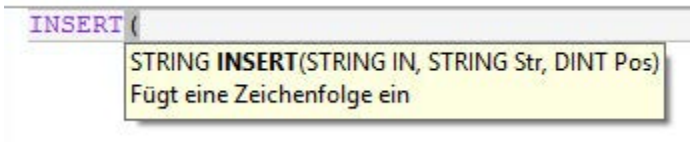
Diagrammbeispiel für Funktionsblock INSERT



INSERT-Kontaktplan-Beispiel



Beispiel für INSERT in strukturiertem Text



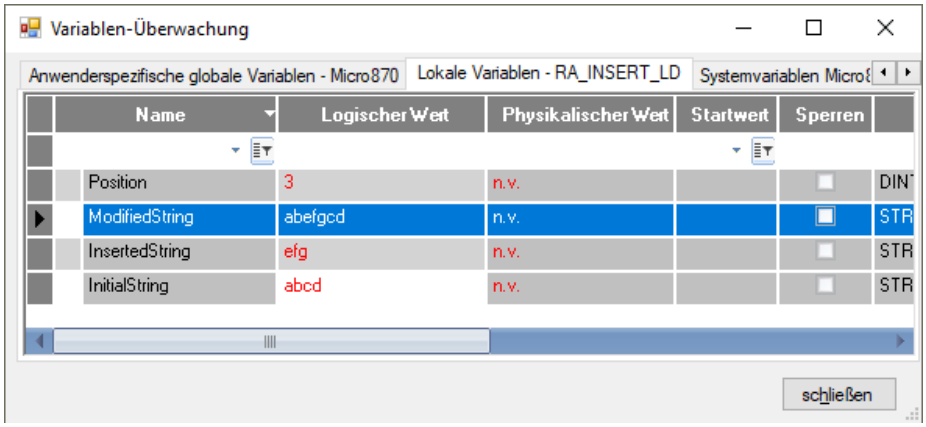

```
1  Position := 3;
2  ModifiedString := INSERT(InitialString, InsertedString, Position);
```

(* ST-Äquivalenz: *)

MyName := INSERT ('Mr JONES', 'Frank ', 4);

(* MyName ist 'Mr Frank JONES' *)

Ergebnisse



LEFT (extrahiert den linken Teil einer Zeichenfolge)

Extrahieren von Zeichen von der linken Seite einer Zeichenfolge.

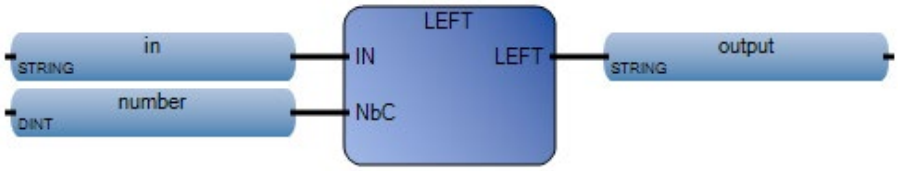
Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

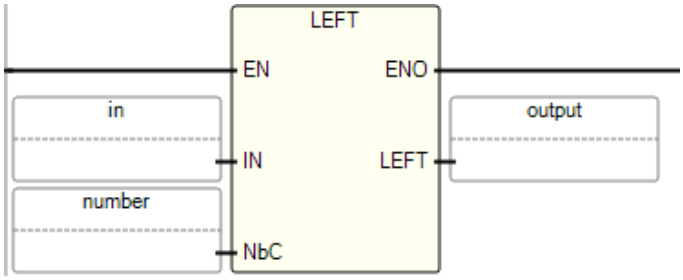
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Berechnung der Zeichenanzahl von der linken Seite der Zeichenfolge. FALSE – Keine Aktion. Gilt für Kontaktplan-Programme.
IN	Eingang	STRING	Beliebige nicht-leere Zeichenfolge.
NbC	Eingang	DINT	Anzahl der zu extrahierenden Zeichen. Diese Zahl kann nicht größer sein als die Länge der IN-Zeichenfolge.
LEFT	Ausgang	STRING	Linker Teil der IN-Zeichenfolge (ihre Länge = NbC). Mögliche Werte: • leere Zeichenfolge, wenn „NbC“ <= 0 • vollständige IN-Zeichenfolge, wenn „NbC“ >= Länge der IN-Zeichenfolge
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

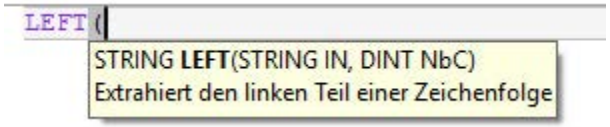
Beispiel für ein LEFT-Funktionsblockdiagramm



LEFT-Kontaktplan-Beispiel



Beispiel für LEFT in strukturiertem Text




```
1  number := 3;
2  output := LEFT(in, number);
```

(* ST-Äquivalenz: *)

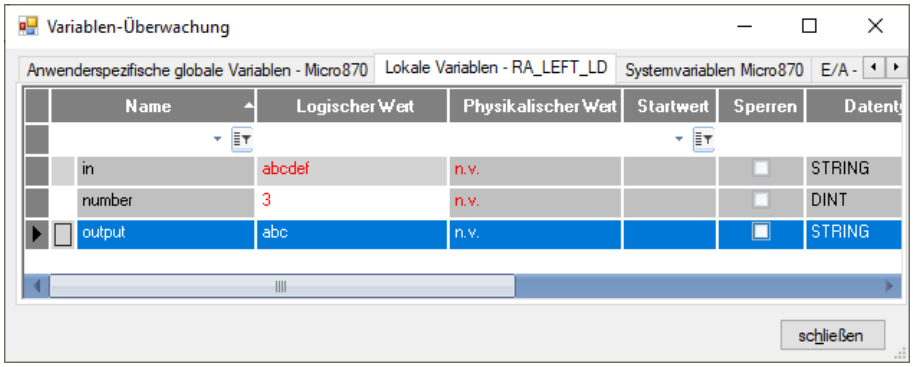
complete_string := RIGHT ('12345678', 4), LEFT ('12345678', 4), 5;

(* complete_string ist '56781234'

Der vom Aufruf RIGHT ausgegebene Wert ist '5678'.

Der vom Aufruf LEFT ausgegebene Wert ist '1234'. *)

Ergebnisse



MID (Extrahiert den mittleren Teil einer Zeichenfolge)

Extrahiert Zeichen aus der Mitte einer Zeichenfolge. Berechnet mithilfe der gegebenen Position und Anzahl von Zeichen die gewünschten Bestandteile von Zeichenfolgen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

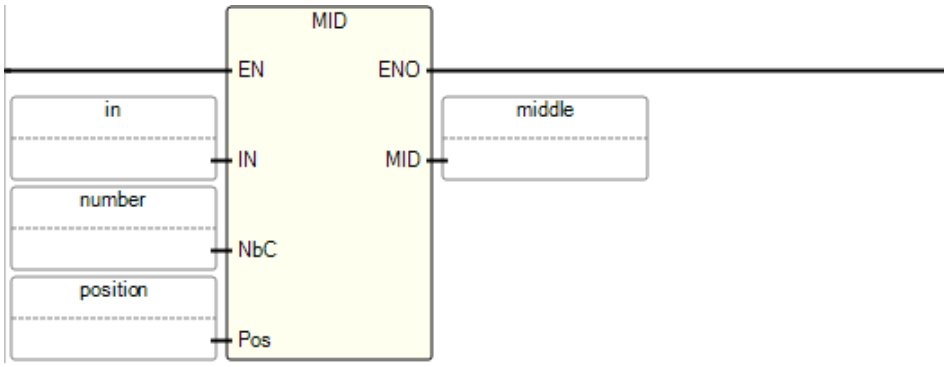
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Teil einer Zeichenfolge generieren. FALSE – kein Generierungsvorgang. Gilt für Kontaktplan-Programme.
IN	Eingang	STRING	Beliebige nicht-leere Zeichenfolge.
NbC	Eingang	DINT	Die Anzahl der zu extrahierenden Zeichen kann nicht größer sein als die Länge der IN-Zeichenfolge.
Pos	Eingang	DINT	Position der Teilzeichenfolge. Dabei gibt der Parameter „Pos“ die Position des ersten Zeichens der Teilzeichenfolge an (die erste gültige Position ist 1).
MID	Ausgang	STRING	Mittlerer Teil der Zeichenfolge (ihre Länge = NbC). Wenn die Anzahl der zu extrahierenden Zeichen die Länge der IN-Zeichenfolge überschreitet, wird „NbC“ automatisch neu berechnet, um nur den verbleibenden Rest der Zeichenfolge zu berücksichtigen. Wenn „NbC“ oder „Pos“ Null betragen oder einen negativen Wert haben, wird eine leere Zeichenfolge zurückgegeben.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispiel für ein MID-Funktionsblockdiagramm

MID-Kontaktplan-Beispiel



Beispiel für MID in strukturiertem Text

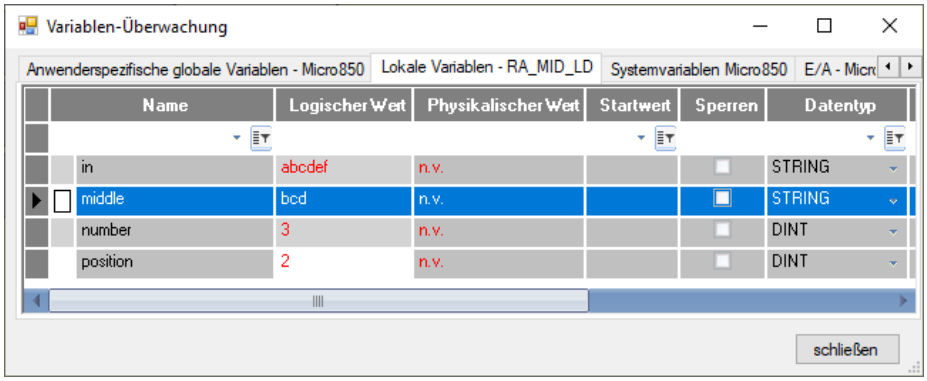
```
1  number := 3;
2  position := 2;
3  middle := MID(in, number, position);
```

(* ST-Äquivalenz: *)

sub_string := MID ('abcdefgh', 2, 4);

(* sub_string ist 'de' *)

Ergebnisse



MLEN (Zeichenfolgenlänge)

Berechnet die Länge der Zeichenfolge.

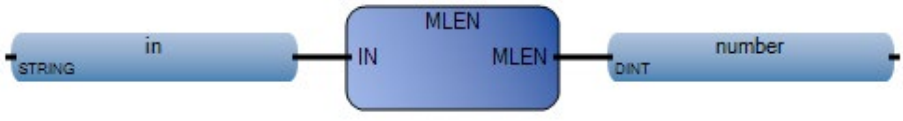
Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

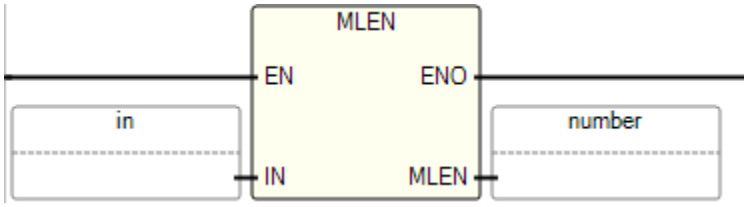
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Länge der Zeichenfolge berechnen. FALSE – Keine Aktion. Gilt für Kontaktplan-Programme.
IN	Eingang	STRING	Beliebige Zeichenfolge.
MLEN	Ausgang	DINT	Zeichenzahl der IN-Zeichenfolge.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

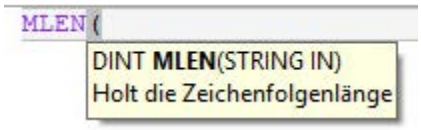
Diagrammbeispiel für Funktionsblock MLEN



MLEN-Kontaktplan-Beispiel



Beispiel für MLEN in strukturiertem Text




```
1  number := MLEN(in);
```

(* ST-Äquivalenz: *)

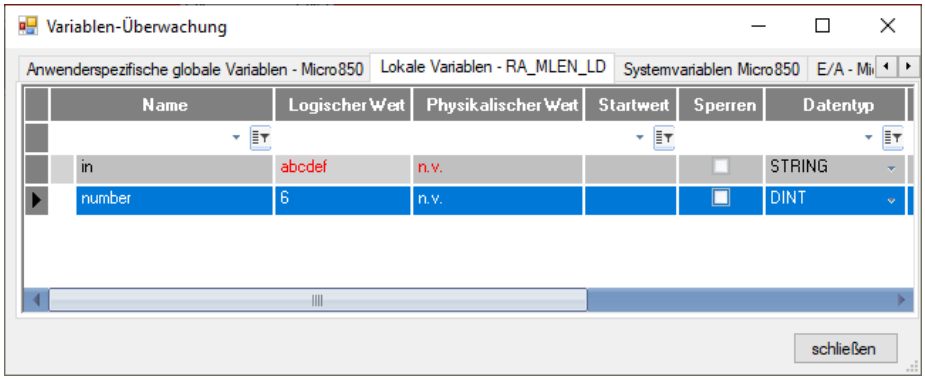
nbchar := MLEN (complete_string);

If (nbchar < 3) Then Return; End_if;

prefix := LEFT (complete_string, 3);

(* Dieses Programm extrahiert die 3 Zeichen auf der linken Seite der Zeichenfolge und schreibt das Ergebnis in die Präfix-Variable der Zeichenfolge. Wenn die Länge der Zeichenfolge weniger als 3 Zeichen beträgt, wird keine Aktion durchgeführt *)

Ergebnisse



RIGHT (rechten Teil einer Zeichenfolge extrahieren)

Extrahiert den Zeichen auf der rechten Seite einer Zeichenfolge.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

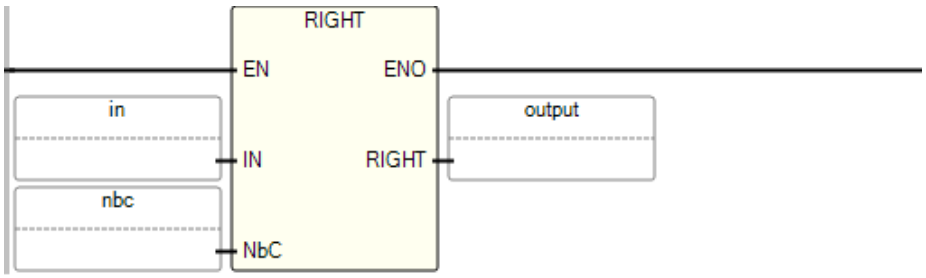
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE – Die angegebene Anzahl Zeichen vom rechten Ende der Zeichenfolge aus gesehen wird extrahiert. FALSE – Keine Aktion. Gilt für Kontaktplan-Programme.
IN	Eingang	STRING	Beliebige nicht-leere Zeichenfolge.
NbC	Eingang	DINT	Anzahl der zu extrahierenden Zeichen. Diese Zahl kann nicht größer sein als die Länge der IN-Zeichenfolge.
RIGHT	Ausgang	STRING	Rechter Teil der Zeichenfolge (Länge = NbC). Mögliche Werte: • leere Zeichenfolge, wenn „NbC“ <= 0 • vollständige Zeichenfolge, wenn „NbC“ >= Länge der Zeichenfolge
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

Beispieldiagramm mit einem RIGHT-Funktionsblock

Beispiel eines RIGHT-Kontaktplans



Beispiel eines strukturierten RIGHT-Texts

```
1  nbc := 2;
2  output := RIGHT(in, nbc);
```

(* ST-Äquivalenz: *)

complete_string := RIGHT ('12345678', 4), LEFT ('12345678', 4),5;

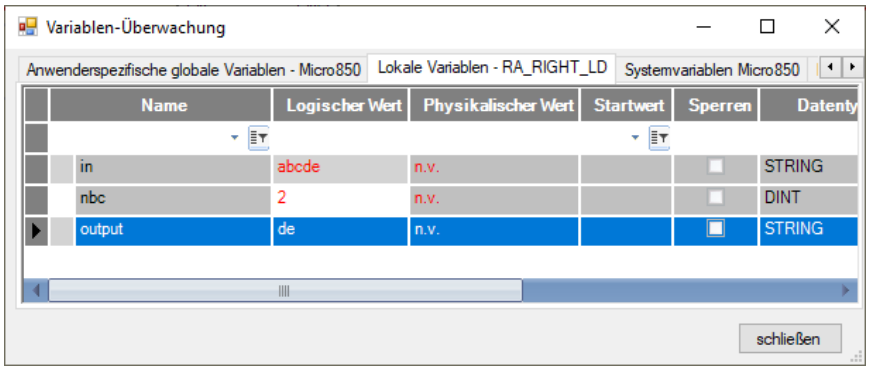
(* complete_string ist '56781234'

Der vom Aufruf RIGHT ausgegebene Wert ist '5678'.

Der vom Aufruf LEFT ausgegebene Wert ist '1234'.

*)

Ergebnisse

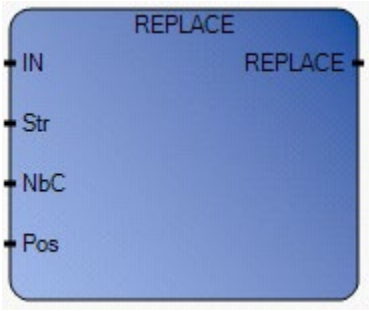


REPLACE (Ersetzen einer Teilzeichenfolge)

Ersetzt Teile einer Zeichenfolge mit einer neuen Gruppe von Zeichen.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

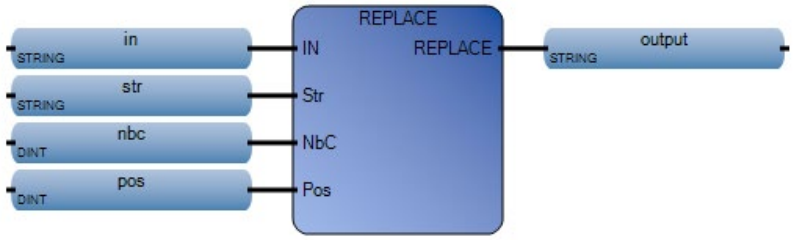
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



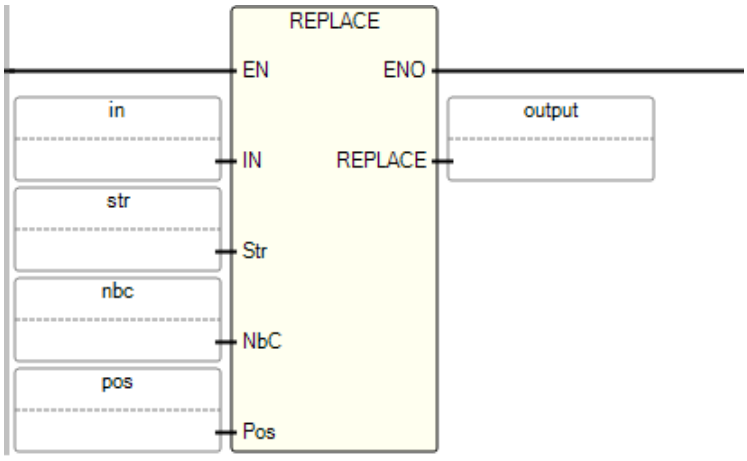
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Funktion aktivieren. TRUE – Teile einer Zeichenfolge werden durch neue Zeichen ersetzt. FALSE – Keine Aktion. Gilt für Kontaktplan-Programme.
IN	Eingang	STRING	Beliebige Zeichenfolge.
Str	Eingang	STRING	Einzufügende Zeichenfolge (als Ersatz für „NbC“ Zeichen).
NbC	Eingang	DINT	Anzahl der zu löschenden Zeichen.
Pos	Eingang	DINT	Position des ersten zu ändernden Zeichens (erste gültige Position ist 1).
REPLACE	Ausgang	STRING	Geänderte Zeichenfolge. Die Zeichenzahl NbC wird ab Position Pos gelöscht. Anschließend wird die Teilzeichenfolge Str an dieser Position eingefügt. Mögliche Werte: • leere Zeichenfolge, wenn „Pos“ < 0 • Verknüpfung beider Zeichenfolgen (IN+Str), wenn „Pos“ die Länge der IN-Zeichenfolge überschreitet • Anfangszeichenfolge IN, wenn „NbC“ <= 0
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.

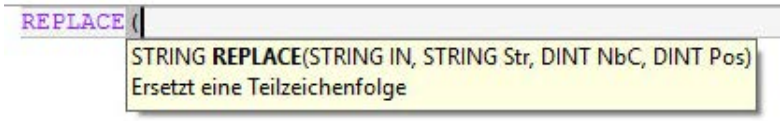
Beispiel eines REPLACE-Funktionsblockdiagramms



Beispiel eines REPLACE-Kontaktplandiagramms



Beispiel eines strukturierten REPLACE-Texts




```
1  nbc := 4;
2  pos := 2;
3  output := REPLACE(in, str, nbc, pos);
```

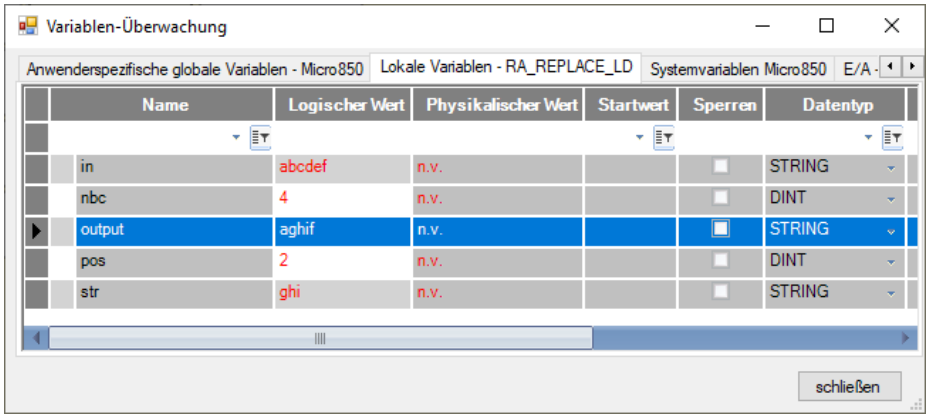
Ersetzt einen Teil einer Zeichenfolge mit einer neuen Gruppe von Zeichen.

(* ST-Äquivalenz: *)

MyName := REPLACE ('Mr X JONES, 'Frank', 1, 4);

(* MyName ist 'Mr Frank JONES' *)

Ergebnisse



Timer-Anweisungen

Verwenden Sie Timer-Anweisungen zur Steuerung von Operationen auf Basis der Zeit.

Anweisung	Beschreibung
TOF auf seite 635	Zeitliche Steuerung der Ausschaltverzögerung. Setzt einen internen Timer um einen vorgegebenen Wert nach oben.
TON auf seite 637	Zeitliche Steuerung der Einschaltverzögerung. Setzt einen internen Timer um einen vorgegebenen Wert nach oben.
TONOFF auf seite 639	Verzögert das Einschalten eines Ausgangs in einem Strompfad mit dem Wert TRUE und verzögert anschließend das Ausschalten des Ausgangs im Strompfad mit dem Wert FALSE.
TP auf seite 642	Zeitliche Impulssteuerung. Setzt einen internen Timer bei einer ansteigenden Flanke um einen gegebenen Wert nach oben.
RTO auf seite 644	Remanenzzeit. Erhöht einen internen Timer, wenn der Eingang aktiv ist, setzt aber den internen Timer nicht zurück, wenn der Eingang inaktiv wird.
DOY auf seite 646	Aktiviert eine Ausgabe, wenn der Wert der Echtzeituhr sich im Bereich der Jahres- und Zeiteinstellung befindet.
TDF auf seite 648	Berechnet die Zeitdifferenz zwischen TimeA und TimeB.
TOW auf seite 650	Aktiviert eine Ausgabe, wenn der Wert der Echtzeituhr sich im Bereich der Wochenzeiteinstellung befindet.

Konfiguration der Timer-Anweisungen

Die Genauigkeit der Zeit bezieht sich auf die Zeit zwischen dem Moment, wenn der Prozessor eine Timer-Anweisung aktiviert und dem Moment, wenn der Prozessor das Zeitintervall abschließt.

Der Prozessor verwendet folgende Informationen aus dem Timer-Befehl auf seite 635:

- **Timer** - Die Adresse der Timer-Steuerung im Timer-Bereich der Datenspeicherung.
- **Zeitbasis** - Bestimmt, wie der Timer betrieben wird.
- **Voreinstellung** - Spezifiziert den Wert, den der Timer erreichen muss, bevor der Prozessor das „Done“-Bit einstellt.
- **Akkumulierter Wert** - Die Anzahl der Zeitschritte, die die Anweisung gezählt hat. Bei einer Aktivierung aktualisiert der Timer diesen Wert kontinuierlich.

TOF (Timer, Ausschaltverzögerung)

Setzt einen internen Timer um einen vorgegebenen Wert nach oben.

Details zum Betrieb:

- Falls der Parameter „EN“ mit diesem Anweisungsblock verwendet wird, beginnt der Timer mit dem Inkrementieren, sobald „EN“ auf TRUE gesetzt wird, und setzt das Inkrementieren auch dann fort, wenn „EN“ auf FALSE gesetzt wird.

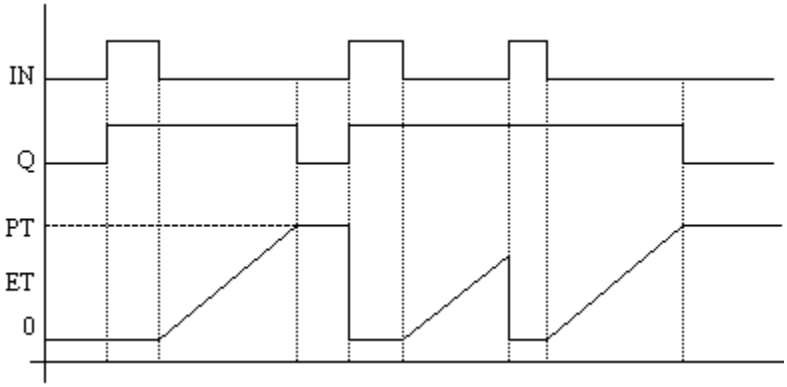
Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

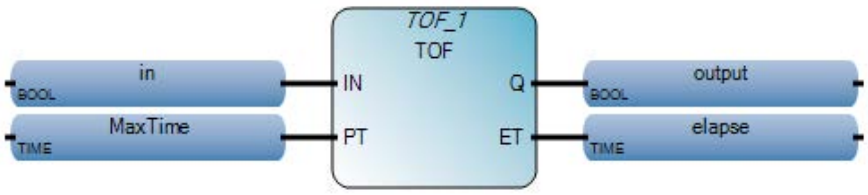
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangssteuerung. TRUE – Bei Abstiegflanke beginnt das Hochzählen des internen Timers. FALSE – Bei Anstiegsflanken wird der interne Timer angehalten und zurückgesetzt.
PT	Eingang	TIME	Maximal programmierte Dauer. Siehe Datentyp „Time“.
Q	Ausgang	BOOL	TRUE– Gesamtzeit ist **nicht** abgelaufen. FALSE – Gesamtzeit ist abgelaufen.
ET	Ausgang	TIME	Derzeit abgelaufene Zeit. Mögliche Werte reichen von 0ms bis 1193h2m47s294ms.

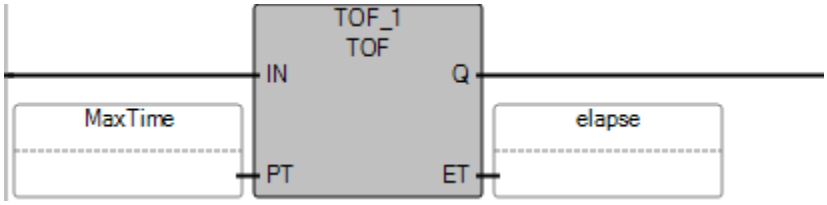
TOF-Timing-Diagramm



Beispiel für ein TOF-Funktionsblockdiagramm



Beispiel für einen TOF-Kontaktplan



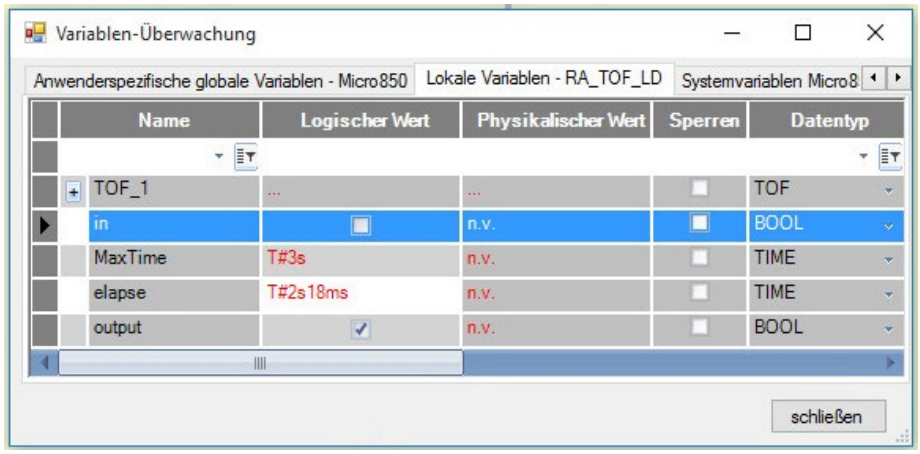
Beispiel für einen strukturierten TOF-Text

```
TOF_1(
```

void **TOF_1**(BOOL **IN**, TIME **PT**)
Typ: TOF, Zeitliche Steuerung der Ausschaltverzögerung

```
1  MaxTime := T#3s;
2  TOF_1(in, MaxTime);
3  output := TOF_1.Q;
4  elapse := TOF_1.ET;
```

Ergebnisse



TON (Timer, Einschaltverzögerung)

Setzt einen internen Timer um einen vorgegebenen Wert nach oben.

Details zum Betrieb:

- Verwenden Sie keinen Sprung, um einen „TON“-Anweisungsblock in einem Kontaktplan (KOP) zu überspringen. Wenn ein Sprung verwendet wird, zählt der TON-Timer nach Ablauf der vorgesehenen Zeitdauer weiter.
- Beispiel: Strompfad 1 enthält ein Sprungelement; Strompfad 2 enthält einen TON-Anweisungsblock mit einer abgelaufenen Zeitdauer von 10 Sekunden. Wenn Sie einen Sprung von Strompfad 1 auf Strompfad 3 aktivieren und das Sprungelement nach 30 Sekunden deaktivieren, beträgt die abgelaufene Zeitdauer 30 Sekunden – und nicht 10 Sekunden, wie in der abgelaufenen Zeit definiert.

- Falls der Parameter „EN“ mit TON verwendet wird, beginnt der Timer mit dem Inkrementieren, sobald „EN“ auf TRUE gesetzt wird, und setzt das Inkrementieren auch dann fort, wenn „EN“ auf FALSE gesetzt wird.

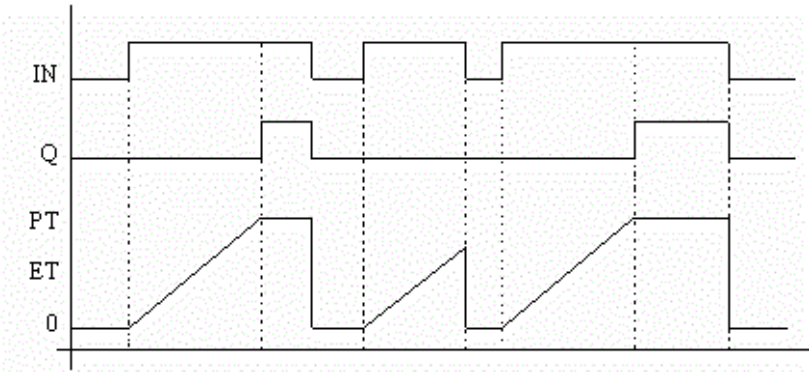
Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

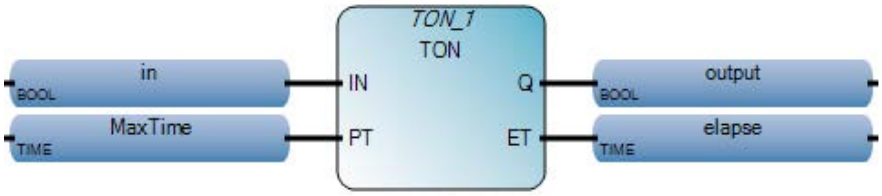
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangssteuerung. TRUE – Bei Anstiegsflanken beginnt das Hochzählen des internen Timers. FALSE – Bei Abstiegsflanken wird der interne Timer angehalten und zurückgesetzt.
PT	Eingang	TIME	Maximale programmierte Dauer, die mit dem Datentyp „Time“ definiert wurde.
Q	Ausgang	BOOL	TRUE – Programmierte Zeit ist abgelaufen. FALSE – Programmierte Zeit ist nicht abgelaufen.
ET	Ausgang	TIME	Derzeit abgelaufene Zeit. Mögliche Werte reichen von 0ms bis 1193h2m47s294ms.

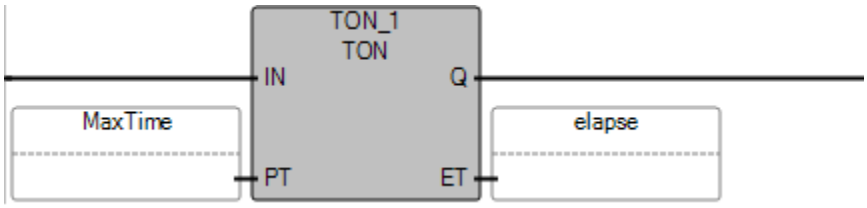
TON-Timing-Diagramm



Beispiel für ein TON-Funktionsblockdiagramm



Beispiel für einen TON-Kontaktplan



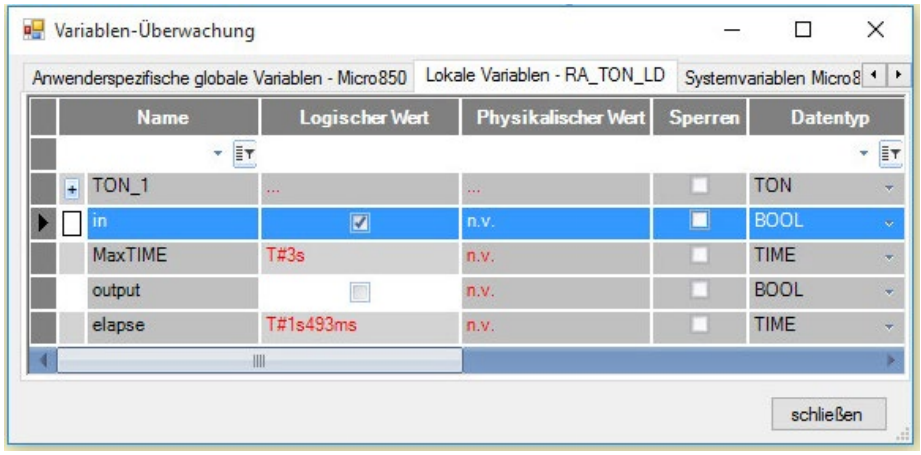
Beispiel für einen strukturierten TON-Text

```
TON_1(
```

void **TON_1**(BOOL **IN**, TIME **PT**)
Typ: TON, Zeitliche Steuerung der Einschaltverzögerung

```
1  MaxTime := T#3s;
2  TON_1(in, MaxTime);
3  output := TON_1.Q;
4  elapse := TON_1.ET;
```

Ergebnisse



TONOFF (Zeitverzögerung, eingeschaltet, ausgeschaltet)

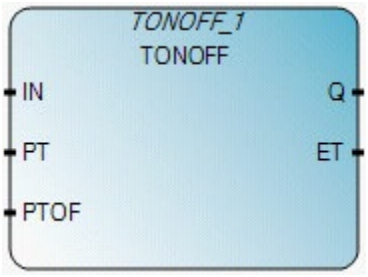
Verzögert das Einschalten eines Ausgangs auf einem korrekten Strompfad und verzögert anschließend das Ausschalten des Ausgangs auf dem falschen Strompfad.

Details zum Betrieb:

- Falls der Parameter „EN“ mit TONOFF verwendet wird, beginnt der Timer mit dem Inkrementieren, sobald „EN“ auf TRUE gesetzt wird, und setzt das Inkrementieren auch dann fort, wenn „EN“ auf FALSE gesetzt wird.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

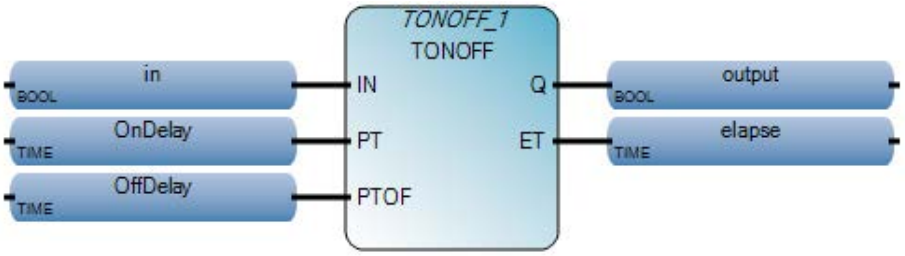
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



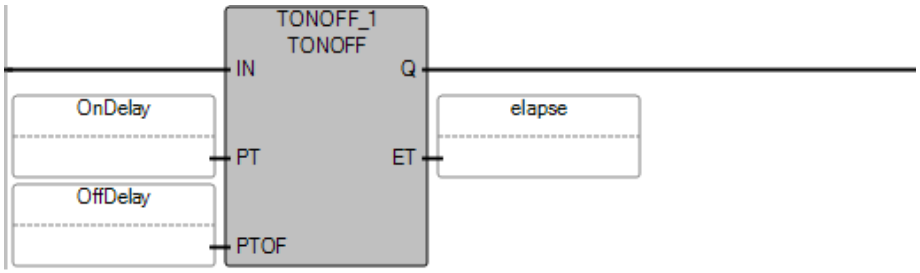
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangssteuerung. TRUE – Anstiegflanke (IN von 0 zu 1) erkannt: • Starten Sie den Einschaltverzögerung (PT). • Wenn die programmierte Off-Delay-Zeit (PTOF) nicht abgelaufen, starten Sie den On-Delay-Timer (PT). FALSE – Absteigsflanke (IN von 1 zu 0) erkannt: • Wenn die programmierte On-Delay-Zeit (PT) nicht abgelaufen ist, stoppen Sie den PT-Timer und setzen Sie ET zurück. • Wenn die programmierte On-Delay-Zeit (PT) abgelaufen ist, starten Sie dden Off-Delay-Timer (PTOF).
PT	Eingang	TIME	Die On-Delay-Zeiteinstellung wird mit Datentyp „Time“ definiert.
PTOF	Eingang	TIME	Die Off-Delay-Zeiteinstellung wird mit Datentyp „Time“ definiert.
Q	Ausgang	BOOL	TRUE – die programmierte On-Delay-Zeit ist abgelaufen, und die programmierte Off-Delay-Zeit ist nicht abgelaufen.
ET	Ausgang	TIME	Derzeit abgelaufene Zeit. Mögliche Werte reichen von 0ms bis 1193h2m47s294ms. Wenn die programmierte On-Delay-Zeit abgelaufen ist und der Off-Delay-Timer noch nicht gestartet ist, bleibt die abgelaufene Zeit (ET) auf dem On-Delay-Wert (PT). Wenn die programmierte Zeit für die Abschaltverzögerung abgelaufen ist und der Timer für die Abschaltverzögerung nicht startet, behält die abgelaufene Zeit (ET) den Wert der Abschaltverzögerung, bis die Anstiegsflanke erneut auftritt.

Beispiel für ein TONOFF-Funktionsblockdiagramm



Beispiel für einen TONOFF-Kontaktplan



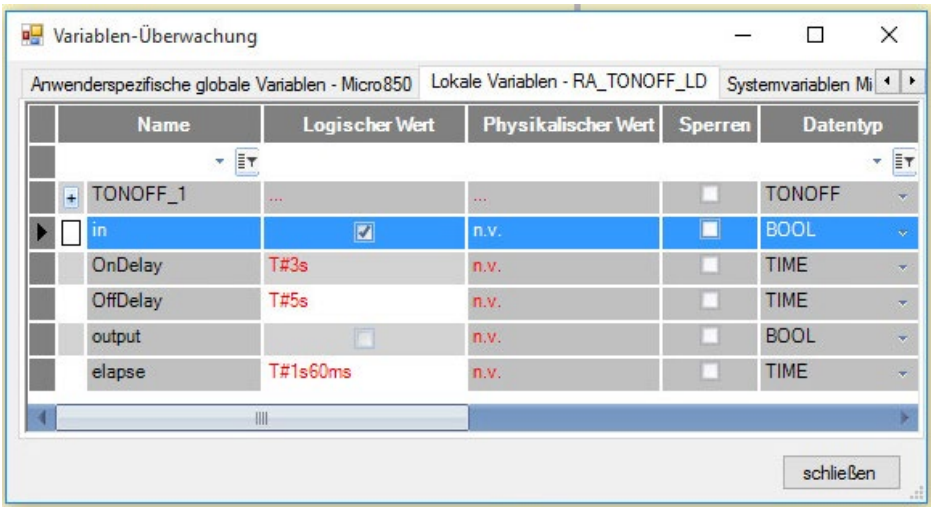
Beispiel für einen strukturierten TONOFF-Text

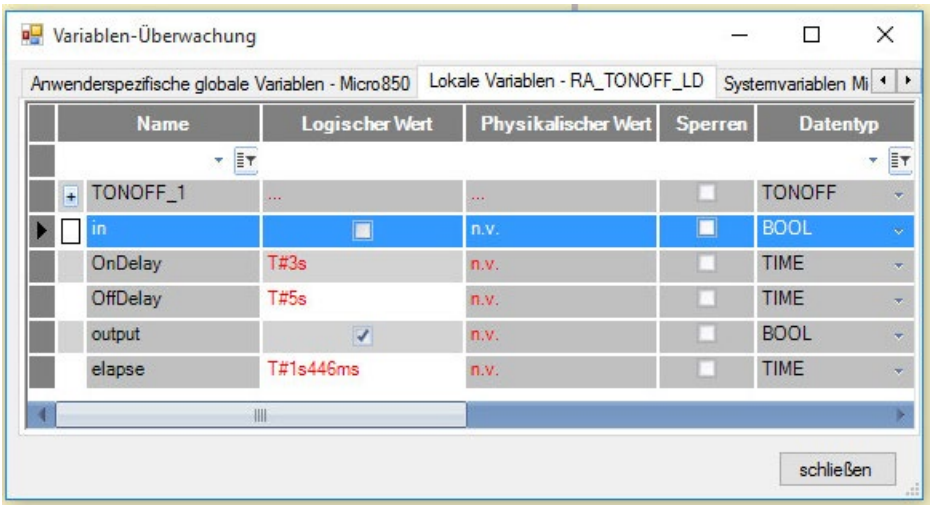
```
TONOFF_1(
```

void **TONOFF_1**(BOOL **IN**, TIME **PT**, TIME **PTOF**)
Typ: TONOFF, Verzögern von einem Ausgang-ein (wahr), dann Verzögern von einem Ausgang-aus (falsch).

```
1  OnDelay := T#3s;
2  OffDelay := T#5s;
3  TONOFF_1(in, OnDelay, OffDelay);
4  output := TONOFF_1.Q;
5  elapse := TONOFF_1.ET;
```

Ergebnisse





TP (zeitliche Impulssteuerung)

Setzt einen internen Timer bei einer ansteigenden Flanke um einen gegebenen Wert nach oben. Wenn der Timer abgelaufen ist, wird die interne Zeit zurückgesetzt.

Details zum Betrieb:

- Falls der Parameter „EN“ mit TP verwendet wird, beginnt der Timer mit dem Inkrementieren, sobald „EN“ auf TRUE gesetzt wird, und setzt das Inkrementieren auch dann fort, wenn „EN“ auf FALSE gesetzt wird.

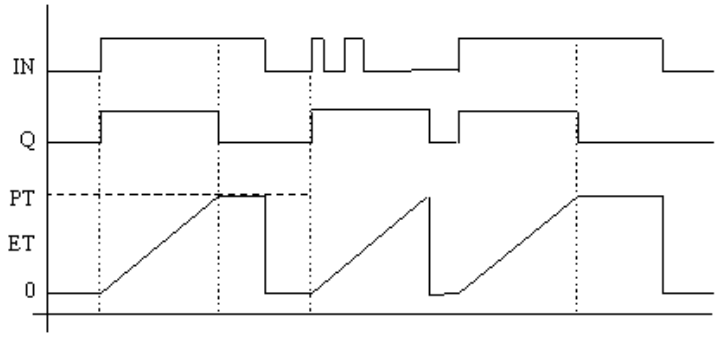
Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

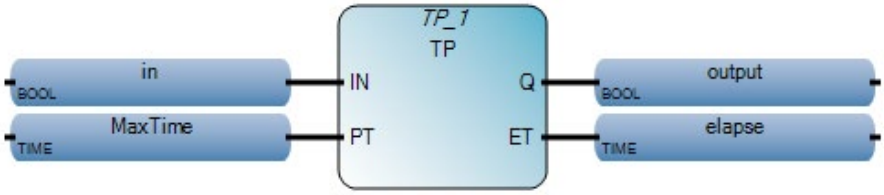
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	TRUE – Bei Anstiegsflanken beginnt das Hochzählen des internen Timers (falls dieser nicht bereits hochgesetzt wird). FALSE – wenn der Timer abgelaufen ist, wird der interne Timer zurückgesetzt. Änderungen an IN während des Zählens haben keinerlei Auswirkungen.
PT	Eingang	TIME	Maximale programmierte Dauer, die mit dem Datentyp „Time" definiert wurde.
Q	Ausgang	BOOL	TRUE – Der Timer zählt. FALSE – Der Timer zählt nicht.
ET	Ausgang	TIME	Derzeit abgelaufene Zeit. Mögliche Werte reichen von 0ms bis 1193h2m47s294ms.

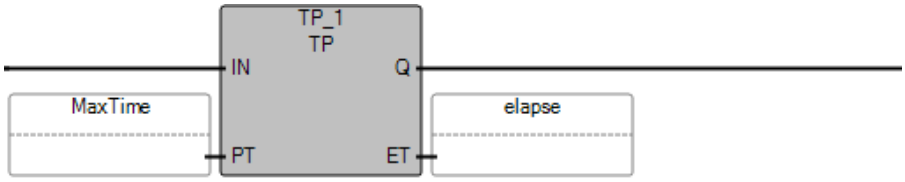
TP-Timing-Diagramm



Beispiel für ein TP-Funktionsblockdiagramm



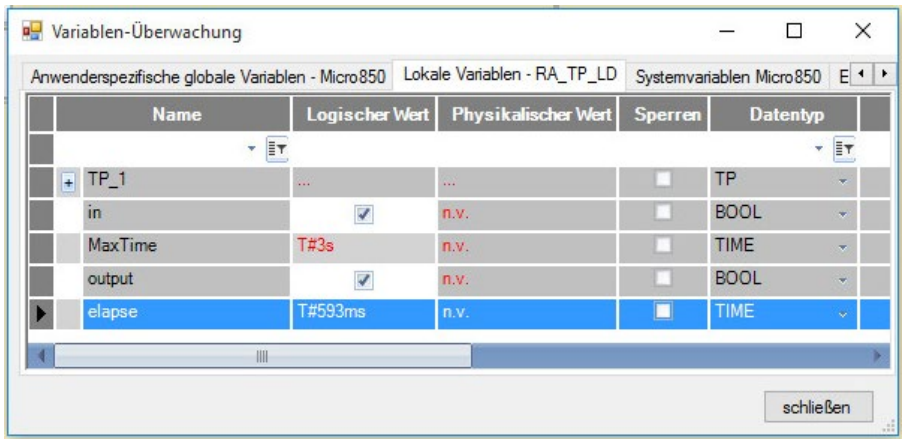
Beispiel für einen TP-Kontaktplan



Beispiel für strukturierten TP-Text

```
1  MaxTime := T#3s;
2  TP_1(in, MaxTime);
3  output := TP_1.Q;
4  elapse := TP_1.ET;
```

Ergebnisse



RTO (Remanenter Timer, Einschaltverzögerung)

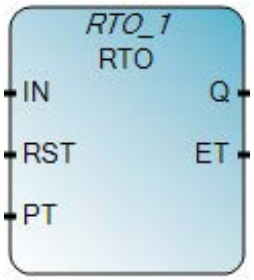
Erhöht einen internen Timer, wenn der Eingang aktiv ist, setzt aber den internen Timer nicht zurück, wenn der Eingang inaktiv wird.

Details zum Betrieb:

- Bei Verwendung eines <Micro810>- oder <Micro820>-Controllers wird der interne RTO-Timer standardmäßig nicht während eines Einschaltzyklus beibehalten. Um den internen Timer beizubehalten, setzen Sie die Konfigurationsparameter „Beibehalten“ auf TRUE.
- Bei Verwendung eines <Micro830>- oder <Micro850>-Controllers wird der interne RTO-Timer während eines Einschaltzyklus beibehalten.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

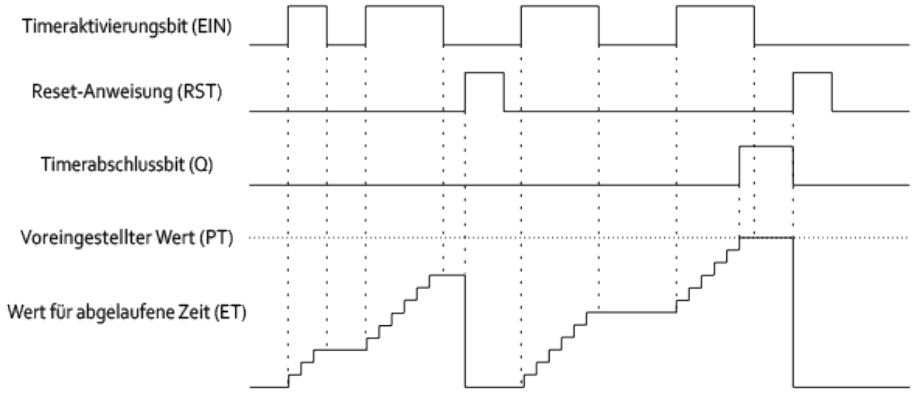
Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



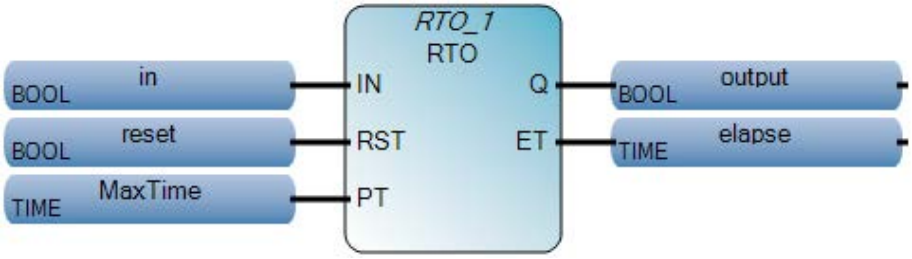
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
IN	Eingang	BOOL	Eingangssteuerung. TRUE – Bei Anstiegflanken wird der interne Timer erhöht. FALSE – Bei Abstiegsflanken wird der interne Timer angehalten und nicht zurückgesetzt.
RST	Eingang	BOOL	TRUE – Bei Anstiegsflanken wird der interne Timer zurückgesetzt. FALSE – Der interne Timer wird nicht zurückgesetzt.
PT	Eingang	TIME	Maximale programmierte On-Delay-Zeit. PT wird definiert mit dem Datentyp „Time“ definiert.
Q	Ausgang	BOOL	TRUE – Die programmierte On-Delay-Zeit ist abgelaufen. FALSE – Die programmierte On-Delay-Zeit ist nicht abgelaufen.
ET	Ausgang	TIME	Derzeit abgelaufene Zeit. Die Werte reichen von 0ms bis 1193h2m47s294ms. ET wird mit den Datentyp –Tiem“ definiert.

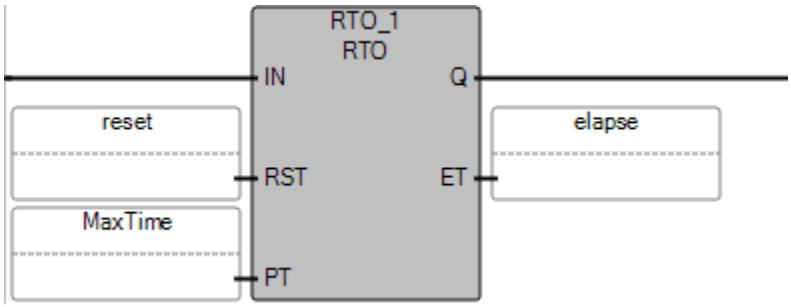
Beispiel eines RTO-Zeitgebungsdiagramms



Beispiel für ein RTO-Funktionsblockdiagramm



Beispiels für einen RTO-Kontaktplan



Beispiel für einen strukturierten RTO-Text

```
RTO_1(
```

void **RTO_1**(BOOL **IN**, BOOL **RST**, TIME **PT**)
Typ: RTO, Verzögern von einem Ausgang-ein (wahr). Abgelaufene Zeit bis Zurücksetzung beibehalten.

```
1 MaxTime := T#3s;
2 RTO_1(in, reset, MaxTime);
3 output := RTO_1.Q;
4 elapse := RTO_1.ET;
```

DOY (Jahreseinstellung für Echtzeituhr)

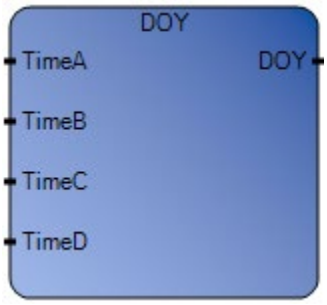
Aktiviert eine Ausgabe, wenn der Wert der Echtzeituhr (RTC) sich im Bereich der Jahr/Zeit-Einstellung befindet.

Details zum Betrieb:

- Wenn keine Echtzeituhr vorhanden ist, ist die Ausgabe immer deaktiviert.
- Konfigurieren Sie die „Zeit“-Eingangsparameter mit gültigen Bereichen; diese entnehmen Sie den Hinweisen zum Datentyp „DOYDATA“. Ein ungültiger Wert führt zu einem Controller-Fehler, wenn TimeX.Enable auf TRUE gesetzt wird und eine Echtzeituhr vorhanden und aktiviert ist.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.



Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

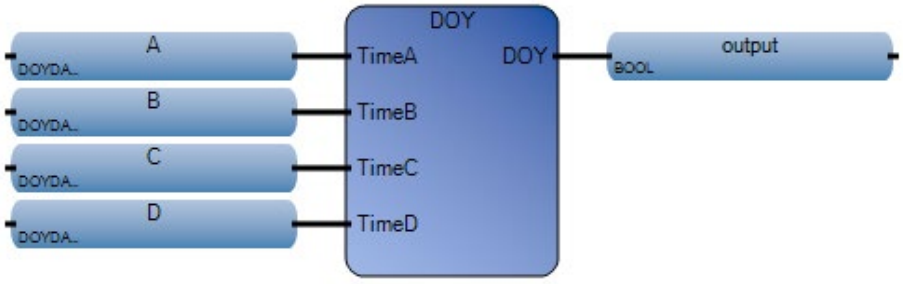
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - die Operation wird durchgeführt. FALSE - die Operation wird nicht durchgeführt.
TimeA	Eingang	DOYDATA	Jahreszeiteinstellung von Kanal A. Verwenden Sie den Datentyp DOYDATA, um TimeA zu konfigurieren.
TimeB	Eingang	DOYDATA	Jahreszeiteinstellung von Kanal B. Verwenden Sie den Datentyp DOYDATA, um TimeB zu konfigurieren.
TimeC	Eingang	DOYDATA	Jahreszeiteinstellung von Kanal C. Verwenden Sie den Datentyp DOYDATA, um TimeC zu konfigurieren.
TimeD	Eingang	DOYDATA	Jahreszeiteinstellung von Kanal D. Verwenden Sie den Datentyp DOYDATA, um TimeD zu konfigurieren.
DOY	Ausgang	BOOL	Wenn TRUE, befindet sich der Wert der Echtzeituhr im Bereich der Jahreszeiteinstellung eines der vier Kanäle.

Datentyp DOYDATA

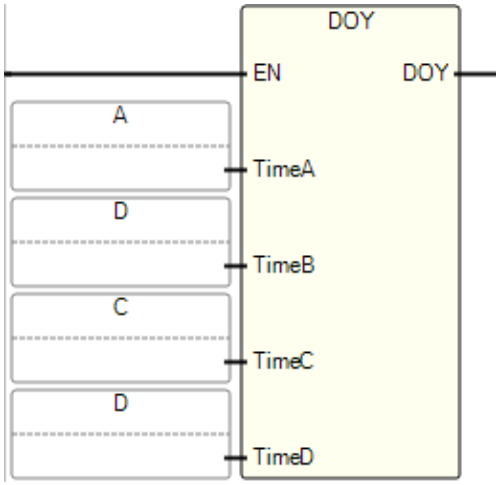
In der folgenden Tabelle wird der Datentyp DOYDATA beschrieben.

Parameter	Datentyp	Beschreibung
Enable	BOOL	TRUE: Aktivieren; FALSE: Deaktivieren
YearlyCenturial	BOOL	Timertyp (0:Jahresschaltuhr; 1:Jahrhundertschaltuhr).
YearOn	UINT	Jahreswert, der zur Aktivierung führt (muss im Bereich [2000...2098] liegen).
MonthOn	USINT	Monatswert, der zur Aktivierung führt (muss im Bereich [1...12] liegen).
DayOn	USINT	Tageswert, der zur Aktivierung führt (muss im Bereich [1...31] liegen, abhängig vom Wert "MonthOn").
YearOff	UINT	Jahreswert, der zur Deaktivierung führt (muss im Bereich [2000...2098] liegen).
MonthOff	USINT	Montagswert, der zur Deaktivierung führt (muss im Bereich [1...12] liegen).
DayOff	USINT	Tageswert, der zur Deaktivierung führt (muss im Bereich [1...31] liegen, abhängig vom Wert "MonthOff").

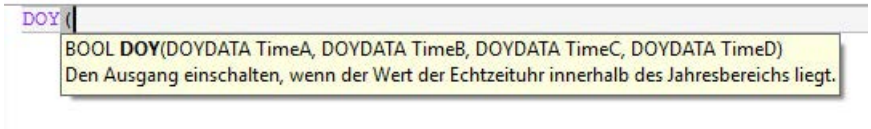
Diagrammbeispiel für Funktionsblock DOY



DOY-Kontaktplan-Beispiel



Beispiel für DOY in strukturiertem Text




```
1  output := DOY(A, B, C, D);
```

(* ST-Äquivalenz: *)

TESTOUTPUT := DOY(TIMEA1, TIMEB1, TIMEC1, TIMED1) ;

TDF (Zeitdifferenz)

Berechnet die Zeitdifferenz zwischen TimeA und TimeB.

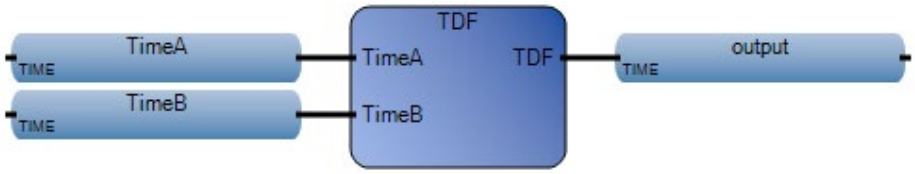
Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

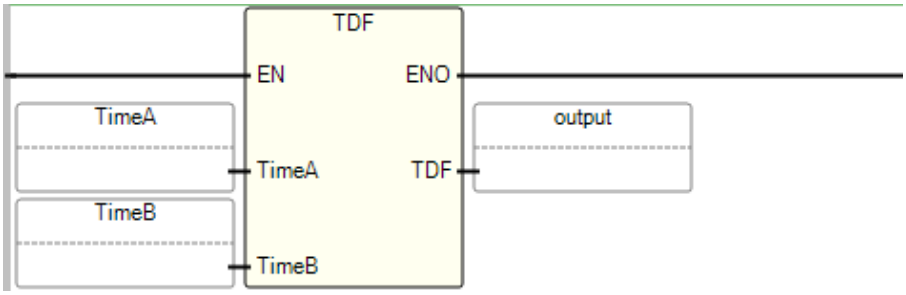
Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. TRUE - Ausführen der aktuellen Berechnung. FALSE – es findet keine Berechnung statt. Gilt für Kontaktplan-Programme.
TimeA	Eingang	TIME	Die Startzeit für die Berechnung der Zeitdifferenz.
TimeB	Eingang	TIME	Die Endzeit für die Berechnung der Zeitdifferenz.
ENO	Ausgang	BOOL	Aktivieren der Ausgabe. Gilt für Kontaktplan-Programme.
TDF	Ausgang	TIME	Die Zeitdifferenz zwischen den zwei eingegangenen Zeitwerten. TDF ist Name oder PIN ID.

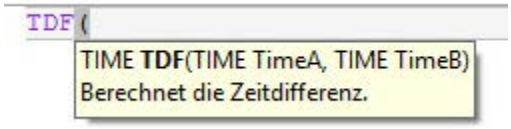
Beispiel für ein TDF-Funktionsblockdiagramm



Beispiel für einen TDF-Kontaktplan



Beispiel für strukturierten TDF-Text

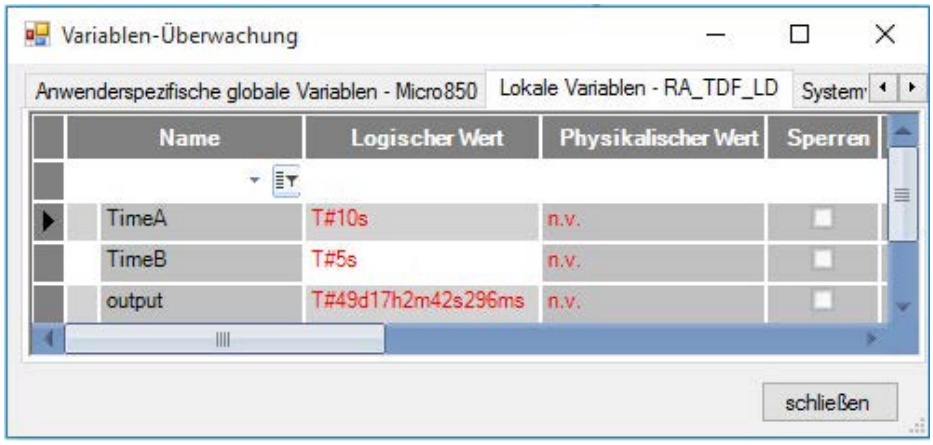



```
1  TimeA := T#10s;
2  TimeB := T#5s;
3  output := TDF(TimeA, TimeB);
```

(* ST-Äquivalenz: *)

TESTTIMEDIFF := TDF(TESTTIME1, TESTTIME2) ;

Ergebnisse



TOW (Woche für Echtzeituhr überprüfen)

Aktiviert eine Ausgabe, wenn sich der Wert der Echtzeituhr (RTC) im Bereich der Wochenzeiteinstellung befindet.

Details zum Betrieb:

- Wenn keine Echtzeituhr vorhanden ist, ist die Ausgabe immer deaktiviert.
- Konfigurieren Sie alle „Zeit“-Eingangsparameter mit gültigen Bereichen; diese entnehmen Sie der Tabelle zum Datentyp „TOWDATA“. Ein ungültiger Wert führt zu einem Controller-Fehler, wenn TimeX.Enable auf TRUE gesetzt wird und eine Echtzeituhr vorhanden und aktiviert ist.

Unterstützte Sprachen: Funktionsblockdiagramm, Leiterdiagramm und Strukturierter Text.

Diese Anweisung gilt für die Controller Micro810, Micro820, Micro830, Micro850, Micro870 und den Micro800 Simulator.

Verwenden Sie diese Tabelle, um die Parameterwerte für diese Anweisung zu ermitteln.

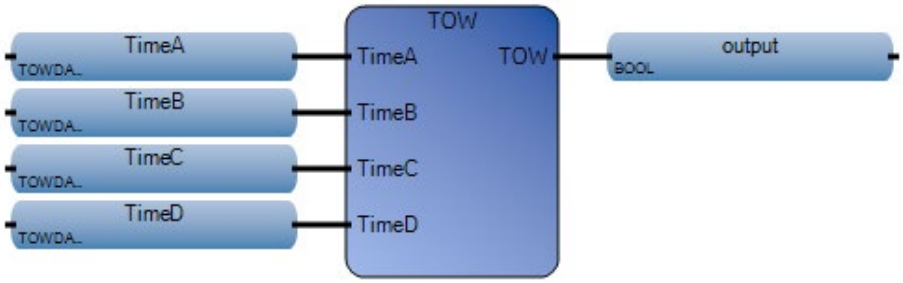
Parameter	Parametertyp	Datentyp	Beschreibung
EN	Eingang	BOOL	Anweisung aktivieren. Wenn EN = TRUE, Vorgang ausführen. Wenn EN = FALSE, Vorgang nicht ausführen.
TimeA	Eingang	TOWDATA	Tageszeiteinstellung von Kanal A. Verwenden Sie den Datentyp „TOWDATA“, um TimeA zu definieren.
TimeB	Eingang	TOWDATA	Tageszeiteinstellung von Kanal B. Verwenden Sie den Datentyp „TOWDATA“, um TimeB zu definieren.
TimeC	Eingang	TOWDATA	Tageszeiteinstellung von Kanal C. Verwenden Sie den Datentyp „TOWDATA“, um TimeC zu definieren.
TimeD	Eingang	TOWDATA	Tageszeiteinstellung von Kanal D. Verwenden Sie den Datentyp „TOWDATA“, um TimeD zu definieren.
TOW	Ausgang	BOOL	Wenn TRUE, befindet sich der Wert der Echtzeituhr im Bereich der Tageszeiteinstellung eines der vier Kanäle.

Datentyp TOWDATA

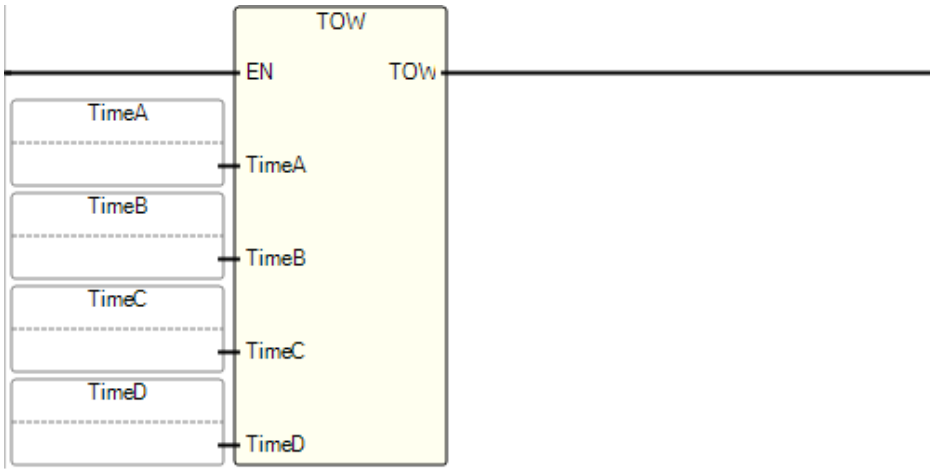
In der folgenden Tabelle wird der Datentyp TOWDATA beschrieben:

Parameter	Datentyp	Beschreibung
Enable	BOOL	TRUE: Aktivieren; FALSE: Deaktivieren.
DailyWeekly	BOOL	Timertyp (0:Tagesschaltuhr; 1:Wochenschaltuhr).
DayOn	USINT	Wochentagswert, der zur Aktivierung führt (muss im Bereich [0...6] liegen).
HourOn	USINT	Stundenwert, der zur Aktivierung führt (muss im Bereich [0...23] liegen).
MinOn	USINT	Minutenwert, der zur Aktivierung führt (muss im Bereich [0...59] liegen).
DayOff	USINT	Wochentagswert, der zur Deaktivierung führt (muss im Bereich [0...6] liegen).
HourOff	USINT	Stundenwert, der zur Deaktivierung führt (muss im Bereich [0...23] liegen).
MinOff	USINT	Minutenwert, der zur Deaktivierung führt (muss im Bereich [0...59] liegen).

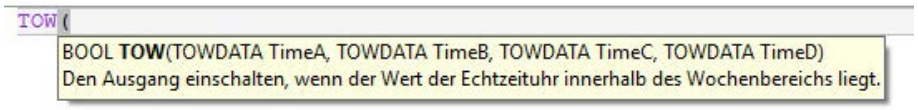
Beispiel für ein TOW-Funktionsblockdiagramm



Beispiel für einen TOW-Kontaktplan



Beispiel für strukturierten TOW-Text




```
1  output := TOW(TimeA, TimeB, TimeC, TimeD);
```

(* ST-Äquivalenz: *)

TESTOUTPUT := TOW(TIMEA, TIMEB, TIMEC, TIMED) ;

Ergebnisse

Name	Logischer Wert	Physikalischer Wert	Sperren	Datentyp
output	☑	n.v.	☐	BOOL
TimeA	...	...	☐	TOWDATA
TimeA.Enable	☑	n.v.	☐	BOOL
TimeA.DailyWeekly	☑	n.v.	☐	BOOL
TimeA.DayOn	1	n.v.	☐	USINT
TimeA.HourOn	10	n.v.	☐	USINT
TimeA.MinOn	20	n.v.	☐	USINT
TimeA.DayOff	2	n.v.	☐	USINT
TimeA.HourOff	15	n.v.	☐	USINT
TimeA.MinOff	30	n.v.	☐	USINT
TimeB	...	...	☐	TOWDATA
TimeB.Enable	☑	n.v.	☐	BOOL
TimeB.DailyWeekly	☑	n.v.	☐	BOOL
TimeB.DayOn	2	n.v.	☐	USINT
TimeB.HourOn	15	n.v.	☐	USINT
TimeB.MinOn	20	n.v.	☐	USINT
TimeB.DayOff	3	n.v.	☐	USINT
TimeB.HourOff	10	n.v.	☐	USINT
TimeB.MinOff	30	n.v.	☐	USINT
TimeC	...	...	☐	TOWDATA
TimeC.Enable	☑	n.v.	☐	BOOL
TimeC.DailyWeekly	☑	n.v.	☐	BOOL
TimeC.DayOn	3	n.v.	☐	USINT
TimeC.HourOn	20	n.v.	☐	USINT
TimeC.MinOn	10	n.v.	☐	USINT
TimeC.DayOff	4	n.v.	☐	USINT
TimeC.HourOff	25	n.v.	☐	USINT
TimeC.MinOff	55	n.v.	☐	USINT
TimeD	...	...	☐	TOWDATA
TimeD.Enable	☑	n.v.	☐	BOOL

Index

- Operator 97

*****

*** Operator 86**

/

/ Operator 79

+

+ Operator 69

<

< Operator 246
<= Operator 248
<> Operator 249

=

= Operator 241

>

> Operator 244
>= Operator 245

1

1Gain-Operator 85

A

ABL, Funktionsblock 103
ABS-Funktion 65
ACB, Funktionsblock 109
ACL, Funktionsblock 105
ACOS_LREAL-Funktion 68
ACOS-Funktion 67
Additionsoperator 69
AFI 521
AHL, Funktionsblock 107
Alarmmeldungen
LIM_ALRM, Funktionsblock 61
AND_MASK-Funktion 125
AND-Operator 150
Anweisungen 29
Anweisungsblockbeschreibung 166, 173, 183, 277, 279
Anweisungsblockbeschreibung für
ABL 103
ACB 109
ACL 105
AHL 107
ARD 111
ARL 114
AVERAGE 277
AWA 116
AWT 118
COP 279
CTD 251
CTU 253
CTUD 255
DERIVATE 475
F_TRIG 145
HSC 290
HSC_SET_STS 306
HYSTER 490
IIM 353
INTEGRAL 492
IOM 355
IPIDCONTROLLER 525
KEY_READ 357
LIM_ALRM 61
MC_AbortTrigger 416
MC_Halt 418
MC_Home 421
MC_MoveAbsolute 424
MC_MoveRelative 428
MC_MoveVelocity 431
MC_Power 435
MC_ReadAxisError 445
MC_ReadBoolParameter 449
MC_ReadParameter 451
MC_ReadStatus 453
MC_Reset 457
MC_SetPosition 460
MC_Stop 462
MC_TouchProbe 465
MC_WriteBoolParameter 469
MC_WriteParameter 472
MM_INFO 361
MSG_CIPGENERIC 166
MSG_CIPSYMBOLIC 173
MSG_MODBUS 177
MSG_MODBUS2 183
PLUGIN_INFO 375
PLUGIN_READ 378
PLUGIN_RESET 379
PLUGIN_WRITE 381
R_TRIG 147
RST 148
RTC_READ 385
RTC_SET 387

RTO 644
SCALER 505
SRT 152
STACKINT 507
SUS 521
SYS_INFO 389
TOF 635
TON 637
TONOFF 639
TP 642
TRIMPOT_READ 392

ANY_TO_BOOL-Operator 257
ANY_TO_BYTE-Operator 258
ANY_TO_DATE-Operator 259
ANY_TO_DINT-Operator 261
ANY_TO_DWORD-Operator 262
ANY_TO_INT-Operator 263
ANY_TO_LINT-Operator 264
ANY_TO_LREAL-Operator 265
ANY_TO_LWORD-Operator 266
ANY_TO_REAL-Operator 267
ANY_TO_SINT-Operator 268
ANY_TO_STRING-Operator 269
ANY_TO_TIME-Operator 270
ANY_TO_UDINT-Operator 271
ANY_TO_UINT-Operator 272
ANY_TO_ULINT-Operator 273
ANY_TO_USINT-Operator 274
ANY_TO_WORD-Operator 275
ARD, Funktionsblock 111
Arithmetisch
ABS-Funktion 65
ACOS_LREAL-Funktion 68
ACOS-Funktion 67
Additionsoperator 69
ASIN_LREAL-Funktion 72
ASIN-Funktion 71
ATAN_LREAL-Funktion 75
ATAN-Funktion 74
COS_LREAL-Funktion 78
COS-Funktion 76
Divisionsoperator 79
EXPT-Funktion 81
LOG-Funktion 82
MOD-Funktion 84
MOV-Operator 85
Multiplikationsoperator 86
Neg-Operator 88
POW-Funktion 89
RAND-Funktion 90
SIN_LREAL 94
SIN-Funktion 92
SQRT-Funktion 95
Subtraktionsoperator 97
TAN_LREAL-Funktion 100
TAN-Funktion 98
TRUNC-Funktion 101
Arithmetische Operationen
1Gain-Operator 85
ARL, Funktionsblock 114
ASCII-Funktion 615
ASIN_LREAL-Funktion 72
ASIN-Funktion 71
ATAN_LREAL-Funktion 75
ATAN-Funktion 74
Aufrufen
Funktionsblöcke 21
Ausgang zurücksetzen in FBD 47
AVERAGE, Funktionsblock 277
AWA, Funktionsblock 116
AWT, Funktionsblock 118

B

Benennungskonventionen
Funktionsblöcke 21
Bewegung
MC_AbortTrigger, Funktionsblock 416
MC_Halt, Funktionsblock 418
MC_Home, Funktionsblock 421
MC_MoveAbsolute, Funktionsblock 424
MC_MoveRelative, Funktionsblock 428
MC_MoveVelocity, Funktionsblock 431
MC_Power, Funktionsblock 435
MC_ReadAxisError, Funktionsblock 445
MC_ReadBoolParameter, Funktionsblock 449
MC_ReadParameter, Funktionsblock 451
MC_ReadStatus, Funktionsblock 453
MC_Reset, Funktionsblock 457
MC_SetPosition, Funktionsblock 460
MC_Stop, Funktionsblock 462
MC_TouchProbe, Funktionsblock 465
MC_WriteBoolParameter, Funktionsblock 469
MC_WriteParameter, Funktionsblock 472
Bezeichnungen
Strompfade, Einfügen 33
Binäre Operationen
AND_MASK-Funktion 125
NOT_MASK-Funktion 133
OR_MASK-Funktion 134
ROL-Funktion 135
ROR-Funktion 137
SHL-Funktion 139
SHR-Funktion 141
XOR_MASK-Funktion 142
Boolesch
MUX4B-Funktion 159
MUX8B-Funktion 156
TTABLE-Funktion 154
Boolesche Vorgänge
AND-Operator 150

F_TRIG, Funktionsblock 145
NOT-Operator 152
OR-Operator 150
R_TRIG, Funktionsblock 147
RS, Funktionsblock 148
SR, Funktionsblock 152
XOR-Operator 151

C

CHAR-Funktion 617
COM_IO_WDOG 163
COP, Funktionsblock 279
COS_LREAL-Funktion 78
COS-Funktion 76
CTD, Funktionsblock 251
CTU, Funktionsblock 253
CTUD, Funktionsblock 255

D

Datenbearbeitung
AVERAGE, Funktionsblock 277
COP, Funktionsblock 279
MAX-Funktion 285
MIN-Funktion 284
Datenkonvertierung
ANY_TO_BOOL-Operator 257
ANY_TO_BYTE-Operator 258
ANY_TO_DATE-Operator 259
ANY_TO_DINT-Operator 261
ANY_TO_DWORD-Operator 262
ANY_TO_INT-Operator 263
ANY_TO_LINT-Operator 264
ANY_TO_LREAL-Operator 265
ANY_TO_LWORD-Operator 266
ANY_TO_REAL-Operator 267
ANY_TO_SINT-Operator 268
ANY_TO_STRING-Operator 269
ANY_TO_TIME-Operator 270
ANY_TO_UDINT-Operator 271
ANY_TO_UINT-Operator 272
ANY_TO_ULINT-Operator 273
ANY_TO_USINT-Operator 274
ANY_TO_WORD-Operator 275
DERIVATE, Funktionsblock 475
Direkt
Kontakte 51
Kontakte in FBD 51
Spulen 43
Spulen in FBD 43
Divisionsoperator 79
DOY-Funktion 646

E

Einfügen
Bezeichnungen von Strompfaden 33
Rücksprünge (FBD-Elemente) 53
Sprünge zu Bezeichnungen (FBD-Elemente) 54
Strompfade (FBD-Elemente) 33
Eingabe/Ausgabe
HSC, Funktionsblock 290
HSC_SET_STS, Funktionsblock 306
IIM, Funktionsblock 353
IOM, Funktionsblock 355
KEY_READ, Funktionsblock 357
LCD-Funktion 341
MMINFO, Funktionsblock 361
PLUGIN_INFO, Funktionsblock 375
PLUGIN_READ, Funktionsblock 378
PLUGIN_RESET, Funktionsblock 379
PLUGIN_WRITE, Funktionsblock 381
RHC-Funktion 348
RPC-Funktion 350
RTC_READ, Funktionsblock 385
RTC_SET, Funktionsblock 387
SYS_INFO, Funktionsblock 389
TRIMPOT_READ, Funktionsblock 392
Erweiterte Steuerungsblöcke
SCALER 505
EXPT-Funktion 81

F

F_TRIG, Funktionsblock 145
FBD (Funktionsblockdiagramm)
Ausgang setzen 46
Direkte Spulen 43
Impulsanstiegflanken-Kontakte (positiv) 52
Invertierter Ausgang 44
Kontakte, Verwendung und verfügbare Typen 48
RESET-Spulen 47
Rücksprünge, Einfügen 53
Schliesser-Kontakt 51
Sprünge zu Bezeichnungen, Einfügen 54
Spulen, Verwendung und verfügbare Typen 40
Strompfade, Einfügen 33
Umkehrkontakte 51
FFL 477
FFU 486
FIND-Funktion 620
Funktionen
ABS 65
ACOS 67
ACOS_LREAL 68
AND_MASK 125
ASCII 615
ASIN 71
ASIN_LREAL 72

ATAN 74
ATAN_LREAL 75
CHAR 617
COS 76
COS_LREAL 78
DELETE 618
DOY 646
EXPT 81
FIND 620
INSERT 622
LCD 341
LEFT 624
LIMIT 517
LOG 82
MAX 285
MID 626
MIN 284
MLEN 628
MOD 84
MUX4B 159
MUX8B 156
NOT_MASK 133
OR_MASK 134
POW 89
RAND 90
REPLACE 631
RHC 348
RIGHT 629
ROL 135
ROR 137
RPC 350
SHL 139
SHR 141
SIN 92
SIN_LREAL 94
SQRT 95
STIS 395
TAN 98
TAN_LREAL 100
TDF 648
TND 516
TOW 650
TRUNC 101
TTABLE 154
UIC 397
UID 398
UIE 400
UIF 402
XOR_MASK 142

Funktionsblöcke

ABL 103
ACB 109
ACL 105
AHL 107
ARD 111
ARL 114
Aufrufen 21
AVERAGE 277
AWA 116
AWT 118
Benennung und Parameter für 21
COP 279
CTD 251
CTU 253
CTUD 255
DERIVATE 475
F_TRIG 145
FFL 477
FFU 486
HSC 290
HSC_SET_STS 306
HYSTER 490
IIM 353
INTEGRAL 492
IOM 355
IPIDCONTROLLER 525
KEY_READ 357
LIM_ALRM 61
MC_AbortTrigger 416
MC_Halt 418
MC_Home 421
MC_MoveAbsolute 424
MC_MoveRelative 428
MC_MoveVelocity 431
MC_Power 435
MC_ReadAxisError 445
MC_ReadBoolParameter 449
MC_ReadParameter 451
MC_ReadStatus 453
MC_Reset 457
MC_SetPosition 460
MC_Stop 462
MC_TouchProbe 465
MC_WriteBoolParameter 469
MC_WriteParameter 472
MMINFO 361
MODULE_INFO 364
MSG_CIPGENERIC 166
MSG_CIPSYMBOLIC 173
MSG_MODBUS 177
MSG_MODBUS2 183
PID 550
PLUGIN_INFO 375
PLUGIN_READ 378
PLUGIN_RESET 379
PLUGIN_WRITE 381
R_TRIG 147
RS 148
RTC_READ 385
RTC_SET 387
RTO 644
SCALER 505
SOCKET_ACCEPT 566
SOCKET_CREATE 570
SOCKET_DELETE 575
SOCKET_DELETEALL 577

SOCKET_INFO 579
SOCKET_OPEN 584
SOCKET_READ 588
SOCKET_WRITE 592
SR 152
STACKINT 507
SUS 521
SYS_INFO 389
TOF 635
TON 637
TONOFF 639
TP 642
TRIMPOT_READ 392

H

HSC, Funktionsblock 290
HSC_SET_STS, Funktionsblock 306
HYSTER, Funktionsblock 490

I

IIM, Funktionsblock 353
Impulsabstiegflanke (negativ)
Kontakte 52
Spulen 45
Impulsanstiegflanke (positiv)
Kontakte 52
Kontakte in FBD 52
Spulen 45
INSERT-Funktion 622
INTEGRAL, Funktionsblock 492
IOM, Funktionsblock 355
IPIDCONTROLLER, Funktionsblock 525

K

KEY_READ, Funktionsblock 357
Kommunikation
ABL, Funktionsblock 103
ACB, Funktionsblöcke 109
ACL, Funktionsblock 105
AHL, Funktionsblock 107
ARD, Funktionsblock 111
ARL, Funktionsblock 114
AWA, Funktionsblock 116
AWT, Funktionsblock 118
MSG_CIPGENERIC, Funktionsblock 166
MSG_CIPSYMBOLIC, Funktionsblock 173
MSG_MODBUS, Funktionsblock 177
MSG_MODBUS2, Funktionsblock 183
Kontakte
Ändern des Typs 48
Direkt 51
Direkt in FBD 51
Einfügen 48
Impulsabstiegflanke (positiv) 52
Impulsanstiegflanke (positiv) 52
Impulsanstiegflanke (positiv) in FBD 52
Parallel, Einfügen 48
Umkehrung 51
Umkehrung in FBD 51
Verwendung und verfügbare Typen 48
Verwendung und verfügbare Typen in FBD 48
Kontaktplan (KOP)
Anweisungen 29
Sprache 29
KOP (Kontaktplan)
Ausgang setzen 46
Direkte Spulen 43
Impulsabstiegflanken-Kontakte (negativ) 52
Impulsanstiegflanken-Kontakte (positiv) 52
Impulsspulen mit abfallender Flanke (negativ) 45
Impulsspulen mit ansteigender Flanke (positiv) 45
Invertierter Ausgang 44
Kontakte, Verwendung und verfügbare Typen 48
RESET-Spulen 47
Return-Anweisungen 53
Schliesser-Kontakt 51
Sprünge 54
Spulen, Verwendung und verfügbare Typen 40
Strompfade, Einfügen 33
Umkehrkontakte 51
Verzweigungen 37
KOP-Sprache
Tastenkombinationen 58

L

LCD-Funktion 341
LEFT-Funktion 624
LIM_ALRM, Funktionsblock 61
LIMIT-Funktion 517
LOG-Funktion 82

M

MAX-Funktion 285
MC_AbortTrigger, Funktionsblock 416
MC_Halt, Funktionsblock 418
MC_Home, Funktionsblock 421
MC_MoveAbsolute, Funktionsblock 424
MC_MoveRelative, Funktionsblock 428
MC_MoveVelocity, Funktionsblock 431
MC_Power, Funktionsblock 435
MC_Read 442
MC_ReadActualPosition 439
MC_ReadAxisError, Funktionsblock 445

MC_ReadBoolParameter, Funktionsblock 449
MC_ReadParameter, Funktionsblock 451
MC_ReadStatus, Funktionsblock 453
MC_Reset, Funktionsblock 457
MC_SetPosition, Funktionsblock 460
MC_Stop, Funktionsblock 462
MC_TouchProbe, Funktionsblock 465
MC_WriteBoolParameter, Funktionsblock 469
MC_WriteParameter, Funktionsblock 472
MID-Funktion 626
MIN-Funktion 284
MLEN-Funktion 628
MMINFO, Funktionsblock 361
MOD-Funktion 84
MODULE_INFO 364
Motion Control-Funktionsblöcke 405
MOV-Operator 85
MSG_CIPGENERIC, Funktionsblock 166
MSG_CIPSYMBOLIC, Funktionsblock 173
MSG_MODBUS, Funktionsblock 177
MSG_MODBUS2, Funktionsblock 183
Multiplikationsoperator 86
MUX4B-Funktion 159
MUX8B-Funktion 156

N

Neg-Operator 88
NOT_MASK-Funktion 133
NOT-Operator 152

O

Operatoren
1Gain 85
Addition 69
AND 150
ANY_TO_BOOL 257
ANY_TO_BYTE 258
ANY_TO_DATE 259
ANY_TO_DINT 261
ANY_TO_DWORD 262
ANY_TO_INT 263
ANY_TO_LINT 264
ANY_TO_LREAL 265
ANY_TO_LWORD 266
ANY_TO_REAL 267
ANY_TO_SINT 268
ANY_TO_STRING 269
ANY_TO_TIME 270
ANY_TO_UDINT 271
ANY_TO_UINT 272
ANY_TO_ULINT 273
ANY_TO_USINT 274
ANY_TO_WORD 275
Division 79
Gleich 241
Größer als 244
Größer als oder gleich 245
Kleiner als 246
Kleiner als oder gleich 248
MOV 85
Multiplikation 86
Neg 88
Nicht gleich 249
NOT 152
OR 150
Subtraktion 97
XOR 151
OR_MASK-Funktion 134
OR-Operator 150

P

Parallelverzweigungen für Kontaktpläne 37
Parameter
Für Funktionsblöcke 21
PID 550
PLUGIN_INFO, Funktionsblock 375
PLUGIN_READ, Funktionsblock 378
PLUGIN_RESET, Funktionsblock 379
PLUGIN_WRITE, Funktionsblock 381
POW-Funktion 89
Programmsteuerung
SUS, Funktionsblock 521
TND-Funktion 516
Prozesssteuerung
DERIVATE, Funktionsblock 475
HYSTER, Funktionsblock 490
INTEGRAL, Funktionsblock 492
IPIDCONTROLLER, Funktionsblock 525
LFL(LIFO-Last) 498
LFU(LIFO-Entlastung) 500
LIMIT-Funktion 517
STACKINT, Funktionsblock 507

R

R_TRIG, Funktionsblock 147
RAND-Funktion 90
REPLACE-Funktion 631
Return-Anweisungen
Einfügen 53
Für Kontaktpläne 53
RHC-Funktion 348
RIGHT-Funktion 629
ROL-Funktion 135
ROR-Funktion 137
RPC-Funktion 350
RS, Funktionsblock 148
RTC_READ, Funktionsblock 385
RTC_SET, Funktionsblock 387
RTO-Funktionsblock 644

Rücksprungsymbole
Einfügen , FBD-Elemente 53

S

SCALLER, Funktionsblock 505
Schablone
Kontaktelemente, verfügbar für Kontaktplandiagramme 48
Spulentypen, verfügbar für FBD 40
Strompfade, Einfügen in KOP-Container 33
Verfügbare Spulentypen für Kontaktplandiagramme 40
SHL-Funktion 139
SHR-Funktion 141
SIN_LREAL 94
SIN-Funktion 92
SOCKET_ACCEPT 566
SOCKET_CREATE 570
SOCKET_DELETE 575
SOCKET_DELETEALL 577
SOCKET_INFO 579
SOCKET_OPEN 584
SOCKET_READ 588
SOCKET_WRITE 592
Sprache
Kontaktplan (KOP) 29
Sprünge
Funktionsblockdiagramm, Einfügen 54
Für Kontaktpläne 54
Kontaktpläne, Einfügen 54
Strompfade,Einfügen 33
Spulen
Ändern des Typs 40
Ändern des Typs in FBD 40
Direkter Typ 43
Direkter Typ in FBD 43
Einfügen 40
Einfügen in FBD 40
einstellen 46
Impulsabstiegflanke (positiv) 45
Impulsanstiegflanke (positiv) 45
Parallel, Einfügen 40
Parallel, Einfügen in FBD 40
RESET in FBD 47
SET in FBD 46
Typ umkehren 44
Umkehrtyp in FBD 44
Verwendung und verfügbare Typen 40
Verwendung und verfügbare Typen für FBD 40
zurücksetzen 47
SQRT-Funktion 95
SR, Funktionsblock 152
STACKINT, Funktionsblock 507
STIS-Funktion 395
Strompfade
Bezeichnungen, Einfügen 33
Einfügen 33
Funktionsblockdiagramm, Einfügen 33
Subtraktionsoperator 97
SUS, Funktionsblock 521
SYS_INFO, Funktionsblock 389

T

TAN_LREAL-Funktion 100
TAN-Funktion 98
Tastenkombinationen
KOP-Sprache 58
TDF-Funktion 648
TND-Funktion 516
TOF, Funktionsblock 635
TON, Funktionsblock 637
TONOFF, Funktionsblock 639
TOW-Funktion 650
TP, Funktionsblock 642
TRIMPOT_READ, Funktionsblock 392
TRUNC-Funktion 101
TTABLE-Funktion 154

U

UIC-Funktion 397
UID-Funktion 398
UIE-Funktion 400
UIF-Funktion 402
Umkehrung
Kontakte 51
Kontakte in FBD 51
Spulen 44
Spulen in FBD 44
Unterbrechung
STIS-Funktion 395
UIC-Funktion 397
UID-Funktion 398
UIE-Funktion 400
UIF-Funktion 402

V

Vergleichsoperationen: 241, 244, 245, 246, 248, 249
Verzweigungen
Einfügen 37
Für KOP 37

X

XOR_MASK-Funktion 142
XOR-Operator 151

Z

Zähler

CTD, Funktionsblock 251
CTU, Funktionsblock 253
CTUD, Funktionsblock 255

Zeichenfolgen-Manipulation

ASCII-Funktion 615
CHAR-Funktion 617
DELETE-Funktion 618
FIND-Funktion 620
INSERT-Funktion 622
LEFT-Funktion 624
MID-Funktion 626
MLEN-Funktion 628
REPLACE-Funktion 631
RIGHT-Funktion 629

Zeit

DOY-Funktion 646
RTO-Funktionsblock 644
TDF-Funktion 648
TOF, Funktionsblock 635
TON, Funktionsblock 637
TONOFF, Funktionsblock 639
TOW-Funktion 650
TP, Funktionsblock 642

zurücksetzen

Spulen 47

Rockwell Automation Support

Verwenden Sie diese Ressourcen, um auf Supportinformationen zuzugreifen.

Technisches Support-Center	Hier finden Sie Hilfe zu Anleitungsvideos, FAQs, Chats, Benutzerforen und Produktbenachrichtigungsupdates.	rok.auto/Support
Knowledgebase	Greifen Sie auf Knowledgebase-Artikel zu.	rok.auto/Wissensdatenbank
Telefonnummern des lokalen technischen Supports	Suchen Sie die Telefonnummer für Ihr Land.	rok.auto/phonesupport
Literaturbibliothek	Hier finden Sie Installationsanleitungen, Handbücher, Broschüren und Technische Datenpublikationen.	rok.auto/Literatur
Produktkompatibilität und Download Center (PCDC)	Erfahren Sie mehr über die Art und Weise, wie Produkte interagieren, funktionen und funktionen überprüfen und die zugehörige Firmware finden.	rok.auto/pcdc

Feedback zur Dokumentation

Ihre Kommentare helfen uns, Ihre Dokumentationsanforderungen besser umzusetzen. Sollten Sie Vorschläge zur Verbesserung des Inhalts haben, füllen Sie das Formular unter rok.auto/docfeedback aus.

Elektro- und Elektronik-Altgeräte (EEA)

Am Ende der Lebensdauer sollte diese Ausrüstung getrennt von unsortierten Siedlungsabfällen gesammelt werden.

Rockwell Automation verwaltet aktuelle Produkt-Umweltinformationen auf seiner Website unter rok.auto/pec.

Allen-Bradley, expanding human possibility, Logix, Rockwell Automation, and Rockwell Software are trademarks of Rockwell Automation, Inc.

EtherNet/IP is a trademark of ODVA, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Rockwell Otomayson Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752, İçerenkÖy, İstanbul, Tel: +90 (216) 5698400 EEE YÖnetmeliğine Uygundur

rockwellautomation.com — expanding **human possibility**®

AMERIKA: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: +1 414 382 2000, Fax: +1 414 382 4444
EUROPA/NAHER OSTEN/AFRIKA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgien, Tel: +32 2 663 0600, Fax: +32 2 663 0640
ASIEN/AUSTRALIEN/PAZIFIKRAUM: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: +852 2887 4788, Fax: +852 2508 1846
DEUTSCHLAND: Rockwell Automation GmbH, Parsevalstraße 11, 40468 Düsseldorf, Tel: +49 (0)211 41553 0, Fax: +49 (0)211 41553 121
SCHWEIZ: Rockwell Automation AG, Industriestrasse 20, CH-5001 Aarau, Tel: +41(62) 889 77 77, Fax: +41(62) 889 77 11, Customer Service – Tel: 0848 000 277
ÖSTERREICH: Rockwell Automation, Kotzinastraße 9, A-4030 Linz, Tel: +43 (0)732 38 909 0, Fax: +43 (0)732 38 909 61

Rockwell Automation-Publikation 2080-RM001L-DE-E - November 2023
Ersetzt die Publikation 2080-RM001K-DE-E - März 2022

Copyright © 2023 Rockwell Automation Technologies, Inc. Alle Rechte vorbehalten. Printed in the U.S.A.