YASKAWA

VIPA SPEED7 Studio

OPL_SP7 | SPEED7 Studio | Handbuch

HB50 | OPL_SP7 | SPEED7 Studio | de | 19-40

Operationsliste SPEED7 Studio - V1.9.0

YASKAWA Europe GmbH
Ohmstraße 4
91074 Herzogenaurach
Tel.: +49 9132 744 0
Fax: +49 9132 744 186
E-Mail: info@yaskawa.eu.com
Internet: www.yaskawa.eu.com

SP7_000_OPL-SP7,13,DE - © 2019

Inhaltsverzeichnis

1 Zu dieser Dokumentation ... 19
1.1 Copyright © YASKAWA Europe GmbH ... 19
1.2 Zweck der Dokumentation ... 20
1.3 Gültigkeit der Dokumentation ... 20
1.4 Aufbau und Inhalt der Dokumentation ... 20
1.5 Darstellung und Kennzeichnungen ... 22
2 Wichtige Hinweise ... 24
2.1 Allgemein ... 24
2.2 Intern verwendete Bausteine ... 24
3 AWL-Operationen ... 25
3.1 Übersicht ... 25
3.2 Abkürzungen ... 29
3.3 Gegenüberstellung der Syntaxsprachen ... 32
3.4 Unterschiede zwischen SPEED7 und 300V Programmierung ... 33
3.5 Register ... 35
3.6 Adressierungsbeispiele ... 36
3.7 Arithmetische Operationen ... 39
3.8 Baustein-Operationen ... 44
3.9 Bildaufbau- und Null-Operationen ... 45
3.10 Flanken-Operationen ... 46
3.11 Lade-Operationen ... 47
3.12 Schiebe-Operationen ... 50
3.13 Setzen/Rücksetzen von Bitoperanden ... 52
3.14 Sprung-Operationen ... 53
3.15 Transfer-Operationen ... 55
3.16 Umwandlungs-Operationen ... 60
3.17 Vergleichs-Operationen ... 62
3.18 Verknüpfungs-Operationen (Bit) ... 64
3.19 Verknüpfungs-Operationen (Wort) ... 72
3.20 Zeit-Operationen ... 73
3.21 Zähl-Operationen ... 74
4 FUP-Operationen ... 75
4.1 Überblick ... 75
4.2 Bitverknüpfungen ... 75
4.2.1 Überblick ... 75
4.2.2 ODER-Verknüpfung - >=1 ... 76
4.2.3 UND-Verknüpfung - & ... 77
4.2.4 EXKLUSIV-ODER-Verknüpfung - XOR ... 78
4.2.5 Binären Eingang einfügen - --| ... 78
4.2.6 Binären Eingang negieren (NICHT-Operation) - -o| ... 79
4.2.7 Zuweisung - = ... 79
4.2.8 Konnektor - # ... 80
4.2.9 Ausgang rücksetzen - R ... 81
4.2.10 Ausgang setzen - S ... 81
4.2.11 Flipflop rücksetzen setzen - RS ... 82
4.2.12 Flipflop setzen rücksetzen - SR ... 83
4.2.13 Flanke 1-0 erkennen - N ... 84

4.2.14 Flanke 0-1 erkennen - P ... 85
4.2.15 Verknüpfungsergebnis in BIE-Register übernehmen - SAVE ... 85
4.2.16 Signalflanke 1-0 erkennen - NEG ... 86
4.2.17 Signalflanke 0-1 erkennen - POS ... 87
4.3 Vergleicher ... 87
4.3.1 Überblick ... 87
4.3.2 Ganze Zahlen vergleichen (16 Bit) - CMP xx I ... 88
4.3.3 Ganze Zahlen vergleichen (32 Bit) - CMP xx D ... 89
4.3.4 Gleitpunktzahlen vergleichen - CMP xx R ... 89
4.4 Umwandler ... 90
4.4.1 Überblick ... 90
4.4.2 BCD-Zahl in Ganzzahl (16 Bit) wandeln - BCD_I ... 91
4.4.3 Ganzzahl (16 Bit) in BCD-Zahl wandeln - I_BCD ... 92
4.4.4 Ganzzahl (16 Bit) in Ganzzahl (32 Bit) wandeln - I_DI ... 93
4.4.5 BCD-Zahl in Ganzzahl (32 Bit) wandeln - BCD_DI ... 93
4.4.6 Ganzzahl (32 Bit) in BCD-Zahl wandeln - DI_BCD ... 94
4.4.7 Ganzzahl (32 Bit) in Gleitpunktzahl wandeln - DI_R ... 95
4.4.8 Einer-Komplement zu Ganzzahl (16 Bit) erzeugen - INV_I ... 96
4.4.9 Einer-Komplement zu Ganzzahl (32 Bit) erzeugen - INV_DI ... 97
4.4.10 Zweier-Komplement zu Ganzzahl (16 Bit) erzeugen - NEG_I ... 97
4.4.11 Zweier-Komplement zu Ganzzahl (32 Bit) erzeugen - NEG_DI ... 98
4.4.12 Vorzeichen einer Gleitpunktzahl wechseln - NEG_R ... 99
4.4.13 Zahl runden - ROUND ... 100
4.4.14 Ganze Zahl erzeugen - TRUNC ... 101
4.4.15 Aus Gleitpunktzahl nächsthöhere Ganzzahl erzeugen - CEIL ... 102
4.4.16 Aus Gleitpunktzahl nächstniedrige Ganzzahl erzeugen - FLOOR ... 102
4.5 Zähler ... 103
4.5.1 Überblick ... 103
4.5.2 Parametrieren und Vorwärts-/Rückwärtszählen - ZAEHLER ... 104
4.5.3 Parametrieren und Vorwärtszählen - Z_VORW ... 105
4.5.4 Parametrieren und Rückwärtszählen - Z_RUECK ... 106
4.5.5 Setzen des Zähleranfangswerts - SZ ... 108
4.5.6 Vorwärtszählen - ZV ... 108
4.5.7 Rückwärtszählen - ZR ... 109
4.6 Festpunkt-Funktionen ... 110
4.6.1 Überblick ... 110
4.6.2 Ganze Zahlen addieren (16 Bit) - ADD_I ... 110
4.6.3 Ganze Zahlen subtrahieren (16 Bit) - SUB_I ... 111
4.6.4 Ganze Zahlen multiplizieren (16 Bit) - MUL_I ... 112
4.6.5 Ganze Zahlen dividieren (16 Bit) - DIV_I ... 113
4.6.6 Ganze Zahlen addieren (32 Bit) - ADD_DI ... 114
4.6.7 Ganze Zahlen subtrahieren (32 Bit) - SUB_DI ... 114
4.6.8 Ganze Zahlen multiplizieren (32 Bit) - MUL_DI ... 115
4.6.9 Ganze Zahlen dividieren (32 Bit) - DIV_DI ... 116
4.6.10 Divisionsrest gewinnen (32 Bit) - MOD_DI ... 117
4.7 Gleitpunkt-Funktionen ... 118
4.7.1 Überblick ... 118
4.7.2 Gleitpunktzahlen addieren - ADD_R ... 119
4.7.3 Gleitpunktzahlen subtrahieren - SUB_R ... 120
4.7.4 Gleitpunktzahlen multiplizieren - MUL_R ... 121

4.7.5 Gleitpunktzahlen dividieren - DIV_R ... 122
4.7.6 Absolutwert einer Gleitpunktzahl - ABS ... 122
4.7.7 Quadratwurzel einer Gleitpunktzahl - SQRT ... 123
4.7.8 Quadrat einer Gleitpunktzahl - SQR ... 124
4.7.9 Natürlichen Logarithmus einer Gleitpunktzahl - LN ... 125
4.7.10 Exponentialwert einer Gleitpunktzahl - EXP ... 126
4.7.11 Sinus einer Gleitpunktzahl - SIN ... 126
4.7.12 Cosinus einer Gleitpunktzahl - COS ... 127
4.7.13 Tangens einer Gleitpunktzahl - TAN ... 128
4.7.14 Arkussinus einer Gleitpunktzahl - ASIN ... 129
4.7.15 Arkuscosinus einer Gleitpunktzahl - ACOS ... 129
4.7.16 Arkustangens einer Gleitpunktzahl - ATAN ... 130
4.8 Verschieben ... 131
4.8.1 Wert übertragen - MOVE ... 131
4.9 Programmsteuerung ... 132
4.9.1 Überblick ... 132
4.9.2 Baustein aufrufen - CALL ... 132
4.9.3 Springe zurück - RET ... 133
4.9.4 Datenbaustein öffnen - OPN ... 134
4.9.5 Zur Sprungmarke springen (absolut oder wenn 1) - JMP ... 135
4.9.6 Zur Sprungmarke springen (wenn 0) - JMPN ... 136
4.9.7 Sprungmarke - Label ... 136
4.10 Schieben/Rotieren ... 137
4.10.1 Überblick ... 137
4.10.2 Ganzzahl (16 Bit) rechts schieben - SHR_I ... 138
4.10.3 Ganzzahl (32 Bit) rechts schieben - SHR_DI ... 139
4.10.4 16 Bit links schieben - SHL_W ... 140
4.10.5 16 Bit rechts schieben - SHR_W ... 140
4.10.6 32 Bit links schieben - SHL_DW ... 141
4.10.7 32 Bit rechts schieben - SHR_DW ... 142
4.10.8 32 Bit links rotieren - ROL_DW ... 143
4.10.9 32 Bit rechts rotieren - ROR_DW ... 144
4.11 Zeiten ... 145
4.11.1 Überblick ... 145
4.11.2 Zeit als Impuls parametrieren und starten - S_IMPULS ... 146
4.11.3 Zeit als verlängerten Impuls parametrieren und starten - S_VIMP ... 147
4.11.4 Zeit als Einschaltverzögerung parametrieren und starten - S_EVERZ ... 149
4.11.5 Zeit als speichernde Einschaltverzögerung parametrieren und starten - S_SEVERZ ... 151
4.11.6 Zeit als Ausschaltverzögerung parametrieren und starten - S_AVERZ ... 153
4.11.7 Zeit als Impuls starten - SI ... 154
4.11.8 Zeit als verlängerten Impuls starten - SV ... 155
4.11.9 Zeit als Einschaltverzögerung starten - SE ... 156
4.11.10 Zeit als speichernde Einschaltverzögerung starten - SS ... 157
4.11.11 Zeit als Ausschaltverzögerung starten - SA ... 158
4.12 Wortverknüpfungen ... 159
4.12.1 Übersicht ... 159
4.12.2 16 Bit UND-Verknüpfung - WAND_W ... 160
4.12.3 16 Bit ODER-Verknüpfung - WOR_W ... 161
4.12.4 16 Bit EXKLUSIV-ODER-Verknüpfung - WXOR_W ... 162

4.12.5 32 Bit UND-Verknüpfung - WAND_DW ... 163
4.12.6 32 Bit ODER-Verknüpfung - WOR_DW ... 164
4.12.7 32 Bit EXKLUSIV-ODER-Verknüpfung - WXOR_DW ... 165
4.13 Statusbits ... 166
4.13.1 Überblick ... 166
4.13.2 Störungsbit Überlauf - OV ... 166
4.13.3 Störungsbit Überlauf gespeichert - OS ... 167
4.13.4 Störungsbit Ungültige Operation - UO ... 168
4.13.5 Störungsbit BIE-Register - BIE ... 169
4.13.6 Ergebnisbit - x0 ... 170
5 KOP-Operationen ... 172
5.1 Überblick ... 172
5.2 Bitverknüpfungen ... 172
5.2.1 Überblick ... 172
5.2.2 Schließerkontakt - ---| |--- ... 173
5.2.3 Öffnerkontakt - ---|/|--- ... 174
5.2.4 EXKLUSIV-ODER-Verknüpfung - XOR ... 174
5.2.5 Verknüpfungsergebnis invertieren - ---|NOT|--- ... 175
5.2.6 Relaisspule, Ausgang - ---( ) ... 176
5.2.7 Konnektor - ---(#)--- ... 176
5.2.8 Ausgang rücksetzen - ---(R) ... 177
5.2.9 Ausgang setzen - ---(S) ... 178
5.2.10 Flipflop rücksetzen setzen - RS ... 178
5.2.11 Flipflop setzen rücksetzen - SR ... 179
5.2.12 Flanke 1-0 erkennen - ---(N)--- ... 180
5.2.13 Flanke 0-1 erkennen - ---(P)--- ... 181
5.2.14 Verknüpfungsergebnis in BIE-Register übernehmen - ---(SAVE) ... 182
5.2.15 Signalflanke 1 - 0 erkennen - NEG ... 183
5.2.16 Signalflanke 0 - 1 erkennen - POS ... 183
5.3 Vergleicher ... 184
5.3.1 Überblick ... 184
5.3.2 Ganze Zahlen vergleichen (16 Bit) - CMP xx I ... 185
5.3.3 Ganze Zahlen vergleichen (32 Bit) - CMP xx D ... 185
5.3.4 Gleitpunktzahlen vergleichen - CMP xx R ... 186
5.4 Umwandler ... 187
5.4.1 Überblick ... 187
5.4.2 BCD-Zahl in Ganzzahl (16 Bit) wandeln - BCD_I ... 188
5.4.3 Ganzzahl (16 Bit) in BCD-Zahl wandeln - I_BCD ... 189
5.4.4 Ganzzahl (16 Bit) in Ganzzahl (32 Bit) wandeln - I_DI ... 189
5.4.5 BCD-Zahl in Ganzzahl (32 Bit) wandeln - BCD_DI ... 190
5.4.6 Ganzzahl (32 Bit) in BCD-Zahl wandeln - DI_BCD ... 191
5.4.7 Ganzzahl (32 Bit) in Gleitpunktzahl wandeln - DI_R ... 192
5.4.8 Einer-Komplement zu Ganzzahl (16 Bit) erzeugen - INV_I ... 193
5.4.9 Einer-Komplement zu Ganzzahl (32 Bit) erzeugen - INV_DI ... 194
5.4.10 Zweier-Komplement zu Ganzzahl (16 Bit) erzeugen - NEG_I ... 194
5.4.11 Zweier-Komplement zu Ganzzahl (32 Bit) erzeugen - NEG_DI ... 195
5.4.12 Vorzeichen einer Gleitpunktzahl wechseln - NEG_R ... 196
5.4.13 Zahl runden - ROUND ... 197
5.4.14 Ganze Zahl erzeugen - TRUNC ... 198

5.4.15 Aus Gleitpunktzahl nächsthöhere Ganzzahl erzeugen - CEIL ... 199
5.4.16 Aus Gleitpunktzahl nächstniedrige Ganzzahl erzeugen - FLOOR ... 200
5.5 Zähler ... 200
5.5.1 Überblick ... 200
5.5.2 Parametrieren und Vorwärts-/Rückwärtszählen - ZAEHLER ... 201
5.5.3 Parametrieren und Vorwärtszählen - Z_VORW ... 202
5.5.4 Parametrieren und Rückwärtszählen - Z_RUECK ... 204
5.5.5 Setzen des Zähleranfangswerts - --(SZ) ... 205
5.5.6 Vorwärtszählen - --(ZV) ... 206
5.5.7 Rückwärtszählen - --(ZR) ... 207
5.6 Festpunkt-Funktionen ... 207
5.6.1 Überblick ... 207
5.6.2 Ganze Zahlen addieren (16 Bit) - ADD_I ... 208
5.6.3 Ganze Zahlen subtrahieren (16 Bit) - SUB_I ... 209
5.6.4 Ganze Zahlen multiplizieren (16 Bit) - MUL_I ... 210
5.6.5 Ganze Zahlen dividieren (16 Bit) - DIV_I ... 210
5.6.6 Ganze Zahlen addieren (32 Bit) - ADD_DI ... 211
5.6.7 Ganze Zahlen subtrahieren (32 Bit) - SUB_DI ... 212
5.6.8 Ganze Zahlen multiplizieren (32 Bit) - MUL_DI ... 213
5.6.9 Ganze Zahlen dividieren (32 Bit) - DIV_DI ... 214
5.6.10 Divisionsrest gewinnen (32 Bit) - MOD_DI ... 215
5.7 Gleitpunkt-Funktionen ... 216
5.7.1 Überblick ... 216
5.7.2 Gleitpunktzahlen addieren - ADD_R ... 217
5.7.3 Gleitpunktzahlen subtrahieren - SUB_R ... 218
5.7.4 Gleitpunktzahlen multiplizieren - MUL_R ... 219
5.7.5 Gleitpunktzahlen dividieren - DIV_R ... 220
5.7.6 Absolutwert einer Gleitpunktzahl - ABS ... 221
5.7.7 Quadratwurzel einer Gleitpunktzahl - SQRT ... 222
5.7.8 Quadrat einer Gleitpunktzahl - SQR ... 222
5.7.9 Natürlichen Logarithmus einer Gleitpunktzahl - LN ... 223
5.7.10 Exponentialwert einer Gleitpunktzahl - EXP ... 224
5.7.11 Sinus einer Gleitpunktzahl - SIN ... 225
5.7.12 Cosinus einer Gleitpunktzahl - COS ... 226
5.7.13 Tangens einer Gleitpunktzahl - TAN ... 226
5.7.14 Arkussinus einer Gleitpunktzahl - ASIN ... 227
5.7.15 Arkuscosinus einer Gleitpunktzahl - ACOS ... 228
5.7.16 Arkustangens einer Gleitpunktzahl - ATAN ... 229
5.8 Verschieben ... 230
5.8.1 Wert übertragen - MOVE ... 230
5.9 Programmsteuerung ... 231
5.9.1 Überblick ... 231
5.9.2 Baustein aufrufen - --(CALL) ... 231
5.9.3 Springe zurück - --(RET) ... 232
5.9.4 Datenbaustein öffnen - --(OPN) ... 232
5.9.5 Zur Sprungmarke springen (absolut oder wenn 1) - --(JMP) ... 233
5.9.6 Zur Sprungmarke springen (wenn 0) - --(JMPN) ... 234
5.9.7 Sprungmarke - Label ... 235
5.10 Schieben/Rotieren ... 236
5.10.1 Überblick ... 236

5.10.2 Ganzzahl (16 Bit) rechts schieben - SHR_I ... 236
5.10.3 Ganzzahl (32 Bit) rechts schieben - SHR_DI ... 237
5.10.4 16 Bit links schieben - SHL_W ... 238
5.10.5 16 Bit rechts schieben - SHR_W ... 239
5.10.6 32 Bit links schieben - SHL_DW ... 240
5.10.7 32 Bit rechts schieben - SHR_DW ... 241
5.10.8 32 Bit links rotieren - ROL_DW ... 242
5.10.9 32 Bit rechts rotieren - ROR_DW ... 243
5.11 Zeiten ... 244
5.11.1 Überblick ... 244
5.11.2 Zeit als Impuls parametrieren und starten - S_IMPULS ... 244
5.11.3 Zeit als verlängerten Impuls parametrieren und starten - S_VIMP ... 246
5.11.4 Zeit als Einschaltverzögerung parametrieren und starten - S_EVERZ ... 248
5.11.5 Zeit als speichernde Einschaltverzögerung parametrieren und starten - S_SEVERZ ... 249
5.11.6 Zeit als Ausschaltverzögerung parametrieren und starten - S_AVERZ ... 251
5.11.7 Zeit als Impuls starten - --(SI) ... 253
5.11.8 Zeit als verlängerten Impuls starten - --(SV) ... 254
5.11.9 Zeit als Einschaltverzögerung starten - --(SE) ... 255
5.11.10 Zeit als speichernde Einschaltverzögerung starten - --(SS) ... 256
5.11.11 Zeit als Ausschaltverzögerung starten - --(SA) ... 257
5.12 Wortverknüpfungen ... 258
5.12.1 Übersicht ... 258
5.12.2 16 Bit UND-Verknüpfung - WAND_W ... 258
5.12.3 16 Bit ODER-Verknüpfung - WOR_W ... 259
5.12.4 16 Bit EXKLUSIV-ODER-Verknüpfung - WXOR_W ... 260
5.12.5 32 Bit UND-Verknüpfung - WAND_DW ... 261
5.12.6 32 Bit ODER-Verknüpfung - WOR_DW ... 262
5.12.7 32 Bit EXKLUSIV-ODER-Verknüpfung - WXOR_DW ... 263
5.13 Statusbits ... 264
5.13.1 Überblick ... 264
5.13.2 Störungsbit Überlauf - OV --||-- ... 265
5.13.3 Störungsbit Überlauf gespeichert - OS --||-- ... 266
5.13.4 Störungsbit Ungültige Operation - UO --||-- ... 266
5.13.5 Störungsbit BIE-Register - BIE --||-- ... 267
5.13.6 Ergebnisbit - x0 --||-- ... 268

6 Bausteinparameter ... 269
6.1 Allgemeine und spezifische Fehlercodes RET_VAL ... 269

7 Organisationsbausteine - *"Organization Blocks"* ... 272
7.1 Übersicht ... 272
7.2 Hauptprogramm - *"Main"* ... 272
7.2.1 OB 1 - Main - Zyklisches Programm ... 272
7.3 Anlauf - *"Startup"* ... 273
7.3.1 OB 100, OB 102 - Complete / Cold Restart - Anlauf ... 273
7.4 Kommunikationsalarme - *"Communication Interrupts"* ... 275
7.4.1 OB 55 - DP: Status Alarm - Statusalarm ... 275
7.4.2 OB 56 - DP: Update Alarm - Update-Alarm ... 276
7.4.3 OB 57 - DP: Manufacture Alarm - Herstellerspezifische Alarme ... 277
7.5 Verzögerungsalarme - *"Time delay Interrupts"* ... 278

7.5.1 OB 20, OB 21 - DEL_INTx - Verzögerungsalarm ... 278
7.6 Uhrzeitalarme - *"Time of day Interrupts"* ... 279
7.6.1 OB 10, OB 11 - TOD_INTx - Uhrzeitalarm ... 279
7.7 Zyklusalarme - *"Cyclic Interrupts"* ... 281
7.7.1 OB 28, 29, 32, 33, 34, 35 - CYC_INTx - Weckalarm ... 281
7.8 Prozessalarme - *"Hardware Interrupts"* ... 283
7.8.1 OB 40, OB 41 - HW_INTx - Prozessalarm ... 283
7.9 Asynchrone Fehleralarme - *"Asynchronous error Interrupts"* ... 284
7.9.1 OB 80 - CYCL_FLT - Zeitfehler ... 284
7.9.2 OB 81 - PS_FLT - Stromversorgungsfehler ... 287
7.9.3 OB 82 - I/O_FLT1 - Diagnosealarm ... 288
7.9.4 OB 83 - I/O_FLT2 - Ziehen / Stecken ... 290
7.9.5 OB 85 - OBNL_FLT - Programmablauffehler ... 293
7.9.6 OB 86 - RACK_FLT - Slaveausfall / -wiederkehr ... 297
7.10 Synchronalarme - *"Synchronous Interrupts"* ... 299
7.10.1 OB 121 - PROG_ERR - Programmierfehler ... 299
7.10.2 OB 122 - MOD_ERR - Peripheriezugriffsfehler ... 302
7.11 Taktsynchronalarme - *"Cycle synchronous Interrupts"* ... 303
7.11.1 OB 60 - MULTI_INT - Multicomputingalarm ... 303
7.11.2 OB 61 - SYNC_1 - Taktsynchronalarm ... 303

8 Gebäude-Automatisierung - *"Building Control"* ... 305
8.1 Übersicht ... 305
8.1.1 Aufruf-Beispiel - Instanz-DB ... 305
8.1.2 Aufruf-Beispiel - Multiinstanzen-DB ... 305
8.2 Raumsteuerung - *"Room"* ... 306
8.2.1 FB 45 - LAMP - Leuchte / Steckdose steuern ... 306
8.2.2 FB 46 - BLIND - Jalousie steuern ... 307
8.2.3 FB 47 - DSTRIKE - Elektrischer Türöffner ... 309
8.3 Zugangskontrolle - *"Access Control"* ... 310
8.3.1 FB 48 - ACONTROL - Zutrittssteuerung ... 310
8.3.2 UDT 3 - ACLREC - Datenstruktur für FB 48 ... 311
8.3.3 UDT 4 - ACL - Datenstruktur für FB 48 ... 312
8.3.4 FB 49 - KEYPAD - Tastatur ... 313
8.3.5 FB 50 - KEYPAD2 - Tastatur ... 315

9 Netzwerkkommunikation - *"Network Communication"* ... 318
9.1 Offene Kommunikation - *"Open Communication"* ... 318
9.1.1 Verbindungsorientierte Protokolle ... 318
9.1.2 Verbindungslose Protokolle ... 318
9.1.3 FB 63 - TSEND - Daten senden - TCP native und ISO on TCP ... 319
9.1.4 FB 64 - TRCV - Daten empfangen - TCP native und ISO on TCP ... 322
9.1.5 FB 65 - TCON - Verbindung aufbauen ... 326
9.1.6 UDT 65 - TCON_PAR - Datenstruktur für FB 65 ... 328
9.1.7 FB 66 - TDISCON - Verbindung abbauen ... 333
9.2 Ethernet-Kommunikation - *"Ethernet Communication"* ... 335
9.2.1 Kommunikation - FC 5...6 für CP 343 ... 335
9.2.2 FC 5 - AG_SEND - Senden an CP 343 ... 337
9.2.3 FC 6 - AG_RECV - Empfangen von CP 343 ... 340
9.2.4 FC 10 - AG_CNTRL - Control CP 343 ... 342
9.2.5 FC 62 - C_CNTR - Zustand einer Verbindung abfragen ... 350

9.2.6 FB/SFB 8 - FB 55 - Übersicht ... 351
9.2.7 FB/SFB 8 - USEND - Unkoordiniertes Senden ... 352
9.2.8 FB/SFB 9 - URCV - Unkoordiniertes Empfangen ... 354
9.2.9 FB/SFB 12 - BSEND - Blockorientiertes Senden ... 356
9.2.10 FB/SFB 13 - BRCV - Blockorientiertes Empfangen ... 359
9.2.11 FB/SFB 14 - GET - Remote CPU lesen ... 361
9.2.12 FB/SFB 15 - PUT - Remote CPU schreiben ... 363
9.2.13 FB 55 - IP_CONF - Progr. Kommunikationsverbindungen ... 365

10 Modbus-Kommunikation - *"Modbus Communication"* ... 380
10.1 TCP ... 380
10.1.1 FB 70 - TCP_MB_CLIENT - Modbus/TCP-Client ... 380
10.1.2 FB 71 - TCP_MB_SERVER - Modbus/TCP-Server ... 383
10.2 RTU ... 387
10.2.1 FB 72 - RTU_MB_MASTER - Modbus-RTU-Master ... 387
10.2.2 FB 73 - RTU_MB_SLAVE - Modbus-RTU-Slave ... 390
10.3 Modbus Exception Codes - *Exception Codes* ... 395
10.4 FKT Modbus-Funktionscodes - *FKT Codes* ... 396

11 Serielle Kommunikation - *"Serial Communication"* ... 402
11.1 Serielle Kommunikation - *"Serial Communication"* ... 402
11.1.1 SFC 207 - SER_CTRL - Modemfunktionalität PtP ... 402
11.1.2 FC/SFC 216 - SER_CFG - Parametrierung PtP ... 403
11.1.3 FC/SFC 217 - SER_SND - Senden an PtP ... 406
11.1.4 FC/SFC 218 - SER_RCV - Empfangen von PtP ... 412
11.1.5 FB 1 - RECEIVE_ASCII - Empfangen mit definierter Länge von PtP ... 414
11.1.6 FB 7 - P_RCV_RK - Empfangen von CP 341 ... 415
11.1.7 FB 8 - P_SND_RK - Senden an CP 341 ... 416
11.2 CP040 ... 418
11.2.1 Übersicht ... 418
11.2.2 FB 60 - SEND - Senden an System SLIO CP 040 ... 419
11.2.3 FB 61 - RECEIVE - Empfangen von System SLIO CP 040 ... 421
11.2.4 FB 65 - CP040_COM - Kommunikation SLIO CP 040 ... 424
11.3 CP240 ... 427
11.3.1 FC 0 - SEND_ASCII_STX_3964 - Senden an CP 240 ... 427
11.3.2 FC 1 - RECEIVE_ASCII_STX_3964 - Empfangen von CP 240 ... 429
11.3.3 FC 8 - STEUERBIT - Modemfunktionalität CP 240 ... 430
11.3.4 FC 9 - SYNCHRON_RESET - Synchronisation CPU und CP 240 ... 431
11.3.5 FC 11 - ASCII_FRAGMENT - Fragment Datenempfang CP 240 ... 432

12 EtherCAT-Kommunikation - *"EtherCAT Communication"* ... 434
12.1 SDO-Kommunikation - *"SDO Communication"* ... 434
12.1.1 FB 52 - SDO_READ - Lesezugriff auf Objektverzeichnis ... 434
12.1.2 FB 53 - SDO_WRITE - Schreibzugriff auf Objektverzeichnis ... 438

13 Modulspezifisch - *"Device Specific"* ... 442
13.1 Frequenzmessung - *"Frequency Measurement"* ... 442
13.1.1 FC 300 ... 303 - Frequenzmessung SLIO konsistent ... 442
13.1.2 FC 300 - FM_SET_CONTROL - Control Frequenzmessung konsistent ... 442
13.1.3 FC 301 - FM_GET_PERIOD - Periodendauer berechnen konsistent ... 444
13.1.4 FC 302 - FM_GET_FREQUENCY - Frequenz berechnen konsistent ... 446
13.1.5 FC 303 - FM_GET_SPEED - Drehzahl berechnen konsistent ... 448
13.1.6 FC 310 ... 313 - Frequenzmessung SLIO ... 450

13.1.7 FC 310 - FM_CONTROL - Control Frequenzmessung ... 450
13.1.8 FC 311 - FM_CALC_PERIOD - Periodendauer berechnen ... 452
13.1.9 FC 312 - FM_CALC_FREQUENCY - Frequenz berechnen ... 454
13.1.10 FC 313 - FM_CALC_SPEED - Drehzahl berechnen ... 456
13.2 Energiemessung - *"Energy Measurement"* ... 458
13.2.1 Übersicht ... 458
13.2.2 FB 325 - EM_COM_R1 - Kommunikation mit 031-1PAxx ... 461
13.2.3 UDT 325 - EM_DATA_R1 - Datenstruktur für FB 325 ... 462
13.3 Motion-Module - *"Motion Modules"* ... 465
13.3.1 Übersicht ... 465
13.3.2 FB 320 - ACYC_RW - Azyklischer Zugriff auf System SLIO Motion-Modul ... 467
13.3.3 FB 321 - ACYC_DS - Azyklische Parametrierung System SLIO Motion-Modul ... 470
13.3.4 UDT 321 - ACYC_OBJECT-DATA - Datenstruktur für FB 321 ... 474
13.4 RAM nach WLD - *"WLD"* ... 475
13.4.1 FB 240 - RAM_to_s7prog.wld - RAM nach s7prog.wld ... 475
13.4.2 FB 241 - RAM_to_autoload.wld - RAM nach autoload.wld ... 475
14 Antriebssteuerung - *"Motion Control"* ... **476**
14.1 Übersicht ... 476
14.1.1 Funktionsbausteine ... 476
14.1.2 Funktionsweise ... 478
14.1.3 Zustände ... 480
14.1.4 Verhalten der Ein- und Ausgänge ... 481
14.1.5 Ablöseverhalten von Bewegungsaufträgen ... 482
14.1.6 Erweiterte Vorgaben von Achsdaten ... 484
14.2 Typen - Datenstrukturen ... 485
14.2.1 UDT 8189 - MC_TRIGGER_REF - Datenstruktur Triggersignal ... 485
14.2.2 UDT 8190 - MC_DIAGNOSTIC_REF - Datenstruktur interner Status ... 485
14.2.3 UDT 8191 - MC_SETUP_UDT - Datenstruktur Konfigurationsdaten ... 485
14.2.4 UDT 8192 - MC_AXIS_REF - Datenstruktur Achsdaten ... 485
14.3 Single Axis - Einfachachsen ... 486
14.3.1 FB 700 - MC_Power - Achsenfreigabe ... 486
14.3.2 FB 701 - MC_Home - Achse referenzieren ... 487
14.3.3 FB 702 - MC_Stop - Achse stoppen ... 489
14.3.4 FB 703 - MC_Halt - Achse anhalten ... 491
14.3.5 FB 704 - MC_MoveRelative - Achse relativ verfahren ... 493
14.3.6 FB 705 - MC_MoveVelocity - Achse verfahren mit konstanter Geschwindigkeit ... 495
14.3.7 FB 708 - MC_MoveAbsolute - Achse auf absolute Position verfahren ... 497
14.3.8 FB 711 - MC_Reset - Reset Achse ... 500
14.3.9 FB 712 - MC_ReadStatus - PLCopen-State ... 502
14.3.10 FB 713 - MC_ReadAxisError - Fehler Achse lesen ... 504
14.3.11 FB 714 - MC_ReadParameter - Parameter Achse lesen ... 506
14.3.12 FB 715 - MC_WriteParameter - Parameter Achse schreiben ... 507
14.3.13 FB 716 - MC_ReadActualPosition - Position Achse auslesen ... 509
14.3.14 FB 717 - MC_ReadActualVelocity - Geschwindigkeit der Achse lesen ... 510
14.3.15 FB 718 - MC_ReadAxisInfo - Zusatzinformationen der Achse lesen ... 512
14.3.16 FB 719 - MC_ReadMotionState - Zustand Bewegungsauftrag lesen ... 513
14.3.17 FB 722 - MC_MoveSuperimposed - überlagerte Positionierung ... 515

14.3.18 FB 723 - MC_TouchProbe - Achsposition erfassen ... 517
14.3.19 FB 724 - MC_AbortTrigger - Achsposition erfassen abbrechen ... 518
14.3.20 FB 725 - MC_ReadBoolParameter - Boolean-Parameter von Achse lesen ... 519
14.3.21 FB 726 - MC_WriteBoolParameter - Boolean-Parameter an Achse schreiben ... 521
14.3.22 FB 727 - VMC_ReadDWordParameter - Doppelwort-Parameter von Achse lesen ... 523
14.3.23 FB 728 - VMC_WriteDWordParameter - Doppelwort-Parameter an Achse schreiben ... 525
14.3.24 FB 729 - VMC_ReadWordParameter - Wort-Parameter von Achse lesen ... 527
14.3.25 FB 730 - VMC_WriteWordParameter - Wort-Parameter an Achse schreiben ... 529
14.3.26 FB 731 - VMC_ReadByteParameter - Byte-Parameter von Achse lesen ... 531
14.3.27 FB 732 - VMC_WriteByteParameter - Byte-Parameter an Achse schreiben ... 533
14.3.28 FB 733 - VMC_ReadDriveParameter - Antriebsparameter lesen ... 535
14.3.29 FB 734 - VMC_WriteDriveParameter - Antriebsparameter schreiben ... 537
14.3.30 FB 735 - VMC_HomeInit_LimitSwitch - Initialisierung Referenzfahrt auf Endschalter ... 539
14.3.31 FB 736 - VMC_HomeInit_HomeSwitch - Initialisierung Referenzfahrt auf Referenzschalter ... 541
14.3.32 FB 737 - VMC_HomeInit_ZeroPulse - Initialisierung Referenzfahrt auf Null Impuls ... 543
14.3.33 FB 738 - VMC_HomeInit_SetPosition - Initialisierung Referenzfahrt auf aktuelle Position ... 544
14.3.34 FB 739 - MC_TorqueControl - Drehmomentsteuerung ... 545
14.3.35 FB 740 - MC_ReadActualTorque - Drehmoment der Achse lesen ... 547
14.4 Multi axis - Mehrfachachsen ... 549
14.4.1 FB 706 - MC_CamIn - Kopplung Master-Slave-Achse über Kurvenscheibe ... 549
14.4.2 FB 707 - MC_GearIn - Kopplung Master-Slave-Achse über Geschwindigkeit ... 559
14.4.3 FB 709 - MC_CamOut - Deaktivierung Master-Slave-Kopplung über Kurvenscheibe ... 562
14.4.4 FB 710 - MC_GearOut - Deaktivierung Master-Slave-Kopplung über Geschwindigkeit ... 564
14.4.5 FB 720 - MC_PhasingAbsolute - Phasenverschiebung absolut ... 566
14.4.6 FB 721 - MC_PhasingRelative - Phasenverschiebung relativ ... 568
14.5 ErrorID - Zusätzliche Fehlerinformationen ... 569
14.6 PLCopen Parameter ... 574
14.7 VIPA-spezifische Parameter ... 575
15 Antriebssteuerung - *Simple Motion Control Library* ... **577**
15.1 Übersicht ... 577
15.2 Einsatz *Sigma-5/7* EtherCAT ... 578
15.2.1 Einsatz *Sigma-5* EtherCAT ... 578
15.2.2 Einsatz *Sigma-7S* EtherCAT ... 598
15.2.3 Einsatz *Sigma-7W* EtherCAT ... 619
15.3 Einsatz *Sigma-5/7* PROFINET ... 643
15.3.1 Einsatz *Sigma-5* PROFINET ... 643
15.3.2 Einsatz *Sigma-7* PROFINET ... 658

15.3.3 Antriebsspezifische Bausteine ... 672
15.4 Einsatz *Sigma-5/7* Pulse Train ... 679
15.4.1 Übersicht ... 679
15.4.2 Parameter am Antrieb einstellen ... 679
15.4.3 Beschaltung ... 680
15.4.4 Einsatz im VIPA *SPEED7 Studio* ... 682
15.4.5 Antriebsspezifischer Baustein ... 686
15.5 Einsatz Frequenzumrichter über PWM ... 694
15.5.1 Übersicht ... 694
15.5.2 Parameter am Frequenzumrichter einstellen ... 695
15.5.3 Beschaltung ... 696
15.5.4 Einsatz im VIPA *SPEED7 Studio* ... 697
15.5.5 Antriebsspezifischer Baustein ... 702
15.6 Einsatz Frequenzumrichter über Modbus RTU ... 705
15.6.1 Übersicht ... 705
15.6.2 Parameter am Frequenzumrichter einstellen ... 706
15.6.3 Beschaltung ... 708
15.6.4 Einsatz im VIPA *SPEED7 Studio* ... 711
15.6.5 Antriebsspezifische Bausteine ... 726
15.7 Einsatz Frequenzumrichter über EtherCAT ... 735
15.7.1 Übersicht ... 735
15.7.2 Parameter am Frequenzumrichter einstellen ... 735
15.7.3 Beschaltung ... 736
15.7.4 Einsatz im VIPA *SPEED7 Studio* ... 737
15.7.5 Antriebsspezifische Bausteine ... 750
15.8 Bausteine zur Achskontrolle ... 751
15.8.1 Übersicht ... 751
15.8.2 Einfache Bewegungsaufgaben ... 754
15.8.3 Komplexe Bewegungsaufgaben - PLCopen-Bausteine ... 758
15.9 Antrieb über HMI steuern ... 825
15.9.1 Übersicht ... 825
15.9.2 Neues Projekt erstellen ... 826
15.9.3 Projekt in Movicon anpassen ... 831
15.9.4 Inbetriebnahme ... 841
15.10 Zustände und Verhalten der Ausgänge ... 845
15.10.1 Zustände ... 845
15.10.2 Ablöseverhalten von Bewegungsaufträgen ... 847
15.10.3 Verhalten der Ein- und Ausgänge ... 849
15.11 ErrorID - Zusätzliche Fehlerinformationen ... 851
16 Integrierte Standardfunktionen - *"Integrated Standard"* ... 859
16.1 System-SFCs - *"System Functions"* ... 859
16.1.1 SFC 0 - SET_CLK - Uhrzeit stellen ... 859
16.1.2 SFC 1 - READ_CLK - Uhrzeit lesen ... 859
16.1.3 SFC 2 ... 4 - Betriebsstundenzähler ... 860
16.1.4 SFC 2 - SET_RTM - Betriebsstundenzähler setzen ... 860
16.1.5 SFC 3 - CTRL_RTM - Betriebsstundenzähler starten/stoppen ... 861
16.1.6 SFC 4 - READ_RTM - Betriebsstundenzähler auslesen ... 862
16.1.7 SFC 5 - GADR_LGC - Logische Adresse eines Kanals ermitteln ... 862
16.1.8 SFC 6 - RD_SINFO - Startinformation auslesen ... 864

16.1.9 SFC 7 - DP_PRAL - Prozessalarm beim DP-Master auslösen ... 866
16.1.10 SFC 12 - D_ACT_DP - DP-Slave aktivieren und deaktivieren ... 867
16.1.11 SFC 13 - DPNRM_DG - Slave-Diagnosedaten lesen ... 871
16.1.12 SFC 14 - DPRD_DAT - Konsistente Nutzdaten lesen ... 873
16.1.13 SFC 15 - DPWR_DAT - Konsistente Nutzdaten schreiben ... 874
16.1.14 SFC 17 - ALARM_SQ und SFC 18 - ALARM_S ... 876
16.1.15 SFC 19 - ALARM_SC - Quittierzustand der letzten Meldung ... 878
16.1.16 SFC 20 - BLKMOV - Variable kopieren ... 879
16.1.17 SFC 21 - FILL - Feld vorbesetzen ... 880
16.1.18 SFC 22 - CREAT_DB - Datenbaustein erzeugen ... 881
16.1.19 SFC 23 - DEL_DB - Datenbaustein löschen ... 883
16.1.20 SFC 24 - TEST_DB - Datenbaustein testen ... 884
16.1.21 FC/SFC 25 - COMPRESS - Komprimieren Anwenderspeicher ... 884
16.1.22 SFC 28 ... SFC 31 - Uhrzeitalarm ... 885
16.1.23 SFC 32 - SRT_DINT - Verzögerungsalarm starten ... 888
16.1.24 SFC 33 - CAN_DINT - Verzögerungsalarm stornieren ... 889
16.1.25 SFC 34 - QRY_DINT - Verzögerungsalarm Status abfragen ... 890
16.1.26 SFC 36 - MSK_FLT - Synchronfehlerereignisse maskieren ... 891
16.1.27 SFC 37 - DMSK_FLT - Synchronfehlerereignisse demaskieren ... 892
16.1.28 SFC 38 - READ_ERR - Ereignisstatusregister lesen ... 892
16.1.29 SFC 39 - DIS_IRT - Alarmereignisse sperren ... 893
16.1.30 SFC 40 - EN_IRT - Gesperrte Alarmereignisse freigeben ... 894
16.1.31 SFC 41 - DIS_AIRT - Alarmereignisse verzögern ... 895
16.1.32 SFC 42 - EN_AIRT - Verzögerte Alarmereignissen freigeben ... 896
16.1.33 SFC 43 - RE_TRIGR - Zykluszeitüberwachung neu starten ... 896
16.1.34 SFC 44 - REPL_VAL - Ersatzwert in AKKU1 übertragen ... 897
16.1.35 SFC 46 - STP - CPU in STOP überführen ... 897
16.1.36 SFC 47 - WAIT - Verzögern des Anwenderprogramms ... 897
16.1.37 SFC 49 - LGC_GADR - Steckplatz ermitteln ... 898
16.1.38 SFC 50 - RD_LGADR - Alle Adressen eines Moduls lesen ... 899
16.1.39 SFC 51 - RDSYSST - Auslesen der Informationen der SZL ... 900
16.1.40 SFC 52 - WR_USMSG - Eintrag in Diagnosepuffer schreiben ... 902
16.1.41 FC/SFC 54 - RD_DPARM - Vordefinierte Parameter lesen ... 904
16.1.42 SFC 55 - WR_PARM - Dynamische Parameter schreiben ... 905
16.1.43 SFC 56 - WR_DPARM - Vordefinierte Parameter schreiben ... 907
16.1.44 SFC 57 - PARM_MOD - Modul parametrieren ... 909
16.1.45 SFC 58 - WR_REC - Datensatz schreiben ... 912
16.1.46 SFC 59 - RD_REC - Datensatz lesen ... 914
16.1.47 SFC 64 - TIME_TCK - Systemzeit lesen ... 916
16.1.48 SFC 65 - X_SEND - Daten senden ... 917
16.1.49 SFC 66 - X_RCV - Daten empfangen ... 919
16.1.50 SFC 67 - X_GET - Daten lesen ... 923
16.1.51 SFC 68 - X_PUT - Daten schreiben ... 926
16.1.52 SFC 69 - X_ABORT - Verbindung abbrechen ... 929
16.1.53 SFC 70 - GEO_LOG - Anfangsadresse einer Baugruppe ermitteln ... 931
16.1.54 SFC 71 - LOG_GEO - zu logischer Adresse gehörenden Slot ermitteln ... 932
16.1.55 SFC 81 - UBLKMOV - Variable ununterbrechbar kopieren ... 935
16.1.56 SFC 101 - RTM - Hantierung Betriebsstundenzähler ... 936
16.1.57 SFC 102 - RD_DPARA - Vordefinierte Parameter lesen ... 937

16.1.58 SFC 105 - READ_SI - Auslesen dyn. Systemressourcen ... 938
16.1.59 SFC 106 - DEL_SI - Freigeben dyn. belegter Systemressourcen ... 941
16.1.60 SFC 107 - ALARM_DQ und SFC 108 - ALARM_D ... 942
16.2 System-SFBs - *"System Function Blocks"* ... 944
16.2.1 SFB 0 - CTU - Vorwärtszählen ... 944
16.2.2 SFB 1 - CTD - Rückwärtszählen ... 945
16.2.3 SFB 2 - CTUD - Vorwärts-/Rückwärtszählen ... 946
16.2.4 SFB 3 - TP - Impuls erzeugen ... 948
16.2.5 SFB 4 - TON - Einschaltverzögerung ... 949
16.2.6 SFB 5 - TOF - Ausschaltverzögerung ... 951
16.2.7 FB/SFB 12 - BSEND - Blockorientiertes Senden ... 952
16.2.8 FB/SFB 13 - BRCV - Blockorientiertes Empfangen ... 955
16.2.9 FB/SFB 14 - GET - Remote CPU lesen ... 957
16.2.10 FB/SFB 15 - PUT - Remote CPU schreiben ... 959
16.2.11 SFB 31 - NOTIFY_8P - Meldung ohne Quittierungsanzeige (8x) ... 961
16.2.12 SFB 32 - DRUM - Schrittschaltwerk ... 963
16.2.13 SFB 33 - ALARM - Meldungen mit Quittierungsanzeige ... 967
16.2.14 SFB 34 - ALARM_8 - Meldungen ohne Begleitwerte (8x) ... 970
16.2.15 SFB 35 - ALARM_8P - Meldungen mit Begleitwerten (8x) ... 971
16.2.16 SFB 36 - NOTIFY - Meldungen ohne Quittierungsanzeige ... 974
16.2.17 SFB 47 - COUNT - Zähler steuern ... 976
16.2.18 SFB 48 - FREQUENC - Frequenzmessung steuern ... 981
16.2.19 SFB 49 - PULSE - Pulsweitenmodulation ... 983
16.2.20 SFB 52 - RDREC - Datensatz lesen ... 991
16.2.21 SFB 53 - WRREC - Datensatz schreiben ... 992
16.2.22 SFB 54 - RALRM - Alarm von einer Peripheriebaugruppe empfangen. 993
17 Standard-Bausteine - *"Standard"* ... **1012**
17.1 Konvertierung - *"Converting"* ... 1012
17.1.1 FB 80 - LEAD_LAG - Lead/Lag Algorithmus ... 1012
17.1.2 FC 93 - SEG - Bitmuster für 7-Segment-Anzeige erzeugen ... 1013
17.1.3 FC 94 - ATH - ASCII-Zeichenkette in Hexadezimalzahl wandeln ... 1014
17.1.4 FC 95 - HTA - Hexadezimalzahl in ASCII-Zeichenkette wandeln ... 1015
17.1.5 FC 96 - ENCO - Bitnummer des niederwertigsten gesetzten Bits lesen 1015
17.1.6 FC 97 - DECO - Vorgegebenes Bit im Wort setzen ... 1016
17.1.7 FC 98 - BCDCPL - Zehnerkomplement erzeugen ... 1016
17.1.8 FC 99 - BITSUM - Anzahl der gesetzten Bits zählen ... 1017
17.1.9 FC 105 - SCALE - Werte skalieren ... 1017
17.1.10 FC 106 - UNSCALE - Werte deskalieren ... 1018
17.1.11 FC 108 - RLG_AA1 - Analogwert ausgeben ... 1019
17.1.12 FC 109 - RLG_AA2 - Write Analog Value 2 ... 1020
17.1.13 FC 110 - PER_ET1 - Read/Write Ext. Per. 1 ... 1021
17.1.14 FC 111 - PER_ET2 - Read/Write Ext. Per. 2 ... 1021
17.2 IEC-Funktionen - *"IEC"* ... 1022
17.2.1 Datum und Uhrzeit als zusammengesetzte Datentypen ... 1022
17.2.2 FC 1 - AD_DT_TM - Zeitdauer auf einen Zeitpunkt addieren ... 1023
17.2.3 FC 2 - CONCAT - Zusammenfassen zweier STRING-Variablen ... 1023
17.2.4 FC 3 - D_TOD_DT - Zusammenfassen DATE und TIME_OF_DAY ... 1023
17.2.5 FC 4 - DELETE - Löschen in einer STRING-Variable ... 1024
17.2.6 FC 5 - DI_STRNG - Formatwandlung DINT nach STRING ... 1024

17.2.7 FC 6 - DT_DATE - Extrahieren DATE aus DT ... 1025
17.2.8 FC 7 - DT_DAY - Extrahieren des Wochentags aus DT ... 1025
17.2.9 FC 8 - DT_TOD - Extrahieren TIME_OF_DAY aus DT ... 1025
17.2.10 FC 9 - EQ_DT - Vergleich DT auf gleich ... 1026
17.2.11 FC 10 - EQ_STRNG - Vergleich STRING auf gleich ... 1026
17.2.12 FC 11 - FIND - Suchen in einer STRING-Variable ... 1026
17.2.13 FC 12 - GE_DT - Vergleich DT auf größer oder gleich ... 1027
17.2.14 FC 13 - GE_STRNG - Vergleich STRING auf größer oder gleich ... 1027
17.2.15 FC 14 - GT_DT - Vergleich DT auf größer ... 1028
17.2.16 FC 15 - GT_STRNG - Vergleich STRING auf größer ... 1028
17.2.17 FC 16 - I_STRNG - Formatwandlung INT nach STRING ... 1028
17.2.18 FC 17 - INSERT - Einfügen in eine STRING-Variable ... 1029
17.2.19 FC 18 - LE_DT - Vergleich DT auf kleiner oder gleich ... 1029
17.2.20 FC 19 - LE_STRNG - Vergleich STRING auf kleiner oder gleich ... 1030
17.2.21 FC 20 - LEFT - Linker Teil einer STRING-Variable ... 1030
17.2.22 FC 21 - LEN - Länge einer STRING-Variablen ... 1030
17.2.23 FC 22 - LIMIT - Begrenzer ... 1031
17.2.24 FC 23 - LT_DT - Vergleich DT auf kleiner ... 1031
17.2.25 FC 24 - LT_STRNG - Vergleich STRING auf kleiner ... 1032
17.2.26 FC 25 - MAX - Maximumauswahl ... 1032
17.2.27 FC 26 - MID - Mittlerer Teil einer STRING-Variablen ... 1033
17.2.28 FC 27 - MIN - Minimumauswahl ... 1033
17.2.29 FC 28 - NE_DT - Vergleich DT auf ungleich ... 1034
17.2.30 FC 29 - NE_STRNG - Vergleich STRING auf ungleich ... 1035
17.2.31 FC 30 - R_STRNG - Formatwandlung REAL nach STRING ... 1035
17.2.32 FC 31 - REPLACE - Ersetzen in einer STRING-Variable ... 1035
17.2.33 FC 32 - RIGHT - Rechter Teil einer STRING-Variable ... 1036
17.2.34 FC 33 - S5TI_TIM - Formatwandlung S5TIME nach TIME ... 1036
17.2.35 FC 34 - SB_DT_DT - Zwei Zeitpunkte subtrahieren ... 1037
17.2.36 FC 35 - SB_DT_TM - Zeitdauer von einem Zeitpunkt subtrahieren ... 1037
17.2.37 FC 36 - SEL - Binärauswahl ... 1038
17.2.38 FC 37 - STRNG_DI - Formatwandlung STRING nach DINT ... 1038
17.2.39 FC 38 - STRNG_I - Formatwandlung STRING nach INT ... 1039
17.2.40 FC 39 - STRNG_R - Formatwandlung STRING nach REAL ... 1039
17.2.41 FC 40 - TIM_S5TI - Formatwandlung TIME nach S5TIME ... 1039
17.3 Ein-/Ausgabe - *"IO"* ... 1040
17.3.1 FB 20 - GETIO - PROFIBUS/PROFINET alle Eingänge lesen ... 1040
17.3.2 FB 21 - SETIO - PROFIBUS/PROFINET alle Ausgänge schreiben ... 1040
17.3.3 FB 22 - GETIO_PART - PROFIBUS/PROFINET Teil-Eingänge lesen ... 1041
17.3.4 FB 23 - SETIO_PART - PROFIBUS/PROFINET Teil-Ausgänge schreiben ... 1042
17.4 S5-Konvertierung - *"S5 Converting"* ... 1044
17.4.1 FC 112 - Sine(x) - Sinus ... 1044
17.4.2 FC 113 - Cosine(x) - Cosinus ... 1045
17.4.3 FC 114 - Tangent(x) - Tangens ... 1045
17.4.4 FC 115 - Cotangent(x) - Cotangens ... 1046
17.4.5 FC 116 - Arc Sine(x) - Arcussinus ... 1047
17.4.6 FC 117 - Arc Cosine(x) - Arcuscosinus ... 1048
17.4.7 FC 118 - Arc Tangent(x) - Arcustangens ... 1048
17.4.8 FC 119 - Arc Cotangent(x) - Arcuscotangens ... 1049

17.4.9 FC 120 - Naperian Logarithm ln(x) - Natürlicher Logarithmus 1050
17.4.10 FC 121 - Decimal Logarithm lg(x) - Dezimaler Logarithmus 1051
17.4.11 FC 122 - Gen. Logarithm to Base b - Allgemeiner Logarithmus log (x) zur Basis b 1051
17.4.12 FC 123 - E to Power n - E hoch n 1052
17.4.13 FC 124 - 10 to Power n - 10 hoch n 1053
17.4.14 FC 125 - ACCU 2 to Power ACCU 1 - AKKU 2 hoch AKKU 1 1053
17.5 PID-Steuerung - *"PID Control"* 1054
17.5.1 FB 41 - CONT_C - Kontinuierliches Regeln 1054
17.5.2 FB 42 - CONT_S - Schrittregeln 1061
17.5.3 FB 43 - PULSGEN - Impulsformen 1066
17.5.4 FB 58 - TCONT_CP - Kontinuierliches Temperaturregeln 1075
17.5.5 FB 59 - TCONT_S - Temperatur-Schrittregeln 1093
17.6 Zeitfunktionen - *"Time Functions"* 1100
17.6.1 UDT 60 - WS_RULES - Regel DB 1100
17.6.2 FC 61 - BT_LT - Umrechnung Basiszeit in Lokalzeit 1101
17.6.3 FC 62 - LT_BT - Umrechnung Lokalzeit in Basiszeit 1102
17.6.4 FC 63 - S_LTINT - Einstellen Uhrzeitalarm in Lokalzeit 1103
18 Systembausteine - *"System Blocks"* 1106
18.1 Fetch/Write - *"Fetch/Write Communication"* 1106
18.1.1 SFC 228 - RW_KACHEL - Kacheldirektzugriff 1106
18.1.2 SFC 230 ... 238 - Kachelkommunikation 1108
18.1.3 SFC 230 - SEND - Senden an Kachel 1120
18.1.4 SFC 231 - RECEIVE - Empfangen von Kachel 1121
18.1.5 SFC 232 - FETCH - Anfordern von Kachel 1122
18.1.6 SFC 233 - CONTROL - Control Kachel 1123
18.1.7 SFC 234 - RESET - Rücksetzen Kachel 1124
18.1.8 SFC 235 - SYNCHRON - Synchronisieren Kachel 1125
18.1.9 SFC 236 - SEND_ALL - Alles senden an Kachel 1126
18.1.10 SFC 237 - RECEIVE_ALL - Alles empfangen von Kachel 1127
18.1.11 SFC 238 - CTRL1 - Control1 Kachel 1128
18.2 Datei-Funktionen SPEED7-CPUs - *"File Functions SPEED7 CPUs"* 1128
18.2.1 FC/SFC 195 und FC/SFC 208...215 - Speicherkarten-Zugriff 1128
18.2.2 FC/SFC 195 - FILE_ATT - Datei-Attribute ändern 1129
18.2.3 FC/SFC 208 - FILE_OPN - Datei öffnen 1131
18.2.4 FC/SFC 209 - FILE_CRE - Datei anlegen 1132
18.2.5 FC/SFC 210 - FILE_CLO - Datei schließen 1133
18.2.6 FC/SFC 211 - FILE_RD - Datei lesen 1134
18.2.7 FC/SFC 212 - FILE_WR - Datei schreiben 1135
18.2.8 FC/SFC 213 - FILE_SEK - Position Schreib-/Lesemarke 1136
18.2.9 FC/SFC 214 - FILE_REN - Datei umbenennen 1137
18.2.10 FC/SFC 215 - FILE_DEL - Datei löschen 1139
18.3 Datei-Funktionen Standard-CPUs - *"File Functions Standard CPUs"* 1141
18.3.1 SFC 220 ... 222 - MMC-Zugriff 1141
18.3.2 SFC 220 - MMC_CR_F - MMC-Datei erstellen oder öffnen 1141
18.3.3 SFC 221 - MMC_RD_F - MMC-Datei lesen 1143
18.3.4 SFC 222 - MMC_WR_F - MMC-Datei schreiben 1144
18.4 Systemfunktions-Blöcke - *"System Function Blocks"* 1146
18.4.1 FB/SFB 7 - TIMEMESS - Zeitmessung 1146
18.5 Systemfunktionen - *"System Functions"* 1146

18.5.1 FC/SFC 53 - uS_Tick - Zeitmessung 1146
18.5.2 SFC 75 - SET_ADDR - PROFIBUS MAC-Adresse setzen 1147
18.5.3 FC/SFC 193 - AI_OSZI - Oszilloskop-/FIFO-Funktion 1147
18.5.4 FC/SFC 194 - DP_EXCH - Datenaustausch mit CP342S 1151
18.5.5 FC/SFC 219 - CAN_TLGR - CANopen-Kommunikation 1152
18.5.6 FC/SFC 254 - RW_SBUS - IBS-Kommunikation 1155
19 Index 1156

Copyright © YASKAWA Europe GmbH

1 Zu dieser Dokumentation

1.1 Copyright © YASKAWA Europe GmbH

All Rights Reserved

Dieses Dokument enthält geschützte Informationen von YASKAWA und darf außer in Übereinstimmung mit anwendbaren Vereinbarungen weder offengelegt noch benutzt werden.

Dieses Material ist durch Urheberrechtsgesetze geschützt. Ohne schriftliches Einverständnis von YASKAWA und dem Besitzer dieses Materials darf dieses Material weder reproduziert, verteilt, noch in keiner Form von keiner Einheit (sowohl YASKAWA-intern als auch -extern) geändert werden, es sei denn in Übereinstimmung mit anwendbaren Vereinbarungen, Verträgen oder Lizenzen.

Zur Genehmigung von Vervielfältigung oder Verteilung wenden Sie sich bitte an: YASKAWA Europe GmbH, European Headquarters, Hauptstraße 185, 65760 Eschborn, Germany

Tel.: +49 6196 569 300

Fax.: +49 6196 569 398

E-Mail: info@yaskawa.eu.com

Internet: www.yaskawa.eu.com

Es wurden alle Anstrengungen unternommen, um sicherzustellen, dass die in diesem Dokument enthaltenen Informationen zum Zeitpunkt der Veröffentlichung vollständig und richtig sind. Das Recht auf Änderungen der Informationen bleibt jedoch vorbehalten.

Die vorliegende Kundendokumentation beschreibt alle heute bekannten Hardware-Einheiten und Funktionen. Es ist möglich, dass Einheiten beschrieben sind, die beim Kunden nicht vorhanden sind. Der genaue Lieferumfang ist im jeweiligen Kaufvertrag beschrieben.

EG-Konformitätserklärung

Hiermit erklärt YASKAWA Europe GmbH, dass die Produkte und Systeme mit den grundlegenden Anforderungen und den anderen relevanten Vorschriften übereinstimmen. Die Übereinstimmung ist durch CE-Zeichen gekennzeichnet.

Informationen zur Konformitätserklärung

Für weitere Informationen zur CE-Kennzeichnung und Konformitätserklärung wenden Sie sich bitte an Ihre Landesvertretung der YASKAWA Europe GmbH.

Warenzeichen

VIPA, SLIO, System 100V, System 200V, System 300V, System 300S, System 400V, System 500S und Commander Compact sind eingetragene Warenzeichen der YASKAWA Europe GmbH.

SPEED7 ist ein eingetragenes Warenzeichen der YASKAWA Europe GmbH.

SIMATIC, STEP, SINEC, TIA Portal, S7-300, S7-400 und S7-1500 sind eingetragene Warenzeichen der Siemens AG.

Microsoft und Windows sind eingetragene Warenzeichen von Microsoft Inc., USA.

Portable Document Format (PDF) und Postscript sind eingetragene Warenzeichen von Adobe Systems, Inc.

Alle anderen erwähnten Firmennamen und Logos sowie Marken- oder Produktnamen sind Warenzeichen oder eingetragene Warenzeichen ihrer jeweiligen Eigentümer.

Dokument-Support

Wenden Sie sich an Ihre Landesvertretung der YASKAWA Europe GmbH, wenn Sie Fehler anzeigen oder inhaltliche Fragen zu diesem Dokument stellen möchten. Ist eine solche Stelle nicht erreichbar, können Sie YASKAWA Europe GmbH über folgenden Kontakt erreichen:

YASKAWA Europe GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Fax: +49 9132 744 29 1204

E-Mail: Documentation.HER@yaskawa.eu.com

Technischer Support

Wenden Sie sich an Ihre Landesvertretung der YASKAWA Europe GmbH, wenn Sie Probleme mit dem Produkt haben oder Fragen zum Produkt stellen möchten. Ist eine solche Stelle nicht erreichbar, können Sie den YASKAWA Kundenservice über folgenden Kontakt erreichen:

YASKAWA Europe GmbH,
European Headquarters, Hauptstraße 185, 65760 Eschborn, Germany

Tel.: +49 6196 569 500 (Hotline)

E-Mail: support@yaskawa.eu.com

1.2 Zweck der Dokumentation

In dieser Dokumentation wird die Operationsliste zum Softwarepaket VIPA *SPEED7 Studio* beschrieben. ↳ *Kap. 1.4 "Aufbau und Inhalt der Dokumentation" Seite 20*

Das Handbuch richtet sich an Personen, die Steuerungsaufgaben mit VIPA *SPEED7 Studio* für VIPA SPEED7-Automatisierungssysteme realisieren.

1.3 Gültigkeit der Dokumentation

Diese Beschreibung ist gültig für das Softwarepaket VIPA *SPEED7 Studio* ab Version 1.9.0

Informationen zu neueren Versionen oder Servicepacks, die nach Veröffentlichung dieser Softwarebeschreibung erscheinen, finden Sie unter *www.vipa.com*

1.4 Aufbau und Inhalt der Dokumentation

Kapitel	Inhalt
↳ *Kap. 1 "Zu dieser Dokumentation" Seite 19* (dieses Kapitel)	Zu dieser Dokumentation ■ Informationen zu dieser und weiteren Dokumentationen ■ Darstellung von Hinweisen und Informationen ■ Bezeichnung der Oberflächenelemente
↳ *Kap. 2 "Wichtige Hinweise" Seite 24*	Wichtige Hinweise ■ Wichtige Hinweise, welche grundsätzlich beim Einsatz der Bausteine zu beachten sind. ■ Intern verwendete Bausteinen, welche nicht überschrieben werden dürfen.
↳ *Kap. 3 "AWL-Operationen" Seite 25*	Liste der AWL-Operationen ■ Die Befehle sind in Themengruppen zusammengefasst, die in alphabetisch sortierter Reihenfolge aufgelistet sind.

Kapitel	Inhalt
Kap. 4 "FUP-Operationen" Seite 75	Liste der FUP-Operationen ■ FUP (Funktionsplan) ist eine grafische Programmiersprache zur Signalverarbeitung. ■ Mit Funktionsplan können unterschiedliche Funktionselemente miteinander verknüpft werden, um den Signalfluss zu steuern.
Kap. 5 "KOP-Operationen" Seite 172	Liste der KOP-Operationen ■ KOP (Kontaktplan) ist eine grafische Programmiersprache zur Signalverarbeitung. ■ Die Darstellung ist vergleichbar mit einem Stromlaufplan.
Kap. 6 "Bausteinparameter" Seite 269	Bausteinparameter ■ Beschreibung des Fehlercodes RET_VAL
Kap. 7 "Organisationsbausteine - "Organization Blocks"" Seite 272	Liste der OBs ■ OBs (Organisationsbausteine) bilden die Schnittstelle zwischen Betriebssystem und Anwenderprogramm.
Kap. 8 "Gebäude-Automatisierung - "Building Control"" Seite 305	■ Bausteine für die Gebäudeleittechnik
Kap. 9 "Netzwerkkommunikation - "Network Communication"" Seite 318	Bausteine für die Netzwerkkommunikation ■ Offene Kommunikation (TCP, ISO on TCP, UDP) ■ Ethernet-Kommunikation (CP343, S7-Kommunikation)
Kap. 10 "Modbus-Kommunikation - "Modbus Communication"" Seite 380	Bausteine für die Modbus-Kommunikation ■ Modbus TCP/RTU
Kap. 11 "Serielle Kommunikation - "Serial Communication"" Seite 402	Bausteine für die serielle Kommunikation
Kap. 12 "EtherCAT-Kommunikation - "EtherCAT Communication"" Seite 434	Bausteine für die EtherCAT-Kommunikation
Kap. 13 "Modulspezifisch - "Device Specific"" Seite 442	Modulspezifische Bausteine ■ Frequenzmessung ■ Energiemessung ■ Motion-Module ■ RAM- und WLD-Zugriff
Kap. 14 "Antriebssteuerung - "Motion Control"" Seite 476	Bausteine zur Antriebssteuerung ■ Komplexer Bewegungsführung an Produktionsmaschinen ■ Einsatz von Kurvenscheiben ■ Einfachachsen ■ Mehrfachachsen
Kap. 15 "Antriebssteuerung - Simple Motion Control Library" Seite 577	Bausteine zur einfachen Antriebssteuerung (Simple motion control) ■ Einsatz Sigma-5/7 ■ Einsatz Frequenzumrichter ■ Antrieb über HMI steuern
Kap. 16 "Integrierte Standardfunktionen - "Integrated Standard"" Seite 859	Integrierte Standardfunktionen ■ System-SFCs ■ System-SFBs

Kapitel	Inhalt
Kap. 17 "Standard-Bausteine - "Standard"" Seite 1012	Standardbausteine ■ Konvertierung ■ IEC-Funktionen ■ Ein-/Ausgabe ■ PID-Steuerung ■ Zeitfunktionen
Kap. 18 "Systembausteine - "System Blocks"" Seite 1106	Systembausteine ■ FETCH/WRITE ■ Datei-Funktionen ■ Systemfunktionen

1.5 Darstellung und Kennzeichnungen

Hinweise, Tipps, Empfehlungen, Beispiele und Handlungsanweisungen werden in dieser Dokumentation wie folgt dargestellt:

Sicherheitshinweise

GEFAHR!
Dieses Symbol weist auf Gefahren hin, bei denen Tod oder schwere Körperverletzung eintreten wird, wenn die genannten Vorsichtsmaßnahmen nicht getroffen werden.

VORSICHT!
Dieses Symbol weist auf Gefahren hin, bei denen Tod oder schwere Körperverletzung eintreten kann, wenn die genannten Vorsichtsmaßnahmen nicht getroffen werden.

HINWEIS!
Dieses Symbol weist auf wichtige Informationen hin, bei deren Missachtung es zu Systemfehlern oder Datenverlust kommen kann.

Tipps und Empfehlungen

Dieses Symbol weist auf Informationen hin, die Ihnen den Umgang mit dem System erleichtern.

Beispiel

Hier werden Ihnen Beispiele für die Anwendung der zuvor erläuterten Handlungen oder Programmierbeispiele gezeigt.

Handlungsanweisungen

Diese Dokumentation enthält Handlungsanweisungen zu vielen Funktionen, die Sie Schritt für Schritt nachvollziehen können. Handlungsanweisungen enthalten folgende Elemente:

Im Handlungsschritt erfahren Sie, was Sie tun müssen. Falls die Handlungsanweisung aus mehreren Schritten besteht, so sind die einzelnen Schritte ihrer Reihenfolge nach nummeriert.

⇨ Hier wird das Ergebnis des Handlungsschrittes gezeigt.

Weitere Kennzeichnungen und Formatierungen

Zum Hervorheben bestimmter Informationen werden in dieser Dokumentation folgende Kennzeichnungen verwendet:

Kennzeichnung/Formatierung	Erläuterung
"Menü ➔ Menüpunkt"	Menübefehl Beispiel: *"Datei ➔ Projekt öffnen"*
"Text"	Text der Programmoberfläche, z. B. Editierfeld in einem Dialogfenster Beispiel: *"Benutzername"* - oder - Schaltfläche Beispiel: *"Abbrechen"*
[Taste]	Taste oder Tastenkombination der Computertastatur Beispiel: *[Strg]+[C]*
Produkt	Produktbezeichnung (kursive Schrift) Beispiel: *SPEED7 Studio*
`Programmcode`	Programmcode (Festbreitenschrift)
	Symbol/Schaltfläche der Programmoberfläche
①	Bezugsziffer in Abbildungen (fortlaufende Nummer im Kreis) Beispiel: (1) Symbolleiste
■	Liste, z. B. zur Auflistung mehrerer Bedienungsmöglichkeiten
	Querverweis auf weiterführende Informationen Beispiel: *Kap. 1 "Zu dieser Dokumentation" Seite 19*

2 Wichtige Hinweise

2.1 Allgemein

Nachfolgend finden Sie wichtige Hinweise, die grundsätzlich beim Einsatz der Bausteine zu beachten sind.

2.2 Intern verwendete Bausteine

VORSICHT!
Folgende Bausteine werden intern verwendet und dürfen nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB! Bitte verwenden Sie für den Aufruf immer die zugehörige Funktion.

FC/SFC	Bezeichnung	Beschreibung
FC/SFC 131	TSEND_	wird intern für FB 63 verwendet
FC/SFC 132	TRECV_	wird intern für FB 64 verwendet
FC/SFC 133	TCON_	wird intern für FB 65 verwendet
FC/SFC 134	TDISCON_	wird intern für FB 66 verwendet
FC/SFC 135	TUSEND_	wird intern für FB 67 verwendet
FC/SFC 136	TURECV_	wird intern für FB 68 verwendet
FC/SFC 192	CP_S_R	wird intern für FB 7 und FB 8 verwendet
FC/SFC 196	AG_CNTRL	wird intern für FC 10 verwendet
FC/SFC 198	USEND_	wird intern für FB 8 verwendet
FC/SFC 199	URCV_	wird intern für FB 9 verwendet
FC/SFC 200	AG_GET	wird intern für FB/SFB 14 verwendet
FC/SFC 201	AG_PUT	wird intern für FB/SFB 15 verwendet
FC/SFC 202	AG_BSEND	wird intern für FB/SFB 12 verwendet
FC/SFC 203	AG_BRCV	wird intern für FB/SFB 13 verwendet
FC/SFC 204	IP_CONF	wird intern für FB 55 IP_CONF verwendet
FC/SFC 205	AG_SEND	wird intern für FC 5 AG_SEND verwendet
FC/SFC 206	AG_RECV	wird intern für FC 6 AG_RECV verwendet
FC/SFC 253	IBS_ACCESS	wird intern für SPEED-Bus-INTERBUS-Master verwendet
SFB 238	EC_RWOD	wird intern für EtherCAT-Kommunikation verwendet
SFB 239	FUNC	wird intern für FB 240, FB 241 verwendet

3 AWL-Operationen

3.1 Übersicht

Das folgende Kapitel beinhaltet die Befehlsliste für die SPEED7 VIPA-CPUs. Die Befehlsliste soll Ihnen einen schnellen Überblick der Befehle und deren Syntax geben. Die Befehle sind in Themengruppen zusammengefasst, die in alphabetisch sortierter Reihenfolge aufgelistet sind. Da die Parameter in die Befehlsliste integriert sind, wurde auf eine gesonderte Parameterliste verzichtet.

Befehle	Beschreibung	Seite
)	Verknüpfungs-Operationen (Bit)	*64*
+	Arithmetische Operationen	*39*
+AR1	Arithmetische Operationen	*39*
+AR2	Arithmetische Operationen	*39*
+I	Arithmetische Operationen	*39*
+D	Arithmetische Operationen	*39*
+R	Arithmetische Operationen	*39*
-D	Arithmetische Operationen	*39*
-I	Arithmetische Operationen	*39*
-R	Arithmetische Operationen	*39*
*D	Arithmetische Operationen	*39*
*I	Arithmetische Operationen	*39*
*R	Arithmetische Operationen	*39*
/D	Arithmetische Operationen	*39*
/I	Arithmetische Operationen	*39*
/R	Arithmetische Operationen	*39*
==D	Vergleichs-Operationen	*62*
==I	Vergleichs-Operationen	*62*
==R	Vergleichs-Operationen	*62*
<=D	Vergleichs-Operationen	*62*
<=I	Vergleichs-Operationen	*62*
<=R	Vergleichs-Operationen	*62*
<D	Vergleichs-Operationen	*62*
<I	Vergleichs-Operationen	*62*
<R	Vergleichs-Operationen	*62*
<>D	Vergleichs-Operationen	*62*
<>I	Vergleichs-Operationen	*62*
<>R	Vergleichs-Operationen	*62*
>=D	Vergleichs-Operationen	*62*
>=I	Vergleichs-Operationen	*62*

Übersicht

Befehle	Beschreibung	Seite
>=R	Vergleichs-Operationen	62
>D	Vergleichs-Operationen	62
>I	Vergleichs-Operationen	62
>R	Vergleichs-Operationen	62
ABS	Arithmetische Operationen	39
ACOS	Arithmetische Operationen	39
ASIN	Arithmetische Operationen	39
ATAN	Arithmetische Operationen	39
AUF	Baustein-Operationen	44
BE	Baustein-Operationen	44
BEA	Baustein-Operationen	44
BEB	Baustein-Operationen	44
BLD	Baustein-Operationen	44
BTD	Umwandlungs-Operationen	60
BTI	Umwandlungs-Operationen	60
CALL	Baustein-Operationen	44
CC	Baustein-Operationen	44
CLR	Setzen/Rücksetzen von Bitoperanden	52
COS	Arithmetische Operationen	39
DEC	Transfer-Operationen	55
DTB	Umwandlungs-Operationen	60
DTR	Umwandlungs-Operationen	60
EXP	Arithmetische Operationen	39
FN	Flanken-Operationen	46
FP	Flanken-Operationen	46
FR	Zähl-Operationen Zeit-Operationen	74 73
INC	Transfer-Operationen	55
INVD	Umwandlungs-Operationen	60
INVI	Umwandlungs-Operationen	60
ITB	Umwandlungs-Operationen	60
ITD	Umwandlungs-Operationen	60
L	Lade-Operationen	47
LAR1	Transfer-Operationen	55
LAR2	Transfer-Operationen	55
LC	Lade-Operationen	47

Befehle	Beschreibung	Seite
LN	Arithmetische Operationen	39
LOOP	Sprung-Operationen	53
MOD	Arithmetische Operationen	39
NEGD	Umwandlungs-Operationen	60
NEGI	Umwandlungs-Operationen	60
NEGR	Arithmetische Operationen	39
NOP	Baustein-Operationen	44
NOT	Setzen/Rücksetzen von Bitoperanden	52
O	Verknüpfungs-Operationen (Bit)	64
O(	Verknüpfungs-Operationen (Bit)	64
OD	Verknüpfungs-Operationen (Wort)	72
ON	Verknüpfungs-Operationen (Bit)	64
ON(	Verknüpfungs-Operationen (Bit)	64
OW	Verknüpfungs-Operationen (Wort)	72
POP	Transfer-Operationen	55
PUSH	Transfer-Operationen	55
R	Setzen/Rücksetzen von Bitoperanden	52
RLD	Schiebe-Operationen	50
RLDA	Schiebe-Operationen	50
RND	Umwandlungs-Operationen	60
RND+	Umwandlungs-Operationen	60
RND-	Umwandlungs-Operationen	60
RRD	Schiebe-Operationen	50
RRDA	Schiebe-Operationen	50
S	Setzen/Rücksetzen von Bitoperanden	52
SA	Zeit-Operationen	73
SAVE	Setzen/Rücksetzen von Bitoperanden	52
SE	Zeit-Operationen	73
SET	Setzen/Rücksetzen von Bitoperanden	52
SI	Zeit-Operationen	73
SIN	Arithmetische Operationen	39
SLD	Schiebe-Operationen	50
SLW	Schiebe-Operationen	50
SPA	Sprung-Operationen	53
SPB	Sprung-Operationen	53
SPBB	Sprung-Operationen	53

Übersicht

Befehle	Beschreibung	Seite
SPBI	Sprung-Operationen	53
SPBIN	Sprung-Operationen	53
SPBN	Sprung-Operationen	53
SPBNB	Sprung-Operationen	53
SPL	Sprung-Operationen	53
SPM	Sprung-Operationen	53
SPMZ	Sprung-Operationen	53
SPN	Sprung-Operationen	53
SPO	Sprung-Operationen	53
SPP	Sprung-Operationen	53
SPPZ	Sprung-Operationen	53
SPS	Sprung-Operationen	53
SPU	Sprung-Operationen	53
SPZ	Sprung-Operationen	53
SQR	Arithmetische Operationen	39
SQRT	Arithmetische Operationen	39
SRD	Schiebe-Operationen	50
SRW	Schiebe-Operationen	50
SS	Zeit-Operationen	73
SSD	Schiebe-Operationen	50
SSI	Schiebe-Operationen	50
SV	Zeit-Operationen	73
T	Transfer-Operationen	55
TAD	Transfer-Operationen	55
TAK	Transfer-Operationen	55
TAN	Arithmetische Operationen	39
TAR	Transfer-Operationen	55
TAR1	Transfer-Operationen	55
TAR2	Transfer-Operationen	55
TAW	Transfer-Operationen	55
TDB	Baustein-Operationen	44
TRUNC	Umwandlungs-Operationen	60
U	Verknüpfungs-Operationen (Bit)	64
U(	Verknüpfungs-Operationen (Bit)	64
UC	Baustein-Operationen	44
UD	Verknüpfungs-Operationen (Wort)	72

Befehle	Beschreibung	Seite
UN	Verknüpfungs-Operationen (Bit)	*64*
UN(	Verknüpfungs-Operationen (Bit)	*64*
UW	Verknüpfungs-Operationen (Wort)	*72*
X	Verknüpfungs-Operationen (Bit)	*64*
X(	Verknüpfungs-Operationen (Bit)	*64*
XN	Verknüpfungs-Operationen (Bit)	*64*
XN(	Verknüpfungs-Operationen (Bit)	*64*
XOD	Verknüpfungs-Operationen (Wort)	*72*
XOW	Verknüpfungs-Operationen (Wort)	*72*
ZR	Zähl-Operationen	*74*
ZV	Zähl-Operationen	*74*

3.2 Abkürzungen

Abkürzung	Erläuterung
2#	Binärkonstante
a	Byteadresse
A	Ausgang (im PAA)
A0	Ergebnisanzeige
A1	Ergebnisanzeige
AB	Ausgangsbyte (im PAA)
AD	Ausgangsdoppelwort (im PAA)
AKKU	Register für die Verarbeitung von Bytes, Worten und Doppelworten.
AR	Adressregister, enthalten die bereichsinternen oder bereichsübergreifenden Adressen für die registerindirekt adressierten Operationen
AW	Ausgangswort (im PAA)
b	Bitadresse
B	bereichsübergreifend, registerindirekt adressiertes Byte
B (b1,b2)	Konstante, 2Byte
B (b1,b2,b3,b4)	Konstante, 4Byte
BIE	Binärergebnis
B#16#	Byte hexadezimal
c	Operandenbereich
C#	Zählerkonstante (BCD-codiert)
D	bereichsübergreifend, registerindirekt adressiertes Doppelwort

Abkürzung	Erläuterung
D#	IEC Datumskonstante
DB	Datenbaustein
DBB	Datenbyte im Datenbaustein
DBD	Datendoppelwort im Datenbaustein
DBW	Datenwort im Datenbaustein
DBX	Datenbit im Datenbaustein
DI	Instanz-Datenbaustein
DIB	Datenbyte im Instanz-DB
DID	Datendoppelwort im Instanz-DB
DIW	Datenwort im Instanz-DB
DIX	Datenbit im Instanz-DB
DW#16#	Doppelwort hexadezimal
E	Eingang (im PAE)
EB	Eingangsbyte (im PAE)
ED	Eingangsdoppelwort (im PAE)
/ER	Erstabfrage
EW	Eingangswort (im PAE)
f	Timer-/Zähler-Nr.
FB	Funktionsbaustein
FC	Funktionen
g	Operandenbereich
h	Operandenbereich
i	Operandenbereich
i8	Ganzzahl (8Bit)
i16	Ganzzahl (16Bit)
i32	Ganzzahl (32Bit)
k8	Konstante (8Bit)
k16	Konstante (16Bit)
k32	Konstante (32Bit)
L	Lokaldaten
L#	Ganzzahlkonstante (32Bit)
LB	Lokaldatenbyte
LD	Lokaldaten-Doppelwort
LW	Lokaldatenwort
m	Pointer-Konstante
M	Merker

Abkürzung	Erläuterung
MARKE	Symbolische Sprungadresse mit max. 4 Zeichen. Diese 4 Zeichen können sich aus Buchstaben, Zahlen und dem Unterstrich "_" zusammensetzen, wobei das 1. Zeichen ein Buchstabe sein muss. Groß- und Kleinschreibung werden unterschieden. Die Sprungmarke endet mit ":".
MB	Merkerbyte
MD	Merkerdoppelwort
MW	Merkerwort
n	Binärkonstante
OB	Operationsbaustein
OR	Oder
OS	Überlaufspeichernd
OV	Überlauf
p	Hexadezimalkonstante
P#	Pointerkonstante
PAA	Prozessabbild der Ausgänge
PAB	Peripherieausgangsbyte (direkter Peripheriezugriff)
PAD	Peripherieausgangsdoppelwort (direkter Peripheriezugriff)
PAE	Prozessabbild der Eingänge
PAW	Peripherieausgangswort (direkter Peripheriezugriff)
PEB	Peripherieeingangsbyte (direkter Peripheriezugriff)
PED	Peripherieeingangsdoppelwort (direkter Peripheriezugriff)
PEW	Peripherieeingangswort (direkter Peripheriezugriff)
q	Realzahl (32-Gleitpunktzahl)
r	Baustein-Nr.
S5T#	S5-Zeitkonstante (16Bit), dient zum Laden der S5-Timer
SFB	Systemfunktionsbausteine
SFC	Systemfunktionen
STA	Status
T	Timer (Zeiten)
T#	Zeitkonstante (16/32Bit)
TOD#	IEC-Zeitkonstante
VKE	Verknüpfungsergebnis
W	bereichsübergreifend, registerindirekt adressiertes Wort
W#16#	Wort hexadezimal
Z	Zähler

3.3 Gegenüberstellung der Syntaxsprachen

Gegenüberstellung

In der folgenden tabellarischen Übersicht werden die deutschen und internationalen Sprachelemente von AWL gegenübergestellt.

Bereich	deutsch	international
Eingang	E	I
Ausgang	A	Q
Zähler	Z	C
Peripherie-Eingangsbyte	PEB	PIB
Peripherie-Eingangswort	PEW	PIW
Peripherie-Eingangsdoppelwort	PED	PID
Peripherie-Ausgangsbyte	PAB	PQB
Peripherie-Ausgangswort	PAW	PQW
Peripherie-Ausgangsdoppelwort	PAD	PQD
Verknüpfungen	U	A
	UN	AN
	U(	A(
	UN(	AN(
	UW	AW
	UD	AD
Zeitfunktionen	SI	SP
	SV	SE
	SE	SD
	SA	SF
Zählerfunktionen	ZV	CU
	ZR	CD
Laden und Transferieren	TAR	CAR
	TAW	CAW
	TAD	CAD
Programmsteuerung	AUF	OPN
	BEA	BEU
	BEB	BEC
	TDB	CDB
	UW	AW
	UD	AD
Sprungfunktionen	SPA	JU
	SPBB	JCB
	SPBIN	JNBI

Bereich	deutsch	international
	SPBNB	JNB
	SPBI	JBI
	SPBN	JCN
	SPB	JC
	SPO	JO
	SPS	JOS
	SPU	JUO
	SPZ	JZ
	SPN	JN
	SPMZ	JMZ
	SPPZ	JPZ
	SPL	JL
	SPM	JM
	SPP	JP

3.4 Unterschiede zwischen SPEED7 und 300V Programmierung

Allgemeines

Die SPEED7-CPUs lehnen sich im Befehlsablauf an die Siemens S7-400 an und unterscheiden sich hierbei zur Siemens S7-300 (VIPA 300V).

Die Unterschiede finden Sie nachfolgend aufgeführt.

Im Folgenden wird eine CPU 318 von Siemens der Siemens S7-400 Serie angerechnet.

Statusregister

Im Gegensatz zur Siemens S7-300 werden von SPEED7 VIPA-CPUs und Siemens S7-400 (CPU 318) die Statusregister-Bits OR, STA, /ER benutzt.

Falls sich das Anwenderprogramm darauf stützt, dass im Statusregister die o.g. Bits immer Null sind (wie Siemens S7-300), so ist das Programm auf SPEED7 VIPA-CPUs und Siemens S7-400 (CPU 318) nicht lauffähig.

AKKU-Handling bei arithmetischen Operationen

Die CPUs der Siemens S7-300 besitzen 2 AKKUs. Bei einer arithmetischen Operation wird der Inhalt des 2. AKKUs nicht verändert.

Die SPEED7-CPUs hingegen besitzt 4 AKKUs. Nach einer arithmetischen Operation (+I, -I, *I, /I, +D, -D, *D, /D, MOD, +R, -R, *R, /R) wird der Inhalt des AKKUs 3 und 4 in die AKKUs 2 und 3 geladen.

Dies kann bei Programmen, die einen unveränderten AKKU2 voraussetzen, zu Konflikten führen.

VKE bei Sprüngen

Durch die fehlende Implementierung des Erstbefehlsbits /ER in der Siemens S7-300, kann es unter bestimmten Bedingungen zu Abweichungen bei der Befehlsausführung von Bit-Befehlen zwischen Siemens S7-300 und SPEED7 VIPA-CPUs bzw. Siemens S7-400 kommen, insbesondere bei einem Sprung in eine Bit-Verknüpfungskette.

Beispiele VKE bei Sprüngen

Beispiel A:

```
U E0.0
U M1.1
= M2.0 // VKE =1 Befehlsende
SPA =SP001 // springt
.....
U M7.6
U M3.0
U M3.1
→SP001:
U A2.2 // nach dem Sprung ...
// Siemens S7-300 verknüpft weiter
// SPEED7 VIPA-CPUs, Siemens S7-400 und CPU 318
// verwenden diesen Befehl als Erstabfrage
```

Beispiel B:

```
U E0.0
U M1.1
= M2.0 // VKE =1 Befehlsende
U A3.3 // Erstabfrage
SPA =SP001 // springt
.....
U M3.0
U M3.1
→SP001:
U M3.2 // nach Sprung ...
..... // verknüpfen alle CPUs weiter
```

BCD-Konsistenz

Beim Setzen eines Timers oder Zählers muss in AKKU1 ein gültiger BCD-Wert vorhanden sein. Die Überprüfung dieses BCD-Wertes erfolgt bei S7- 300 von Siemens nur, wenn der Timer oder Zähler übernommen wird (Flankenwechsel). In den SPEED7-CPUs erfolgt (wie bei S7-400 von Siemens) immer eine Überprüfung.

Beispiel:

```
......
U E5.4
L MW20
S T30
// Siemens S7-300 prüft nur falls
// Timer aktiv abgearbeitet wird
// SPEED7, Siemens S7-400 und CPU 318
// prüfen hier immer (auch wenn
// die Kondition nicht gegeben ist)
......
```

3.5 Register

AKKU1 ... AKKU4 (32Bit)

Die AKKUs sind Register für die Verarbeitung von Bytes, Worten oder Doppelworten. Hierzu werden die Operanden in die AKKUs geladen und dort verknüpft. Das Ergebnis der Operation steht immer im AKKU1.

AKKU	Bit
AKKUx (x=1 ... 4)	Bit 0 ... Bit 31
AKKUx-L	Bit 0 ... Bit 15
AKKUx-H	Bit 16 ... Bit 31
AKKUx-LL	Bit 0 ... Bit 7
AKKUx-LH	Bit 8 ... Bit 15
AKKUx-HL	Bit 16 ... Bit 23
AKKUx-HH	Bit 24 ... Bit 31

Adressregister AR1 und AR2 (32Bit)

Die Adressregister enthalten die bereichsinternen oder bereichsübergreifenden Adressen für die registerindirekt adressierenden Operationen. Die Adressregister sind 32Bit breit.

Die bereichsinternen bzw. bereichsübergreifenden Adressen haben folgenden Aufbau:

bereichsinterne Adresse:

00000000 00000bbb bbbbbbbb bbbbbxxx

bereichsübergreifende Adresse:

10000yyy 00000bbb bbbbbbbb bbbbbxxx

Legende:	b	Byteadresse
	x	Bitnummer
	Y	Bereichskennung *Kap. 3.6 "Adressierungsbeispiele" Seite 36*

Statuswort (16Bit)

Die Anzeigen werden durch die Operationen ausgewertet oder gesetzt. Das Statuswort ist 16Bit breit.

Bit	Belegung	Bedeutung
0	/ER	Erstabfrage
1	VKE	Verknüpfungsergebnis
2	STA	Status
3	OR	Oder
4	OS	Überlauf speichernd
5	OV	Überlauf
6	A0	Ergebnisanzeige
7	A1	Ergebnisanzeige
8	BIE	Binärergebnis
9 ... 15	nicht belegt	-

3.6 Adressierungsbeispiele

Adressierungsbeispiel	Beschreibung
Unmittelbare Adressierung	
L +27	Lade 16-Bit-Ganzzahlkonstante "27" in AKKU1
L L#-1	Lade 32-Bit-Ganzzahlkonstante "-1" in AKKU1
L 2#1010101010101010	Lade Binärkonstante in AKKU1
L DW#16#A0F0_BCFD	Lade Hexadezimalkonstante in AKKU1
L "Ende"	Lade ASCII-Zeichen in AKKU1
L T#500ms	Lade Zeitwert in AKKU1
L C#100	Lade Zählerwert in AKKU1
L B#(100,12)	Lade Konstante als 2Byte
L B#(100,12,50,8)	Lade Konstante als 4Byte
L P#10.0	Lade bereichsinterne Pointer in AKKU1
L P#E20.6	Lade bereichsübergreifende Pointer in AKKU1
L -2.5	Lade Realzahl in AKKU1
L D#1995-01-20	Lade Datum
L TOD#13:20:33.125	Lade Uhrzeit
Direkte Adressierung	
U E 0.0	UND-Verknüpfung des Eingangsbits 0.0
L EB 1	Lade Eingangsbyte 1 in AKKU1

Adressierungsbeispiel	Beschreibung
L EW 0	Lade Eingangswort 0 in AKKU1
L ED 0	Lade Eingangsdoppelwort 0 in AKKU1
Indirekte Adressierung Timer/Zähler	
SI T [LW 8]	Starte Timer; die Timer-Nr. steht im Lokaldatenwort 8
ZV Z [LW 10]	Starte Zähler; die Zähler-Nr. steht im Lokaldatenwort 10
Speicherindirekte, bereichsinterne Adressierung	
U E [LD 12] Bsp.: LP#22.2 T LD 12 U E [LD 12]	UND-Operation; die Adresse des Eingangs steht als Pointer im Lokaldaten-Doppelwort 12
U E [DBD 1]	UND-Operation; die Adresse des Eingangs steht als Pointer im Datendoppelwort 1 des DB
U A [DID 12]	UND-Operation; die Adresse des Ausgangs steht als Pointer im Datendoppelwort 12 des Instanz-DB
U A [MD 12]	UND-Operation; die Adresse des Ausgangs steht als Pointer im Merkerdoppelwort 12
Registerindirekte, bereichsinterne Adressierung	
U E [AR1,P#12.2]	UND-Operation; die Adresse des Eingangs errechnet sich zu "Pointerwert im Adressregister 1 + Pointer P#12.2"
Registerindirekte, bereichsübergreifende Adressierung	
Für die bereichsübergreifende, registerindirekte Adressierung muss die Adresse zusätzlich eine Bereichskennung in den Bits 24-26 enthalten. Die Adresse steht im Adressregister.	

Bereichskennung	Codierung binär	hex.	Bereich
P	1000 0**000**	80	Peripheriebereich
E	1000 0**001**	81	Eingangsbereich
A	1000 0**010**	82	Ausgangsbereich
M	1000 0**011**	83	Merkerbereich
DB	1000 0**100**	84	Datenbereich
DI	1000 0**101**	85	Instanz-Datenbereich
L	1000 0**110**	86	Lokaldatenbereich
VL	1000 0**111**	87	Vorgänger-Lokaldatenbereich (Zugriff auf Lokaldaten des aufrufenden Bausteins)

Adressierungsbeispiel	Beschreibung
L B [AR1,P#8.0]	Lade Byte in AKKU1; die Adresse errechnet sich aus "Pointerwert im Adressregister 1 + Pointer P#8.0"
U [AR1,P#32.3]	UND-Operation; die Adresse des Operanden errechnet sich aus "Pointerwert im Adressregister 1 + Pointer P#32.3"
Adressierung über Parameter	
U Parameter	Der Operand wird über den Parameter adressiert.

Beispiel zur Pointerberechnung

Beispiel bei Summe der Bitadressen ≤ 7:

```
LAR1 P#8.2
U E [AR1,P#10.2]
```

Ergebnis: Adressiert wird Eingang 18.4

(durch jeweilige Addition der Byte- und Bitadressen).

Beispiel bei Summe der Bitadressen > 7:

`L MD 0` beliebig berechneter Pointer, z.B. P#10.5

```
LAR1
U E [AR1,P#10.7]
```

Ergebnis: Adressiert wird Eingang 21.4

(durch Addition der Byte und Bitadressen mit Übertrag).

3.7 Arithmetische Operationen

Festpunktarithmetik (16Bit)

Arithmetische Operationen zweier 16-Bit-Zahlen.
Das Ergebnis steht im AKKU1 bzw. AKKU1-L.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
+I	-		Addiere zwei Ganzzahlen (16Bit) (AKKU1-L)=(AKKU1-L)+(AKKU2-L)	1
-I	-		Subtrahiere zwei Ganzzahlen (16Bit) (AKKU1-L)=(AKKU2-L)-(AKKU1-L)	1
*I	-		Multipliziere zwei Ganzzahlen (16Bit) (AKKU1)=(AKKU2-L)*(AKKU1-L)	1
/I	-		Dividiere zwei Ganzzahlen (16Bit) (AKKU1-L)=(AKKU2-L):(AKKU1-L) Im AKKU1-H steht der Rest der Division	1

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	✓	✓	✓	✓	-	-	-	-

Festpunktarithmetik (32Bit)

Arithmetische Operationen zweier 32-Bit-Zahlen.

Das Ergebnis steht im AKKU1.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
+D	-		Addiere zwei Ganzzahlen (32Bit) (AKKU1)=(AKKU2)+(AKKU1)	1
-D	-		Subtrahiere zwei Ganzzahlen (32Bit) (AKKU1)=(AKKU2)-(AKKU1)	1
*D	-		Multipliziere zwei Ganzzahlen (32Bit) (AKKU1)=(AKKU2)*(AKKU1)	1
/D	-		Dividiere zwei Ganzzahlen (32Bit) (AKKU1)=(AKKU2):(AKKU1)	1
MOD	-		Dividiere zwei Ganzzahlen (32Bit) und lade den Rest der Division in AKKU1 (AKKU1)=Rest von [(AKKU2):(AKKU1)]	1

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	✓	✓	✓	✓	-	-	-	-

Gleitpunktarithmetik (32Bit)

Das Ergebnis der arithmetische Operationen steht im AKKU1. Die Ausführungszeit der Operation hängt vom Wert ab, der berechnet werden soll.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
+R	-		Addiere zwei Realzahlen (32Bit) (AKKU1)=(AKKU2)+(AKKU1)	1
-R	-		Subtrahiere zwei Realzahlen (32Bit) (AKKU1)=(AKKU2)-(AKKU1)	1
*R	-		Multipliziere zwei Realzahlen (32Bit) (AKKU1)=(AKKU2)*(AKKU1)	1
/R	-		Dividiere zwei Realzahlen (32Bit) (AKKU1)=(AKKU2):(AKKU1)	1
NEGR	-		Negiere Realzahl im AKKU1	1
ABS	-		Bilde Betrag der Realzahl im AKKU1	1

Statuswort für: R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	✓	✓	✓	✓	-	-	-	-

Statuswort für: NEGR, ABS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	-	-	-	-	-	-	-	-

Quadratwurzel, Quadrat (32Bit)

Das Ergebnis der Operationen steht im AKKU1.

Operationen sind durch Alarme unterbrechbar.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
SQRT	-		Berechne die Quadratwurzel einer Realzahl in AKKU1	1
SQR	-		Quadriere die Restzahl in AKKU1	1

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	✓	✓	✓	✓	-	-	-	-

Logarithmusfunktion (32Bit)

Das Ergebnis der Logarithmusfunktion steht im AKKU1.

Die Operationen sind durch Alarme unterbrechbar.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
LN	-		Bilde den natürlichen Logarithmus einer Realzahl in AKKU1	1
EXP	-		Berechne den Exponentialwert einer Realzahl in AKKU1 zur Basis e (=2,71828)	1

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	✓	✓	✓	✓	-	-	-	-

Trigonometrische Funktionen (32Bit)

Das Ergebnis der Operationen steht im AKKU1.

Operationen sind durch Alarme unterbrechbar.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
SIN[1]	-		Berechne den Sinus der Realzahl	1
ASIN[2]	-		Berechne den Arcussinus der Realzahl	1
COS[1]	-		Berechne den Cosinus der Realzahl	1
ACOS[2]	-		Berechne den Arcuscosinus der Realzahl	1
TAN[1]	-		Berechne den Tangens der Realzahl	1
ATAN[2]	-		Berechne den Arcustangens der Realzahl	1

1) Geben Sie den Winkel im Bogenmaß an. Der Winkel muss als Gleitpunktzahlenwert in AKKU1 angegeben werden.

2) Das Ergebnis ist ein Winkel im Bogenmaß.

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	✓	✓	✓	✓	-	-	-	-

Addition von Konstanten

Addition von Ganzzahl-Konstanten zum AKKU1.
Die Anzeigen werden nicht beeinflusst.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
+	i8		Addiere eine 8-Bit Integer-Konstante	1
+	i16		Addiere eine 16-Bit Integer-Konstante	2
+	i32		Addiere eine 32-Bit Integer-Konstante	3

Addition über Adressregister

Addition einer Ganzzahl (16Bit) zum Inhalt des Adressregisters.
Der Wert steht in der Operation oder im AKKU 1-L.
Die Anzeigen werden nicht beeinflusst.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
+AR1	-		Addiere Inhalt von AKKU1-L zum AR1	1
+AR1	m		Addiere Pointer-Konstante zum AR1	2
+AR2	-		Addiere Inhalt von AKKU1-L zum AR2	1
+AR2	m		Addiere Pointer-Konstante zum AR2	2

3.8 Baustein-Operationen

Baustein-Aufrufoperationen

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
CALL	FB p DB r	0 ... 8191 0 ... 8191	Unbedingter Aufruf eines FB, mit Parameterübergabe	
CALL	SFB p DB r	0 ... 8191 0 ... 8191	Unbedingter Aufruf eines SFB, mit Parameterübergabe	
CALL	FC p		Unbedingter Aufruf einer Funktion, mit Parameterübergabe	
CALL	SFC p		Unbedingter Aufruf einer SFC, mit Parameterübergabe	
UC	FB q FC q Parameter	0 ... 8191	Unbedingter Aufruf von Bausteinen, ohne Parameterübergabe FB/FC-Aufruf über Parameter	1/2
CC	FB q FC q Parameter	0 ... 8191	Bedingter Aufruf von Bausteinen, ohne Parameterübergabe FB/FC-Aufruf über Parameter	1/2
AUF	DB p DI p Parameter	0 ... 8191	Aufschlagen eines Datenbausteins Aufschlagen eines Instanz-Datenbausteins Aufschlagen eines Datenbausteins über Parameter	1/2 2 2

Statuswort für: CALL, UC	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	-	-	-	0	0	1	-	0

Statuswort für: CC	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	✓	-
Operation beeinflusst	-	-	-	-	0	0	1	-	0

Statuswort für: AUF	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	-	-	-	-	-	-	-	-

Baustein-Endeoperationen

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
BE			Beende Baustein	1
BEA			Beende Baustein absolut	1
BEB			Beende Baustein bei VKE="1"	1

Statuswort für BE, BEA:	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	-	-	-	0	0	1	-	0

Statuswort für BEB	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	✓	-
Operation beeinflusst	-	-	-	-	✓	0	1	1	0

Tausche Datenbausteine

Tauschen der beiden aktuellen Datenbausteine. Der aktuelle Datenbaustein wird zum aktuellen Instanz-Datenbaustein und umgekehrt.

Die Anzeigen werden nicht beeinflusst.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
TDB			Tausche Datenbausteine	1

3.9 Bildaufbau- und Null-Operationen

Das Statuswort wird nicht beeinflusst.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
BLD	0 ... 255		Bildaufbau-Operation; wird von der CPU wie eine Null-Operation behandelt	1
NOP	0 1		Nulloperation	1

3.10 Flanken-Operationen

Flanken-Operationen

Erkennen eines Flankenwechsels. Der aktuelle Signalzustand im VKE wird verglichen mit dem Signalzustand im Operanden, dem "Flankenmerker".

FP erkennt einen Flankenwechsel von "0" nach "1".

FN erkennt einen Flankenwechsel von "1" nach "0".

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
FP	E/A a.b	0.0 ... 2047.7	Anzeigen der steigenden Flanke im VKE, Flankenhilfsmerker ist das in der Operation adressierte Bit.	2
	M a.b	0.0 ... 8191.7		2
	L a.b	parametrierbar		2
	DBX a.b	0.0 ... 65535.7		2
	DIX a.b	0.0 ... 65535.7		2
	c [AR1,m]			2
	c [AR2,m]			2
	[AR1,m]			2
	[AR2,m]			2
	Parameter			2
FN	E/A a.b	0.0 ... 2047.7	Anzeigen der fallenden Flanke im VKE, Flankenhilfsmerker ist das in der Operation adressierte Bit	2
	M a.b	0.0 ... 8191.7		2
	L a.b	parametrierbar		2
	DBX a.b	0.0 ... 65535.7		2
	DIX a.b	0.0 ... 65535.7		2
	c [AR1,m]			2
	c [AR2,m]			2
	[AR1,m]			2
	[AR2,m]			2
	Parameter			2

Statuswort für: FP, FN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	✓	-
Operation beeinflusst	-	-	-	-	-	0	✓	✓	1

3.11 Lade-Operationen

Lade-Operationen

Laden der Operanden in AKKU1, zuvor wird der alte Inhalt von AKKU1 in AKKU2 gerettet.

Das Statuswort wird nicht beeinflusst.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
L			Lade ...	
	EB a		Eingangsbyte	1/2
	AB a		Ausgangsbyte	1/2
	PEB a		Peripherie-Eingangsbyte	2
	MB a	0.0 ... 8191	Merkerbyte	1/2
	LB a	parametrierbar	Lokaldatenbyte	2
	DBB a	0.0 ... 65535	Datenbyte	2
	DIB a	0.0 ... 65535	Instanz-Datenbyte	2
			... in AKKU1	
	g [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	g [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	B [AR1,m]		bereichsübergreifend (AR1)	2
	B [AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2
L			Lade ...	
	EW a	0.0 ... 2046	Eingangswort	1/2
	AW a	0.0 ... 2046	Ausgangswort	1/2
	PEW a	0.0 ... 8190	Peripherie-Eingangswort	2
	MW a	0.0 ... 8190	Merkerwort	1/2
	LW a	parametrierbar	Lokaldatenwort	2
	DBW a	0.0 ... 65534	Datenwort	1/2
	DIW a	0.0 ... 65534	Instanz-Datenwort	1/2
			... in AKKU1-L	
	h [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	h [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	W [AR1,m]		bereichsübergreifend (AR1)	2
	W [AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2
L			Lade ...	
	ED a	0.0 ... 2044	Eingangsdoppelwort	1/2
	AD a	0.0 ... 2044	Ausgangsdoppelwort	1/2
	PED a	0.0 ... 8188	Peripherie-Eingangsdoppelwort	2

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
	MD a	0.0 ... 8188	Merkerdoppelwort	1/2
	LD a	parametrierbar	Lokaldatendoppelwort	2
	DBD a	0.0 ... 65532	Datendoppelwort	2
	DID a	0.0 ... 65532	Instanz-Datendoppelwort	2
			... in AKKU1-L.	
	i [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	i [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	D [AR1,m]		bereichsübergreifend (AR1)	2
	D [AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2
L			Lade ...	
	k8		8-Bit-Konstante in AKKU1-LL	1
	k16		16-Bit-Konstante in AKKU1-L	2
	k32		32-Bit-Konstante in AKKU1	3
	Parameter		Lade Konstante in AKKU1 (über Parameter adressiert)	2
L	2#n		Lade 16-Bit-Binärkonstante in AKKU1-L	2
			Lade 32-Bit-Binärkonstante in AKKU1	3
L	B#8#p		Lade 8-Bit-Hexadezimalkonstante in AKKU1-LL	1
	W#16#p		Lade 16-Bit-Hexadezimalkonstante in AKKU1-L	2
	DW#16#p		Lade 32-Bit-Hexadezimalkonstante in AKKU1	3
L	x		Lade ein Zeichen	
L	xx		Lade zwei Zeichen	2
L	xxx		Lade drei Zeichen	
L	xxxx		Lade vier Zeichen	3
L	D# Datum		Lade IEC-Datum (BCD-codiert)	3
L	S5T# Zeitwert		Lade Zeitkonstante (16-Bit)	2
L	TOD# Zeitwert		Lade 32-Bit-Zeitkonstante (IEC-Tageszeit)	3
L	T# Zeitwert		Lade 16-Bit-Zeitkonstante Lade 32-Bit-Zeitkonstante	2 3
L	C# Zählwert		Lade 16-Bit-Zählerkonstante	2
L	P# Bitpointer		Lade Bitpointer	3
L	L# Integer		Lade 32-Bit-Ganzzahlkonstante	3
L	Realzahl		Lade Realzahl	3

Lade-Operationen für Timer und Zähler

Laden eines Zeitwertes oder Zählwertes in AKKU1, zuvor wird der alte Inhalt von AKKU1 in AKKU2 gerettet.

Die Anzeigen werden nicht beeinflusst.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
L	T f Timerpara.	0 ... 511	Lade Zeitwert Lade Zeitwert (über Parameter adressiert)	1/2 2
L	Z f Zählerpara.	0 ... 511	Lade Zählwert Lade Zählwert (über Parameter adressiert)	1/2 2
LC	T f Timerpara.	0 ... 511	Lade Zeitwert BCD-codiert Lade Zeitwert BCD-codiert (über Parameter adressiert)	1/2 2
LC	Z f Zählerpara.	0 ... 511	Lade Zählwert BCD-codiert Lade Zählwert BCD-codiert (über Parameter adressiert)	1/2 2

3.12 Schiebe-Operationen

Schiebe-Operationen

Schiebe Inhalt von AKKU1 oder AKKU1-L um die angegebene Anzahl von Stellen nach links/rechts.

Ist kein Operand angegeben, schiebe Anzahl in AKKU2-LL. Freiwerdende Stellen werden mit Nullen bzw. mit dem Vorzeichen aufgefüllt.

Zuletzt geschobenes Bit steht im Anzeigenbit A1.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
SLW SLW	- 0 ... 15		Schiebe Inhalt von AKKU1-L nach links, freiwerdende Stellen werden mit Nullen aufgefüllt.	1
SLD SLD	- 0 ... 32		Schiebe Inhalt von AKKU1 nach links, freiwerdende Stellen werden mit Nullen aufgefüllt.	1
SRW SRW	- 0 ... 15		Schiebe Inhalt von AKKU1-L nach rechts, freiwerdende Stellen werden mit Nullen aufgefüllt.	1
SRD SRD	- 0 ... 32		Schiebe Inhalt von AKKU1 nach rechts, freiwerdende Stellen werden mit Nullen aufgefüllt.	1
SSI SSI	- 0 ... 15		Schiebe den Inhalt von AKKU1-L mit Vorzeichen nach rechts, freiwerdende Stellen werden mit den Vorzeichen (Bit 15) aufgefüllt.	1
SSD SSD	- 0 ... 32		Schiebe den Inhalt von AKKU1 mit Vorzeichen nach rechts.	1

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	✓	✓	✓	-	-	-	-	-

Rotier-Operationen

Rotiere Inhalt von AKKU1 um die angegebene Anzahl von Stellen nach links/rechts.
Ist kein Operand angegeben, rotiere Anzahl in AKKU2-LL.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
RLD RLD	- 0 ... 32		Rotiere Inhalt von AKKU1 nach links	1
RRD RRD	- 0 ... 32		Rotiere Inhalt von AKKU1 nach rechts	1
RLDA	-		Rotiere Inhalt von AKKU1 um eine Bitposition nach links über Anzeigebit A1	
RRDA	-		Rotiere Inhalt von AKKU1 um eine Bitposition nach rechts über Anzeigebit A1	

Statuswort für: RLD, RRD	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	✓	✓	✓	-	-	-	-	-

Statuswort für: RLDA, RRDA	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	✓	0	0	-	-	-	-	-

3.13 Setzen/Rücksetzen von Bitoperanden

Setzen/Rücksetzen von Bitoperanden Zuweisen des Wertes "1" oder "0" bzw. des VKE an den adressierten Operanden.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
S			Setze ...	
	E/A a.b	0.0 ... 2047.7	Eingang/Ausgang auf "1"	1/2
	M a.b	0.0 ... 8191.7	Merker auf "1"	1/2
	L a.b	parametrierbar	Lokaldatenbit auf "1"	2
	DBX a.b	0.0 ... 65535.7	Datenbit auf "1"	2
	DIX a.b	0.0 ... 65535.7	Instanz-Datenbit auf "1"	2
	c [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	c [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	[AR1,m]		bereichsübergreifend (AR1)	2
	[AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2
R			Rücksetze ...	
	E/A a.b	0.0 ... 2047.7	Eingang/Ausgang auf "0"	1/2
	M a.b	0.0 ... 8191.7	Merker auf "0"	1/2
	L a.b	parametrierbar	Lokaldatenbit auf "0"	2
	DBX a.b	0.0 ... 65535.7	Datenbit auf "0"	2
	DIX a.b	0.0 ... 65535.7	Instanz-Datenbit auf "0"	2
	c [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	c [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	[AR1,m]		bereichsübergreifend (AR1)	2
	[AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2
=			Zuweisen ...	
	E/A a.b	0.0 ... 2047.7	des VKE an Eingang/Ausgang	1/2
	M a.b	0.0 ... 8191.7	des VKE an Merker	1/2
	L a.b	parametrierbar	des VKE an Lokaldatenbit	2
	DBX a.b	0.0 ... 65535.7	des VKE an Datenbit	2
	DIX a.b	0.0 ... 65535.7	des VKE an Instanz-Datenbit	2
	c [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	c [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	[AR1,m]		bereichsübergreifend (AR1)	2
	[AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2

Statuswort für: S, R, =	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	✓	-
Operation beeinflusst	-	-	-	-	-	0	✓	-	0

VKE direkt beeinflussende Operationen

Die folgenden Operationen bearbeiten direkt das VKE.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
CLR			Setze VKE auf "0"	1
SET			Setze VKE auf "1"	1
NOT			Negiere das VKE	1
SAVE			Rette das VKE in das BIE-Bit	1

Statuswort für: CLR	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	-	-	-	-	0	0	0	0

Statuswort für: SET	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	-	-	-	-	0	1	1	0

Statuswort für: NOT	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	✓	-	✓	-
Operation beeinflusst	-	-	-	-	-	-	1	✓	-

Statuswort für: SAVE	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	✓	-
Operation beeinflusst	✓	-	-	-	-	-	-	-	-

3.14 Sprung-Operationen

Sprungmarke

Die Sprungmarke ist eine symbolische Sprungadresse mit max. 4 Zeichen. Diese 4 Zeichen können sich aus Buchstaben, Zahlen und dem Unterstrich "_" zusammensetzen, wobei das 1. Zeichen ein Buchstabe sein muss. Groß- und Kleinschreibung werden unterschieden. Der Doppelpunkt ":" hinter der Sprungmarke kennzeichnet die Sprungmarke und leitet den Anweisungsteil ein.

Sprung, abhängig von der Bedingung.

Bei 8-Bit-Operanden liegt die Sprungweite zwischen (-128 ... +127),
bei 16-Bit-Operanden zwischen (-32768 ... -129) oder (+128 ... +32767)

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
SPA	MARKE		Springe unbedingt	1/2
SPB	MARKE		Springe bedingt bei VKE="1"	1/2
SPBN	MARKE		Springe bedingt bei VKE="0"	2
SPBB	MARKE		Springe bedingt bei VKE="1" Retten des VKE in das BIE-Bit	2
SPBNB	MARKE		Springe bedingt bei VKE="0" Retten des VKE in das BIE-Bit	2
SPBI	MARKE		Springe bedingt bei BIE="1"	2
SPBIN	MARKE		Springe bedingt bei BIE="0"	2
SPO	MARKE		Springe bedingt bei Überlauf (OV="1")	1/2
SPS	MARKE		Springe bedingt bei Überlauf speichernd (OS="1")	2
SPU	MARKE		Springe bei "Unzulässiger Operation" (A1=1 und A0=1)	2
SPZ	MARKE		Spring bedingt bei Ergebnis=0 (A1=0 und A0=0)	1/2
SPP	MARKE		Spring bedingt bei Ergebnis> 0 (A1=1 und A0=0)	1/2
SPM	MARKE		Spring bedingt bei Ergebnis < 0 (A1=0 und A0=1)	1/2
SPN	MARKE		Spring bedingt bei Ergebnis ≠ 0 (A1=1 und A0=0) oder (A1=0) und (A0=1)	1/2
SPMZ	MARKE		Springe bedingt bei Ergebnis ≤ 0 (A1=0 und A0=1) oder (A1=0 und A0=0)	2
SPPZ	MARKE		Springe bedingt bei Ergebnis ≥ 0 (A1=1 und A0=0) oder (A1=0 und A0=0)	2
SPL	MARKE		Sprungverteiler Der Operation folgt eine Liste von Sprungoperationen. Der Operand ist eine Sprungmarke auf die der Liste folgenden Operationen. AKKU1-L-L enthält die Nr. der Sprungoperation, die ausgeführt werden soll.	2
LOOP	MARKE		Dekrementiere AKKU1-L und springe bei AKKU1-L ≠ 0 (Schleifenprogrammierung)	2

Statuswort für: SPA, SPL, LOOP	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	-	-	-	-	-	-	-	-

Statuswort für: SPB, SPBN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	✓	-
Operation beeinflusst	-	-	-	-	-	0	1	1	0

Statuswort für: SPBB, SPBNB	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	✓	-
Operation beeinflusst	✓	-	-	-	-	0	1	1	0

Statuswort für: SPBI, SPBIN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	✓	-	-	-	-	-	-	-	-
Operation beeinflusst	-	-	-	-	-	0	1	-	0

Statuswort für: SPO	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	✓	-	-	-	-	-
Operation beeinflusst	-	-	-	-	-	-	-	-	-

Statuswort für: SPS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	✓	-	-	-	-
Operation beeinflusst	-	-	-	-	0	-	-	-	-

Statuswort für: SPU, SPZ, SPP, SPM, SPN, SPMZ, SPPZ	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	✓	✓	-	-	-	-	-	-
Operation beeinflusst	-	-	-	-	-	-	-	-	-

3.15 Transfer-Operationen

Transfer-Operationen

Transferieren des Inhalts von AKKU1 in den adressierten Operanden.

Das Statuswort wird nicht beeinflusst.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
T			Transferiere Inhalt von AKKU1-LL zum ...	
	EB a	0.0 ... 2047	Eingangsbyte	1/2
	AB a	0.0 ... 2047	Ausgangsbyte	1/2
	PAB a	0.0 ... 8191	Peripherie-Ausgangsbyte	1/2
	MB a	0.0 ... 8191	Merkerbyte	1/2
	LB a	parametrierbar	Lokaldatenbyte	2
	DBB a	0.0 ... 65535	Datenbyte	2
	DIB a	0.0 ... 65535	Instanz-Datenbyte	2
	g [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	g [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	B [AR1,m]		bereichsübergreifend (AR1)	2
	B [AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2
T			Transferiere Inhalt von AKKU1-L zum ...	
	EW	0.0 ... 2046	Eingangswort	1/2
	AW	0.0 ... 2046	Ausgangswort	1/2
	PAW	0.0 ... 8190	Peripherie-Ausgangswort	1/2
	MW	0.0 ... 8190	Merkerwort	1/2
	LW	parametrierbar	Lokaldatenwort	2
	DBW	0.0 ... 65534	Datenwort	2
	DIW	0.0 ... 65534	Instanz-Datenwort	2
	h [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	h [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	W [AR1,m]		bereichsübergreifend (AR1)	2
	W [AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2
T			Transferiere Inhalt von AKKU1 zum ...	

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
	ED	0.0 ... 2044	Eingangsdoppelwort	1/2
	AD	0.0 ... 2044	Ausgangsdoppelwort	1/2
	PAD	0.0 ... 8188	Peripherie-Ausgangsdoppelwort	1/2
	MD	0.0 ... 8188	Merkerdoppelwort	1/2
	LD	parametrierbar	Lokaldatendoppelwort	2
	DBD	0.0 ... 65532	Datendoppelwort	2
	DID	0.0 ... 65532	Instanz-Datendoppelwort	2
	i [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	i [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	D [AR1,m]		bereichsübergreifend (AR1)	2
	D [AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2

Lade- und Transfer-Operationen für Adressregister

Laden eines Doppelwortes aus einem Speicher oder einem Register in AR1 oder AR2. Das Statuswort wird nicht beeinflusst.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
LAR1			Lade Inhalt aus ...	
	-		AKKU1	1
	AR2		Adressregister 2	1
	DBD a	0 ... 65532	Datendoppelwort	2
	DID a	0 ... 65532	Instanz-Datendoppelwort	2
	m		32-Bit-Konstante als Pointer	3
	LD a	parametrierbar	Lokaldatendoppelwort	2
	MD a	0 ... 8188	Merkerdoppelwort	2
			... in AR1	
LAR2			Lade Inhalt aus ...	
	-		AKKU1	1
	DBD a	0 ... 65532	Datendoppelwort	2
	DID a	0 ... 65532	Instanz-Datendoppelwort	2
	m		32-Bit-Konstante als Pointer	3
	LD a	parametrierbar	Lokaldatendoppelwort	2
	MD a	0 ... 8188	Merkerdoppelwort.	2
			... in AR2	
TAR1			Transferiere Inhalt aus AR1 in ...	
	-		AKKU1	1
	AR2		Adressregister 2	1
	DBD a	0 ... 65532	Datendoppelwort	2
	DID a	0 ... 65532	Instanz-Datendoppelwort	2
	LD a	parametrierbar	Lokaldatendoppelwort	2
	MD a	0 ... 8188	Merkerdoppelwort	2
TAR2			Transferiere Inhalt aus AR2 in ...	
	-		AKKU1	1
	DBD a	0 ... 65532	Datendoppelwort	2
	DID a	0 ... 65532	Instanz-Datendoppelwort	2
	LD a	parametrierbar	Lokaldatendoppelwort	2
	MD a	0 ... 8188	Merkerdoppelwort	2
TAR			Tausche die Inhalte von AR1 und AR2	1

Lade- und Transfer-Operationen für das Statuswort

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
L	STW		Lade Statuswort in AKKU1 Wenn OR gesetzt wird, wird auch zwingend VKE und /ER gesetzt	
T	STW		Transferiere AKKU1 (Bits 0 ... 8) in das Statuswort	

Statuswort für: L STW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	✓	✓	✓	✓	✓	✓	✓	✓	0
Operation beeinflusst	-	-	-	-	-	-	-	-	-

Statuswort für: T STW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	✓	✓	✓	✓	✓	-	-	✓	-

Lade-Operationen für DB-Nummer und DB-Länge

Laden der Nummer/Länge eines Datenbausteins in AKKU1. Der alte Inhalt von AKKU1 wird in AKKU2 gerettet.

Die Anzeigen werden nicht beeinflusst.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
L	DBNO		Lade Nummer des Datenbausteins	1
L	DINO		Lade Nummer des Instanz-Datenbausteins	1
L	DBLG		Lade Länge des Datenbausteins in Byte	1
L	DILG		Lade Länge des Instanz-Datenbausteins in Byte	1

AKKU-Transferoperationen, Inkrementieren, Dekrementieren

Das Statuswort wird nicht beeinflusst.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
TAW	-		Umkehr der Reihenfolge der Bytes im AKKU1-L Aus LL, LH wird LH, LL	1
TAD	-		Umkehr der Reihenfolge der Bytes im AKKU1 Aus LL, LH, HL, HH wird HH, HL, LH, LL	1
TAK	-		Tausche Inhalte von AKKU1 und AKKU2	1
ENT	-		Der Inhalt von AKKU2 und AKKU3 wird in AKKU3 und AKKU4 übertragen	
LEAVE	-		Der Inhalt von AKKU3 und AKKU4 wird in AKKU2 und AKKU3 übertragen	
PUSH	-		Der Inhalt von AKKU1, AKKU2 und AKKU3 wird in AKKU2, AKKU3 und AKKU4 übertragen	1
POP	-		Der Inhalt von AKKU2, AKKU3 und AKKU4 wird in AKKU1, AKKU2 und AKKU3 übertragen	1
INC	0 ... 255		Inkrementiere AKKU1-LL	1
DEC	0 ... 255		Dekrementiere AKKU1-LL	1

3.16 Umwandlungs-Operationen

Datentyp-Umwandlungsoperationen

Die Ergebnisse der Wandlung stehen im AKKU1. Bei der Wandlung von Realzahlen ist die Ausführungszeit abhängig vom Wert.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
BTI	-		Konvertiere AKKU1 von BCD nach Ganzzahl (16Bit) (BCD to Int.)	1
BTD	-		Konvertiere AKKU1 von BCD nach Ganzzahl (32Bit) (BCD to Doubleint.)	1
DTR	-		Konvertiere AKKU1 von Ganzzahl (32Bit) nach Realzahl (32Bit) (Doubleint. to Real)	1
ITD	-		Konvertiere AKKU1 von Ganzzahl (16Bit) nach Ganzzahl (32Bit) (Int. to Doubleint.)	1

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	-	-	-	-	-	-	-	-

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
ITB	-		Konvertiere AKKU1 von Ganzzahl (16Bit) nach BCD 0 ... +/-999 (Int. To BCD)	1
DTB	-		Konvertiere AKKU1 von Ganzzahl (32Bit) nach BCD 0 ... +/-9 999 999 (Doubleint. To BCD)	1
RND	-		Wandle Realzahl in 32-Bit-Ganzzahl um	1
RND-	-		Wandle Realzahl in 32-Bit-Ganzzahl um Es wird abgerundet zur nächsten ganzen Zahl	1
RND+	-		Wandle Realzahl in 32-Bit-Integerzahl um Es wird aufgerundet zur nächsten ganzen Zahl	1
TRUNC	-		Wandle Realzahl in 32-Bit-Integerzahl um Es werden die Nachkommastellen abgeschnitten	1

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	-	-	✓	✓	-	-	-	-

Komplementbildung

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
INVI	-		Bilde 1er-Komplement von AKKU1-L	1
INVD	-		Bilde 1er-Komplement von AKKU1	1
NEGI	-		Bilde 2er-Komplement von AKKU1-L (Integerzahl)	1
NEGD	-		Bilde 2er-Komplement von AKKU1 (Double-Integerzahl)	1

Statuswort für: INVI, INVD	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	-	-	-	-	-	-	-	-

Statuswort für: NEGI, NEGD	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	✓	✓	✓	✓	-	-	-	-

3.17 Vergleichs-Operationen

Vergleichs-Operationen mit Ganzzahl (16Bit)

Vergleich der Ganzzahl (16Bit) in AKKU1-L und AKKU2-L.
VKE=1, wenn Bedingung erfüllt.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
==I	-		AKKU2-L = AKKU1-L	1
<>I	-		AKKU2-L ungleich AKKU1-L	1
<I	-		AKKU2-L < AKKU1-L	1
<=I	-		AKKU2-L <= AKKU1-L	1
>I	-		AKKU2-L > AKKU1-L	1
>=I	-		AKKU2-L >= AKKU1-L	1

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	✓	✓	0	-	0	✓	✓	1

Vergleichs-Operationen mit Ganzzahl (32Bit)

Vergleich der Ganzzahl (32Bit) in AKKU1 und AKKU2.
VKE=1, wenn Bedingung erfüllt.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
==D	-		AKKU2 = AKKU1	1
<>D	-		AKKU2 ungleich AKKU1	1
<D	-		AKKU2 < AKKU1	1
<=D	-		AKKU2 <= AKKU1	1
>D	-		AKKU2 > AKKU1	1
>=D	-		AKKU2 >= AKKU1	1

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	✓	✓	0	-	0	✓	✓	1

Vergleichs-Operationen mit (32-Bit-Realzahl)

Vergleich der 32-Bit-Realzahlen in AKKU1 und AKKU2.

VKE=1, wenn Bedingung erfüllt.

Die Ausführungszeit der Operation hängt vom Wert ab, der verglichen werden soll.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
==R	-		AKKU2 = AKKU1	1
<>R	-		AKKU2 ungleich AKKU1	1
<R	-		AKKU2 < AKKU1	1
<=R	-		AKKU2 <= AKKU1	1
>R	-		AKKU2 > AKKU1	1
>=R	-		AKKU2 >= AKKU1	1

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	✓	✓	✓	✓	0	✓	✓	1

3.18 Verknüpfungs-Operationen (Bit)

Verknüpfungs-Operationen mit Bitoperanden

Abfrage des adressierten Operanden auf seinen Signalzustand und Verknüpfung des Ergebnisses mit dem VKE nach der entsprechenden Funktion.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
U			UND-Verknüpfung auf Signalzustand "1"	
	E/A a.b	0.0 ... 2047.7	Eingang/Ausgang	1/2
	M a.b	0.0 ... 8191.7	Merker	1/2
	L a.b	parametrierbar	Lokaldatenbit	2
	DBX a.b	0.0 ... 65535.7	Datenbit	2
	DIX a.b	0.0 ... 65535.7	Instanz-Datenbit	2
	c [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	c [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	[AR1,m]		bereichsübergreifend (AR1)	2
	[AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2
UN			UND-Verknüpfung auf Signalzustand "0"	
	E/A a.b	0.0 ... 2047.7	Eingang/Ausgang	1/2
	M a.b	0.0 ... 8191.7	Merker	1/2
	L a.b	parametrierbar	Lokaldatenbit	2
	DBX a.b	0.0 ... 65535.7	Datenbit	2
	DIX a.b	0.0 ... 65535.7	Instanz-Datenbit	2
	c [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	c [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	[AR1,m]		bereichsübergreifend (AR1)	2
	[AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2

Statuswort für: U, UN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	✓	-	✓	✓
Operation beeinflusst	-	-	-	-	-	✓	✓	✓	1

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
O			ODER-Verknüpfung auf Signalzustand "1"	
	E/A a.b	0.0 ... 2047.7	Eingang/Ausgang	1/2
	M a.b	0.0 ... 8191.7	Merker	1/2

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
	L a.b	parametrierbar	Lokaldatenbit	2
	DBX a.b	0.0 ... 65535.7	Datenbit	2
	DIX a.b	0.0 ... 65535.7	Instanz-Datenbit	2
	c [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	c [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	[AR1,m]		bereichsübergreifend (AR1)	2
	[AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2
ON			ODER-Verknüpfung auf Signalzustand "0"	
	E/A a.b	0.0 ... 2047.7	Eingang/Ausgang	1/2
	M a.b	0.0 ... 8191.7	Merker	1/2
	L a.b	parametrierbar	Lokaldatenbit	2
	DBX a.b	0.0 ... 65535.7	Datenbit	2
	DIX a.b	0.0 ... 65535.7	Instanz-Datenbit	2
	c [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	c [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	[AR1,m]		bereichsübergreifend (AR1)	2
	[AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2

Statuswort für: O, ON	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	✓	✓
Operation beeinflusst	-	-	-	-	-	0	✓	✓	1

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
X			EXKLUSIV-ODER-Verknüpfung auf Signalzustand "1"	
	E/A a.b	0.0 ... 2047.7	Eingang/Ausgang	1/2
	M a.b	0.0 ... 8191.7	Merker	1/2
	L a.b	parametrierbar	Lokaldatenbit	2
	DBX a.b	0.0 ... 65535.7	Datenbit	2
	DIX a.b	0.0 ... 65535.7	Instanz-Datenbit	2
	c [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	c [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	[AR1,m]		bereichsübergreifend (AR1)	2
	[AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2
XN			EXKLUSIV-ODER-Verknüpfung auf Signalzustand "0"	
	E/A a.b	0.0 ... 2047.7	Eingang/Ausgang	1/2
	M a.b	0.0 ... 8191.7	Merker	1/2
	L a.b	parametrierbar	Lokaldatenbit	2
	DBX a.b	0.0 ... 65535.7	Datenbit	2
	DIX a.b	0.0 ... 65535.7	Instanz-Datenbit	2
	c [AR1,m]		registerindirekt, bereichsintern (AR1)	2
	c [AR2,m]		registerindirekt, bereichsintern (AR2)	2
	[AR1,m]		bereichsübergreifend (AR1)	2
	[AR2,m]		bereichsübergreifend (AR2)	2
	Parameter		über Parameter	2

Statuswort für: X, XN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	✓	✓
Operation beeinflusst	-	-	-	-	-	0	✓	✓	1

Verknüpfungs-Operationen von Klammerausdrücken

Retten der Bits BIE, VKE, OR und einer Funktionskennung (U, UN, ...) auf den Klammerstack.

Pro Baustein sind 7 Klammerebenen möglich.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
U(			UND-Klammer auf	1
UN(			UND-NICHT-Klammer auf	1
O(			ODER-Klammer auf	1
ON(			ODER-NICHT-Klammer auf	1
X(			EXCLUSIV-ODER-Klammer auf	1
XN(			EXKLUSIV-ODER-NICHT-Klammer auf	1
)			Klammer zu; entfernen eines Eintrags vom Klammerstack. Verknüpfen des VKE mit dem aktuellen VKE im Prozessor.	1

Statuswort für: U(, UN(, O(, ON(X(, XN(	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	✓	-	-	-	-	✓	-	✓	✓
Operation beeinflusst	-	-	-	-	-	0	1	-	0

Statuswort für:)	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	✓	-
Operation beeinflusst	✓	-	-	-	-	✓	1	✓	1

ODER-Verknüpfung von UND-Funktionen

Es erfolgt die ODER-Verknüpfung von UND-Funktionen nach der Regel: UND vor ODER.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
O			ODER-Verknüpfung von UND-Funktionen nach der Regel: UND vor ODER	1

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	✓	-	✓	✓
Operation beeinflusst	-	-	-	-	-	✓	1	-	✓

Verknüpfungs-Operationen mit Timern und Zählern

Abfrage des adressierten Timer/Zähler auf seinen Signalzustand und Verknüpfen des Ergebnisses mit dem VKE nach der entspr. Funktion.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
U			UND-Verknüpfung auf Signalzustand	
	T f	0 ... 511	Timer	1/2
	Z f	0 ... 511	Zähler	1/2
	Timerpara.		Timer über Parameter adressiert	2
	Zählerpara.		Zähler über Parameter adressiert	2
UN			UND-Verknüpfung auf Signalzustand	
	T f	0 ... 511	Timer	1/2
	Z f	0 ... 511	Zähler	1/2
	Timerpara.		Timer über Parameter adressiert	2
	Zählerpara.		Zähler über Parameter adressiert	2

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	✓	-	✓	✓
Operation beeinflusst	-	-	-	-	-	✓	✓	✓	1

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
O			ODER-Verknüpfung auf Signalzustand	
	T f	0 ... 511	Timer	1/2
	Z f	0 ... 511	Zähler	1/2
	Timerpara.		Timer über Parameter adressiert	2
	Zählerpara.		Zähler über Parameter adressiert	2
ON			ODER-Verknüpfung auf Signalzustand	
	T f	0 ... 511	Timer	1/2
	Z f	0 ... 511	Zähler	1/2
	Timerpara.		Timer über Parameter adressiert	2
	Zählerpara.		Zähler über Parameter adressiert	2

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	✓	✓
Operation beeinflusst	-	-	-	-	-	0	✓	✓	1

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
X			EXKLUSIV-ODER-Verknüpfung auf Signalzustand	
	T f	0 ... 511	Timer	1/2
	Z f	0 ... 511	Zähler	1/2
	Timerpara.		Timer über Parameter adressiert	2
	Zählerpara.		Zähler über Parameter adressiert	2
XN			EXKLUSIV-ODER-Verknüpfung auf Signalzustand	
	T f	0 ... 511	Timer	1/2
	Z f	0 ... 511	Zähler	1/2
	Timerpara.		Timer über Parameter adressiert	2
	Zählerpara.		Zähler über Parameter adressiert	2

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	✓	✓
Operation beeinflusst	-	-	-	-	-	0	✓	✓	1

Verknüpfungs-Operationen mit Anzeigenbits

Abfrage der angegebenen Bedingungen auf ihren Signalzustand und Verknüpfen des Ergebnisses mit dem VKE nach der entspr. Funktion.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
U, O, X			UND, ODER, EXCLUSIV ODER-Verknüpfung auf Signalzustand "1"	
	==0		Ergebnis = 0 (A1=0) und (A0=0)	1
	>0		Ergebnis > 0 (A1=1) und (A0=0)	1
	<0		Ergebnis < 0 (A1=0) und (A0=1)	1
	<>0		Ergebnis ungleich 0 ((A1=0) und (A0=1)) oder ((A1=1) und (A0=0))	1
	≤0		Ergebnis < 0 ((A1=0) und (A0=1)) oder ((A1=0) und (A0=0))	1
	≥0		Ergebnis ≥ 0 ((A1=1) und (A0=0)) oder ((A1=1) und (A0=0))	1
	UO		unordered (A1=1) und (A0=1)	1
	OS		OS=1	1
	BIE		BIE=1	1
	OV		OV=1	1

Statuswort für: U	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	✓	✓	✓	✓	✓	✓	-	✓	✓
Operation beeinflusst	-	-	-	-	-	✓	✓	✓	1

Statuswort für: O, X	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	✓	✓	✓	✓	✓	-	-	✓	✓
Operation beeinflusst	-	-	-	-	-	0	✓	✓	1

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
UN ON XN			UND NICHT/ODER NICHT/EXCLUSIV ODER NICHT Verknüpfung auf Signalzustand "0"	1
	==0		Ergebnis = 0 (A1=0) und (A0=0)	1
	>0		Ergebnis > 0 (A1=1) und (A0=0)	1
	<0		Ergebnis < 0 (A1=0) und (A0=1)	1
	<>0		Ergebnis ungleich 0 ((A1=0) und (A0=1)) oder ((A1=1) und (A0=0))	1
	<=0		Ergebnis < 0 ((A1=0) und (A0=1)) oder ((A1=0) und (A0=0))	1
	>=0		Ergebnis >= 0 ((A1=1) und (A0=0)) oder ((A1=1) und (A0=0))	1
	UO		unordered (A1=1) und (A0=1)	1
	OS		OS=0	1
	BIE		BIE=0	1
	OV		OV=0	1

Statuswort für: UN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	✓	✓	✓	✓	✓	✓	-	✓	✓
Operation beeinflusst	-	-	-	-	-	✓	✓	✓	1

Statuswort für: ON, XN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	✓	✓	✓	✓	✓	-	-	✓	✓
Operation beeinflusst	-	-	-	-	-	-	✓	✓	1

3.19 Verknüpfungs-Operationen (Wort)

Verknüpfungs-Operationen mit dem Inhalt von AKKU1

Verknüpfung des Inhalts von AKKU1 bzw. AKKU1-L mit einem Wort bzw. Doppelwort nach der entsprechenden Funktion. Das Wort bzw. Doppelwort steht entweder als Konstante in der Operation oder im AKKU2. Das Ergebnis steht im AKKU1 bzw. AKKU1-L.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
UW	k16		UND AKKU2-L	1
UW			UND 16-Bit-Konstante	2
OW	k16		ODER AKKU2-L	1
OW			ODER 16-Bit-Konstante	2
XOW	k16		EXKLUSIV ODER AKKU2-L	1
XOW			EXKLUSIV ODER 16-Bit-Konstante	2
UD	k32		UND AKKU2	1
UD			UND 32-Bit-Konstante	3
OD	k32		ODER AKKU2	1
OD			ODER 32-Bit-Konstante	3
XOD	k32		EXKLUSIV ODER AKKU2	1
XOD			EXKLUSIV ODER 32-Bit-Konstante	3

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	-	-
Operation beeinflusst	-	✓	0	0	-	-	-	-	-

3.20 Zeit-Operationen

Starten bzw. Rücksetzen eines Timers (direkt adressiert oder über Parameter adressiert).

Die Zeitdauer muss in AKKU1-L stehen.

Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
SI	T f	0 ... 511	Starte Zeit als Impuls bei Flankenwechsel von "0" nach "1"	1/2
	Timerpara.			2
SV	T f	0 ... 511	Starte Zeit als verlängerten Impuls bei Flankenwechsel von "0" nach "1"	1/2
	Timerpara.			2
SE	T f	0 ... 511	Starte Zeit als Einschaltverzögerung bei Flanken-wechsel von "0" nach "1"	1/2
	Timerpara.			2
SS	T f	0 ... 511	Starte Zeit als speichernde Einschaltverzögerung bei Flankenwechsel von "0" nach "1"	1/2
	Timerpara.			2
SA	T f	0 ... 511	Starte Zeit als Ausschaltverzögerung bei Flanken-wechsel "1" nach "0"	1/2
	Timerpara.			2
FR	T f	0 ... 511	Freigabe eines Timers für das erneute Starten bei Flan-kenwechsel (Löschen des Flankenmerkers für das Starten der Zeit)	1/2
	Timerpara.			2
R	T f	0 ... 511	Rücksetzen einer Zeit	1/2
	Timerpara.			2

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	✓	-
Operation beeinflusst	-	-	-	-	-	0	-	-	0

3.21 Zähl-Operationen

Der Zählwert steht im AKKU1-L bzw. in der als Parameter übergebenen Adresse.

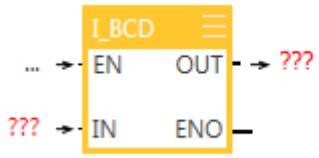
Befehl	Operand	Parameter	Funktionsbezeichnung	Länge in Worten
S	Z f	0 ... 511	Vorbelegen eines Zählers bei Flankenwechsel von "0" nach "1"	1/2
	Zählerpara.			2
R	Z f	0 ... 511	Rücksetzen des Zählers auf "0" bei Flankenwechsel von "0" nach "1"	1/2
	Zählerpara.			2
ZV	Z f	0 ... 511	Zähle um 1 vorwärts bei Flankenwechsel von "0" nach "1"	1/2
	Zählerpara.			2
ZR	Z f	0 ... 511	Zähle um 1 rückwärts bei Flankenwechsel von "0" nach "1"	1/2
	Zählerpara.			2
FR	Z f	0 ... 511	Freigabe eines Zählers bei Flankenwechsel "0" nach "1" (Löschen des Flankenmerkers für Vorwärts- und Rückwärtszählen eines Zählers)	1/2
	Zählerpara.			2

Statuswort	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation hängt ab von	-	-	-	-	-	-	-	✓	-
Operation beeinflusst	-	-	-	-	-	0	-	-	0

4 FUP-Operationen

4.1 Überblick

Funktionsplan (FUP) ist eine grafische Programmiersprache zur Signalverarbeitung. Mit Funktionsplan können unterschiedliche Funktionselemente miteinander verknüpft werden, um den Signalfluss zu steuern.



Zeichen	Bedeutung
➞	links vom Element: Eingangsparameter (eingehender Wert) rechts vom Element: Ausgangsparameter (ausgehender Wert)
???	Angabe des Parameters ist zwingend
...	Angabe des Parameters ist optional

4.2 Bitverknüpfungen

4.2.1 Überblick

Mit den Bitverknüpfungen können Sie binäre (boolesche) Operationen programmieren. Binäre Operationen basieren auf den beiden Signalzuständen "0" und "1".

Operation	Bitverknüpfung
>=1	ODER-Verknüpfung
&	UND-Verknüpfung
XOR	EXKLUSIV-ODER-Verknüpfung
=	Zuweisung
#	Konnektor
S	Ausgang setzen
R	Ausgang rücksetzen
RS	Flipflop rücksetzen setzen
SR	Flipflop setzen rücksetzen
P	Flanke 0-1 erkennen
N	Flanke 1-0 erkennen
POS	Signalflanke 0-1 erkennen
NEG	Signalflanke 1-0 erkennen

Bei den Elementen ">=1", "&" und "XOR" können Sie mit "--|" weitere binäre Eingänge einfügen.

Mit "-o|" können Sie binäre Eingänge negieren.

Mit der Operation "SAVE" können Sie das Verknüpfungsergebnis "VKE" in das BIE-Register übernehmen.

Wahrheitstabelle

Die folgende Wahrheitstabelle zeigt die Verknüpfungsergebnisse für verschiedene binäre Operationen mit zwei Eingangsparametern.

Operand 1	Operand 2	ODER	UND	XOR
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

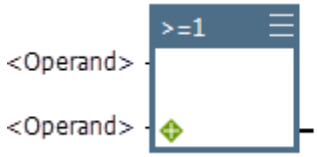
4.2.2 ODER-Verknüpfung - >=1

Mit der ODER-Verknüpfung können Sie die Signalzustände mehrerer Operanden an den Eingängen abfragen und miteinander logisch verknüpfen, wie in der ODER-Wahrheitstabelle dargestellt.

Wenn mindestens ein Eingang den Signalzustand "1" führt, nimmt der Ausgang den Wert "1" an. Wenn alle Eingänge den Signalzustand "0" führen, nimmt der Ausgang den Wert "0" an.

Wenn die ODER-Verknüpfung die erste Operation in einer Verknüpfungskette ist, dann wird das Ergebnis der Signalzustandsabfrage im VKE-Bit gespeichert.

Wenn die ODER-Verknüpfung nicht die erste Operation in einer Verknüpfungskette ist, dann wird das Ergebnis der Signalzustandsabfrage mit dem Wert verknüpft, der im VKE-Bit gespeichert ist.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, T, Z, D, L	Bit-Operand, dessen Signalzustand abgefragt wird

Statuswort für: >=1	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel

Wenn die Eingänge `E4.0` **oder** `E7.7` den Signalzustand "1" führen, liegt am Ausgang `A2.0` der Wert "1" an.

4.2.3 UND-Verknüpfung - &

Mit der UND-Verknüpfung können Sie die Signalzustände mehrerer Operanden an den Eingängen abfragen und miteinander logisch verknüpfen, wie in der UND-Wahrheitstabelle dargestellt.

Wenn sämtliche Eingänge den Signalzustand "1" führen, nimmt der Ausgang den Wert "1" an. Wenn auch nur ein Eingang den Signalzustand "0" führt, nimmt der Ausgang den Wert "0" an.

Wenn die UND-Verknüpfung die erste Operation in einer Verknüpfungskette ist, dann wird das Ergebnis der Signalzustandsabfrage im VKE-Bit gespeichert.

Wenn die UND-Verknüpfung nicht die erste Operation in einer Verknüpfungskette ist, dann wird das Ergebnis der Signalzustandsabfrage mit dem Wert verknüpft, der im VKE-Bit gespeichert ist.

Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, T, Z, D, L	Der Operand gibt das Bit an, dessen Signalzustand abgefragt wird

Statuswort für: &	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

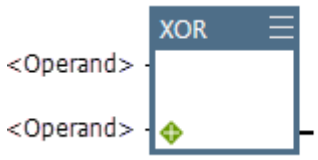
Beispiel

Wenn die Eingänge `E4.0` **und** `E7.7` den Signalzustand "1" führen, liegt am Ausgang `A2.0` der Wert "1" an.

4.2.4 EXKLUSIV-ODER-Verknüpfung - XOR

Mit der XOR-Verknüpfung können Sie die Signalzustände mehrerer Operanden an den Eingängen abfragen und miteinander logisch verknüpfen, wie in der EXKLUSIV-ODER-Wahrheitstabelle dargestellt.

Wenn die Signale der beiden Eingänge unterschiedlich sind, nimmt der Ausgang den Wert "1" an. Bei XOR-Verknüpfungen mit mehr als zwei Eingängen nimmt der Ausgang den Wert "1" an, wenn eine ungerade Anzahl von Eingängen den Signalzustand "1" führt.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, T, Z, D, L	Bit-Operand, dessen Signalzustand abgefragt wird

Statuswort für: XOR	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel

Wenn **entweder** der Eingang `E4.0` **oder** der Merker `M1.0` den Signalzustand "1" führt, liegt am Ausgang `A2.0` der Wert "1" an. Wenn beide Eingänge den gleichen Signalzustand "0" oder "1" führen, liegt am Ausgang der Wert "0" an.

4.2.5 Binären Eingang einfügen - --|

Mit der Funktion "Binären Eingang einfügen" können Sie UND-, ODER- sowie XOR-Operationen um einen weiteren binären Eingang erweitern.

Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, T, Z, D, L	Bit-Operand, dessen Signalzustand abgefragt wird

Statuswort für: --|	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	-	1	✓	-

Beispiel

Wenn die Eingänge `E4.0` **und** `E7.7` **und** der Merker `M8.4` den Signalzustand "1" führen, liegt am Ausgang `A2.0` der Wert "1" an.

4.2.6 Binären Eingang negieren (NICHT-Operation) - -o|

Mit der Operation "Binären Eingang negieren" können Sie den Eingängen den entgegengesetzten Signalzustand zuweisen (Negation, Inverter).

Statuswort für: -o|	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	-	1	✓	-

Beispiel

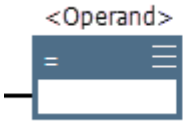
Wenn der Eingang `E4.0` den Signalzustand "1" führt und der Eingang `E7.7` **nicht** den Signalzustand "1" führt, liegt am Ausgang `A2.0` der Wert "1" an.

4.2.7 Zuweisung - =

Mit der "Zuweisung" können Sie das Verknüpfungsergebnis (VKE) einem Operanden zuweisen. Sie können die Zuweisung an den Ausgang einer Verknüpfung setzen.

Wenn die Bedingungen der Verknüpfung vor dem Zuweisungselement erfüllt sind (VKE =1), nimmt der Ausgang den Wert "1" an. Wenn die Bedingungen der Verknüpfung vor dem Zuweisungselement nicht erfüllt sind (VKE =0), nimmt der Ausgang den Wert "0" an.

Sie können eine Zuweisung negieren, indem Sie deren Eingang negieren ↬ *Kap. 4.2.6 " Binären Eingang negieren (NICHT-Operation) - -o|" Seite 79*.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, dem das Verknüpfungsergebnis (VKE) zugewiesen wird

Statuswort für: =	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	✓	-	0

Beispiel

Wenn die Eingänge `E4.0` und `E7.7` den Signalzustand "1" führen, liegt am Ausgang `A2.0` der Wert "1" an.

4.2.8 Konnektor -

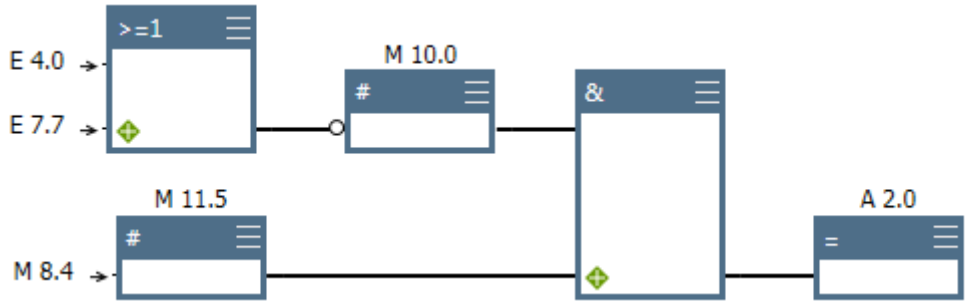
Mit dem "Konnektor" können Sie das Verknüpfungsergebnis VKE in einem Operanden speichern.

Sie können einen Konnektor negieren, indem Sie dessen Eingang negieren *Kap. 4.2.6 " Binären Eingang negieren (NICHT-Operation) - -o|" Seite 79*.

Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, *L	Bit-Operand, in dem das Verknüpfungsergebnis VKE gespeichert wird
* Im Lokaldaten-Stack können Operanden nur dann verwendet werden, wenn sie im TEMP-Bereich der Variablendeklaration eines Programmbausteins (OB, FB, FC) deklariert worden sind.			

Statuswort für: #	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	✓	-	1

Beispiel



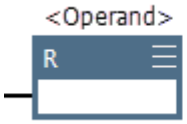
In den Konnektoren werden folgende Verknüpfungsergebnisse gespeichert:

- `M10.0` speichert das negierte VKE der davor liegenden ODER-Verknüpfung. Das VKE wird unverändert an die folgende UND-Verknüpfung weitergeleitet.
- `M11.5` speichert das VKE von `M8.4`.

4.2.9 Ausgang rücksetzen - R

Mit den Operationen "Ausgang setzen (S)" und "Ausgang rücksetzen (R)" können Sie eine Signalspeicherung programmieren.

Der Operand wird nur dann auf den Wert "0" zurückgesetzt, wenn das Verknüpfungsergebnis VKE = 1 ist. Wenn das VKE = 0 ist, bleibt der Operand unverändert.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, T, Z, D, L	Bit-Operand, der zurückgesetzt werden soll

Statuswort für: R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	✓	-	0

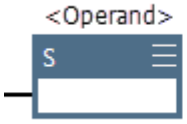
Beispiel

Wenn die Eingänge `E4.0` oder `E7.7` den Signalzustand "1" führen, dann wird der Ausgang `A2.0` auf den Wert "0" zurückgesetzt. Wenn das VKE der Verknüpfung anschließend wieder nach "0" wechselt, bleibt der Signalzustand am Ausgang unverändert auf dem Wert "0" .

4.2.10 Ausgang setzen - S

Mit den Operationen "Ausgang setzen (S)" und "Ausgang rücksetzen (R)" können Sie eine Signalspeicherung programmieren.

Der Operand wird nur dann auf den Wert "1" gesetzt, wenn das Verknüpfungsergebnis VKE = 1 ist. Wenn das VKE = 0 ist, bleibt der Operand unverändert.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, der gesetzt werden soll

Statuswort für: S	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	✓	-	0

Beispiel

Wenn die Eingänge `E4.7` und `E8.0` den Signalzustand "1" führen, dann wird der Ausgang `A2.0` auf den Wert "1" gesetzt. Wenn das VKE der Verknüpfung anschließend wieder nach "0" wechselt, bleibt der Signalzustand am Ausgang unverändert auf dem Wert "1" .

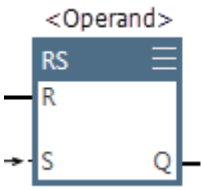
4.2.11 Flipflop rücksetzen setzen - RS

Mit einem Flipflop (bistabiles Element) können Sie ein binäres Signal speichern.

Der Operand wird nur dann gesetzt (S) bzw. zurückgesetzt (R), wenn das Verknüpfungsergebnis VKE = 1 ist. Wenn das Verknüpfungsergebnis VKE = 0 ist, bleibt der Operand unverändert.

Wenn die Eingänge den Signalzustand *R* = 0 und *S* = 1 führen, wird der Operand auf den Wert "1" gesetzt. Wenn die Eingänge den Signalzustand *R* = 1 und *S* = 0 führen, wird der Operand auf den Wert "0" zurückgesetzt.

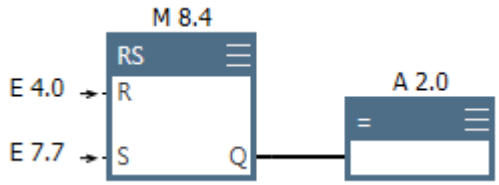
Mit der Operation "Flipflop rücksetzen setzen" wird der Operand **vorrangig gesetzt**. Das bedeutet, wenn die beiden Eingänge *R* und *S* gleichzeitig den Signalzustand "1" führen, wird der Operand auf den Wert "1" gesetzt.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, der gesetzt oder zurückgesetzt werden soll
R	BOOL	E, A, M, D, L, T, Z	Speicher zurücksetzen
S	BOOL	E, A, M, D, L, T, Z	Speicher setzen
Q	BOOL	E, A, M, D, L	Ausgang, Signalzustand des Operanden

Statuswort für: RS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



Wenn der Eingang `E4.0` den Signalzustand "0" führt und der Eingang `E7.7` den Signalzustand "1" führt, werden der Merker `M8.4` und der Ausgang `A2.0` auf den Wert "1" gesetzt. Wenn der Eingang `E4.0` den Signalzustand "1" führt und der Eingang `E7.7` den Signalzustand "0" führt, werden der Merker `M8.4` und der Ausgang `A2.0` auf den Wert "0" zurückgesetzt.

Wenn am Eingang beide Signalzustände "0" führen, bleiben die Signalzustände am Merker und Ausgang unverändert. Wenn am Eingang beide Signalzustände "1" führen, werden der Merker und der Ausgang auf den Wert "1" gesetzt.

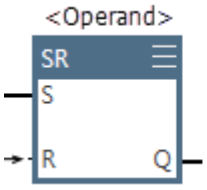
4.2.12 Flipflop setzen rücksetzen - SR

Mit einem Flipflop (bistabiles Element) können Sie ein binäres Signal speichern.

Der Operand wird nur dann gesetzt (S) bzw. zurückgesetzt (R), wenn das Verknüpfungsergebnis VKE = 1 ist. Wenn das Verknüpfungsergebnis VKE = 0 ist, bleibt der Operand unverändert.

Wenn die Eingänge den Signalzustand *R* = 0 und *S* = 1 führen, wird der Operand auf den Wert "1" gesetzt. Wenn die Eingänge den Signalzustand *R* = 1 und *S* = 0 führen, wird der Operand auf den Wert "0" zurückgesetzt.

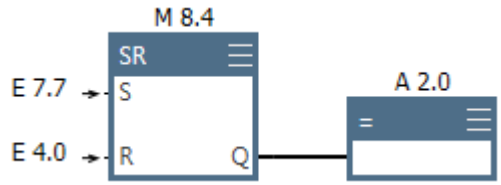
Mit der Operation "Flipflop setzen rücksetzen" wird der Operand **vorrangig rückgesetzt**. Das bedeutet, wenn die beiden Eingänge *R* und *S* gleichzeitig den Signalzustand "1" führen, wird der Operand auf den Wert "0" zurückgesetzt.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, der gesetzt oder zurückgesetzt werden soll
S	BOOL	E, A, M, D, L, T, Z	Speicher setzen
R	BOOL	E, A, M, D, L, T, Z	Speicher zurücksetzen
Q	BOOL	E, A, M, D, L	Ausgang, Signalzustand des Operanden

Statuswort für: SR	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



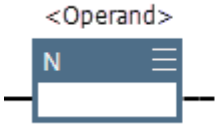
Wenn der Eingang `E4.0` den Signalzustand "0" führt und der Eingang `E7.7` den Signalzustand "1" führt, werden der Merker `M8.4` und der Ausgang `A2.0` auf den Wert "1" gesetzt. Wenn der Eingang `E4.0` den Signalzustand "1" führt und der Eingang `E7.7` den Signalzustand "0" führt, werden der Merker `M8.4` und der Ausgang `A2.0` auf den Wert "0" zurückgesetzt.

Wenn am Eingang beide Signalzustände "0" führen, bleiben die Signalzustände am Merker und Ausgang unverändert. Wenn am Eingang beide Signalzustände "1" führen, werden der Merker und der Ausgang auf den Wert "0" zurückgesetzt.

4.2.13 Flanke 1-0 erkennen - N

Die Operation "Flanke 1-0 erkennen" erkennt den Übergang des Verknüpfungsergebnisses VKE von "1" auf "0" (fallende Flanke).

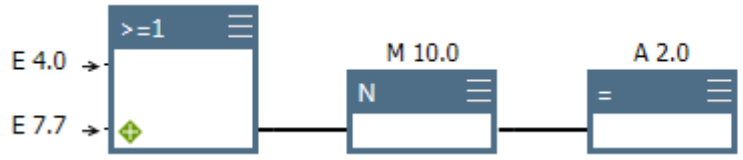
Die Operation vergleicht den Wert des VKE am Eingang der N-Operation mit dem Wert der vorherigen Abfrage, der im Operanden (Flankenmerker) gespeichert ist. Nur wenn der Flankenmerker den Wert "1" führt und das aktuelle VKE den Wert "0" führt, liegt eine **fallende Flanke** vor. Das VKE am Ausgang der N-Operation wird kurzzeitig (als Impuls) auf den Wert "1" gesetzt.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, der den vorherigen Zustand des VKE speichert (Flankenmerker)

Statuswort für: N	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	✓	✓	1

Beispiel

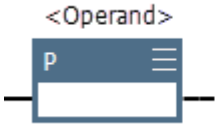


Wenn das VKE der ODER-Verknüpfung von "1" nach "0" wechselt, wird der Ausgang `A2.0` kurzzeitig auf den Wert "1" gesetzt. Der Flankenmerker `M10.0` wird dabei als Speicher verwendet, um die Flanke festzustellen.

4.2.14 Flanke 0-1 erkennen - P

Die Operation "Flanke 0-1 erkennen" erkennt den Übergang des Verknüpfungsergebnisses VKE von "0" auf "1" (steigende Flanke).

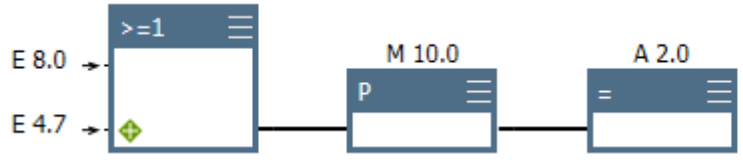
Die Operation vergleicht den Wert des VKE am Eingang der P-Operation mit dem Wert der vorherigen Abfrage, der im Operanden (Flankenmerker) gespeichert ist. Nur wenn der Flankenmerker den Wert "0" führt und das aktuelle VKE den Wert "1" führt, liegt eine **steigende Flanke** vor. Das VKE am Ausgang der P-Operation wird kurzzeitig (als Impuls) auf den Wert "1" gesetzt.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, der den vorherigen Zustand des VKE speichert (Flankenmerker)

Statuswort für: P	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	✓	✓	1

Beispiel

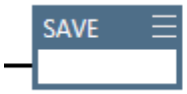


Wenn das VKE der ODER-Verknüpfung von "0" nach "1" wechselt, wird der Ausgang `A2.0` kurzzeitig auf den Wert "1" gesetzt. Der Flankenmerker `M10.0` wird dabei als Speicher verwendet, um die Flanke festzustellen.

4.2.15 Verknüpfungsergebnis in BIE-Register übernehmen - SAVE

Mit der Operation "SAVE" wird das Verknüpfungsergebnis VKE in das Binärergebnis-Bit "BIE" des Statuswortes übernommen. Das Erstabfragebit "/ER" wird nicht zurückgesetzt. Am Bausteinende wird der Ausgang "ENO" auf den Wert des VKE gesetzt. Die nächste Operation wird mit dem VKE des aktuellen Netzwerks verknüpft.

Die Operation "SAVE" kann beispielsweise vor dem Verlassen eines Bausteins verwendet werden, um das Verknüpfungsergebnis im aufrufenden Baustein auszuwerten. Da sich das Binärergebnis-Bit "BIE" durch folgende Operationen ändern kann, sollten Sie die Operation "SAVE" direkt vor dem Verlassen eines Bausteins einsetzen.



Statuswort für: SAVE	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	-	-	-	-	-	-

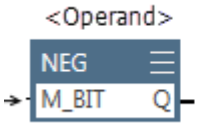
Beispiel

Das Verknüpfungsergebnis VKE wird im Binärergebnis-Bit "BIE" gespeichert.

4.2.16 Signalflanke 1-0 erkennen - NEG

Die Operation "Signalflanke 1-0 erkennen" erkennt den Übergang des Signalzustands von "1" auf "0" (fallende Flanke).

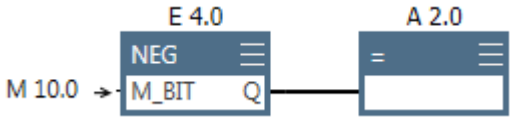
Die Operation vergleicht den Signalzustand des Operanden mit dem Signalzustand der vorherigen Abfrage, der im Eingang *M_BIT* (Flankenmerker) gespeichert ist. Nur wenn *M_BIT* den Wert "1" führt und der Operand den Wert "0" führt, liegt eine **fallende Flanke** vor. Das VKE am Ausgang der NEG-Operation wird kurzzeitig (als Impuls) auf den Wert "1" gesetzt.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, dessen Flankenwechsel erkannt werden soll
M_BIT	BOOL	A, M, D	Bit-Operand, der den vorherigen Signalzustand speichert (Flankenmerker).
Q	BOOL	E, A, M, D, L	Ausgang, Impuls bei Flankenwechsel

Statuswort für: NEG	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	1	✓	1

Beispiel

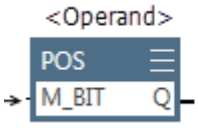


Wenn der Signalzustand am Eingang `E4.0` von "1" nach "0" wechselt, wird der Ausgang `A2.0` kurzzeitig auf den Wert "1" gesetzt. Der Flankenmerker `M10.0` wird dabei als Speicher verwendet, um die Flanke festzustellen.

4.2.17 Signalflanke 0-1 erkennen - POS

Die Operation "Signalflanke 1-0 erkennen" erkennt den Übergang des Signalzustands von "0" auf "1" (steigende Flanke).

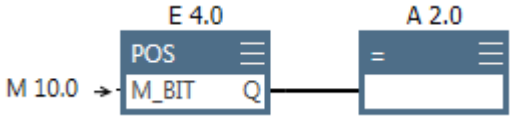
Die Operation vergleicht den Signalzustand des Operanden mit dem Signalzustand der vorherigen Abfrage, der im Eingang *M_BIT* (Flankenmerker) gespeichert ist. Nur wenn *M_BIT* den Wert "0" führt und der Operand den Wert "1" führt, liegt eine **steigende Flanke** vor. Das VKE am Ausgang der POS-Operation wird kurzzeitig (als Impuls) auf den Wert "1" gesetzt.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, dessen Flankenwechsel erkannt werden soll
M_BIT	BOOL	A, M, D	Bit-Operand, der den vorherigen Signalzustand speichert (Flankenmerker).
Q	BOOL	E, A, M, D, L	Ausgang, Impuls bei Flankenwechsel

Statuswort für: POS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	1	✓	1

Beispiel



Wenn der Signalzustand am Eingang `E4.0` von "0" nach "1" wechselt, wird der Ausgang `A2.0` kurzzeitig auf den Wert "1" gesetzt. Der Flankenmerker `M10.0` wird dabei als Speicher verwendet, um die Flanke festzustellen.

4.3 Vergleicher

4.3.1 Überblick

Mit den Vergleichsoperationen können Sie zwei Werte miteinander vergleichen.

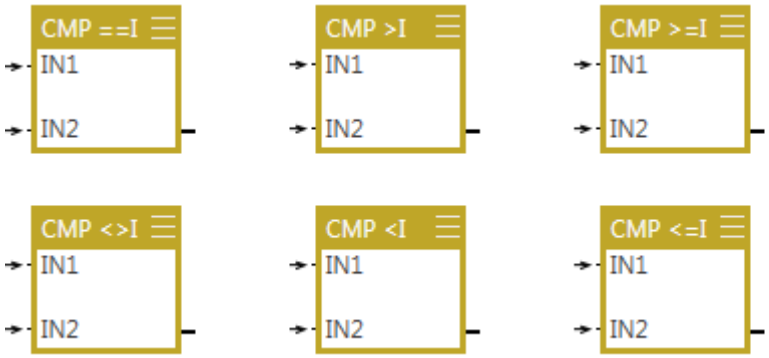
Operation	Vergleich
==	Wert 1 ist gleich Wert 2
<>	Wert 1 ist ungleich Wert 2
>	Wert 1 ist größer als Wert 2
<	Wert 1 ist kleiner als Wert 2
>=	Wert 1 ist größer als oder gleich Wert 2
<=	Wert 1 ist kleiner als oder gleich Wert 2

Sie können Werte der folgenden Datentypen vergleichen:

- Ganze Zahlen (16 Bit) - CMP xx I
- Ganze Zahlen (32 Bit) - CMP xx D
- Gleitpunktzahlen - CMP xx R

4.3.2 Ganze Zahlen vergleichen (16 Bit) - CMP xx I

Mit der Operation "Ganze Zahlen vergleichen (16 Bit)" können Sie zwei 16 Bit-Festpunktzahlen an den Eingängen *IN1* und *IN2* miteinander vergleichen.



Parameter	Datentyp	Speicherbereich	Beschreibung
IN1	INT	E, A, M, D, L oder Konstante	erster Vergleichswert
IN2	INT	E, A, M, D, L oder Konstante	zweiter Vergleichswert
Ausgang	BOOL	E, A, M, D, L	Vergleichsergebnis

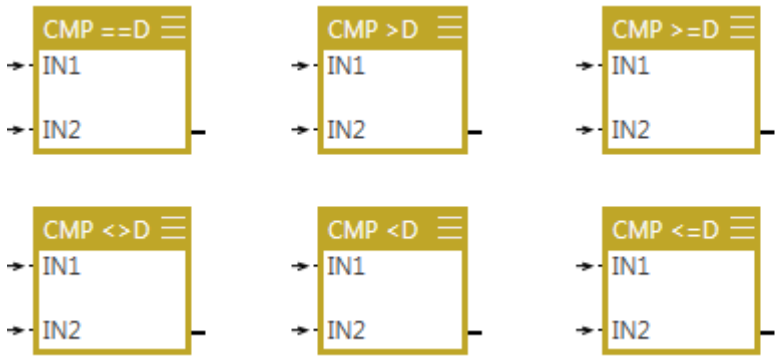
Statuswort für: CMP xx I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	0	-	0	✓	✓	1

Beispiel

Wenn `MW38` **ungleich** `MW40` ist, wird der Ausgang `A2.0` zurückgesetzt.

4.3.3 Ganze Zahlen vergleichen (32 Bit) - CMP xx D

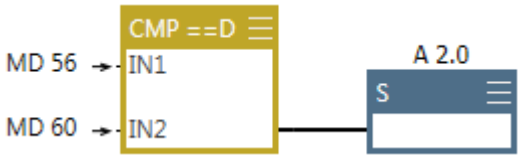
Mit der Operation "Ganze Zahlen vergleichen (32 Bit)" können Sie zwei 32 Bit-Festpunktzahlen an den Eingängen *IN1* und *IN2* miteinander vergleichen.



Parameter	Datentyp	Speicherbereich	Beschreibung
IN1	DINT	E, A, M, D, L oder Konstante	erster Vergleichswert
IN2	DINT	E, A, M, D, L oder Konstante	zweiter Vergleichswert
Ausgang	BOOL	E, A, M, D, L	Vergleichsergebnis

Statuswort für: CMP xx D	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	✓	✓	0	-	0	✓	✓	1

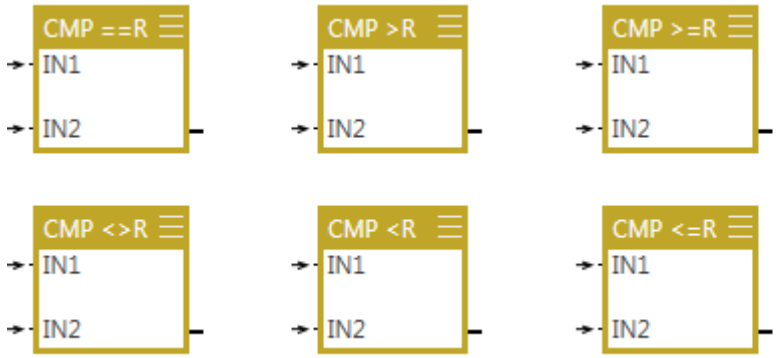
Beispiel



Wenn `MD56` **gleich** `MD60` ist, wird der Ausgang `A2.0`gesetzt.

4.3.4 Gleitpunktzahlen vergleichen - CMP xx R

Mit der Operation "Gleitpunktzahlen vergleichen" können Sie zwei Gleitpunktzahlen (32 Bit-Realzahlen) an den Eingängen *IN1* und *IN2* miteinander vergleichen.



Parameter	Datentyp	Speicherbereich	Beschreibung
IN1	REAL	E, A, M, D, L oder Konstante	erster Vergleichswert
IN2	REAL	E, A, M, D, L oder Konstante	zweiter Vergleichswert
Ausgang	BOOL	E, A, M, D, L	Vergleichsergebnis

Statuswort für: CMP xx R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel

Wenn `MD56` **größer oder gleich** `MD60` ist, wird der Ausgang `A2.0`gesetzt.

4.4 Umwandler

4.4.1 Überblick

Mit den Umwandlungsoperationen können Sie Werte von einem Zahlenformat in ein anderes umwandeln.

Operation	Umwandlung
BCD_I	BCD-Zahl in Ganzzahl (16 Bit) wandeln
I_BCD	Ganzzahl (16 Bit) in BCD-Zahl wandeln
I_DI	Ganzzahl (16 Bit) in Ganzzahl (32 Bit) wandeln
BCD_DI	BCD-Zahl in Ganzzahl (32 Bit) wandeln
DI_BCD	Ganzzahl (32 Bit) in BCD-Zahl wandeln
DI_R	Ganzzahl (32 Bit) in Gleitpunktzahl wandeln
INV_I	Einer-Komplement zu Ganzzahl (16 Bit) erzeugen

Operation	Umwandlung
INV_DI	Einer-Komplement zu Ganzzahl (32 Bit) erzeugen
NEG_I	Zweier-Komplement zu Ganzzahl (16 Bit) erzeugen
NEG_DI	Zweier-Komplement zu Ganzzahl (32 Bit) erzeugen
NEG_R	Vorzeichen einer Gleitpunktzahl wechseln
ROUND	Zahl runden
TRUNC	Ganze Zahl erzeugen
CEIL	aus Gleitpunktzahl nächsthöhere Ganzzahl erzeugen
FLOOR	aus Gleitpunktzahl nächstniedrige Ganzzahl erzeugen

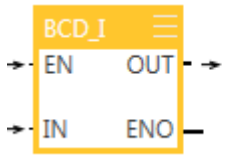
4.4.2 BCD-Zahl in Ganzzahl (16 Bit) wandeln - BCD_I

Die Operation "BCD-Zahl in Ganzzahl (16 Bit) wandeln" wandelt die dreistellige binär-codierte Dezimalzahl (BCD, ± 999) am Eingang *IN* in einen ganzzahligen Wert (16 Bit) am Ausgang *OUT* um. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt.

Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.

Wenn eine binär-codierte Dezimalstelle der BCD-Zahl im ungültigen Bereich liegt (größer als 9), tritt ein Wandlungsfehler auf:

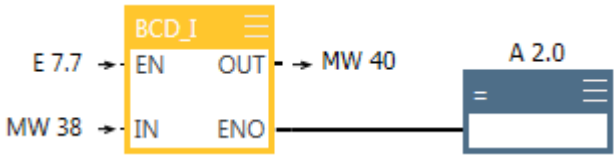
- Die CPU schaltet in den Betriebszustand STOP. Im Diagnosespeicher wird ein "BCD-Umwandlungsfehler" mit der Ereignisnummer 2521 eingetragen.
- Wenn der Organisationsbaustein OB121 programmiert worden ist, wird er aufgerufen.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	WORD	E, A, M, D, L oder Konstante	BCD-Zahl
OUT	INT	E, A, M, D, L	ganzzahliger Wert (16 Bit) der BCD-Zahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: BCD_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	-	-	-	-	0	1	1	1

Beispiel

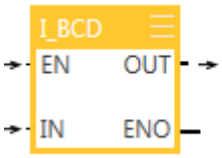


Wenn `E7.7` = 1 ist, wird die dreistellige BCD-Zahl im Merkerwort `MW38` in eine Ganzzahl (16 Bit) umgewandelt und im Merkerwort `MW40` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.4.3 Ganzzahl (16 Bit) in BCD-Zahl wandeln - I_BCD

Die Operation "Ganzzahl (16 Bit) in BCD-Zahl wandeln" wandelt den ganzzahligen Wert (16 Bit) am Eingang *IN* in eine dreistellige binär-codierte Dezimalzahl (BCD, ± 999) am Ausgang *OUT* um. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt.

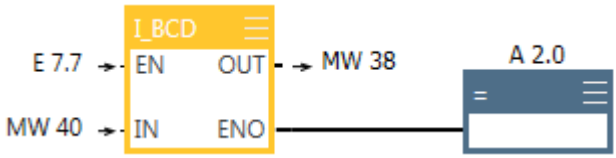
Wenn ein Überlauf eintritt, ist der Parameter *ENO* = 0 und die Umwandlung wird nicht ausgeführt.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	INT	E, A, M, D, L oder Konstante	Ganzzahl (16 Bit)
OUT	WORD	E, A, M, D, L	BCD-Wert der Ganzzahl (16 Bit)
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: I_BCD	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	✓	✓	0	✓	✓	1

Beispiel

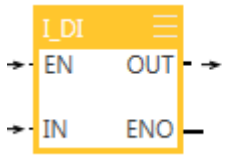


Wenn `E7.7` = 1 ist, wird die Ganzzahl (16 Bit) im Merkerwort `MW40` in eine dreistellige BCD-Zahl umgewandelt und im Merkerwort `MW38` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN). Wenn ein Überlauf eintritt, ist `A2.0` = 0 und die Umwandlung wird nicht ausgeführt.

4.4.4 Ganzzahl (16 Bit) in Ganzzahl (32 Bit) wandeln - I_DI

Die Operation "Ganzzahl (16 Bit) in Ganzzahl (32 Bit) wandeln" wandelt die Ganzzahl (16 Bit) am Eingang *IN* in eine Ganzzahl (32 Bit) am Ausgang *OUT* um. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt.

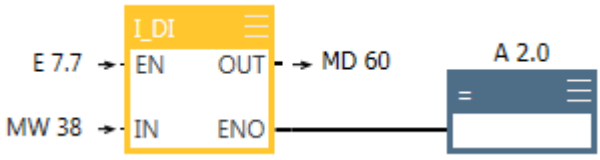
Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	INT	E, A, M, D, L oder Konstante	Ganzzahl (16 Bit)
OUT	DINT	E, A, M, D, L	Ganzzahl (32 Bit)
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: I_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	-	-	0	1	1	1

Beispiel



Wenn `E7.7` = 1 ist, wird die Ganzzahl (16 Bit) im Merkerwort `MW38` in eine Ganzzahl (32 Bit) umgewandelt und im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

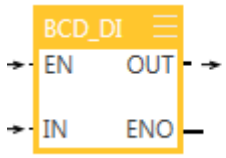
4.4.5 BCD-Zahl in Ganzzahl (32 Bit) wandeln - BCD_DI

Die Operation "BCD-Zahl in Ganzzahl (32 Bit) wandeln" wandelt die siebenstellige binär-codierte Dezimalzahl (BCD, ± 9.999.999) am Eingang *IN* in einen ganzzahligen Wert (32 Bit) am Ausgang *OUT* um. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt.

Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.

Wenn eine binär-codierte Dezimalstelle der BCD-Zahl im ungültigen Bereich liegt (größer als 9), tritt ein Wandlungsfehler auf:

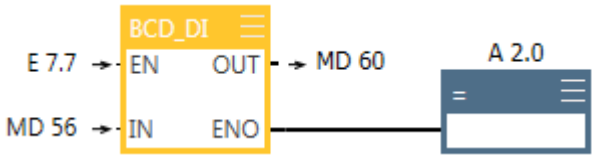
- Die CPU schaltet in den Betriebszustand STOP. Im Diagnosespeicher wird ein "BCD-Umwandlungsfehler" mit der Ereignisnummer 2521 eingetragen.
- Wenn der Organisationsbaustein OB121 programmiert worden ist, wird er aufgerufen.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DWORD	E, A, M, D, L oder Konstante	BCD-Zahl
OUT	DINT	E, A, M, D, L	ganzzahliger Wert (32 Bit) der BCD-Zahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: BCD_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	-	-	-	-	0	1	1	1

Beispiel

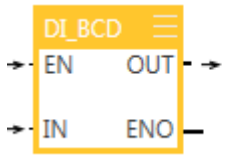


Wenn `E7.7` = 1 ist, wird die siebenstellige BCD-Zahl im Merkerdoppelwort `MD56` in eine Ganzzahl (32 Bit) umgewandelt und im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.4.6 Ganzzahl (32 Bit) in BCD-Zahl wandeln - DI_BCD

Die Operation "Ganzzahl (32 Bit) in BCD-Zahl wandeln" wandelt den ganzzahligen Wert (32 Bit) am Eingang *IN* in eine siebenstellige binär-codierte Dezimalzahl (BCD, ± 9.999.999) am Ausgang *OUT* um. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt.

Wenn ein Überlauf eintritt, ist der Parameter *ENO* = 0 und die Umwandlung wird nicht ausgeführt.

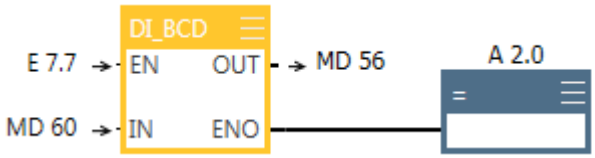


Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DINT	E, A, M, D, L oder Konstante	Ganzzahl (32 Bit)

Parameter	Datentyp	Speicherbereich	Beschreibung
OUT	DWORD	E, A, M, D, L	BCD-Wert der Ganzzahl (32 Bit)
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: DI_BCD	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	✓	✓	0	✓	✓	1

Beispiel

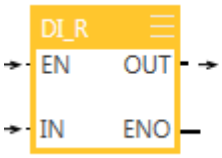


Wenn `E7.7` = 1 ist, wird die Ganzzahl (32 Bit) im Merkerdoppelwort `MD60` in eine siebenstellige BCD-Zahl umgewandelt und im Merkerdoppelwort `MD56` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN). Wenn ein Überlauf eintritt, ist `A2.0` = 0 und die Umwandlung wird nicht ausgeführt.

4.4.7 Ganzzahl (32 Bit) in Gleitpunktzahl wandeln - DI_R

Die Operation "Ganzzahl (32 Bit) in Gleitpunktzahl wandeln" wandelt die Ganzzahl (32 Bit) am Eingang *IN* in eine Gleitpunktzahl am Ausgang *OUT* um. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt.

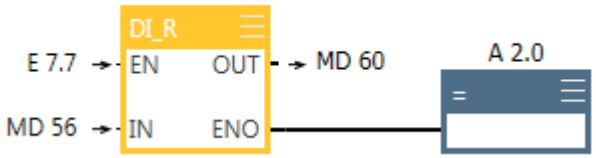
Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DINT	E, A, M, D, L oder Konstante	Ganzzahl (32 Bit)
OUT	REAL	E, A, M, D, L	Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: DI_R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	-	-	-	-	0	1	1	1

Beispiel

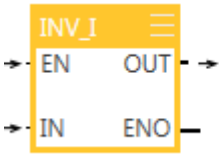


Wenn `E7.7` = 1 ist, wird die Ganzzahl (32 Bit) im Merkerdoppelwort `MD56` in eine Gleitpunktzahl umgewandelt und im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.4.8 Einer-Komplement zu Ganzzahl (16 Bit) erzeugen - INV_I

Die Operation "Einer-Komplement zu Ganzzahl (16 Bit) erzeugen" führt eine Wortverknüpfungsoperation "EXKLUSIV ODER verknüpfen" des Eingangs *IN* mit der Hexadezimalschablone FFFF aus. Dadurch werden die Werte der einzelnen Bits umgekehrt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

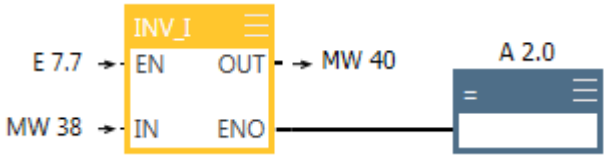
Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	INT	E, A, M, D, L oder Konstante	Ganzzahl (16 Bit)
OUT	INT	E, A, M, D, L	Einer-Komplement der Ganzzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: INV_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	-	-	-	-	0	1	1	1

Beispiel



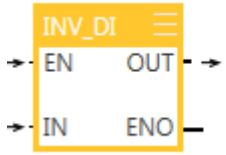
Wenn `E7.7` = 1 ist, wird aus der Ganzzahl (16 Bit) im Merkerwort `MW38` das Einer-Komplement gebildet. Jedes Bit wird umgekehrt. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel: `MW38` = 10011100 01101010, `MW40` = 01100011 10010101

4.4.9 Einer-Komplement zu Ganzzahl (32 Bit) erzeugen - INV_DI

Die Operation "Einer-Komplement zu Ganzzahl (32 Bit) erzeugen" führt eine Wortverknüpfungsoperation "EXKLUSIV ODER verknüpfen" des Eingangs *IN* mit der Hexadezimalschablone FFFF aus. Dadurch werden die Werte der einzelnen Bits umgekehrt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

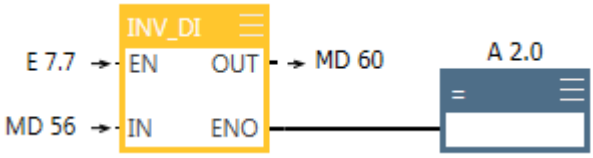
Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	INT	E, A, M, D, L oder Konstante	Ganzzahl (32 Bit)
OUT	INT	E, A, M, D, L	Einer-Komplement der Ganzzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: INV_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	-	-	-	-	0	1	1	1

Beispiel



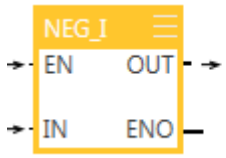
Wenn `E7.7` = 1 ist, wird aus der Ganzzahl (32 Bit) im Merkerdoppelwort `MD56` das Einer-Komplement gebildet. Jedes Bit wird umgekehrt. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel: `MD56` = FE01 5B83, `MD60` = 01FE A47C

4.4.10 Zweier-Komplement zu Ganzzahl (16 Bit) erzeugen - NEG_I

Die Operation "Zweier-Komplement zu Ganzzahl (16 Bit) erzeugen" kehrt das Vorzeichen des Werts am Eingang *IN* um, z. B. von einem positiven in einen negativen Wert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

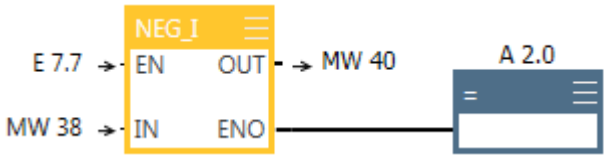
Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand. Ausnahme: Wenn der Signalzustand von *EN* gleich "1" ist und ein Überlauf eintritt, ist der Signalzustand von *ENO* gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	INT	E, A, M, D, L oder Konstante	Ganzzahl (16 Bit)
OUT	INT	E, A, M, D, L	Zweier-Komplement der Ganzzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: NEG_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel



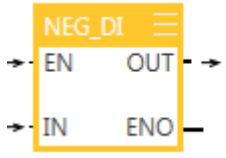
Wenn `E7.7` = 1 ist, wird das Vorzeichen der Ganzzahl (16 Bit) im Merkerwort `MW38` umgekehrt. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel: `MW38` = 42, `MW40` = -42

4.4.11 Zweier-Komplement zu Ganzzahl (32 Bit) erzeugen - NEG_DI

Die Operation "Zweier-Komplement zu Ganzzahl (32 Bit) erzeugen" kehrt das Vorzeichen des Werts am Eingang *IN* um, z. B. von einem positiven in einen negativen Wert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

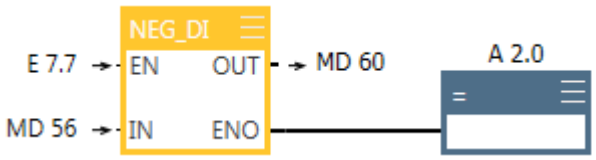
Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand. Ausnahme: Wenn der Signalzustand von *EN* gleich "1" ist und ein Überlauf eintritt, ist der Signalzustand von *ENO* gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DINT	E, A, M, D, L oder Konstante	Ganzzahl (32 Bit)
OUT	DINT	E, A, M, D, L	Zweier-Komplement der Ganzzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: NEG_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel



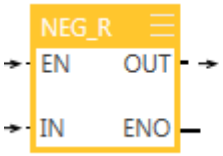
Wenn `E7.7` = 1 ist, wird das Vorzeichen der Ganzzahl (32 Bit) im Merkerdoppelwort `MD56` umgekehrt. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel: `MD56` = 100.000, `MD60` = -100.000

4.4.12 Vorzeichen einer Gleitpunktzahl wechseln - NEG_R

Die Operation "Vorzeichen einer Gleitpunktzahl wechseln" kehrt das Vorzeichenbit des Werts am Eingang *IN* um, z. B. von"0" für positiv in "1" für negativ. Die Bits für Exponent und Mantisse bleiben unverändert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

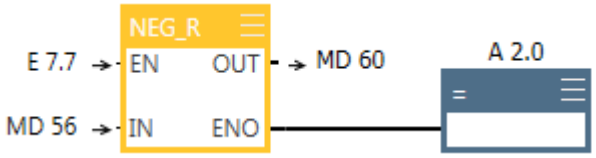
Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand. Ausnahme: Wenn der Signalzustand von *EN* gleich "1" ist und ein Überlauf eintritt, ist der Signalzustand von *ENO* gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Gleitpunktzahl, Vorzeichen gewechselt
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: NEG_R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	-	-	0	✓	✓	1

Beispiel



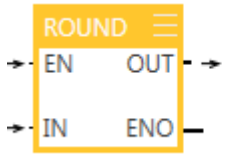
Wenn `E7.7` = 1 ist, wird das Vorzeichen der Gleitpunktzahl im Merkerdoppelwort `MD56` umgekehrt. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel: `MD56` = 42e2, `MD60` = -42e2

4.4.13 Zahl runden - ROUND

Die Operation "Zahl runden" rundet die Gleitpunktzahl am Eingang *IN* zu einer Ganzzahl (32 Bit). Wenn die erste Ziffer nach dem Komma 0, 1, 2, 3 oder 4 ist, wird abgerundet (Beispiel: "2,49" wird abgerundet zu "2"). Wenn die erste Ziffer nach dem Komma 5, 6, 7, 8 oder 9 ist, wird aufgerundet (Beispiel: "2,5" wird aufgerundet zu "3"). Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

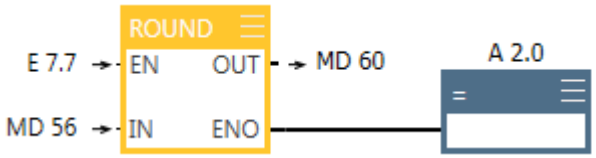
Wenn ein Überlauf eintritt, ist der Signalzustand von *ENO* gleich "0". Wenn der Wert am Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	DINT	E, A, M, D, L	Ganzzahl (32 Bit), zur nächsten ganzen Zahl gerundet
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ROUND	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	✓	✓	0	✓	✓	1

Beispiel

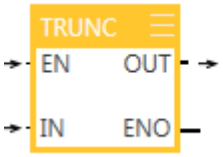


Wenn `E7.7` = 1 ist, wird die Gleitpunktzahl im Merkerdoppelwort `MD56` auf- bzw. abgerundet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.4.14 Ganze Zahl erzeugen - TRUNC

Die Operation "Ganze Zahl erzeugen" wandelt den Wert der Gleitpunktzahl am Eingang *IN* in eine Ganzzahl (32 Bit) um, indem die Nachkommastellen abgeschnitten werden (Beispiel: "2,49" wird zu "2", "2,5" wird ebenfalls zu "2"). Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

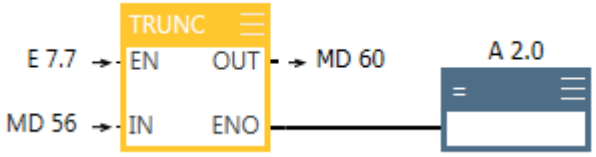
Wenn ein Überlauf eintritt, ist der Signalzustand von *ENO* gleich "0". Wenn der Wert am Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	DINT	E, A, M, D, L	Ganzzahl (32 Bit), Nachkommastellen abgeschnitten
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: TRUNC	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	✓	✓	0	✓	✓	1

Beispiel

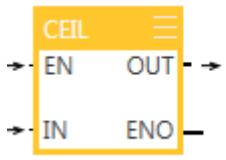


Wenn `E7.7` = 1 ist, werden die Nachkommastellen der Gleitpunktzahl im Merkerdoppelwort `MD56` abgeschnitten. Das Ergebnis wird als Ganzzahl (32 Bit) im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.4.15 Aus Gleitpunktzahl nächsthöhere Ganzzahl erzeugen - CEIL

Die Operation "Aus Gleitpunktzahl nächsthöhere Ganzzahl erzeugen" rundet die Gleitpunktzahl am Eingang *IN* zu einer Ganzzahl (32 Bit) auf. (Beispiel: "2,3" wird zu "3", "-2,8" wird zu "-2"). Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

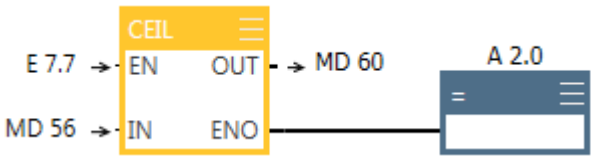
Wenn ein Überlauf eintritt, ist der Signalzustand von *ENO* gleich "0". Wenn der Wert am Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	DINT	E, A, M, D, L	Ganzzahl (32 Bit), aufgerundet
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: CEIL	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	✓	✓	0	✓	✓	1

Beispiel

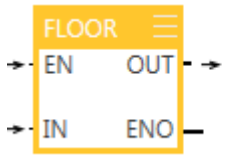


Wenn `E7.7` = 1 ist, wird die Gleitpunktzahl im Merkerdoppelwort `MD56` aufgerundet. Das Ergebnis wird als Ganzzahl (32 Bit) im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.4.16 Aus Gleitpunktzahl nächstniedrige Ganzzahl erzeugen - FLOOR

Die Operation "Aus Gleitpunktzahl nächstniedrige Ganzzahl erzeugen" rundet die Gleitpunktzahl am Eingang *IN* zu einer Ganzzahl (32 Bit) ab. (Beispiel: "2,8" wird zu "2", "-2,3" wird zu "-3"). Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

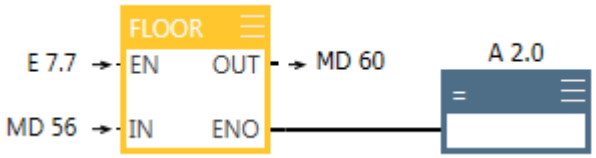
Wenn ein Überlauf eintritt, ist der Signalzustand von *ENO* gleich "0". Wenn der Wert am Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	DINT	E, A, M, D, L	Ganzzahl (32 Bit), abgerundet
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: FLOOR	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	✓	✓	0	✓	✓	1

Beispiel



Wenn `E7.7` = 1 ist, wird die Gleitpunktzahl im Merkerdoppelwort `MD56` abgerundet. Das Ergebnis wird als Ganzzahl (32 Bit) im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.5 Zähler

4.5.1 Überblick

In der CPU ist ein eigener Speicherbereich für Zähler reserviert. Die Anzahl der Zähler ist abhängig von der CPU.

Operation	Zähler
Z_VORW	Parametrieren und Vorwärtszählen
Z_RUECK	Parametrieren und Rückwärtszählen
ZAEHLER	Parametrieren und Vorwärts-/ Rückwärtszählen
SZ	Setzen des Zähleranfangswerts
ZV	Vorwärtszählen
ZR	Rückwärtszählen

4.5.2 Parametrieren und Vorwärts-/Rückwärtszählen - ZAEHLER

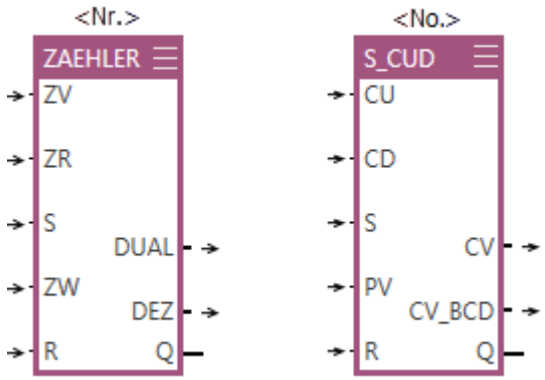
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird der Zähler auf den Zählwert *ZW* gesetzt. Wenn der Eingang *R* den Signalzustand "1" führt, wird der Zählwert *ZW* auf "0" gesetzt.

Bei jeder steigenden Flanke am Eingang *ZV* wird der Zählwert um 1 erhöht. Der Zählwert wird nicht weiter erhöht, wenn er "999" erreicht hat. Bei jeder steigenden Flanke am Eingang *ZR* wird der Zählwert um 1 vermindert. Der Zählwert wird nicht weiter vermindert, wenn er "0" erreicht hat. Wenn gleichzeitig an beiden Eingängen eine steigende Flanke vorliegt, ändert sich der Zählwert nicht.

Wenn der Zählwert gesetzt wird und der Eingang *ZV* oder *ZR* den Signalzustand "1" führt, wird der Zählwert einmalig im nächsten Zyklus erhöht bzw. vermindert, auch wenn kein Flankenwechsel eingetreten ist.

Wenn der Zählwert größer als "0" ist, führt der Ausgang *Q* den Wert "1". Wenn der Zählwert gleich "0" ist, führt der Ausgang *Q* den Wert "0".

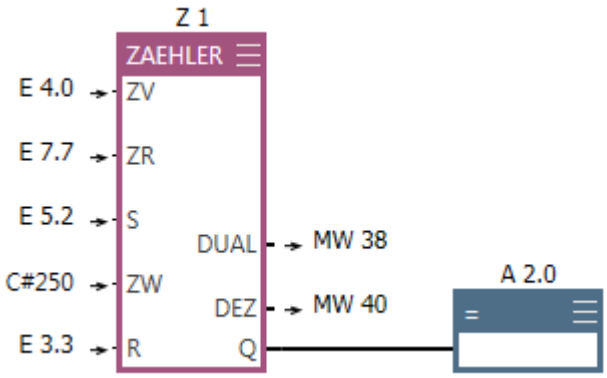
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	COUNTER	Z	Nummer des Zählers, Bereich ist abhängig von der CPU
ZV	CU	BOOL	E, A, M, D, L	Vorwärtszählen
ZR	CD	BOOL	E, A, M, D, L	Rückwärtszählen
S	S	BOOL	E, A, M, D, L, T, Z	Zählwert setzen
ZW	PV	WORD	E, A, M, D, L oder Konstante	Zählwert, BCD-kodierte Zählerkonstante: C#<Wert>, <Wert> zwischen 0 und 999
R	R	BOOL	E, A, M, D, L, T, Z	Zählwert rücksetzen
DUAL	CV	WORD	E, A, M, D, L	aktueller Zählwert, hexadezimal
DEZ	CV_BCD	WORD	E, A, M, D, L	aktueller Zählwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status des Zählers

Statuswort für: ZAEHLER	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird der Zähler `Z1` auf den Wert "250" gesetzt. Wenn der Signalzustand am Eingang `E4.0` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` um 1 erhöht. Wenn der Signalzustand am Eingang `E7.7` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` um 1 vermindert. Wenn der Signalzustand am Eingang `E3.3` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` auf "0" gesetzt.

Wenn der Wert des Zählers `Z1` größer als "0" ist, ist `A2.0` = 1. Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zählwert.

4.5.3 Parametrieren und Vorwärtszählen - Z_VORW

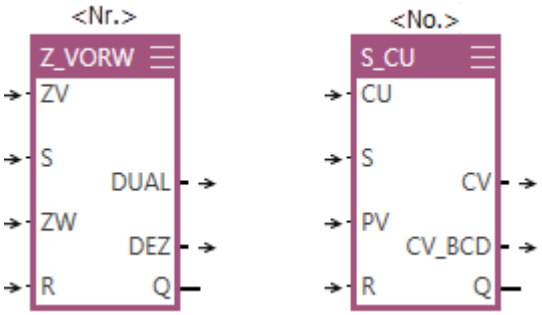
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird der Zähler auf den Zählwert *ZW* gesetzt. Wenn der Eingang *R* den Signalzustand "1" führt, wird der Zählwert *ZW* auf "0" gesetzt.

Bei jeder steigenden Flanke am Eingang *ZV* wird der Zählwert um 1 erhöht. Der Zählwert wird nicht weiter erhöht, wenn er "999" erreicht hat.

Wenn der Zählwert gesetzt wird und der Eingang *ZV* den Signalzustand "1" führt, wird der Zählwert einmalig im nächsten Zyklus erhöht, auch wenn kein Flankenwechsel eingetreten ist.

Wenn der Zählwert größer als "0" ist, führt der Ausgang *Q* den Wert "1". Wenn der Zählwert gleich "0" ist, führt der Ausgang *Q* den Wert "0".

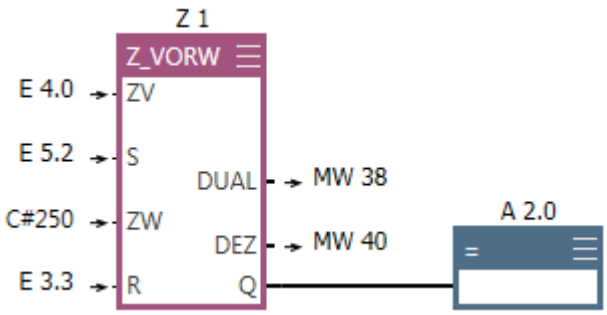
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	COUNTER	Z	Nummer des Zählers, Bereich ist abhängig von der CPU
ZV	CU	BOOL	E, A, M, D, L	Vorwärtszählen
S	S	BOOL	E, A, M, D, L, T, Z	Zählwert setzen
ZW	PV	WORD	E, A, M, D, L oder Konstante	Zählwert, BCD-kodierte Zählerkonstante: C#<Wert>, <Wert> zwischen 0 und 999
R	R	BOOL	E, A, M, D, L, T, Z	Zählwert rücksetzen
DUAL	CV	WORD	E, A, M, D, L	aktueller Zählwert, hexadezimal
DEZ	CV_BCD	WORD	E, A, M, D, L	aktueller Zählwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status des Zählers

Statuswort für: Z_VORW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird der Zähler `Z1` auf den Wert "250" gesetzt. Wenn der Signalzustand am Eingang `E4.0` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` um 1 erhöht. Wenn der Signalzustand am Eingang `E3.3` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` auf "0" gesetzt.

Wenn der Wert des Zählers `Z1` größer als "0" ist, ist `A2.0` = 1. Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zählwert.

4.5.4 Parametrieren und Rückwärtszählen - Z_RUECK

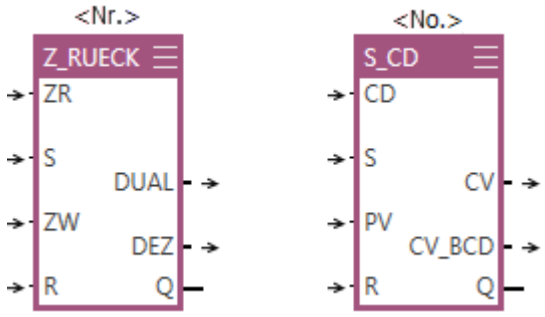
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird der Zähler auf den Zählwert *ZW* gesetzt. Wenn der Eingang *R* den Signalzustand "1" führt, wird der Zählwert *ZW* auf "0" gesetzt.

Bei jeder steigenden Flanke am Eingang *ZR* wird der Zählwert um 1 vermindert. Der Zählwert wird nicht weiter vermindert, wenn er "0" erreicht hat.

Wenn der Zählwert gesetzt wird und der Eingang *ZR* den Signalzustand "1" führt, wird der Zählwert einmalig im nächsten Zyklus vermindert, auch wenn kein Flankenwechsel eingetreten ist.

Wenn der Zählwert größer als "0" ist, führt der Ausgang *Q* den Wert "1". Wenn der Zählwert gleich "0" ist, führt der Ausgang *Q* den Wert "0".

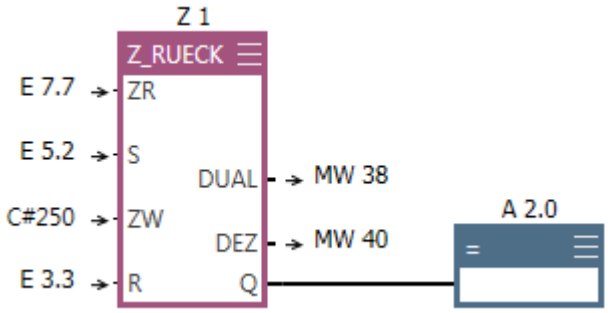
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	COUNTER	Z	Nummer des Zählers, Bereich ist abhängig von der CPU
ZR	CD	BOOL	E, A, M, D, L	Rückwärtszählen
S	S	BOOL	E, A, M, D, L, T, Z	Zählwert setzen
ZW	PV	WORD	E, A, M, D, L oder Konstante	Zählwert, BCD-kodierte Zählerkonstante: C#<Wert>, <Wert> zwischen 0 und 999
R	R	BOOL	E, A, M, D, L, T, Z	Zählwert rücksetzen
DUAL	CV	WORD	E, A, M, D, L	aktueller Zählwert, hexadezimal
DEZ	CV_BCD	WORD	E, A, M, D, L	aktueller Zählwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status des Zählers

Statuswort für: Z_RUECK	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird der Zähler `Z1` auf den Wert "250" gesetzt. Wenn der Signalzustand am Eingang `E7.7` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` um 1 vermindert. Wenn der Signalzustand am Eingang `E3.3` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` auf "0" gesetzt.

Wenn der Wert des Zählers `Z1` größer als "0" ist, ist `A2.0` = 1. Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zählwert.

4.5.5 Setzen des Zähleranfangswerts - SZ

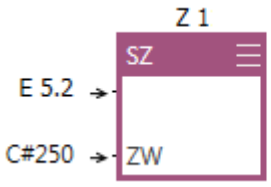
Die Operation "Setzen des Zähleranfangswerts" setzt den Zählwert eines Zählers. Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird der Zähler auf den Wert *ZW* gesetzt.

Deutsch - Englisch

Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	COUNTER	Z	Nummer des Zählers, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Zählwert setzen
ZW	PV	WORD	E, A, M, D, L oder Konstante	Zählwert, BCD-kodierte Zählerkonstante: C#<Wert>, <Wert> zwischen 0 und 999

Statuswort für: SZ	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	-	-	0

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird der Zähler `Z1` auf den Wert "250" gesetzt.

4.5.6 Vorwärtszählen - ZV

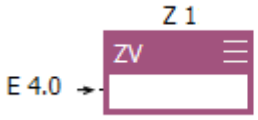
Die Operation "Vorwärtszählen" erhöht den Zählwert eines Zählers. Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird der Zählwert um 1 erhöht. Der Zählwert wird nicht weiter erhöht, wenn er "999" erreicht hat.

Deutsch - Englisch

Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	COUNTER	Z	Nummer des Zählers, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Vorwärtszählen

Statuswort für: ZV	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



Wenn der Signalzustand am Eingang `E4.0` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` um 1 erhöht.

4.5.7 Rückwärtszählen - ZR

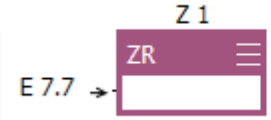
Die Operation "Rückwärtszählen" vermindert den Zählwert eines Zählers. Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird der Zählwert um 1 vermindert. Der Zählwert wird nicht weiter vermindert, wenn er "0" erreicht hat.

Deutsch - Englisch

Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	COUNTER	Z	Nummer des Zählers, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Rückwärtszählen

Statuswort für: ZR	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	-	-	0

Beispiel



Wenn der Signalzustand am Eingang `E7.7` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` um 1 vermindert.

4.6 Festpunkt-Funktionen

4.6.1 Überblick

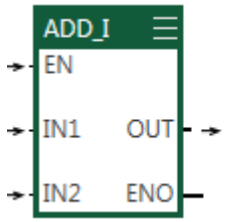
Mit den Festpunkt-Funktionen können Sie arithmetische Operationen mit zwei Ganzzahlen durchführen.

Operation	Festpunkt-Funktion
ADD_I	ganze Zahlen addieren (16 Bit)
SUB_I	ganze Zahlen subtrahieren (16 Bit)
MUL_I	ganze Zahlen multiplizieren (16 Bit)
DIV_I	ganze Zahlen dividieren (16 Bit)
ADD_DI	ganze Zahlen addieren (32 Bit)
SUB_DI	ganze Zahlen subtrahieren (32 Bit)
MUL_DI	ganze Zahlen multiplizieren (32 Bit)
DIV_DI	ganze Zahlen dividieren (32 Bit)
MOD_DI	Divisionsrest gewinnen (32 Bit)

4.6.2 Ganze Zahlen addieren (16 Bit) - ADD_I

Mit der Operation "Ganze Zahlen addieren (16 Bit)" können Sie zwei 16 Bit-Festpunktzahlen an den Eingängen *IN1* und *IN2* addieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

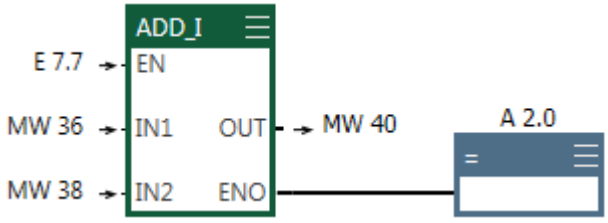
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (16 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	INT	E, A, M, D, L oder Konstante	erster Summand
IN2	INT	E, A, M, D, L oder Konstante	zweiter Summand
OUT	INT	E, A, M, D, L	Additionsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ADD_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

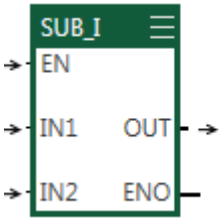


Wenn `E7.7` = 1 ist, werden die Ganzzahlen (16 Bit) in den Merkerwörtern `MW36` und `MW38` addiert. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Addition ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.6.3 Ganze Zahlen subtrahieren (16 Bit) - SUB_I

Mit der Operation "Ganze Zahlen subtrahieren (16 Bit)" können Sie 16 Bit-Festpunktzahlen subtrahieren. Dabei wird der Wert am Eingang *IN2* vom Wert am Eingang *IN1* subtrahiert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

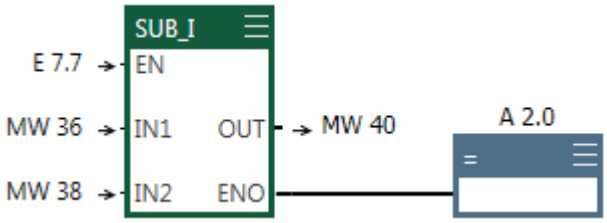
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (16 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	INT	E, A, M, D, L oder Konstante	Minuend
IN2	INT	E, A, M, D, L oder Konstante	Subtrahend
OUT	INT	E, A, M, D, L	Subtraktionsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SUB_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

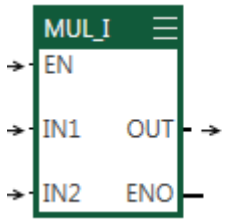


Wenn `E7.7` = 1 ist, wird die Ganzzahl (16 Bit) im Merkerwort `MW38` von der Ganzzahl (16 Bit) im Merkerwort `MW36` subtrahiert. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Subtraktion ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.6.4 Ganze Zahlen multiplizieren (16 Bit) - MUL_I

Mit der Operation "Ganze Zahlen multiplizieren (16 Bit)" können Sie zwei 16 Bit-Festpunktzahlen an den Eingängen *IN1* und *IN2* miteinander multiplizieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

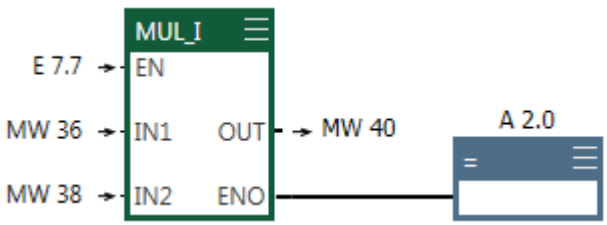
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (16 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	INT	E, A, M, D, L oder Konstante	Multiplikand
IN2	INT	E, A, M, D, L oder Konstante	Multiplikator
OUT	INT	E, A, M, D, L	Multiplikationsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: MUL_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

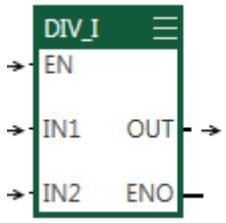


Wenn E7.7 = 1 ist, werden die Ganzzahlen (16 Bit) in den Merkerwörtern MW36 und MW38 miteinander multipliziert. Das Ergebnis wird im Merkerwort MW40 gespeichert. Wenn die Multiplikation ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

4.6.5 Ganze Zahlen dividieren (16 Bit) - DIV_I

Mit der Operation "Ganze Zahlen dividieren (16 Bit)" können Sie 16 Bit-Festpunktzahlen dividieren. Dabei wird der Wert am Eingang *IN1* durch den Wert am Eingang *IN2* dividiert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis (Quotient) wird im Ausgang *OUT* gespeichert. Der Divisionsrest wird nicht gespeichert.

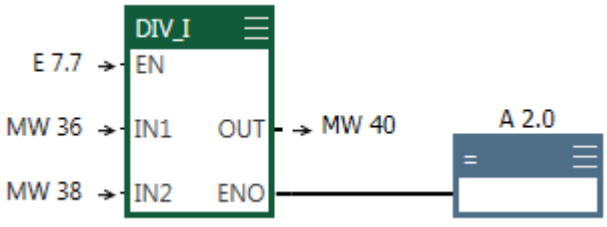
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (16 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	INT	E, A, M, D, L oder Konstante	Dividend
IN2	INT	E, A, M, D, L oder Konstante	Divisor
OUT	INT	E, A, M, D, L	Divisionsergebnis (Quotient)
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: DIV_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

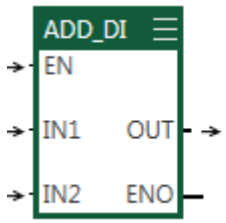


Wenn E7.7 = 1 ist, wird die Ganzzahl (16 Bit) im Merkerwort MW36 durch die Ganzzahl (16 Bit) im Merkerwort MW38 dividiert. Das Ergebnis (Quotient) wird im Merkerwort MW40 gespeichert. Wenn die Division ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

4.6.6 Ganze Zahlen addieren (32 Bit) - ADD_DI

Mit der Operation "Ganze Zahlen addieren (32 Bit)" können Sie zwei 32 Bit-Festpunktzahlen an den Eingängen *IN1* und *IN2* addieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

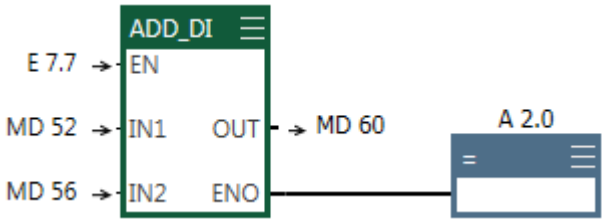
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (32 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	DINT	E, A, M, D, L oder Konstante	erster Summand
IN2	DINT	E, A, M, D, L oder Konstante	zweiter Summand
OUT	DINT	E, A, M, D, L	Additionsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ADD_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

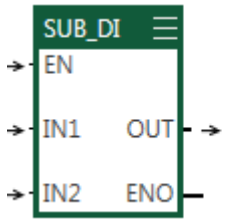


Wenn `E7.7` = 1 ist, werden die Ganzzahlen (32 Bit) in den Merkerdoppelwörtern `MD52` und `MD56` addiert. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Addition ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.6.7 Ganze Zahlen subtrahieren (32 Bit) - SUB_DI

Mit der Operation "Ganze Zahlen subtrahieren (32 Bit)" können Sie 32 Bit-Festpunktzahlen subtrahieren. Dabei wird der Wert am Eingang *IN2* vom Wert am Eingang *IN1* subtrahiert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

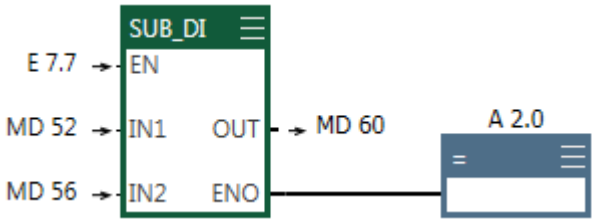
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (32 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	DINT	E, A, M, D, L oder Konstante	Minuend
IN2	DINT	E, A, M, D, L oder Konstante	Subtrahend
OUT	INT	E, A, M, D, L	Subtraktionsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SUB_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

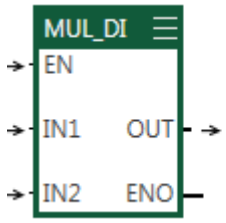


Wenn `E7.7` = 1 ist, wird die Ganzzahl (32 Bit) im Merkerdoppelwort `MD56` von der Ganzzahl (32 Bit) im Merkerdoppelwort `MD52` subtrahiert. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Subtraktion ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.6.8 Ganze Zahlen multiplizieren (32 Bit) - MUL_DI

Mit der Operation "Ganze Zahlen multiplizieren (32 Bit)" können Sie zwei 32 Bit-Festpunktzahlen an den Eingängen *IN1* und *IN2* miteinander multiplizieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

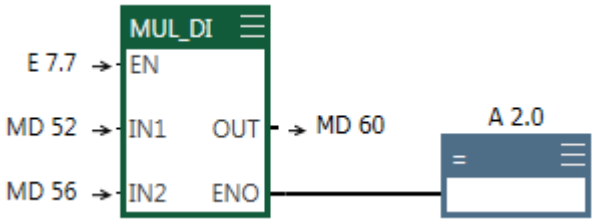
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (32 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	DINT	E, A, M, D, L oder Konstante	Multiplikand
IN2	DINT	E, A, M, D, L oder Konstante	Multiplikator
OUT	DINT	E, A, M, D, L	Multiplikationsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: MUL_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

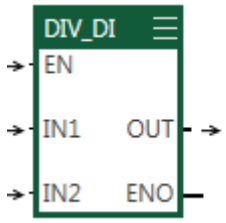


Wenn `E7.7` = 1 ist, werden die Ganzzahlen (32 Bit) in den Merkerdoppelwörtern `MD52` und `MD56` miteinander multipliziert. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Multiplikation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.6.9 Ganze Zahlen dividieren (32 Bit) - DIV_DI

Mit der Operation "Ganze Zahlen dividieren (32 Bit)" können Sie 32 Bit-Festpunktzahlen dividieren. Dabei wird der Wert am Eingang *IN1* durch den Wert am Eingang *IN2* dividiert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis (Quotient) wird im Ausgang *OUT* gespeichert. Der Divisionsrest wird nicht gespeichert.

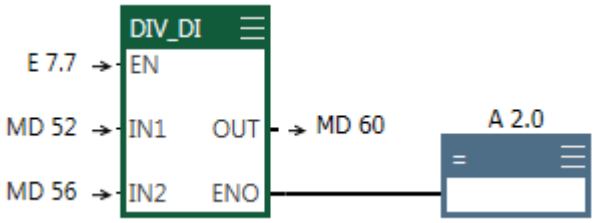
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (32 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	DINT	E, A, M, D, L oder Konstante	Dividend
IN2	DINT	E, A, M, D, L oder Konstante	Divisor
OUT	DINT	E, A, M, D, L	Divisionsergebnis (Quotient)
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: DIV_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

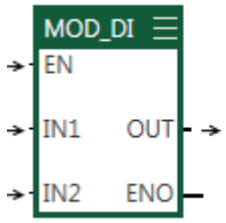


Wenn `E7.7` = 1 ist, wird die Ganzzahl (32 Bit) im Merkerdoppelwort `MD52` durch die Ganzzahl (32 Bit) im Merkerdoppelwort `MD56` dividiert. Das Ergebnis (Quotient) wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Division ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.6.10 Divisionsrest gewinnen (32 Bit) - MOD_DI

Mit der Operation "Divisionsrest gewinnen (32 Bit)" können Sie 32 Bit-Festpunktzahlen dividieren. Als Ergebnis erhalten Sie den Divisionsrest. Dabei wird der Wert am Eingang *IN1* durch den Wert am Eingang *IN2* dividiert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis (Divisionsrest) wird im Ausgang *OUT* gespeichert.

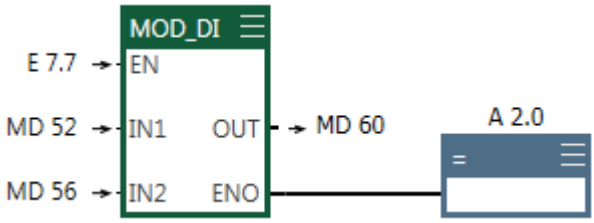
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (32 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	INT	E, A, M, D, L oder Konstante	Dividend
IN2	INT	E, A, M, D, L oder Konstante	Divisor
OUT	INT	E, A, M, D, L	Divisionsrest
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: MOD_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel



Wenn `E7.7` = 1 ist, wird die Ganzzahl (32 Bit) im Merkerdoppelwort `MD52` durch die Ganzzahl (32 Bit) im Merkerdoppelwort `MD56` dividiert. Das Ergebnis (Divisionsrest) wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Division ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.7 Gleitpunkt-Funktionen

4.7.1 Überblick

Mit den Gleitpunkt-Funktionen können Sie arithmetische und trigonometrische Operationen mit einer bzw. zwei Gleitpunktzahlen durchführen.

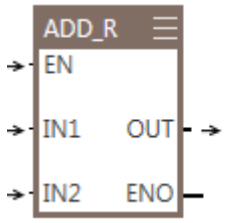
Operation	Gleitpunkt-Funktion
ADD_R	Gleitpunktzahlen addieren
SUB_R	Gleitpunktzahlen subtrahieren
MUL_R	Gleitpunktzahlen multiplizieren
DIV_R	Gleitpunktzahlen dividieren
ABS	Absolutwert einer Gleitpunktzahl bilden

Operation	Gleitpunkt-Funktion
SQRT	Quadratwurzel einer Gleitpunktzahl ziehen
SQR	Quadrat einer Gleitpunktzahl bilden
LN	natürlichen Logarithmus einer Gleitpunktzahl bilden
EXP	Exponentialwert einer Gleitpunktzahl bilden
SIN	Sinus einer Gleitpunktzahl bilden
COS	Cosinus einer Gleitpunktzahl bilden
TAN	Tangens einer Gleitpunktzahl bilden
ASIN	Arkussinus einer Gleitpunktzahl bilden
ACOS	Arkuscosinus einer Gleitpunktzahl bilden
ATAN	Arkustangens einer Gleitpunktzahl bilden

4.7.2 Gleitpunktzahlen addieren - ADD_R

Mit der Operation "Gleitpunktzahlen addieren" können Sie zwei Gleitpunktzahlen an den Eingängen *IN1* und *IN2* addieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

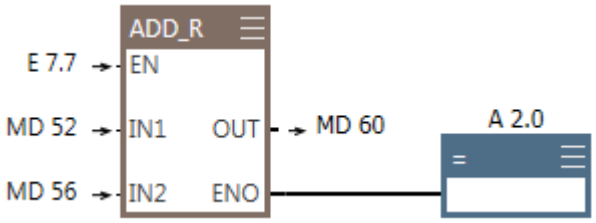
Wenn das Ergebnis oder einer der Eingänge keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	REAL	E, A, M, D, L oder Konstante	erster Summand
IN2	REAL	E, A, M, D, L oder Konstante	zweiter Summand
OUT	REAL	E, A, M, D, L	Additionsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ADD_R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

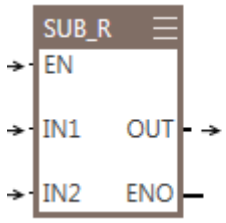


Wenn `E7.7` = 1 ist, werden die Gleitpunktzahlen in den Merkerdoppelwörtern `MD52` und `MD56` addiert. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Addition ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.7.3 Gleitpunktzahlen subtrahieren - SUB_R

Mit der Operation "Gleitpunktzahlen subtrahieren" können Sie Gleitpunktzahlen subtrahieren. Dabei wird der Wert am Eingang *IN2* vom Wert am Eingang *IN1* subtrahiert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

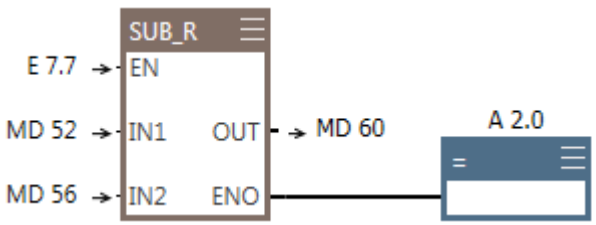
Wenn das Ergebnis oder einer der Eingänge keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	REAL	E, A, M, D, L oder Konstante	Minuend
IN2	REAL	E, A, M, D, L oder Konstante	Subtrahend
OUT	REAL	E, A, M, D, L	Subtraktionsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SUB_R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

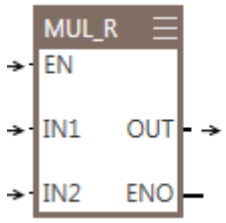


Wenn E7.7 = 1 ist, wird die Gleitpunktzahl im Merkerdoppelwort MD56 von der Gleitpunktzahl im Merkerdoppelwort MD52 subtrahiert. Das Ergebnis wird im Merkerdoppelwort MD60 gespeichert. Wenn die Subtraktion ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

4.7.4 Gleitpunktzahlen multiplizieren - MUL_R

Mit der Operation "Ganze Zahlen multiplizieren (32 Bit)" können Sie zwei 32 Bit-Festpunktzahlen an den Eingängen *IN1* und *IN2* miteinander multiplizieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

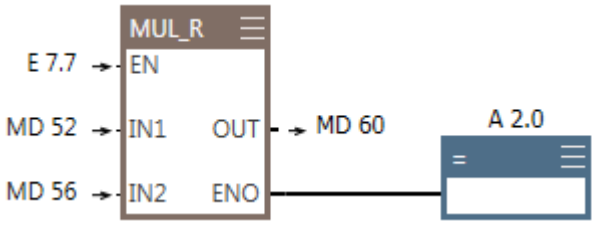
Wenn das Ergebnis oder einer der Eingänge keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	REAL	E, A, M, D, L oder Konstante	Multiplikand
IN2	REAL	E, A, M, D, L oder Konstante	Multiplikator
OUT	REAL	E, A, M, D, L	Multiplikationsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: MUL_R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

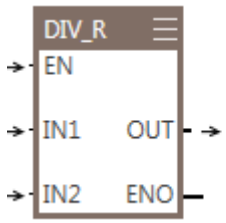


Wenn E7.7 = 1 ist, werden die Gleitpunktzahlen in den Merkerdoppelwörtern MD52 und MD56 miteinander multipliziert. Das Ergebnis wird im Merkerdoppelwort MD60 gespeichert. Wenn die Multiplikation ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

4.7.5 Gleitpunktzahlen dividieren - DIV_R

Mit der Operation "Gleitpunktzahlen dividieren" können Sie Gleitpunktzahlen dividieren. Dabei wird der Wert am Eingang *IN1* durch den Wert am Eingang *IN2* dividiert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

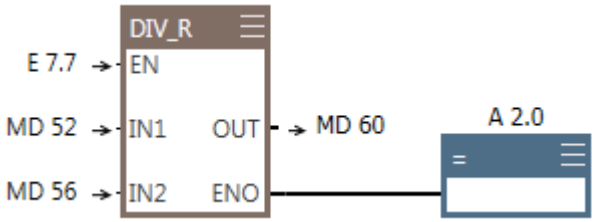
Wenn das Ergebnis oder einer der Eingänge keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	REAL	E, A, M, D, L oder Konstante	Dividend
IN2	REAL	E, A, M, D, L oder Konstante	Divisor
OUT	REAL	E, A, M, D, L	Divisionsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: DIV_R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

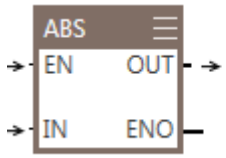
Beispiel



Wenn `E7.7` = 1 ist, wird die Gleitpunktzahl im Merkerdoppelwort `MD52` durch die Gleitpunktzahl im Merkerdoppelwort `MD56` dividiert. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Division ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.7.6 Absolutwert einer Gleitpunktzahl - ABS

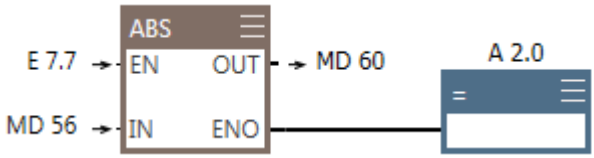
Mit der Operation "Absolutwert einer Gleitpunktzahl" können Sie den Absolutwert einer Gleitpunktzahl am Eingang *IN* bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Absolutwert der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ABS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel



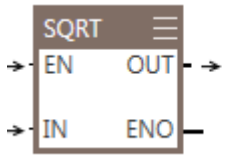
Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der Absolutwert gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel: `MD56` = -25,75, `MD60` = 25,75

4.7.7 Quadratwurzel einer Gleitpunktzahl - SQRT

Mit der Operation "Quadratwurzel einer Gleitpunktzahl" können Sie die Quadratwurzel einer Gleitpunktzahl am Eingang *IN* ziehen. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

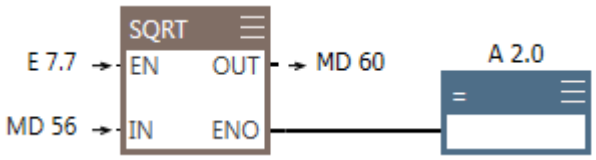
Wenn das Ergebnis oder der Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Quadratwurzel der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SQRT	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

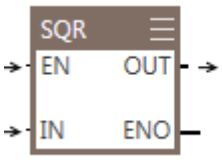


Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` die Quadratwurzel gezogen. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.7.8 Quadrat einer Gleitpunktzahl - SQR

Mit der Operation "Quadrat einer Gleitpunktzahl" können Sie eine Gleitpunktzahl am Eingang *IN* quadrieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

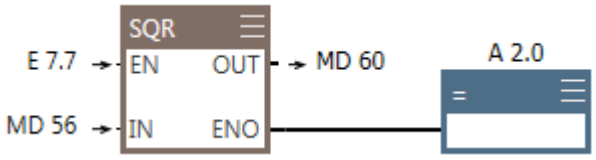
Wenn das Ergebnis oder der Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Quadrat der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SQR	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

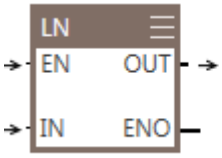


Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` das Quadrat gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.7.9 Natürlichen Logarithmus einer Gleitpunktzahl - LN

Mit der Operation "Natürlichen Logarithmus einer Gleitpunktzahl" können Sie den natürlichen Logarithmus einer Gleitpunktzahl am Eingang *IN* bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

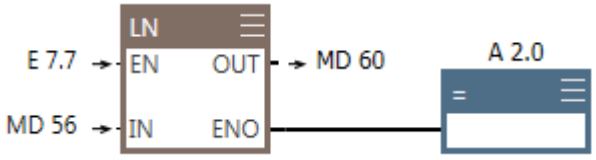
Wenn das Ergebnis oder der Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Zahl
OUT	REAL	E, A, M, D, L	natürlicher Logarithmus der Zahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: LN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

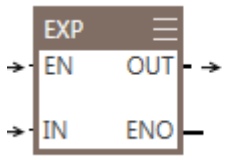


Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der natürliche Logarithmus gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.7.10 Exponentialwert einer Gleitpunktzahl - EXP

Mit der Operation "Exponentialwert einer Gleitpunktzahl" können Sie den Exponentialwert einer Gleitpunktzahl am Eingang *IN* auf Basis der Eulerschen Zahl *e* bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

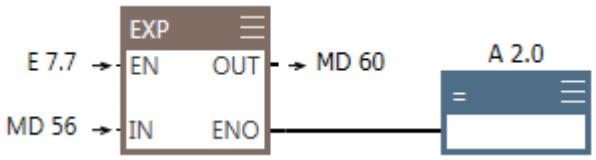
Wenn das Ergebnis oder der Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Zahl
OUT	REAL	E, A, M, D, L	Exponent der Zahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: EXP	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

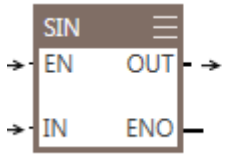
Beispiel



Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der Exponentialwert gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.7.11 Sinus einer Gleitpunktzahl - SIN

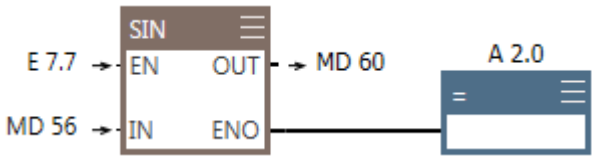
Mit der Operation "Sinus einer Gleitpunktzahl" können Sie von einem Winkel (Bogenmaß) am Eingang *IN* den Sinus (sin) bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Sinus der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SIN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

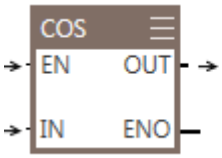
Beispiel



Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der Sinus gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.7.12 Cosinus einer Gleitpunktzahl - COS

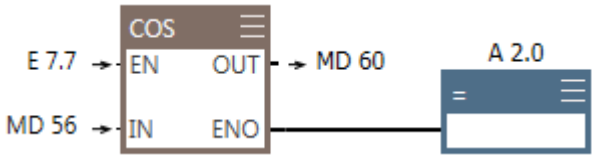
Mit der Operation "Cosinus einer Gleitpunktzahl" können Sie von einem Winkel (Bogenmaß) am Eingang *IN* den Cosinus (cos) bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Cosinus der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: COS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

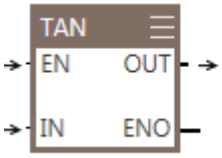
Beispiel



Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der Cosinus gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.7.13 Tangens einer Gleitpunktzahl - TAN

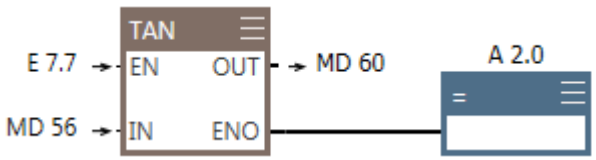
Mit der Operation "Tangens einer Gleitpunktzahl" können Sie von einem Winkel (Bogenmaß) am Eingang *IN* den Tangens (tan) bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Tangens der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: TAN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

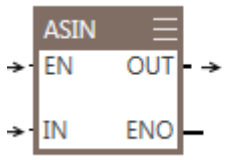


Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der Tangens gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.7.14 Arkussinus einer Gleitpunktzahl - ASIN

Mit der Operation "Arkussinus einer Gleitpunktzahl" können Sie aus einer Gleitpunktzahl am Eingang *IN* den Arkussinus ($\sin^{-1}$) bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis ist ein Winkel im Bogenmaß und wird im Ausgang *OUT* gespeichert.

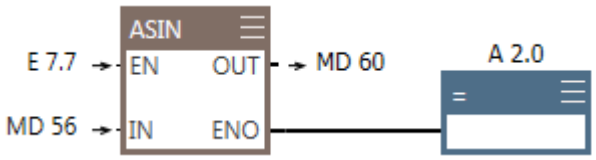
Der Ausgabewert liegt im folgenden Bereich: $-\frac{1}{2}\pi \leq$ Arkussinus $\leq +\frac{1}{2}\pi$



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Arkussinus der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ASIN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

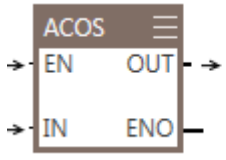


Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der Arkussinus gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.7.15 Arkuscosinus einer Gleitpunktzahl - ACOS

Mit der Operation "Arkuscosinus einer Gleitpunktzahl" können Sie aus einer Gleitpunktzahl am Eingang *IN* den Arkuscosinus ($\cos^{-1}$) bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis ist ein Winkel im Bogenmaß und wird im Ausgang *OUT* gespeichert.

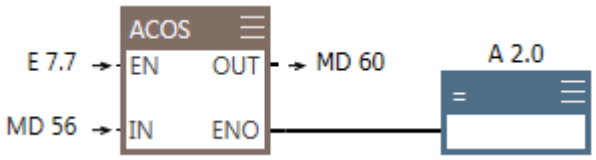
Der Ausgabewert liegt im folgenden Bereich: $0 \leq$ Arkuscosinus $\leq +\pi$



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Arkuscosinus der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ACOS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

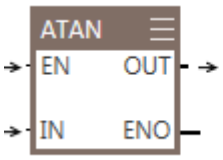


Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der Arkuscosinus gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.7.16 Arkustangens einer Gleitpunktzahl - ATAN

Mit der Operation "Arkustangens einer Gleitpunktzahl" können Sie aus einer Gleitpunktzahl am Eingang *IN* den Arkustangens ($\tan^{-1}$) bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis ist ein Winkel im Bogenmaß und wird im Ausgang *OUT* gespeichert.

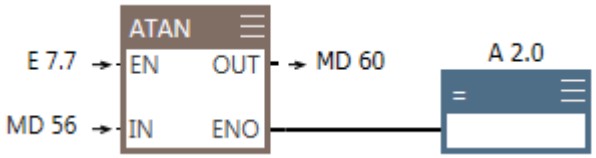
Der Ausgabewert liegt im folgenden Bereich: $-\frac{1}{2}\pi \leq$ Arkustangens $\leq +\frac{1}{2}\pi$



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Arkustangens der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ATAN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel



Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der Arkustangens gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

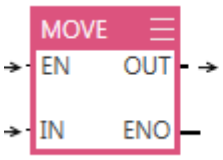
4.8 Verschieben

4.8.1 Wert übertragen - MOVE

Mit der Operation "Wert übertragen" können Sie den Wert am Eingang *IN* in den Ausgangsoperanden *OUT* kopieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Sie können alle elementaren Datentypen mit einer Länge von 8, 16 oder 32 Bit verwenden, z. B. BYTE, INT, WORD, REAL. Erweiterte Datentypen, wie z. B. Felder oder Strukturen, können jedoch nicht mit dieser Operation, sondern nur über die Systemfunktion "SFC20 BLKMOV" kopiert werden.

Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.

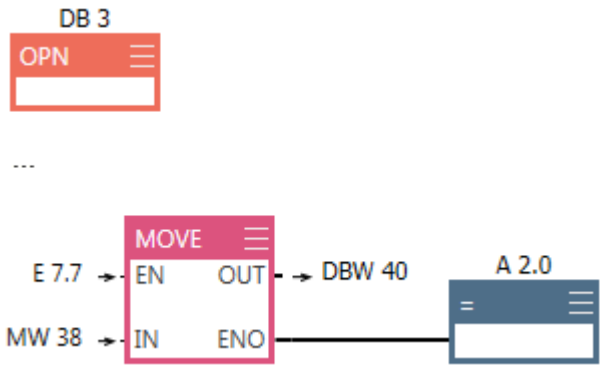
Wenn der Datentyp am Eingang **mehr** Bitstellen hat als der am Ausgang, werden die höherwertigen Bits abgeschnitten. Wenn der Datentyp am Eingang **weniger** Bitstellen hat als der am Ausgang, werden die höherwertigen Bits mit Nullen aufgefüllt.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	Elementare Datentypen (8, 16 oder 32 Bit)	E, A, M, D, L oder Konstante	Eingabewert
OUT	Elementare Datentypen (8, 16 oder 32 Bit)	E, A, M, D, L	Zieladresse
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: MOVE	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	-	-	-	-	0	1	1	1

Beispiel



Im ersten Netzwerk wird der Datenbaustein `DB3` mit der Operation "OPN" geöffnet.

Wenn `E7.7` = 1 ist, wird der Inhalt vom Merkerwort `MW38` in das Datenwort `DBW40` des Datenbausteins `DB3` kopiert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.9 Programmsteuerung

4.9.1 Überblick

Mit den Operationen zur Programmsteuerung können Sie Bausteine aufrufen und verlassen sowie innerhalb eines Bausteins an eine Sprungmarke springen.

Operation	Programmsteuerung
CALL	Funktion aufrufen (FC/SFC)
RET	aktuellen Baustein verlassen
OPN	Datenbaustein öffnen
JMP	zu einer Sprungmarke springen (absolut oder wenn 1)
JMPN	zu einer Sprungmarke springen (wenn 0)

Mit der Funktion "Sprungmarke - Label" können Sie das Ziel für die Sprungoperationen "JMP" bzw. "JMPN" setzen.

4.9.2 Baustein aufrufen - CALL

Mit der Operation "Baustein aufrufen" können Sie Funktionen (FC) oder Systemfunktion (SFC) aufrufen. Die Programmbearbeitung wird in der aufgerufenen Funktion bzw. dem aufgerufenen Funktionsbaustein fortgesetzt.

Parameter	Datentyp	Speicherbereich	Beschreibung
Nr.	-	-	Nummer des FC oder SFC, Bereich ist abhängig von der CPU

Aufruf ohne Bedingung (absoluter Aufruf)

Wenn Sie am Eingang des Aufruf-Elements keinen Parameter angeben (keine Verknüpfung vor dem Element), wird der Baustein ohne Bedingung aufgerufen.

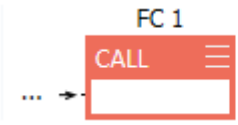
Statuswort für: CALL	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	0	0	1	-	0

Aufruf, wenn VKE = 1 (bedingter Aufruf)

Wenn vor dem Aufruf-Element Verknüpfungen angeben sind und das Verknüpfungsergebnis VKE = 1 ist, wird der Baustein aufgerufen.

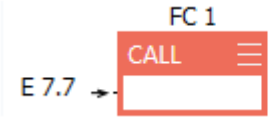
Statuswort für: CALL	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	0	0	1	1	0

Beispiel (absoluter Aufruf)



Am Eingang des Aufruf-Elements ist kein Parameter angeben (keine Verknüpfung vor dem Element). Die Funktion `FC1` wird immer aufgerufen. Die Programmbearbeitung wird im Baustein `FC1` fortgesetzt.

Beispiel (bedingter Aufruf)



Nur wenn `E7.7` = 1 ist, wird die Funktion `FC1` aufgerufen. Die Programmbearbeitung wird im Baustein `FC1` fortgesetzt.

4.9.3 Springe zurück - RET

Mit der Operation "Springe zurück" können Sie den aktuellen Baustein verlassen.

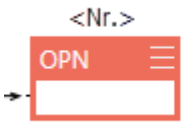
Statuswort für: DB	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	-	-	-	-	-	-

Beispiel

Wenn `E7.7` = 1 ist, wird der Baustein verlassen.

4.9.4 Datenbaustein öffnen - OPN

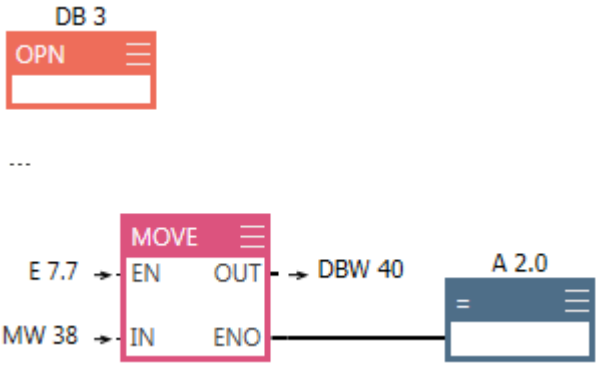
Mit der Operation "Datenbaustein öffnen" können Sie einen Datenbaustein als globalen Datenbaustein (DB) oder als Instanz-Datenbaustein (DI) öffnen, um anschließend darauf zuzugreifen. Die Operation überträgt die DB-Nummer in das DB- bzw. DI-Register. Alle folgenden DB- bzw. DI-Operationen greifen auf den geöffneten Baustein zu.



Parameter	Datentyp	Speicherbereich	Beschreibung
Nr.	-	-	Nummer des DB oder DI, Bereich ist abhängig von der CPU

Statuswort für: DB	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	-	-	-	-

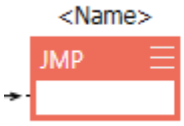
Beispiel



Im ersten Netzwerk wird der Datenbaustein `DB3` geöffnet. Alle folgenden Datenbaustein-Operation beziehen sich auf den geöffneten Datenbaustein `DB3`. Wenn `E7.7` = 1 ist, wird der Inhalt vom Merkerwort `MW38` in das Datenwort `DBW40` des Datenbausteins `DB3` kopiert.

4.9.5 Zur Sprungmarke springen (absolut oder wenn 1) - JMP

Mit der Operation "Zur Sprungmarke springen (absolut oder wenn 1)" können Sie innerhalb des aktuellen Bausteins zu der im *Operand* angegebenen Sprungmarke springen, um die Programmbearbeitung dort fortzusetzen.



Parameter	Datentyp	Speicherbereich	Beschreibung
Name	-	-	Sprungmarke, zu der absolut oder bei VKE = 1 gesprungen wird

Sprung ohne Bedingung (absoluter Sprung)

Wenn Sie am Eingang des Sprung-Elements keinen Parameter angeben (keine Verknüpfung vor dem Element), wird ohne Bedingung zur Sprungmarke gesprungen.

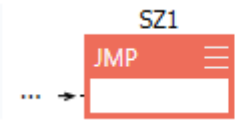
Statuswort für: JMP	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	-	-	-	-

Sprung, wenn VKE = 1 (bedingter Sprung)

Wenn vor dem Sprung-Element Verknüpfungen angeben sind und das Verknüpfungsergebnis VKE = 1 ist, wird zur Sprungmarke gesprungen.

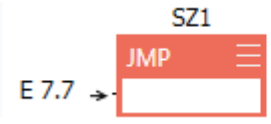
Statuswort für: JMP	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	1	1	0

Beispiel (absoluter Sprung)



Am Eingang des Sprung-Elements ist kein Parameter angeben (keine Verknüpfung vor dem Element). Der Sprung zur Sprungmarke `SZ1` wird immer ausgeführt.

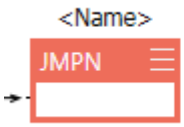
Beispiel (bedingter Sprung)



Nur wenn `E7.7` = 1 ist, wird zur Sprungmarke `SZ1` gesprungen.

4.9.6 Zur Sprungmarke springen (wenn 0) - JMPN

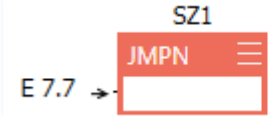
Mit der Operation "Zur Sprungmarke springen (wenn 0)" können Sie innerhalb des aktuellen Bausteins zu der im *Operand* angegebenen Sprungmarke verzweigen, um die Programmbearbeitung dort fortzusetzen. Wenn das Verknüpfungsergebnis VKE vor dem Sprung-Element gleich "0" ist, wird zur Sprungmarke gesprungen.



Parameter	Datentyp	Speicherbereich	Beschreibung
Name	-	-	Sprungmarke, zu der bei VKE = 0 gesprungen wird

Statuswort für: JMPN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	1	1	0

Beispiel

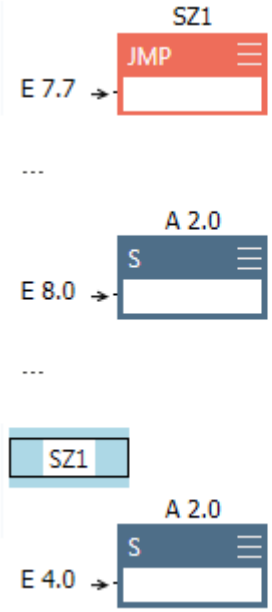


Nur wenn `E7.7` = 0 ist, wird zur Sprungmarke `SZ1` gesprungen.

4.9.7 Sprungmarke - Label

Die Sprungmarke bezeichnet das Ziel der Sprungoperationen "JMP" bzw. "JMPN". Das in *Label* angegebene Sprungziel darf aus maximal vier Zeichen bestehen. Das erste Zeichen muss ein Buchstabe sein, die anderen Zeichen dürfen Buchstaben oder Zahlen sein.

Beispiel



Nur wenn `E7.7` im ersten Netzwerk gleich "1" ist, wird zur Sprungmarke `SZ1` gesprungen. Die Programmbearbeitung wird mit der Setze-Operation fortgesetzt, die den Eingang `E4.0` auswertet.

Wenn jedoch `E7.7` gleich "0" ist, wird **nicht** zur Sprungmarke `SZ1` gesprungen. Die Programmbearbeitung wird mit der Setze-Operation im nächsten Netzwerk fortgesetzt - hier die Operation, die den Eingang `E8.0` auswertet.

4.10 Schieben/Rotieren

4.10.1 Überblick

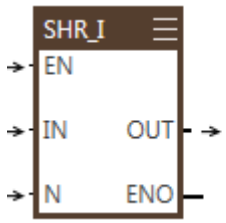
Mit den Schiebe-/Rotier-Operationen können Sie die Bits eines Operanden nach links oder rechts schieben bzw. rotieren.

Operation	Schieben/Rotieren
SHR_I	Ganzzahl (16 Bit) nach rechts schieben
SHR_DI	Ganzzahl (32 Bit) nach rechts schieben
SHL_W	Bitfolge (16 Bit) nach links schieben
SHR_W	Bitfolge (16 Bit) nach rechts schieben
SHL_DW	Bitfolge (32 Bit) nach links schieben
SHR_DW	Bitfolge (32 Bit) nach rechts schieben
ROL_DW	Bitfolge (32 Bit) nach links rotieren
ROR_DW	Bitfolge (32 Bit) nach rechts rotieren

4.10.2 Ganzzahl (16 Bit) rechts schieben - SHR_I

Mit der Operation "Ganzzahl (16 Bit) rechts schieben" können Sie die Bits 0 - 15 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach rechts schieben. Die Stellen, die durch die Schiebeoperation frei werden, werden entweder mit Nullen oder mit dem Signalzustand des Vorzeichenbits aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

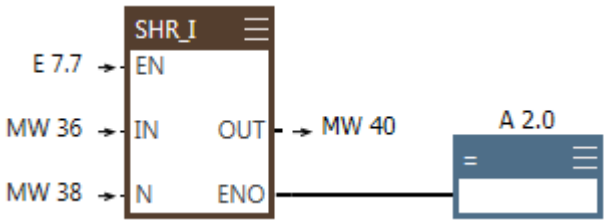
Die Bits am Eingang *IN* werden um maximal 16 Bitstellen geschoben, auch wenn der Wert *N* größer als "16" ist. Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	INT	E, A, M, D, L	Ganzzahl (16 Bit), die geschoben werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die geschoben werden soll, maximal 16
OUT	INT	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SHR_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel

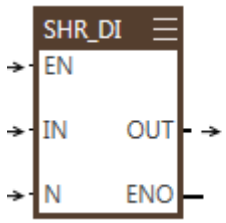


Wenn `E7.7` = 1 ist, wird die Ganzzahl (16 Bit) im Merkerwort `MW36` um die Anzahl an Bits nach rechts geschoben, die im Merkerwort `MW38` angegeben ist. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.10.3 Ganzzahl (32 Bit) rechts schieben - SHR_DI

Mit der Operation "Ganzzahl (32 Bit) rechts schieben" können Sie die Bits 0 - 32 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach rechts schieben. Die Stellen, die durch die Schiebeoperation frei werden, werden entweder mit Nullen oder mit dem Signalzustand des Vorzeichenbits aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

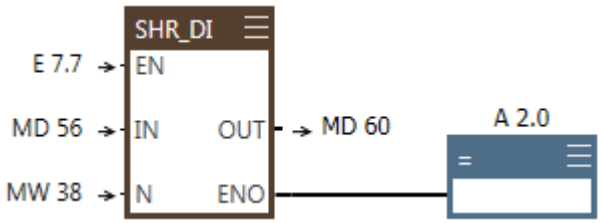
Die Bits am Eingang *IN* werden um maximal 32 Bitstellen geschoben, auch wenn der Wert *N* größer als "32" ist. Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	WORD	E, A, M, D, L	Ganzzahl (32 Bit), die geschoben werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die geschoben werden soll, maximal 32
OUT	WORD	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SHR_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel

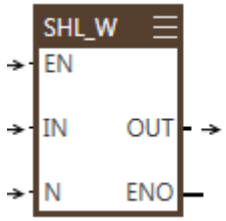


Wenn `E7.7` = 1 ist, wird die Ganzzahl (32 Bit) im Merkerdoppelwort `MD56` um die Anzahl an Bits nach rechts geschoben, die im Merkerwort `MW38` angegeben ist. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.10.4 16 Bit links schieben - SHL_W

Mit der Operation "16 Bit links schieben" können Sie die Bits 0 - 15 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach links schieben. Die Stellen, die durch die Schiebeoperation frei werden, werden mit Nullen aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

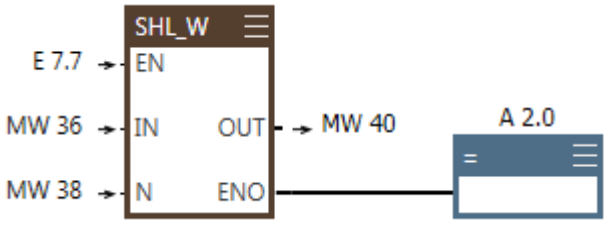
Wenn der Wert *N* größer als "16" ist, wird der Ausgang *OUT* auf den Wert "0" gesetzt. Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	WORD	E, A, M, D, L	Bitfolge (16 Bit), die geschoben werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die geschoben werden soll, maximal 16
OUT	WORD	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SHL_W	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel

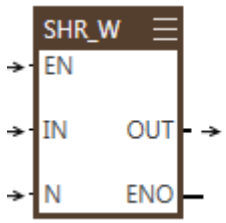


Wenn `E7.7` = 1 ist, wird die Bitfolge (16 Bit) im Merkerwort `MW36` um die Anzahl an Bits nach links geschoben, die im Merkerwort `MW38` angegeben ist. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.10.5 16 Bit rechts schieben - SHR_W

Mit der Operation "16 Bit rechts schieben" können Sie die Bits 0 - 15 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach rechts schieben. Die Stellen, die durch die Schiebeoperation frei werden, werden mit Nullen aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

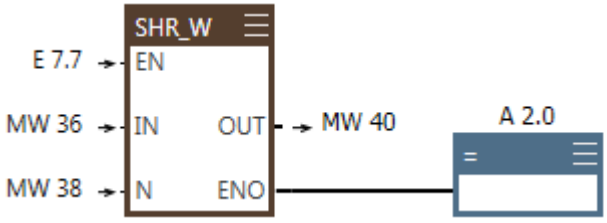
Wenn der Wert *N* größer als "16" ist, wird der Ausgang *OUT* auf den Wert "0" gesetzt. Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	WORD	E, A, M, D, L	Bitfolge (16 Bit), die geschoben werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die geschoben werden soll, maximal 16
OUT	WORD	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SHR_W	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel

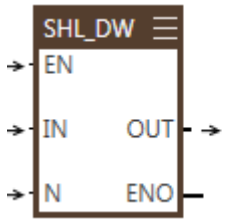


Wenn `E7.7` = 1 ist, wird die Bitfolge (16 Bit) im Merkerwort `MW36` um die Anzahl an Bits nach rechts geschoben, die im Merkerwort `MW38` angegeben ist. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.10.6 32 Bit links schieben - SHL_DW

Mit der Operation "32 Bit links schieben" können Sie die Bits 0 - 31 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach links schieben. Die Stellen, die durch die Schiebeoperation frei werden, werden mit Nullen aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

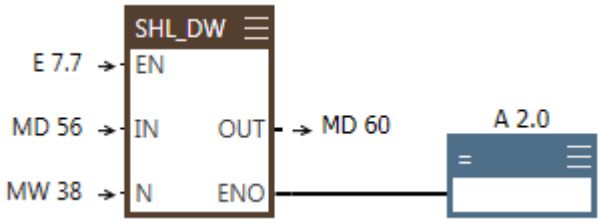
Wenn der Wert *N* größer als "32" ist, wird der Ausgang *OUT* auf den Wert "0" gesetzt. Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DWORD	E, A, M, D, L	Bitfolge (32 Bit), die geschoben werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die geschoben werden soll, maximal 32
OUT	DWORD	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SHL_DW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel

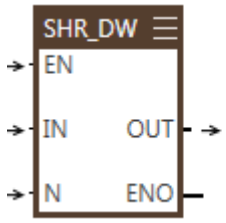


Wenn `E7.7` = 1 ist, wird die Bitfolge (32 Bit) im Merkerdoppelwort `MD56` um die Anzahl an Bits nach links geschoben, die im Merkerwort `MW38` angegeben ist . Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.10.7 32 Bit rechts schieben - SHR_DW

Mit der Operation "32 Bit rechts schieben" können Sie die Bits 0 - 31 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach rechts schieben. Die Stellen, die durch die Schiebeoperation frei werden, werden mit Nullen aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

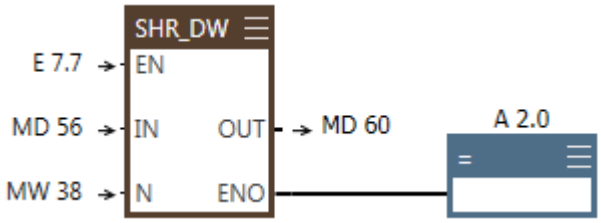
Wenn der Wert *N* größer als "32" ist, wird der Ausgang *OUT* auf den Wert "0" gesetzt. Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DWORD	E, A, M, D, L	Bitfolge (32 Bit), die geschoben werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die geschoben werden soll, maximal 32
OUT	DWORD	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SHR_DW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel

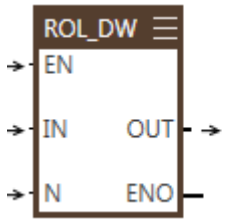


Wenn `E7.7` = 1 ist, wird die Bitfolge (32 Bit) im Merkerdoppelwort `MD56` um die Anzahl an Bits nach rechts geschoben, die im Merkerwort `MW38` angegeben ist. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.10.8 32 Bit links rotieren - ROL_DW

Mit der Operation "32 Bit links rotieren" können Sie die Bits 0 - 31 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach links rotieren. Die Stellen, die durch die Rotier-Operation frei werden, werden mit den Signalzuständen der hinausgeschobenen (rotierenden) Bits aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

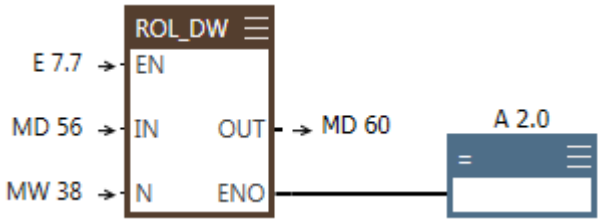
Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DWORD	E, A, M, D, L	Bitfolge (32 Bit), die rotiert werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die rotiert werden soll
OUT	DWORD	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ROL_DW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel

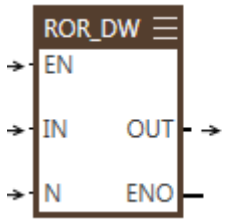


Wenn `E7.7` = 1 ist, wird die Bitfolge (32 Bit) im Merkerdoppelwort `MD56` um die Anzahl an Bits nach links rotiert, die im Merkerwort `MW38` angegeben ist. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.10.9 32 Bit rechts rotieren - ROR_DW

Mit der Operation "32 Bit rechts rotieren" können Sie die Bits 0 - 31 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach rechts rotieren. Die Stellen, die durch die Rotier-Operation frei werden, werden mit den Signalzuständen der hinausgeschobenen (rotierenden) Bits aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

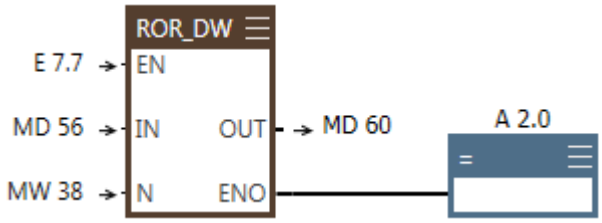
Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DWORD	E, A, M, D, L	Bitfolge (32 Bit), die rotiert werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die rotiert werden soll
OUT	DWORD	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ROR_DW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel



Wenn `E7.7` = 1 ist, wird die Bitfolge (32 Bit) im Merkerdoppelwort `MD56` um die Anzahl an Bits nach rechts rotiert, die im Merkerwort `MW38` angegeben ist. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

4.11 Zeiten

4.11.1 Überblick

In der CPU ist ein eigener Speicherbereich für Zeiten reserviert. Die Anzahl der Zeiten ist abhängig von der CPU.

Operation	Zeit
S_IMPULS	Zeit als Impuls parametrieren und starten
S_VIMP	Zeit als verlängerten Impuls parametrieren und starten
S_EVERZ	Zeit als Einschaltverzögerung parametrieren und starten
S_SEVERZ	Zeit als speichernde Einschaltverzögerung parametrieren und starten
S_AVERZ	Zeit als Ausschaltverzögerung parametrieren und starten

Operation	Zeit
SI	Zeit als Impuls starten
SV	Zeit als verlängerten Impuls starten
SE	Zeit als Einschaltverzögerung starten
SS	Zeit als speichernde Einschaltverzögerung starten
Sa	Zeit als Ausschaltverzögerung starten

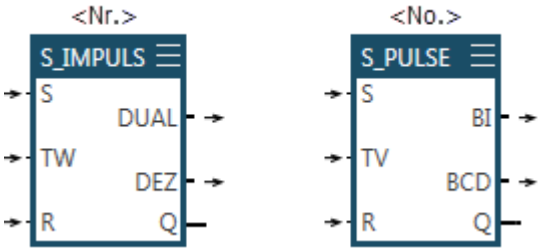
4.11.2 Zeit als Impuls parametrieren und starten - S_IMPULS

Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird die Operation "Zeit als Impuls parametrieren und starten" gestartet und der Ausgang *Q* wird auf den Signalzustand "1" gesetzt.

Der Ausgang *Q* wird erst dann wieder auf den Signalzustand "0" zurücksetzt, wenn mindestens einer der folgenden Zustände eintritt:

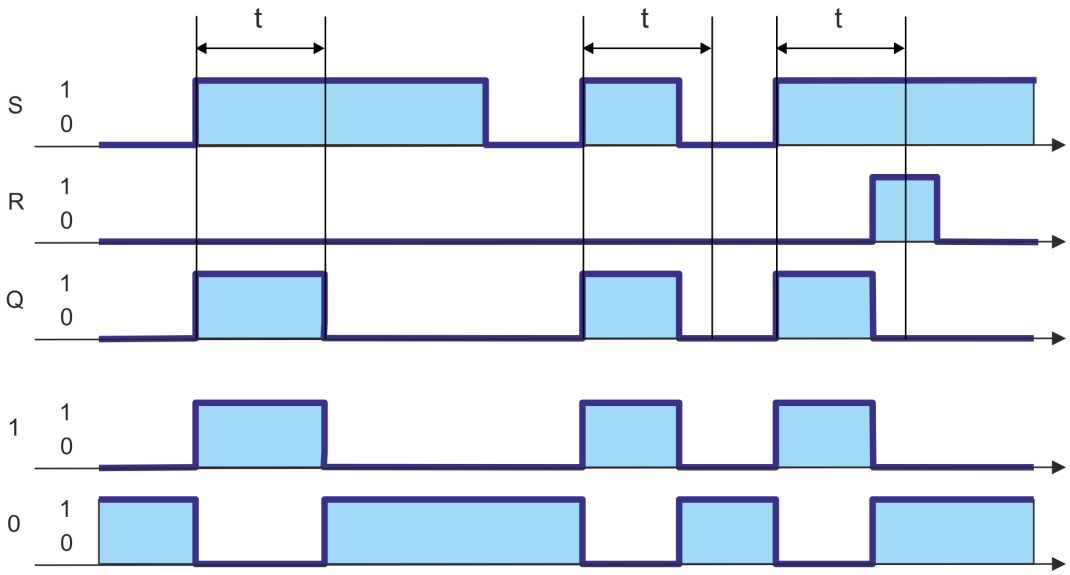
- Der Signalzustand am Eingang *S* wechselt von "1" auf "0".
- Die am Eingang *TW* voreingestellte Zeit ist abgelaufen.
- Der Signalzustand am Eingang *R* wechselt von "0" auf "1".

Deutsch - Englisch



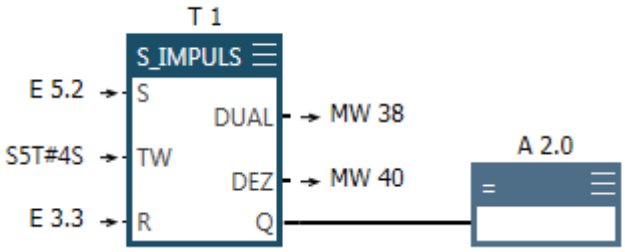
Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
S	S	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden
R	R	BOOL	E, A, M, D, L, T, Z	Rücksetzen
DUAL	BI	WORD	E, A, M, D, L	aktueller Zeitwert, Ganzzahlenformat
DEZ	BCD	WORD	E, A, M, D, L	aktueller Zeitwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status der Zeit

Statuswort für: S_IMPULS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1



t Programmierte Zeit
S VKE am Eingang *S*
R VKE am Eingang *R*
Q Signalzustand am Ausgang *Q*
1 Abfrage am Ausgang *Q* auf "1"
0 Abfrage am Ausgang *Q* auf "0"

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird für vier Sekunden auf den Wert "1" gesetzt.

Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden von "1" nach "0" wechselt, wird der Ausgang `A2.0` auf "0" zurückgesetzt.

Wenn der Signalzustand am Eingang `E3.3` vor Ablauf der vier Sekunden von "0" nach "1" wechselt, wird der Ausgang `A2.0` auf "0" zurückgesetzt.

Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zeitwert.

4.11.3 Zeit als verlängerten Impuls parametrieren und starten - S_VIMP

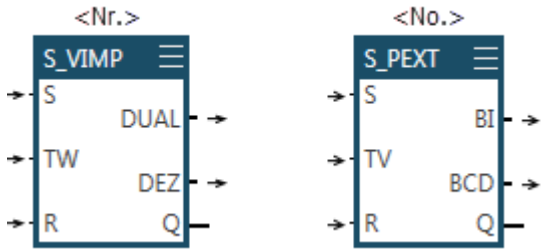
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird die Operation "Zeit als verlängerten Impuls parametrieren und starten" gestartet und der Ausgang *Q* wird auf den Signalzustand "1" gesetzt.

Der Ausgang *Q* wird erst dann wieder auf den Signalzustand "0" zurücksetzt, wenn mindestens einer der folgenden Zustände eintritt:

- Die am Eingang *TW* voreingestellte Zeit ist abgelaufen.
- Der Signalzustand am Eingang *R* wechselt von "0" auf "1".

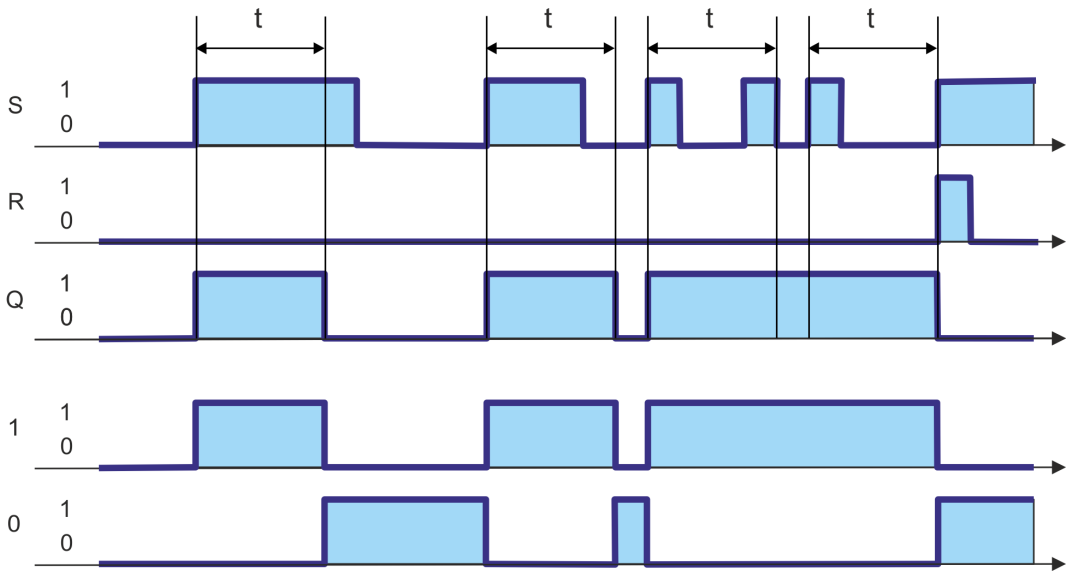
Wenn der Signalzustand am Eingang *S* erneut von "0" auf "1" wechselt, während die gestartete Zeit läuft, wird die Zeit mit dem Zeitwert am Eingang *TW* nochmals gestartet. Der Impuls wird verlängert.

Deutsch - Englisch



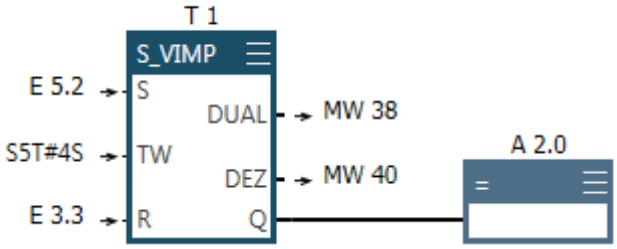
Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
S	S	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden
R	R	BOOL	E, A, M, D, L, T, Z	Rücksetzen
DUAL	BI	WORD	E, A, M, D, L	aktueller Zeitwert, Ganzzahlenformat
DEZ	BCD	WORD	E, A, M, D, L	aktueller Zeitwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status der Zeit

Statuswort für: S_VIMP	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1



t Programmierte Zeit
S VKE am Eingang *S*
R VKE am Eingang *R*
Q Signalzustand am Ausgang *Q*
1 Abfrage am Ausgang *Q* auf "1"
0 Abfrage am Ausgang *Q* auf "0"

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird für vier Sekunden auf den Wert "1" gesetzt.

Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden erneut von "0" nach "1" wechselt, wird die Zeit um weitere vier Sekunden verlängert und der Ausgang `A2.0` erst nach Ablauf dieser Zeit auf "0" zurückgesetzt.

Wenn der Signalzustand am Eingang `E3.3` vor Ablauf der vier Sekunden von "0" nach "1" wechselt, wird der Ausgang `A2.0` auf "0" zurückgesetzt.

Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zeitwert.

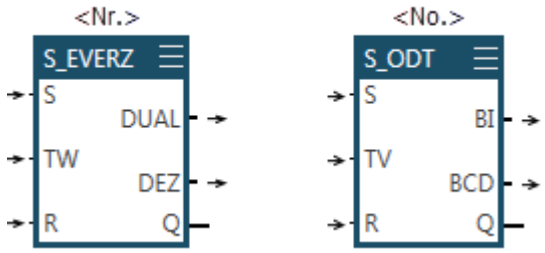
4.11.4 Zeit als Einschaltverzögerung parametrieren und starten - S_EVERZ

Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird die Operation "Zeit als Einschaltverzögerung parametrieren und starten" gestartet. Der Ausgang *Q* wird auf den Signalzustand "1" gesetzt, sobald die am Eingang *TW* voreingestellte Zeit abgelaufen ist und der Eingang *S* noch immer den Signalzustand "1" führt.

Der Ausgang *Q* wird erst dann wieder auf den Signalzustand "0" zurücksetzt, wenn mindestens einer der folgenden Zustände eintritt:

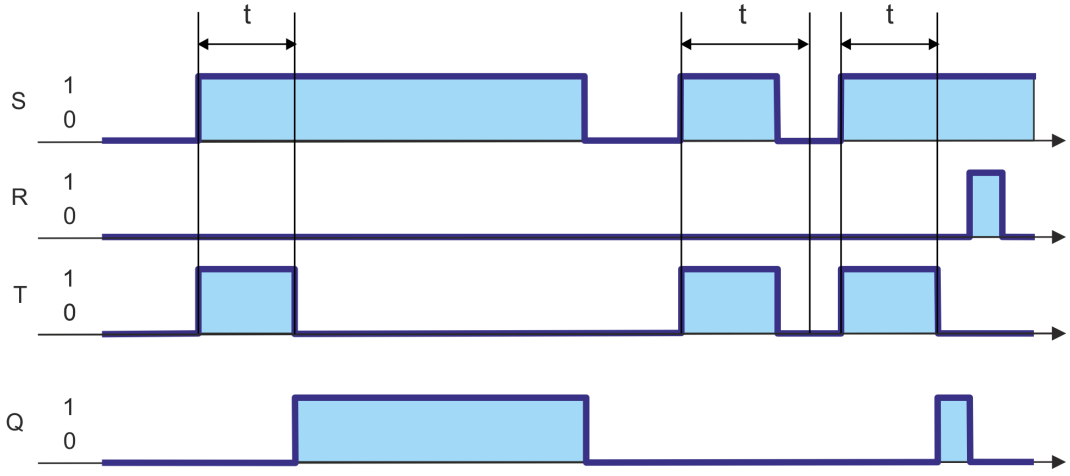
- Der Signalzustand am Eingang *S* wechselt von "1" auf "0".
- Der Signalzustand am Eingang *R* wechselt von "0" auf "1".

Deutsch - Englisch



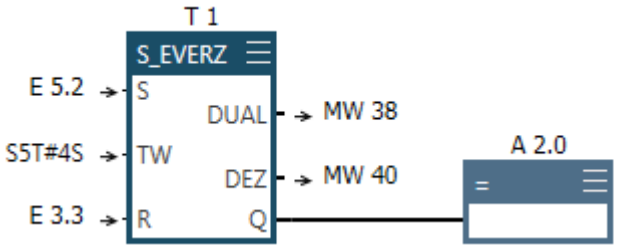
Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
S	S	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden
R	R	BOOL	E, A, M, D, L, T, Z	Rücksetzen
DUAL	BI	WORD	E, A, M, D, L	aktueller Zeitwert, Ganzzahlenformat
DEZ	BCD	WORD	E, A, M, D, L	aktueller Zeitwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status der Zeit

Statuswort für: S_EVERZ	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1



t Programmierte Zeit
S VKE am Eingang *S*
R VKE am Eingang *R*
T Zeit läuft
Q Signalzustand am Ausgang *Q*

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Wenn der Eingang `E5.2` nach vier Sekunden noch immer den Signalzustand "1" führt , wird der Ausgang `A2.0` auf den Wert "1" gesetzt. Wenn der Eingang `E5.2` jedoch vor Ablauf der vier Sekunden den Signalzustand "0" führt, bleibt der Ausgang `A2.0` auf dem Wert "0".

Wenn der Signalzustand am Eingang `E3.3` von "0" nach "1" wechselt, wird der Ausgang `A2.0` auf "0" zurückgesetzt.

Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zeitwert.

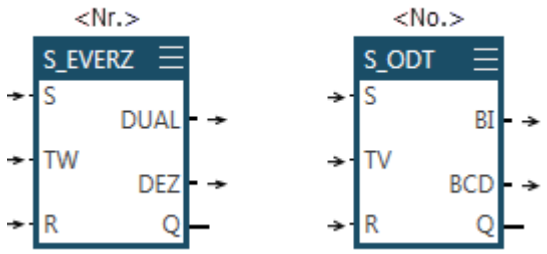
4.11.5 Zeit als speichernde Einschaltverzögerung parametrieren und starten - S_SEVERZ

Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird die Operation "Zeit als speichernde Einschaltverzögerung parametrieren und starten" gestartet. Der Ausgang *Q* wird auf den Signalzustand "1" gesetzt, sobald die am Eingang *TW* voreingestellte Zeit abgelaufen ist.

Der Ausgang *Q* wird erst dann wieder auf den Signalzustand "0" zurücksetzt, wenn der Signalzustand am Eingang *R* von "0" auf "1" wechselt.

Wenn der Signalzustand am Eingang *S* erneut von "0" auf "1" wechselt, während die gestartete Zeit läuft, wird die Zeit mit dem Zeitwert am Eingang *TW* nochmals gestartet. Der Ausgang *Q* wird erst nach dieser weiteren Zeit auf den Signalzustand "1" gesetzt.

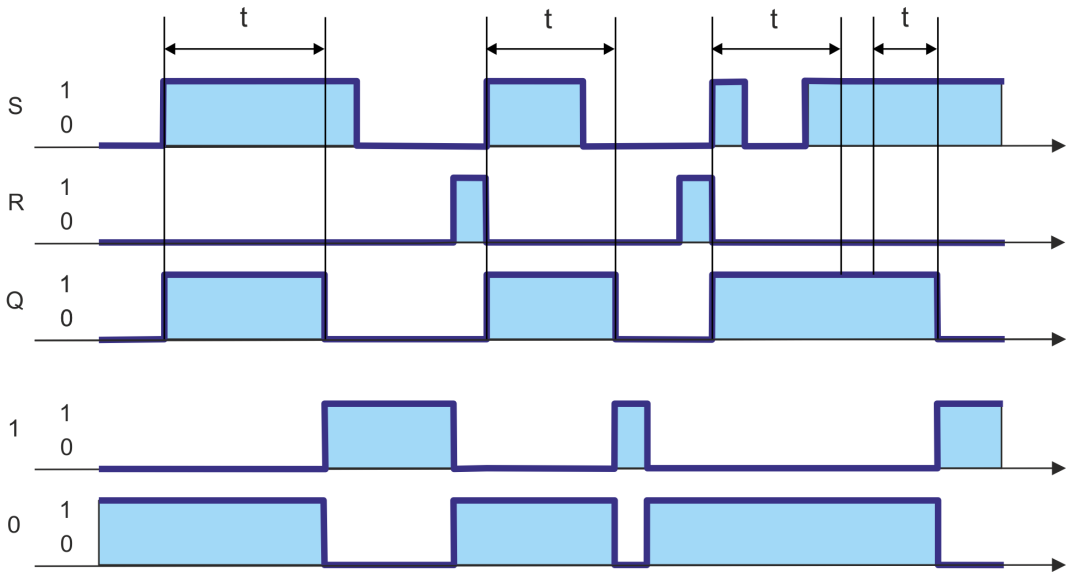
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
S	S	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden
R	R	BOOL	E, A, M, D, L, T, Z	Rücksetzen

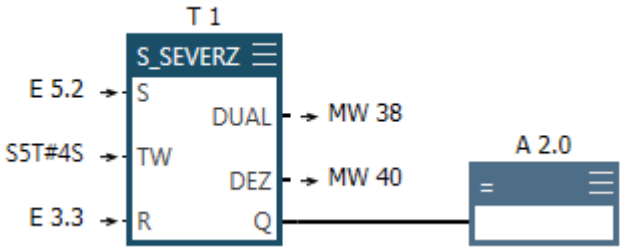
Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
DUAL	BI	WORD	E, A, M, D, L	aktueller Zeitwert, Ganzzahlenformat
DEZ	BCD	WORD	E, A, M, D, L	aktueller Zeitwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status der Zeit

Statuswort für: S_SEVERZ	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1



t Programmierte Zeit
S VKE am Eingang *S*
R VKE am Eingang *R*
Q Signalzustand am Ausgang *Q*
1 Abfrage am Ausgang *Q* auf "1"
0 Abfrage am Ausgang *Q* auf "0"

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird nach vier Sekunden auf den Wert "1" gesetzt, auch wenn der Eingang `E5.2` nicht mehr den Signalzustand "1" führt.

Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden erneut von "0" nach "1" wechselt, wird die Zeit um weitere vier Sekunden verlängert und der Ausgang `A2.0` erst nach Ablauf dieser Zeit auf "1" gesetzt.

Wenn der Signalzustand am Eingang `E3.3` von "0" nach "1" wechselt, wird der Ausgang `A2.0` auf "0" zurückgesetzt.

Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zeitwert.

4.11.6 Zeit als Ausschaltverzögerung parametrieren und starten - S_AVERZ

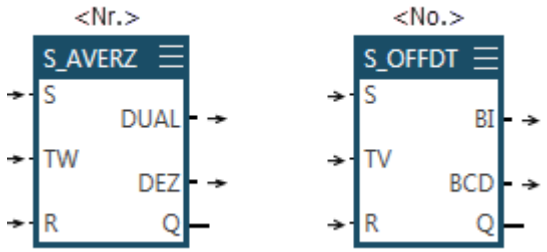
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird die Operation "Zeit als Ausschaltverzögerung parametrieren und starten" gestartet. Der Ausgang *Q* wird auf den Signalzustand "1" gesetzt. Erst bei einem Flankenwechsel von "1" auf "0" (fallende Flanke) am Eingang *S* wird die am Eingang *TW* voreingestellte Zeit gestartet.

Der Ausgang *Q* wird erst dann wieder auf den Signalzustand "0" zurücksetzt, wenn mindestens einer der folgenden Zustände eintritt:

- Die am Eingang *TW* voreingestellte Zeit, die durch die fallende Flanke am Eingang *S* gestartet wurde, ist abgelaufen.
- Der Signalzustand am Eingang *R* wechselt von "0" auf "1".

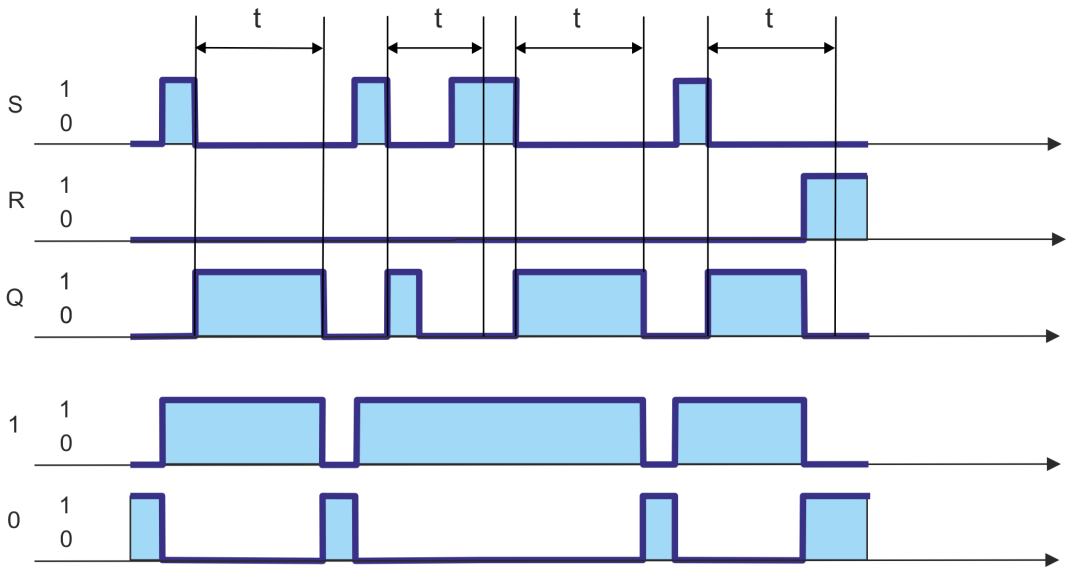
Wenn der Signalzustand am Eingang *S* erneut von "1" auf "0" wechselt, während die gestartete Zeit läuft, wird die Zeit mit dem Zeitwert am Eingang *TW* neu gestartet. Der Ausgang *Q* wird erst nach dieser weiteren Zeit auf den Signalzustand "0" gesetzt.

Deutsch - Englisch



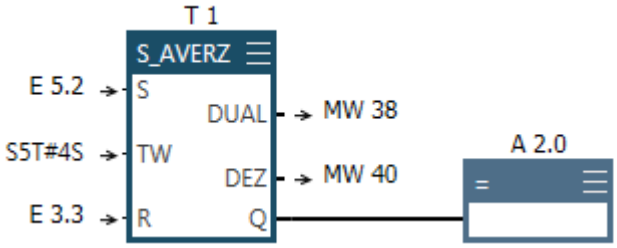
Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
S	S	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden
R	R	BOOL	E, A, M, D, L, T, Z	Rücksetzen
DUAL	BI	WORD	E, A, M, D, L	aktueller Zeitwert, Ganzzahlenformat
DEZ	BCD	WORD	E, A, M, D, L	aktueller Zeitwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status der Zeit

Statuswort für: S_AVERZ	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1



t Programmierte Zeit
S VKE am Eingang *S*
R VKE am Eingang *R*
Q Signalzustand am Ausgang *Q*
1 Abfrage am Ausgang *Q* auf "1"
0 Abfrage am Ausgang *Q* auf "0"

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird auf den Wert "1" gesetzt. Wenn der Signalzustand am Eingang `E5.2` von "1" zurück nach "0" wechselt ", wird die voreingestellte Zeit von vier Sekunden gestartet. Nach Ablauf dieser Zeit wird der Ausgang `A2.0` auf "0" zurückgesetzt.

Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden erneut von "1" nach "0" wechselt, wird die Zeit um weitere vier Sekunden verlängert und der Ausgang `A2.0` erst nach Ablauf dieser Zeit auf "0" zurückgesetzt.

Wenn der Signalzustand am Eingang `E3.3` von "0" nach "1" wechselt, wird der Ausgang `A2.0` auf "0" zurückgesetzt.

Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zeitwert.

4.11.7 Zeit als Impuls starten - SI

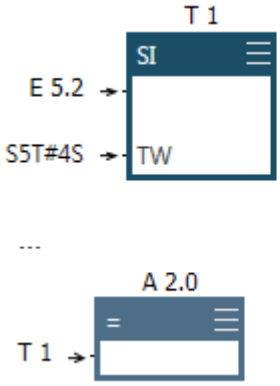
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird die Operation "Zeit als Impuls starten" mit der am Eingang *TW* voreingestellten Zeit gestartet. Siehe dazu *Kap. 4.11.2 "Zeit als Impuls parametrieren und starten - S_IMPULS" Seite 146*.

Deutsch - Englisch

Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden

Statuswort für: SI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	-	-	0

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird für vier Sekunden auf den Wert "1" gesetzt. Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden von "1" nach "0" wechselt, wird der Ausgang `A2.0` auf "0" zurückgesetzt.

4.11.8 Zeit als verlängerten Impuls starten - SV

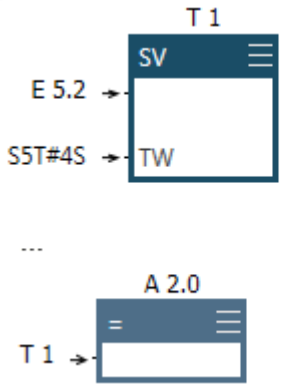
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird die Operation "Zeit als verlängerten Impuls starten" mit der am Eingang *TW* voreingestellten Zeit gestartet. Siehe dazu ↳ *Kap. 4.11.3 "Zeit als verlängerten Impuls parametrieren und starten - S_VIMP" Seite 147*.

Deutsch - Englisch

Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden

Statuswort für: SV	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	-	-	0

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird für vier Sekunden auf den Wert "1" gesetzt.

Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden erneut von "0" nach "1" wechselt, wird die Zeit um weitere vier Sekunden verlängert und der Ausgang `A2.0` erst nach Ablauf dieser Zeit auf "0" zurückgesetzt.

4.11.9 Zeit als Einschaltverzögerung starten - SE

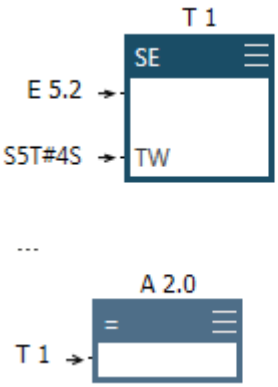
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird die Operation "Zeit als Einschaltverzögerung starten" mit der am Eingang *TW* voreingestellten Zeit gestartet. Siehe dazu ↳ *Kap. 4.11.4 "Zeit als Einschaltverzögerung parametrieren und starten - S_EVERZ" Seite 149*.

Deutsch - Englisch

Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden

Statuswort für: SE	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	-	-	0

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Wenn der Eingang `E5.2` nach vier Sekunden noch immer den Signalzustand "1" führt , wird der Ausgang `A2.0` auf den Wert "1" gesetzt. Wenn der Eingang `E5.2` jedoch vor Ablauf der vier Sekunden den Signalzustand "0" führt, bleibt der Ausgang `A2.0` auf dem Wert "0".

4.11.10 Zeit als speichernde Einschaltverzögerung starten - SS

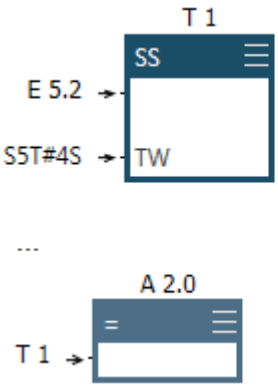
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird die Operation "Zeit als speichernde Einschaltverzögerung starten" mit der am Eingang *TW* voreingestellten Zeit gestartet. Siehe dazu *Kap. 4.11.5 "Zeit als speichernde Einschaltverzögerung parametrieren und starten - S_SEVERZ" Seite 151*.

Deutsch - Englisch

Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden

Statuswort für: SS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	-	-	0

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird nach vier Sekunden auf den Wert "1" gesetzt, auch wenn der Eingang `E5.2` nicht mehr den Signalzustand "1" führt.

Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden erneut von "0" nach "1" wechselt, wird die Zeit um weitere vier Sekunden verlängert und der Ausgang `A2.0` erst nach Ablauf dieser Zeit auf "1" gesetzt.

4.11.11 Zeit als Ausschaltverzögerung starten - SA

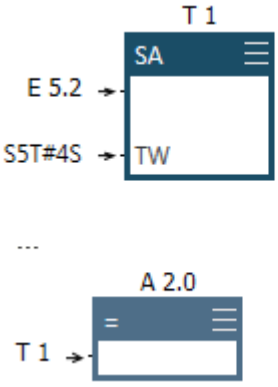
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird die Operation "Zeit als Ausschaltverzögerung starten" mit der am Eingang *TW* voreingestellten Zeit gestartet. Siehe dazu ↬ *Kap. 4.11.6 "Zeit als Ausschaltverzögerung parametrieren und starten - S_AVERZ" Seite 153*.

Deutsch - Englisch

Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden

Statuswort für: SA	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	-	-	0

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird auf den Wert "1" gesetzt. Wenn der Signalzustand am Eingang `E5.2` von "1" zurück nach "0" wechselt ", wird die voreingestellte Zeit von vier Sekunden gestartet. Nach Ablauf dieser Zeit wird der Ausgang `A2.0` auf "0" zurückgesetzt.

Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden erneut von "1" nach "0" wechselt, wird die Zeit um weitere vier Sekunden verlängert und der Ausgang `A2.0` erst nach Ablauf dieser Zeit auf "0" zurückgesetzt.

4.12 Wortverknüpfungen

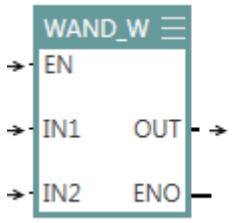
4.12.1 Übersicht

Mit den Wortverknüpfungen können Sie binäre (boolesche) Operationen mit zwei Wort- bzw. Doppelwortoperanden durchführen.

Operation	Wortverknüpfung
WAND_W	16 Bit UND-Verknüpfung
WOR_W	16 Bit ODER-Verknüpfung
WXOR_W	16 Bit EXKLUSIV-ODER-Verknüpfung
WAND_DW	32 Bit UND-Verknüpfung
WOR_DW	32 Bit ODER-Verknüpfung
WXOR_DW	32 Bit EXKLUSIV-ODER-Verknüpfung

4.12.2 16 Bit UND-Verknüpfung - WAND_W

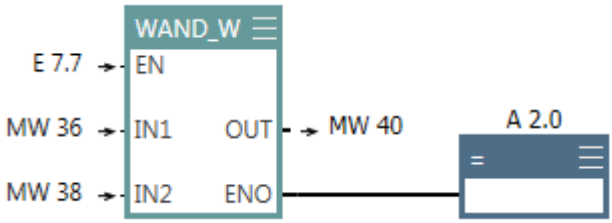
Mit der Operation "16 Bit UND-Verknüpfung" können Sie die einzelnen Bits an den Eingängen *IN1* und *IN2* logisch miteinander verknüpfen, wie in der UND-Wahrheitstabelle dargestellt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	WORD	E, A, M, D, L oder Konstante	erstes Bitfeld
IN2	WORD	E, A, M, D, L oder Konstante	zweites Bitfeld
OUT	WORD	E, A, M, D, L	Verknüpfungsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: WAND_W	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	✓	0	0	-	✓	1	1	1

Beispiel



Wenn E7.7 = 1 ist, werden die Bitfelder in den Merkerwörtern MW36 und MW38 miteinander verknüpft. Das Ergebnis wird im Merkerwort MW40 gespeichert. Wenn die Operation ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

Beispiel:

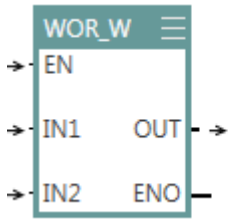
MW36 = 10011100 01101010

MW38 = 11110000 11110000

MW40 = 10010000 01100000

4.12.3 16 Bit ODER-Verknüpfung - WOR_W

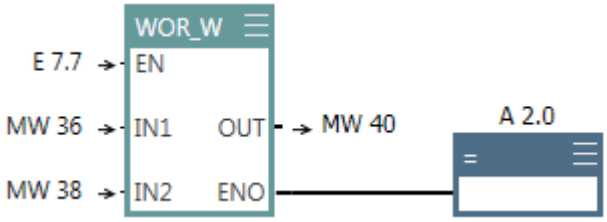
Mit der Operation "16 Bit ODER-Verknüpfung" können Sie die einzelnen Bits an den Eingängen *IN1* und *IN2* logisch miteinander verknüpfen, wie in der ODER-Wahrheitstabelle dargestellt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	WORD	E, A, M, D, L oder Konstante	erstes Bitfeld
IN2	WORD	E, A, M, D, L oder Konstante	zweites Bitfeld
OUT	WORD	E, A, M, D, L	Verknüpfungsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: WOR_W	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	✓	0	0	-	✓	1	1	1

Beispiel



Wenn E7.7 = 1 ist, werden die Bitfelder in den Merkerwörtern MW36 und MW38 miteinander verknüpft. Das Ergebnis wird im Merkerwort MW40 gespeichert. Wenn die Operation ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

Beispiel:

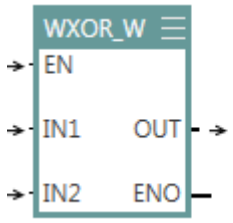
MW36 = 10011100 01101010

MW38 = 11110000 11110000

MW40 = 11111100 11111010

4.12.4 16 Bit EXKLUSIV-ODER-Verknüpfung - WXOR_W

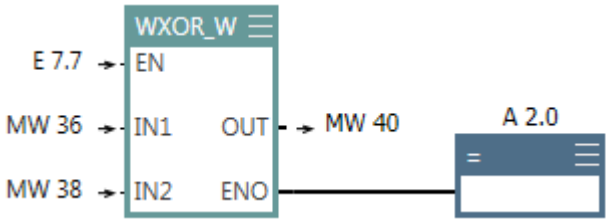
Mit der Operation "16 Bit EXKLUSIV-ODER-Verknüpfung" können Sie die einzelnen Bits an den Eingängen *IN1* und *IN2* logisch miteinander verknüpfen, wie in der EXKLUSIV-ODER-Wahrheitstabelle dargestellt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	WORD	E, A, M, D, L oder Konstante	erstes Bitfeld
IN2	WORD	E, A, M, D, L oder Konstante	zweites Bitfeld
OUT	WORD	E, A, M, D, L	Verknüpfungsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: WXOR_W	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	✓	0	0	-	✓	1	1	1

Beispiel



Wenn E7.7 = 1 ist, werden die Bitfelder in den Merkerwörtern MW36 und MW38 miteinander verknüpft. Das Ergebnis wird im Merkerwort MW40 gespeichert. Wenn die Operation ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

Beispiel:

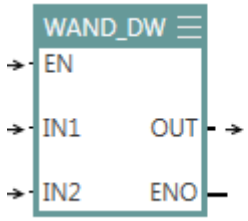
MW36 = 10011100 01101010

MW38 = 11110000 11110000

MW40 = 01101100 10011010

4.12.5 32 Bit UND-Verknüpfung - WAND_DW

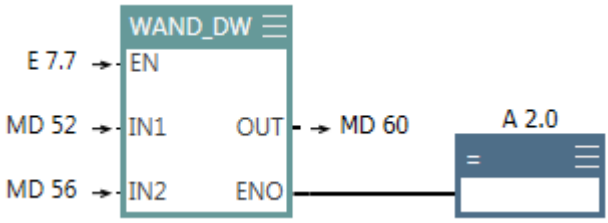
Mit der Operation "32 Bit UND-Verknüpfung" können Sie die einzelnen Bits an den Eingängen *IN1* und *IN2* logisch miteinander verknüpfen, wie in der UND-Wahrheitstabelle dargestellt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	DWORD	E, A, M, D, L oder Konstante	erstes Bitfeld
IN2	DWORD	E, A, M, D, L oder Konstante	zweites Bitfeld
OUT	DWORD	E, A, M, D, L	Verknüpfungsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: WAND_DW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	✓	0	0	-	✓	1	1	1

Beispiel



Wenn E7.7 = 1 ist, werden die Bitfelder in den Merkerdoppelwörtern MD52 und MD56 miteinander verknüpft. Das Ergebnis wird im Merkerdoppelwort MD60 gespeichert. Wenn die Operation ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

Beispiel:

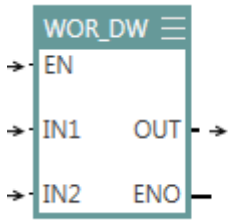
MD52 = 10011100 01101010 10011100 01101010

MD56 = 11110000 11110000 00000000 11111111

MD60 = 10010000 01100000 00000000 01101010

4.12.6 32 Bit ODER-Verknüpfung - WOR_DW

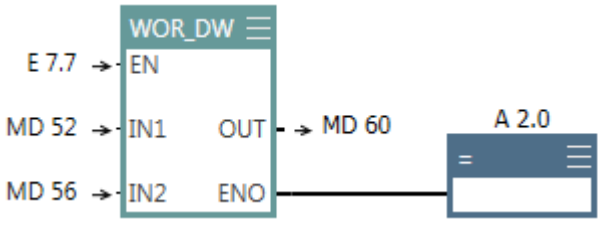
Mit der Operation "32 Bit ODER-Verknüpfung" können Sie die einzelnen Bits an den Eingängen *IN1* und *IN2* logisch miteinander verknüpfen, wie in der ODER-Wahrheitstabelle dargestellt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	DWORD	E, A, M, D, L oder Konstante	erstes Bitfeld
IN2	DWORD	E, A, M, D, L oder Konstante	zweites Bitfeld
OUT	DWORD	E, A, M, D, L	Verknüpfungsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: WOR_DW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	✓	0	0	-	✓	1	1	1

Beispiel



Wenn `E7.7` = 1 ist, werden die Bitfelder in den Merkerdoppelwörtern `MD52` und `MD56` miteinander verknüpft. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel:

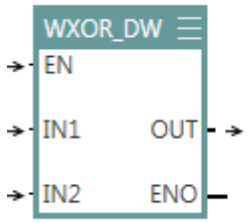
`MD52` = 10011100 01101010 10011100 01101010

`MD56` = 11110000 11110000 00000000 11111111

`MD60` = 11111100 11111010 10011100 11111111

4.12.7 32 Bit EXKLUSIV-ODER-Verknüpfung - WXOR_DW

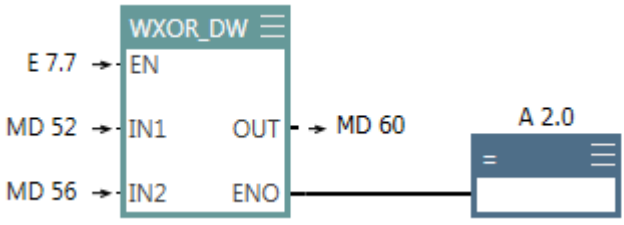
Mit der Operation "32 Bit EXKLUSIV-ODER-Verknüpfung" können Sie die einzelnen Bits an den Eingängen *IN1* und *IN2* logisch miteinander verknüpfen, wie in der EXKLUSIV-ODER-Wahrheitstabelle dargestellt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	WORD	E, A, M, D, L oder Konstante	erstes Bitfeld
IN2	WORD	E, A, M, D, L oder Konstante	zweites Bitfeld
OUT	WORD	E, A, M, D, L	Verknüpfungsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: WXOR_DW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	✓	0	0	-	✓	1	1	1

Beispiel



Wenn `E7.7` = 1 ist, werden die Bitfelder in den Merkerdoppelwörtern `MD52` und `MD56` miteinander verknüpft. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel:

`MD52` = 10011100 01101010 10011100 01101010

`MD56` = 11110000 11110000 00000000 11111111

`MD60` = 01101100 10011100 10011100 10010101

4.13 Statusbits

4.13.1 Überblick

Mit den Statusbit-Operationen können Sie binäre Verknüpfungen programmieren, die auf den Bits des Statusworts im Speicher der CPU basieren.

Operation	Statusbit
OV	Störungsbit "Überlauf": In der letzten arithmetischen Operation ist ein Überlauf aufgetreten
OS	Störungsbit "Überlauf gespeichert": In einer arithmetischen Operation ist ein Überlauf aufgetreten
UO	Störungsbit "Ungültige Operation": Das Ergebnis einer arithmetischen Operation ist ungültig
BIE	Störungsbit BIE-Register abfragen
==0 <>0 >0 <0 >=0 <=0	Ergebnisbit: Ergebnis einer arithmetischen Operation mit dem Wert Null vergleichen

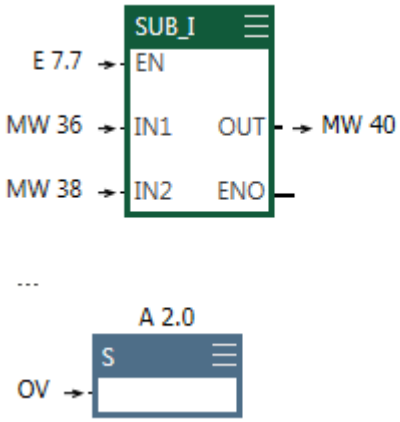
4.13.2 Störungsbit Überlauf - OV

Mit der Operation "Störungsbit Überlauf" können Sie den Signalzustand des Störungsbits "Überlauf (OV)" abfragen. Wenn das Ergebnis der letzten arithmetischen Operation außerhalb des zulässigen negativen oder positiven Bereichs liegt, wird das Bit "OV" gesetzt.

Statuswort für: OV	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Wenn am Element "OV" nur ein Eingang belegt ist und Sie am Ausgang ein weiteres Element einfügen, z. B. "Ausgang setzen (S)", werden beide Elemente miteinander vereint:

Beispiel



Im ersten Netzwerk wird die Ganzzahl (16 Bit) im Merkerwort `MW38` von der Ganzzahl (16 Bit) im Merkerwort `MW36` subtrahiert. Wenn das Ergebnis im Merkerwort `MW40` außerhalb des zulässigen Bereichs für eine Ganzzahl (16 Bit) liegt, wird das Bit "OV" auf den Wert "1" gesetzt. Der Ausgang `A2.0` wird gesetzt. Das Bit "OV" wird zurückgesetzt, wenn weitere arithmetische Operationen zwischen den beiden Netzwerken liegen, die keinen Überlauf verursachen.

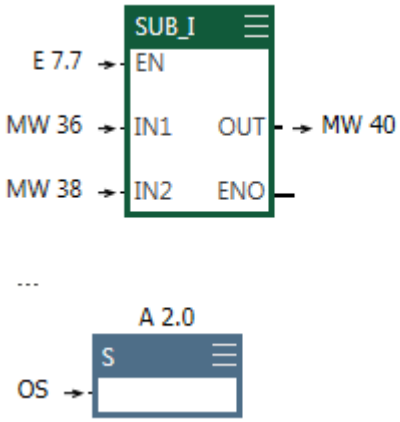
4.13.3 Störungsbit Überlauf gespeichert - OS

Mit der Operation "Störungsbit Überlauf gespeichert" können Sie den Signalzustand des Störungsbits "Überlauf gespeichert (OS)" abfragen. Wenn das Ergebnis einer arithmetischen Operation außerhalb des zulässigen negativen oder positiven Bereichs liegt, wird das Bit "OV" gesetzt.

Statuswort für: OS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Wenn am Element "OS" nur ein Eingang belegt ist und Sie am Ausgang ein weiteres Element einfügen, z. B. "Ausgang setzen (S)", werden beide Elemente miteinander vereint:

Beispiel



Im ersten Netzwerk wird die Ganzzahl (16 Bit) im Merkerwort `MW38` von der Ganzzahl (16 Bit) im Merkerwort `MW36` subtrahiert. Wenn das Ergebnis im Merkerwort `MW40` außerhalb des zulässigen Bereichs für eine Ganzzahl (16 Bit) liegt, wird das Bit "OS" auf den Wert "1" gesetzt. Der Ausgang `A2.0` wird gesetzt. Das Bit "OS" bleibt auch dann gesetzt, wenn weitere arithmetische Operationen zwischen den beiden Netzwerken liegen, die keinen Überlauf verursachen.

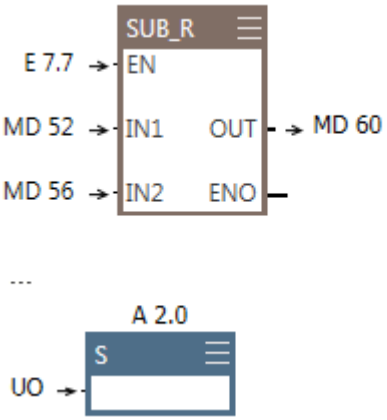
4.13.4 Störungsbit Ungültige Operation - UO

Mit der Operation "Störungsbit Ungültige Operation" können Sie den Signalzustand des Störungsbits "Ungültige Operation (UO)" abfragen. Wenn bei arithmetischen Operationen mit Gleitpunktzahlen ein Eingang keine gültige Gleitpunktzahl enthält, wird das Bit "UO" gesetzt.

Statuswort für: UO	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Wenn am Element "UO" nur ein Eingang belegt ist und Sie am Ausgang ein weiteres Element einfügen, z. B. "Ausgang setzen (S)", werden beide Elemente miteinander vereint:

Beispiel



Im ersten Netzwerk wird die Gleitpunktzahl im Merkerdoppelwort `MD56` von der Gleitpunktzahl im Merkerdoppelwort `MD52` subtrahiert. Wenn mindestens einer der beiden Eingänge keine gültige Gleitpunktzahl enthält, wird das Bit "UO" auf den Wert "1" gesetzt. Der Ausgang `A2.0` wird gesetzt.

4.13.5 Störungsbit BIE-Register - BIE

Mit der Operation "Störungsbit BIE-Register" können Sie den Signalzustand des "Binärergebnis-Bits (BIE)" abfragen.

Statuswort für: BIE	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Wenn am Element "BIE" nur ein Eingang belegt ist und Sie am Ausgang ein weiteres Element einfügen, z. B. "Ausgang setzen (S)", werden beide Elemente miteinander vereint:

Beispiel

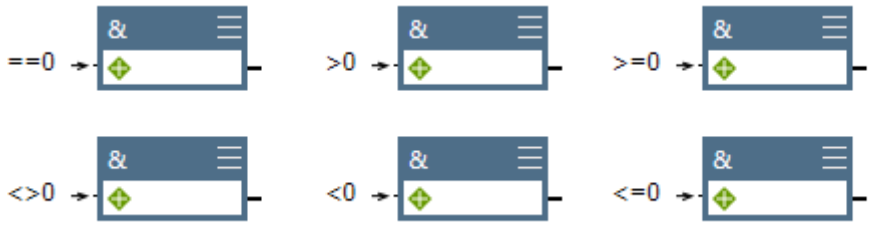
Wenn der Eingang `E4.0` und das Binärergebnis-Bits "BIE" den Signalzustand "1" führen, liegt am Ausgang `A2.0` der Wert "1" an.

4.13.6 Ergebnisbit - x0

Mit der Operation "Ergebnisbit" können Sie das Ergebnis einer arithmetischen Operation mit dem Wert Null vergleichen. Wenn das Ergebnis der letzten arithmetischen Operation der Vergleichsbedingung entspricht, wird das Ergebnisbit gesetzt.

Sie können aus folgenden Vergleichsbedingungen wählen:

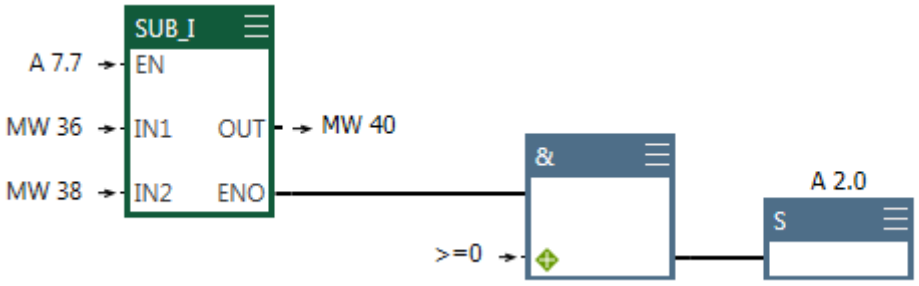
Operation	Beschreibung
== 0	Setzt das Ergebnisbit, wenn das Ergebnis der Operation gleich 0 ist
<> 0	Setzt das Ergebnisbit, wenn das Ergebnis der Operation ungleich 0 ist
> 0	Setzt das Ergebnisbit, wenn das Ergebnis der Operation größer als 0 ist
< 0	Setzt das Ergebnisbit, wenn das Ergebnis der Operation kleiner als 0 ist
>= 0	Setzt das Ergebnisbit, wenn das Ergebnis der Operation größer als oder gleich 0 ist
<= 0	Setzt das Ergebnisbit, wenn das Ergebnis der Operation kleiner als oder gleich 0 ist



Statuswort für: <> 0	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Wenn am Element "Ergebnisbit" nur ein Eingang belegt ist und Sie am Ausgang ein weiteres Element einfügen, z. B. "Ausgang setzen (S)", werden beide Elemente miteinander vereint:

Beispiel



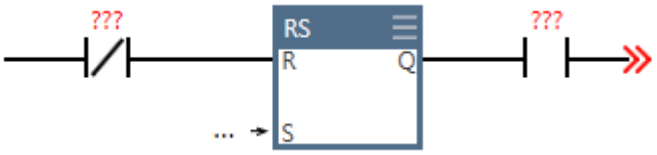
Wenn `E7.7` = 1 ist, wird der Wert im Merkerwort `MW38` von `MW36` subtrahiert. Wenn das Ergebnis im Merkerwort `MW40` größer als oder gleich Null ist, wird der Ausgang `A2.0` gesetzt. Wenn das Ergebnis kleiner als Null ist, wird `A2.0` nicht gesetzt.

5 KOP-Operationen

5.1 Überblick

Kontaktplan (KOP) ist eine grafische Programmiersprache zur Signalverarbeitung. Mit dem Kontaktplan können unterschiedliche Funktionselemente miteinander verknüpft werden, um den Signalfluss zu steuern. Die Darstellung ist vergleichbar mit einem Stromlaufplan. Verbindungen von Kontakten und Spulen beschreiben den Stromfluss zwischen zwei Stromschienen.

Kontaktplan ist für Steuerungen ausgelegt, bei denen einfache Elemente wie Öffner, Schließer und Ausgänge verwendet werden. Komplexere Elemente, wie z. B. Zeitglieder oder Zähler, werden ähnlich wie bei der Programmiersprache Funktionsplan (FUP) dargestellt, siehe Abbildung.



Zeichen	Bedeutung
–	links vom Element: Eingangsparameter (eingehender Wert)
>>	rechts vom Element: Ausgangsparameter (ausgehender Wert), Signalfluss ist nicht abgeschlossen
┤	Signalfluss ist abgeschlossen
???	Angabe des Parameters ist zwingend
...	Angabe des Parameters ist optional

5.2 Bitverknüpfungen

5.2.1 Überblick

Mit den Bitverknüpfungen können Sie binäre (boolesche) Operationen programmieren. Binäre Operationen basieren auf den beiden Signalzuständen "0" und "1".

Operation	Bitverknüpfung
---| |---	Schließerkontakt
---|/|---	Öffnerkontakt
---|NOT|---	Verknüpfungsergebnis invertieren
---( )	Relaisspule, Ausgang
---(#)---	Konnektor
---(R)	Ausgang rücksetzen
---(S)	Ausgang setzen
RS	Flipflop rücksetzen setzen
SR	Flipflop setzen rücksetzen
---(N)---	Flanke 1-0 erkennen

Operation	Bitverknüpfung
---(P)---	Flanke 0-1 erkennen
---(SAVE)	Verknüpfungsergebnis in BIE-Register laden
NEG	Signalflanke 1-0 erkennen
POS	Signalflanke 0-1 erkennen

Wahrheitstabelle

Die folgende Wahrheitstabelle zeigt die Verknüpfungsergebnisse für verschiedene binäre Operationen mit zwei Eingangsparametern.

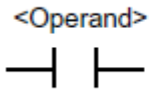
Operand 1	Operand 2	ODER	UND	XOR
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

5.2.2 Schließerkontakt - ---| |---

Der "Schließerkontakt" wird geschlossen, wenn der Wert des abgefragten Bits, der am angegebenen Operanden gespeichert wird gleich "1" ist. Bei geschlossenem Kontakt kann Strom fließen und das Verknüpfungsergebnis (VKE) ist "1".

Der "Schließerkontakt" wird geöffnet, wenn der Wert des abgefragten Bits, der am angegebenen Operanden gespeichert wird gleich "0" ist. Bei geöffnetem Kontakt kann kein Strom fließen und das Verknüpfungsergebnis (VKE) ist "0".

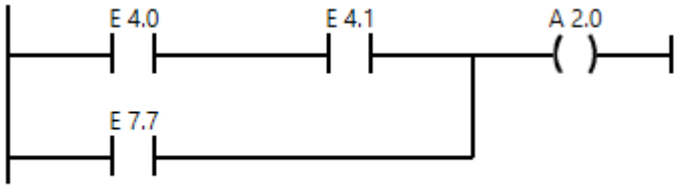
Bei Reihenschaltungen wird der Kontakt ---| |--- mit dem VKE bitweise durch UND verknüpft. Bei Parallelschaltungen wird der Kontakt mit dem VKE durch ODER verknüpft.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L, T, Z	Bit-Operand, dessen Signalzustand abgefragt wird

Statuswort für: =	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



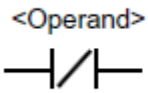
Strom kann fließen, wenn der Zustand an den Eingängen `E4.0` **und** `E4.1` "1" ist **oder** der Zustand an Eingang `E7.7` "1" ist.

5.2.3 Öffnerkontakt - ---|/|---

Der "Öffnerkontakt" wird geschlossen, wenn der Wert des abgefragten Bits, der am angegebenen Operanden gespeichert wird gleich "0" ist. Bei geschlossenem Kontakt kann Strom fließen und das Verknüpfungsergebnis (VKE) ist "1".

Der "Öffnerkontakt" wird geöffnet, wenn der Wert des abgefragten Bits, der am angegebenen Operanden gespeichert wird gleich "1" ist. Bei geöffnetem Kontakt kann kein Strom fließen und das Verknüpfungsergebnis (VKE) ist "0".

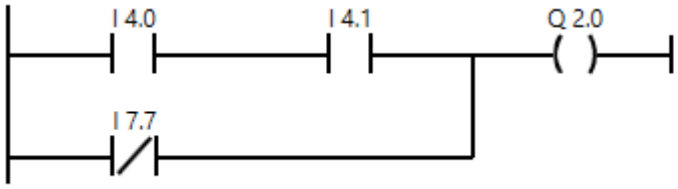
Bei Reihenschaltungen wird der Kontakt ---|/|--- mit dem VKE bitweise durch UND verknüpft. Bei Parallelschaltungen wird der Kontakt mit dem VKE durch ODER verknüpft.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L, T, Z	Bit-Operand, dessen Signalzustand abgefragt wird

Statuswort für: =	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



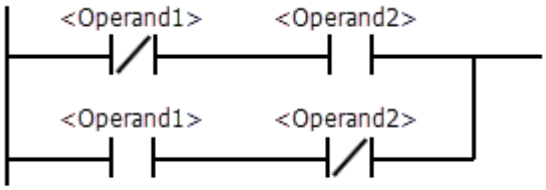
Strom kann fließen, wenn der Zustand an den Eingängen `E4.0` **und** `E4.1` "1" ist **oder** der Zustand an Eingang `E7.7` "0" ist.

5.2.4 EXKLUSIV-ODER-Verknüpfung - XOR

Mit der XOR können Sie die Signalzustände mehrerer Operanden an den Eingängen abfragen und miteinander logisch verknüpfen, siehe "XOR" in *"Wahrheitstabelle" Seite 173*.

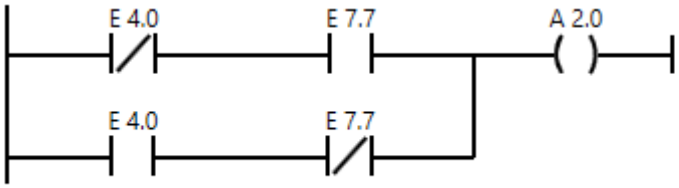
Wenn die Signale der beiden Eingänge unterschiedlich sind, nimmt der Ausgang den Wert "1" an. Bei XOR-Verknüpfungen mit mehr als zwei Eingängen nimmt der Ausgang den Wert "1" an, wenn eine ungerade Anzahl von Eingängen den Signalzustand "1" führt.

Die Programmierung einer XOR-Verknüpfung besteht aus einem Netzwerk von Schließern und Öffnern.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand1	BOOL	E, A, M, T, Z, D, L	Bit-Operand, dessen Signalzustand abgefragt wird
Operand2	BOOL	E, A, M, T, Z, D, L	Bit-Operand, dessen Signalzustand abgefragt wird

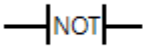
Beispiel



Wenn **entweder** der Eingang `E4.0` **oder** `E7.7` den Signalzustand "1" führt, liegt am Ausgang `A2.0` der Wert "1" an. Wenn beide Eingänge den gleichen Signalzustand "0" oder "1" führen, liegt am Ausgang der Wert "0" an.

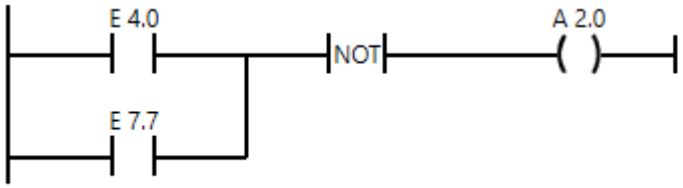
5.2.5 Verknüpfungsergebnis invertieren - ---|NOT|---

Mit der Operation "Verknüpfungsergebnis invertieren" können Sie das VKE-Bit umkehren.



Statuswort für: -o|	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	-	1	✓	-

Beispiel



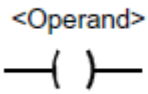
Der Ausgang `A2.0` ist "1", wenn der Zustand an beiden Eingängen `E4.0` **und** `E7.7` "0" ist.

5.2.6 Relaisspule, Ausgang - ---( )

Mit der "Relaisspule" können Sie das Verknüpfungsergebnis (VKE) einem Operanden zuweisen. Sie können die Zuweisung an den Ausgang einer Verknüpfung setzen.

Wenn die Bedingungen der Verknüpfung vor der Relaisspule erfüllt sind (VKE = 1), nimmt der Ausgang den Wert "1" an. Wenn die Bedingungen der Verknüpfung vor der Relaisspule nicht erfüllt sind (VKE = 0), nimmt der Ausgang den Wert "0" an.

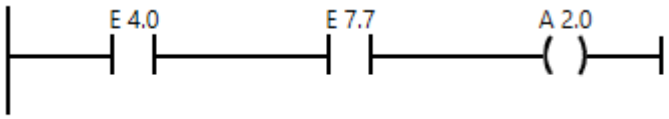
Sie können eine Zuweisung negieren, indem Sie davor das VKE-Bit invertieren, siehe *Kap. 5.2.5 " Verknüpfungsergebnis invertieren - ---|NOT|---" Seite 175*.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, dem das Verknüpfungsergebnis (VKE) zugewiesen wird

Statuswort für: =	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	✓	-	0

Beispiel

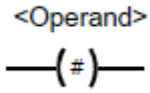


Wenn die Eingänge `E4.0` und `E7.7` den Signalzustand "1" führen, liegt am Ausgang `A2.0` der Wert "1" an.

5.2.7 Konnektor - ---(#)---

Mit dem "Konnektor" können Sie das Verknüpfungsergebnis VKE in einem Operanden speichern.

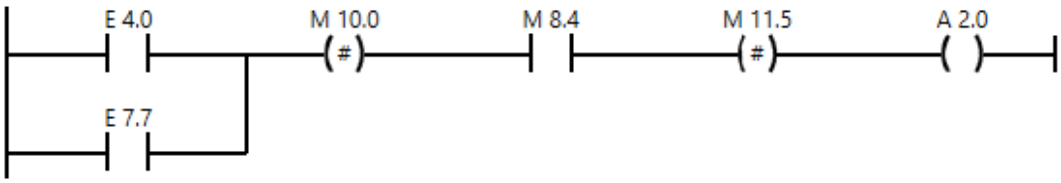
Sie können einen Konnektor negieren, indem Sie davor das VKE-Bit invertieren, siehe *Kap. 5.2.5 " Verknüpfungsergebnis invertieren - ---|NOT|---" Seite 175*.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, *L	Bit-Operand, in dem das Verknüpfungsergebnis VKE gespeichert wird
* Im Lokaldaten-Stack können Operanden nur dann verwendet werden, wenn sie im TEMP-Bereich der Variablendeklaration eines Programmbausteins (OB, FB, FC) deklariert worden sind.			

Statuswort für: #	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	✓	-	1

Beispiel



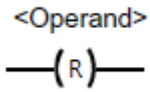
In den Konnektoren werden folgende Verknüpfungsergebnisse gespeichert:

- `M10.0` speichert das VKE der davor liegenden ODER-Verknüpfung. Das VKE wird unverändert an die folgende UND-Verknüpfung weitergeleitet.
- `M11.5` speichert das VKE von `M8.4`.

5.2.8 Ausgang rücksetzen - ---(R)

Mit den Operationen "Ausgang setzen (S)" und "Ausgang rücksetzen (R)" können Sie eine Signalspeicherung programmieren.

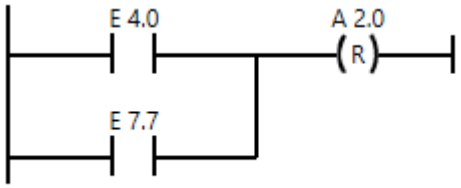
Der Operand wird nur dann auf den Wert "0" zurückgesetzt, wenn das Verknüpfungsergebnis VKE = 1 ist. Wenn das VKE = 0 ist, bleibt der Operand unverändert.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, T, Z, D, L	Bit-Operand, der zurückgesetzt werden soll

Statuswort für: R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	✓	-	0

Beispiel

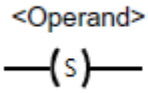


Wenn die Eingänge `E4.0` oder `E7.7` den Signalzustand "1" führen, dann wird der Ausgang `A2.0` auf den Wert "0" zurückgesetzt. Wenn das VKE der Verknüpfung anschließend wieder nach "0" wechselt, bleibt der Signalzustand am Ausgang unverändert auf dem Wert "0".

5.2.9 Ausgang setzen - ---(S)

Mit den Operationen "Ausgang setzen (S)" und "Ausgang rücksetzen (R)" können Sie eine Signalspeicherung programmieren.

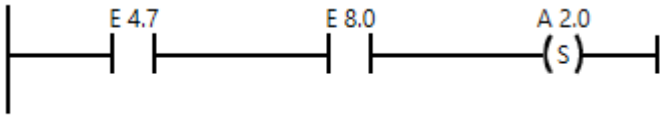
Der Operand wird nur dann auf den Wert "1" gesetzt, wenn das Verknüpfungsergebnis VKE = 1 ist. Wenn das VKE = 0 ist, bleibt der Operand unverändert.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, der gesetzt werden soll

Statuswort für: S	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	✓	-	0

Beispiel



Wenn die Eingänge `E4.7` und `E8.0` den Signalzustand "1" führen, dann wird der Ausgang `A2.0` auf den Wert "1" gesetzt. Wenn das VKE der Verknüpfung anschließend wieder nach "0" wechselt, bleibt der Signalzustand am Ausgang unverändert auf dem Wert "1" .

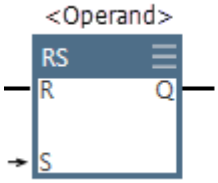
5.2.10 Flipflop rücksetzen setzen - RS

Mit einem Flipflop (bistabiles Element) können Sie ein binäres Signal speichern.

Der Operand wird nur dann gesetzt (S) bzw. zurückgesetzt (R), wenn das Verknüpfungsergebnis VKE = 1 ist. Wenn das Verknüpfungsergebnis VKE = 0 ist, bleibt der Operand unverändert.

Wenn die Eingänge den Signalzustand *R* = 0 und *S* = 1 führen, wird der Operand auf den Wert "1" gesetzt. Wenn die Eingänge den Signalzustand *R* = 1 und *S* = 0 führen, wird der Operand auf den Wert "0" zurückgesetzt.

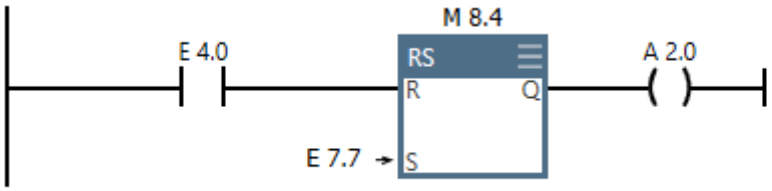
Mit der Operation "Flipflop rücksetzen setzen" wird der Operand **vorrangig gesetzt**. Das bedeutet, wenn die beiden Eingänge *R* und *S* gleichzeitig den Signalzustand "1" führen, wird der Operand auf den Wert "1" gesetzt.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, der gesetzt oder zurückgesetzt werden soll
R	BOOL	E, A, M, D, L, T, Z	Speicher zurücksetzen
S	BOOL	E, A, M, D, L, T, Z	Speicher setzen
Q	BOOL	E, A, M, D, L	Ausgang, Signalzustand des Operanden

Statuswort für: RS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



Wenn der Eingang `E4.0` den Signalzustand "0" führt und der Eingang `E7.7` den Signalzustand "1" führt, werden der Merker `M8.4` und der Ausgang `A2.0` auf den Wert "1" gesetzt. Wenn der Eingang `E4.0` den Signalzustand "1" führt und der Eingang `E7.7` den Signalzustand "0" führt, werden der Merker `M8.4` und der Ausgang `A2.0` auf den Wert "0" zurückgesetzt.

Wenn am Eingang beide Signalzustände "0" führen, bleiben die Signalzustände am Merker und Ausgang unverändert. Wenn am Eingang beide Signalzustände "1" führen, werden der Merker und der Ausgang auf den Wert "1" gesetzt.

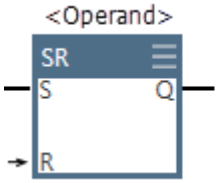
5.2.11 Flipflop setzen rücksetzen - SR

Mit einem Flipflop (bistabiles Element) können Sie ein binäres Signal speichern.

Der Operand wird nur dann gesetzt (S) bzw. zurückgesetzt (R), wenn das Verknüpfungsergebnis VKE = 1 ist. Wenn das Verknüpfungsergebnis VKE = 0 ist, bleibt der Operand unverändert.

Wenn die Eingänge den Signalzustand *R* = 0 und *S* = 1 führen, wird der Operand auf den Wert "1" gesetzt. Wenn die Eingänge den Signalzustand *R* = 1 und *S* = 0 führen, wird der Operand auf den Wert "0" zurückgesetzt.

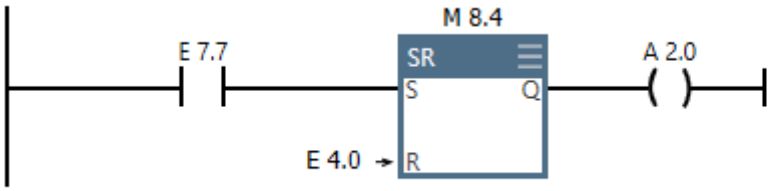
Mit der Operation "Flipflop setzen rücksetzen" wird der Operand **vorrangig rückgesetzt**. Das bedeutet, wenn die beiden Eingänge *R* und *S* gleichzeitig den Signalzustand "1" führen, wird der Operand auf den Wert "0" zurückgesetzt.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, der gesetzt oder zurückgesetzt werden soll
S	BOOL	E, A, M, D, L, T, Z	Speicher setzen
R	BOOL	E, A, M, D, L, T, Z	Speicher zurücksetzen
Q	BOOL	E, A, M, D, L	Ausgang, Signalzustand des Operanden

Statuswort für: SR	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



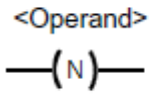
Wenn der Eingang `E4.0` den Signalzustand "0" führt und der Eingang `E7.7` den Signalzustand "1" führt, werden der Merker `M8.4` und der Ausgang `A2.0` auf den Wert "1" gesetzt. Wenn der Eingang `E4.0` den Signalzustand "1" führt und der Eingang `E7.7` den Signalzustand "0" führt, werden der Merker `M8.4` und der Ausgang `A2.0` auf den Wert "0" zurückgesetzt.

Wenn am Eingang beide Signalzustände "0" führen, bleiben die Signalzustände am Merker und Ausgang unverändert. Wenn am Eingang beide Signalzustände "1" führen, werden der Merker und der Ausgang auf den Wert "0" zurückgesetzt.

5.2.12 Flanke 1-0 erkennen - ---(N)---

Die Operation "Flanke 1-0 erkennen" erkennt den Übergang des Verknüpfungsergebnisses VKE von "1" auf "0" (fallende Flanke).

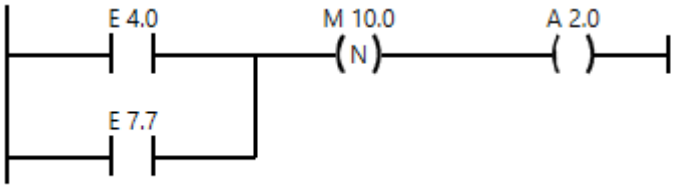
Die Operation vergleicht den Wert des VKE am Eingang der N-Operation mit dem Wert der vorherigen Abfrage, der im Operanden (Flankenmerker) gespeichert ist. Nur wenn der Flankenmerker den Wert "1" führt und das aktuelle VKE den Wert "0" führt, liegt eine **fallende Flanke** vor. Das VKE am Ausgang der N-Operation wird kurzzeitig (als Impuls) auf den Wert "1" gesetzt.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, der den vorherigen Zustand des VKE speichert (Flankenmerker)

Statuswort für: N	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	✓	✓	1

Beispiel

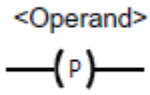


Wenn das VKE der ODER-Verknüpfung von "1" nach "0" wechselt, wird der Ausgang `A2.0` kurzzeitig auf den Wert "1" gesetzt. Der Flankenmerker `M10.0` wird dabei als Speicher verwendet, um die Flanke festzustellen.

5.2.13 Flanke 0-1 erkennen - ---(P)---

Die Operation "Flanke 0-1 erkennen" erkennt den Übergang des Verknüpfungsergebnisses VKE von "0" auf "1" (steigende Flanke).

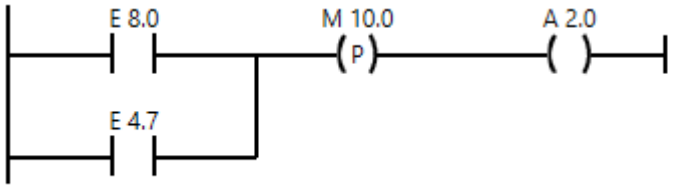
Die Operation vergleicht den Wert des VKE am Eingang der P-Operation mit dem Wert der vorherigen Abfrage, der im Operanden (Flankenmerker) gespeichert ist. Nur wenn der Flankenmerker den Wert "0" führt und das aktuelle VKE den Wert "1" führt, liegt eine **steigende Flanke** vor. Das VKE am Ausgang der P-Operation wird kurzzeitig (als Impuls) auf den Wert "1" gesetzt.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, der den vorherigen Zustand des VKE speichert (Flankenmerker)

Statuswort für: P	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	✓	✓	1

Beispiel

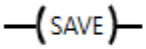


Wenn das VKE der ODER-Verknüpfung von "0" nach "1" wechselt, wird der Ausgang `A2.0` kurzzeitig auf den Wert "1" gesetzt. Der Flankenmerker `M10.0` wird dabei als Speicher verwendet, um die Flanke festzustellen.

5.2.14 Verknüpfungsergebnis in BIE-Register übernehmen - ---(SAVE)

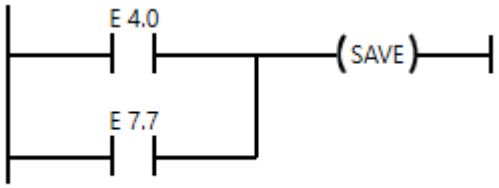
Mit der Operation "SAVE" wird das Verknüpfungsergebnis VKE in das Binärergebnis-Bit "BIE" des Statuswortes übernommen. Das Erstabfragebit "/ER" wird nicht zurückgesetzt. Am Bausteinende wird der Ausgang "ENO" auf den Wert des VKE gesetzt. Die nächste Operation wird mit dem VKE des aktuellen Netzwerks verknüpft.

Die Operation "SAVE" kann beispielsweise vor dem Verlassen eines Bausteins verwendet werden, um das Verknüpfungsergebnis im aufrufenden Baustein auszuwerten. Da sich das Binärergebnis-Bit "BIE" durch folgende Operationen ändern kann, sollten Sie die Operation "SAVE" direkt vor dem Verlassen eines Bausteins einsetzen.



Statuswort für: SAVE	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	-	-	-	-	-	-

Beispiel

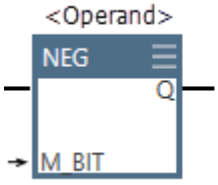


Das Verknüpfungsergebnis VKE wird im Binärergebnis-Bit "BIE" gespeichert.

5.2.15 Signalflanke 1 - 0 erkennen - NEG

Die Operation "Signalflanke 1 - 0 erkennen" erkennt den Übergang des Signalzustands von "1" auf "0" (fallende Flanke).

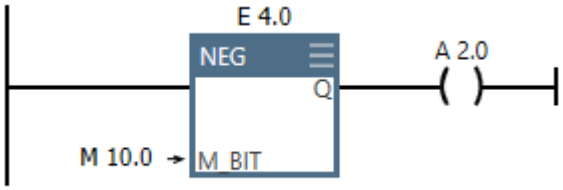
Die Operation vergleicht den Signalzustand des Operanden mit dem Signalzustand der vorherigen Abfrage, der im Eingang *M_BIT* (Flankenmerker) gespeichert ist. Nur wenn *M_BIT* den Wert "1" führt und der Operand den Wert "0" führt, liegt eine **fallende Flanke** vor. Das VKE am Ausgang der NEG-Operation wird kurzzeitig (als Impuls) auf den Wert "1" gesetzt.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, dessen Flankenwechsel erkannt werden soll
M_BIT	BOOL	A, M, D	Bit-Operand, der den vorherigen Signalzustand speichert (Flankenmerker).
Q	BOOL	E, A, M, D, L	Ausgang, Impuls bei Flankenwechsel

Statuswort für: NEG	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	1	✓	1

Beispiel

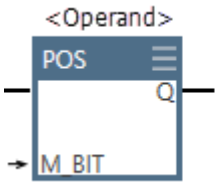


Wenn der Signalzustand am Eingang `E4.0` von "1" nach "0" wechselt, wird der Ausgang `A2.0` kurzzeitig auf den Wert "1" gesetzt. Der Flankenmerker `M10.0` wird dabei als Speicher verwendet, um die Flanke festzustellen.

5.2.16 Signalflanke 0 - 1 erkennen - POS

Die Operation "Signalflanke 1 - 0 erkennen" erkennt den Übergang des Signalzustands von "0" auf "1" (steigende Flanke).

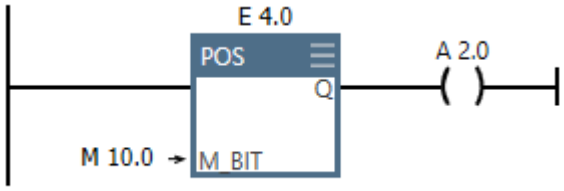
Die Operation vergleicht den Signalzustand des Operanden mit dem Signalzustand der vorherigen Abfrage, der im Eingang *M_BIT* (Flankenmerker) gespeichert ist. Nur wenn *M_BIT* den Wert "0" führt und der Operand den Wert "1" führt, liegt eine **steigende Flanke** vor. Das VKE am Ausgang der POS-Operation wird kurzzeitig (als Impuls) auf den Wert "1" gesetzt.



Parameter	Datentyp	Speicherbereich	Beschreibung
Operand	BOOL	E, A, M, D, L	Bit-Operand, dessen Flankenwechsel erkannt werden soll
M_BIT	BOOL	A, M, D	Bit-Operand, der den vorherigen Signalzustand speichert (Flankenmerker).
Q	BOOL	E, A, M, D, L	Ausgang, Impuls bei Flankenwechsel

Statuswort für: POS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	1	✓	1

Beispiel



Wenn der Signalzustand am Eingang `E4.0` von "0" nach "1" wechselt, wird der Ausgang `A2.0` kurzzeitig auf den Wert "1" gesetzt. Der Flankenmerker `M10.0` wird dabei als Speicher verwendet, um die Flanke festzustellen.

5.3 Vergleicher

5.3.1 Überblick

Mit den Vergleichsoperationen können Sie zwei Werte miteinander vergleichen.

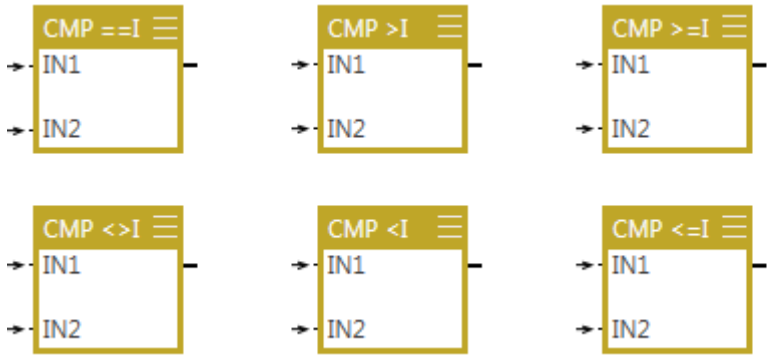
Operation	Vergleich
==	Wert 1 ist gleich Wert 2
<>	Wert 1 ist ungleich Wert 2
>	Wert 1 ist größer als Wert 2
<	Wert 1 ist kleiner als Wert 2
>=	Wert 1 ist größer als oder gleich Wert 2
<=	Wert 1 ist kleiner als oder gleich Wert 2

Sie können Werte der folgenden Datentypen vergleichen:

- Ganze Zahlen (16 Bit) - CMP xx I
- Ganze Zahlen (32 Bit) - CMP xx D
- Gleitpunktzahlen - CMP xx R

5.3.2 Ganze Zahlen vergleichen (16 Bit) - CMP xx I

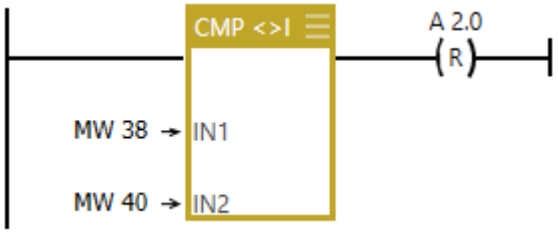
Mit der Operation "Ganze Zahlen vergleichen (16 Bit)" können Sie zwei 16 Bit-Festpunktzahlen an den Eingängen *IN1* und *IN2* miteinander vergleichen.



Parameter	Datentyp	Speicherbereich	Beschreibung
IN1	DINT	E, A, M, D, L oder Konstante	erster Vergleichswert
IN2	DINT	E, A, M, D, L oder Konstante	zweiter Vergleichswert
Ausgang	BOOL	E, A, M, D, L	Vergleichsergebnis

Statuswort für: CMP xx I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	0	-	0	✓	✓	1

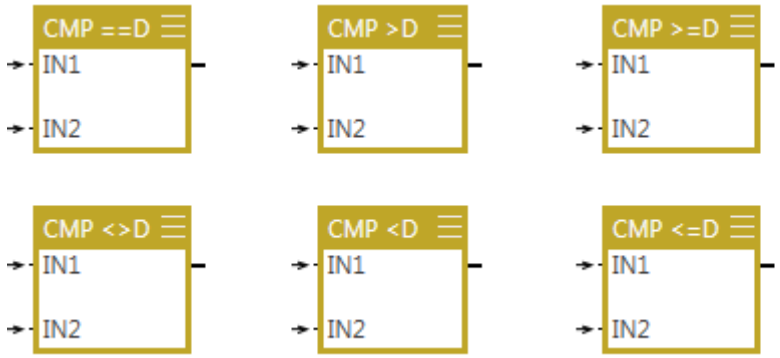
Beispiel



Wenn `MW38` **ungleich** `MW40` ist, wird der Ausgang `A2.0` zurückgesetzt.

5.3.3 Ganze Zahlen vergleichen (32 Bit) - CMP xx D

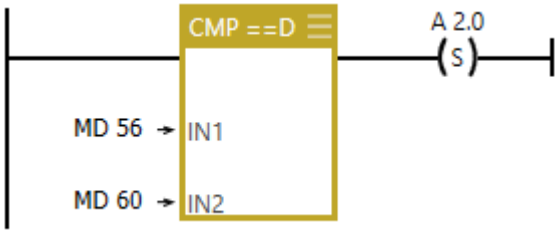
Mit der Operation "Ganze Zahlen vergleichen (32 Bit)" können Sie zwei 32 Bit-Festpunktzahlen an den Eingängen *IN1* und *IN2* miteinander vergleichen.



Parameter	Datentyp	Speicherbereich	Beschreibung
IN1	DINT	E, A, M, D, L oder Konstante	erster Vergleichswert
IN2	DINT	E, A, M, D, L oder Konstante	zweiter Vergleichswert
Ausgang	BOOL	E, A, M, D, L	Vergleichsergebnis

Statuswort für: CMP xx D	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	✓	✓	0	-	0	✓	✓	1

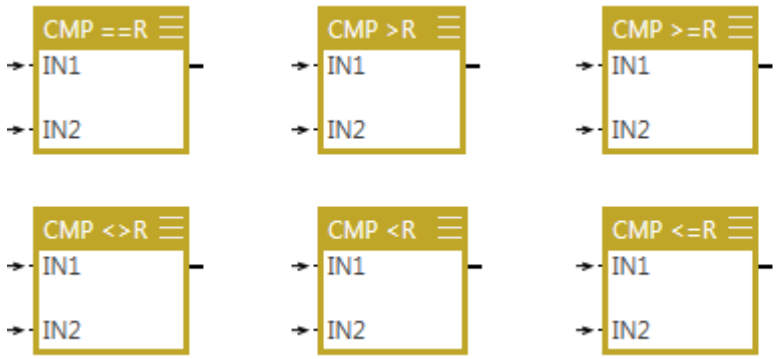
Beispiel



Wenn `MD56` **gleich** `MD60` ist, wird der Ausgang `A2.0`gesetzt.

5.3.4 Gleitpunktzahlen vergleichen - CMP xx R

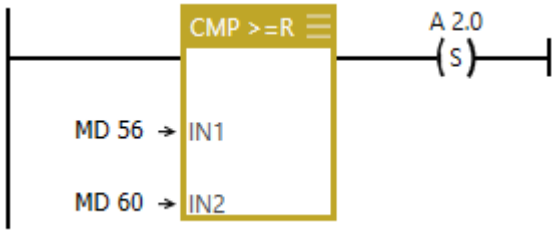
Mit der Operation "Gleitpunktzahlen vergleichen" können Sie zwei Gleitpunktzahlen (32 Bit-Realzahlen) an den Eingängen *IN1* und *IN2* miteinander vergleichen.



Parameter	Datentyp	Speicherbereich	Beschreibung
IN1	REAL	E, A, M, D, L oder Konstante	erster Vergleichswert
IN2	REAL	E, A, M, D, L oder Konstante	zweiter Vergleichswert
Ausgang	BOOL	E, A, M, D, L	Vergleichsergebnis

Statuswort für: CMP xx R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



Wenn `MD56` **größer oder gleich** `MD60` ist, wird der Ausgang `A2.0`gesetzt.

5.4 Umwandler

5.4.1 Überblick

Mit den Umwandlungsoperationen können Sie Werte von einem Zahlenformat in ein anderes umwandeln.

Operation	Umwandlung
BCD_I	BCD-Zahl in Ganzzahl (16 Bit) wandeln
I_BCD	Ganzzahl (16 Bit) in BCD-Zahl wandeln
I_DI	Ganzzahl (16 Bit) in Ganzzahl (32 Bit) wandeln
BCD_DI	BCD-Zahl in Ganzzahl (32 Bit) wandeln
DI_BCD	Ganzzahl (32 Bit) in BCD-Zahl wandeln
DI_R	Ganzzahl (32 Bit) in Gleitpunktzahl wandeln
INV_I	Einer-Komplement zu Ganzzahl (16 Bit) erzeugen
INV_DI	Einer-Komplement zu Ganzzahl (32 Bit) erzeugen
NEG_I	Zweier-Komplement zu Ganzzahl (16 Bit) erzeugen
NEG_DI	Zweier-Komplement zu Ganzzahl (32 Bit) erzeugen
NEG_R	Vorzeichen einer Gleitpunktzahl wechseln
ROUND	Zahl runden
TRUNC	Ganze Zahl erzeugen

Operation	Umwandlung
CEIL	aus Gleitpunktzahl nächsthöhere Ganzzahl erzeugen
FLOOR	aus Gleitpunktzahl nächstniedrige Ganzzahl erzeugen

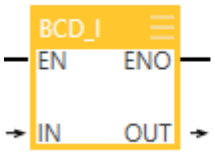
5.4.2 BCD-Zahl in Ganzzahl (16 Bit) wandeln - BCD_I

Die Operation "BCD-Zahl in Ganzzahl (16 Bit) wandeln" wandelt die dreistellige binär-codierte Dezimalzahl (BCD, ± 999) am Eingang *IN* in einen ganzzahligen Wert (16 Bit) am Ausgang *OUT* um. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt.

Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.

Wenn eine binär-codierte Dezimalstelle der BCD-Zahl im ungültigen Bereich liegt (größer als 9), tritt ein Wandlungsfehler auf:

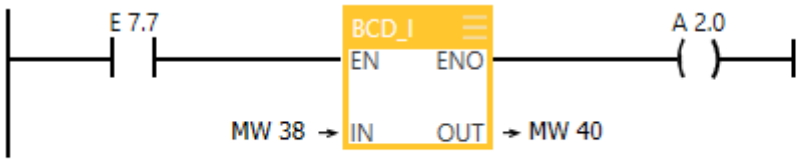
- Die CPU schaltet in den Betriebszustand STOP. Im Diagnosespeicher wird ein "BCD-Umwandlungsfehler" mit der Ereignisnummer 2521 eingetragen.
- Wenn der Organisationsbaustein OB121 programmiert worden ist, wird er aufgerufen.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	WORD	E, A, M, D, L oder Konstante	BCD-Zahl
OUT	INT	E, A, M, D, L	ganzzahliger Wert (16 Bit) der BCD-Zahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: BCD_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	-	-	-	-	0	1	1	1

Beispiel

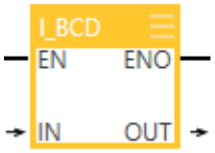


Wenn `E7.7` = 1 ist, wird die dreistellige BCD-Zahl im Merkerwort `MW38` in eine Ganzzahl (16 Bit) umgewandelt und im Merkerwort `MW40` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.4.3 Ganzzahl (16 Bit) in BCD-Zahl wandeln - I_BCD

Die Operation "Ganzzahl (16 Bit) in BCD-Zahl wandeln" wandelt den ganzzahligen Wert (16 Bit) am Eingang *IN* in eine dreistellige binär-codierte Dezimalzahl (BCD, ± 999) am Ausgang *OUT* um. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt.

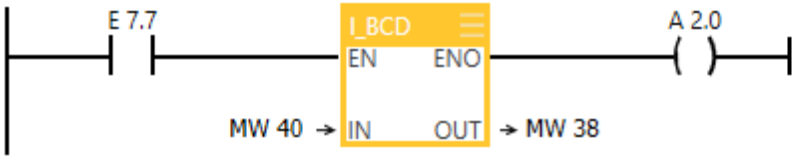
Wenn ein Überlauf eintritt, ist der Parameter *ENO* = 0 und die Umwandlung wird nicht ausgeführt.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	INT	E, A, M, D, L oder Konstante	Ganzzahl (16 Bit)
OUT	WORD	E, A, M, D, L	BCD-Wert der Ganzzahl (16 Bit)
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: I_BCD	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	✓	✓	0	✓	✓	1

Beispiel

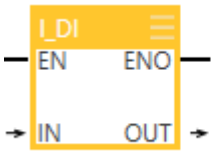


Wenn `E7.7` = 1 ist, wird die Ganzzahl (16 Bit) im Merkerwort `MW40` in eine dreistellige BCD-Zahl umgewandelt und im Merkerwort `MW38` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN). Wenn ein Überlauf eintritt, ist `A2.0` = 0 und die Umwandlung wird nicht ausgeführt.

5.4.4 Ganzzahl (16 Bit) in Ganzzahl (32 Bit) wandeln - I_DI

Die Operation "Ganzzahl (16 Bit) in Ganzzahl (32 Bit) wandeln" wandelt die Ganzzahl (16 Bit) am Eingang *IN* in eine Ganzzahl (32 Bit) am Ausgang *OUT* um. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt.

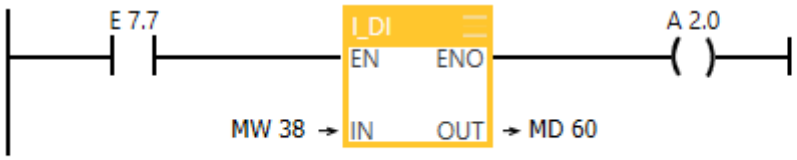
Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	INT	E, A, M, D, L oder Konstante	Ganzzahl (16 Bit)
OUT	DINT	E, A, M, D, L	Ganzzahl (32 Bit)
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: I_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	-	-	0	1	1	1

Beispiel



Wenn `E7.7` = 1 ist, wird die Ganzzahl (16 Bit) im Merkerwort `MW38` in eine Ganzzahl (32 Bit) umgewandelt und im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

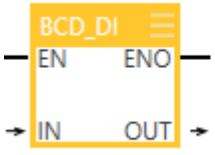
5.4.5 BCD-Zahl in Ganzzahl (32 Bit) wandeln - BCD_DI

Die Operation "BCD-Zahl in Ganzzahl (32 Bit) wandeln" wandelt die siebenstellige binär-codierte Dezimalzahl (BCD, ± 9.999.999) am Eingang *IN* in einen ganzzahligen Wert (32 Bit) am Ausgang *OUT* um. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt.

Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.

Wenn eine binär-codierte Dezimalstelle der BCD-Zahl im ungültigen Bereich liegt (größer als 9), tritt ein Wandlungsfehler auf:

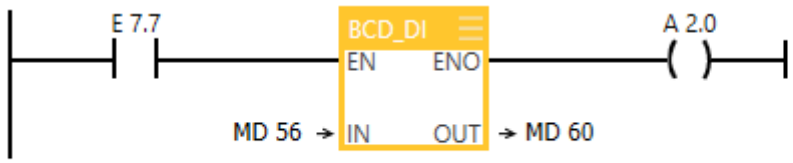
- Die CPU schaltet in den Betriebszustand STOP. Im Diagnosespeicher wird ein "BCD-Umwandlungsfehler" mit der Ereignisnummer 2521 eingetragen.
- Wenn der Organisationsbaustein OB121 programmiert worden ist, wird er aufgerufen.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DWORD	E, A, M, D, L oder Konstante	BCD-Zahl
OUT	DINT	E, A, M, D, L	ganzzahliger Wert (32 Bit) der BCD-Zahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: BCD_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	-	-	-	-	0	1	1	1

Beispiel

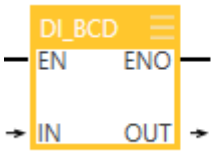


Wenn `E7.7` = 1 ist, wird die siebenstellige BCD-Zahl im Merkerdoppelwort `MD56` in eine Ganzzahl (32 Bit) umgewandelt und im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.4.6 Ganzzahl (32 Bit) in BCD-Zahl wandeln - DI_BCD

Die Operation "Ganzzahl (32 Bit) in BCD-Zahl wandeln" wandelt den ganzzahligen Wert (32 Bit) am Eingang *IN* in eine siebenstellige binär-codierte Dezimalzahl (BCD, ± 9.999.999) am Ausgang *OUT* um. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt.

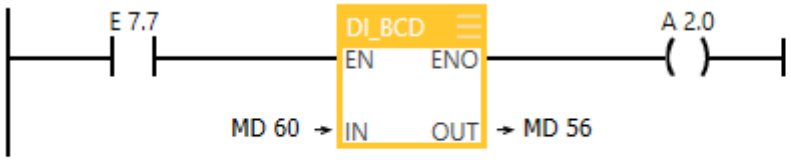
Wenn ein Überlauf eintritt, ist der Parameter *ENO* = 0 und die Umwandlung wird nicht ausgeführt.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DINT	E, A, M, D, L oder Konstante	Ganzzahl (32 Bit)
OUT	DWORD	E, A, M, D, L	BCD-Wert der Ganzzahl (32 Bit)
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: DI_BCD	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	✓	✓	0	✓	✓	1

Beispiel

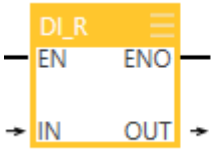


Wenn `E7.7` = 1 ist, wird die Ganzzahl (32 Bit) im Merkerdoppelwort `MD60` in eine siebenstellige BCD-Zahl umgewandelt und im Merkerdoppelwort `MD56` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN). Wenn ein Überlauf eintritt, ist `A2.0` = 0 und die Umwandlung wird nicht ausgeführt.

5.4.7 Ganzzahl (32 Bit) in Gleitpunktzahl wandeln - DI_R

Die Operation "Ganzzahl (32 Bit) in Gleitpunktzahl wandeln" wandelt die Ganzzahl (32 Bit) am Eingang *IN* in eine Gleitpunktzahl am Ausgang *OUT* um. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt.

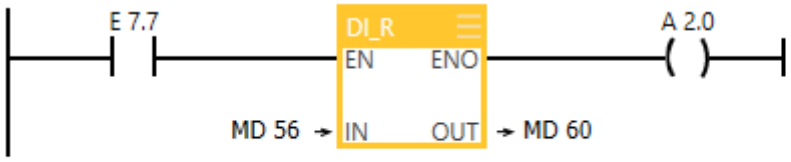
Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DINT	E, A, M, D, L oder Konstante	Ganzzahl (32 Bit)
OUT	REAL	E, A, M, D, L	Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: DI_R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	-	-	-	-	0	1	1	1

Beispiel

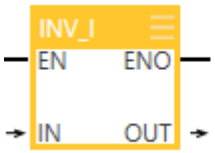


Wenn E7.7 = 1 ist, wird die Ganzzahl (32 Bit) im Merkerdoppelwort MD56 in eine Gleitpunktzahl umgewandelt und im Merkerdoppelwort MD60 gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

5.4.8 Einer-Komplement zu Ganzzahl (16 Bit) erzeugen - INV_I

Die Operation "Einer-Komplement zu Ganzzahl (16 Bit) erzeugen" führt eine Wortverknüpfungsoperation "EXKLUSIV ODER verknüpfen" des Eingangs *IN* mit der Hexadezimalschablone FFFF aus. Dadurch werden die Werte der einzelnen Bits umgekehrt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

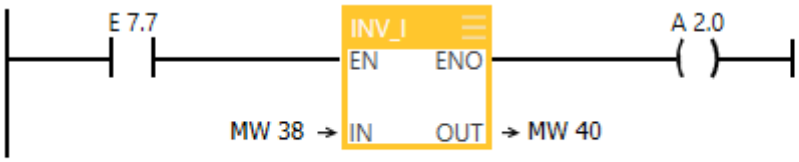
Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	INT	E, A, M, D, L oder Konstante	Ganzzahl (16 Bit)
OUT	INT	E, A, M, D, L	Einer-Komplement der Ganzzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: INV_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	-	-	-	-	0	1	1	1

Beispiel



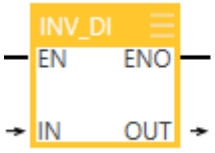
Wenn E7.7 = 1 ist, wird aus der Ganzzahl (16 Bit) im Merkerwort MW38 das Einer-Komplement gebildet. Jedes Bit wird umgekehrt. Das Ergebnis wird im Merkerwort MW40 gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

Beispiel: MW38 = 10011100 01101010, MW40 = 01100011 10010101

5.4.9 Einer-Komplement zu Ganzzahl (32 Bit) erzeugen - INV_DI

Die Operation "Einer-Komplement zu Ganzzahl (32 Bit) erzeugen" führt eine Wortverknüpfungsoperation "EXKLUSIV ODER verknüpfen" des Eingangs *IN* mit der Hexadezimalschablone FFFF aus. Dadurch werden die Werte der einzelnen Bits umgekehrt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

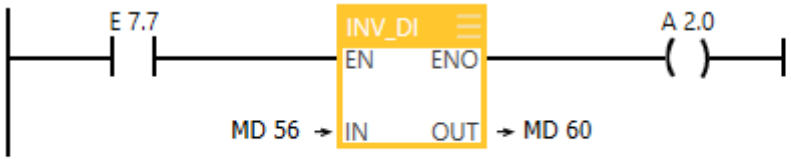
Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	INT	E, A, M, D, L oder Konstante	Ganzzahl (32 Bit)
OUT	INT	E, A, M, D, L	Einer-Komplement der Ganzzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: INV_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	-	-	-	-	0	1	1	1

Beispiel



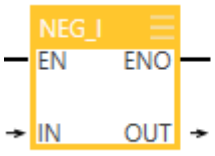
Wenn `E7.7` = 1 ist, wird aus der Ganzzahl (32 Bit) im Merkerdoppelwort `MD56` das Einer-Komplement gebildet. Jedes Bit wird umgekehrt. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel: `MD56` = FE01 5B83, `MD60` = 01FE A47C

5.4.10 Zweier-Komplement zu Ganzzahl (16 Bit) erzeugen - NEG_I

Die Operation "Zweier-Komplement zu Ganzzahl (16 Bit) erzeugen" kehrt das Vorzeichen des Werts am Eingang *IN* um, z. B. von einem positiven in einen negativen Wert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

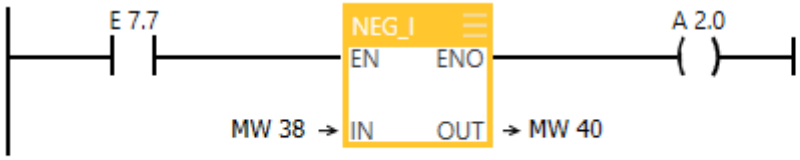
Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand. Ausnahme: Wenn der Signalzustand von *EN* gleich "1" ist und ein Überlauf eintritt, ist der Signalzustand von *ENO* gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	INT	E, A, M, D, L oder Konstante	Ganzzahl (16 Bit)
OUT	INT	E, A, M, D, L	Zweier-Komplement der Ganzzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: NEG_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel



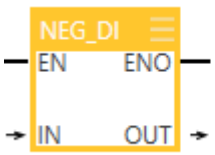
Wenn `E7.7` = 1 ist, wird das Vorzeichen der Ganzzahl (16 Bit) im Merkerwort `MW38` umgekehrt. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel: `MW38` = 42, `MW40` = -42

5.4.11 Zweier-Komplement zu Ganzzahl (32 Bit) erzeugen - NEG_DI

Die Operation "Zweier-Komplement zu Ganzzahl (32 Bit) erzeugen" kehrt das Vorzeichen des Werts am Eingang *IN* um, z. B. von einem positiven in einen negativen Wert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

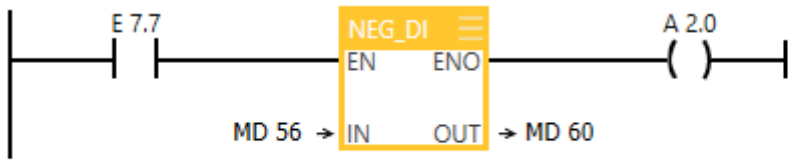
Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand. Ausnahme: Wenn der Signalzustand von *EN* gleich "1" ist und ein Überlauf eintritt, ist der Signalzustand von *ENO* gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DINT	E, A, M, D, L oder Konstante	Ganzzahl (32 Bit)
OUT	DINT	E, A, M, D, L	Zweier-Komplement der Ganzzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: NEG_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel



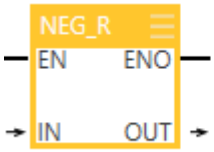
Wenn `E7.7` = 1 ist, wird das Vorzeichen der Ganzzahl (32 Bit) im Merkerdoppelwort `MD56` umgekehrt. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel: `MD56` = 100.000, `MD60` = -100.000

5.4.12 Vorzeichen einer Gleitpunktzahl wechseln - NEG_R

Die Operation "Vorzeichen einer Gleitpunktzahl wechseln" kehrt das Vorzeichenbit des Werts am Eingang *IN* um, z. B. von"0" für positiv in "1" für negativ. Die Bits für Exponent und Mantisse bleiben unverändert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

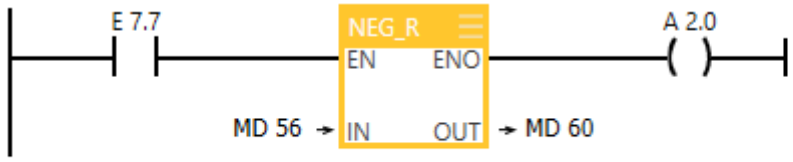
Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand. Ausnahme: Wenn der Signalzustand von *EN* gleich "1" ist und ein Überlauf eintritt, ist der Signalzustand von *ENO* gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Gleitpunktzahl, Vorzeichen gewechselt
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: NEG_R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	-	-	0	✓	✓	1

Beispiel



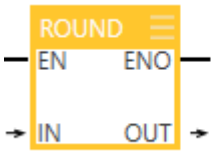
Wenn `E7.7` = 1 ist, wird das Vorzeichen der Gleitpunktzahl im Merkerdoppelwort `MD56` umgekehrt. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel: `MD56` = 42e2, `MD60` = -42e2

5.4.13 Zahl runden - ROUND

Die Operation "Zahl runden" rundet die Gleitpunktzahl am Eingang *IN* zu einer Ganzzahl (32 Bit). Wenn die erste Ziffer nach dem Komma 0, 1, 2, 3 oder 4 ist, wird abgerundet (Beispiel: "2,49" wird abgerundet zu "2"). Wenn die erste Ziffer nach dem Komma 5, 6, 7, 8 oder 9 ist, wird aufgerundet (Beispiel: "2,5" wird aufgerundet zu "3"). Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

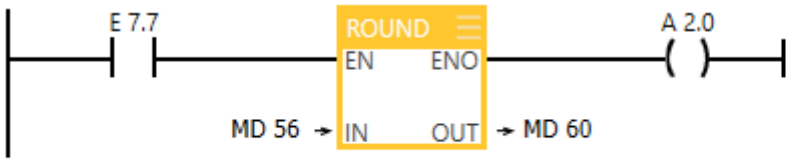
Wenn ein Überlauf eintritt, ist der Signalzustand von *ENO* gleich "0". Wenn der Wert am Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	DINT	E, A, M, D, L	Ganzzahl (32 Bit), zur nächsten ganzen Zahl gerundet
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ROUND	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	✓	✓	0	✓	✓	1

Beispiel

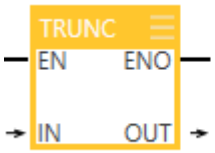


Wenn `E7.7` = 1 ist, wird die Gleitpunktzahl im Merkerdoppelwort `MD56` auf- bzw. abgerundet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.4.14 Ganze Zahl erzeugen - TRUNC

Die Operation "Ganze Zahl erzeugen" wandelt den Wert der Gleitpunktzahl am Eingang *IN* in eine Ganzzahl (32 Bit) um, indem die Nachkommastellen abgeschnitten werden (Beispiel: "2,49" wird zu "2", "2,5" wird ebenfalls zu "2"). Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

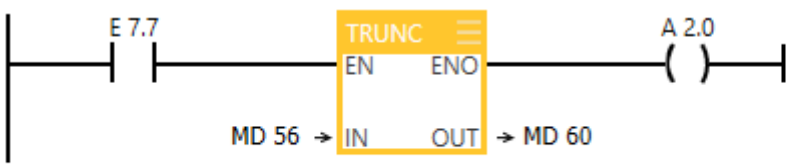
Wenn ein Überlauf eintritt, ist der Signalzustand von *ENO* gleich "0". Wenn der Wert am Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	DINT	E, A, M, D, L	Ganzzahl (32 Bit), Nachkommastellen abgeschnitten
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: TRUNC	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	✓	✓	0	✓	✓	1

Beispiel

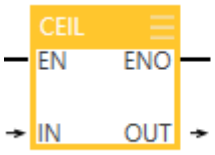


Wenn E7.7 = 1 ist, werden die Nachkommastellen der Gleitpunktzahl im Merkerdoppelwort MD56 abgeschnitten. Das Ergebnis wird als Ganzzahl (32 Bit) im Merkerdoppelwort MD60 gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

5.4.15 Aus Gleitpunktzahl nächsthöhere Ganzzahl erzeugen - CEIL

Die Operation "Aus Gleitpunktzahl nächsthöhere Ganzzahl erzeugen" rundet die Gleitpunktzahl am Eingang *IN* zu einer Ganzzahl (32 Bit) auf. (Beispiel: "2,3" wird zu "3", "-2,8" wird zu "-2"). Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

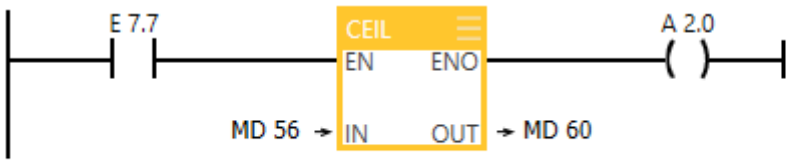
Wenn ein Überlauf eintritt, ist der Signalzustand von *ENO* gleich "0". Wenn der Wert am Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	DINT	E, A, M, D, L	Ganzzahl (32 Bit), aufgerundet
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: CEIL	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	✓	✓	0	✓	✓	1

Beispiel

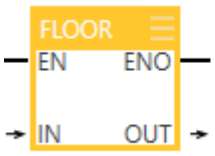


Wenn E7.7 = 1 ist, wird die Gleitpunktzahl im Merkerdoppelwort MD56 aufgerundet. Das Ergebnis wird als Ganzzahl (32 Bit) im Merkerdoppelwort MD60 gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

5.4.16 Aus Gleitpunktzahl nächstniedrige Ganzzahl erzeugen - FLOOR

Die Operation "Aus Gleitpunktzahl nächstniedrige Ganzzahl erzeugen" rundet die Gleitpunktzahl am Eingang *IN* zu einer Ganzzahl (32 Bit) ab. (Beispiel: "2,8" wird zu "2", "-2,3" wird zu "-3"). Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

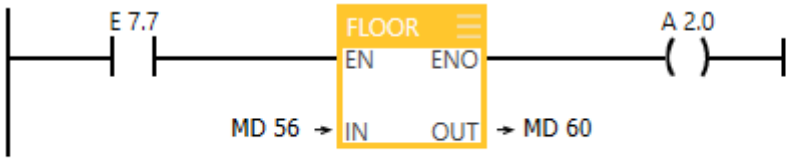
Wenn ein Überlauf eintritt, ist der Signalzustand von *ENO* gleich "0". Wenn der Wert am Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	DINT	E, A, M, D, L	Ganzzahl (32 Bit), abgerundet
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: FLOOR	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	✓	✓	0	✓	✓	1

Beispiel



Wenn `E7.7` = 1 ist, wird die Gleitpunktzahl im Merkerdoppelwort `MD56` abgerundet. Das Ergebnis wird als Ganzzahl (32 Bit) im Merkerdoppelwort `MD60` gespeichert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.5 Zähler

5.5.1 Überblick

In der CPU ist ein eigener Speicherbereich für Zähler reserviert. Die Anzahl der Zähler ist abhängig von der CPU.

Operation	Zähler
Z_VORW	Parametrieren und Vorwärtszählen
Z_RUECK	Parametrieren und Rückwärtszählen
ZAEHLER	Parametrieren und Vorwärts-/ Rückwärtszählen
--(SZ)	Setzen des Zähleranfangswerts
--(ZV)	Vorwärtszählen
--(ZR)	Rückwärtszählen

5.5.2 Parametrieren und Vorwärts-/Rückwärtszählen - ZAEHLER

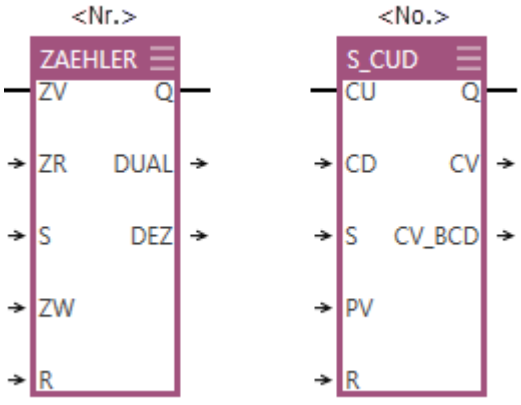
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird der Zähler auf den Zählwert *ZW* gesetzt. Wenn der Eingang *R* den Signalzustand "1" führt, wird der Zählwert *ZW* auf "0" gesetzt.

Bei jeder steigenden Flanke am Eingang *ZV* wird der Zählwert um 1 erhöht. Der Zählwert wird nicht weiter erhöht, wenn er "999" erreicht hat. Bei jeder steigenden Flanke am Eingang *ZR* wird der Zählwert um 1 vermindert. Der Zählwert wird nicht weiter vermindert, wenn er "0" erreicht hat. Wenn gleichzeitig an beiden Eingängen eine steigende Flanke vorliegt, ändert sich der Zählwert nicht.

Wenn der Zählwert gesetzt wird und der Eingang *ZV* oder *ZR* den Signalzustand "1" führt, wird der Zählwert einmalig im nächsten Zyklus erhöht bzw. vermindert, auch wenn kein Flankenwechsel eingetreten ist.

Wenn der Zählwert größer als "0" ist, führt der Ausgang *Q* den Wert "1". Wenn der Zählwert gleich "0" ist, führt der Ausgang *Q* den Wert "0".

Deutsch - Englisch

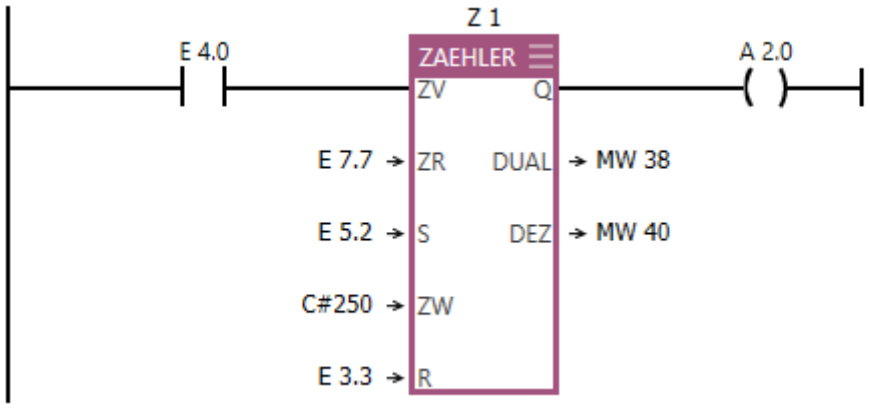


Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	COUNTER	Z	Nummer des Zählers, Bereich ist abhängig von der CPU
ZV	CU	BOOL	E, A, M, D, L	Vorwärtszählen
ZR	CD	BOOL	E, A, M, D, L	Rückwärtszählen
S	S	BOOL	E, A, M, D, L, T, Z	Zählwert setzen

Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
ZW	PV	WORD	E, A, M, D, L oder Konstante	Zählwert, BCD-kodierte Zählerkonstante: C#<Wert>, <Wert> zwischen 0 und 999
R	R	BOOL	E, A, M, D, L, T, Z	Zählwert rücksetzen
DUAL	CV	WORD	E, A, M, D, L	aktueller Zählwert, hexadezimal
DEZ	CV_BCD	WORD	E, A, M, D, L	aktueller Zählwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status des Zählers

Statuswort für: ZAEHLER	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird der Zähler `Z1` auf den Wert "250" gesetzt. Wenn der Signalzustand am Eingang `E4.0` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` um 1 erhöht. Wenn der Signalzustand am Eingang `E7.7` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` um 1 vermindert. Wenn der Signalzustand am Eingang `E3.3` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` auf "0" gesetzt.

Wenn der Wert des Zählers `Z1` größer als "0" ist, ist `A2.0` = 1. Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zählwert.

5.5.3 Parametrieren und Vorwärtszählen - Z_VORW

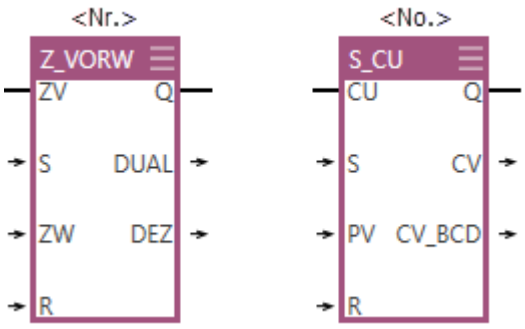
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird der Zähler auf den Zählwert *ZW* gesetzt. Wenn der Eingang *R* den Signalzustand "1" führt, wird der Zählwert *ZW* auf "0" gesetzt.

Bei jeder steigenden Flanke am Eingang *ZV* wird der Zählwert um 1 erhöht. Der Zählwert wird nicht weiter erhöht, wenn er "999" erreicht hat.

Wenn der Zählwert gesetzt wird und der Eingang *ZV* den Signalzustand "1" führt, wird der Zählwert einmalig im nächsten Zyklus erhöht, auch wenn kein Flankenwechsel eingetreten ist.

Wenn der Zählwert größer als "0" ist, führt der Ausgang *Q* den Wert "1". Wenn der Zählwert gleich "0" ist, führt der Ausgang *Q* den Wert "0".

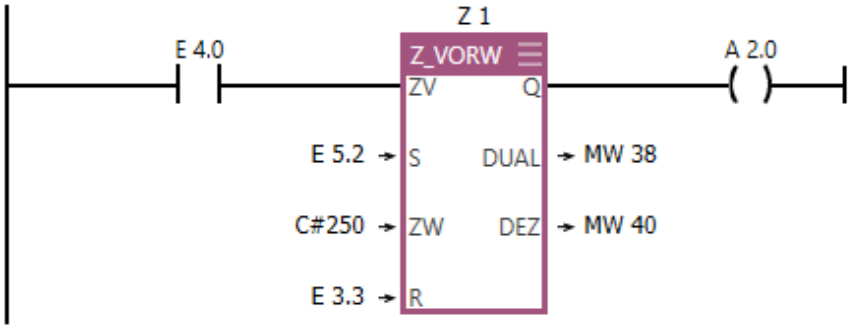
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	COUNTER	Z	Nummer des Zählers, Bereich ist abhängig von der CPU
ZV	CU	BOOL	E, A, M, D, L	Vorwärtszählen
S	S	BOOL	E, A, M, D, L, T, Z	Zählwert setzen
ZW	PV	WORD	E, A, M, D, L oder Konstante	Zählwert, BCD-kodierte Zählerkonstante: C#<Wert>, <Wert> zwischen 0 und 999
R	R	BOOL	E, A, M, D, L, T, Z	Zählwert rücksetzen
DUAL	CV	WORD	E, A, M, D, L	aktueller Zählwert, hexadezimal
DEZ	CV_BCD	WORD	E, A, M, D, L	aktueller Zählwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status des Zählers

Statuswort für: Z_VORW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird der Zähler `Z1` auf den Wert "250" gesetzt. Wenn der Signalzustand am Eingang `E4.0` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` um 1 erhöht. Wenn der Signalzustand am Eingang `E3.3` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` auf "0" gesetzt.

Wenn der Wert des Zählers `Z1` größer als "0" ist, ist `A2.0` = 1. Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zählwert.

5.5.4 Parametrieren und Rückwärtszählen - Z_RUECK

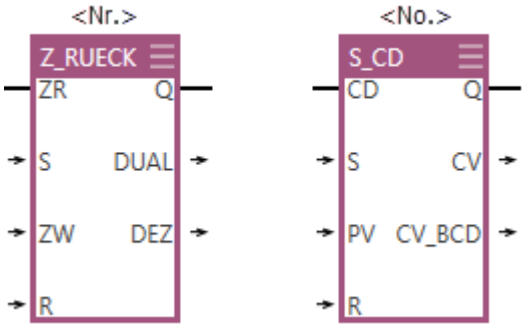
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird der Zähler auf den Zählwert *ZW* gesetzt. Wenn der Eingang *R* den Signalzustand "1" führt, wird der Zählwert *ZW* auf "0" gesetzt.

Bei jeder steigenden Flanke am Eingang *ZR* wird der Zählwert um 1 vermindert. Der Zählwert wird nicht weiter vermindert, wenn er "0" erreicht hat.

Wenn der Zählwert gesetzt wird und der Eingang *ZR* den Signalzustand "1" führt, wird der Zählwert einmalig im nächsten Zyklus vermindert, auch wenn kein Flankenwechsel eingetreten ist.

Wenn der Zählwert größer als "0" ist, führt der Ausgang *Q* den Wert "1". Wenn der Zählwert gleich "0" ist, führt der Ausgang *Q* den Wert "0".

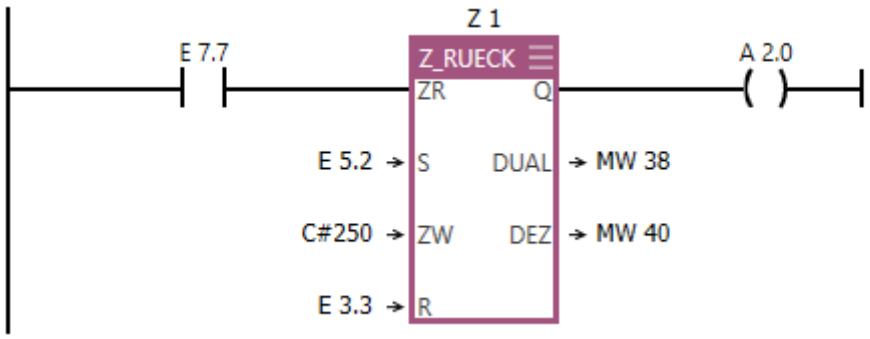
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	COUNTER	Z	Nummer des Zählers, Bereich ist abhängig von der CPU
ZR	CD	BOOL	E, A, M, D, L	Rückwärtszählen
S	S	BOOL	E, A, M, D, L, T, Z	Zählwert setzen
ZW	PV	WORD	E, A, M, D, L oder Konstante	Zählwert, BCD-kodierte Zählerkonstante: C#<Wert>, <Wert> zwischen 0 und 999
R	R	BOOL	E, A, M, D, L, T, Z	Zählwert rücksetzen
DUAL	CV	WORD	E, A, M, D, L	aktueller Zählwert, hexadezimal
DEZ	CV_BCD	WORD	E, A, M, D, L	aktueller Zählwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status des Zählers

Statuswort für: Z_RUECK	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



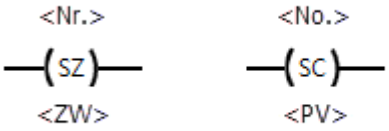
Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird der Zähler `Z1` auf den Wert "250" gesetzt. Wenn der Signalzustand am Eingang `E7.7` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` um 1 vermindert. Wenn der Signalzustand am Eingang `E3.3` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` auf "0" gesetzt.

Wenn der Wert des Zählers `Z1` größer als "0" ist, ist `A2.0` = 1. Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zählwert.

5.5.5 Setzen des Zähleranfangswerts - --(SZ)

Die Operation "Setzen des Zähleranfangswerts" setzt den Zählwert eines Zählers. Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird der Zähler auf den Wert *ZW* gesetzt.

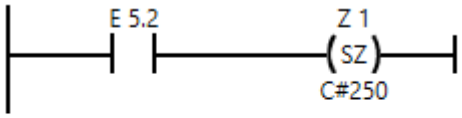
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	COUNTER	Z	Nummer des Zählers, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Zählwert setzen
ZW	PV	WORD	E, A, M, D, L oder Konstante	Zählwert, BCD-kodierte Zählerkonstante: C#<Wert>, <Wert> zwischen 0 und 999

Statuswort für: SZ	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	-	-	0

Beispiel

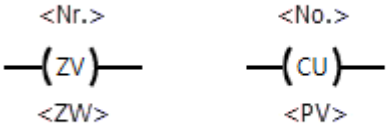


Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird der Zähler `Z1` auf den Wert "250" gesetzt.

5.5.6 Vorwärtszählen - --(ZV)

Die Operation "Vorwärtszählen" erhöht den Zählwert eines Zählers. Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird der Zählwert um 1 erhöht. Der Zählwert wird nicht weiter erhöht, wenn er "999" erreicht hat.

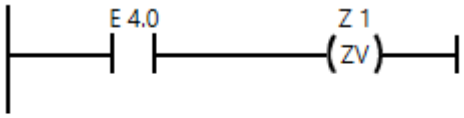
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	COUNTER	Z	Nummer des Zählers, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Vorwärtszählen

Statuswort für: ZV	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel

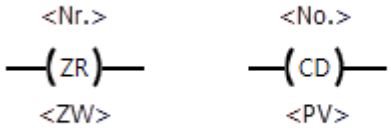


Wenn der Signalzustand am Eingang `E4.0` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` um 1 erhöht.

5.5.7 Rückwärtszählen - --(ZR)

Die Operation "Rückwärtszählen" vermindert den Zählwert eines Zählers. Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird der Zählwert um 1 vermindert. Der Zählwert wird nicht weiter vermindert, wenn er "0" erreicht hat.

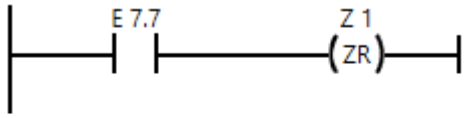
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	COUNTER	Z	Nummer des Zählers, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Rückwärtszählen

Statuswort für: ZR	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	-	-	0

Beispiel



Wenn der Signalzustand am Eingang `E7.7` von "0" nach "1" wechselt, wird der Wert des Zählers `Z1` um 1 vermindert.

5.6 Festpunkt-Funktionen

5.6.1 Überblick

Mit den Festpunkt-Funktionen können Sie arithmetische Operationen mit zwei Ganzzahlen durchführen.

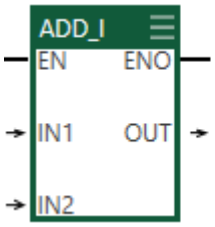
Operation	Festpunkt-Funktion
ADD_I	ganze Zahlen addieren (16 Bit)
SUB_I	ganze Zahlen subtrahieren (16 Bit)
MUL_I	ganze Zahlen multiplizieren (16 Bit)
DIV_I	ganze Zahlen dividieren (16 Bit)

Operation	Festpunkt-Funktion
ADD_DI	ganze Zahlen addieren (32 Bit)
SUB_DI	ganze Zahlen subtrahieren (32 Bit)
MUL_DI	ganze Zahlen multiplizieren (32 Bit)
DIV_DI	ganze Zahlen dividieren (32 Bit)
MOD_DI	Divisionsrest gewinnen (32 Bit)

5.6.2 Ganze Zahlen addieren (16 Bit) - ADD_I

Mit der Operation "Ganze Zahlen addieren (16 Bit)" können Sie zwei 16 Bit-Festpunktzahlen an den Eingängen *IN1* und *IN2* addieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

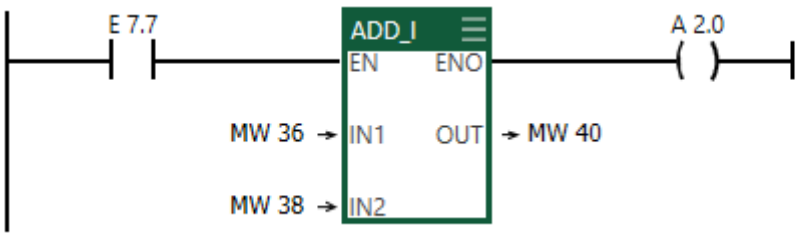
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (16 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	INT	E, A, M, D, L oder Konstante	erster Summand
IN2	INT	E, A, M, D, L oder Konstante	zweiter Summand
OUT	INT	E, A, M, D, L	Additionsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ADD_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

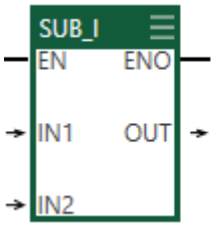


Wenn E7.7 = 1 ist, werden die Ganzzahlen (16 Bit) in den Merkerwörtern MW36 und MW38 addiert. Das Ergebnis wird im Merkerwort MW40 gespeichert. Wenn die Addition ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

5.6.3 Ganze Zahlen subtrahieren (16 Bit) - SUB_I

Mit der Operation "Ganze Zahlen subtrahieren (16 Bit)" können Sie 16 Bit-Festpunktzahlen subtrahieren. Dabei wird der Wert am Eingang *IN2* vom Wert am Eingang *IN1* subtrahiert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

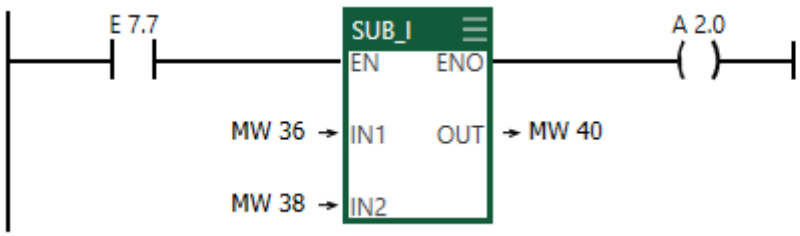
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (16 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	INT	E, A, M, D, L oder Konstante	Minuend
IN2	INT	E, A, M, D, L oder Konstante	Subtrahend
OUT	INT	E, A, M, D, L	Subtraktionsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SUB_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

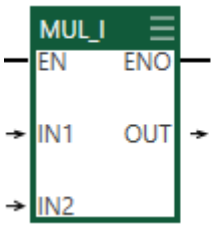


Wenn E7.7 = 1 ist, wird die Ganzzahl (16 Bit) im Merkerwort MW38 von der Ganzzahl (16 Bit) im Merkerwort MW36 subtrahiert. Das Ergebnis wird im Merkerwort MW40 gespeichert. Wenn die Subtraktion ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

5.6.4 Ganze Zahlen multiplizieren (16 Bit) - MUL_I

Mit der Operation "Ganze Zahlen multiplizieren (16 Bit)" können Sie zwei 16 Bit-Festpunktzahlen an den Eingängen *IN1* und *IN2* miteinander multiplizieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

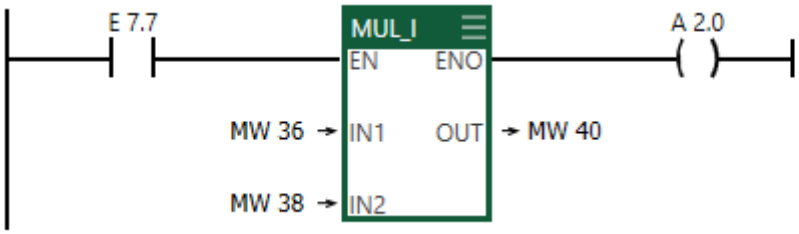
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (16 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	INT	E, A, M, D, L oder Konstante	Multiplikand
IN2	INT	E, A, M, D, L oder Konstante	Multiplikator
OUT	INT	E, A, M, D, L	Multiplikationsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: MUL_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

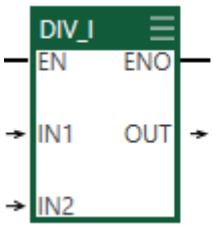


Wenn `E7.7` = 1 ist, werden die Ganzzahlen (16 Bit) in den Merkerwörtern `MW36` und `MW38` miteinander multipliziert. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Multiplikation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.6.5 Ganze Zahlen dividieren (16 Bit) - DIV_I

Mit der Operation "Ganze Zahlen dividieren (16 Bit)" können Sie 16 Bit-Festpunktzahlen dividieren. Dabei wird der Wert am Eingang *IN1* durch den Wert am Eingang *IN2* dividiert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis (Quotient) wird im Ausgang *OUT* gespeichert. Der Divisionsrest wird nicht gespeichert.

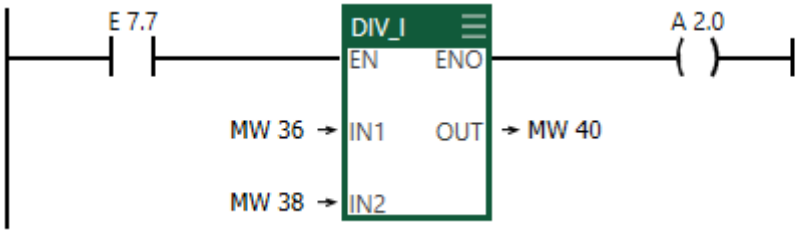
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (16 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	INT	E, A, M, D, L oder Konstante	Dividend
IN2	INT	E, A, M, D, L oder Konstante	Divisor
OUT	INT	E, A, M, D, L	Divisionsergebnis (Quotient)
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: DIV_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

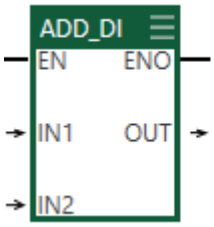


Wenn `E7.7` = 1 ist, wird die Ganzzahl (16 Bit) im Merkerwort `MW36` durch die Ganzzahl (16 Bit) im Merkerwort `MW38` dividiert. Das Ergebnis (Quotient) wird im Merkerwort `MW40` gespeichert. Wenn die Division ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.6.6 Ganze Zahlen addieren (32 Bit) - ADD_DI

Mit der Operation "Ganze Zahlen addieren (32 Bit)" können Sie zwei 32 Bit-Festpunktzahlen an den Eingängen *IN1* und *IN2* addieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

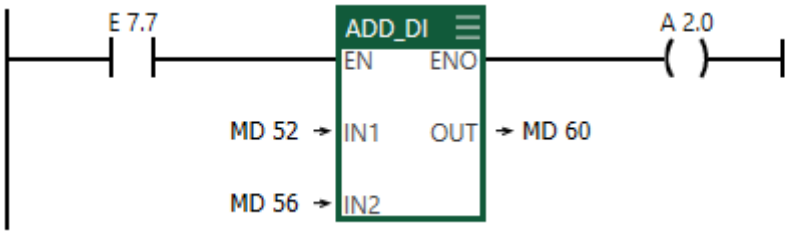
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (32 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	DINT	E, A, M, D, L oder Konstante	erster Summand
IN2	DINT	E, A, M, D, L oder Konstante	zweiter Summand
OUT	DINT	E, A, M, D, L	Additionsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ADD_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

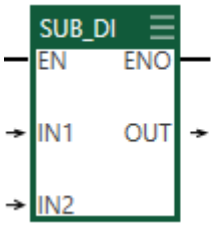


Wenn `E7.7` = 1 ist, werden die Ganzzahlen (32 Bit) in den Merkerdoppelwörtern `MD52` und `MD56` addiert. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Addition ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.6.7 Ganze Zahlen subtrahieren (32 Bit) - SUB_DI

Mit der Operation "Ganze Zahlen subtrahieren (32 Bit)" können Sie 32 Bit-Festpunktzahlen subtrahieren. Dabei wird der Wert am Eingang *IN2* vom Wert am Eingang *IN1* subtrahiert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

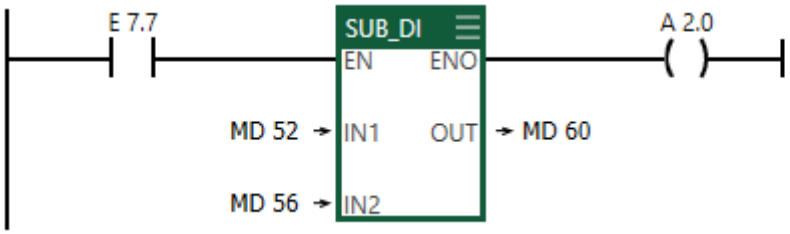
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (32 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	DINT	E, A, M, D, L oder Konstante	Minuend
IN2	DINT	E, A, M, D, L oder Konstante	Subtrahend
OUT	INT	E, A, M, D, L	Subtraktionsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SUB_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

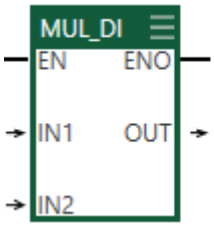


Wenn `E7.7` = 1 ist, wird die Ganzzahl (32 Bit) im Merkerdoppelwort `MD56` von der Ganzzahl (32 Bit) im Merkerdoppelwort `MD52` subtrahiert. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Subtraktion ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.6.8 Ganze Zahlen multiplizieren (32 Bit) - MUL_DI

Mit der Operation "Ganze Zahlen multiplizieren (32 Bit)" können Sie zwei 32 Bit-Festpunktzahlen an den Eingängen *IN1* und *IN2* miteinander multiplizieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

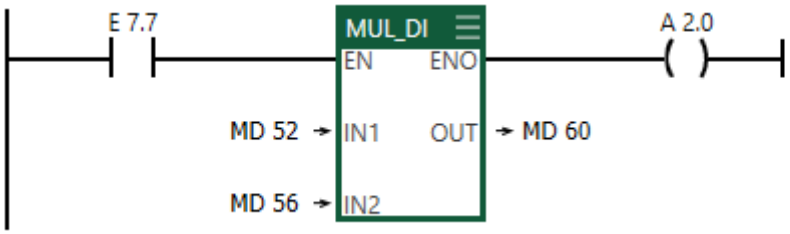
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (32 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	DINT	E, A, M, D, L oder Konstante	Multiplikand
IN2	DINT	E, A, M, D, L oder Konstante	Multiplikator
OUT	DINT	E, A, M, D, L	Multiplikationsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: MUL_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

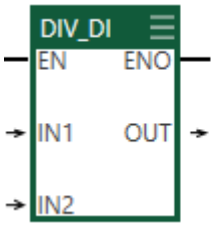


Wenn `E7.7` = 1 ist, werden die Ganzzahlen (32 Bit) in den Merkerdoppelwörtern `MD52` und `MD56` miteinander multipliziert. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Multiplikation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.6.9 Ganze Zahlen dividieren (32 Bit) - DIV_DI

Mit der Operation "Ganze Zahlen dividieren (32 Bit)" können Sie 32 Bit-Festpunktzahlen dividieren. Dabei wird der Wert am Eingang *IN1* durch den Wert am Eingang *IN2* dividiert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis (Quotient) wird im Ausgang *OUT* gespeichert. Der Divisionsrest wird nicht gespeichert.

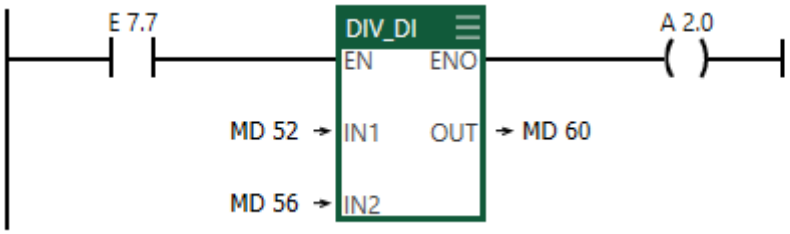
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (32 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	DINT	E, A, M, D, L oder Konstante	Dividend
IN2	DINT	E, A, M, D, L oder Konstante	Divisor
OUT	DINT	E, A, M, D, L	Divisionsergebnis (Quotient)
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: DIV_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

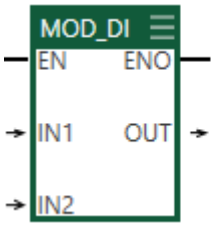


Wenn `E7.7` = 1 ist, wird die Ganzzahl (32 Bit) im Merkerdoppelwort `MD52` durch die Ganzzahl (32 Bit) im Merkerdoppelwort `MD56` dividiert. Das Ergebnis (Quotient) wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Division ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.6.10 Divisionsrest gewinnen (32 Bit) - MOD_DI

Mit der Operation "Divisionsrest gewinnen (32 Bit)" können Sie 32 Bit-Festpunktzahlen dividieren. Als Ergebnis erhalten Sie den Divisionsrest. Dabei wird der Wert am Eingang *IN1* durch den Wert am Eingang *IN2* dividiert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis (Divisionsrest) wird im Ausgang *OUT* gespeichert.

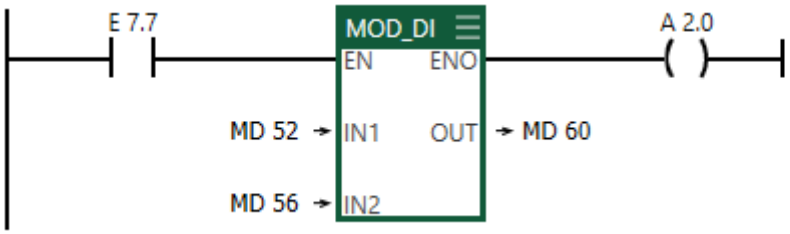
Wenn das Ergebnis außerhalb des zulässigen Bereichs für Ganzzahlen (32 Bit) liegt, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	INT	E, A, M, D, L oder Konstante	Dividend
IN2	INT	E, A, M, D, L oder Konstante	Divisor
OUT	INT	E, A, M, D, L	Divisionsrest
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: MOD_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel



Wenn `E7.7` = 1 ist, wird die Ganzzahl (32 Bit) im Merkerdoppelwort `MD52` durch die Ganzzahl (32 Bit) im Merkerdoppelwort `MD56` dividiert. Das Ergebnis (Divisionsrest) wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Division ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.7 Gleitpunkt-Funktionen

5.7.1 Überblick

Mit den Gleitpunkt-Funktionen können Sie arithmetische und trigonometrische Operationen mit einer bzw. zwei Gleitpunktzahlen durchführen.

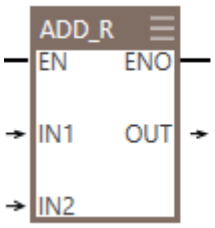
Operation	Gleitpunkt-Funktion
ADD_R	Gleitpunktzahlen addieren
SUB_R	Gleitpunktzahlen subtrahieren
MUL_R	Gleitpunktzahlen multiplizieren
DIV_R	Gleitpunktzahlen dividieren

Operation	Gleitpunkt-Funktion
ABS	Absolutwert einer Gleitpunktzahl bilden
SQRT	Quadratwurzel einer Gleitpunktzahl ziehen
SQR	Quadrat einer Gleitpunktzahl bilden
LN	natürlichen Logarithmus einer Gleitpunktzahl bilden
EXP	Exponentialwert einer Gleitpunktzahl bilden
SIN	Sinus einer Gleitpunktzahl bilden
COS	Cosinus einer Gleitpunktzahl bilden
TAN	Tangens einer Gleitpunktzahl bilden
ASIN	Arkussinus einer Gleitpunktzahl bilden
ACOS	Arkuscosinus einer Gleitpunktzahl bilden
ATAN	Arkustangens einer Gleitpunktzahl bilden

5.7.2 Gleitpunktzahlen addieren - ADD_R

Mit der Operation "Gleitpunktzahlen addieren" können Sie zwei Gleitpunktzahlen an den Eingängen *IN1* und *IN2* addieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

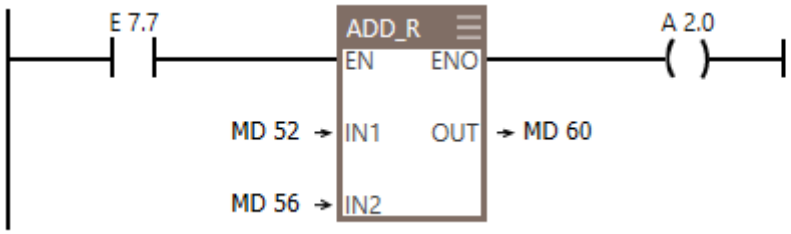
Wenn das Ergebnis oder einer der Eingänge keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	REAL	E, A, M, D, L oder Konstante	erster Summand
IN2	REAL	E, A, M, D, L oder Konstante	zweiter Summand
OUT	REAL	E, A, M, D, L	Additionsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ADD_R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

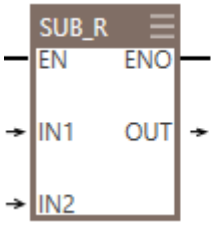


Wenn `E7.7` = 1 ist, werden die Gleitpunktzahlen in den Merkerdoppelwörtern `MD52` und `MD56` addiert. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Addition ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.7.3 Gleitpunktzahlen subtrahieren - SUB_R

Mit der Operation "Gleitpunktzahlen subtrahieren" können Sie Gleitpunktzahlen subtrahieren. Dabei wird der Wert am Eingang *IN2* vom Wert am Eingang *IN1* subtrahiert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

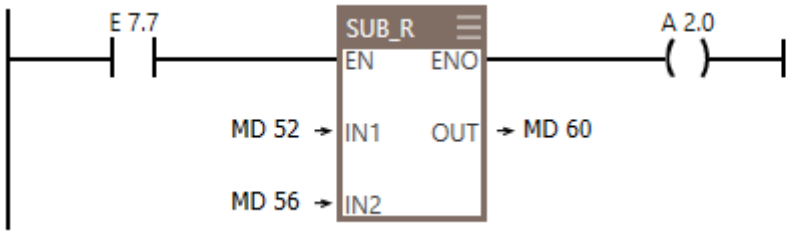
Wenn das Ergebnis oder einer der Eingänge keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	REAL	E, A, M, D, L oder Konstante	Minuend
IN2	REAL	E, A, M, D, L oder Konstante	Subtrahend
OUT	REAL	E, A, M, D, L	Subtraktionsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SUB_R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

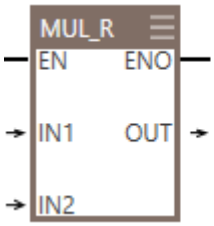


Wenn `E7.7` = 1 ist, wird die Gleitpunktzahl im Merkerdoppelwort `MD56` von der Gleitpunktzahl im Merkerdoppelwort `MD52` subtrahiert. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Subtraktion ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.7.4 Gleitpunktzahlen multiplizieren - MUL_R

Mit der Operation "Ganze Zahlen multiplizieren (32 Bit)" können Sie zwei 32 Bit-Festpunktzahlen an den Eingängen *IN1* und *IN2* miteinander multiplizieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

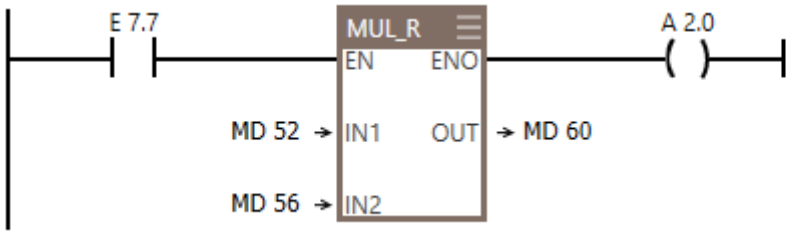
Wenn das Ergebnis oder einer der Eingänge keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	REAL	E, A, M, D, L oder Konstante	Multiplikand
IN2	REAL	E, A, M, D, L oder Konstante	Multiplikator
OUT	REAL	E, A, M, D, L	Multiplikationsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: MUL_R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

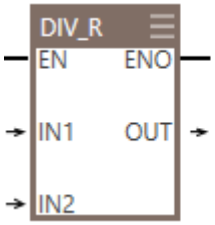


Wenn `E7.7` = 1 ist, werden die Gleitpunktzahlen in den Merkerdoppelwörtern `MD52` und `MD56` miteinander multipliziert. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Multiplikation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.7.5 Gleitpunktzahlen dividieren - DIV_R

Mit der Operation "Gleitpunktzahlen dividieren" können Sie Gleitpunktzahlen dividieren. Dabei wird der Wert am Eingang *IN1* durch den Wert am Eingang *IN2* dividiert. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

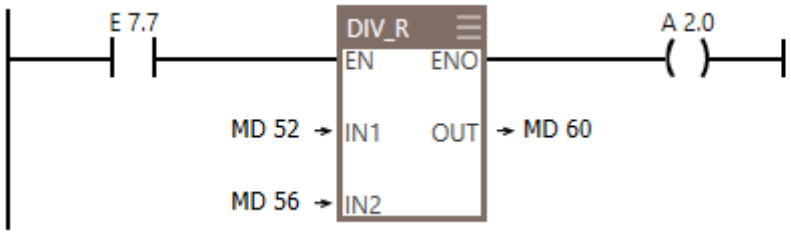
Wenn das Ergebnis oder einer der Eingänge keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	REAL	E, A, M, D, L oder Konstante	Dividend
IN2	REAL	E, A, M, D, L oder Konstante	Divisor
OUT	REAL	E, A, M, D, L	Divisionsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: DIV_R	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

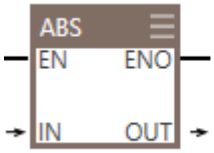
Beispiel



Wenn `E7.7` = 1 ist, wird die Gleitpunktzahl im Merkerdoppelwort `MD52` durch die Gleitpunktzahl im Merkerdoppelwort `MD56` dividiert. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Division ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.7.6 Absolutwert einer Gleitpunktzahl - ABS

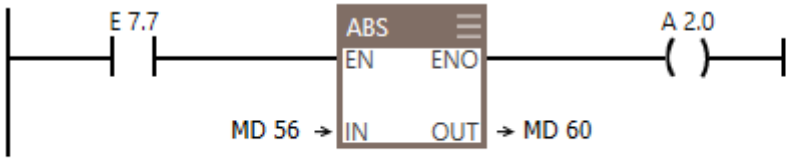
Mit der Operation "Absolutwert einer Gleitpunktzahl" können Sie den Absolutwert einer Gleitpunktzahl am Eingang *IN* bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Absolutwert der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ABS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel



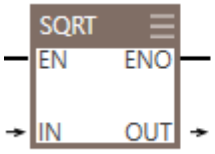
Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der Absolutwert gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel: `MD56` = -25,75, `MD60` = 25,75

5.7.7 Quadratwurzel einer Gleitpunktzahl - SQRT

Mit der Operation "Quadratwurzel einer Gleitpunktzahl" können Sie die Quadratwurzel einer Gleitpunktzahl am Eingang *IN* ziehen. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

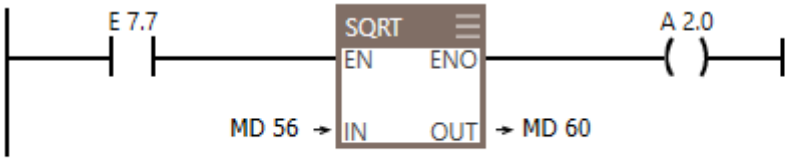
Wenn das Ergebnis oder der Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Quadratwurzel der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SQRT	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

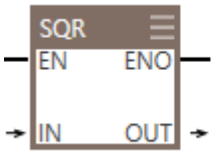


Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` die Quadratwurzel gezogen. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.7.8 Quadrat einer Gleitpunktzahl - SQR

Mit der Operation "Quadrat einer Gleitpunktzahl" können Sie eine Gleitpunktzahl am Eingang *IN* quadrieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

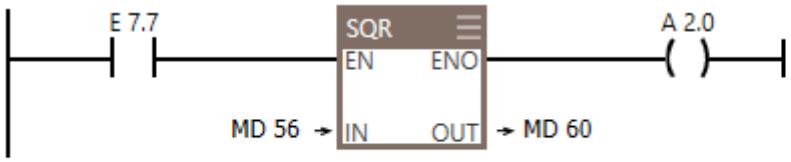
Wenn das Ergebnis oder der Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Quadrat der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SQR	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

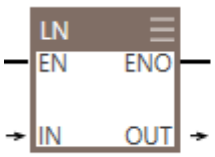


Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` das Quadrat gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.7.9 Natürlichen Logarithmus einer Gleitpunktzahl - LN

Mit der Operation "Natürlichen Logarithmus einer Gleitpunktzahl" können Sie den natürlichen Logarithmus einer Gleitpunktzahl am Eingang *IN* bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

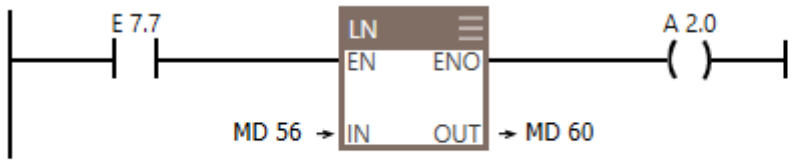
Wenn das Ergebnis oder der Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Zahl
OUT	REAL	E, A, M, D, L	natürlicher Logarithmus der Zahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: LN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

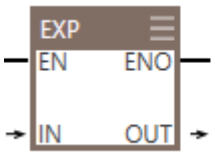


Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der natürliche Logarithmus gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.7.10 Exponentialwert einer Gleitpunktzahl - EXP

Mit der Operation "Exponentialwert einer Gleitpunktzahl" können Sie den Exponentialwert einer Gleitpunktzahl am Eingang *IN* auf Basis der Eulerschen Zahl *e* bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

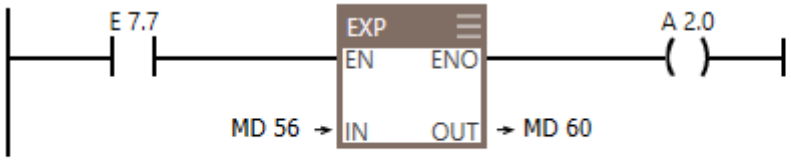
Wenn das Ergebnis oder der Eingang keine Gleitpunktzahl ist, haben die Bits "OV" und "OS" den Wert "1" und *ENO* ist gleich "0".



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Zahl
OUT	REAL	E, A, M, D, L	Exponent der Zahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: EXP	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

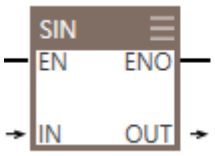
Beispiel



Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der Exponentialwert gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.7.11 Sinus einer Gleitpunktzahl - SIN

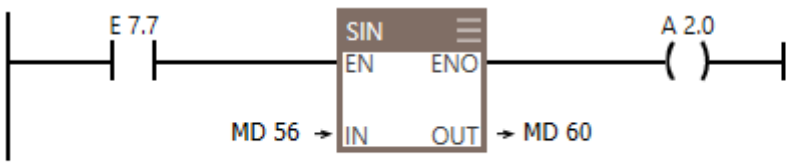
Mit der Operation "Sinus einer Gleitpunktzahl" können Sie von einem Winkel (Bogenmaß) am Eingang *IN* den Sinus (sin) bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Sinus der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SIN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

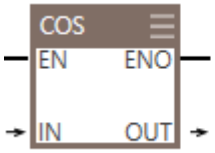
Beispiel



Wenn E7.7 = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort MD56 der Sinus gebildet. Das Ergebnis wird im Merkerdoppelwort MD60 gespeichert. Wenn die Operation ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

5.7.12 Cosinus einer Gleitpunktzahl - COS

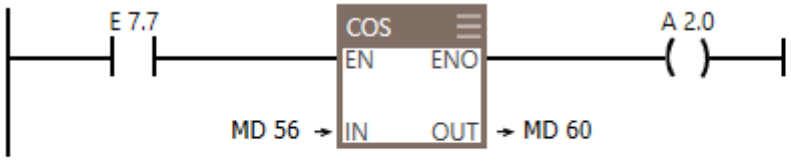
Mit der Operation "Cosinus einer Gleitpunktzahl" können Sie von einem Winkel (Bogenmaß) am Eingang *IN* den Cosinus (cos) bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Cosinus der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: COS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

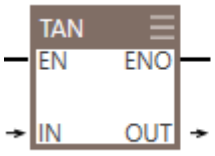
Beispiel



Wenn E7.7 = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort MD56 der Cosinus gebildet. Das Ergebnis wird im Merkerdoppelwort MD60 gespeichert. Wenn die Operation ausgeführt worden ist, ist A2.0 = 1 (ENO = EN).

5.7.13 Tangens einer Gleitpunktzahl - TAN

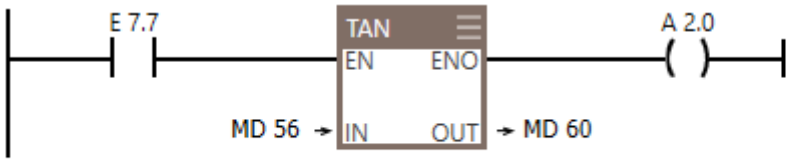
Mit der Operation "Tangens einer Gleitpunktzahl" können Sie von einem Winkel (Bogenmaß) am Eingang *IN* den Tangens (tan) bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Tangens der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: TAN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

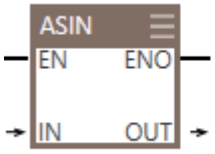


Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der Tangens gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.7.14 Arkussinus einer Gleitpunktzahl - ASIN

Mit der Operation "Arkussinus einer Gleitpunktzahl" können Sie aus einer Gleitpunktzahl am Eingang *IN* den Arkussinus ($\sin^{-1}$) bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis ist ein Winkel im Bogenmaß und wird im Ausgang *OUT* gespeichert.

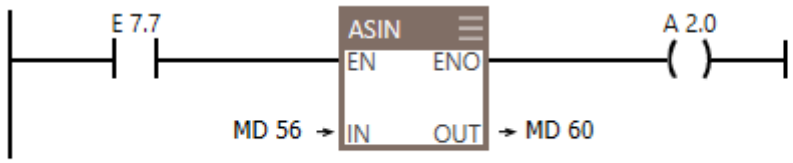
Der Ausgabewert liegt im folgenden Bereich: $-\frac{1}{2}\pi \leq \text{Arkussinus} \leq +\frac{1}{2}\pi$



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Arkussinus der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ASIN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

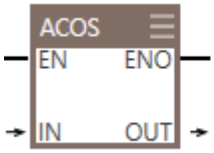


Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der Arkussinus gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.7.15 Arkuscosinus einer Gleitpunktzahl - ACOS

Mit der Operation "Arkuscosinus einer Gleitpunktzahl" können Sie aus einer Gleitpunktzahl am Eingang *IN* den Arkuscosinus ($\cos^{-1}$) bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis ist ein Winkel im Bogenmaß und wird im Ausgang *OUT* gespeichert.

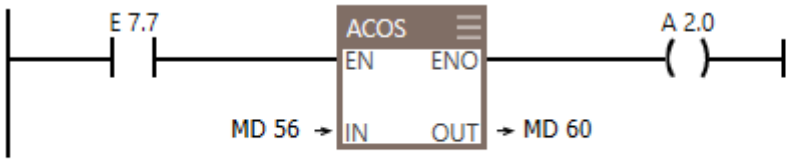
Der Ausgabewert liegt im folgenden Bereich: $0 \leq$ Arkuscosinus $\leq + \pi$



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Arkuscosinus der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ACOS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel

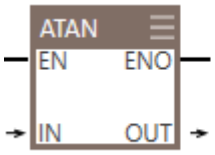


Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der Arkuscosinus gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.7.16 Arkustangens einer Gleitpunktzahl - ATAN

Mit der Operation "Arkustangens einer Gleitpunktzahl" können Sie aus einer Gleitpunktzahl am Eingang *IN* den Arkustangens ($\tan^{-1}$) bilden. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis ist ein Winkel im Bogenmaß und wird im Ausgang *OUT* gespeichert.

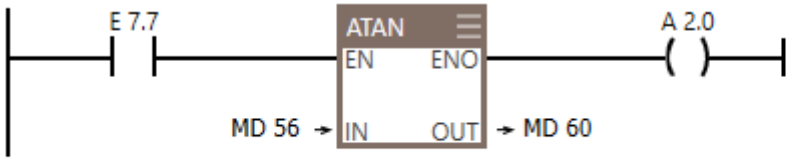
Der Ausgabewert liegt im folgenden Bereich: $-\frac{1}{2}\pi \leq$ Arkustangens $\leq +\frac{1}{2}\pi$



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	REAL	E, A, M, D, L oder Konstante	Gleitpunktzahl
OUT	REAL	E, A, M, D, L	Arkustangens der Gleitpunktzahl
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ATAN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	✓	0	✓	✓	1

Beispiel



Wenn `E7.7` = 1 ist, wird von der Gleitpunktzahl im Merkerdoppelwort `MD56` der Arkustangens gebildet. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

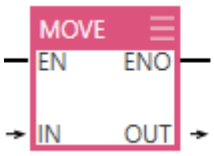
5.8 Verschieben

5.8.1 Wert übertragen - MOVE

Mit der Operation "Wert übertragen" können Sie den Wert am Eingang *IN* in den Ausgangsoperanden *OUT* kopieren. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Sie können alle elementaren Datentypen mit einer Länge von 8, 16 oder 32 Bit verwenden, z. B. BYTE, INT, WORD, REAL. Erweiterte Datentypen, wie z. B. Felder oder Strukturen, können jedoch nicht mit dieser Operation, sondern nur über die Systemfunktion "SFC 20 BLKMOV" kopiert werden.

Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.

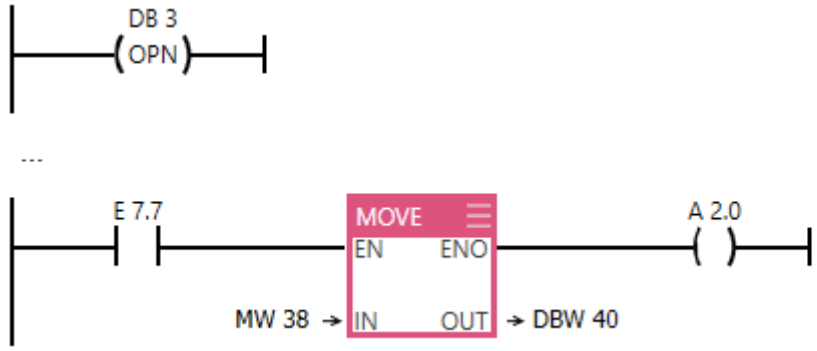
Wenn der Datentyp am Eingang **mehr** Bitstellen hat als der am Ausgang, werden die höherwertigen Bits abgeschnitten. Wenn der Datentyp am Eingang **weniger** Bitstellen hat als der am Ausgang, werden die höherwertigen Bits mit Nullen aufgefüllt.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	Elementare Datentypen (8, 16 oder 32 Bit)	E, A, M, D, L oder Konstante	Eingabewert
OUT	Elementare Datentypen (8, 16 oder 32 Bit)	E, A, M, D, L	Zieladresse
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: MOVE	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	-	-	-	-	0	1	1	1

Beispiel



Im ersten Netzwerk wird der Datenbaustein `DB3` mit der Operation "OPN" geöffnet.

Wenn `E7.7` = 1 ist, wird der Inhalt vom Merkerwort `MW38` in das Datenwort `DBW40` des Datenbausteins `DB3` kopiert. Wenn die Umwandlung ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.9 Programmsteuerung

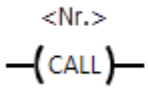
5.9.1 Überblick

Mit den Operationen zur Programmsteuerung können Sie Bausteine aufrufen und verlassen sowie innerhalb eines Bausteins an eine Sprungmarke springen.

Operation	Programmsteuerung
--(CALL)	Funktion aufrufen (FC/SFC)
--(RET)	aktuellen Baustein verlassen
--(OPN)	Datenbaustein öffnen
--(JMP)	zu einer Sprungmarke springen (absolut oder wenn 1)
--(JMPN)	zu einer Sprungmarke springen (wenn 0)
Label	Ziel für die Sprungoperationen "JMP" bzw. "JMPN" setzen

5.9.2 Baustein aufrufen - --(CALL)

Mit der Operation "Baustein aufrufen" können Sie Funktionen (FC) oder Systemfunktion (SFC) aufrufen. Die Programmbearbeitung wird in der aufgerufenen Funktion bzw. dem aufgerufenen Funktionsbaustein fortgesetzt.



Parameter	Datentyp	Speicherbereich	Beschreibung
Nr.	-	-	Nummer des FC oder SFC, Bereich ist abhängig von der CPU

Aufruf ohne Bedingung (absoluter Aufruf)

Wenn Sie am Eingang des Aufruf-Elements keinen Parameter angeben (keine Verknüpfung vor dem Element), wird der Baustein ohne Bedingung aufgerufen.

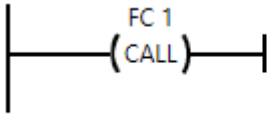
Statuswort für: CALL	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	0	0	1	-	0

Aufruf, wenn VKE = 1 (bedingter Aufruf)

Wenn vor dem Aufruf-Element Verknüpfungen angeben sind und das Verknüpfungsergebnis VKE = 1 ist, wird der Baustein aufgerufen.

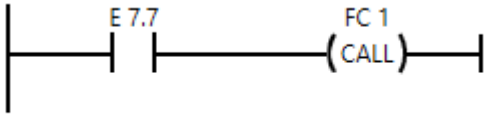
Statuswort für: CALL	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	0	0	1	1	0

Beispiel (absoluter Aufruf)



Am Eingang des Aufruf-Elements ist kein Parameter angeben (keine Verknüpfung vor dem Element). Die Funktion `FC1` wird immer aufgerufen. Die Programmbearbeitung wird im Baustein `FC1` fortgesetzt.

Beispiel (bedingter Aufruf)



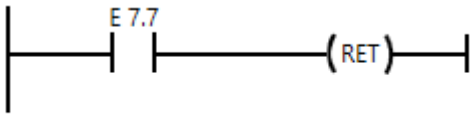
Nur wenn `E7.7` = 1 ist, wird die Funktion `FC1` aufgerufen. Die Programmbearbeitung wird im Baustein `FC1` fortgesetzt.

5.9.3 Springe zurück - --(RET)

Mit der Operation "Springe zurück" können Sie den aktuellen Baustein verlassen.

Statuswort für: DB	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	-	-	-	-	-	-	-	-

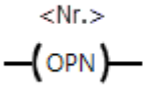
Beispiel



Wenn `E7.7` = 1 ist, wird der Baustein verlassen.

5.9.4 Datenbaustein öffnen - --(OPN)

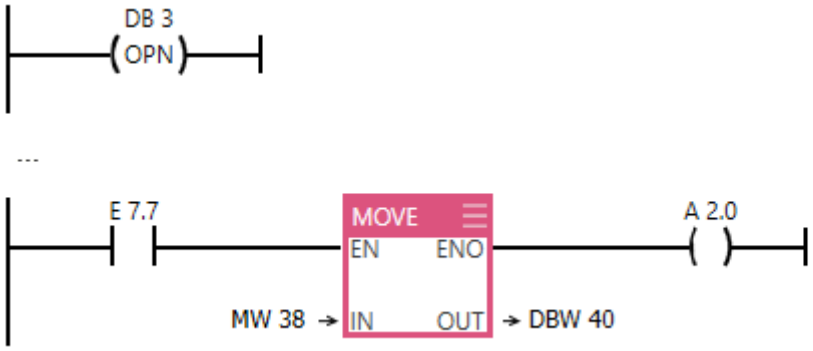
Mit der Operation "Datenbaustein öffnen" können Sie einen Datenbaustein als globalen Datenbaustein (DB) oder als Instanz-Datenbaustein (DI) öffnen, um anschließend darauf zuzugreifen. Die Operation überträgt die DB-Nummer in das DB- bzw. DI-Register. Alle folgenden DB- bzw. DI-Operationen greifen auf den geöffneten Baustein zu.



Parameter	Datentyp	Speicherbereich	Beschreibung
Nr.	-	-	Nummer des DB oder DI, Bereich ist abhängig von der CPU

Statuswort für: DB	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	-	-	-	-

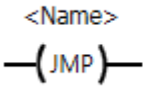
Beispiel



Im ersten Netzwerk wird der Datenbaustein `DB3` geöffnet. Alle folgenden Datenbaustein-Operation beziehen sich auf den geöffneten Datenbaustein `DB3`. Wenn `E7.7` = 1 ist, wird der Inhalt vom Merkerwort `MW38` in das Datenwort `DBW40` des Datenbausteins `DB3` kopiert.

5.9.5 Zur Sprungmarke springen (absolut oder wenn 1) - --(JMP)

Mit der Operation "Zur Sprungmarke springen (absolut oder wenn 1)" können Sie innerhalb des aktuellen Bausteins zu der im *Operand* angegebenen Sprungmarke springen, um die Programmbearbeitung dort fortzusetzen.



Parameter	Datentyp	Speicherbereich	Beschreibung
Name	-	-	Sprungmarke, zu der absolut oder bei VKE = 1 gesprungen wird

Sprung ohne Bedingung (absoluter Sprung)

Wenn Sie am Eingang des Sprung-Elements keinen Parameter angeben (keine Verknüpfung vor dem Element), wird ohne Bedingung zur Sprungmarke gesprungen.

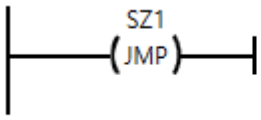
Statuswort für: JMP	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	-	-	-	-

Sprung, wenn VKE = 1 (bedingter Sprung)

Wenn vor dem Sprung-Element Verknüpfungen angeben sind und das Verknüpfungsergebnis VKE = 1 ist, wird zur Sprungmarke gesprungen.

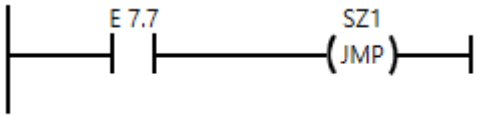
Statuswort für: JMP	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	1	1	0

Beispiel (absoluter Sprung)



Am Eingang des Sprung-Elements ist kein Parameter angeben (keine Verknüpfung vor dem Element). Der Sprung zur Sprungmarke `SZ1` wird immer ausgeführt.

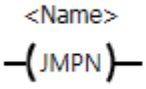
Beispiel (bedingter Sprung)



Nur wenn `E7.7` = 1 ist, wird zur Sprungmarke `SZ1` gesprungen.

5.9.6 Zur Sprungmarke springen (wenn 0) - --(JMPN)

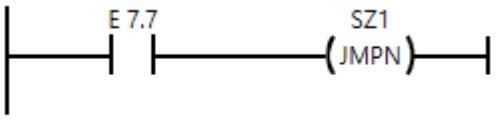
Mit der Operation "Zur Sprungmarke springen (wenn 0)" können Sie innerhalb des aktuellen Bausteins zu der im *Operand* angegebenen Sprungmarke verzweigen, um die Programmbearbeitung dort fortzusetzen. Wenn das Verknüpfungsergebnis VKE vor dem Sprung-Element gleich "0" ist, wird zur Sprungmarke gesprungen.



Parameter	Datentyp	Speicherbereich	Beschreibung
Name	-	-	Sprungmarke, zu der bei VKE = 0 gesprungen wird

Statuswort für: JMPN	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	1	1	0

Beispiel

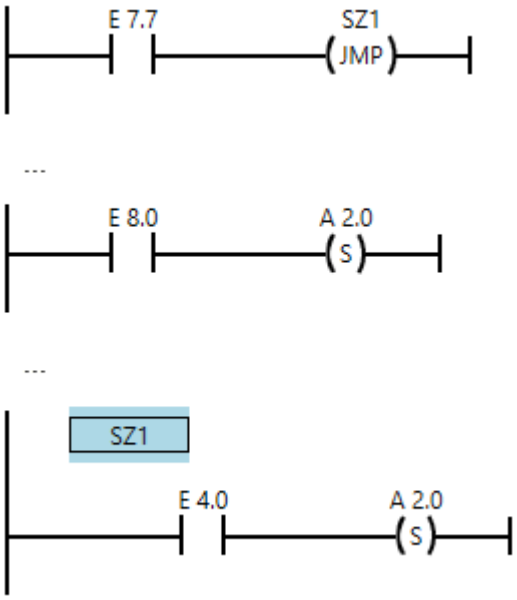


Nur wenn `E7.7` = 0 ist, wird zur Sprungmarke `SZ1` gesprungen.

5.9.7 Sprungmarke - Label

Die Sprungmarke bezeichnet das Ziel der Sprungoperationen "JMP" bzw. "JMPN". Das in *Label* angegebene Sprungziel darf aus maximal vier Zeichen bestehen. Das erste Zeichen muss ein Buchstabe sein, die anderen Zeichen dürfen Buchstaben oder Zahlen sein.

Beispiel



Nur wenn `E7.7` im ersten Netzwerk gleich "1" ist, wird zur Sprungmarke `SZ1` gesprungen. Die Programmbearbeitung wird mit der Setze-Operation fortgesetzt, die den Eingang `E4.0` auswertet.

Wenn jedoch `E7.7` gleich "0" ist, wird **nicht** zur Sprungmarke `SZ1` gesprungen. Die Programmbearbeitung wird mit der Setze-Operation im nächsten Netzwerk fortgesetzt - hier die Operation, die den Eingang `E8.0` auswertet.

5.10 Schieben/Rotieren

5.10.1 Überblick

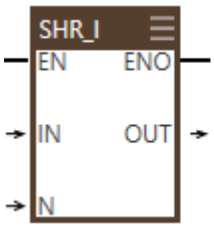
Mit den Schiebe-/Rotier-Operationen können Sie die Bits eines Operanden nach links oder rechts schieben bzw. rotieren.

Operation	Schieben/Rotieren
SHR_I	Ganzzahl (16 Bit) nach rechts schieben
SHR_DI	Ganzzahl (32 Bit) nach rechts schieben
SHL_W	Bitfolge (16 Bit) nach links schieben
SHR_W	Bitfolge (16 Bit) nach rechts schieben
SHL_DW	Bitfolge (32 Bit) nach links schieben
SHR_DW	Bitfolge (32 Bit) nach rechts schieben
ROL_DW	Bitfolge (32 Bit) nach links rotieren
ROR_DW	Bitfolge (32 Bit) nach rechts rotieren

5.10.2 Ganzzahl (16 Bit) rechts schieben - SHR_I

Mit der Operation "Ganzzahl (16 Bit) rechts schieben" können Sie die Bits 0 - 15 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach rechts schieben. Die Stellen, die durch die Schiebeoperation frei werden, werden entweder mit Nullen oder mit dem Signalzustand des Vorzeichenbits aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

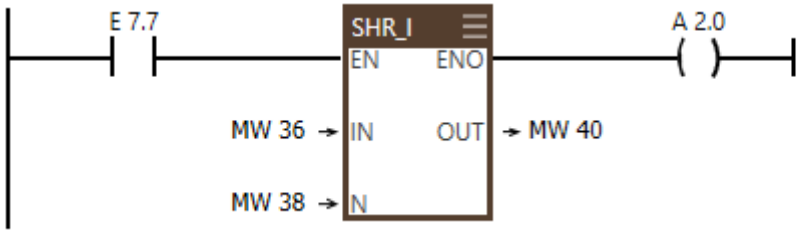
Die Bits am Eingang *IN* werden um maximal 16 Bitstellen geschoben, auch wenn der Wert *N* größer als "16" ist. Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	INT	E, A, M, D, L	Ganzzahl (16 Bit), die geschoben werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die geschoben werden soll, maximal 16
OUT	INT	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SHR_I	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel

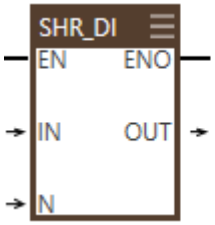


Wenn `E7.7` = 1 ist, wird die Ganzzahl (16 Bit) im Merkerwort `MW36` um die Anzahl an Bits nach rechts geschoben, die im Merkerwort `MW38` angegeben ist. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.10.3 Ganzzahl (32 Bit) rechts schieben - SHR_DI

Mit der Operation "Ganzzahl (32 Bit) rechts schieben" können Sie die Bits 0 - 32 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach rechts schieben. Die Stellen, die durch die Schiebeoperation frei werden, werden entweder mit Nullen oder mit dem Signalzustand des Vorzeichenbits aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

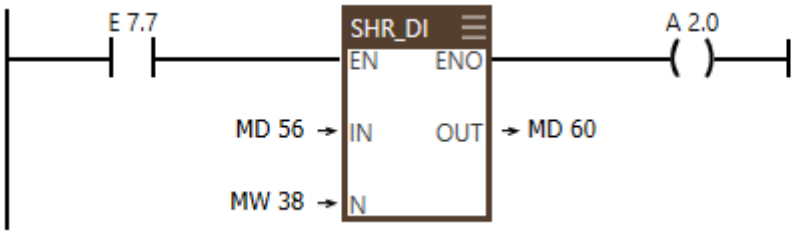
Die Bits am Eingang *IN* werden um maximal 32 Bitstellen geschoben, auch wenn der Wert *N* größer als "32" ist. Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	WORD	E, A, M, D, L	Ganzzahl (32 Bit), die geschoben werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die geschoben werden soll, maximal 32
OUT	WORD	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SHR_DI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel

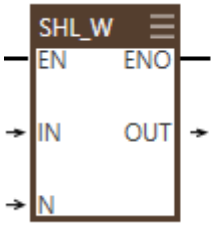


Wenn `E7.7` = 1 ist, wird die Ganzzahl (32 Bit) im Merkerdoppelwort `MD56` um die Anzahl an Bits nach rechts geschoben, die im Merkerwort `MW38` angegeben ist. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.10.4 16 Bit links schieben - SHL_W

Mit der Operation "16 Bit links schieben" können Sie die Bits 0 - 15 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach links schieben. Die Stellen, die durch die Schiebeoperation frei werden, werden mit Nullen aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

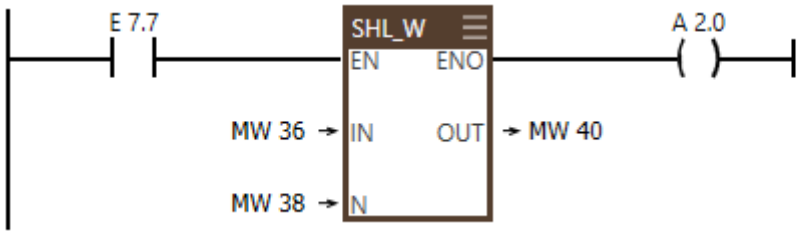
Wenn der Wert *N* größer als "16" ist, wird der Ausgang *OUT* auf den Wert "0" gesetzt. Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	WORD	E, A, M, D, L	Bitfolge (16 Bit), die geschoben werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die geschoben werden soll, maximal 16
OUT	WORD	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SHL_W	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel

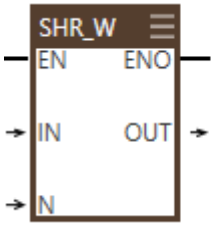


Wenn `E7.7` = 1 ist, wird die Bitfolge (16 Bit) im Merkerwort `MW36` um die Anzahl an Bits nach links geschoben, die im Merkerwort `MW38` angegeben ist. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.10.5 16 Bit rechts schieben - SHR_W

Mit der Operation "16 Bit rechts schieben" können Sie die Bits 0 - 15 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach rechts schieben. Die Stellen, die durch die Schiebeoperation frei werden, werden mit Nullen aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

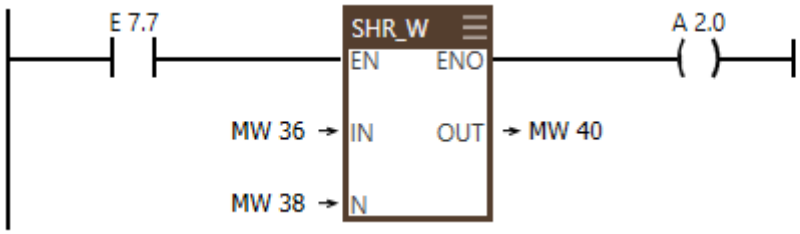
Wenn der Wert *N* größer als "16" ist, wird der Ausgang *OUT* auf den Wert "0" gesetzt. Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	WORD	E, A, M, D, L	Bitfolge (16 Bit), die geschoben werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die geschoben werden soll, maximal 16
OUT	WORD	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SHR_W	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel

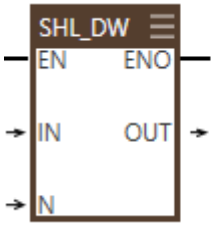


Wenn `E7.7` = 1 ist, wird die Bitfolge (16 Bit) im Merkerwort `MW36` um die Anzahl an Bits nach rechts geschoben, die im Merkerwort `MW38` angegeben ist. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.10.6 32 Bit links schieben - SHL_DW

Mit der Operation "32 Bit links schieben" können Sie die Bits 0 - 31 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach links schieben. Die Stellen, die durch die Schiebeoperation frei werden, werden mit Nullen aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

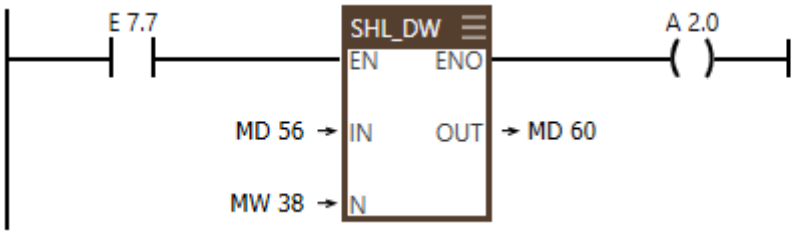
Wenn der Wert *N* größer als "32" ist, wird der Ausgang *OUT* auf den Wert "0" gesetzt. Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DWORD	E, A, M, D, L	Bitfolge (32 Bit), die geschoben werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die geschoben werden soll, maximal 32
OUT	DWORD	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SHL_DW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel

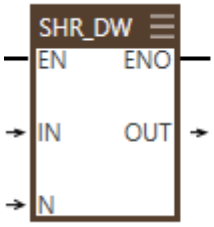


Wenn `E7.7` = 1 ist, wird die Bitfolge (32 Bit) im Merkerdoppelwort `MD56` um die Anzahl an Bits nach links geschoben, die im Merkerwort `MW38` angegeben ist . Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.10.7 32 Bit rechts schieben - SHR_DW

Mit der Operation "32 Bit rechts schieben" können Sie die Bits 0 - 31 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach rechts schieben. Die Stellen, die durch die Schiebeoperation frei werden, werden mit Nullen aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

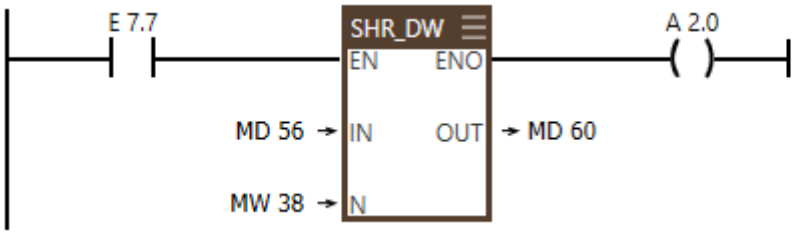
Wenn der Wert *N* größer als "32" ist, wird der Ausgang *OUT* auf den Wert "0" gesetzt. Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DWORD	E, A, M, D, L	Bitfolge (32 Bit), die geschoben werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die geschoben werden soll, maximal 32
OUT	DWORD	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: SHR_DW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel

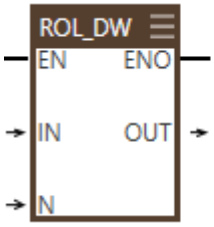


Wenn `E7.7` = 1 ist, wird die Bitfolge (32 Bit) im Merkerdoppelwort `MD56` um die Anzahl an Bits nach rechts geschoben, die im Merkerwort `MW38` angegeben ist. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.10.8 32 Bit links rotieren - ROL_DW

Mit der Operation "32 Bit links rotieren" können Sie die Bits 0 - 31 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach links rotieren. Die Stellen, die durch die Rotier-Operation frei werden, werden mit den Signalzuständen der hinausgeschobenen (rotierenden) Bits aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

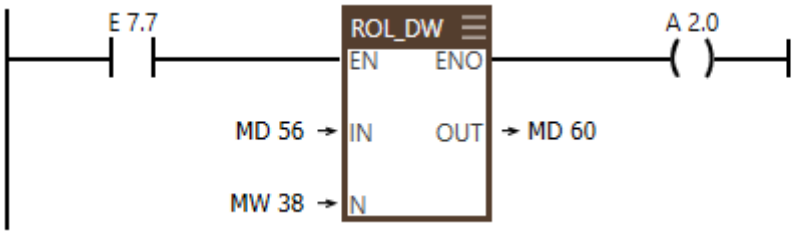
Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DWORD	E, A, M, D, L	Bitfolge (32 Bit), die rotiert werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die rotiert werden soll
OUT	DWORD	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ROL_DW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel

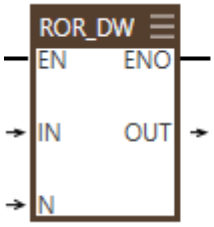


Wenn `E7.7` = 1 ist, wird die Bitfolge (32 Bit) im Merkerdoppelwort `MD56` um die Anzahl an Bits nach links rotiert, die im Merkerwort `MW38` angegeben ist. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.10.9 32 Bit rechts rotieren - ROR_DW

Mit der Operation "32 Bit rechts rotieren" können Sie die Bits 0 - 31 am Eingang *IN* um eine bestimmte Anzahl, die am Eingang *N* angegeben ist, nach rechts rotieren. Die Stellen, die durch die Rotier-Operation frei werden, werden mit den Signalzuständen der hinausgeschobenen (rotierenden) Bits aufgefüllt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert.

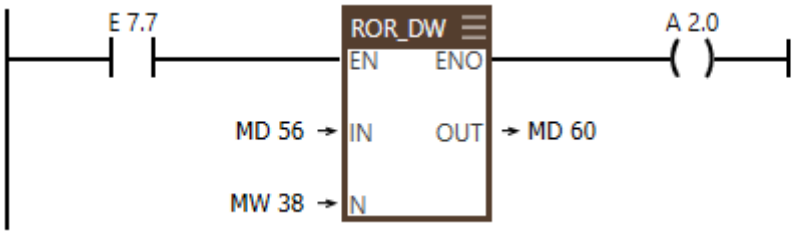
Wenn *N* ungleich "0" ist, werden die Bits "A0" und "OV" auf den Wert "0" gesetzt. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN	DWORD	E, A, M, D, L	Bitfolge (32 Bit), die rotiert werden soll
N	WORD	E, A, M, D, L	Anzahl der Bitpositionen, um die rotiert werden soll
OUT	DWORD	E, A, M, D, L	Ergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: ROR_DW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	✓	✓	✓	✓	-	✓	✓	✓	1

Beispiel



Wenn `E7.7` = 1 ist, wird die Bitfolge (32 Bit) im Merkerdoppelwort `MD56` um die Anzahl an Bits nach rechts rotiert, die im Merkerwort `MW38` angegeben ist. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

5.11 Zeiten

5.11.1 Überblick

In der CPU ist ein eigener Speicherbereich für Zeiten reserviert. Die Anzahl der Zeiten ist abhängig von der CPU.

Operation	Zeit
S_IMPULS	Zeit als Impuls parametrieren und starten
S_VIMP	Zeit als verlängerten Impuls parametrieren und starten
S_EVERZ	Zeit als Einschaltverzögerung parametrieren und starten
S_SEVERZ	Zeit als speichernde Einschaltverzögerung parametrieren und starten
S_AVERZ	Zeit als Ausschaltverzögerung parametrieren und starten
--(SI)	Zeit als Impuls starten
--(SV)	Zeit als verlängerten Impuls starten
--(SE)	Zeit als Einschaltverzögerung starten
--(SS)	Zeit als speichernde Einschaltverzögerung starten
--(SA)	Zeit als Ausschaltverzögerung starten

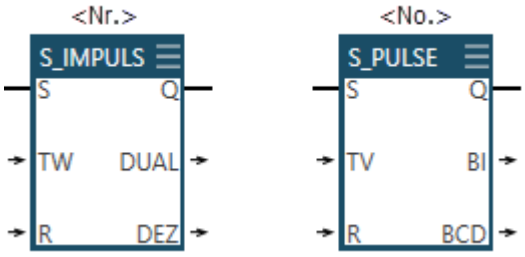
5.11.2 Zeit als Impuls parametrieren und starten - S_IMPULS

Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird die Operation "Zeit als Impuls parametrieren und starten" gestartet und der Ausgang *Q* wird auf den Signalzustand "1" gesetzt.

Der Ausgang *Q* wird erst dann wieder auf den Signalzustand "0" zurücksetzt, wenn mindestens einer der folgenden Zustände eintritt:

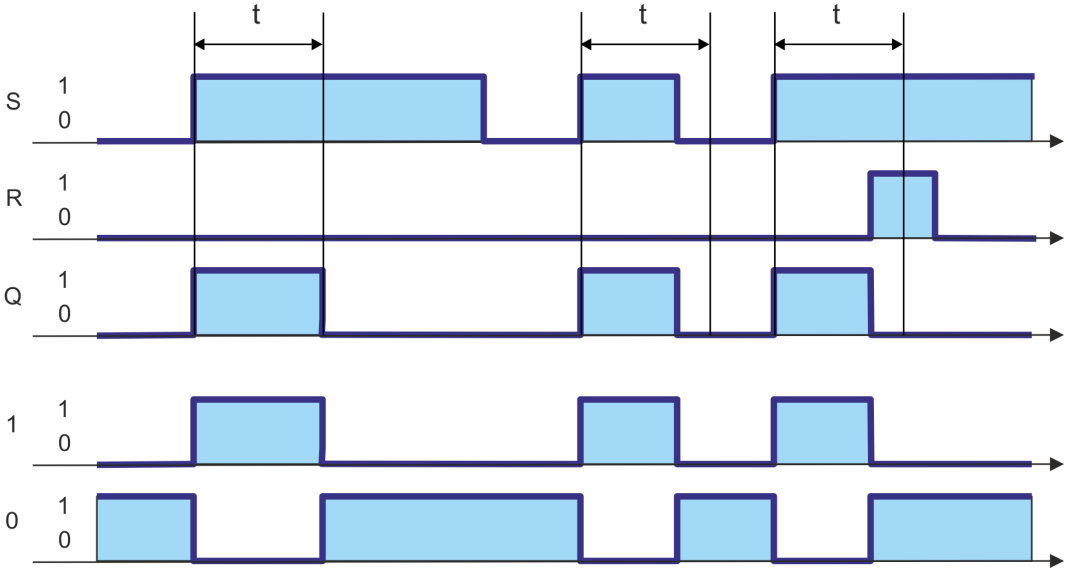
- Der Signalzustand am Eingang *S* wechselt von "1" auf "0".
- Die am Eingang *TW* voreingestellte Zeit ist abgelaufen.
- Der Signalzustand am Eingang *R* wechselt von "0" auf "1".

Deutsch - Englisch



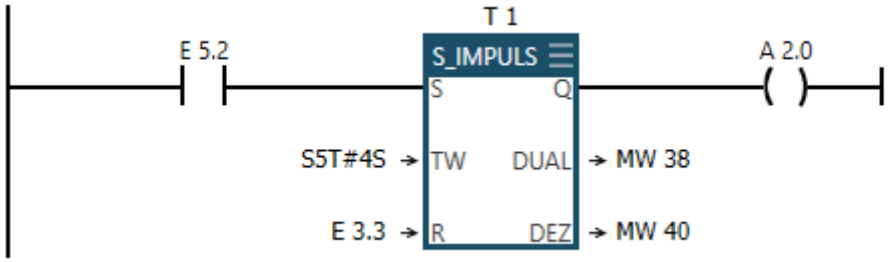
Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
S	S	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden
R	R	BOOL	E, A, M, D, L, T, Z	Rücksetzen
DUAL	BI	WORD	E, A, M, D, L	aktueller Zeitwert, Ganzzahlenformat
DEZ	BCD	WORD	E, A, M, D, L	aktueller Zeitwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status der Zeit

Statuswort für: S_IMPULS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1



t Programmierte Zeit
S VKE am Eingang *S*
R VKE am Eingang *R*
Q Signalzustand am Ausgang *Q*
1 Abfrage am Ausgang *Q* auf "1"
0 Abfrage am Ausgang *Q* auf "0"

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird für vier Sekunden auf den Wert "1" gesetzt.

Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden von "1" nach "0" wechselt, wird der Ausgang `A2.0` auf "0" zurückgesetzt.

Wenn der Signalzustand am Eingang `E3.3` vor Ablauf der vier Sekunden von "0" nach "1" wechselt, wird der Ausgang `A2.0` auf "0" zurückgesetzt.

Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zeitwert.

5.11.3 Zeit als verlängerten Impuls parametrieren und starten - S_VIMP

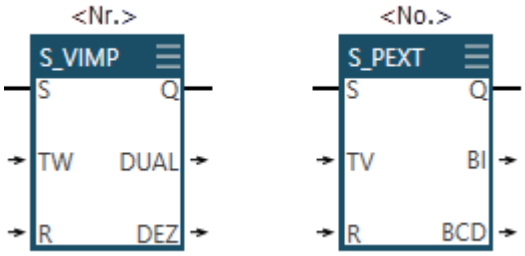
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird die Operation "Zeit als verlängerten Impuls parametrieren und starten" gestartet und der Ausgang *Q* wird auf den Signalzustand "1" gesetzt.

Der Ausgang *Q* wird erst dann wieder auf den Signalzustand "0" zurücksetzt, wenn mindestens einer der folgenden Zustände eintritt:

- Die am Eingang *TW* voreingestellte Zeit ist abgelaufen.
- Der Signalzustand am Eingang *R* wechselt von "0" auf "1".

Wenn der Signalzustand am Eingang *S* erneut von "0" auf "1" wechselt, während die gestartete Zeit läuft, wird die Zeit mit dem Zeitwert am Eingang *TW* nochmals gestartet. Der Impuls wird verlängert.

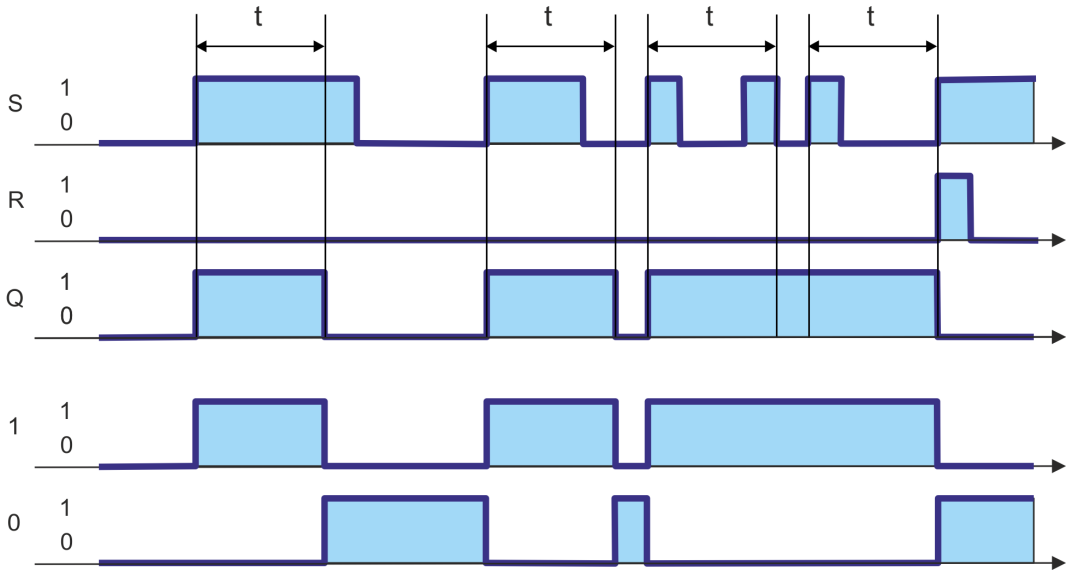
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
S	S	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden

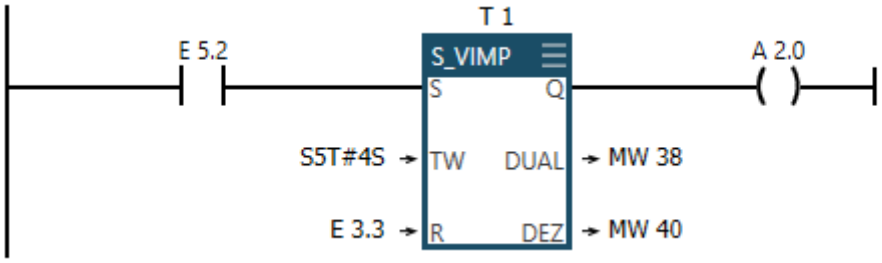
Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
R	R	BOOL	E, A, M, D, L, T, Z	Rücksetzen
DUAL	BI	WORD	E, A, M, D, L	aktueller Zeitwert, Ganzzahlenformat
DEZ	BCD	WORD	E, A, M, D, L	aktueller Zeitwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status der Zeit

Statuswort für: S_VIMP	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1



t Programmierte Zeit
S VKE am Eingang *S*
R VKE am Eingang *R*
Q Signalzustand am Ausgang *Q*
1 Abfrage am Ausgang *Q* auf "1"
0 Abfrage am Ausgang *Q* auf "0"

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird für vier Sekunden auf den Wert "1" gesetzt.

Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden erneut von "0" nach "1" wechselt, wird die Zeit um weitere vier Sekunden verlängert und der Ausgang `A2.0` erst nach Ablauf dieser Zeit auf "0" zurückgesetzt.

Wenn der Signalzustand am Eingang `E3.3` vor Ablauf der vier Sekunden von "0" nach "1" wechselt, wird der Ausgang `A2.0` auf "0" zurückgesetzt.

Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zeitwert.

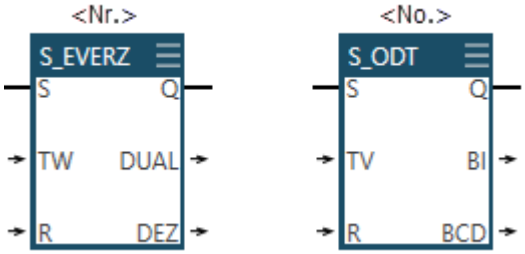
5.11.4 Zeit als Einschaltverzögerung parametrieren und starten - S_EVERZ

Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird die Operation "Zeit als Einschaltverzögerung parametrieren und starten" gestartet. Der Ausgang *Q* wird auf den Signalzustand "1" gesetzt, sobald die am Eingang *TW* voreingestellte Zeit abgelaufen ist und der Eingang *S* noch immer den Signalzustand "1" führt.

Der Ausgang *Q* wird erst dann wieder auf den Signalzustand "0" zurücksetzt, wenn mindestens einer der folgenden Zustände eintritt:

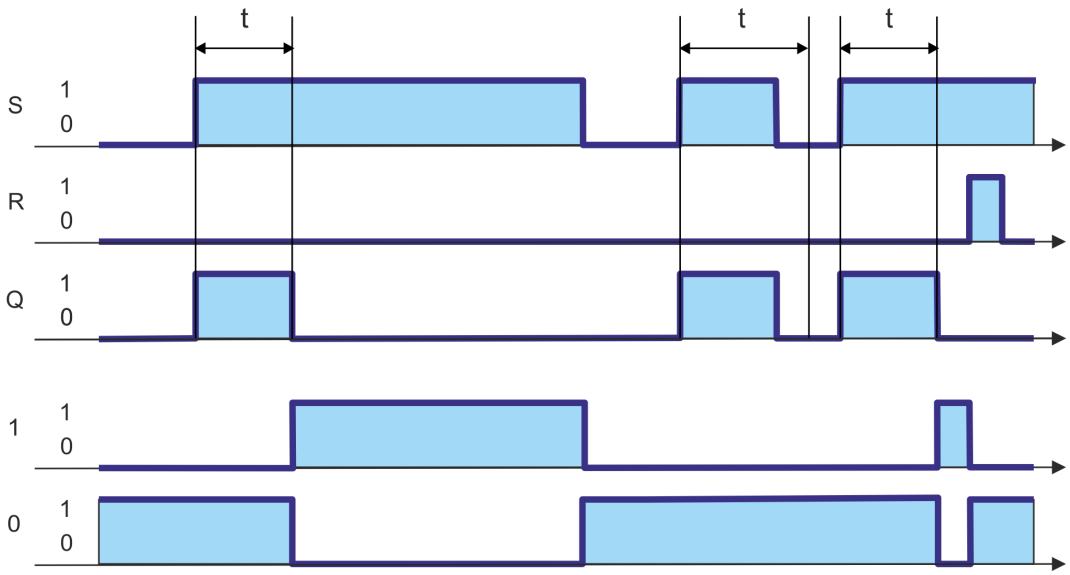
- Der Signalzustand am Eingang *S* wechselt von "1" auf "0".
- Der Signalzustand am Eingang *R* wechselt von "0" auf "1".

Deutsch - Englisch



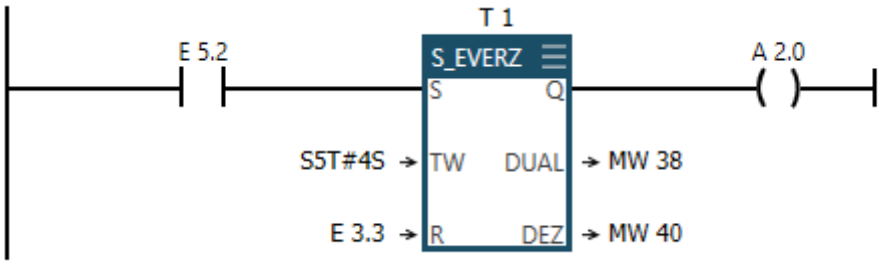
Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
S	S	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden
R	R	BOOL	E, A, M, D, L, T, Z	Rücksetzen
DUAL	BI	WORD	E, A, M, D, L	aktueller Zeitwert, Ganzzahlenformat
DEZ	BCD	WORD	E, A, M, D, L	aktueller Zeitwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status der Zeit

Statuswort für: S_EVERZ	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1



t Programmierte Zeit
S VKE am Eingang *S*
R VKE am Eingang *R*
Q Signalzustand am Ausgang *Q*
1 Abfrage am Ausgang *Q* auf "1"
0 Abfrage am Ausgang *Q* auf "0"

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Wenn der Eingang `E5.2` nach vier Sekunden noch immer den Signalzustand "1" führt , wird der Ausgang `A2.0` auf den Wert "1" gesetzt. Wenn der Eingang `E5.2` jedoch vor Ablauf der vier Sekunden den Signalzustand "0" führt, bleibt der Ausgang `A2.0` auf dem Wert "0".

Wenn der Signalzustand am Eingang `E3.3` von "0" nach "1" wechselt, wird der Ausgang `A2.0` auf "0" zurückgesetzt.

Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zeitwert.

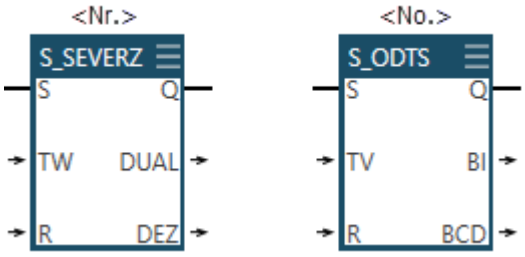
5.11.5 Zeit als speichernde Einschaltverzögerung parametrieren und starten - S_SEVERZ

Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird die Operation "Zeit als speichernde Einschaltverzögerung parametrieren und starten" gestartet. Der Ausgang *Q* wird auf den Signalzustand "1" gesetzt, sobald die am Eingang *TW* voreingestellte Zeit abgelaufen ist.

Der Ausgang *Q* wird erst dann wieder auf den Signalzustand "0" zurücksetzt, wenn der Signalzustand am Eingang *R* von "0" auf "1" wechselt.

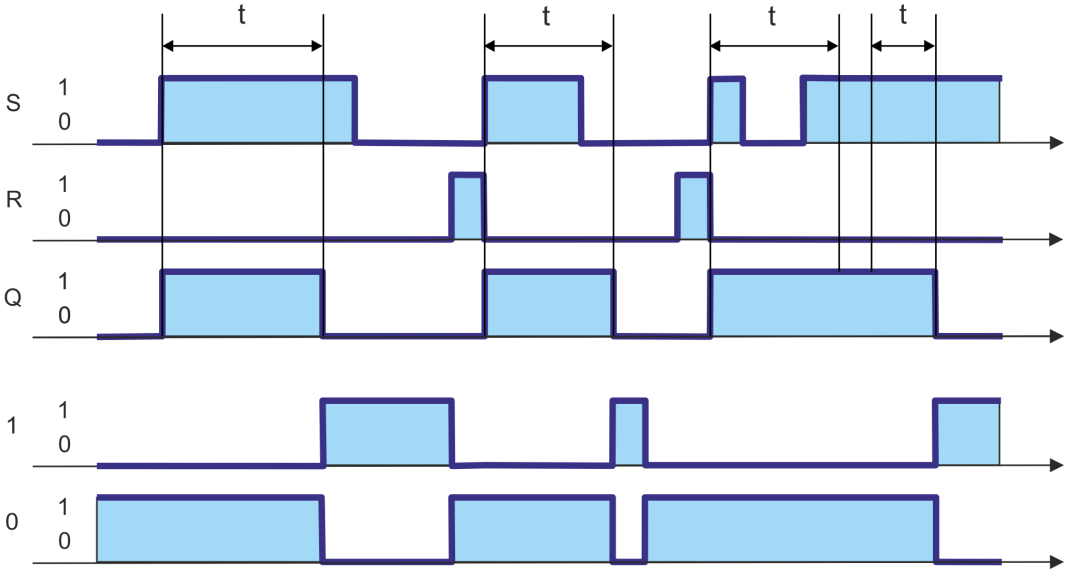
Wenn der Signalzustand am Eingang *S* erneut von "0" auf "1" wechselt, während die gestartete Zeit läuft, wird die Zeit mit dem Zeitwert am Eingang *TW* nochmals gestartet. Der Ausgang *Q* wird erst nach dieser weiteren Zeit auf den Signalzustand "1" gesetzt.

Deutsch - Englisch



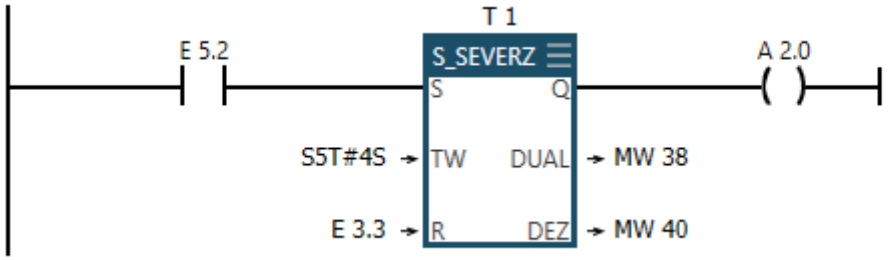
Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
S	S	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden
R	R	BOOL	E, A, M, D, L, T, Z	Rücksetzen
DUAL	BI	WORD	E, A, M, D, L	aktueller Zeitwert, Ganzzahlenformat
DEZ	BCD	WORD	E, A, M, D, L	aktueller Zeitwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status der Zeit

Statuswort für: S_SEVERZ	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1



t Programmierte Zeit
S VKE am Eingang *S*
R VKE am Eingang *R*
Q Signalzustand am Ausgang *Q*
1 Abfrage am Ausgang *Q* auf "1"
0 Abfrage am Ausgang *Q* auf "0"

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird nach vier Sekunden auf den Wert "1" gesetzt, auch wenn der Eingang `E5.2` nicht mehr den Signalzustand "1" führt.

Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden erneut von "0" nach "1" wechselt, wird die Zeit um weitere vier Sekunden verlängert und der Ausgang `A2.0` erst nach Ablauf dieser Zeit auf "1" gesetzt.

Wenn der Signalzustand am Eingang `E3.3` von "0" nach "1" wechselt, wird der Ausgang `A2.0` auf "0" zurückgesetzt.

Die Merkerworte `MW38` und `MW40` enthalten den aktuellen Zeitwert.

5.11.6 Zeit als Ausschaltverzögerung parametrieren und starten - S_AVERZ

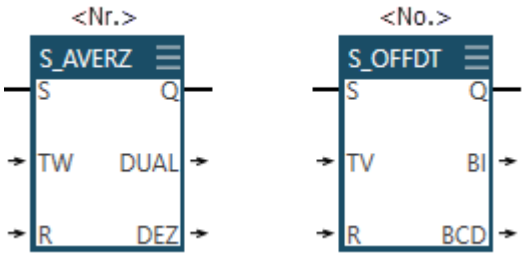
Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang *S* wird die Operation "Zeit als Ausschaltverzögerung parametrieren und starten" gestartet. Der Ausgang *Q* wird auf den Signalzustand "1" gesetzt. Erst bei einem Flankenwechsel von "1" auf "0" (fallende Flanke) am Eingang *S* wird die am Eingang *TW* voreingestellte Zeit gestartet.

Der Ausgang *Q* wird erst dann wieder auf den Signalzustand "0" zurücksetzt, wenn mindestens einer der folgenden Zustände eintritt:

- Die am Eingang *TW* voreingestellte Zeit, die durch die fallende Flanke am Eingang *S* gestartet wurde, ist abgelaufen.
- Der Signalzustand am Eingang *R* wechselt von "0" auf "1".

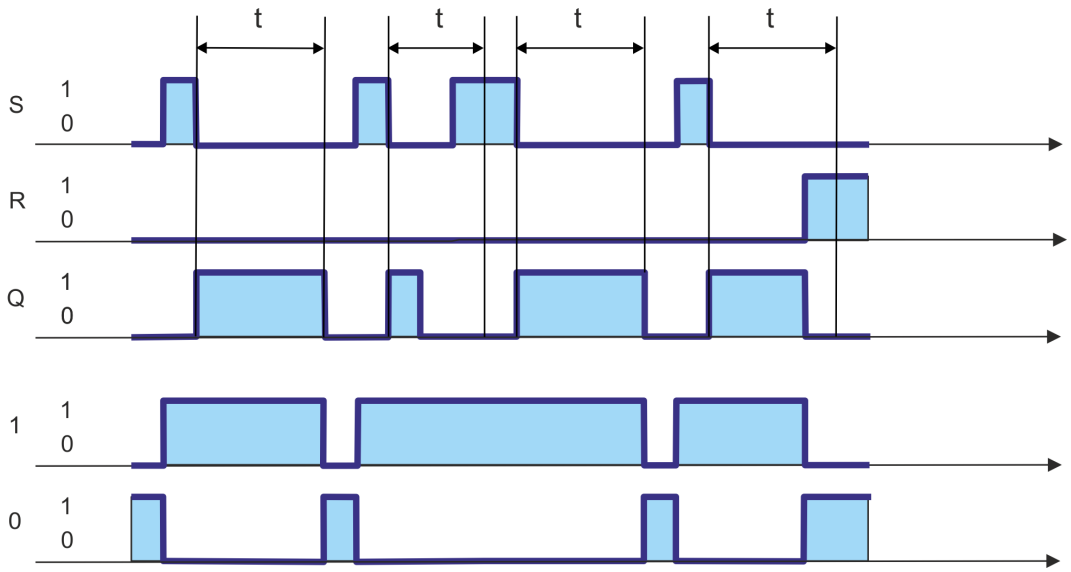
Wenn der Signalzustand am Eingang *S* erneut von "1" auf "0" wechselt, während die gestartete Zeit läuft, wird die Zeit mit dem Zeitwert am Eingang *TW* neu gestartet. Der Ausgang *Q* wird erst nach dieser weiteren Zeit auf den Signalzustand "0" gesetzt.

Deutsch - Englisch



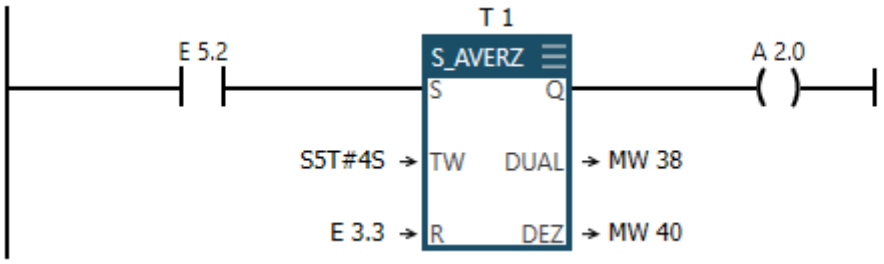
Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
S	S	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden
R	R	BOOL	E, A, M, D, L, T, Z	Rücksetzen
DUAL	BI	WORD	E, A, M, D, L	aktueller Zeitwert, Ganzzahlenformat
DEZ	BCD	WORD	E, A, M, D, L	aktueller Zeitwert, BCD-Format
Q	Q	BOOL	E, A, M, D, L	Status der Zeit

Statuswort für: S_AVERZ	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1



t Programmierte Zeit
S VKE am Eingang *S*
R VKE am Eingang *R*
Q Signalzustand am Ausgang *Q*
1 Abfrage am Ausgang *Q* auf "1"
0 Abfrage am Ausgang *Q* auf "0"

Beispiel



Wenn der Signalzustand am Eingang E5.2 von "0" nach "1" wechselt, wird die Zeit T1 gestartet. Der Ausgang A2.0 wird auf den Wert "1" gesetzt. Wenn der Signalzustand am Eingang E5.2 von "1" zurück nach "0" wechselt ", wird die voreingestellte Zeit von vier Sekunden gestartet. Nach Ablauf dieser Zeit wird der Ausgang A2.0 auf "0" zurückgesetzt.

Wenn der Signalzustand am Eingang E5.2 vor Ablauf der vier Sekunden erneut von "1" nach "0" wechselt, wird die Zeit um weitere vier Sekunden verlängert und der Ausgang A2.0 erst nach Ablauf dieser Zeit auf "0" zurückgesetzt.

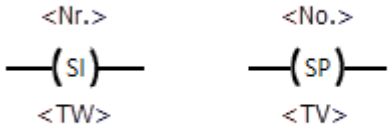
Wenn der Signalzustand am Eingang E3.3 von "0" nach "1" wechselt, wird der Ausgang A2.0 auf "0" zurückgesetzt.

Die Merkerworte MW38 und MW40 enthalten den aktuellen Zeitwert.

5.11.7 Zeit als Impuls starten - --(SI)

Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird die Operation "Zeit als Impuls starten" mit der am Eingang *TW* voreingestellten Zeit gestartet.
↳ *Kap. 5.11.2 "Zeit als Impuls parametrieren und starten - S_IMPULS" Seite 244*.

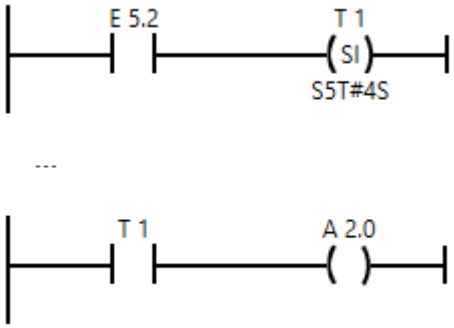
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden

Statuswort für: SI	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	-	-	0

Beispiel

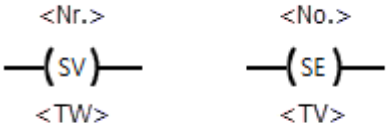


Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird für vier Sekunden auf den Wert "1" gesetzt. Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden von "1" nach "0" wechselt, wird der Ausgang `A2.0` auf "0" zurückgesetzt.

5.11.8 Zeit als verlängerten Impuls starten - --(SV)

Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird die Operation "Zeit als verlängerten Impuls starten" mit der am Eingang *TW* voreingestellten Zeit gestartet. *Kap. 5.11.3 "Zeit als verlängerten Impuls parametrieren und starten - S_VIMP" Seite 246*.

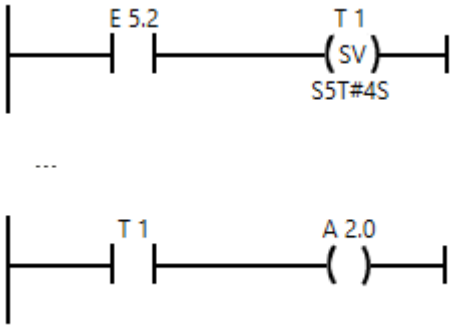
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden

Statuswort für: SV	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	-	-	0

Beispiel



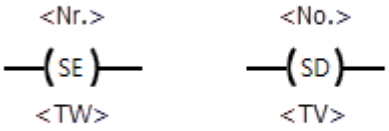
Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird für vier Sekunden auf den Wert "1" gesetzt.

Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden erneut von "0" nach "1" wechselt, wird die Zeit um weitere vier Sekunden verlängert und der Ausgang `A2.0` erst nach Ablauf dieser Zeit auf "0" zurückgesetzt.

5.11.9 Zeit als Einschaltverzögerung starten - --(SE)

Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird die Operation "Zeit als Einschaltverzögerung starten" mit der am Eingang *TW* voreingestellten Zeit gestartet. *Kap. 5.11.4 "Zeit als Einschaltverzögerung parametrieren und starten - S_EVERZ" Seite 248*.

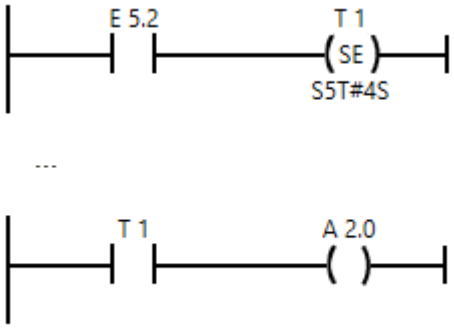
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden

Statuswort für: SE	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	-	-	0

Beispiel

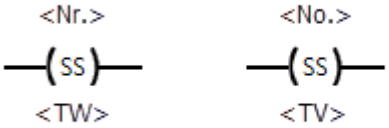


Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Wenn der Eingang `E5.2` nach vier Sekunden noch immer den Signalzustand "1" führt , wird der Ausgang `A2.0` auf den Wert "1" gesetzt. Wenn der Eingang `E5.2` jedoch vor Ablauf der vier Sekunden den Signalzustand "0" führt, bleibt der Ausgang `A2.0` auf dem Wert "0".

5.11.10 Zeit als speichernde Einschaltverzögerung starten - --(SS)

Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird die Operation "Zeit als speichernde Einschaltverzögerung starten" mit der am Eingang *TW* voreingestellten Zeit gestartet. *Kap. 5.11.5 "Zeit als speichernde Einschaltverzögerung parametrieren und starten - S_SEVERZ" Seite 249*.

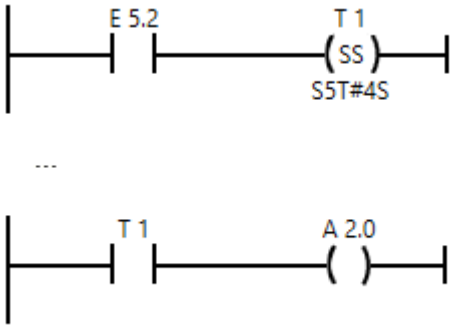
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden

Statuswort für: SS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	-	-	0

Beispiel



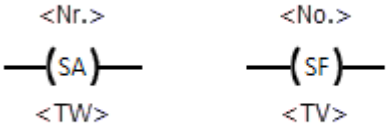
Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird nach vier Sekunden auf den Wert "1" gesetzt, auch wenn der Eingang `E5.2` nicht mehr den Signalzustand "1" führt.

Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden erneut von "0" nach "1" wechselt, wird die Zeit um weitere vier Sekunden verlängert und der Ausgang `A2.0` erst nach Ablauf dieser Zeit auf "1" gesetzt.

5.11.11 Zeit als Ausschaltverzögerung starten - --(SA)

Bei einem Flankenwechsel von "0" auf "1" (steigende Flanke) am Eingang wird die Operation "Zeit als Ausschaltverzögerung starten" mit der am Eingang *TW* voreingestellten Zeit gestartet. ↬ *Kap. 5.11.6 "Zeit als Ausschaltverzögerung parametrieren und starten - S_AVERZ" Seite 251*.

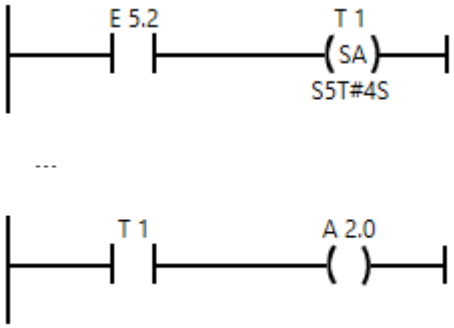
Deutsch - Englisch



Parameter Deutsch	Parameter Englisch	Datentyp	Speicherbereich	Beschreibung
Nr.	No.	TIMER	T	Nummer der Zeit, Bereich ist abhängig von der CPU
-	-	BOOL	E, A, M, D, L, T, Z	Zeit starten
TW	TV	S5TIME	E, A, M, D, L oder Konstante	Zeitwert, maximal 9.990 Sekunden

Statuswort für: SA	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	0	-	-	0

Beispiel



Wenn der Signalzustand am Eingang `E5.2` von "0" nach "1" wechselt, wird die Zeit `T1` gestartet. Der Ausgang `A2.0` wird auf den Wert "1" gesetzt. Wenn der Signalzustand am Eingang `E5.2` von "1" zurück nach "0" wechselt ", wird die voreingestellte Zeit von vier Sekunden gestartet. Nach Ablauf dieser Zeit wird der Ausgang `A2.0` auf "0" zurückgesetzt.

Wenn der Signalzustand am Eingang `E5.2` vor Ablauf der vier Sekunden erneut von "1" nach "0" wechselt, wird die Zeit um weitere vier Sekunden verlängert und der Ausgang `A2.0` erst nach Ablauf dieser Zeit auf "0" zurückgesetzt.

5.12 Wortverknüpfungen

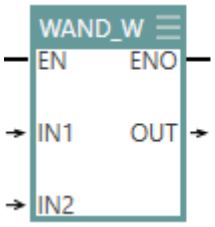
5.12.1 Übersicht

Mit den Wortverknüpfungen können Sie binäre (boolesche) Operationen mit zwei Wort- bzw. Doppelwortoperanden durchführen.

Operation	Wortverknüpfung
WAND_W	16 Bit UND-Verknüpfung
WOR_W	16 Bit ODER-Verknüpfung
WXOR_W	16 Bit EXKLUSIV-ODER-Verknüpfung
WAND_DW	32 Bit UND-Verknüpfung
WOR_DW	32 Bit ODER-Verknüpfung
WXOR_DW	32 Bit EXKLUSIV-ODER-Verknüpfung

5.12.2 16 Bit UND-Verknüpfung - WAND_W

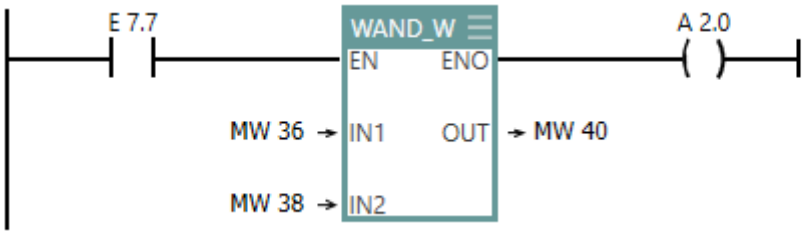
Mit der Operation "16 Bit UND-Verknüpfung" können Sie die einzelnen Bits an den Eingängen *IN1* und *IN2* logisch miteinander verknüpfen, wie in der UND-Wahrheitstabelle dargestellt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	WORD	E, A, M, D, L oder Konstante	erstes Bitfeld
IN2	WORD	E, A, M, D, L oder Konstante	zweites Bitfeld
OUT	WORD	E, A, M, D, L	Verknüpfungsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: WAND_W	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	✓	0	0	-	✓	1	1	1

Beispiel



Wenn `E7.7` = 1 ist, werden die Bitfelder in den Merkerwörtern `MW36` und `MW38` miteinander verknüpft. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel:

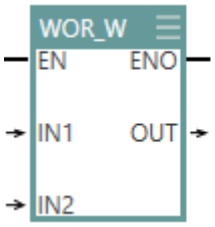
`MW36` = 10011100 01101010

`MW38` = 11110000 11110000

`MW40` = 10010000 01100000

5.12.3 16 Bit ODER-Verknüpfung - WOR_W

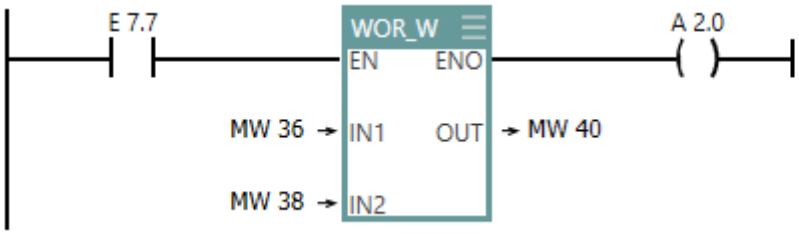
Mit der Operation "16 Bit ODER-Verknüpfung" können Sie die einzelnen Bits an den Eingängen *IN1* und *IN2* logisch miteinander verknüpfen, wie in der ODER-Wahrheitstabelle dargestellt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	WORD	E, A, M, D, L oder Konstante	erstes Bitfeld
IN2	WORD	E, A, M, D, L oder Konstante	zweites Bitfeld
OUT	WORD	E, A, M, D, L	Verknüpfungsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: WOR_W	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	✓	0	0	-	✓	1	1	1

Beispiel



Wenn `E7.7` = 1 ist, werden die Bitfelder in den Merkerwörtern `MW36` und `MW38` miteinander verknüpft. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel:

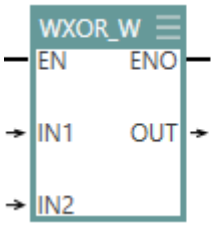
`MW36` = 10011100 01101010

`MW38` = 11110000 11110000

`MW40` = 11111100 11111010

5.12.4 16 Bit EXKLUSIV-ODER-Verknüpfung - WXOR_W

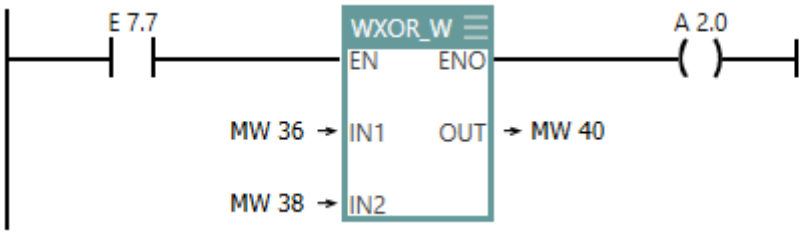
Mit der Operation "16 Bit EXKLUSIV-ODER-Verknüpfung" können Sie die einzelnen Bits an den Eingängen *IN1* und *IN2* logisch miteinander verknüpfen, wie in der EXKLUSIV-ODER-Wahrheitstabelle dargestellt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	WORD	E, A, M, D, L oder Konstante	erstes Bitfeld
IN2	WORD	E, A, M, D, L oder Konstante	zweites Bitfeld
OUT	WORD	E, A, M, D, L	Verknüpfungsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: WXOR_W	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	✓	0	0	-	✓	1	1	1

Beispiel



Wenn `E7.7` = 1 ist, werden die Bitfelder in den Merkerwörtern `MW36` und `MW38` miteinander verknüpft. Das Ergebnis wird im Merkerwort `MW40` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel:

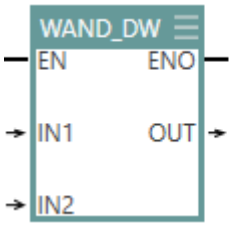
`MW36` = 10011100 01101010

`MW38` = 11110000 11110000

`MW40` = 01101100 10011010

5.12.5 32 Bit UND-Verknüpfung - WAND_DW

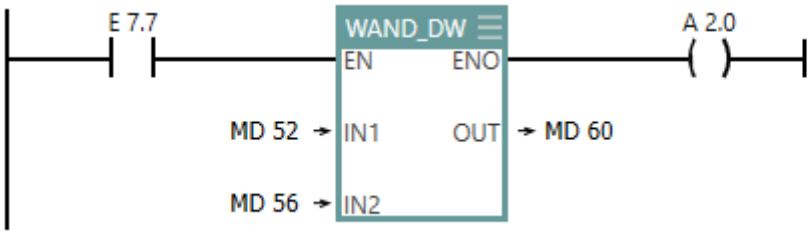
Mit der Operation "32 Bit UND-Verknüpfung" können Sie die einzelnen Bits an den Eingängen *IN1* und *IN2* logisch miteinander verknüpfen, wie in der UND-Wahrheitstabelle dargestellt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	DWORD	E, A, M, D, L oder Konstante	erstes Bitfeld
IN2	DWORD	E, A, M, D, L oder Konstante	zweites Bitfeld
OUT	DWORD	E, A, M, D, L	Verknüpfungsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: WAND_DW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	✓	0	0	-	✓	1	1	1

Beispiel



Wenn `E7.7` = 1 ist, werden die Bitfelder in den Merkerdoppelwörtern `MD52` und `MD56` miteinander verknüpft. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel:

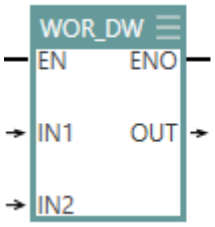
`MD52` = 10011100 01101010 10011100 01101010

`MD56` = 11110000 11110000 00000000 11111111

`MD60` = 10010000 01100000 00000000 01101010

5.12.6 32 Bit ODER-Verknüpfung - WOR_DW

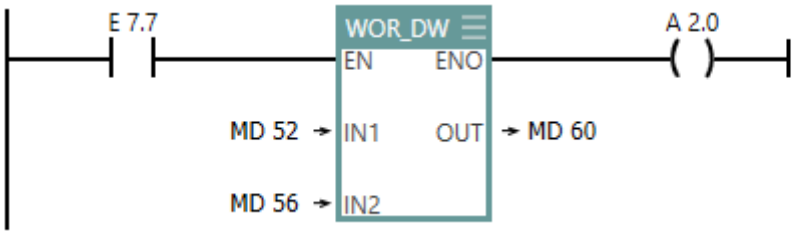
Mit der Operation "32 Bit ODER-Verknüpfung" können Sie die einzelnen Bits an den Eingängen *IN1* und *IN2* logisch miteinander verknüpfen, wie in der ODER-Wahrheitstabelle dargestellt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	DWORD	E, A, M, D, L oder Konstante	erstes Bitfeld
IN2	DWORD	E, A, M, D, L oder Konstante	zweites Bitfeld
OUT	DWORD	E, A, M, D, L	Verknüpfungsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: WOR_DW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	✓	0	0	-	✓	1	1	1

Beispiel



Wenn `E7.7` = 1 ist, werden die Bitfelder in den Merkerdoppelwörtern `MD52` und `MD56` miteinander verknüpft. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel:

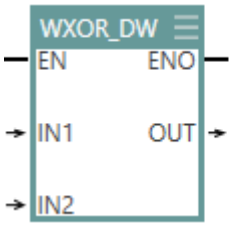
`MD52` = 10011100 01101010 10011100 01101010

`MD56` = 11110000 11110000 00000000 11111111

`MD60` = 11111100 11111010 10011100 11111111

5.12.7 32 Bit EXKLUSIV-ODER-Verknüpfung - WXOR_DW

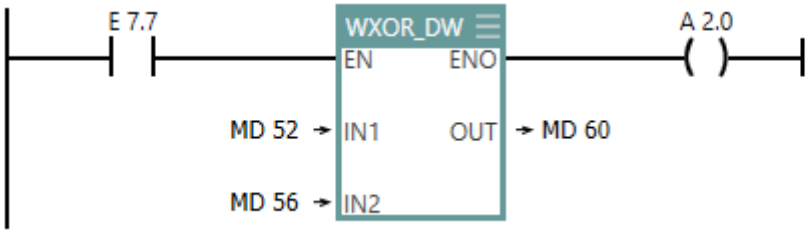
Mit der Operation "32 Bit EXKLUSIV-ODER-Verknüpfung" können Sie die einzelnen Bits an den Eingängen *IN1* und *IN2* logisch miteinander verknüpfen, wie in der EXKLUSIV-ODER-Wahrheitstabelle dargestellt. Die Operation wird nur dann ausgeführt, wenn der Freigabeeingang *EN* den Signalzustand "1" führt. Das Ergebnis wird im Ausgang *OUT* gespeichert. Die Parameter *ENO* und *EN* haben immer den gleichen Signalzustand.



Parameter	Datentyp	Speicherbereich	Beschreibung
EN	BOOL	E, A, M, D, L, T, Z	Freigabeeingang
IN1	WORD	E, A, M, D, L oder Konstante	erstes Bitfeld
IN2	WORD	E, A, M, D, L oder Konstante	zweites Bitfeld
OUT	WORD	E, A, M, D, L	Verknüpfungsergebnis
ENO	BOOL	E, A, M, D, L	Freigabeausgang

Statuswort für: WXOR_DW	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	1	✓	0	0	-	✓	1	1	1

Beispiel



Wenn `E7.7` = 1 ist, werden die Bitfelder in den Merkerdoppelwörtern `MD52` und `MD56` miteinander verknüpft. Das Ergebnis wird im Merkerdoppelwort `MD60` gespeichert. Wenn die Operation ausgeführt worden ist, ist `A2.0` = 1 (ENO = EN).

Beispiel:

`MD52` = 10011100 01101010 10011100 01101010

`MD56` = 11110000 11110000 00000000 11111111

`MD60` = 01101100 10011100 10011100 10010101

5.13 Statusbits

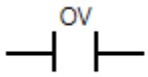
5.13.1 Überblick

Mit den Statusbit-Operationen können Sie binäre Verknüpfungen programmieren, die auf den Bits des Statusworts im Speicher der CPU basieren.

Operation	Statusbit
OV --||--	Störungsbit "Überlauf": In der letzten arithmetischen Operation ist ein Überlauf aufgetreten
OS --||--	Störungsbit "Überlauf gespeichert": In einer arithmetischen Operation ist ein Überlauf aufgetreten
UO --||--	Störungsbit "Ungültige Operation": Das Ergebnis einer arithmetischen Operation ist ungültig
BIE --||--	Störungsbit BIE-Register abfragen
==0 --||-- <>0 --||-- >0 --||-- <0 --||-- >=0 --||-- <=0 --||--	Ergebnisbit: Ergebnis einer arithmetischen Operation mit dem Wert Null vergleichen

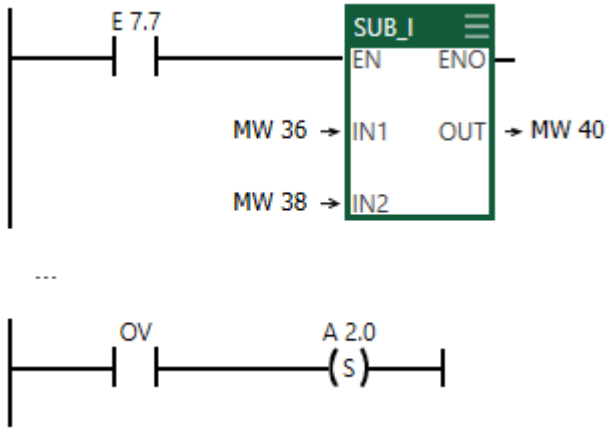
5.13.2 Störungsbit Überlauf - OV --||--

Mit der Operation "Störungsbit Überlauf" können Sie den Signalzustand des Störungsbits "Überlauf (OV)" abfragen. Wenn das Ergebnis der letzten arithmetischen Operation außerhalb des zulässigen negativen oder positiven Bereichs liegt, wird das Bit "OV" gesetzt.



Statuswort für: OV	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

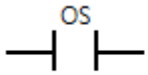
Beispiel



Im ersten Netzwerk wird die Ganzzahl (16 Bit) im Merkerwort MW38 von der Ganzzahl (16 Bit) im Merkerwort MW36 subtrahiert. Wenn das Ergebnis im Merkerwort MW40 außerhalb des zulässigen Bereichs für eine Ganzzahl (16 Bit) liegt, wird das Bit "OV" auf den Wert "1" gesetzt. Der Ausgang A2.0 wird gesetzt. Das Bit "OV" wird zurückgesetzt, wenn weitere arithmetische Operationen zwischen den beiden Netzwerken liegen, die keinen Überlauf verursachen.

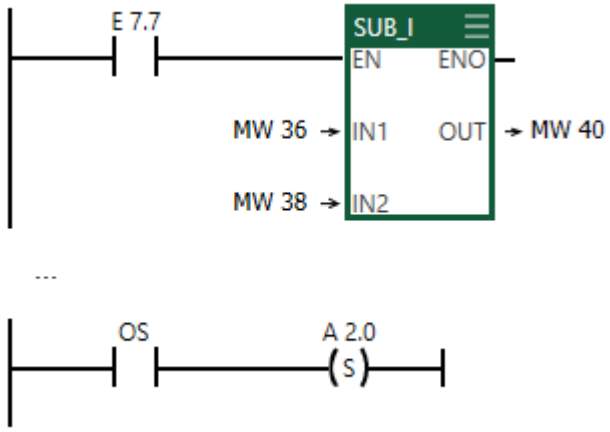
5.13.3 Störungsbit Überlauf gespeichert - OS --||--

Mit der Operation "Störungsbit Überlauf gespeichert" können Sie den Signalzustand des Störungsbits "Überlauf gespeichert (OS)" abfragen. Wenn das Ergebnis einer arithmetischen Operation außerhalb des zulässigen negativen oder positiven Bereichs liegt, wird das Bit "OV" gesetzt.



Statuswort für: OS	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

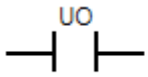
Beispiel



Im ersten Netzwerk wird die Ganzzahl (16 Bit) im Merkerwort MW38 von der Ganzzahl (16 Bit) im Merkerwort MW36 subtrahiert. Wenn das Ergebnis im Merkerwort MW40 außerhalb des zulässigen Bereichs für eine Ganzzahl (16 Bit) liegt, wird das Bit "OS" auf den Wert "1" gesetzt. Der Ausgang A2.0 wird gesetzt. Das Bit "OS" bleibt auch dann gesetzt, wenn weitere arithmetische Operationen zwischen den beiden Netzwerken liegen, die keinen Überlauf verursachen.

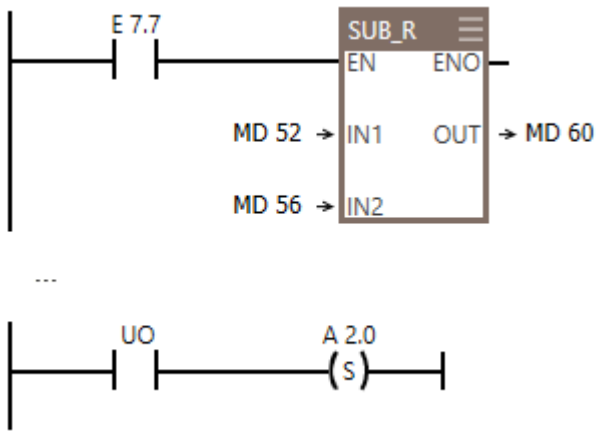
5.13.4 Störungsbit Ungültige Operation - UO --||--

Mit der Operation "Störungsbit Ungültige Operation" können Sie den Signalzustand des Störungsbits "Ungültige Operation (UO)" abfragen. Wenn bei arithmetischen Operationen mit Gleitpunktzahlen ein Eingang keine gültige Gleitpunktzahl enthält, wird das Bit "UO" gesetzt.



Statuswort für: UO	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

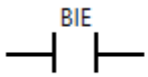
Beispiel



Im ersten Netzwerk wird die Gleitpunktzahl im Merkerdoppelwort `MD56` von der Gleitpunktzahl im Merkerdoppelwort `MD52` subtrahiert. Wenn mindestens einer der beiden Eingänge keine gültige Gleitpunktzahl enthält, wird das Bit "UO" auf den Wert "1" gesetzt. Der Ausgang `A2.0` wird gesetzt.

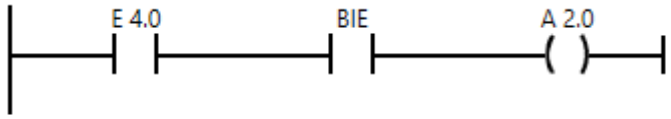
5.13.5 Störungsbit BIE-Register - BIE --||--

Mit der Operation "Störungsbit BIE-Register" können Sie den Signalzustand des "Binärergebnis-Bits (BIE)" abfragen.



Statuswort für: BIE	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



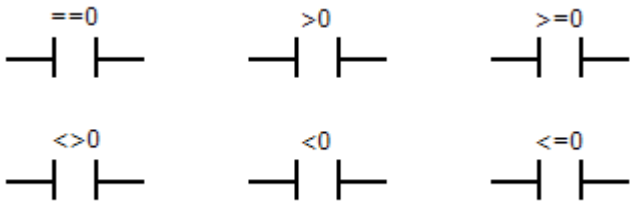
Wenn der Eingang `E4.0` und das Binärergebnis-Bits "BIE" den Signalzustand "1" führen, liegt am Ausgang `A2.0` der Wert "1" an.

5.13.6 Ergebnisbit - x0 --||--

Mit der Operation "Ergebnisbit" können Sie das Ergebnis einer arithmetischen Operation mit dem Wert Null vergleichen. Wenn das Ergebnis der letzten arithmetischen Operation der Vergleichsbedingung entspricht, wird das Ergebnisbit gesetzt.

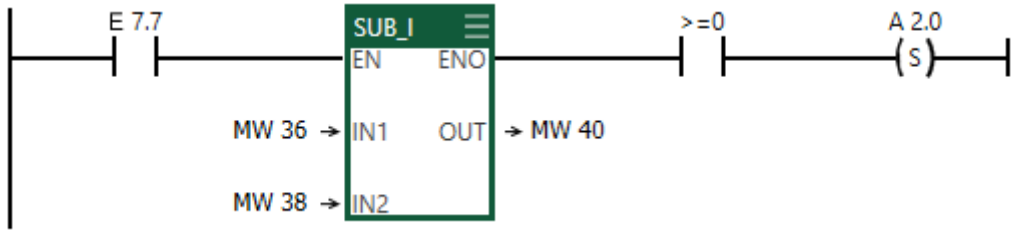
Sie können aus folgenden Vergleichsbedingungen wählen:

Operation	Beschreibung
== 0	Setzt das Ergebnisbit, wenn das Ergebnis der Operation gleich 0 ist
<> 0	Setzt das Ergebnisbit, wenn das Ergebnis der Operation ungleich 0 ist
> 0	Setzt das Ergebnisbit, wenn das Ergebnis der Operation größer als 0 ist
< 0	Setzt das Ergebnisbit, wenn das Ergebnis der Operation kleiner als 0 ist
>= 0	Setzt das Ergebnisbit, wenn das Ergebnis der Operation größer als oder gleich 0 ist
<= 0	Setzt das Ergebnisbit, wenn das Ergebnis der Operation kleiner als oder gleich 0 ist



Statuswort für: <> 0	BIE	A1	A0	OV	OS	OR	STA	VKE	/ER
Operation schreibt	-	-	-	-	-	✓	✓	✓	1

Beispiel



Wenn `E7.7` = 1 ist, wird der Wert im Merkerwort `MW38` von `MW36` subtrahiert. Wenn das Ergebnis im Merkerwort `MW40` größer als oder gleich Null ist, wird der Ausgang `A2.0` gesetzt. Wenn das Ergebnis kleiner als Null ist, wird `A2.0` nicht gesetzt.

6 Bausteinparameter

6.1 Allgemeine und spezifische Fehlercodes RET_VAL

Übersicht

Der Rückgabewert *RET_VAL* einer Systemfunktion stellt einen der beiden folgenden Fehlercodes zur Verfügung:

- *Allgemeiner Fehlercode*, der sich auf jeden beliebigen SFC beziehen kann.
- *Spezifischer Fehlercode*, der sich auf den jeweiligen SFC bezieht.

Es handelt sich beim Datentyp des Ausgangsparameters *RET_VAL* zwar um eine Ganzzahl (INT), doch die Fehlercodes der Systemfunktionen werden nach hexadezimalen Werten gegliedert.

Wenn Sie einen Rückgabewert auswerten und den Wert mit den Fehlercodes vergleichen, so lassen Sie sich den Fehlercode im Hexadezimalformat ausgeben.

RET_VAL (Rückgabewert)

Die folgende Tabelle zeigt den Aufbau eines Fehlercodes:

Bit	Bedeutung
7 ... 0	Ereignisnummer bzw. Fehlerklasse und Einzelfehler
14 ... 8	Bit 14 ... 8 = "0": **Spezifischer Fehlercode** Den spezifischen Fehlercode finden Sie in der Beschreibung der einzelnen SFCs. Bit 14 ... 8 > "0": **Allgemeiner Fehlercode** Die möglichen allgemeinen Fehlercodesfinden Sie auf der folgenden Seite.
15	Bit 15 = "1": zeigt an, dass ein Fehler aufgetreten ist.

Spezifischer Fehlercode

Dieser Fehlercode zeigt an, dass ein Fehler, der zu einer bestimmten Systemfunktion gehört, während der Bearbeitung aufgetreten ist.

Ein spezifischer Fehlercode besteht aus:

- Fehlerklasse zwischen 0 und 7
- Einzelfehler zwischen 0 und 15

Bit	Bedeutung
3 ... 0	Einzelfehler
6 ... 4	Fehlerklasse
7	Bit 7 = "1"
14 ... 8	Bit 14 ... 8 = "0"
15	Bit 15 = "1": zeigt an, dass ein Fehler aufgetreten ist.

Allgemeine Fehlercodes RET_VAL

Der Parameter *RET_VAL* verschiedener SFCs liefert keine spezifischen, sondern nur allgemeine Fehlerinformationen zurück.

Der allgemeine Fehlercode enthält Fehlerinformationen, die bei allen Systemfunktionen auftreten können. Ein allgemeiner Fehlercode besteht aus den beiden folgenden Nummern:

- Eine Parameternummer zwischen 1 und 111, wobei 1 den ersten Parameter, 2 den zweiten Parameter usw. des aufgerufenen SFC anzeigt.
- Eine Ereignisnummer zwischen 0 und 127. Die Ereignisnummer zeigt einen synchronen Fehler an.

Bit	Bedeutung
7 ... 0	Ereignisnummer
14 ... 8	Parameternummer
15	Bit 15 = "1": zeigt an, dass ein Fehler aufgetreten ist.

Allgemeine Fehlercodes

In der folgenden Tabelle werden die allgemeinen Fehlercodes eines Rückgabewerts erläutert. Die Darstellung erfolgt im Hexadezimalformat, wobei der Buchstabe x in jeder Codenummer nur als Platzhalter dient und die Nummer des Parameters der Systemfunktion darstellt, die den Fehler verursacht hat.

Fehlercode	Beschreibung
8x7Fh	Interner Fehler. Dieser Fehlercode zeigt einen internen Fehler am Parameter x an. Dieser Fehler wurde nicht vom Anwender verursacht und kann von ihm auch nicht behoben werden.
8x01h	Unzulässige Syntaxkennung bei einem ANY-Parameter.
8x22h	Bereichslängenfehler beim Lesen eines Parameters.
8x23h	Bereichslängenfehler beim Schreiben eines Parameters. Dieser Fehlercode zeigt an, dass sich der Parameter x vollständig oder teilweise außerhalb des Operandenbereichs befindet oder die Länge eines Bitfeldes bei einem ANY-Parameter nicht durch 8 teilbar ist.
8x24h	Bereichsfehler beim Lesen eines Parameters.
8x25h	Bereichsfehler beim Schreiben eines Parameters. Dieser Fehlercode zeigt an, dass sich der Parameter x in einem Bereich befindet, der für die Systemfunktion unzulässig ist. Die Beschreibung der jeweiligen Funktion gibt die Bereiche an, die für die Funktion unzulässig sind.
8x26h	Der Parameter enthält eine zu große Nummer einer Zeitzelle. Dieser Fehlercode zeigt an, dass die Zeitzelle, die in Parameter x angegeben wird, nicht vorhanden ist.
8x27h	Der Parameter enthält eine zu große Nummer einer Zählerzelle (Nummernfehler des Zählers). Dieser Fehlercode zeigt an, dass die Zählerzelle, die in Parameter x angegeben wird, nicht vorhanden ist.
8x28h	Ausrichtungsfehler beim Lesen eines Parameters.
8x29h	Ausrichtungsfehler beim Schreiben eines Parameters. Dieser Fehlercode zeigt an, dass der Verweis auf den Parameter x ein Operand ist, dessen Bitadresse ungleich 0 ist.
8x30h	Der Parameter befindet sich in dem schreibgeschützten Global-DB.
8x31h	Der Parameter befindet sich in dem schreibgeschützten Instanz-DB. Dieser Fehlercode zeigt an, dass der Parameter x sich in einem schreibgeschützten Datenbaustein befindet. Wenn der Datenbaustein von der Systemfunktion selbst geöffnet wurde, gibt die Systemfunktion immer den Wert 8x30h aus.
8x32h	Der Parameter enthält eine zu große DB-Nummer (Nummernfehler des DBs).

Fehlercode	Beschreibung
8x34h	Der Parameter enthält eine zu große FC-Nummer (Nummernfehler des FCs).
8x35h	Der Parameter enthält eine zu große FB-Nummer (Nummernfehler des FBs). Dieser Fehlercode zeigt an, dass der Parameter x eine Bausteinnummer enthält, die größer ist als die maximal zulässige Bausteinnummer.
8x3Ah	Der Parameter enthält die Nummer eines DBs, der nicht geladen ist.
8x3Ch	Der Parameter enthält die Nummer eines FCs, der nicht geladen ist.
8x3Eh	Der Parameter enthält die Nummer eines FBs, der nicht geladen ist.
8x42h	Es ist ein Zugriffsfehler aufgetreten, während das System einen Parameter aus dem Peripheriebereich der Eingänge auslesen wollte.
8x43h	Es ist ein Zugriffsfehler aufgetreten, während das System einen Parameter in den Peripheriebereich der Ausgänge schreiben wollte.
8x44h	Fehler beim n-ten (n > 1) Lesezugriff nach Auftreten eines Fehlers.
8x45h	Fehler beim n-ten (n > 1) Schreibzugriff nach Auftreten eines Fehlers. Dieser Fehlercode zeigt an, dass der Zugriff auf den gewünschten Parameter verweigert wird.

7 Organisationsbausteine - *"Organization Blocks"*

7.1 Übersicht

OBs (Organisationsbausteine) bilden die Schnittstelle zwischen Betriebssystem und Anwenderprogramm. Das Hauptprogramm ist immer in OB 1 abzulegen. Die anderen OBs haben entsprechend den Aufrufereignissen festgelegte Nummern. OBs werden entsprechend ihrer zugeordneten Priorität bearbeitet. Das Betriebssystem der CPU ruft die OBs bei folgenden Ereignissen auf:

- Anlauf der CPU
- Zyklisch oder zeitlich getaktete Programmfolge
- Auftreten von Fehlern
- Auftreten von Prozessalarmen

7.2 Hauptprogramm - *"Main"*

7.2.1 OB 1 - Main - Zyklisches Programm

Beschreibung

Das Betriebssystem der CPU bearbeitet den OB 1 zyklisch. Nach dem Übergang von ANLAUF nach RUN startet die zyklische Bearbeitung des OB 1. Der OB 1 hat die niedrigste Priorität (Priorität 1) aller laufzeitüberwachten OBs. Innerhalb des OB 1 haben Sie die Möglichkeit Funktionsbausteine und Funktionen aufzurufen.

Funktionsweise

Wenn die Bearbeitung des OB 1 beendet ist, sendet das Betriebssystem Globaldaten. Bevor der OB 1 neu gestartet wird, schreibt das Betriebssystem das Prozessabbild der Ausgänge in die Ausgabe-Module, aktualisiert das Prozessabbild der Eingänge und empfängt Globaldaten für die CPU.

Zykluszeit

Die Zeit, die für die gesamte Bearbeitung des OB 1 erforderlich ist, nennt man *Zykluszeit*. Hierin eingeschlossen sind beispielsweise die Bearbeitungszeiten für höhere Prioritätsklassen, die das Hauptprogramm unterbrechen bzw. Kommunikationsprozesse des Betriebssystems. Hierzu zählen auch Systemsteuerung der zyklischen Bearbeitung, Aktualisierung des Prozessabbilds und Aktualisierung der Zeitfunktionen.

Sofern Sie mit der CPU online sind, haben Sie die Möglichkeit die aktuelle Zykluszeit mit dem *SPEED7 Studio* zu ermitteln. Gehen Sie im *Projektbaum* auf Ihre CPU und öffnen Sie mit *"Kontextmenü ➔ Baugruppenzustand"* den Dialog "Baugruppenzustand". Unter *"Zykluszeiten"* können Sie die minimale, maximale und aktuelle Zykluszeit ausgeben lassen.

Zykluszeitüberwachung

Die CPU bietet eine Überwachung der *maximalen Zykluszeit*. Der Wert für die *maximale Zykluszeit* ist auf 150ms voreingestellt. Sie können diesen Wert umparametrieren oder aber mit dem SFC 43 (RE_TRIGR) an jeder Stelle Ihres Programms die Zeitüberwachung neu starten. Überschreitet Ihr Programm die *maximale Zykluszeit* für den OB 1, dann ruft das Betriebssystem den OB 80 (Zykluszeitüberschreitung) auf. Ist der OB 80 nicht programmiert, dann geht die CPU in den Betriebszustand STOP. Neben der Überwachung der *maximalen Zykluszeit* kann die Einhaltung einer *minimalen Zykluszeit* garantiert werden. Hierbei verzögert das Betriebssystem den Beginn eines neuen Zyklus (Schreiben des Prozessabbilds der Ausgänge), solange, bis die *minimale Zykluszeit* erreicht ist.

Zugriff auf die Lokaldaten

Das Betriebssystem der CPU übergibt dem OB 1, wie jedem OB, in den ersten 20Byte der temporären Lokaldaten eine Startinformation. Mit der Systemfunktion SFC 6 RD_SINFO können Sie auf diese Startinformation zugreifen. Bitte beachten Sie hierbei, dass Sie die Startinformationen eines OBs nur im OB selbst lesen können, da es sich hier um temporäre Daten handelt.

Lokaldaten

Die folgende Tabelle zeigt die Belegung der Startinformationen für den OB 1, die defaultmäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB1_EV_CLASS	BYTE	Ereignisklasse und Kennung: 11h: OB 1 ist aktiv
OB1_SCAN_1	BYTE	01h: Abschluss des Neustart 03h: Abschluss des freien Zyklus
OB1_PRIORITY	BYTE	Prioritätsklasse: 1
OB1_OB_NUMBR	BYTE	OB-NR. (01)
OB1_RESERVED_1	BYTE	reserviert
OB1_RESERVED_2	BYTE	reserviert
OB1_PREV_CYCLE	INT	Laufzeit des vorherigen Zyklus (ms)
OB1_MIN_CYCLE	INT	Minimale Zykluszeit (ms) seit dem letzten Anlauf
OB1_MAX_CYCLE	INT	Maximale Zykluszeit (ms) seit dem letzten Anlauf
OB1_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde

7.3 Anlauf - *"Startup"*

7.3.1 OB 100, OB 102 - Complete / Cold Restart - Anlauf

Beschreibung

Bei einem Neustart bzw. Anlauf versetzt die CPU sich selbst und die angebundenen Module in den projektierten Grundzustand, löscht die nicht remanenten Daten im Systemspeicher, ruft den Anlauf-OB auf und bearbeitet danach den OB 1 von Anfang an. Hierbei bleiben das aktuelle Programm und die aktuellen Daten im Arbeitsspeicher erhalten, ebenso die per SFC erzeugten Datenbausteine.

Bei der CPU unterscheidet man folgende Anlaufarten:

- OB 100: Complete Restart - Neustart
- OB 102: Cold Restart - Kaltstart

Startereignisse für den Anlauf:

- nach NetzEIN mit Betriebsartenschalter in RUN
- nach Umschalten des Betriebsartenschalters von STOP in RUN
- nach einer RUN/STOP-Anforderung durch eine Kommunikationsfunktion

Auch wenn kein Anlauf-OB in die CPU geladen ist, geht diese ohne Fehlermeldung in RUN.

Lokaldaten

Nachfolgend sehen Sie die Belegung der Startinformationen für den Anlauf-OB, die defaultmäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB10x_EV_CLASS	BYTE	Ereignisklasse und Kennung: 13h: aktiv
OB10x_STRTUP	BYTE	Anlaufanforderung ■ 81h: Manuelle Neustart-Anforderung ■ 82h: Automatische Neustart-Anforderung ■ 85h: Manuelle Kaltstartanforderung ■ 86h: Automatische Kaltstartanforderung ■ 8Ah: Master: Manuelle Neustart-Anforderung ■ 8Bh: Master: Automatische Neustart-Anforderung
OB10x_PRIORITY	BYTE	Prioritätsklasse: 27
OB10x_OB_NUMBR	BYTE	OB-NR. (100 oder 102)
OB10x_RESERVED_1	BYTE	reserviert
OB10x_RESERVED_2	BYTE	reserviert
OB10x_STOP	WORD	Nummer des Ereignisses, das die CPU in STOP versetzt hat
OB10x_STRT_INFO	DWORD	Ergänzende Informationen zum aktuellen Anlauf
OB10x_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde

Informationen, wie Sie auf die Lokaldaten zugreifen können finden Sie in der Beschreibung zum OB 1.

Belegung
OB10x_STRT_INFO

Bit-Nr.	Bedeutung	mögliche Werte (binär)	Beschreibung
31...24	Anlaufinformation	xxxx xxx0	keine Soll-Istausbau-Differenz vorhanden
		xxxx xxx1	Soll-Istausbau-Differenz vorhanden
		xxxx 0xxx	Uhr für Zeitstempel bei letztem NetzEIN nicht gepuffert
		xxxx 1xxx	Uhr für Zeitstempel bei letztem NetzEIN gepuffert
23...16	Soeben durchgeführte Anlaufart	0000 0011	Neustart über Betriebsartenschalter
		0000 0100	Neustart über MPI-Bedienung
		0000 0111	Kaltstart über Betriebsartenschalter
		0000 1000	Kaltstart über MPI-Bedienung
		0001 0000	Automatischer Neustart nach gepuffertem NetzEIN
		0001 0011	Neustart über Betriebsartenschalter; letzter NetzEIN gepuffert
		0001 0100	Neustart über MPI-Bedienung; letzter NetzEIN gepuffert
		0010 0000	Automatischer Neustart nach ungepuffertem NetzEIN (mit systemseitigem Urlöschen)
		0010 0011	Neustart über Betriebsartenschalter; letzter NetzEIN ungepuffert
		0010 0100	Neustart über MPI-Bedienung; letzter NetzEIN ungepuffert

Bit-Nr.	Bedeutung	mögliche Werte (binär)	Beschreibung
15...12	Zulässigkeit automatischer Anlaufarten	0000	Automatischer Anlauf unzulässig, Urlöschen angefordert
		0001	Automatischer Anlauf unzulässig, Parameteränderung erforderlich
		0111	Automatischer Neustart zulässig
11...8	Zulässigkeit manueller Anlaufarten	0000	Manueller Anlauf unzulässig, Urlöschen angefordert
		0001	Manueller Anlauf unzulässig, Parameteränderung erforderlich
		0111	Manueller Neustart zulässig
7...0	Letzte gültige Bedienung oder Einstellung der automatischen Anlaufart bei NetzEIN	0000 0000	Keine Anlaufart
		0000 0011	Neustart durch Betriebsartenschalter
		0000 0100	Neustart über MPI-Bedienung
		0001 0000	Automatischer Neustart nach gepuffertem NetzEIN
		0001 0011	Neustart über Betriebsartenschalter, letzter NetzEIN gepuffert
		0001 0100	Neustart über MPI-Bedienung, letzter NetzEIN gepuffert
		0010 0000	Automatischer Neustart nach ungepuffertem NetzEIN (mit systemseitigem Urlöschen)
		0010 0011	Neustart über Betriebsartenschalter, letzter NetzEIN ungepuffert
		0010 0100	Neustart über MPI-Bedienung, letzter NetzEIN ungepuffert

7.4 Kommunikationsalarme - *"Communication Interrupts"*

7.4.1 OB 55 - DP: Status Alarm - Statusalarm

Beschreibung

Den Statusalarm-OB (OB 55) gibt es nur bei DP-V1-fähigen CPUs.

Das Betriebssystem der CPU ruft den OB 55 auf, wenn von einem Steckplatz eines DP-V1-Slaves ein Statusalarm ausgelöst wurde. Dies kann der Fall sein, wenn eine Komponente (Modul) eines DP-V1-Slaves ihren Betriebszustand wechselt, z.B. von RUN nach STOP. Die genauen Ereignisse, die einen Statusalarm auslösen, entnehmen Sie der Dokumentation des DP-V1-Slave-Herstellers.

Lokaldaten

Nachfolgend sehen Sie die Belegung der Startinformationen für den OB 55, die defaultmäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB55_EV_CLASS	BYTE	Ereignisklasse und Kennung: 11h: kommendes Ereignis
OB55_STRT_INF	BYTE	55h: Statusalarm bei DP 58h: Statusalarm bei PROFINET IO

Variable	Datentyp	Beschreibung
OB55_PRIORITY	BYTE	Parametrierte Prioritätsklasse: Defaultwert: 2
OB55_OB_NUMBR	BYTE	OB-NR. (55)
OB55_RESERVED_1	BYTE	reserviert
OB55_IO_FLAG	BYTE	Eingabe-Module: 54h Ausgabe-Module: 55h
OB55_MDL_ADDR	WORD	Logische Basisadresse des Moduls, das den Alarm auslöst
OB55_LEN	BYTE	Länge des Datenblocks, den der Alarm liefert
OB55_TYPE	BYTE	Kennung für den Alarmtyp "Statusalarm"
OB55_SLOT	BYTE	Steckplatz-Nr. der Alarm auslösenden Komponente
OB55_SPEC	BYTE	Specifier: ■ Bit 1, 0: Alarm-Specifier ■ Bit 2: Add_Ack ■ Bit 7... 3: Seq-Nummer
OB55_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde

Die vollständige Alarmzusatzinformation aus dem Telegramm erhalten Sie durch Aufruf des SFB 54 "RALRM" innerhalb des OB 55. *Kap. 16.2.22 "SFB 54 - RALRM - Alarm von einer Peripheriebaugruppe empfangen" Seite 993*

7.4.2 OB 56 - DP: Update Alarm - Update-Alarm

Beschreibung

Den Update-Alarm-OB (OB 56) gibt es nur bei DP-V1-fähigen CPUs.

Das Betriebssystem der CPU ruft den OB 56 auf, wenn von einem Steckplatz eines DP-V1-Slaves ein Update-Alarm ausgelöst wurde. Dies kann der Fall sein, wenn Sie an einem Steckplatz eines DP-V1-Slaves eine Parameteränderung durchgeführt haben. Die genauen Ereignisse, die einen Update-Alarm auslösen, entnehmen Sie der Dokumentation des DP-V1-Slave-Herstellers.

Lokaldaten

Nachfolgend sehen Sie die Belegung der Startinformationen für den OB 56, die defaultmäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB56_EV_CLASS	BYTE	Ereignisklasse und Kennung: 11h: kommendes Ereignis
OB56_STRT_INF	BYTE	56h: Update-Alarm bei DP 59h: Update-Alarm bei PROFINET IO

Variable	Datentyp	Beschreibung
OB56_PRIORITY	BYTE	Parametrierte Prioritätsklasse: Defaultwert: 2
OB56_OB_NUMBR	BYTE	OB-NR. (56)
OB56_RESERVED_1	BYTE	reserviert
OB56_IO_FLAG	BYTE	Eingabe-Module: 54h Ausgabe-Module: 55h
OB56_MDL_ADDR	WORD	Logische Basisadresse des Moduls, das den Alarm auslöst
OB56_LEN	BYTE	Länge des Datenblocks, den der Alarm liefert
OB56_TYPE	BYTE	Kennung für den Alarmtyp "Update-Alarm"
OB56_SLOT	BYTE	Steckplatz-Nr. der Alarm auslösenden Komponente
OB56_SPEC	BYTE	Specifier: ■ Bit 1, 0: Alarm-Specifier ■ Bit 2: Add_Ack ■ Bit 7... 3: Seq-Nummer
OB56_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde

Die vollständige Alarmzusatzinformation aus dem Telegramm erhalten Sie durch Aufruf des SFB 54 "RALRM" innerhalb des OB 56. *↳ Kap. 16.2.22 "SFB 54 - RALRM - Alarm von einer Peripheriebaugruppe empfangen" Seite 993*

7.4.3 OB 57 - DP: Manufacture Alarm - Herstellerspezifische Alarme

Beschreibung

Das Betriebssystem der CPU ruft den OB 57 auf, wenn von einem Steckplatz eines Slaves-Systems ein herstellerspezifischer Alarm ausgelöst wurde.

Lokaldaten

Nachfolgend sehen Sie die Belegung der Startinformationen für den OB 57, die defaultmäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB57_EV_CLASS	BYTE	Ereignisklasse und Kennung: 11h: kommendes Ereignis
OB57_STRT_INF	BYTE	57h: Startinformation für OB 57
OB57_PRIORITY	BYTE	Parametrierte Prioritätsklasse: Defaultwert: 2
OB57_OB_NUMBR	BYTE	OB-NR. (57)
OB57_RESERVED_1	BYTE	reserviert
OB57_IO_FLAG	BYTE	Eingabe-Module: 54h Ausgabe-Module: 55h
OB57_MDL_ADDR	WORD	Logische Basisadresse des Moduls, das den Alarm auslöst
OB57_LEN	BYTE	reserviert
OB57_TYPE	BYTE	reserviert
OB57_SLOT	BYTE	reserviert
OB57_SPEC	BYTE	reserviert
OB57_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde

Die vollständige Alarmzusatzinformation aus dem Telegramm erhalten Sie durch Aufruf des SFB 54 "RALRM" innerhalb des OB 57.

7.5 Verzögerungsalarme - *"Time delay Interrupts"*

7.5.1 OB 20, OB 21 - DEL_INTx - Verzögerungsalarm

Beschreibung

Mit einem Verzögerungsalarm haben Sie die Möglichkeit, unabhängig von den Zeitfunktionen eine Zeitverzögerung zu realisieren. Sie können die Verzögerungsalarme in der Hardware-Konfiguration projektieren bzw. mit Systemfunktionen vom Programm aus zur Laufzeit steuern.

Aktivierung

Für die Aktivierung ist keine Hardware-Konfiguration erforderlich. Sie starten den Verzögerungsalarm durch Aufruf des SFC 32 SRT_DINT und durch Laden des entsprechenden OBs in die CPU. Übergeben Sie hierbei der Funktion OB-Nr., Verzögerungszeit und eine Kennzeichnung. Nach der angegebenen Verzögerungszeit wird vom Betriebssystem der zugehörige OB aufgerufen. Sie haben die Möglichkeit mit dem SFC 33 CAN_DINT einen Verzögerungsalarm, der noch nicht aktiviert ist, zu stornieren bzw. mit dem SFC 34 QRY_DINT den Zustand des Verzögerungsalarms zu ermitteln. Mit den SFC 39 DIS_IRT können Sie Verzögerungsalarme sperren und mit dem SFC 40 EN_IRT wieder freigeben. Mittels einer Hardware-Konfiguration können Sie die Priorität der entsprechenden OBs verändern. Hierzu öffnen sie die CPU mit **Bearbeiten** > *Objekteigenschaften* > *Alarme*. Hier können Sie die entsprechenden Prioritäten einstellen.

Fehlerverhalten

Fehlt zum Zeitpunkt seines Aufrufs der Verzögerungsalarm-OB im Anwenderprogramm, ruft das Betriebssystem den OB 85 auf. Ist dieser nicht vorhanden, geht die CPU in STOP.

Lokaldaten

Nachfolgend sehen Sie die Belegung der Startinformationen für OB 20 und OB 21, die defaultmäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB20_EV_CLASS	BYTE	Ereignisklasse und Kennung: 11h: Verzögerungsalarm ist aktiv
OB20_STRT_INF	BYTE	21h: Startanforderung für OB 20 22h: Startanforderung für OB 21
OB20_PRIORITY	BYTE	Parametrierte Prioritätsklasse: Defaultwerte: 3 (OB 20) 4 (OB 21)
OB20_OB_NUMBR	BYTE	OB-NR. (20, 21)
OB20_RESERVED_1	BYTE	reserviert
OB20_RESERVED_2	BYTE	reserviert
OB20_SIGN	WORD	Anwenderkennung: Eingangsparameter SIGN aus dem Aufruf des SFC 32 (SRT_DINT)
OB20_DTIME	TIME	Abgelaufene Verzögerungszeit in ms
OB20_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde

Informationen, wie Sie auf die Lokaldaten zugreifen können finden Sie in der Beschreibung zum OB 1.

7.6 Uhrzeitalarme - *"Time of day Interrupts"*

7.6.1 OB 10, OB 11 - TOD_INTx - Uhrzeitalarm

Beschreibung

Sie verwenden Uhrzeitalarme, wenn Sie ein Programm zu einer bestimmten Uhrzeit einmalig oder periodisch bearbeiten lassen möchten. Sie können die Uhrzeitalarme in der Hardware-Konfiguration projektieren oder mit Systemfunktionen vom Programm aus zur Laufzeit steuern. Voraussetzung für die ordnungsgemäße Bearbeitung der Uhrzeitalarme ist eine richtig eingestellte Echtzeituhr auf der CPU. Bei der Ausführung haben Sie folgende Auswahlmöglichkeiten:

- einmalig
- minütlich
- stündlich
- täglich
- wöchentlich
- monatlich
- jährlich
- am Monatsende

Bei monatlicher Bearbeitung eines Uhrzeitalarm-OBs sind beim Startdatum nur die Tage 1, 2, ... 28 möglich.

Aktivierung

Um einen Uhrzeitalarm zu starten, müssen Sie den Alarm erst einstellen und dann aktivieren. Hierzu gibt es folgende Möglichkeiten:

1. Die Aktivierung der Uhrzeitalarme geschieht über die Hardware- Konfiguration. Sie öffnen die CPU mit **Bearbeiten** > *Objekteigenschaften* > *Uhrzeitalarme*. Hier können Sie die entsprechenden Uhrzeitalarm einstellen und aktivieren. Nach Übertragung in die CPU und dem Übergang von Anlauf in RUN wird die Uhrzeitalarmüberwachung automatisch gestartet.
2. Sie stellen den Uhrzeitalarm wie oben gezeigt über die Hardware- Konfiguration und aktivieren ihn durch Aufrufen des SFC 30 ACT_TINT aus Ihrem Anwenderprogramm heraus.
3. Sie stellen den Uhrzeitalarm durch Aufruf des SFC 28 SET_TINT und aktivieren ihn durch Aufruf des SFC 30 ACT_TINT.

Sie können den Uhrzeitalarm mit Hilfe des SFC 41 DIS_AIRT verzögern bzw. die Verzögerung mit dem SFC 42 EN_AIRT aufheben.

Fehlerverhalten

Fehlt zum Zeitpunkt seines Aufrufs der Uhrzeitalarm-OB im Anwenderprogramm, ruft das Betriebssystem den OB 85 auf. Ist dieser nicht vorhanden, geht die CPU in STOP. Kommt es zu einem Fehler bei der Uhrzeitalarmbearbeitung z.B. Startzeitpunkt liegt in der Vergangenheit, wird der Zeitfehler-OB OB 80 aufgerufen und danach der Uhrzeitalarm-OB einmalig bearbeitet.

Möglichkeiten bei der Aktivierung

Die folgende Tabelle zeigt die prinzipiellen Möglichkeiten beim Aktivieren eines Uhrzeitalarms:

Intervall	Beschreibung
Nicht aktiviert	Der Uhrzeitalarm-OB wird nicht bearbeitet. Sie können den Uhrzeitalarm aktivieren durch Aufruf des SFC 30.
Einmalig aktiviert	Der Uhrzeitalarm-OB wird automatisch storniert, nachdem er einmal wie angegeben bearbeitet wurde. Ihr Programm kann mit dem SFC 28 den Uhrzeitalarm erneut stellen und ihn mit dem SFC 30 wieder aktivieren.
Periodisch aktiviert	Die CPU ermittelt zum Zeitpunkt des Uhrzeitalarms aus der aktuellen Uhrzeit und der Periode den nächsten Startzeitpunkt des Uhrzeitalarms.

Lokaldaten des Uhrzeitalarm-OBs

Nachfolgend sehen Sie die Belegung der Startinformationen für OB 10 ... OB 11, die defaultmäßige symbolische Bezeichnung und die Datentypen. Als Variablennamen wurden die Defaultnamen des OB 10 gewählt.

Variable	Datentyp	Beschreibung
OB10_EV_CLASS	BYTE	Ereignisklasse und Kennung: 11h: Alarm ist aktiv
OB10_STRT_INFO	BYTE	11h: Startanforderung für OB 10 12h: Startanforderung für OB 11
OB10_PRIORITY	BYTE	Parametrierte Prioritätsklasse: Defaultwert 2
OB10_OB_NUMBR	BYTE	OB-NR. (10 ... 11)
OB10_RESERVED_1	BYTE	reserviert
OB10_RESERVED_2	BYTE	reserviert
OB10_PERIOD_EXE	WORD	Der OB wird zu dem angegebenen Intervall bearbeitet: 0000h: einmalig 0201h: minütlich 0401h: stündlich 1001h: täglich 1201h: wöchentlich 1401h: monatlich 1801h: jährlich 2001h: am Monatsende
OB10_RESERVED_3	INT	reserviert
OB10_RESERVED_4	INT	reserviert
OB10_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde

Informationen, wie Sie auf die Lokaldaten zugreifen können finden Sie in der Beschreibung zum OB 1.

7.7 Zyklusalarme - *"Cyclic Interrupts"*

7.7.1 OB 28, 29, 32, 33, 34, 35 - CYC_INTx - Weckalarm

Beschreibung

Mit Hilfe eines Weckalarm OBs können Sie die zyklische Programmbearbeitung in äquidistanten Zeitabständen unterbrechen. Startzeitpunkt des Zeittakts ist der Betriebszustandwechsel von ANLAUF in RUN nach Abarbeitung des Anlauf-OB OB 100.

Weckalarm-OB	Defaultwert für Zeitraster	Defaultwert für die Prioritätsklasse	Möglichkeit zur Phasenverschiebung
OB 28	250µs	24	nein*
OB 29	500µs	24	nein*
OB 32	1s	09	ja
OB 33	500ms	10	ja
OB 34	200ms	11	ja
OB 35	100ms	12	ja

*) Sind beide OBs aktiviert, wird zuerst OB 28 und direkt danach OB 29 abgearbeitet. Aufgrund der sehr kleinen Zeitraster und der hohen Prioritäten sollte auf den gleichzeitigen Einsatz von OB 28 und OB 29 verzichtet werden.

Aktivierung

Einen Weckalarm aktivieren Sie, indem Sie den entsprechenden OB in Ihrer CPU programmieren. Sie können den Weckalarm mit Hilfe des SFC 41 DIS_AIRT verzögern bzw. die Verzögerung mit dem SFC 42 EN_AIRT aufheben.

Funktionsweise

Nach dem Übergang von ANLAUF in RUN werden unter Berücksichtigung der Phasenverschiebung die aktivierten Weckalarm-OBs in den vorgegebenen äquidistanten Intervallen aufgerufen. Die äquidistanten Startzeitpunkte der Weckalarm-OBs ergeben sich aus dem jeweiligen Zeitraster und der Phasenverschiebung. Durch Programmieren der entsprechenden OBs können Sie auf diese Weise zeitgesteuert ein Unter-Programm ausführen.

Phasenverschiebung

Die Phasenverschiebung dient zur zeitversetzten Bearbeitung von Weckalarmen mit einem gemeinsamen Vielfachen im Zeitintervall. Auf diese Weise erhalten Sie eine höhere Genauigkeit der Zeitintervalle. Der Startzeitpunkt der Zeitintervalle ist der Übergang des Betriebszustands ANLAUF nach RUN. Danach ist der Aufrufzeitpunkt für einen Weckalarm- OB das Zeitintervall plus die parametrierte Phasenverschiebung.

Parametrierung

Ausführzeit, Phasenverschiebung (nicht OB 28, 29) und Priorität können Sie über den Hardware-Konfigurator parametrieren.

Achten Sie beim Vorgeben der Zeittakte darauf, dass zwischen den Startereignissen der einzelnen Weckalarme genügend Zeit für die Bearbeitung der Weckalarme bleibt. Der fehlerverursachende Weckalarm wird nachgeholt.

Lokaldaten

Nachfolgend sehen Sie die Belegung der Startinformationen, die defaultmäßige symbolische Bezeichnung und die Datentypen. Als Variablennamen wurden die Defaultnamen des OB 35 gewählt.

Variable	Datentyp	Beschreibung
OB35_EV_CLASS	BYTE	Ereignisklasse und Kennung: 11h: Weckalarm ist aktiv
OB35_STRT_INF	BYTE	2Fh: Startanforderung für OB 28 30h: Startanforderung für OB 29 33h: Startanforderung für OB 32 34h: Startanforderung für OB 33 35h: Startanforderung für OB 34 36h: Startanforderung für OB 35
OB35_PRIORITY	BYTE	Parametrierte Prioritätsklasse; Defaultwerte: 24 (OB 28, 29), 9 (OB 32) ... 12 (OB 35)
OB35_OB_NUMBR	BYTE	OB-NR. (28, 29, 32 ... 35)
OB35_RESERVED_1	BYTE	reserviert
OB35_RESERVED_2	BYTE	reserviert

Variable	Datentyp	Beschreibung
OB35_PHASE_OFFSET	WORD	Phasenverschiebung in ms
OB35_RESERVED_3	INT	reserviert
OB35_EXC_FREQ	INT	Zeittakt in ms
OB35_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit der OB-Anforderung

Informationen, wie Sie auf die Lokaldaten zugreifen können finden Sie in der Beschreibung zum OB 1.

Da die Bausteine SFC58/59 bzw. SFB52/53 zum Schreiben und Lesen von Datensätzen nicht unterbrechbar sind, kann es in Verbindung mit OB 28 und OB 29 zu einem CPU-STOP kommen!

7.8 Prozessalarme - *"Hardware Interrupts"*

7.8.1 OB 40, OB 41 - HW_INTx - Prozessalarm

Beschreibung

Sie verwenden Prozessalarme zur sofortigen Erfassung von Ereignisse aus dem gesteuerten Prozess in ihrem Anwenderprogramm. Hier können Sie dem entsprechenden Programm darauf reagieren. Hierzu stehen Ihnen die OB 40 und OB 41 zur Verfügung. Durch Parametrierung legen Sie für jedes Modul fest, welche Kanäle bei welcher Randbedingung einen Prozessalarm auslösen. Mit den SFC 55 WR_PARM, SFC 56 WR_DPARM und SFC 57 PARM_MOD können Sie die prozessalarmerfassenden Module auch während des laufenden Betriebs (um-)parametrieren. *Kap. 16.1.42 "SFC 55 - WR_PARM - Dynamische Parameter schreiben" Seite 905* *Kap. 16.1.43 "SFC 56 - WR_DPARM - Vordefinierte Parameter schreiben" Seite 907* *Kap. 16.1.44 "SFC 57 - PARM_MOD - Modul parametrieren" Seite 909*

Aktivierung

Die Prozessalarmverwaltung der CPU ist immer aktiv. Damit ein Modul einen Prozessalarm auslösen kann, müssen Sie auf dem entsprechenden Modul über eine Hardware-Konfiguration die Auslösung eines Prozessalarms freigeben. Hierbei können Sie wählen, ob der Prozessalarm bei kommendem, gehenden oder bei beiden Ereignis ausgelöst werden soll.

Funktionsweise

Nach Auslösen eines Prozessalarms durch das Modul identifiziert das Betriebssystem den Steckplatz und ruft den Prozessalarm-OB auf. Hat dieser eine höhere Priorität als die momentan aktive Prioritätsklasse, wird dieser gestartet. Nach der Bearbeitung des zum Prozessalarm gehörenden OB wird der Prozessalarm auf dem Modul quittiert. Tritt in der Zeit zwischen der Identifikation und der Quittung eines Prozessalarms auf demselben Modul erneut ein Ereignis auf, das einen Prozessalarm auslösen soll, gilt:

- Tritt das Ereignis bei dem Kanal auf, der zuvor den Prozessalarm ausgelöst hat, so geht der zugehörige Alarm verloren.
- Tritt das Ereignis bei einem anderen Kanal desselben Moduls auf, so kann momentan kein Prozessalarm ausgelöst werden. Steht nach der Quittierung des vorhergehenden Ereignisses das "neue" Alarmereignis noch an, wird ein Prozessalarm ausgelöst. Ansonsten geht der Alarm verloren.
- Soll ein Prozessalarm ausgelöst werden, dessen OB aufgrund eines Prozessalarms eines anderen Moduls aktiv ist, so kann nur dann ein Prozessalarm ausgelöst werden, wenn nach der Quittierung das auslösende Ereignis noch ansteht.

Im Anlauf erzeugt die Baugruppe keine Prozessalarme. Die Alarmbearbeitung beginnt beim Übergang in den Betriebszustand RUN. Beim Übergang anstehende Prozessalarme gehen verloren.

Fehlerverhalten

Fehlt zum Zeitpunkt seines Aufrufs der Prozessalarm-OB im Anwenderprogramm, ruft das Betriebssystem den OB 85 auf. Der Prozessalarm wird quittiert. Ist der OB 85 nicht vorhanden, geht die CPU in STOP.

Diagnosealarm

Während der Bearbeitung eines Prozessalarms kann ein Diagnosealarm ausgelöst werden. Tritt in der Zeit vom Auslösen des Prozessalarms bis zu dessen Quittierung auf demselben Kanal ein weiterer Prozessalarm auf, wird über einen Diagnosealarm der Prozessalarmverlust zur Systemdiagnose gemeldet.

Lokaldaten

Nachfolgend sehen Sie die Belegung der Startinformationen für OB 40 und OB 41, die defaultmäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB40_EV_CLASS	BYTE	Ereignisklasse und Kennung: 11h: Alarm ist aktiv
OB40_STRT_INF	BYTE	41h: Alarm über Interruptleitung 1
OB40_PRIORITY	BYTE	Parametrierte Prioritätsklasse: Defaultwert: 16 (OB 40) Defaultwert: 17 (OB 41)
OB40_OB_NUMBR	BYTE	OB-NR. (40, 41)
OB40_RESERVED_1	BYTE	reserviert
OB40_IO_FLAG	BYTE	Eingabe-Module 54h Ausgabe-Module 55h
OB40_MDL_ADDR	WORD	Logische Basisadresse des Moduls, das den Alarm auslöst
OB40_POINT_ADDR	DWORD	■ Bei Digital-Modulen – Bitfeld mit den Zuständen der Eingänge auf dem Modul (Bit 0 entspricht dem ersten Eintrag). ■ Bei Analog-Modulen – Bitfeld mit Informationen welcher Kanal welche Grenzen erreicht hat. ■ Bei CPs oder IMs – Informiert über den Alarmzustand des Moduls.
OB40_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde

Informationen, wie Sie auf die Lokaldaten zugreifen können finden Sie in der Beschreibung zum OB 1.

7.9 Asynchrone Fehleralarme - *"Asynchronous error Interrupts"*

7.9.1 OB 80 - CYCL_FLT - Zeitfehler

Beschreibung

Das Betriebssystem der CPU ruft den OB 80 auf, wenn einer der folgenden Fehler auftritt:

- Überschreiten der Zyklusüberwachungszeit
- Quittungsfehler bei der Bearbeitung eines OBs d.h. der abgeforderte OB ist noch in Bearbeitung bzw. ein OB wird innerhalb einer Prioritätsklasse zu oft angefordert.
- Uhrzeitfehleralarm, d.h. abgelaufener Uhrzeitalarm durch Vorstellen der Uhrzeit oder nach Übergang in RUN.

Sie können den Zeitfehler-OB mit Hilfe der SFC 39 ... 42 sperren bzw. verzögern und wieder freigeben.

Haben Sie den OB 80 nicht programmiert, geht die CPU in den Betriebszustand STOP.

Wird der OB 80 in demselben Zyklus zweimal aufgrund der Zykluszeitüberschreitung aufgerufen, geht die CPU in STOP. Sie können dies durch Aufruf des SFC 43 RE_TRIGR an geeigneter Stelle verhindern.

Lokaldaten

Nachfolgend sehen Sie die Belegung der Startinformationen für den OB 80, die defaultmäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB80_EV_CLASS	BYTE	Ereignisklasse und Kennung: 35h
OB80_FLT_ID	BYTE	Fehlercode (mögliche Werte: 01h, 02h, 05h, 06h, 07h, 08h, 09h, 0Ah)
OB80_PRIORITY	BYTE	Prioritätsklasse: 26 (Betriebszustand RUN) 28 (Überlauf des OB-Anforderungspuffers)
OB80_OB_NUMBR	BYTE	OB-NR. (80)
OB80_RESERVED_1	BYTE	reserviert
OB80_RESERVED_2	BYTE	reserviert
OB80_ERROR_INFO	WORD	Fehlerinformation: abhängig vom Fehlercode
OB80_ERR_EV_CLASS	BYTE	Klasse des Ereignisses, das den Fehler ausgelöst hat
OB80_ERR_EV_NUM	BYTE	Nummer des Ereignisses, das den Fehler ausgelöst hat
OB80_OB_PRIORITY	BYTE	Fehlerinformation: abhängig vom Fehlercode
OB80_OB_NUM	BYTE	Fehlerinformation: abhängig vom Fehlercode
OB80_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde

Informationen, wie Sie auf die Lokaldaten zugreifen können finden Sie in der Beschreibung zum OB 1.

Belegung der Variablen nach Fehlercode

Die vom Fehlercode abhängigen Variablen haben folgende Bedeutung:

Fehlercode	Variable	Bit	Beschreibung
01h			*Zykluszeit überschritten*
	OB80_ERROR_INFO		Laufzeit des letzten Zyklus (ms)

Fehlercode	Variable	Bit	Beschreibung
	OB80_ERR_EV_CLASS		Klasse des Ereignisses, das den Alarm ausgelöst hat
	OB80_ERR_EV_NUM		Nummer des Ereignisses, das den Alarm ausgelöst hat
	OB80_OB_PRIORITY		Prioritätsklasse des OBs, der bearbeitet wurde, als der Fehler auftrat
	OB80_OB_NUM		Nummer des OBs, der bearbeitet wurde, als der Fehler auftrat
02h			*Der angeforderte OB ist noch in Bearbeitung*
	OB80_ERROR_INFO		Die zugehörige temporäre Variable des angeforderten OB. Diese ist bestimmt durch: OB80_ERR_EV_CLASS und OB80_ERR_EV_NUM
	OB80_ERR_EV_CLASS		Klasse des Ereignisses, das den Alarm ausgelöst hat
	OB80_ERR_EV_NUM		Nummer des Ereignisses, das den Alarm ausgelöst hat
	OB80_OB_PRIORITY		Prioritätsklasse des fehlerverursachenden OBs
	OB80_OB_NUM		Nummer des fehlerverursachenden OBs
05h und 06h			*Abgelaufener Uhrzeitalarm durch Uhrzeitsprung*
			Abgelaufener Uhrzeitalarm bei Wiedereintritt in RUN nach HALT
	OB80_ERROR_INFO	Bit 0 = "1"	Für den Uhrzeitalarm 0 liegt der Startzeitpunkt in der Vergangenheit.
		...	....
		Bit 7 = "1"	Für den Uhrzeitalarm 7 liegt der Startzeitpunkt in der Vergangenheit
		Bit 15 ... 8	nicht verwendet
	OB80_ERR_EV_CLASS		nicht verwendet
	OB80_ERR_EV_NUM		nicht verwendet
	OB80_OB_PRIORITY		nicht verwendet
	OB80_OB_NUM		nicht verwendet

Fehlercode	Variable	Bit	Beschreibung
07h	Bedeutung der Parameter siehe Fehlercode 02h		*Überlauf des OB-Anforderungspuffers für die aktuelle Prioritätsklasse* (jede OB-Startanforderung für eine Prioritätsklasse wird in den zugehörigen OB-Anforderungspuffer eingetragen; nach Beendigung des OBs wird der Eintrag wieder gelöscht. Falls für eine Prioritätsklasse mehr Startanforderungen vorliegen als die maximal mögliche Anzahl der Einträge im zugehörigen OB-Anforderungspuffer, wird der OB 80 mit dem Fehlercode 07h aufgerufen.)
08h			*Taktsynchronalarm-Zeitfehler*
09h			*Alarmverlust durch zu hohe Alarmlast*
0Ah	OB80_ERROR_INFO		*Wiedereintritt in RUN nach CiR (**C**onfiguration **i**n **R**UN)* CiR-Synchronisationszeit in ms

7.9.2 OB 81 - PS_FLT - Stromversorgungsfehler

Beschreibung

Das Betriebssystem der CPU ruft den OB 81 auf, wenn ein Ereignis auftritt, das durch einen Fehler in der Stromversorgung oder der Pufferung ausgelöst wird. Eine Überprüfung erfolgt ausschließlich nach Netz EIN.

Ist im Fehlerfall der OB 81 in der CPU nicht vorhanden, bleibt diese im RUN.

Sie können den Stromversorgungsfehler-OB mit Hilfe der SFCs 39 ... 42 sperren bzw. verzögern und wieder freigeben.

Lokaldaten

Nachfolgend sehen Sie die Belegung der Startinformationen für den OB 81, die defaultmäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB81_EV_CLASS	BYTE	Ereignisklasse und Kennung: 39h: kommendes Ereignis
OB81_FLT_ID	BYTE	Fehlercode: 22h: Pufferspannung in CPU fehlt
OB81_PRIORITY	BYTE	Prioritätsklasse: 28 (Betriebszustand ANLAUF)
OB81_OB_NUMBR	BYTE	OB-NR. (81)
OB81_RESERVED_1	BYTE	reserviert
OB81_RESERVED_2	BYTE	reserviert
OB81_RACK_CPU	WORD	Bit 2 ... 0: 000 (Baugruppenträgernummer) Bit 3: 1 (Master-CPU) Bit 7 ... 4: 1111 (fix)
OB81_RESERVED_3	BYTE	reserviert

Variable	Datentyp	Beschreibung
OB81_RESERVED_4	BYTE	reserviert
OB81_RESERVED_5	BYTE	reserviert
OB81_RESERVED_6	BYTE	reserviert
OB81_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde.

Informationen, wie Sie auf die Lokaldaten zugreifen können finden Sie in der Beschreibung zum OB 1.

7.9.3 OB 82 - I/O_FLT1 - Diagnosealarm

Beschreibung

- Die Systemdiagnose ist die Erkennung, Auswertung und die Meldung von Fehlern, die innerhalb Ihres Automatisierungssystems auftreten. Dies können Fehler im Anwenderprogramm oder Ausfälle auf Modulen aber auch Drahtbrüche bei Signal-Modulen sein.
- Wenn ein diagnosefähiges Modul, bei dem Sie den Diagnosealarm freigegeben haben, einen Fehler erkennt, stellt dies eine Diagnosealarmanforderung sowohl bei kommendem als auch bei gehendem Ereignis an die CPU. Daraufhin ruft das Betriebssystem den OB 82 auf.
- Der OB 82 enthält in seinen lokalen Variablen die logische Basisadresse sowie eine vier Byte lange Diagnoseinformation des fehlerhaften Moduls.
- Haben Sie den OB 82 nicht programmiert, geht die CPU in den Betriebszustand STOP. Sie können den Diagnosealarm mit Hilfe des SFC 41 DIS_AIRT verzögern bzw. die Verzögerung mit dem SFC 42 EN_AIRT aufheben.

Diagnose in Ringpuffer

Alle an das Betriebssystem der CPU gemeldeten Diagnoseereignisse werden in einem Diagnosepuffer in der Reihenfolge ihres Auftretens mit Datum und Uhrzeit abgelegt. Dies ist ein gepufferter Speicherbereich der CPU, der seinen Inhalt auch nach Urlöschen behält.

- Dieser Diagnosepuffer ist als Ringpuffer aufgebaut und bietet bei den VIPA-CPUs Platz für 100 Diagnoseeinträge.
- Ist der Diagnosepuffer voll, wird der jeweils älteste Eintrag vom aktuellen Diagnoseereignis überschrieben.
- Sofern Sie mit der CPU online sind, haben Sie die Möglichkeit den Diagnosepuffer mit dem *SPEED7 Studio* zu ermitteln. Gehen Sie im *Projektbaum* auf Ihre CPU und öffnen Sie mit *"Kontextmenü ➔ Baugruppenzustand"* den Dialog "Baugruppenzustand". Unter *"Diagnosepuffer"* können Sie den Diagnosepuffer ausgeben lassen.

Parametrierbare Diagnose

Bei den parametrierbaren Diagnoseereignissen erfolgt eine Meldung nur dann, wenn Sie mittels Parametrierung die Diagnose freigegeben haben. Die nicht parametrierbaren Diagnoseereignisse werden unabhängig von der Diagnosefreigabe immer gemeldet.

Diagnose mittels SFC eintragen

Mit der Systemfunktion SFC 52 WR_USMSG können Sie einen Eintrag in den Diagnosepuffer schreiben.

Diagnosedaten mittels SFC 59 auslesen

Innerhalb des OB 82 können Sie mittels SFC 59 RD_REC (Datensatz lesen) auf detaillierte Fehlerinformationen zugreifen. Die Diagnosedaten sind bis zum Verlassen des OB 82 konsistent. Mit dem Verlassen des OB 82 wird der Diagnosealarm quittiert. Die Diagnosedaten befinden sich in Datensatz 0 (DS 0) und Datensatz 1 (DS 1). DS 0 besteht aus

4 Byte, die den aktuellen Zustand des Moduls beschreiben. Die Belegung dieser Byte entspricht der Belegung der Byte 8 ... 11 der Startinformationen des OB 82. DS 1 setzt sich aus den 4Byte des DS 0 und den modulspezifischen Diagnosedaten zusammen. Näheres zu den modulspezifischen Diagnosedaten finden Sie bei der Beschreibung des entsprechenden Moduls.

Lokaldaten

Informationen, wie Sie auf die Lokaldaten zugreifen können finden Sie in der Beschreibung zum OB 1. Nachfolgend sehen Sie die Belegung der Startinformationen für den OB 82, die defaultmäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB82_EV_CLASS	BYTE	Ereignisklasse und Kennung: 38h: gehendes Ereignis 39h: kommendes Ereignis
OB82_FLT_ID	BYTE	Fehlercode (42h)
OB82_PRIORITY	BYTE	Prioritätsklasse: parametrierbar über Hardware-Konfiguration
OB82_OB_NUMBR	BYTE	OB-NR. (82)
OB82_RESERVED_1	BYTE	reserviert
OB82_IO_FLAG	BYTE	Eingabe-Module 54h Ausgabe-Module 55h
OB82_MDL_ADDR	INT	Logische Basisadresse des Moduls, in dem der Fehler aufgetreten ist
OB82_MDL_DEFECT	BOOL	Modulstörung
OB82_INT_FAULT	BOOL	Interner Fehler
OB82_EXT_FAULT	BOOL	Externer Fehler
OB82_PNT_INFO	BOOL	Kanalfehler vorhanden
OB82_EXT_VOLTAGE	BOOL	Externe Hilfsspannung fehlt
OB82_FLD_CONNCTR	BOOL	Frontstecker fehlt
OB82_NO_CONFIG	BOOL	Modul nicht parametriert
OB82_CONFIG_ERR	BOOL	Falsche Parameter im Modul
OB82_MDL_TYPE	BYTE	Bit 3 ... 0: Modulklasse Bit 4: Kanalinformation vorhanden Bit 5: Anwenderinformation vorhanden Bit 6: Diagnosealarm von Stellvertreter Bit 7: reserviert
OB82_SUB_MDL_ERR	BOOL	Anwendermodul falsch/fehlt
OB82_COMM_FAULT	BOOL	Kommunikationsstörung
OB82_MDL_STOP	BOOL	Betriebszustand (0: RUN, 1:STOP)
OB82_WTCH_DOG_FLT	BOOL	Zeitüberwachung hat angesprochen
OB82_INT_PS_FLT	BOOL	Modulinterne Versorgungsspannung ausgefallen

Variable	Datentyp	Beschreibung
OB82_PRIM_BATT_FLT	BOOL	Batterie leer
OB82_BCKUP_BATT_FLT	BOOL	Gesamte Pufferung ausgefallen
OB82_RESERVED_2	BOOL	Reserviert
OB82_RACK_FLT	BOOL	Erweiterungsgeräteausfall
OB82_PROC_FLT	BOOL	Prozessorausfall
OB82_EPROM_FLT	BOOL	EPROM-Fehler
OB82_RAM_FLT	BOOL	RAM-Fehler
OB82_ADU_FLT	BOOL	ADU/DAU-Fehler
OB82_FUSE_FLT	BOOL	Sicherungsausfall
OB82_HW_INTR_FLT	BOOL	Prozessalarm verloren
OB82_RESERVED_3	BOOL	reserviert
OB82_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde

7.9.4 OB 83 - I/O_FLT2 - Ziehen / Stecken

Beschreibung

Das Betriebssystem der CPU ruft den OB 83 in folgenden Fällen auf:

- nach dem Ziehen oder Stecken eines projektierten Moduls
- nachdem Sie im Rahmen einer Anlagenänderung im laufenden Betrieb (CiR-Vorgang) Parameteränderungen eines Moduls vorgenommen und diese im RUN in die CPU geladen haben

Wenn Sie den OB 83 nicht programmiert haben, geht die CPU in den Betriebszustand STOP. Sie können den OB 83 mit Hilfe der SFCs 39 bis 42 sperren bzw. verzögern und wieder freigeben.

Ziehen/Stecken von Modulen

Jedes Ziehen und Stecken eines projektierten Moduls (nicht erlaubt sind: Netzteile, CPUs und Bus-Koppler) in den Betriebszuständen RUN, STOP und ANLAUF führt zu einem Ziehen/Stecken-Alarm. Dieser Alarm führt bei der zugehörigen CPU zu je einem Eintrag in den Diagnosepuffer und die Systemzustandsliste. Darüber hinaus wird im Betriebszustand RUN der Ziehen/Stecken-OB gestartet. Haben Sie diesen OB nicht programmiert, geht die CPU in den Betriebszustand STOP. Das Ziehen und Stecken von Modulen wird innerhalb des Systems sekündlich überwacht. Damit das Ziehen und Stecken von der CPU erkannt wird, muss zwischen dem Ziehen und Stecken eines Moduls eine Mindestzeit von zwei Sekunden liegen. Beim Ziehen eines projektierten Moduls im Betriebszustand RUN wird der OB 83 gestartet. Zuvor kann ein Zugriffsfehler beim Direktzugriff oder der Prozessabbildaktualisierung erkannt werden. Beim Stecken eines Moduls in einen projektierten Steckplatz im Zustand RUN überprüft das Betriebssystem, ob der Modultyp des gesteckten Moduls mit der Projektierung übereinstimmt. Anschließend wird der OB 83 gestartet, und bei Übereinstimmung der Modultypen erfolgt die Parametrierung.

Umparametrieren

Im Rahmen einer Anlagenänderung im laufenden Betrieb können Sie vorhandene Module umparametrieren. Die Umparametrierung erfolgt durch Übertragung der Parameter-Datensätze an die gewünschten Module. Der Ablauf ist wie folgt:

1. Nachdem Sie die Parameteränderungen eines Moduls vorgenommen und im RUN in die CPU geladen haben, wird der OB 83 gestartet (Startereignis: 3367h). Aus der OB-Startinformation relevant sind die logische Basisadresse (OB83_MDL_ADDR) und der Baugruppentyp (OB83_MDL_TYPE). Ab jetzt sind die Ein- bzw. Ausgangsdaten des Moduls evtl. nicht mehr korrekt, und es dürfen keine SFCs mehr aktiv sein, die Datensätze an dieses Modul senden.
2. Nach Beendigung des OB 83 erfolgt die Umparametrierung des Moduls.
3. Nach Beendigung des Umparametriervorgangs wird der OB 83 erneut gestartet.
   - Startereignis: 3267h, falls die Parametrierung erfolgreich war, bzw.
   - 3968h, falls sie nicht erfolgreich war

   Die Ein- bzw. Ausgangsdaten des Moduls verhalten sich wie nach einem Stecken-Alarm, d.h. sie sind zum jetzigen Zeitpunkt unter Umständen noch nicht korrekt. Sie dürfen ab sofort wieder SFCs aufrufen, die Datensätze an das Modul senden.

Lokaldaten

Nachfolgend sehen Sie die Belegung der Startinformationen für den OB 83, die defaultmäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB83_EV_CLASS	BYTE	Ereignisklasse und Kennungen: 32h: Ende Umparametrierung des Moduls 33h: Beginn Umparametrierung des Moduls 38h: Modul gesteckt 39h: Modul gezogen bzw. nicht ansprechbar bzw. Ende Umparametrierung
OB83_FLT_ID	BYTE	Fehlercode: (mögliche Werte: 51h, 54h ... 56h, 58h, 61, 63h ... 68h)
OB83_PRIORITY	BYTE	Prioritätsklasse: parametrierbar über Hardware-Konfiguration
OB83_OB_NUMBR	BYTE	OB-Nummer (83)
OB83_RESERVED_1	BYTE	Kennung für Modul bzw. Submodul/Schnittstellenmodul
OB83_MDL_ID	BYTE	54h: Peripheriebereich der Eingänge (PE) 55h: Peripheriebereich der Ausgänge (PA)
OB83_MDL_ADDR	WORD	■ zentral oder dezentral PROFIBUS-DP: – Logische Basisadresse des betroffenen Moduls, bei einem Mischmodul die kleinste verwendete logische Adresse des Moduls. – Sind die logischen E- und A-Adressen des Mischmoduls gleich, erhält die logische Basisadresse die E-Kennung. ■ dezentral PROFINET-IO: – Logische Basisadresse des Moduls/Submoduls

Variable	Datentyp	Beschreibung
OB83_RACK_NUM	WORD	■ Falls OB83_RESERVED_1 = A0h: Nr. des Submoduls/Schnittstellenmoduls (low byte) ■ Falls OB83_RESERVED_1 = C4h: – zentral: Nr. des Baugruppenträgers – dezentral PROFIBUS-DP: Nr. der DP-Station (low byte) und DP-Master-system-ID (high byte) – dezentral PROFINET-IO: physikalische Adresse: Kennungsbit (Bit 15, 1 = PROFINET-IO), IO-System-ID (Bit 11 ... 14) und Stationsnummer (Bit 0 ... 10)
OB83_MDL_TYPE	WORD	■ zentral oder dezentral PROFIBUS-DP: Modultyp des betroffenen Moduls (x: nicht anwenderrelevant) – x5xxh: Analogmodul – x8xxh: Funktionsmodul – xCxxh: CP – xFxxh: Digitalmodul – 8340h: Ersatztypkennung für Eingabemodul – 9340h: Ersatztypkennung für Ausgabemodul – A340h: Ersatztypkennung für Mischmodul (E/A) – F340h: Ersatztypkennung für nicht eindeutig identifizierbares Modul (z.B. bei gepackten Adressen) ■ dezentral PROFINET-IO: – 8101h: Modultyp des gesteckten Moduls ist gleich dem Modultyp des gezogenen Moduls – 8102h: Modultyp des gesteckten Moduls ist ungleich dem Modultyp des gezogenen Moduls
OB83_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde

OB83_EV_CLASS Ereignis, das den Start des OB 83 verursacht hat.

OB83_EV_CLASS	OB83_FLT_ID	Beschreibung
39h	51h	PROFINET-IO Modul gezogen
	54h	PROFINET-IO Submodul gezogen
38h	54h	PROFINET-IO Submodul gesteckt und entspricht parametriertem Sub-modul
	55h	PROFINET-IO Submodul gesteckt, entspricht aber nicht dem paramet-rierten Submodul
	56h	PROFINET-IO Submodul gesteckt, jedoch Fehler bei Parametrierung
	58h	PROFINET-IO Submodul Zugriffsfehler beseitigt
39h	61h	Modul gezogen bzw. nicht ansprechbar OB83_MDL_TYPE: Istmodultyp

OB83_EV_CLASS	OB83_FLT_ID	Beschreibung
38h	61h	Modul gesteckt, Modultyp in Ordnung OB83_MDL_TYPE: Istmodultyp
	63h	Modul gesteckt, jedoch falscher Modultyp OB83_MDL_TYPE: Istmodultyp
	64h	Modul gesteckt, jedoch gestört (Modulkennung nicht lesbar) OB83_MDL_TYPE: Sollmodultyp
	65h	Modul gesteckt, jedoch Fehler bei Modulparametrierung OB83_MDL_TYPE: Istmodultyp
39h	66h	Modul nicht ansprechbar, Lastspannungsfehler
38h	66h	Modul wieder ansprechbar, Lastspannungsfehler beseitigt
33h	67h	Beginn Umparametrierung eines Moduls
32h	67h	Ende Umparametrierung eines Moduls
39h	68h	Umparametrierung eines Moduls mit Fehler beendet

Falls Sie eine DP-V1- oder PROFINET-fähige CPU einsetzen, können Sie mit Hilfe des SFB 54 "RALRM" weitere Informationen über den Alarm erhalten, die über die Startinformation des OB hinausgehen.

7.9.5 OB 85 - OBNL_FLT - Programmablauffehler

Beschreibung

Das Betriebssystem der CPU ruft den OB 85 auf wenn:

- ein OB aufgerufen wird, der nicht in der CPU geladen wurde
- ein Fehler beim Zugriff des Betriebssystems auf einen Baustein auftritt
- ein Peripheriezugriffsfehler bei der systemseitigen Aktualisierung des Prozessabbilds auftritt

Sie können den OB 85 mit Hilfe des SFC 41 verzögern bzw. mit SFC 42 wieder freigeben.

Haben Sie den OB 85 nicht programmiert, geht die CPU in den Betriebszustand STOP.

Lokaldaten

Nachfolgend sehen Sie die Belegung der Startinformationen für den OB 85, die defaultmäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB85_EV_CLASS	BYTE	Ereignisklasse und Kennung: 35h 38h (nur bei Fehlercode B3h, B4h) 39h (nur bei Fehlercode B1h, B2h, B3h, B4h)
OB85_FLT_ID	BYTE	Fehlercode (mögliche Werte: A1h, A2h, A3h, A4h, B1h, B2h, B3h, B4h)
OB85_PRIORITY	BYTE	Prioritätsklasse: 26 (Defaultwert Betriebszustand RUN) 28 (Betriebszustand ANLAUF)
OB85_OB_NUMBR	BYTE	OB-NR. (85)
OB85_RESERVED_1	BYTE	reserviert
OB85_RESERVED_2	BYTE	reserviert
OB85_RESERVED_3	INT	reserviert
OB85_ERR_EV_CLASS	BYTE	Klasse des Ereignisses, das den Fehler ausgelöst hat.
OB85_ERR_EV_NUM	BYTE	Nummer des Ereignisses, das den Fehler ausgelöst hat.
OB85_OB_PRIOR	BYTE	Prioritätsklasse des OBs, der bearbeitet wurde, als der Fehler auftrat.
OB85_OB_NUM	BYTE	Nummer des OBs, der bearbeitet wurde, als der Fehler auftrat.
OB85_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde.

Informationen, wie Sie auf die Lokaldaten zugreifen können finden Sie in der Beschreibung zum OB 1.

OB 85 in Abhängigkeit von Fehlercodes

Falls Sie den OB 85 in Abhängigkeit von den möglichen Fehlercodes programmieren wollen, wird empfohlen die lokalen Variablen wie folgt zu organisieren:

Variable	Datentyp
OB85_EV_CLASS	BYTE
OB85_FLT_ID	BYTE
OB85_PRIORITY	BYTE
OB85_OB_NUMBR	BYTE
OB85_DKZ23	BYTE
OB85_RESERVED_2	BYTE
OB85_Z1	WORD
OB85_Z23	DWORD
OB85_DATE_TIME	DATE_AND_TIME

Die nachfolgende Tabelle beinhaltet das Ereignis, das den OB 85 aufgerufen hat.

OB85_EV_CLASS	OB85_FLT_ID	Variable	Bedeutung
35h	A1h, A2h		Ihr Programm oder das Betriebssystem erzeugt ein Startereignis für einen OB, der nicht in der CPU geladen ist.
	A1h, A2h	OB85_Z1	Die zugehörige temporäre Variable des angeforderten OBs. Dieser ist bestimmt durch OB85_Z23.
	A1h, A2h	OB85_Z23	High-Word: Klasse und Nummer des verursachenden Ereignisses
			Low-Word, High-Byte: Zum Fehlerzeitpunkt aktive Programmebene Low-Word, Low-Byte: Aktiver OB
35h	A3h		Fehler beim Zugriff des Betriebssystems auf einen Baustein
		OB85_Z1	Fehlerkennung des Betriebssystems
			High-Byte: 1: Integrierte Funktion 2: IEC-Timer
			Low-Byte: 0: keine Fehlerauflösung 1: Baustein nicht geladen 2: Bereichslängenfehler 3: Schreibschutzfehler
		OB85_Z23	High-Word: Bausteinnummer
			Low-Word: Relativadresse des Fehler verursachenden MC7-Befehls. Der Bausteintyp ist OB85_DKZ23 zu entnehmen. (88h: OB, 8Ch: FC, 8Eh: FB, 8Ah: DB)
35h	A4h		PROFINET-DB nicht ansprechbar
34h	A4h		PROFINET-DB wieder ansprechbar
39h	B1h		Peripheriezugriffsfehler beim Aktualisieren des Prozessabbilds der Eingänge.
	B2h		Peripheriezugriffsfehler beim Aktualisieren des Prozessabbilds der Ausgänge zu den Ausgabe-Modulen.

OB85_EV_CLASS	OB85_FLT_ID	Variable	Bedeutung
	B1h, B2h	OB85_DKZ23	Kennung für die Art des Prozessabbildtransfers bei dem der Peripheriezugriffsfehler auftrat: 10h: Bytezugriff 20h: Wortzugriff 30h: Doppelwortzugriff 57h: Übertragung eines projektierten Konsistenzbereichs
	B1h, B2h	OB85_Z1	reserviert für interne Verwendung der CPU: logische Basisadresse des Moduls Falls OB85_RESERVED_2 den Wert 76h hat, enthält OB85_Z1 den Rückgabewert der betroffenen SFC.
	B1h, B2h	OB85_Z23	Byte 0: Teilprozessabbild-Nummer
			Byte 1: Irrelevant, falls OB85_DKZ23=10, 20 oder 30 OB85_DKZ23=57: Länge des Konsistenzbereichs in Bytes
			Byte 2, 3: Nummer des PZF-verursachenden Peripheriebytes falls OB85_DKZ23=10, 20 oder 30 OB85_DKZ23=57: logische Anfangsadresse des Konsistenzbereichs
Sie erhalten die Fehlercodes B1h und B2h, falls Sie für die systemseitige Prozessabbildaktualisierung das wiederholte Melden von Peripheriezugriffsfehlern projektiert haben.			
38h, 39h	B3h		Peripheriezugriffsfehler beim Aktualisieren des Prozessabbilds der Eingänge kommend/gehend.
38h, 39h	B4h		Peripheriezugriffsfehler beim Aktualisieren des Prozessabbilds der Ausgänge zu den Ausgabe-Modulen kommend/gehend.
	B3h, B4h	OB85_DKZ23	Kennung für die Art des Prozessabbildtransfers bei dem der Peripheriezugriffsfehler auftrat: 10h: Bytezugriff 20h: Wortzugriff 30h: Doppelwortzugriff 57h: Übertragung eines projektierten Konsistenzbereichs
	B3h, B4h	OB85_Z1	reserviert für interne Verwendung der CPU: logische Basisadresse des Moduls Falls OB85_RESERVED_2 den Wert 76h hat, enthält OB85_Z1 den Rückgabewert der betroffenen SFC

OB85_EV_CLASS	OB85_FLT_ID	Variable	Bedeutung
	B3h, B4h	OB85_Z23	Byte 0: Teilprozessabbild-Nummer Byte 1: Irrelevant, falls OB85_DKZ23=10, 20 oder 30 OB85_DKZ23=57: Länge des Konsistenzbereichs in Bytes Byte 2, 3 Nummer des PZF-verursachenden Peripheriebytes falls OB85_DKZ23=10, 20 oder 30 OB85_DKZ23=57: logische Anfangsadresse des Konsistenzbereichs

Sie erhalten die Fehlercodes B3h und B4h, falls Sie für die systemseitige Prozessabbildaktualisierung das Melden von kommenden und gehenden Peripheriezugriffsfehlern projektiert haben. Nach einem Neustart werden bei der nächsten Prozessabbildaktualisierung alle Zugriffe auf nicht vorhandenen Ein- und Ausgänge als kommende Peripheriezugriffsfehler gemeldet.

7.9.6 OB 86 - RACK_FLT - Slaveausfall / -wiederkehr

Beschreibung

Das Betriebssystem der CPU ruft den OB 86 auf, wenn der Ausfall eines Slaves erkannt wird (sowohl bei kommendem als auch bei gehendem Ereignis).

Haben Sie den OB 86 nicht programmiert, geht die CPU in den Betriebszustand STOP.

Sie können den OB 86 mit Hilfe des SFC 41 verzögern bzw. mit SFC 42 wieder freigeben.

Lokaldaten

Nachfolgend sehen Sie die Belegung der Startinformationen für den OB 86, die defaultmäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB86_EV_CLASS	BYTE	Ereignisklasse und Kennung: 38h: gehendes Ereignis 39h: kommendes Ereignis
OB86_FLT_ID	BYTE	Fehlercode (mögliche Werte: C4h, C5h, C7h, C8h)
OB86_PRIORITY	BYTE	Prioritätsklasse: wird über den Hardware-Konfigurator eingestellt.
OB86_OB_NUMBR	BYTE	OB-NR. (86)
OB86_RESERVED_1	BYTE	reserviert

Variable	Datentyp	Beschreibung
OB86_RESERVED_2	BYTE	reserviert
OB86_MDL_ADDR	WORD	Abhängig vom Fehlercode
OB86_RACKS_FLTD	ARRAY (0 ... 31) OF BOOL	Abhängig vom Fehlercode
OB86_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde

Informationen, wie Sie auf die Lokaldaten zugreifen können finden Sie in der Beschreibung zum OB 1.

OB 86 in Abhängigkeit von Fehlercodes

Falls Sie den OB 86 in Abhängigkeit von den möglichen Fehlercodes programmieren wollen, wird empfohlen die lokalen Variablen wie folgt zu organisieren:

Variable	Datentyp
OB86_EV_CLASS	BYTE
OB86_FLT_ID	BYTE
OB86_PRIORITY	BYTE
OB86_OB_NUMBR	BYTE
OB86_RESERVED_1	BYTE
OB86_RESERVED_2	BYTE
OB86_MDL_ADDR	WORD
OB86_Z23	DWORD
OB86_DATE_TIME	DATE_AND_TIME

Die Variablen, deren Inhalt abhängig vom Fehlercode ist, haben folgende Bedeutung:

EV_CLASS	FLT_ID	Variable	Bit ...	Beschreibung
39h, 38h	C4h			Ausfall einer DP-Station
	C5h			Störung einer DP-Station
	C4h, C5h	OB86_MDL_ADDR		Logische Basisadresse des DP-Masters
		OB86_Z23		Adresse des betroffenen DP-Slaves:
			Bit 7 ... 0	Nummer der DP-Station
			Bit 15 ... 8	DP-Mastersystem-ID
			Bit 30 ... 16	Logische Basisadresse des DP-Slave
			Bit 31	I/O Kennung
38h	C7h			Wiederkehr einer DP-Station, jedoch Fehler bei der Modulparametrierung

EV_CLASS	FLT_ID	Variable	Bit ...	Beschreibung
		OB86_MDL_ADDR		Logische Basisadresse des DP-Masters
		OB86_Z23		Adresse des betroffenen DP-Slaves:
			Bit 7 ... 0	Nummer der DP-Station
			Bit 15 ... 8	DP-Mastersystem-ID
			Bit 30 ... 16	Logische Basisadresse des DP-Slave
			Bit 31	I/O Kennung
	C8h			Wiederkehr einer DP-Station, jedoch Abweichung Soll-/ Istausbau
		OB86_MDL_ADDR		Logische Basisadresse des DP-Masters
		OB86_Z23		Adresse des betroffenen DP-Slaves:
			Bit 7 ... 0	Nummer der DP-Station
			Bit 15 ... 8	DP-Mastersystem-ID
			Bit 30 ... 16	Logische Basisadresse des DP-Slave
			Bit 31	I/O Kennung

7.10 Synchronalarme - *"Synchronous Interrupts"*

7.10.1 OB 121 - PROG_ERR - Programmierfehler

Beschreibung

Das Betriebssystem der CPU generiert ein Fehler-Ereignis, wenn in unmittelbarem Zusammenhang mit der Programmbearbeitung ein Fehler auftritt. Ist der OB 121 nicht programmiert, wechselt die CPU in den Betriebszustand STOP.

Wenn Sie beispielsweise in Ihrem Programm einen Baustein aufrufen, der nicht in die CPU geladen wurde, dann wird der OB 121 aufgerufen.

Der OB 121 läuft in derselben Prioritätsklasse wie der unterbrochene Baustein. Sie können deshalb innerhalb des OB 121 auf die Register des unterbrochenen Bausteins zugreifen und diese ggf. mit geänderten Parametern an den OB zurückgeben.

Startereignisse maskieren

Die CPU verfügt über die folgenden SFCs, mit denen Sie Startereignisse des OB 121 maskieren und demaskieren können, während ihr Programm bearbeitet wird:

- Die SFC 36 MSK_FLT maskiert bestimmte Fehlercodes.
- Die SFC 37 DMSK_FLT demaskiert die Fehlercodes, die von der SFC 36 maskiert wurden.
- Die SFC 38 READ_ERR liest das Ereignisstatusregister.

Lokaldaten

Nachfolgend sehen Sie die Belegung der Startinformationen für den OB 121, die default-mäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB121_EV_CLASS	BYTE	Ereignisklasse und Kennung: 25h
OB121_SW_FLT	BYTE	Fehlercode
OB121_PRIORITY	BYTE	Prioritätsklasse: Prioritätsklasse des OBs, in dem der Fehler aufgetreten ist.
OB121_OB_NUMBR	BYTE	OB-Nr. (121)
OB121_BLK_TYPE	BYTE	Art des Bausteins, in dem der Fehler aufgetreten ist 88h: OB, 8Ah: DB, 8Ch: FC, 8Eh: FB
OB121_RESEVED_1	BYTE	reserviert (Datenbereich und die Zugriffsart)
OB121_FLT_REG	WORD	Fehlerquelle (abhängig vom Fehlercode), z.B.: ■ Register, in dem der Konvertierungsfehler aufgetreten ist. ■ Fehlerhafte Adresse (Lese-/Schreibfehler) ■ Fehlerhafte Nummer eines Timers, eines Zählers oder eines Bausteins. ■ Fehlerhafter Speicherbereich
OB121_BLK_NUM	WORD	Nummer des Bausteins mit dem fehlerverursachenden Befehl.
OB121_PRG_ADDR	WORD	Relativadresse des fehlerverursachenden Befehls.
OB121_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde.

Informationen, wie Sie auf die Lokaldaten zugreifen können finden Sie in der Beschreibung zum OB 1.

Fehlercodes

Die vom Fehlercode abhängigen Variablen haben folgende Bedeutung:

Fehlercode	Variable	Beschreibung
21h	OB121_FLT_REG:	BCD-Konvertierungsfehler
		Kennung für das betroffene Register (0000h: Akku 1)
22h	OB121_RESERVED_1	Bereichslängenfehler beim Lesen
23h		Bereichslängenfehler beim Schreiben
28h		lesender Zugriff auf ein Byte, Wort oder Doppelwort mit einem Pointer, dessen Bitadresse ungleich 0 ist.
29h		schreibender Zugriff auf ein Byte, Wort oder Doppelwort mit einem Pointer, dessen Bitadresse ungleich 0 ist.
		fehlerhafte Byteadresse Der Datenbereich und die Zugriffsart sind OB121_RESERVED_1 zu entnehmen

Fehlercode	Variable	Beschreibung
		Bit 3 ... 0 Speicherbereich: 0: Peripheriebereich 1: Prozessabbild der Eingänge 2: Prozessabbild der Ausgänge 3: Merker 4: Global-DB 5: Instanz-DB 6: Eigene Lokaldaten 7: Lokaldaten des Aufrufers Bit 7 ... 4 Zugriffsart: 0: Bitzugriff 1: Bytezugriff 2: Wortzugriff 3: Doppelwortzugriff
24h	OB121_FLT_REG	Bereichsfehler beim Lesen
25h		Bereichsfehler beim Schreiben
		enthält im Low-Byte die Kennung des unzulässigen Bereichs (86h eigener Lokaldatenbereich)
26h	OB121_FLT_REG	Fehler bei Timernummer
27h		Fehler bei Zählernummer
		unzulässige Nummer
30h	OB121_FLT_REG	Schreibender Zugriff auf einen schreibgeschützten Global-DB
31h		Schreibender Zugriff auf einen schreibgeschützten Instanz-DB
32h		DB-Nummernfehler beim Zugriff auf einen Global-DB
33h		DB-Nummernfehler beim Zugriff auf einen Instanz-DB
		unzulässige DB-Nummer
34h	OB121_FLT_REG	Nummernfehler beim FC-Aufruf
35h		FB-Nummernfehler beim FB-Aufruf
3Ah		Zugriff auf einen nicht geladenen DB; die DB-Nummer liegt im zulässigen Bereich
3Ch		Zugriff auf einen nicht geladenen FC; die FC-Nummer liegt im zulässigen Bereich
3Dh		Zugriff auf eine nicht geladenen SFC; die SFC-Nummer liegt im zulässigen Bereich
3Eh		Zugriff auf einen nicht geladenen FB; die FB-Nummer liegt im zulässigen Bereich
3Fh		Zugriff auf einen nicht geladenen SFB; die SFB-Nummer liegt im zulässigen Bereich
		unzulässige Nummer

7.10.2 OB 122 - MOD_ERR - Peripheriezugriffsfehler

Beschreibung

Das Betriebssystem der CPU ruft den OB 122 auf, wenn beim Zugreifen auf Daten eines Moduls ein Fehler auftritt. Wenn die CPU beispielsweise einen Lesefehler beim Zugriff auf Daten eines I/O-Moduls erkennt, dann ruft das Betriebssystem den OB 122 auf. Ist der OB 122 nicht programmiert, wechselt die CPU in den Betriebszustand STOP.

Der OB 122 läuft in derselben Prioritätsklasse wie der unterbrochene Baustein. Sie können deshalb innerhalb des OB 122 auf die Register des unterbrochenen Bausteins zugreifen und diese ggf. mit geänderten Parametern an den OB zurückgeben.

Startereignisse maskieren

Die CPU verfügt über die folgenden SFCs, mit denen Sie Startereignisse des OB 122 maskieren und demaskieren können, während ihr Programm bearbeitet wird:

- Die SFC 36 MSK_FLT maskiert bestimmte Fehlercodes.
- Die SFC 37 DMSK_FLT demaskiert die Fehlercodes, die von der SFC 36 maskiert wurden.
- Die SFC 38 READ_ERR liest das Ereignisstatusregister.

Lokaldaten

Nachfolgend sehen Sie die Belegung der Startinformationen für den OB 122, die default-mäßige symbolische Bezeichnung und die Datentypen:

Variable	Datentyp	Beschreibung
OB122_EV_CLASS	BYTE	Ereignisklasse und Kennung: 29h
OB122_SW_FLT	BYTE	Fehlercode: 42h: Peripheriezugriffsfehler - lesend 43h: Peripheriezugriffsfehler - schreibend
OB122_PRIORITY	BYTE	Prioritätsklasse: Prioritätsklasse des OBs, der Fehler verursachte
OB122_OB_NUMBR	BYTE	OB-NR. (122)
OB122_BLK_TYPE	BYTE	hier wird kein gültiger Wert eingetragen
OB122_MEM_AREA	BYTE	Speicherbereich und Zugriffsart: Bit 3 ... 0: Speicherbereich 0: Peripheriebereich 1: Prozessabbild der Eingänge 2: Prozessabbild der Ausgänge Bit 7 ... 4: Zugriffsart: 0: Bitzugriff, 1: Bytezugriff, 2: Wortzugriff, 3: Doppelwortzugriff
OB122_MEM_ADDR	WORD	Adresse des Fehlers im Speicher
OB122_BLK_NUM	WORD	hier wird kein gültiger Wert eingetragen
OB122_PGR_ADDR	WORD	hier wird kein gültiger Wert eingetragen
OB122_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde

Informationen, wie Sie auf die Lokaldaten zugreifen können finden Sie in der Beschreibung zum OB 1.

7.11 Taktsynchronalarme - *"Cycle synchronous Interrupts"*

7.11.1 OB 60 - MULTI_INT - Multicomputingalarm

Beschreibung

Durch Aktivierung der Funktion "Motion Control" im SPEED7 Studio wird der OB 60 automatisch angelegt. Der OB wird intern verwendet und kann nicht bearbeitet werden. Er dient der Verwaltung der Service-Daten-Objekten (SDO) und Diagnosedaten. Der OB 60 besitzt eine höhere Priorität als der OB 1. Die Zykluszeit für diesen OB können Sie im SPEED7 Studio vorgeben.

7.11.2 OB 61 - SYNC_1 - Taktsynchronalarm

Beschreibung

Durch Aktivierung der Funktion "Motion Control" im SPEED7 Studio wird der OB 61 automatisch angelegt. Innerhalb des OB 61 sind die Funktionen abzulegen, welche synchron auszuführen sind. Für den OB wird ein gesondertes Prozessabbild PA OB 61 angelegt, dessen Daten während der Abarbeitung des OBs konsistent sind. OB 61 besitzt eine höhere Priorität als OB 60.

Lokaldaten

Nachfolgend sehen Sie die temporären (TEMP) Variablen des Taktsynchronalarm-OB. Als Variablennamen wurden die Defaultnamen des OB 61 gewählt.

Variable	Datentyp	Beschreibung
OB61_EV_CLASS	BYTE	Ereignisklasse und Kennungen: B#16#11; Alarm ist aktiv
OB61_STRT_INF	BYTE	B#16#64: Startanforderung für OB 61
OB61_PRIORITY	BYTE	Parametrierte Prioritätsklasse; Defaultwert: 28
OB61_OB_NUMBR	BYTE	OB-Nummer 61
OB61_LAST_EXEC_TIME	WORD	Ausführungszeit des vorherigen OB 61 (µs) seit dem letzten Anlauf. Beim ersten Anlauf ist der Wert 0. Bei Werten die den Speicherbereich überschreiten würden ist der Wert 0xFFFF (65535).
OB61_RESERVED_1	BOOL	Reserviert
OB61_FIRST	BOOL	Erste Ausführung nach Anlauf bzw. Haltzustand
OB61_CYCLE_ERROR	BOOL	Ausführungszeitverletzung: 0: Ausführungszeit beim letzten OB 61 war kleiner als X% der Zykluszeit. 1: OB 61 hat länger als X% der Zykluszeit gedauert. X ist zwischen 0 und 100% konfigurierbar.
OB61_READ_PI_ERROR	BOOL	Beim Einlesen der OB61-Prozessabbild-Eingänge ist ein Fehler aufgetreten.
OB61_WRITE_PI_ERROR	BOOL	Beim Schreiben der OB61-Prozessabbild-Ausgänge ist beim vorangegangenen OB 61 ein Fehler aufgetreten.
OB61_DC_STATUS_MASTER	BOOL	Status des DistributedClocks-Masters: 0=InSync / 1=OutOfSync

Variable	Datentyp	Beschreibung
OB61_DC_STATUS_SLAVE	BOOL	Status des DistributedClocks-Slave: 0=InSync / 1=OutOfSync
OB61_MISSED_EXEC	BYTE	Anzahl der ausgefallenen OB 61-Starts seit der letzten OB 61-Ausführung. ■ Bit 0: 0 ... 127 ■ Bit 7: Anzeige eines Überlaufs, es sind mehr als 127 OB 61 Starts ausgefallen.
OB61_MIN_EXEC_TIME	WORD	Minimale Ausführungszeit (µs) seit dem letzten Anlauf. Beim ersten Anlauf ist der Wert 0. Bei Werten die den Speicherbereich überschreiten würden ist der Wert 0xFFFF (65535).
OB61_MAX_EXEC_TIME	WORD	Maximale Ausführungszeit (µs) seit dem letzten Anlauf. Beim ersten Anlauf ist der Wert 0. Bei Werten die den Speicherbereich überschreiten würden ist der Wertf 0xFFFF (65535).
OB61_DATE_TIME	DATE_AND_TIME	Datum und Uhrzeit, zu denen der OB angefordert wurde.

8 Gebäude-Automatisierung - *"Building Control"*

8.1 Übersicht

In diesem Kapitel finden Sie die Funktionsbausteine (FB45 ... FB50) für die Gebäudeleittechnik (GLT). Die Bausteine verwenden die Systemzeit der CPU. Es sind keine S7-Timer erforderlich. Sie haben die Möglichkeit für jeden Baustein einen Instanz-Datenbaustein oder Multiinstanzen zu verwenden. Es gibt folgende Bausteine:

FB		Beschreibung
FB 45	LAMP	Steuern einer Leuchte oder Steckdose
FB 46	BLIND	Steuern einer Jalousie
FB 47	DSTRIKE	Steuern eines elektrischen Türöffners
FB 48	ACONTROL	Zutrittsteuerung
FB 49	KEYPAD	Abfrage eines Tastenfelds mit externer Spannungsversorgung
FB 50	KEYPAD2	Abfrage eines Tastenfelds ohne externe Spannungsversorgung

8.1.1 Aufruf-Beispiel - Instanz-DB

Netzwerk 1

```
CALL "Deckenleuchte", DB 1
  ON       :=M20.0
  OFF      :=20.1
  ONOFF    :=20.2
  Duration :=T#5M
  Output   :=M20.3
  PulseOn  :=
  PulseOff :=
```

8.1.2 Aufruf-Beispiel - Multiinstanzen-DB

Inhalt von: "Umgebung \Schnittstelle\Stat"

Nachfolgend sehen Sie einen Beispielaufruf für mehrere Leuchten und eine Jalousie in AWL mit Multiinstanzen.

Name	Datentyp	Adresse
Deckenleuchte	LAMP	0.0
Stehleuchte	LAMP	46.0
Spiegelleuchte	LAMP	92.0
Jalousie	BLIND	138.0

Netzwerk 1

```
CALL #Deckenleuchte
  ON       :=M20.0
  OFF      :=20.1
  ONOFF    :=20.2
```

```
  Duration :=T#5M
  Output   :=M20.3
  PulseOn  :=
  PulseOff :=
```

Netzwerk 2

```
CALL #Jalousie
  Up              :=M30.0
  Down            :=M30.1
  CentralUp       :=
  CentralDown     :=
  TimeMaxDuration :=T#10S
  TimePause       :=T#1S
  TimeShortLong   :=T#2S
  Endable         :=
  BlindUp         :=M30.6
  BlindDown       :=M30.7
```

8.2 Raumsteuerung - *"Room"*

8.2.1 FB 45 - LAMP - Leuchte / Steckdose steuern

Beschreibung

Mit diesem Baustein können Sie Lastrelais für Leuchten und Steckdosen ansteuern. Die Ansteuerung kann über einen Ein/Aus-Taster oder über getrennte Ein- und Aus-Taster erfolgen. Zusätzlich haben Sie die Möglichkeit über *Duration* eine Zeit für das automatische Abschalten vorzugeben. Über *TimeDebounce* können Sie für die Eingangssignale eine Entprellzeit vorgeben.

- Bei Ansteuerung eines monostabilen Relais bleibt ein Ausgang solange gesetzt, solange das Relais aktiv sein soll. Mit einem Flankenwechsel 0-1 an *OnOff* bzw. *On* wird der statische Ausgang *Output* gesetzt. Dieser bleibt solange gesetzt, bis Sie diesen mit Flankenwechsel 0-1 an *OnOff* bzw. *Off* wieder zurücksetzen oder die über *Duration* vorgegebene Zeit abgelaufen ist.
- Bei Ansteuerung eines bistabilen Relais werden 2 Ausgänge verwendet. Hierbei steuert *PulseOn* den Einschaltvorgang und *PulseOff* den Ausschaltvorgang. Über *TimePulse* geben Sie die Impulsdauer und über *TimePause* die Umschaltzeit der beiden Ausgänge vor.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
OnOff	INPUT	BOOL	Mit Flankenwechsel 0-1 wird der Ausgang *Output* aktiviert bzw. deaktiviert und *PulseOn* oder *PulseOff* aktiviert. Default: FALSE
On	INPUT	BOOL	Mit Flankenwechsel 0-1 wird der Ausgang *Output* bzw. *PulseOn* aktiviert. Default: FALSE
Off	INPUT	BOOL	Mit Flankenwechsel 0-1 wird der Ausgang *Output* deaktiviert und *PulseOff* aktiviert. Default: FALSE

Parameter	Deklaration	Datentyp	Beschreibung
Duration	INPUT	TIME	Zeit für die Dauer, nach der der Ausgang *Output* deaktiviert bzw. *PulseOff* aktiviert wird. Mit Angabe von 0ms ist das automatische Abschalten deaktiviert. Default: 0ms
Output	OUTPUT	BOOL	Statischer Ausgang zur Ansteuerung eines monostabilen Relais.
PulseOn	OUTPUT	BOOL	Impulsausgang zur Ansteuerung eines bistabilen Relais (Ein-Signal).
PulseOff	OUTPUT	BOOL	Impulsausgang zur Ansteuerung eines bistabilen Relais (Aus-Signal).
TimeDebounce	CONSTANT	TIME	Zeitvorgabe zum Entprellen der Eingänge. Default: 100ms
TimePulse	CONSTANT	TIME	Zeitvorgabe für die Impulsdauer von *PulseOn* bzw. *PulseOff*. Default: 100ms
TimePause	CONSTANT	TIME	Zeitvorgabe für die Pause zwischen dem Rücksetzen und Setzen von *PulseOn* bzw. *PulseOff*. Default: 100ms

8.2.2 FB 46 - BLIND - Jalousie steuern

Beschreibung

Mit diesem Baustein können Sie eine motorisch angetriebene Jalousie ansteuern. Hierzu müssen Sie den Antrieb mit *Enable* freigeben.

- Die Ansteuerung für "Heben" *BlindUp* und "Senken" *BlindDown* hat über 2 Taster (*Up*/*Down* bzw. *CentralUp*/*CentralDown*) zu erfolgen.
  - *CentralUp*/*CentralDown*: Dient zur zentralen Ansteuerung aller Jalousien eines Gebäudes
  - *Up*/*Down*: Dient zur lokalen Ansteuerung einer Jalousie. Hier wird ein anstehendes *CentralUp*/*CentralDown*-Signal ignoriert.
- Wird der entsprechende Taster länger als die Zeit *TimeShortLong* gedrückt, fährt der Jalousie-Motor in die entsprechende Endlage. Durch Tippen eines der beiden Taster (*Up/Down* bzw. *CentralUp/CentralDown*) können Sie die Bewegung stoppen und diese gegebenenfalls umkehren.
- Mit *TimeMaxDuration* geben Sie die Maximal Laufzeit des Motors vor und mit *TimePause* die Pause für den Richtungswechsel.
- Durch Tippen fährt der Jalousie-Motor nur kurz an. Mit dieser Funktion können Sie die Jalousie-Lamellen fein justieren.
- Über *TimeDebounce* können Sie für die Eingangssignale eine Entprellzeit vorgeben.
- Über Status können Sie die Stellung der Jalousie abfragen
  - 0: Endposition oben
  - 50: Unbestimmte Position zwischen den beiden Endpositionen
  - 100: Endposition unten

VORSICHT!
Der Jalousie-Motor muss über eigene Endlagenschalter verfügen, die diesen selbsttätig abschalten!

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Up	INPUT	BOOL	Mit Flankenwechsel 0-1 wird der Ausgang *BlindUp* aktiviert. Abhängig vom Eingangs-Signal fährt die Jalousie in die Endlage oben oder wird nur kurz bewegt. Solange dieses Signal ansteht werden die Signale *CentralUp/CentralDown* ignoriert. Default: FALSE
Down	INPUT	BOOL	Mit Flankenwechsel 0-1 wird der Ausgang *Blind-Down* aktiviert. Abhängig vom Eingangs-Signal fährt die Jalousie in die Endlage unten oder wird nur kurz bewegt. Solange dieses Signal ansteht werden die Signale *CentralUp/CentralDown* ignoriert. Default: FALSE
CentralUp	INPUT	BOOL	Mit Flankenwechsel 0-1 wird der Ausgang *BlindUp* aktiviert. Hierbei fährt die Jalousie in die Endlage oben. Default: FALSE
CentralDown	INPUT	BOOL	Mit Flankenwechsel 0-1 wird der Ausgang *Blind-Down* aktiviert. Hierbei fährt die Jalousie in die Endlage unten. Default: FALSE
TimeMaxDuration	INPUT	TIME	Maximale Laufzeit des Motors für die Fahrt in die jeweilige Endlage. Default: 30s
TimePause	INPUT	TIME	Zeitvorgabe für die Pause für den Richtungs-wechsel. Default: 2s
TimeShortLong	INPUT	TIME	Zeitvorgabe für die Unterscheidung zwischen Tipp- und Dauer-Betrieb. Default: 1s
Enable	INPUT	BOOL	Freigabe für den Antrieb (statisch) Default: TRUE
BlindUp	OUTPUT	BOOL	Statischer Ausgang Jalousie "heben".
BlindDown	OUTPUT	BOOL	Statischer Ausgang Jalousie "senken".
Status	OUTPUT	INT	■ Status - Position der Jalousie – 0: Endposition oben – 50: Unbestimmte Position zwischen den beiden Endpositionen – 100: Endposition unten
TimeDebounce	CONSTANT	TIME	Zeitvorgabe zum Entprellen der Eingänge. Default: 100ms

8.2.3 FB 47 - DSTRIKE - Elektrischer Türöffner

Beschreibung

Mit diesem Baustein können Sie einen elektrischen Türöffner ansteuern, sofern dieser nicht mit *DoorIsLocked* "verriegelt" ist.

- Mit einem Flankenwechsel 0-1 am Eingang *Open* wird für die Zeit *"TimeOpening"* der Ausgang *"Output"* angesteuert.
- Mit einem Flankenwechsel 0-1 des Eingangsignals *EnableAlwaysOpen* bzw. *DisableAlwaysOpen* wird der Ausgang *Open* dauerhaft aktiviert bzw. deaktiviert. Zusätzlich wird bei gesetztem *EnableAlwaysOpen* der statische Ausgang *AlwaysOpen* gesetzt.
- An die Eingangsignale *DoorIsClosed* und *DoorIsLocked* können Sie Ihre Tür-Kontakte anbinden. *DoorIsClosed* wird gesetzt, sobald Ihre Türe geschlossen ist. *DoorIsLocked* ist bei verriegelter Tür aktiv, d.h. der Kontakt wird über den Schließmechanismus ausgelöst und das Öffnen mittels Türöffner ist nicht möglich.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Open	INPUT	BOOL	Mit Flankenwechsel 0-1 wird der Ausgang *Output* für die Zeit *TimeOpening* aktiviert. Default: FALSE
EnableAlwaysOpen	INPUT	BOOL	Mit Flankenwechsel 0-1 wird der Ausgang *Output* dauerhaft angesteuert. Default: FALSE
DisableAlwaysOpen	INPUT	BOOL	Mit Flankenwechsel 0-1 wird der Ausgang *Output* dauerhaft deaktiviert. Default: FALSE
TimeOpening	INPUT	TIME	Zeit für die Dauer der Ansteuerung von *Output*. Default: 3s
DoorIsClosed	INPUT	BOOL	■ Optional - Position der Tür TRUE: Tür ist geschlossen FALSE: Tür ist geöffnet Default: FALSE
DoorIsLocked	INPUT	BOOL	■ Optional - Schließzustand der Tür – TRUE: Tür ist verriegelt – FALSE: Tür ist nicht verriegelt Default: FALSE
Output	OUTPUT	BOOL	Statischer Ausgang zur Ansteuerung eines monostabilen Relais.
AlwaysOpen	OUTPUT	BOOL	Statischer Ausgang für Anzeige "Tür ist dauerhaft geöffnet".

8.3 Zugangskontrolle - *"Access Control"*

8.3.1 FB 48 - ACONTROL - Zutrittssteuerung

Beschreibung

Mit diesem Baustein können Sie eine Zugangssteuerung realisieren. Nach Vorgabe eines Codes von einem externen Tastenfeld, Panel oder RFID-Lesers wird dieser Code mit einer Liste verglichen. Je nach Ergebnis werden daraufhin entsprechende Ausgänge angesteuert.

- Die Zugangscodes sind in einem Datenbaustein anzulegen, welchen Sie über *ACL-Block* angeben. Hier bestimmen Sie auch welche Ausgänge *Access1...6* angesteuert werden und wie (Impuls/statisch) diese angesteuert werden sollen. Mit dem Datenbaustein können Sie bis zu 16 Zugangscodes verwalten.
- Über *AccessCode1...4* geben Sie den Code des entsprechenden Eingabegeräts vor.
- Mit *CheckCode1...4* wird der Code mit den Codes in Ihrem Datenbaustein *ACLBlock* verglichen.
  - Ist der Zugangscode im Datenbaustein vorhanden, werden die entsprechenden Ausgänge nach Vorgabe angesteuert. Bei projektierter Impuls-Ausgabe können Sie über *TimePulse* die Impulsdauer vorgeben.
  - Ist der Zugangscode im Datenbaustein nicht vorhanden, wird für die Zeit *TimeError* der Ausgang *Error* gesetzt.
- Mit einem Flankenwechsel 0-1 an *CentralLock* werden alle Zugangscodes deaktiviert. Hierbei wird der Ausgang *CentralLocked* gesetzt.
- Mit einem Flankenwechsel 0-1 an *CentralUnlock* werden alle Zugangscodes aktiviert und der Ausgang *CentralLocked* zurückgesetzt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
AccessCode1	INPUT	STRING[16]	Zutrittscode, z.B. von Keypad.
CheckCode1	INPUT	BOOL	Mit Flankenwechsel 0-1 wird der *AccessCode1* mit dem Zugangscode im Datenbaustein *ACL-Block* verglichen. Default: 0
AccessCode2	INPUT	STRING[16]	Zutrittscode, z.B. von Panel
CheckCode2	INPUT	BOOL	Mit Flankenwechsel 0-1 wird der *AccessCode2* mit dem Zugangscode im Datenbaustein *ACL-Block* verglichen. Default: 0
AccessCode3	INPUT	STRING[16]	Zutrittscode, z.B. von RFID-Lesegerät
CheckCode3	INPUT	BOOL	Mit Flankenwechsel 0-1 wird der *AccessCode3* mit dem Zugangscode im Datenbaustein *ACL-Block* verglichen. Default: 0
AccessCode4	INPUT	STRING[16]	Zutrittscode, z.B. von sonstigem System
CheckCode4	INPUT	BOOL	Mit Flankenwechsel 0-1 wird der *AccessCode4* mit dem Zugangscode im Datenbaustein *ACL-Block* verglichen. Default: 0
CentralLock	INPUT	BOOL	Mit Flankenwechsel 0-1 werden alle Zugangscodes deaktiviert. Hierbei wird der Ausgang *CentralLocked* gesetzt.

Parameter	Deklaration	Datentyp	Beschreibung
CentralUnlock	INPUT	BOOL	Mit Flankenwechsel 0-1 an *CentralUnlock* werden alle Zugangscodes aktiviert und der Ausgang *CentralLocked* zurückgesetzt.
ACLBlock	INPUT	BLOCK	Datenbaustein, in dem die Zugangscodes hinterlegt sind vom Typ UDT 4 - ACL. ↪ *Kap. 8.3.3 "UDT 4 - ACL - Datenstruktur für FB 48" Seite 312*
Access1	OUTPUT	BOOL	Ausgang 1, kann als Impuls oder statisch angesteuert werden.
Access2	OUTPUT	BOOL	Ausgang 2, kann als Impuls oder statisch angesteuert werden.
Access3	OUTPUT	BOOL	Ausgang 3, kann als Impuls oder statisch angesteuert werden.
Access4	OUTPUT	BOOL	Ausgang 4, kann als Impuls oder statisch angesteuert werden.
Access5	OUTPUT	BOOL	Ausgang 5, kann als Impuls oder statisch angesteuert werden.
Access6	OUTPUT	BOOL	Ausgang 6, kann als Impuls oder statisch angesteuert werden.
Error	OUTPUT	BOOL	Ist der Zugangscode im Datenbaustein nicht vorhanden, wird für die Zeit *TimeError* der Ausgang *Error* gesetzt.
CentralLocked	OUTPUT	BOOL	■ Zugang – TRUE: verriegelt - Zugang nicht möglich – FALSE: nicht verriegelt - Zugang möglich Default: TRUE
TimePulse	CONSTANT	Time	Zeit für die Impulsdauer an einem Ausgang. Default: 3s
TimeError	CONSTANT	Time	Zeit für die Dauer des Error-Signals. Default: 500ms

8.3.2 UDT 3 - ACLREC - Datenstruktur für FB 48

Beschreibung

Adresse	Name	Typ	Anfangswert	Kommentar
0.0		**STRUCT**		
+0.0	Code	STRING[16]	' '	Byte 0 ... 17: Zugangscode S7String mit max. 16 ASCII-Zeichen für Zugriffscode
+18.0	EnableOutput1	BOOL	FALSE	Byte 18: Signal für anzusteuernde Ausgänge TRUE: aktiviere Ausgang, FALSE: deaktiviere Ausgang
+18.1	EnableOutput2	BOOL	FALSE	
+18.2	EnableOutput3	BOOL	FALSE	

Adresse	Name	Typ	Anfangswert	Kommentar
0.0		**STRUCT**		
+18.3	EnableOutput4	BOOL	FALSE	
+18.4	EnableOutput5	BOOL	FALSE	
+18.5	EnableOutput6	BOOL	FALSE	
+18.6	EnableRes7	BOOL	FALSE	
+18.7	EnableRes8	BOOL	FALSE	
+19.0	SignalOutput1	BOOL	FALSE	Byte 19: Signaltyp FALSE: Impuls, TRUE: statische 1, Deaktivierung mit weiterem Code
+19.1	SignalOutput2	BOOL	FALSE	
+19.2	SignalOutput3	BOOL	FALSE	
+19.3	SignalOutput4	BOOL	FALSE	
+19.4	SignalOutput5	BOOL	FALSE	
+19.5	SignalOutput6	BOOL	FALSE	
+19.6	SignalRes7	BOOL	FALSE	
+19.7	SignalRes8	BOOL	FALSE	
=20.0				

8.3.3 UDT 4 - ACL - Datenstruktur für FB 48

Beschreibung

Adresse	Name	Typ	Anfangswert	Kommentar
0.0		**STRUCT**		
+0.0	RecordCount	INT	16	DBW0: Anzahl gültiger Datensätze (0 ... n)
+2.0	RecordLen	INT	20	DBW2: Länge eines Datensatzes in Bytes (20)
+4.0	Record	ARRAY[0...15]		Ab DBB4 beginnt der erste Datensatz
*20.0		"UDT 3 - ACLREC"		*Kap. 8.3.2 "UDT 3 - ACLREC - Datenstruktur für FB 48" Seite 311*
=324.0		BOOL		

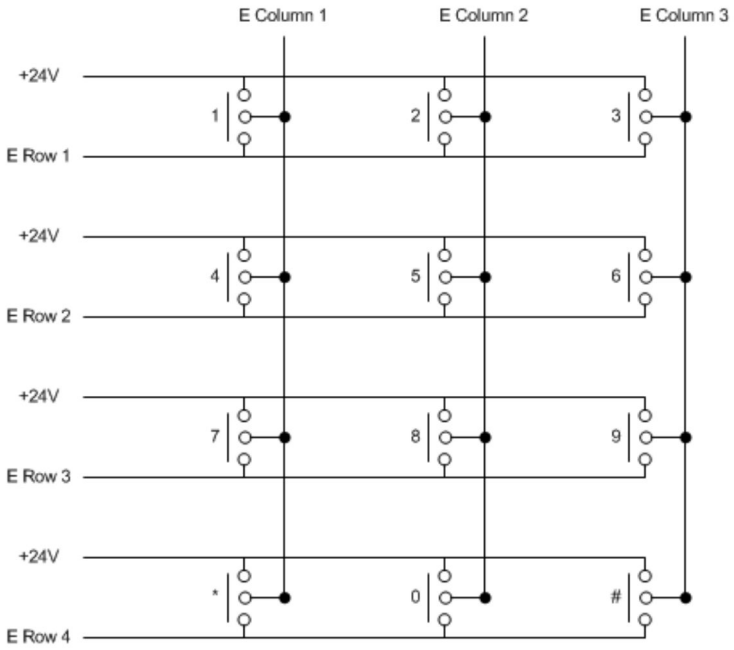
VORSICHT!
Ein Code darf nur 1 x in der ganzen Liste vorkommen. Doppelte Codes sind unzulässig.

8.3.4 FB 49 - KEYPAD - Tastatur

Beschreibung

Dieser Baustein dient zur Anbindung eines externen Tastenfelds (0...9,*,#) mit externer DC 24V Spannungsversorgung. Abhängig von der betätigten Taste liefert die Tastatur Zeilen und Spaltensignale (24V). Der Baustein wertet die Signale intern mittels einer Bitmuster-Tabelle aus und übergibt den ermittelten ASCII-Code in den Tastaturpuffer. Bei Bedarf oder automatisch wird der Tastaturpuffer als max. 16Byte lange Zeichenkette ausgegeben.

- Über *Row 1...4* werden die Zeilen 1...4 der Tastaturmatrix angebunden.
- Über *Column 1...3* werden die Spalten 1...3 der Tastaturmatrix angebunden.
- Über *ClearCode* können Sie einen Tastencode vorgeben, mit dem der Eingabepuffer gelöscht wird.
- Über *EnterCode* können Sie einen Tastencode vorgeben, mit dem der Tastaturpuffer für die Länge eines Zyklus an *Output* ausgegeben wird. Während dieser Zeit wird der Ausgang *Valid* aktiv.
- Über Flankenwechsel 0-1 an *Clear* wird der Tastaturpuffer gelöscht.
- Über *TimeAutoClear* bestimmen Sie Zeit, innerhalb der eine Taste gedrückt werden muss. Ansonsten wird der Tastaturpuffer wieder gelöscht.
- Über *CountCharAutoEnter* können Sie die Anzahl der Zeichen vorgeben, nach deren Eingabe der Tastaturpuffer für die Länge eines Zyklus an *Output* ausgegeben wird. Während dieser Zeit wird der Ausgang *Valid* aktiv.
- *Error* wird für die Zeit *TimeError* aktiviert, wenn eine Taste betätigt wird, der Tastaturpuffer aber voll ist.
- Über *TimeDebounce* können Sie für die Eingangssignale eine Entprellzeit vorgeben.



Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Row1	INPUT	BOOL	Zeile 1 der Tastaturmatrix. Default: FALSE
Row2	INPUT	BOOL	Zeile 2 der Tastaturmatrix. Default: FALSE

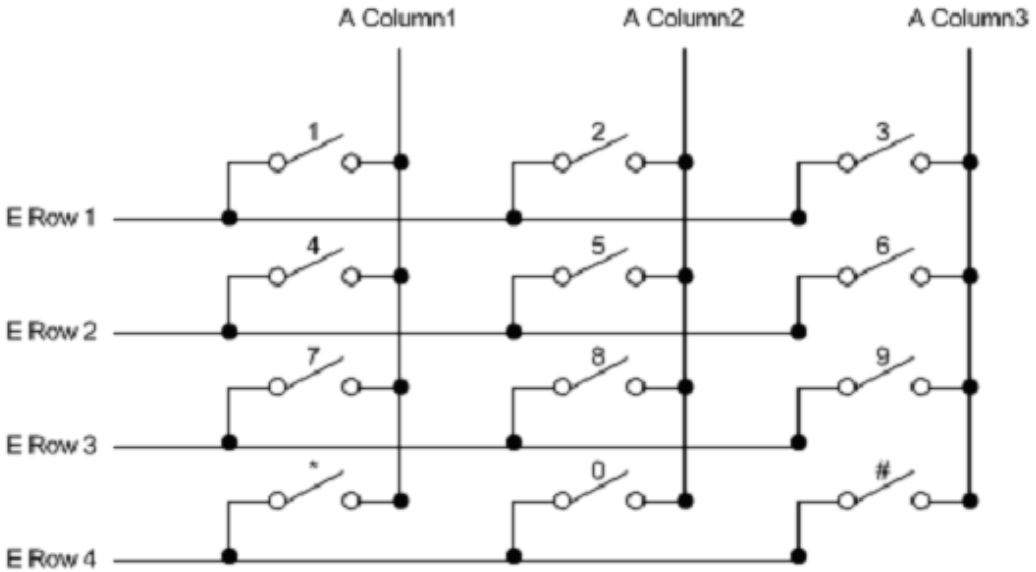
Parameter	Deklaration	Datentyp	Beschreibung
Row3	INPUT	BOOL	Zeile 3 der Tastaturmatrix. Default: FALSE
Row4	INPUT	BOOL	Zeile 4 der Tastaturmatrix. Default: FALSE
Column1	INPUT	BOOL	Spalte 1 der Tastaturmatrix. Default: FALSE
Column2	INPUT	BOOL	Spalte 2 der Tastaturmatrix. Default: FALSE
Column3	INPUT	BOOL	Spalte 3 der Tastaturmatrix. Default: FALSE
ClearCode	INPUT	BYTE	Der Wert, bei dem der Tastaturpuffer gelöscht werden soll. 0: deaktiviert Default: 42 = *
EnterCode	INPUT	BYTE	Der Wert, bei dem der Tastaturpuffer ausgegeben werden soll. 0: deaktiviert Default: 35 = #
Clear	INPUT	BOOL	Flankenwechsel 0-1 löscht den Tastaturpuffer. Default: FALSE
TimeAutoClear	INPUT	TIME	Zeitvorgabe für die Dauer innerhalb der eine Taste gedrückt werden muss. Ansonsten wird der Tastaturpuffer wieder gelöscht. 0: keine Löschung Default: 10s
CountCharAutoEnter	INPUT	INT	Anzahl der Zeichen nach deren Eingabe der Tastaturpuffer automatisch ausgegeben wird. 0: deaktiviert Default: 0
Output	OUTPUT	STRING[16]	Inhalt des Tastaturpuffers als max. 16 Byte lange Zeichenkette.
Valid	OUTPUT	BOOL	Statischer Ausgang, zeigt an, dass der über *Output* ausgegebene String gültig ist. Das Signal steht für einen Zyklus an.
Error	OUTPUT	BOOL	*Error* wird für die Zeit *TimeError* aktiviert, wenn eine Taste betätigt wird, der Tastaturpuffer aber voll ist.
TimeDebounce	CONSTANT	TIME	Zeitvorgabe zum Entprellen der Eingänge. Default: 100ms
TimeError	CONSTANT	TIME	Zeit für die Dauer des Error-Signals Default: 500ms

8.3.5 FB 50 - KEYPAD2 - Tastatur

Beschreibung

Dieser Baustein dient zur Anbindung eines externen Tastenfelds (0...9,*,#) ohne eigene Spannungsversorgung. Der Baustein liefert Ausgangs-Spalten-Signale. Abhängig von der betätigten Taste liefert die Tastatur das entsprechende Zeilen-Signal. Der Baustein wertet das Signal intern mittels einer Bitmuster-Tabelle aus und übergibt den ermittelten ASCII-Code in den Tastaturpuffer. Bei Bedarf oder automatisch wird der Tastaturpuffer als max. 16Byte lange Zeichenkette ausgegeben.

- Über die Eingänge *Row 1...4* werden die Zeilen 1...4 der Tastaturmatrix angebunden.
- Über die Ausgänge *Column 1...3* werden die Spalten 1...3 der Tastaturmatrix angebunden.
- Über *TimeDelay* geben Sie eine Wartezeit nach dem Setzen der Spalten-Ausgänge bis zum Einlesen der entsprechenden Zeilen-Eingänge vor. Diese Zeit muss größer sein als die Verzögerungszeit der verwendeten Baugruppe.
- Über *ClearCode* können Sie einen Tastencode vorgeben, mit dem der Eingabepuffer gelöscht wird.
- Über *EnterCode* können Sie einen Tastencode vorgeben, mit dem der Tastaturpuffer für die Länge eines Zyklus an *Output* ausgegeben wird. Während dieser Zeit wird der Ausgang *Valid* aktiv.
- Über Flankenwechsel 0-1 an *Clear* wird der Tastaturpuffer gelöscht.
- Über *TimeAutoClear* bestimmen Sie Zeit, innerhalb der eine Taste gedrückt werden muss. Ansonsten wird der Tastaturpuffer wieder gelöscht.
- Über *CountCharAutoEnter* können Sie die Anzahl der Zeichen vorgeben, nach deren Eingabe der Tastaturpuffer für die Länge eines Zyklus an *Output* ausgegeben wird. Während dieser Zeit wird der Ausgang *Valid* aktiv.
- *Error* wird für die Zeit *TimeError* aktiviert, wenn eine Taste betätigt wird, der Tastaturpuffer aber voll ist.
- Über *TimeDebounce* können Sie für die Eingangssignale eine Entprellzeit vorgeben.



Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Row1	INPUT	BOOL	Zeile 1 der Tastaturmatrix. Default: FALSE
Row2	INPUT	BOOL	Zeile 2 der Tastaturmatrix. Default: FALSE

Parameter	Deklaration	Datentyp	Beschreibung
Row3	INPUT	BOOL	Zeile 3 der Tastaturmatrix. Default: FALSE
Row4	INPUT	BOOL	Zeile 4 der Tastaturmatrix. Default: FALSE
ClearCode	INPUT	BYTE	Der Wert, bei dem der Tastaturpuffer gelöscht werden soll. 0: deaktiviert Default: 42 = *
EnterCode	INPUT	BYTE	Der Wert, bei dem der Tastaturpuffer ausgegeben werden soll. 0: deaktiviert Default: 35 = #
Clear	INPUT	BOOL	Flankenwechsel 0-1 löscht den Tastaturpuffer. Default: FALSE
TimeAutoClear	INPUT	TIME	Zeitvorgabe für die Dauer innerhalb der eine Taste gedrückt werden muss. Ansonsten wird der Tastaturpuffer wieder gelöscht. 0: keine Löschung Default: 10s
CountCharAutoEnter	INPUT	INT	Anzahl der Zeichen nach deren Eingabe der Tastaturpuffer automatisch ausgegeben wird. 0: deaktiviert Default: 0
Column1	OUTPUT	BOOL	Spalte 1 der Tastaturmatrix. Default: FALSE
Column2	OUTPUT	BOOL	Spalte 2 der Tastaturmatrix. Default: FALSE
Column3	OUTPUT	BOOL	Spalte 3 der Tastaturmatrix. Default: FALSE
Output	OUTPUT	BYTE	Inhalt des Tastaturpuffers als max. 16 Byte lange Zeichenkette.
Valid	OUTPUT	BOOL	Statischer Ausgang, zeigt an, dass der über *Output* ausgegebene String gültig ist. Das Signal steht für einen Zyklus an.
Error	OUTPUT	BOOL	*Error* wird für die Zeit *TimeError* aktiviert, wenn eine Taste betätigt wird, der Tastaturpuffer aber voll ist.
TimeDebounce	CONSTANT	TIME	Zeitvorgabe zum Entprellen der Eingänge. Default: 100ms

Parameter	Deklaration	Datentyp	Beschreibung
TimeError	CONSTANT	TIME	Zeit für die Dauer des Error-Signals Default: 500ms
TimeDelay	CONSTANT	TIME	Zeitvorgabe für die Dauer nach dem nach dem Setzen der Spalten-Ausgänge bis zum Einlesen der entsprechenden Zeilen-Eingänge vor. Diese Zeit muss größer sein als die Verzögerungszeit der verwendeten Baugruppe. Default: 10ms

9 Netzwerkkommunikation - *"Network Communication"*

9.1 Offene Kommunikation - *"Open Communication"*

9.1.1 Verbindungsorientierte Protokolle

- Verbindungsorientierte Protokolle bauen vor der Datenübertragung eine (logische) Verbindung zum Kommunikationspartner auf und bauen diese nach Abschluss der Datenübertragung ggf. wieder ab.
- Verbindungsorientierte Protokolle werden eingesetzt, wenn es bei der Datenübertragung insbesondere auf Sicherheit ankommt. Auch wird hier die richtige Reihenfolge der empfangenen Pakete gewährleistet.
- Über eine physikalische Leitung können in der Regel mehrere logische Verbindungen bestehen.
- Bei den FBs zur Offenen Kommunikation über Industrial Ethernet werden die folgenden verbindungsorientierten Protokolle unterstützt:
  - TCP native gemäß RFC 793 (Verbindungstypen 01h und 11h)
  - ISO on TCP gemäß RFC 1006 (Verbindungstyp 12h)

TCP native

- Bei der Datenübertragung über TCP nativ werden weder Informationen zur Länge noch über Anfang und Ende einer Nachricht übertragen. Auch besteht keine Möglichkeit zu erkennen, wo ein Datenstrom endet und der nächste beginnt.
- Die Übertragung ist stream-orientiert. Aus diesem Grund sollten Sie in den FBs bei Sender und Empfänger identische Datenlängen angeben.
- Falls die empfangene Anzahl der Daten von der parametrierten Länge abweicht, erhalten Sie entweder Daten, welche nicht die vollständigen Telegrammdaten enthalten oder mit dem Inhalt eines nachfolgenden Telegramms aufgefüllt sind.
- Der Empfangsbaustein kopiert so viele Bytes in den Empfangsbereich, wie Sie als Länge parametriert haben. Anschließend setzt er NDR auf TRUE und beschreibt RCVD_LEN mit dem Wert von LEN. Mit jedem weiteren Aufruf erhalten Sie damit einen weiteren Block der gesendeten Daten.

ISO on TCP

- Bei der Datenübertragung werden Informationen zur Länge und zum Ende einer Nachricht übertragen. Die Übertragung ist blockorientiert.
- Falls Sie die Länge der zu empfangenden Daten größer gewählt haben als die Länge der gesendeten Daten, kopiert der Empfangsbaustein die gesendeten Daten vollständig in den Empfangsdatenbereich. Anschließend setzt er NDR auf TRUE und beschreibt RCVD_LEN mit der Länge der gesendeten Daten.
- Falls Sie die Länge der zu empfangenden Daten kleiner gewählt haben als die Länge der gesendeten Daten, kopiert der Empfangsbaustein keine Daten in den Empfangsdatenbereich, sondern liefert folgende Fehlerinformation: ERROR = 1, STATUS = 8088h.

9.1.2 Verbindungslose Protokolle

Bei den verbindungslosen Protokollen entfallen Verbindungsauf- und Verbindungsabbau zum remoten Partner. Verbindungslose Protokolle übertragen die Daten unquittiert und damit ungesichert zum remoten Partner. Bei den FBs zur Offenen Kommunikation über Industrial Ethernet wird das folgende verbindungslose Protokoll unterstützt:

- UDP gemäß RFC 768 (Verbindungstyp 13h)

UDP

- Bei Aufruf des Sendebausteins ist ein Verweis auf die Adressparameter des Empfängers (IP-Adresse und Port-Nr.) anzugeben. Auch werden Informationen zur Länge und zum Ende einer Nachricht übertragen.
- Analog erhalten Sie nach Abschluss des Empfangsbausteins einen Verweis auf die Adressparameter des Senders (IP-Adresse und Port-Nr.).

- Damit sie Sende- und Empfangsbaustein nutzen können, müssen Sie zuvor sowohl auf der Sender- als auch auf der Empfängerseite einen lokalen Kommunikationszugangspunkt einrichten.
- Bei jedem Sendauftrag können Sie den remoten Partner durch Angabe seiner IP-Adresse und seiner Port-Nr. neu referenzieren.
- Falls Sie die Länge der zu empfangenden Daten größer gewählt haben als die Länge der gesendeten Daten, kopiert der Empfangsbaustein die gesendeten Daten vollständig in den Empfangsdatenbereich. Anschließend setzt er NDR auf TRUE und beschreibt RCVD_LEN mit der Länge der gesendeten Daten.
- Falls Sie die Länge der zu empfangenden Daten kleiner gewählt haben als die Länge der gesendeten Daten, kopiert der Empfangsbaustein keine Daten in den Empfangsdatenbereich, sondern liefert folgende Fehlerinformation: ERROR = 1, STATUS = 8088h.

9.1.3 FB 63 - TSEND - Daten senden - TCP native und ISO on TCP

Beschreibung

- Der FB 63 TSEND sendet Daten über eine bestehende Kommunikationsverbindung. Er ist ein asynchron arbeitender FB, d.h. die Bearbeitung erstreckt sich über mehrere FB-Aufrufe.
- Sie starten den Sendevorgang, indem Sie den FB 63 mit *REQ* = 1 aufrufen.
- Über den Ausgangsparameter *BUSY* und den Ausgangsparameter *STATUS* wird der Zustand des Auftrags angezeigt. Dabei entspricht *STATUS* dem Ausgangsparameter *RET_VAL* der asynchron arbeitenden SFCs (siehe auch Bedeutung von *REQ*, *RET_VAL* und *BUSY* bei asynchron arbeitenden SFCs).
- In der folgenden Tabelle ist der Zusammenhang zwischen *BUSY*, *DONE* und *ERROR* angegeben. Mit ihrer Hilfe können Sie feststellen, in welchem Zustand sich der FB 63 aktuell befindet bzw. wann der Verbindungsaufbau beendet ist.

BUSY	DONE	ERROR	Beschreibung
TRUE	irrelevant	irrelevant	Der Auftrag ist in Bearbeitung.
FALSE	TRUE	FALSE	Der Auftrag wurde erfolgreich abgeschlossen.
FALSE	FALSE	TRUE	Der Auftrag wurde mit einem Fehler beendet. Die Fehlerursache können Sie dem Parameter *STATUS* entnehmen.
FALSE	FALSE	FALSE	Dem FB wurde kein (neuer) Auftrag erteilt.

Durch die asynchrone Arbeitsweise des FB 63 TSEND müssen Sie die Daten im Sendebereich so lange konsistent halten, bis der Parameter DONE oder der Parameter ERROR den Wert TRUE annimmt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L	Steuerparameter *REQ*, aktiviert den Sendevorgang der durch *ID* gegebenen Verbindung bei steigender Flanke. Beim erstmaligen Aufruf mit *REQ* = 1 werden Daten aus dem mit Parameter *DATA* angegebenen Bereich übergeben.
ID	INPUT	WORD	M, D, Konstante	Referenz auf die zugehörige Verbindung. *ID* muss identisch sein mit dem zugehörigen Parameter *ID* in der lokalen Verbindungsbeschreibung. Wertebereich: 0001h ... 0FFFh
LEN	INPUT	INT	E, A, M, D, L	Anzahl der Bytes, die mit dem Auftrag gesendet werden sollen Wertebereich: ■ 1 ... 1460, falls Verbindungstyp = 01h ■ 1 ... 8192, falls Verbindungstyp = 11h ■ 1 ... 1452, falls Verbindungstyp = 12h und ein CP benutzt wird ■ 1 ... 8192, falls Verbindungstyp = 12h und kein CP benutzt wird.
DONE	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *DONE*: ■ 0: Auftrag wurde noch nicht gestartet oder wird noch ausgeführt. ■ 1: Auftrag wurde fehlerfrei ausgeführt.
BUSY	OUTPUT	BOOL	E, A, M, D, L	■ *BUSY* = 1: Der Auftrag ist noch nicht beendet. Es kann kein neuer Auftrag angestoßen werden. ■ *BUSY* = 0: Der Auftrag ist beendet.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter ERROR: ■ *ERROR* = 1: Bei der Bearbeitung ist ein Fehler aufgetreten. *STATUS* liefert detaillierte Auskunft über die Art des Fehlers.
STATUS	OUTPUT	WORD	M, D	Zustandsparameter *STATUS*: Statusinformationen
DATA	IN_OUT	ANY	E, A, M, D	Sendebereich, enthält Adresse und Länge. Die Adresse verweist auf: ■ das Prozessabbild der Eingänge ■ das Prozessabbild der Ausgänge ■ einen Merker ■ einen Datenbaustein Zulässige referenzierte Datentypen: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING

Statusinformationen

ERROR	STATUS	Beschreibung
0	0000h	Sendeauftrag wurde ohne Fehler abgeschlossen.
0	7000h	Erstaufruf mit *REQ* = 0, kein Anstoß zum Senden.
0	7001h	Erstaufruf mit *REQ* = 1, Anstoß des Sendevorgangs.
0	7002h	Zwischenaufruf (*REQ* irrelevant), Auftrag ist in Bearbeitung **Hinweis:** In dieser Bearbeitungsphase greift das Betriebssystem auf die Daten im Sendepuffer *DATA* zu.
1	8085h	Parameter *LEN* hat den Wert 0 oder ist größer als der größte zulässige Wert.
1	8086h	Parameter *ID* liegt in einem unzulässigen Wertebereich.
0	8088h	Parameter *LEN* ist größer als der in *DATA* angegebene Speicherbereich.
1	80A1h	Kommunikationsfehler: ■ Zu der angegebenen *ID* wurde noch kein FB 65 TCON aufgerufen. ■ Die angegebene Verbindung wird momentan abgebaut. Ein Sendevorgang über diese Verbindung ist nicht möglich. ■ Schnittstelle wird neu initialisiert.
1	80B3h	Der parametrierte Verbindungstyp (Parameter *connection_type* in der Verbindungsbeschreibung) ist UDP. Bitte verwenden Sie den FB 67 TUSEND.
1	80C3h	Die Betriebsmittel (Speicher) der CPU sind temporär belegt.
1	80C4h	Temporärer Kommunikationsfehler: ■ Die Verbindung zum Kommunikationspartner kann momentan nicht aufgebaut werden. ■ Die Schnittstelle wird neu parametriert.
1	8822h	Parameter *DATA*: Quellbereich ungültig, Bereich im DB nicht vorhanden.
1	8824h	Parameter *DATA*: Bereichsfehler im ANY-Pointer.
1	8832h	Parameter *DATA*: DB-Nummer ist zu groß.
1	883Ah	Parameter *DATA*: Zugriff auf Sendepuffer nicht möglich (z.B. gelöschter DB).
1	887Fh	Parameter *DATA*: Interner Fehler, z.B. unzulässige ANY-Referenz.
1	8F7Fh	Interner Fehler (VIPA-spezifisch)
1	8xyyh	Allgemeine Fehlerinformation ↳ *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

9.1.4 FB 64 - TRCV - Daten empfangen - TCP native und ISO on TCP

Beschreibung

Der FB 64 TRCV empfängt Daten über eine bestehende Kommunikationsverbindung. Für den Empfang und die Weiterverarbeitung der Daten gibt es folgende Varianten:

- Variante 1: Empfangenen Datenblock sofort weiterverarbeiten.
- Variante 2: Empfangene Datenblöcke in einem Empfangspuffer speichern und erst dann weiterverarbeiten, wenn dieser gefüllt ist.

Der Zusammenhang zwischen Verbindungstyp und den beiden Varianten ist in folgender Tabelle dargestellt.

Verbindungstyp	Variante
01h und 11h	Sie können die Variante selbst bestimmen.
12h	Variante 2 (fix)

Die beiden Varianten sind in der folgenden Tabelle näher beschrieben.

Empfangene Daten ...	Wertebereich von *LEN*	Wertebereich von *RCVD_LEN*	Beschreibung
stehen sofort zur Verfügung.	0	1 ... x	Sie übergeben einen Puffer, dessen Länge x im ANY-Pointer des Empfangspuffers hinterlegt ist (Parameter *DATA*). Nachdem ein Datenblock empfangen wurde, wird dieser sofort im Empfangspuffer zur Verfügung gestellt. Die Anzahl der empfangenen Daten (Parameter *RCVD_LEN*) kann maximal so groß sein wie die im Parameter *DATA* hinterlegte Größe. Der Empfang wird angezeigt durch *NDR* = 1.
im Empfangspuffer speichern. Sie stehen zur Verfügung, sobald die projektierte Länge erreicht wird.	1 ... 1460, falls Verbindungstyp = 01h 1 ... 8192, falls Verbindungstyp = 11h 1 ... 1452, falls Verbindungstyp = 12h und ein CP benutzt wird 1 ... 8192, falls Verbindungstyp = 12h und kein CP benutzt wird	gleicher Wert wie im Parameter LEN	Sie übergeben die Empfangslänge am Parameter *LEN*. Wenn diese parametrierte Länge erreicht ist, werden die Empfangsdaten im Parameter *DATA* zur Verfügung gestellt (*NDR* = 1).

Arbeitsweise

- Der FB 64 TRCV ist ein asynchron arbeitender FB, d.h. die Bearbeitung erstreckt sich über mehrere FB-Aufrufe. Sie starten den Empfangsvorgang, indem Sie den FB 64 mit *REQ* = 1 aufrufen.
- Über den Ausgangsparameter *BUSY* und den Ausgangsparameter STATUS wird der Zustand des Auftrags angezeigt. Dabei entspricht STATUS dem Ausgangsparameter *RET_VAL* der asynchron arbeitenden SFCs (siehe auch Bedeutung von *REQ*, *RET_VAL* und *BUSY* bei asynchron arbeitenden SFCs).
- In der folgenden Tabelle ist der Zusammenhang zwischen *BUSY*, *NDR* und *ERROR* angegeben. Mit ihrer Hilfe können Sie feststellen, in welchem Zustand sich der FB 64 TRCV aktuell befindet bzw. wann der Empfangsvorgang beendet ist.

BUSY	DONE	ERROR	Beschreibung
TRUE	irrelevant	irrelevant	Der Auftrag ist in Bearbeitung.
FALSE	TRUE	FALSE	Der Auftrag wurde erfolgreich abgeschlossen.
FALSE	FALSE	TRUE	Der Auftrag wurde mit einem Fehler beendet. Die Fehlerursache können Sie dem Parameter *STATUS* entnehmen.
FALSE	FALSE	FALSE	Dem FB wurde kein (neuer) Auftrag erteilt.

Durch die asynchrone Arbeitsweise des FB 64 TRCV sind die Daten im Empfangsbereich erst dann konsistent, wenn der Parameter NDR den Wert TRUE annimmt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN_R	INPUT	BOOL	E, A, M, D, L	Mit EN_R = 1 wird der FB 64 TRCV empfangsbereit (Steuerparameter). Der Empfangsauftrag wird bearbeitet.
ID	INPUT	WORD	M, D, Konstante	Referenz auf die zugehörige Verbindung. *ID* muss identisch sein mit dem zugehörigen Parameter *id* in der lokalen Verbindungsbeschreibung. Wertebereich: 0001h ... 0FFFh

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
LEN	INPUT	INT	E, A, M, D, L	■ *LEN* = 0 (Ad-Hoc-Mode) empfohlen bei ISO on TCP: implizite Längenangabe im ANY-Pointer DATA verwenden. Die empfangenen Daten werden beim Bausteinaufruf unmittelbar bereitgestellt. Die Anzahl der empfangenen Daten steht in RCVD_LEN zur Verfügung. ■ 1 ≤ LEN ≤ max: Anzahl der Bytes, die empfangen werden sollen. Die Anzahl der tatsächlich empfangenen Daten steht in *RCVD_LEN* zur Verfügung. Die Daten stehen Ihnen dann zur Verfügung, wenn sie vollständig empfangen wurden. max hängt vom Verbindungstyp ab: – max = 1460 beim Verbindungstyp 01h – max = 8192 beim Verbindungstyp 11h – max = 1452 beim Verbindungstyp 12h mit CP-Einsatz – max = 8192 beim Verbindungstyp 12h ohne CP-Einsatz
NDR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *NDR*: ■ *NDR* = 0: Auftrag wurde noch nicht gestartet oder läuft noch ■ *NDR* = 1: Auftrag wurde erfolgreich abgeschlossen
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*: ■ *ERROR* = 1: Bei der Bearbeitung ist ein Fehler aufgetreten. *STATUS* liefert detaillierte Auskunft über die Art des Fehlers.
BUSY	OUTPUT	BOOL	E, A, M, D, L	■ *BUSY* = 1: Der Auftrag ist noch nicht beendet. Es kann kein neuer Auftrag angestoßen werden. ■ *BUSY* = 0: Der Auftrag ist beendet.
STATUS	OUTPUT	WORD	M, D	Zustandsparameter *STATUS*: Statusinformationen
RCVD_LEN	OUTPUT	INT	E, A, M, D, L	Anzahl der tatsächlich empfangenen Daten in Bytes
DATA	IN_OUT	ANY	E, A, M, D	Empfangsbereich (Adresse und Länge). Die Adresse verweist auf: ■ das Prozessabbild der Eingänge ■ das Prozessabbild der Ausgänge ■ einen Merker ■ einen Datenbaustein Zulässige referenzierte Datentypen: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING

Statusinformationen

ERROR	STATUS	Beschreibung
0	0000h	Neue Daten wurden übernommen. Die aktuelle Länge der empfangenen Daten wird in *RCVD_LEN* angezeigt.
0	7000h	Erstaufruf mit *REQ* = 0, kein Anstoß zum Empfangen
0	7001h	Baustein ist empfangsbereit. Empfangsauftrag wurde aktiviert.
0	7002h	Zwischenaufruf, Auftrag ist in Bearbeitung **Hinweis:** In dieser Bearbeitungsphase schreibt das Betriebssystem Daten in den Empfangspuffer *DATA*. Deshalb können im Fehlerfall inkonsistente Daten im Empfangspuffer stehen.
1	8085h	Parameter *LEN* ist größer als der größte zulässige Wert, oder Sie haben den Wert von *LEN* gegenüber dem Erstaufruf geändert
1	8086h	Parameter *ID* liegt in einem unzulässigen Wertebereich
1	8088h	■ Zielpuffer (*DATA*) ist zu klein Wert in *LEN* ist größer als der durch *DATA* vorgegebene Empfangsbereich. Fehlerbehebung falls Verbindungstyp = 12h: Vergrößern Sie den Zielpuffer *DATA*.
1	80A1h	Kommunikationsfehler: ■ Zu der angegebenen *ID* wurde noch kein FB 65 TCON aufgerufen ■ Die angegebene Verbindung wird momentan abgebaut. Ein Empfangsvorgang über diese Verbindung ist nicht möglich. ■ Die Schnittstelle wird neu parametriert.
1	80B3h	Der parametrierte Verbindungstyp (Parameter *connection_type* in der Verbindungsbeschreibung) ist UDP. Bitte verwenden Sie den FB 68 TURCV.
1	80C3h	Die Betriebsmittel (Speicher) der CPU sind temporär belegt
1	80C4h	Temporärer Kommunikationsfehler: Die Verbindung wird gerade aufgebaut.
1	8922h	Parameter *DATA*: Zielbereich ungültig, Bereich im DB nicht vorhanden
1	8924h	Parameter *DATA*: Bereichsfehler im ANY-Pointer
1	8932h	Parameter *DATA*: DB-Nummer ist zu groß
1	893Ah	Parameter *DATA*: Zugriff auf Empfangspuffer nicht möglich (z.B. gelöschter DB)
1	897Fh	Parameter *DATA*: Interner Fehler, z.B. unzulässige ANY-Referenz
1	8F7Fh	Interner Fehler (VIPA-spezifisch)
1	8xyyh	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

9.1.5 FB 65 - TCON - Verbindung aufbauen

Verwendung bei TCP native und ISO on TCP

Beide Kommunikationspartner rufen den FB 65 TCON zum Aufbau der Kommunikationsverbindung auf. In der Parametrierung hinterlegen Sie, welcher der aktive und welcher der passive Kommunikationsendpunkt ist. Die Anzahl der möglichen Verbindungen entnehmen Sie den Technischen Daten Ihrer CPU. Nach dem Aufbau der Verbindung wird diese automatisch von der CPU überwacht und gehalten. Bei Verbindungsabbruch durch z.B. Leitungsunterbrechung oder durch den remoten Kommunikationspartner versucht der aktive Partner die Verbindung wieder aufzubauen. Sie müssen den FB 65 TCON nicht erneut aufrufen. Mit dem Aufruf des FB 66 TDISCON oder im Betriebszustand STOP der CPU wird eine bestehende Verbindung abgebrochen. Zum erneuten Aufbau der Verbindung müssen Sie den FB 65 TCON nochmals aufrufen.

Verwendung bei UDP

Beide Kommunikationspartner rufen den FB 65 TCON auf, um ihren lokalen Kommunikationszugangspunkt einzurichten. Dabei wird eine Verbindung zwischen Anwenderprogramm und der Kommunikationsschicht des Betriebssystems eingerichtet. Es erfolgt kein Verbindungsaufbau zum remoten Partner. Der lokale Zugangspunkt wird zum Senden und Empfangen von UDP Telegrammen verwendet.

Beschreibung

Der FB 65 TCON ist ein asynchron arbeitender FB, d.h. die Bearbeitung erstreckt sich über mehrere FB-Aufrufe. Sie starten den Verbindungsaufbau, indem Sie den FB 65 mit *REQ* = 1 aufrufen. Über den Ausgangsparameter *BUSY* und den Ausgangsparameter *STATUS* wird der Zustand des Auftrags angezeigt. Dabei entspricht *STATUS* dem Ausgangsparameter *RET_VAL* der asynchron arbeitenden SFCs (siehe auch Bedeutung von *REQ*, *RET_VAL* und *BUSY* bei asynchron arbeitenden SFCs). In der folgenden Tabelle ist der Zusammenhang zwischen *BUSY*, *DONE* und *ERROR* angegeben. Mit ihrer Hilfe können Sie feststellen, in welchem Zustand sich der FB 65 aktuell befindet bzw. wann der Verbindungsaufbau beendet ist.

BUSY	DONE	ERROR	Beschreibung
TRUE	irrelevant	irrelevant	Der Auftrag ist in Bearbeitung.
FALSE	TRUE	FALSE	Der Auftrag wurde erfolgreich abgeschlossen.
FALSE	FALSE	TRUE	Der Auftrag wurde mit einem Fehler beendet. Die Fehlerursache können Sie dem Parameter *STATUS* entnehmen.
FALSE	FALSE	FALSE	Dem FB wurde kein (neuer) Auftrag erteilt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L	Steuerparameter *REQ*, aktiviert den Verbindungsaufbau bei steigender Flanke
ID	INPUT	WORD	M, D, Konstante	Referenz auf die aufzubauende Verbindung zum remoten Partner bzw. zwischen Anwenderprogramm und Kommunikationsschicht des Betriebssystems. ID muss identisch sein mit dem zugehörigen Parameter id in der lokalen Verbindungsbeschreibung. Wertebereich: 0001h ... 0FFFh
DONE	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *DONE*: ■ 0: Auftrag wurde noch nicht gestartet oder wird noch ausgeführt. ■ 1: Auftrag wurde fehlerfrei ausgeführt.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
BUSY	OUTPUT	BOOL	E, A, M, D, L	■ *BUSY* = 1: Der Auftrag ist noch nicht beendet. ■ *BUSY* = 0: Der Auftrag ist beendet.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*: ■ *ERROR* = 1: Bei der Bearbeitung ist ein Fehler aufgetreten. *STATUS* liefert detaillierte Auskunft über die Art des Fehlers.
STATUS	OUTPUT	WORD	M, D	Zustandsparameter *STATUS*: Statusinformationen
CONNECT	IN_OUT	ANY	D	Zeiger auf die zugehörige Verbindungsbeschreibung. *Kap. 9.1.6 "UDT 65 - TCON_PAR - Datenstruktur für FB 65" Seite 328*

Statusinformationen

ERROR	STATUS	Beschreibung
0	0000h	Verbindung konnte aufgebaut werden
0	7000h	Aufruf mit *REQ* = 0, kein Anstoß zum Verbindungsaufbau
0	7001h	Erstaufruf mit *REQ* = 1, Verbindung wird aufgebaut
0	7002h	Zwischenaufruf (*REQ* irrelevant), Verbindung wird aufgebaut
1	8086h	Der Parameter ID darf nicht den Wert Null haben.
0	8087h	Maximale Anzahl der Verbindungen ist erreicht, keine weitere Verbindung möglich.
1	8089h	Der Parameter *CONNECT* zeigt nicht auf einen Datenbaustein.
1	809Ah	Der Parameter *CONNECT* zeigt auf ein Feld, das nicht die Länge der Verbindungsbeschreibung (UDT 65) hat.
1	809Bh	Die über *local_device_id* und *next_staddr* angegebene Kommunikationsschnittstelle wird von der CPU nicht unterstützt.
1	80A1h	Verbindung bzw. Port ist bereits vom Anwender belegt.
1	80A2h	Lokaler oder remoter Port ist vom System belegt.
1	80A3h	Es wird versucht, eine bestehende Verbindung erneut aufzubauen.
1	80A4h	IP-Adresse des remoten Verbindungsendpunkts ist ungültig.
1	80A7h	Kommunikationsfehler: Sie haben nach einem nicht abgeschlossenen TCON einen TDISCON aufgerufen. Der Verbindungsaufbau wurde durch Aufruf eines TDISCON vorzeitig abgebrochen.
1	80B4h	Beim Protokoll ISO on TCP sind beim passiven Verbindungsaufbau eine oder mehrere der folgenden Bedingungen verletzt worden: ■ local_tsap_id_len ≥ 02h ■ local_tsap_id[1] = E0h bei local_tsap_id_len = 02h ■ local_tsap_id[1] ein ASCII-Zeichen bei local_tsap_id_len ≥ 03h ■ local_tsap_id[1] ist ein ASCII-Zeichen und local_tsap_id_len ≥ 03h
1	80B5h	Parameter active_est (UDT 65) ist TRUE bei der Protokollvariante UDP.
1	80B6h	Parameter connection_type ist ungültig (UDT 65).

ERROR	STATUS	Beschreibung
1	80B7h	Fehler in einem der folgenden Parameter des UDT 65: ■ block_length ■ local_tsap_id_len ■ rem_subnet_id_len ■ rem_staddr_len ■ rem_tsap_id_len ■ next_staddr_len
1	80B8h	Parameter *id* in der lokalen Verbindungsbeschreibung (UDT 65) und Parameter ID sind unterschiedlich.
1	80C3h	Temporärer Ressourcenmangel der CPU
1	80C4h	Temporärer Kommunikationsfehler: ■ Verbindung kann momentan nicht aufgebaut werden. ■ Die Schnittstelle wird neu parametriert .
1	8F7Fh	Interner Fehler (VIPA-spezifisch)
1	8xyyh	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

9.1.6 UDT 65 - TCON_PAR - Datenstruktur für FB 65

9.1.6.1 Datenstruktur für Verbindung

In der Verbindungsparametrierung von TCP native bzw. ISO on TCP legen Sie fest, welcher Kommunikationspartner den Verbindungsaufbau aktiviert und welcher auf eine Anforderung des Kommunikationspartners hin einen passiven Verbindungsaufbau durchführt. Wenn beide Kommunikationspartner ihren Verbindungsaufbau angestoßen haben, kann das Betriebssystem die Kommunikationsverbindung herstellen. Für die Kommunikation ist ein DB erforderlich. Hierbei erhält der DB seine Datenstruktur aus dem UDT 65 TCON_PAR. Für jede Verbindung ist solch eine Datenstruktur erforderlich, die Sie auch in einem globalen DB zusammenfassen können. Der Verbindungsparameter CONNECT des FB 65 TCON enthält einen Verweis auf die Adresse der zugehörigen Verbindungsbeschreibung (z.B. P#DB10.DBX0.0 Byte 64).

Datenstruktur

Byte	Parameter	Datentyp	Anfangswert	Beschreibung
0 ... 1	block_length	WORD	40h	Länge des UDT 65: 64 Bytes (fest)
2 ... 3	id	WORD	0000h	■ Referenz auf diese Verbindung (Wertebereich: 0001h ... 0FFFh) ■ Den Wert dieses Parameters müssen Sie im jeweiligen Baustein bei *ID* angeben.
4	connection_type	BYTE	01h	Verbindungstyp: ■ 11h: TCP/IP native ■ 12h: ISO on TCP ■ 13h: UDP ■ 01h: TCP/IP native - Kompatibilitätsmode

Byte	Parameter	Datentyp	Anfangswert	Beschreibung
5	active_est	BOOL	FALSE	Kennung für die Art des Verbindungsaufbaus: TCP, TCP, IoT: ■ FALSE: passiver Verbindungsaufbau ■ TRUE: aktiver Verbindungsaufbau UDP: ■ FALSE
6	local_device_id	BYTE	02h	Kommunikationsschnittstelle ■ 00h: Ethernet-PG/OP-Kanal der CPU ■ 02h: Ethernet-CP der CPU
7	local_tsap_id_len	BYTE	02h	Verwendete Länge des Parameters *local_tsap_id*; mögliche Werte: TCP ■ Aktive Seite: 0 (dynamische Portvergabe) oder 2 ■ Passive Seite: 2 ISO on TCP ■ 2 ... 16 UDP ■ 2 TCP ■ Aktive Seite: 0 ■ Passive Seite: 2
8	rem_subnet_id_len	BYTE	00h	Dieser Parameter wird derzeit nicht verwendet. Sie müssen ihn mit 00h belegen.
9	rem_staddr_len	BYTE	00h	Länge der Adresse des remoten Verbindungsendpunkts: TCP/ISO on TCP/TCP (Komp.) ■ 0: unspezifiziert, d.h. Parameter *rem_staddr* ist irrelevant. ■ 4: gültige IP-Adr. im Parameter *rem_staddr* UDP ■ 0*

Byte	Parameter	Datentyp	Anfangswert	Beschreibung
10	rem_tsap_id_len	BYTE	00h	Verwendete Länge des Parameters *rem_tsap_id*; mögliche Werte: TCP ■ Aktive Seite: 2 (Der Port muss spezifiziert werden.) ■ Passive Seite: 0 oder 2 ISO on TCP ■ 0 oder 2 ... 16 UDP ■ Dieser Parameter wird nicht verwendet. Parameter mit 00h belegen. TCP (Komp.) ■ Aktive Seite: 2 (Der Port muss spezifiziert werden.) Für die passive Seite ist nur der Wert 0 zulässig.
11	next_staddr_len	BYTE	00h	Verwendete Länge des Parameters *next_staddr* ■ 00h: Ethernet-CP der CPU ■ 01h: Ethernet-PG/OP-Kanal der CPU
12 ... 27	local_tsap_id	ARRAY [1..16] of BYTE	00h ...	Bei *connection_type*: TCP, UDP ■ local_tsap_id[1] = high byte der Port-Nr. in hexadezimaler Darstellung ■ local_tsap_id[2] = low byte der Port-Nr. in hexadezimaler Darstellung ■ local_tsap_id[3-16] = 00h ISO on TCP ■ lokale TSAP-ID (mögliche Werte: 2000 ... 5000) – local_tsap_id[1] = E0h (Verbindungstyp T-Verbindung) – local_tsap_id[2] = Rack und Steckplatz der eigenen CPU (Bits 0 ... 4 Steckplatz, Bits 5 ... 7: Racknummer) – local_tsap_id[3-16] = TSAP-Erweiterung TCP (Komp.) ■ local_tsap_id[1] = low byte der Port-Nr. in hexadezimaler Darstellung ■ local_tsap_id[2] = high byte der Port-Nr. in hexadezimaler Darstellung ■ local_tsap_id[3-16] = 00h **Hinweis:** Stellen Sie sicher, dass jeder Wert von *local_tsap_id*, den Sie auf Ihrer CPU verwenden, eindeutig ist.
28 ... 33	rem_subnet_id	ARRAY [1..6] of BYTE	00h ...	Dieser Parameter wird derzeit nicht verwendet. Sie müssen ihn mit 00h belegen.

Byte	Parameter	Datentyp	Anfangswert	Beschreibung
34 ... 39	rem_staddr	ARRAY [1..6] of BYTE	00h ...	IP-Adresse des remoten Verbindungsendpunkts, z.B. 192.168.002.003: bei *connection_type* ■ TCP / ISO on TCP – rem_staddr[1] = C0h (192) – rem_staddr[2] = A8h (168) – rem_staddr[3] = 02h (002) – rem_staddr[4] = 03h (003) – rem_staddr[5-6] = irrelevant ■ UDP – Dieser Parameter wird nicht verwendet. Sie müssen ihn mit 00h belegen. ■ TCP (Komp.) – rem_staddr[1] = 03h (003) – rem_staddr[2] = 02h (002) – rem_staddr[3] = A8h (168) – rem_staddr[4] = C0h (192) – rem_staddr[5-6] = irrelevant
40 ... 55	rem_tsap_id	ARRAY [1..16] of BYTE	00h ...	Bei *connection_type* ■ TCP: remote Port-Nr. (mögliche Werte: 2000 ... 5000), – rem_tsap_id[1] = High-Byte der Port-Nr. in hexadezimaler Darstellung, – rem_tsap_id[2] = Low-Byte der Port-Nr. in hexadezimaler Darstellung, – rem_tsap_id[3-16] = 00h ■ ISO on TCP: remote TSAP-ID: – rem_tsap_id[1] = E0h (Verbindungstyp T-Verbindung), – rem_tsap_id[2] = Rack und Steckplatz des remoten Verbindungsendpunkts (CPU) (Bits 0 ... 4: Steckplatz, Bits 5 ... 7: Racknummer), – rem_tsap_id[3-16] = TSAP-Erweiterung ■ UDP Dieser Parameter wird nicht verwendet. Sie müssen ihn mit 00h belegen. ■ TCP (Komp.): remote Port-Nr. (mögliche Werte: 2000 ... 5000), – local_tsap_id[1] = Low-Byte der Port-Nr. in hexadezimaler Darstellung, – local_tsap_id[2] = High-Byte der Port-Nr. in hexadezimaler Darstellung, – local_tsap_id[3-16] = 00h
56 ... 61	next_staddr	ARRAY [1..6] of BYTE	00h ...	Rack und Steckplatz des projektierten CP für die PG/OP-Schnittstelle ■ 00h (Ethernet-PG/OP-Kanal) – next_staddr[1]: 04h – next_staddr[2-6]: 00h ■ 02h (Ethernet-CP) – next_staddr[1-6]: 00h

Byte	Parameter	Datentyp	Anfangswert	Beschreibung
62 ... 63	spare	WORD	0000h	irrelevant

*) Die Partner IP-Adresse wird beim Aufruf von TUSEND/TURECV über den Parameter *ADDR* angegeben.

9.1.6.2 Datenstruktur für Kommunikationszugangspunkt

Ein Kommunikationszugangspunkt stellt die Verbindung zwischen Anwenderprogramm der Kommunikationsschicht des Betriebssystems dar. Für die Kommunikation über UDP muss jeder Kommunikationspartner einen Kommunikationszugangspunkt mittels eines DB definieren. Hierbei erhält der DB seine Datenstruktur aus dem UDT 65 "TCON_PAR".

Datenstruktur

Byte	Parameter	Datentyp	Anfangswert	Beschreibung
0 ... 1	block_length	WORD	40h	Länge des UDT 65: 64 Bytes (fest)
2 ... 3	id	WORD	0000h	■ Referenz auf diese Verbindung zwischen-Anwenderprogramm und Kommunikations-schicht des Betriebssystems (Wertebereich: 0001h ... 0FFFh) ■ Den Wert dieses Parameters müssen Sie im jeweiligen Baustein bei ID angeben.
4	connection_type	BYTE	01h	Verbindungstyp: ■ 13h: UDP
5	active_est	BOOL	FALSE	Kennung für die Art des Verbindungsaufbaus: Diesen Parameter müssen Sie mit FALSE belegen, da über den Kommunikationszugangs-punkt Daten sowohl gesendet als auch emp-fangen werden können.
6	local_device_id	BYTE	02h	Kommunikationsschnittstelle ■ 00h: Ethernet-PG/OP-Kanal der CPU ■ 02h: Ethernet-CP der CPU
7	local_tsap_id_len	BYTE	02h	Verwendete Länge des Parameters *local_tsap_id*; möglicher Wert: 2
8	rem_subnet_id_len	BYTE	00h	Dieser Parameter wird nicht verwendet. Wert 00h (fix)
9	rem_staddr_len	BYTE	00h	Dieser Parameter wird nicht verwendet. Wert 00h (fix)
10	rem_tsap_id_len	BYTE	00h	Dieser Parameter wird nicht verwendet. Wert 00h (fix)
11	next_staddr_len	BYTE	00h	Dieser Parameter wird nicht verwendet. Wert 00h (fix)

Byte	Parameter	Datentyp	Anfangswert	Beschreibung
12 ... 27	local_tsap_id	ARRAY [1..16] of BYTE	00h ...	■ lokale Port-Nr. (mögliche Werte: 2000 ... 5000) – local_tsap_id[1] = High-Byte der Port-Nr. in hexadezimaler Darstellung – local_tsap_id[2] = Low-Byte der Port-Nr. in hexadezimaler Darstellung – local_tsap_id[3-16] = irrelevant **Hinweis:** Stellen Sie sicher, dass jeder Wert von *local_tsap_id*, den Sie auf Ihrer CPU verwenden, eindeutig ist.
28 ... 33	rem_subnet_id	ARRAY [1..6] of BYTE	00h ...	Dieser Parameter wird nicht verwendet. Wert 00h (fix)
34 ... 39	rem_staddr	ARRAY [1..6] of BYTE	00h ...	Dieser Parameter wird nicht verwendet. Wert 00h (fix)
40 ... 55	rem_tsap_id	ARRAY [1..16] of BYTE	00h ...	Dieser Parameter wird nicht verwendet. Wert 00h (fix)
56 ... 61	next_staddr	ARRAY [1..6] of BYTE	00h ...	Dieser Parameter wird nicht verwendet. Wert 00h (fix)
62 ... 63	spare	WORD	0000h	irrelevant

9.1.7 FB 66 - TDISCON - Verbindung abbauen

Verwendung bei TCP native und ISO on TCP

Der FB 66 TDISCON baut eine Kommunikationsverbindung der CPU zu einem Kommunikationspartner ab.

Verwendung bei UDP

Der FB 66 TDISCON löst den lokalen Kommunikationszugangspunkt auf, d.h. die Verbindung zwischen Anwenderprogramm und Kommunikationsschicht des Betriebssystems wird abgebaut.

Beschreibung

Der FB 66 TDISCON ist ein asynchron arbeitender FB, d.h. die Bearbeitung erstreckt sich über mehrere FB-Aufrufe. Sie starten den Verbindungsabbau, indem Sie den FB 66 mit *REQ* = 1 aufrufen.

Nach dem erfolgreichen Durchlauf des FB 66 TDISCON ist die beim FB 65 TCON angegebene ID nicht mehr gültig und kann damit weder zum Senden noch zum Empfangen verwendet werden.

Über den Ausgangsparameter *BUSY* und den Ausgangsparameter *STATUS* wird der Zustand des Auftrags angezeigt. Dabei entspricht *STATUS* dem Ausgangsparameter *RET_VAL* der asynchron arbeitenden SFCs (siehe auch Bedeutung von *REQ*, *RET_VAL* und *BUSY* bei asynchron arbeitenden SFCs).

In der folgenden Tabelle ist der Zusammenhang zwischen *BUSY*, *DONE* und *ERROR* angegeben. Mit ihrer Hilfe können Sie feststellen, in welchem Zustand sich der FB 66 aktuell befindet bzw. wann der Verbindungsaufbau beendet ist.

BUSY	DONE	ERROR	Beschreibung
TRUE	irrelevant	irrelevant	Der Auftrag ist in Bearbeitung.
FALSE	TRUE	FALSE	Der Auftrag wurde erfolgreich abgeschlossen.
FALSE	FALSE	TRUE	Der Auftrag wurde mit einem Fehler beendet. Die Fehlerursache können Sie dem Parameter *STATUS* entnehmen.
FALSE	FALSE	FALSE	Dem FB wurde kein (neuer) Auftrag erteilt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L	Steuerparameter *REQ*, aktiviert den Abbau der durch *ID* gegebenen Verbindung. Der Anstoß erfolgt bei steigender Flanke.
ID	INPUT	WORD	M, D, Konstante	Referenz auf die abzubauende Verbindung zum remoten Partner bzw. zwischen Anwenderprogramm und Kommunikationsschicht des Betriebssystems. *ID* muss identisch sein mit dem zugehörigen Parameter *ID* in der lokalen Verbindungsbeschreibung. Wertebereich: 0001h ... 0FFFh
DONE	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *DONE*: ■ 0: Auftrag wurde noch nicht gestartet oder wird noch ausgeführt. ■ 1: Auftrag wurde fehlerfrei ausgeführt.
BUSY	OUTPUT	BOOL	E, A, M, D, L	■ *BUSY* = 1: Der Auftrag ist noch nicht beendet. ■ *BUSY* = 0: Der Auftrag ist beendet.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*: ■ *ERROR* = 1: Bei der Bearbeitung ist ein Fehler aufgetreten. *STATUS* liefert detaillierte Auskunft über die Art des Fehlers.
STATUS	OUTPUT	WORD	M, D	Zustandsparameter *STATUS*: Statusinformationen

Statusinformationen

ERROR	STATUS	Erläuterung
0	0000h	Verbindung konnte abgebaut werden
0	7000h	Erstaufruf mit *REQ* = 0, kein Anstoß zum Verbindungsabbau
0	7001h	Erstaufruf mit *REQ* = 1, Start der Auftragsbearbeitung, Verbindung wird abgebaut
0	7002h	Zwischenaufruf (*REQ* irrelevant), Verbindung wird abgebaut
1	8086h	Parameter *ID* liegt in einem unzulässigen Wertebereich
1	80A3h	Es wird versucht, eine nicht bestehende Verbindung abzubauen
1	80C4h	Temporärer Kommunikationsfehler: Schnittstelle wird neu parametriert

ERROR	STATUS	Erläuterung
1	8F7Fh	Interner Fehler (VIPA-spezifisch)
1	8xyyh	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

9.2 Ethernet-Kommunikation - *"Ethernet Communication"*

9.2.1 Kommunikation - FC 5...6 für CP 343

Die beiden Bausteine dienen der Verarbeitung von Verbindungsaufträgen auf SPS-Seite eines Ethernet-CP 343. Durch Einbindung dieser Bausteine in den Zyklus-Baustein OB1 können Sie zyklisch Daten senden und empfangen. Innerhalb dieser Bausteine werden die FCs 205 und 206 aufgerufen, die als Sonderfunktionsbausteine in der CPU abliegen.

Bitte beachten Sie, dass Sie in Ihrem Anwenderprogramm für die Kommunikation mit VIPA-CPs ausschließlich die VIPA-spezifischen SEND/RECV-FCs einsetzen dürfen. Bei Wechsel zu VIPA-CPs in einem schon bestehenden Projekt können die bestehenden AG_SEND / AG_LSEND bzw. AG_RECV / AG_LRECV durch die VIPA-spezifischen AG_SEND bzw. AG_RECV ohne Anpassung ersetzt werden. Da sich der CP automatisch an die Länge der zu übertragenden Daten anpasst ist die L-Variante von SEND bzw. RECV bei VIPA-CPs nicht erforderlich.

Kommunikationsbausteine

Für die Kommunikation zwischen CPU und Ethernet-CP 343 stehen Ihnen folgende FCs zur Verfügung:

- AG_SEND (FC 5)
  - Dieser Baustein übergibt die Nutzdaten aus dem über *SEND* angegebenen Datenbereich an den über *ID* und *LADDR* spezifizierten CP. Als Datenbereich können Sie einen PA-, Merker- oder Datenbaustein-Bereich angeben. Wurde der Datenbereich fehlerfrei übertragen, so wird "Auftrag fertig ohne Fehler" zurückgemeldet.
- AG_RECV (FC 6)
  - Der Baustein übernimmt vom CP die Nutzdaten und legt sie in dem über *RECV* definierten Datenbereich ab. Als Datenbereich können Sie einen PE-, Merker- oder Datenbaustein-Bereich angeben. Wurde der Datenbereich fehlerfrei übernommen, so wird "Auftrag fertig ohne Fehler" zurückgemeldet.

Statusanzeigen

Der CP bearbeitet Sende- und Empfangsaufträge unabhängig vom CPU Zyklus und benötigt hierzu eine Übertragungszeit. Die Schnittstelle mit den FC-Bausteinen zum Anwenderprogramm wird hierbei über Quittungen synchronisiert. Für die Statusauswertung liefern die Kommunikationsbausteine Parameter zurück, die Sie in Ihrem Anwenderprogramm direkt auswerten können. Diese Statusanzeigen werden bei jedem Baustein-Aufruf aktualisiert.

Einsatz unter hoher Kommunikationslast

Verwenden Sie keine zyklischen Aufrufe der Kommunikationsbausteine im OB 1. Dies führt zu einer ständigen Kommunikation zwischen CPU und CP. Programmieren Sie statt dessen Ihre Kommunikationsbausteine in einem Zeit-OB, deren Zykluszeit größer ist als die des OB1 bzw. ereignisgesteuert.

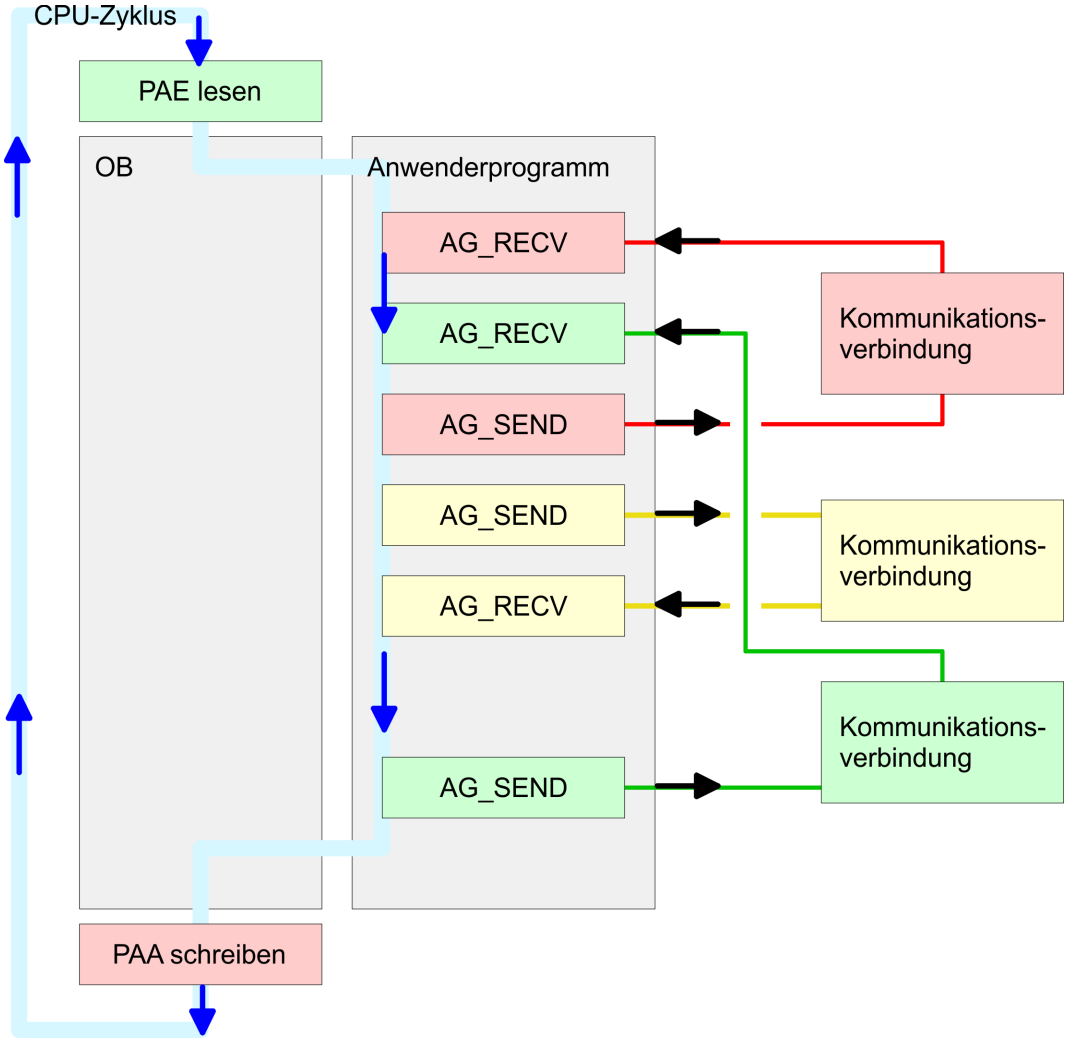
Aufruf FC schneller als CP-Übertragungszeit

Wird ein Baustein im Anwenderprogramm erneut aufgerufen, bevor die Daten vollständig gesendet oder empfangen wurden, wird an der Schnittstelle der FC-Bausteine wie folgt verfahren:

- AG_SEND
  - Es wird kein Auftrag entgegen genommen, bis die Datenübertragung über die Verbindung vom Partner quittiert wurde. Solange erhalten Sie die Meldung "Auftrag läuft", bis der CP den nächsten Auftrag für die gleiche Verbindung übernehmen kann.
- AG_RECV
  - Der Auftrag wird mit der Meldung "Es liegen noch keine Daten vor" quittiert, solange der CP die Empfangsdaten noch nicht vollständig empfangen hat.

AG_SEND, AG_RECV im Anwenderprogramm

Eine mögliche Ablaufsequenz für die FC-Bausteine zusammen mit den Organisations- und Programmbausteinen im CPU-Zyklus ist nachfolgend dargestellt:



Die FC-Bausteine mit zugehöriger Kommunikationsverbindung sind farblich zusammengefasst. Hier können Sie auch erkennen, dass Ihr Anwenderprogramm aus beliebig vielen Bausteinen bestehen kann. Somit können Sie ereignis- bzw. programmgesteuert an beliebiger Stelle im CPU-Zyklus mit AG_SEND Daten senden bzw. mit AG_RECV Daten empfangen. Sie können die Bausteine für **eine** Kommunikationsverbindung auch mehrmals in einem Zyklus aufrufen.

9.2.2 FC 5 - AG_SEND - Senden an CP 343

Mit AG_SEND werden die zu sendenden Daten von der CPU an einen Ethernet-CP übertragen.

Bitte beachten Sie, dass dieser Baustein intern den FC bzw. SFC 205 AG_SEND aufruft. Dieser darf nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB!

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
ACT	INPUT	BOOL	Aktivierung des Senders 0: Aktualisiert die *DONE, ERROR* und *STATUS* 1: Der unter *SEND* mit der Länge *LEN* abgelegte Datenbereich wird gesendet
ID	INPUT	INT	Verbindungsnummer 1 ... 16 (identisch mit dem gleichnamigen Parameter in *"Lokale ID"* unter *"Eigenschaften der Verbindung "* in *"Geräte und Netze"*)
LADDR	INPUT	WORD	Logische Basisadresse des CPs (identisch mit dem gleichnamigen Parameter in *"Lokale ID"* unter *"Eigenschaften der Verbindung "* in *"Geräte und Netze"*)
SEND	INPUT	ANY	Datenbereich
LEN	INPUT	INT	Anzahl der Bytes, die aus dem Datenbereich zu übertragen sind
DONE	OUTPUT	BOOL	Zustandsparameter für den Auftrag 0: Auftrag läuft 1: Auftrag fertig ohne Fehler
ERROR	OUTPUT	BOOL	Fehleranzeige 0: Auftrag läuft (bei *DONE* = 0) 0: Auftrag fertig ohne Fehler (bei *DONE* = 1) 1: Auftrag fertig mit Fehler
STATUS	OUTPUT	WORD	Statusanzeige, die in Verbindung mit *DONE* und *ERROR* zurückgeliefert wird. Näheres hierzu finden Sie in der nachfolgenden Tabelle.

DONE, ERROR, STATUS

In der nachfolgenden Tabelle sind alle Meldungen aufgeführt, die der Ethernet-CP nach einem SEND-Auftrag bzw. RECV-Auftrag zurückliefern kann. Ein "-" bedeutet, dass diese Meldung für den entsprechenden SEND- bzw. RECV-Auftrag nicht existiert.

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Beschreibung
1	-	0	0000h	Auftrag fertig ohne Fehler.
-	1	0	0000h	Neue Daten wurden ohne Fehler übernommen.
0	-	0	0000h	Kein Auftrag in Bearbeitung.

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Beschreibung
-	0	0	8180h	Es liegen noch keine Daten vor.
0	0	0	8181h	Auftrag läuft
0	0	1	8183h	Für diesen Auftrag gibt es keine CP-Projektierung.
0	-	1	8184h	Es ist ein Systemfehler aufgetreten.
-	0	1	8184h	Es ist ein Systemfehler aufgetreten (Quelldatenbereich fehlerhaft)
0	-	1	8185h	Parameter *LEN* größer als Quell-Bereich *SEND*.
	0	1	8185h	Ziel-Puffer (RECV) ist zu klein.
0	0	1	8186h	Parameter *ID* ungültig (nicht im Bereich 1 ... 16).
0	-	1	8302h	Keine Empfangsressourcen bei Ziel-Station, Empfänger-Station kann empfangene Daten nicht schnell genug verarbeiten bzw. hat keine Empfangsressourcen bereitgestellt.
0	-	1	8304h	Die Verbindung ist nicht aufgebaut. Der Sendeauftrag sollte erst nach einer Wartezeit > 100ms erneut abgesetzt werden.
-	0	1	8304h	Die Verbindung ist nicht aufgebaut. Der Empfangsauftrag sollte erst nach einer Wartezeit > 100ms erneut abgesetzt werden.
0	-	1	8311h	Zielstation ist unter der angegebenen Ethernet-Adresse nicht erreichbar.
0	-	1	8312h	Ethernet-Fehler im CP
0		1	8F22h	Quell-Bereich ungültig, wenn beispielsweise Bereich im DB nicht vorhanden Parameter *LEN* < 0.
-	0	1	8F23h	Quell-Bereich ungültig, wenn beispielsweise Bereich im DB nicht vorhanden Parameter *LEN* < 0.
0	-	1	8F24h	Bereichsfehler beim Lesen eines Parameters.
-	0	1	8F25h	Bereichsfehler beim Schreiben eines Parameters.
0	-	1	8F28h	Ausrichtungsfehler beim Lesen eines Parameters.
-	0	1	8F29h	Ausrichtungsfehler beim Schreiben eines Parameters.
-	0	1	8F30h	Parameter liegt im schreibgeschützten 1. akt. Datenbaustein
-	0	1	8F31h	Parameter liegt im schreibgeschützten 2. akt. Datenbaustein
0	0	1	8F32h	Parameter enthält zu große DB-Nummer.
0	0	1	8F33h	DB-Nummer Fehler
0	0	1	8F3Ah	Bereich nicht geladen (DB)
0	-	1	8F42h	Quittungsverzug beim Lesen eines Parameters aus dem Peripheriebereich.
-	0	1	8F43h	Quittungsverzug beim Schreiben eines Parameters in den Peripheriebereich.
0	-	1	8F44h	Adresse des zu lesenden Parameters in der Zugriffsspur gesperrt.

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Beschreibung
-	0	1	8F45h	Adresse des zu schreibenden Parameters in der Zugriffsspur gesperrt.
0	0	1	8F7Fh	Interner Fehler z.B. unzulässige ANY-Referenz z.B. Parameter *LEN* = 0.
0	0	1	8090h	Baugruppe mit dieser Baugruppen-Anfangsadresse nicht vorhanden oder CPU in STOP.
0	0	1	8091h	Baugruppen-Anfangsadresse nicht auf Doppel-Wort-Raster.
0	0	1	8092h	In ANY-Referenz ist eine Typangabe ungleich BYTE angegeben.
-	0	1	80A0h	Negative Quittung beim Lesen von Baugruppe.
0	0	1	80A4h	reserviert
0	0	1	80B0h	Baugruppe kennt den Datensatz nicht.
0	0	1	80B1h	Die Längenangabe (im Parameter *LEN*) ist falsch.
0	0	1	80B2h	reserviert
0	0	1	80C0h	Datensatz kann nicht gelesen werden.
0	0	1	80C1h	Der angegebene Datensatz ist gerade in Bearbeitung.
0	0	1	80C2h	Es liegt ein Auftragsstau vor.
0	0	1	80C3h	Die Betriebsmittel (Speicher) der CPU sind temporär belegt.
0	0	1	80C4h	Kommunikationsfehler (tritt temporär auf; daher ist eine Wiederholung im Anwenderprogramm sinnvoll).
0	0	1	80D2h	Baugruppen-Anfangsadresse ist falsch.

Status-Parameter bei Neuanlauf

Bei einem Neuanlauf des CP werden die Ausgabe-Parameter wie folgt zurückgesetzt:

- DONE = 0
- NDR = 0
- ERROR = 0
- STATUS = 8180h (bei AG_RECV)
- STATUS = 8181h (bei AG_SEND)

9.2.3 FC 6 - AG_RECV - Empfangen von CP 343

Mit dem 1. Aufruf von AG_RECV richten Sie einen Empfangspuffer zwischen der CPU und einem Ethernet CP 343 ein. Von jetzt ab werden empfangene Daten automatisch in diesem Puffer abgelegt. Sobald nach einem Aufruf von AG_RECV der Rückgabewert *NDR* = 1 zurückgeliefert wird, liegen gültige Daten ab. Da mit einem weiteren Aufruf von AG_RECV der Empfangspuffer für den Empfang neuer Daten wieder freigegeben wird, müssen Sie die zuvor empfangenen Daten sichern.

Bitte beachten Sie, dass dieser Baustein intern den FC bzw. SFC 206 AG_RECV aufruft. Dieser darf nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB!

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
ID	INPUT	INT	Verbindungsnummer 1 ... 16 (identisch mit dem gleichnamigen Parameter in *"Lokale ID"* unter *"Eigenschaften der Verbindung "* in *"Geräte und Netze"*)
LADDR	INPUT	WORD	Logische Basisadresse des CPs (identisch mit dem gleichnamigen Parameter in *"Lokale ID"* unter *"Eigenschaften der Verbindung "* in *"Geräte und Netze"*)
RECV	INPUT	ANY	Datenbereich für die empfangenen Daten.
NDR	OUTPUT	BOOL	Zustandsparameter für den Auftrag 0: Auftrag läuft 1: Auftrag fertig Daten wurden ohne Fehler übernommen
ERROR	OUTPUT	BOOL	Fehleranzeige 0: Auftrag läuft (bei *NDR* = 0) 0: Auftrag fertig ohne Fehler (*NDR* = 1) 1: Auftrag fertig mit Fehler
STATUS	OUTPUT	WORD	Statusanzeige, die in Verbindung mit *NDR* und *ERROR* zurückgeliefert wird. Näheres hierzu finden Sie in der nachfolgenden Tabelle.
LEN	OUTPUT	INT	Anzahl der Bytes, die empfangen wurden.

DONE, ERROR, STATUS

In der nachfolgenden Tabelle sind alle Meldungen aufgeführt, die der Ethernet-CP 343 nach einem SEND-Auftrag bzw. RECV-Auftrag zurückliefern kann.

Ein "-" bedeutet, dass diese Meldung für den entsprechenden SEND- bzw. RECV-Auftrag nicht existiert.

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Beschreibung
1	-	0	0000h	Auftrag fertig ohne Fehler.
-	1	0	0000h	Neue Daten wurden ohne Fehler übernommen.
0	-	0	0000h	Kein Auftrag in Bearbeitung.
-	0	0	8180h	Es liegen noch keine Daten vor.

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Beschreibung
0	0	0	8181h	Auftrag läuft
0	0	1	8183h	Für diesen Auftrag gibt es keine CP-Projektierung.
0	-	1	8184h	Es ist ein Systemfehler aufgetreten.
-	0	1	8184h	Es ist ein Systemfehler aufgetreten (Quelldatenbereich fehlerhaft)
0	-	1	8185h	Parameter *LEN* größer als Quell-Bereich *SEND*.
	0	1	8185h	Ziel-Puffer (RECV) ist zu klein.
0	0	1	8186h	Parameter *ID* ungültig (nicht im Bereich 1 ... 16).
0	-	1	8302h	Keine Empfangsressourcen bei Ziel-Station, Empfänger-Station kann empfangene Daten nicht schnell genug verarbeiten bzw. hat keine Empfangsressourcen bereitgestellt.
0	-	1	8304h	Die Verbindung ist nicht aufgebaut. Der Sendeauftrag sollte erst nach einer Wartezeit > 100ms erneut abgesetzt werden.
-	0	1	8304h	Die Verbindung ist nicht aufgebaut. Der Empfangsauftrag sollte erst nach einer Wartezeit > 100ms erneut abgesetzt werden.
0	-	1	8311h	Zielstation ist unter der angegebenen Ethernet-Adresse nicht erreichbar.
0	-	1	8312h	Ethernet-Fehler im CP
0		1	8F22h	Quell-Bereich ungültig, wenn beispielsweise Bereich im DB nicht vorhanden Parameter *LEN* < 0.
-	0	1	8F23h	Quell-Bereich ungültig, wenn beispielsweise Bereich im DB nicht vorhanden Parameter *LEN* < 0.
0	-	1	8F24h	Bereichsfehler beim Lesen eines Parameters.
-	0	1	8F25h	Bereichsfehler beim Schreiben eines Parameters.
0	-	1	8F28h	Ausrichtungsfehler beim Lesen eines Parameters.
-	0	1	8F29h	Ausrichtungsfehler beim Schreiben eines Parameters.
-	0	1	8F30h	Parameter liegt im schreibgeschützten 1. akt. Datenbaustein
-	0	1	8F31h	Parameter liegt im schreibgeschützten 2. akt. Datenbaustein
0	0	1	8F32h	Parameter enthält zu große DB-Nummer.
0	0	1	8F33h	DB-Nummer Fehler
0	0	1	8F3Ah	Bereich nicht geladen (DB)
0	-	1	8F42h	Quittungsverzug beim Lesen eines Parameters aus dem Peripheriebereich.
-	0	1	8F43h	Quittungsverzug beim Schreiben eines Parameters in den Peripheriebereich.
0	-	1	8F44h	Adresse des zu lesenden Parameters in der Zugriffsspur gesperrt.
-	0	1	8F45h	Adresse des zu schreibenden Parameters in der Zugriffsspur gesperrt.

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Beschreibung
0	0	1	8F7Fh	Interner Fehler z.B. unzulässige ANY-Referenz z.B. Parameter *LEN* = 0.
0	0	1	8090h	Baugruppe mit dieser Baugruppen-Anfangsadresse nicht vorhanden oder CPU in STOP.
0	0	1	8091h	Baugruppen-Anfangsadresse nicht auf Doppel-Wort-Raster.
0	0	1	8092h	In ANY-Referenz ist eine Typangabe ungleich BYTE angegeben.
-	0	1	80A0h	Negative Quittung beim Lesen von Baugruppe.
0	0	1	80A4h	reserviert
0	0	1	80B0h	Baugruppe kennt den Datensatz nicht.
0	0	1	80B1h	Die Längenangabe (im Parameter *LEN*) ist falsch.
0	0	1	80B2h	reserviert
0	0	1	80C0h	Datensatz kann nicht gelesen werden.
0	0	1	80C1h	Der angegebene Datensatz ist gerade in Bearbeitung.
0	0	1	80C2h	Es liegt ein Auftragsstau vor.
0	0	1	80C3h	Die Betriebsmittel (Speicher) der CPU sind temporär belegt.
0	0	1	80C4h	Kommunikationsfehler (tritt temporär auf; daher ist eine Wiederholung im Anwenderprogramm sinnvoll).
0	0	1	80D2h	Baugruppen-Anfangsadresse ist falsch.

Status-Parameter bei Neuanlauf

Bei einem Neuanlauf des CP werden die Ausgabe-Parameter wie folgt zurückgesetzt:

- DONE = 0
- NDR = 0
- ERROR = 0
- STATUS = 8180h (bei AG_RECV)
- STATUS = 8181h (bei AG_SEND)

9.2.4 FC 10 - AG_CNTRL - Control CP 343

Beschreibung

Mit dem FC 10 besteht die Möglichkeit, Verbindungen des Ethernet-CP 343 zu diagnostizieren bzw. einen neuen Verbindungsaufbau zu initialisieren.

Folgende Aktionen können über parametrierbare Kommandos ausgeführt werden:

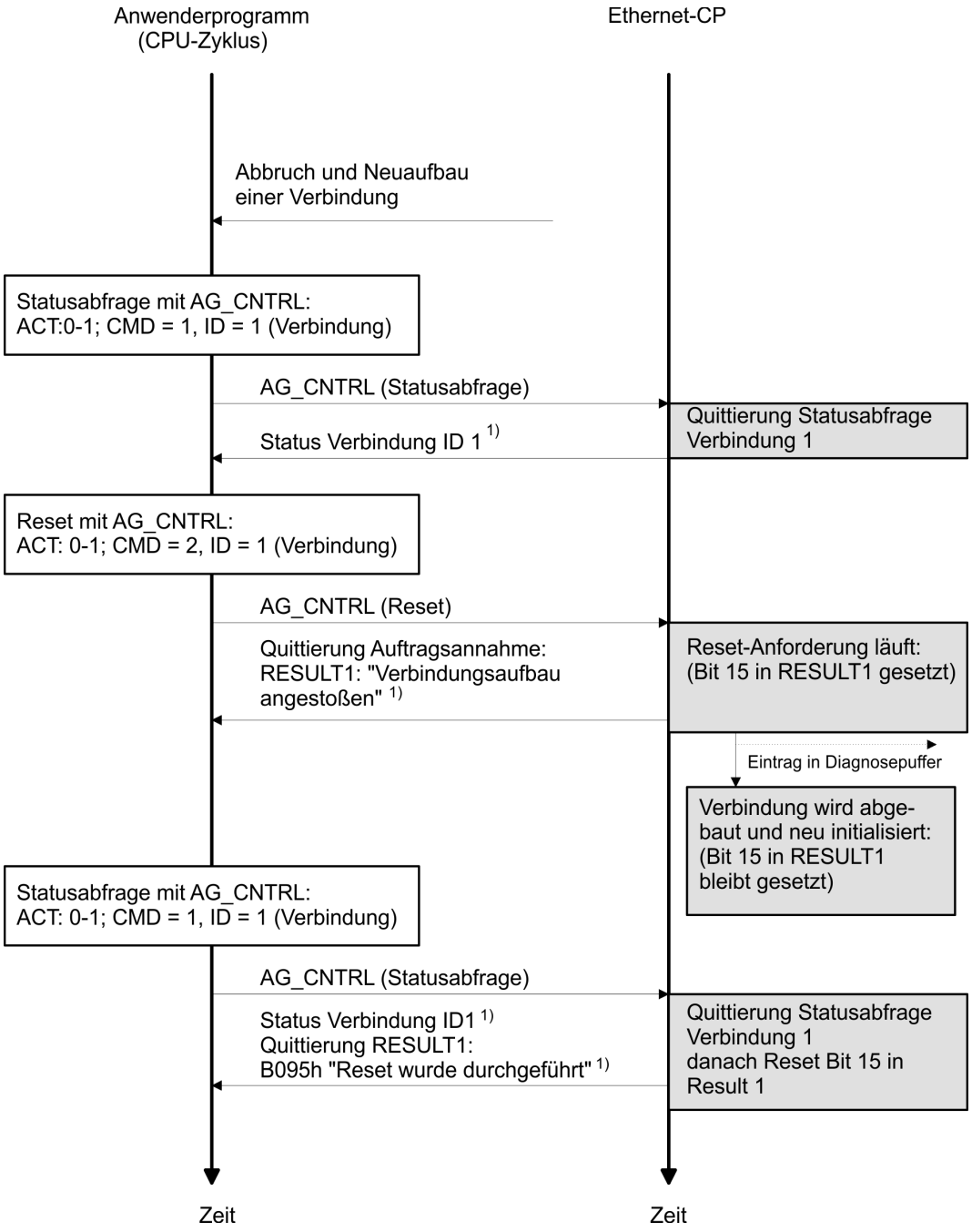
- Auslesen von Verbindungsinformationen
- Rücksetzen von projektierten Verbindungen

Die Kommandos dieses Bausteins sind nur für SEND-/RECV-Verbindungen zulässig, die auf die Protokolle ISO/RFC/TCP und UDP aufsetzen.

Bitte beachten Sie, dass dieser Baustein intern den FC bzw. SFC 196 AG_CNTRL aufruft. Dieser darf nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB!

FC 10 im Anwenderprogramm

Die folgende Darstellung zeigt einen typischen Ablauf von AG_CNTRL. Hier sehen Sie, wie zunächst der Verbindungsstatus abgefragt und in einem zweiten Auftrag der Verbindungsaufbau mittels Reset-Kommando veranlasst wird.



1) Parameterübergabe *DONE*, *ERROR*, *STATUS* und *RESULT1/2*

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
ACT	INPUT	BOOL	Auftrag wird durch Flankenwechsel 0-1 von *ACT* ausgeführt.
ID	INPUT	INT	Verbindungsnummer aus der Projektierung
LADDR	INPUT	WORD	Anfangsadresse des CP in der Hardware-Konfiguration

Parameter	Deklaration	Datentyp	Beschreibung
CMD	INPUT	INT	Kommando-Nr.
DONE	OUTPUT	BOOL	Zustandsparameter des Auftrags
ERROR	OUTPUT	BOOL	Fehler des Auftrags
STATUS	OUTPUT	WORD	Status des Auftrags
RESULT1	OUTPUT	DWORD	Rückmeldung 1 gemäß Kommando
RESULT2	OUTPUT	DWORD	Rückmeldung 2 gemäß Kommando

ACT

Wertebereich: 0, 1

Der Aufruf des FC muss mit Flankenwechsel 0-1 von *ACT* erfolgen.

Beim Aufruf mit *ACT* = 0 erfolgt kein Funktionsaufruf und der Baustein wird sofort wieder verlassen.

ID

Wertebereich: 1, 2 ... n, oder 0

Im Parameter *ID* wird die Verbindungsnummer der Verbindung angegeben. Die Verbindungsnummer ist der Projektierung zu entnehmen. Die maximale Anzahl der Verbindungen ist n.

Bei einem Aufruf, der alle Verbindungen anspricht, muss als *ID* 0 angegeben werden (_ALL-Funktion mit *CMD* 3 bzw. *CMD* 4).

LADDR

Baugruppen-Anfangsadresse

Bei der Konfiguration des CP im Hardware-Konfigurator wird die Baugruppen-Anfangsadresse in der Konfigurationstabelle ausgegeben.

Geben Sie diese Adresse hier an.

CMD

Kommandos an den FC AG_CNTRL

DONE

0: Auftrag ist noch in Bearbeitung bzw. noch nicht angestoßen

1: Auftrag ist ausgeführt

Der Zustandsparameter zeigt an, ob der Auftrag fehlerfrei abgewickelt wurde.

Bei *DONE* = 1 kann *RESULT* ausgewertet werden.

ERROR

0: kein Fehler

1: Fehleranzeige

STATUS

Statusanzeige

RESULT1/2

Rückmeldung gemäß Kommando an den FC AG_CNTRL

DONE, ERROR, STATUS

In der nachfolgenden Tabelle sind alle Meldungen aufgeführt, die der Ethernet-CP 343 nach einem AG_CNTRL zurückliefern kann.

Zusätzlich sind die Kommando-Ergebnisse in den Parametern *RESULT1* und *RESULT2* auszuwerten.

DONE	ERROR	STATUS	Beschreibung
1	0	0000h	Auftrag fertig ohne Fehler
0	0	0000h	Kein Auftrag in Bearbeitung
0	0	8181h	Auftrag läuft, der Bausteinaufruf ist mit den gleichen Parametern zu wiederholen, bis *DONE* oder *ERROR* gemeldet werden.
0	1	8183h	Für diesen Auftrag gibt es keine CP-Projektierung oder der Dienst im Ethernet-CP 343 ist noch nicht gestartet.
0	1	8186h	Parameter *ID* ungültig. Die zulässige *ID* ist abhängig vom gewählten Kommando.
0	1	8187h	Der Parameter *CMD* ist ungültig
0	1	8188h	Sequenzfehler bei der *ACT*-Steuerung
0	1	8090h	Baugruppe mit dieser Baugruppen-Anfangsadresse nicht vorhanden oder CPU in STOP.
0	1	8091h	Baugruppen-Anfangsadresse nicht auf Doppel-Wort Raster.
0	1	80B0h	Baugruppe kennt den Datensatz nicht.
0	1	80C0h	Datensatz kann nicht gelesen werden.
0	1	80C1h	Der angegebene Datensatz ist gerade in Bearbeitung.
0	1	80C2h	Es liegt ein Auftragsstau vor.
0	1	80C3h	Die Betriebsmittel (Speicher) der CPU sind temporär belegt.
0	1	80C4h	Kommunikationsfehler (tritt temporär auf; daher ist eine Wiederholung im Anwenderprogramm sinnvoll).
0	1	80D2h	Baugruppen-Anfangsadresse ist falsch.

Status-Parameter bei Neuanlauf

Bei einem Neuanlauf des CP werden die Ausgabe-Parameter wie folgt zurückgesetzt:

- *DONE* = 0
- *NDR* = 0
- *ERROR* = 8180h (bei AG_RECV)
- *ERROR* = 8181h (bei AG_SEND)

Hinweis!

Bitte beachten Sie, dass Sie den Baustein nur dann mit neuen Parametern aufrufen können, wenn ein zuvor gestarteter Auftrag mit DONE = 1 abgeschlossen wurde.

Kommandos und Auswertung der Auftragsergebnisse

In der nachfolgenden Tabelle finden Sie die möglichen Kommandos und die in den Parametern *RESULT1* und *RESULT2* auswertbaren Ergebnisse.

CMD 0

NOP - no operation

Der Baustein wird ohne Auftrag an den CP durchlaufen.

RESULT	Hex-Wert/Bereich	Beschreibung
RESULT 1	0000 0001h	Ablauf ohne Fehler
RESULT 2	0000 0000h	Default

CMD 1

CN_STATUS - connection status

Dieses Kommando liefert den Status der mit *ID* gewählten Verbindung des über *LADDR* ausgewählten CP. Sollte das Bit 15 (RESET-Kennung) gesetzt sein, wird dieses automatisch zurückgesetzt (dieses Verhalten entspricht CMD 5 - CN_CLEAR_RESET).

RESULT	Hex-Wert/Bereich	Beschreibung
RESULT 1	0000 000xh	Bit 3 ... 0: Anzeige Senderichtung (ausgeschlossen: 0010_b) Bit 0: Sende/Empfangsverbindung reserviert Bit 1: Sendeauftrag in Bearbeitung Bit 3, 2: Vorangegangener Auftrag 00: Keine Information 01: Sendeauftrag positiv abgeschlossen 10: Sendeauftrag negativ abgeschlossen
	0000 00x0h	Bit 7 ... 4: Anzeige Empfangsrichtung (ausgeschlossen: 0010_b) Bit 4: Sende/Empfangsverbindung reserviert Bit 5: Empfangsauftrag in Bearbeitung Bit 7, 6: Vorangegangener Auftrag 00: Keine Information 01: Empfangsauftrag positiv abgeschlossen 10: Empfangsauftrag negativ abgeschlossen

RESULT	Hex-Wert/Bereich	Beschreibung
	0000 0x00h	Bit 11 ... 8: Anzeigen für FETCH/WRITE (ausgeschlossen: 0011_b, 0111_b, 1000_b, 1011_b, 0010_b) Bit 8: Verbindungstyp 0: keine FETCH-Verbindung 1: Verbindung für FETCH-Aufträge reserviert Bit 9: Verbindungstyp 0: keine WRITE-Verbindung 1: Verbindung für WRITE-Aufträge reserviert Bit 10: Auftragsstatus (FETCH/ WRITE) 0: Auftragsstatus OK 1: Auftragsstatus nicht OK Diese Kennung wird in folgenden Fällen gesetzt: - Der Auftrag wurde von der CPU negativ quittiert - Der Auftrag wurde von der CPU negativ quittiert - Der Auftrag konnte nicht zur CPU weitergeleitet werden, weil sich die Verbindung im Zustand "LOCKED" befand. - Der Auftrag wurde abgelehnt, weil der FETCH/WRITE-Header nicht korrekt aufgebaut war. Bit 11: Status FETCH/WRITE-Auftrag 0: es läuft kein Auftrag 1: es läuft ein Auftrag von LAN
	0000 x000h	Bit 15 ... 12: Allgemeine CP-Informationen (ausgeschlossen: 0011_b, 1011_b) Bit 13, 12: Verbindungszustand (nur für SEND/RECV-Verbindungen verfügbar, die auf die Protokolle ISO/RFC/TCP aufsetzen; bei UDP werden die entsprechenden internen Informationen ausgegeben) 00: Verbindung ist abgebaut 01: Verbindungsaufbau läuft 10: Verbindungsabbau läuft 11: Verbindung ist aufgebaut Bit 14: CP-Information 0: CP in STOP 1: CP in RUN Bit 15: Reset-Kennung 0: Es wurde durch den FC 10 noch kein Verbindungs-Reset durchgeführt bzw. die Reset-Kennung wurde zurück genommen. 1: Es wurde durch den FC 10 ein Verbindungs-Reset durchgeführt.
	xxxx 0000h	Bit 31 ... 16: Reserviert für spätere Erweiterungen
RESULT 2	0000 0000h	Reserviert für spätere Erweiterungen

CMD 2

CN_RESET - connection reset

Dieses Kommando setzt die mit *ID* gewählte Verbindung des über *LADDR* ausgewählten CP zurück.

Das Rücksetzen der Verbindung bewirkt einen Verbindungsabbruch und einen erneuten Verbindungsaufbau (aktiv oder passiv, abhängig von der Projektierung).

Es wird zusätzlich ein Eintrag im Diagnosepuffer erzeugt, dem das Auftragsergebnis zu entnehmen ist.

RESULT	Hex-Wert/Bereich	Beschreibung
RESULT 1	0000 0001h	Der Reset-Auftrag wurde erfolgreich an den CP übertragen. Der Verbindungsabbruch und der anschließende Verbindungsaufbau wurden angestoßen.
	0000 0002h	Der Reset-Auftrag konnte nicht an den CP übertragen werden, da der Dienst im CP nicht gestartet ist (z.B. CP in STOP).
RESULT 2	0000 0000h	Default

CMD 3

CN_STATUS_ALL - all connections status

Dieses Kommando liefert in den Parametern *RESULT1/2* (insgesamt 8Byte Sammelinformation) den Verbindungsstatus aller Verbindungen (aufgebaut/abgebaut) des über *LADDR* ausgewählten CP.

Hierbei muss der Parameter *ID* auf "0" gesetzt sein (wird auf "0" überprüft).

Detailinformationen einer abgebauten oder nicht projektierten Verbindung erhält man bei Bedarf über einen erneuten, auf die Verbindung gezielten Status-Aufruf mit *CMD* = 1.

RESULT	Hex-Wert/Bereich	Beschreibung
RESULT 1	xxxx xxxxh	32 Bit: Verbindung 1 ... 32 0: Verbindung abgebaut / nicht projektiert 1: Verbindung aufgebaut
RESULT 2	xxxx xxxxh	32 Bit: Verbindung 33 ... 64 0: Verbindung abgebaut / nicht projektiert 1: Verbindung aufgebaut

CMD 4

CN_RESET_ALL - all connections reset

Dieses Kommando setzt alle Verbindungen des über *LADDR* angewählten CP zurück.

Der Parameter *ID* muss auf "0" gesetzt sein (wird auf "0" überprüft).

Das Rücksetzen der Verbindung bewirkt einen Verbindungsabbruch und einen erneuten Verbindungsaufbau (aktiv oder passiv, abhängig von der Projektierung).

Es wird zusätzlich ein Eintrag im Diagnosepuffer erzeugt, dem das Auftragsergebnis zu entnehmen ist.

RESULT	Hex-Wert/ Bereich	Beschreibung
RESULT 1	0000 0001h	Der Reset-Auftrag wurde erfolgreich an den CP übertragen. Der Verbindungsabbruch und der anschließende Verbindungsaufbau aller Verbindungen wurden angestoßen.
	0000 0002h	Der Reset-Auftrag konnte nicht an den CP übertragen werden, da der Dienst im CP nicht gestartet ist (z.B. CP in STOP).
RESULT 2	0000 0000h	Default

CMD 5

CN_CLEAR_RESET - clear the reset *ID*

Dieses Kommando setzt die Reset-Kennung (Bit 15 in RESULT1) für die mit *ID* gewählte Verbindung des über *LADDR* ausgewählten CP zurück.

Dieser Auftrag wird auch automatisch beim Lesen des Verbindungsstatus ausgeführt (*CMD* = 1); der hier beschriebene separat absetzbare Auftrag ist nur in Sonderfällen erforderlich.

RESULT	Hex-Wert/ Bereich	Beschreibung
RESULT 1	0000 0001h	Der Clear-Auftrag wurde erfolgreich an den CP übertragen.
	0000 0002h	Der Clear-Auftrag konnte nicht an den CP übertragen werden, da der Dienst im CP nicht gestartet ist (z.B. CP in STOP).
RESULT 2	0000 0000h	Default

CMD 6

CN_DISCON - connection disconnect

Dieses Kommando setzt die Verbindung zurück, die mit *ID* und *LADDR* ausgewählt wurde. Das Rücksetzen der Verbindung wird durch einen Verbindungsabbruch durchgeführt.

Eventuell im Stack gespeicherte Daten gehen ohne Hinweis verloren. Im Anschluss erfolgt kein automatischer Aufbau der Verbindung. Die Verbindung kann durch den Control-Auftrag CN_STARTCON wieder aufgebaut werden. Es wird ein Diagnosepuffereintrag erzeugt, dem Sie das Auftragsergebnis entnehmen können.

RESULT	Hex-Wert/ Bereich	Beschreibung
RESULT 1	0000 0001h	Der Auftrag wurde erfolgreich an den CP übertragen. Der Verbindungsabbruch wurde eingeleitet.
	0000 0002h	Der Auftrag konnte nicht an den CP übertragen werden, da der Dienst im CP nicht gestartet ist (z.B. CP in STOP).
RESULT 2	0000 0000h	Default

CMD 7 CN_STARTCON - start connection

Dieses Kommando baut eine Verbindung auf, die mit *ID* und *LADDR* ausgewählt und zuvor mit dem Control-Auftrag CN_DISCON abgebrochen wurde. Es wird ein Diagnosepuffereintrag erzeugt, dem Sie das Auftragsergebnis entnehmen können.

RESULT	Hex-Wert/ Bereich	Beschreibung
RESULT 1	0000 0001h	Der Auftrag wurde erfolgreich an den CP übertragen. Der Verbindungsaufbau wurde eingeleitet.
	0000 0002h	Der Auftrag konnte nicht an den CP übertragen werden, da der Dienst im CP nicht gestartet ist (z.B. CP in STOP).
RESULT 2	0000 0000h	Default

9.2.5 FC 62 - C_CNTR - Zustand einer Verbindung abfragen

Beschreibung

Mit dem FC 62 ermitteln Sie den Zustand einer Verbindung. Nach Aufruf der Systemfunktion mit dem Wert 1 am Steuereingang *EN_R* wird der momentane Zustand der über ID adressierten Verbindung ermittelt. Intern wird der SFC196 aufgerufen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN_R	INPUT	BOOL	E, A, M, D, L, Konst.	Steuerparameter enabled to receive, signalisiert Empfangsbereitschaft, wenn der Eingang gesetzt ist.
ID	INPUT	WORD	M, D, Konst.	Adressierungsparameter *ID*
RET_VAL	OUTPUT	INT	E, A, M, D, L	Fehlerinformation
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR* und *STATUS*
STATUS	OUTPUT	WORD	E, A, M, D, L	■ *ERROR*=0 und *STATUS* hat den Wert: – 0000h: weder Warnung noch Fehler – <> 0000h: Warnung, *STATUS* liefert detaillierte Auskunft. ■ *ERROR*=1 – Es liegt ein Fehler vor. *STATUS* liefert detaillierte Auskunft über die Art des Fehlers.
C_CONN	OUTPUT	BOOL	E, A, M, D, L	Zustand der zugehörigen Verbindung. Mögliche Werte: ■ 0: Verbindung abgebrochen oder nicht aufgebaut. ■ 1: Verbindung vorhanden.
C_STATUS	OUTPUT	WORD	E, A, M, D, L	Verbindungszustand: ■ W#16#0000: Verbindung ist nicht eingerichtet ■ W#16#0001: Verbindung wird gerade eingerichtet ■ W#16#0002: Verbindung ist eingerichtet ■ W#16#000F: Keine Daten für Verbindungsstatus verfügbar (z. B. beim CP-Anlauf) ■ W#16#00FF: Verbindung ist nicht projektiert

Fehlerinformationen

Der Ausgangsparameter *RET_VAL* kann bei der FC 62 C_CNTRL die folgenden beiden Werte annehmen:

- 0000h: Bei der Ausführung des FC ist kein Fehler aufgetreten.
- 8000h: Bei der Ausführung des FC ist ein Fehler aufgetreten.

Auch dann, wenn im Ausgangsparameter RET_VAL der Wert 0000h angezeigt wird, sind die Ausgangsparameter ERROR und STATUS auszuwerten.

ERROR	STATUS (dezimal)	Beschreibung
1	10	Fehler beim Zugriff auf den CP, weil gerade ein anderer Auftrag läuft. Auftragswiederholung zu einem späteren Zeitpunkt.
1	27	Für diesen Baustein existiert in der CPU kein Funktionscode.

9.2.6 FB/SFB 8 - FB 55 - Übersicht

Mit der Siemens S7-Kommunikation können Sie größere Datenmengen zwischen über Ethernet verbundenen SPS-Systemen auf Basis von Siemens STEP7® übertragen. Die Kommunikationsverbindungen sind statisch, d.h. sie sind über eine Verbindungstabelle zu projektieren.

Möglichkeiten der Kommunikationsfunktionen

- Siemens S7-300-Kommunikationsfunktionen
  - Durch Verwendung der VIPA-spezifischen Funktionsbausteine FB 8 ... FB 55 können Sie auf die Siemens S7-300-Kommunikationsfunktionen zugreifen.
- Siemens S7-400-Kommunikationsfunktionen
  - Für die Siemens S7-400-Kommunikationsfunktionen verwenden Sie die SFB 8... SFB 23, die im Betriebssystem der CPU integriert sind. Hierzu kopieren Sie die Schnittstellenbeschreibung der SFBs aus dem *"Katalog" "Bausteine"* in das Verzeichnis *"Systembausteine"* Ihres Projekts. Generieren Sie für jeden Aufruf einen Instanz-Datenbaustein und rufen den SFB mit dem zugehörigen Instanz-Datenbaustein auf.

Projektierung

Voraussetzung für die Siemens S7-Kommunikation ist eine projektierte Verbindungstabelle, in der die Kommunikationsverbindungen definiert werden. Eine Kommunikationsverbindung ist durch eine Verbindungs-ID für jeden Kommunikationspartner spezifiziert. Die lokale ID verwenden Sie für die Parametrierung des FB/SFB der SPS von der aus die Verbindung betrachtet wird und die Partner-ID für die Parametrierung des FB/SFB in der Partner-SPS.

Funktionsbausteine

FB/SFB	Bezeichnung	Beschreibung
FB/SFB 8	USEND	Unkoordiniertes Senden
FB/SFB 9	URCV	Unkoordiniertes Empfangen
FB/SFB 12	BSEND	Blockorientiertes Senden
FB/SFB 13	BRCV	Blockorientiertes Empfangen
FB/SFB 14	GET	Remote CPU lesen

FB/SFB	Bezeichnung	Beschreibung
FB/SFB 15	PUT	Remote CPU schreiben
FB 55	IP_CONF	Programmierbare Verbindungen

Bitte verwenden Sie für die Siemens S7-Kommunikation ausschließlich die hier aufgeführten FB/SFBs. Der direkte Aufruf der zugehörigen internen FC/SFCs führt zu Fehlern im entsprechenden Instanz-DB!

9.2.7 FB/SFB 8 - USEND - Unkoordiniertes Senden

Beschreibung

Mit dem FB/SFB 8 USEND können Daten an einen remoten Partner- FB/SFB vom Typ URCV (FB/SFB 9) gesendet werden, wobei darauf zu achten ist, dass der Parameter *R_ID* bei beiden FB/SFBs identisch ist. Der Sendevorgang wird mit einer positiven Flanke am Steuereingang *REQ* gestartet und verläuft ohne Koordination mit dem Partner-FB/SFB.

Je nach Kommunikationsfunktion haben Sie folgendes Verhalten:

- Siemens S7-300-Kommunikation (FB 8)
  - Der Sendevorgang erfolgt nach einer positiven Flanke an *REQ*. Mit jeder positiven Flanke an *REQ* werden die Parameter *R_ID*, *ID* und *SD_1* übernommen. Nach Abschluss eines Auftrags können Sie den Parametern *R_ID*, *ID* und *SD_1* neue Werte zuweisen.
- Siemens S7-400-Kommunikation (SFB 8)
  - Der Sendevorgang erfolgt nach einer positiven Flanke an *REQ*. Die zu sendenden Daten werden durch die Parameter *SD_1 ... SD_4* referenziert, wobei diese vier Sendeparameter nicht alle belegt sein müssen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L	Steuerparameter request, aktiviert den Datenaustausch bei steigender Flanke (gegenüber letztem FB/SFB-Aufruf)
ID	INPUT	WORD	E, A, M, D, Konstante	Referenz auf die Verbindung. *ID* muss in der Form wxyzh angegeben werden.
R_ID	INPUT	DWORD	E, A, M, D, L, Konstante	Adressierungsparameter *R_ID*. Ist in der Form DW#16#wxyzWXYZ anzugeben.
DONE	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *DONE*: ■ 0: Auftrag wurde noch nicht gestartet oder wird noch ausgeführt. ■ 1: Auftrag wurde fehlerfrei ausgeführt.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – Weder Warnung noch Fehler. ■ *ERROR* = 0 + *STATUS* ungleich 0000h – Es liegt eine Warnung vor. *STATUS* liefert detaillierte Auskunft. ■ *ERROR* = 1 – Es liegt ein Fehler vor.
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter *STATUS*, liefert detaillierte Auskunft über die Art des Fehlers.
SD_i,1≤ i ≤4	IN_OUT	ANY	E, A, M, D, T, Z	Zeiger auf i-ten Sendebereich. Zulässig sind nur die Datentypen: BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

Bitte beachten Sie , dass die über die Parameter SD_1/SD_1...SD_4 und RD_1/RD_1...RD_4 (beim zugehörigen Partner-FB/SFB URCV) definierten Bereiche übereinstimmen in Anzahl, Länge und Datentyp.

Der Parameter R_ID muss bei beiden FB/SFBs identisch sein. Der erfolgreiche Abschluss des Sendevorgangs wird am Zustandsparameter DONE mit einer logischen 1 angezeigt.

Fehlerinformationen

ERROR	STATUS (dezimal)	Bedeutung
0	11	Warnung: Neuer Auftrag ist unwirksam, da vorangegangener Auftrag noch nicht abgeschlossen ist.
0	25	Die Kommunikation wurde angestoßen. Der Auftrag ist in Bearbeitung.
1	1	Kommunikationsprobleme, z.B. ■ Verbindungsbeschreibung nicht geladen (lokal oder remote) ■ Verbindung unterbrochen (z.B. Kabel, CPU ausgeschaltet, CP in STOP)
1	4	Fehler in den Sendebereichszeigern *SD_i* bezüglich der Datenlänge oder des Datentyps.
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z.B. Zugriff auf gelöschten DB).
1	12	Beim Aufruf des FB/SFB wurde ■ ein Instanz-DB, der nicht zum FB/SFB 8 gehört, angegeben ■ kein Instanz-DB, sondern ein Global-DB angegeben ■ kein Instanz-DB gefunden (Laden eines neuen Instanz-DB vom PG)

ERROR	STATUS (dezimal)	Bedeutung
1	18	*R_ID* existiert bereits in der Verbindung *ID*.
1	20	Zu wenig Arbeitsspeicher vorhanden.

Datenkonsistenz

Um sicherzustellen das die Datenkonsistenz nicht beeinträchtigt wird, dürfen die aktuell benutzten Sendebereiche *SD_i* erst dann wieder beschrieben werden, wenn der aktuelle Sendevorgang abgeschlossen ist. Dazu muss der Parameter *DONE* ausgewertet werden. Der gesamte Sendevorgang ist erst dann abgeschlossen, wenn der Zustandsparameter *DONE* den Wert 1 annimmt.

9.2.8 FB/SFB 9 - URCV - Unkoordiniertes Empfangen

Beschreibung

Mit dem FB/SFB 9 URCV können Daten asynchron von einem remoten Partner-FB/SFB vom Typ USEND (FB/SFB 8) empfangen werden, wobei darauf zu achten ist, dass der Parameter *R_ID* bei beiden FB/SFBs identisch ist. Der Baustein ist empfangsbereit, wenn am Eingang *EN_R* eine logische 1 anliegt. Mit *EN_R*=0 kann ein laufender Auftrag abgebrochen werden.

Je nach Kommunikationsfunktion haben Sie folgendes Verhalten:

- Siemens S7-300-Kommunikation (FB 9)
  - Mit jeder positiven Flanke an *EN_R* werden die Parameter *R_ID, ID* und *RD_1* übernommen. Nach Abschluss eines Auftrags können Sie den Parametern *R_ID, ID* und *RD_1* neue Werte zuweisen.
- Siemens S7-400-Kommunikation (SFB 9)
  - Die Empfangsdatenbereiche werden durch die Parameter *RD_1...RD_4* referenziert.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN_R	INPUT	BOOL	E, A, M, D, L	Steuerparameter enabled to receive, signalisiert Empfangsbereitschaft
ID	INPUT	WORD	E, A, M, D, Konstante	Referenz auf die Verbindung. *ID* muss in der Form wxyzh angegeben werden.
R_ID	INPUT	DWORD	E, A, M, D, L, Konstante	Adressierungsparameter *R_ID*. Er muss in der Form DW#16#wxyzWXYZ angegeben werden.
NDR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *NDR*: neue Daten übernommen.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – Weder Warnung noch Fehler. ■ *ERROR* = 0 + *STATUS* ungleich 0000h – Es liegt eine Warnung vor. *STATUS* liefert detaillierte Auskunft. ■ *ERROR* = 1 – Es liegt ein Fehler vor.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter *STATUS*, liefert detaillierte Auskunft über die Art des Fehlers.
RD_i,1≤ i ≤4	IN_OUT	ANY	E, A, M, D, T, Z	Zeiger auf i-ten Empfangsbereich. Zulässig sind nur die Datentypen: BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

Es ist darauf zu achten, dass die über die Parameter SD_i und RD_i, 1 ≤ i ≤ 4, definierten Bereiche in Anzahl, Länge und im Datentyp übereinstimmen (SD_i ist der Sendebereich des zugehörigen Partner-FB/SFB, siehe FB/SFB 8). Beim Erstaufruf des FB/SFB 9 wird das "Empfangsfach" angelegt, bei den Nachfolgeaufrufen müssen die zu empfangenden Daten in dieses Empfangsfach hineinpassen. Wird der Kopiervorgang erfolgreich abgeschlossen, hat der Parameter NDR den Wert 1.

Fehlerinformationen

ERROR	STATUS (dezimal)	Beschreibung
0	9	Overrun-Warnung: Ältere Empfangsdaten wurden von neueren Empfangsdaten überschrieben.
0	11	Warnung: Neuer Auftrag ist unwirksam, da vorangegangener Auftrag noch nicht abgeschlossen ist.
0	25	Die Kommunikation wurde angestoßen. Der Auftrag ist in Bearbeitung.
1	1	Kommunikationsprobleme, z.B. ■ Verbindungsbeschreibung nicht geladen (lokal oder remote) ■ Verbindung unterbrochen (z.B. Kabel, CPU ausgeschaltet, CP in STOP)
1	4	Fehler in den Empfangsbereichszeigern *RD_i* bezüglich der Datenlänge oder des Datentyps.
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z.B. Zugriff auf gelöschten DB).
1	12	Beim Aufruf des FB/SFB wurde ■ ein Instanz-DB, der nicht zum FB/SFB 9 gehört, angegeben ■ kein Instanz-DB, sondern ein Global-DB angegeben ■ kein Instanz-DB gefunden (Laden eines neuen Instanz-DB vom PG)
1	18	*R_ID* existiert bereits in der Verbindung *ID*.
1	19	Der zugehörige FB/SFB USEND sendet schneller Daten als diese vom FB/SFB URCV in die Empfangsbereiche kopiert werden können.
1	20	Zu wenig Arbeitsspeicher vorhanden.

Datenkonsistenz

Die Daten werden konsistent empfangen, wenn Sie folgendes beachten:

- Siemens S7-300-Kommunikation:
  - Nachdem der Zustandsparameter *NDR* den Wert 1 angenommen hat, müssen Sie den FB 9 URCV mit dem Wert 0 an *EN_R* sofort erneut aufrufen. Damit wird gewährleistet, dass der Empfangsbereich nicht bereits überschrieben wird, bevor Sie ihn ausgewertet haben. Werten Sie den Empfangsbereich *RD_1* vollständig aus, bevor Sie den Baustein wieder mit dem Wert 1 am Steuereingang *EN_R* aufrufen.
- Siemens S7-400-Kommunikation:
  - Nachdem der Zustandsparameter *NDR* den Wert 1 angenommen hat, befinden sich neue Empfangsdaten in Ihren Empfangsbereichen (*RD_i*). Ein erneuter Bausteinaufruf kann diese Daten mit neuen Empfangsdaten überschreiben. Wenn Sie dies verhindern wollen, müssen Sie den SFB 9 URCV mit dem Wert 0 an *EN_R* so lange aufrufen (z.B. bei zyklischer Bausteinbearbeitung), bis Sie die Bearbeitung der Empfangsdaten beendet haben.

9.2.9 FB/SFB 12 - BSEND - Blockorientiertes Senden

Beschreibung

Mit dem FB/SFB 12 BSEND können Daten an einen remoten Partner- FB/SFB vom Typ BRCV (FB/SFB 13) gesendet werden. Der zu sendende Datenbereich wird segmentiert. Jedes Segment wird einzeln an den Partner gesendet. Das letzte Segment wird vom Partner bereits bei seiner Ankunft quittiert, unabhängig vom zugehörigen Aufruf des FB/SFB BRCV. Aufgrund der Segmentierung können Sie mit einem Sendeauftrag bis zu 65534Byte große Daten übertragen.

Bitte beachten Sie, dass dieser Baustein intern den FC bzw. SFC 202 AG_BSEND aufruft. Dieser darf nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB!

Je nach Kommunikationsfunktion haben Sie folgendes Verhalten:

- *Siemens S7-300-Kommunikation (FB 12)*
  - Der Sendevorgang erfolgt nach einer positiven Flanke an *REQ*. Mit jeder positiven Flanke an *REQ* werden die Parameter *R_ID, ID, SD_1* und *LEN* übernommen. Nach Abschluss eines Auftrags können Sie den Parametern *R_ID, ID, SD_1* und *LEN* neue Werte zuweisen. Zur Übertragung von segmentierten Daten ist der Baustein zyklisch im Anwenderprogramm aufzurufen. Die Anfangsadresse und die maximale Länge der zu sendenden Daten werden durch *SD_1* vorgegeben. Die Länge des Datenblocks legen Sie auftragsbezogen durch *LEN* fest.
- *Siemens S7-400-Kommunikation (SFB 12)*
  - Die Aktivierung des Sendevorgangs erfolgt nach Aufruf des Bausteins und positiver Flanke an *REQ*. Das Senden der Daten aus dem Anwenderspeicher erfolgt asynchron zur Bearbeitung des Anwenderprogramms. Die Anfangsadresse der zu sendenden Daten wird durch *SD_1* vorgegeben. Die Länge der Sendedaten legen Sie auftragsbezogen durch *LEN* fest. *LEN* ersetzt damit den Längenanteil von *SD_1*.

Funktion

- Bei einer positiven Flanke am Steuereingang *R* wird ein laufender Sendevorgang abgebrochen.
- Der erfolgreiche Abschluss des Sendevorgangs wird am Zustandsparameter *DONE* mit 1 angezeigt.

- Nach Abschluss eines Sendevorganges, kann erst wieder ein neuer Sendeauftrag bearbeitet werden, wenn die Zustandsparameter *DONE* oder *ERROR* den Wert 1 angenommen haben.
- Aufgrund der asynchronen Datenübertragung kann ein erneutes Senden von Daten erst gestartet werden, wenn die vorhergehenden Daten durch Aufruf des Partner-FB/SFB abgeholt wurden. Bis die Daten abgeholt werden, wird beim Aufruf des FB/SFB BSEND der Statuswert 7 ausgegeben.

Der Parameter R_ID muss bei den zusammengehörenden FB/SFBs identisch sein.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L	Steuerparameter request, aktiviert den Datenaustausch bei steigender Flanke (gegenüber letztem FB/SFB-Aufruf)
R	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter reset: Abbruch des aktuellen Auftrags
ID	INPUT	WORD	E, A, M, D, Konstante	Referenz auf die Verbindung. *ID* muss in der Form W#16#xxxx angegeben werden.
R_ID	INPUT	DWORD	E, A, M, D, L, Konstante	Adressierungsparameter *R_ID*. Er muss in der Form DW#16#wxyzWXYZ angegeben werden.
DONE	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *DONE*: ■ 0: Auftrag wurde noch nicht gestartet oder wird noch ausgeführt. ■ 1: Auftrag wurde fehlerfrei ausgeführt.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – Weder Warnung noch Fehler. ■ *ERROR* = 0 + *STATUS* ungleich 0000h – Es liegt eine Warnung vor. *STATUS* liefert detaillierte Auskunft. ■ *ERROR* = 1 – Es liegt ein Fehler vor.
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter *STATUS*, liefert detaillierte Auskunft über die Art des Fehlers.
SD_1	IN_OUT	ANY	E, A, M, D, T, Z	Zeiger auf Sendebereich. Die Längenangabe wird nur beim ersten Bausteinaufruf nach dem Start ausgewertet. Sie gibt die maximale Länge des Sendebereichs vor. Zulässig sind nur die Datentypen: BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.
LEN	IN_OUT	WORD	E, A, M, D, L	Länge des zu sendenden Datenblocks in Byte.

Fehlerinformationen

ERROR	STATUS (dezimal)	Bedeutung
0	11	Warnung: Neuer Auftrag ist unwirksam, da vorangegangener Auftrag noch nicht abgeschlossen ist.
0	25	Die Kommunikation wurde angestoßen. Der Auftrag ist in Bearbeitung.
1	1	Kommunikationsprobleme, z.B.: ■ Verbindungsbeschreibung nicht geladen (lokal oder remote) ■ Verbindung unterbrochen (z.B. Kabel, CPU ausgeschaltet, CP in STOP)
1	2	Negative Quittung vom Partner-FB/SFB. Die Funktion ist nicht ausführbar.
1	3	*R_ID* ist auf der durch *ID* vorgegebenen Kommunikationsverbindung nicht bekannt, oder der Empfangsbaustein wurde noch nie aufgerufen.
1	4	Fehler im Sendebereichszeiger *SD_1* bezüglich der Datenlänge oder des Datentyps, oder am Parameter *LEN* wurde der Wert 0 übergeben oder Fehler im Empfangsbereichszeiger *RD_1* des zugehörigen FB/SFB 13 BRCV
1	5	Resetanforderung wurde ausgeführt.
1	6	Partner-FB/SFB befindet sich im Zustand DISABLED (*EN_R* hat den Wert 0).
1	7	Partner-FB/SFB befindet sich in falschem Zustand. (Der Empfangsbaustein wurde nach der letzten Datenübertragung nicht mehr aufgerufen).
1	8	Zugriff auf remotes Objekt im Anwenderspeicher wurde abgelehnt.
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z.B. Zugriff auf gelöschten DB).
1	12	Beim Aufruf des FB/SFB wurde ■ ein Instanz-DB, der nicht zum FB/SFB 12 gehört, angegeben ■ kein Instanz-DB, sondern ein Global-DB angegeben ■ kein Instanz-DB gefunden (Laden eines neuen Instanz-DB vom PG)
1	18	*R_ID* existiert bereits in der Verbindung *ID*.
1	20	Zu wenig Arbeitsspeicher vorhanden.

Datenkonsistenz

Um sicherzustellen das die Datenkonsistenz nicht beeinträchtigt wird, darf der aktuell benutzte Teil des Sendebereichs *SD_1* erst dann wieder beschrieben werden, wenn der aktuelle Sendevorgang abgeschlossen ist. Dazu muss der Parameter *DONE* ausgewertet werden.

9.2.10 FB/SFB 13 - BRCV - Blockorientiertes Empfangen

Beschreibung

Mit dem FB/SFB 13 BRCV können Daten von einem remoten Partner- FB/SFB vom Typ BSEND (FB/SFB 12) empfangen werden, wobei darauf zu achten ist, dass der Parameter *R_ID* bei beiden FB/SFBs identisch ist. Nach jedem empfangenen Datensegment wird eine Quittung an den Partner-FB/SFB geschickt, und der Parameter LEN aktualisiert.

Bitte beachten Sie, dass dieser Baustein intern den FC bzw. SFC 203 AG_BRCV aufruft. Dieser darf nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB!

Je nach Kommunikationsfunktion haben Sie folgendes Verhalten:

- *Siemens S7-300-Kommunikation (FB 13)*
  - Mit jeder positiven Flanke an *EN_R* werden die Parameter *R_ID, ID* und *RD_1* übernommen. Nach Abschluss eines Auftrags können Sie den Parametern *R_ID, ID* und *RD_1* neue Werte zuweisen. Zur Übertragung von segmentierten Daten muss der Baustein zyklisch im Anwenderprogramm aufgerufen werden.
- *Siemens S7-400-Kommunikation (SFB 13)*
  - Der Empfang der Daten aus dem Anwenderspeicher erfolgt asynchron zur Bearbeitung des Anwenderprogramms.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN_R	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter enabled to receive, signalisiert Empfangsbereitschaft
ID	INPUT	WORD	E, A, M, D, Konstante	Referenz auf die Verbindung. *ID* muss in der Form W#16#xxxx angegeben werden.
R_ID	INPUT	DWORD	E, A, M, D, L, Konstante	Adressierungsparameter *R_ID*. Er muss in der Form DW#16#wxyzWXYZ angegeben werden.
NDR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *NDR*: neue Daten übernommen.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – Weder Warnung noch Fehler. ■ *ERROR* = 0 + *STATUS* ungleich 0000h – Es liegt eine Warnung vor. *STATUS* liefert detaillierte Auskunft. ■ *ERROR* = 1 – Es liegt ein Fehler vor.
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter *STATUS*, liefert detaillierte Auskunft über die Art des Fehlers.
RD_1	IN_OUT	ANY	E, A, M, D, T, Z	Zeiger auf Empfangsbereich. Die Längenangabe gibt die maximale Länge des zu empfangenden Blocks vor. Zulässig sind nur die Datentypen: BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.
LEN	IN_OUT	WORD	E, A, M, D, L	Länge der bisher empfangenen Daten in Byte.

Funktion

- Der FB/SFB 13 ist empfangsbereit, wenn am Steuereingang *EN_R* der Wert 1 anliegt. Durch den Parameter *RD_1* wird die Anfangsadresse des Empfangsbereichs angegeben. Der Partner-FB/SFB erhält nach jedem empfangenen Datensegment eine Quittung und der Parameter *LEN* des FB/SFB 13 wird aktualisiert. Wenn der Baustein während des asynchronen Empfangs aufgerufen wird, erfolgt die Ausgabe einer Warnung im Zustandsparameter *STATUS*.
- Wenn dieser Aufruf mit dem Wert 0 am Steuereingang *EN_R* erfolgt, wird der Empfangsvorgang abgebrochen, und der FB/SFB in seinen Grundzustand zurückversetzt. Wird der fehlerfreie Empfang sämtlicher Datensegmente erfolgreich abgeschlossen, hat der Parameter *NDR* den Wert 1. Die empfangenen Daten bleiben solange unverändert, bis der FB/SFB 13 erneut mit dem Parameter *EN_R* = 1 aufgerufen wird.

Fehlerinformationen

ERROR	STATUS (dezimal)	Bedeutung
0	11	Warnung: Neuer Auftrag ist unwirksam, da vorangegangener Auftrag noch nicht abgeschlossen ist.
0	17	Warnung: Baustein empfängt asynchron Daten.
0	25	Die Kommunikation wurde angestoßen. Der Auftrag ist in Bearbeitung.
1	1	Kommunikationsprobleme, z.B. ■ Verbindungsbeschreibung nicht geladen (lokal oder remote) ■ Verbindung unterbrochen (z.B. Kabel, CPU ausgeschaltet, CP in STOP)
1	2	Funktion nicht ausführbar.
1	4	Fehler im Empfangsbereichszeiger *RD_1* bezüglich der Datenlänge oder des Datentyps (gesendeter Datenblock ist länger als der Empfangsbereich).
1	5	Resetanforderung eingetroffen, unvollständige Übertragung.
1	8	Zugriff auf remotes Objekt im Anwenderspeicher wurde abgelehnt.
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z.B. Zugriff auf gelöschten DB).
1	12	Beim Aufruf des FB/SFB wurde ■ ein Instanz-DB, der nicht zum FB/SFB 13 gehört, angegeben ■ kein Instanz-DB, sondern ein Global-DB angegeben ■ kein Instanz-DB gefunden (Laden eines neuen Instanz-DB vom PG)
1	18	*R_ID* existiert bereits in der Verbindung *ID*.
1	20	Zu wenig Arbeitsspeicher vorhanden.

Datenkonsistenz

Die Daten werden konsistent empfangen, wenn Sie folgendes beachten:

- Nach Abschluss des Kopiervorganges (der Parameter *NDR* hat den Wert 1), muss der FB/SFB 13 erneut mit dem Wert 0 am Parameter *EN_R* aufgerufen werden, um zu gewährleisten, dass der Empfangsbereich nicht bereits wieder überschrieben wird, bevor er ausgewertet wurde.
- Der zuletzt benutzte Empfangsbereich *RD_1* muss vollständig ausgewertet werden, bevor der Baustein wieder empfangsbereit gemacht werden kann (Aufruf mit dem Wert 1 am Parameter *EN_R*).

Datenempfang Siemens S7-400

- Falls die Empfänger-CPU mit einem empfangsbereiten BRCV-Baustein (d.h. ein Aufruf mit *EN_R* =1 ist bereits erfolgt) in STOP geht, bevor der zugehörige Sendebaustein das erste Datensegment eines Auftrags abgeschickt hat, geschieht folgendes:
- Die Daten des ersten Auftrags nach Übergang der Empfänger-CPU in STOP werden vollständig in den Empfangsbereich eingetragen,
- Der Partner-SFB BSEND erhält darüber eine positive Quittung.
- Weitere BSEND-Aufträge können von der Empfänger-CPU im STOP-Zustand nicht mehr angenommen werden.
- Solange sie sich im STOP-Zustand befindet, haben *NDR* und *LEN* den Wert 0.
- Damit Ihnen die Information über die empfangenen Daten nicht verloren geht, müssen Sie bei der Empfänger-CPU einen Wiederanlauf durchführen und die SFB 13 BRCV mit *EN_R* = 1 aufrufen.

9.2.11 FB/SFB 14 - GET - Remote CPU lesen

Beschreibung

Mit dem FB/SFB 14 GET können Daten aus einer remoten CPU ausgelesen werden, wobei sich die CPU im Betriebszustand RUN oder STOP befinden kann.

Bitte beachten Sie, dass dieser Baustein intern den FC bzw. SFC 200 AG_GET aufruft. Dieser darf nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB!

Je nach Kommunikationsfunktion haben Sie folgendes Verhalten:

- *Siemens S7-300-Kommunikation (FB 14)*
  - Der Lesevorgang erfolgt nach einer positiven Flanke an *REQ*. Mit jeder positiven Flanke an *REQ* werden die Parameter *ID, ADDR_1* und *RD_1* übernommen. Nach Abschluss eines Auftrags können Sie den Parametern *ID, ADDR_1* und *RD_1* neue Werte zuweisen.
- *Siemens S7-400-Kommunikation (SFB 14)*
  - Bei einer positiven Flanke an *REQ* wird der SFB gestartet. Dabei werden die relevanten Zeiger auf die auszulesenden Bereiche (*ADDR_i*) an die Partner-CPU gesendet.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L	Steuerparameter request, aktiviert den Datenaustausch bei steigender Flanke (gegenüber letztem FB/SFB-Aufruf)
ID	INPUT	WORD	E, A, M, D, Konstante	Referenz auf die Verbindung. *ID* muss in der Form W#16#xxxx angegeben werden.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
NDR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *NDR*: Daten aus Partner-CPU übernommen.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – Weder Warnung noch Fehler. ■ *ERROR* = 0 + *STATUS* ungleich 0000h – Es liegt eine Warnung vor. *STATUS* liefert detaillierte Auskunft. ■ *ERROR* = 1 – Es liegt ein Fehler vor.
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter *STATUS*, liefert detaillierte Auskunft über die Art des Fehlers.
ADDR_1	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, die gelesen werden sollen
ADDR_2	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, die gelesen werden sollen
ADDR_3	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, die gelesen werden sollen
ADDR_4	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, die gelesen werden sollen
RD_i,1≤ i ≤4	IN_OUT	ANY	E, A, M, D, T, Z	Zeiger auf diejenigen Bereiche in der eigenen CPU, in der die gelesenen Daten abgelegt werden. Zulässig sind nur die Datentypen: BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

Funktion

- Die Remote-CPU sendet die Dateninhalte zurück und die Antwort wird daraufhin auf Zugriffsprobleme beim Lesen der Daten ausgewertet, des weiteren wird eine Datentypprüfung vorgenommen.
- Bei einer fehlerfreien Datenübertragung werden die empfangenen Daten beim nächsten Aufruf des FB/SFB 14 in die projektierten Empfangsbereiche (*RD_i*) kopiert und der Parameter *NDR* erhält den Wert 1.
- Ein erneuter Lesevorgang kann erst dann wieder aktiviert werden, wenn der letzte Lesevorgang abgeschlossen wurde. Sie müssen darauf achten, dass die über die Parameter *ADDR_i* und *RD_i* definierten Bereiche in der Anzahl, in der Länge und im Datentyp zueinander passen.

Fehlerinformationen

ERROR	STATUS (dezimal)	Bedeutung
0	11	Warnung: Neuer Auftrag ist unwirksam, da voran-gegangener Auftrag noch nicht abgeschlossen ist.
0	25	Die Kommunikation wurde angestoßen. Der Auftrag ist in Bearbeitung.

ERROR	STATUS (dezimal)	Bedeutung
1	1	Kommunikationsprobleme, z.B. ■ Verbindungsbeschreibung nicht geladen (lokal oder remote) ■ Verbindung unterbrochen (z.B.: Kabel, CPU ausgeschaltet, CP in STOP)
1	2	Negative Quittung vom Partnergerät. Die Funktion ist nicht ausführbar.
1	4	Fehler in den Empfangsbereichszeigern *RD_i* bezüglich der Datenlänge oder des Datentyps.
1	8	Zugriffsfehler bei der Partner-CPU
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z.B. Zugriff auf gelöschten DB).
1	12	Beim Aufruf des FB/SFB wurde ■ ein Instanz-DB, der nicht zum FB/SFB 14 gehört, angegeben ■ kein Instanz-DB, sondern ein Global-DB angegeben ■ kein Instanz-DB gefunden (Laden eines neuen Instanz-DB vom PG)
1	20	Zu wenig Arbeitsspeicher vorhanden.

Datenkonsistenz

Die Daten werden konsistent empfangen, wenn Sie den aktuell verwendeten des Empfangsbereichs *RD_i* vollständig auswerten, bevor Sie einen erneuten Auftrag aktivieren.

9.2.12 FB/SFB 15 - PUT - Remote CPU schreiben

Beschreibung

Mit dem FB/SFB 15 PUT können Daten in eine remote CPU geschrieben werden, wobei sich die CPU im Betriebszustand RUN oder STOP befinden kann.

Bitte beachten Sie, dass dieser Baustein intern den FC bzw. SFC 201 AG_PUT aufruft. Dieser darf nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB!

Je nach Kommunikationsfunktion haben Sie folgendes Verhalten:

- *Siemens S7-300-Kommunikation (FB 15)*
  - Der Sendevorgang erfolgt nach einer positiven Flanke an *REQ*. Mit jeder positiven Flanke an *REQ* werden die Parameter *ID, ADDR_1* und *SD_1* übernommen. Nach Abschluss eines Auftrags können Sie den Parametern *ID, ADDR_1* und *SD_1* neue Werte zuweisen.
- *Siemens S7-400-Kommunikation (SFB 15)*
  - Bei einer positiven Flanke an *REQ* wird der SFB gestartet. Dabei werden die Zeiger auf die zu schreibenden Bereiche (*ADDR_i*) und die Daten (*SD_i*) an die Partner-CPU gesendet.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L	Steuerparameter request, aktiviert den Datenaustausch bei steigender Flanke (gegenüber letztem FB/SFB-Aufruf)
ID	INPUT	WORD	E, A, M, D, Konstante	Referenz auf die Verbindung. *ID* muss in der Form W#16#xxxx angegeben werden.
DONE	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *DONE*: Funktion ausgeführt.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – Weder Warnung noch Fehler. ■ *ERROR* = 0 + *STATUS* ungleich 0000h – Es liegt eine Warnung vor. *STATUS* liefert detaillierte Auskunft. ■ *ERROR* = 1 – Es liegt ein Fehler vor.
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter STATUS, liefert detaillierte Auskunft über die Art des Fehlers.
ADDR_1	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, in die geschrieben werden soll
ADDR_2	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, in die geschrieben werden soll
ADDR_3	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, in die geschrieben werden soll
ADDR_4	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, in die geschrieben werden soll
SD_i,1≤i ≤4	IN_OUT	ANY	E, A, M, D, T, Z	Zeiger auf diejenigen Bereiche in der eigenen CPU, die die zu versendenden Daten enthalten. Zulässig sind nur die Datentypen BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

Funktion

- Die Partner-CPU legt die gesendeten Daten unter den mitgeführten Adressen ab und sendet eine Ausführungsquittung zurück.
- Diese Ausführungsquittung wird ausgewertet und bei einer fehlerfreien Datenübertragung erhält der Parameter *DONE* beim nächsten Aufruf des FB/SFB 15 den Wert 1.
- Ein erneuter Schreibvorgang kann erst dann wieder aktiviert werden, wenn der letzte Schreibvorgang abgeschlossen wurde. Die über die Parameter *ADDR_i* und *SD_i,* 1 ≤ i ≤ 4, definierten Bereiche müssen in Anzahl, Länge und im Datentyp übereinstimmen.

Fehlerinformationen

ERROR	STATUS (dezimal)	Bedeutung
0	11	Warnung: Neuer Auftrag ist unwirksam, da vorangegangener Auftrag noch nicht abgeschlossen ist.
0	25	Die Kommunikation wurde angestoßen. Der Auftrag ist in Bearbeitung.

ERROR	STATUS (dezimal)	Bedeutung
1	1	Kommunikationsprobleme, z.B. ■ Verbindungsbeschreibung nicht geladen (lokal oder remote) ■ Verbindung unterbrochen (z.B.: Kabel, CPU ausgeschaltet, CP in STOP)
1	2	Negative Quittung vom Partnergerät. Die Funktion ist nicht ausführbar.
1	4	Fehler in den Sendebereichszeigern *SD_i* bezüglich der Datenlänge oder des Datentyps.
1	8	Zugriffsfehler bei der Partner-CPU
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z.B. Zugriff auf gelöschten DB).
1	12	Beim Aufruf des FB/SFB wurde ein Instanz-DB, der nicht zum FB/SFB 15 gehört, angegeben. kein Instanz-DB, sondern ein Global-DB angegeben. kein Instanz-DB gefunden (laden eines neuen Instanz-DB vom PG).
1	20	Zu wenig Arbeitsspeicher vorhanden.

Datenkonsistenz

- *Siemens S7-300-Kommunikation*
  - Um Datenkonsistenz zu gewährleisten, dürfen Sie den Sendebereich *SD_1* erst dann wieder beschreiben, wenn der aktuelle Sendevorgang abgeschlossen ist. Dies ist der Fall, wenn der Zustandsparameter *DONE* den Wert "1" annimmt.
- *Siemens S7-400-Kommunikation*
  - Mit dem Aktivieren eines Sendevorgangs (steigende Flanke an *REQ*) sind die zu sendenden Daten der Sendebereiche *SD_i* aus dem Anwenderprogramm kopiert. Sie können diese Bereiche nach dem Bausteinaufruf neu beschreiben, ohne die aktuellen Sendedaten zu verfälschen.

9.2.13 FB 55 - IP_CONF - Progr. Kommunikationsverbindungen

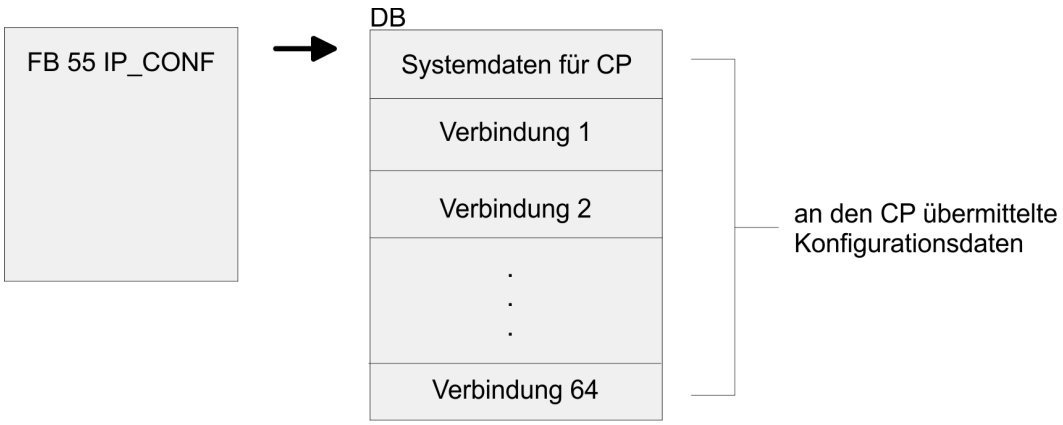
9.2.13.1 Übersicht

Der FB 55 - IP_CONF ermöglicht zur Einrichtung flexibler Kommunikationsverbindungen die programmgesteuerte Übergabe von Datenbausteinen mit Projektierdaten an einen CP.

Bitte beachten Sie, dass dieser Baustein intern den FC bzw. SFC 204 IP_CONF aufruft. Dieser darf nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB!

Prinzip

Über den FB 55, der im Anwenderprogramm aufgerufen wird, können Konfigurationsdaten für Kommunikationsverbindungen an den CP übertragen werden. Der Konfigurations-DB kann jederzeit in den CP geladen werden.



VORSICHT!
Sobald das Anwenderprogramm über den FB 55 IP_CONF die Verbindungsdaten übergibt, schaltet die CPU den CP kurzzeitig in STOP. Der CP übernimmt die Systemdaten (inklusive IP-Adresse) und die neuen Verbindungsdaten und arbeitet diese im Anlauf ab (RUN).

9.2.13.2 FB 55 - IP_CONF

Je nach Größe des Konfigurations-DB erfolgt die Übertragung zum CP in mehreren Segmenten. Sie müssen daher den FB solange erneut aufrufen, bis der FB mit dem *DONE*-Bit = 1 die vollständige Übertragung signalisiert. Der Auftrag wird ausgeführt, sobald *ACT* = 1 übergeben wird.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
ACT	INPUT	BOOL	E, A, M, D, L	■ Beim FB-Aufruf mit *ACT* = 1 wird der DBxx an den CP gesendet. ■ Beim FB-Aufruf mit *ACT* = 0 werden nur die Statusanzeigen *DONE, ERROR* und *STATUS* aktualisiert.
LADDR	INPUT	WORD	E, A, M, D, Konstante	Baugruppen-Anfangsadresse Bei der Hardware-Konfiguration wird die Baugruppen-Anfangsadresse in der Konfigurationstabelle ausgegeben. Geben Sie diese Adresse hier an.
CONF_DB	INPUT	ANY	E, A, M, D	Der Parameter zeigt die Anfangsadresse des Konfigurations-Datenbereichs in einem DB.
LEN	INPUT	INT	E, A, M, D, Konstante	Längenangabe in Byte für den Konfigurations-Datenbereich.
DONE	OUTPUT	BOOL	E, A, M, D, L	Der Parameter zeigt an, ob der Konfigurations-Datenbereich vollständig übertragen wurde. Beachten Sie, dass der FB je nach Größe des Konfigurations-Datenbereichs (in mehreren Zyklen) mehrfach angestoßen werden muss, bis die Anzeige *DONE* = 1 den Abschluss signalisiert.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Fehleranzeige

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
STATUS	OUTPUT	WORD	E, A, M, D	Statusanzeige
EXT_STATUS	OUTPUT	WORD	E, A, M, D	Der Parameter zeigt bei einer fehlerhaften Auftragsausführung an, welcher Parameter im Konfigurations-DB als Fehlerursache erkannt wurde. ■ High-Byte: Index des Parameter-Blocks ■ Low-Byte: Index des Subblocks innerhalb des Parameterblocks

Fehlerinformationen

ERROR	STATUS	Beschreibung
0	0000h	Auftrag fertig ohne Fehler
0	8181h	Auftrag läuft
1	80B1h	Anzahl der zu sendenden Daten überschreitet die für diesen Dienst zulässige Obergrenze.
1	80C4h	Kommunikationsfehler Fehler kann temporär auftreten, deshalb ist eine Wiederholung im Anwenderprogramm sinnvoll.
1	80D2h	Projektierungsfehler, die eingesetzte Baugruppe unterstützt diesen Dienst nicht.
1	8183h	CP lehnt die angeforderte Datensatznummer ab.
1	8184h	Systemfehler bzw. unzulässiger Parametertyp.
1	8185h	Wert des Parameters *LEN* ist größer als der *CONF_DB* abzüglich des reservierten Headers (4Byte) oder die Längenangabe ist falsch.
1	8186h	Unzulässigen Parameter erkannt. ANY-Pointer *CONF_DB* verweist nicht auf einen DB.
1	8187h	Ungültiger Zustand des FBs. Daten im Header des *CONF_DB* überschrieben oder Instanz-DB nicht vorhanden.
1	8A01h	Statusanzeige im gelesenen Datensatz ist ungültig (Wert >=3)
1	8A02h	Es läuft kein Auftrag auf dem CP; der FB hat jedoch eine Quittung für gelaufenen Auftrag erwartet.
1	8A03h	Es läuft kein Auftrag auf dem CP und der CP ist nicht bereit; der FB hat einen ersten Auftrag für Datensatz-Lesen angestoßen.
1	8A04h	Es läuft kein Auftrag auf dem CP und der CP ist nicht bereit; der FB hat jedoch eine Quittung für den gelaufenen Auftrag erwartet.
1	8A05h	Auftrag läuft, eine Quittung ist jedoch noch nicht erfolgt; der FB hat einen ersten Auftrag für Datensatz-Lesen angestoßen.
1	8A06h	Auftrag ist fertig; der FB hat jedoch einen ersten Auftrag für Datensatz-Lesen angestoßen.
1	8B01h	Kommunikationsfehler, DB konnte nicht übertragen werden.
1	8B02h	Parameterfehler, Doppelter Parameterblock
1	8B03h	Parameterfehler, Subblock im Parameterblock ist nicht erlaubt.

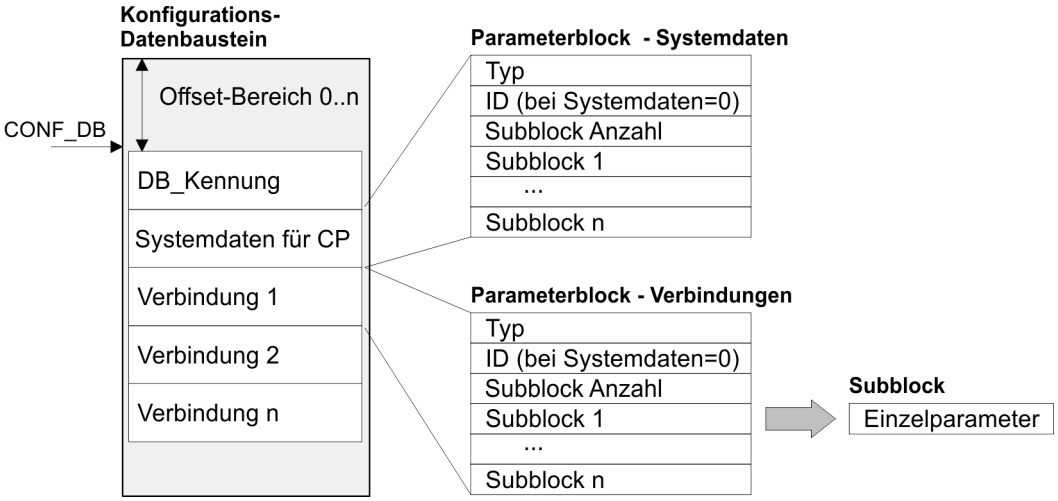
ERROR	STATUS	Beschreibung
1	8B04h	Parameterfehler, Länge die im FB angegeben wurde, stimmt nicht mit der Länge der Parameterblöcke/Subblöcke überein.
1	8B05h	Parameterfehler, Länge des Parameterblocks ist ungültig.
1	8B06h	Parameterfehler, Länge des Subblocks ist ungültig.
1	8B07h	Parameterfehler, ID des Parameterblocks ist ungültig.
1	8B08h	Parameterfehler, ID des Subblocks ist ungültig.
1	8B09h	Systemfehler, Verbindung existiert nicht.
1	8B0Ah	Datenfehler, Inhalt des Subblocks ist nicht korrekt.
1	8B0Bh	Strukturfehler, Subblock ist doppelt aufgeführt.
1	8B0Ch	Datenfehler, im Parameterblock sind nicht alle erforderlichen Parameter enthalten.
1	8B0Dh	Datenfehler, der *CONF_DB* enthält keinen Parameterblock für Systemdaten.
1	8B0Eh	Datenfehler/Strukturfehler, Typ des *CONF_DB* ist ungültig.
1	8B0Fh	Systemfehler, CP hat zu wenig Ressourcen, um den *CONF_DB* vollständig bearbeiten zu können.
1	8B10	Datenfehler, es ist nicht eingestellt, das die Konfiguration über das Anwenderprogramm erfolgt.
1	8B11	Datenfehler, der angegebene Typ des Parameterblocks ist ungültig.
1	8B12	Datenfehler, es wurden zu viele Verbindungen angegeben.
1	8B13	CP interner Fehler
1	8F22h	Bereichslängenfehler beim Lesen eines Parameters.
1	8F23h	Bereichslängenfehler beim Schreiben eines Parameters.
1	8F24h	Bereichsfehler beim Lesen eines Parameters.
1	8F25h	Bereichsfehler beim Schreiben eines Parameters.
1	8F28h	Ausrichtungsfehler beim Lesen eines Parameters.
1	8F29h	Ausrichtungsfehler beim Schreiben eines Parameters.
1	8F30h	Parameter liegt im schreibgeschützten 1. aktuellen DB
1	8F31h	Parameter liegt im schreibgeschützten 2. aktuellen DB
1	8F32h	Parameter enthält eine zu große DB-Nummer.
1	8F33h	DB-Nummernfehler
1	8F3Ah	Zielbereich wurde nicht geladen (DB).
1	8F42h	Quittungsverzug beim Lesen eines Parameters aus dem Peripheriebereich.
1	8F43h	Quittungsverzug beim Schreiben eines Parameters in den Peripheriebereich.
1	8F44h	Zugriff auf einen in der Bausteinbearbeitung zu lesenden Parameter ist gesperrt.
1	8F45h	Zugriff auf einen in der Bausteinbearbeitung zu schreibenden Parameter ist gesperrt.
1	8F7Fh	Interner Fehler

9.2.13.3 Konfigurations- Datenbaustein

Der Konfigurations-Datenbaustein (*CONF_DB*) enthält sämtliche Verbindungsdaten und Konfigurationsdaten (IP-Adresse, Subnetz-Maske, Default-Router, NTP-Uhrzeit-Server und weitere) für einen Ethernet-CP. Der Konfigurations-DB wird mit dem FB 55 an den CP übergeben.

Aufbau

Der *CONF_DB* kann über einen Offset-Bereich an beliebiger Stelle innerhalb eines Datenbausteines beginnen. Jede Verbindung sowie die Systemdaten werden durch einen identisch aufgebauten Parameterblock beschrieben. Einzelne Parameter werden durch Subblöcke typisiert.



Parameterblock für Systemdaten für CP

Nachfolgend finden Sie die für die Vernetzung des CPs relevanten Subblöcke. Diese sind im Parameterblock für Systemdaten für CP anzugeben. Je nach Anwendungsfall sind nicht alle Subblock-Typen erforderlich.

Aufbau

Typ = 0
ID = 0
Subblock Anzahl = n
Subblock 1
Subblock 2
Subblock n

Subblock				**Parameter**	
ID	**Typ**	**Länge (Byte)**	**Bedeutung**	**Besonderheiten**	**Anwendung**
1	SUB_IP_V4	4 + 4	IP-Adresse lokale Station gemäß IPv4		zwingend
2	SUB_NETMASK	4 + 4	Subnetzmaske der lokalen Station		zwingend
4	SUB_DNS_SERV_ADDR	4 + 4	DNS Server Adresse	Kann bis zu 4 mal vorkommen. Erster Eintrag ist der primäre DNS Server.	optional
8	SUB_DEF_ROUTER	4 + 4	IP Adresse des Default Routers		optional

Subblock				Parameter	
ID	**Typ**	**Länge (Byte)**	**Bedeutung**	**Besonderheiten**	**Anwendung**
14	SUB_DHCP_ENABLE	4 + 1	IP-Adresse von einem DHCP-Server beziehen	0: kein DHCP 1: DHCP	optional
15	SUB_CLIENT_ID	Länge Client-ID + 4	-	-	optional
51	MAC-ADR	4 + 6	MAC-Adresse lokaler Knoten		optional

Parameterblöcke für Verbindung

Sie erfahren nachfolgend, welche Werte in die Parameterblöcke einzutragen sind und welche Subblöcke zu den jeweiligen Verbindungstypen zu verwenden sind. Je nach Anwendungsfall sind nicht alle Subblock-Typen erforderlich. Von besonderer Bedeutung ist der ID-Parameter, der jedem Verbindungsparameterblock neben der Typkennung vorangestellt wird. Bei programmierten Verbindungen können Sie diese ID innerhalb des zulässigen Wertebereichs frei vergeben. Sie müssen diese ID dann an der Aufrufschnittstelle der FCs für die SEND/RECV-Schnittstelle zur Identifizierung der Verbindung verwenden.

Wertebereich für die Verbindungs-ID: 1, 2 ... 64

TCP-Verbindung

Typ = 1
ID = Verbindungs-ID
Subblock Anzahl = n
Subblock 1
Subblock 2
Subblock n

Subblock				Parameter	
ID	**Typ**	**Länge (Byte)**	**Bedeutung**	**Besonderheiten**	**Anwendung**
1	SUB_IP_V4	4 + 4	IP-Adresse lokale Station gemäß IPv4		zwingend[1]
9	SUB_LOC_PORT	4 + 2	Port der lokalen Station		zwingend
10	SUB_REM_PORT	4 + 2	Port der fernen Station		zwingend[1]
18	SUB_CONNECT_NAME	Länge Name + 4	Name der Verbindung		optional

Subblock				Parameter	
ID	**Typ**	**Länge (Byte)**	**Bedeutung**	**Besonderheiten**	**Anwendung**
19	SUB_LOC_MODE	4 + 1	Lokale Betriebsart der Verbindung, Wertebereich: 0x00 = SEND/RECV 0x10 = S5-Adressierungsmodus bei FETCH/WRITE [2] 0x80 = FETCH [2] 0x40 = WRITE [2] Defaulteinstellung bei Verzicht auf den Parameter ist SEND/RECV. FETCH/WRITE erfordern die Einstellung passiver Verbindungsaufbau.		optional
21	SUB_KBUS_ADR	-	-	Wert: fix 2	optional
22	SUB_CON_ESTABL	4 + 1	Typ des Verbindungsaufbaus. Legen Sie mit dieser Option fest, ob der Verbindungsaufbau von dieser Station aus erfolgen soll. Wertebereich: 0 = passiv 1 = aktiv		zwingend

1) optional bei passiver Verbindung

2) die Codierungen können mittels ODER-Verknüpfungen kombiniert werden

UDP-Verbindung

Typ = 2
ID = Verbindungs-ID
Subblock Anzahl = n
Subblock 1
Subblock 2
Subblock n

Subblock				Parameter	
ID	**Typ**	**Länge (Byte)**	**Bedeutung**	**Besonder-heiten**	**Anwendung**
1	SUB_IP_V4	4 + 4	IP-Adresse lokale Station gemäß IPv4		zwingend
9	SUB_LOC_PORT	4 + 2	Port der lokalen Station		zwingend
10	SUB_REM_PORT	4 + 2	Port der fernen Station		zwingend
18	SUB_CONNECT_NAME	Länge Name + 4	Name der Verbindung		optional

Subblock				Parameter	
ID	**Typ**	**Länge (Byte)**	**Bedeutung**	**Besonder-heiten**	**Anwendung**
19	SUB_LOC_MODE	4 + 1	Lokale Betriebsart der Verbindung, Wertebereich: 0x00 = SEND/RECV 0x10 = S5-Adressierungsmodus bei FETCH/ WRITE [1] 0x80 = FETCH [1] 0x40 = WRITE [1] Defaulteinstellung bei Verzicht auf den Parameter ist SEND/RECV. FETCH/WRITE erfordern die Einstellung passiver Verbindungsaufbau.		optional
21	SUB_KBUS_ADR	-	-	Wert: fest auf 2	optional
23	SUB_ADDR_IN_DATA_ BLOCK	4 + 1	Freie UDP-Verbindung wählen. Der ferne Teilnehmer wird vom Anwenderprogramm beim AG_SEND Aufruf im Auftragsheader des Auftragspuffers eingetragen. Damit kann jeder beliebige Teilnehmer am Ethernet/LAN/WAN erreicht werden. Wertebereich: 1 = Freie UDP-Verbindung 0 = sonstige		optional

1) die Codierungen können mittels ODER-Verknüpfungen kombiniert werden.

ISO-on-TCP-Verbindung

Typ = 3
ID = Verbindungs-ID
Subblock Anzahl = n
Subblock 1
Subblock 2
Subblock n

Subblock				Parameter	
ID	**Typ**	**Länge (Byte)**	**Bedeutung**	**Besonderheiten**	**Anwendung**
1	SUB_IP_V4	4 + 4	IP-Adresse der fernen Station gemäß IPv4		zwingend[1]
11	SUB_LOC_PORT	TSAP-Länge + 4	TSAP der lokalen Station		zwingend

Subblock				Parameter	
ID	**Typ**	**Länge (Byte)**	**Bedeutung**	**Besonderheiten**	**Anwendung**
12	SUB_REM_PORT	TSAP-Länge + 4	TSAP der fernen Station		zwingend[1]
18	SUB_CONNECT_NAME	Länge Name + 4	Name der Verbindung		optional
19	SUB_LOC_MODE	4 + 1	Lokale Betriebsart der Verbindung, Wertebereich: 0x00 = SEND/RECV 0x10 = S5-Adressierungsmodus bei FETCH/WRITE [2] 0x80 = FETCH [2] 0x40 = WRITE [2] Defaulteinstellung bei Verzicht auf den Parameter ist SEND/RECV. FETCH/WRITE erfordern die Einstellung passiver Verbindungsaufbau.		optional
21	SUB_KBUS_ADR	-	-	Wert fest auf 2	optional
22	SUB_CON_ESTABL	4 + 1	Typ des Verbindungsaufbaus. Legen Sie mit dieser Option fest, ob der Verbindungsaufbau von dieser Station aus erfolgen soll. Wertebereich: 0 = passiv 1 = aktiv		zwingend

1) optional bei passiver Verbindung

2) die Codierungen können mittels ODER-Verknüpfungen kombiniert werden

H1-Verbindung (ISO)

Typ = 10
ID = Verbindungs-ID
Subblock Anzahl = n
Subblock 1
Subblock 2
Subblock n

Subblock				Parameter	
ID	**Typ**	**Länge (Byte)**	**Bedeutung**	**Besonderheiten**	**Anwendung**
51	SUB_MAC	4 + 6	MAC-Adresse der fernen Station		zwingend
11	SUB_LOC_TSAP	TSAP-Länge + 4	TSAP der lokalen Station		zwingend

Subblock				Parameter	
ID	**Typ**	**Länge (Byte)**	**Bedeutung**	**Besonderheiten**	**Anwendung**
12	SUB_REM_TSAP	TSAP-Länge + 4	TSAP der fernen Station		zwingend[1]
18	SUB_CONNECT_NAME	Länge Name + 4	Name der Verbindung		optional
19	SUB_LOC_MODE	4 + 1	Lokale Betriebsart der Verbindung, Wertebereich: 0x00 = SEND/RECV 0x10 = S5-Adressierungsmodus bei FETCH/WRITE [2] 0x80 = FETCH [2] 0x40 = WRITE [2] Defaulteinstellung bei Verzicht auf den Parameter ist SEND/RECV. FETCH/WRITE erfordern die Einstellung passiver Verbindungsaufbau.		optional
22	SUB_CON_ESTABL	4 + 1	Typ des Verbindungsaufbaus. Legen Sie mit dieser Option fest, ob der Verbindungsaufbau von dieser Station aus erfolgen soll. Wertebereich: 0 = passiv; 1 = aktiv		zwingend
52	SUB_TIME_CON_RETRAN	4 + 2	Zeitabstand nach welchem ein fehlgeschlagener Verbindungsaufbau erneut angestoßen wird. (1...60s, Default: 5s)	irrelevant bei passivem Verbindungsaufbau	optional
53	SUB_TIME_DAT_RETRAN	4 + 2	Zeitabstand nach welchem ein fehlgeschlagener Sendeversuch erneut angestoßen wird. (100...30000ms, Default: 1000ms)		optional
54		4 + 2	Anzahl der Sendeversuche inkl. 1. Versuch (1...100, Default: 5)		optional
55		4 + 2	Zeitabstand nach welchem eine Verbindung abgebaut wird, sofern von der Partner-Station keine Rückmeldung kommt.(6...160s, Default: 30s)		optional

1) optional bei passiver Verbindung

2) die Codierungen können mittels ODER-Verknüpfungen kombiniert werden

Siemens-S7-Verbindung

Typ = 11
ID = Verbindungs-ID
Subblock Anzahl = n
Subblock 1
Subblock 2
Subblock n

Subblock				**Parameter**	
ID	**Typ**	**Länge (Byte)**	**Bedeutung**	**Besonderheiten**	**Anwendung**
56	SUB_S/_C_DETAIL	4 + 14	Verbindungsspezifische Parameter		zwingend
18	SUB_CONNECT_NAME	Länge Name + 4	Name der Verbindung		optional
1	SUB_IP_V4	4 + 4	IP-Adresse gemäß IPv4	IP-Adresse des Partners	zwingend[1]
51	SUB_MAC	4 + 6	MAC-Adresse der Partnerstation		zwingend
22	SUB_CON_ESTABL	4 + 1	Typ des Verbindungsaufbaus. Legen Sie mit dieser Option fest, ob der Verbindungsaufbau von dieser Station aus erfolgen soll. Wertebereich: 0 = passiv 1 = aktiv		zwingend

1) optional bei passiver Verbindung

SUB_S/_C_DETAIL

Parameter	**Deklaration**	**Datentyp**	**Beschreibung**
SubBlockID	IN	WORD	Kennung
SubBlockLen	IN	WORD	Länge
TcpIpActive	IN	INT	Verbindung über MAC- oder IP-Adresse (MAC=0, IP=1)
LocalResource	IN	WORD	Lokale Ressource 0001h ... 00DFh (1=PG, 2=OP, 0010h ... 00DFh=unspezifiziert)
LocalRack	IN	WORD	Nummer lokales Rack 0000h ... 0002h
LocalSlot	IN	WORD	Nummer lokaler Steckplatz 0002h ... 000Fh (2=CPU, 4=VIPA-PG/OP, 5=CP int., 6=CP ext.)
RemoteResource	IN	WORD	Remote Ressource 0001h ... 00DFh (1=PG, 2=OP, 0010h ... 00DFh=unspezifiziert)

Parameter	Deklaration	Datentyp	Beschreibung
RemoteRack	IN	WORD	Nummer remote Rack 0000h ... 0002h
RemoteSlot	IN	WORD	Nummer remote Steckplatz 0002h ... 000Fh (2=CPU, 4=VIPA-PG/OP, 5=CP int., 6=CP ext.)

Der "Lokale TSAP" wird gebildet aus *LocalResource*, *LocalRack* und *LocalSlot*.

Der "Remote TSAP" wird gebildet aus *RemoteResource*, *RemoteRack* und *RemoteSlot*.

Zusätzliche Parameterblöcke

Block_VIPA_ BACNET

Sobald der Block_VIPA_BACNET (Sonderkennung 100) im DB enthalten ist, wird eine BACNET-Konfiguration aus dem DB abgeleitet und es werden keine weiteren Blöcke danach ausgewertet.

Typ = 100
Subblock Anzahl = 0

Block_VIPA_IPK

Typ = 101
ID = Verbindungs-ID
Subblock Anzahl = n
Subblock 1
Subblock 2
Subblock n

Subblock				Parameter	
ID	**Typ**	**Länge (Byte)**	**Bedeutung**	**Besonderheiten**	**Anwendung**
1	VIPA_IPK_CYCLE	4 + 4	IPK Zykluszeit für Verbindungs-ID	VIPA-spezifisch	optional

Beispiel-DB

Adresse	Name	Typ	Anfangswert	Aktualwert	Kommentar
0.0	DB_Ident	WORD	W#16#1	W#16#1	
2.0	Systemdaten.Typ	INT	0	0	Systemdaten
4.0	Systemdaten.VerbId	INT	0	0	fix 0
6.0	Systemdaten.SubBlock_Anzahl	INT	3	3	
8.0	Systemdaten.ip.SUB_IP_V4	WORD	W#16#1	W#16#1	
10.0	Systemdaten.ip.SUB_IP_V4_LEN	WORD	W#16#8	W#16#8	
12.0	Systemdaten.ip.IP_0	BYTE	B#16#0	B#16#AC	
13.0	Systemdaten.ip.IP_1	BYTE	B#16#0	B#16#14	
14.0	Systemdaten.ip.IP_2	BYTE	B#16#0	B#16#8B	
15.0	Systemdaten.ip.IP_3	BYTE	B#16#0	B#16#61	
16.0	Systemdaten.netmask.SUB_NETMASK	WORD	W#16#2	W#16#2	

Adresse	Name	Typ	Anfangswert	Aktualwert	Kommentar
18.0	Systemdaten.netmask.SUB_NETMASK_LEN	WORD	W#16#8	W#16#8	
20.0	Systemdaten.netmask.NETMASK_0	BYTE	B#16#0	B#16#FF	
21.0	Systemdaten.netmask.NETMASK_1	BYTE	B#16#0	B#16#FF	
22.0	Systemdaten.netmask.NETMASK_2	BYTE	B#16#0	B#16#FF	
23.0	Systemdaten.netmask.NETMASK_3	BYTE	B#16#0	B#16#0	
24.0	Systemdaten.router.SUB_DEF_ROUTER	WORD	W#16#8	W#16#8	
26.0	Systemdaten.router.SUB_DEF_ROUTER_LEN	WORD	W#16#8	W#16#8	
28.0	Systemdaten.router.ROUTER_0	BYTE	B#16#0	B#16#AC	
29.0	Systemdaten.router.ROUTER_1	BYTE	B#16#0	B#16#14	
30.0	Systemdaten.router.ROUTER_2	BYTE	B#16#0	B#16#8B	
31.0	Systemdaten.router.ROUTER_3	BYTE	B#16#0	B#16#61	
32.0	Con_TCP_ID1.Typ	INT	1	1	TCP-Verbindung
34.0	Con_TCP_ID1.VerbId	INT	0	1	Verbindungs-ID
36.0	Con_TCP_ID1.SubBlock_Anzahl	INT	4	4	
38.0	Con_TCP_ID1.ip1.SUB_IP_V4	WORD	W#16#1	W#16#1	
40.0	Con_TCP_ID1.ip1. SUB_IP_V4_LEN	WORD	W#16#8	W#16#8	
42.0	Con_TCP_ID1.ip1.IP_0	BYTE	B#16#0	B#16#AC	
43.0	Con_TCP_ID1.ip1.IP_1	BYTE	B#16#0	B#16#14	
44.0	Con_TCP_ID1.ip1.IP_2	BYTE	B#16#0	B#16#8B	
45.0	Con_TCP_ID1.ip1.IP_3	BYTE	B#16#0	B#16#62	
46.0	Con_TCP_ID1.locport.SUB_LOC_PORT	WORD	W#16#9	W#16#9	
48.0	Con_TCP_ID1.locport.SUB_LOC_PORT_LEN	WORD	W#16#6	W#16#6	
50.0	Con_TCP_ID1.locport.LOC_PORT	WORD	W#16#0	W#16#3E9	
52.0	Con_TCP_ID1.remport.SUB_REM_PORT	WORD	W#16#A	W#16#A	
54.0	Con_TCP_ID1.remport.SUB_REM_PORT_LEN	WORD	W#16#6	W#16#6	
56.0	Con_TCP_ID1.remport.REM_PORT	WORD	W#16#0	W#16#3E9	
58.0	Con_TCP_ID1.con_est.SUB_CON_ESTABL	WORD	W#16#16	W#16#16	
60.0	Con_TCP_ID1.con_est.SUB_CON_ESTABL_LEN	WORD	W#16#6	W#16#6	
62.0	Con_TCP_ID1.con_est.CON_ESTABL	BYTE	B#16#0	B#16#1	
64.0	Con_ISO_ID3.Typ	INT	3	3	ISO-on-TCP-Verbindung
66.0	Con_ISO_ID3.VerbId	INT	0	3	Verbindungs-ID
68.0	Con_ISO_ID3.SubBlock_Anzahl	INT	4	4	
70.0	Con_ISO_ID3.ip1. SUB_IP_V4	WORD	W#16#1	W#16#1	
72.0	Con_ISO_ID3.ip1. SUB_IP_V4_LEN	WORD	W#16#8	W#16#8	
74.0	Con_ISO_ID3.ip1.IP_0	BYTE	B#16#0	B#16#AC	
75.0	Con_ISO_ID3.ip1.IP_1	BYTE	B#16#0	B#16#10	
76.0	Con_ISO_ID3.ip1.IP_2	BYTE	B#16#0	B#16#8B	
77.0	Con_ISO_ID3.ip1.IP_3	BYTE	B#16#0	B#16#62	
78.0	Con_ISO_ID3.loc_TSAP.SUB_LOC_PORT	WORD	W#16#B	W#16#B	
80.0	Con_ISO_ID3.loc_TSAP.SUB_LOC_PORT_LEN	WORD	W#16#A	W#16#A	

Adresse	Name	Typ	Anfangswert	Aktualwert	Kommentar
82.0	Con_ISO_ID3.loc_TSAP.LOC_TSAP[0]	BYTE	B#16#0	B#16#54	
83.0	Con_ISO_ID3.loc_TSAP.LOC_TSAP[1]	BYTE	B#16#0	B#16#53	
84.0	Con_ISO_ID3.loc_TSAP.LOC_TSAP[2]	BYTE	B#16#0	B#16#41	
85.0	Con_ISO_ID3.loc_TSAP.LOC_TSAP[3]	BYTE	B#16#0	B#16#50	
86.0	Con_ISO_ID3.loc_TSAP.LOC_TSAP[4]	BYTE	B#16#0	B#16#30	
87.0	Con_ISO_ID3.loc_TSAP.LOC_TSAP[5]	BYTE	B#16#0	B#16#31	
88.0	Con_ISO_ID3.rem_TSAP.SUB_REM_PORT	WORD	W#16#C	W#16#C	
90.0	Con_ISO_ID3.rem_TSAP.SUB_REM_PORT_LEN	WORD	W#16#A	W#16#A	
92.0	Con_ISO_ID3.rem_TSAP.REM_TSAP[0]	BYTE	B#16#0	B#16#54	
93.0	Con_ISO_ID3.rem_TSAP.REM_TSAP[1]	BYTE	B#16#0	B#16#53	
94.0	Con_ISO_ID3.rem_TSAP.REM_TSAP[2]	BYTE	B#16#0	B#16#41	
95.0	Con_ISO_ID3.rem_TSAP.REM_TSAP[3]	BYTE	B#16#0	B#16#50	
96.0	Con_ISO_ID3.rem_TSAP.REM_TSAP[4]	BYTE	B#16#0	B#16#30	
97.0	Con_ISO_ID3.rem_TSAP.REM_TSAP[5]	BYTE	B#16#0	B#16#31	
98.0	Con_ISO_ID3.con_est.SUB_CON_ESTABL	WORD	W#16#16	W#16#16	
100.0	Con_ISO_ID3.con_est.SUB_CON_ESTABL_LEN SUB_CON_ESTABL SUB_CON_ESTABL_LEN	WORD	W#16#6	W#16#6	
102.0	Con_ISO_ID3.con_est.CON_ESTABL	BYTE	B#16#0	B#16#1	
104.0	S7_Verb.Typ	INT	11	11	S7-Verbindung
106.0	S7_Verb.Verb_ID	INT	0	0	Verb.-ID
108.0	S7_Verb.SubBlock_Anzahl	INT	5	5	
110.0	S7_Verb.Verb_Parameter.SUB_S7_C_DETAIL	INT	56	56	
112.0	S7_Verb.Verb_Parameter. SUB_S7_C_DETAIL_LEN	INT	18	18	
114.0	S7_Verb.Verb_Parameter.TcpIpActive	INT	0	1	
116.0	S7_Verb.Verb_Parameter.LocalResource	INT	0	2	
118.0	S7_Verb.Verb_Parameter.LocalRack	INT	0	0	
120.0	S7_Verb.Verb_Parameter.LocalsSlot	INT	0	2	
122.0	S7_Verb.Verb_Parameter.RemoteResource	INT	0	2	
124.0	S7_Verb.Verb_Parameter.RemoteRack	INT	0	0	
126.0	S7_Verb.Verb_Parameter.RemoteSlot	INT	0	2	
128.0	S7_Verb.ipl.SUB_IP_V4	WORD	W#16#1	W#16#1	
130.0	S7_Verb.ipl. SUB_IP_V4_LEN	WORD	W#16#8	W#16#8	
132.0	S7_Verb.ipl.IP_0	BYTE	B#16#0	B#16#AC	
133.0	S7_Verb.ipl.IP_1	BYTE	B#16#0	B#16#10	
134.0	S7_Verb.ipl.IP_2	BYTE	B#16#0	B#16#8B	
135.0	S7_Verb.ipl.IP_3	BYTE	B#16#0	B#16#62	
136.0	S7_Verb.Mac.SUB_MAC	INT	51	51	
138.0	S7_Verb.Mac.SUB_MAC_LEN	INT	10	10	
140.0	S7_Verb.Mac.MAC_0	BYTE	B#16#0	B#16#0	
141.0	S7_Verb.Mac.MAC_1	BYTE	B#16#0	B#16#20	
142.0	S7_Verb.Mac.MAC_2	BYTE	B#16#0	B#16#D5	

Adresse	Name	Typ	Anfangswert	Aktualwert	Kommentar
143.0	S7_Verb.Mac.MAC_3	BYTE	B#16#0	B#16#77	
144.0	S7_Verb.Mac.MAC_4	BYTE	B#16#0	B#16#53	
145.0	S7_Verb.Mac.MAC_5	BYTE	B#16#0	B#16#9B	
146.0	S7_Verb.con_est .SUB_CON_ESTABL	WORD	W#16#16	W#16#16	
148.0	S7_Verb.con_est.SUB_CON_ESTABL_LEN	WORD	W#16#6	W#16#6	
150.0	S7_Verb.con_est.CON_ESTABL	BYTE	B#16#0	B#16#1	
152.0	S7_Verb.name_verb.SUB_CONNECT_NAME	WORD	W#16#12	W#16#12	
154.0	S7_Verb.name_verb.SUB_CONNECT_NAME_LEN	WORD	W#16#23	W#16#23	
156.0	S7_Verb.name_verb.CONNECT_NAME[0]	CHAR	' '	'V'	S7-Verbindung mit IP-Konfig. 1
157.0	S7_Verb.name_verb.CONNECT_NAME[1]	CHAR	' '	'e'	
158.0	S7_Verb.name_verb.CONNECT_NAME[2]	CHAR	' '	'r'	
159.0	S7_Verb.name_verb.CONNECT_NAME[3]	CHAR	' '	'b'	
160.0	S7_Verb.name_verb.CONNECT_NAME[4]	CHAR	' '	'i'	
161.0	S7_Verb.name_verb.CONNECT_NAME[5]	CHAR	' '	'n'	
162.0	S7_Verb.name_verb.CONNECT_NAME[6]	CHAR	' '	'd'	
163.0	S7_Verb.name_verb.CONNECT_NAME[7]	CHAR	' '	'u'	
164.0	S7_Verb.name_verb.CONNECT_NAME[8]	CHAR	' '	'n'	
165.0	S7_Verb.name_verb.CONNECT_NAME[9]	CHAR	' '	'g'	
166.0	S7_Verb.name_verb.CONNECT_NAME[10]	CHAR	' '	' '	
167.0	S7_Verb.name_verb.CONNECT_NAME[11]	CHAR	' '	'S'	
168.0	S7_Verb.name_verb.CONNECT_NAME[12]	CHAR	' '	'7'	
169.0	S7_Verb.name_verb.CONNECT_NAME[13]	CHAR	' '	' '	
170.0	S7_Verb.name_verb.CONNECT_NAME[14]	CHAR	' '	'm'	
171.0	S7_Verb.name_verb.CONNECT_NAME[15]	CHAR	' '	'i'	
172.0	S7_Verb.name_verb.CONNECT_NAME[16]	CHAR	' '	't'	
173.0	S7_Verb.name_verb.CONNECT_NAME[17]	CHAR	' '	' '	
174.0	S7_Verb.name_verb.CONNECT_NAME[18]	CHAR	' '	'I'	
175.0	S7_Verb.name_verb.CONNECT_NAME[19]	CHAR	' '	'P'	
176.0	S7_Verb.name_verb.CONNECT_NAME[20]	CHAR	' '	'-'	
177.0	S7_Verb.name_verb.CONNECT_NAME[21]	CHAR	' '	'C'	
178.0	S7_Verb.name_verb.CONNECT_NAME[22]	CHAR	' '	'o'	
179.0	S7_Verb.name_verb.CONNECT_NAME[23]	CHAR	' '	'n'	
180.0	S7_Verb.name_verb.CONNECT_NAME[24]	CHAR	' '	'f'	
181.0	S7_Verb.name_verb.CONNECT_NAME[25]	CHAR	' '	'i'	
182.0	S7_Verb.name_verb.CONNECT_NAME[26]	CHAR	' '	'g'	
183.0	S7_Verb.name_verb.CONNECT_NAME[27]	CHAR	' '	' '	
184.0	S7_Verb.name_verb.CONNECT_NAME[28]	CHAR	' '	'1'	
185.0	S7_Verb.name_verb.CONNECT_NAME[29]	CHAR	' '	' '	
186.0	S7_Verb.name_verb.CONNECT_NAME[30]	CHAR	' '	' '	

10 Modbus-Kommunikation - *"Modbus Communication"*

10.1 TCP

10.1.1 FB 70 - TCP_MB_CLIENT - Modbus/TCP-Client

10.1.1.1 Beschreibung

Dieser Funktionsbaustein ermöglicht den Betrieb einer Ethernet-Schnittstelle als Modbus/TCP-Client.

Aufrufparameter

Name	Deklaration	Typ	Beschreibung
REQ	IN	BOOL	Auftrag starten mit Flanke 0-1.
ID	IN	WORD	ID von TCON.
MB_FUNCTION	IN	BYTE	Modbus: *Funktions-Code*.
MB_DATA_ADDR	IN	WORD	Modbus: Startadresse oder *Sub-Funktions-Code*.
MB_DATA_LEN	IN	INT	Modbus: Anzahl der Register/Bits.
MB_DATA_PTR	IN	ANY	Modbus: Datenpuffer (nur Merkerbereich oder Datenbaustein vom Datentyp Byte zulässig) für den Zugriff mit *Funktions-Code 03h*, *06h* und *10h*.
DONE *	OUT	BOOL	Auftrag fertig ohne Fehler.
BUSY	OUT	BOOL	Auftrag ist in Bearbeitung.
ERROR *	OUT	BOOL	Auftrag fertig mit Fehler - Parameter STATUS enthält die Fehlerinformation.
STATUS *	OUT	WORD	Erweiterte Status- und Fehlerinformationen.

*) Parameter steht bis zum nächsten Aufruf des FBs zur Verfügung.

Parameter im Instanz-DB

Name	Deklaration	Typ	Beschreibung
PROTOCOL_TIMEOUT	STAT	INT	Sperrzeit bevor ein aktiver Auftrag vom Anwender abgebrochen werden kann. Default: 3s
RCV_TIMEOUT	STAT	INT	Überwachungszeit für einen Auftrag. Default: 2s
MB_TRANS_ID	STAT	WORD	Modbus: Startwert für den Transaktions Identifier. Default: 1
MB_UNIT_ID	STAT	BYTE	Modbus: Geräteidentifikation. Default: 255

Hierbei ist folgendes zu beachten:

- Die *Aufrufparameter* sind beim Baustein-Aufruf anzugeben. Neben den *Aufrufparametern* finden Sie alle Parameter im Instanz-DB.
- Die Kommunikationsverbindung muss zuvor über FB 65 (TCON) initialisiert werden.

- FB 63 (TSEND) und FB 64 (TRCV) sind für die Verwendung des Baustein erforderlich.
- Während einer Auftragsbearbeitung wird der Instanz-DB für andere Clients gesperrt.
- Während einer Auftragsbearbeitung werden Änderungen an den Eingangsparametern nicht ausgewertet.
- Unter einer der folgenden Bedingungen ist eine Auftragsbearbeitung abgeschlossen bzw. wird abgebrochen:
  - DONE = 1 bei Auftrag ohne Fehler
  - ERROR = 1 bei Auftrag mit Fehler
  - Ablauf von RCV_TIMEOUT
  - REQ = FALSE nach Ablauf von PROTOCOL_TIMEOUT
- Wird REQ zurückgesetzt bevor DONE oder ERROR gesetzt oder PROTOCOL_TIMEOUT abgelaufen ist, wird STATUS 8200h geliefert. Hierbei wird der aktive Auftrag weiterhin bearbeitet.

Status- und Fehleranzeige

Der Funktionsbaustein liefert über STATUS die folgenden Status- und Fehlerinformationen.

STATUS	DONE	BUSY	ERROR	Beschreibung
0000h	1	0	0	Anweisung fehlerfrei ausgeführt.
7000h	0	0	0	Keine Verbindung aufgebaut oder Kommunikationsfehler (TCON).
7004h	0	0	0	Verbindung hergestellt und überwacht. Keine Auftragsbearbeitung aktiv.
7005h	0	1	0	Daten werden gesendet.
7006h	0	1	0	Daten werden empfangen.
8210h	0	0	1	Die Hardware ist inkompatibel zur Baustein-Bibliothek Modbus RTU/TCP.
8380h	0	0	1	Empfangenes Modbus-Frame hat nicht das richtige Format oder eine ungültige Länge.
8381h	0	0	1	Server liefert *Exception Code 01h*. *395*
8382h	0	0	1	Server liefert *Exception Code 03h* oder falsche Startadresse. *395*
8383h	0	0	1	Server liefert *Exception Code 02h*. *395*
8384h	0	0	1	Server liefert *Exception Code 04h*. *395*
8386h	0	0	1	Server liefert falschen *Funktions-Code*.
8387h	0	0	1	Verbindungs-ID (TCON) passt nicht zur Instanz oder Server liefert falsche Protokoll-ID.
8388h	0	0	1	Server liefert falschen Wert oder falsche Quantity.
80C8h	0	0	1	Keine Antwort des Servers im definierten Zeitraum (RCV_TIMEOUT).
8188h	0	0	1	MB_FUNCTION ungültig.
8189h	0	0	1	MB_DATA_ADDR ungültig.
818Ah	0	0	1	MB_DATA_LEN ungültig.
818Bh	0	0	1	MB_DATA_PTR ungültig.

STATUS	DONE	BUSY	ERROR	Beschreibung
818Ch	0	0	1	BLOCKED_PROC_TIMEOUT oder RCV_TIMEOUT ungültig.
818Dh	0	0	1	Server liefert falsche Transaktions-ID.
8200h	0	0	1	Eine andere Modbus-Anfrage wird zur Zeit über den Port verarbeitet (PROTOCOL_TIMEOUT).

10.1.1.2 Beispiel

Aufgabenstellung

Von einem Modbus/TCP-Server sollen mit *Funktions-Code 03h* 100 Register ab Startadresse 2000 gelesen werden und im Merkerbereich ab MB200 abgelegt werden. Fehler sollen abgespeichert werden.

OB1

```
 CALL  FB    65 , DB65
       REQ    :=M100.0
       ID     :=W#16#1
       DONE   :=M100.1
       BUSY   :=
       ERROR  :=M100.2
       STATUS :=MW102
       CONNECT:=P#DB255.DBX 0.0 BYTE 64

      UN    M    100.2
      SPB   ERR1
      L     MW   102
      T     MW   104
ERR1: NOP   0
      U     M    100.1
      R     M    100.0

CALL  FB    70 , DB70
       REQ          :=M101.0
       ID           :=W#16#1
       MB_FUNCTION  :=B#16#3
       MB_DATA_ADDR:=W#16#7D0
       MB_DATA_LEN  :=100
       MB_DATA_PTR  :=P#M 200.0 BYTE 200
       DONE         :=M101.1
       BUSY         :=
       ERROR        :=M101.2
       STATUS       :=MW106

      UN    M    101.2
      SPB   ERR2
      L     MW   106
      T     MW   108
ERR2: NOP   0
      U     M    101.1
      R     M    101.0
```

OB1 - Beschreibung

1. Aufruf von FB 65 (TCON) zur Herstellung der Kommunikationsverbindung mit der Partnerstation.
2. Aufruf des Modbus/TCP-Client Hantierungsbausteins mit den korrekten Parametern.
3. Es ist keine Verbindung zur Partnerstation aufgebaut und MW102 liefert 7000h.

4. M100.0 in der CPU auf TRUE setzten.
   ⇨ Wenn M100.0 automatisch zurück gesetzt wird, ist die Verbindung zur Partnerstation aufgebaut und MW108 liefert 7004h.
5. M101.0 in der CPU auf TRUE setzen.
   ⇨ Der Modbus-Request wird versendet und auf eine Antwort gewartet.

   Wenn M101.0 automatisch zurück gesetzt wird, wurde der Auftrag fehlerfrei bearbeitet und die gelesenen Daten liegen ab Merkerbyte 200 in der CPU. MW108 liefert 7004h und signalisiert die Bereitschaft für einen neuen Auftrag.

   Wenn M101.0 nicht automatisch zurück gesetzt wird und MW108 einen Wert ungleich 0 liefert, ist ein Fehler aufgetreten. Die Fehlerursache kann über den Code in MW108 ausgelesen werden (z.B. MW108 = 8382h wenn die Startadresse 2000 im Server nicht vorhanden ist). MW108 liefert 7004h und signalisiert die Bereitschaft für einen neuen Auftrag.

10.1.2 FB 71 - TCP_MB_SERVER - Modbus/TCP-Server

10.1.2.1 Beschreibung

Dieser Funktionsbaustein ermöglicht den Betrieb einer Ethernet-Schnittstelle als Modbus/TCP-Server.

Aufrufparameter

Name	Deklaration	Typ	Beschreibung
ENABLE	IN	BOOL	Aktivierung/Deaktivierung Modbus-Server.
MB_DATA_PTR	IN	ANY	Modbus: Datenpuffer (nur Merkerbereich oder Datenbaustein vom Datentyp Byte zulässig) für den Zugriff mit *Funktions-Code 03h*, *06h* und *10h*.
ID	IN	WORD	ID von TCON.
NDR*	OUT	BOOL	Neue Daten wurden durch den Modbus-Client geschrieben.
DR*	OUT	BOOL	Daten wurden vom Modbus Client gelesen.
ERROR*	OUT	BOOL	Auftrag fertig mit Fehler - Parameter STATUS enthält die Fehlerinformation.
STATUS*	OUT	WORD	Erweiterte Status- und Fehlerinformationen.

*) Parameter steht bis zum nächsten Aufruf des FBs zur Verfügung.

Parameter im Instanz-DB

Name	Deklaration	Typ	Beschreibung
REQUEST_COUNT	STAT	WORD	Zähler für jedes empfangene Telegramm.
MESSAGE_COUNT	STAT	WORD	Zähler für jeden gültigen Modbus-Request.
XMT_RCV_COUNT	STAT	WORD	Zähler für jedes empfangene Telegramm, welches keinen gültigen Modbus-Request enthält.
EXCEPTION_COUNT	STAT	WORD	Zähler für jeden negativ quittierten Modbus-Request.

Name	Deklaration	Typ	Beschreibung
SUCCESS_COUNT	STAT	WORD	Zähler für jeden positiv quittierten Modbus-Request.
FC1_ADDR_OUTPUT_START	STAT	WORD	Modbus *Funktions-Code 01h* Startregister für A0.0 Default: 0
FC1_ADDR_OUTPUT_END	STAT	WORD	Modbus *Funktions-Code 01h* Endregister für Ax.y Default: 19999
FC1_ADDR_MEMORY_START	STAT	WORD	Modbus *Funktions-Code 01h* Startregister für M0.0 Default: 20000
FC1_ADDR_MEMORY_END	STAT	WORD	Modbus *Funktions-Code 01h* Endregister für Mx.y Default: 39999
FC2_ADDR_INPUT_START	STAT	WORD	Modbus *Funktions-Code 02h* Startregister für E0.0 Default: 0
FC2_ADDR_INPUT_END	STAT	WORD	Modbus *Funktions-Code 02h* Endregister für Ex.y Default: 19999
FC2_ADDR_MEMORY_START	STAT	WORD	Modbus *Funktions-Code 02h* Startregister für M0.0 Default: 20000
FC2_ADDR_MEMORY_END	STAT	WORD	Modbus *Funktions-Code 02h* Endregister für Mx.y Default: 39999
FC4_ADDR_INPUT_START	STAT	WORD	Modbus *Funktions-Code 04h* Startregister für EW0 Default: 0
FC4_ADDR_INPUT_END	STAT	WORD	Modbus *Funktions-Code 04h* Endregister für EWx Default: 19999
FC4_ADDR_MEMORY_START	STAT	WORD	Modbus *Funktions-Code 04h* Startregister für MW0 Default: 20000
FC4_ADDR_MEMORY_END	STAT	WORD	Modbus *Funktions-Code 04h* Endregister für MWx Default: 39999
FC5_ADDR_OUTPUT_START	STAT	WORD	Modbus *Funktions-Code 05h* Startregister für A0.0 Default: 0
FC5_ADDR_OUTPUT_END	STAT	WORD	Modbus *Funktions-Code 05h* Endregister für Ax.y Default: 19999
FC5_ADDR_MEMORY_START	STAT	WORD	Modbus *Funktions-Code 05h* Startregister für M0.0 Default: 20000
FC5_ADDR_MEMORY_END	STAT	WORD	Modbus *Funktions-Code 05h* Endregister für Mx.y Default: 39999
FC15_ADDR_OUTPUT_START	STAT	WORD	Modbus *Funktions-Code 0Fh* Startregister für A0.0 Default: 0
FC15_ADDR_OUTPUT_END	STAT	WORD	Modbus *Funktions-Code 0Fh* Endregister für Ax.y Default: 19999

Name	Deklaration	Typ	Beschreibung
FC15_ADDR_MEMORY_START	STAT	WORD	Modbus *Funktions-Code 0Fh* Startregister für M0.0 Default: 20000
FC15_ADDR_MEMORY_END	STAT	WORD	Modbus *Funktions-Code 0Fh* Endregister für Mx.y Default: 39999

Hierbei ist folgendes zu beachten:

- Die *Aufrufparameter* sind beim Baustein-Aufruf anzugeben. Neben den *Aufrufparametern* finden Sie alle Parameter im Instanz-DB.
- Die Kommunikationsverbindung muss zuvor über FB 65 (TCON) initialisiert werden.
- FB 63 (TSEND) und FB 64 (TRCV) sind für die Verwendung des Bausteins erforderlich.
- Die INPUT/OUTPUT Modbus-Adressen eines *Funktions-Codes* müssen vor den MEMORY Modbus-Adressen liegen und somit immer kleiner sein.
- Innerhalb eines *Funktions-Code* darf keine Modbus-Adresse mehrfach definiert werden - auch die 0 nicht!
- Der Server kann nur einen Auftrag gleichzeitig bearbeiten. Neue Modbus-Anfragen während einer Auftragsbearbeitung werden ignoriert und nicht beantwortet.

Status- und Fehleranzeige

Der Funktionsbaustein liefert über *STATUS* die folgenden Status- und Fehlerinformationen.

STATUS	NDR	DR	ERROR	Beschreibung
0000h	0 oder 1*		0	Anweisung fehlerfrei ausgeführt.
7000h	0	0	0	Keine Verbindung aufgebaut oder Kommunikationsfehler (TCON).
7005h	0	0	0	Daten werden gesendet.
7006h	0	0	0	Daten werden empfangen.
8210h	0	0	1	Die Hardware ist inkompatibel zur Baustein-Bibliothek Modbus RTU/TCP.
8380h	0	0	1	Empfangenes Modbus-Frame hat nicht das richtige Format oder es wurden zu wenige Bytes empfangen.
8381h	0	0	1	*Exception Code 01h*, *Funktion-Code* wird nicht unterstützt. ↳ *395*
8382h	0	0	1	*Exception Code 03h*, Datenlänge oder Datenwert ungültig. ↳ *395*
8383h	0	0	1	*Exception Code 02h*, Ungültige Startadresse bzw. Adressbereich. ↳ *395*
8384h	0	0	1	*Exception Code 04h*, Bereichslängenfehler beim Zugriff auf Eingänge, Ausgänge oder Merker. ↳ *395*
8387h	0	0	1	Verbindungs-ID (TCON) passt nicht zur Instanz oder Client liefert falsche Protokoll-ID.
8187h	0	0	1	MB_DATA_PTR ungültig.

*) Fehlerfreier Modbus-Auftrag mit *Funktions-Code 05h, 06h, 0Fh* oder *10h* liefert NDR=1 und DR=0 bzw. fehlerfreier Modbus-Auftrag mit *Funktons-Code 01h, 02h, 03h, 04h* liefert DR=1 und NDR=0.

10.1.2.2 Beispiel

Aufgabenstellung

Die CPU stellt 100 Byte Daten im Merkerbereich ab MB200 für einen Modbus-Client über die Modbus-Register 0...49 zur Verfügung. Die Daten können vom Modbus-Client mit dem *Funktions-Code 03h* gelesen und mit *Funktions-Code 06h, 10h* geschrieben werden. Der Ausgang A1.0 in der CPU soll von einem Modbus-Client über den *Funktions-Code 05h* und die Startadresse 5008 angesteuert werden können. Fehler sollen abgespeichert werden.

OB1

```
CALL  FB    65 , DB65
       REQ    :=M100.0
       ID     :=W#16#1
       DONE   :=M100.1
       BUSY   :=
       ERROR  :=M100.2
       STATUS :=MW102
       CONNECT:=P#DB255.DBX 0.0 BYTE 64

      UN    M    100.2
      SPB   ERR1
      L     MW   102
      T     MW   104
ERR1: NOP   0
      U     M    100.1
      R     M    100.0

      L     5000
      T     DB71.DBW   52

      CALL  FB    71 , DB71
       ENABLE     :=M101.0
       MB_DATA_PTR:=P#M 200.0 BYTE 100
       ID         :=W#16#1
       NDR        :=M101.1
       DR         :=M101.2
       ERROR      :=M101.3
       STATUS     :=MW106

      UN    M    101.3
      SPB   ERR2
      L     MW   106
      T     MW   108
ERR2: NOP   0
```

OB1 - Beschreibung

1. Aufruf von FB 65 (TCON) zur Herstellung der Kommunikationsverbindung mit der Partnerstation.
2. Aufruf des Modbus/TCP-Server Hantierungsbausteins mit den korrekten Parametern.
3. Es ist keine Verbindung zur Partnerstation aufgebaut und MW102 liefert 7000h.
4. M100.0 in der CPU auf TRUE setzten.
   ⇨ Wenn M100.0 automatisch zurück gesetzt wird, ist die Verbindung zur Partnerstation aufgebaut und MW108 liefert 7006h.
5. Das Modbus-Startregister für die über *Funktions-Code 05h* erreichbaren Ausgänge im Prozessabbild wird im Beispiel über den Parameter FC5_ADDR_OUTPUT_START (Wort 52 im Instanz-Datenbaustein) geändert.
6. M101.0 in der CPU auf TRUE setzen.
   ⇨ Der Modbus-Server arbeitet nun.
7. Der Client sendet einen Modbus-Request mit *Funktions-Code 03h*, Startadresse 10 und Quantity 30.

⇨ Der Server antwortet mit 60 Byte ab MB220. DR wird für einen CPU-Zyklus angesteuert und somit M101.2 auf "1" gesetzt.

8. Der Client sendet einen Modbus-Request mit *Funktions-Code 05h*, Startadresse 5008 und dem Wert FF00h.

⇨ Der Server quittiert den Auftrag und schreibt den Ausgang A1.0 auf "1". NDR wird für einen CPU-Zyklus angesteuert und somit M101.1 auf "1" gesetzt.

9. Der Client sendet einen Modbus-Request mit *Funktions-Code 03h*, Startadresse 50 (nicht vorhanden!) und Quantity 1.

⇨ Der Server antwortet mit einem *Exception Code 02h* und steuert ERROR/ STATUS für einen CPU-Zyklus an. MW108 liefert 8383h.

10.2 RTU

10.2.1 FB 72 - RTU_MB_MASTER - Modbus-RTU-Master

10.2.1.1 Beschreibung

Dieser Funktionsbaustein ermöglicht den Betrieb der internen seriellen RS485 Schnittstelle einer SPEED7 VIPA-CPU oder eines VIPA System SLIO CP 040 als Modbus-RTU-Master.

Aufrufparameter

Name	Deklaration	Typ	Beschreibung
REQ	IN	BOOL	Auftrag starten mit Flanke 0-1.
HARDWARE	IN	BYTE	1 = System SLIO CP 040 / 2 = SPEED7 VIPA-CPU
LADDR	IN	INT	Logische Adresse vom System SLIO CP 040 (Parameter wird für SPEED7 VIPA-CPU ignoriert).
MB_UNIT_ID	IN	BYTE	Modbus: Geräteidentifikation = Adresse vom Slave (0 ... 247).
MB_FUNCTION	IN	BYTE	Modbus: *Funktions-Code*. Bitte beachten Sie, dass der *Functions-Code* 16h nicht unterstützt wird!
MB_DATA_ADDR	IN	WORD	Modbus: Startadresse oder *Sub-Funktions-Code*.
MB_DATA_LEN	IN	INT	Modbus: Anzahl der Register/Bits.
MB_DATA_PTR	IN	ANY	Modbus: Datenpuffer (nur Merkerbereich oder Datenbaustein vom Datentyp Byte zulässig) für den Zugriff mit *Funktions-Code 03h*, *06h* und *10h*.
DONE*	OUT	BOOL	Auftrag fertig ohne Fehler.
BUSY	OUT	BOOL	Auftrag ist in Bearbeitung.
ERROR*	OUT	BOOL	Auftrag fertig mit Fehler - Parameter *STATUS* enthält die Fehlerinformation.
STATUS*	OUT	WORD	Erweiterte Status- und Fehlerinformationen.

*) Parameter steht bis zum nächsten Aufruf des FBs zur Verfügung.

Parameter im Instanz-DB

Name	Deklaration	Typ	Beschreibung
INIT	STAT	BOOL	Eine Flanke 0-1 führt einen Synchron Reset am System SLIO CP 040 durch. Nach erfolgreichem Reset wird das Bit automatisch zurück gesetzt.

Hierbei ist folgendes zu beachten:

- Die *Aufrufparameter* sind beim Baustein-Aufruf anzugeben. Neben den *Aufrufparametern* finden Sie alle Parameter im Instanz-DB.
- Die verwendete Schnittstelle muss zuvor konfiguriert werden:
  - VIPA System SLIO CP 040: Projektierung als "Modbus Master RTU" mit 60 Byte IO-Size in der Hardwarekonfiguration.
  - Interne serielle RS485 Schnittstelle einer VIPA-CPU:
    Projektierung über SFC 216 (SER_CFG) mit Protokoll "Modbus Master RTU".
- FB 60 SEND und FB 61 RECEIVE (oder FB 65 SEND_RECV) sind für die Verwendung des Bausteins zwingend erforderlich, auch wenn die interne serielle RS485 Schnittstelle einer VIPA-CPU verwendet wird.
- Während einer Auftragsbearbeitung werden Änderungen an den Eingangsparametern nicht ausgewertet.
- Broadcast Request über MB_UNIT_ID = 0 werden nur für schreibende Funktionen akzeptiert.
- Unter einer der folgenden Bedingungen ist eine Auftragsbearbeitung abgeschlossen bzw. wird abgebrochen:
  - *DONE* = 1 bei Auftrag ohne Fehler
  - *ERROR* = 1 bei Auftrag mit Fehler
  - Ablauf vom Timeout (Parametrierung bei der Schnittstelle)
- Wird *REQ* zurückgesetzt bevor *DONE* oder *ERROR* gesetzt ist, wird *STATUS* 8200h geliefert. Hierbei wird der aktive Auftrag weiterhin bearbeitet.

Status- und Fehleranzeige

Der Funktionsbaustein liefert über STATUS die folgenden Status- und Fehlerinformationen.

STATUS	DONE	BUSY	ERROR	Beschreibung
0000h	1	0	0	Anweisung fehlerfrei ausgeführt.
7000h	0	0	0	Keine Verbindung aufgebaut oder Kommunikationsfehler.
7004h	0	0	0	Verbindung hergestellt und überwacht. Keine Auftragsbearbeitung aktiv.
7005h	0	1	0	Daten werden gesendet.
7006h	0	1	0	Daten werden empfangen.
8210h	0	0	1	Die Hardware ist inkompatibel zur Baustein-Bibliothek Modbus RTU/ TCP.
8381h	0	0	1	Server liefert *Exception Code 01h*. ↯ *395*
8382h	0	0	1	Server liefert *Exception Code 03h* oder falsche Startadresse. ↯ *395*
8383h	0	0	1	Server liefert *Exception Code 02h*. ↯ *395*
8384h	0	0	1	Server liefert *Exception Code 04h*. ↯ *395*
8386h	0	0	1	Server liefert falschen *Funktions-Code*.
8388h	0	0	1	Server liefert falschen Wert oder falsche Quantity.

STATUS	DONE	BUSY	ERROR	Beschreibung
80C8h	0	0	1	Keine Antwort des Servers im definierten Zeitraum (Timeout über Schnittstelle parametrierbar).
8188h	0	0	1	MB_FUNCTION ungültig.
8189h	0	0	1	MB_DATA_ADDR ungültig.
818Ah	0	0	1	MB_DATA_LEN ungültig.
818Bh	0	0	1	MB_DATA_PTR ungültig.
8201h	0	0	1	HARDWARE ungültig.
8202h	0	0	1	MB_UNIT_ID ungültig.
8200h	0	0	1	Eine andere Modbus-Anfrage wird zur Zeit über den Port verarbeitet.

10.2.1.2 Beispiel

Aufgabenstellung

Von einem Modbus-RTU-Slave mit Adresse 99 sollen mit *Funktions-Code 03h*, 100 Register ab Startadresse 2000 gelesen und im Merkerbereich ab MB200 abgelegt werden. Fehler sollen abgespeichert werden. Der Modbus-RTU-Master wird über die interne serielle Schnittstelle einer VIPA-CPU realisiert.

OB100

```
CALL  SFC  216
      Protocol :=B#16#5
      Parameter :=DB10
      Baudrate:=B#16#9
      CharLen:=B#16#3
      Parity:=B#16#2
      StopBits:=B#16#1
      FlowControl:=B#16#1
      RetVal:=MW100
```

OB100 - Beschreibung

1. Aufruf von SFC 216 (SER_CFG) zur Konfiguration der internen seriellen Schnittstelle der VIPA-CPU.
2. Protokoll: "Modbus Master RTU", 9600 Baud, 8 Datenbit, 1 Stoppbit, gerade Parität, kein Flusskontrolle.
3. Der DB10 enthält eine Variable vom Typ WORD mit einem Modbus-Timeout (Wert in ms).

OB1

```
CALL  FB    72 , DB72
       REQ          :=M101.0
       HARDWARE     :=B#16#2
       LADDR        :=
       MB_UNIT_ID   :=B#16#63
       MB_FUNCTION  :=B#16#3
       MB_DATA_ADDR:=W#16#7D0
       MB_DATA_LEN  :=100
       MB_DATA_PTR  :=P#M 200.0 BYTE 200
       DONE         :=M101.1
       BUSY         :=
       ERROR        :=M101.2
       STATUS       :=MW102

      UN    M    101.2
      SPB   ERR1
      L     MW   102
```

```
      T     MW   104
ERR1: NOP   0
      U     M    101.1
      R     M    101.0
```

OB1 - Beschreibung

1. Aufruf des Modbus-RTU-Master Hantierungsbausteins mit den korrekten Parametern.
2. Wenn die Schnittstelle im OB 100 korrekt initialisiert wurde, ist der Master einsatzbereit und MW102 liefert 7004h zurück.
3. M101.0 in der CPU auf TRUE setzen.

   ⇨ Der Modbus-Request wird versendet und auf eine Antwort gewartet.

   Wenn M101.0 automatisch zurück gesetzt wird, wurde der Auftrag fehlerfrei bearbeitet und die gelesenen Daten liegen ab Merkerbyte 200 in der CPU. MW104 liefert 7004h und signalisiert die Bereitschaft für einen neuen Auftrag.

   Wenn M101.0 nicht automatisch zurück gesetzt wird und MW104 einen Wert ungleich 0 liefert, ist ein Fehler aufgetreten. Die Fehlerursache kann über den Code in MW104 ausgelesen werden (z.B. MW104 = 8382h wenn die Startadresse 2000 im Server nicht vorhanden ist). MW102 liefert 7004h und signalisiert die Bereitschaft für einen neuen Auftrag.

10.2.2 FB 73 - RTU_MB_SLAVE - Modbus-RTU-Slave

10.2.2.1 Beschreibung

Dieser Funktionsbaustein ermöglicht den Betrieb der internen seriellen RS485 Schnittstelle einer SPEED7 VIPA-CPU oder eines VIPA System SLIO CP 040 als Modbus-RTU-Slave.

Aufrufparameter

Name	Deklaration	Typ	Beschreibung
ENABLE	IN	BOOL	Aktivierung/Deaktivierung des Modbus-Server.
HARDWARE	IN	BYTE	1 = System SLIO CP 040 / 2 = SPEED7 VIPA-CPU
LADDR	IN	INT	Logische Adresse vom System SLIO CP 040 (Parameter wird für SPEED7 VIPA-CPU ignoriert).
MB_UNIT_ID	IN	BYTE	Modbus: Geräteidentifikation = eigene Adresse (1 ... 247).
MB_DATA_PTR	IN	ANY	Modbus: Datenpuffer (nur Merkerbereich oder Datenbaustein vom Datentyp Byte zulässig) für den Zugriff mit *Funktions-Code 03h*, *06h* und *10h*.
NDR*	OUT	BOOL	Neue Daten wurden durch den Modbus-Client geschrieben.
DR*	OUT	BOOL	Daten wurden vom Modbus-Client gelesen.
ERROR*	OUT	BOOL	Auftrag fertig mit Fehler - Parameter *STATUS* enthält die Fehlerinformation.
STATUS*	OUT	WORD	Erweiterte Status- und Fehlerinformationen.

*) Parameter steht bis zum nächsten Aufruf des FBs zur Verfügung

Parameter im Instanz-DB

Name	Deklaration	Typ	Beschreibung
INIT	STAT	BOOL	Eine Flanke 0-1 führt einen Synchron Reset am System SLIO CP 040 durch.
REQUEST_COUNT	STAT	WORD	Zähler für jedes empfangene Telegramm.
MESSAGE_COUNT	STAT	WORD	Zähler für jeden gültigen Modbus-Request.
BROADCAST_COUNT	STAT	WORD	Zähler für jeden gültigen Modbus-Broadcast-Request.
EXCEPTION_COUNT	STAT	WORD	Zähler für jeden negativ quittierten Modbus-Request.
SUCCESS_COUNT	STAT	WORD	Zähler für jeden positiv quittierten Modbus-Request.
BAD_CRC_COUNT	STAT	WORD	Zähler für jeden gültigen Modbus-Request mit CRC-Fehler.
FC1_ADDR_OUTPUT_START	STAT	WORD	Modbus *Funktions-Code 01h* Startregister für A0.0 Default: 0
FC1_ADDR_OUTPUT_END	STAT	WORD	Modbus *Funktions-Code 01h* Endregister für Ax.y Default: 19999
FC1_ADDR_MEMORY_START	STAT	WORD	Modbus *Funktions-Code 01h* Startregister für M0.0 Default: 20000
FC1_ADDR_MEMORY_END	STAT	WORD	Modbus *Funktions-Code 01h* Endregister für Mx.y Default: 39999
FC2_ADDR_INPUT_START	STAT	WORD	Modbus *Funktions-Code 02h* Startregister für E0.0 Default: 0
FC2_ADDR_INPUT_END	STAT	WORD	Modbus *Funktions-Code 02h* Endregister für Ex.y Default: 19999
FC2_ADDR_MEMORY_START	STAT	WORD	Modbus *Funktions-Code 02h* Startregister für M0.0 Default: 20000
FC2_ADDR_MEMORY_END	STAT	WORD	Modbus *Funktions-Code 02h* Endregister für Mx.y Default: 39999
FC4_ADDR_INPUT_START	STAT	WORD	Modbus *Funktions-Code 04h* Startregister für EW0 Default: 0
FC4_ADDR_INPUT_END	STAT	WORD	Modbus *Funktions-Code 04h* Endregister für EWx Default: 19999
FC4_ADDR_MEMORY_START	STAT	WORD	Modbus *Funktions-Code 04h* Startregister für MW0 Default: 20000
FC4_ADDR_MEMORY_END	STAT	WORD	Modbus *Funktions-Code 04h* Endregister für MWx Default: 39999
FC5_ADDR_OUTPUT_START	STAT	WORD	Modbus *Funktions-Code 05h* Startregister für A0.0 Default: 0
FC5_ADDR_OUTPUT_END	STAT	WORD	Modbus *Funktions-Code 05h* Endregister für Ax.y Default: 19999

Name	Deklaration	Typ	Beschreibung
FC5_ADDR_MEMORY_START	STAT	WORD	Modbus *Funktions-Code 05h* Startregister für M0.0 Default: 20000
FC5_ADDR_MEMORY_END	STAT	WORD	Modbus *Funktions-Code 05h* Endregister für Mx.y Default: 39999
FC15_ADDR_OUTPUT_START	STAT	WORD	Modbus *Funktions-Code 0Fh* Startregister für A0.0 Default: 0
FC15_ADDR_OUTPUT_END	STAT	WORD	Modbus *Funktions-Code 0Fh* Endregister für Ax.y Default: 19999
FC15_ADDR_MEMORY_START	STAT	WORD	Modbus *Funktions-Code 0Fh* Startregister für M0.0 Default: 20000
FC15_ADDR_MEMORY_END	STAT	WORD	Modbus *Funktions-Code 0Fh* Endregister für Mx.y Default: 39999

Hierbei ist folgendes zu beachten:

- Die *Aufrufparameter* sind beim Baustein-Aufruf anzugeben. Neben den *Aufrufparametern* finden Sie alle Parameter im Instanz-DB.
- Die verwendete Schnittstelle muss zuvor konfiguriert werden:
  - VIPA System SLIO CP 040: Projektierung als ASCII-Modul mit 60 Byte IO-Size in der Hardwarekonfiguration.
  - Interne serielle RS485 Schnittstelle einer VIPA-CPU:
    Projektierung über SFC 216 (SER_CFG) mit Protokoll "ASCII".
- FB 60 SEND und FB 61 RECEIVE (oder FB 65 SEND_RECV) sind für die Verwendung des Baustein zwingend erforderlich, auch wenn die interne serielle RS485 Schnittstelle einer VIPA-CPU verwendet wird.
- Broadcast Request über MB_UNIT_ID = 0 werden nur für schreibende Funktionen akzeptiert.
- Die INPUT/OUTPUT Modbus-Adressen eines *Funktions-Codes* müssen vor den MEMORY Modbus-Adressen liegen und somit immer kleiner sein.
- Innerhalb eines *Funktions-Codes* darf keine Modbus-Adresse mehrfach definiert werden, auch die 0 nicht!
- Der Slave kann nur einen Auftrag gleichzeitig bearbeiten. Neue Modbus-Anfragen während einer Auftragsbearbeitung werden ignoriert und nicht beantwortet.

Status- und Fehleranzeige

Der Funktionsbaustein liefert über STATUS die folgenden Status- und Fehlerinformationen.

STATUS	NDR	DR	ERROR	Beschreibung
0000h	0 oder 1*		0	Anweisung fehlerfrei ausgeführt.
7000h	0	0	0	Keine Verbindung aufgebaut oder Kommunikationsfehler.
7005h	0	0	0	Daten werden gesendet.
7006h	0	0	0	Daten werden empfangen.
8210h	0	0	1	Die Hardware ist inkompatibel zur Baustein-Bibliothek Modbus RTU/TCP.
8380h	0	0	1	CRC-Fehler

STATUS	NDR	DR	ERROR	Beschreibung
8381h	0	0	1	*Exception Code 01h*, *Funktions-Code* wird nicht unterstützt. ↳ *395*
8382h	0	0	1	*Exception Code 03h*, Datenlänge oder Datenwert ungültig. ↳ *395*
8383h	0	0	1	*Exception Code 02h*, Ungültige Startadresse bzw. Adressbereich. ↳ *395*
8384h	0	0	1	*Exception Code 04h*, Bereichslängenfehler beim Zugriff auf Eingänge, Ausgänge oder Merker ↳ *395*
8187h	0	0	1	MB_DATA_PTR ungültig.
8201h	0	0	1	HARDWARE ungültig.
8202h	0	0	1	MB_UNIT_ID ungültig.
8203h	0	0	1	Fragmentiertes Empfangstelegramm ungültig (SFC 218).

*) Fehlerfreier Modbus-Auftrag mit *Funktions-Code 05h, 06h, 0Fh* oder *10h* liefert NDR=1 und DR=0 bzw. fehlerfreier Modbus-Auftrag mit *Funktons-Code 01h, 02h, 03h, 04h* liefert DR=1 und NDR=0.

10.2.2.2 Beispiel

Aufgabenstellung

Die CPU stellt 100 Byte Daten im Merkerbereich ab MB200 für einen Modbus-Master über die Modbus-Register 0 ... 49 zur Verfügung. Die Daten können vom Modbus-Master über *Funktions-Code 03h* gelesen und über *Funktions-Code 06h, 10h* geschrieben werden. Der Ausgang A1.0 in der CPU soll von einem Modbus-Master über den *Funktions-Code 05h* und die Startadresse 5008 angesteuert werden können. Fehler sollen abgespeichert werden. Der Modbus-RTU-Slave mit der Adresse 99 wird über die interne serielle Schnittstelle einer VIPA-CPU realisiert.

OB100

```
CALL  SFC  216
      Protocol :=B#16#1
      Parameter :=DB10
      Baudrate:=B#16#9
      CharLen:=B#16#3
      Parity:=B#16#2
      StopBits:=B#16#1
      FlowControl:=B#16#1
      RetVal:=MW100
```

OB100 - Beschreibung

1. Aufruf von SFC 216 (SER_CFG) zur Konfiguration der internen seriellen Schnittstelle der VIPA-CPU.
2. Protokoll: "ASCII", 9600 Baud, 8 Datenbit, 1 Stoppbit, gerade Parität, kein Flusskontrolle.
3. Der DB10 enthält eine Variable vom Typ WORD und muss als "Dummy" übergeben werden.

OB1

```
      L     5000
      T     DB73.DBW   58

      CALL  FB    73 , DB73
       ENABLE     :=M101.0
       HARDWARE   :=B#16#2
       LADDR      :=
       MB_UNIT_ID :=B#16#63
       MB_DATA_PTR:=P#M 200.0 BYTE 100
```

```
          NDR         :=M101.1
          DR          :=M101.2
          ERROR       :=M101.3
          STATUS      :=MW102

        UN    M    101.3
        SPB   ERR1
        L     MW   102
        T     MW   104
ERR1: NOP   0
```

OB1 - Beschreibung

1. Aufruf des Modbus/TCP-Server Hantierungsbausteins mit den korrekten Parametern.
2. Wenn die Schnittstelle im OB100 korrekt initialisiert wurde, ist der Slave einsatzbereit und MW102 wird zu 7006h geliefert.
3. Das Modbus-Startregister für die über *Funktions-Code 05h* erreichbaren Ausgänge im Prozessabbild wird im Beispiel über den Parameter FC5_ADDR_OUTPUT_START (Wort 58 im Instanz-Datenbaustein) geändert.
4. M101.0 in der CPU auf TRUE setzen.

   ⇨ Der Modbus-Slave arbeitet nun.
5. Der Master sendet einen Modbus-Request mit *Funktions-Code 03h*, Startadresse 10 und Quantity 30.

   ⇨ Der Slave antwortet mit 60Byte ab MB200. DR wird für einen CPU-Zyklus angesteuert und somit M101.2 auf "1" gesetzt.
6. Der Master sendet einen Modbus-Request mit *Funktions-Code 05h*, Startadresse 5008 und dem Wert FF00h.

   ⇨ Der Slave quittiert den Auftrag und schreibt den Ausgang A1.0 auf "1". NDR wird für einen CPU-Zyklus angesteuert und somit M101.1 auf "1" gesetzt.
7. Der Master sendet einen Modbus-Request mit *Funktions-Code 03h*, Startadresse 50 (nicht vorhanden!) und Quantity 1.

   ⇨ Der Slave antwortet mit einem *Exception Code 02h* und steuert ERROR/ STATUS für einen CPU-Zyklus an. MW104 liefert 8383h.

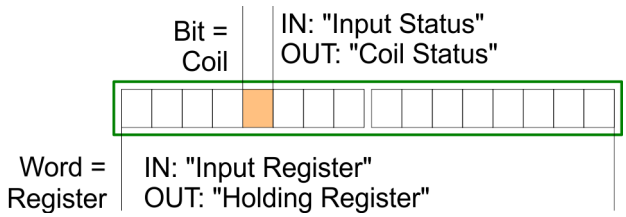
10.3 Modbus Exception Codes - *Exception Codes*

Code	Name	Beschreibung
01	ILLEGAL FUNCTION	Der in der Abfrage empfangene Funktionscode ist keine zulässige Aktion für den Server (oder Slave). Dies kann daran liegen, dass der Funktionscode nur für neuere Geräte gilt und nicht im ausgewählten Modul implementiert wurde. Dies könnte auch bedeuten, dass der Server (oder Slave) sich im falschen Zustand befindet, um eine Anforderung dieses Typs verarbeiten zu können, zum Beispiel da er nicht konfiguriert ist und Registerwerte auslesen soll.
02	ILLEGAL DATA ADDRESS	Der in der Abfrage empfangene Funktionscode ist keine zulässige Aktion für den Server (oder Slave). Insbesondere ist die Kombination aus Referenznummer und Übertragungslänge ungültig. Bei einem Controller mit 100 Registern adressiert die PDU (**P**rotocol **D**ata **U**nit) das erste Register mit 0 und das letzte mit 99. Wenn eine Anfrage mit einer Startregisteradresse von 96 und einer Anzahl von 4 Registern gesendet wird, wird diese Anfrage erfolgreich (zumindest adressenmäßig) in den Registern 96, 97, 98, 99 ausgeführt. Wenn eine Anfrage mit einer Startregisteradresse von 96 und einer Anzahl von 5 Registern gesendet wird, schlägt diese Anfrage mit dem *Exception Code* 0x02 "ILLEGAL DATA ADDRESS" fehl, da versucht wird, die Register 96, 97, 98, 99 und 100 zu bearbeiten. Es gibt aber kein Register mit Adresse 100.
03	ILLEGAL DATA VALUE	Ein Wert in der Datenabfrage ist kein zulässiger Wert für Server (oder Slave). Dies weist auf einen Fehler in der Struktur einer komplexen Datenanforderung hin, z.B. dass die implizierte Länge falsch ist. Dies bedeutet aber NICHT, dass ein zur Speicherung in einem Register übermittelter Datenwert einen Wert hat, welcher außerhalb eines erwarteten Wertes des Anwendungsprogramms liegt, da das Modbus-Protokoll die Bedeutung eines bestimmten Wertes eines bestimmten Registers nicht kennt.
04	SLAVE DEVICE FAILURE	Ein nicht behebbarer Fehler ist aufgetreten während der Server (oder Slave) die angeforderte Aktion ausgeführt hat.
05	ACKNOWLEDGE	Spezieller Einsatz in Verbindung mit Programmierbefehlen. Der Server (oder Slave) hat die Anforderung angenommen und verarbeitet sie. Dies kann länger dauern. Diese Antwort soll im Client (oder Master) einen Timeout-Fehler verhindern. Der Client (oder Master) kann danach zur Ermittlung, ob die Verarbeitung abgeschlossen ist, eine Poll Program Complete Nachricht ausgeben.
06	SLAVE DEVICE BUSY	Spezieller Einsatz in Verbindung mit Programmierbefehlen. Der Server (oder Slave) bearbeitet einen länger dauernden Programmbefehl. Der Client (oder Master) sollte die Nachricht später erneut senden, wenn der Server (oder Slave) frei ist.
08	MEMORY PARITY ERROR	Speziell in Verbindung mit den Funktionscodes 20 und 21 und dem Referenztyp 6, um anzuzeigen, dass der erweiterte Dateibereich die Konsistenzprüfung nicht bestanden hat.
0A	GATEWAY PATH UNAVAILABLE	Zeigt speziell in Verbindung mit Gateways an, dass das Gateway für die Verarbeitung der Anforderung keinen internen Kommunikationspfad vom Eingabeport zum Ausgabeport zuordnen konnte. In der Regel bedeutet dies, dass das Gateway falsch konfiguriert bzw. überlastet ist.
0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	Die spezielle Verwendung in Verbindung mit Gateways zeigt an, dass keine Antwort vom Zielgerät erhalten wurde. In der Regel bedeutet dies, dass das Gerät nicht im Netzwerk vorhanden ist.

10.4 FKT Modbus-Funktionscodes - *FKT Codes*

Namenskonventionen

Für Modbus gibt es Namenskonventionen, die hier kurz aufgeführt sind:



- Modbus unterscheidet zwischen Bit- und Wortzugriff; Bits = "Coils" und Worte = "Register".
- Bit-Eingänge werden als "Input-Status" bezeichnet und Bit-Ausgänge als "Coil-Status".
- Wort-Eingänge werden als "Input-Register" und Wort-Ausgänge als "Holding-Register" bezeichnet.

Bereichsdefinitionen

Üblicherweise erfolgt unter Modbus der Zugriff mittels der Bereiche 0x, 1x, 3x und 4x.

Mit 0x und 1x haben Sie Zugriff auf digitale Bit-Bereiche und mit 3x und 4x auf analoge Wort-Bereiche.

Da aber bei den VIPA-CPs keine Unterscheidung zwischen Digital- und Analogdaten stattfindet, gilt folgende Zuordnung:

0x - Bit-Bereich für Ausgabe-Daten des Masters

Zugriff über Funktions-Code 01h, 05h, 0Fh

1x - Bit-Bereich für Eingabe-Daten des Masters

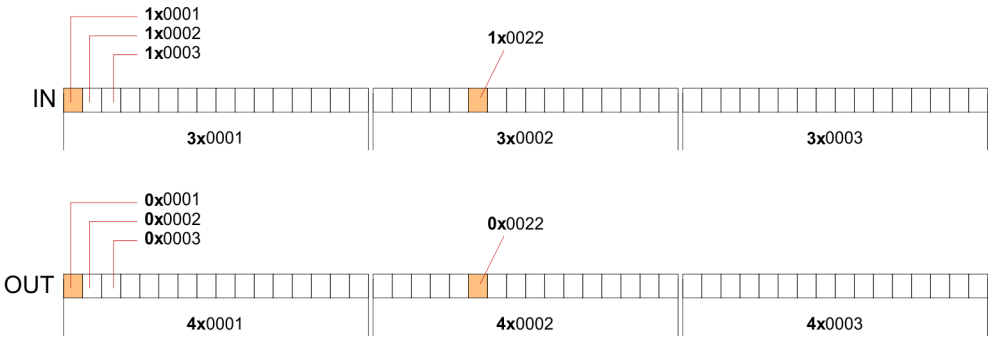
Zugriff über Funktions-Code 02h

3x - Wort-Bereich für Eingabe-Daten des Masters

Zugriff über Funktions-Code 04h

4x - Wort-Bereich für Ausgabe-Daten des Masters

Zugriff über Funktions-Code 03h, 06h, 10h, 16h



Übersicht

Mit folgenden Funktionscodes können Sie von einem Modbus-Master auf einen Slave zugreifen. Die Beschreibung erfolgt immer aus Sicht des Masters:

Code	Befehl	Beschreibung
01h	Read n Bits	n Bit lesen von Master-Ausgabe-Bereich 0x
02h	Read n Bits	n Bit lesen von Master-Eingabe-Bereich 1x

Code	Befehl	Beschreibung
03h	Read n Words	n Worte lesen von Master-Ausgabe-Bereich 4x
04h	Read n Words	n Worte lesen von Master-Eingabe-Bereich 3x
05h	Write 1 Bit	1 Bit schreiben in Master-Ausgabe-Bereich 0x
06h	Write 1 Word	1 Wort schreiben in Master-Ausgabe-Bereich 4x
0Fh	Write n Bits	n Bit schreiben in Master-Ausgabe-Bereich 0x
10h	Write n Words	n Worte schreiben in Master-Ausgabe-Bereich 4x
16h	Mask 1 Word	1 Wort in Master-Ausgabe-Bereich 4x maskieren

Byte-Reihenfolge im Wort

1 Wort	
High Byte	Low Byte

Antwort des Kopplers

Liefert der Slave einen Fehler zurück, so wird der Funktionscode mit 80h "verodert" zurückgesendet. Ist kein Fehler aufgetreten, wird der Funktionscode zurückgeliefert.

Slave-Antwort:	Funktionscode OR 80h	→ Fehler & Fehlernummer
	Funktionscode	→ OK

Zusätzlich erhalten Sie im Fehlerfall in einem weiteren Byte eine Fehlernummer. Hier gibt es folgende Fehlernummern:

01h: Funktionsnummer wird nicht unterstützt

02h: Adressierung fehlerhaft

03h: Daten fehlerhaft

04h: System SLIO Bus ist nicht initialisiert

07h: Allgemeiner Fehler

Read n Bits 01h, 02h

Code 01h: n Bit lesen von Master-Ausgabe-Bereich 0x.

Code 02h: n Bit lesen von Master-Eingabe-Bereich 1x.

Kommandotelegramm

Modbus/TCP-Header						Slave-Adresse	Funktions-Code	Adresse 1. Bit	Anzahl der Bits
x	x	0	0	0	6				
6Byte						1Byte	1Byte	1Wort	1Wort

Antworttelegramm

Modbus/TCP-Header						Slave-Adresse	Funktions-Code	Anzahl gelesene Bytes	Daten 1. Byte	Daten 2. Byte	...
x	x	0	0	0							
6Byte						1Byte	1Byte	1Byte	1Byte	1Byte	
									max. 252Byte		

Read n Words 03h, 04h

03h: n Worte lesen von Master-Ausgabe-Bereich 4x.

04h: n Worte lesen von Master-Eingabe-Bereich 3x.

Kommandotelegramm

Modbus/TCP-Header						Slave-Adresse	Funktions-Code	Adresse Wort	Anzahl der Worte
x	x	0	0	0	6				
6Byte						1Byte	1Byte	1Wort	1Wort

Antworttelegramm

Modbus/TCP-Header						Slave-Adresse	Funktions-Code	Anzahl gelesene Bytes	Daten 1. Wort	Daten 2. Wort	...
x	x	0	0	0							
6Byte						1Byte	1Byte	1Byte	1Wort	1Wort	
									max. 126Worte		

Write 1 Bit 05h

Code 05h: 1 Bit schreiben in Master-Ausgabe-Bereich 0x.

Eine Zustandsänderung erfolgt unter "Zustand Bit" mit folgenden Werten:

"Zustand Bit" = 0000h → Bit = 0

"Zustand Bit" = FF00h → Bit = 1

Kommandotelegramm

Modbus/TCP-Header						Slave-Adresse	Funktions-Code	Adresse Bit	Zustand Bit
x	x	0	0	0	6				
6Byte						1Byte	1Byte	1Wort	1Wort

Antworttelegramm

Modbus/TCP-Header						Slave-Adresse	Funktions-Code	Adresse Bit	Zustand Bit
x	x	0	0	0	6				
6Byte						1Byte	1Byte	1Wort	1Wort

Write 1 Word 06h

Code 06h: 1 Wort schreiben in Master-Ausgabe-Bereich 4x.

Kommandotelegramm

Modbus/TCP-Header						Slave-Adresse	Funktions-Code	Adresse Wort	Wert Wort
x	x	0	0	0	6				
6Byte						1Byte	1Byte	1Wort	1Wort

Antworttelegramm

Modbus/TCP-Header						Slave-Adresse	Funktions-Code	Adresse Wort	Wert Wort
x	x	0	0	0	6				
6Byte						1Byte	1Byte	1Wort	1Wort

Write n Bits 0Fh

Code 0Fh: n Bit schreiben in Master-Ausgabe-Bereich 0x

Bitte beachten Sie, dass die Anzahl der Bits zusätzlich in Byte anzugeben sind.

Kommandotelegramm

Modbus/TCP-Header						Slave-Adresse	Funk-tions-Code	Adresse 1. Bit	Anzahl Bits	Anzahl Bytes	Daten 1. Byte	Daten 2. Byte	...
x	x	0	0	0									
6Byte						1Byte	1Byte	1Wort	1Wort	1Byte	1Byte	1Byte	1Byte
											max. 248Byte		

Antworttelegramm

Modbus/TCP-Header						Slave-Adresse	Funktions-Code	Adresse 1.Bit	Anzahl Bits
x	x	0	0	0	6				
6Byte						1Byte	1Byte	1Wort	1Wort

Write n Words 10h

Code 10h: n Worte schreiben in Master-Ausgabe-Bereich.

Kommandotelegramm

Modbus/TCP-Header						Slave-Adresse	Funk-tions-Code	Adresse 1. Wort	Anzahl Worte	Anzahl Bytes	Daten 1. Wort	Daten 2. Wort	...
x	x	0	0	0									
6Byte						1Byte	1Byte	1Wort	1Wort	1Wort	1Wort	1Wort	1Wort
											max. 124Byte		

Antworttelegramm

Modbus/TCP-Header						Slave-Adresse	Funktions-Code	Adresse 1. Wort	Anzahl Worte
x	x	0	0	0	6				
6Byte						1Byte	1Byte	1Wort	1Wort

Mask 1 Word 16h

Code 16h: Mit dieser Funktion können Sie ein Wort im Master-Ausgabe-Bereich 4x maskieren.

Kommandotelegramm

Modbus/TCP-Header						Slave-Adresse	Funktions-Code	Adresse Wort	AND Mask	OR Mask
x	x	0	0	0	8					
6Byte						1Byte	1Byte	1Wort	1Wort	1Wort

Antworttelegramm

Modbus/TCP-Header						Slave-Adresse	Funktions-Code	Adresse Wort	AND Mask	OR Mask
x	x	0	0	0	8					
6Byte						1Byte	1Byte	1Wort	1Wort	1Wort

11 Serielle Kommunikation - *"Serial Communication"*

11.1 Serielle Kommunikation - *"Serial Communication"*

11.1.1 SFC 207 - SER_CTRL - Modemfunktionalität PtP

Beschreibung

Bitte beachten Sie, dass dieser Baustein von SPEED7 CPUs nicht unterstützt wird!

Bei Einsatz des ASCII-Protokolls über die RS232-Schnittstelle haben Sie mit diesem Baustein zur Laufzeit Zugriff auf die seriellen Modemleitungen. Abhängig vom Parameter *FLOWCONTROL*, den Sie über *SFC 216 (SER_CFG)* vorgeben, bietet der Baustein folgende Funktionalität:

	Lesen	Schreiben
FLOWCONTROL=0:	DTR, RTS, DSR, RI, CTS, CD	DTR, RTS
FLOWCONTROL>0:	DTR, RTS, DSR, RI, CTS, CD	nicht möglich

Parameter

Name	Deklaration	Typ	Beschreibung
WRITE	IN	BYTE	■ Bit 0: Neuer Zustand DTR ■ Bit 1: Neuer Zustand RTS
MASKWRITE	IN	BYTE	■ Bit 0: Zustand an DTR übergeben ■ Bit 1: Zustand an RTS übergeben
READ	OUT	BYTE	Status (CTS, DSR, RI, CD, DTR, RTS)
READDELTA	OUT	BYTE	Status Änderung seit letztem Zugriff
RETVAL	OUT	WORD	Rückgabewert (0 = OK)

WRITE

Mit diesem Parameter geben Sie den Status für DTR und RTS vor, den Sie über *MASKWRITE* aktivieren können. Das Byte hat folgende Belegung:

- Bit 0 = DTR
- Bit 1 = RTS
- Bit 7 ... Bit 2: reserviert

MASKWRITE

Hier wird mit "1" der Status des entsprechenden Parameters übernommen. Das Byte hat folgende Belegung:

- Bit 0 = DTR
- Bit 1 = RTS
- Bit 7 ... Bit 2: reserviert

READ

READ liefert den aktuellen Status der Modem-Leitungen zurück. *READDELTA* liefert den Status der Modem-Leitungen zurück, die sich seit dem letzten Zugriff geändert haben. Die Bytes haben folgenden Aufbau:

Bit-Nr.	7	6	5	4	3	2	1	0
Read	x	x	RTS	DTR	CD	RI	DSR	CTS
ReadDelta	x	x	x	x	CD	RI	DSR	CTS

RETVAL (Rückgabewert)

Wert	Beschreibung
0000h	kein Fehler
8x24h	Fehler in SFC-Parameter x, mit x: ■ 1: Fehler in *WRITE* ■ 2: Fehler in *MASKWRITE* ■ 3: Fehler in *READ* ■ 4: Fehler in *READDELTA*
809Ah	Schnittstelle ist nicht vorhanden
809Bh	Schnittstelle ist nicht konfiguriert (SFC 216)

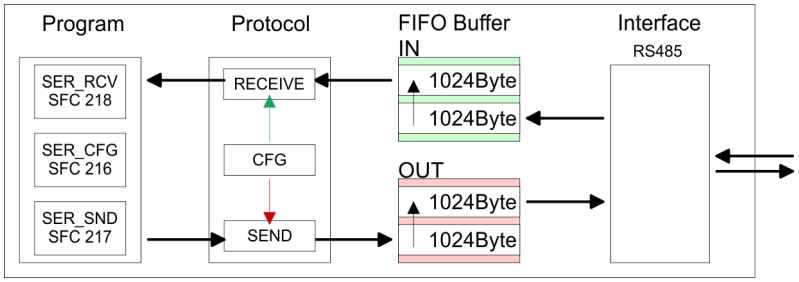
11.1.2 FC/SFC 216 - SER_CFG - Parametrierung PtP

Beschreibung

Über eine Hardware-Konfiguration können Sie unter Objekteigenschaften über den Parameter "Funktion RS485" den in der SPEED7-CPU integrierten DP-Master deaktivieren und die RS485-Schnittstelle für PtP-Kommunikation (point to point) freigeben. Die RS485-Schnittstelle im PtP-Betrieb ermöglicht die serielle Prozessankopplung zu verschiedenen Ziel- oder Quellsystemen. Die Parametrierung erfolgt zur Laufzeit unter Einsatz des FC/SFC 216 (SER_CFG). Hierbei sind für alle Protokolle mit Ausnahme von ASCII die Parameter in einem DB abzulegen.

Kommunikation

- Daten, die von der CPU in den entsprechenden Datenkanal geschrieben werden, werden in einen FIFO-Sendepuffer (first in first out) mit einer Größe von 2x1024Byte abgelegt und von dort über die Schnittstelle ausgegeben.
- Empfängt die Schnittstelle Daten, werden diese in einem FIFO-Empfangspuffer mit einer Größe von 2x1024Byte abgelegt und können dort von der CPU gelesen werden.
- Sofern Daten mittels eines Protokolls übertragen werden, erfolgt die Einbettung der Daten in das entsprechende Protokoll automatisch. Im Gegensatz zu ASCII und STX/ETX erfolgt bei den Protokollen 3964R, USS und Modbus die Datenübertragung mit Quittierung der Gegenseite.
- Durch erneuten Aufruf des FC/SFC 217 SER_SND bekommen Sie über RETVAL einen Rückgabewert geliefert, der unter anderem auch aktuelle Informationen über die Quittierung der Gegenseite beinhaltet. Zusätzlich ist bei USS und Modbus nach einem SER_SND das Quittungstelegramm durch Aufruf des FC/SFC 218 SER_RCV auszulesen.



Parameter

Parameter	Deklaration	Datentyp	Beschreibung
PROTOCOL	IN	BYTE	1=ASCII, 2=STX/ETX, 3=3964R
PARAMETER	IN	ANY	Zeiger zu den Protokoll-Parametern
BAUDRATE	IN	BYTE	Nr. der Baudrate
CHARLEN	IN	BYTE	0=5Bit, 1=6Bit, 2=7Bit, 3=8Bit
PARITY	IN	BYTE	0=Non, 1=Odd, 2=Even
STOPBITS	IN	BYTE	1=1Bit, 2=1,5Bit, 3=2Bit
FLOWCONTROL	IN	BYTE	1 - siehe Hinweis
RETVAL	OUT	WORD	Rückgabewert (0 = OK)

Alle Zeitangaben für Timeouts sind als Hexadezimaler Wert anzugeben. Den Hex-Wert erhalten Sie, indem Sie die gewünschte Zeit in Sekunden mit der Baudrate multiplizieren.

Beispiel:

- Gewünschte Zeit 8ms bei einer Baudrate von 19200Baud
- Berechnung: 19200Bit/s x 0,008s ≈ 154Bit → (9Ah)
- Als Hex-Wert ist 9Ah vorzugeben.

PROTOCOL

Geben Sie hier das Protokoll an, das verwendet werden soll. Zur Auswahl stehen:

- 1: ASCII
- 2: STX/ETX
- 3: 3964R
- 4: USS Master
- 5: Modbus RTU Master
- 6: Modbus ASCII Master

PARAMETER (als DB)

Bei eingestelltem ASCII-Protokoll wird dieser Parameter ignoriert. Für die Protokolle geben Sie hier einen DB an, der die Kommunikationsparameter beinhaltet und für die jeweiligen Protokolle STX/ETX, 3964R, USS und Modbus folgenden Aufbau hat:

Datenbaustein bei STX/ETX			
DBB0:	STX1	BYTE	(1. Start-Zeichen in hexadezimaler Form)
DBB1:	STX2	BYTE	(2. Start-Zeichen in hexadezimaler Form)
DBB2:	ETX1	BYTE	(1. Ende-Zeichen in hexadezimaler Form)
DBB3:	ETX2	BYTE	(2. Ende-Zeichen in hexadezimaler Form)
DBW4:	TIMEOUT	WORD	(max. zeitlicher Abstand zwischen 2 Tele-grammen)

Das Zeichen für Start bzw. Ende sollte immer ein Wert kleiner 20 sein, ansonsten wird das Zeichen ignoriert!

Tragen Sie immer für nicht benutzte Zeichen FFh ein!

Datenbaustein bei 3964R			
DBB0:	Prio	BYTE	(Die Priorität beider Partner muss unterschiedlich sein)
DBB1:	ConnAttmptNr	BYTE	(Anzahl der Verbindungsaufbauversuche)
DBB2:	SendAttmptNr	BYTE	(Anzahl der Telegrammwiederholungen)
DBB4:	CharTimeout	WORD	(Zeichenverzugszeit)
DBW6:	ConfTimeout	WORD	(Quittungsverzugszeit)

Datenbaustein bei USS			
DBW0:	Timeout	WORD	(Verzugszeit)

Datenbaustein bei Modbus-Master			
DBW0:	Timeout	WORD	(Antwort-Verzugszeit)

BAUDRATE

Geschwindigkeit der Datenübertragung in Bit/s (Baud).							
04h:	1200Baud	05h:	1800Baud	06h:	2400Baud	07h:	4800Baud
08h:	7200Baud	09h:	9600Baud	0Ah:	14400Baud	0Bh:	19200Baud
0Ch:	38400Baud	0Dh:	57600Baud	0Eh:	115200Baud		

CHARLEN

Anzahl der Datenbits, auf die ein Zeichen abgebildet wird.			
0: 5Bit	1: 6Bit	2: 7Bit	3: 8Bit

PARITY

Die Parität ist je nach Wert gerade oder ungerade. Zur Paritätskontrolle werden die Informationsbits um das Paritätsbit erweitert, das durch seinen Wert ("0" oder "1") den Wert aller Bits auf einen vereinbarten Zustand ergänzt. Ist keine Parität vereinbart, wird das Paritätsbit auf "1" gesetzt, aber nicht ausgewertet.

0: NONE	1: ODD	2: EVEN

STOPBITS

Die Stopbits werden jedem zu übertragenden Zeichen nachgesetzt und kennzeichnen das Ende eines Zeichens.

1: 1Bit	2: 1,5Bit*	3: 2Bit
*) Nur zulässig bei *CHARLEN* = 0 (5Bit)		

FLOWCONTROL

Der Parameter *FLOWCONTROL* wird ignoriert. Beim Senden ist RTS=1, beim Empfangen ist RTS=0.

Sonderfunktion in System MICRO CPU

Ab der Firmware-Version 2.4.4 können Sie in einer System MICRO CPU zwischen RS422- und RS485-Kommunikation umschalten.

0: RS422-Kommunikation

1: RS485-Kommunikation

RETVAL FC/SFC 216 (Rückgabewert)

Rückgabewerte, die der Baustein liefert:

Fehlercode	Beschreibung
0000h	kein Fehler
809Ah	Schnittstelle ist nicht vorhanden bzw. Schnittstelle wird für PROFIBUS verwendet.
8x24h	Fehler in FC/SFC-Parameter x, mit x: 1: Fehler in *PROTOKOLL* 2: Fehler in *PARAMETER* 3: Fehler in *BAUDRATE* 4: Fehler in *CHARLENGTH* 5: Fehler in *PARITY* 6: Fehler in *STOPBITS* 7: Fehler in *FLOWCONTROL* (Parameter fehlt)
809xh	Fehler in Wert des FC/SFC-Parameter x, mit x: 1: Fehler in *PROTOKOLL* 3: Fehler in *BAUDRATE* 4: Fehler in *CHARLENGTH* 5: Fehler in *PARITY* 6: Fehler in *STOPBITS*
8092h	Zugriffsfehler auf Parameter-DB (DB zu kurz)
828xh	Fehler in Parameter x von DB-Parameter mit x: 1: Fehler im 1. Parameter 2: Fehler im 2. Parameter ...

11.1.3 FC/SFC 217 - SER_SND - Senden an PtP

Beschreibung

Mit diesem Baustein werden Daten über die serielle Schnittstelle gesendet. Durch erneuten Aufruf des FC/SFC 217 SER_SND bekommen Sie bei 3964R, USS und Modbus über RETVAL einen Rückgabewert geliefert, der unter anderem auch aktuelle Informationen über die Quittierung der Gegenseite beinhaltet. Zusätzlich ist bei USS und Modbus nach einem SER_SND das Quittungstelegramm durch Aufruf des FC/SFC 218 SER_RCV auszulesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
DATAPTR	IN	ANY	Zeiger auf Sendedaten
DATALEN	OUT	WORD	Länge der Sendedaten
RETVAL	OUT	WORD	Rückgabewert (0 = OK)

DATAPTR

Geben Sie hier einen Bereich vom Typ Pointer für den Sendepuffer an, in den die Daten, die gesendet werden sollen, abzulegen sind. Anzugeben sind Typ, Anfang und Länge.

Beispiel:

- Daten liegen in DB5 ab 0.0 mit einer Länge von 124Byte
- DataPtr:=P#DB5.DBX0.0 BYTE 124

DATALEN

- Wort, in dem die Anzahl der gesendeten Bytes abgelegt wird.
- Werden unter **ASCII** die Daten intern mittels FC/SFC 217 schneller an die serielle Schnittstelle übertragen als sie gesendet werden können, kann aufgrund eines Puferüberlaufs die zu sendende Datenlänge von *DATALEN* abweichen. Dies sollte im Anwenderprogramm berücksichtigt werden!
- Bei **STX/ETX**, **3964R**, **Modbus** und **USS** wird immer die unter *DATAPTR* angegebene Länge oder 0 eingetragen.

RETVAL FC/SFC 217 (Rückgabewerte)

Rückgabewerte, die der Baustein liefert:

Fehler-code	Beschreibung
0000h	Daten gesendet - fertig
1000h	Nichts gesendet (Datenlänge 0)
20xxh	Protokoll wurde fehlerfrei ausgeführt mit xx-Bitmuster für Diagnose
7001h	Daten liegen im internen Puffer - aktiv (busy)
7002h	Transfer - aktiv
80xxh	Protokoll wurde fehlerhaft ausgeführt mit xx-Bitmuster für Diagnose (keine Quittung der Gegenseite)
90xxh	Protokoll wurde nicht ausgeführt mit xx-Bitmuster für Diagnose (keine Quittung der Gegenseite)
8x24h	Fehler in FC/SFC-Parameter x, mit x: 1: Fehler in *DATAPTR* 2: Fehler in *DATALEN*
8122h	Fehler in Parameter *DATAPTR* (z.B. DB zu kurz)
807Fh	Interner Fehler
809Ah	Schnittstelle nicht vorhanden bzw. Schnittstelle wird für PROFIBUS verwendet
809Bh	Schnittstelle nicht konfiguriert

Protokollspezifische RETVAL-Werte

ASCII

Wert	Beschreibung
9000h	Pufferüberlauf (keine Daten gesendet)
9002h	Daten sind zu kurz (0Byte)

STX/ETX

Wert	Beschreibung
9000h	Pufferüberlauf (keine Daten gesendet)
9001h	Daten sind zu lang (>1024Byte)
9002h	Daten sind zu kurz (0Byte)
9004h	Unzulässiges Zeichen

3964R

Wert	Beschreibung
2000h	Senden fertig ohne Fehler
80FFh	NAK empfangen - Fehler in der Kommunikation
80FEh	Datenübertragung ohne Quittierung der Gegenseite oder mit fehlerhafter Quittierung
9000h	Pufferüberlauf (keine Daten gesendet)
9001h	Daten sind zu lang (>1024Byte)
9002h	Daten sind zu kurz (0Byte)

USS

Fehler-code	Beschreibung
2000h	Senden fertig ohne Fehler
8080h	Empfangspuffer voll (kein Platz für Quittung)
8090h	Quittungsverzugszeit überschritten
80F0h	Falsche Checksumme in Rückantwort
80FEh	Falsches Startzeichen in der Rückantwort
80FFh	Falsche Slave-Adresse in der Rückantwort
9000h	Pufferüberlauf (keine Daten gesendet)
9001h	Daten sind zu lang (>1024Byte)
9002h	Daten sind zu kurz (<2Byte)

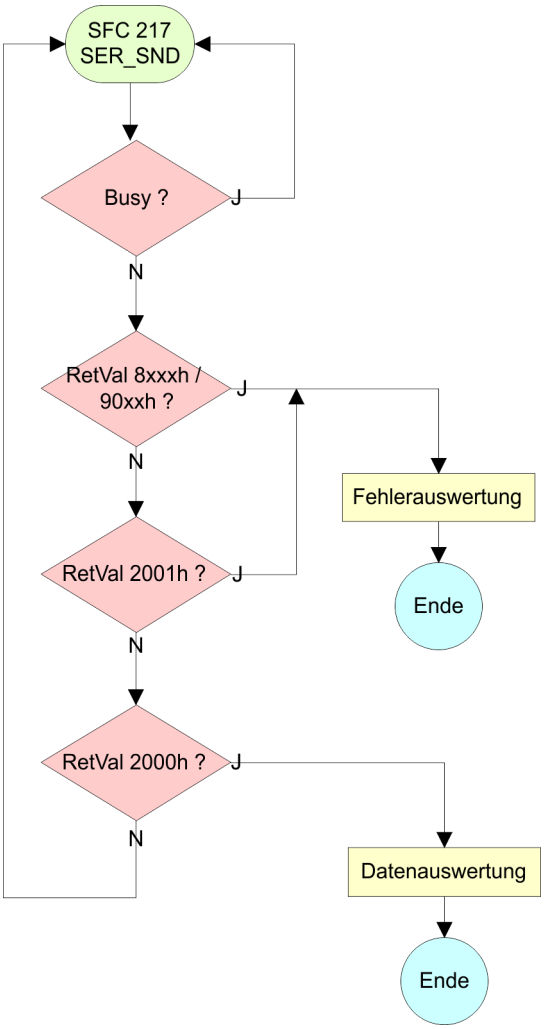
Modbus RTU/ASCII Master

Fehler-code	Beschreibung
2000h	Senden fertig (positive Slave-Rückmeldung vorhanden)
2001h	Senden fertig (negative Slave-Rückmeldung vorhanden)
8080h	Empfangspuffer voll (kein Platz für Quittung)
8090h	Quittungsverzugszeit überschritten
80F0h	Falsche Checksumme in Rückantwort
80FDh	Länge der Rückantwort ist zu lang
80FEh	Falscher Funktionscode in der Rückantwort
80FFh	Falsche Slave-Adresse in der Rückantwort
9000h	Pufferüberlauf (keine Daten gesendet)
9001h	Daten sind zu lang (>1024Byte)
9002h	Daten sind zu kurz (<2Byte)

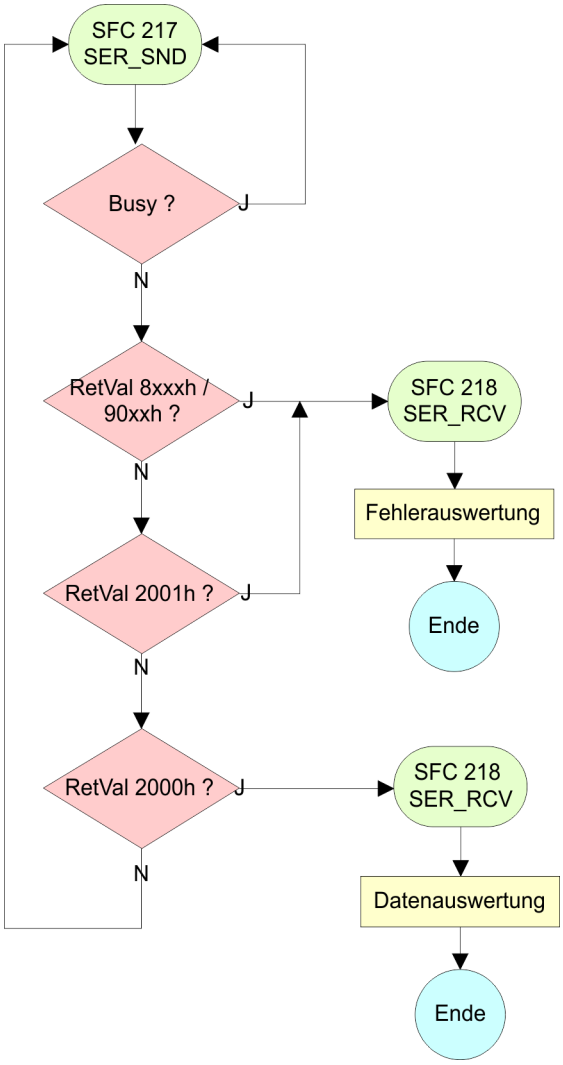
Prinzip der Programmierung

Nachfolgend soll kurz die Struktur zur Programmierung eines Sendeauftrags für die verschiedenen Protokolle gezeigt werden.

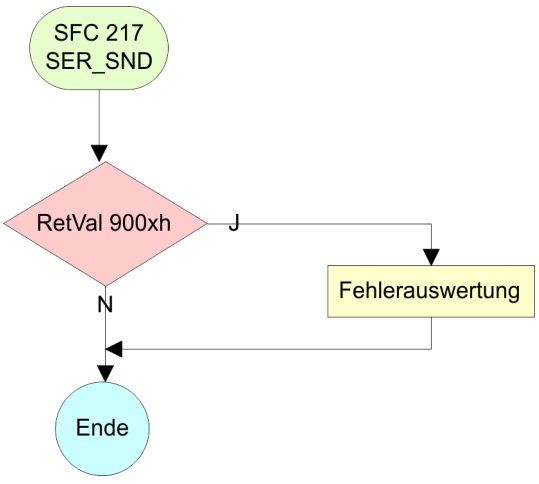
3964R



USS / Modbus



ASCII / STX/ETX



11.1.4 FC/SFC 218 - SER_RCV - Empfangen von PtP

Beschreibung

Mit diesem Baustein werden Daten über die serielle Schnittstelle empfangen. Bei den Protokollen USS und Modbus können Sie durch Aufruf des FC/SFC 218 SER_RCV nach einem SER_SND das Quittungstelegramm auslesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
DATAPTR	IN	ANY	Zeiger auf Empfangspuffer
DATALEN	OUT	WORD	Länge der empfangenen Daten
ERROR	OUT	WORD	Fehler-Nr.
RETVAL	OUT	WORD	Rückgabewert (0 = OK)

DATAPTR

Geben Sie hier einen Bereich vom Typ Pointer für den Empfangspuffer an, in den die Daten, die empfangen werden, abzulegen sind. Anzugeben sind Typ, Anfang und Länge.

Beispiel:

- Daten sind in DB5 ab 0.0 mit einer Länge von 124Byte abzulegen
- DataPtr:=P#DB5.DBX0.0 BYTE 124

DATALEN

- Wort, in dem die Anzahl der empfangenen Bytes abgelegt wird.
- Bei **STX/ETX** und **3964R** wird immer die Länge der empfangenen Nutzdaten oder 0 eingetragen.
- Unter **ASCII** wird hier die Anzahl der gelesenen Zeichen eingetragen. Dieser Wert kann von der Telegrammlänge abweichen.

ERROR

In diesem Wort erfolgt ein Eintrag im Fehlerfall. Folgende Fehlermeldungen können protokollabhängig generiert werden:

ASCII

Bit	Fehler	Beschreibung
0	overrun	Überlauf, ein Zeichen konnte nicht schnell genug aus der Schnittstelle gelesen werden kann
1	framing error	Fehler, der anzeigt, dass ein definierter Bitrahmen nicht übereinstimmt, die zulässige Länge überschreitet oder eine zusätzliche Bitfolge enthält (Stoppbitfehler)
2	parity	Paritätsfehler
3	overflow	Der Puffer ist voll.

STX/ETX

Bit	Fehler	Beschreibung
0	overflow	Das empfangene Telegramm übersteigt die Größe des Empfangspuffers.
1	char	Es wurde ein Zeichen außerhalb des Bereichs 20h ... 7Fh empfangen.
3	overflow	Der Puffer ist voll.

3964R / Modbus RTU/ASCII Master

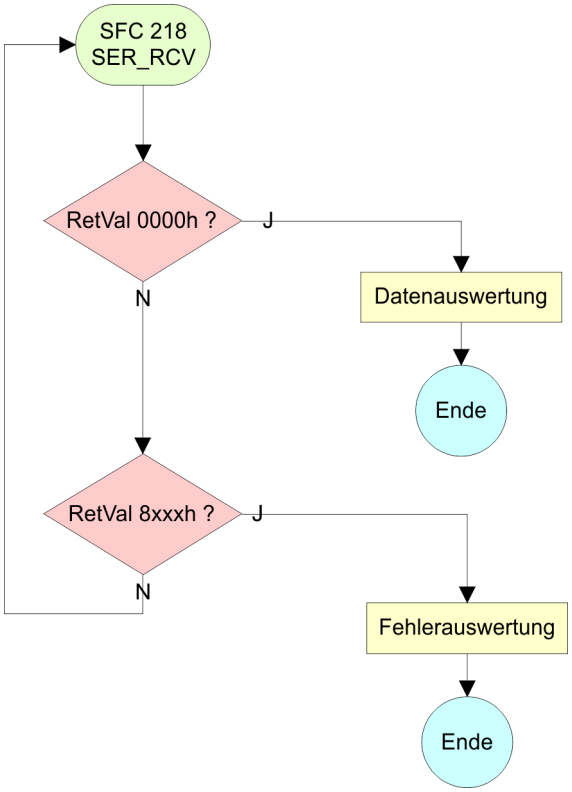
Bit	Fehler	Beschreibung
0	overflow	Das empfangene Telegramm übersteigt die Größe des Empfangspuffers.

RETVAL FC/SFC 218 (Rückgabewert)

Fehlercode	Beschreibung
0000h	kein Fehler
1000h	Empfangspuffer ist zu klein (Datenverlust)
8x24h	Fehler in FC/SFC-Parameter x, mit x: 1: Fehler in *DATAPTR* 2: Fehler in *DATALEN* 3: Fehler in *ERROR*
8122h	Fehler in Parameter *DATAPTR* (z.B. DB zu kurz)
809Ah	Schnittstelle nicht vorhanden bzw. Schnittstelle wird für PROFIBUS verwendet
809Bh	Schnittstelle ist nicht konfiguriert

Prinzip der Programmierung

Nachfolgend sehen Sie die Grundstruktur zur Programmierung eines Receive-Auftrags. Diese Struktur können Sie für alle Protokolle verwenden.



11.1.5 FB 1 - RECEIVE_ASCII - Empfangen mit definierter Länge von PtP

Beschreibung

Dieser FB sammelt die Daten die über die interne serielle Schnittstelle im PtP-Betrieb empfangen werden und kopiert diese in den mittels *EMPF_PUFFER* angegebenen Telegrammpuffer. Wurde das komplette Telegramm empfangen, wird *EMPF_FERTIG* gesetzt und der FB verlassen. Das Einlesen der Daten, kann mehrere FB-Aufrufe erfordern. Das nächste Telegramm wird erst eingelesen, wenn das Bit *EMPF_FERTIG* vom Anwender zurückgesetzt wurde. Mit diesem FB können nur Telegramme mit fester Länge empfangen werden.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
EMPF_PUFFER	IN	ANY	Zeiger auf DB, in den das empfangene Telegramm übertragen wird.
ER_BYTE	OUT	WORD	Fehlercode
EMPF_FERTIG	IN_OUT	BOOL	Status

EMPF_PUFFER

Geben Sie hier einen Bereich vom Typ Pointer an, in den die empfangenen Daten zu kopieren sind. Anzugeben sind Typ, Anfang und Länge.

Beispiel:

- Daten sind in DB5 ab 0.0 mit einer Länge von 124Byte abzulegen
  - DataPtr:=P#DB5.DBX0.0 BYTE 124

ER_BYTE In diesem Wort erfolgt ein Eintrag im Fehlerfall.

Fehlercode	Beschreibung
0003h	DB mit Telegrammpuffer nicht vorhanden.
0004h	DB mit Telegrammpuffer ist zu kurz.
7000h	Empfangspuffer ist zu klein - Daten wurden gelöscht!
8000h	Pointerangabe in *EMPF_PUFFER* fehlerhaft oder nicht vorhanden.
9001h	DB-Angabe in *EMPF_PUFFER* fehlerhaft oder nicht vorhanden.
9002h	Längen-Angabe in *EMPF_PUFFER* fehlerhaft oder nicht vorhanden.

11.1.6 FB 7 - P_RCV_RK - Empfangen von CP 341

Beschreibung

Der FB 7 P_RCV_RK überträgt Daten vom CP in einen Datenbereich der CPU, spezifiziert durch die Parameter *DB_NO, DBB_NO* und *LEN*. Der FB wird zur Datenübertragung im Zyklus oder in einem zeitgesteuerten Programm aufgerufen. Bitte beachten Sie, dass dieser Baustein intern den FC bzw. SFC 192 CP_S_R aufruft. Dieser darf nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB!

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
EN_R	IN	BOOL	Freigabe für Daten lesen
R	IN	BOOL	Auftragsabbruch - der laufende Auftrag wird abgebrochen und Empfang wird gesperrt.
LADDR	IN	INT	Logische Basisadresse des CP - entspricht der Adresse aus der Hardware-Konfiguration des CP.
DB_NO	IN	INT	Datenbausteinnummer - Nummer des Empfangs-DB, Null ist nicht erlaubt.
DBB_NO	IN	INT	Datenbytenummer - Empfangsdaten ab Datenbyte 0 ≤ *DBB_NO* ≤ 8190
L_...	OUT	-	Diese Parameter haben unter ASCII und 3964(R) keine Bedeutung, können aber von ladbaren Protokollen belegt sein.
NDR*	OUT	BOOL	Auftrag fertig ohne Fehler, Daten übernommen Parameter *STATUS* = 00h
ERROR*	OUT	BOOL	Auftrag fertig mit Fehler Parameter *STATUS* enthält die Fehlerinformation
LEN*	OUT	BOOL	Länge des empfangenen Telegramms in Byte 1 ≤ *LEN* ≤ 1024
STATUS*	OUT	WORD	Spezifikation des Fehlers bei *ERROR* = 1

*) Parameter steht bis zum nächsten Aufruf des FBs zur Verfügung.

Auftrag freigeben und abbrechen

- Mit Signalzustand "1" an *EN_R* wird die Überprüfung, ob Daten vom CP zu lesen sind, freigegeben. Je nach Datenmenge kann eine Datenübertragung über mehrere Programmzyklen laufen.
- Sie können jederzeit mit dem Signalzustand "0" an *EN_R* eine laufende Übertragung abbrechen. Hierbei wird der abgebrochene Empfangsauftrag mit einer Fehlermeldung (STATUS) beendet.
- Solange "0" an *EN_R* ansteht, ist der Empfang ausgeschaltet. Mit Signalzustand "1" an R können Sie einen laufenden Auftrag abbrechen und den FB in den Grundzustand zurückversetzen. Steht statisch der Signalzustand "1" am Eingang R an, so ist der Empfang ausgeschaltet.

Mechanismus zur Anlaufsynchronisation

Der FB 7 besitzen einen Mechanismus für die Anlaufsynchronisation zwischen CPU und CP, der automatisch beim erstmaligen Aufruf des FB ausgeführt wird. Bevor ein angestoßener Auftrag nach einem STOP-RUN-Übergang vom CP bearbeitet werden kann, muss die Anlauf-Synchronisation zwischen CP und CPU abgeschlossen sein. Ein in der Zwischenzeit angestoßener Auftrag wird nach der Synchronisation zum CP übertragen.

Zum Erkennen eines Signalwechsels ist eine Mindestimpulsdauer erforderlich. Ausschlaggebend sind die CPU-Zykluszeit, die Aktualisierungszeit auf dem CP und die Reaktionszeit des Kommunikationspartners.

Fehleranzeige

- Der Ausgang *NDR* zeigt "Auftrag fertig ohne Fehler/Daten übernommen" an. Bei *ERROR* wird die entsprechende Ereignisnummer in *STATUS* angezeigt. Ist kein Fehler aufgetreten, hat STATUS den Wert "0".
- *NDR* und *ERROR/STATUS* werden auch bei *RESET* des FB ausgegeben. Bei einem aufgetretenen Fehler wird das Binärergebnis BEI zurückgesetzt. Wird der Baustein ohne Fehler beendet, hat BEI den Zustand "1".
- Bitte beachten Sie, dass die Parameter *NDR, ERROR* und *STATUS* immer nur für einen Bausteinaufruf verfügbar sind. Zur weiteren Auswertung sollten Sie diese in einen freien Datenbereich kopieren.

Adressierung

Mit *LADDR* geben Sie die Adresse des anzusprechenden CP an. Dies ist die Adresse, die Sie über die Hardware-Konfiguration für den CP vergeben haben. Bitte beachten Sie beim CP, dass die Basis-Adressen für Ein- und Ausgabe identisch sind.

Datenbereich

Der FB 7 - P_RCV_RK arbeitet mit einem Instanz-DB I_RCV_RK zusammen. Dieser hat eine Länge von 60Byte. Die DB-Nr. geben Sie mit dem Aufruf mit. Ein Zugriff auf die Daten im Instanz-DB ist nicht zulässig.

11.1.7 FB 8 - P_SND_RK - Senden an CP 341

Beschreibung

Der FB 8 - P_SND_RK überträgt einen Datenblock von einem Datenbaustein zum CP, spezifiziert durch die Parameter *DB_NO*, *DBB_NO* und *LEN*. Hierbei ist der FB zur Datenübertragung im Zyklus oder statisch in einem zeitgesteuerten Programm aufzurufen. Bitte beachten Sie, dass dieser Baustein intern den FC bzw. SFC 192 CP_S_R aufruft. Dieser darf nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB!

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
SF	IN	CHAR	S = Send, F = Fetch. Bei ASCII und 3964(R) kann hier der Default-wert "S" für Senden übernommen werden.
REQ	IN	BOOL	Auftragsanstoß bei positiver Flanke
R	IN	BOOL	Auftragsabbruch - der laufende Auftrag wird abgebrochen und Senden wird gesperrt.
LADDR	IN	INT	Logische Basisadresse des CP - entspricht der Adresse aus der Hardware-Konfiguration des CP.
DB_NO	IN	INT	Datenbausteinnummer - Nummer des Sende-DB, Null ist nicht erlaubt.
DBB_NO	IN	INT	Datenbytenummer - Sendedaten ab Datenbyte 0 ≤ *DBB_NO* ≤ 8190
LEN	IN	INT	Länge des zu sendenden Telegramms in Byte. 1 ≤ *LEN* ≤ 1024
R_...	IN	-	Diese Parameter haben unter ASCII und 3964(R) keine Bedeutung, können aber von ladbaren Protokollen belegt sein. Bei Einsatz unter Modbus ist hier "X" einzutragen.
DONE*	OUT	BOOL	Auftrag fertig ohne Fehler, Daten gesendet Parameter *STATUS* = 00h
ERROR*	OUT	BOOL	Auftrag fertig mit Fehler Parameter *STATUS* enthält die Fehlerinfor-mation
STATUS*	OUT	WORD	Spezifikation des Fehlers bei *ERROR* = 1
*) Parameter steht bis zum nächsten Aufruf des FBs zur Verfügung.			

Auftrag freigeben und abbrechen

- Mit einer positiven Flanke am Eingang *REQ* des FB 8 - P_SND_RK wird die Übertragung der Daten angestoßen. Je nach Datenmenge kann eine Datenübertragung über mehrere Programmzyklen laufen.
- Sie können jederzeit mit dem Signalzustand "1" an *R* einen laufenden Auftrag abbrechen und den FB in den Grundzustand zurückversetzen. Bitte beachten Sie, dass hierbei die Daten, die der CP bereits erhalten hat, noch an den Kommunikationspartner gesendet werden.
- Steht statisch der Signalzustand "1" am Eingang *R* an, so ist das Senden ausgeschaltet.

Mechanismus zur Anlaufsynchronisation

Der FB 8 besitzen einen Mechanismus für die Anlaufsynchronisation zwischen CPU und CP, der automatisch beim erstmaligen Aufruf des FB ausgeführt wird. Bevor ein angestoßener Auftrag nach einem STOP-RUN-Übergang der CPU vom CP bearbeitet werden kann, muss die Anlauf-Synchronisation zwischen CP und CPU abgeschlossen sein. Ein in der Zwischenzeit angestoßener Auftrag wird nach der Synchronisation zum CP übertragen.

Zum Erkennen eines Signalwechsels ist eine Mindestimpulsdauer erforderlich. Ausschlaggebend sind die CPU-Zykluszeit, die Aktualisierungszeit auf dem CP und die Reaktionszeit des Kommunikationspartners.

Fehleranzeige

- Der Ausgang *DONE* zeigt "Auftragsende ohne Fehler" an. Bei *ERROR* wird die entsprechende Ereignisnummer in *STATUS* angezeigt. Ist kein Fehler aufgetreten, hat *STATUS* den Wert "0".
- *DONE* und *ERROR/STATUS* werden auch bei *RESET* des FB ausgegeben. Bei einem aufgetretenen Fehler wird das Binärergebnis BIE zurückgesetzt. Wird der Baustein ohne Fehler beendet, hat BIE den Zustand "1".
- Bitte beachten Sie, dass die Parameter *DONE, ERROR* und *STATUS* immer nur für einen Bausteinaufruf verfügbar sind. Zur weiteren Auswertung sollten Sie diese in einen freien Datenbereich kopieren.

Adressierung

Mit *LADDR* geben Sie die Adresse des anzusprechenden CP an. Dies ist die Adresse, die Sie über die Hardware-Konfiguration für den CP vergeben haben. Bitte beachten Sie beim CP, dass die Basis-Adressen für Ein- und Ausgabe identisch sind.

Datenbereich

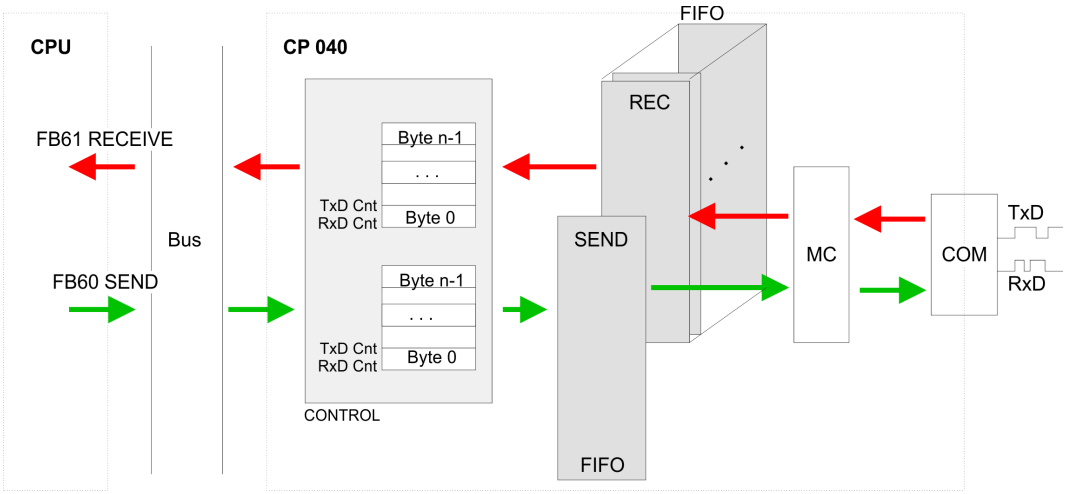
Der FB 8 - P_SND_RK arbeitet mit einem Instanz-DB I_SND_RK zusammen. Dieser hat eine Länge von 62Byte. Die DB-Nr. geben Sie mit dem Aufruf mit. Ein Zugriff auf die Daten im Instanz-DB ist nicht zulässig.

11.2 CP040

11.2.1 Übersicht

Kommunikationsprinzip

- Durch zyklischen Aufruf von FB 60 SEND und FB 61 RECEIVE bzw. FB 65 CP040_COM können Sie mit dem CP zyklisch Daten senden und empfangen.
- Auf dem CP erfolgt die Umsetzung der Übertragungsprotokolle zum Kommunikationspartner, welche Sie mittels der Hardwarekonfiguration parametrieren können.
- Ein zu sendendes Telegramm wird in der CPU, abhängig von der IO-Size, in Blöcke unterteilt und über den Datenkanal an den CP übergeben. Im CP werden diese Blöcke im Sendepuffer zusammengesetzt und bei Vollständigkeit des Telegramms über die serielle Schnittstelle gesendet.
- Der Austausch von empfangenen Telegrammen über den Rückwandbus erfolgt asynchron.
- Ist ein komplettes Telegramm über die serielle Schnittstelle eingetroffen, so wird dies in einem 1024Byte großen Ringpuffer abgelegt. Aus der Länge des noch freien Ringpuffers ergibt sich die max. Länge eines Telegramms.
- Je nach Parametrierung können bis zu 250 Telegramme gepuffert werden, wobei deren Gesamtlänge 1024 nicht überschreiten darf.
- Ist der Puffer voll, werden neu ankommende Telegramme verworfen.
- Ein komplettes Telegramm wird in Blöcke, abhängig von der parametrierten IO-Size unterteilt und an den Rückwandbus übergeben.
- Das Zusammensetzen der Datenblöcke hat in der CPU zu erfolgen.
- Da der Datenaustausch über den Rückwandbus asynchron abläuft, wird ein Software-Handshake zwischen dem CP und der CPU eingesetzt. Hierzu besitzen beide Hantierungsbausteine den gemeinsamen Parameter CONTROL. Für diesen Parameter ist das selbe Merker-Byte zu verwenden.



FIFO Ringpuffer max. 250 Telegramme 1024Byte
CONTROL Software-Handshake über CONTROL-Baustein

Zum Erkennen eines Signalwechsels ist eine Mindestimpulsdauer erforderlich. Ausschlaggebend sind die CPU-Zykluszeit, die Aktualisierungszeit auf dem CP und die Reaktionszeit des Kommunikationspartners.

11.2.2 FB 60 - SEND - Senden an System SLIO CP 040

Beschreibung

Dieser FB dient zur Datenausgabe von der CPU an den System SLIO CP 040. Hierbei legen Sie über die Bezeichner *DB_NO*, *DBB_NO* und *LEN* den Sendebereich fest. Über eine positive Flanke an *REQ* wird das Senden angestoßen und die Daten werden gesendet.

Parameter

Name	Deklaration	Typ	Beschreibung
REQ	IN	BOOL	Sendefreigabe bei positiver Flanke.
R	IN	BOOL	Synchron Reset auslösen.
LADDR	IN	INT	Logische Basisadresse des CP.
DB_NO	IN	INT	Datenbausteinnummer der Sendedaten.
DBB_NO	IN	INT	Datenbytenummer - Sendedaten ab Datenbyte.
LEN	IN	INT	Länge des zu sendenden Telegramms in Byte.
IO_SIZE	IN	WORD	Parametrierte IO Größe des Moduls.
DONE*	OUT	BOOL	Sende-Auftrag fertig ohne Fehler.
ERROR*	OUT	BOOL	Sende-Auftrag fertig mit Fehler. Parameter *STATUS* enthält die Fehlerinformation.
STATUS*	OUT	WORD	Spezifikation des Fehlers bei *ERROR* = 1.
CONTROL	IN_OUT	BYTE	Geteiltes Byte mit RECEIVE Hantierungsbaustein: SEND (Bit 0 … 3), RECEIVE (Bit 4 … 7).

*) Parameter steht bis zum nächsten Aufruf des FBs zur Verfügung.

REQ

Request - Sendefreigabe:

- Mit einer positiven Flanke am Eingang *REQ* wird die Übertragung der Daten angestoßen.
- Je nach Datenmenge kann eine Datenübertragung über mehrere Programmzyklen laufen.

R

Synchron Reset:

- Für die Initialisierung ist SEND im Anlauf-OB mit allen Parametern und mit gesetztem *R* einmalig aufzurufen.
- Sie können jederzeit mit dem Signalzustand "1" an *R* einen laufenden Auftrag abbrechen und den FB in den Grundzustand zurückversetzen. Bitte beachten Sie, dass hierbei die Daten, die der CP bereits erhalten hat, noch an den Kommunikationspartner gesendet werden.
- Steht statisch der Signalzustand "1" am Eingang *R* an, so ist das Senden ausgeschaltet.

LADDR

Peripherieadresse:

- Mit *LADDR* geben Sie die Adresse des anzusprechenden CP an. Dies ist die Adresse, die Sie über die Hardware- Konfiguration für den CP vergeben haben.

DB_NO

Datenbaustein-Nummer:

- Nummer des Datenbausteins, der die zu sendenden Daten beinhaltet.
- Null ist nicht erlaubt.

DBB_NO

Datenbyte-Nummer:

- Nummer des Datenbytes im Datenbaustein, ab dem die Sendedaten abgelegt sind.

LEN

Länge:

- Länge der Nutzdaten, welche zu übertragen sind.
- Es gilt: 1 ≤ *LEN* ≤ 1024.

IO_SIZE

Größe E/A-Bereich:

- Geben Sie hier die Größe des E/A-Bereichs an. Abhängig vom übergeordneten System belegt der CP für Ein- und Ausgabe jeweils folgende Anzahl an Bytes im Adress-Bereich:
  - PROFIBUS: 8Byte, 20Byte oder 60Byte wählbar
  - PROFINET: 20Byte oder 60Byte wählbar
  - CANopen: 8Byte
  - EtherCAT: 60Byte
  - DeviceNET: 60Byte
  - ModbusTCP: 60Byte

DONE

DONE:

- wird gesetzt bei Auftrag fertig ohne Fehler und *STATUS* = 0000h.

ERROR

ERROR:

- wird gesetzt bei Auftrag fertig mit Fehler. Hierbei enthält *STATUS* die entsprechende Fehlerinformation.

STATUS

Bei fehlerfreier Funktion *STATUS* = 0000h oder 8181h. Im Fehlerfall finden Sie hier den entsprechenden Fehlercode. Solange *ERROR* gesetzt ist bleibt der Wert in *STATUS* bestehen. Folgende Statusmeldungen sind möglich:

STATUS	Beschreibung
0000h	Kein Fehler vorhanden.
0202h	Mögliche Fehlerquellen: - Hantierungsbaustein und CP sind nicht synchron (Abhilfe: Synchron Reset auslösen) - *IO_SIZE* ist ungültig (*IO_SIZE* = 0 oder *IO_SIZE* > 60).
0301h	DB ist ungültig.
0517h	*LEN* ist ungültig (*LEN* = 0 oder *LEN* > 1024).
070Ah	Übertragung fehlgeschlagen, Partner antwortet nicht, oder hat den Auftrag negativ quittiert.
8181h	Auftrag läuft (Status und keine Fehlermeldung).

CONTROL

Die Hantierungsbausteine SEND und RECEIVE verwenden für den Handshake den gemeinsamen Parameter *CONTROL*. Weisen Sie diesem Parameter ein gemeinsames Merker-Byte zu.

Fehleranzeige

- Der Ausgang *DONE* zeigt "Auftragsende ohne Fehler" an. Bei *ERROR* wird die entsprechende Ereignisnummer in *STATUS* angezeigt. Ist kein Fehler aufgetreten, hat *STATUS* den Wert "0".
- *DONE*, *ERROR* und *STATUS* werden auch bei RESET des FB ausgegeben. Bei einem aufgetretenen Fehler wird das Binärergebnis BIE zurückgesetzt. Wird der Baustein ohne Fehler beendet, hat BIE den Zustand "1".
- Bitte beachten Sie, dass die Parameter *DONE*, *ERROR* und *STATUS* immer nur für einen Bausteinaufruf verfügbar sind. Zur weiteren Auswertung sollten Sie diese in einen freien Datenbereich kopieren.

11.2.3 FB 61 - RECEIVE - Empfangen von System SLIO CP 040

Beschreibung

Dieser FB dient zum Datenempfang vom System SLIO CP 040. Hierbei legen Sie über die Bezeichner *DB_NO* und *DBB_NO* den Empfangsbereich fest. Die Länge des eingelesenen Telegramms wird in *LEN* abgelegt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
EN_R	IN	BOOL	Freigabe zum Daten lesen.
R	IN	BOOL	Synchron Reset auslösen.
LADDR	IN	INT	Logische Basisadresse des CP.

Parameter	Deklaration	Datentyp	Beschreibung
DB_NO	IN	INT	Datenbausteinnummer der Empfangsdaten.
DBB_NO	IN	INT	Datenbytenummer - Empfangsdaten ab Datenbyte.
IO_SIZE	IN	WORD	Parametrierte I/O Größe des Moduls.
LEN	OUT	INT	Länge des empfangenen Telegramms in Byte.
NDR*	OUT	BOOL	Empfangs-Auftrag fertig ohne Fehler.
ERROR*	OUT	BOOL	Empfangs-Auftrag fertig mit Fehler. Parameter *STATUS* enthält die Fehlerinformation.
STATUS*	OUT	WORD	Spezifikation des Fehlers bei *ERROR* = 1.
CONTROL	IN_OUT	BYTE	Geteiltes Byte mit SEND Hantierungsbaustein: SEND (Bit 0 … 3), RECEIVE (Bit 4 … 7).

*) Parameter steht bis zum nächsten Aufruf des FBs zur Verfügung.

EN_R

Enable Receive - Lesefreigabe:

- Mit Signalzustand "1" an *EN_R* wird die Überprüfung, ob Daten vom CP zu lesen sind, freigegeben. Je nach Datenmenge kann eine Datenübertragung über mehrere Programmzyklen laufen.
- Sie können jederzeit mit dem Signalzustand "0" an *EN_R* eine laufende Übertragung abbrechen. Hierbei wird der abgebrochene Empfangsauftrag mit einer Fehlermeldung (*STATUS*) beendet.
- Solange "0" an *EN_R* ansteht, ist der Empfang ausgeschaltet.

R

Synchron Reset:

- Für die Initialisierung ist RECEIVE im Anlauf-OB mit allen Parametern und mit gesetztem *R* einmalig aufzurufen.
- Sie können jederzeit mit dem Signalzustand "1" an *R* einen laufenden Auftrag abbrechen und den FB in den Grundzustand zurückversetzen.
- Steht statisch der Signalzustand "1" am Eingang *R* an, so ist der Empfang ausgeschaltet.

LADDR

Peripherieadresse:

- Mit *LADDR* geben Sie die Adresse des anzusprechenden CP an. Dies ist die Adresse, die Sie über die Hardware- Konfiguration für den CP vergeben haben.

DB_NO

Datenbaustein-Nummer:

- Nummer des Datenbausteins, der die gelesenen Daten beinhaltet.
- Null ist nicht erlaubt.

DBB_NO

Datenbyte-Nummer:

- Nummer des Datenbytes im Datenbaustein, ab dem die empfangenen Daten abgelegt werden sollen.

IO_SIZE

Größe E/A-Bereich:

- Geben Sie hier die Größe des E/A-Bereichs an. Abhängig vom übergeordneten System belegt der CP für Ein- und Ausgabe jeweils folgende Anzahl an Bytes im Adress-Bereich:
  - PROFIBUS: 8Byte, 20Byte oder 60Byte wählbar
  - PROFINET: 20Byte oder 60Byte wählbar
  - CANopen: 8Byte
  - EtherCAT: 60Byte
  - DeviceNET: 60Byte
  - ModbusTCP: 60Byte

LEN

Länge:

- Länge der Nutzdaten, welche zu übertragen sind.
- Es gilt: 1 ≤ *LEN* ≤ 1024.

NDR

New data ready:

- Neu empfangene Daten stehen für die CPU im CP bereit.

ERROR

ERROR:

- wird gesetzt bei Auftrag fertig mit Fehler. Hierbei enthält *STATUS* die entsprechende Fehlerinformation.

STATUS

Bei fehlerfreier Funktion *STATUS* = 0000h oder 8181h. Im Fehlerfall finden Sie hier den entsprechenden Fehlercode. Solange *ERROR* gesetzt ist bleibt der Wert in *STATUS* bestehen. Folgende Statusmeldungen sind möglich:

STATUS	Beschreibung
0000h	Kein Fehler vorhanden.
0202h	Mögliche Fehlerquellen: ■ Hantierungsbaustein und CP sind nicht synchron (Abhilfe: Synchron Reset auslösen). ■ *IO_SIZE* ist ungültig (*IO_SIZE* = 0 oder *IO_SIZE* > 60).
0301h	DB ist ungültig.
070Ah	Übertragung fehlgeschlagen, Partner antwortet nicht, oder hat den Auftrag negativ quittiert.
8181h	Auftrag läuft (Status und keine Fehlermeldung).

CONTROL

- Die Hantierungsbausteine SEND und RECEIVE verwenden für den Handshake den gemeinsamen Parameter CONTROL.
- Weisen Sie diesem Parameter ein gemeinsames Merker-Byte zu.

Fehleranzeige

- Der Ausgang *NDR* zeigt "Auftrag fertig ohne Fehler/Daten übernommen" an. Bei *ERROR* wird die entsprechende Ereignisnummer in *STATUS* angezeigt. Ist kein Fehler aufgetreten, hat *STATUS* den Wert "0".
- *NDR, ERROR* und *STATUS* werden auch bei RESET des FBs ausgegeben. Bei einem aufgetretenen Fehler wird das Binärergebnis BIE zurückgesetzt. Wird der Baustein ohne Fehler beendet, hat BIE den Zustand "1".
- Bitte beachten Sie, dass die Parameter *NDR, ERROR* und *STATUS* immer nur für einen Bausteinaufruf verfügbar sind. Zur weiteren Auswertung sollten Sie diese in einen freien Datenbereich kopieren.

11.2.4 FB 65 - CP040_COM - Kommunikation SLIO CP 040

Beschreibung

Dieser FB dient zur Datenübergabe von der System SLIO CPU an den CP 040. Hierbei legen Sie über die Bezeichner *DB_NO_SEND*, *DB_NO_RECV* den Sende-/Empfangsbereich fest. Über eine positive Flanke an *REQ_SEND* wird das Senden angestoßen und die Daten werden gesendet. Über *EN_RECV* wird die Freigabe zum Daten empfangen gegeben.

Parameter

Name	Deklaration	Typ	Beschreibung
REQ_SEND	IN	BOOL	Anstoß Sende-Auftrag bei positiver Flanke.
EN_RECV	IN	BOOL	Freigabe für Daten empfangen.
RESET	IN	BOOL	Synchron Reset auslösen.
ADDR_OUT	IN	INT	Ausgangsadresse des CP aus der HW-Konfiguration.
ADDR_IN	IN	INT	Eingangsadresse des CP aus der HW-Konfiguration.
IO_SIZE	IN	WORD	Parametrierte IO Größe des Moduls.
DB_NO_SEND	IN	INT	Datenbaustein-Nummer - Nummer des Sende-DB. Null ist nicht erlaubt.
DBB_NO_SEND	IN	INT	Datenbyte-Nummer - Sendedaten ab Datenbyte.
LEN_SEND	IN	INT	Länge des zu sendenden Telegramms in Byte. 1 ≤ *LEN_SEND* ≤ 1024
DB_NO_RECV	IN	INT	Datenbaustein-Nummer - Nummer des Empfang-DB. Null ist nicht erlaubt.
DBB_NO_RECV	IN	INT	Datenbyte-Nummer - Empfangsdaten ab Datenbyte.
DONE_SEND*	OUT	BOOL	Sende-Auftrag fertig ohne Fehler. Daten gesendet: Parameter *STATUS_SEND* = 0000h.
ERROR_SEND*	OUT	BOOL	Sende-Auftrag fertig mit Fehler. Parameter *STATUS_SEND* enthält die Fehlerinformation.
NDR_RCV*	OUT	BOOL	Empfangs-Auftrag fertig ohne Fehler. Daten gesendet: Parameter *STATUS_RCV* = 0000h. Bleibt für einen Zyklus stehen.
ERROR_RCV*	OUT	BOOL	Empfangs-Auftrag fertig mit Fehler. Parameter *STATUS_RCV* enthält die Fehlerinformation.
STATUS_SEND*	OUT	WORD	Spezifikation des Fehlers beim Senden bei *ERROR_SEND* = 1

Name	Deklaration	Typ	Beschreibung
LEN_RCV	OUT	INT	Länge des empfangenen Telegramms in Byte. 1 ≤ *LEN_RCV* ≤ 1024
STATUS_RCV*	OUT	WORD	Spezifikation des Fehlers beim Empfangen bei *ERROR_RCV* = 1

*) Parameter steht bis zum nächsten Aufruf des FBs zur Verfügung.

REQ_SEND

Request - Sendefreigabe:

- Mit einer positiven Flanke am Eingang *REQ_SEND* wird die Übertragung der Daten angestoßen.
- Je nach Datenmenge kann eine Datenübertragung über mehrere Programmzyklen laufen.

EN_RECV

Freigabe für Daten empfangen.

RESET

Synchron Reset:

- Für die Initialisierung ist der FB 65 im Anlauf-OB mit allen Parametern und mit gesetztem *RESET* einmalig aufzurufen.
- Sie können jederzeit mit dem Signalzustand "1" an *RESET* einen laufenden Auftrag abbrechen und den FB in den Grundzustand zurückversetzen. Bitte beachten Sie, dass hierbei die Daten, die der CP bereits erhalten hat, noch an den Kommunikationspartner gesendet werden.
- Steht statisch der Signalzustand "1" am Eingang *RESET* an, so ist das Senden ausgeschaltet.

ADDR_IN

Eingangs-Peripherieadresse:

- Mit *ADDR_IN* geben Sie die Eingangs-Adresse des anzusprechenden CP an. Dies ist die Adresse, die Sie über die Hardware-Konfiguration für den CP vergeben haben.

ADDR_OUT

Ausgangs-Peripherieadresse:

- Mit *ADDR_OUT* geben Sie die Ausgangs-Adresse des anzusprechenden CP an. Dies ist die Adresse, die Sie über die Hardware-Konfiguration für den CP vergeben haben.

DB_NO_SEND

Datenbaustein-Nummer SEND:

- Nummer des Datenbausteins, der die zu sendenden Daten beinhaltet.
- Null ist nicht erlaubt.

DBB_NO_SEND

Datenbyte-Nummer SEND:

- Nummer des Datenbytes im Datenbaustein, ab dem die Sendedaten abgelegt sind.

LEN_SEND

Länge SEND:

- Länge der Nutzdaten, welche zu übertragen sind.
- Es gilt: 1 ≤ *LEN_SEND* ≤ 1024.

DB_NO_RECV

Datenbaustein-Nummer RECV:

- Nummer des Empfang-DBs.
- Null ist nicht erlaubt.

DBB_NO_RECV

Datenbyte-Nummer RECV:

- Nummer des Datenbytes im Datenbaustein, ab dem die Empfangsdaten abgelegt sind.

IO_SIZE

Größe E/A-Bereich:

- Geben Sie hier die Größe des E/A-Bereichs an. Abhängig vom übergeordneten System belegt der CP für Ein- und Ausgabe jeweils folgende Anzahl an Bytes im Adress-Bereich:
  - SLIO CPU: 8Byte, 20Byte oder 60Byte wählbar
  - PROFIBUS: 8Byte, 20Byte oder 60Byte wählbar
  - PROFINET: 20Byte oder 60Byte wählbar
  - CANopen: 8Byte
  - EtherCAT: 60Byte
  - DeviceNET: 60Byte
  - ModbusTCP: 60Byte

DONE_SEND

Sende-Auftrag fertig ohne Fehler, Daten gesendet. Parameter *STATUS_SEND* = 0000h.

ERROR_SEND

ERROR_SEND wird gesetzt bei Auftrag fertig mit Fehler. Hierbei enthält *STATUS_SEND* die entsprechende Fehlerinformation.

STATUS_SEND

Bei fehlerfreier Funktion ist *STATUS_SEND* 0000h oder 8181h. Im Fehlerfall finden Sie hier den entsprechenden Fehlercode. Solange *ERROR_SEND* gesetzt ist bleibt der Wert in *STATUS_SEND* bestehen. Folgende Statusmeldungen sind möglich:

STATUS	Beschreibung
0000h	Kein Fehler vorhanden.
0202h	*IO_SIZE* ist ungültig (*IO_SIZE* = 0 oder *IO_SIZE* > 60).
0301h	DB ist ungültig.
070Ah	Übertragung fehlgeschlagen, Partner antwortet nicht, oder hat den Auftrag negativ quittiert
0517h	*LEN* ist ungültig (*LEN* = 0 oder *LEN* > 1024).
8181h	Auftrag läuft (Status und keine Fehlermeldung).

LEN_RCV

Länge Receive:

- Länge des empfangenen Telegramms in Byte.
- 1 ≤ *LEN_RCV* ≤ 1024

NDR_RCV

New data ready:

- Neu empfangene Daten stehen im Empfangs-DB zur Verfügung. *NDR_RCV* bleibt für einen Zyklus stehen.
- Daten ohne Fehler empfangen: Parameter *STATUS_RCV* = 0000h.

ERROR_RCV

ERROR_RCV wird gesetzt bei Auftrag fertig mit Fehler. Hierbei enthält *STATUS_REC* die entsprechende Fehlerinformation.

STATUS_RCV

Bei fehlerfreier Funktion ist *STATUS_RCV* 0000h oder 8181h. Im Fehlerfall finden Sie hier den entsprechenden Fehlercode. Solange *ERROR_RCV* gesetzt ist bleibt der Wert in *STATUS_RCV* bestehen. Folgende Statusmeldungen sind möglich:

STATUS	Beschreibung
0000h	Kein Fehler vorhanden.
0202h	*IO_SIZE* ist ungültig (*IO_SIZE* = 0 oder *IO_SIZE* > 60).
0301h	DB ist ungültig.
070Ah	Übertragung fehlgeschlagen, Partner antwortet nicht, oder hat den Auftrag negativ quittiert.
080Ah	Ein freier Empfangspuffer ist nicht vorhanden.
080Ch	Fehlerhaftes Zeichen empfangen (Zeichenrahmen- oder Paritätsfehler)
8181h	Auftrag läuft (Status und keine Fehlermeldung).

Fehleranzeige

- Der Ausgang *DONE_SEND* zeigt "Auftragsende ohne Fehler" an.
- Der Ausgang *NDR_RCV* zeigt "Auftragsempfang ohne Fehler" an.
- Bei *ERROR_SEND* oder *ERROR_RCV* wird die entsprechende Ereignisnummer in *STATUS_SEND* oder *STATUS_RCV* angezeigt. Ist kein Fehler aufgetreten, hat *STATUS_SEND* und *STATUS_RCV* den Wert 0000h.
- *DONE_SEND, NDR_RCV, ERROR_SEND, ERROR_RCV* und *STATUS_SEND, STATUS_RCV* werden auch bei RESET des FBs ausgegeben. Bei einem aufgetretenen Fehler wird das Binärergebnis BIE zurückgesetzt. Wird der Baustein ohne Fehler beendet, hat BIE den Zustand "1".
- Bitte beachten Sie, dass die Parameter *DONE_SEND, NDR_RCV, ERROR_SEND, ERROR_RCV* und *STATUS_SEND, STATUS_RCV* immer nur für einen Bausteinaufruf verfügbar sind. Zur weiteren Auswertung sollten Sie diese in einen freien Datenbereich kopieren.

11.3 CP240

11.3.1 FC 0 - SEND_ASCII_STX_3964 - Senden an CP 240

Beschreibung

Dieser FC dient zur Datenausgabe von der CPU an den CP 240. Hierbei legen Sie über die Bezeichner *_DB, ABD* und *ANZ* den Sendebereich fest. Über das Bit *FRG* wird der Sendeanstoß gesetzt und die Daten werden gesendet. Nach dem Übertragen der Daten setzt der Hantierungsbaustein das Bit *FRG* wieder zurück.

Parameter

Name	Deklaration	Typ	Beschreibung
ADR	IN	INT	Peripherieadresse
_DB	IN	BLOCK_DB	DB-Nummer mit den Sendedaten.
ABD	IN	WORD	Nummer des 1. Datenworts.
ANZ	IN	WORD	Anzahl der Bytes.
PAFE	OUT	BYTE	Parametrierfehler Code (0 = OK).
FRG	IN_OUT	BOOL	Auftrag anstoßen.
GESE	IN_OUT	WORD	Wird intern verwendet.
ANZ_INT	IN_OUT	WORD	Wird intern verwendet.
ENDE_KOM	IN_OUT	BOOL	Wird intern verwendet.
LETZTER_BLOCK	IN_OUT	BOOL	Wird intern verwendet.
SENDEN_LAEUFT	IN_OUT	BOOL	Status der Funktion.
FEHLER_KOM	IN_OUT	BOOL	Wird intern verwendet.

ADR

Peripherieadresse unter welcher der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

_DB

Nummer des Datenbausteins, der die zu sendenden Daten für den CP beinhaltet.

ABD

Wortvariable, welche die Nummer des Datenworts enthält, ab dem die auszugebenden Zeichen abgelegt sind.

ANZ

Anzahl der Bytes, die zu übertragen sind.

PAFE

Alle Bits dieses Merker-Bytes sind bei richtiger Funktion "0". Bei Fehlfunktion wird ein Fehlercode eingetragen. Die Fehlerangabe ist selbstquittierend, d.h. nach Beseitigung der Fehlerursache wird das Byte wieder auf "0" gesetzt. Folgende Fehler sind möglich:

- 1 = Datenbaustein nicht vorhanden
- 2 = Datenbaustein zu kurz
- 3 = Datenbausteinnummer nicht im gültigen Bereich

FRG Sendefreigabe

Bei *FRG* = "1" werden die über *_DB, ADB* und *ANZ* definieren Daten einmalig an den über *ADR* adressierten CP übertragen. Nach der Übertragung wird *FRG* wieder zurückgesetzt. Ist beim Aufruf *FRG* = "0", wird der Baustein sofort wieder verlassen!

GESE, ANZ_INT, ENDE_KOM, LETZTER_BLOCK, SENDEN_LAEUFT, FEHLER_KOM

Diese Parameter werden intern verwendet. Sie dienen dem Informationsaustausch zwischen den Hantierungsbausteinen. Für den Einsatz des SYNCHRON_RESET (FC9) sind die Steuerbits FRG, ENDE_KOM, LETZTER _BLOCK, SENDEN_LAEUFT und FEHLER_KOM immer in einem Merker-Byte abzulegen.

11.3.2 FC 1 - RECEIVE_ASCII_STX_3964 - Empfangen von CP 240

Beschreibung

Dieser FC dient zum Datenempfang vom CP 240. Hierbei legen Sie über die Bezeichner *_DB* und *ABD* den Empfangsbereich fest. Ist der Ausgang *EMFR* gesetzt, so ist ein neues Telegramm komplett eingelesen worden. Die Länge des eingelesenen Telegramms wird in *ANZ* abgelegt. Nach der Auswertung des Telegramms ist dieses Bit vom Anwender zurückzusetzen, da ansonsten kein weiteres Telegramm in der CPU übernommen werden kann.

Parameter

Name	Deklaration	Typ	Beschreibung
ADR	IN	INT	Peripherieadresse
_DB	IN	BLOCK_DB	DB-Nummer mit den Empfangsdaten.
ABD	IN	WORD	Nummer des 1. Datenworts.
ANZ	OUT	WORD	Anzahl der empfangenen Bytes.
PAFE	OUT	BYTE	Parametrierfehler Code (0 = OK).
EMFR	IN_OUT	BOOL	Empfangsbestätigung
GEEM	IN_OUT	WORD	Wird intern verwendet.
ANZ_INT	IN_OUT	WORD	Wird intern verwendet.
EMPF_LAEUFT	IN_OUT	BOOL	Status der Funktion.
LETZTER_BLOCK	IN_OUT	BOOL	Wird intern verwendet.
FEHLER_EMPF	IN_OUT	BOOL	Wird intern verwendet.
OFFSET	IN_OUT	WORD	Wird intern verwendet.

ADR

Peripherieadresse unter welcher der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

_DB

Nummer des Datenbausteins, der die zu empfangenden Daten beinhaltet.

ABD

Wortvariable, welche die Nummer des Datenworts enthält, ab dem die auszugebenden Zeichen abgelegt sind.

ANZ

Wort-Variable, welche die Anzahl der Bytes enthält, die empfangen wurden.

PAFE

Alle Bits dieses Merker-Bytes sind bei richtiger Funktion "0". Bei Fehlfunktion wird ein Fehlercode eingetragen. Die Fehlerangabe ist selbstquittierend, d.h. nach Beseitigung der Fehlerursache wird das Byte wieder auf "0" gesetzt. Folgende Fehler sind möglich:

- 1 = Datenbaustein nicht vorhanden
- 2 = Datenbaustein zu kurz
- 3 = Datenbausteinnummer nicht im gültigen Bereich

EMFR

Durch Setzen des *EMFR* zeigt der Hantierungsbaustein an, dass Daten empfangen wurden. Erst durch Rücksetzen von *EMFR* im Anwenderprogramm können weitere Daten empfangen werden.

GEEM, ANZ_INT, LETZTER_BLOCK, EMPF_LAEUFT, FEHLER_EMPF, OFFSET

Diese Parameter werden intern verwendet. Sie dienen dem Informationsaustausch zwischen den Hantierungsbausteinen. Für den Einsatz des SYNCHRON_RESET (FC9) sind die Steuerbits EMFR, LETZTER_BLOCK, EMPF_LAEUFT und FEHLER_EMPF immer in einem Merker-Byte abzulegen.

11.3.3 FC 8 - STEUERBIT - Modemfunktionalität CP 240

Beschreibung

Mit diesem Baustein haben Sie folgenden Zugriff auf die seriellen Modemleitungen:

Lesen:	DTR, RTS, DSR, RI, CTS, CD
Schreiben:	DTR, RTS

Parameter

Name	Deklaration	Typ	Beschreibung
ADR	IN	INT	Peripherieadresse
RTS	IN	BOOL	Neuer Zustand RTS
DTR	IN	BOOL	Neuer Zustand DTR
MASKE_RTS	IN	BOOL	■ 0: nichts ändern ■ 1: Zustand an RTS übergeben
MASKE_DTR	IN	BOOL	■ 0: nichts ändern ■ 1: Zustand an DTR übergeben
RET_VAL	OUT	WORD	Rückgabewert (0 = OK)
STATUS	OUT	BYTE	Status
DELTA_STATUS	OUT	BYTE	Statusänderungen seit dem letzten Zugriff
START	IN_OUT	BOOL	Auftrag anstoßen
AUFTRAG_LAEUFT	IN_OUT	BOOL	Status der Funktion

Dieser Baustein darf nicht aufgerufen werden, solange ein Sendeauftrag läuft, ansonsten kann dies zu Datenverlust führen.

ADR

Peripherieadresse unter welcher der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

RTS, DTR

Mit diesem Parameter geben Sie den Status für *RTS* bzw. *DTR* vor, den Sie über *MASK_RTS* bzw. *MASK_DTR* aktivieren können.

MASK_RTS, MASK_DTR

Hier wird mit 1 der Status des entsprechenden Parameters übernommen, sobald Sie *START* auf 1 setzen.

RET_VAL

Dieser Parameter liefert zur Zeit immer 00h zurück und dient zukünftigen Fehlermeldungen.

STATUS, DELTA_STATUS

STATUS liefert den aktuellen Status der Modem-Leitungen zurück. *DELTA_STATUS* liefert den Status der Modem-Leitungen zurück, die sich seit dem letzten Zugriff geändert haben. Die Bytes haben folgenden Aufbau:

Bit-Nr.	7	6	5	4	3	2	1	0
STATUS	x	x	RTS	DTR	CD	RI	DSR	CTS
DELTA_STATUS	x	x	x	x	CD	RI	DSR	CTS

START

Durch Setzen von *START* wird der über die Maske aktivierte Status übernommen.

AUFTRAG_LAEUFT

Solange die Funktion abgearbeitet wird, bleibt dieses Bit gesetzt.

11.3.4 FC 9 - SYNCHRON_RESET - Synchronisation CPU und CP 240

Beschreibung

Der Baustein ist im zyklischen Programmteil aufzurufen. Mit dieser Funktion wird die Anlaufkennung des CP 240 quittiert, und so die Synchronisation zwischen CPU und CP hergestellt. Weiterhin kann bei einer Kommunikationsunterbrechung der CP rückgesetzt werden und so ein synchroner Anlauf erfolgen.

Eine Kommunikation mit SEND- und RECEIVE-Bausteinen ist nur möglich, wenn zuvor im Anlauf-OB der Parameter ANL des SYNCHRON-Bausteins gesetzt wurde.

Parameter

Name	Deklaration	Typ	Beschreibung
ADR	IN	INT	Peripherieadresse
TIMER_NR	IN	TIMER	Timer
ANL	IN_OUT	BOOL	CPU-Neustart erfolgt
ZERO	IN_OUT	BOOL	wird intern verwendet
RESET	IN_OUT	BOOL	Reset an CP
TIME_AN	IN_OUT	BOOL	wird intern verwendet
STEUERB_S	IN_OUT	BYTE	wird intern verwendet
STEUERB_R	IN_OUT	BYTE	wird intern verwendet

ADR

Peripherieadresse unter der der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

TIMER_NR

Timer für die Wartezeit.

ANL

Mit *ANL* = 1 wird dem Hantierungsbaustein mitgeteilt, dass an der CPU STOP/START bzw. NETZ-AUS/NETZ-EIN erfolgt ist und nun eine Synchronisation erfolgen muss. Nach der Synchronisation wird ANL automatisch zurückgesetzt.

ZERO Parameter wird intern verwendet.

RESET Mit *RESET* = 1 können Sie den CP aus Ihrem Anwenderprogramm zurücksetzen.

TIME_AN Parameter wird intern verwendet.

STEUERB_S

Hier ist das Merkerbyte anzugeben, in dem die Steuerbits FRG, ENDE_KOM, LETZTER_BLOCK, SENDEN_LAEUFT und FEHLER_KOM für den SEND-FC abgelegt sind.

STEUERB_R Hier ist das Merkerbyte anzugeben, in dem die Steuerbits EMFR, LETZTER_BLOCK, EMPF_LAEUFT und FEHLER_EMPF für den RECEIVE-FC abgelegt sind.

11.3.5 FC 11 - ASCII_FRAGMENT - Fragment Datenempfang CP 240

Beschreibung

Dieser FC dient zum fragmentierten ASCII-Datenempfang. Hiermit haben Sie die Möglichkeit große Telegramme in 12Byte-Blöcken direkt nach dem Erhalt an die CPU weiterzureichen. Hierbei wartet der CP nicht, bis das komplette Telegramm empfangen wurde. Der Einsatz des FC 11 setzt voraus, dass Sie beim Empfänger "ASCII-fragmentiert" parametriert haben. Im FC 11 legen Sie über die Bezeichner *_DB* und *ABD* den Empfangsbereich fest. Ist der Ausgang *EMFR* gesetzt, so ist ein neues Telegramm komplett eingelesen worden. Die Länge des eingelesenen Telegramms wird in *ANZ* abgelegt. Nach der Auswertung des Telegramms ist dieses Bit vom Anwender zurückzusetzen, da ansonsten kein weiteres Telegramm in der CPU übernommen werden kann.

Parameter

Name	Deklaration	Typ	Beschreibung
ADR	IN	INT	Peripherieadresse
_DB	IN	BLOCK_DB	DB-Nummer mit den Empfangsdaten
ABD	IN	WORD	Nummer des 1. Datenworts
ANZ	OUT	WORD	Anzahl der empfangenen Bytes
EMFR	IN_OUT	BOOL	Empfangsbestätigung
GEEM	IN_OUT	WORD	wird intern verwendet
ANZ_INT	IN_OUT	WORD	wird intern verwendet
EMPF_LAEUFT	IN_OUT	BOOL	wird intern verwendet
LETZTER_BLOCK	IN_OUT	BOOL	wird intern verwendet
FEHLER_EMPF	IN_OUT	BOOL	wird intern verwendet
PAFE	OUT	BYTE	Parametrierfehler (0 = OK)

ADR Peripherieadresse unter welcher der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

_DB	Nummer des Datenbausteins, der die empfangenen Daten beinhaltet.
ABD	Wortvariable, welche die Nummer des Datenworts enthält, ab dem die empfangenen Zeichen abgelegt sind.
ANZ	Wort-Variable, die die Anzahl der Bytes enthält, die empfangen wurden.
EMFR	Durch Setzen des *EMFR* zeigt der Hantierungsbaustein an, dass Daten empfangen wurden. Erst durch Rücksetzen von *EMFR* im Anwenderprogramm können weitere Daten empfangen werden.
PAFE	Alle Bits dieses Merker-Bytes sind bei richtiger Funktion "0". Bei Fehlfunktion wird ein Fehlercode eingetragen. Die Fehlerangabe ist selbstquittierend, d.h. nach Beseitigung der Fehlerursache wird das Byte wieder "0" gesetzt. Folgende Fehler sind möglich: ■ 1 = Datenbaustein nicht vorhanden ■ 2 = Datenbaustein zu kurz ■ 3 = Datenbausteinnummer nicht im gültigen Bereich
GEEM, ANZ_INT, LETZTER_BLOCK, EMPF_LAEUFT, FEHLER_EMPF	Diese Parameter werden intern verwendet. Sie dienen dem Informationsaustausch zwischen den Hantierungsbausteinen. Für den Einsatz des SYNCHRON_RESET sind die Steuerbits LETZTER_BLOCK, EMPF_ LAEUFT und FEHLER_EMPF immer in einem Merker-Byte abzulegen.

12 EtherCAT-Kommunikation - *"EtherCAT Communication"*

12.1 SDO-Kommunikation - *"SDO Communication"*

12.1.1 FB 52 - SDO_READ - Lesezugriff auf Objektverzeichnis

Beschreibung

Mit diesem Baustein können Sie auf das Objektverzeichnis von EtherCAT-Slave-Stationen und EtherCAT-Master lesend zugreifen. Hierbei handelt es sich um einen asynchron arbeitenden Baustein, d.h. die Bearbeitung erstreckt sich über mehrere Baustein-Aufrufe. Sie starten den SDO-Auftrag, indem Sie den FB 52 mit REQ = 1 aufrufen. Über den Ausgangsparameter BUSY und den Ausgangsparameter RETVAL wird der Zustand des Auftrags angezeigt. Die Datensatzübertragung ist abgeschlossen, wenn der Ausgangsparameter BUSY den Wert FALSE angenommen hat.

Die Fehlerbehandlung erfolgt über die Parameter ERROR, ERROR_ID und RETVAL

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
REQ	IN	BOOL	REQ = 1: Aktiviert den SDO-Zugriff bei steigender Flanke.
ID	IN	WORD	Logische Basisadresse der EtherCAT-Slave-Station bzw. des Masters in der Hardwarekonfiguration. Bei einer Ausgabebaugruppe muss Bit 15 gesetzt werden (Bsp. für Adresse 5: ID:=DW#16#8005). Bei einer Mischbaugruppe ist die kleinere der beiden Adressen anzugeben.
INDEX	IN	WORD	Index des Objekts für den SDO-Zugriff.
SUBINDEX	IN	BYTE	Subindex des Objekts für den SDO-Zugriff.
COMPL_ACCESS	IN	BOOL	Mit diesem Parameter wird bestimmt, ob nur ein einzelner Subindex oder das gesamte Objekt gelesen werden soll.
MLEN	IN	INT	Maximale Länge der zu lesenden Daten.
VALID	OUT	BOOL	Gibt an, ob ein neuer Datensatz empfangen wurde und gültig ist.
BUSY	OUT	BOOL	Dieser Parameter gibt den Bearbeitungsstatus des SDO-Zugriffs an. *BUSY* = 1: SDO-Zugriff ist noch in Bearbeitung.
ERROR	OUT	BOOL	*ERROR* = 1: Beim Lesevorgang trat ein Fehler auf.
RETVAL	OUT	INT	Rückgabewert (0 = OK)
ERROR_ID	OUT	DWORD	Busspezifischer Fehlercode. Ist während der Bearbeitung des SDO-Zugriffs ein Fehler aufgetreten, so ist in diesem Parameter der SDO-Abort-Fehlercode (EtherCAT-Fehlercode) angegeben.
LEN	OUT	INT	Länge der gelesenen Daten.
RECORD	INOUT	ANY	Bereich für die gelesenen Daten.

Bitte beachten Sie, dass die an RECORD übergebenen Daten nicht in einem temporären Bereich liegen.

Besonderheiten bei *COMPL_ACCESS* (CompleteAccess)

Bei Aktivierung des Parameters *COMPL_ACCESS* ist folgendes zu beachten:

- Bei *COMPL_ACCESS* = true darf der *SUBINDEX* nur 0 oder 1 betragen! Ansonsten bekommen Sie eine Fehlermeldung.
- Bei *COMPL_ACCESS* = true werden für *SUBINDEX* 0 2 Byte ausgelesen, da *SUBINDEX* 1 einen Offset von 2 Byte besitzt.

RETVAL (Rückgabewert)

Zusätzlich zu den hier aufgeführten modulspezifischen Fehlercodes sind auch noch die allgemeingültigen Fehlercodes für FC/SFCs als Rückgabewert möglich. ↳ *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

RETVAL	Beschreibung	Fehlercode in *ERROR_ID*
0x80A0	Negative Quittung beim Lesen von der Baugruppe.	ja
0x80A1	Negative Quittung beim Schreiben zur Baugruppe.	ja
0x80A3	Allgemeiner Protokollfehler.	ja
0x80A5	Interner Fehler.	Wert = 0: nein
		Wert <> 0: ja
0x80A7	Baugruppe beschäftigt (Timeout).	ja
0x80A9	Funktion wird von der Baugruppe nicht unterstützt.	ja
0x80AA	Baugruppe meldet einen herstellerspezifischen Fehler seiner Anwendung.	ja
0x80B0	Baugruppe kennt den Datensatz / das Objekt nicht.	ja
0x80B4	Baugruppe meldet einen Zugriff auf einen unzulässigen Bereich.	ja
0x80B5	Baugruppe ist nicht bereit.	ja
0x80B6	Baugruppe verweigert den Zugriff.	ja
0x80B7	Baugruppe meldet einen unzulässigen Bereich eines Parameters oder eines Wertes.	ja
0x80B8	Baugruppe meldet einen unzulässigen Parameter.	ja
0x80B9	Baugruppe meldet einen unzulässigen Typ: Puffer zu klein (Lesen von Teilmengen ist nicht möglich).	ja
0x80C2	Die Baugrupe bearbeitet momentan das mögliche Maximum an Aufträgen für eine CPU.	ja
0x80C3	Die benötigten Betriebsmittel sind momentan belegt.	nein
0x80C4	Interner temporärer Fehler: Auftrag konnte nicht ausgeführt werden.	ja
0x80C5	Baugruppe nicht verfügbar.	ja
0x80D2	Fehler beim Lesen eines SDO aufgrund falscher Aufruf-Parameter.	ja

ERROR_ID

Bei einem *RETVAL* finden Sie nähere Informationen in der *ERROR_ID* sofern verfügbar. Ansonsten ist *ERROR_ID* 0.

Interner Fehler	Beschreibung
0x00000000	No error
0x98110001	Feature not supported
0x98110002	Invalid Index
0x98110003	Invalid Offset
0x98110005	Invalid Size
0x98110006	Invalid Data
0x98110007	Not ready
0x98110008	Busy
0x9811000A	No Memory left
0x9811000B	Invalid Parameter
0x9811000C	Not Found
0x9811000E	Invalid state
0x98110010	Timeout
0x98110011	Open Failed
0x98110012	Send Failed
0x98110014	Invalid Command
0x98110015	Unknown Mailbox Protocol Command
0x98110016	Access Denied
0x98110024	Slave error
0x9811002D	Ethernet link cable disconnected
0x98110031	No mailbox support

CoE-Fehlerwerte	Beschreibung	CoE slave abort code
0x98110040	SDO: Toggle bit not alternated	0x05030000
0x98110041	SDO protocol timed out	0x05040000
0x98110042	SDO: Client/server command specifier not valid or unknown	0x05040001
0x98110043	SDO: Invalid block size (block mode only)	0x05040002
0x98110044	SDO: Invalid sequence number (block mode only)	0x05040003
0x98110045	SDO: CRC error (block mode only)	0x05040004
0x98110046	SDO: Out of memory	0x05040005
0x98110047	SDO: Unsupported access to an object	0x06010000
0x98110048	SDO: Attempt to read a write only object	0x06010001
0x98110049	SDO: Attempt to write a read only object	0x06010002
0x9811004A	SDO: Object does not exist in the object dictionary	0x06020000
0x9811004B	SDO: Object cannot be mapped to the PDO	0x06040041

CoE-Fehlerwerte	Beschreibung	CoE slave abort code
0x9811004C	SDO: The number and length of the objects to be mapped would exceed PDO length	0x06040042
0x9811004D	SDO: General parameter incompatibility reason	0x06040043
0x9811004E	SDO: General internal incompatibility in the device	0x06040047
0x9811004F	SDO: Access failed due to an hardware error	0x06060000
0x98110050	SDO: Data type does not match, length of service parameter does not match	0x06070010
0x98110051	SDO: Data type does not match, length of service parameter too high	0x06070012
0x98110052	SDO: Data type does not match, length of service parameter too low	0x06070013
0x98110053	SDO: Sub-index does not exist	0x06090011
0x98110054	SDO: Value range of parameter exceeded (only for write access)	0x06090030
0x98110055	SDO: Value of parameter written too high	0x06090031
0x98110056	SDO: Value of parameter written too low	0x06090032
0x98110057	SDO: Maximum value is less than minimum value	0x06090036
0x98110058	SDO: General error	0x08000000
0x98110059	SDO: Data cannot be transferred or stored to the application	0x08000020
0x9811005A	SDO: Data cannot be transferred or stored to the application because of local control	0x08000021
0x9811005B	SDO: Data cannot be transferred or stored to the application because of the present device state	0x08000022
0x9811005C	SDO: Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error)	0x08000023
0x9811005D	SDO: Unknown code	unknown
0x9811010E	Command not executed	Slave is not present at the bus

12.1.2 FB 53 - SDO_WRITE - Schreibzugriff auf Objektverzeichnis

Beschreibung

Mit diesem Baustein können Sie auf das Objektverzeichnis von EtherCAT-Slave-Stationen und EtherCAT-Master schreibend zugreifen. Hierbei handelt es sich um einen asynchron arbeitenden Baustein, d.h. die Bearbeitung erstreckt sich über mehrere Baustein-Aufrufe. Sie starten den SDO-Auftrag, indem Sie den FB 53 mit REQ = 1 aufrufen. Über den Ausgangsparameter BUSY und den Ausgangsparameter RETVAL wird der Zustand des Auftrags angezeigt. Die Datensatzübertragung ist abgeschlossen, wenn der Ausgangsparameter BUSY den Wert FALSE angenommen hat.

Die Fehlerbehandlung erfolgt über die Parameter ERROR, ERROR_ID und RETVAL

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
REQ	IN	BOOL	REQ = 1: Aktiviert den SDO-Zugriff bei steigender Flanke.
ID	IN	WORD	Logische Basisadresse der EtherCAT-Slave-Station bzw. des Masters in der Hardwarekonfiguration. Bei einer Ausgabebaugruppe muss Bit 15 gesetzt werden (Bsp. für Adresse 5: ID:=DW#16#8005). Bei einer Mischbaugruppe ist die kleinere der beiden Adressen anzugeben.
INDEX	IN	WORD	Index des Objekts für den SDO-Zugriff.
SUBINDEX	IN	BYTE	Subindex des Objekts für den SDO-Zugriff.
COMPL_ACCESS	IN	BOOL	Mit diesem Parameter wird bestimmt, ob nur ein einzelner Subindex oder das gesamte Objekt geschrieben werden soll.
LEN	IN	INT	Maximale Länge der zu schreibenden Daten.
DONE	OUT	BOOL	Gibt an, ob ein neuer Datensatz übertragen wurde.
BUSY	OUT	BOOL	Dieser Parameter gibt den Bearbeitungsstatus des SDO-Zugriffs an. *BUSY* = 1: SDO-Zugriff ist noch in Bearbeitung.
ERROR	OUT	BOOL	*ERROR* = 1: Beim Schreibvorgang trat ein Fehler auf.
RETVAL	OUT	INT	Rückgabewert (0 = OK)
ERROR_ID	OUT	DWORD	Busspezifischer Fehlercode. Ist während der Bearbeitung des SDO-Zugriffs ein Fehler aufgetreten, so ist in diesem Parameter der SDO-Abort-Fehlercode (EtherCAT-Fehlercode) angegeben.
LEN	OUT	INT	Länge der zu schreibenden Daten.
RECORD	INOUT	ANY	Bereich für die zu schreibenden Daten.

Bitte beachten Sie, dass die an RECORD übergebenen Daten nicht in einem temporären Bereich liegen.

Besonderheiten bei *COMPL_ACCESS* (CompleteAccess)

Bei Aktivierung des Parameters *COMPL_ACCESS* ist folgendes zu beachten:

- Bei *COMPL_ACCESS* = true darf der *SUBINDEX* nur 0 oder 1 betragen! Ansonsten bekommen Sie eine Fehlermeldung.
- Bei *COMPL_ACCESS* = true werden für *SUBINDEX* 0 2 Byte geschrieben, da *SUBINDEX* 1 einen Offset von 2 Byte besitzt.

RETVAL (Rückgabewert)

Zusätzlich zu den hier aufgeführten modulspezifischen Fehlercodes sind auch noch die allgemeingültigen Fehlercodes für FC/SFCs als Rückgabewert möglich. ↳ *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

RETVAL	Beschreibung	Fehlercode in *ERROR_ID*
0x80A0	Negative Quittung beim Lesen von der Baugruppe.	ja
0x80A1	Negative Quittung beim Schreiben zur Baugruppe.	ja
0x80A3	Allgemeiner Protokollfehler.	ja
0x80A5	Interner Fehler.	Wert = 0: nein
		Wert ≠ 0: ja
0x80A7	Baugruppe beschäftigt (Timeout).	ja
0x80A9	Funktion wird von der Baugruppe nicht unterstützt.	ja
0x80AA	Baugruppe meldet einen herstellerspezifischen Fehler seiner Anwendung.	ja
0x80B0	Baugruppe kennt den Datensatz / das Objekt nicht.	ja
0x80B4	Baugruppe meldet einen Zugriff auf einen unzulässigen Bereich.	ja
0x80B5	Baugruppe ist nicht bereit.	ja
0x80B6	Baugruppe verweigert den Zugriff.	ja
0x80B7	Baugruppe meldet einen unzulässigen Bereich eines Parameters oder eines Wertes.	ja
0x80B8	Baugruppe meldet einen unzulässigen Parameter.	ja
0x80B9	Baugruppe meldet einen unzulässigen Typ: Puffer zu klein (Schreiben von Teilmengen ist nicht möglich).	ja
0x80C2	Die Baugrupe bearbeitet momentan das mögliche Maximum an Aufträgen für eine CPU.	ja
0x80C3	Die benötigten Betriebsmittel sind momentan belegt.	nein
0x80C4	Interner temporärer Fehler: Auftrag konnte nicht ausgeführt werden.	ja
0x80C5	Baugruppe nicht verfügbar.	ja
0x80D2	Fehler beim Lesen eines SDO aufgrund falscher Aufruf-Parameter.	ja

ERROR_ID

Bei einem *RETVAL* finden Sie nähere Informationen in der *ERROR_ID* sofern verfügbar. Ansonsten ist *ERROR_ID* 0.

Interner Fehler	Beschreibung
0x00000000	No error
0x98110001	Feature not supported
0x98110002	Invalid Index
0x98110003	Invalid Offset
0x98110005	Invalid Size
0x98110006	Invalid Data
0x98110007	Not ready
0x98110008	Busy
0x9811000A	No Memory left
0x9811000B	Invalid Parameter
0x9811000C	Not Found
0x9811000E	Invalid state
0x98110010	Timeout
0x98110011	Open Failed
0x98110012	Send Failed
0x98110014	Invalid Command
0x98110015	Unknown Mailbox Protocol Command
0x98110016	Access Denied
0x98110024	Slave error
0x9811002D	Ethernet link cable disconnected
0x98110031	No mailbox support

CoE-Fehlerwerte	Beschreibung	CoE slave abort code
0x98110040	SDO: Toggle bit not alternated	0x05030000
0x98110041	SDO protocol timed out	0x05040000
0x98110042	SDO: Client/server command specifier not valid or unknown	0x05040001
0x98110043	SDO: Invalid block size (block mode only)	0x05040002
0x98110044	SDO: Invalid sequence number (block mode only)	0x05040003
0x98110045	SDO: CRC error (block mode only)	0x05040004
0x98110046	SDO: Out of memory	0x05040005
0x98110047	SDO: Unsupported access to an object	0x06010000
0x98110048	SDO: Attempt to read a write only object	0x06010001
0x98110049	SDO: Attempt to write a read only object	0x06010002
0x9811004A	SDO: Object does not exist in the object dictionary	0x06020000
0x9811004B	SDO: Object cannot be mapped to the PDO	0x06040041

CoE-Fehlerwerte	Beschreibung	CoE slave abort code
0x9811004C	SDO: The number and length of the objects to be mapped would exceed PDO length	0x06040042
0x9811004D	SDO: General parameter incompatibility reason	0x06040043
0x9811004E	SDO: General internal incompatibility in the device	0x06040047
0x9811004F	SDO: Access failed due to an hardware error	0x06060000
0x98110050	SDO: Data type does not match, length of service parameter does not match	0x06070010
0x98110051	SDO: Data type does not match, length of service parameter too high	0x06070012
0x98110052	SDO: Data type does not match, length of service parameter too low	0x06070013
0x98110053	SDO: Sub-index does not exist	0x06090011
0x98110054	SDO: Value range of parameter exceeded (only for write access)	0x06090030
0x98110055	SDO: Value of parameter written too high	0x06090031
0x98110056	SDO: Value of parameter written too low	0x06090032
0x98110057	SDO: Maximum value is less than minimum value	0x06090036
0x98110058	SDO: General error	0x08000000
0x98110059	SDO: Data cannot be transferred or stored to the application	0x08000020
0x9811005A	SDO: Data cannot be transferred or stored to the application because of local control	0x08000021
0x9811005B	SDO: Data cannot be transferred or stored to the application because of the present device state	0x08000022
0x9811005C	SDO: Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error)	0x08000023
0x9811005D	SDO: Unknown code	unknown
0x9811010E	Command not executed	Slave is not present at the bus

13 Modulspezifisch - *"Device Specific"*

13.1 Frequenzmessung - *"Frequency Measurement"*

13.1.1 FC 300 ... 303 - Frequenzmessung SLIO konsistent

Übersicht

Mit folgenden VIPA-spezifischen Funktionen können Sie System SLIO Frequenzmess-Module ansteuern, welche über PROFIBUS, PROFINET oder EtherCAT angebunden sind. Der Einsatz unter EtherCAT ist ausschließlich mit einer VIPA EtherCAT-CPU möglich. Von diesen Funktionen wird intern der SFC 14 - DPRD_DAT bzw. SFC 15 - DPWR_DAT für konsistentes Lesen bzw. Schreiben von Nutzdaten aufgerufen. Fehlermeldungen dieser Bausteine werden über den Parameter *ERROR* zurückgeliefert.

Baustein	Symbol	Kommentar
FC 300	FM_SET_CONTROL	Funktion zur Steuerung der Frequenzmessung mit integriertem konsistentem Zugriff.
FC 301	FM_GET_PERIOD	Funktion zur Berechnung der Periodendauer mit integriertem konsistentem Zugriff.
FC 302	FM_GET_FREQUENCY	Funktion zur Berechnung der Frequenz mit integriertem konsistentem Zugriff.
FC 303	FM_GET_SPEED	Funktion zur Berechnung der Drehzahl mit integriertem konsistentem Zugriff.

13.1.2 FC 300 - FM_SET_CONTROL - Control Frequenzmessung konsistent

Beschreibung

Mit dem FC 300 FM_SET_CONTROL können Sie das System SLIO Frequenzmess-Modul steuern. Von dieser Funktion wird intern der SFC 15 - DPWR_DAT für konsistentes Schreiben von Nutzdaten aufgerufen. Hierbei werden Fehlermeldungen des Bausteins über *ERROR* ausgegeben.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
ENABLE_FM	INPUT	BOOL	E, A, M, D, L	Freigabe Frequenzmessung
LADDR_OUT	INPUT	WORD	E, A, M, D, L	Logische Basis-Ausgabeadresse des Frequenzmess-Moduls.
PRESET_CH0	INPUT	DINT	E, A, M, D, L	Kanal 0: Messperiode
PRESET_CH1	INPUT	DINT	E, A, M, D, L	Kanal 1: Messperiode
DONE	OUTPUT	BOOL	E, A, M, D, L	Fertigmeldung (TRUE = OK)
ERROR	OUTPUT	WORD	E, A, M, D, L	Rückgabewert (0 = OK)

ENABLE_FM

Durch Setzen von *ENABLE_FM* werden die über *PRESET_CH0/1* vorgegebenen *Messperioden* an die Kanäle übergeben und bei beiden Kanälen die Messung gestartet. Durch Rücksetzen von *ENABLE_FM* werden beide Kanäle gestoppt.

Nur solange ENABLE_FM gesetzt ist, können ermittelte Werte vom Modul abgerufen werden. Ansonsten erhalten Sie die Fehlermeldung, dass die Kanäle deaktiviert sind.

LADDR_OUT

Projektierte Anfangsadresse aus dem Ausgabebereich des System SLIO Frequenz-Messmoduls, in welchen geschrieben werden soll. Die Adresse wird hexadezimal angegeben.

(Beispiel: Adresse 100: *LADDR_OUT*: = W#16#64)

PRESET_CHx

Geben Sie hier die Messperiode in µs für den entsprechenden Kanal an.

Wertebereich: 1µs ... 8 388 607µs

DONE

Fertigmeldung der Funktion

- TRUE: Funktion wurde ohne Fehler beendet.
- FALSE: Funktion nicht aktiv bzw. es ist ein Fehler aufgetreten.

ERROR (Rückgabewert)

Folgende Codes können zurückgeliefert werden:

Code	Beschreibung
0x0000	Kein Fehler
0x80D2	Kanal 0: Eingangswert Messperiode ≤ 0
0x80D3	Kanal 1: Eingangswert Messperiode ≤ 0
0x80D4	Kanal 0: Eingangswert Messperiode > 8 388 607µs
0x80D5	Kanal 1: Eingangswert Messperiode > 8 388 607µs

Fehler des intern aufgerufenen SFC 15

Code	Beschreibung
0x808x	Systemfehler am Bus-Koppler
0x8090	*LADDR_OUT* ist falsch, mögliche Gründe: - auf dieser Adresse ist kein Modul projektiert - Einschränkung über die Länge der konsistenten Daten wurde nicht beachtet - Anfangsadresse im Parameter LADDR_OUT wurde nicht hexadezimal angegeben
0x8093	Für *LADDR_OUT* existiert kein Bus-Koppler, von dem Sie konsistente Daten lesen können.
0x80A0	Beim Zugriff auf die Peripherie wurde ein Zugriffsfehler erkannt.
0x80B0	Systemfehler am Bus-Koppler
0x80B1	Angegebene Länge des Quellbereichs entspricht nicht der projektierten Nutzdatenlänge.
0x80B2	Systemfehler am Bus-Koppler
0x80B3	Systemfehler am Bus-Koppler

Code	Beschreibung
0x80C1	Die Daten des auf der Baugruppe vorangegangenen Leseauftrags sind von der Baugruppe noch nicht bearbeitet.
0x80C2	Systemfehler am Bus-Koppler
0x80Fx	Systemfehler am Bus-Koppler
0x85xy	Systemfehler am Bus-Koppler
0x8xyy	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

13.1.3 FC 301 - FM_GET_PERIOD - Periodendauer berechnen konsistent

Beschreibung

Mit dem FC 301 FM_GET_PERIOD können Sie die Periodendauer der Eingangssignale beider Kanäle des System SLIO Frequenzmess-Moduls berechnen. Von dieser Funktion wird intern der SFC 14 - DPRD_DAT für konsistentes Lesen von Nutzdaten aufgerufen. Hierbei werden Fehlermeldungen des Bausteins über *ERROR* ausgegeben.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
LADDR_IN	INPUT	WORD	E, A, M, D, L	Logische Basis-Eingabeadresse des Frequenzmess-Moduls.
DONE	OUTPUT	BOOL	E, A, M, D, L	Fertigmeldung (TRUE = OK)
ERROR	OUTPUT	WORD	E, A, M, D, L	Rückgabewert (0 = OK)
PERIOD_CH0	OUTPUT	DINT	E, A, M, D, L	Kanal 0: Periodendauer
PERIOD_CH1	OUTPUT	DINT	E, A, M, D, L	Kanal 1: Periodendauer

LADDR_IN

Projektierte Anfangsadresse aus dem Eingabebereich des System SLIO Frequenzmess-Moduls, aus welchem gelesen werden soll. Die Adresse wird hexadezimal angegeben.

(Beispiel: Adresse 100: *LADDR_IN*: = W#16#64)

DONE

Fertigmeldung der Funktion

- TRUE: Funktion wurde ohne Fehler beendet.
- FALSE: Funktion nicht aktiv bzw. es ist ein Fehler aufgetreten.

PERIOD_CHx

Aktuell ermittelte Periodendauer des entsprechenden Kanals in 100ns.

ERROR (Rückgabewert) Folgende Codes können zurückgeliefert werden:

Code	Beschreibung
0x0000	Kein Fehler
0x80D0	Kanal 0 nicht im Status aktiv
0x80D1	Kanal 1 nicht im Status aktiv
0x80DC	Kanal 0: Gemessener Zeitwert < 0
0x80DD	Kanal 1: Gemessener Zeitwert < 0
0x80DE	Kanal 0: Gemessener Zeitwert > 0x7FFFFFF
0x80DF	Kanal 1: Gemessener Zeitwert > 0x7FFFFFF
0x80E0	Kanal 0: Anzahl ermittelter Flanken = 0
0x80E1	Kanal 1: Anzahl ermittelter Flanken = 0
0x80E2	Kanal 0: Anzahl ermittelter Flanken < 0
0x80E3	Kanal 1: Anzahl ermittelter Flanken < 0
0x80E4	Kanal 0: Anzahl ermittelter Flanken > 0xFFFFFF
0x80E5	Kanal 1: Anzahl ermittelter Flanken > 0xFFFFFF
0x80E8	Kanal 0: Keine gültige Messung innerhalb der vorgegebenen Messperiode
0x80E9	Kanal 1: Keine gültige Messung innerhalb der vorgegebenen Messperiode

Fehler des intern aufgerufenen SFC 14

Code	Beschreibung
0x808x	Systemfehler am Bus-Koppler
0x8090	*LADDR_IN* ist falsch, mögliche Gründe: ■ auf dieser Adresse ist kein Modul projektiert ■ Einschränkung über die Länge der konsistenten Daten wurde nicht beachtet ■ Anfangsadresse im Parameter LADDR_IN wurde nicht hexadezimal angegeben
0x8093	Für *LADDR_IN* existiert kein Bus-Koppler, von dem Sie konsistente Daten lesen können.
0x80A0	Beim Zugriff auf die Peripherie wurde ein Zugriffsfehler erkannt.
0x80B0	Systemfehler am Bus-Koppler
0x80B1	Angegebene Länge des Quellbereichs entspricht nicht der projektierten Nutzdatenlänge.
0x80B2	Systemfehler am Bus-Koppler
0x80B3	Systemfehler am Bus-Koppler
0x80C1	Die Daten des auf der Baugruppe vorangegangenen Leseauftrags sind von der Baugruppe noch nicht bearbeitet.
0x80C2	Systemfehler am Bus-Koppler
0x80Fx	Systemfehler am Bus-Koppler

Code	Beschreibung
0x85xy	Systemfehler am Bus-Koppler
0x8xyy	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

13.1.4 FC 302 - FM_GET_FREQUENCY - Frequenz berechnen konsistent

Beschreibung

Mit dem FC 302 FM_GET_FREQUENCY können Sie die Frequenz der Eingangssignale beider Kanäle des System SLIO Frequenzmess-Moduls berechnen. Von dieser Funktion wird intern der SFC 14 - DPRD_DAT für konsistentes Lesen von Nutzdaten aufgerufen. Hierbei werden Fehlermeldungen des Bausteins über *ERROR* ausgegeben.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
LADDR_IN	INPUT	WORD	E, A, M, D, L	Logische Basis-Eingabeadresse des Frequenzmess-Moduls.
DONE	OUTPUT	BOOL	E, A, M, D, L	Fertigmeldung (TRUE = OK)
ERROR	OUTPUT	WORD	E, A, M, D, L	Rückgabewert (0 = OK)
FREQUENCY_CH0	OUTPUT	DINT	E, A, M, D, L	Kanal 0: Frequenz
FREQUENCY_CH1	OUTPUT	DINT	E, A, M, D, L	Kanal 1: Frequenz

LADDR_IN

Projektierte Anfangsadresse aus dem Eingabebereich des System SLIO Frequenzmess-Moduls, aus welchem gelesen werden soll. Die Adresse wird hexadezimal angegeben.

(Beispiel: Adresse 100: *LADDR_IN*: = W#16#64)

DONE

Fertigmeldung der Funktion

- TRUE: Funktion wurde ohne Fehler beendet.
- FALSE: Funktion nicht aktiv bzw. es ist ein Fehler aufgetreten.

FREQUENCY_CHx

Aktuell ermittelte Frequenz des entsprechenden Kanals in mHz.

ERROR (Rückgabewert)

Folgende Codes können zurückgeliefert werden:

Code	Beschreibung
0x0000	Kein Fehler
0x80D0	Kanal 0 nicht im Status aktiv
0x80D1	Kanal 1 nicht im Status aktiv
0x80DA	Kanal 0: Gemessener Zeitwert = 0

Code	Beschreibung
0x80DB	Kanal 1: Gemessener Zeitwert = 0
0x80DC	Kanal 0: Gemessener Zeitwert < 0
0x80DD	Kanal 1: Gemessener Zeitwert < 0
0x80DE	Kanal 0: Gemessener Zeitwert > 0x7FFFFFF
0x80DF	Kanal 1: Gemessener Zeitwert > 0x7FFFFFF
0x80E2	Kanal 0: Anzahl ermittelter Flanken < 0
0x80E3	Kanal 1: Anzahl ermittelter Flanken < 0
0x80E4	Kanal 0: Anzahl ermittelter Flanken > 0xFFFFFF
0x80E5	Kanal 1: Anzahl ermittelter Flanken > 0xFFFFFF
0x80E6	Kanal 0: Frequenz > 600kHz
0x80E7	Kanal 1: Frequenz > 600kHz
0x80E8	Kanal 0: Keine gültige Messung innerhalb der Messperiode.
0x80E9	Kanal 1: Keine gültige Messung innerhalb der Messperiode.

Fehler des intern aufgerufenen SFC 14

Code	Beschreibung
0x808x	Systemfehler am Bus-Koppler
0x8090	*LADDR_IN* ist falsch, mögliche Gründe: ■ auf dieser Adresse ist kein Modul projektiert ■ Einschränkung über die Länge der konsistenten Daten wurde nicht beachtet ■ Anfangsadresse im Parameter LADDR_IN wurde nicht hexadezimal angegeben
0x8093	Für *LADDR_IN* existiert kein Bus-Koppler, von dem Sie konsistente Daten lesen können.
0x80A0	Beim Zugriff auf die Peripherie wurde ein Zugriffsfehler erkannt.
0x80B0	Systemfehler am Bus-Koppler
0x80B1	Angegebene Länge des Quellbereichs entspricht nicht der projektierten Nutzdatenlänge.
0x80B2	Systemfehler am Bus-Koppler
0x80B3	Systemfehler am Bus-Koppler
0x80C1	Die Daten des auf der Baugruppe vorangegangenen Leseauftrags sind von der Baugruppe noch nicht bearbeitet.
0x80C2	Systemfehler am Bus-Koppler
0x80Fx	Systemfehler am Bus-Koppler
0x85xy	Systemfehler am Bus-Koppler
0x8xyy	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

13.1.5 FC 303 - FM_GET_SPEED - Drehzahl berechnen konsistent

Beschreibung

Mit dem FC 303 FM_GET_SPEED können Sie die Drehzahl der Eingangssignale beider Kanäle des System SLIO Frequenzmess-Moduls berechnen. Von dieser Funktion wird intern der SFC 14 - DPRD_DAT für konsistentes Lesen von Nutzdaten aufgerufen. Hierbei werden Fehlermeldungen des Bausteins über *ERROR* ausgegeben.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
LADDR_IN	INPUT	WORD	E, A, M, D, L	Logische Basis-Eingabeadresse des Frequenzmess-Moduls.
RESOLUTION_CH0	INPUT	DINT	E, A, M, D, L	Kanal 0: Auflösung des Gebers
RESOLUTION_CH1	INPUT	DINT	E, A, M, D, L	Kanal 1: Auflösung des Gebers
DONE	OUTPUT	BOOL	E, A, M, D, L	Fertigmeldung (TRUE = OK)
ERROR	OUTPUT	WORD	E, A, M, D, L	Rückgabewert (0 = OK)
SPEED_CH0	OUTPUT	DINT	E, A, M, D, L	Kanal 0: Drehzahl
SPEED_CH1	OUTPUT	DINT	E, A, M, D, L	Kanal 1: Drehzahl

LADDR_IN

Projektierte Anfangsadresse aus dem Eingabebereich des System SLIO Frequenz-Messmoduls, aus welchem gelesen werden soll. Die Adresse wird hexadezimal angegeben.

(Beispiel: Adresse 100: *LADDR_IN*: = W#16#64)

RESOLUTION_CHx

Geben Sie hier die Auflösung in Inkremente pro Umdrehung für den entsprechenden Kanal an.

DONE

Fertigmeldung der Funktion

- TRUE: Funktion wurde ohne Fehler beendet.
- FALSE: Funktion nicht aktiv bzw. es ist ein Fehler aufgetreten.

SPEED_CHx

Aktuell ermittelte Drehzahl des entsprechenden Kanals in Umdrehungen pro Minute (rpm).

ERROR (Rückgabewert) Folgende Codes können zurückgeliefert werden:

ERROR	Beschreibung
0x0000	Kein Fehler
0x80D0	Kanal 0 nicht im Status aktiv
0x80D1	Kanal 1 nicht im Status aktiv
0x80D6	Kanal 0: Eingangswert RESOLUTION_CH0 = 0
0x80D7	Kanal 1: Eingangswert RESOLUTION_CH1 = 0
0x80D8	Kanal 0: Eingangswert RESOLUTION_CH0 < 0
0x80D9	Kanal 1: Eingangswert RESOLUTION_CH1 < 0
0x80DA	Kanal 0: Gemessener Zeitwert = 0
0x80DB	Kanal 1: Gemessener Zeitwert = 0
0x80DC	Kanal 0: Gemessener Zeitwert < 0
0x80DD	Kanal 1: Gemessener Zeitwert < 0
0x80DE	Kanal 0: Gemessener Zeitwert > 0x7FFFFFF
0x80DF	Kanal 1: Gemessener Zeitwert > 0x7FFFFFF
0x80E2	Kanal 0: Anzahl ermittelter Flanken < 0
0x80E3	Kanal 1: Anzahl ermittelter Flanken < 0
0x80E4	Kanal 0: Anzahl ermittelter Flanken > 0xFFFFFF
0x80E5	Kanal 1: Anzahl ermittelter Flanken > 0xFFFFFF
0x80E6	Kanal 0: Ermittelte Drehzahl > max. (DINT)
0x80E7	Kanal 1: Ermittelte Drehzahl > max. (DINT)
0x80E8	Kanal 0: Keine gültige Messung innerhalb der vorgegebenen Messperiode
0x80E9	Kanal 1: Keine gültige Messung innerhalb der vorgegebenen Messperiode

Fehler des intern aufgerufenen SFC 14

Code	Beschreibung
0x808x	Systemfehler am Bus-Koppler
0x8090	*LADDR_IN* ist falsch, mögliche Gründe: ■ auf dieser Adresse ist kein Modul projektiert ■ Einschränkung über die Länge der konsistenten Daten wurde nicht beachtet ■ Anfangsadresse im Parameter LADDR_IN wurde nicht hexadezimal angegeben
0x8093	Für *LADDR_IN* existiert kein Bus-Koppler, von dem Sie konsistente Daten lesen können.
0x80A0	Beim Zugriff auf die Peripherie wurde ein Zugriffsfehler erkannt.
0x80B0	Systemfehler am Bus-Koppler
0x80B1	Angegebene Länge des Quellbereichs entspricht nicht der projektierten Nutzdatenlänge.

Code	Beschreibung
0x80B2	Systemfehler am Bus-Koppler
0x80B3	Systemfehler am Bus-Koppler
0x80C1	Die Daten des auf der Baugruppe vorangegangenen Leseauftrags sind von der Baugruppe noch nicht bearbeitet.
0x80C2	Systemfehler am Bus-Koppler
0x80Fx	Systemfehler am Bus-Koppler
0x85xy	Systemfehler am Bus-Koppler
0x8xyy	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

13.1.6 FC 310 ... 313 - Frequenzmessung SLIO

Übersicht

Mit folgenden VIPA-spezifischen Funktionen können Sie System SLIO Frequenzmess-Module ansteuern, wenn die Konsistenz der Nutzdaten über das Bus-Protokoll sichergestellt ist und konsistentes Lesen bzw. Schreiben mittels SFC 14 bzw. SFC 15 nicht möglich ist. Innerhalb der Funktionen befinden sich "FM_..."-Parameter, deren Inhalte konsistent über das Bus-System mit dem entsprechenden Ein- bzw. Ausgabebereich des Frequenzmess-Moduls zu verschalten sind. Mit dem Aufruf der Funktionen werden die entsprechenden "FM_..."-Parameter von der Funktion automatisch befüllt.

Baustein	Symbol	Kommentar
FC 310	FM_CONTROL	Funktion zur Steuerung der Frequenzmessung
FC 311	FM_CALC_PERIOD	Funktion zur Berechnung der Periodendauer
FC 312	FM_CALC_FREQUENCY	Funktion zur Berechnung der Frequenz
FC 313	FM_CALC_SPEED	Funktion zur Berechnung der Drehzahl

13.1.7 FC 310 - FM_CONTROL - Control Frequenzmessung

Beschreibung

Mit dem FC 310 FM_CONTROL können Sie das System SLIO Frequenzmess-Modul steuern. Da dieser FC keinen Baustein für konsistentes Schreiben intern aufruft, müssen Sie in Ihrem System die konsistente Übertragung der Daten sicherstellen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
ENABLE_FM	INPUT	BOOL	E, A, M, D, L	Freigabe Frequenzmessung
PRESET_CH0	INPUT	DINT	E, A, M, D, L	Kanal 0: Messperiode
PRESET_CH1	INPUT	DINT	E, A, M, D, L	Kanal 1: Messperiode

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
DONE	OUTPUT	BOOL	E, A, M, D, L	Fertigmeldung (TRUE = OK)
ERROR	OUTPUT	WORD	E, A, M, D, L	Rückgabewert (0 = OK)
FM_PRESET_PERIOD_CH0	OUTPUT	DWORD	E, A, M, D, L	Sollwertvorgabe für Frequenz-mess-Modul Ausgabe-Adresse: +0
FM_PRESET_PERIOD_CH1	OUTPUT	DWORD	E, A, M, D, L	Sollwertvorgabe für Frequenz-mess-Modul Ausgabe-Adresse: +4
FM_CONTROL_CH0	OUTPUT	WORD	E, A, M, D, L	Sollwertvorgabe für Frequenz-mess-Modul Ausgabe-Adresse: +8
FM_CONTROL_CH1	OUTPUT	WORD	E, A, M, D, L	Sollwertvorgabe für Frequenz-mess-Modul Ausgabe-Adresse: +10

ENABLE_FM

Durch Setzen von *ENABLE_FM* wird das entsprechende CONTROL generiert und über *FM_CONTROL_CHx* ausgegeben. Sobald der Inhalt von *FM_CONTROL_CHx* über das Bussystem konsistent an das Frequenzmess-Modul übertragen wurde, werden beide Messungen der Kanäle gestartet. Durch Rücksetzen von *ENABLE_FM* wird die Messung beider Kanäle gestoppt, nachdem *FM_CONTROL_CHx* über das Bussystem konsistent an das Frequenzmess-Modul übertragen wurde.

Nur solange die Messung gestartet ist, können ermittelte Werte vom Modul abgerufen werden. Ansonsten erhalten Sie die Fehlermeldung, dass die Kanäle deaktiviert sind.

PRESET_CHx

Geben Sie hier die Messperiode in µs für den entsprechenden Kanal an.

Wertebereich: 1µs ... 8 388 607µs

DONE

Fertigmeldung der Funktion

- TRUE: Funktion wurde ohne Fehler beendet.
- FALSE: Funktion nicht aktiv bzw. es ist ein Fehler aufgetreten.

FM_PRESET_ PERIOD_CHx

Dieser Parameter enthält die Messperiode für Kanal 0 bzw. Kanal 1. Der Inhalt ist über das entsprechende Bussystem konsistent mit Adresse +0 bzw. +4 im Ausgabebereich des Frequenzmess-Moduls zu verschalten.

FM_CONTROL_CHx

Dieser Parameter enthält das CONTROL, welches über *ENABLE_FM* generiert wird. Der jeweilige Inhalt ist für Kanal 0 bzw. Kanal 1 über das entsprechende Bussystem konsistent mit Adresse +8 bzw. +10 im Ausgabebereich des Frequenzmess-Moduls zu verschalten.

ERROR (Rückgabewert) Folgende Codes können zurückgeliefert werden:

Code	Beschreibung
0x0000	Kein Fehler
0x80D2	Kanal 0: Eingangswert Messperiode ≤ 0
0x80D3	Kanal 1: Eingangswert Messperiode ≤ 0
0x80D4	Kanal 0: Eingangswert Messperiode > 8 388 607µs
0x80D5	Kanal 1: Eingangswert Messperiode > 8 388 607µs

13.1.8 FC 311 - FM_CALC_PERIOD - Periodendauer berechnen

Beschreibung Mit dem FC 311 FM_CALC_PERIOD können die Periodendauer der Eingangssignale beider Kanäle berechnen. Da dieser FC keinen Baustein für konsistentes Lesen intern aufruft, müssen Sie in Ihrem System die konsistente Übertragung der Daten sicherstellen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
FM_PERIOD_CH0	INPUT	DWORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +0
FM_PERIOD_CH1	INPUT	DWORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +8
FM_RISING_EDGES_CH0	INPUT	DWORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +4
FM_RISING_EDGES_CH1	INPUT	DWORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +12
FM_STATUS_CH0	INPUT	WORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +16
FM_STATUS_CH1	INPUT	WORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +18
DONE	OUTPUT	BOOL	E, A, M, D, L	Fertigmeldung (TRUE = OK)

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
ERROR	OUTPUT	WORD	E, A, M, D, L	Rückgabewert (0 = OK)
PERIOD_CH0	OUTPUT	DINT	E, A, M, D, L	Kanal 0: Berechnete Periodendauer
PERIOD_CH1	OUTPUT	DINT	E, A, M, D, L	Kanal 1: Berechnete Periodendauer

FM_PERIOD_CHx

Dieser Parameter enthält den gemessenen Zeitwert von Kanal 0 bzw. Kanal 1. Der Inhalt ist über das entsprechende Bussystem konsistent mit Adresse +0 bzw. +4 im Eingabebereich des Frequenzmess-Moduls zu verschalten.

FM_RISING_ EDGES_CHx

Dieser Parameter enthält die ermittelte Anzahl steigender Flanken für Kanal 0 bzw. Kanal 1. Der Inhalt ist über das entsprechende Bussystem konsistent mit Adresse +8 bzw. +12 im Eingabebereich des Frequenzmess-Moduls zu verschalten.

FM_STATUS_CHx

Dieser Parameter enthält den Status für Kanal 0 bzw. Kanal 1. Der Inhalt ist über das entsprechende Bussystem konsistent mit Adresse +16 bzw. +18 im Eingabebereich des Frequenzmess-Moduls zu verschalten.

DONE

Fertigmeldung der Funktion

- TRUE: Funktion wurde ohne Fehler beendet.
- FALSE: Funktion nicht aktiv bzw. es ist ein Fehler aufgetreten.

PERIOD_CHx

Aktuell ermittelte Periodendauer des entsprechenden Kanals in 100ns.

ERROR (Rückgabewert) Folgende Codes können zurückgeliefert werden:

Code	Beschreibung
0x0000	Kein Fehler
0x80D0	Kanal 0 nicht im Status aktiv
0x80D1	Kanal 1 nicht im Status aktiv
0x80DC	Kanal 0: Gemessener Zeitwert < 0
0x80DD	Kanal 1: Gemessener Zeitwert < 0
0x80DE	Kanal 0: Gemessener Zeitwert > 0x7FFFFFF
0x80DF	Kanal 1: Gemessener Zeitwert > 0x7FFFFFF
0x80E0	Kanal 0: Anzahl ermittelter Flanken = 0
0x80E1	Kanal 1: Anzahl ermittelter Flanken = 0
0x80E2	Kanal 0: Anzahl ermittelter Flanken < 0
0x80E3	Kanal 1: Anzahl ermittelter Flanken < 0
0x80E4	Kanal 0: Anzahl ermittelter Flanken > 0xFFFFFF
0x80E5	Kanal 1: Anzahl ermittelter Flanken > 0xFFFFFF
0x80E8	Kanal 0: Keine gültige Messung innerhalb der vorgegebenen Messperiode
0x80E9	Kanal 1: Keine gültige Messung innerhalb der vorgegebenen Messperiode

13.1.9 FC 312 - FM_CALC_FREQUENCY - Frequenz berechnen

Beschreibung Mit dem FC 312 FM_CALC_FREQUENCY können Sie die Frequenz der Eingangssignale beider Kanäle berechnen. Da dieser FC keinen Baustein für konsistentes Lesen intern aufruft, müssen Sie in Ihrem System die konsistente Übertragung der Daten sicherstellen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
FM_PERIOD_CH0	INPUT	DWORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +0
FM_PERIOD_CH1	INPUT	DWORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +8
FM_RISING_EDGES_CH0	INPUT	DWORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +4
FM_RISING_EDGES_CH1	INPUT	DWORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +12

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
FM_STATUS_CH0	INPUT	WORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +16
FM_STATUS_CH1	INPUT	WORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +18
DONE	OUTPUT	BOOL	E, A, M, D, L	Rückmeldung (TRUE = OK)
ERROR	OUTPUT	WORD	E, A, M, D, L	Rückgabewert (0 = OK)
FREQUENCY_CH0	OUTPUT	DINT	E, A, M, D, L	Kanal 0: Berechnete Frequenz
FREQUENCY_CH1	OUTPUT	DINT	E, A, M, D, L	Kanal 1: Berechnete Frequenz

FM_PERIOD_CHx

Dieser Parameter enthält den gemessenen Zeitwert für Kanal 0 bzw. Kanal 1. Der Inhalt ist über das entsprechende Bussystem konsistent mit Adresse +0 bzw. +4 im Eingabebereich des Frequenzmess-Moduls zu verschalten.

FM_RISING_ EDGES_CHx

Dieser Parameter enthält die ermittelte Anzahl steigender Flanken für Kanal 0 bzw. Kanal 1. Der Inhalt ist über das entsprechende Bussystem konsistent mit Adresse +8 bzw. +12 im Eingabebereich des Frequenzmess-Moduls zu verschalten.

FM_STATUS_CHx

Dieser Parameter enthält den Status für Kanal 0 bzw. Kanal 1. Der Inhalt ist über das entsprechende Bussystem konsistent mit Adresse +16 bzw. +18 im Eingabebereich des Frequenzmess-Moduls zu verschalten.

DONE

Fertigmeldung der Funktion

- TRUE: Funktion wurde ohne Fehler beendet.
- FALSE: Funktion nicht aktiv bzw. es ist ein Fehler aufgetreten.

FREQUENCY_CHx

Aktuell ermittelte Frequenz des entsprechenden Kanals in mHz.

ERROR (Rückgabewert)

Folgende Codes können zurückgeliefert werden:

Code	Beschreibung
0x0000	Kein Fehler
0x80D0	Kanal 0 nicht im Status aktiv
0x80D1	Kanal 1 nicht im Status aktiv
0x80DA	Kanal 0: Gemessener Zeitwert = 0
0x80DB	Kanal 1: Gemessener Zeitwert = 0

Code	Beschreibung
0x80DC	Kanal 0: Gemessener Zeitwert < 0
0x80DD	Kanal 1: Gemessener Zeitwert < 0
0x80DE	Kanal 0: Gemessener Zeitwert > 0x7FFFFFF
0x80DF	Kanal 1: Gemessener Zeitwert > 0x7FFFFFF
0x80E2	Kanal 0: Anzahl ermittelter Flanken < 0
0x80E3	Kanal 1: Anzahl ermittelter Flanken < 0
0x80E4	Kanal 0: Anzahl ermittelter Flanken > 0xFFFFFF
0x80E5	Kanal 1: Anzahl ermittelter Flanken > 0xFFFFFF
0x80E6	Kanal 0: Frequenz > 600kHz
0x80E7	Kanal 1: Frequenz > 600kHz
0x80E8	Kanal 0: Keine gültige Messung innerhalb der Messperiode.
0x80E9	Kanal 1: Keine gültige Messung innerhalb der Messperiode.

13.1.10 FC 313 - FM_CALC_SPEED - Drehzahl berechnen

Beschreibung

Mit dem FC 313 FM_CALC_SPEED können Sie die Drehzahl der Eingangssignale beider Kanäle berechnen. Da dieser FC keinen Baustein für konsistentes Lesen intern aufruft, müssen Sie in Ihrem System die konsistente Übertragung der Daten sicherstellen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
FM_PERIOD_CH0	INPUT	DWORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +0
FM_PERIOD_CH1	INPUT	DWORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +8
FM_RISING_EDGES_CH0	INPUT	DWORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +4
FM_RISING_EDGES_CH1	INPUT	DWORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +12
FM_STATUS_CH0	INPUT	WORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +16
FM_STATUS_CH1	INPUT	WORD	E, A, M, D, L	Istwert von Frequenzmess-Modul Eingabe-Adresse: +18

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
RESOLUTION_CH0	INPUT	DINT	E, A, M, D, L	Kanal 0: Auflösung des Gebers
RESOLUTION_CH1	INPUT	DINT	E, A, M, D, L	Kanal 1: Auflösung des Gebers
DONE	OUTPUT	BOOL	E, A, M, D, L	Fertigmeldung (TRUE = OK)
ERROR	OUTPUT	WORD	E, A, M, D, L	Rückgabewert (0 = OK)
SPEED_CH0	OUTPUT	DINT	E, A, M, D, L	Kanal 0: Berechnete Drehzahl
SPEED_CH1	OUTPUT	DINT	E, A, M, D, L	Kanal 1: Berechnete Drehzahl

FM_PERIOD_CHx

Dieser Parameter enthält den gemessenen Zeitwert für Kanal 0 bzw. Kanal 1. Der Inhalt ist über das entsprechende Bussystem konsistent mit Adresse +0 bzw. +4 im Eingabebereich des Frequenzmess-Moduls zu verschalten.

FM_RISING_ EDGES_CHx

Dieser Parameter enthält die ermittelte Anzahl steigender Flanken für Kanal 0 bzw. Kanal 1. Der Inhalt ist über das entsprechende Bussystem konsistent mit Adresse +8 bzw. +12 im Eingabebereich des Frequenzmess-Moduls zu verschalten.

FM_STATUS_CHx

Dieser Parameter enthält den Status für Kanal 0 bzw. Kanal 1. Der Inhalt ist über das entsprechende Bussystem konsistent mit Adresse +16 bzw. +18 im Eingabebereich des Frequenzmess-Moduls zu verschalten.

RESOLUTION_CHx

Geben Sie hier die Auflösung in Inkremente pro Umdrehung für den entsprechenden Kanal an.

DONE

Zustandsparameter des Funktionsbausteins

- TRUE: Funktion wurde ohne Fehler beendet.
- FALSE: Funktion nicht aktiv bzw. es ist ein Fehler aufgetreten.

SPEED_CHx

Aktuell ermittelte Drehzahl des entsprechenden Kanals in Umdrehungen pro Minute (rpm).

ERROR (Rückgabewert) Folgende Codes können zurückgeliefert werden:

ERROR	Beschreibung
0x0000	Kein Fehler
0x80D0	Kanal 0 nicht im Status aktiv
0x80D1	Kanal 1 nicht im Status aktiv
0x80D6	Kanal 0: Eingangswert RESOLUTION_CH0 = 0
0x80D7	Kanal 1: Eingangswert RESOLUTION_CH1 = 0
0x80D8	Kanal 0: Eingangswert RESOLUTION_CH0 < 0
0x80D9	Kanal 1: Eingangswert RESOLUTION_CH1 < 0
0x80DA	Kanal 0: Gemessener Zeitwert = 0
0x80DB	Kanal 1: Gemessener Zeitwert = 0
0x80DC	Kanal 0: Gemessener Zeitwert < 0
0x80DD	Kanal 1: Gemessener Zeitwert < 0
0x80DE	Kanal 0: Gemessener Zeitwert > 0x7FFFFFF
0x80DF	Kanal 1: Gemessener Zeitwert > 0x7FFFFFF
0x80E2	Kanal 0: Anzahl ermittelter Flanken < 0
0x80E3	Kanal 1: Anzahl ermittelter Flanken < 0
0x80E4	Kanal 0: Anzahl ermittelter Flanken > 0xFFFFFF
0x80E5	Kanal 1: Anzahl ermittelter Flanken > 0xFFFFFF
0x80E6	Kanal 0: Ermittelte Drehzahl > max. (DINT)
0x80E7	Kanal 1: Ermittelte Drehzahl > max. (DINT)
0x80E8	Kanal 0: Keine gültige Messung innerhalb der vorgegebenen Messperiode
0x80E9	Kanal 1: Keine gültige Messung innerhalb der vorgegebenen Messperiode

13.2 Energiemessung - *"Energy Measurement"*

13.2.1 Übersicht

13.2.1.1 Begriffe

Messgröße Eine *Messgröße* ist eine physikalische Größe, die zu messen ist, z.B. Strom, Spannung oder Temperatur.

Messwert Ein *Messwert* ist ein Wert einer Messgröße, der durch Messung oder durch Berechnung ermittelt wird.

ID Im Modul ist jeder *Messgröße* eine *ID* zugeordnet. Der Zugriff auf den Messwert einer Messgröße erfolgt durch Angabe der entsprechenden *ID*.

DS-ID

Sobald das Modul über die DC 24V Leistungsversorgung versorgt wird, beginnt die Messwerterfassung und der Zählvorgang der Energiezähler wird mit den remanent gespeicherten Zählerständen fortgesetzt. Die Messwerte aller Messgrößen werden unter einer Datensatz-ID *DS-ID* im Modul gespeichert. Hierbei ist folgendes zu beachten:

- Alle Messwerte mit der gleichen *DS-ID* stammen aus der gleichen Messung und sind konsistent.
- Durch Angabe der *DS-ID* können Sie die einzelnen Messwerte aus der gleichen Messung adressieren.
- Die *DS-ID* umfasst die Werte 1 ... 15.
- Zur Aktualisierung der Messwerte ist die *DS-ID* um 1 zu inkrementieren. Nach dem Wert 15 muss wieder die 1 folgen.
- Wird die *DS-ID* inkrementiert und es liegt noch kein neuer Wert vor, wird der aktuelle Wert geliefert. Hierbei meldet das Energiemess-Modul einen Fehler.
- *DS-ID* = 0 - Autoinkrement-Modus
  - Mit *DS-ID* = 0 erfolgt eine Anfrage im *Autoinkrement-Modus*. Hierbei liefert das Modul immer den aktuellsten Messwert zurück. Sobald hier ein neuer Messwert vorhanden ist, wird die *DS-ID* innerhalb der Werte 1 ... 15 um 1 inkrementiert. Sollte noch kein neuer Messwert vorliegen, bleibt die *DS-ID* unverändert. Hierbei meldet das Energiemess-Modul einen Fehler.
- Die Eindeutigkeit eines Messwerts besteht immer aus der *ID* der Messgröße und der *DS-ID*.

Frame

Im Modul können Sie mehrere Messgrößen zu einem Datenpaket (Frame) zusammenfassen, welches in einem Durchgang übertragen wird. Ein Datenpaket umfasst 12Byte Nutzdaten. Unter Berücksichtigung der Nutzdatenlänge von 12Byte können Sie durch Angabe der Messgrößen-*ID* die Inhalte eines Frames definieren. Sie können bis zu 256 Frames (*Frame 0* ... *Frame 255*) konfigurieren. Hierbei ist folgendes zu beachten:

- Die Definition von *Frame 1* bis *Frame 255* erfolgt mit dem Befehl *Set_Frame*. .
- *Frame 0* mit den entsprechenden Messgrößen kann ausschließlich über die Parametrierung konfiguriert werden.
- Mit dem Telegrammtyp *Zero Frame* haben Sie Zugriff auf das Datenpaket von Frame 0. Nach dem Hochlauf des Moduls erfolgen automatische *Zero Frame*-Anforderungen solange bis vom Kopfmodul die Prozessdatenkommunikation übernommen wird.

FR-ID

Bei der Definition von Frames mittels *"Set Frame"* werden diese über die *FR-ID* mit einer beliebigen Ziffer aus 0 ... 255 versehen. Durch Angabe der *FR-ID* können Sie das entsprechende Frame anfordern.

Datentyp

Nachfolgend sind die Datentypen aufgeführt, welche im Modul verwendet werden. Die Länge ist insbesondere bei der Definition von *Frames* zu berücksichtigen.

Datentyp	Länge in Byte	Beschreibung
UINT_8	1	Ganzzahl 8Bit
UINT_16	2	Ganzzahl 16Bit
UINT_32	4	Ganzzahl 32Bit
INT_8	1	Vorzeichenbehaftete Ganzzahl 8Bit
INT_16	2	Vorzeichenbehaftete Ganzzahl 16Bit
INT_32	4	Vorzeichenbehaftete Ganzzahl 32Bit
FLOAT	4	32Bit Gleitkommazahl nach IEEE 754

13.2.1.2 Funktionsweise

Übersicht

- Das Energiemess-Modul dient der Energiemessung eines 3-Phasen Anschlusses. Hierbei ermittelt das Modul neben Spannung, Strom und Phase viele weitere Messgrößen.
- Für manche Messgrößen lassen sich Grenzwerte parametriert. Bei Über- bzw. Unterschreiten werden entsprechende Alarm-Status-Bits gesetzt. Das Modul unterstützt mehrere Kommandos (CMD). Beispielsweise lassen sich hiermit Alarm-Status-Bits wieder zurücksetzen.
- Mit dem Funktionsbaustein FB 325 und der zugehörigen Datenstruktur vom Typ UDT 325 können Sie Energie-Messwerte und Alarm-Status-Bits des Energiemess-Moduls lesen und Kommandos auf dem Modul ausführen. Hierbei kommuniziert der FB 325 über die zyklischen E/A-Daten (je 16 Byte) des Moduls, welche beim Aufruf des FB 325 entsprechend anzugeben sind.
- Die eigentliche Auftragsschnittstelle ist über die Datenstruktur vom Typ UDT 325 realisiert. Hierdurch ist eine einfache Ansteuerung und Auswertung beispielsweise über ein Touch Panel möglich.

Beschaltung des FB 325

- Bei der Projektierung ist auf eine korrekte Verschaltung der Parameter CHANNEL_IN und CHANNEL_OUT des FB 325 zu achten. Ansonsten erhalten Sie eine Timeout-Fehlermeldung.
  - CHANNEL_IN ist auf die 16Byte Eingabedaten des Energiemess-Moduls zu verschalten.
  - CHANNEL_OUT auf die 16Byte Ausgabedaten des Energiemess-Moduls zu verschalten.

Zyklische Messwert-Erfassung

- Indem Sie nach einem PowerON einen manuellen Reset am Energiemess-Modul durchführen, können Sie temporäre Fehlermeldungen vermeiden. Setzen Sie hierzu in der Datenstruktur *MEAS_DATA* des FB 325 Bit 7 der Variable *Header.Control_Global*.
- Mit den Grundeinstellungen der UDT 325 werden alle Messwerte des Energiemess-Moduls mit einer Periode von 1s gelesen und in der Datenstruktur *MEAS_DATA* gespeichert. Die Periode können Sie über Variable *Header.Polltime* in der Datenstruktur *MEAS_DATA* des FB 325 anpassen.

Zur Validierung der Aktualität Ihrer Messwerte können Sie den Parameter DS-ID auf den Zeitpunkt seiner letzten Änderung überprüfen. Sobald ein neuer Messwert vorhanden ist, wird die DS-ID innerhalb der Werte 1 ... 15 um 1 inkrementiert. Der Zeitpunkt der letzten Änderung ist gleichzeitig das Alter der letzten Messwerte.

Manuelle Messwerterfassung

Zur manuellen Erfassung der Messwerte setzen Sie in der Datenstruktur *MEAS_DATA* des FB 325 Bit 1 der Variable *Header.Control_Global*. Ist das Bit gesetzt, werden die Messwerte einmalig vom Energiemess-Modul gelesen und danach das Bit wieder zurück gesetzt.

Selektion von *Messgrößen*

Per Default werden die Messwerte aller *Messgrößen* periodisch gelesen. Sie haben aber die Möglichkeit die *Messgrößen* in der Datenstruktur *MEAS_DATA* zu selektieren. Über Bit 0 der Variable *Data.[Name der Messgröße].Read_Mode* können Sie den Zugriff auf den Messwert der entsprechenden *Messgröße* einstellen. Bitte beachten Sie, dass hier die Messgrößen-IDs in Gruppen zusammengefasst sind. Sobald mindestens ein Messwert einer *Messgröße* einer Gruppe gelesen werden soll, werden die Messwerte aller *Messgrößen* dieser Gruppe gelesen. Soll beispielsweise der Wert der Messgröße mit der ID-Nr. 4 gelesen werden, so werden auch die mit ID-Nr. 5 und 6 gelesen. Es gibt folgende Gruppierungen:

Gruppe	IDs der Messgrößen	Gruppe	IDs der Messgrößen	Gruppe	IDs der Messgrößen
1	1, 2, 3	6	16, 17, 18	11	31, 32, 33
2	4, 5, 6	7	19, 20, 21	12	34, 35, 36
3	7, 8, 9	8	22, 23, 24	13	37, 38, 39
4	10, 11, 12	9	25, 26, 27	14	40, 41
5	13, 14, 15	10	28, 29, 30		

Die gelesenen Messwerte werden in die entsprechenden Variablen von *Data.[Name der Messgröße].Value* eingetragen. Bei nicht gelesenen Messwerten ist *Value* = 0.

Kommando-Schnittstelle

Kommandos können Sie über die Datenstruktur *MEAS_DATA* durch Setzten der entsprechenden Bits in der Variablen *Header.Cmd* auslösen. Sind mehrere Bits gesetzt, werden sie nacheinander abgearbeitet. Hierbei stehen Ihnen folgende Kommandos zur Verfügung:

- Bit 0: Rücksetzen aller Wirkenergiezähler
- Bit 1: Reset auf dem Stromwandler auslösen
- Bit 2: Rücksetzen von *Status Messung*
- Bit 3: Schreiben der Energiesollwerte aus "SetValues" auf die ID3...ID8.

Fehlerverhalten

- Fehlermeldungen, welche bei der Initialisierung des Bausteins bzw. beim Lesen von Messwerten auftreten, finden Sie in der Datenstruktur *MEAS_DATA* unter *Header.Status_Global*.
- Fehlermeldungen, welche bei der Kommando-Abarbeitung auftreten, finden Sie unter *Header.Status_Cmd* und die Detailinformationen unter *Header.Error_ID*
- Im Fehlerfall setzt der Funktionsbaustein die Auftragsbearbeitung fort. Hierbei werden die fehlerhaft ausgeführten Aufträge wiederholt. Die Messwerte in der Datenstruktur *MEAS_DATA* werden durch Fehlermeldungen nicht beeinflusst.

13.2.2 FB 325 - EM_COM_R1 - Kommunikation mit 031-1PAxx

Übersicht

Dieser Baustein ermöglicht die Kommunikation mit den Modulen 031-1PAxx zur Energiezählung und Leistungsmessung. Für die Kommunikation ist ein Datenbaustein erforderlich. Hierbei erhält der DB seine Struktur aus der UDT 325 EM_DATA_R1. Der Baustein besitzt folgende Funktionalitäten:

- Grundparameter laden nach dem Anlauf
- Ablage von Parameter, Grenzwerte, Messwerte und Meldungen
- Übertragung konsistenter Messwerte
- Schreiben von Sollwerten
- Definition der Messwerte mittels UDT-Struktur
- Kommunikation mittels Telegrammtyp und ID
- Funktionsdiagnose, Verbindungsüberwachung und Störmeldeauswertung

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
MODE	INPUT	BYTE	■ 0x01 = Datenaustausch via Prozessdaten Aktuell wird nur MODE = 0x01 unterstützt
CHANNEL_IN	INPUT	ANY	Zeiger auf die Eingangsdaten ■ Mit MODE = 0x01 ist ausschließlich Datentyp BYTE und Länge 16 zulässig. Beispiel: P#E100.0 BYTE 16 oder P#DB10.DBX0.0 BYTE 16
CHANNEL_OUT	INPUT	ANY	Zeiger auf die Ausgangsdaten ■ Mit MODE = 0x01 ist ausschließlich Datentyp BYTE und Länge 16 zulässig. Beispiel: P#A100.0 BYTE 16 oder P#DB10.DBX16.0 BYTE 16
MEAS_DATA	IN_OUT	UDT	■ UDT für die Messwerte *Kap. 13.2.3 "UDT 325 - EM_DATA_R1 - Datenstruktur für FB 325" Seite 462* ■ Bitte beachten Sie, dass diese Struktur nicht in den temporären Lokaldaten liegen darf!

13.2.3 UDT 325 - EM_DATA_R1 - Datenstruktur für FB 325

13.2.3.1 Struktur

UDT 325 Die UDT 325 ist dynamisch aufgebaut und besitzt folgende Grundstruktur.

UDT-Bereiche	Beschreibung
UDT - Header	Struktur für die Header-Daten
UDT - Daten	Gleiche Datenstruktur für die einzelnen *Messgrößen*. Eine *Messgröße* ist eine physikalische Größe, die zu messen ist, z.B. Strom, Spannung oder Temperatur. Eine Übersicht der Messgrößen finden Sie im Handbuch zu ihrem Energiemess-Modul.
...	
UDT - Daten	
UDT - SetValues	Struktur für die Sollwertvorgabe

UDT - Header	Deklaration	Datentyp	Beschreibung
Timeout	INPUT	TIME	■ Timeout-Zeit für die Auftragsbearbeitung. Bei Überschreiten von *Timeout* wird der Auftrag abgebrochen und eine entsprechende Fehlermeldung ausgegeben.
Polltime	INPUT	TIME	■ Intervall für das periodische Lesen ■ *Polltime* ist nur relevant, wenn die Messwerte im Intervall von *Polltime* periodisch gelesen werden d.h. wenn Bit 0 von *Header.Control_Global* gesetzt ist. Ist *Polltime* kleiner als das schnellstmögliche Intervall, werden die Messwerte im schnellstmöglichen Intervall gelesen.
Control_Global	INPUT	BYTE	0: nicht aktiv, 1: aktiv ■ Bit 0: Periodische Ausführung entsprechend der *Polltime* (Default) ■ Bit 1: Einmalige Ausführung - Bit wird nach der Ausführung zurückgesetzt. ■ Bit 6 ... 2: reserviert ■ Bit 7: Neuinitialisierung des Bausteins, indem die Konfiguration neu gesendet wird

UDT - Header	Deklaration	Datentyp	Beschreibung
Status_Global	OUTPUT	BYTE	Bausteinstatus ■ 0x00: Nicht bearbeitet ■ 0x01: In Bearbeitung (BUSY) ■ 0x02: Fertig ohne Fehler (DONE) ■ 0x80: Fehler bei der Bearbeitung (ERROR)
Status Alarm_Global	OUTPUT	BYTE	Entspricht B3: Header-Byte 3 - *Sammelstatus* ■ Bit 0: Frequenz *F_MAX* überschritten ■ Bit 1: Frequenz *F_MIN* unterschritten ■ Bit 2: Temperatur *T_MAX* überschritten ■ Bit 3: Spannung *VRMS_MAX* überschritten ■ Bit 4: Spannung *VRMS_MIN* unterschritten ■ Bit 5: Wirkungsgrad *PF_MIN* unterschritten ■ Bit 6: Strom *IRMS_MAX* überschritten ■ Bit 7: reserviert
Cmd	INPUT	BYTE	0: nicht aktiv, 1: aktiv ■ Bit 0: Rücksetzen aller Wirkenergiezähler ■ Bit 1: Reset auf dem Stromwandler auslösen ■ Bit 2: Rücksetzen von *Status Messung* ■ Bit 3: Schreiben der Energiesollwerte aus "SetValues" auf die ID3 ... 8 und ID 39 ... 40. Sind mehrere Bits gesetzt, werden sie nacheinander abgearbeitet. Hinweis: Schreiben der Energiesollwerte setzt eine Protokollversion Major ≥ 1 Minor ≥ 1 voraus!
Status_Cmd	OUTPUT	BYTE	Status Kommando ■ 0x00: Nicht bearbeitet ■ 0x01: In Bearbeitung (BUSY) ■ 0x02: Fertig ohne Fehler (DONE) ■ 0x80: Fehler bei der Bearbeitung (ERROR) - siehe Error_ID
Jobtime	OUTPUT	TIME	■ Zeitdauer, welche für das Auslesen der Messwerte bzw. für das Ausführen eines Kommandos erforderlich war.
DsID	OUTPUT	BYTE	Nummer der aktuellen DS-ID *"DS-ID" Seite 459*
Frame_ID	OUTPUT	BYTE	Nummer der aktuellen FR-ID *"FR-ID" Seite 459*
Error_ID	OUTPUT	WORD	Detaillierte Fehlerinformationen
Status_ReadVersion	OUTPUT	BYTE	Status *Read FW Version* ■ 0x00: nie ausgeführt ■ 0x01: Busy ■ 0x02: Done ■ 0x80: Error
Reserve	STATIC	ARRAY of BYTE (1...15)	reserviert
VersionInfo		Struct	Die Firmware-Version wird automatisch ermittelt
FirmwareMajor	OUTPUT	Byte	Firmware Version: Major
FirmwareMinor	OUTPUT	Byte	Firmware Version: Minor

UDT - Header	Deklaration	Datentyp	Beschreibung
FirmwareRevision	OUTPUT	Byte	Firmware Revision
ProtocollMajor	OUTPUT	Byte	Protokoll Version: Major
ProtocollMinor	OUTPUT	Byte	Protokoll Version: Minor
ProtocollRevsion	OUTPUT	Byte	Protokoll Version: Revision
ChipDateYear	OUTPUT	Word	Datum Mess-Chip: Jahr
ChipDateMonth	OUTPUT	Byte	Datum Mess-Chip: Monat
ChipDateDay	OUTPUT	Byte	Datum Mess-Chip: Tag

Gleiche Datenstruktur für die einzelnen *Messgrößen*. Eine Übersicht der *Messgrößen* finden Sie im Handbuch zu ihrem Energiemess-Modul.

UDT - Daten	Deklaration	Datentyp	Beschreibung
Name	IN_OUT	STRUCT	■ Name der *Messgröße*
Read_Mode	INPUT	BYTE	■ Bit 0: Zugriff auf den Messwert der Messgröße – 0: Messwert soll nicht gelesen werden. – 1: Messwert soll gelesen werden.
Value	OUTPUT	DWORD	■ Aktueller Messwert

UDT - SetValues	Deklaration	Datentyp	Beschreibung
SetValues		STRUCT	
EN_L1_CONSUMED	INPUT	DWORD	Sollwert für Wirkenergie L1 Verbraucher: UINT32, 1Wh
EN_L1_ DELIVERED	INPUT	DWORD	Sollwert für Wirkenergie L1 Erzeuger: UINT32, 1Wh
EN_L2_CONSUMED	INPUT	DWORD	Sollwert für Wirkenergie L2 Verbraucher: UINT32, 1Wh
EN_L2_ DELIVERED	INPUT	DWORD	Sollwert für Wirkenergie L2 Erzeuger: UINT32, 1Wh
EN_L3_CONSUMED	INPUT	DWORD	Sollwert für Wirkenergie L3 Verbraucher: UINT32, 1Wh
EN_L3_ DELIVERED	INPUT	DWORD	Sollwert für Wirkenergie L3 Erzeuger: UINT32, 1Wh
EXCESS_ACTIVE_EN_CONSUME	INPUT	DWORD	Sollwert für Überlauf Energiezähler Phase 1 ... 3 Verbraucher ■ 0xXX112233 – XX: nicht genutzt – 11: Sollwert (Byte) für Überlauf Energiezähler Phase 1 Verbraucher – 22: Sollwert (Byte) für Überlauf Energiezähler Phase 2 Verbraucher – 33: Sollwert (Byte) für Überlauf Energiezähler Phase 3 Verbraucher Wird bei Überlauf des Energiezählers (ID = 1) um 1 inkrementiert
EXCESS_ACTIVE_EN_DELIVERED	INPUT	DWORD	Sollwert für Überlauf Energiezähler Phase 1 ... 3 Erzeuger ■ 0xXX112233 – XX: nicht genutzt – 11: Sollwert (Byte) für Überlauf Energiezähler Phase 1 Erzeuger – 22: Sollwert (Byte) für Überlauf Energiezähler Phase 2 Erzeuger – 33: Sollwert (Byte) für Überlauf Energiezähler Phase 3 Erzeuger Wird bei Überlauf des Energiezählers (ID = 2) um 1 inkrementiert

13.2.3.2 Fehlerrückmeldung

ERROR ID	Beschreibung
0x0000	kein Fehler
0x8060	Fehler: Eine aktuellere Protokollversion ist erforderlich
0x8070	Fehler: Parameter MODE
0x8073	Fehler: Parameter CHANNEL_IN passt nicht zu MODE
0x8074	Fehler: Parameter CHANNEL_OUT passt nicht zu MODE
0x8080	Fehler: *"SET Frame"*: Timeout beim Zugriff erkannt
0x8081	Fehler: *"READ Frame"*: Timeout beim Zugriff erkannt
0x8082	Fehler: *"CMD Frame"*: Timeout beim Zugriff erkannt
0x8083	Fehler: Timeout beim automatischen Lesen der Firmwareinformation
0x80A1	Status Kommunikation: Fehler: Datensatz konnte nicht aktualisiert werden
0x80A2	Status Kommunikation: Fehler: *"DS-ID"*
0x80A3	Status Kommunikation: Fehler: Telegrammlänge
0x80A4	Status Kommunikation: Fehler: *Frame* zu groß
0x80A5	Status Kommunikation: Fehler: *Frame* nicht definiert
0x80A6	Status Kommunikation: Fehler: Messgröße nicht vorhanden
0x80A7	Status Kommunikation: *"CMD Frame"* - Kommando konnte nicht ausgeführt werden
0x80A8	Status Kommunikation: Fehler: *"SetFrame"* - Ungültige Framedefinition (Set Frame)
0x80A9	Status Kommunikation: Fehler: Telegrammtyp nicht vorhanden - ungültige Anfrage
0x80AA	Status Kommunikation: Fehler: Parameter - der letzte Parametersatz war ungültig
0x80AB	Fehler: Messmodul BUSY, es werden keine neuen Daten geliefert
0x80AE	Externer Fehler - Bitte kontaktieren Sie unseren Support
0x80AF	Interner Fehler: Aufgrund einer temporären Störung bei der Verarbeitung der Messdaten konnten diese nicht aktualisiert werden. Sollte dieser Fehler öfter auftreten, kontaktieren Sie bitte unsere Hotline.

13.3 Motion-Module - *"Motion Modules"*

13.3.1 Übersicht

Bausteine

Mit den nachfolgend aufgeführten Bausteinen haben Sie Zugriff auf die System SLIO Motion-Module:

- FB 320 - ACYC_RW - Azyklischer Zugriff auf System SLIO Motion-Modul
- FB 321 - ACYC_DS - Azyklische Parametrierung System SLIO Motion-Modul
- UDT 321 - ACYC_OBJECT-DATA - Datenstruktur für FB 321

Unterstützte Motion-Module

Folgende System SLIO Motion-Module werden unterstützt:

- 054-1BA00: FM 054 Motion Modul - Stepper
- 054-1CB00: FM 054 Motion Modul - 2xDC
- 054-1DA00: FM 054 Motion Modul - Pulse Train RS422

Index - Subindex

Die System SLIO Motion-Modul stellen ihre Daten wie z.B. "Profilgeschwindigkeit" über ein Objektverzeichnis zur Verfügung. In diesem Objektverzeichnis sind die Objekte organisiert und durch eine eindeutige Nummer, bestehend aus *Index* und *Subindex* adressierbar. Die Nummer wird wie folgt angegeben:

0x	Index (hexadezimal)	-	Subindex (dezimal)
Beispiel: 0x8400-03			

Zur besseren Strukturierung und Erweiterung wurde beim System SLIO Motion-Modul eine andere Objektnummerierung (Index-Vergabe) gegenüber dem Standard CiA 402 gewählt.

Index - Bereiche

Durch die Aufteilung in *Index* und *Subindex* ist eine Gruppierung möglich. Die einzelnen Bereiche sind in Gruppen zusammengehöriger Objekte gegliedert. Dieses Objektverzeichnis ist bei den System SLIO Motion Modul wie folgt strukturiert:

Index-Bereich	Inhalt
0x1000 bis 0x6FFF	Allgemeine Daten und Systemdaten
0x7000 bis 0x7FFF	Daten der digitalen Ein- und Ausgabeeinheit
ab 0x8000	Daten der Achse bzw. Antriebe

Näheres zum Aufbau des Objektverzeichnis finden Sie im Handbuch zu ihrem Motion-Modul.

Jedes Objekt verfügt über einen Subindex 0. Durch Aufruf eines Objekts mit Subindex 0 bekommen Sie die Anzahl der verfügbaren Subindizes des entsprechenden Objekts zurückgeliefert.

E/A-Adressbereich

Die Motion-Module belegen eine gewisse Anzahl an Bytes im E/A-Adressbereich.

Kopfmodul	Rückwandbus	Motion-Modul	
CPU bzw. Buskoppler	→ ←	Prozessdaten	Azyklischer Kanal

Über den *Azyklischen Kanal* können Sie azyklisch Schreib- und Lesebefehle ausführen. Hierzu wurden in den Ein-/Ausgabe-Bereich der Motion-Module Datenbereiche für die azyklische Kommunikation implementiert. Dieser Bereich umfasst 8Byte Ausgabe- und 8Byte Eingabe-Daten. Bei Einsatz der Bausteine erfolgt die Kommunikation über den *Azyklischen Kanal*.

Der Datenaustausch mit dem Motion-Modul muss über die Länge der Ein- bzw. Ausgabedaten konsistent sein! Es wird daher die Ansteuerung über das Prozessabbild empfohlen. Sie können aber auch SFC 14 und 15 für konsistentes Lesen und Schreiben der Eingabe- bzw. Ausgabe-Daten verwenden.

Beschaltung der FBs

- Bei der Projektierung ist auf eine korrekte Verschaltung der Parameter CHANNEL_IN und CHANNEL_OUT der FBs zu achten.
  - CHANNEL_IN ist auf die Eingabedaten des *Azyklischen Kanals* des Motion-Moduls zu verschalten.
  - CHANNEL_OUT auf die Ausgabedaten des *Azyklischen Kanals* des Motion-Moduls zu verschalten.

Ausgehend von der Basisadresse ist die Anfangsadresse des *Azyklischen Kanals* für die Ein- und Ausgabedaten über folgenden Offset zu erreichen:

- 054-1BA00: FM 054 - Stepper: Basis-Adresse + 26
- 054-1CB00: FM 054 - 2xDC: Basis-Adresse + 50
- 054-1DA00: FM 054 - Pulse Train RS422: Basis-Adresse + 26

Beispiel mit Basisadresse 256:

```
CHANNEL_IN   :=P#E 282.0 BYTE 10  // Basisadresse 256 + 26
CHANNEL_OUT  :=P#A 282.0 BYTE 10  // Basisadresse 256 + 26
```

Bitte beachten Sie, dass Sie eine Länge von 10Byte angeben, obwohl der Azyklische Kanal intern 8Byte verwendet!

13.3.2 FB 320 - ACYC_RW - Azyklischer Zugriff auf System SLIO Motion-Modul

Beschreibung

Mit diesem Baustein können Sie aus Ihrem Anwenderprogramm auf das Objektverzeichnis der System SLIO Motion-Module zugreifen. Hierbei verwendet der Baustein einen azyklischen Kommunikationskanal, auf Basis einer Anfrage-/Antwort-Sequenz. Dieser ist Bestandteil des Ein-/Ausgabereichs des Motion-Moduls.

Folgende System SLIO Motion-Module werden unterstützt:

- 054-1BA00: FM 054 - Stepper
- 054-1CB00: FM 054 - 2xDC
- 054-1DA00: FM 054 - Pulse Train RS422

Da der FB 321 intern den FB 320 aufruft und beide Bausteine auf die gleiche Datenbasis zugreifen, dürfen Sie je Antriebskanal (sofern mehrkanalig) nur einen dieser Bausteine in Ihrem Anwenderprogramm verwenden! Auch darf dieser Baustein nur einmal pro Zyklus aufgerufen werden!

Der Datenaustausch mit dem Motion-Modul muss über die Länge der Ein- bzw. Ausgabedaten konsistent sein! Es wird daher die Ansteuerung über das Prozessabbild empfohlen. Sie können aber auch SFC 14 und 15 für konsistentes Lesen und Schreiben der Eingabe- bzw. Ausgabe-Daten verwenden.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
REQUEST	IN	BOOL	Mit Flanke 0-1 wird die Auftragsbearbeitung gestartet.
MODE	IN	BYTE	Geben Sie hier 0x01 für das azyklische Protokoll an
COMMAND	IN	BYTE	0x11 = Lesen eines Datenobjekts (max. 4Byte) 0x21 = Schreiben eines Datenobjekts (max. 4Byte)
INDEX	IN	WORD	Index des Objekts im Objektverzeichnis - siehe Handbuch zum System SLIO Motion-Modul.
SUBINDEX	IN	BYTE	Subindex des Objekts im Objektverzeichnis - siehe Handbuch zum System SLIO Motion-Modul.
WRITE_LENGTH	IN	DINT	Länge der zu schreibenden Daten in Byte (max. 4Byte)
WRITE_DATA	IN	ANY	Zeiger auf die zu schreibenden Daten.
READ_DATA	IN	ANY	Zeiger auf die gelesenen Daten.
CHANNEL_IN	IN	ANY	Zeiger auf den Beginn des azyklischen Kanals im Eingabe-Bereich des Motion-Moduls. Tragen Sie als Länge 10Byte ein. Beispiele: P#E100.0 BYTE 10 oder P#DB10.DBX0.0 BYTE 10
CHANNEL_OUT	IN	ANY	Zeiger auf den Beginn des azyklischen Kanal im Ausgabe-Bereich des Motion-Moduls. Tragen Sie als Länge 10Byte ein. Beispiele: P#A100.0 BYTE 10 oder P#DB10.DBX10.0 BYTE 10
READ_LENGTH	OUT	DINT	Länge der empfangenen Daten in Byte. Dieser Wert ist auf ein Vielfaches von 4 aufzurunden, da die Längenangabe nicht übertragen wird.
DONE	OUT	BOOL	1: Auftrag wurde fehlerfrei ausgeführt
BUSY	OUT	BOOL	0: Kein Auftrag in Bearbeitung 1: Auftrag wird bearbeitet
ERROR	OUT	BOOL	0: Kein Fehler 1: Fehler aufgetreten. Die Fehlerursache wird über den Parameter *ERROR_ID* angezeigt
ERROR_ID	OUT	WORD	Detaillierte Fehlerinformationen

Bitte beachten Sie, dass bei den Parametern WRITE_DATA und READ_DATA keine Überprüfung auf Datentyp und Länge stattfindet!

Verhalten der Bausteinparameter

- Ausschließlichkeit der Ausgänge:
  - Die Ausgänge *BUSY*, *DONE* und *ERROR* schließen sich gegenseitig aus. Es kann immer nur einer dieser Ausgänge zur gleichen Zeit TRUE sein.
  - Sobald der Eingang *REQUEST* TRUE wird, muss einer der Ausgänge TRUE werden.
- Ausgangs-Zustand
  - Die Ausgänge *DONE, ERROR, ERROR_ID* und *READ_LENGTH* werden mit einer Flanke 1-0 am Eingang *REQUEST* zurückgesetzt, wenn der Funktionsbaustein nicht aktiv ist (*BUSY* = FALSE).
  - Eine Flanke 1-0 an *REQUEST* beeinflusst die Auftragsbearbeitung nicht.
  - Falls *REQUEST* bereits während der Auftragsbearbeitung zurückgesetzt wird, so ist sichergestellt, dass einer der Ausgänge am Ende des Auftrags für einen SPS-Zyklus gesetzt wird. Erst danach werden die Ausgänge zurückgesetzt.
- Eingangs-Parameter
  - Die Eingangs-Parameter werden mit Flanke 0-1 an *REQUEST* übernommen. Zur Änderung von Parametern, müssen Sie den Auftrag neu triggern.
  - Tritt während der Auftragsbearbeitung erneut eine Flanke 0-1 an *REQUEST* auf, wird ein Fehler ausgegeben, kein neues Kommando aktiviert und die Antwort vom laufenden Kommando verworfen!
- Fehlerbehandlung
  - Der Baustein besitzt 2 Fehlerausgänge zur Anzeige von Fehlern während der Auftragsbearbeitung. *ERROR* zeigt den Fehler an und *ERROR_ID* gibt eine ergänzende Fehlernummer aus.
  - Die Ausgänge *DONE* und *READ_LENGTH* bezeichnen eine erfolgreiche Auftragsbearbeitung und werden nicht gesetzt, wenn *ERROR* TRUE wird.
- Verhalten des *DONE* Ausgangs
  - Der *DONE* Ausgang wird gesetzt, wenn ein Auftrag erfolgreich ausgeführt wurde.
- Verhalten des *BUSY* Ausgangs
  - Der *BUSY* Ausgang zeigt an, dass der Funktionsbaustein aktiv ist.
  - *BUSY* wird sofort mit der Flanke 0-1 an *REQUEST* gesetzt und wird erst zurückgesetzt, wenn der Auftrag erfolgreich oder auch nicht erfolgreich beendet wurde.
  - Solange *BUSY* TRUE ist, muss der Baustein zyklisch aufgerufen werden um das Kommando ausführen zu können.

Tritt während der Auftragsbearbeitung erneut eine Flanke 0-1 an REQUEST auf, wird ein Fehler ausgegeben, kein neues Kommando aktiviert und die Antwort vom laufenden Kommando verworfen!

ERROR_ID

ERROR_ID	Beschreibung
0x0000	Kein Fehler vorhanden
0x8070	Fehlerhafter Parameter *MODE*
0x8071	Fehlerhafter Parameter *COMMAND*
0x8072	Parameter *WRITE_LENGTH* überschreitet die maximal Größe
0x8073	Parameter *CHANNEL_IN* passt nicht zum Parameter *MODE*

ERROR_ID	Beschreibung
0x8074	Parameter *CHANNEL_OUT* passt nicht zum Parameter *MODE*
0x8075	Nicht zulässiger Befehl (Flanke 0-1 bei *REQUEST* während Auftrag ausgeführt wird)
0x8081	Fehler - Lesezugriff - Daten nicht vorhanden Befehl wurde nicht ausgeführt!
0x8091	Fehler - Schreibzugriff - Daten nicht vorhanden Befehl wurde nicht ausgeführt!
0x8092	Fehler - Schreibzugriff - Datenbereich überschritten Befehl wurde nicht ausgeführt!
0x8093	Fehler - Schreibzugriff - Daten können nur gelesen werden Befehl wurde nicht ausgeführt!
0x8094	Fehler - Schreibzugriff - Daten sind schreibgeschützt Befehl wurde nicht ausgeführt!
0x8099	Fehler während der azyklischen Kommunikation Befehl wurde nicht ausgeführt!

Programm-Struktur

Ist kein Auftrag aktiv, so sind alle Ausgabe-Parameter auf 0 zu setzen. Mit einer Flanke 0-1 an *REQUEST* aktivieren Sie nach folgender Vorgehensweise einen Auftrag:

1. Überprüfen Sie, ob bereits ein Auftrag aktiv ist, ggf. Auftrag abbrechen und Fehler ausgeben.
   ⇨ Abfragen auf DONE = 1 bzw. BUSY = 0
2. Beschalten Sie die Eingabeparameter:
   - MODE
   - COMMAND
   - WRITE_LENGTH
   - CHANNEL_IN
   - CHANNEL_OUT

   ⇨ Im Fehlerfall Auftrag abrechen, ansonsten weiter mit Schritt 3.
3. Eingangsparameter intern speichern.
4. Das gewünschte Kommando ausführen und warten bis dieses ausgeführt wurde.
5. Ergebnis der Kommandoausführung intern speichern und ausgeben.
6. Alle Ausgabeparameter wieder auf 0 setzen.

13.3.3 FB 321 - ACYC_DS - Azyklische Parametrierung System SLIO Motion-Modul

Beschreibung

Mit diesem Baustein können Sie aus Ihrem Anwenderprogramm Ihr Motion-Modul parametrieren. Hierbei können Sie in einem Datenbaustein Ihre Parameter in Form einer *Objektliste* ablegen und diese über den azyklischen Kommunikationskanal in Ihr Motion-Modul übertragen.

Folgende System SLIO Module werden unterstützt:

- 054-1BA00: FM 054 Motion Modul - Stepper
- 054-1CB00: FM 054 Motion Modul - 2xDC
- 054-1DA00: FM 054 Motion Modul - Pulse Train RS422

Da der FB 321 intern den FB 320 aufruft und beide Bausteine auf die gleiche Datenbasis zugreifen, dürfen Sie je Antriebskanal (sofern mehrkanalig) nur einen dieser Bausteine in Ihrem Anwenderprogramm verwenden! Auch darf dieser Baustein nur einmal pro Zyklus aufgerufen werden!

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
REQUEST	IN	BOOL	Mit Flanke 0-1 wird die Auftragsbearbeitung gestartet.
MODE	IN	BYTE	Geben Sie hier 0x01 für das azyklische Protokoll an.
READ_BACK	IN	BOOL	0: Geschriebene Objekte werden nicht zurückgelesen. 1: Geschrieben Objekte werden direkt nach dem Schreibvorgang zurückgelesen und einem Vergleich unterzogen.
GROUP	IN	WORD	0x01...0x7F: Selektion einer Gruppe in der Objektliste. 0xFF: Selektion aller Objekte in der Objektliste.
OBJECT_DATA	IN	ANY	Zeiger auf die UDT. *Kap. 13.3.4 "UDT 321 - ACYC_OBJECT-DATA - Datenstruktur für FB 321" Seite 474*
CHANNEL_IN	IN	ANY	Zeiger auf den Beginn des azyklischen Kanals im Eingabe-Bereich des Motion-Moduls. Tragen Sie als Länge 10Byte ein. Beispiele: P#E100.0 BYTE 10 oder P#DB10.DBX0.0 BYTE 10
CHANNEL_OUT	IN	ANY	Zeiger auf den Beginn des azyklischen Kanal im Ausgabe-Bereich des Motion-Moduls. Tragen Sie als Länge 10Byte ein. Beispiele: P#A100.0 BYTE 10 oder P#DB10.DBX10.0 BYTE 10
DONE	OUT	BOOL	1: Auftrag wurde fehlerfrei ausgeführt.
BUSY	OUT	BOOL	0: Kein Auftrag in Bearbeitung. 1: Auftrag wird bearbeitet.
DATASET_INDEX	OUT	INT	Objekt, welches aktuell bearbeitet wird.
ERROR	OUT	BOOL	0: Kein Fehler 1: Fehler aufgetreten. Die Fehlerursache wird über den Parameter *ERROR_ID* angezeigt.
ERROR_ID	OUT	WORD	Detaillierte Fehlerinformationen

Verhalten der Bausteinparameter

- Ausschließlichkeit der Ausgänge:
  - Die Ausgänge *BUSY*, *DONE* und *ERROR* schließen sich gegenseitig aus. Es kann immer nur einer dieser Ausgänge zur gleichen Zeit TRUE sein.
  - Sobald der Eingang *REQUEST* TRUE wird, muss einer der Ausgänge TRUE werden.
- Ausgangs-Zustand
  - Nach Ende der Abarbeitung der Objektliste werden mit einer Flanke 1-0 an *REQUEST* die Ausgänge *DONE, ERROR, ERROR_ID* und *DATASET_INDEX* zurückgesetzt.
  - Falls *REQUEST* bereits während der Abarbeitung der Objektliste zurückgesetzt wird, so ist sichergestellt, dass die ganze Objektliste abgearbeitet wird.
  - Am Ende der Abarbeitung wird bei fehlerfreier Ausführung *DONE* für einen SPS-Zyklus gesetzt wird. Erst danach werden die Ausgänge zurückgesetzt.
- Eingangs-Parameter
  - Die Eingangs-Parameter werden mit Flanke 0-1 an *REQUEST* übernommen. Zur Änderung von Parametern, müssen Sie den Auftrag neu triggern.
  - Tritt während der Auftragsbearbeitung erneut eine Flanke 0-1 an *REQUEST* auf, wird ein Fehler ausgegeben (unzulässige Kommandofolge) und die Abarbeitung der Objektliste beendet.
- Eingangs-Parameter *READ_BACK*
  - Bei aktiviertem Parameter *READ_BACK* werden zu schreibende Objekte unmittelbar nach dem Schreibauftrag des Objekts mit einem Leseauftrag zurückgelesen.
  - Der geschriebene und zurückgelesene Wert werden einem Vergleich unterzogen.

    Bei Gleichheit wird das nächste Objekt bearbeitet.

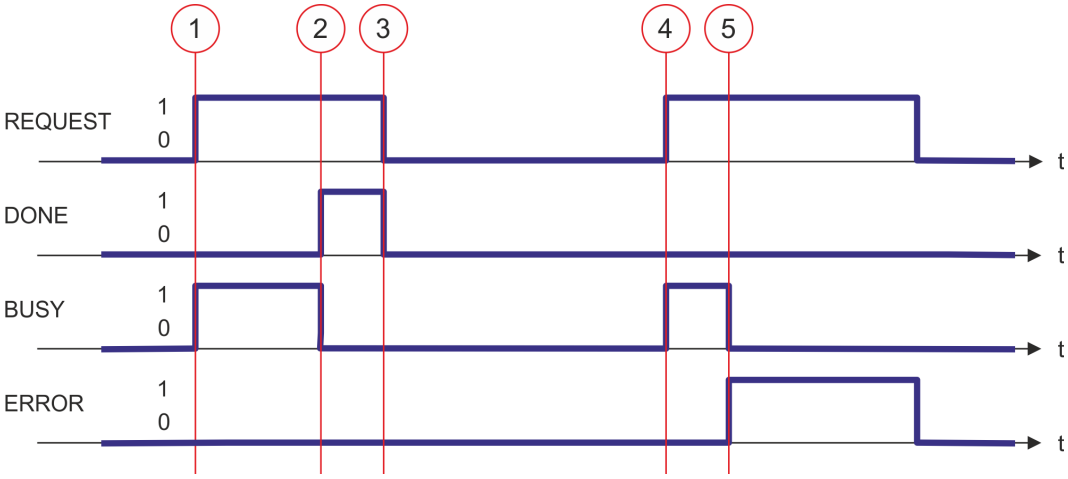
    Bei Ungleichheit erfolgt eine Fehlermeldung (*ERROR_ID* = 0x8079) und die Abarbeitung der Objektliste wird beendet.
- Eingangs-Parameter *GROUP*
  - In der Objektliste können Sie zur besseren Strukturierung jedem Objekt eine Gruppe zuordnen.
  - Über *GROUP* definieren Sie die Gruppe, deren Parameter zu übertragen sind.

    0x01...0x7F: Objekte der gewählten Gruppe übertragen.

    0xFF: Objekte aller Gruppen übertragen.
- Fehlerbehandlung
  - Der Baustein besitzt Fehlerausgänge zur Anzeige von Fehlern während der Auftragsbearbeitung. *ERROR* zeigt den Fehler an, *ERROR_ID* gibt eine ergänzende Fehlernummer aus und *DATASET_INDEX* zeigt an, bei welchem Objekt der Fehler aufgetreten ist.
  - Der Ausgang *DONE* bezeichnet eine erfolgreiche Auftragsbearbeitung und wird nicht gesetzt, wenn *ERROR* TRUE wird.
- Verhalten des *DONE* Ausgangs
  - Der *DONE* Ausgang wird gesetzt, wenn ein Auftrag erfolgreich ausgeführt wurde.
- Verhalten des *BUSY* Ausgangs
  - Der *BUSY* Ausgang zeigt an, dass der Funktionsbaustein aktiv ist.
  - *BUSY* wird sofort mit der Flanke 0-1 an *REQUEST* gesetzt und wird erst zurückgesetzt, wenn der Auftrag erfolgreich oder auch nicht erfolgreich beendet wurde.
  - Solange *BUSY* TRUE ist, muss der Baustein zyklisch aufgerufen werden, um das Kommando ausführen zu können.
- Verhalten des *DATASET_INDEX* Ausgangs
  - Der *DATASET_INDEX* Ausgang zeigt an, welches Objekt der Objektliste aktuell bearbeitet wird.
  - Ist kein Auftrag aktiv, wird *DATASET_INDEX* = 0 geliefert.
  - Tritt während der Objektbearbeitung ein Fehler auf, zeigt *DATASET_INDEX* das fehlerverursachende Objekt an.

Tritt während der Auftragsbearbeitung erneut eine Flanke 0-1 an REQUEST auf, wird ein Fehler (ERROR_ID = 0x8075) ausgegeben, kein neues Kommando aktiviert und die Antwort vom laufenden Kommando verworfen!

Zustandsdiagramm



(1) Mit Flanke 0-1 an *REQUEST* zum Zeitpunkt (1) wird die Abarbeitung gestartet und *BUSY* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist die Abarbeitung abgeschlossen. *BUSY* liefert den Wert FALSE und *DONE* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist die Abarbeitung abgeschlossen und *REQUEST* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.
(4) Zum Zeitpunkt (4) wird erneut die Abarbeitung mit einer Flanke 0-1 an *REQUEST* gestartet und *BUSY* liefert den Wert TRUE.
(5) Zum Zeitpunkt (5) tritt ein Fehler bei der Abarbeitung auf. *BUSY* liefert den Wert FALSE und *ERROR* den Wert TRUE.

ERROR_ID

ERROR_ID	Beschreibung
0x0000	Kein Fehler vorhanden
0x8070	Fehlerhafter Parameter *MODE*
0x8071	Fehlerhafter Parameter *OBJECT_DATA*
0x8075	Nicht zulässiger Befehl (Flanke 0-1 bei *REQUEST* während Auftrag ausgeführt wird)
0x8078	Fehlerhafter Parameter *GROUP*
0x8079	*READ_BACK* erkennt einen Fehler (geschriebener und gelesener Wert ungleich)
0x807A	Zeiger bei *OBJECT_DATA* ungültig

Innerhalb des Funktionsbausteins erfolgt ein Aufruf des FB 320. Hierbei werden eventuelle Fehler des FB 320 an den FB 321 durchgereicht.
"ERROR_ID" Seite 469

13.3.4 UDT 321 - ACYC_OBJECT-DATA - Datenstruktur für FB 321

Datenstruktur für die Objektliste

Die Parameter sind in einem Datenbaustein als *Objektliste* abzulegen, welche aus einzelnen *Objekten* besteht. Die Struktur eines *Objekts* ist über eine UDT definiert.

Struktur eines Objekts

Variable	Deklaration	Datentyp	Beschreibung
Group	IN	WORD	0 < *Group* < 0x80 zulässig
Command	IN	BYTE	0x11 = Lesen aus dem Objektverzeichnis 0x21 = Schreiben in das Objektverzeichnis
Index	IN	WORD	Index des Objekts
Subindex	IN	BYTE	Subindex des Objekts
Write_Length	IN	BYTE	Länge der zu schreibenden Daten in Byte
Data_Write	IN	DWORD	Zu schreibende Daten.
Data_Read	OUT	DWORD	Gelesene Daten
State	OUT	BYTE	0x00 = nie bearbeitet 0x01 = *BUSY* - in Bearbeitung 0x02 = *DONE* - erfolgreich bearbeitet 0x80 = *ERROR* - bei der Bearbeitung ist ein Fehler aufgetreten

Bitte beachten Sie, dass Sie bei einem Schreibauftrag immer die zum Objekt passende Länge angeben!

Beispiel-DB

Adr.	Name	Typ	Anfangswert	Aktualwert	Kommentar
0.0	Object(1).Group	WORD			1. Objekt
2.0	Object(1).Command	BYTE			
4.0	Object(1).Index	WORD			
6.0	Object(1).Subindex	BYTE			
7.0	Object(1).Write_Length	BYTE			
8.0	Object(1).Data_Write	DWORD			
12.0	Object(1).Data_Read	DWORD			
16.0	Object(1).State	BYTE			
18.0	Object(2).Group	WORD			2. Objekt
...	...	...			
34.0	Object(2).State	BYTE			
36.0	Object(3).Group	WORD			3. Objekt
...	...	...			

Adr.	Name	Typ	Anfangswert	Aktualwert	Kommentar
52.0	Object(3).State	BYTE			
...	...	...			...

13.4 RAM nach WLD - *"WLD"*

13.4.1 FB 240 - RAM_to_s7prog.wld - RAM nach s7prog.wld

Beschreibung

Mit *REQ* = TRUE kopiert dieser Baustein das aktuell geladenen Projekt im RAM einer CPU auf eine gesteckte Speicherkarte als s7prog.wld. Bei einer VIPA SPEED7-CPU wird die s7prog.wld immer automatisch nach Urlöschen von einer gesteckten Speicherkarte gelesen. Der FB 240 ruft intern den Baustein SFB 239 mit den entsprechenden Parametern auf. Hierbei werden die Werte für *BUSY* und *RET_VAL* vom SFB 239 an den FB 240 zurückgegeben.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	IN	BOOL	E, A, M, D, L	Funktionsanstoß mit *REQ* = 1
BUSY	OUT	BOOL	E, A, M, D, L	Rückgabewert des SFB 239
RET_VAL	OUT	WORD	E, A, M, D, L	Rückgabewert des SFB 239

13.4.2 FB 241 - RAM_to_autoload.wld - RAM nach autoload.wld

Beschreibung

Mit *REQ* = TRUE kopiert dieser Baustein das aktuell geladenen Projekt im RAM einer CPU auf eine gesteckte Speicherkarte als autoload.wld. Bei einer VIPA SPEED7-CPU wird die autoload.wld immer automatisch nach NetzEIN von einer gesteckten Speicherkarte gelesen. Der FB 241 ruft intern den Baustein SFB 239 mit den entsprechenden Parametern auf. Hierbei werden die Werte für *BUSY* und *RET_VAL* vom SFB 239 an den FB 241 zurückgegeben.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	IN	BOOL	E, A, M, D, L	Funktionsanstoß mit *REQ* = 1
BUSY	OUT	BOOL	E, A, M, D, L	Rückgabewert des SFB 239
RET_VAL	OUT	WORD	E, A, M, D, L	Rückgabewert des SFB 239

14 Antriebssteuerung - *"Motion Control"*

14.1 Übersicht

14.1.1 Funktionsbausteine

Baustein	Aufruf im OB 1	Aufruf im OB 61	Positionierungsachse	Drehzahlachse	Externer Geber	Virtuelle Achse	Siehe Seite
FB 700 - MC_Power Achse freigeben bzw. sperren	Ja	Nein	Ja	Ja	Ja	Ja	*486*
FB 701 - MC_Home Achse referenzieren	Ja	Nein	Ja	Ja	ja	Ja	*487*
FB 702 - MC_Stop Achse stoppen	Ja	Ja[2]	Ja	Ja	Nein	Ja	*489*
FB 703 - MC_Halt Achse anhalten	Ja	Ja[2]	Ja	Ja	Nein	Ja	*491*
FB 704 - MC_MoveRelative Achse relativ verfahren	Ja	Ja[2]	Ja	Nein	Nein	Ja	*493*
FB 705 - MC_MoveVelocity Achse verfahren mit konstanter Geschwindigkeit	Ja	Ja[2]	Ja	Ja	Nein	Ja	*495*
FB 706 - MC_CamIn Kopplung Master-Slave-Achse über Kurvenscheibe	Nein	Ja[1]	Ja	Nein	Nein	Ja	*549*
FB 707 - MC_GearIn Kopplung Master-Slave-Achse über Geschwindigkeit	Nein	Ja[1]	Ja	Nein	Nein	Ja	*559*
FB 708 - MC_MoveAbsolute Achse auf absolute Position verfahren	Ja	Ja[2]	Ja	Nein	Nein	Ja	*497*
FB 709 - MC_CamOut Deaktivierung Master-Slave-Kopplung über Kurvenscheibe	Ja	Ja[2]	Ja	Nein	Nein	Ja	*562*
FB 710 - MC_GearOut Deaktivierung Master-Slave-Kopplung über Geschwindigkeit	Ja	Ja[2]	Ja	Nein	Nein	Ja	*564*
FB 711 - MC_Reset Achse zurücksetzen (Neuinitialisierung)	Ja	Nein	Ja	Ja	Nein	Ja	*500*
FB 712 - MC_ReadStatus PLCopen-State der Achse lesen	Ja	Nein	Ja	Ja	Ja	Ja	*502*
FB 713 - MC_ReadAxisError Fehler Achse lesen	Ja	Nein	Ja	Ja	Nein	Ja	*504*
FB 714 - MC_ReadParameter Parameter Achse lesen	Ja	Nein	Ja	Ja	Ja	Ja	*506*
FB 715 - MC_WriteParameter Parameter Achse schreiben	Ja	Nein	Ja	Ja	Ja	Ja	*507*

Baustein	Aufruf im OB 1	Aufruf im OB 61	Positionierungsachse	Drehzahlachse	Externer Geber	Virtuelle Achse	Siehe Seite
FB 716 - MC_ReadActualPosition Position Achse auslesen	Ja	Ja[2]	Ja	Nein	Ja	Ja	*509*
FB 717 - MC_ReadActualVelocity Geschwindigkeit der Achse lesen	Ja	Ja[2]	Ja	Ja	Ja	Ja	*510*
FB 718 - MC_ReadAxisInfo Zusatzinformationen der Achse lesen	Ja	Nein	Ja	Ja	Ja	Ja	*512*
FB 719 - MC_ReadMotionState Zustand Bewegungsauftrag lesen	Ja	Nein	Ja	Ja	Ja	Ja	*513*
FB 720 - MC_PhasingAbsolute Phasenverschiebung absolut	Nein	Ja[1]	Ja	Nein	Nein	Ja	*566*
FB 721 - MC_PhasingRelative Phasenverschiebung relativ	Nein	Ja[1]	Ja	Nein	Nein	Ja	*568*
FB 722 - MC_MoveSuperimposed Überlagerte Positionierung	Nein	Ja[1]	Ja	Nein	Nein	Ja	*515*
FB 723 - MC_TouchProbe Achsposition erfassen	Nein	Ja[1]	Ja	Nein	Nein	Nein	*517*
FB 724 - MC_AbortTrigger Achsposition erfassen abbrechen	Ja	Ja	Ja	Nein	Nein	Nein	*518*
FB 725 - MC_ReadBoolParameter Parameter-Boolean Achse lesen	Ja	Nein	Ja	Ja	Ja	Ja	*519*
FB 726 - MC_WriteBoolParameter Parameter-Boolean Achse schreiben	Ja	Nein	Ja	Ja	Ja	Ja	*521*
FB 727 - VMC_ReadDWordParameter Parameter-Doppelwort Achse lesen	Ja	Nein	Ja	Ja	Ja	Ja	*523*
FB 728 - VMC_WriteDWordParameter Parameter-Doppelwort Achse schreiben	Ja	Nein	Ja	Ja	Ja	Ja	*525*
FB 729 - VMC_ReadWordParameter Parameter-Wort Achse lesen	Ja	Nein	Ja	Ja	Ja	Ja	*527*
FB 730 - VMC_WriteWordParameter Parameter-Wort Achse schreiben	Ja	Nein	Ja	Ja	Ja	Ja	*529*
FB 731 - VMC_ReadByteParameter Parameter-Byte Achse lesen	Ja	Nein	Ja	Ja	Ja	Ja	*531*
FB 732 - VMC_WriteByteParameter Parameter-Byte Achse schreiben	Ja	Nein	Ja	Ja	Ja	Ja	*533*
FB 733 - VMC_ReadDriveParameter Antriebsparameter lesen	Ja	Nein	Ja	Ja	Nein	Nein	*535*
FB 734 - VMC_WriteDriveParameter Antriebsparameter schreiben	Ja	Nein	Ja	Ja	Nein	Nein	*537*

Baustein	Aufruf im OB 1	Aufruf im OB 61	Positionierungsachse	Drehzahlachse	Externer Geber	Virtuelle Achse	Siehe Seite
FB 735 - VMC_HomeInit_LimitSwitch Initialisierung Referenzfahrt auf Endschalter	Ja	Nein	Ja	Nein	Nein	Nein	*539*
FB 736 - VMC_HomeInit_HomeSwitch Initialisierung Referenzfahrt auf Referenzschalter	Ja	Nein	Ja	Nein	Nein	Nein	*541*
FB 737 - VMC_HomeInit_ZeroPulse Initialisierung Referenzfahrt auf Nullimpuls	Ja	Nein	Ja	Nein	Nein	Nein	*543*
FB 738 - VMC_HomeInit_SetPosition Initialisierung Referenzfahrt setze Position	Ja	Nein	Ja	Nein	Nein	Nein	*544*
FB 739 - MC_TorqueControl Drehmomentsteuerung	Ja	Ja	Ja	Nein	Nein	Nein	*545*
FB 740 - MC_ReadActualTorque - Drehmoment Achse auslesen	Ja	Ja	Ja	Nein	Nein	Nein	*547*

1): Diese Bausteine müssen zwingend im OB61 aufgerufen werden, um eine korrekte funktionsweise zu gewährleisten.

2): Es wird geraten die Bausteine im OB1 zu verwenden. In zeitkritischen Spezialfällen (Kommandos müssen zeitgleich gestartet werden) können die Bausteine auch im OB61 aufgerufen werden, was jedoch zu Lasten der Performance geht.

14.1.2 Funktionsweise

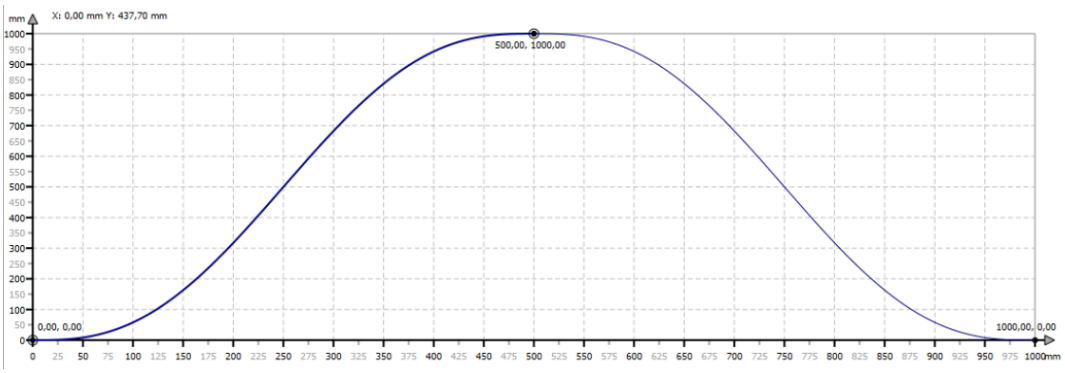
Einsatzgebiet

Die hier aufgeführten Motion-Bausteine eignen sich speziell für Bewegungsaufgaben mit komplexer Bewegungsführung an Produktionsmaschinen. Hiermit können unter Einsatz elektronischer Kurvenscheiben folgende Bewegungen realisiert werden:

- Absolute Zielvorgabe
- Relative Verfahrensstrecken
- Endlosfahrten
- Synchrone Bewegungen

Begriffserklärung

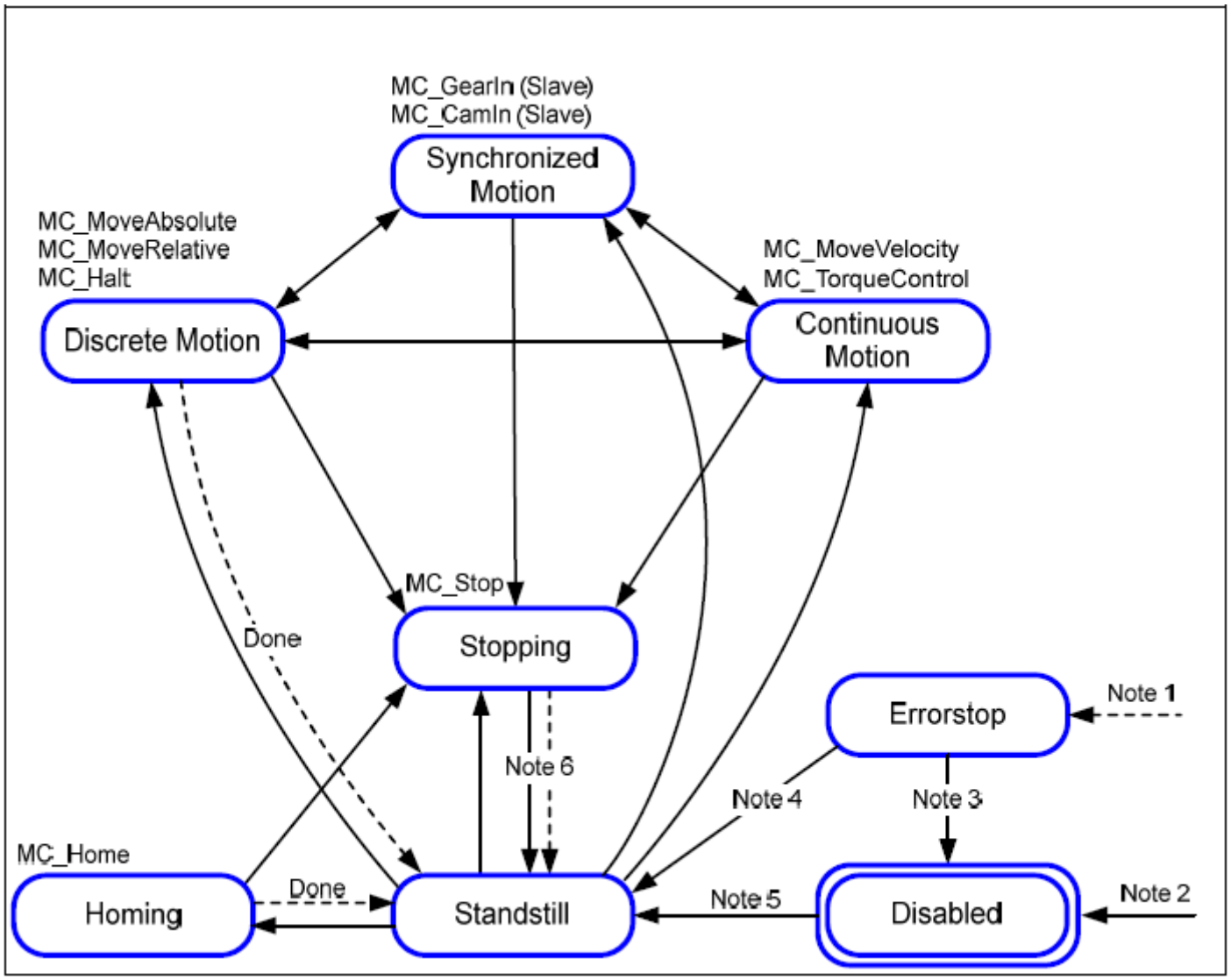
- Zustandsdiagramm
  - Das Zustandsdiagramm ist eine Abstraktionsebene, die den realen Zustand der Achse widerspiegelt.
  - Alle Kommandos arbeiten in diesem Achs-Zustandsdiagramm. Die Achse befindet sich immer in einem der definierten Zustände.
  - Der Zustand der Achse wechselt sofort mit dem auslösenden Kommando.
- Bewegungskommando
  - Bewegungskommandos werden immer sequentiell abgearbeitet.
  - Jedes Bewegungskommando, das eine Transition verursacht, ändert den Zustand der Achse und ändert als Folge daraus, das Bewegungsprofil.
- Kurvenscheibe
  - Eine elektronische Kurvenscheibe stellt eine eindeutige Zuordnung zwischen einer Position der Master-Achse zu einer Position der Slave-Achse her.
  - Die Zuordnung erfolgt über ein 2-dimensionales Koordinatensystem. Hierbei bildet die Horizontale die Position der Master-Achse und die Vertikale die Position der Slave-Achse ab.
  - Der Positionsumfang der Horizontalen entspricht der Masterumlauflänge. Der Positionsumfang der Vertikalen entspricht der Slaveumlauflänge. Eine Gerade mit der Steigung 1 entspricht somit einer 1:1-Kopplung von Master- zu Slave-Achse.
  - Bei einer periodischen Bearbeitung wird ein Kurvenzug wiederholt abgearbeitet.



Kurvenscheibe

14.1.3 Zustände

Zustandsdiagramm



Note 1 Aus jedem Status: An der Achse ist ein Fehler aufgetreten

Note 2 Aus jedem Status: MC_Power.Enable = FALSE und es gibt keinen Fehler an der Achse

Note 3 MC_Reset und MC_Power.Status = FALSE

Note 4 MC_Reset und MC_Power.Status = TRUE und MC_Power.Enable = TRUE

Note 5 MC_Power.Enable = TRUE und MC_Power.Status = TRUE

Note 6 MC_Stop.Done = TRUE und MC_Stop.Execute = FALSE

- Der Zustand *Disabled* beschreibt den Grundzustand einer Achse. In diesem Zustand kann die Achse durch keinen Funktionsbaustein bewegt werden. Nach dem Aufruf des MC_Power Bausteins mit Enable = TRUE wechselt die Achse in den Zustand *Standstill* oder im Fehlerfall in den Zustand *ErrorStop*.
- Wenn MC_Power mit Enable = FALSE aufgerufen wird, wechselt der Zustand nach *Disabled*. Der Zweck des Zustands *ErrorStop* ist die Achse zu stoppen und anschließend keine weiteren Kommandos anzunehmen, bis ein Reset ausgelöst wurde.
- Der Zustandsübergang *Error* bezieht sich nur auf tatsächliche Achsfehler und nicht auf Ausführungsfehler eines Funktionsbausteins. Achsfehler können aber auch am Fehlerausgang eines Funktionsbausteins angezeigt werden.

- Funktionsbausteine, die im Zustandsdiagramm nicht aufgeführt werden, beeinflussen den Zustand der Achse nicht.
- Der Zustand *Stopping* zeigt an, dass sich die Achse in einer Stopprampe befindet. Der Zustand wechselt nach dem vollständigen Stopp und Aufruf von MC_Stop mit Execute = FALSE nach *StandStill*.

14.1.4 Verhalten der Ein- und Ausgänge

Ausschließlichkeit der Ausgänge

- Die Ausgänge *Busy*, *Done*, *Error* und *CommandAborted* schließen sich gegenseitig aus, es kann also an einem Funktionsbaustein nur einer dieser Ausgänge zu einer Zeit TRUE sein.
- Sobald der Eingang *Execute* TRUE wird, muss einer der Ausgänge TRUE werden. Ebenfalls kann nur einer der Ausgänge *Active*, *Error*, *Done* und *CommandAborted* zu einer Zeit TRUE sein.

Ausgangs-Zustand

- Die Ausgänge *Done*, *InGear*, *InSync*, *InVelocity*, *Error*, *ErrorID* und *CommandAborted* werden mit einer Flanke 1-0 am Eingang *Execute* zurückgesetzt, wenn der Funktionsbaustein nicht aktiv ist (*Busy* = FALSE).
- Die Kommandoausführung wird durch eine Flanke 1-0 an *Execute* nicht beeinflusst.
- Falls *Execute* bereits während der Kommandoausführung zurückgesetzt wird, so ist sichergestellt, dass einer der Ausgänge am Ende des Kommandos für einen SPS-Zyklus gesetzt wird. Erst danach werden die Ausgänge zurückgesetzt.

Eingangs-Parameter

- Die Eingangs-Parameter werden mit Flanke 0-1 an *Execute* übernommen.
- Zur Änderung der Parameter muss das Kommando neu getriggert werden.
- Falls ein Eingangs-Parameter nicht an den Funktionsbaustein übergeben wird, so bleibt der zuletzt an diesen Baustein übergebene Wert gültig.
- Beim ersten Aufruf muss ein sinnvoller Default-Wert übergeben werden.

Position und Distanz

- Der Eingang *Position* bezeichnet einen definierten Wert innerhalb eines Koordinatensystems.
- *Distance* bezeichnet ein relatives Maß als Abstand zweier Positionen.
- Sowohl *Position*, als auch *Distance* werden in technischen Einheiten, z.B. [mm] oder [°], entsprechend der Skalierung der Achse angegeben.

Parameter für das dynamische Verhalten

- Die Dynamikparameter für *Move*-Funktionen werden in technischen Einheiten mit der Zeitbasis Sekunde angegeben.
  Ist eine Achse beispielsweise in Millimetern skaliert, so sind die Einheiten für *Velocity* [mm/s], *Acceleration* [mm/s^2], *Deceleration* [mm/s^2] und *Jerk* [mm/s^3].

Fehlerbehandlung

- Alle Funktionsbausteine haben zwei Fehlerausgänge um Fehler während der Kommandoausführung anzuzeigen.
- *Error* zeigt den Fehler an und *ErrorID* gibt eine ergänzende Fehlernummer aus.
- Die Ausgänge *Done*, *InVelocity*, *InGear* und *InSync* bezeichnen eine erfolgreiche Kommandoausführung und werden nicht gesetzt, wenn *Error* TRUE wird.

Fehlertypen

- Funktionsbausteinfehler
  - Funktionsbausteinfehler sind Fehler, die ausschließlich den Funktionsbaustein und nicht die Achse betreffen wie z.B. fehlerhafte Parametrierung.
  - Funktionsbausteinfehler müssen nicht explizit zurückgesetzt werden, sondern werden selbständig zurückgesetzt, wenn der Eingang *Execute* zurückgesetzt wird.
- Kommunikationsfehler
  - Kommunikationsfehler wie z.B. der Funktionsbaustein kann die Achse nicht adressieren.
  - Kommunikationsfehler deuten oft auf eine fehlerhafte Konfiguration oder Parametrierung hin.
  - Ein Reset ist nicht möglich, sondern der Funktionsbaustein kann neu getriggert werden, nachdem die Konfiguration korrigiert wurde.
- Achsfehler
  - Achsfehler treten üblicherweise während der Fahrt auf wie z.B. Schleppabstandsfehler.
  - Ein Achsfehler muss durch MC_Reset zurückgesetzt werden.

Verhalten des *Done*-Ausgangs

- Der *Done*-Ausgang wird gesetzt, wenn ein Kommando erfolgreich ausgeführt wurde.
- Wenn mit mehreren Funktionsbausteinen an einer Achse gearbeitet wird und das laufende Kommando durch einen weiteren Baustein unterbrochen wird, so wird der *Done*-Ausgang des ersten Bausteins nicht gesetzt.

Verhalten des *CommandAborted*-Ausgangs

- *CommandAborted* wird gesetzt, wenn ein Kommando durch einen anderen Baustein unterbrochen wird.

Verhalten des *Busy*-Ausgangs

- Der *Busy*-Ausgang zeigt an, dass der Funktionsbaustein aktiv ist.
- *Busy* wird sofort mit Flanke 0-1 an *Execute* gesetzt und wird erst zurückgesetzt, wenn das Kommando erfolgreich oder auch nicht erfolgreich beendet wurde.
- Solange *Busy* TRUE ist, muss der Funktionsbaustein zyklisch aufgerufen werden, um das Kommando ausführen zu können.

Verhalten des *Active*-Ausgangs

- Wenn die Bewegung einer Achse durch mehrere Funktionsbausteine gesteuert wird, so zeigt der *Active*-Ausgang jedes Bausteins an, dass das Kommando durch die Achse ausgeführt wird.

***Enable*-Eingang und *Valid*-Ausgang**

- Im Gegensatz zu *Execute* führt der *Enable*-Eingang dazu, dass eine Aktion permanent und wiederholt ausgeführt wird, solange *Enable* TRUE ist. MC_ReadStatus aktualisiert beispielsweise zyklisch den Zustand einer Achse solange *Enable* TRUE ist.
- Ein Funktionsbaustein mit einem *Enable*-Eingang zeigt durch den *Valid*-Ausgang an, dass die an den Ausgängen angezeigten Daten gültig sind. Die Daten können jedoch ständig aktualisiert werden während *Valid* TRUE ist.

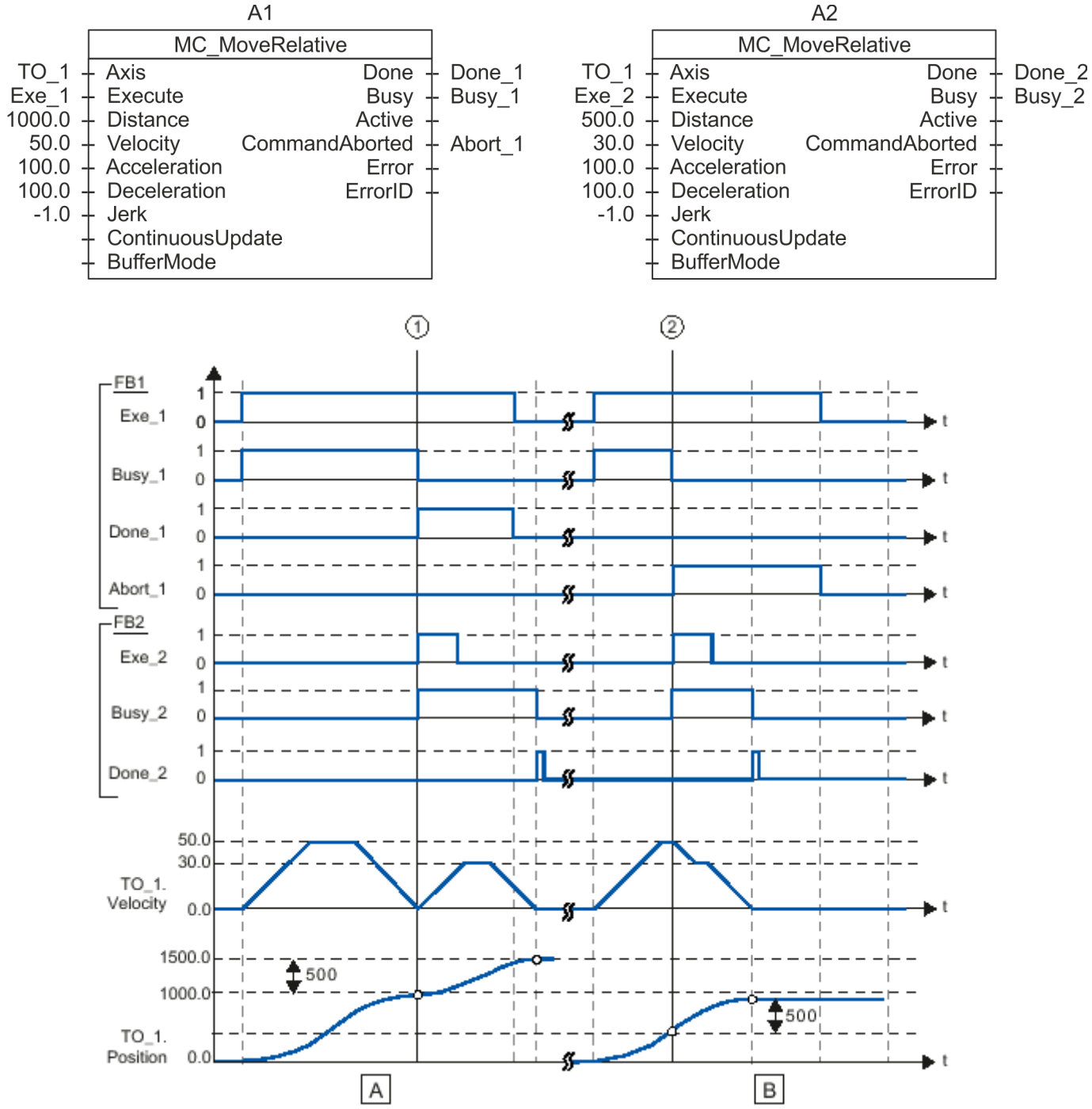
BufferMode

- *BufferMode* wird nicht unterstützt.

14.1.5 Ablöseverhalten von Bewegungsaufträgen

Beispiel

Das Ablöseverhalten von Bewegungsaufträgen wird nachfolgend am Beispiel von MC_MoveRelative erklärt. ↳ *Kap. 14.3.5 "FB 704 - MC_MoveRelative - Achse relativ verfahren" Seite 493*



(A) Die Achse wird durch einen "MC_MoveRelative"-Auftrag (A1) um die Wegstrecke *Distance* 1000.0 verfahren (Startposition ist hier die Position 0.0).

(1) Das Erreichen der Zielposition wird zum Zeitpunkt (1) über *Done_1* gemeldet. Zu diesem Zeitpunkt (1) wird ein weiterer MC_MoveRelative-Auftrag (A2) mit der Wegstrecke 500.0 gestartet. Das erfolgreiche Erreichen der neuen Zielposition wird über *Done_2* gemeldet. Da *Exe_2* vorher zurückgesetzt wurde, steht *Done_2* nur für einen Zyklus an.

(B) Ein laufender MC_MoveRelative-Auftrag (A1) wird durch einen weiteren MC_MoveRelative-Auftrag (A2) abgelöst.

(2) Der Abbruch wird zum Zeitpunkt (2) über *Abort_1* gemeldet. Die Achse wird anschließend mit der neuen Geschwindigkeit um die Wegstrecke *Distance* 500.0 verfahren. Das Erreichen der neuen Zielposition wird über *Done_2* gemeldet.

14.1.6 Erweiterte Vorgaben von Achsdaten

Lokale Achsdaten mit "#@" übernehmen

Die erweiterte Vorgabe von Achsdaten wird nur innerhalb von Multiinstanzen unterstützt, d.h. Aufruf eines MC-FBs innerhalb eines Rahmen-FBs.

Für den Einsatz immer wiederkehrender MC-FB-Abfolgen können Sie diese in einem Rahmen-FB (z.B. FB100) zusammenfassen. Diesem FB können Sie die Achsdaten vom Typ *VAR_IN_OUT* beim Aufruf als Lokaldaten übergeben. Im *SPEED7 Studio* haben Sie die Möglichkeit innerhalb des FB diese lokalen Achsdaten für den Aufruf der MC-FBs für die Achsdaten vom Typ *MC_AXIS_REF* zu verwenden, indem Sie in der Deklaration den Variablennamen des FB verwenden und diesem #@ voranstellen.

Beispiel

Aufruf FB100 aus OB1

```
CALL  FB   100, DB100
         Execute         :=
         Done            :=
         Busy            :=
         Error           :=
         ErrorID         :=
         Axis            := "MC_Axis_01".AXIS

CALL  FB   101, DB101
         Execute         :=
         Done            :=
         Busy            :=
         Error           :=
         ErrorID         :=
         Axis            := "MC_Axis_02".AXIS
```

Aufruf MC-FB aus FB100 bzw. FB101

```
CALL  #MC_Halt_1
         Execute         :=
         Deceleration    :=
         Jerk            :=
         BufferMode      :=
         Done            :=
         Busy            :=
         Active          :=
         CommandAborted  :=
         Error           :=
         ErrorID         :=
         Axis            := #@Axis
```

14.2 Typen - Datenstrukturen

14.2.1 UDT 8189 - MC_TRIGGER_REF - Datenstruktur Triggersignal

Beschreibung

In dieser Struktur werden Informationen zum ausgewählten Trigger-Event eingetragen. Bei Aktivierung von *Motion Control* im Projekt wird diese UDT automatisch dem Projekt hinzugefügt. Die UDT besitzt folgende Struktur:

Parameter

Name	Datentyp	Beschreibung
Probe	Byte	■ 0x00: ungültig ■ 0x01: Probe 1 ■ 0x02: Probe 2 ■ 0x03 ... 0xFF: reserviert
TriggerSource	Byte	■ 0x00: Eingang ■ 0x01: Geber Nullspur (Phase C) ■ 0x02 ... 0xFF: reserviert
TriggerMode	Byte	■ 0x00: Single trigger mode ■ 0x01 ... 0xFF: reserviert
Reserved	Byte	Reserviert.

14.2.2 UDT 8190 - MC_DIAGNOSTIC_REF - Datenstruktur interner Status

Beschreibung

Diese Struktur wird intern zur Status- und Diagnoseverarbeitung verwendet. Bei Aktivierung von *Motion Control* im Projekt wird diese UDT automatisch dem Projekt hinzugefügt.

14.2.3 UDT 8191 - MC_SETUP_UDT - Datenstruktur Konfigurationsdaten

Beschreibung

Diese Struktur wird zur internen Verwaltung von Konfigurationsdaten verwendet. Bei Aktivierung von *Motion Control* im Projekt wird diese UDT automatisch dem Projekt hinzugefügt.

14.2.4 UDT 8192 - MC_AXIS_REF - Datenstruktur Achsdaten

Beschreibung

Diese Struktur dient als Referenz zur Achse. Bei Aktivierung von *Motion Control* im Projekt wird diese UDT automatisch dem Projekt hinzugefügt.

14.3 Single Axis - Einfachachsen

14.3.1 FB 700 - MC_Power - Achsenfreigabe

Beschreibung

Mit MC_Power kann eine Achse freigegeben bzw. gesperrt werden.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Enable	INPUT	BOOL	■ Achsenfreigabe – TRUE: Die Achse wird freigegeben – FALSE: Die Achse wird gesperrt
EnablePositive	INPUT	BOOL	Parameter aktuell nicht unterstützt; Aufruf mit FALSE
EnableNegative	INPUT	BOOL	Parameter aktuell nicht unterstützt; Aufruf mit FALSE
Status	OUTPUT	BOOL	■ Status Achse – TRUE: Achse nimmt Bewegungsaufträge an – FALSE: Achse nimmt keine Bewegungsaufträge an
Valid	OUTPUT	BOOL	Immer FALSE
Error	OUTPUT	BOOL	■ Fehler – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter ErrorID entnommen werden. Die Achse wird gesperrt.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  - OB 1
- Anwendbar auf:
  - Positionierungsachse
  - Drehzahlachse
  - Externer Geber
  - Virtuelle Achse

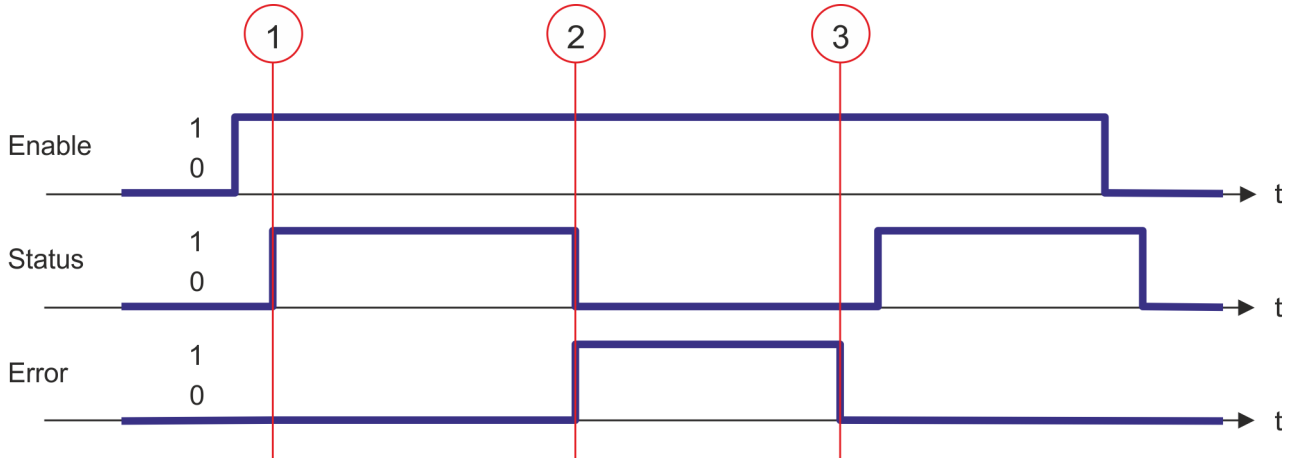
Achse freigeben

Aufruf von MC_Power mit *Enable* = TRUE. Sobald *Status* den Wert TRUE zeigt, ist die Achse freigegeben. In diesem Zustand können Bewegungsaufträge aktiviert werden.

Achse sperren

Aufruf von MC_Power mit *Enable* = FALSE. Sobald *Status* den Wert FALSE zeigt, ist die Achse gesperrt. Bei Sperren der Achse wird ein ggf. aktiver Bewegungsauftrag abgebrochen und die Achse gestoppt.

Zustandsdiagramm der Bausteinparameter



(1) Die Achse wird mit *Enable* = TRUE freigegeben. Zum Zeitpunkt (1) ist die Freigabe erfolgt. Anschließend können Bewegungsaufträge aktiviert werden.
(2) Zum Zeitpunkt (2) tritt ein Fehler auf, der das Sperren der Achse zur Folge hat. Ein ggf. aktiver Bewegungsauftrag wird abgebrochen und die Achse gestoppt.
(3) Der Fehler wird beseitigt und zum Zeitpunkt (3) quittiert. Da *Enable* weiterhin gesetzt ist, wird die Achse wieder freigegeben. Zuletzt wird die Achse mit *Enable* = FALSE gesperrt.

14.3.2 FB 701 - MC_Home - Achse referenzieren

Beschreibung

Mit MC_Home kann eine Achse referenziert werden. Dadurch kann ein Bezug zwischen der Position der Achse und der mechanischen Stellung hergestellt werden. Die Referenzfahrt-Methode und die zugehörigen Parameter müssen Sie an der realen Achse konfigurieren. Bei einer virtuellen Achse ist keine Konfiguration möglich. Hier wird die Istposition der Achse gleich dem Eingangsparameter *Position* gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Referenzfahrt – Flanke 0-1: Referenzfahrt wird gestartet
Position	INPUT	REAL	Bei erfolgreicher Referenzierung wird die Istposition der Achse einmalig gleich *Position* gesetzt. *Position* ist in der verwendeten Anwendereinheit anzugeben.
BufferMode	INPUT	BYTE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung

Parameter	Deklaration	Datentyp	Beschreibung
CommandA-borted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen.
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse
  – Drehzahlachse
  – Virtuelle Achse

PLCopen-State

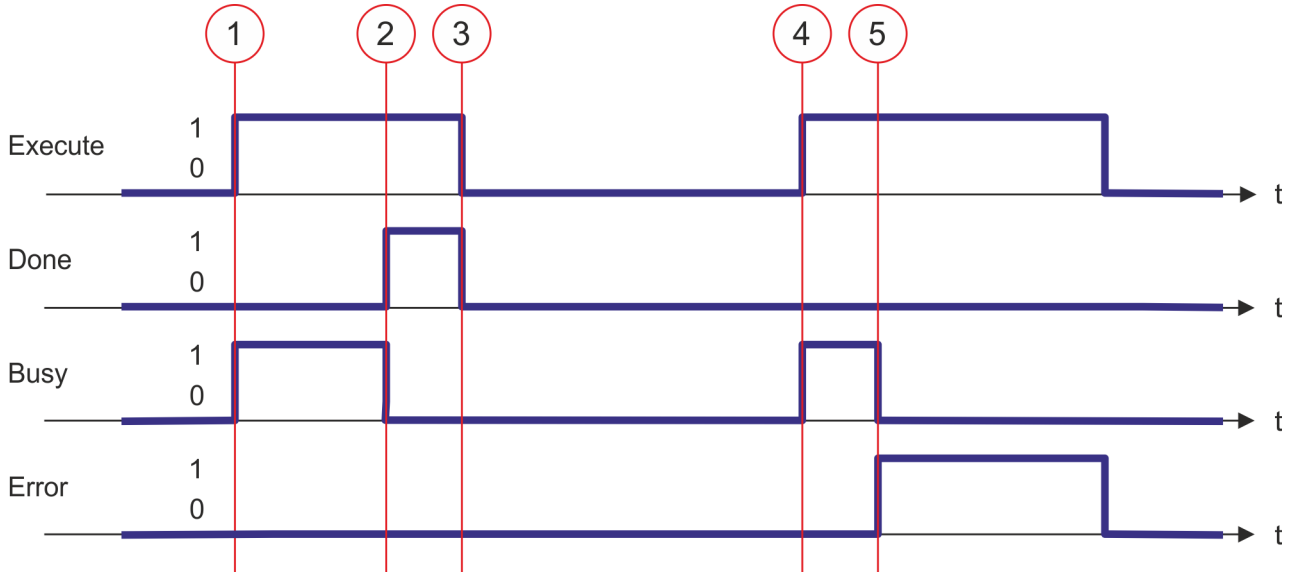
Start des Auftrags nur im PLCopen-State *Standstill* möglich.

Achse referenzieren

Mit einer Flanke 0-1 an *Execute* wird die Referenzierung gestartet. Solange die Referenzierung läuft zeigt *Busy* den Wert TRUE. Sobald *Done* den Wert TRUE hat, ist die Referenzierung erfolgreich abgeschlossen. Die Istposition der Achse wurde auf den Wert von *Position* gesetzt.

– *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.*
– *Ein laufender Auftrag kann durch einen Bewegungsauftrag (z.B. MC_MoveRelative) nicht abgebrochen werden.*

Zustandsdiagramm der Bausteinparameter



(1) Mit Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird die Referenzierung gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist die Referenzierung abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.
(4) Zum Zeitpunkt (4) wird erneut die Referenzierung mit einer Flanke 0-1 an *Execute* gestartet und *Busy* liefert den Wert TRUE.
(5) Zum Zeitpunkt (5) tritt ein Fehler bei der Referenzierung auf. *Busy* liefert den Wert FALSE und *Error* den Wert TRUE.

14.3.3 FB 702 - MC_Stop - Achse stoppen

Beschreibung

Mit MC_Stop wird die Achse gestoppt. Mit den Parametern *Deceleration* und *Jerk* kann das dynamische Verhalten beim Stoppvorgang bestimmt werden.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Achse stoppen – Flanke 0-1: Stoppen der Achse wird gestartet
Deceleration	INPUT	REAL	Verzögerung beim Stoppen in [Anwendereinheiten/s^2]
Jerk	INPUT	REAL	Ruck beim Stoppen in [Anwendereinheiten/s^3]
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt

Parameter	Deklaration	Datentyp	Beschreibung
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
CommandA-borted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  - OB 1
  - OB 61
- Anwendbar auf:
  - Positionierungsachse
  - Drehzahlachse
  - Virtuelle Achse

PLCopen-State

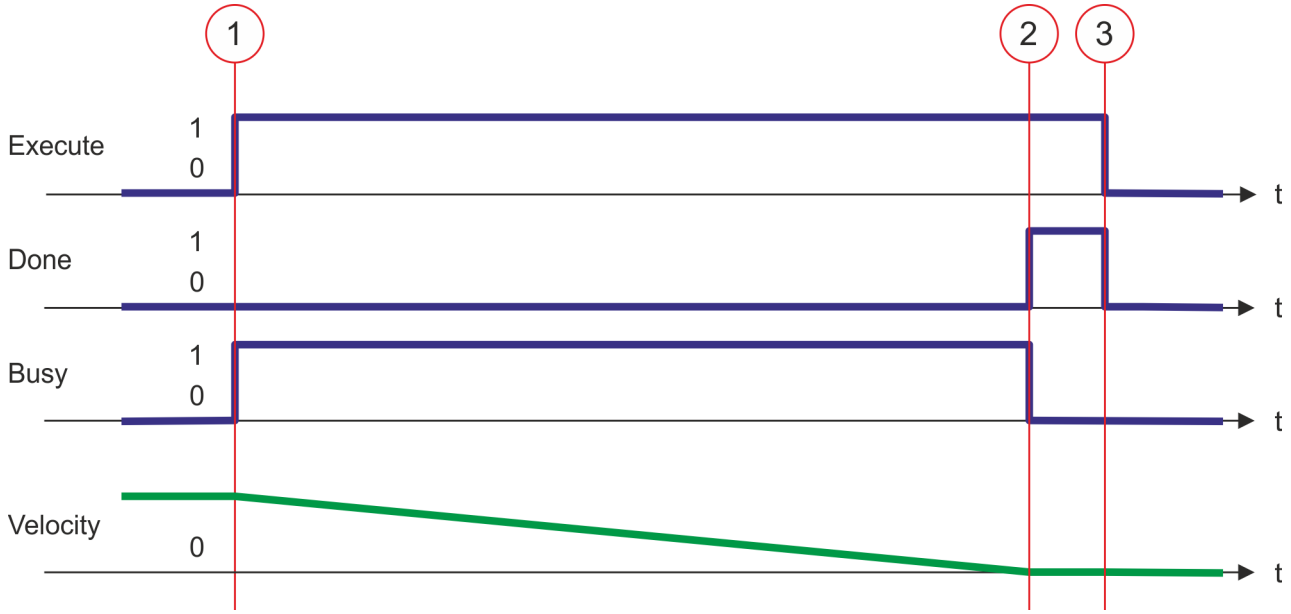
- Start des Auftrags in den PLCopen-States *Standstill*, *Homing*, *Discrete Motion*, *Synchronized Motion* und *Continuous Motion* möglich.
- MC_Stop führt die Achse in den PLCopen-State *Stopping* über. In *Stopping* können keine Bewegungsaufträge gestartet werden. Solange *Execute* gleich TRUE ist, bleibt die Achse im PLCopen-State *Stopping*. Wird *Execute* gleich FALSE gesetzt, geht die Achse in den PLCopen-State *Standstill* über. In *Standstill* können Bewegungsaufträge gestartet werden.

Achse stoppen

Mit einer Flanke 0-1 an *Execute* wird das Stoppen der Achse gestartet. Solange das Stoppen der Achse läuft, zeigt *Busy* den Wert TRUE. Nachdem die Achse gestoppt wurde und somit die Geschwindigkeit 0 erreicht hat, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE bis zum Stopp der Achse ausgeführt.*
- *Ein laufender Auftrag kann durch einen Bewegungsauftrag (z.B. MC_MoveRelative) nicht abgebrochen werden.*

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Stoppen der Achse gestartet und *Busy* liefert den Wert TRUE. Die Geschwindigkeit der Achse wird unter Berücksichtigung der Parameter *Deceleration* und *Jerk* bis auf null verringert.
(2) Zum Zeitpunkt (2) ist das Stoppen der Achse abgeschlossen, die Achse ist gestoppt. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.4 FB 703 - MC_Halt - Achse anhalten

Beschreibung

Mit MC_Halt wird die Achse bis zum Stillstand abgebremst. Mit den Parametern *Deceleration* und *Jerk* kann das dynamische Verhalten beim Bremsvorgang bestimmt werden.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Achse anhalten – Flanke 0-1: Anhalten der Achse wird gestartet
Deceleration	INPUT	REAL	Verzögerung beim Bremsen in [Anwendereinheiten/s^2]
Jerk	INPUT	REAL	Ruck beim Stoppen in [Anwendereinheiten/s^3]
BufferMode	INPUT	BYTE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung

Parameter	Deklaration	Datentyp	Beschreibung
Active	OUTPUT	BOOL	■ Status – TRUE: Baustein hat die Kontrolle über die Achse
CommandA-borted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehler-informationen können dem Parameter *ErrorID* ent-nommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  - OB 1
- Anwendbar auf:
  - Positionierungsachse
  - Drehzahlachse
  - Virtuelle Achse

PLCopen-State

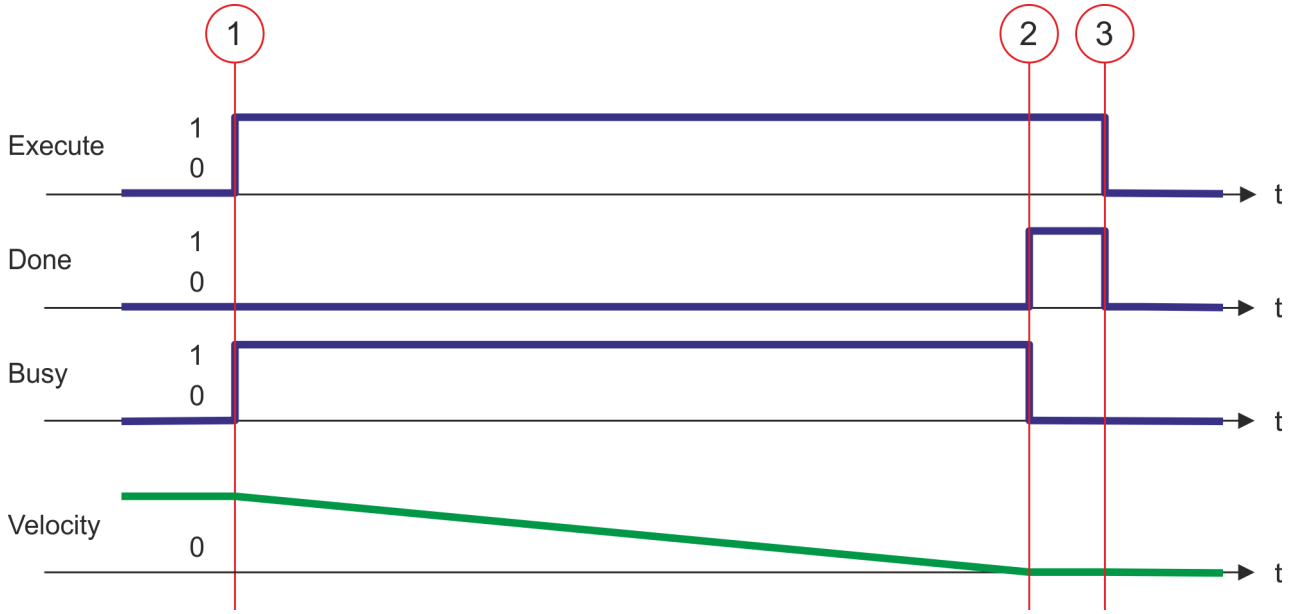
- Start des Auftrags in den PLCopen-States *Discrete Motion*, *Synchronized Motion* und *Continuous Motion* möglich.
- MC_Halt führt die Achse in den PLCopen-State *Discrete Motion* über.

Achse anhalten

Mit einer Flanke 0-1 an *Execute* wird das Anhalten der Achse gestartet. Solange das Anhalten der Achse läuft, zeigt *Busy* den Wert TRUE. Nachdem die Achse angehalten wurde und somit die Geschwindigkeit 0 erreicht hat, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE bis zum Anhalten der Achse ausgeführt.*
- *Ein laufender Auftrag kann durch einen anderen Bewegungsauftrag (z.B. MC_MoveRelative) abgebrochen werden.*

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Anhalten der Achse gestartet und *Busy* liefert den Wert TRUE. Die Geschwindigkeit der Achse wird unter Berücksichtigung der Parameter *Deceleration* und *Jerk* bis auf 0 verringert.
(2) Zum Zeitpunkt (2) ist das Anhalten der Achse abgeschlossen, die Achse steht. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.5 FB 704 - MC_MoveRelative - Achse relativ verfahren

Beschreibung

Mit MC_MoveRelative wird die Achse relativ zu der Position bei Auftragsstart um eine spezifizierte Distanz verfahren. Mit den Parametern *Velocity*, *Jerk*, *Acceleration* und *Deceleration* wird das dynamische Verhalten beim Bewegungsvorgang bestimmt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Achse relativ verfahren – Flanke 0-1: Das relative Verfahren der Achse wird gestartet
ContinuousUpdate	INPUT	BOOL	Parameter aktuell nicht unterstützt; Aufruf mit FALSE
Distance	INPUT	REAL	Relative Wegstrecke in [Anwendereinheiten]
Velocity	INPUT	REAL	Max. Geschwindigkeit (muss nicht zwingend erreicht werden) in [Anwendereinheiten/s]
Acceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2]

Parameter	Deklaration	Datentyp	Beschreibung
Deceleration	INPUT	REAL	Verzögerung in [Anwendereinheiten/s^2]
Jerk	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
BufferMode	INPUT	BYTE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt; Zielposition erreicht
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Active	OUTPUT	BOOL	■ Status – TRUE: Baustein hat die Kontrolle über die Achse
CommandA-borted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  - OB 1
  - OB 61
- Anwendbar auf:
  - Positionierungsachse
  - Virtuelle Achse

PLCopen-State

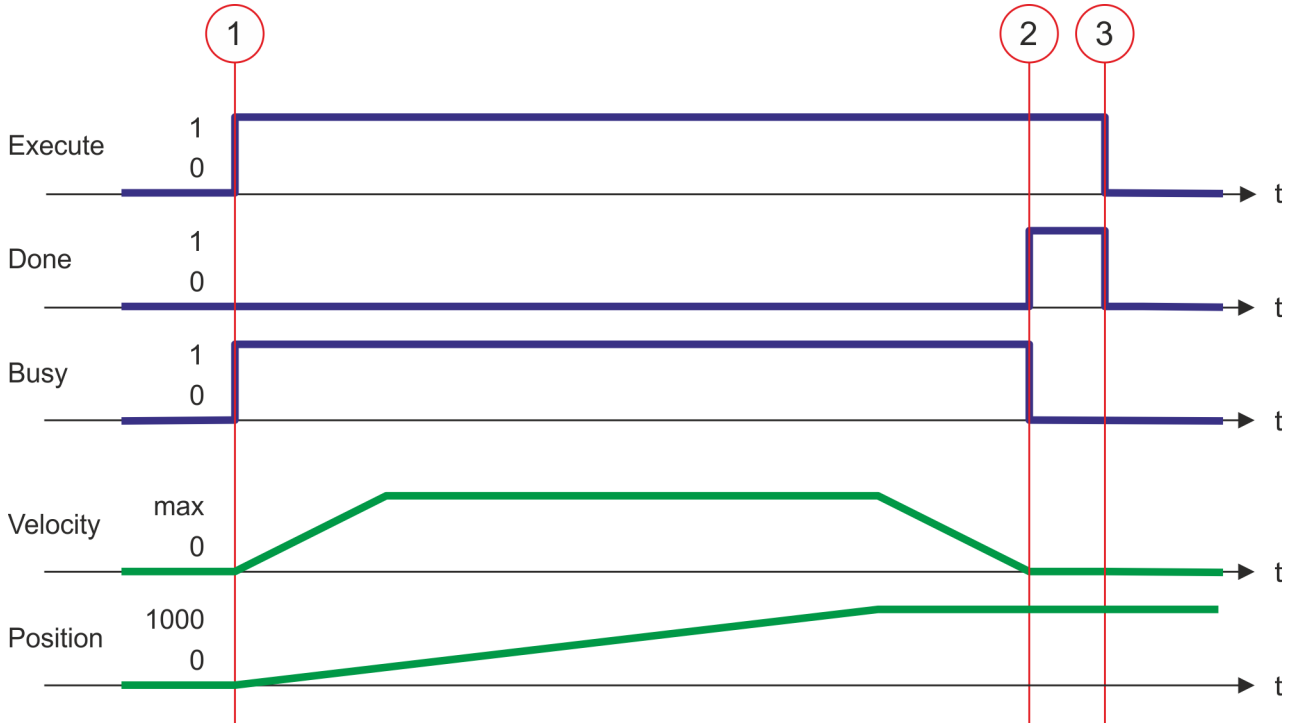
- Start des Auftrags in den PLCopen-States *Standstill*, *Discrete Motion*, *Synchronized Motion* und *Continuous Motion* möglich.
- MC_MoveRelative führt die Achse in den PLCopen-State *Discrete Motion* über.

Achse relativ verfahren

Mit einer Flanke 0-1 an *Execute* wird das Verfahren der Achse gestartet. Solange das Verfahren der Achse läuft, zeigt *Busy* den Wert TRUE. Nachdem die Zielposition erreicht wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Die Geschwindigkeit der Achse ist dann gleich 0.

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE bis die Achse die Zielposition erreicht hat, ausgeführt.*
- *Ein laufender Auftrag kann durch einen anderen Bewegungsauftrag (z.B. MC_MoveAbsolute) abgebrochen werden.*

Zustandsdiagramm der Bausteinparameter



(1) Die Achse wird mit MC_MoveRelative um eine *Distance* = 1000.0 verfahren (Startposition bei Auftragsstart gleich 0.0). Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Verfahren der Achse gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) wurde die Achse um die *Distance* = 1000.0 verfahren, d.h. die Zielposition wurde erreicht. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.6 FB 705 - MC_MoveVelocity - Achse verfahren mit konstanter Geschwindigkeit

Beschreibung

Mit MC_MoveVelocity wird die Achse mit einer konstanten Geschwindigkeit verfahren. Mit den Parametern *Velocity*, *Jerk*, *Acceleration* und *Deceleration* wird das dynamische Verhalten beim Bewegungsvorgang bestimmt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Achse mit konstanter Geschwindigkeit verfahren – Flanke 0-1: Das Verfahren der Achse mit konstanter Geschwindigkeit wird gestartet
ContinuousUpdate	INPUT	BOOL	Parameter aktuell nicht unterstützt; Aufruf mit FALSE

Parameter	Deklaration	Datentyp	Beschreibung
Velocity	INPUT	REAL	Geschwindigkeitsvorgabe (vorzeichenbehafteter Wert) in [Anwendereinheiten/s]
Acceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2]
Deceleration	INPUT	REAL	Verzögerung in [Anwendereinheiten/s^2]
Jerk	INPUT	REAL	Ruck in [Anwendereinheiten/s^3]
BufferMode	INPUT	BYTE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0
InVelocity	OUTPUT	BOOL	■ Geschwindigkeitsvorgabe – TRUE: Geschwindigkeitsvorgabe erreicht
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Active	OUTPUT	BOOL	■ Status – TRUE: Baustein hat die Kontrolle über die Achse
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  - OB 1
  - OB 61
- Anwendbar auf:
  - Positionierungsachse
  - Drehzahlachse
  - Virtuelle Achse

PLCopen-State

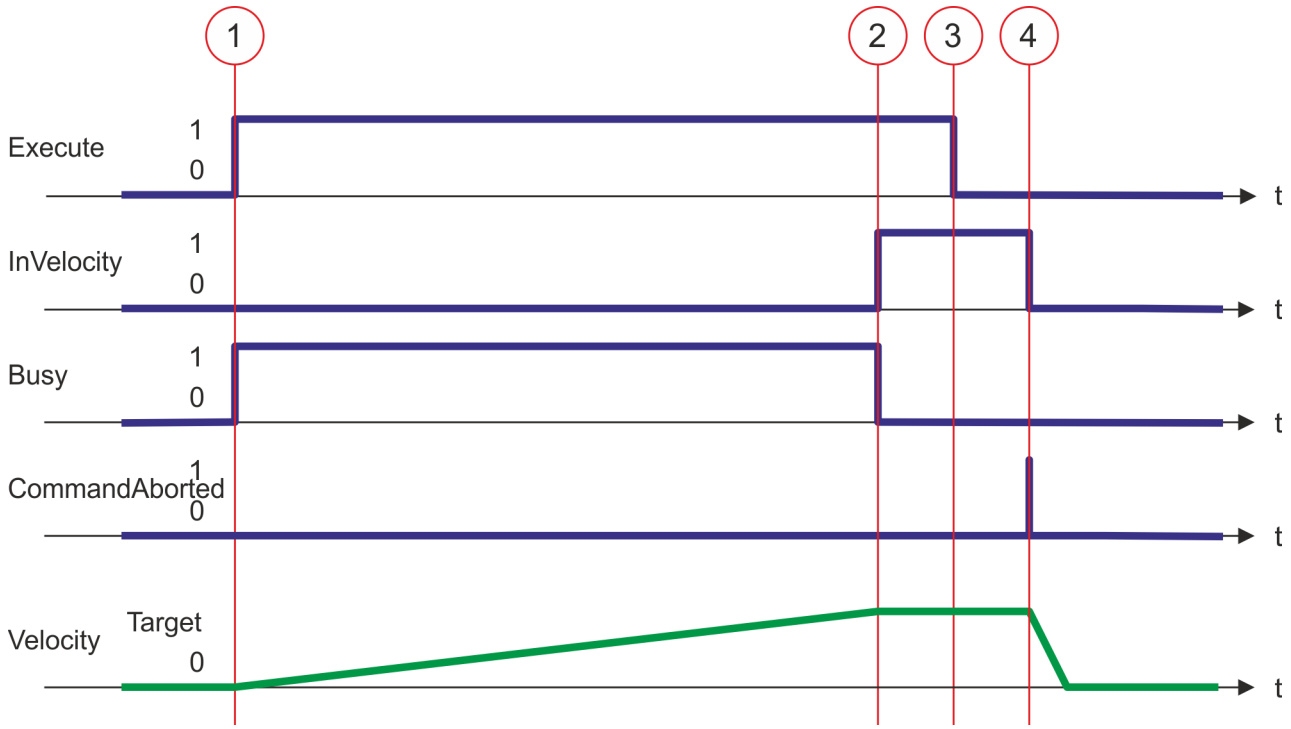
- Start des Auftrags in den PLCopen-States *Standstill*, *Discrete Motion*, *Synchronized Motion* und *Continuous Motion* möglich.
- MC_MoveVelocity führt die Achse in den PLCopen-State *Continuous Motion* über.

Achse mit Geschwindigkeitsvorgabe verfahren

Mit einer Flanke 0-1 an *Execute* wird das Verfahren der Achse mit Geschwindigkeitsvorgabe gestartet. Solange die Geschwindigkeitsvorgabe nicht erreicht ist, zeigt *Busy* den Wert TRUE und *InVelocity* den Wert FALSE. Ist die Geschwindigkeitsvorgabe erreicht, wird *Busy* gleich FALSE und *InVelocity* gleich TRUE. Die Achse wird mit dieser Geschwindigkeit konstant weiter verfahren.

– *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt, auch wenn die Geschwindigkeitsvorgabe erreicht wurde.*
– *Ein laufender Auftrag kann durch einen anderen Bewegungsauftrag (z.B. MC_MoveAbsolute) abgebrochen werden.*

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an Execute zum Zeitpunkt (1) wird das Verfahren der Achse mit Geschwindigkeitsvorgabe gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) erreicht die Achse die Geschwindigkeitsvorgabe und *Busy* liefert den Wert FALSE und *InVelocity* den Wert TRUE.
(3) Das Rücksetzen von Execute auf FALSE zum Zeitpunkt (3) hat keine Auswirkung auf die Achse. Die Achse wird weiterhin konstant mit der Geschwindigkeitsvorgabe verfahren und *InVelocity* liefert weiterhin den Wert TRUE.
(4) Zum Zeitpunkt (4) wird der MC_Velocity-Auftrag durch einen MC_Halt-Auftrag abgebrochen. Die Achse wird bis zum Halt abgebremst.

14.3.7 FB 708 - MC_MoveAbsolute - Achse auf absolute Position verfahren

Beschreibung

Mit MC_MoveAbsolute wird die Achse auf eine absolute Position verfahren. Mit den Parametern *Velocity*, *Jerk*, *Acceleration* und *Deceleration* wird das dynamische Verhalten beim Bewegungsvorgang bestimmt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Verfahren der Achse starten – Flanke 0-1: Das Verfahren der Achse wird gestartet
ContinuousUpdate	INPUT	BOOL	Parameter aktuell nicht unterstützt; Aufruf mit FALSE
Position	INPUT	REAL	Absolute Position in [Anwendereinheiten]
Velocity	INPUT	REAL	Maximale Geschwindigkeit (muss nicht zwingend erreicht werden) vorzeichenbehafteter Wert in [Anwendereinheiten/s]
Acceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2]
Deceleration	INPUT	REAL	Verzögerung in [Anwendereinheiten/s^2]
Jerk	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
Direction	INPUT	Byte	■ Richtung – 0: Kürzeste Entfernung – 1: Positive Richtung – 2: Negative Richtung – 3: Aktuelle Richtung
BufferMode	INPUT	BYTE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Die Zielposition wurde erreicht.
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Active	OUTPUT	BOOL	■ Status – TRUE: Baustein hat die Kontrolle über die Achse
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Bei Achsen vom Achstyp begrenzt wird nur Direction == 0 unterstützt. Ist Direction <> 0, erhalten Sie einen Fehler (ErrorID: 0x8012).

Verwendung

- Bausteinaufruf im:
  - OB 1
  - OB 61
- Anwendbar auf:
  - Positionierungsachse
  - Virtuelle Achse

PLCopen-State

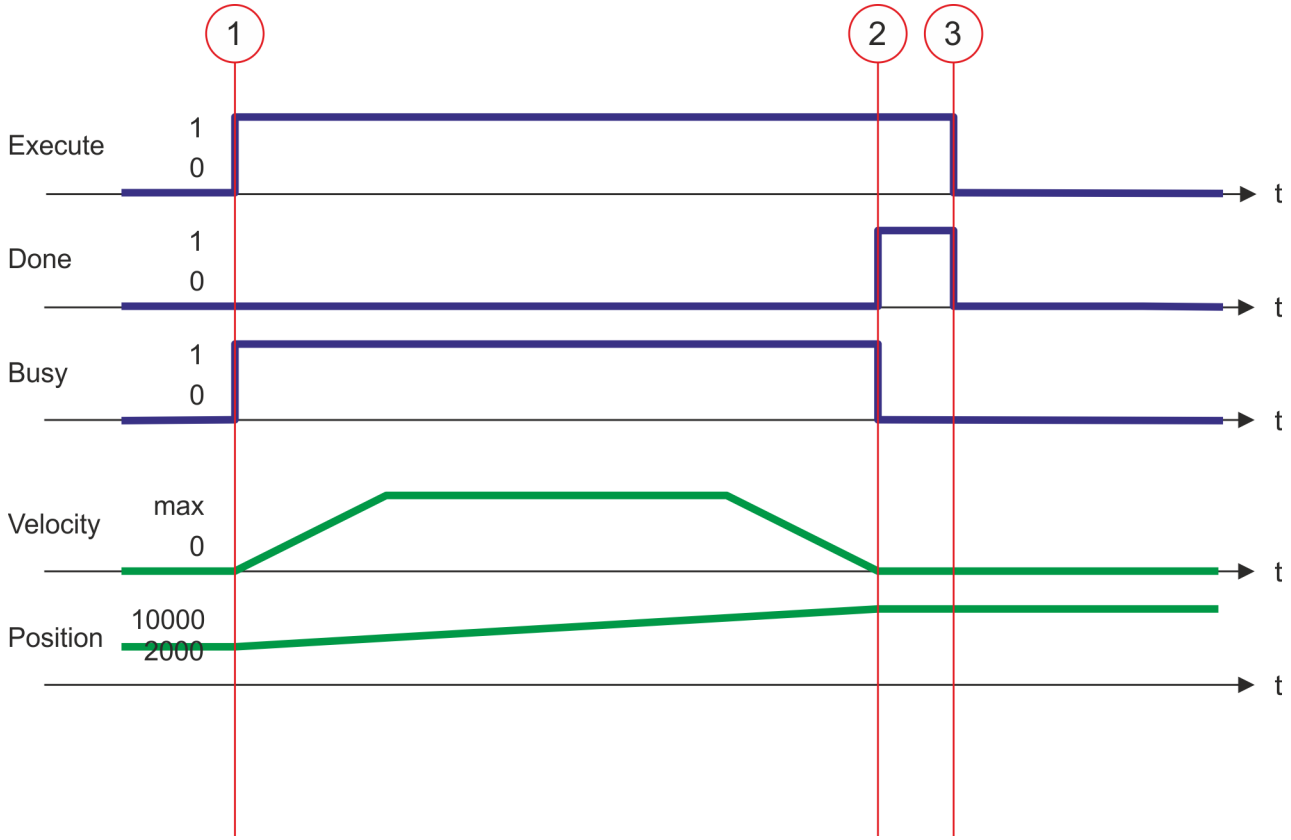
- Start des Auftrags in den PLCopen-States *Standstill*, *Discrete Motion*, *Synchronized Motion* und *Continuous Motion* möglich.
- MC_MoveVelocity führt die Achse in den PLCopen-State *Discrete Motion* über.

Achse absolute verfahren

Mit einer Flanke 0-1 an *Execute* wird das Verfahren der Achse gestartet. Solange das Verfahren der Achse läuft, zeigt *Busy* den Wert TRUE. Nachdem die Zielposition erreicht wurde, wird *Busy* = FALSE und *Done* = TRUE geliefert. Die Geschwindigkeit der Achse ist dann gleich 0.

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE, bis die Achse die Zielposition erreicht hat, ausgeführt.*
- *Ein laufender Auftrag kann durch einen anderen Bewegungsauftrag (z.B. MC_MoveVelocity) abgebrochen werden.*

Zustandsdiagramm der Bausteinparameter



(1) Die Achse wird mit MC_MoveAbsolute auf die absolute Position = 10000.0 verfahren (Startposition bei Auftragsstart gleich 2000.0). Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Verfahren der Achse gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) hat die Achse die Zielposition erreicht. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.8 FB 711 - MC_Reset - Reset Achse

Beschreibung

Mit MC_Reset wird ein Reset (Neuinitialisieren) der Achse durchgeführt. Dabei werden alle internen Fehler der Achse zurückgesetzt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Reset Achse – Flanke 0-1: Reset der Achse wird durchgeführt
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Reset wurde durchgeführt

Parameter	Deklaration	Datentyp	Beschreibung
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse
  – Drehzahlachse
  – Virtuelle Achse

PLCopen-State

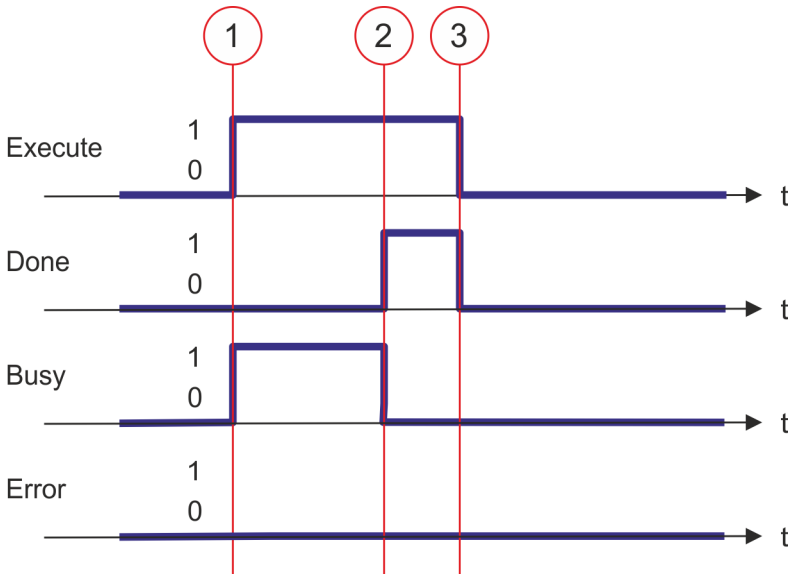
- Start des Auftrags im PLCopen-State *ErrorStop* möglich.
- MC_Reset führt die Achse in Abhängigkeit von MC_Power entweder in den PLCopen-State *Standstill* (Aufruf von MC_Power mit *Enable* = TRUE) oder *Disabled* (Aufruf von MC_Power mit *Enable* = FALSE) über.

Reset an Achse durchführen

Mit einer Flanke 0-1 an *Execute* wird der Reset der Achse gestartet. Solange der Reset der Achse läuft, zeigt *Busy* den Wert TRUE. Nachdem die Achse neu initialisiert wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE ausgeführt, bis der Auftrag abgeschlossen ist.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird der Reset der Achse gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist der Reset erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.9 FB 712 - MC_ReadStatus - PLCopen-State

Beschreibung Mit MC_ReadStatus kann der PLCopen-State der Achse ermittelt werden.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Slave-Achse
Enable	INPUT	BOOL	■ Statusanzeige – TRUE: Der Status wird an den Ausgängen permanent angezeigt – FALSE: Alle Ausgänge werden gleich FALSE bzw. 0
Valid	OUTPUT	BOOL	■ Status gültig – TRUE: Der angezeigte Status ist gültig
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Parameter	Deklaration	Datentyp	Beschreibung
ErrorStop	OUTPUT	BOOL	■ Achsfehler – TRUE: Achsfehler aufgetreten; ein Bewegungsauftrag kann nicht aktiviert werden
Disabled	OUTPUT	BOOL	■ Status Achse: Sperrung – TRUE: Achse ist gesperrt; ein Bewegungsauftrag kann nicht aktiviert werden
Stopping	OUTPUT	BOOL	■ Status Achse: Stop – TRUE: Achse wird gestoppt (MC_Stop ist aktiv)
Homing	OUTPUT	BOOL	■ Status Achse: Referenzierung – TRUE: Achse wird referenziert (MC_Homing ist aktiv)
Standstill	OUTPUT	BOOL	■ Status Bewegungsauftrag – TRUE: Kein Bewegungsauftrag aktiv; Bewegungsauftrag kann aktiviert werden
DiscreteMotion	OUTPUT	BOOL	■ Status Achsbewegung: Diskret – TRUE: Achse wird durch eine diskrete Bewegung verfahren (MC_MoveRelative, MC_MoveAbsolute oder MC_Halt ist aktiv)
ContinuousMotion	OUTPUT	BOOL	■ Status Achsbewegung: Kontinuierlich – TRUE: Achse wird durch eine kontinuierliche Bewegung verfahren (MC_MoveVelocity ist aktiv)
SynchronizedMotion	OUTPUT	BOOL	■ Status Achse: Slave-Achse – TRUE: Achse ist eine Slave-Achse (MC_CamIn oder MC_GearIn ist aktiv)

Verwendung

- Bausteinaufruf im:
  - OB 1
- Anwendbar auf:
  - Positionierungsachse
  - Drehzahlachse
  - Externer Geber
  - Virtuelle Achse

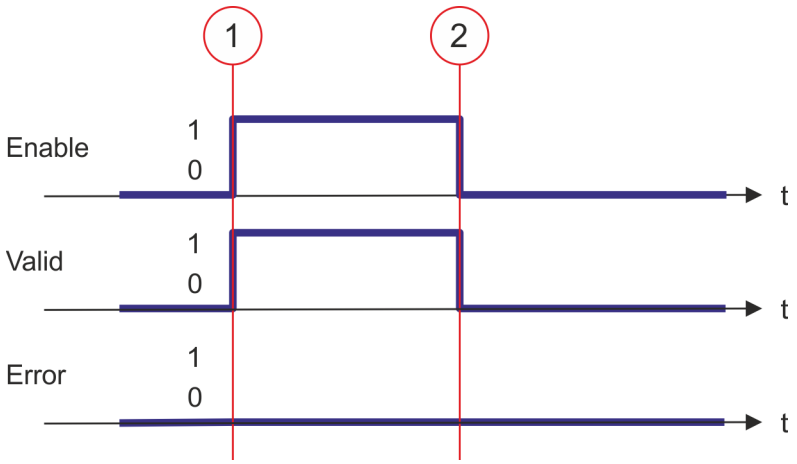
PLCopen-State

- Start des Auftrags in jedem PLCopen-State möglich.

Status der Achse ermitteln

Mit *Enable* = TRUE wird an den Ausgängen der Zustand der Achse entsprechend dem Zustandsdiagramm nach PLCopen geliefert.

Zustandsdiagramm der Bausteinparameter



(1) Zum Zeitpunkt (1) wird *Enable* = TRUE gesetzt. Damit liefert *Valid* den Wert TRUE und an den Ausgängen wird der Zustand entsprechend des PLCopen-Zustandsdiagramms angezeigt.
(2) Zum Zeitpunkt (2) wird *Enable* = FALSE gesetzt. Damit werden sämtliche Ausgänge gleich FALSE bzw. 0 gesetzt.

14.3.10 FB 713 - MC_ReadAxisError - Fehler Achse lesen

Beschreibung Mit MC_ReadAxisError wird der aktuell anstehende Fehler der Achse gelesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Reset Achse – Flanke 0-1: Achsfehler wird gelesen.
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Achsfehler ausgelesen.
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung.
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
AxisErrorID	OUTPUT	WORD	Achsfehler-ID; der gelieferte Wert ist Hersteller-spezifisch kodiert.

Verwendung

- Bausteinaufruf im:
  - OB 1
- Anwendbar auf:
  - Positionierungsachse
  - Drehzahlachse
  - Virtuelle Achse

PLCopen-State

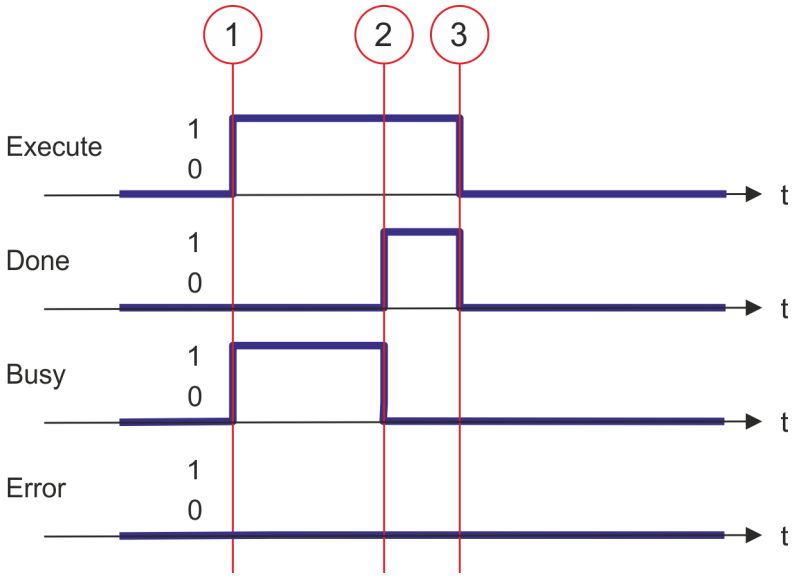
- Start des Auftrags in jedem PLCopen-State möglich.

Fehler der Achse lesen

Mit einer Flanke 0-1 an *Execute* wird das Lesen des Achsfehlers gestartet. Solange das Lesen des Achsfehlers läuft, zeigt *Busy* den Wert TRUE. Nachdem der Achsfehler gelesen wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der Ausgang *AxisErrorID* zeigt den aktuell anstehenden Achsfehler an.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Lesen des Achsfehlers gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Lesen des Achsfehlers erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.11 FB 714 - MC_ReadParameter - Parameter Achse lesen

Beschreibung

Mit MC_ReadParameter wird der Parameter, der über die Parameter-Nummer festgelegt ist, von der Achse gelesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Parameter Achse lesen – Flanke 0-1: Das Lesen des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher gelesen werden soll. ↳ *Kap. 14.6 "PLCopen Parameter" Seite 574*
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde ausgelesen
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
Value	OUTPUT	REAL	Wert des gelesenen Parameters

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse
  – Drehzahlachse
  – Virtuelle Achse (nur Parameter-Nummer < 1000 zulässig)

PLCopen-State

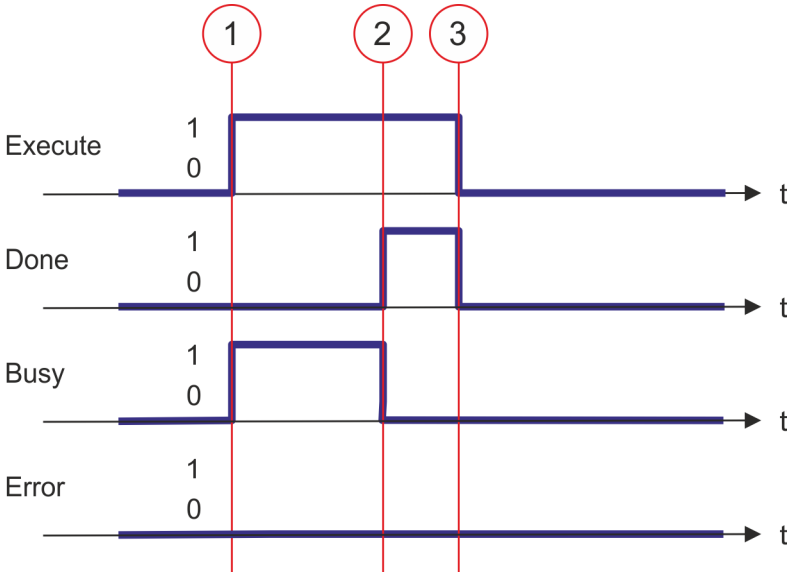
- Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse lesen

Mit einer Flanke 0-1 an *Execute* wird das Lesen des Parameters gestartet. Solange das Lesen des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter gelesen wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der Ausgang *Value* zeigt den Wert des Parameters an.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Lesen des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Lesen des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.12 FB 715 - MC_WriteParameter - Parameter Achse schreiben

Beschreibung

Mit MC_WriteParameter wird der Wert des Parameters, der über die Parameter-Nummer festgelegt ist, zur Achse geschrieben.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Parameter Achse schreiben – Flanke 0-1: Das Schreiben des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher geschrieben werden soll. ↳ *Kap. 14.6 "PLCopen Parameter" Seite 574*
Value	INPUT	REAL	Wert des geschriebenen Parameters
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde geschrieben
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung

Parameter	Deklaration	Datentyp	Beschreibung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse
  – Drehzahlachse
  – Virtuelle Achse (nur Parameter-Nummer < 1000 zulässig)

PLCopen-State

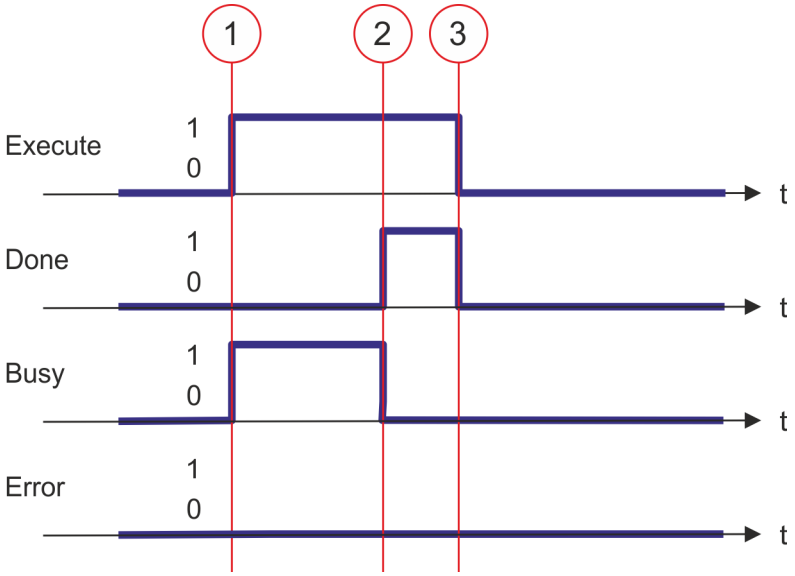
- Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse schreiben

Mit einer Flanke 0-1 an *Execute* wird das Schreiben des Parameters gestartet. Solange das Schreiben des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter geschrieben wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Schreiben des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Schreiben des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.13 FB 716 - MC_ReadActualPosition - Position Achse auslesen

Beschreibung Mit MC_ReadActualPosition wird die aktuelle Position der Achse gelesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Enable	INPUT	BOOL	■ Position Achse lesen – TRUE: Die Position der Achse wird kontinuierlich gelesen – FALSE: Alle Ausgänge werden gleich FALSE bzw. 0
Valid	OUTPUT	BOOL	■ Position gültig – TRUE: Die gelesene Position ist gültig
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
Position	OUTPUT	REAL	Position der Achse in [Anwendereinheiten]

Verwendung

- Bausteinaufruf im:
  - OB 1
  - OB 61
- Anwendbar auf:
  - Positionierungsachse
  - Drehzahlachse
  - Externer Geber
  - Virtuelle Achse

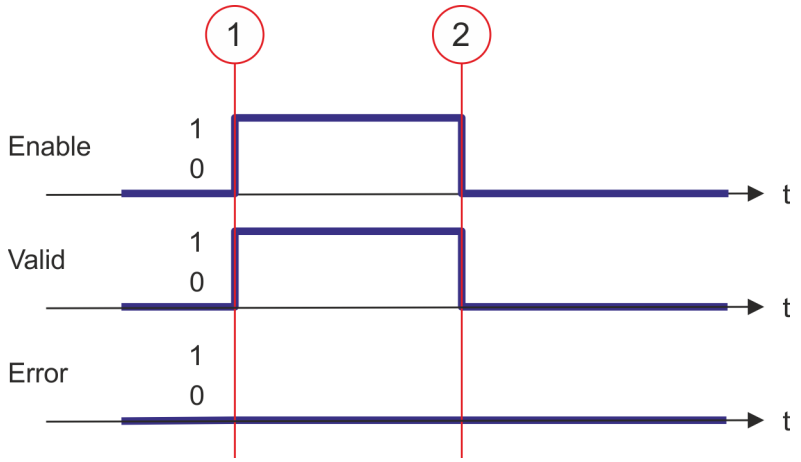
PLCopen-State

- Start des Auftrags in jedem PLCopen-State möglich.

Position der Achse lesen

Mit *Enable* gleich TRUE wird die aktuelle Position der Achse ermittelt und unter *Position* abgelegt.

Zustandsdiagramm der Bausteinparameter



(1) Zum Zeitpunkt (1) wird *Enable* = TRUE gesetzt. Damit liefert *Valid* den Wert TRUE und am Ausgang *Position* wird die aktuelle Position der Achse angezeigt.
(2) Zum Zeitpunkt (2) wird *Enable* = FALSE gesetzt. Damit werden sämtliche Ausgänge gleich FALSE bzw. 0 gesetzt.

14.3.14 FB 717 - MC_ReadActualVelocity - Geschwindigkeit der Achse lesen

Beschreibung

Mit MC_ReadActualVelocity wird die aktuelle Geschwindigkeit der Achse gelesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Enable	INPUT	BOOL	■ Geschwindigkeit Achse lesen – TRUE: Die Geschwindigkeit der Achse wird kontinuierlich gelesen – FALSE: Alle Ausgänge werden gleich FALSE bzw. 0

Parameter	Deklaration	Datentyp	Beschreibung
Valid	OUTPUT	BOOL	■ Geschwindigkeit gültig – TRUE: Die gelesene Geschwindigkeit ist gültig
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
Velocity	OUTPUT	REAL	Geschwindigkeit der Achse in [Anwendereinheiten/s]

Verwendung

- Bausteinaufruf im:
  - OB 1
  - OB 61
- Anwendbar auf:
  - Positionierungsachse
  - Drehzahlachse
  - Externer Geber
  - Virtuelle Achse

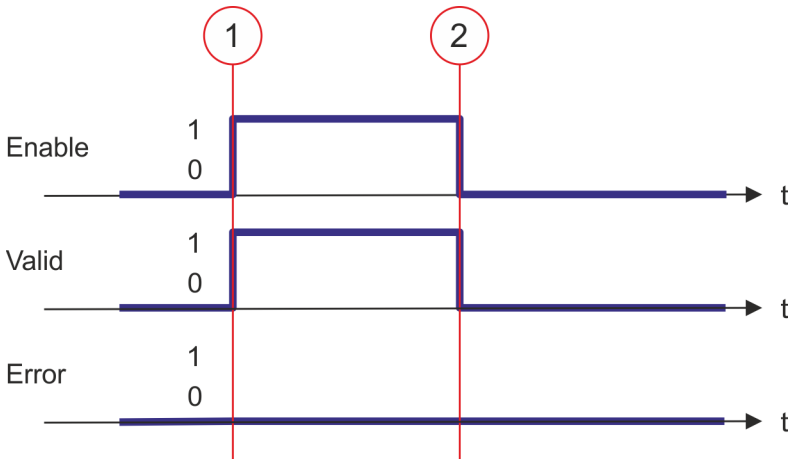
PLCopen-State

- Start des Auftrags in jedem PLCopen-State möglich.

Geschwindigkeit der Achse lesen

Mit *Enable* gleich TRUE wird die aktuelle Geschwindigkeit der Achse ermittelt und unter *Velocity* abgelegt.

Zustandsdiagramm der Bausteinparameter



(1) Zum Zeitpunkt (1) wird *Enable* = TRUE gesetzt. Damit liefert *Valid* den Wert TRUE und am Ausgang *Velocity* wird die aktuelle Geschwindigkeit der Achse angezeigt.
(2) Zum Zeitpunkt (2) wird *Enable* = FALSE gesetzt. Damit werden sämtliche Ausgänge gleich FALSE bzw. 0 gesetzt.

14.3.15 FB 718 - MC_ReadAxisInfo - Zusatzinformationen der Achse lesen

Beschreibung

Mit MC_ReadAxisInfo werden einige Zusatzinformationen der Achse angezeigt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Enable	INPUT	BOOL	■ Zusatzinformationen Achse lesen – TRUE: Die Zusatzinformationen der Achse werden kontinuierlich gelesen – FALSE: Alle Ausgänge werden gleich FALSE bzw. 0
Valid	OUTPUT	BOOL	■ Zusatzinformationen gültig – TRUE: Die gelesene Zusatzinformationen sind gültig
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
HomeAbsSwitch	OUTPUT	BOOL	Parameter aktuell nicht unterstützt; immer FALSE
LimitSwitchPos	OUTPUT	BOOL	Parameter aktuell nicht unterstützt; immer FALSE
LimitSwitchNeg	OUTPUT	BOOL	Parameter aktuell nicht unterstützt; immer FALSE
Simulation	OUTPUT	BOOL	Parameter aktuell nicht unterstützt; immer FALSE
Communication-Ready	OUTPUT	BOOL	■ Information Achse: Datenaustausch – TRUE: Datenaustausch mit der Achse initialisiert; Achse ist kommunikationsbereit
ReadyForPowerOn	OUTPUT	BOOL	■ Information Achse: Freigabe möglich – TRUE: Die Freigabe der Achse ist möglich
PowerOn	OUTPUT	BOOL	■ Information Achse: freigegeben – TRUE: Die Freigabe der Achse ist erfolgt
IsHomed	OUTPUT	BOOL	■ Information Achse: referenziert – TRUE: Die Achse ist referenziert
AxisWarning	OUTPUT	BOOL	■ Information Achse: Fehler – TRUE: Mindestens 1 Fehler wird von der Achse gemeldet

Verwendung

- Bausteinaufruf im:
  - OB 1
- Anwendbar auf:
  - Positionierungsachse
  - Drehzahlachse
  - Externer Geber
  - Virtuelle Achse

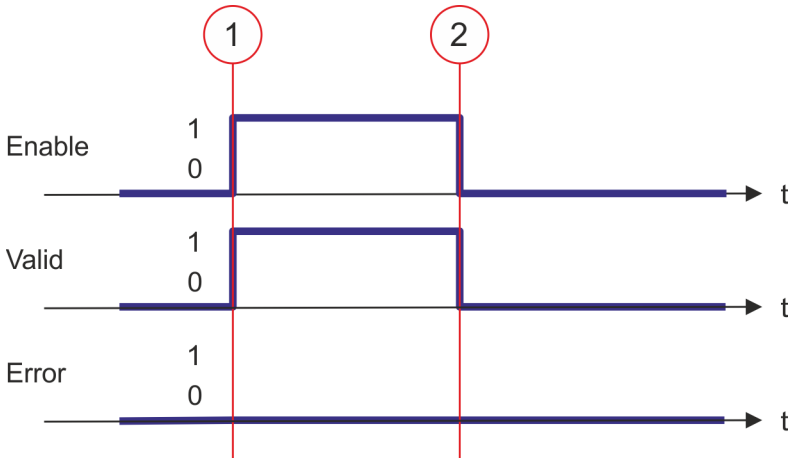
PLCopen-State

- Start des Auftrags in jedem PLCopen-State möglich.

Status der Achse ermitteln

Mit *Enable* gleich TRUE werden an den Ausgängen die Zusatzinformationen zur Achse geliefert.

Zustandsdiagramm der Bausteinparameter



(1) Zum Zeitpunkt (1) wird *Enable* = TRUE gesetzt. Damit liefert *Valid* den Wert TRUE und an den Ausgängen werden die Zusatzinformationen zur Achse angezeigt.
(2) Zum Zeitpunkt (2) wird *Enable* = FALSE gesetzt. Damit werden sämtliche Ausgänge gleich FALSE bzw. 0 gesetzt.

14.3.16 FB 719 - MC_ReadMotionState - Zustand Bewegungsauftrag lesen

Beschreibung

Mit MC_ReadMotionState wird der aktuelle Zustand des Bewegungsauftrags angezeigt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Enable	INPUT	BOOL	■ Zustand Bewegungsauftrag lesen – TRUE: Zustand Bewegungsauftrag wird kontinuierlich gelesen – FALSE: Alle Ausgänge werden gleich FALSE bzw. 0
Source	INPUT	Byte	Nur Source = 0 wird unterstützt; an den Ausgängen wird der Sollzustand des Bewegungsauftrags angezeigt.
Valid	OUTPUT	BOOL	■ Zustand gültig – TRUE: Die gelesene Zustand des Bewegungsauftrags ist gültig
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.

Parameter	Deklaration	Datentyp	Beschreibung
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
ConstantVelocity	OUTPUT	BOOL	■ Zustand Bewegungsauftrag: Geschwindigkeit – TRUE: Geschwindigkeit ist konstant
Accelerating	OUTPUT	BOOL	■ Zustand Bewegungsauftrag: Beschleunigung – TRUE: Achse wird beschleunigt; die Geschwindigkeit der Achse erhöht sich.
Decelerating	OUTPUT	BOOL	■ Zustand Bewegungsauftrag: Bremsvorgang – TRUE: Achse wird gebremst; die Geschwindigkeit der Achse wird geringer.
DirectionPositive	OUTPUT	BOOL	■ Zustand Bewegungsauftrag: Position zunehmend – TRUE: Die Position der Achse nimmt zu
DirectionNegative	OUTPUT	BOOL	■ Zustand Bewegungsauftrag: Position abnehmend – TRUE: Die Position der Achse nimmt ab

Verwendung

- Bausteinaufruf im:
  - OB 1
- Anwendbar auf:
  - Positionierungsachse
  - Drehzahlachse
  - Externer Geber
  - Virtuelle Achse

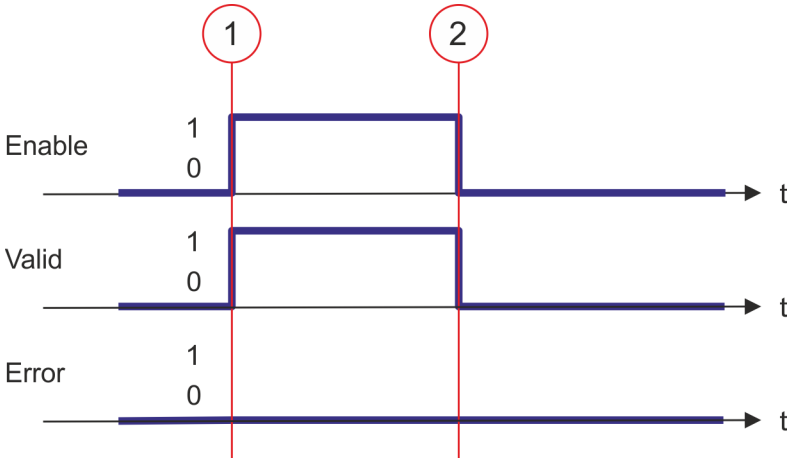
PLCopen-State

- Start des Auftrags in jedem PLCopen-State möglich.

Zustand des Bewegungsauftrags lesen

Mit *Enable* gleich TRUE wird an den Ausgängen der Zustand des Bewegungsauftrags der Achse geliefert.

Zustandsdiagramm der Bausteinparameter



(1) Zum Zeitpunkt (1) wird *Enable* = TRUE gesetzt. Damit liefert *Valid* den Wert TRUE und an den Ausgängen wird der Zustand des Bewegungsauftrags angezeigt.
(2) Zum Zeitpunkt (2) wird *Enable* = FALSE gesetzt. Damit werden sämtliche Ausgänge gleich FALSE bzw. 0 gesetzt.

14.3.17 FB 722 - MC_MoveSuperimposed - überlagerte Positionierung

Beschreibung

Mit dem Funktionsbaustein MC_MoveSuperImposed wird eine überlagerte relative Positionierung durchgeführt. Die Achse legt eine zusätzliche Wegstrecke zurück, welche Sie über *Distance* vorgeben können. Hierbei wird die aktuelle Bewegung nicht unterbrochen, sondern durch die zusätzliche Bewegung überlagert. Bei einer negativen *Distance* legt die Achse eine um diesen Betrag kürzere Strecke zurück.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	Mit einer Flanke 0-1 an *Execute* wird die überlagerte Positionierung gestartet
ContinuousUpdate	INPUT	BOOL	Parameter aktuell nicht unterstützt; Aufruf mit FALSE
Distance	INPUT	REAL	Relative Wegstrecke die aufgeholt werden soll. Ein positiver Wert bedeutet eine betragsmäßige Geschwindigkeitserhöhung, um diese Strecke zusätzlich zur unbeeinflussten Bewegung zurückzulegen. Ein negativer Wert bedeutet ein Abbremsen und Zurückfallen um diese Strecke [Anwendereinheiten].
VelocityDiff	INPUT	REAL	Maximale Geschwindigkeitsüberhöhung mit der gefahren werden soll [Anwendereinheiten/s^2]
Acceleration	INPUT	REAL	Beschleunigung (Energiesteigerung des Motors) [Anwendereinheiten/s^2]
Deceleration	INPUT	REAL	Verzögerung (Energieverringerung des Motors) [Anwendereinheiten/s^2]

Parameter	Deklaration	Datentyp	Beschreibung
Jerk	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Die überlagerte Positionierung ist aktiv
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
CoveredDistance	OUTPUT	REAL	Zeigt kontinuierlich die überlagerte Position [Anwendereinheiten], solange *Busy* gesetzt ist.

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE, solange ausgeführt, bis dieser abgearbeitet ist.*
- *Ein laufender Auftrag kann durch Abbruch des untergeordneten Auftrags abgebrochen werden.*
- *Solange MC_MoveSuperimposed aktiv ist, können Sie mit jedem Abbruch-Kommando die überlagerte und die untergeordnete Positionierung abbrechen.*
- *Ein laufender Auftrag kann durch einen neuen MC_MoveSuperimposed-Auftrag zur überlagerten Positionierung für die selbe Achse abgebrochen werden. Die untergeordnete Positionierung bleibt hiervon unberührt.*
- *MC_MoveSuperimposed bewirkt eine Änderung der Geschwindigkeit und, falls erforderlich, die Sollposition einer laufenden Bewegung in allen relevanten Zuständen*
- *Im PLCopen-State Standstill verhält sich der FB wie MC_MoveRelative*
- *Die Werte Acceleration und Deceleration sind zusätzliche Werte für die laufende Positionierung und keine Absolutwerte.*

Verwendung

- Bausteinaufruf im:
  - OB 61
- Anwendbar auf:
  - Positionierungsachse
  - Virtuelle Achse

PLCopen-State

- Start des Auftrags ist im PLCopen-State *Synchronized Motion*, *Standstill*, *Discrete Motion* und *Continuous Motion* möglich

Überlagerte Positionierung

Mit einer Flanke 0-1 an *Execute* wird die überlagerte Positionierung der Achse ausgeführt. Solange der Befehl abgearbeitet wird, zeigt *Busy* den Wert TRUE. Nach Abarbeitung des Befehls, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

14.3.18 FB 723 - MC_TouchProbe - Achsposition erfassen

Beschreibung

Dieser Baustein erfasst einmalig die Achsposition in Abhängigkeit eines Trigger-Signals. Das Trigger-Signal kann über die am Eingang *TriggerInput* angegebene Variable konfiguriert werden. Als Trigger-Signal kann z.B. ein Digitaleingang oder die Gebernullspur dienen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse.
TriggerInput	IN_OUT	MC_TRIGGER_REF	Referenz zum Trigger-Eingang. Struktur ■ .Probe – 01: TouchProbe-Register 1 – 02: TouchProbe-Register 2 ■ .TriggerSource – 00: Eingang – 00: Encoder Nullimpuls ■ .Triggermode – 00: SingleTrigger (fix) ■ .Reserved (0 fix)
Execute	INPUT	BOOL	Mit einer Flanke 0-1 an *Execute* wird die Erfassung der Achsposition aktiviert.
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Die Achsposition wurde erfasst.
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung.
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen.
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
RecordedPosition	OUTPUT	REAL	Erfasste Achsposition zum Zeitpunkt des Trigger-Signals [Anwendereinheiten].

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE, solange ausgeführt, bis dieser abgearbeitet ist. Die erfasste Achsposition wird dann für einen Zyklus am Ausgang RecordedPosition ausgegeben.* *Kap. 14.1.4 "Verhalten der Ein- und Ausgänge" Seite 481*
- *Damit der Befehl ausgeführt werden kann, muss die Kommunikation mit der Achse OK und der PLCopen-State ungleich Homing sein.*
- *Ein laufender Auftrag kann durch einen neuen MC_TouchProbe auf der gleichen Achse abgebrochen werden.*
- *Ein laufender Auftrag kann durch den Befehl MC_AbortTrigger abgebrochen werden.*
- *Ein laufender Auftrag kann durch den Befehl MC_Home abgebrochen werden.*

Verwendung

- Bausteinaufruf im:
  - OB 61
- Anwendbar auf:
  - Positionierungsachse

Achsposition erfassen

Mit einer Flanke 0-1 an *Execute* wir die Erfassung der Achsposition aktiviert. Solange der Befehl abgearbeitet wird, zeigt *Busy* den Wert TRUE. Nach Abarbeitung des Befehls, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der erfasste Wert wird in *RecordedPosition* ausgegeben.

14.3.19 FB 724 - MC_AbortTrigger - Achsposition erfassen abbrechen

Beschreibung

Dieser Baustein bricht die durch MC_TouchProbe gestartete Erfassung der Achsposition ab.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse.
TriggerInput	IN_OUT	MC_TRIGGER_REF	Referenz zum Trigger-Eingang. Struktur ■ .Probe – 01: TouchProbe-Register 1 – 02: TouchProbe-Register 2 ■ .TriggerSource – 00: Eingang – 00: Encoder Nullimpuls ■ .Triggermode – 00: SingleTrigger (fix) ■ .Reserved (0 fix)
Execute	INPUT	BOOL	Mit einer Flanke 0-1 an *Execute* wird die Erfassung der Achsposition abgebrochen.

Parameter	Deklaration	Datentyp	Beschreibung
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Die Erfassung der Achsposition wurde abgebrochen.
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung.
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Damit der Befehl ausgeführt werden kann, muss die Kommunikation mit der Achse OK sein.

Verwendung

- Bausteinaufruf im:
  – OB 61
- Anwendbar auf:
  – Positionierungsachse

Erfassung der Achsposition abbrechen

Mit einer Flanke 0-1 an *Execute* wir die Erfassung der Achsposition abgebrochen. Solange der Befehl abgearbeitet wird, zeigt *Busy* den Wert TRUE. Nach Abarbeitung des Befehls, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

14.3.20 FB 725 - MC_ReadBoolParameter - Boolean-Parameter von Achse lesen

Beschreibung

Mit MC_ReadBoolParameter wird der Parameter, der über die Parameter-Nummer festgelegt ist, vom Datentyp BOOL von der Achse gelesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Parameter Achse lesen – Flanke 0-1: Das Lesen des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher gelesen werden soll. ↳ *Kap. 14.6 "PLCopen Parameter" Seite 574*
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde ausgelesen
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung

Parameter	Deklaration	Datentyp	Beschreibung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
Value	OUTPUT	BOOL	Wert des gelesenen Parameters

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse
  – Drehzahlachse
  – Virtuelle Achse (nur Parameter-Nummer < 1000 zulässig)

PLCopen-State

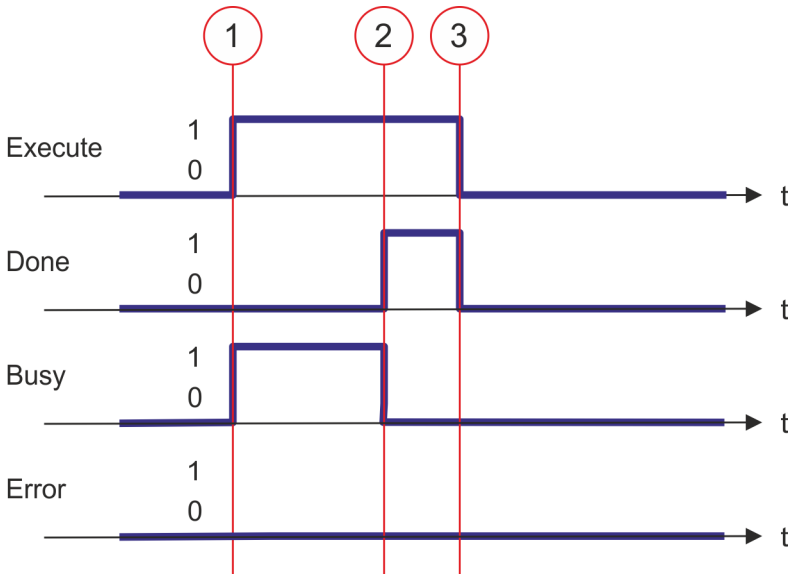
- Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse lesen

Mit einer Flanke 0-1 an *Execute* wird das Lesen des Parameters gestartet. Solange das Lesen des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter gelesen wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der Ausgang *Value* zeigt den Wert des Parameters an.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Lesen des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Lesen des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.21 FB 726 - MC_WriteBoolParameter - Boolean-Parameter an Achse schreiben

Beschreibung

Mit MC_WriteBoolParameter wird der Wert des Parameters, der über die Parameter-Nummer festgelegt ist, vom Datentyp BOOL zur Achse geschrieben.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Parameter Achse schreiben – Flanke 0-1: Das Schreiben des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher geschrieben werden soll. ↳ *Kap. 14.6 "PLCopen Parameter" Seite 574*
Value	INPUT	BOOL	Wert des geschriebenen Parameters
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde geschrieben
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung

Parameter	Deklaration	Datentyp	Beschreibung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse
  – Drehzahlachse
  – Virtuelle Achse (nur Parameter-Nummer < 1000 zulässig)

PLCopen-State

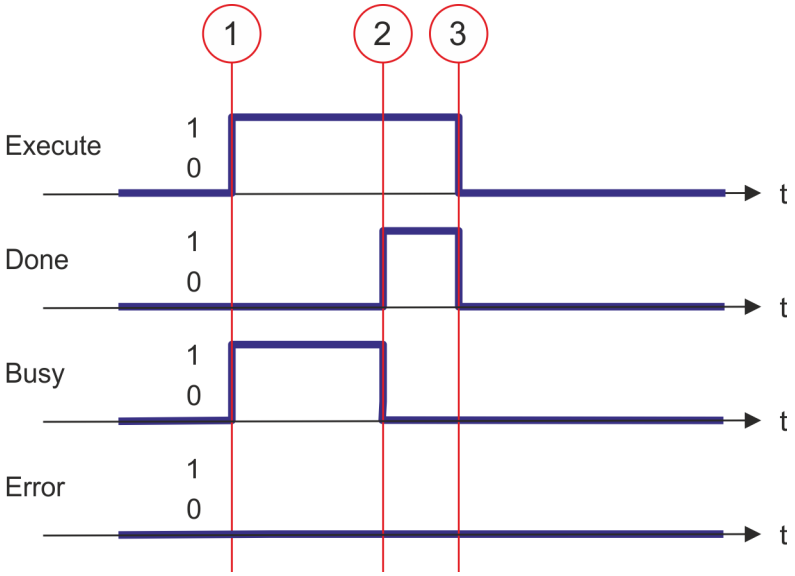
- Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse schreiben

Mit einer Flanke 0-1 an *Execute* wird das Schreiben des Parameters gestartet. Solange das Schreiben des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter geschrieben wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Schreiben des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Schreiben des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.22 FB 727 - VMC_ReadDWordParameter - Doppelwort-Parameter von Achse lesen

Beschreibung

Mit VMC_ReadDWordParameter wird der Parameter, der über die Parameter-Nummer festgelegt ist, vom Datentyp DWORD von der Achse gelesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Parameter Achse lesen – Flanke 0-1: Das Lesen des Parameters wird durchgeführt
ParameterNumber	INPUT	INT	Nummer des Parameters, welcher gelesen werden soll. ↳ *Kap. 14.6 "PLCopen Parameter" Seite 574*
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde ausgelesen
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung

Parameter	Deklaration	Datentyp	Beschreibung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
Value	OUTPUT	DWORD	Wert des gelesenen Parameters

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse
  – Drehzahlachse
  – Virtuelle Achse (nur Parameter-Nummer < 1000 zulässig)

PLCopen-State

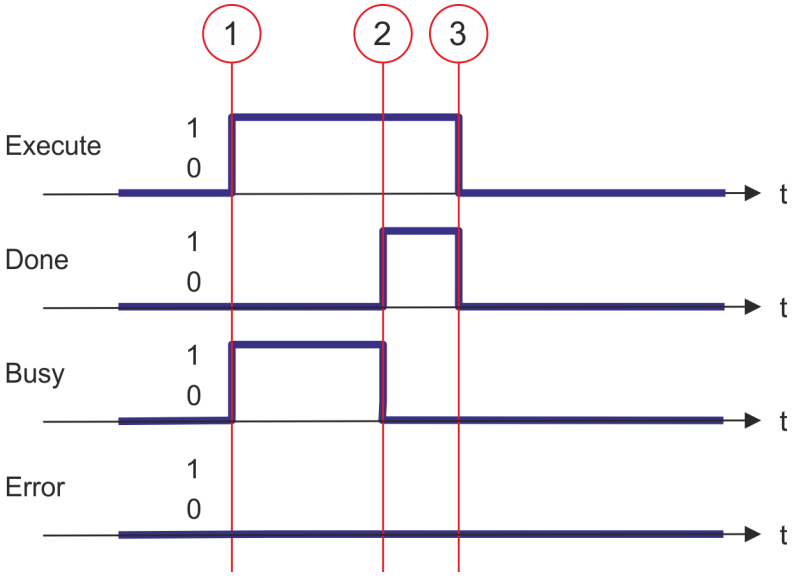
- Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse lesen

Mit einer Flanke 0-1 an *Execute* wird das Lesen des Parameters gestartet. Solange das Lesen des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter gelesen wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der Ausgang *Value* zeigt den Wert des Parameters an.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Lesen des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Lesen des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.23 FB 728 - VMC_WriteDWordParameter - Doppelwort-Parameter an Achse schreiben

Beschreibung

Mit VMC_WriteDWordParameter wird der Wert des Parameters, der über die Parameter-Nummer festgelegt ist, vom Datentyp DWORD zur Achse geschrieben.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Parameter Achse schreiben – Flanke 0-1: Das Schreiben des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher geschrieben werden soll. ↳ *Kap. 14.6 "PLCopen Parameter" Seite 574*
Value	INPUT	DWORD	Wert des geschriebenen Parameters
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde geschrieben
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung

Parameter	Deklaration	Datentyp	Beschreibung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse
  – Drehzahlachse
  – Virtuelle Achse (nur Parameter-Nummer < 1000 zulässig)

PLCopen-State

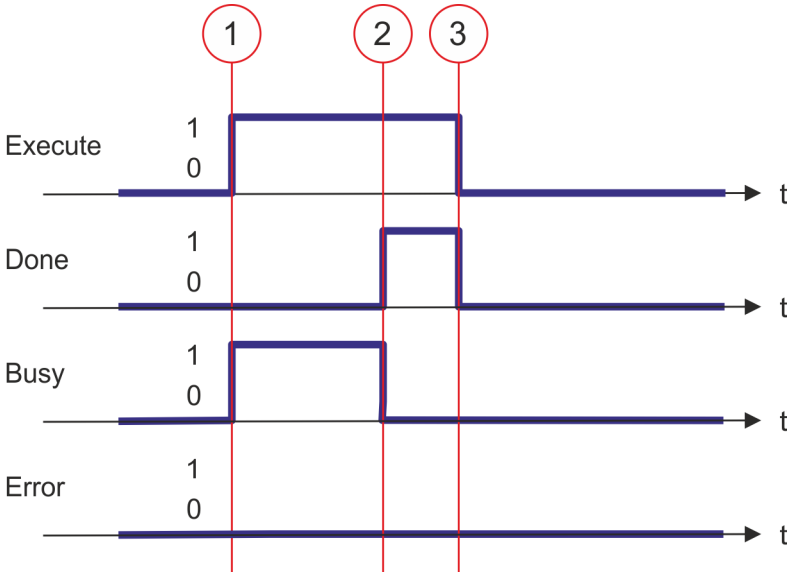
- Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse schreiben

Mit einer Flanke 0-1 an *Execute* wird das Schreiben des Parameters gestartet. Solange das Schreiben des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter geschrieben wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Schreiben des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Schreiben des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.24 FB 729 - VMC_ReadWordParameter - Wort-Parameter von Achse lesen

Beschreibung

Mit VMC_ReadWordParameter wird der Parameter, der über die Parameter-Nummer festgelegt ist, vom Datentyp WORD von der Achse gelesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Parameter Achse lesen – Flanke 0-1: Das Lesen des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher gelesen werden soll. ↳ *Kap. 14.6 "PLCopen Parameter" Seite 574*
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde ausgelesen
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung

Parameter	Deklaration	Datentyp	Beschreibung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
Value	OUTPUT	WORD	Wert des gelesenen Parameters

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse
  – Drehzahlachse
  – Virtuelle Achse (nur Parameter-Nummer < 1000 zulässig)

PLCopen-State

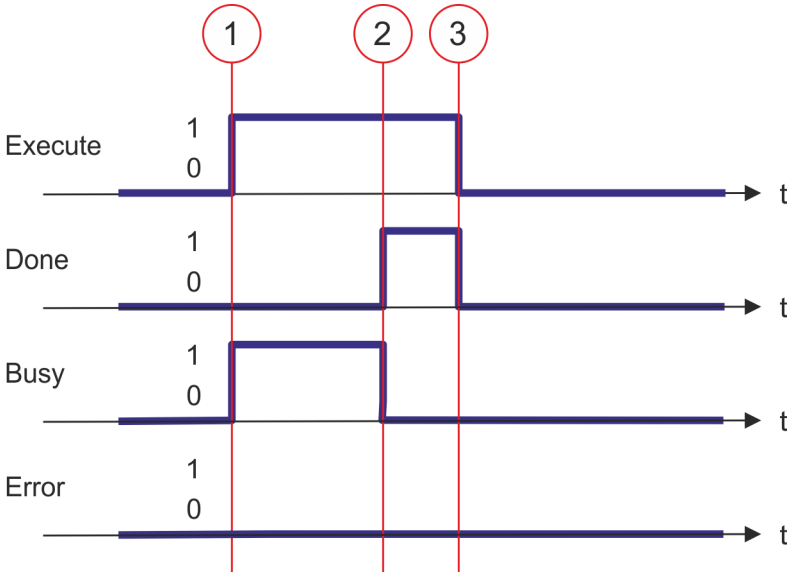
- Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse lesen

Mit einer Flanke 0-1 an *Execute* wird das Lesen des Parameters gestartet. Solange das Lesen des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter gelesen wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der Ausgang *Value* zeigt den Wert des Parameters an.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Lesen des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Lesen des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.25 FB 730 - VMC_WriteWordParameter - Wort-Parameter an Achse schreiben

Beschreibung

Mit VMC_WriteWordParameter wird der Wert des Parameters, der über die Parameter-Nummer festgelegt ist, vom Datentyp WORD zur Achse geschrieben.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Parameter Achse schreiben – Flanke 0-1: Das Schreiben des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher geschrieben werden soll. ↳ *Kap. 14.6 "PLCopen Parameter" Seite 574*
Value	INPUT	WORD	Wert des geschriebenen Parameters
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde geschrieben
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung

Parameter	Deklaration	Datentyp	Beschreibung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse
  – Drehzahlachse
  – Virtuelle Achse (nur Parameter-Nummer < 1000 zulässig)

PLCopen-State

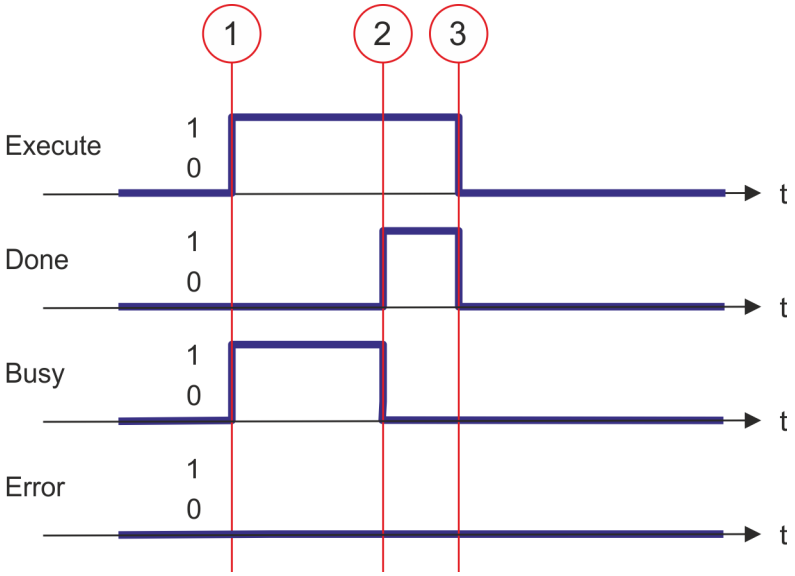
- Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse schreiben

Mit einer Flanke 0-1 an *Execute* wird das Schreiben des Parameters gestartet. Solange das Schreiben des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter geschrieben wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Schreiben des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Schreiben des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.26 FB 731 - VMC_ReadByteParameter - Byte-Parameter von Achse lesen

Beschreibung

Mit VMC_ReadByteParameter wird der Parameter, der über die Parameter-Nummer festgelegt ist, vom Datentyp BYTE von der Achse gelesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Parameter Achse lesen – Flanke 0-1: Das Lesen des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher gelesen werden soll. ↳ *Kap. 14.6 "PLCopen Parameter" Seite 574*
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde ausgelesen
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung

Parameter	Deklaration	Datentyp	Beschreibung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
Value	OUTPUT	BYTE	Wert des gelesenen Parameters

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse
  – Drehzahlachse
  – Virtuelle Achse (nur Parameter-Nummer < 1000 zulässig)

PLCopen-State

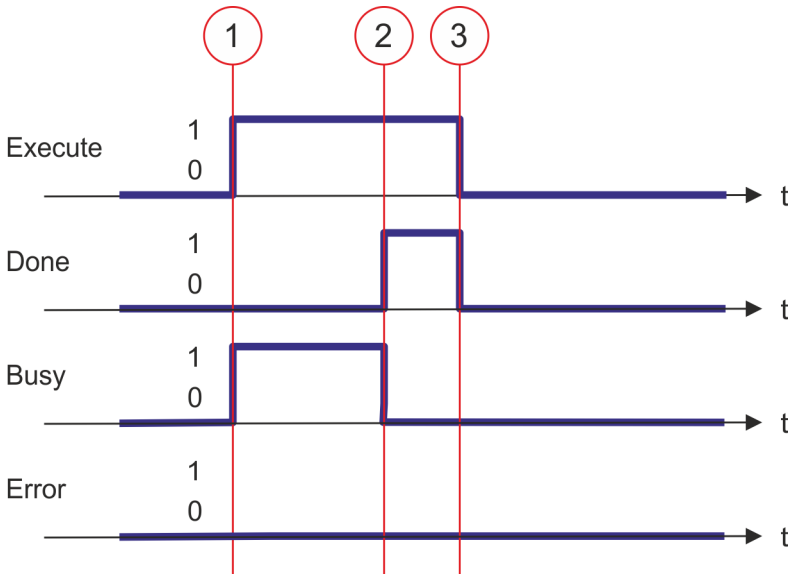
- Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse lesen

Mit einer Flanke 0-1 an *Execute* wird das Lesen des Parameters gestartet. Solange das Lesen des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter gelesen wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der Ausgang *Value* zeigt den Wert des Parameters an.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Lesen des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Lesen des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.27 FB 732 - VMC_WriteByteParameter - Byte-Parameter an Achse schreiben

Beschreibung

Mit VMC_WriteByteParameter wird der Wert des Parameters, der über die Parameter-Nummer festgelegt ist, vom Datentyp BYTE zur Achse geschrieben.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Parameter Achse schreiben – Flanke 0-1: Das Schreiben des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher geschrieben werden soll. ↳ *Kap. 14.6 "PLCopen Parameter" Seite 574*
Value	INPUT	BYTE	Wert des geschriebenen Parameters
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde geschrieben
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung

Parameter	Deklaration	Datentyp	Beschreibung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse
  – Drehzahlachse
  – Virtuelle Achse (nur Parameter-Nummer < 1000 zulässig)

PLCopen-State

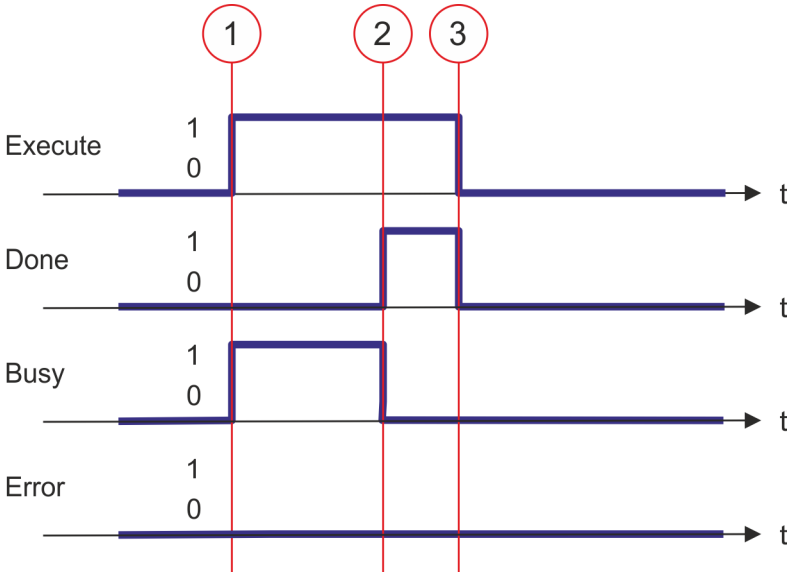
- Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse schreiben

Mit einer Flanke 0-1 an *Execute* wird das Schreiben des Parameters gestartet. Solange das Schreiben des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter geschrieben wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Schreiben des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Schreiben des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.28 FB 733 - VMC_ReadDriveParameter - Antriebsparameter lesen

Beschreibung

Mit dem VMC_ReadDriveParameter wird ein Parameterwert aus dem angebundenen Antrieb gelesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Antriebsparameter lesen – Flanke 0-1: Das Lesen des Antriebsparameters wird durchgeführt.
Index	INPUT	WORD	Index des Antriebsparameters
Subindex	INPUT	BYTE	Subindex des Antriebsparameters
Length	INPUT	BYTE	Datenlänge ■ 1: BYTE ■ 2: WORD ■ 4: DWORD
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde ausgelesen

Parameter	Deklaration	Datentyp	Beschreibung
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
Value	OUTPUT	DWORD	Wert des gelesenen Parameters

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse
  – Drehzahlachse

PLCopen-State

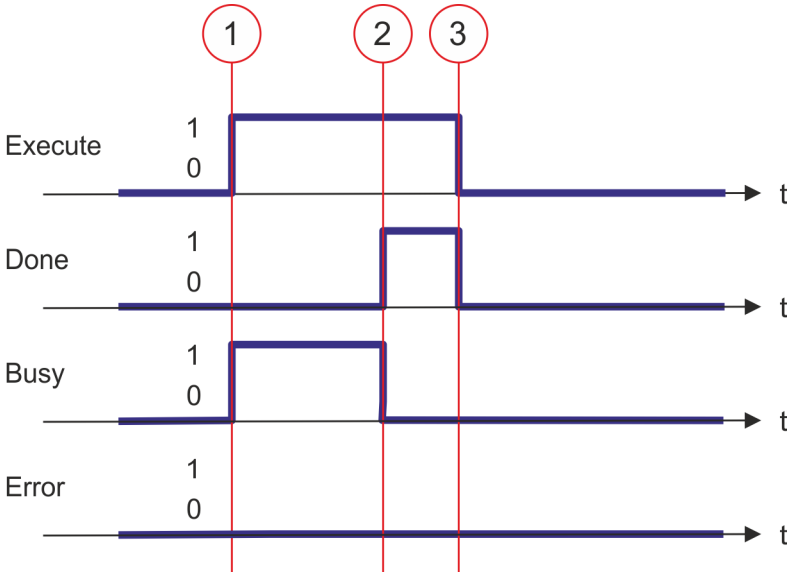
- Start des Auftrags in jedem PLCopen-State möglich.

Antriebsparameter lesen

Mit einer Flanke 0-1 an *Execute* wird das Lesen des Antriebsparameters gestartet. Solange das Lesen des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter gelesen wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der Ausgang *Value* zeigt den Wert des Parameters an.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Lesen des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Lesen des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.29 FB 734 - VMC_WriteDriveParameter - Antriebsparameter schreiben

Beschreibung

Mit dem VMC_WriteDriveParameter wird ein Parameterwert in den angebundenen Antrieb geschrieben.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Execute	INPUT	BOOL	■ Antriebsparameter schreiben – Flanke 0-1: Das Schreiben des Antriebsarameters wird durchgeführt.
Index	INPUT	WORD	Index des Antriebsparameters
Subindex	INPUT	BYTE	Subindex des Antriebsparameters
Length	INPUT	BYTE	Datenlänge ■ 1: BYTE ■ 2: WORD ■ 4: DWORD
Value	INPUT	DWORD	Wert des geschriebenen Parameters

Parameter	Deklaration	Datentyp	Beschreibung
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde ausgelesen
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse
  – Drehzahlachse

PLCopen-State

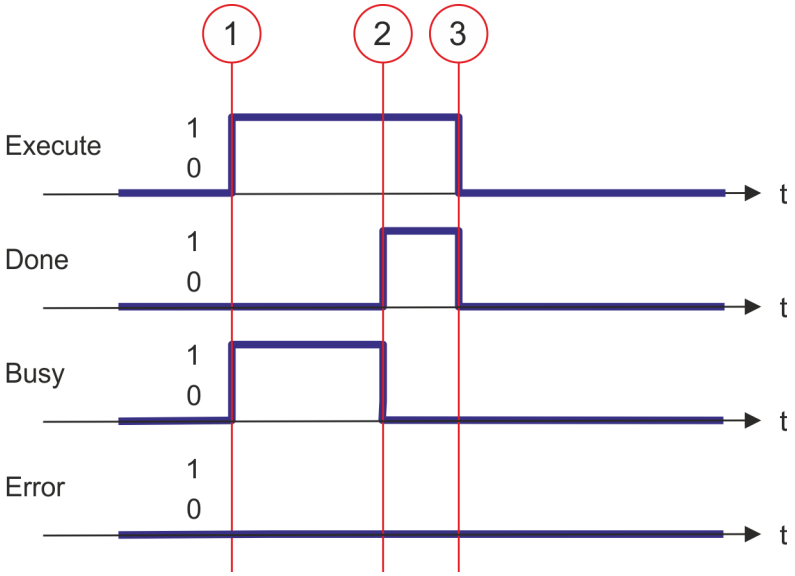
- Start des Auftrags in jedem PLCopen-State möglich.

Antriebsparameter schreiben

Mit einer Flanke 0-1 an *Execute* wird das Schreiben des Parameters gestartet. Solange das Schreiben des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter geschrieben wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Schreiben des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Schreiben des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.3.30 FB 735 - VMC_HomeInit_LimitSwitch - Initialisierung Referenzfahrt auf Endschalter

Beschreibung Dieser Baustein initialisiert die Referenzfahrt (Homing) auf den Endschalter.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	UDT8192	Referenz zur Achse.
Execute	INPUT	BOOL	■ Initialisierung der Referenzfahrt Methode – Flanke 0-1: Werte der Eingangsparameter werden übernommen und die Initialisierung der Referenzfahrt Methode gestartet.
Direction	INPUT	BOOL	■ Richtung der Referenzfahrt – TRUE: auf positiven Endschalter – FALSE: auf negativen Endschalter
Velocity-SearchSwitch	INPUT	REAL	Geschwindigkeit für die Suche nach dem Schalter in [Anwendereinheiten/s]
VelocitySearch-Zero	INPUT	REAL	Geschwindigkeit für die Suche nach dem Index in [Anwendereinheiten/s]
Acceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2]
Done	OUTPUT	BOOL	■ Status – TRUE: Initialisierung wurde ohne Fehler beendet.
Busy	OUTPUT	BOOL	■ Status – TRUE: Initialisierung ist aktiv

Parameter	Deklaration	Datentyp	Beschreibung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse

Initialisierung Referenzfahrt auf Endschalter

Mit einer Flanke 0-1 an *Execute* werden die Werte der Eingangsparameter übernommen und die Initialisierung der Referenzfahrt Methode gestartet. So lange die Initialisierung aktiv ist, wird der Ausgang *Busy* auf True gesetzt. Ist die Initialisierung ohne Fehler beendet worden, wird der Ausgang *Done* auf True gesetzt. Tritt bei der Initialisierung ein Fehler auf, wird der Ausgang *Error* auf True gesetzt und am Ausgang *ErrorID* eine Fehlernummer ausgegeben.

Initialisierung der Referenzfahrt Methode

1. Überprüfen der Kommunikation zur Achse.
2. Prüfen auf erlaubte PLCopen Zustände.
3. Prüfung der Eingangswerte:
   - Eingang VelocitySearchSwitch [UserUnits] > 0.0
   - VelocitySearchSwitch [InternalUnits] > 0
   - VelocitySearchSwitch [InternalUnits] ≤ VelocityMax
   - Eingang VelocitySearchZero [UserUnits] > 0.0
   - VelocitySearchZero [InternalUnits] > 0
   - VelocitySearchZero [InternalUnits] ≤ VelocityMax
   - Eingang Acceleration [UserUnits] > 0.0
   - Acceleration [InternalUnits] > 0
   - Acceleration [InternalUnits] ≤ AccelerationMax
4. Übertragung der Antriebsparameter:
   - "Homing Method" Referenzfahrtmethode in Abhängigkeit vom Eingang "Direction"
     Siehe Tabelle unten!
   - "Homing Speed during search for switch" [Inc/s]
     Geschwindigkeit für die Schaltersuche
   - "Homing Speed during search for zero" [Inc/s]
     Geschwindigkeit für die Indexsuche
   - "Homing Acceleration" [Inc/s^2]
     Anfahr- und Bremsbeschleunigung für die Referenzfahrt

Homing Method	Direction
1	false
2	true

14.3.31 FB 736 - VMC_HomeInit_HomeSwitch - Initialisierung Referenzfahrt auf Referenzschalter

Beschreibung Dieser Baustein initialisiert die Referenzfahrt (Homing) auf den Referenzschalter.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
AXIS	IN_OUT	UDT8192	Referenz zur Achse.
Execute	INPUT	BOOL	■ Initialisierung der Referenzfahrt Methode – Flanke 0-1: Werte der Eingangsparameter werden übernommen und die Initialisierung der Referenzfahrt Methode gestartet.
InitialDirection	INPUT	BOOL	■ Startrichtung der Referenzfahrt – TRUE: auf positiven Endschalter – FALSE: auf negativen Endschalter
WithIndexPulse	INPUT	BOOL	■ Referenzfahrt – TRUE: mit Nullimpuls – FALSE: ohne Nullimpuls
OnRisingEdge	INPUT	BOOL	■ Flanke Referenzschalter – TRUE: Flanke 0-1 – FALSE: Flanke 1-0
SameDirIndex-Pulse	INPUT	BOOL	■ Nullimpulssuche – TRUE: Nach dem Erkennen des Referenzschalters ohne Richtungswechsel den Nullimpuls suchen. – FALSE: Nach dem Erkennen des Referenzschalters einen Richtungswechsel zur Nullimpulssuche durchführen.
Velocity-SearchSwitch	INPUT	REAL	Geschwindigkeit für die Schaltersuche in [Anwendereinheiten/s]
VelocitySearch-Zero	INPUT	REAL	Geschwindigkeit für die Indexsuche in [Anwendereinheiten/s]
Acceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2]
Done	OUTPUT	BOOL	■ Status – TRUE: Initialisierung wurde ohne Fehler beendet.
Busy	OUTPUT	BOOL	■ Status – TRUE: Initialisierung ist aktiv
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse

Initialisierung Referenzfahrt auf Referenzschalter

Mit einer Flanke 0-1 an *Execute* werden die Werte der Eingangsparameter übernommen und die Initialisierung der Referenzfahrt Methode gestartet. So lange die Initialisierung aktiv ist, wird der Ausgang *Busy* auf True gesetzt. Ist die Initialisierung ohne Fehler beendet worden, wird der Ausgang *Done* auf True gesetzt. Tritt bei der Initialisierung ein Fehler auf, wird der Ausgang *Error* auf True gesetzt und am Ausgang *ErrorID* eine Fehlernummer ausgegeben.

Initialisierung der Referenzfahrt Methode

1. Überprüfen der Kommunikation zur Achse.
2. Prüfen auf erlaubte PLCopen Zustände.
3. Prüfung der Eingangswerte:
   - Eingang VelocitySearchSwitch [UserUnits] > 0.0
   - VelocitySearchSwitch [InternalUnits] > 0
   - VelocitySearchSwitch [InternalUnits] ≤ VelocityMax
   - Eingang VelocitySearchZero [UserUnits] > 0.0
   - VelocitySearchZero [InternalUnits] > 0
   - VelocitySearchZero [InternalUnits] ≤ VelocityMax
   - Eingang Acceleration [UserUnits] > 0.0
   - Acceleration [InternalUnits] > 0
   - Acceleration [InternalUnits] ≤ AccelerationMax
4. Übertragung der Antriebsparameter:
   - "Homing Method" Referenzfahrtmethode in Abhängigkeit vom Eingang "Direction"
     Siehe Tabelle unten!
   - "Homing Speed during search for switch" [Inc/s]
     Geschwindigkeit für die Schaltersuche
   - "Homing Speed during search for zero" [Inc/s]
     Geschwindigkeit für die Indexsuche
   - "Homing Acceleration" [Inc/s^2]
     Anfahr- und Bremsbeschleunigung für die Referenzfahrt

Homing Method	InitialDirection	WithIndexPulse	OnRisingEdge	SameDirIndexPulse
7	positive	true	true	false
8	positive	true	true	true
9	positive	true	false	false
10	positive	true	false	true
11	negative	true	true	false
12	negative	true	true	true
13	negative	true	false	false
14	negative	true	false	true
24	positive	false	true	false

Homing Method	InitialDirection	WithIndexPulse	OnRisingEdge	SameDirIndexPulse
24	positive	false	true	true
24	positive	false	false	false
24	positive	false	false	true
28	negative	false	true	false
28	negative	false	true	true
28	negative	false	false	false
28	negative	false	false	true

14.3.32 FB 737 - VMC_HomeInit_ZeroPulse - Initialisierung Referenzfahrt auf Null Impuls

Beschreibung Dieser Baustein initialisiert die Referenzfahrt (Homing) auf Null Impuls.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
AXIS	IN_OUT	UDT8192	Referenz zur Achse.
Execute	INPUT	BOOL	■ Initialisierung der Referenzfahrt Methode – Flanke 0-1: Werte der Eingangsparameter werden übernommen und die Initialisierung der Referenzfahrt Methode gestartet.
Direction	INPUT	BOOL	■ Richtung der Referenzfahrt – TRUE: Positive Richtung – FALSE: Negative Richtung
VelocitySearch-Zero	INPUT	REAL	Geschwindigkeit für die Indexsuche in [Anwendereinheiten/s]
Acceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2]
Done	OUTPUT	BOOL	■ Status – TRUE: Initialisierung wurde ohne Fehler beendet.
Busy	OUTPUT	BOOL	■ Status – TRUE: Initialisierung ist aktiv
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  - OB 1
- Anwendbar auf:
  - Positionierungsachse

Initialisierung Referenzfahrt auf Nullimpuls

Mit einer Flanke 0-1 an *Execute* werden die Werte der Eingangsparameter übernommen und die Initialisierung der Referenzfahrt Methode gestartet. So lange die Initialisierung aktiv ist, wird der Ausgang *Busy* auf True gesetzt. Ist die Initialisierung ohne Fehler beendet worden, wird der Ausgang *Done* auf True gesetzt. Tritt bei der Initialisierung ein Fehler auf, wird der Ausgang *Error* auf True gesetzt und am Ausgang *ErrorID* eine Fehlernummer ausgegeben.

Initialisierung der Referenzfahrt Methode

1. Überprüfen der Kommunikation zur Achse.
2. Prüfen auf erlaubte PLCopen Zustände.
3. Prüfung der Eingangswerte:
   - Eingang VelocitySearchZero [UserUnits] > 0.0
   - VelocitySearchZero [InternalUnits] > 0
   - VelocitySearchZero [InternalUnits] ≤ VelocityMax
   - Eingang Acceleration [UserUnits] > 0.0
   - Acceleration [InternalUnits] > 0
   - Acceleration [InternalUnits] ≤ AccelerationMax
4. Übertragung der Antriebsparameter:
   - "Homing Method" Referenzfahrtmethode in Abhängigkeit vom Eingang "Direction"
     Siehe Tabelle unten!
   - "Homing Speed during search for switch" [Inc/s]
     Geschwindigkeit für die Schaltersuche
   - "Homing Speed during search for zero" [Inc/s]
     Geschwindigkeit für die Indexsuche
   - "Homing Acceleration" [Inc/s^2]
     Anfahr- und Bremsbeschleunigung für die Referenzfahrt

Homing Method	Direction
33	false
34	true

14.3.33 FB 738 - VMC_HomeInit_SetPosition - Initialisierung Referenzfahrt auf aktuelle Position

Beschreibung

Dieser Baustein initialisiert die Referenzfahrt (Homing) auf die aktuelle Position.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
AXIS	IN_OUT	UDT8192	Referenz zur Achse.
Execute	INPUT	BOOL	■ Initialisierung der Referenzfahrt Methode – Flanke 0-1: Werte der Eingangsparameter werden übernommen und die Initialisierung der Referenzfahrt Methode gestartet.
Done	OUTPUT	BOOL	■ Status – TRUE: Initialisierung wurde ohne Fehler beendet.
Busy	OUTPUT	BOOL	■ Status – TRUE: Initialisierung ist aktiv
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  – OB 1
- Anwendbar auf:
  – Positionierungsachse

Initialisierung Referenzfahrt auf Endschalter

Mit einer Flanke 0-1 an *Execute* werden die Werte der Eingangsparameter übernommen und die Initialisierung der Referenzfahrt Methode gestartet. So lange die Initialisierung aktiv ist, wird der Ausgang *Busy* auf True gesetzt. Ist die Initialisierung ohne Fehler beendet worden, wird der Ausgang *Done* auf True gesetzt. Tritt bei der Initialisierung ein Fehler auf, wird der Ausgang *Error* auf True gesetzt und am Ausgang *ErrorID* eine Fehlernummer ausgegeben.

Initialisierung der Referenzfahrt Methode

1. Überprüfen der Kommunikation zur Achse.
2. Prüfen auf erlaubte PLCopen Zustände.
3. Übertragung der Antriebsparameter:
   - "Homing Method" = 35

14.3.34 FB 739 - MC_TorqueControl - Drehmomentsteuerung

Beschreibung

Mit diesem Baustein geben Sie ein Drehmoment vor.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
AXIS	IN_OUT	UDT8192	Referenz zur Achse.
Execute	INPUT	BOOL	■ Achse mit konstantem Drehmoment verfahren – Flanke 0-1: Das Verfahren der Achse mit konstantem Drehmoment wird gestartet

Parameter	Deklaration	Datentyp	Beschreibung
ContinuousUpdate	INPUT	BOOL	Parameter aktuell nicht unterstützt; Aufruf mit FALSE
Torque	INPUT	REAL	Drehmomentwert in [Anwendereinheiten]
TorqueRamp	INPUT	REAL	maximale Zeit, innerhalb der der Drehmomentwert erreicht werden soll in [Anwendereinheiten/s]
Velocity	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
Acceleration	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
Deceleration	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
Jerk	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
Direction	INPUT	BYTE	Drehrichtung ■ 01h: positiv ■ 02h: negativ
BufferMode	INPUT	BYTE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0
InTorque	OUTPUT	BOOL	■ Drehmoment – TRUE: Das vorgegebene Drehmoment wurde erreicht
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung: Drehmoment ist noch nicht erreicht
Active	OUTPUT	BOOL	■ Status – TRUE: Baustein hat die Kontrolle über die Achse
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  - OB 1
  - OB 61
- Anwendbar auf:
  - Positionierungsachse

PLCopen-State

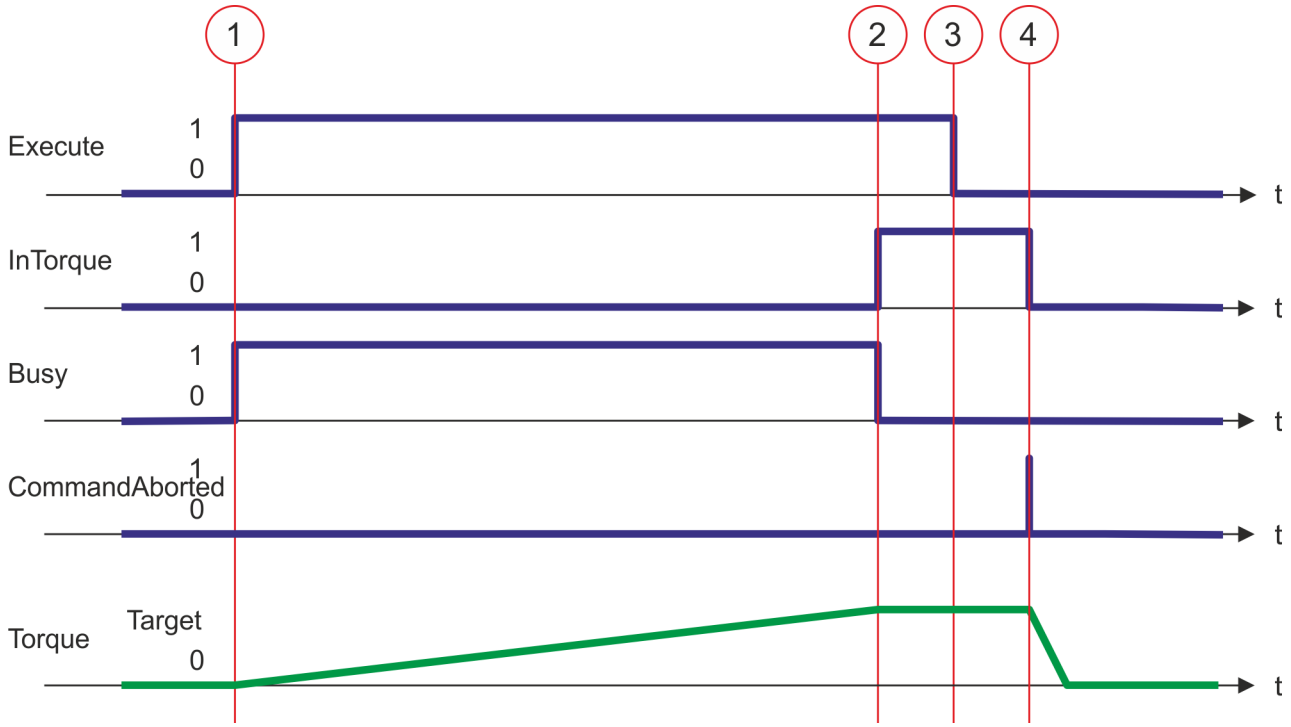
- Start des Auftrags in den PLCopen-States *Standstill*, *Discrete Motion*, *Synchronized Motion* und *Continuous Motion* möglich.
- MC_TorqueControl führt die Achse in den PLCopen-State *Continuous Motion* über.

Achse mit Drehmoment-vorgabe verfahren

Mit einer Flanke 0-1 an *Execute* wird das Verfahren der Achse mit Drehmomentvorgabe gestartet. Solange die Drehmomentvorgabe nicht erreicht ist, zeigt *Busy* den Wert TRUE und *InTorque* den Wert FALSE. Ist die Drehmomentvorgabe erreicht, wird *Busy* gleich FALSE und *InTorque* gleich TRUE. Die Achse wird mit diesem Drehmoment konstant weiter verfahren.

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt, auch wenn die Drehmomentvorgabe erreicht wurde.*
- *Ein laufender Auftrag kann durch einen anderen Bewegungsauftrag (z.B. MC_MoveAbsolute) abgebrochen werden.*

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an Execute zum Zeitpunkt (1) wird das Verfahren der Achse mit Drehmomentvorgabe gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) erreicht die Achse die Drehmoment-vorgabe und *Busy* liefert den Wert FALSE und *InTorque* den Wert TRUE.
(3) Das Rücksetzen von Execute auf FALSE zum Zeit-punkt (3) hat keine Auswirkung auf die Achse. Die Achse wird weiterhin konstant mit der Drehmomentvor-gabe verfahren und *InTorque* liefert weiterhin den Wert TRUE.
(4) Zum Zeitpunkt (4) wird der MC_TorqueControl durch einen MC_Halt-Auftrag abgebrochen. Die Achse wird bis zum Halt abgebremst.

14.3.35 FB 740 - MC_ReadActualTorque - Drehmoment der Achse lesen

Beschreibung

Mit MC_ReadActualTorque wird das aktuelle Drehmoment der Achse gelesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse
Enable	INPUT	BOOL	■ Drehmoment Achse lesen – TRUE: Das Drehmoment der Achse wird kontinuierlich gelesen – FALSE: Alle Ausgänge werden gleich FALSE bzw. 0
Valid	OUTPUT	BOOL	■ Drehmoment gültig – TRUE: Das gelesene Drehmoment ist gültig
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
Torque	OUTPUT	REAL	Drehmoment der Achse in [Anwendereinheiten/s]

Verwendung

- Bausteinaufruf im:
  – OB 1
  – OB 61
- Anwendbar auf:
  – Positionierungsachse

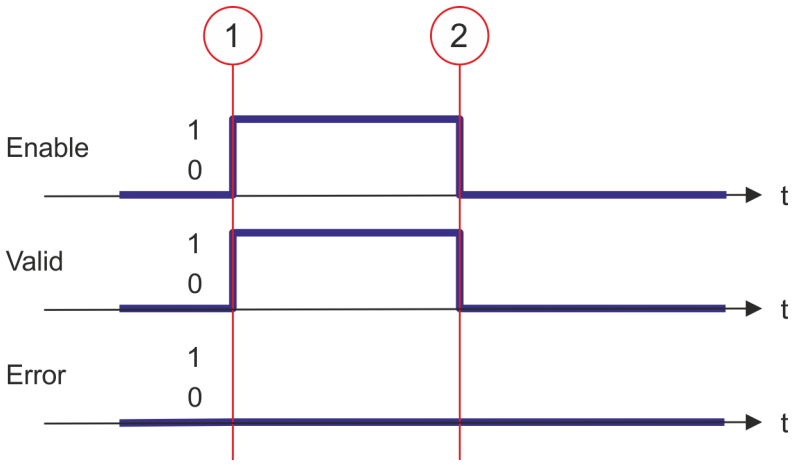
PLCopen-State

- Start des Auftrags in jedem PLCopen-State möglich.

Drehmoment der Achse lesen

Mit *Enable* gleich TRUE wird das aktuelle Drehmoment der Achse ermittelt und unter *Torque* abgelegt.

Zustandsdiagramm der Bausteinparameter



(1) Zum Zeitpunkt (1) wird *Enable* = TRUE gesetzt. Damit liefert *Valid* den Wert TRUE und am Ausgang *Torque* wird das aktuelle Drehmoment der Achse angezeigt.

(2) Zum Zeitpunkt (2) wird *Enable* = FALSE gesetzt. Damit werden sämtliche Ausgänge gleich FALSE bzw. 0 gesetzt.

14.4 Multi axis - Mehrfachachsen

14.4.1 FB 706 - MC_CamIn - Kopplung Master-Slave-Achse über Kurvenscheibe

14.4.1.1 Beschreibung

Mit MC_CamIn wird eine Kopplung zwischen Slave-Achse und Master-Achse über eine bestimmte Kurvenscheibe eingerichtet.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Master	IN_OUT	MC_AXIS_REF	Referenz zur Master-Achse
Slave	IN_OUT	MC_AXIS_REF	Referenz zur Slave-Achse
Execute	INPUT	BOOL	Mit einer Flanke 0-1 an *Execute* wird das Koppeln der Slave-Achse an die Master-Achse gestartet bzw. auf ein anderes Kurvenscheibenprofil umgeschaltet
ContinuousUpdate	INPUT	BOOL	Parameter aktuell nicht unterstützt; Aufruf mit FALSE
MasterOffset	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
SlaveOffset	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
MasterScaling	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
SlaveScaling	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
StartMode	INPUT	BYTE	■ Start-Position zum Zeitpunkt der Kopplung – 1: Absolute: Dem aktuellen Masterwert wird der entsprechende Absolutwert der Kurvenscheibe zugrunde gelegt. – 2: Relative: Dem aktuellen Masterwert wird der Anfang der Kurvenscheibe zugrunde gelegt.
ActivationMode	INPUT	BYTE	■ Aktivierungsmodus – 1: Kopplung erfolgt sofort – 2: Kopplung erfolgt mit dem nächsten Zyklus der Kurvenscheibe
BufferMode	INPUT	BYTE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0
CamTableDB	INPUT	INT	Datenbaustein-Nummer mit Kurvenscheibenprofil
InSync	OUTPUT	BOOL	■ Synchronisation – TRUE: Synchronisation erreicht
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung: Synchronisation ist noch nicht erreicht
Active	OUTPUT	BOOL	■ Status – TRUE: Baustein hat die Kontrolle über die Achse
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.

Parameter	Deklaration	Datentyp	Beschreibung
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
EndOfProfile	OUTPUT	BOOL	■ Ende Kurvenscheibenprofil – TRUE: Ende des Kurvenscheibenprofils wurde erreicht (TRUE nur für einen Aufruf-Zyklus)

Verwendung

- Bausteinaufruf im:
  - OB 61
- Anwendbar auf:
  - Positionierungsachse
  - Externer Encoder
  - Virtuelle Achse

PLCopen-State

- Master-Achse
  - Start des Auftrags in den PLCopen-States *Standstill*, *Discrete Motion*, *Synchronized Motion*, *Continuous Motion* und *Stopping* möglich.
- Slave-Achse
  - Start des Auftrags in den PLCopen-States *Standstill*, *Discrete Motion*, *Synchronized Motion* und *Continuous Motion* möglich.
- MC_CamIn führt die Achse in den PLCopen-State *Synchronized Motion* über.

Zulässige Kombinationen von *StartMode* und *ActivationMode*

Kurvenscheibe	StartMode	ActivationMode	Zulässig
Koppeln	relative	sofort	Ja
Koppeln	relative	nächster Zyklus der Kurvenscheibe	Nein
Koppeln	absolute	sofort	Ja
Koppeln	absolute	nächster Zyklus der Kurvenscheibe	Nein
Umschalten	relative	sofort	Ja
Umschalten	relative	nächster Zyklus der Kurvenscheibe	Ja
Umschalten	absolute	sofort	Ja
Umschalten	absolute	nächster Zyklus der Kurvenscheibe	Nein

VORSICHT!

Das Umschalten mit einem unterschiedlichen *StartMode* ist nicht zulässig!

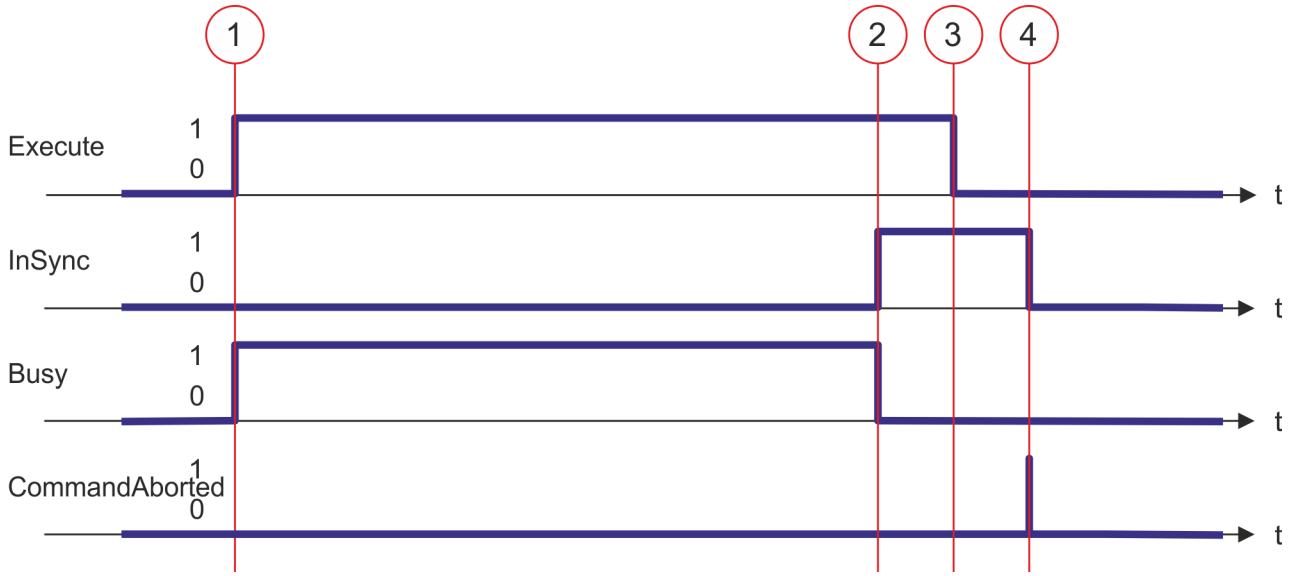
Eine gegenseitige Verriegelung zweier oder mehrerer Achsen kann zu einem Aufschwingen des Systems führen!

Slave-Achse an Master-Achse koppeln

Mit einer Flanke 0-1 an *Execute* wird das Ankoppeln der Slave-Achse an die Master-Achse gestartet. Solange die Slave-Achse auf die über das Kurvenscheibenprofil festgelegte Position synchronisiert wird, wird *Busy* gleich TRUE und *InSync* = FALSE geliefert. Wird die Position erreicht, wird *Busy* gleich FALSE und *InSync* = TRUE geliefert.

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt, auch wenn die Synchronisation erreicht ist.*
- *Ein laufender Auftrag kann durch einen anderen Bewegungsauftrag (z.B. MC_MoveAbsolute) abgebrochen werden. Anschließend sind Master- und Slave-Achse entkoppelt.*

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Koppeln der Slave-Achse an die Master-Achse gestartet, *Busy* liefert den Wert TRUE. Die Slave-Achse wird an die Position entsprechend des Kurvenscheibenprofils verfahren.
(2) Zum Zeitpunkt (2) erreicht die Slave-Achse, die über das Kurvenscheibenprofil festgelegte Position und *Busy* liefert den Wert FALSE und *InSync* den Wert TRUE.
(3) Das Rücksetzen von *Execute* auf FALSE zum Zeitpunkt (3) hat keine Auswirkung auf die Slave-Achse. Die Slave-Achse ist weiterhin an die Master-Achse über das Kurvenscheibenprofil gekoppelt.
(4) Zum Zeitpunkt (4) wird die Slave-Achse durch einen MC_CamOut-Auftrag von der Master-Achse abgekoppelt. Die Slave-Achse wird weiterhin mit konstanter Geschwindigkeit verfahren, bis ein weiterer Bewegungsauftrag gestartet wird.

14.4.1.2 Beispiele mit *StartMode* = *"Relative"*

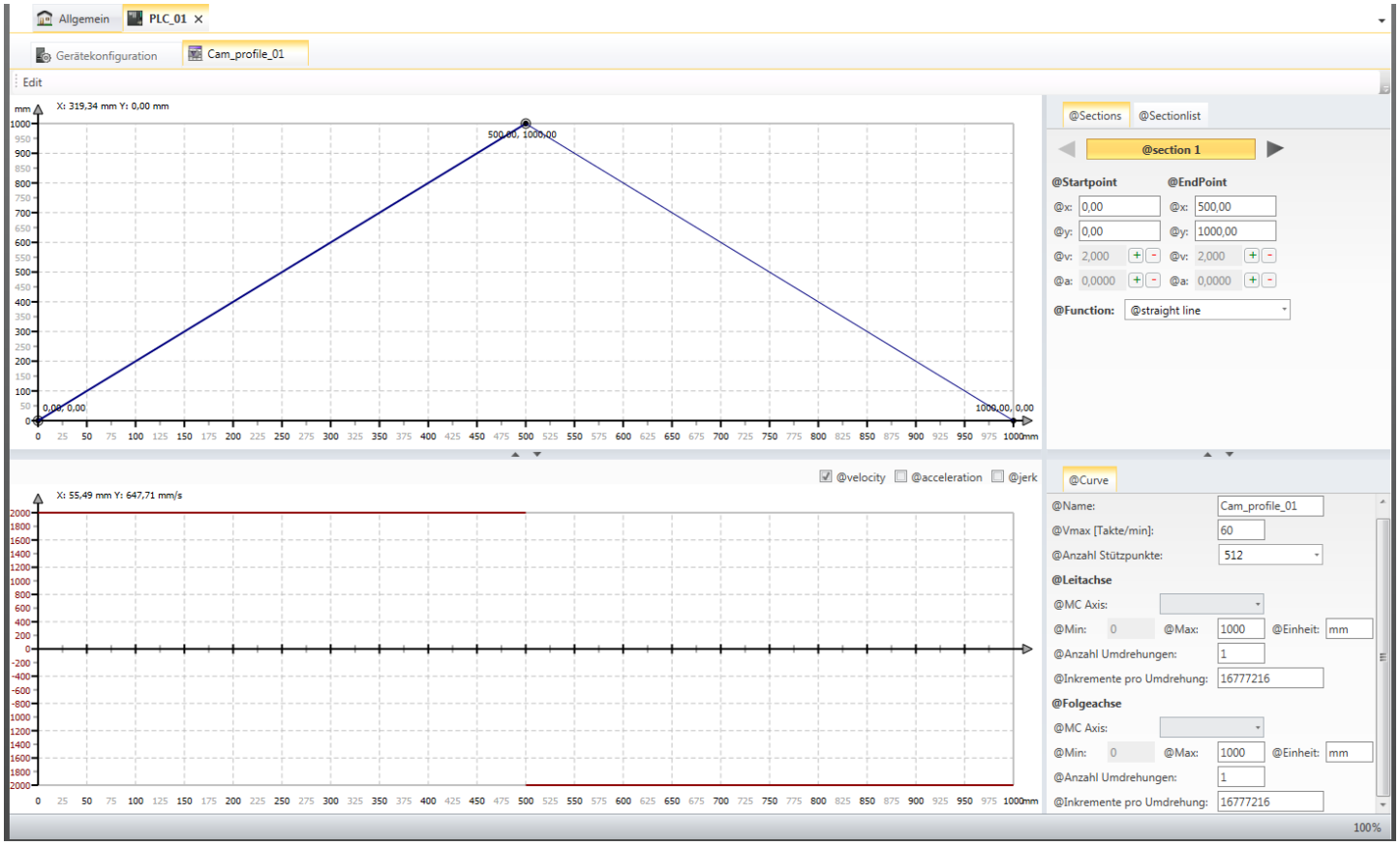
14.4.1.2.1 Beschreibung

Die absoluten Positionen von Master und Slave zum Zeitpunkt des Koppelns entsprechen dem Anfangswert des Kurvenscheibenprofils.

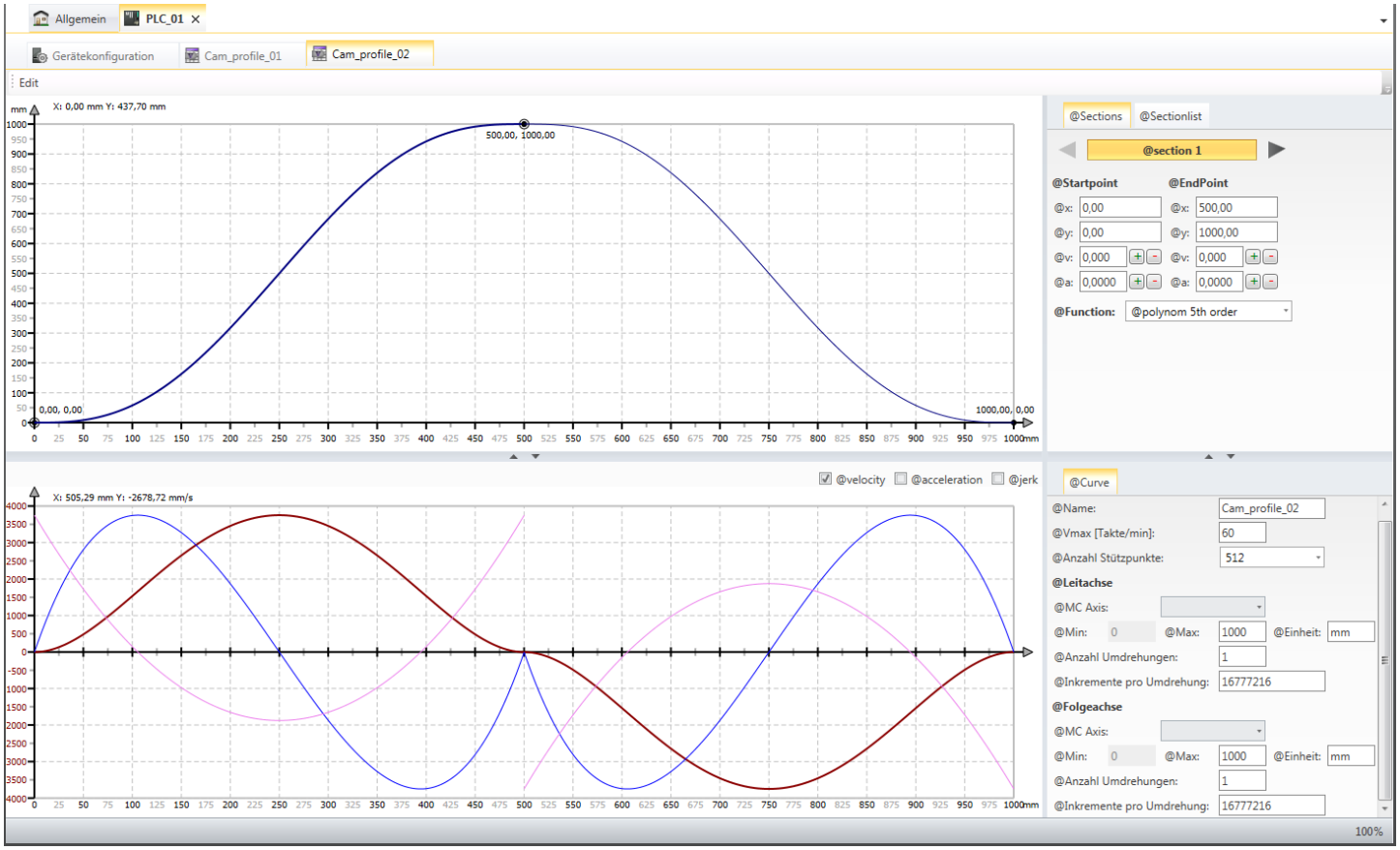
- *Beim Koppeln oder Umschalten mit StartMode = "Relative" kann es zu keinem Sprung der Slave-Achse kommen.*
- *Beim Umschalten in positiver Master-Richtung entsprechen die absoluten Positionen von Master und Slave dem Anfangswert (erster Wert) des Kurvenscheibenprofils. Beim Umschalten in negativer Master-Richtung entsprechen die absoluten Positionen von Master und Slave dem Endwert (letzter Wert) des Kurvenscheibenprofils.*

14.4.1.2.2 Verwendete Kurvenscheibenprofile

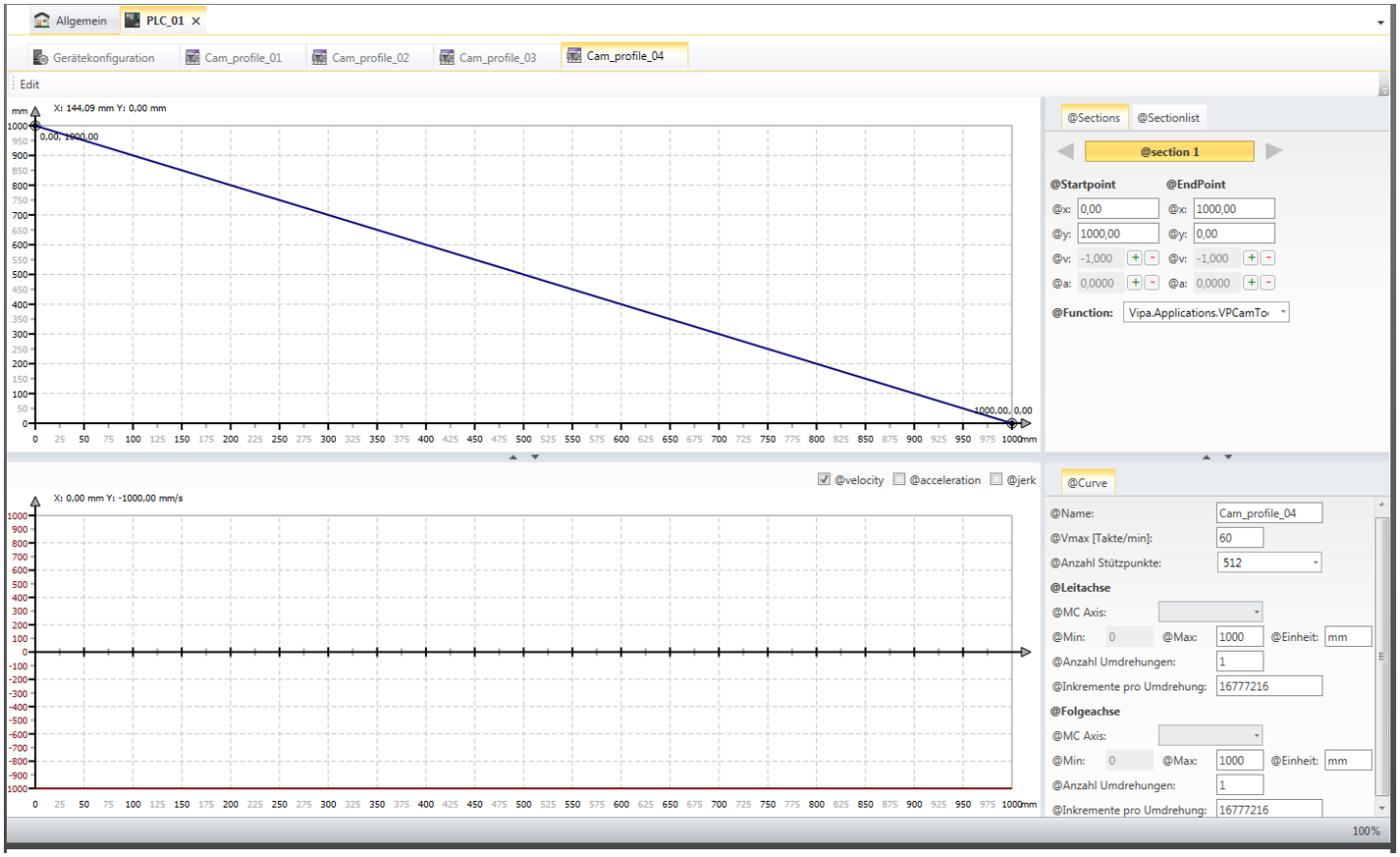
Cam_profile_01



Cam_profile_02

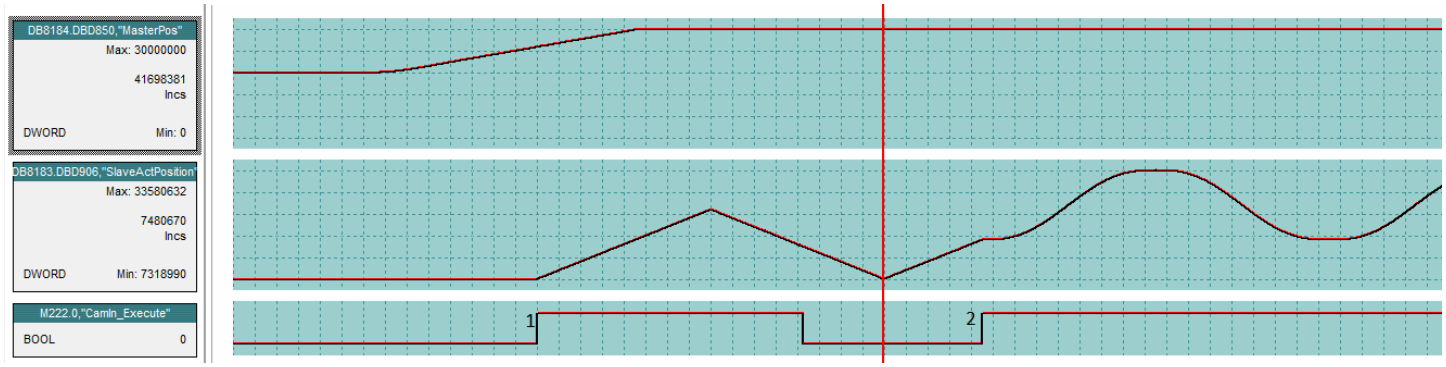


Cam_profile_04



14.4.1.2.3 Beispiel für Koppeln sofort und Umschalten sofort (Kurvenscheibenprofil ohne Hub)

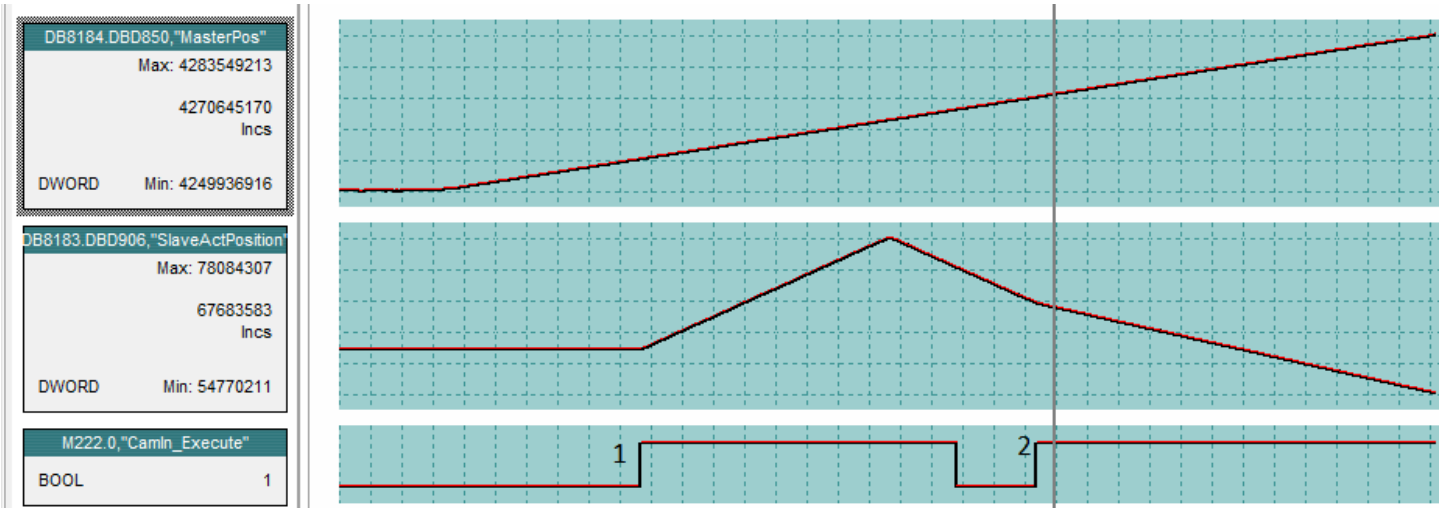
Position



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird die Slave-Achse über das Kurvenscheibenprofil "Cam_Profile_01" an die Master-Achse gekoppelt. Die absoluten Positionen von Master und Slave beim Zeitpunkt des Koppelns entsprechen dem Anfangswert des Kurvenscheibenprofils. Die Slave-Achse folgt entsprechend dem Kurvenscheibenprofil "Cam_Profile_01" der Master-Achse.

(2) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (2) wird sofort auf das Kurvenscheibenprofil "Cam_Profile_02" umgeschaltet. Die absoluten Positionen von Master und Slave zum Zeitpunkt des Umschaltens entsprechen dem Anfangswert des Kurvenscheibenprofils. Die Slave-Achse folgt entsprechend dem Kurvenscheibenprofil "Cam_Profile_02" der Master-Achse.

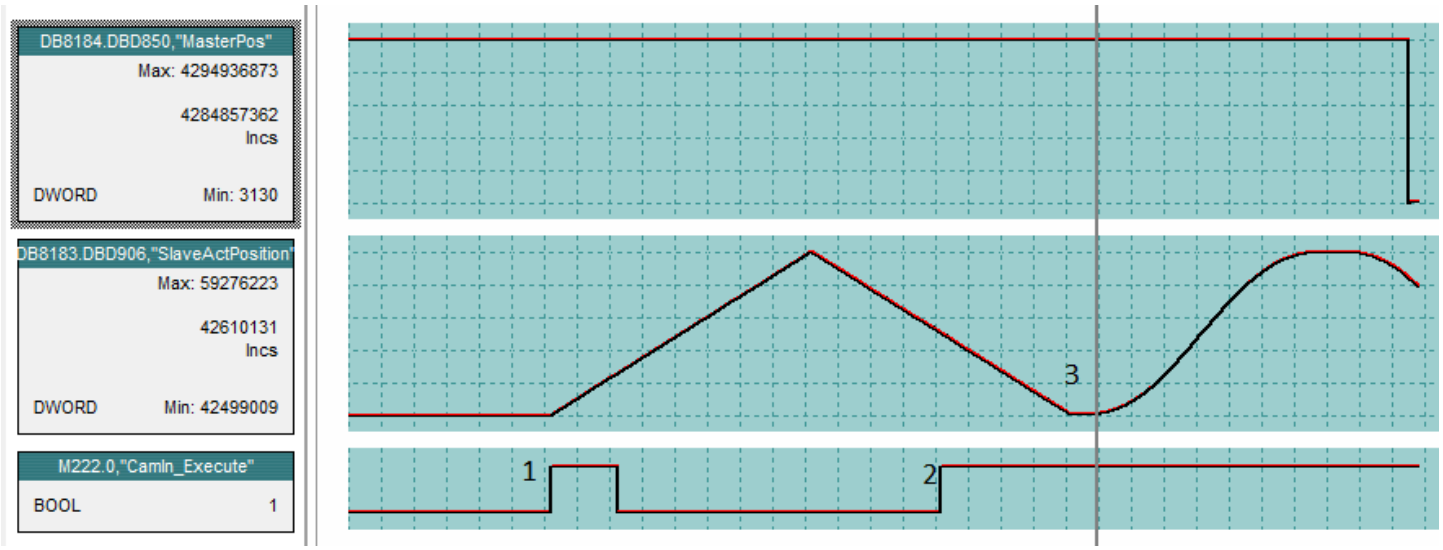
14.4.1.2.4 Beispiel für Koppeln sofort und Umschalten sofort (Kurvenscheibenprofil mit Hub)



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird die Slave-Achse über das Kurvenscheibenprofil "Cam_Profile_01" an die Master-Achse gekoppelt. Die absoluten Positionen von Master und Slave zum Zeitpunkt des Koppelns entsprechen dem Anfangswert vom Kurvenscheibenprofil. Die Slave-Achse folgt entsprechend des Kurvenscheibenprofils "Cam_Profile_01" der Master-Achse.

(2) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (2) wird sofort auf das Kurvenscheibenprofil "Cam_Profile_04" umgeschaltet. Die absoluten Positionen von Master und Slave zum Zeitpunkt des Umschaltens entsprechen dem Anfangswert des Kurvenscheibenprofils. Die Slave-Achse folgt entsprechend dem Kurvenscheibenprofil "Cam_Profile_04" der Master-Achse.

14.4.1.2.5 Beispiel für Koppeln sofort und Umschalten nächster Zyklus (Kurvenscheibenprofil ohne Hub)



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird die Slave-Achse über das Kurvenscheibenprofil "Cam_Profile_01" an die Master-Achse gekoppelt. Die absoluten Positionen von Master und Slave zum Zeitpunkt des Koppelns entsprechen dem Anfangswert des Kurvenscheibenprofils. Die Slave-Achse folgt entsprechend dem Kurvenscheibenprofil "Cam_Profile_01" der Master-Achse.

(2) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (2) wird zu Beginn des nächsten Kurvenscheibenzyklus zum Zeitpunkt (3) auf das Kurvenscheibenprofil "Cam_Profile_02" umgeschaltet. Die absoluten Positionen von Master und Slave zum Zeitpunkt des Umschaltens entsprechen dem Anfangswert des Kurvenscheibenprofils. Die Slave-Achse folgt entsprechend dem Kurvenscheibenprofil "Cam_Profile_02" der Master-Achse.

14.4.1.3 Beispiele mit *StartMode = "Absolute"*

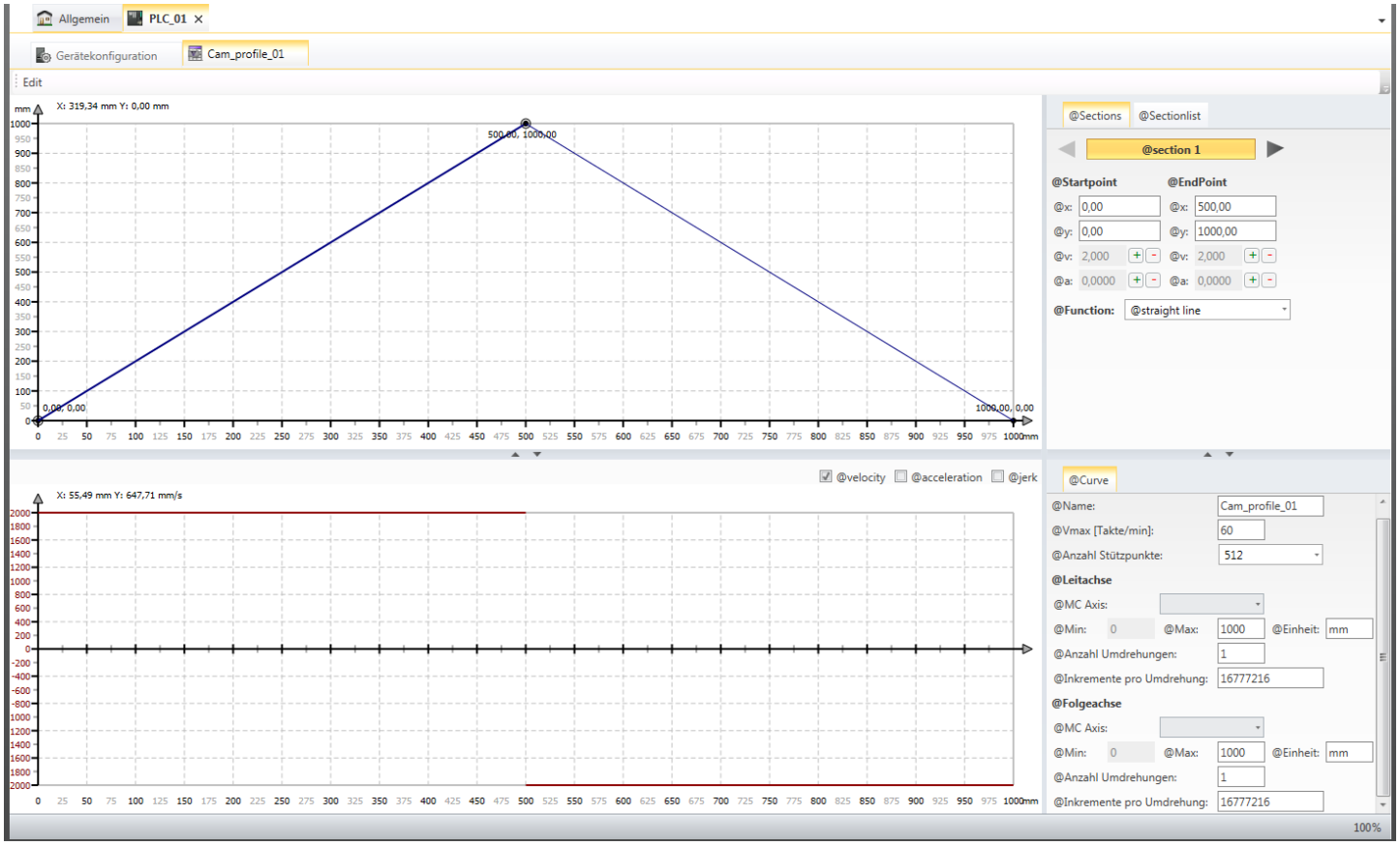
14.4.1.3.1 Beschreibung

Das Koordinatensystem von Master und Slave ist deckungsgleich mit dem Koordinatensystem vom Kurvenscheibenprofil. Hier wird dem aktuellen Masterwert der entsprechende Absolutwert der Kurvenscheibe zugrunde gelegt.

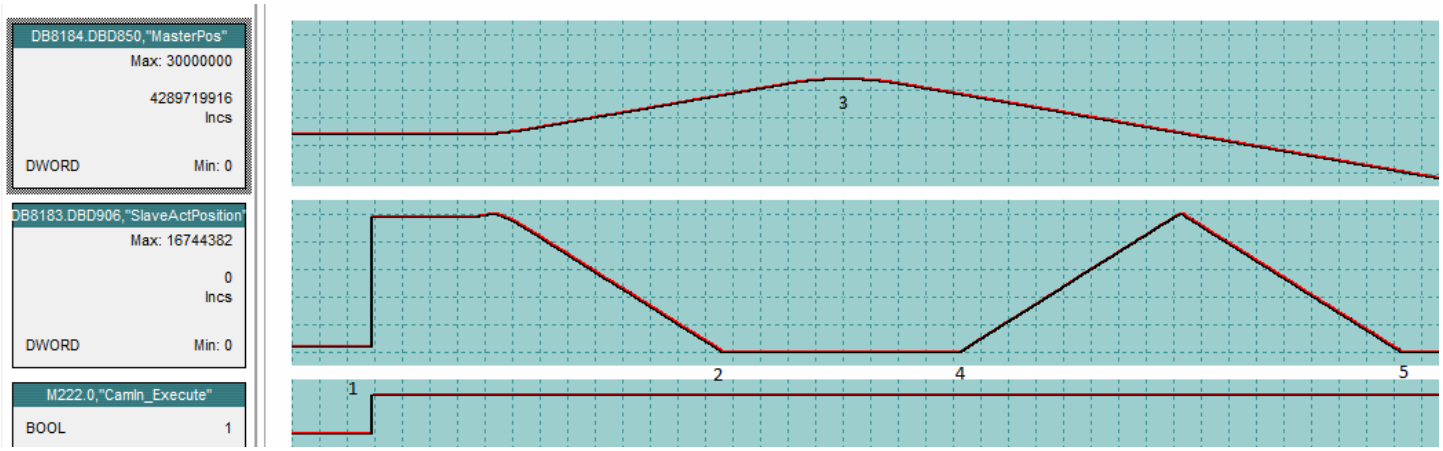
Beim Koppeln oder Umschalten mit StartMode = "Absolute" kann es zu einem Sprung der Slave-Achse kommen.

14.4.1.3.2 Verwendete Kurvenscheiben-Profile

Cam_profile_01



14.4.1.3.3 Beispiel für Koppeln sofort innerhalb der Kurvenscheibe



(1) Zum Zeitpunkt (1) befindet sich die Master-Achse innerhalb des über das Kurvenscheibenprofil festgelegten Bereichs. Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird die Slave-Achse über das Kurvenscheibenprofil "Cam_Profile_01" an die Master-Achse gekoppelt. Das Koordinatensystem von Master und Slave ist deckungsgleich mit dem Koordinatensystem des Kurvenscheibenprofils. Aus diesem Grund "springt" die Slave-Achse an die über das Kurvenscheibenprofil festgelegte Position.

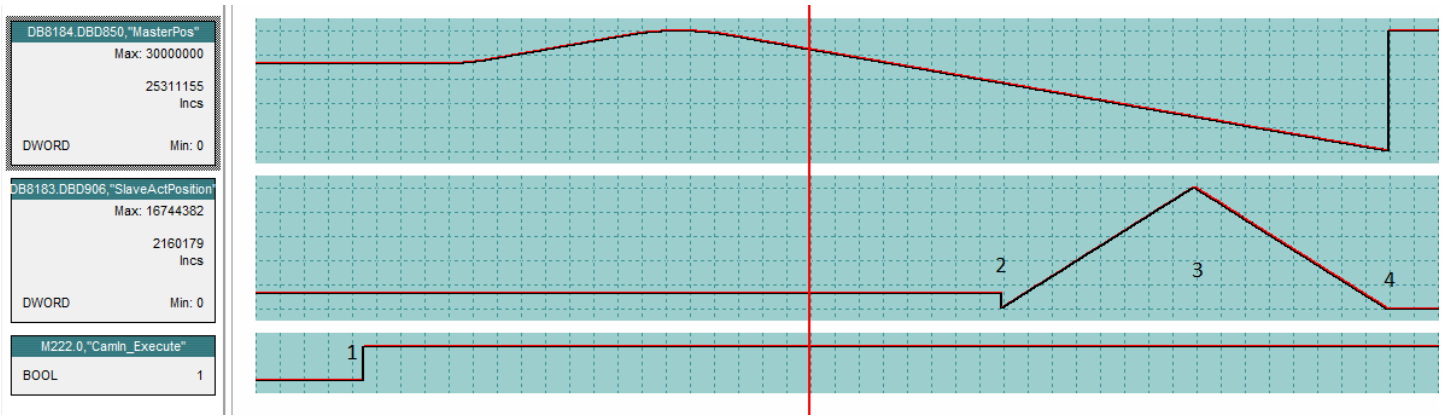
(2) Zum Zeitpunkt (2) verlässt die Master-Achse den über das Kurvenscheibenprofil festgelegten Bereich. Aus diesem Grund bleibt die Slave-Achse stehen.

(3) Zum Zeitpunkt (3) dreht sich die Richtung der Master-Achse um.

(4) Zum Zeitpunkt (4) kommt die Master-Achse in den über das Kurvenscheibenprofil festgelegten Bereich zurück. Die Slave-Achse wird entsprechend des Kurvenscheibenprofils verfahren.

(5) Zum Zeitpunkt (5) verlässt die Master-Achse wiederum den über das Kurvenscheibenprofil festgelegten Bereich. Aus diesem Grund bleibt die Slave-Achse stehen.

14.4.1.3.4 Beispiel für Koppeln sofort außerhalb der Kurvenscheibe



(1) Zum Zeitpunkt (1) befindet sich die Master-Achse außerhalb des über das Kurvenscheibenprofil festgelegten Bereichs. Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird die Slave-Achse über das Kurvenscheibenprofil "Cam_Profile_01" an die Master-Achse gekoppelt. Das Koordinatensystem von Master und Slave ist deckungsgleich mit dem Koordinatensystem vom Kurvenscheibenprofil. Da die Master-Achse außerhalb des über das Kurvenscheibenprofil festgelegten Bereichs ist, wird die Slave-Achse nicht verfahren.

(2) Zum Zeitpunkt (2) tritt die Master-Achse in den über das Kurvenscheibenprofil festgelegten Bereich ein. Aus diesem Grund "springt" die Slave-Achse an die über das Kurvenscheibenprofil festgelegte Position.

(3) Während der Zeitphase (3) wird die Slave-Achse entsprechend des Kurvenscheibenprofils verfahren.

(4) Zum Zeitpunkt (4) verlässt die Master-Achse den über das Kurvenscheibenprofil festgelegten Bereich. Aus diesem Grund bleibt die Slave-Achse stehen.

14.4.2 FB 707 - MC_GearIn - Kopplung Master-Slave-Achse über Geschwindigkeit

Beschreibung

Mit MC_GearIn wird eine Kopplung zwischen Slave-Achse und Master-Achse über die Geschwindigkeit eingerichtet. Mit den Parametern *Acceleration*, *Deceleration* und *Jerk* wird das dynamische Verhalten bei der Synchronisation der Slave-Achse bestimmt. Nach der Synchronisation ergibt sich das dynamische Verhalten der Slave-Achse aus der Bewegung der Master-Achse und der gewählten Geschwindigkeitsübersetzung. Mit den Parametern *RatioNumerator* und *RatioDenominator* kann die Geschwindigkeitsübersetzung zwischen Master-Achse und Slave-Achse eingestellt werden.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Master	IN_OUT	MC_AXIS_REF	Referenz zur Master-Achse
Slave	IN_OUT	MC_AXIS_REF	Referenz zur Slave-Achse
Execute	INPUT	BOOL	Mit einer Flanke 0-1 an *Execute* wird das Koppeln der Slave-Achse an die Master-Achse gestartet
ContinuousUpdate	INPUT	BOOL	Parameter aktuell nicht unterstützt; Aufruf mit FALSE
RatioNumerator	INPUT	INT	Zähler der Geschwindigkeitsübersetzung -2048 ... 2047
RatioDenominator	INPUT	INT	Nenner der Geschwindigkeitsübersetzung

Parameter	Deklaration	Datentyp	Beschreibung
Acceleration	INPUT	REAL	Beschleunigung während der Synchronisation in [Anwendereinheiten/s^2]
Deceleration	INPUT	REAL	Verzögerung während der Synchronisation in [Anwendereinheiten/s^2]
Jerk	INPUT	REAL	Ruck während der Synchronisation in [Anwendereinheiten/s^3]
BufferMode	INPUT	BYTE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0
InGear	OUTPUT	BOOL	■ Synchronisation – TRUE: Synchronisation erreicht
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung: Synchronisation ist noch nicht erreicht
Active	OUTPUT	BOOL	■ Status – TRUE: Baustein hat die Kontrolle über die Achse
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Einsatz mit mechanischem Getriebe

Bitte beachten Sie, dass der Baustein aktuell kein mechanisches Getriebe unterstützt. Möchten Sie ein mechanisches Getriebe verwenden, so müssen Sie das Getriebeverhältnis in die Parameter RatioNumerator und RatioDenominator mit einrechnen.

Beispiel:

- *Vorgabe:*
  - *Achse 1 folgt Achse 2 mit Verhältnis 1:1 (x:y)*
  - *An Achse 1 ist ein mechanisches Getriebe von 1:3 (a:b) verbaut*
  - *An Achse 2 ist ein mechanisches Getriebe von 2:1 (c:d) verbaut*
- *Berechnung:*
  - *RatioNumerator = x · b · c = 1 · 3 · 2 = 6*
  - *RatioDenominator = y · a · d = 1 · 1 · 1 = 1*

Verwendung

- Bausteinaufruf im:
  - OB 61
- Anwendbar auf:
  - Positionierungsachse
  - Virtuelle Achse

PLCopen-State

- Master-Achse
  - Start des Auftrags in den PLCopen-States *Standstill*, *Discrete Motion*, *Synchronized Motion*, *Continuous Motion* und *Stopping* möglich.
- Slave-Achse
  - Start des Auftrags in den PLCopen-States *Standstill*, *Discrete Motion*, *Synchronized Motion* und *Continuous Motion* möglich.
- MC_GearIn führt die Achse in den PLCopen-State *Synchronized Motion* über.

Slave-Achse an Master-Achse koppeln

Mit einer Flanke 0-1 an *Execute* wird das Ankoppeln der Slave-Achse an die Master-Achse gestartet. Solange die Slave-Achse, die über die Geschwindigkeitsübersetzung festgelegte Geschwindigkeit nicht erreicht hat, wird *Busy* = TRUE und *InGear* = FALSE geliefert. Wird die Geschwindigkeit erreicht, wird *Busy* = FALSE und *InGear* = TRUE geliefert. Die Wegstrecke, die bei der Synchronisation verloren geht, wird nicht kompensiert.

VORSICHT!
Eine gegenseitige Verriegelung zweier oder mehrerer Achsen kann zu einem Aufschwingen des Systems führen!

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt, auch wenn die durch die Geschwindigkeitsübersetzung definierte Geschwindigkeit erreicht wurde.*
- *Ein laufender Auftrag kann durch einen anderen Bewegungsauftrag (z.B. MC_MoveAbsolute) abgebrochen werden. Anschließend sind Master- und Slave-Achse entkoppelt.*

Geschwindigkeit der Slave-Achse

$V_{Slave} = V_{Master} \times RD / RN$

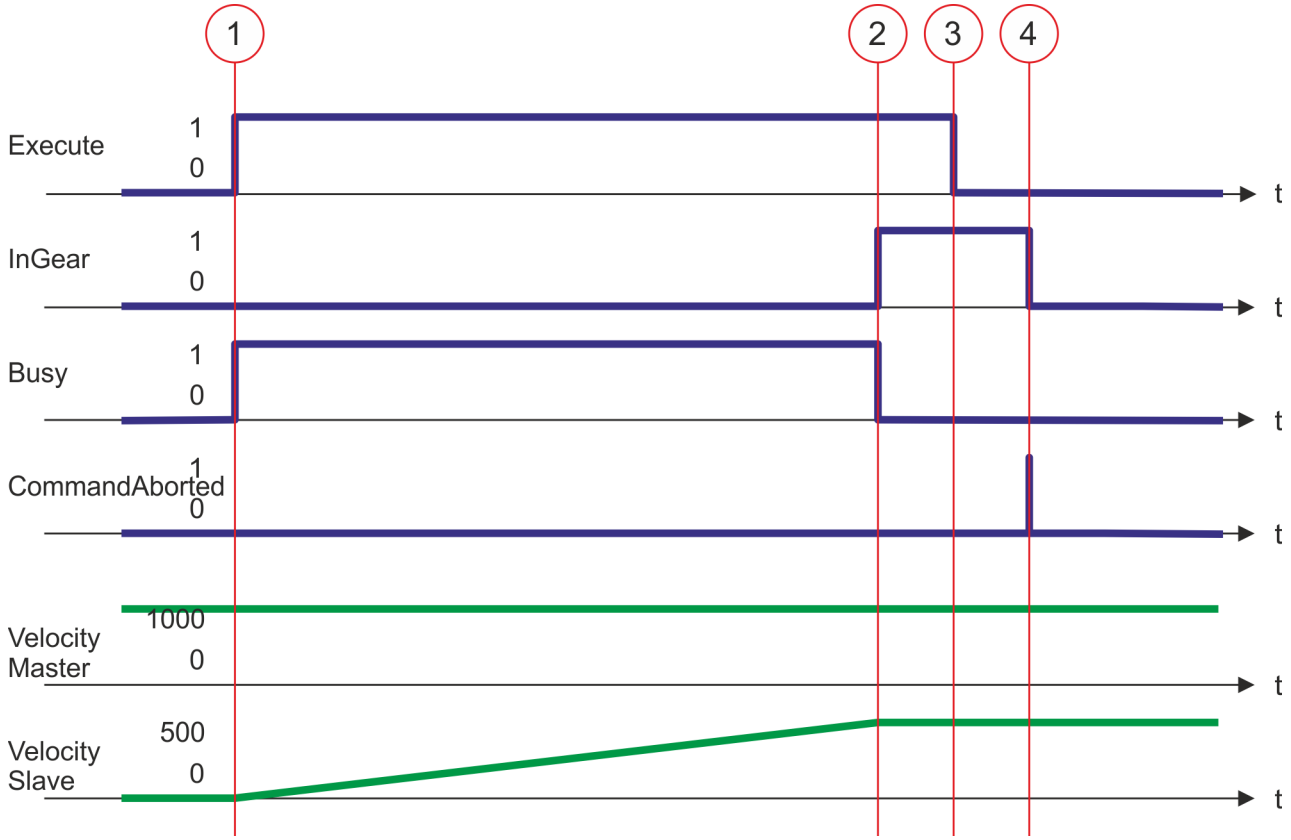
V_{Slave} - Geschwindigkeit Slave-Achse

V_{Master} - Geschwindigkeit Master-Achse

RD - *RatioDenominator*

RN - *RatioNumerator*

Zustandsdiagramm der Bausteinparameter



(1) Die Master-Achse wird mit MC_MoveVelocity mit einer Geschwindigkeit von 1000.0 verfahren. Zum Zeitpunkt (1) steht die Slave-Achse. Die eingestellte Geschwindigkeitsübersetzung ist 1:2. Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Koppeln der Slave-Achse an die Master-Achse gestartet, *Busy* liefert den Wert TRUE. Die Slave-Achse wird beschleunigt.
(2) Zum Zeitpunkt (2) erreicht die Slave-Achse, die über die Geschwindigkeitsübersetzung festgelegte Geschwindigkeit von 500.0 und *Busy* = FALSE und *InGear* = TRUE.
(3) Das Rücksetzen von *Execute* auf FALSE zum Zeitpunkt (3) hat keine Auswirkung auf die Slave-Achse. Die Slave-Achse ist weiterhin an die Master-Achse über die Geschwindigkeitsübersetzung gekoppelt.
(4) Zum Zeitpunkt (4) wird die Slave-Achse durch einen MC_GearOut-Auftrag von der Master-Achse abgekoppelt. Die Slave-Achse wird weiterhin mit konstanter Geschwindigkeit verfahren, bis ein weiterer Bewegungsauftrag gestartet wird.

14.4.3 FB 709 - MC_CamOut - Deaktivierung Master-Slave-Kopplung über Kurvenscheibe

Beschreibung Mit MC_CamOut wird eine Master-Slave-Kopplung über eine Kurvenscheibe deaktiviert.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Slave	IN_OUT	MC_AXIS_REF	Referenz zur Slave-Achse
Execute	INPUT	BOOL	■ Slave-Achse abkoppeln – Flanke 0-1: Das Abkoppeln der Slave-Achse von der Master-Achse wird gestartet.
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Slave-Achse wurde von Master-Achse abgekoppelt.
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  - OB 1
  - OB 61
- Anwendbar auf:
  - Positionierungsachse
  - Externer Encoder
  - Virtuelle Achse

PLCopen-State

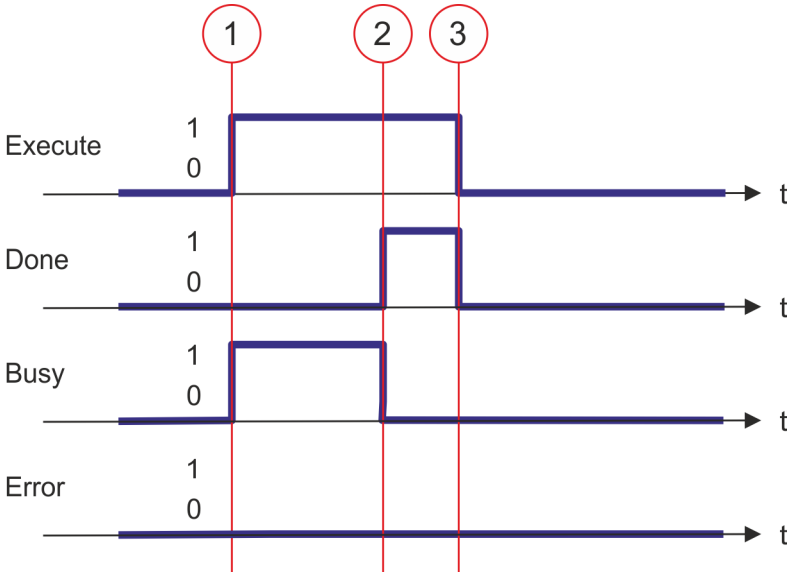
- Start des Auftrags ist im PLCopen-State *Synchronized Motion* nur dann möglich, wenn die Slave-Achse über eine Kurvenscheibe mit der Master-Achse gekoppelt ist.
- MC_CamOut führt die Achse in den PLCopen-State *Continuous Motion* über.

Slave-Achse abkoppeln

Mit einer Flanke 0-1 an *Execute* wird das Abkoppeln der Slave-Achse von der Master-Achse gestartet. Solange das Abkoppeln der Slave-Achse läuft, zeigt *Busy* den Wert TRUE. Nachdem das Abkoppeln erfolgreich durchgeführt wurde, wird *Busy* = FALSE und *Done* = TRUE geliefert.

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE ausgeführt, bis die Slave-Achse abgekoppelt ist.*
- *Ein laufender Auftrag kann durch einen anderen Bewegungsauftrag (z.B. MC_MoveVelocity) abgebrochen werden. Anschließend sind Master- und Slave-Achse entkoppelt.*
- *Wenn die Slave-Achse während einer Bewegungsfahrt abgekoppelt wird, fährt die Slave-Achse mit konstanter Geschwindigkeit endlos weiter, bis ein anderer Bewegungsauftrag (z.B. MC_Halt) ausgeführt wird.*

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Abkoppeln der Slave-Achse von der Master-Achse gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Abkoppeln erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.4.4 FB 710 - MC_GearOut - Deaktivierung Master-Slave-Kopplung über Geschwindigkeit

Beschreibung Mit MC_GearOut wird eine Master-Slave-Kopplung über die Geschwindigkeit deaktiviert.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Slave	IN_OUT	MC_AXIS_REF	Referenz zur Slave-Achse
Execute	INPUT	BOOL	■ Slave-Achse abkoppeln – Flanke 0-1: Das Abkoppeln der Slave-Achse von der Master-Achse wird gestartet
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Slave-Achse wurde von Master-Achse abgekoppelt.
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung

Parameter	Deklaration	Datentyp	Beschreibung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*

Verwendung

- Bausteinaufruf im:
  - OB 1
  - OB 61
- Anwendbar auf:
  - Positionierungsachse
  - Externer Encoder
  - Virtuelle Achse

PLCopen-State

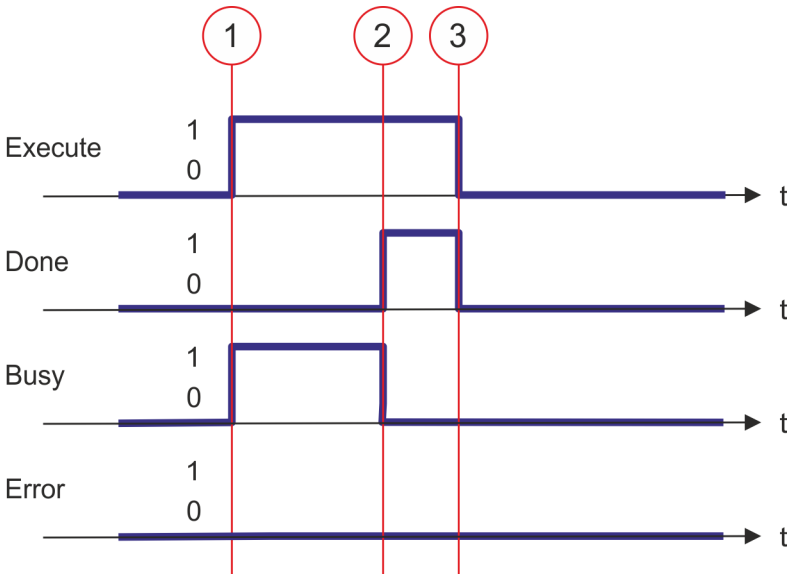
- Start des Auftrags ist im PLCopen-State *Synchronized Motion* nur dann möglich, wenn die Slave-Achse über eine Kurvenscheibe mit der Master-Achse gekoppelt ist.
- MC_CamOut führt die Achse in den PLCopen-State *Continuous Motion* über.

Slave-Achse abkoppeln

Mit einer Flanke 0-1 an *Execute* wird das Abkoppeln der Slave-Achse von der Master-Achse gestartet. Solange das Abkoppeln der Slave-Achse läuft, zeigt *Busy* den Wert TRUE. Nachdem das Abkoppeln erfolgreich durchgeführt wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE, bis die Slave-Achse abgekoppelt ist, ausgeführt.*
- *Ein laufender Auftrag kann durch einen anderen Bewegungsauftrag (z.B. MC_MoveVelocity) abgebrochen werden. Anschließend sind Master- und Slave-Achse entkoppelt.*
- *Wenn die Slave-Achse während einer Bewegungsfahrt abgekoppelt wird, fährt die Slave-Achse mit konstanter Geschwindigkeit endlos weiter, bis ein anderer Bewegungsauftrag (z.B. MC_Halt) ausgeführt wird.*

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Abkoppeln der Slave-Achse von der Master-Achse gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Abkoppeln erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

14.4.5 FB 720 - MC_PhasingAbsolute - Phasenverschiebung absolut

Beschreibung

Beim MC_PhasingAbsolute wird aus Sicht der Slave-Achse die Master-Achsen-Position um einen absoluten Positions-Wert verschoben. Mit MC_GearIn oder MC_CamIn wird aus der Summe der Master-Achsen-Position und der Phasenverschiebung eine neue Position für die Slave-Achse errechnet. Die Phasenverschiebung erfolgt immer aus Sicht der Slave-Achse. Die Master-Achse bleibt hiervon unberührt. Die Phasenverschiebung bleibt solange bestehen, bis diese durch ein weiteres "Phasen"-Kommando überschrieben wird.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Master	IN_OUT	MC_AXIS_REF	Referenz zur Master-Achse.
Slave	IN_OUT	MC_AXIS_REF	Referenz zur Slave-Achse.
Execute	IN	BOOL	Mit einer Flanke 0-1 an *Execute* wird die Phasenverschiebung gestartet.
PhaseShift	IN	REAL	Absolute Phasenverschiebung der Masterposition gegenüber der Slave-Achse [Anwendereinheiten].
Velocity	IN	REAL	Maximale Geschwindigkeit (muss nicht zwingend erreicht werden) vorzeichenbehafteter Wert in [Anwendereinheiten/s].
Acceleration	IN	REAL	Beschleunigung in [Anwendereinheiten/s^2].

Parameter	Deklaration	Datentyp	Beschreibung
Deceleration	IN	REAL	Verzögerung in [Anwendereinheiten/s^2].
Jerk	IN	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
BufferMode	IN	MC_BUFFER_MODE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0.
Done	OUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Die Phasenverschiebung ist aktiv.
Busy	OUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung.
Active	OUT	BOOL	■ Status – TRUE: Baustein hat die Kontrolle über die Achse.
CommandAborted	OUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen.
Error	OUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUT	WORD	Zusätzliche Fehlerinformationen *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
AbsolutePhaseShift	OUT	REAL	Zeigt die aktuelle absolute Phasenverschiebung [Anwendereinheiten], solange *Busy* oder *Done* gesetzt ist.

PhaseShift, Velocity, Acceleration und Deceleration einer Phasenverschiebung werden über den FB gesteuert.

Verwendung

- Bausteinaufruf im:
  – OB 61
- Anwendbar auf:
  – Positionierungsachse
  – Virtuelle Achse

PLCopen-State

- Start des Auftrags ist im PLCopen-State *Synchronized Motion* nur dann möglich, wenn die Slave-Achse über eine Kurvenscheibe oder ein Getriebe mit der Master-Achse gekoppelt ist.

Slave-Achse abkoppeln

Mit einer Flanke 0-1 an *Execute* wird die absolute Phasenverschiebung der Master-Achse gegenüber der Slave-Achse gestartet. Solange der Befehl abgearbeitet wird, zeigt *Busy* den Wert TRUE. Nach Abarbeitung des Befehls, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE, solange ausgeführt, bis dieser abgearbeitet ist.*
- *Ein laufender Auftrag kann durch einen neuen Auftrag zur Phasenverschiebung für die selbe Achse abgebrochen werden.*
- *Ein laufender Auftrag kann durch Zustandswechsel aus dem PLCopen-State Synchronized Motion abgebrochen werden.*

14.4.6 FB 721 - MC_PhasingRelative - Phasenverschiebung relativ

Beschreibung

Beim MC_PhasingRelative wird aus Sicht der Slave-Achse die Master-Achsen-Position um einen relativen Positions-Wert verschoben. Mit MC_GearIn oder MC_CamIn wird aus der Summe der Master-Achsen-Position und der Phasenverschiebung eine neue Position für die Slave-Achse errechnet. Die Phasenverschiebung erfolgt immer aus Sicht der Slave-Achse. Die Master-Achse bleibt hiervon unberührt. Die Phasenverschiebung bleibt solange bestehen, bis diese durch ein weiteres "Phasen"-Kommando überschrieben wird.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Master	IN_OUT	MC_AXIS_REF	Referenz zur Master-Achse.
Slave	IN_OUT	MC_AXIS_REF	Referenz zur Slave-Achse.
Execute	IN	BOOL	Mit einer Flanke 0-1 an *Execute* wird die Phasenverschiebung gestartet.
PhaseShift	IN	REAL	Relative Phasenverschiebung der Masterposition gegenüber der Slave-Achse [Anwendereinheiten].
Velocity	IN	REAL	Maximale Geschwindigkeit (muss nicht zwingend erreicht werden) vorzeichenbehafteter Wert in [Anwendereinheiten/s].
Acceleration	IN	REAL	Beschleunigung in [Anwendereinheiten/s^2].
Deceleration	IN	REAL	Verzögerung in [Anwendereinheiten/s^2].
Jerk	IN	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
BufferMode	IN	MC_BUFFER_MODE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0.
Done	OUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Die Phasenverschiebung ist aktiv.
Busy	OUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung.
Active	OUT	BOOL	■ Status – TRUE: Baustein hat die Kontrolle über die Achse.
CommandAborted	OUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen.

Parameter	Deklaration	Datentyp	Beschreibung
Error	OUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 14.5 "ErrorID - Zusätzliche Fehlerinformationen" Seite 569*
CoveredPhaseShift	OUT	REAL	Zeigt kontinuierlich die relative Phasenverschiebung [Anwendereinheiten], solange *Busy* gesetzt ist.

PhaseShift, Velocity, Acceleration und Deceleration einer Phasenverschiebung werden über den FB gesteuert.

Verwendung

- Bausteinaufruf im:
  – OB 61
- Anwendbar auf:
  – Positionierungsachse
  – Virtuelle Achse

PLCopen-State

- Start des Auftrags ist im PLCopen-State *Synchronized Motion* nur dann möglich, wenn die Slave-Achse über eine Kurvenscheibe oder ein Getriebe mit der Master-Achse gekoppelt ist.

Slave-Achse abkoppeln

Mit einer Flanke 0-1 an *Execute* wird die relative Phasenverschiebung der Master-Achse gegenüber der Slave-Achse gestartet. Solange der Befehl abgearbeitet wird, zeigt *Busy* den Wert TRUE. Nach Abarbeitung des Befehls, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

– *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE, solange ausgeführt, bis dieser abgearbeitet ist.*
– *Ein laufender Auftrag kann durch einen neuen Auftrag zur Phasenverschiebung für die selbe Achse abgebrochen werden.*
– *Ein laufender Auftrag kann durch Zustandswechsel aus dem PLCopen-State Synchronized Motion abgebrochen werden.*

14.5 ErrorID - Zusätzliche Fehlerinformationen

ErrorID	Beschreibung	Bemerkung
0x0000	Kein Fehler	
0x8001	Unzulässiger Wert beim Parameter *Position*.	
0x8002	Unzulässiger Wert beim Parameter *Distance*.	
0x8003	Unzulässiger Wert beim Parameter *Velocity*.	
0x8004	Unzulässiger Wert beim Parameter *Acceleration*.	

ErrorID	Beschreibung	Bemerkung
0x8005	Unzulässiger Wert beim Parameter *Deceleration*.	
0x8006	Unzulässiger Wert beim Parameter *Jerk*.	
0x8007	Unzulässiger Wert beim Parameter *ContinuousUpdate*.	
0x8008	Unzulässiger Wert beim Parameter *BufferMode*.	
0x8009	Unzulässiger Wert beim Parameter *EnablePositive*.	
0x800A	Unzulässiger Wert beim Parameter *EnableNegative*.	
0x800B	Unzulässiger Wert beim Parameter *MasterOffset*.	
0x800C	Unzulässiger Wert beim Parameter *SlaveOffset*.	
0x800D	Unzulässiger Wert beim Parameter *MasterScaling*.	
0x800E	Unzulässiger Wert beim Parameter *SlaveScaling*.	
0x800F	Unzulässiger Wert beim Parameter *StartMode*.	
0x8010	Unzulässiger Wert beim Parameter *ActivationMode*.	
0x8011	Unzulässiger Wert beim Parameter *Source*.	
0x8012	Unzulässiger Wert beim Parameter *Direction*.	
0x8013	Unzulässiger Parameter der virtuellen Achse.	MC_ReadParameter
0x8014	Unzulässiger Parameter der physikalischen Achse.	MC_ReadParameter
0x8015	Unzulässiger Index oder Subindex.	MC_ReadParameter
0x8016	Unzulässige Parameterlänge.	MC_ReadParameter
0x8017	Unzulässige LADDR.	MC_ReadParameter
0x8018	Unzulässiger Wert beim Parameter *RatioDenominator*.	MC_GearIn
0x8019	Unzulässiger Wert beim Parameter *RatioNumerator*.	MC_GearIn
0x801A	Parameternummer nicht bekannt.	MC_ReadParameter, MC_Write-Parameter
0x801B	Parameter kann nicht geschrieben werden, Parameter ist schreib-geschützt.	MC_WriteParameter
0x801C	Parameter Kommunikation mit unbekanntem Mode.	MC_Home, MC_WriteParameter
0x801D	Parameterkommunikation mit allgemeinem Fehler. Die Fehlerur-sache ist nicht näher beschrieben.	
0x801E	SDO-Parameterwert außerhalb des zulässigen Bereichs.	MC_Home, MC_WriteParameter
0x801F	Der Typ in ANY ist nicht BYTE.	Parameter lesen/schreiben
0x8020	Unterschiedliche Konfiguration der Anwendereinheiten in Cam und Master-Achse.	
0x8021	Unterschiedliche Konfiguration der Anwendereinheiten in Cam und Slave-Achse.	
0x8022	Auf der über LADDR angegebenen logischen Adresse gibt es kein PROFIBUS/PROFINET-Device, von dem konsistente Daten gelesen werden können.	Parameter lesen/schreiben
0x8023	Es wurde ein Zugriffsfehler beim Zugriff auf ein I/O-Geräte erkannt.	Parameter lesen/schreiben
0x8024	Slave-Fehler an externem DP-Slave.	Parameter lesen/schreiben

ErrorID	Beschreibung	Bemerkung
0x8025	Systemfehler an externem DP-Slave.	Parameter lesen/schreiben
0x8026	Systemfehler an externem DP-Slave.	Parameter lesen/schreiben
0x8027	Die Daten wurden noch nicht vom Modul gelesen.	Parameter lesen/schreiben
0x8028	Systemfehler an externem DP-Slave.	Parameter lesen/schreiben
0x8029	Schreibversuch auf eine Objekt, welches nur gelesen werden kann.	Parameter lesen/schreiben
0x802A	Leseversuch auf ein Objekt, welches nur geschrieben werden kann.	Parameter lesen/schreiben
0x802B	Nicht unterstützte Zugriff auf ein Objekt.	Parameter lesen/schreiben
0x802C	Falscher Datentyp.	Parameter lesen/schreiben
0x802D	Fehler im Geräteprofil.	Parameter lesen/schreiben
0x802E	Fehler Kommando-Typ.	Parameter lesen/schreiben
0x802F	Keine Systemressourcen verfügbar.	Parameter lesen/schreiben
0x8101	Keine zyklische Kommunikation mit der Achse möglich.	
0x8102	Befehl ist im aktuellen PLCopen-State nicht zulässig.	
0x8103	Befehl wird von der Achse nicht unterstützt.	
0x8104	Achse ist nicht einschaltbereit, mögliche Gründe: ■ Kommunikation zur Achse nicht bereit. ■ Antrieb ist nicht im Zustand *"eingeschaltet"* → Antriebsfehler evtl. mit MC_Reset zurücksetzen. ■ Kommunikation wurde unterbrochen, z.B. durch Aus- Einschalten der CPU. Fehler mit MC_Reset zurücksetzen.	*PreOperational* muss auch im Status *Operational* gesetzt werden
0x8105	Befehl wird von virtueller Achse nicht unterstützt.	
0x8106	PLCopen-State ist nicht definiert.	
0x8201	Wegen Mangels an internen Ressourcen kann der Befehl aktuell nicht ausgeführt werden (kein freier Slot im CommandBuffer).	
0x8202	Fehler beim Schreiben des Homing Offsets (kein freier Slot im CommandBuffer).	DriveManager → Homing (aktives Kommando)
0x8301	Keine zyklische Kommunikation mit der Master-Achse möglich.	
0x8302	Befehl ist im aktuellen PLCopen-State der Master-Achse nicht zulässig.	
0x8303	Befehl wird von der Master-Achse nicht unterstützt.	
0x8304	Master-Achse befindet sich nicht im Status *Pre-Operational*.	
0x8305	Die Nummer des Datenbausteins der Master-Achse hat sich geändert.	
0x8306	Kommunikationsfehler an der Master Achse. Slave Achse wird mit Schnellhalt gestoppt.	
0x8311	Keine zyklische Kommunikation mit der Slave-Achse möglich.	
0x8312	Befehl ist im aktuellen PLCopen-State der Slave-Achse nicht zulässig.	
0x8313	Befehl wird von der Slave-Achse nicht unterstützt.	

ErrorID	Beschreibung	Bemerkung
0x8314	Slave-Achse befindet sich nicht im Status *Pre-Operational*.	
0x8315	Die Nummer des Datenbausteins der Slave-Achse hat sich geändert.	
0x8321	Koppeln mit *StartMode* = relative und *ActivationMode* = nextcycle ist nicht zulässig	
0x8322	Koppeln oder schalten mit *StartMode* = absolute und *ActivationMode* = nextcycle ist nicht zulässig	
0x8323	Schalten mit einem unterschiedlichen *StartMode* (*StartMode* der Kopplung ist zu verwenden).	
0x8331	MC_CamIn ist nicht aktiv.	
0x8332	MC_GearIn ist nicht aktiv.	
0x8340	Ungültiger Wert in TriggerInput.Probe.	
0x8341	Ungültiger Wert in TriggerInput.Source.	
0x8342	Ungültiger Wert in TriggerInput.TriggerMode.	
0x8350	Ungültiger Wert in VelocitySearchSwitch.	Homing Initialisierung
0x8351	Ungültiger Wert in VelocitySearchZero.	Homing Initialisierung
0x8352	Ungültige Kombination von Eingängen.	Homing Initialisierung
0x8400	MC_Power: Unerwarteter Drive-State Drive-State <> Operation enabled	
0x8401	MC_Power: Unerwarteter Drive-State Drive-State = Quick stop active	
0x8402	MC_Power: Unerwarteter Drive-State Drive-State = Fault reaction active	
0x8403	MC_Power: Unerwarteter Drive-State Drive-State = Fault	
0x8410	Zeitüberschreitung	MC_Reset
0x8603	Homing-Fehler am Antrieb, Geschwindigkeit <> 0.	MC_Home
0x8604	Homing-Fehler am Antrieb, Geschwindigkeit = 0.	MC_Home
0x8700	FEHLER: Ungültige Größe	
0x8710	FEHLER SDO: Wechsel-Bit nicht wechselnd	
0x8711	FEHLER SDO: SDO-Protokoll Zeitüberschreitung	
0x8712	FEHLER SDO: Client/Server Befehl nicht gültig oder unbekannt	
0x8713	FEHLER SDO: Ungültige Blockgröße (nur im Blockmodus)	
0x8714	FEHLER SDO: Ungültige Sequenznummer (nur im Blockmodus)	
0x8715	FEHLER SDO: CRC-Fehler (nur im Blockmodus)	
0x8716	FEHLER SDO: Speicher voll	
0x8717	FEHLER SDO: Nicht unterstützter Zugriff auf ein Objekt	

ErrorID	Beschreibung	Bemerkung
0x8718	FEHLER SDO: Versuch, ein Nur-Schreiben-Objekt zu lesen	
0x8719	FEHLER SDO: Versuche, ein Nur-Lesen-Objekt zu schreiben	
0x871A	FEHLER SDO: Objekt existiert im Objektverzeichnis nicht	
0x871B	FEHLER SDO: Objekt kann nicht auf ein PDO gemappt werden	
0x871C	FEHLER SDO: Anzahl und Länge der mappenden Objekte überschreiten die PDO-Länge	
0x871D	FEHLER SDO: Allgemeine Parameter-Inkompatibilität	
0x871E	FEHLER SDO: Allgemeine interne Inkompatibilität im Gerät	
0x871F	FEHLER SDO: Zugriff fehlgeschlagen aufgrund eines Hardwarefehlers	
0x8720	FEHLER SDO: Datentyp passt nicht, die Länge des Auftragsparameters passt nicht	
0x8721	FEHLER SDO: Datentyp passt nicht, Serviceparameter zu lang	
0x8722	FEHLER SDO: Datentyp passt nicht, Serviceparameter zu kurz	
0x8723	FEHLER SDO: Subindex existiert nicht	
0x8724	FEHLER SDO: Schreibzugriff - Parameterwert außerhalb des Bereichs	
0x8725	FEHLER SDO: Schreibzugriff - Parameterwert über dem Bereich	
0x8726	FEHLER SDO: Schreibzugriff - Parameterwert unterhalb des Bereichs	
0x8727	FEHLER SDO: Maximalwert kleiner als Minimalwert	
0x8728	FEHLER SDO: Allgemeiner Fehler	
0x8729	FEHLER SDO: Daten können nicht an die Applikation übertragen bzw. dort gespeichert werden	
0x872A	FEHLER SDO: Daten können nicht an die Applikation übertragen werden, da diese zentral gesteuert wird	
0x872B	FEHLER SDO: Daten können nicht an die Applikation übertragen werden, da diese zentral gesteuert wird	
0x872C	FEHLER SDO: Die dynamische Erzeugung eines Ojektverzeichnisses ist fehlgeschlagen bzw. das Objektverzeichnis fehlt.	
0x872D	FEHLER SDO: Unbekannter Code	
0x8750	LADDR ungültig	
0x8751	Typ ist nicht BYTE in ANY-Pointer	
0x8752	Auf der LADDR angegebenen logischen Adresse existiert kein DP-Modul / PROFINET-IO-Gerät, von dem aus konsistente Daten gelesen werden können	
0x8753	Zugriffsfehler erkannt, während auf die E/A-Geräte zugegriffen wurde	
0x8754	Slave-Fehler am externen DP-Slave	
0x8755	Länge der SFB-Daten entspricht nicht denen der Benutzerdaten	
0x8756	System-Fehler am externen DP-Slave	

ErrorID	Beschreibung	Bemerkung
0x8757	System-Fehler am externen DP-Slave	
0x8758	Die Daten wurden noch nicht vom Modul gelesen	
0x8759	System-Fehler am externen DP-Slave	
0x875A	Keine Systemressourcen verfügbar	
0x8799	SDO: Anderer Fehler aufgetreten >>> Info1 und Info2 Daten überprüfen	

14.6 PLCopen Parameter

PN	Name	Datentyp	R/W	Beschreibung
1	CommandedPosition	REAL	R	Sollposition Zugriff auf: `"Axis".Control.ControlPositioning.PositionCommanded`
2	SWLimitPos	REAL	R/W	Positive Software Endschalter Position Zugriff auf: `"Axis".AxisConfiguration.PositionLimits.MaxPosition`
3	SWLimitNeg	REAL	R/W	Negative Software Endschalter Position Zugriff auf: `"Axis".AxisConfiguration.PositionLimits.MinPosition`
4	EnableLimitPos	BOOL	R/W	Positiven Endschalter aktivieren Zugriff auf: `"Axis".AxisConfiguration.PositionLimits.EnableMaxPos`
5	EnableLimitNeg	BOOL	R/W	Negativen Endschalter aktivieren Zugriff auf: `"Axis".AxisConfiguration.PositionLimits.EnableMinPos`
6	EnablePosLagMonitoring	BOOL	R/W	Überwachung Schleppabstand aktivieren Funktion wird nicht unterstützt
7	MaxPositionLag	REAL	R/W	Maximaler Schleppabstand Funktion wird nicht unterstützt
8	MaxVelocitySystem	REAL	R	Maximal zulässige Geschwindigkeit der Achse im Bewegungssystem Zugriff auf: `"Axis".AxisConfiguration.DynamicLimits.MaxVelocitySystem`
9	MaxVelocityAppl	REAL	R/W	Maximal zulässige Geschwindigkeit der Achse in der Anwendung Zugriff auf: `"Axis".AxisConfiguration.DynamicLimits.MaxVelocityApplication`

PN	Name	Datentyp	R/W	Beschreibung
10	ActualVelocity	REAL	R	Aktuelle Geschwindigkeit Zugriff auf: `"Axis".Status.StatusPositioning.ActVelocity`
11	CommandedVelocity	REAL	R	Sollgeschwindigkeit Zugriff auf: `"Axis".Control.ControlPositioning.PositionCommanded`
12	MaxAccelerationSystem	REAL	R	Maximal zulässige Beschleunigung der Achse im Bewegungssystem Zugriff auf: `"Axis".AxisConfiguration.DynamicLimits.MaxAccelerationSystem`
13	MaxAccelerationAppl	REAL	R/W	Maximal zulässige Beschleunigung der Achse in der Anwendung Zugriff auf: `"Axis".AxisConfiguration.DynamicLimits.MaxAccelerationApplication`
14	MaxDecelerationSystem	REAL	R	Maximal zulässige Verzögerung der Achse im Bewegungssystem Zugriff auf: `"Axis".AxisConfiguration.DynamicLimits.MaxDecelerationSystem`
15	MaxDecelerationAppl	REAL	R/W	Maximal zulässige Verzögerung der Achse in der Anwendung Zugriff auf: `"Axis".AxisConfiguration.DynamicLimits.MaxDecelerationApplication`
16	MaxJerkSystem	REAL	R	Maximal zulässiger Ruck der Achse im Bewegungssystem Zugriff auf: `"Axis".AxisConfiguration.DynamicLimits.MaxJerkSystem`
17	MaxJerkAppl	REAL	R/W	Maximal zulässiger Ruck der Achse in der Anwendung Zugriff auf: `"Axis".AxisConfiguration.DynamicLimits.MaxJerkApplication`

14.7 VIPA-spezifische Parameter

Positionierachse: Yaskawa Sigma5 / Sigma7

PN	Name	Datentyp	R/W	Beschreibung
900	HomingDone	BOOL	R/W	In der Struktur *AxisREF* können Sie *HomingDone* setzen bzw. rücksetzen.
1000	ErrorCode	WORD	R/W	
1001	HomeOffset	DINT	R/W	

PN	Name	Datentyp	R/W	Beschreibung
1002	HomingMethod	BYTE	R/W	
1003	SpeedSearchSwitch	DWORD	R/W	
1004	SpeedSearchZero	DWORD	R/W	
1005	HomingAcc	DWORD	R/W	
1006	PositiveTorqueLimit	UINT	R/W	
1007	NegativeTorqueLimit	UINT	R/W	
1008	MotorRatedTorque	DWORD	R/W	
1009	FollowingErrorWindow	DWORD	R/W	
1010	FollowingErrorTimeOut	WORD	R/W	
1011	PositionWindow	DWORD	R/W	
1012	PositionTime	WORD	R/W	
1013	Min Position Limit	DWORD	R/W	
1014	Max Position Limit	DWORD	R/W	
1015	Digital outputs / physical outputs	DWORD	R/W	
1016	Digital outputs / mask	DWORD	R/W	
1017	Quick stop deceleration	DWORD	R/W	

Drehzahlachse: Yaskawa V1000

PN	Name	Datentyp	R/W	Beschreibung
1000	ErrorCode	WORD	R/W	
1001	Vel. Max. Amount	INT	R/W	

15 Antriebssteuerung - *Simple Motion Control Library*

15.1 Übersicht

Leistungsmerkmale

Mit den Bausteinen der *Simple Motion Control Library* können Sie auf einfache Weise ohne Detailwissen Antriebe in Ihre Applikationen integrieren. Hierbei werden verschiedene Antriebe und Bussystem unterstützt. Mittels der PLCopen-Bausteine können Sie einfache Antriebsaufgaben in Ihrer Steuerung realisieren. Dieses System bietet folgenden Leistungsumfang:

- Einsetzbar im VIPA *SPEED7 Studio*
- Umsetzung von einfachen Antriebsfunktionen
  - Einschalten bzw. Ausschalten
  - Drehzahlvorgabe
  - Relative bzw. absolute Positionierung
  - Referenzfahrt (Homing)
  - Lesen und Schreiben von Parametern
  - Abfrage von Achsposition und Status
- Einfache Inbetriebnahme und Diagnose ohne detaillierte Kenntnisse der Antriebe
- Unterstützung verschiedener Antriebe und Feldbusse
- Visualisierung einzelner Achsen
- Skalierbar durch Einsatz von PLCopen-Bausteinen

Struktur

Die *Simple Motion Control Library* ist in folgende Gruppen gegliedert:

- Axis Control
  - Allgemeine Bausteine zur Steuerung der Antriebe.
- Sigma5 EtherCAT
  - Spezifische Bausteine für den Einsatz von *Sigma-5*-Antrieben, welche über EtherCAT angebunden sind.
- Sigma7 EtherCAT
  - Spezifische Bausteine für den Einsatz von *Sigma-7S*-Antrieben, welche über EtherCAT angebunden sind.
  - Spezifische Bausteine für den Einsatz von *Sigma-7W*-Antrieben, welche über EtherCAT angebunden sind.
- Sigma5+7 PROFINET
  - Spezifische Bausteine für den Einsatz von *Sigma-5* bzw. *Sigma-7*-Antrieben, welche über PROFINET angebunden sind.
- Sigma5+7 PulseTrain
  - Spezifischer Baustein für den Einsatz von *Sigma-5* bzw. *Sigma-7*-Antrieben, welche über Pulse Train angebunden sind.
- V1000 PWM
  - Spezifischer Baustein für den Einsatz von *V1000*-Frequenzumrichter, welche über PWM angebunden sind.
- V1000 Modbus RTU
  - Spezifische Bausteine für den Einsatz von *V1000*-Frequenzumrichter, welche über Modbus-RTU angebunden sind.
- Inverter EtherCAT
  - Spezifische Bausteine für den Einsatz von Frequenzumrichter, welche über EtherCAT angebunden sind.

15.2 Einsatz *Sigma-5/7* EtherCAT

15.2.1 Einsatz *Sigma-5* EtherCAT

15.2.1.1 Übersicht

Voraussetzung

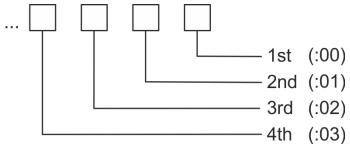
- SPEED7 Studio ab V1.6.1
- CPU mit EtherCAT-Master wie z.B. CPU 015-CEFNR00
- *Sigma-5*-Antrieb mit EtherCAT-Optionskarte

Schritte der Projektierung

1. Parameter am Antrieb einstellen
   - Die Einstellung der Parameter hat mit dem Softwaretool *Sigma Win+* zu erfolgen.
2. Hardwarekonfiguration im VIPA *SPEED7 Studio*
   - Projektierung einer CPU mit EtherCAT-Master-Funktionalität.
   - Projektierung eines *Sigma-5* EtherCAT-Antriebs.
   - Projektierung der EtherCAT-Anbindung über *SPEED7 EtherCAT Manager*.
3. Programmierung im VIPA *SPEED7 Studio*
   - *Init*-Baustein zur Konfiguration der Achse beschalten.
   - *Kernel*-Baustein zur Kommunikation mit der Achse beschalten.
   - Bausteine für die Bewegungsabläufe beschalten.

15.2.1.2 Parameter am Antrieb einstellen

Parameter-Digits



VORSICHT!
Vor der Erstinbetriebnahme müssen Sie Ihren Antrieb mit dem Softwaretool *Sigma Win+* an Ihre Applikation anpassen! Näheres hierzu finden Sie im Handbuch zu ihrem Antrieb.

Zur Abstimmung auf die *Simple Motion Control Library* sind folgende Parameter über *Sigma Win+* einzustellen:

Sigma-5 (20Bit Encoder)

Servopack Parameter	Adresse:Digit	Name	Wert
Pn205	(2205h)	Multiturn Limit Setting	65535
Pn20E	(220Eh)	Electronic Gear Ratio (Numerator)	1
Pn210	(2210h)	Electronic Gear Ratio (Denominator)	1
PnB02	(2701h:01)	Position User Unit (Numerator)	1
PnB04	(2701h:02)	Position User Unit (Denominator)	1
PnB06	(2702h:01)	Velocity User Unit (Numerator)	1
PnB08	(2702h:02)	Velocity User Unit (Denominator)	1
PnB0A	(2703h:01)	Acceleration User Unit (Numerator)	1
PnB0C	(2703h:02)	Acceleration User Unit (Denominator)	1

Bitte beachten Sie, dass Sie gemäß ihren Anforderungen die entsprechende Fahrtrichtung für Ihren Antrieb freigeben. Verwenden Sie hierzu die Parameter Pn50A (P-OT) bzw. Pn50B (N-OT) in Sigma Win+.

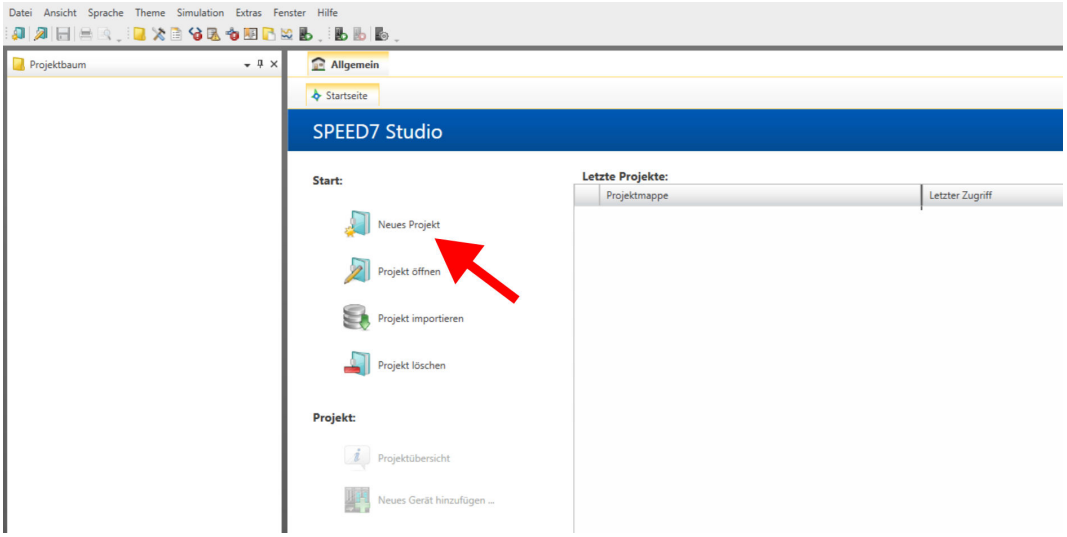
15.2.1.3 Einsatz im VIPA *SPEED7 Studio*

15.2.1.3.1 Hardware-Konfiguration

CPU im Projekt anlegen

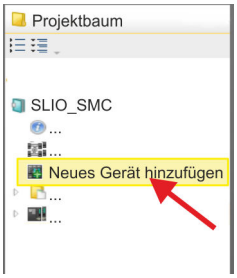
Bitte verwenden Sie für die Projektierung das *SPEED7 Studio* ab V1.6.1.

1. Starten Sie das *SPEED7 Studio*.

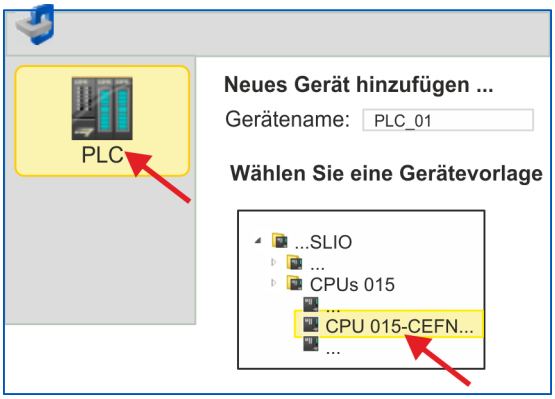


2. Erstellen sie auf der Startseite mit *"Neues Projekt"* ein neues Projekt und vergeben Sie einen *"Projektnamen"*.

⇨ Ein neues Projekt wird angelegt und in die Sicht *"Geräte und Netze"* gewechselt.



3. Klicken Sie im *Projektbaum* auf *"Neues Gerät hinzufügen ..."*.



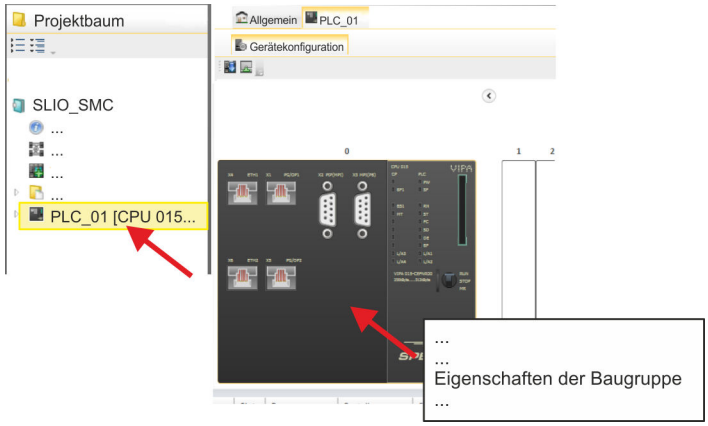
⇨ Es öffnet sich ein Dialog für die Geräteauswahl.

4. Wählen Sie unter den *"Gerätevorlagen"* eine CPU mit EtherCAT-Master-Funktionalität wie z.B. die CPU 015-CEFNR00 und klicken Sie auf [OK].

⇨ Die CPU wird in *"Geräte und Netze"* eingefügt und die *"Gerätekonfiguration"* geöffnet.

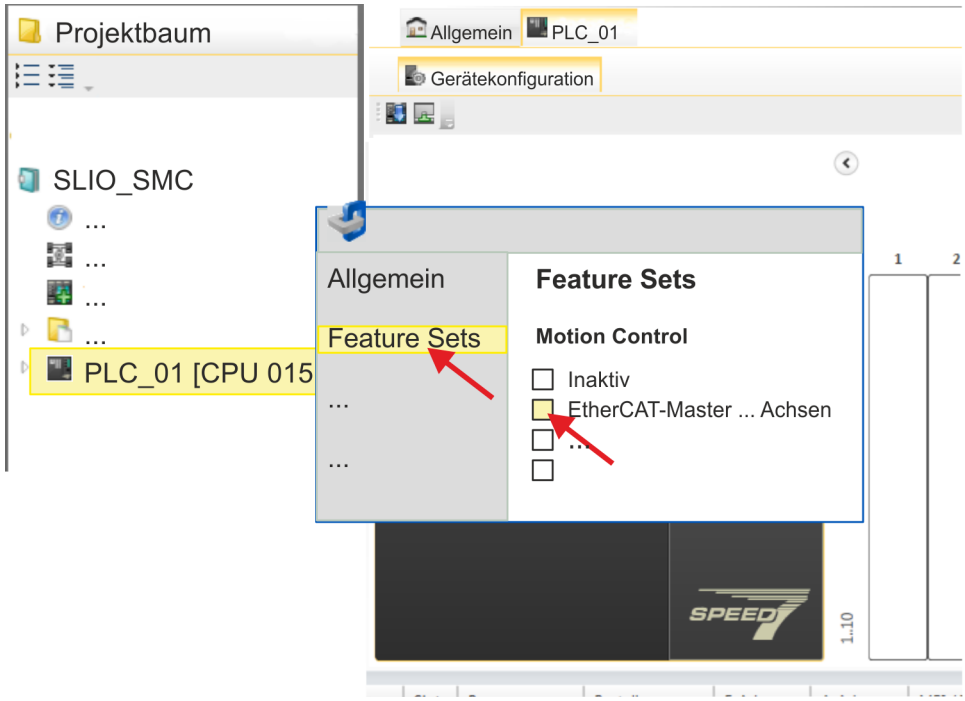
Motion-Control-Funktionen aktivieren

Sofern bei Ihrer CPU die EtherCAT-Master-Funktionalität noch nicht aktiviert ist, erfolgt die Aktivierung nach folgenden Vorgehensweise:



1. Klicken Sie in der *"Gerätekonfiguration"* auf die CPU und wählen Sie *"Kontextmenü ➔ Eigenschaften der Baugruppe"*.

   ⇨ Es öffnet sich der Eigenschaften-Dialog der CPU.



2. Klicken Sie auf *"Feature Sets"* und aktivieren Sie unter *"Motion Control"* einen der Parameter *"EtherCAT-Master ... Achsen"*. Die Anzahl der Achsen ist in diesem Beispiel nicht relevant.

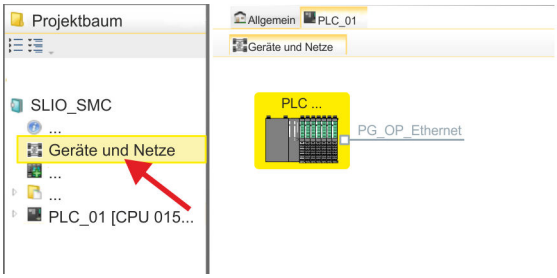
3. Bestätigen Sie Ihre Angaben mit [OK].

   ⇨ Die Motion-Control-Funktionen steht Ihnen nun in Ihrem Projekt zur Verfügung.

VORSICHT!
Bitte beachten Sie, dass bei jeder Änderung der Feature-Set-Einstellungen systembedingt das EtherCAT-Feldbus-System zusammen mit der Motion-Control-Konfiguration aus Ihrem Projekt gelöscht werden!

Ethernet-PG/OP-Kanal parametrieren

1. Klicken Sie im *Projektbaum* auf *"Geräte und Netze"*.
   ⇨ Sie erhalten eine grafische Objekt-Ansicht Ihrer CPU.



2. Klicken Sie auf das Netzwerk *"PG_OP_Ethernet"*.
3. Wählen Sie *"Kontextmenü ➔ Eigenschaften der Schnittstelle"*.
   ⇨ Es öffnet sich ein Dialogfenster. Hier können Sie IP-Adressdaten für Ihren Ethernet-PG/OP-Kanal angeben. Gültige IP-Adress-Parameter erhalten Sie von Ihrem Systemadministrator.
4. Bestätigen Sie Ihre Eingabe mit [OK].
   ⇨ Die IP-Adressdaten werden in Ihr Projekt übernommen und in *"Geräte und Netze"* unter *"Lokale Baugruppen"* aufgelistet.

   Nach der Übertragung Ihres Projekts ist Ihre CPU über die angegebenen IP-Adressdaten via Ethernet-PG/OP-Kanal erreichbar.

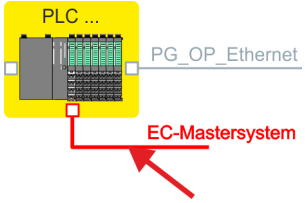
ESI-Datei installieren

Damit der *Sigma-5* EtherCAT Antrieb im *SPEED7 EtherCAT Manager* konfiguriert werden kann, muss die entsprechende ESI-Datei installiert sein. In der Regel wird das *SPEED7 Studio* mit aktuellen ESI-Dateien ausgeliefert und Sie können diesen Teil überspringen. Sollte Ihre ESI-Datei veraltet sein, finden Sie die aktuellste ESI-Datei für den *Sigma-5* EtherCAT Antrieb unter *www.yaskawa.eu.com* unter *"Service ➔ Drives & Motion Software"*.

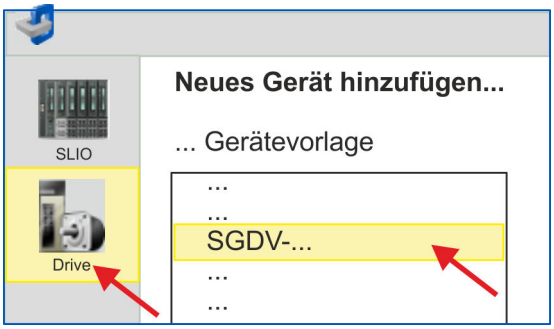
1. Laden Sie die zu Ihrem Antrieb passende ESI-Datei herunter. Entpacken Sie diese falls erforderlich.
2. Gehen Sie in Ihr *SPEED7 Studio*.
3. Öffnen Sie mit *"Extras ➔ Gerätebeschreibungsdatei installieren (EtherCAT - ESI)"* das zugehörige Dialogfenster.
4. Geben Sie unter *"Quellpfad"* die ESI-Datei an und installieren Sie diese mit [Installieren].
   ⇨ Die Geräte der ESI-Datei steht Ihnen nun zur Verfügung.

***Sigma-5* Antrieb hinzufügen**

1. Klicken Sie im Projektbaum auf *"Geräte und Netze"*.
2. Klicken Sie hier auf *"EC-Mastersystem"* und wählen sie *"Kontextmenü ➔ Neues Gerät hinzufügen"*.



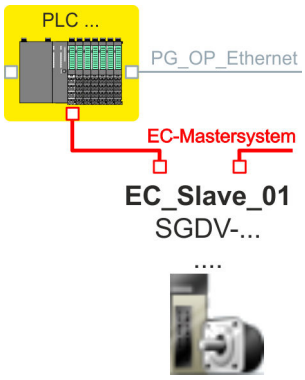
   ⇨ Es öffnet sich die Gerätevorlage zur Auswahl eines EtherCAT-Devices.



3. Wählen Sie Ihren *Sigma-5* Antrieb aus:

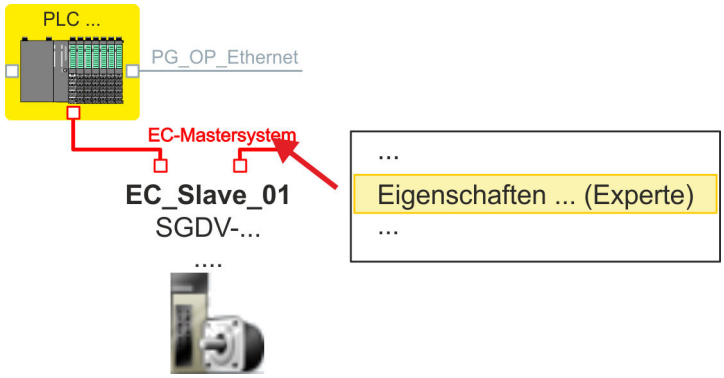
- SGDV-xxxxE5...
- SGDV-xxxxE1...

Bestätigen Sie Ihre Angaben mit [OK]. Sollte Ihr Antrieb nicht vorhanden sein, müssen Sie die entsprechende ESI-Datei wie weiter oben beschrieben installieren.



⇨ Der *Sigma-5* Antrieb wird an Ihr EC-Mastersystem angebunden.

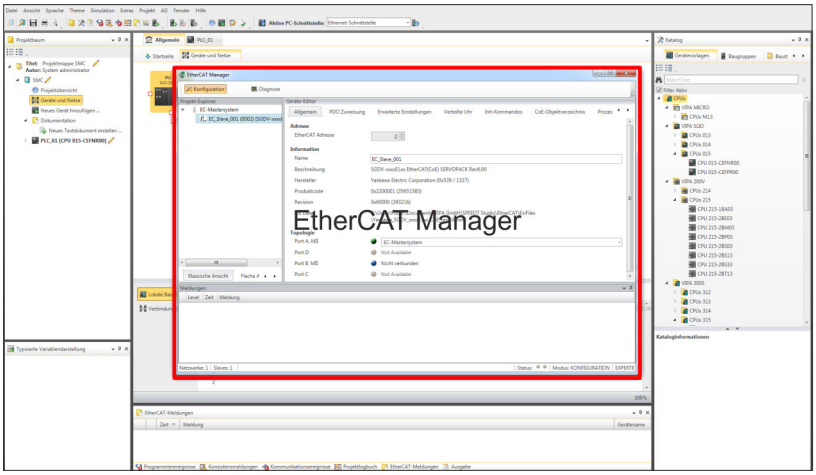
Sigma-5 Antrieb konfigurieren



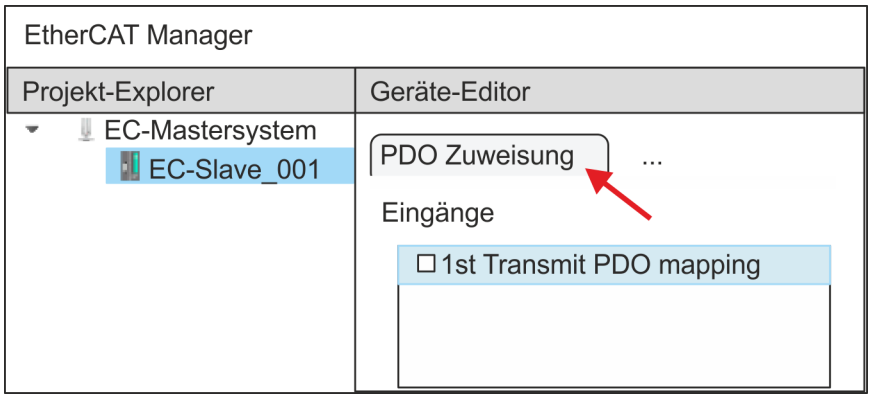
1. Klicken Sie auf *"EC-Mastersystem"* und wählen sie *"Kontextmenü ➔ Eigenschaft des Bussystems (Experte)"*.

> *PDOs können Sie nur im "Experten-Modus" bearbeiten! Ansonsten werden die Schaltflächen ausgeblendet.*

⇨ Der *SPEED7 EtherCAT Manager* wird gestartet. Hier können Sie die EtherCAT-Kommunikation zu Ihrem *Sigma-5* Antrieb konfigurieren.



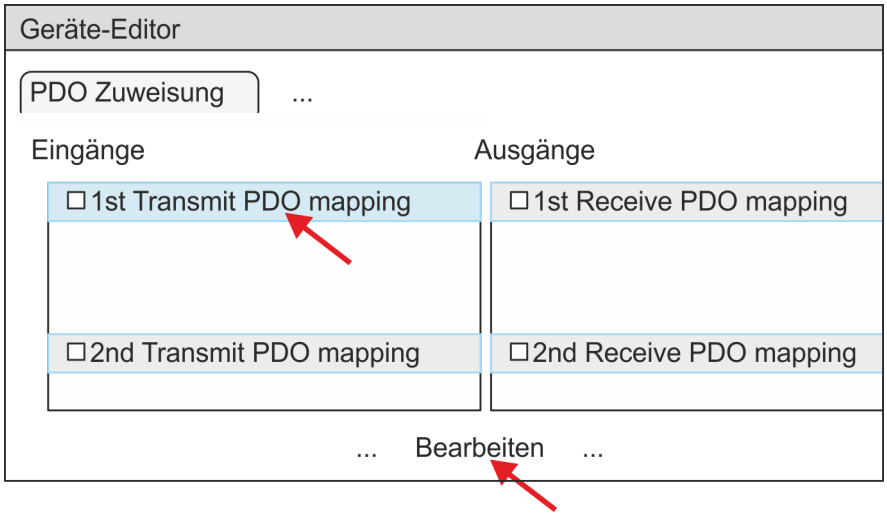
2. Klicken Sie im *SPEED7 EtherCAT Manager* auf den Slave und wählen Sie im *"Geräte-Editor"* den Reiter *"PDO-Zuweisung"* an.



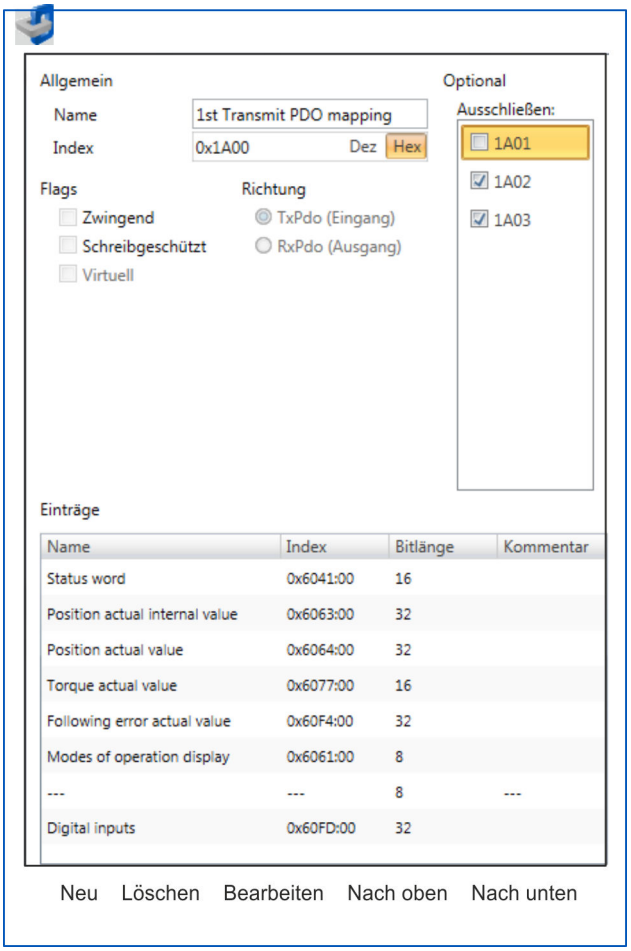
⇨ Dieser Dialog zeigt eine Auflistung aller PDOs.

3. Durch Anwahl des entsprechenden PDO-Mappings können Sie mit [Bearbeiten] die PDOs bearbeiten. Wählen Sie das Mapping *"1st Transmit PDO mapping"* an und klicken Sie auf [Bearbeiten].

Bitte beachten Sie, dass aufgrund der Voreinstellung manche PDOs nicht bearbeitet werden können. Durch Deaktivierung bereits aktivierter PDOs können Sie die Bearbeitung von gesperrten PDOs frei geben.



⇨ Es öffnet sich der Dialog *"PDO bearbeiten"*. Bitte überprüfen Sie hier die aufgeführten PDO-Einstellungen und passen Sie diese ggf. an. Bitte berücksichtigen Sie hierbei auch die Reihenfolge der *"Einträge"* und ergänzen Sie diese entsprechend.



Für die Bearbeitung der *"Einträge"* stehen folgende Funktionen zur Verfügung:

- Neu
  - Hiermit können Sie in einem Dialogfenster einen neuen Eintrag anlegen, indem Sie aus dem *"CoE-Objektverzeichnis"* den entsprechenden Eintrag auswählen und Ihre Einstellungen vornehmen. Mit [OK] wird der Eintrag übernommen und in der Liste der Einträge aufgeführt.
- Löschen
  - Hiermit können Sie den angewählte Eintrag löschen.
- Bearbeiten
  - Hiermit können Sie allgemeinen Daten eines Eintrags bearbeiten.
- Nach oben/unten
  - Hiermit können Sie den angewählten Eintrag in der Liste nach oben bzw. nach unten bewegen.

4. Führen Sie folgende Einstellungen durch:

Eingänge: 1st Transmit PDO 0x1A00

- Allgemein
  - Name: 1st Transmit PDO mapping
  - Index: 0x1A00
- Flags
  - Alles deaktiviert
- Richtung
  - TxPdo (Eingang): aktiviert
- Ausschließen
  Bitte diese Einstellungen beachten, da ansonsten die PDO-Mappings nicht zeitgleich aktiviert werden können!
  - 1A01: deaktiviert
- Einträge

Name	Index	Bitlänge
Status word	0x6041:00	16Bit
Position actual internal value	0x6063:00	32Bit
Position actual value	0x6064:00	32Bit
Torque actual value	0x6077:00	16Bit
Following error actual value	0x60F4:00	32Bit
Modes of operation display	0x6061:00	8Bit
---	---	8Bit
Digital inputs	0x60FD:00	32Bit

Schließen Sie den Dialog *"PDO bearbeiten"* mit [OK].

5. Wählen Sie das Mapping *"2nd Transmit PDO mapping"* an und klicken Sie auf [Bearbeiten]. Führen Sie folgende Einstellungen durch:

Eingänge: 2nd Transmit PDO 0x1A01

- Allgemein
  - Name: 2nd Transmit PDO mapping
  - Index: 0x1A01
- Flags
  - Alles deaktiviert
- Richtung
  - TxPdo (Eingang): aktiviert
- Ausschließen

  Bitte diese Einstellungen beachten, da ansonsten die PDO-Mappings nicht zeitgleich aktiviert werden können!
  - 1A00: deaktiviert
  - 1A02: deaktiviert
  - 1A03: deaktiviert
- Einträge

Name	Index	Bitlänge
Touch probe status	0x60B9:00	16Bit
Touch probe 1 position value	0x60BA:00	32Bit
Touch probe 2 position value	0x60BC:00	32Bit
Velocity actual value	0x606C:00	32Bit

Schließen Sie den Dialog *"PDO bearbeiten"* mit [OK].

6. Wählen Sie das Mapping *"1st Receive PDO mapping"* an und klicken Sie auf [Bearbeiten]. Führen Sie folgende Einstellungen durch:

Ausgänge: 1st Receive PDO 0x1600

- Allgemein
  - Name: 1st Receive PDO mapping
  - Index: 0x1600
- Flags
  - Alles deaktiviert
- Richtung
  - RxPdo (Ausgang): aktiviert
- Ausschließen

  Bitte diese Einstellungen beachten, da ansonsten die PDO-Mappings nicht zeitgleich aktiviert werden können!
  - 1601: deaktiviert
  - 1602: deaktiviert
  - 1603: deaktiviert
- Einträge

Name	Index	Bitlänge
Control word	0x6040:00	16Bit
Target position	0x607A:00	32Bit
Target velocity	0x60FF:00	32Bit
Modes of operation	0x6060:00	8Bit
---	---	8Bit
Touch probe function	0x60B8:00	16Bit

Schließen Sie den Dialog *"PDO bearbeiten"* mit [OK].

7. Wählen Sie das Mapping *"2nd Receice PDO mapping"* an und klicken Sie auf [Bearbeiten]. Führen Sie folgende Einstellungen durch:

Ausgänge: 2nd Receive PDO 0x1601

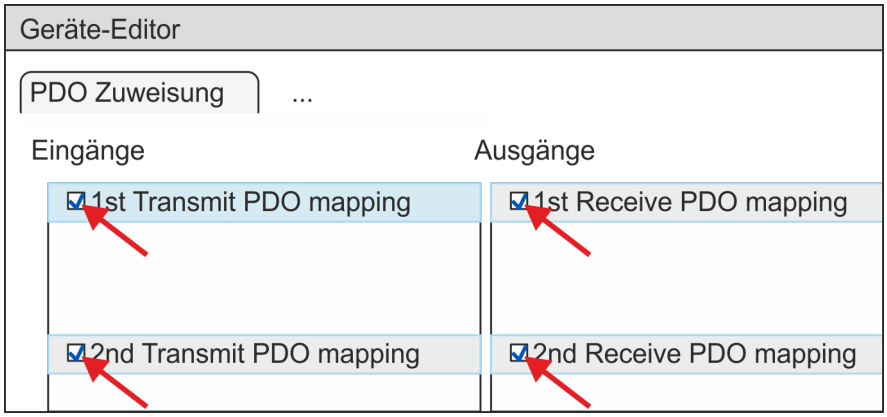
- Allgemein
  - Name: 2nd Receive PDO mapping
  - Index: 0x1601
- Flags
  - Alles deaktiviert
- Richtung
  - RxPdo (Ausgang): aktiviert
- Ausschließen

  Bitte diese Einstellungen beachten, da ansonsten die PDO-Mappings nicht zeitgleich aktiviert werden können!
  - 1600: deaktiviert
  - 1602: aktiviert
  - 1603: aktiviert
- Einträge

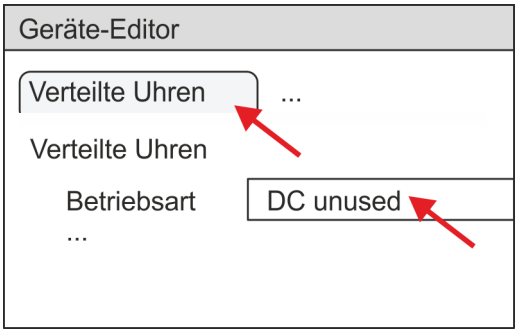
Name	Index	Bitlänge
Profile velocity	0x6081:00	32Bit
Profile acceleration	0x6083:00	32Bit
Profile deceleration	0x6084:00	32Bit

Schließen Sie den Dialog *"PDO bearbeiten"* mit [OK].

8. Aktivieren Sie in PDO-Zuweisung die PDOs 1 und 2 für die Ein und Ausgänge. Alle nachfolgenden PDOs müssen deaktiviert bleiben. Sollte dies nicht möglich sein, überprüfen Sie bitte den jeweiligen PDO-Parameter *"Ausschließen"*.

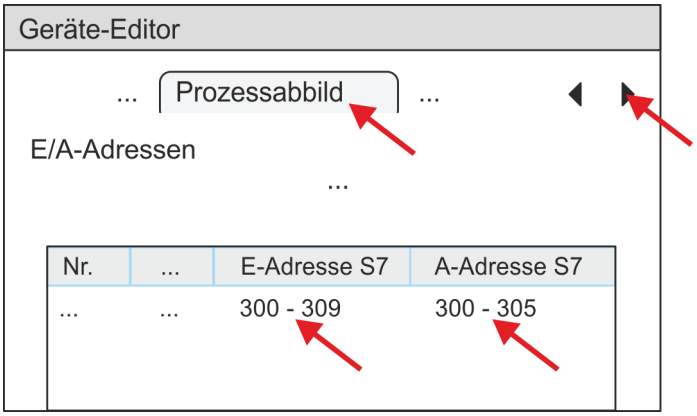


9. Wählen Sie im *"Geräte-Editor"* des *SPEED7 EtherCAT Manager* den Reiter *"Verteilte Uhren"* an und stellen Sie *"DC unused"* als *"Betriebsart"* ein.



10. Wählen Sie im *"Geräte-Editor"* über die Pfeiltaste den Reiter *"Prozessabbild"* an und notieren Sie sich für die Parameter des Bausteins FB 871 - VMC_InitSigma5_EC folgende PDO-Anfangsadressen:

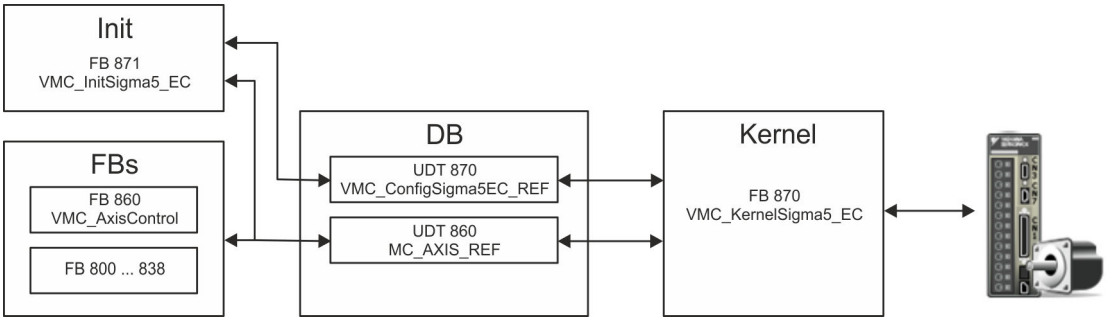
- *"E-Adresse S7"* → *"InputsStartAddressPDO"*
- *"A-Adresse S7"* → *"OutputsStartAddressPDO"*



11. Indem Sie den Dialog des *SPEED7 EtherCAT Manager* mit [X] schließen, wird die Konfiguration in das *SPEED7 Studio* übernommen.

15.2.1.3.2 Anwender-Programm

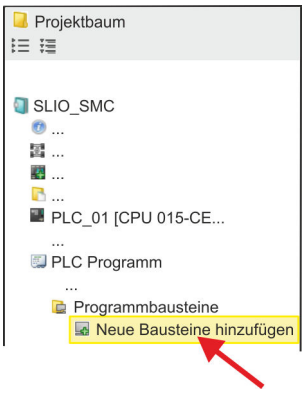
Programmstruktur



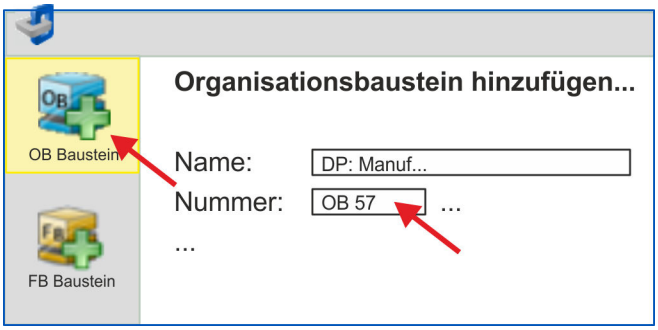
- DB
  Für jede Achse ist ein Datenbaustein (Achs-DB) für Konfiguration und Statusdaten anzulegen. Der Datenbaustein besteht aus folgenden Datenstrukturen:
  – UDT 870 - *VMC_Config*Sigma5EC_REF
    Die Datenstruktur beschreibt den Aufbau der Konfiguration des Antriebs.
    Spezifische Datenstruktur für *Sigma-5* EtherCAT.
  – UDT 860 - *MC_AXIS_REF*
    Die Datenstruktur beschreibt den Aufbau der Parameter und Statusinformationen von Antrieben.
    Allgemeine Datenstruktur für alle Antriebe und Bussysteme.
- FB 871 - *VMC_Init*Sigma5_EC
  – Der *Init*-Baustein dient zur Konfiguration einer Achse.
  – Spezifischer Baustein für *Sigma-5* EtherCAT.
  – Die Konfigurationsdaten für die Initialisierung sind im *Achs-DB* abzulegen.
- FB 870 - *VMC_Kernel*Sigma5_EC
  – Der *Kernel*-Baustein kommuniziert mit dem Antrieb über das entsprechende Bussystem, verarbeitet die Benutzeraufträge und liefert Statusmeldungen zurück.
  – Spezifischer Baustein für *Sigma-5* EtherCAT.
  – Der Austausch der Daten erfolgt mittels des *Achs-DB*.
- FB 860 - *VMC_AxisControl*
  – Universal-Baustein für alle Antriebe und Bussysteme.
  – Unterstützt einfache Bewegungskommandos und liefert alle relevanten Statusmeldungen.
  – Der Austausch der Daten erfolgt mittels des *Achs-DB*.
  – Über die Instanzdaten des Bausteins können Sie zur Bewegungssteuerung und Statusabfrage eine Visualisierung anbinden.
  – Zusätzlich zum FB 860 - *VMC_AxisControl* haben Sie die Möglichkeit *PLCopen*-Bausteine zu nutzen.
- FB 800 ... FB 838 - *PLCopen*
  – Die PLCopen-Bausteine dienen zur Programmierung von Bewegungsabläufen und Statusabfragen.
  – Allgemeine Bausteine für alle Antriebe und Bussysteme.

Programmierung

Bausteine in Projekt kopieren

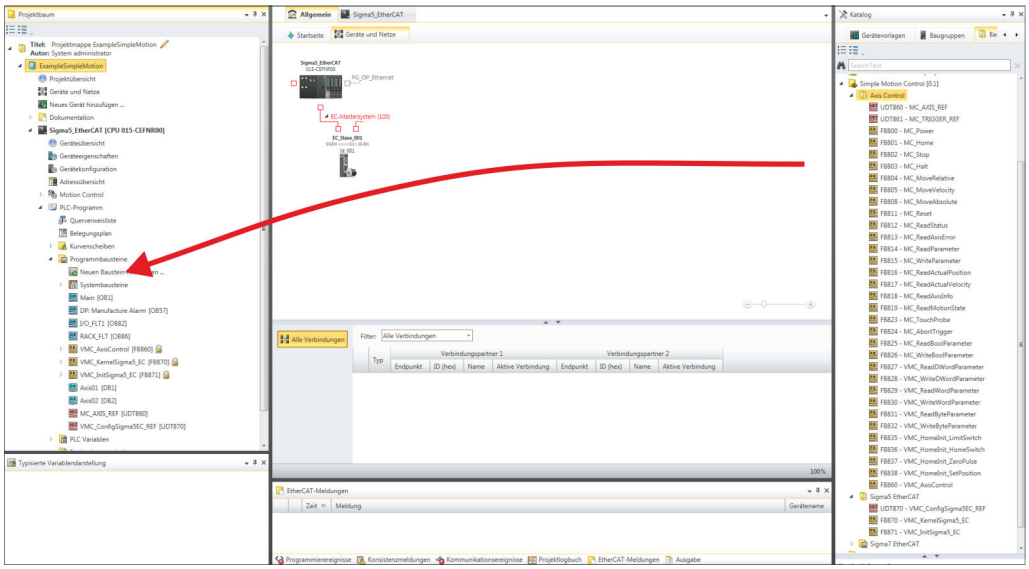


1. Klicken Sie im *Projektbaum* innerhalb der CPU unter *"PLC-Programm"*, *"Programmbausteine"* auf *"Neuen Baustein hinzufügen"*.



⇨ Das Dialogfenster *"Baustein hinzufügen"* öffnet sich.

2. Wählen Sie den Bausteintyp *"OB Baustein"* und fügen Sie nacheinander OB 57, OB 82 und OB 86 Ihrem Projekt hinzu.



3. Öffnen Sie im *"Katalog"* unter *"Bausteine"* *"Simple Motion Control"* und ziehen Sie per Drag&Drop folgende Bausteine in *"Programmbausteine"* des *Projektbaums*:

- *Sigma-5* EtherCAT:
  - UDT 870 - VMC_ConfigSigma5EC_REF
  - FB 870 - VMC_KernelSigma5_EC
  - FB 871 - VMC_InitSigma5_EC
- Axis Control
  - UDT 860 - MC_AXIS_REF
  - Bausteine für die gewünschten Bewegungsabläufe

Achs-DB anlegen

1. Fügen Sie Ihrem Projekt einen neuen DB als *Achs-DB* hinzu. Klicken Sie hierzu im *Projektbaum* innerhalb der CPU unter *"PLC-Programm"*, *"Programmbausteine"* auf *"Neuen Baustein hinzufügen"*, wählen Sie den Bausteintyp *"DB Baustein"* und vergeben Sie diesem den Namen "Axis01". Die DB-Nr. können Sie frei wählen wie z.B. DB 10.

⇨ Der Baustein wird angelegt und geöffnet.

2. ■ Legen Sie in "Axis01" die Variable "Config" vom Typ UDT 870 an. Dies sind spezifische Achs-Konfigurationsdaten.
■ Legen Sie in "Axis01" die Variable "Axis" vom Typ UDT 860 an. Während des Betriebs werden hier alle Betriebsdaten der Achse abgelegt.

Axis01 [DB10]
Bausteinstruktur

		Adr...	Name	Datentyp	...
		...	Config	UDT	[870]
		...	Axis	UDT	[860]

OB 1

Konfiguration der Achse

Öffnen Sie den OB 1 und programmieren Sie folgende FB-Aufrufe mit zugehörigen DBs:

FB 871 - VMC_InitSigma5_EC, DB 871 *Kap. 15.2.1.4.3 "FB 871 - VMC_InitSigma5_EC - Sigma-5 EtherCAT Initialisierung" Seite 596*

Geben Sie unter *InputsStartAddressPDO* bzw. *OutputsStartAddressPDO* die Adresse aus dem *SPEED7 EtherCAT Manager* an. *590*

```
⇨ CALL  "VMC_InitSigma5_EC" , "DI_InitSgm5ETC01"
  Enable                :="InitS5EC1_Enable"
  LogicalAddress        :=300
  InputsStartAddressPDO :=300 (EtherCAT-Man.: E-Adresse S7)
  OutputsStartAddressPDO:=300 (EtherCAT-Man.: A-Adresse S7)
  EncoderType           :=1
  EncoderResolutionBits :=20
  FactorPosition        :=1.048576e+006
  FactorVelocity        :=1.048576e+006
  FactorAcceleration    :=1.048576e+002
  OffsetPosition        :=0.000000e+000
  MaxVelocityApp        :=5.000000e+001
  MaxAccelerationApp    :=1.000000e+002
  MaxDecelerationApp    :=1.000000e+002
  MaxVelocityDrive      :=6.000000e+001
  MaxAccelerationDrive  :=1.500000e+002
  MaxDecelerationDrive  :=1.500000e+002
  MaxPosition           :=1.048500e+003
  MinPosition           :=-1.048514e+003
  EnableMaxPosition     :=TRUE
  EnableMinPosition     :=TRUE
  MinUserPosition       :="InitS5EC1_MinUserPos"
  MaxUserPosition       :="InitS5EC1_MaxUserPos"
  Valid                 :="InitS5EC1_Valid"
  Error                 :="InitS5EC1_Error"
  ErrorID               :="InitS5EC1_ErrorID"
  Config                :="Axis01".Config
  Axis                  :="Axis01".Axis
```

Kernel für Achse beschalten

Der *Kernel* verarbeitet die Benutzerkommandos und gibt sie entsprechend aufbereitet an den Antrieb über das jeweilige Bussystem weiter.

FB 870 - VMC_KernelSigma5_EC, DB 870 *Kap. 15.2.1.4.2 "FB 870 - VMC_KernelSigma5_EC - Sigma-5 EtherCAT Kernel" Seite 596*

```
⇨       CALL  "VMC_KernelSigma5_EC" , "DI_KernelSgm5ETC01"
         Init  :="KernelS5EC1_Init"
         Config:="Axis01".Config
         Axis  :="Axis01".Axis
```

Baustein für Bewegungsabläufe beschalten

Zur Vereinfachung soll hier die Beschaltung des FB 860 - VMC_AxisControl gezeigt werden. Dieser Universalbaustein unterstützt einfache Bewegungskommandos und liefert Statusmeldungen zurück. Die Ein- und Ausgänge können Sie individuell beschalten. Bitte geben Sie unter *"Axis"* die Referenz zu den entsprechenden Achsdaten im *Achs-DB* an.

FB 860 - VMC_AxisControl, DB 860 *Kap. 15.8.2.2 "FB 860 - VMC_AxisControl - Control-Baustein Achskontrolle" Seite 754*

```
⇨       CALL  "VMC_AxisControl" , "DI_AxisControl01"
         AxisEnable       :="AxCtrl1_AxisEnable"
         AxisReset        :="AxCtrl1_AxisReset"
         HomeExecute      :="AxCtrl1_HomeExecute"
         HomePosition     :="AxCtrl1_HomePosition"
         StopExecute      :="AxCtrl1_StopExecute"
         MvVelocityExecute:="AxCtrl1_MvVelExecute"
         MvRelativeExecute:="AxCtrl1_MvRelExecute"
         MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
         PositionDistance :="AxCtrl1_PositionDistance"
         Velocity         :="AxCtrl1_Velocity"
         Acceleration     :="AxCtrl1_Acceleration"
         Deceleration     :="AxCtrl1_Deceleration"
         JogPositive      :="AxCtrl1_JogPositive"
         JogNegative      :="AxCtrl1_JogNegative"
         JogVelocity      :="AxCtrl1_JogVelocity"
         JogAcceleration  :="AxCtrl1_JogAcceleration"
         JogDeceleration  :="AxCtrl1_JogDeceleration"
         AxisReady        :="AxCtrl1_AxisReady"
         AxisEnabled      :="AxCtrl1_AxisEnabled"
         AxisError        :="AxCtrl1_AxisError"
         AxisErrorID      :="AxCtrl1_AxisErrorID"
         DriveWarning     :="AxCtrl1_DriveWarning"
         DriveError       :="AxCtrl1_DriveError"
         DriveErrorID     :="AxCtrl1_DriveErrorID"
         IsHomed          :="AxCtrl1_IsHomed"
         ModeOfOperation  :="AxCtrl1_ModeOfOperation"
         PLCopenState     :="AxCtrl1_PLCopenState"
         ActualPosition   :="AxCtrl1_ActualPosition"
         ActualVelocity   :="AxCtrl1_ActualVelocity"
         CmdDone          :="AxCtrl1_CmdDone"
         CmdBusy          :="AxCtrl1_CmdBusy"
         CmdAborted       :="AxCtrl1_CmdAborted"
         CmdError         :="AxCtrl1_CmdError"
         CmdErrorID       :="AxCtrl1_CmdErrorID"
         DirectionPositive:="AxCtrl1_DirectionPos"
         DirectionNegative:="AxCtrl1_DirectionNeg"
         SWLimitMinActive :="AxCtrl1_SWLimitMinActive"
         SWLimitMaxActive :="AxCtrl1_SWLimitMaxActive"
         HWLimitMinActive :="AxCtrl1_HWLimitMinActive"
         HWLimitMaxActive :="AxCtrl1_HWLimitMaxActive"
         Axis             :="Axis01".Axis
```

Für komplexe Bewegungsaufgaben können Sie die PLCopen-Bausteine verwenden. Hier müssen Sie ebenfalls unter Axis die Referenz zu den Achsdaten im Achs-DB angeben.

Ihr Projekt beinhaltet nun folgende Bausteine:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 860 - VMC_AxisControl mit Instanz-DB

- FB 870 - VMC_KernelSigma5_EC mit Instanz-DB
- FB 871 - VMC_InitSigma5_EC mit Instanz-DB
- UDT 860 - MC_Axis_REF
- UDT 870 - VMC_ConfigSigma5EC_REF

Zeitlicher Ablauf

1. Wählen Sie *"Projekt ➔ Alles übersetzen"* und übertragen Sie das Projekt in Ihre CPU.

   ⇨ Sie können jetzt Ihre Applikation in Betrieb nehmen.

VORSICHT!
Bitte beachten Sie immer die Sicherheitshinweise zu ihrem Antrieb, insbesondere bei der Inbetriebnahme!

2. Bevor eine Achse gesteuert werden kann, muss diese initialisiert werden. Rufen Sie hierzu den *Init*-Baustein FB 871 - VMC_InitSigma5_EC mit *Enable* = TRUE auf.

   ⇨ Der Ausgang *Valid* meldet TRUE zurück. Im Fehlerfall können Sie durch Auswertung der *ErrorID* den Fehler ermitteln.

   Den *Init*-Baustein müssen Sie erneut aufrufen, wenn Sie einen neuen Achs-DB laden oder Parameter am *Init*-Baustein geändert wurden.

Fahren Sie erst fort, wenn der Init-Baustein keinen Fehler meldet!

3. Stellen Sie sicher, dass der *Kernel*-Baustein FB 870 - VMC_KernelSigma5_EC zyklisch aufgerufen wird. Auf diese Weise werden Steuersignale an den Antrieb übergeben und Statusmeldungen übermittelt.
4. Programmieren Sie Ihre Applikation mit dem FB 860 - VMC_AxisControl oder mit den PLCopen Bausteinen.

Steuerung des Antriebs über HMI

Sie haben die Möglichkeit über ein HMI Ihren Antrieb zu steuern. Hierzu gibt es für Movicon eine vorgefertigte Symbolbibliothek für den Zugriff auf den VMC_AxisControl Funktionsbaustein. ↬ *Kap. 15.9 "Antrieb über HMI steuern" Seite 825*

15.2.1.4 Antriebsspezifische Bausteine

Die PLCopen-Bausteine zur Achskontrolle finden Sie hier: Kap. 15.8 "Bausteine zur Achskontrolle" Seite 751

15.2.1.4.1 UDT 870 - VMC_ConfigSigma5EC_REF - *Sigma-5* EtherCAT Datenstruktur Achskonfiguration

Dies ist eine benutzerdefinierte Datenstruktur, die Informationen zu den Konfigurationsdaten beinhaltet. Die UDT ist speziell angepasst an die Verwendung eines *Sigma-5*-Antriebs, welcher über EtherCAT angebunden ist.

15.2.1.4.2 FB 870 - VMC_KernelSigma5_EC - *Sigma-5* EtherCAT Kernel

Beschreibung

Dieser Baustein setzt die Antriebskommandos für eine *Sigma-5* Achse über EtherCAT um und kommuniziert mit dem Antrieb. Je *Sigma-5* Achse ist eine Instanz dieses FBs zyklisch aufzurufen.

Bitte beachten Sie, dass dieser Baustein intern den SFB 238 aufruft.

Im SPEED7 Studio wird dieser Baustein automatisch in Ihr Projekt eingefügt.

Parameter	Deklaration	Datentyp	Beschreibung
Init	INPUT	BOOL	Mit einer Flanke 0-1 wird der Baustein intern zurückgesetzt. Hierbei werden bestehende Bewegungskommandos abgebrochen und der Baustein wird initialisiert.
Config	IN_OUT	UDT870	Datenstruktur zur Übergabe von achsabhängigen Konfigurationsdaten an den *AxisKernel*.
Axis	IN_OUT	MC_AXIS_REF	Datenstruktur zur Übergabe von achsabhängigen Informationen an *AxisKernel* und PLCopen-Bausteine.

15.2.1.4.3 FB 871 - VMC_InitSigma5_EC - *Sigma-5* EtherCAT Initialisierung

Beschreibung

Dieser Baustein dient zur Konfiguration der Achse. Der Baustein ist speziell angepasst an die Verwendung eines *Sigma-5*-Antriebs, welcher über EtherCAT angebunden ist.

Parameter	Deklaration	Datentyp	Beschreibung
Enable	INPUT	BOOL	Freigabe der Initialisierung
LogicalAddress	INPUT	INT	Startadresse der PDO-Eingangsdaten
InputsStartAddressPDO	INPUT	INT	Startadresse der Eingabe-PDOs
OutputsStartAddressPDO	INPUT	INT	Startadresse der Ausgabe-PDOs
EncoderType	INPUT	INT	Encoder-Typ ■ 1: Absolut-Encoder ■ 2: Inkremental-Encoder

Parameter	Deklaration	Datentyp	Beschreibung
EncoderResolutionBits	INPUT	INT	Anzahl der Bits, die einer Geber-Umdrehung entsprechen. Default: 20
FactorPosition	INPUT	REAL	Faktor zur Umrechnung der Position von Benutzereinheiten [u] in Antriebseinheiten [Inkremente] und zurück. Es gilt: $p_{[Inkremente]} = p_{[u]}$ x *FactorPosition* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2701:1 und 0x2701:2 vorgeben können. Dieser sollte 1 sein.
FactorVelocity	INPUT	REAL	Faktor zur Umrechnung der Geschwindigkeit von Benutzereinheiten [u/s] in Antriebseinheiten [Inkremente/s] und zurück. Es gilt: $v_{[Inkremente/s]} = v_{[u/s]}$ x *FactorVelocity* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2702:1 und 0x2702:2 vorgeben können. Dieser sollte 1 sein.
FactorAcceleration	INPUT	REAL	Faktor zur Umrechnung der Beschleunigung von Benutzereinheiten [u/s^2] in Antriebseinheiten [10^{-4} x Inkremente/s^2] und zurück. Es gilt: 10^{-4} x $a_{[Inkremente/s^2]} = a_{[u/s^2]}$ x *FactorAcceleration* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2703:1 und 0x2703:2 vorgeben können. Dieser sollte 1 sein.
OffsetPosition	INPUT	REAL	Offset für die Nullposition [u].
MaxVelocityApp	INPUT	REAL	Maximale Geschwindigkeit der Applikation [u/s]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.
MaxAccelerationApp	INPUT	REAL	Maximale Beschleunigung der Applikation [u/s^2]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.
MaxDecelerationApp	INPUT	REAL	Maximale Verzögerung der Applikation [u/s^2]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.
MaxPosition	INPUT	REAL	Maximale Position für die Überwachung der Softwarelimits [u].
MinPosition	INPUT	REAL	Minimale Position für die Überwachung der Softwarelimits [u].
EnableMaxPosition	INPUT	BOOL	Überwachung maximale Position ■ TRUE: Aktivierung der Überwachung der maximalen Position.
EnableMinPosition	INPUT	BOOL	Überwachung minimale Position ■ TRUE: Aktivierung der Überwachung der minimalen Position.
MinUserPosition	OUTPUT	REAL	Minimale Benutzerposition basierend auf dem minimalen Encoder Wert von 0x80000000 und dem *FactorPosition* [u].

Parameter	Deklaration	Datentyp	Beschreibung
MaxUserPosition	OUTPUT	REAL	Maximale Benutzerposition basierend auf dem maximalen Encoder Wert von 0x7FFFFFFF und dem *Factor-Position* [u].
Valid	OUTPUT	BOOL	Initialisierung ■ TRUE: Initialisierung ist gültig.
Error	OUTPUT	BOOL	■ Fehler – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden. Die Achse wird gesperrt.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Config	IN_OUT	UDT870	Datenstruktur zur Übergabe von achsabhängigen Konfigurationsdaten an den *AxisKernel*.
Axis	IN_OUT	MC_AXIS_REF	Datenstruktur zur Übergabe von achsabhängigen Informationen an *AxisKernel* und PLCopen-Bausteine.

15.2.2 Einsatz *Sigma-7S* EtherCAT

15.2.2.1 Übersicht

Einsatz des Doppelachs-Antriebs *Kap. 15.2.3 "Einsatz Sigma-7W EtherCAT" Seite 619*

Voraussetzung

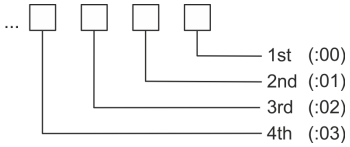
- SPEED7 Studio ab V1.6.1
- CPU mit EtherCAT-Master wie z.B. CPU 015-CEFNR00
- *Sigma-7S*-Antrieb mit EtherCAT-Optionskarte

Schritte der Projektierung

1. Parameter am Antrieb einstellen
   - Die Einstellung der Parameter hat mit dem Softwaretool *Sigma Win+* zu erfolgen.
2. Hardwarekonfiguration im VIPA *SPEED7 Studio*
   - Projektierung einer CPU mit EtherCAT-Master-Funktionalität.
   - Projektierung eines *Sigma-7S* EtherCAT-Antriebs.
   - Projektierung der EtherCAT-Anbindung über *SPEED7 EtherCAT Manager*.
3. Programmierung im VIPA *SPEED7 Studio*
   - *Init*-Baustein zur Konfiguration der Achse beschalten.
   - *Kernel*-Baustein zur Kommunikation mit der Achse beschalten.
   - Bausteine für die Bewegungsabläufe beschalten.

15.2.2.2 Parameter am Antrieb einstellen

Parameter-Digits



VORSICHT!
Vor der Erstinbetriebnahme müssen Sie Ihren Antrieb mit dem Software-tool *Sigma Win+* an Ihre Applikation anpassen! Näheres hierzu finden Sie im Handbuch zu ihrem Antrieb.

Zur Abstimmung auf die *Simple Motion Control Library* sind folgende Parameter über *Sigma Win+* einzustellen:

Sigma-7S (24Bit Encoder)

Servopack Parameter	Adresse:Digit	Name	Wert
Pn205	(2205h)	Multiturn Limit Setting	65535
Pn20E	(220Eh)	ElectronicGear Ratio (Numerator)	16
Pn210	(2210h)	Electronic Gear Ratio (Denominator)	1
PnB02	(2701h:01)	Position User Unit (Numerator)	1
PnB04	(2701h:02)	Position User Unit (Denominator)	1
PnB06	(2702h:01)	Velocity User Unit (Numerator)	1
PnB08	(2702h:02)	Velocity User Unit (Denominator)	1
PnB0A	(2703h:01)	Acceleration User Unit (Numerator)	1
PnB0C	(2703h:02)	Acceleration User Unit (Denominator)	1

Bitte beachten Sie, dass Sie gemäß ihren Anforderungen die entsprechende Fahrtrichtung für Ihren Antrieb freigeben. Verwenden Sie hierzu die Parameter Pn50A (P-OT) bzw. Pn50B (N-OT) in Sigma Win+.

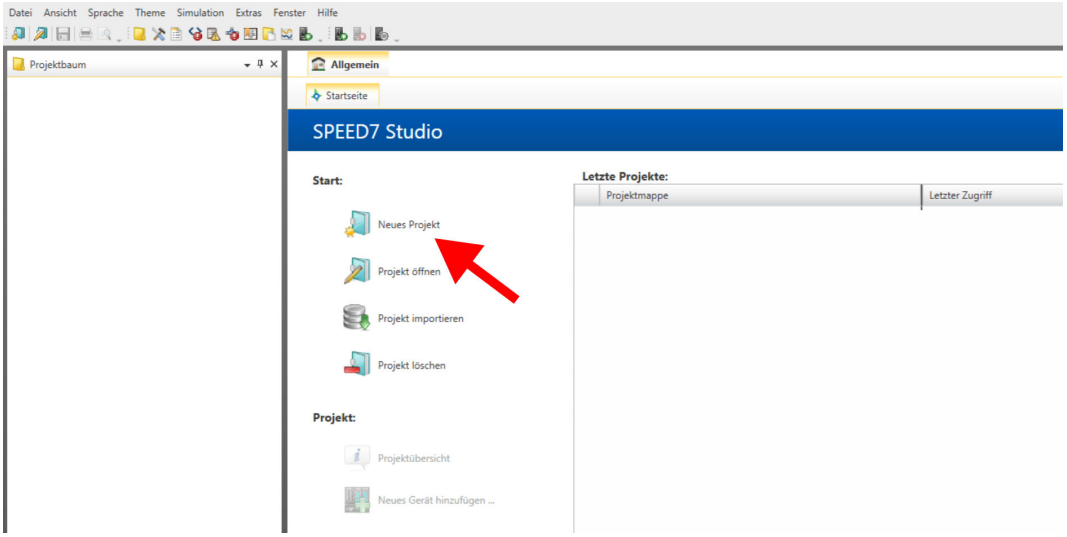
15.2.2.3 Einsatz im VIPA *SPEED7 Studio*

15.2.2.3.1 Hardware-Konfiguration

CPU im Projekt anlegen

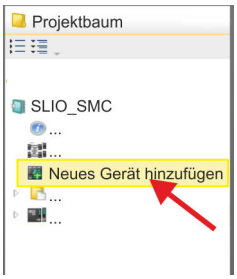
Bitte verwenden Sie für die Projektierung das *SPEED7 Studio* ab V1.6.1.

1. Starten Sie das *SPEED7 Studio*.

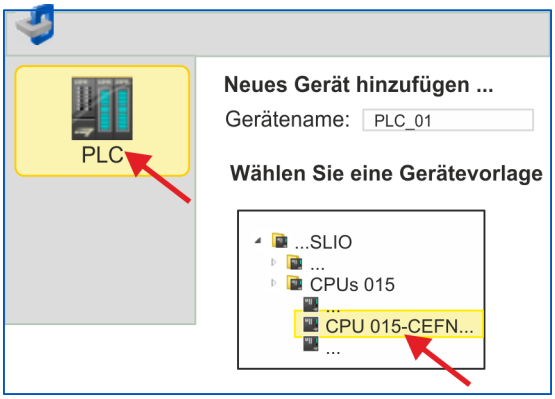


2. Erstellen sie auf der Startseite mit *"Neues Projekt"* ein neues Projekt und vergeben Sie einen *"Projektnamen"*.

⇨ Ein neues Projekt wird angelegt und in die Sicht *"Geräte und Netze"* gewechselt.



3. Klicken Sie im *Projektbaum* auf *"Neues Gerät hinzufügen ..."*.



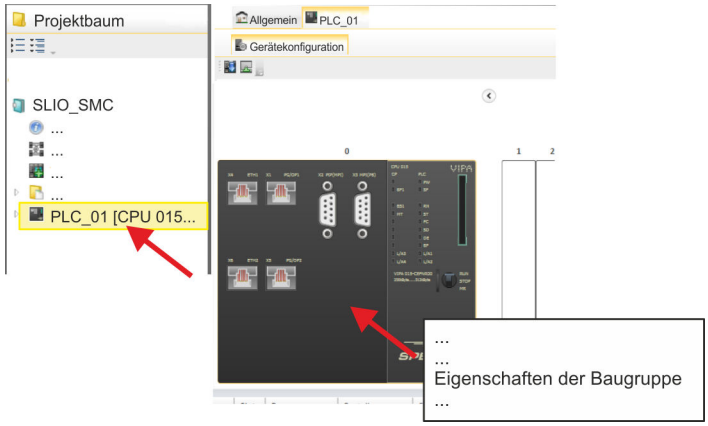
⇨ Es öffnet sich ein Dialog für die Geräteauswahl.

4. Wählen Sie unter den *"Gerätevorlagen"* eine CPU mit EtherCAT-Master-Funktionalität wie z.B. die CPU 015-CEFNR00 und klicken Sie auf [OK].

⇨ Die CPU wird in *"Geräte und Netze"* eingefügt und die *"Gerätekonfiguration"* geöffnet.

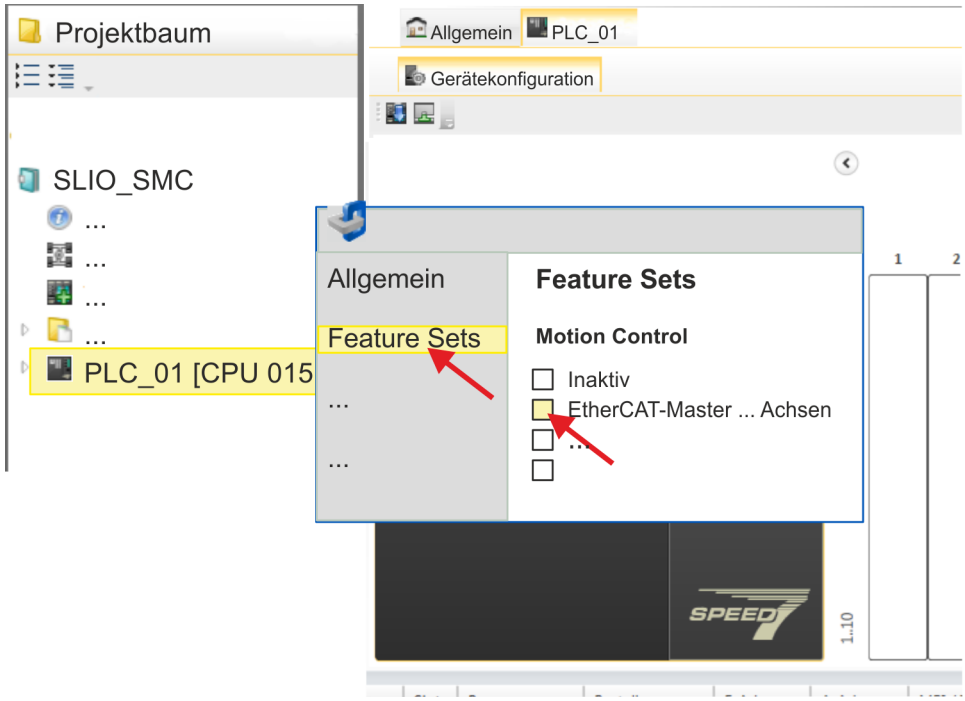
Motion-Control-Funktionen aktivieren

Sofern bei Ihrer CPU die EtherCAT-Master-Funktionalität noch nicht aktiviert ist, erfolgt die Aktivierung nach folgenden Vorgehensweise:



1. Klicken Sie in der *"Gerätekonfiguration"* auf die CPU und wählen Sie *"Kontextmenü ➔ Eigenschaften der Baugruppe"*.

   ⇨ Es öffnet sich der Eigenschaften-Dialog der CPU.



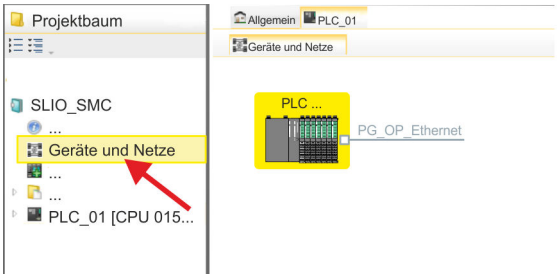
2. Klicken Sie auf *"Feature Sets"* und aktivieren Sie unter *"Motion Control"* einen der Parameter *"EtherCAT-Master ... Achsen"*. Die Anzahl der Achsen ist in diesem Beispiel nicht relevant.
3. Bestätigen Sie Ihre Angaben mit [OK].

   ⇨ Die Motion-Control-Funktionen steht Ihnen nun in Ihrem Projekt zur Verfügung.

VORSICHT!
Bitte beachten Sie, dass bei jeder Änderung der Feature-Set-Einstellungen systembedingt das EtherCAT-Feldbus-System zusammen mit der Motion-Control-Konfiguration aus Ihrem Projekt gelöscht werden!

Ethernet-PG/OP-Kanal parametrieren

1. Klicken Sie im *Projektbaum* auf *"Geräte und Netze"*.
   ⇨ Sie erhalten eine grafische Objekt-Ansicht Ihrer CPU.



2. Klicken Sie auf das Netzwerk *"PG_OP_Ethernet"*.
3. Wählen Sie *"Kontextmenü ➔ Eigenschaften der Schnittstelle"*.
   ⇨ Es öffnet sich ein Dialogfenster. Hier können Sie IP-Adressdaten für Ihren Ethernet-PG/OP-Kanal angeben. Gültige IP-Adress-Parameter erhalten Sie von Ihrem Systemadministrator.
4. Bestätigen Sie Ihre Eingabe mit [OK].
   ⇨ Die IP-Adressdaten werden in Ihr Projekt übernommen und in *"Geräte und Netze"* unter *"Lokale Baugruppen"* aufgelistet.

   Nach der Übertragung Ihres Projekts ist Ihre CPU über die angegebenen IP-Adressdaten via Ethernet-PG/OP-Kanal erreichbar.

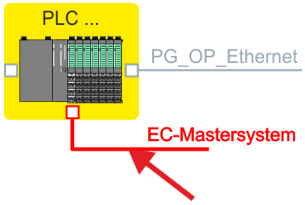
ESI-Datei installieren

Damit der *Sigma-7* EtherCAT Antrieb im *SPEED7 EtherCAT Manager* konfiguriert werden kann, muss die entsprechende ESI-Datei installiert sein. In der Regel wird das *SPEED7 Studio* mit aktuellen ESI-Dateien ausgeliefert und Sie können diesen Teil überspringen. Sollte Ihre ESI-Datei veraltet sein, finden Sie die aktuellste ESI-Datei für den *Sigma-7* EtherCAT Antrieb unter *www.yaskawa.eu.com* unter *"Service ➔ Drives & Motion Software"*.

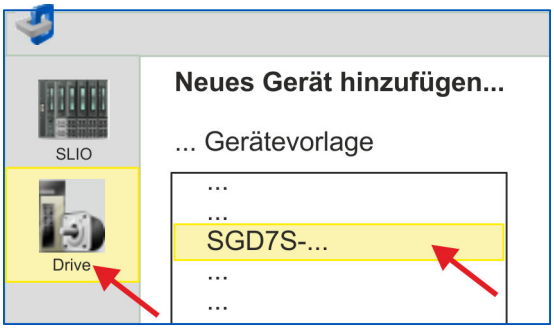
1. Laden Sie die zu Ihrem Antrieb passende ESI-Datei herunter. Entpacken Sie diese falls erforderlich.
2. Gehen Sie in Ihr *SPEED7 Studio*.
3. Öffnen Sie mit *"Extras ➔ Gerätebeschreibungsdatei installieren (EtherCAT - ESI)"* das zugehörige Dialogfenster.
4. Geben Sie unter *"Quellpfad"* die ESI-Datei an und installieren Sie diese mit [Installieren].
   ⇨ Die Geräte der ESI-Datei steht Ihnen nun zur Verfügung.

Sigma-7S Singleachs-Antrieb hinzufügen

1. Klicken Sie im Projektbaum auf *"Geräte und Netze"*.
2. Klicken Sie hier auf *"EC-Mastersystem"* und wählen sie *"Kontextmenü ➔ Neues Gerät hinzufügen"*.



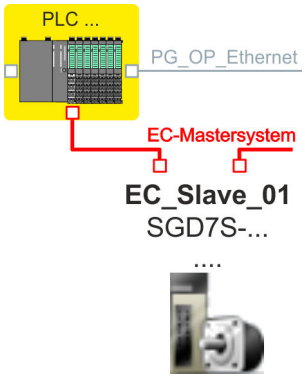
   ⇨ Es öffnet sich die Gerätevorlage zur Auswahl eines EtherCAT-Devices.



3. Wählen Sie Ihren *Sigma-7* Antrieb aus:

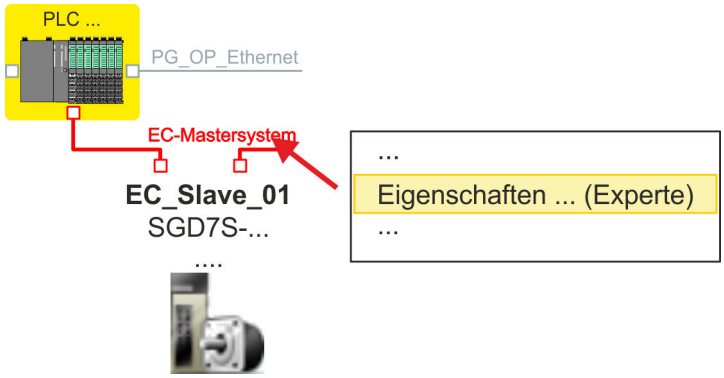
- SGD7S-xxxAA0...
- SGD7S-xxxDA0...
- SGD7S-xxxxA0...

Bestätigen Sie Ihre Angaben mit [OK]. Sollte Ihr Antrieb nicht vorhanden sein, müssen Sie die entsprechende ESI-Datei wie weiter oben beschrieben installieren.



⇨ Der *Sigma-7* Antrieb wird an Ihr EC-Mastersystem angebunden.

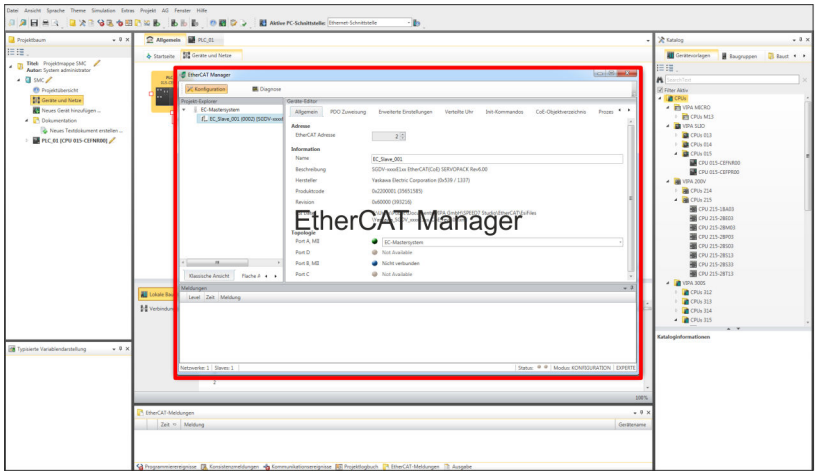
Sigma-7S Singleachs-Antrieb konfigurieren



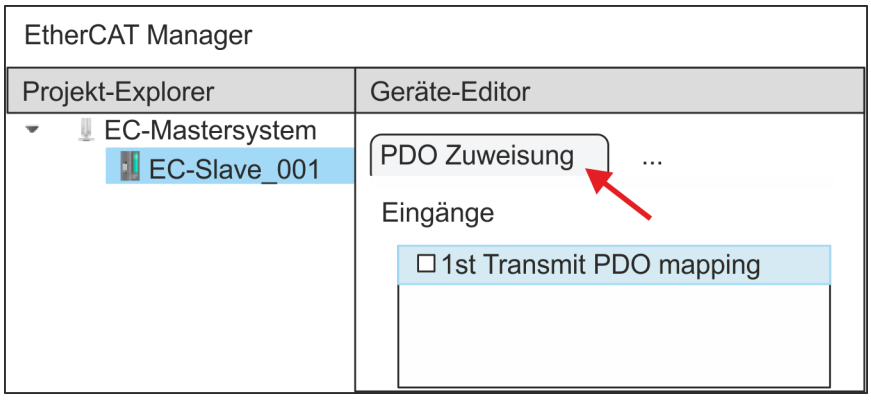
1. Klicken Sie auf *"EC-Mastersystem"* und wählen sie *"Kontextmenü ➔ Eigenschaft des Busystems (Experte)"*.

PDOs können Sie nur im "Experten-Modus" bearbeiten! Ansonsten werden die Schaltflächen ausgeblendet.

⇨ Der *SPEED7 EtherCAT Manager* wird gestartet. Hier können Sie die EtherCAT-Kommunikation zu Ihrem *Sigma-7* Antrieb konfigurieren.



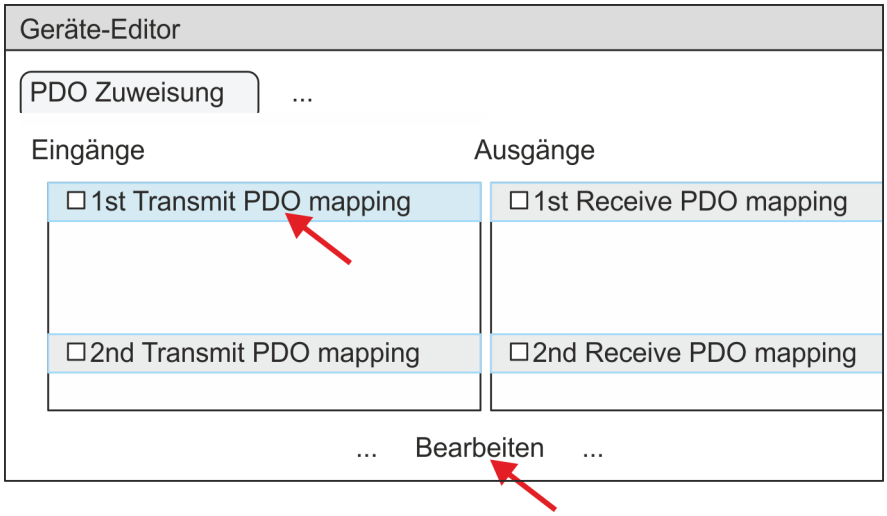
2. Klicken Sie im *SPEED7 EtherCAT Manager* auf den Slave und wählen Sie im *"Geräte-Editor"* den Reiter *"PDO-Zuweisung"* an.



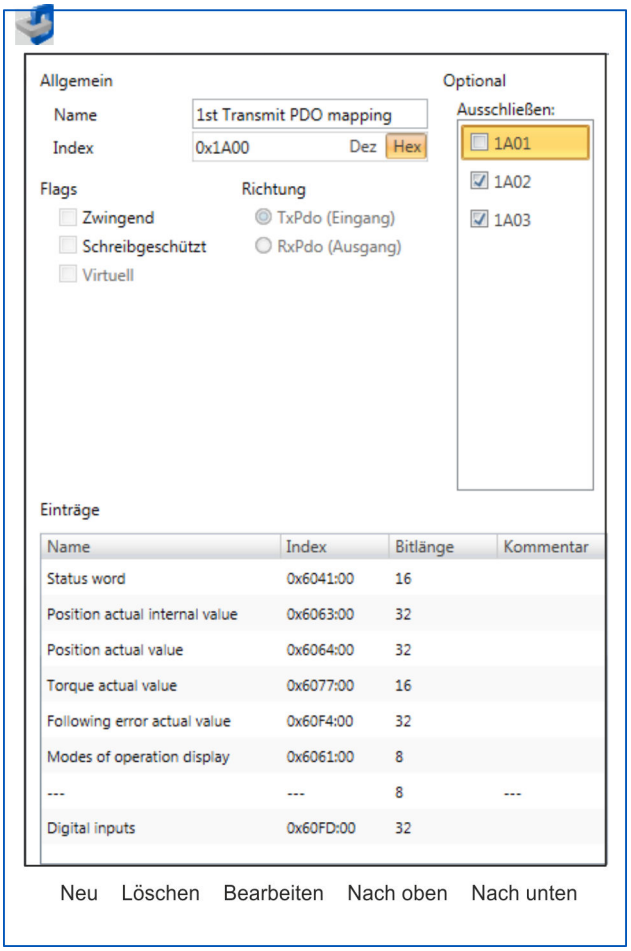
⇨ Dieser Dialog zeigt eine Auflistung aller PDOs.

3. Durch Anwahl des entsprechenden PDO-Mappings können Sie mit [Bearbeiten] die PDOs bearbeiten. Wählen Sie das Mapping *"1st Transmit PDO mapping"* an und klicken Sie auf [Bearbeiten].

Bitte beachten Sie, dass aufgrund der Voreinstellung manche PDOs nicht bearbeitet werden können. Durch Deaktivierung bereits aktivierter PDOs können Sie die Bearbeitung von gesperrten PDOs frei geben.



⇨ Es öffnet sich der Dialog *"PDO bearbeiten"*. Bitte überprüfen Sie hier die aufgeführten PDO-Einstellungen und passen Sie diese ggf. an. Bitte berücksichtigen Sie hierbei auch die Reihenfolge der *"Einträge"* und ergänzen Sie diese entsprechend.



Für die Bearbeitung der *"Einträge"* stehen folgende Funktionen zur Verfügung:

- Neu
  - Hiermit können Sie in einem Dialogfenster einen neuen Eintrag anlegen, indem Sie aus dem *"CoE-Objektverzeichnis"* den entsprechenden Eintrag auswählen und Ihre Einstellungen vornehmen. Mit [OK] wird der Eintrag übernommen und in der Liste der Einträge aufgeführt.
- Löschen
  - Hiermit können Sie den angewählte Eintrag löschen.
- Bearbeiten
  - Hiermit können Sie allgemeinen Daten eines Eintrags bearbeiten.
- Nach oben/unten
  - Hiermit können Sie den angewählten Eintrag in der Liste nach oben bzw. nach unten bewegen

4. Führen Sie folgende Einstellungen durch:

Eingänge: 1st Transmit PDO 0x1A00

- Allgemein
  - Name: 1st Transmit PDO mapping
  - Index: 0x1A00
- Flags
  - Alles deaktiviert
- Richtung
  - TxPdo (Eingang): aktiviert
- Ausschließen
  Bitte diese Einstellungen beachten, da ansonsten die PDO-Mappings nicht zeitgleich aktiviert werden können!
  - 1A01: deaktiviert
- Einträge

Name	Index	Bitlänge
Status word	0x6041:00	16Bit
Position actual internal value	0x6063:00	32Bit
Position actual value	0x6064:00	32Bit
Torque actual value	0x6077:00	16Bit
Following error actual value	0x60F4:00	32Bit
Modes of operation display	0x6061:00	8Bit
---	---	8Bit
Digital inputs	0x60FD:00	32Bit

Schließen Sie den Dialog *"PDO bearbeiten"* mit [OK].

5. Wählen Sie das Mapping *"2nd Transmit PDO mapping"* an und klicken Sie auf [Bearbeiten]. Führen Sie folgende Einstellungen durch:

Eingänge: 2nd Transmit PDO 0x1A01

- Allgemein
  - Name: 2nd Transmit PDO mapping
  - Index: 0x1A01
- Flags
  - Alles deaktiviert
- Richtung
  - TxPdo (Eingang): aktiviert
- Ausschließen

  Bitte diese Einstellungen beachten, da ansonsten die PDO-Mappings nicht zeitgleich aktiviert werden können!
  - 1A00: deaktiviert
  - 1A02: deaktiviert
  - 1A03: deaktiviert
- Einträge

Name	Index	Bitlänge
Touch probe status	0x60B9:00	16Bit
Touch probe 1 position value	0x60BA:00	32Bit
Touch probe 2 position value	0x60BC:00	32Bit
Velocity actual value	0x606C:00	32Bit

Schließen Sie den Dialog *"PDO bearbeiten"* mit [OK].

6. Wählen Sie das Mapping *"1st Receive PDO mapping"* an und klicken Sie auf [Bearbeiten]. Führen Sie folgende Einstellungen durch:

Ausgänge: 1st Receive PDO 0x1600

- Allgemein
  - Name: 1st Receive PDO mapping
  - Index: 0x1600
- Flags
  - Alles deaktiviert
- Richtung
  - RxPdo (Ausgang): aktiviert
- Ausschließen

  Bitte diese Einstellungen beachten, da ansonsten die PDO-Mappings nicht zeitgleich aktiviert werden können!
  - 1601: deaktiviert
  - 1602: deaktiviert
  - 1603: deaktiviert
- Einträge

Name	Index	Bitlänge
Control word	0x6040:00	16Bit
Target position	0x607A:00	32Bit
Target velocity	0x60FF:00	32Bit
Modes of operation	0x6060:00	8Bit
---	---	8Bit
Touch probe function	0x60B8:00	16Bit

Schließen Sie den Dialog *"PDO bearbeiten"* mit [OK].

7. Wählen Sie das Mapping *"2nd Receice PDO mapping"* an und klicken Sie auf [Bearbeiten]. Führen Sie folgende Einstellungen durch:

Ausgänge: 2nd Receive PDO 0x1601

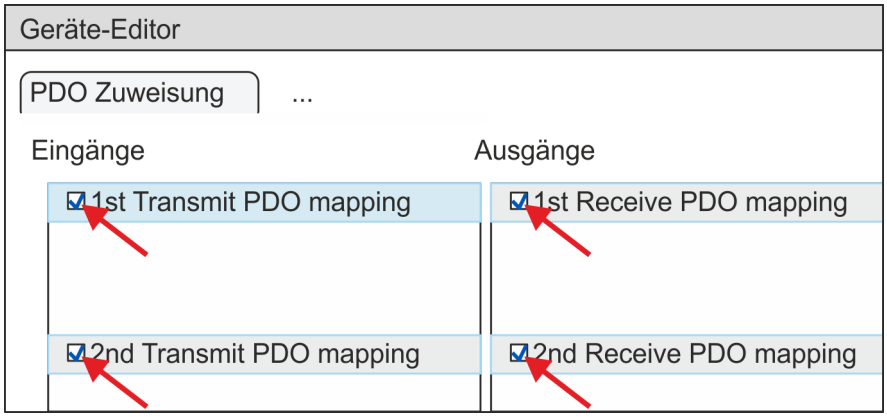
- Allgemein
  - Name: 2nd Receive PDO mapping
  - Index: 0x1601
- Flags
  - Alles deaktiviert
- Richtung
  - RxPdo (Ausgang): aktiviert
- Ausschließen

  Bitte diese Einstellungen beachten, da ansonsten die PDO-Mappings nicht zeitgleich aktiviert werden können!
  - 1600: deaktiviert
  - 1602: aktiviert
  - 1603: aktiviert
- Einträge

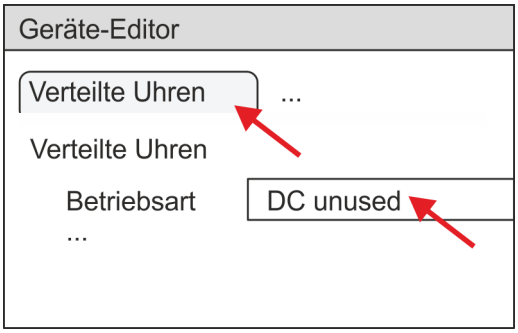
Name	Index	Bitlänge
Profile velocity	0x6081:00	32Bit
Profile acceleration	0x6083:00	32Bit
Profile deceleration	0x6084:00	32Bit

Schließen Sie den Dialog *"PDO bearbeiten"* mit [OK].

8. Aktivieren Sie in PDO-Zuweisung die PDOs 1 und 2 für die Ein und Ausgänge. Alle nachfolgenden PDOs müssen deaktiviert bleiben. Sollte dies nicht möglich sein, überprüfen Sie bitte den jeweiligen PDO-Parameter *"Ausschließen"*.

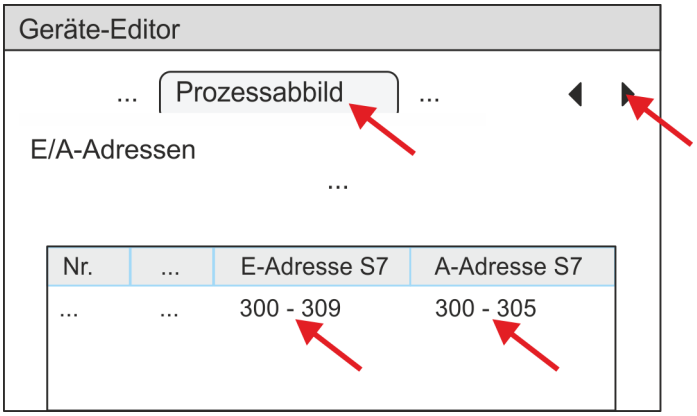


9. Wählen Sie im *"Geräte-Editor"* des *SPEED7 EtherCAT Manager* den Reiter *"Verteilte Uhren"* an und stellen Sie *"DC unused"* als *"Betriebsart"* ein.

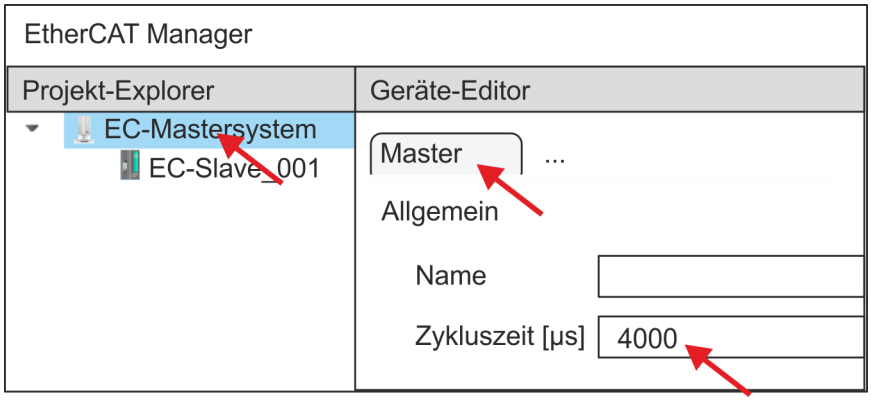


10. Wählen Sie im *"Geräte-Editor"* über die Pfeiltaste den Reiter *"Prozessabbild"* an und notieren Sie sich für die Parameter des Bausteins FB 873 - VMC_InitSigma7S_EC folgende PDO-Anfangsadressen:

- *"E-Adresse S7"* → *"InputsStartAddressPDO"*
- *"A-Adresse S7"* → *"OutputsStartAddressPDO"*



11. Klicken Sie im *SPEED7 EtherCAT Manager* auf *"EC-Mastersystem"* und wählen Sie im *"Geräte-Editor"* den Reiter *"Master"* an.

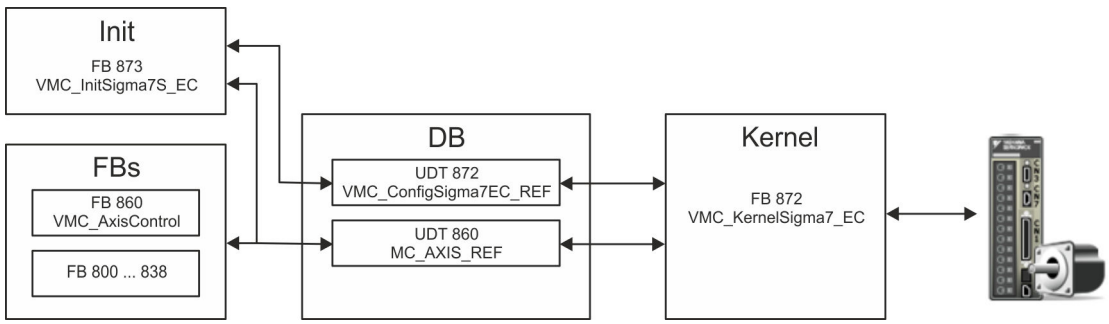


⇨ Stellen Sie für Sigma-7S (400V) Antriebe (SGD7S-xxxDA0... und SGD7S-xxxxA0...) eine Zykluszeit von mindestens 4ms ein. Ansonsten lassen Sie den Wert bei 1ms.

12. Indem Sie den Dialog des *SPEED7 EtherCAT Manager* mit [X] schließen, wird die Konfiguration in das *SPEED7 Studio* übernommen.

15.2.2.3.2 Anwender-Programm

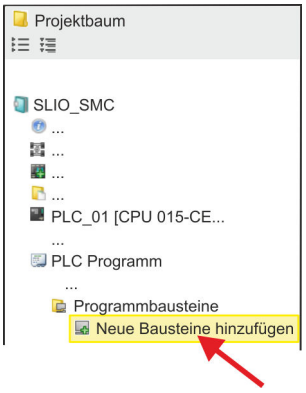
Programmstruktur



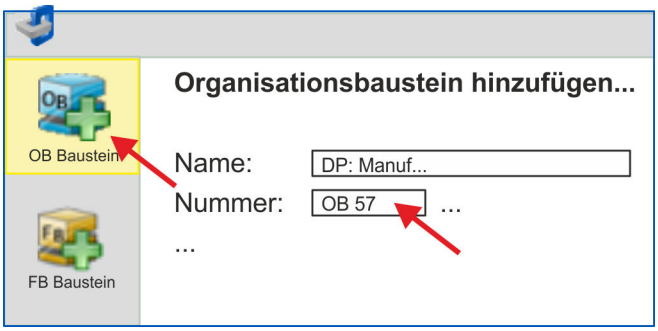
- DB
  Für jede Achse ist ein Datenbaustein (Achs-DB) für Konfiguration und Statusdaten anzulegen. Der Datenbaustein besteht aus folgenden Datenstrukturen:
  – UDT 872 - *VMC_Config*Sigma7EC_REF
    Die Datenstruktur beschreibt den Aufbau der Konfiguration des Antriebs.
    Spezifische Datenstruktur für *Sigma-7* EtherCAT.
  – UDT 860 - *MC_AXIS_REF*
    Die Datenstruktur beschreibt den Aufbau der Parameter und Statusinformationen von Antrieben.
    Allgemeine Datenstruktur für alle Antriebe und Bussysteme.
- FB 873 - *VMC_Init*Sigma7S_EC
  – Der *Init*-Baustein dient zur Konfiguration einer Achse.
  – Spezifischer Baustein für *Sigma-7S* EtherCAT.
  – Die Konfigurationsdaten für die Initialisierung sind im *Achs-DB* abzulegen.
- FB 872 - *VMC_Kernel*Sigma7_EC
  – Der *Kernel*-Baustein kommuniziert mit dem Antrieb über das entsprechende Bussystem, verarbeitet die Benutzeraufträge und liefert Statusmeldungen zurück.
  – Spezifischer Baustein für *Sigma-7* EtherCAT.
  – Der Austausch der Daten erfolgt mittels des *Achs-DB*.
- FB 860 - *VMC_AxisControl*
  – Universal-Baustein für alle Antriebe und Bussysteme.
  – Unterstützt einfache Bewegungskommandos und liefert alle relevanten Statusmeldungen.
  – Der Austausch der Daten erfolgt mittels des *Achs-DB*.
  – Über die Instanzdaten des Bausteins können Sie zur Bewegungssteuerung und Statusabfrage eine Visualisierung anbinden.
  – Zusätzlich zum FB 860 - *VMC_AxisControl* haben Sie die Möglichkeit *PLCopen*-Bausteine zu nutzen.
- FB 800 ... FB 838 - *PLCopen*
  – Die PLCopen-Bausteine dienen zur Programmierung von Bewegungsabläufen und Statusabfragen.
  – Allgemeine Bausteine für alle Antriebe und Bussysteme.

Programmierung

Bausteine in Projekt kopieren

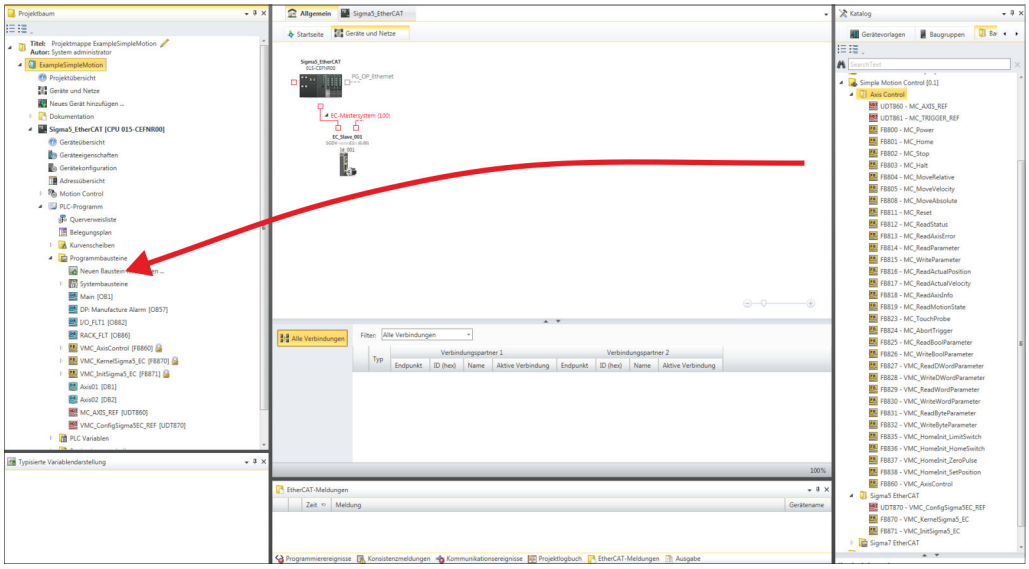


1. Klicken Sie im *Projektbaum* innerhalb der CPU unter *"PLC-Programm"*, *"Programmbausteine"* auf *"Neuen Baustein hinzufügen"*.



⇨ Das Dialogfenster *"Baustein hinzufügen"* öffnet sich.

2. Wählen Sie den Bausteintyp *"OB Baustein"* und fügen Sie nacheinander OB 57, OB 82 und OB 86 Ihrem Projekt hinzu.



3. Öffnen Sie im *"Katalog"* unter *"Bausteine"* *"Simple Motion Control"* und ziehen Sie per Drag&Drop folgende Bausteine in *"Programmbausteine"* des *Projektbaums*:
   - *Sigma-7* EtherCAT:
     - UDT 872 - VMC_ConfigSigma7EC_REF
     - FB 872 - VMC_KernelSigma7_EC
     - FB 873 - VMC_InitSigma7S_EC
   - Axis Control
     - UDT 860 - MC_AXIS_REF
     - Bausteine für die gewünschten Bewegungsabläufe

Achs-DB anlegen

1. Fügen Sie Ihrem Projekt einen neuen DB als *Achs-DB* hinzu. Klicken Sie hierzu im *Projektbaum* innerhalb der CPU unter *"PLC-Programm"*, *"Programmbausteine"* auf *"Neuen Baustein hinzufügen"*, wählen Sie den Bausteintyp *"DB Baustein"* und vergeben Sie diesem den Namen "Axis01". Die DB-Nr. können Sie frei wählen wie z.B. DB10.

   ⇨ Der Baustein wird angelegt und geöffnet.

2. - Legen Sie in "Axis01" die Variable "Config" vom Typ UDT 872 an. Dies sind spezifische Achs-Konfigurationsdaten.
   - Legen Sie in "Axis01" die Variable "Axis" vom Typ UDT 860 an. Während des Betriebs werden hier alle Betriebsdaten der Achse abgelegt.

Axis01 [DB10]

Bausteinstruktur

		Adr...	Name	Datentyp	...
		...	Config	UDT	[872]
		...	Axis	UDT	[860]

OB 1

Konfiguration der Achse

Öffnen Sie den OB 1 und programmieren Sie folgende FB-Aufrufe mit zugehörigen DBs:

FB 873 - VMC_InitSigma7S_EC, DB 873 *Kap. 15.2.2.4.3 "FB 873 - VMC_InitSigma7S_EC - Sigma-7S EtherCAT Initialisierung" Seite 617*

Geben Sie unter *InputsStartAddressPDO* bzw. *OutputsStartAddressPDO* die Adresse aus dem *SPEED7 EtherCAT Manager* an. *611*

```
⇨ CALL  "VMC_InitSigma7S_EC" , "DI_InitSgm7SETC01"
  Enable                :="InitS7SEC1_Enable"
  LogicalAddress        :=300
  InputsStartAddressPDO :=300 (EtherCAT-Man.: E-Adresse S7)
  OutputsStartAddressPDO:=300 (EtherCAT-Man.: A-Adresse S7)
  EncoderType           :=1
  EncoderResolutionBits :=20
  FactorPosition        :=1.048576e+006
  FactorVelocity        :=1.048576e+006
  FactorAcceleration    :=1.048576e+002
  OffsetPosition        :=0.000000e+000
  MaxVelocityApp        :=5.000000e+001
  MaxAccelerationApp    :=1.000000e+002
  MaxDecelerationApp    :=1.000000e+002
  MaxVelocityDrive      :=6.000000e+001
  MaxAccelerationDrive  :=1.500000e+002
  MaxDecelerationDrive  :=1.500000e+002
  MaxPosition           :=1.048500e+003
  MinPosition           :=-1.048514e+003
  EnableMaxPosition     :=TRUE
  EnableMinPosition     :=TRUE
  MinUserPosition       :="InitS7SEC1_MinUserPos"
  MaxUserPosition       :="InitS7SEC1_MaxUserPos"
  Valid                 :="InitS7SEC1_Valid"
  Error                 :="InitS7SEC1_Error"
  ErrorID               :="InitS7SEC1_ErrorID"
  Config                :="Axis01".Config
  Axis                  :="Axis01".Axis
```

Kernel für Achse beschalten

Der *Kernel* verarbeitet die Benutzerkommandos und gibt sie entsprechend aufbereitet an den Antrieb über das jeweilige Bussystem weiter.

FB 872 - VMC_KernelSigma7_EC, DB 872 *Kap. 15.2.2.4.2 "FB 872 - VMC_KernelSigma7_EC - Sigma-7 EtherCAT Kernel" Seite 617*

```
⇨       CALL  "VMC_KernelSigma7_EC" , "DI_KernelSgm5ETC01"
         Init  :="KernelS7SEC1_Init"
         Config:="Axis01".Config
         Axis  :="Axis01".Axis
```

Baustein für Bewegungsabläufe beschalten

Zur Vereinfachung soll hier die Beschaltung des FB 860 - VMC_AxisControl gezeigt werden. Dieser Universalbaustein unterstützt einfache Bewegungskommandos und liefert Statusmeldungen zurück. Die Ein- und Ausgänge können Sie individuell beschalten. Bitte geben Sie unter *"Axis"* die Referenz zu den entsprechenden Achsdaten im *Achs-DB* an.

FB 860 - VMC_AxisControl, DB 860 ↬ *Kap. 15.8.2.2 "FB 860 - VMC_AxisControl - Control-Baustein Achskontrolle" Seite 754*

```
⇨       CALL  "VMC_AxisControl" , "DI_AxisControl01"
         AxisEnable       :="AxCtrl1_AxisEnable"
         AxisReset        :="AxCtrl1_AxisReset"
         HomeExecute      :="AxCtrl1_HomeExecute"
         HomePosition     :="AxCtrl1_HomePosition"
         StopExecute      :="AxCtrl1_StopExecute"
         MvVelocityExecute:="AxCtrl1_MvVelExecute"
         MvRelativeExecute:="AxCtrl1_MvRelExecute"
         MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
         PositionDistance :="AxCtrl1_PositionDistance"
         Velocity         :="AxCtrl1_Velocity"
         Acceleration     :="AxCtrl1_Acceleration"
         Deceleration     :="AxCtrl1_Deceleration"
         JogPositive      :="AxCtrl1_JogPositive"
         JogNegative      :="AxCtrl1_JogNegative"
         JogVelocity      :="AxCtrl1_JogVelocity"
         JogAcceleration  :="AxCtrl1_JogAcceleration"
         JogDeceleration  :="AxCtrl1_JogDeceleration"
         AxisReady        :="AxCtrl1_AxisReady"
         AxisEnabled      :="AxCtrl1_AxisEnabled"
         AxisError        :="AxCtrl1_AxisError"
         AxisErrorID      :="AxCtrl1_AxisErrorID"
         DriveWarning     :="AxCtrl1_DriveWarning"
         DriveError       :="AxCtrl1_DriveError"
         DriveErrorID     :="AxCtrl1_DriveErrorID"
         IsHomed          :="AxCtrl1_IsHomed"
         ModeOfOperation  :="AxCtrl1_ModeOfOperation"
         PLCopenState     :="AxCtrl1_PLCopenState"
         ActualPosition   :="AxCtrl1_ActualPosition"
         ActualVelocity   :="AxCtrl1_ActualVelocity"
         CmdDone          :="AxCtrl1_CmdDone"
         CmdBusy          :="AxCtrl1_CmdBusy"
         CmdAborted       :="AxCtrl1_CmdAborted"
         CmdError         :="AxCtrl1_CmdError"
         CmdErrorID       :="AxCtrl1_CmdErrorID"
         DirectionPositive:="AxCtrl1_DirectionPos"
         DirectionNegative:="AxCtrl1_DirectionNeg"
         SWLimitMinActive :="AxCtrl1_SWLimitMinActive"
         SWLimitMaxActive :="AxCtrl1_SWLimitMaxActive"
         HWLimitMinActive :="AxCtrl1_HWLimitMinActive"
         HWLimitMaxActive :="AxCtrl1_HWLimitMaxActive"
         Axis             :="Axis01".Axis
```

Für komplexe Bewegungsaufgaben können Sie die PLCopen-Bausteine verwenden. Hier müssen Sie ebenfalls unter Axis die Referenz zu den Achsdaten im Achs-DB angeben.

Ihr Projekt beinhaltet nun folgende Bausteine:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 860 - VMC_AxisControl mit Instanz-DB

- FB 872 - VMC_KernelSigma7_EC mit Instanz-DB
- FB 873 - VMC_InitSigma7S_EC mit Instanz-DB
- UDT 860 - MC_Axis_REF
- UDT 872 - VMC_ConfigSigma7EC_REF

Zeitlicher Ablauf

1. Wählen Sie *"Projekt ➔ Alles übersetzen"* und übertragen Sie das Projekt in Ihre CPU.

   ⇨ Sie können jetzt Ihre Applikation in Betrieb nehmen.

VORSICHT!
Bitte beachten Sie immer die Sicherheitshinweise zu ihrem Antrieb, insbesondere bei der Inbetriebnahme!

2. Bevor eine Achse gesteuert werden kann, muss diese initialisiert werden. Rufen Sie hierzu den *Init*-Baustein FB 873 - VMC_InitSigma7S_EC mit *Enable* = TRUE auf.

   ⇨ Der Ausgang *Valid* meldet TRUE zurück. Im Fehlerfall können Sie durch Auswertung der *ErrorID* den Fehler ermitteln.

   Den *Init*-Baustein müssen Sie erneut aufrufen, wenn Sie einen neuen Achs-DB laden oder Parameter am *Init*-Baustein geändert wurden.

Fahren Sie erst fort, wenn der Init-Baustein keinen Fehler meldet!

3. Stellen Sie sicher, dass der *Kernel*-Baustein FB 872 - VMC_KernelSigma7_EC zyklisch aufgerufen wird. Auf diese Weise werden Steuersignale an den Antrieb übergeben und Statusmeldungen übermittelt.
4. Programmieren Sie Ihre Applikation mit dem FB 860 - VMC_AxisControl oder mit den PLCopen Bausteinen.

Steuerung des Antriebs über HMI

Sie haben die Möglichkeit über ein HMI Ihren Antrieb zu steuern. Hierzu gibt es für Movicon eine vorgefertigte Symbolbibliothek für den Zugriff auf den VMC_AxisControl Funktionsbaustein. ↬ *Kap. 15.9 "Antrieb über HMI steuern" Seite 825*

15.2.2.4 Antriebsspezifische Bausteine

Die PLCopen-Bausteine zur Achskontrolle finden Sie hier: Kap. 15.8 "Bausteine zur Achskontrolle" Seite 751

15.2.2.4.1 UDT 872 - VMC_ConfigSigma7EC_REF - *Sigma-7* EtherCAT Datenstruktur Achskonfiguration

Dies ist eine benutzerdefinierte Datenstruktur, die Informationen zu den Konfigurationsdaten beinhaltet. Die UDT ist speziell angepasst an die Verwendung eines *Sigma-7*-Antriebs, welcher über EtherCAT angebunden ist.

15.2.2.4.2 FB 872 - VMC_KernelSigma7_EC - *Sigma-7* EtherCAT Kernel

Beschreibung

Dieser Baustein setzt die Antriebskommandos für eine *Sigma-7* Achse über EtherCAT um und kommuniziert mit dem Antrieb. Je *Sigma-7* Achse ist eine Instanz dieses FBs zyklisch aufzurufen.

Bitte beachten Sie, dass dieser Baustein intern den SFB 238 aufruft.

Im SPEED7 Studio wird dieser Baustein automatisch in Ihr Projekt eingefügt.

Parameter	Deklaration	Datentyp	Beschreibung
Init	INPUT	BOOL	Mit einer Flanke 0-1 wird der Baustein intern zurückgesetzt. Hierbei werden bestehende Bewegungskommandos abgebrochen und der Baustein wird initialisiert.
Config	IN_OUT	UDT872	Datenstruktur zur Übergabe von achsabhängigen Konfigurationsdaten an den *AxisKernel*.
Axis	IN_OUT	MC_AXIS_REF	Datenstruktur zur Übergabe von achsabhängigen Informationen an *AxisKernel* und PLCopen-Bausteine.

15.2.2.4.3 FB 873 - VMC_InitSigma7S_EC - *Sigma-7S* EtherCAT Initialisierung

Beschreibung

Dieser Baustein dient zur Konfiguration der Achse. Der Baustein ist speziell angepasst an die Verwendung eines *Sigma-7*-Antriebs, welcher über EtherCAT angebunden ist.

Parameter	Deklaration	Datentyp	Beschreibung
Enable	INPUT	BOOL	Freigabe der Initialisierung
LogicalAddress	INPUT	INT	Startadresse der PDO-Eingangsdaten
InputsStartAddressPDO	INPUT	INT	Startadresse der Eingabe-PDOs
OutputsStartAddressPDO	INPUT	INT	Startadresse der Ausgabe-PDOs
EncoderType	INPUT	INT	Encoder-Typ ■ 1: Absolut-Encoder ■ 2: Inkremental-Encoder

Parameter	Deklaration	Datentyp	Beschreibung
EncoderResolutionBits	INPUT	INT	Anzahl der Bits, die einer Geber-Umdrehung entsprechen. Default: 20
FactorPosition	INPUT	REAL	Faktor zur Umrechnung der Position von Benutzereinheiten [u] in Antriebseinheiten [Inkremente] und zurück. Es gilt: $p_{[Inkremente]} = p_{[u]}$ x *FactorPosition* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2701:1 und 0x2701:2 vorgeben können. Dieser sollte 1 sein.
FactorVelocity	INPUT	REAL	Faktor zur Umrechnung der Geschwindigkeit von Benutzereinheiten [u/s] in Antriebseinheiten [Inkremente/s] und zurück. Es gilt: $v_{[Inkremente/s]} = v_{[u/s]}$ x *FactorVelocity* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2702:1 und 0x2702:2 vorgeben können. Dieser sollte 1 sein.
FactorAcceleration	INPUT	REAL	Faktor zur Umrechnung der Beschleunigung von Benutzereinheiten [u/s^2] in Antriebseinheiten [10^{-4} x Inkremente/s^2] und zurück. Es gilt: 10^{-4} x $a_{[Inkremente/s^2]} = a_{[u/s^2]}$ x *FactorAcceleration* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2703:1 und 0x2703:2 vorgeben können. Dieser sollte 1 sein.
OffsetPosition	INPUT	REAL	Offset für die Nullposition [u].
MaxVelocityApp	INPUT	REAL	Maximale Geschwindigkeit der Applikation [u/s]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.
MaxAccelerationApp	INPUT	REAL	Maximale Beschleunigung der Applikation [u/s^2]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.
MaxDecelerationApp	INPUT	REAL	Maximale Verzögerung der Applikation [u/s^2]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.
MaxPosition	INPUT	REAL	Maximale Position für die Überwachung der Softwarelimits [u].
MinPosition	INPUT	REAL	Minimale Position für die Überwachung der Softwarelimits [u].
EnableMaxPosition	INPUT	BOOL	Überwachung maximale Position ■ TRUE: Aktivierung der Überwachung der maximalen Position.
EnableMinPosition	INPUT	BOOL	Überwachung minimale Position ■ TRUE: Aktivierung der Überwachung der minimalen Position.
MinUserPosition	OUTPUT	REAL	Minimale Benutzerposition basierend auf dem minimalen Encoder Wert von 0x80000000 und dem *FactorPosition* [u].

Parameter	Deklaration	Datentyp	Beschreibung
MaxUserPosition	OUTPUT	REAL	Maximale Benutzerposition basierend auf dem maximalen Encoder Wert von 0x7FFFFFFF und dem *Factor-Position* [u].
Valid	OUTPUT	BOOL	Initialisierung ■ TRUE: Initialisierung ist gültig.
Error	OUTPUT	BOOL	■ Fehler – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden. Die Achse wird gesperrt.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Config	IN_OUT	UDT872	Datenstruktur zur Übergabe von achsabhängigen Konfigurationsdaten an den *AxisKernel*.
Axis	IN_OUT	MC_AXIS_REF	Datenstruktur zur Übergabe von achsabhängigen Informationen an *AxisKernel* und PLCopen-Bausteine.

15.2.3 Einsatz *Sigma-7W* EtherCAT

15.2.3.1 Übersicht

Einsatz des Singelachs-Antriebs *Kap. 15.2.2 "Einsatz Sigma-7S EtherCAT" Seite 598*

Voraussetzung

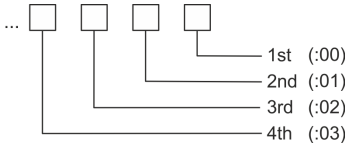
- SPEED7 Studio ab V1.6.1
- CPU mit EtherCAT-Master wie z.B. CPU 015-CEFNR00
- *Sigma-7W* Doppelachs-Antrieb mit EtherCAT-Optionskarte

Schritte der Projektierung

1. Parameter am Antrieb einstellen
   - Die Einstellung der Parameter hat mit dem Softwaretool *Sigma Win+* zu erfolgen.
2. Hardwarekonfiguration im VIPA *SPEED7 Studio*
   - Projektierung einer CPU mit EtherCAT-Master-Funktionalität.
   - Projektierung der *Sigma-7W* EtherCAT Doppelachsen.
   - Projektierung der EtherCAT-Anbindung über *SPEED7 EtherCAT Manager*.
3. Programmierung im VIPA *SPEED7 Studio*
   - *Init*-Baustein zur Konfiguration der Doppel-Achsen beschalten.
   - *Kernel*-Baustein zur Kommunikation mit je einer Achse beschalten.
   - Bausteine für die Bewegungsabläufe beschalten.

15.2.3.2 Parameter am Antrieb einstellen

Parameter-Digits



VORSICHT!
Vor der Erstinbetriebnahme müssen Sie Ihren Antrieb mit dem Software-tool *Sigma Win+* an Ihre Applikation anpassen! Näheres hierzu finden Sie im Handbuch zu ihrem Antrieb.

Zur Abstimmung auf die *Simple Motion Control Library* sind folgende Parameter über *Sigma Win+* einzustellen:

Achse 1 - Module 1 (24Bit Encoder)

Servopack Parameter	Adresse:Digit	Name	Wert
Pn205	(2205h)	Multiturn Limit Setting	65535
Pn20E	(220Eh)	Electronic Gear Ratio (Numerator)	16
Pn210	(2210h)	Electronic Gear Ratio (Denominator)	1
PnB02	(2701h:01)	Position User Unit (Numerator)	1
PnB04	(2701h:02)	Position User Unit (Denominator)	1
PnB06	(2702h:01)	Velocity User Unit (Numerator)	1
PnB08	(2702h:02)	Velocity User Unit (Denominator)	1
PnB0A	(2703h:01)	Acceleration User Unit (Numerator)	1
PnB0C	(2703h:02)	Acceleration User Unit (Denominator)	1

Achse 2 - Module 2 (24Bit Encoder)

Servopack Parameter	Adresse:Digit	Name	Wert
Pn205	(2A05h)	Multiturn Limit Setting	65535
Pn20E	(2A0Eh)	Electronic Gear Ratio (Numerator)	16
Pn210	(2A10h)	Electronic Gear Ratio (Denominator)	1
PnB02	(2F01h:01)	Position User Unit (Numerator)	1
PnB04	(2F01h:02)	Position User Unit (Denominator)	1
PnB06	(2F02h:01)	Velocity User Unit (Numerator)	1
PnB08	(2F02h:02)	Velocity User Unit (Denominator)	1
PnB0A	(2F03h:01)	Acceleration User Unit (Numerator)	1
PnB0C	(2F03h:02)	Acceleration User Unit (Denominator)	1

Bitte beachten Sie, dass Sie gemäß ihren Anforderungen die entsprechende Fahrtrichtung für Ihren Antrieb freigeben. Verwenden Sie hierzu die Parameter Pn50A (P-OT) bzw. Pn50B (N-OT) in Sigma Win+.

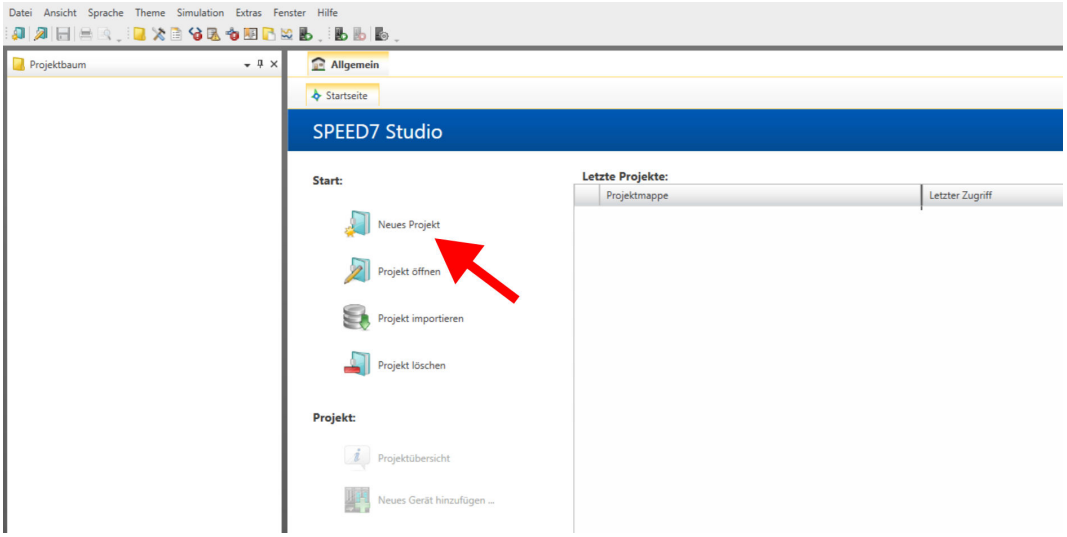
15.2.3.3 Einsatz im VIPA *SPEED7 Studio*

15.2.3.3.1 Hardware-Konfiguration

CPU im Projekt anlegen

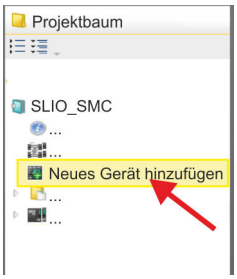
Bitte verwenden Sie für die Projektierung das *SPEED7 Studio* ab V1.6.1.

1. Starten Sie das *SPEED7 Studio*.

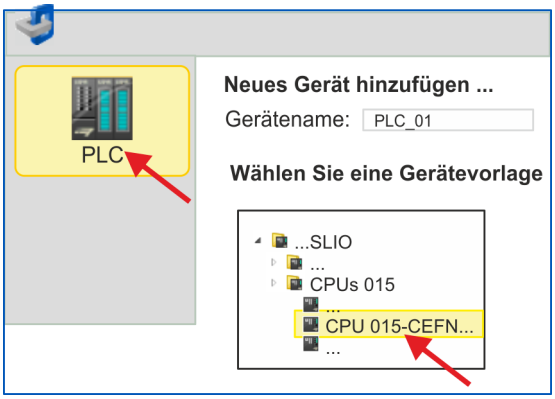


2. Erstellen sie auf der Startseite mit *"Neues Projekt"* ein neues Projekt und vergeben Sie einen *"Projektnamen"*.

⇨ Ein neues Projekt wird angelegt und in die Sicht *"Geräte und Netze"* gewechselt.



3. Klicken Sie im *Projektbaum* auf *"Neues Gerät hinzufügen ..."*.



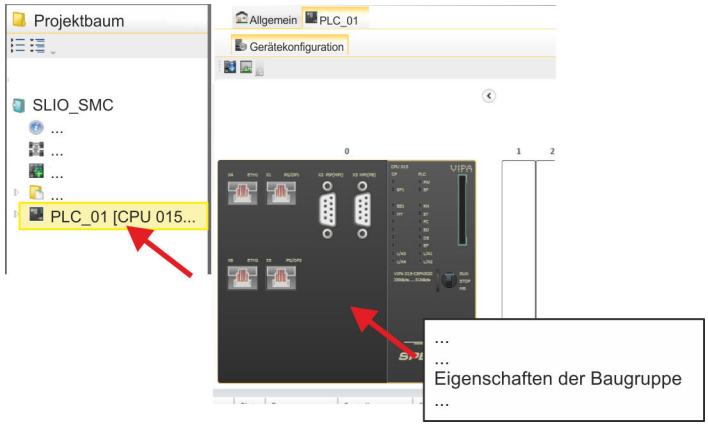
⇨ Es öffnet sich ein Dialog für die Geräteauswahl.

4. Wählen Sie unter den *"Gerätevorlagen"* eine CPU mit EtherCAT-Master-Funktionalität wie z.B. die CPU 015-CEFNR00 und klicken Sie auf [OK].

⇨ Die CPU wird in *"Geräte und Netze"* eingefügt und die *"Gerätekonfiguration"* geöffnet.

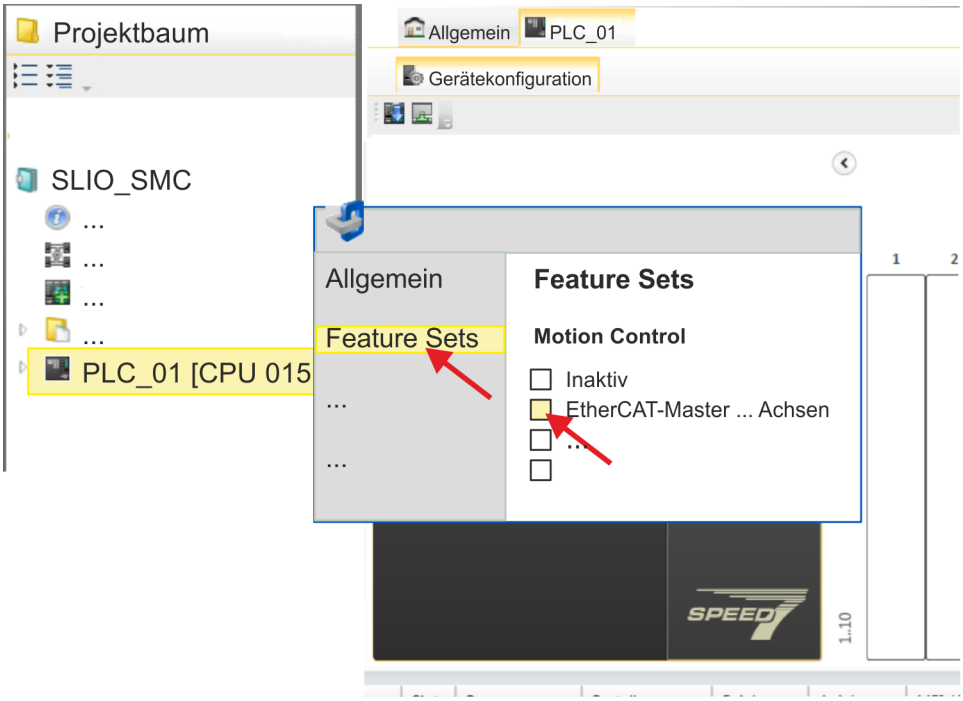
Motion-Control-Funktionen aktivieren

Sofern bei Ihrer CPU die EtherCAT-Master-Funktionalität noch nicht aktiviert ist, erfolgt die Aktivierung nach folgenden Vorgehensweise:



1. Klicken Sie in der *"Gerätekonfiguration"* auf die CPU und wählen Sie *"Kontextmenü ➔ Eigenschaften der Baugruppe"*.

   ⇨ Es öffnet sich der Eigenschaften-Dialog der CPU.



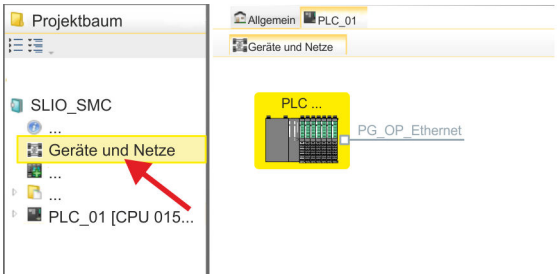
2. Klicken Sie auf *"Feature Sets"* und aktivieren Sie unter *"Motion Control"* einen der Parameter *"EtherCAT-Master ... Achsen"*. Die Anzahl der Achsen ist in diesem Beispiel nicht relevant.
3. Bestätigen Sie Ihre Angaben mit [OK].

   ⇨ Die Motion-Control-Funktionen steht Ihnen nun in Ihrem Projekt zur Verfügung.

VORSICHT!
Bitte beachten Sie, dass bei jeder Änderung der Feature-Set-Einstellungen systembedingt das EtherCAT-Feldbus-System zusammen mit der Motion-Control-Konfiguration aus Ihrem Projekt gelöscht werden!

Ethernet-PG/OP-Kanal parametrieren

1. Klicken Sie im *Projektbaum* auf *"Geräte und Netze"*.
   ⇨ Sie erhalten eine grafische Objekt-Ansicht Ihrer CPU.



2. Klicken Sie auf das Netzwerk *"PG_OP_Ethernet"*.
3. Wählen Sie *"Kontextmenü ➔ Eigenschaften der Schnittstelle"*.
   ⇨ Es öffnet sich ein Dialogfenster. Hier können Sie IP-Adressdaten für Ihren Ethernet-PG/OP-Kanal angeben. Gültige IP-Adress-Parameter erhalten Sie von Ihrem Systemadministrator.
4. Bestätigen Sie Ihre Eingabe mit [OK].
   ⇨ Die IP-Adressdaten werden in Ihr Projekt übernommen und in *"Geräte und Netze"* unter *"Lokale Baugruppen"* aufgelistet.

   Nach der Übertragung Ihres Projekts ist Ihre CPU über die angegebenen IP-Adressdaten via Ethernet-PG/OP-Kanal erreichbar.

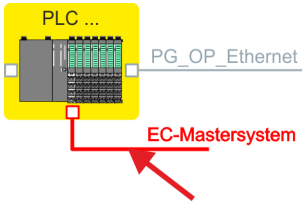
ESI-Datei installieren

Damit der *Sigma-7* EtherCAT Antrieb im *SPEED7 EtherCAT Manager* konfiguriert werden kann, muss die entsprechende ESI-Datei installiert sein. In der Regel wird das *SPEED7 Studio* mit aktuellen ESI-Dateien ausgeliefert und Sie können diesen Teil überspringen. Sollte Ihre ESI-Datei veraltet sein, finden Sie die aktuellste ESI-Datei für den *Sigma-7* EtherCAT Antrieb unter *www.yaskawa.eu.com* unter *"Service ➔ Drives & Motion Software"*.

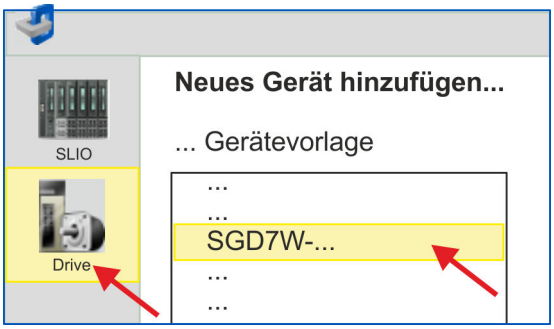
1. Laden Sie die zu Ihrem Antrieb passende ESI-Datei herunter. Entpacken Sie diese falls erforderlich.
2. Gehen Sie in Ihr *SPEED7 Studio*.
3. Öffnen Sie mit *"Extras ➔ Gerätebeschreibungsdatei installieren (EtherCAT - ESI)"* das zugehörige Dialogfenster.
4. Geben Sie unter *"Quellpfad"* die ESI-Datei an und installieren Sie diese mit [Installieren].
   ⇨ Die Geräte der ESI-Datei steht Ihnen nun zur Verfügung.

***Sigma-7W* Doppelachs-Antrieb hinzufügen**

1. Klicken Sie im Projektbaum auf *"Geräte und Netze"*.
2. Klicken Sie hier auf *"EC-Mastersystem"* und wählen sie *"Kontextmenü ➔ Neues Gerät hinzufügen"*.



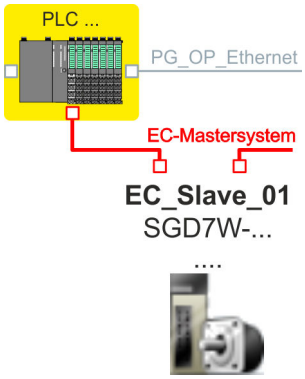
   ⇨ Es öffnet sich die Gerätevorlage zur Auswahl eines EtherCAT-Devices.



3. Wählen Sie Ihren *Sigma-7W* Doppelachs-Antrieb aus:

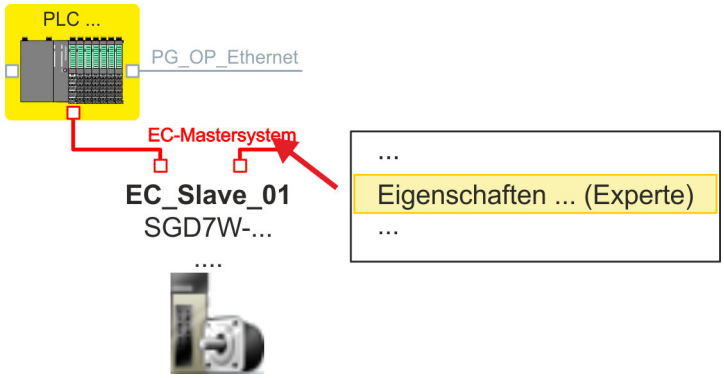
- SGD7W-xxxxA0...

Bestätigen Sie Ihre Angaben mit [OK]. Sollte Ihr Antrieb nicht vorhanden sein, müssen Sie die entsprechende ESI-Datei wie weiter oben beschrieben installieren.



⇨ Der *Sigma-7W* Doppelachs-Antrieb wird an Ihr EC-Mastersystem angebunden.

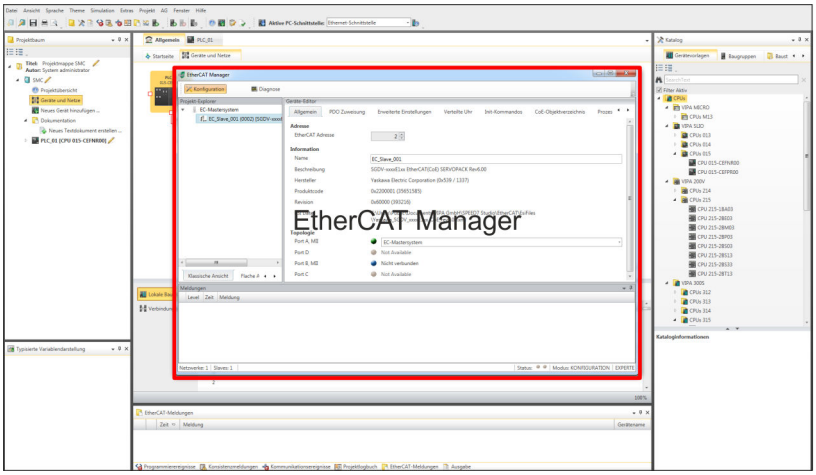
Sigma-7W Doppelachs-Antrieb konfigurieren



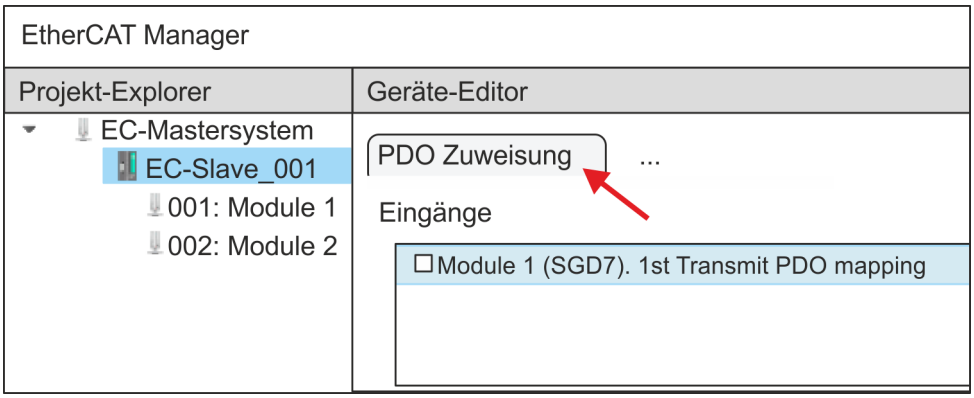
1. Klicken Sie auf *"EC-Mastersystem"* und wählen sie *"Kontextmenü ➔ Eigenschaft des Busystems (Experte)"*.

> *PDOs können Sie nur im "Experten-Modus" bearbeiten! Ansonsten werden die Schaltflächen ausgeblendet.*

⇨ Der *SPEED7 EtherCAT Manager* wird gestartet. Hier können Sie die EtherCAT-Kommunikation zu Ihrem *Sigma-7W* Doppelachs-Antrieb konfigurieren.



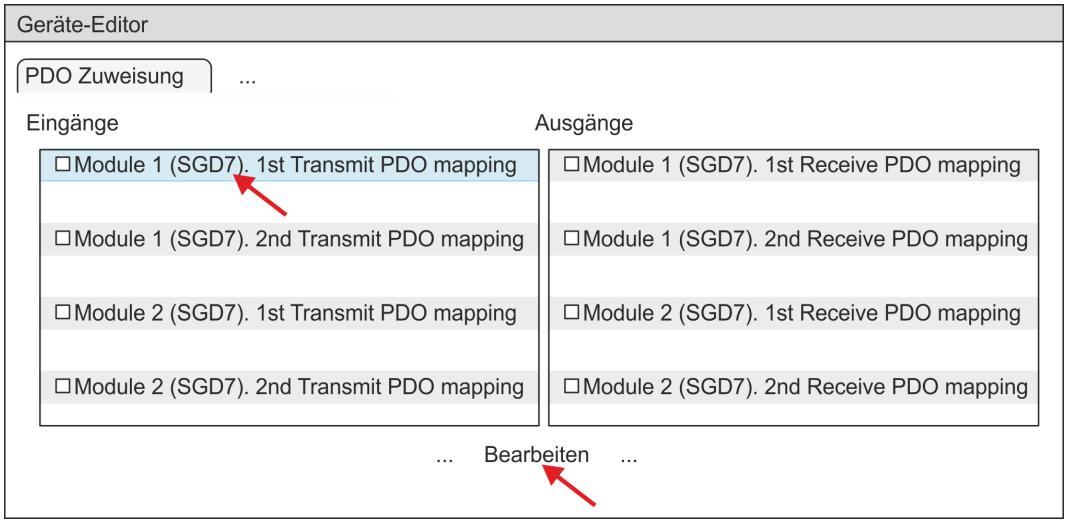
2. Klicken Sie im *SPEED7 EtherCAT Manager* auf den Slave und wählen Sie im *"Geräte-Editor"* den Reiter *"PDO-Zuweisung"* an.



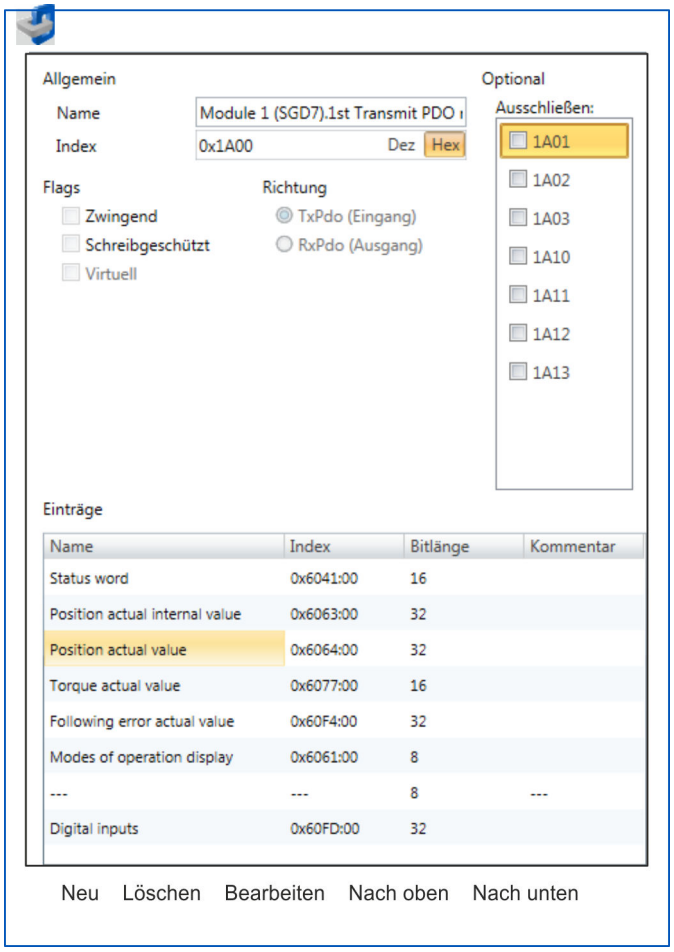
⇨ Dieser Dialog zeigt eine Auflistung aller PDOs für *"Module 1"* (Achse 1) und *"Module 2"* (Achse 2).

3. Durch Anwahl des entsprechenden PDO-Mappings können Sie mit [Bearbeiten] die PDOs bearbeiten. Wählen Sie das Mapping *"Module 1 (SGD7). 1st Transmit PDO mapping"* an und klicken Sie auf [Bearbeiten].

Bitte beachten Sie, dass aufgrund der Voreinstellung manche PDOs nicht bearbeitet werden können. Durch Deaktivierung bereits aktivierter PDOs können Sie die Bearbeitung von gesperrten PDOs frei geben.



⇨ Es öffnet sich der Dialog *"PDO bearbeiten"*. Bitte überprüfen Sie hier die aufgeführten PDO-Einstellungen und passen Sie diese ggf. an. Bitte berücksichtigen Sie hierbei auch die Reihenfolge der *"Einträge"* und ergänzen Sie diese entsprechend.



Für die Bearbeitung der *"Einträge"* stehen folgende Funktionen zur Verfügung:

- Neu
  - Hiermit können Sie in einem Dialogfenster einen neuen Eintrag anlegen, indem Sie aus dem *"CoE-Objektverzeichnis"* den entsprechenden Eintrag auswählen und Ihre Einstellungen vornehmen. Mit [OK] wird der Eintrag übernommen und in der Liste der Einträge aufgeführt.
- Löschen
  - Hiermit können Sie den angewählte Eintrag löschen.
- Bearbeiten
  - Hiermit können Sie allgemeinen Daten eines Eintrags bearbeiten.
- Nach oben/unten
  - Hiermit können Sie den angewählten Eintrag in der Liste nach oben bzw. nach unten bewegen.

4. Führen Sie für die Transmit PDOs folgende Einstellungen durch:

Eingänge: 1st Transmit PDO

Module 1 (SGD7). 1st Transmit PDO mapping	**Module 2 (SGD7). 1st Transmit PDO mapping**
Name: Module 1 (SGD7). 1st Transmit PDO mapping	Name: Module 2 (SGD7). 1st Transmit PDO mapping
Index: 0x1A00	Index: 0x1A10
Flags: Alles deaktiviert	
Richtung: TxPdo (Eingang): aktiviert	
Ausschließen: 1A01: deaktiviert	1A11: deaktiviert
Bitte diese Einstellungen beachten, da ansonsten die PDO-Mappings nicht zeitgleich aktiviert werden können!	

Einträge	**Modul 1 (Achse 1)**	**Modul 2 (Achse 2)**	**Bitlänge**
Name	**Index**	**Index**	
Status word	0x6041:00	0x6841:00	16Bit
Position actual internal value	0x6063:00	0x6863:00	32Bit
Position actual value	0x6064:00	0x6864:00	32Bit
Torque actual value	0x6077:00	0x6877:00	16Bit
Following error actual value	0x60F4:00	0x68F4:00	32Bit
Modes of operation display	0x6061:00	0x6861:00	8Bit
---	---	---	8Bit
Digital inputs	0x60FD:00	0x68FD:00	32Bit

Eingänge: 2nd Transmit PDO

Module 1 (SGD7). 2nd Transmit PDO mapping	**Module 2 (SGD7). 2st Transmit PDO mapping**
Name: Module 1 (SGD7). 2nd Transmit PDO mapping	Name: Module 2 (SGD7). 2st Transmit PDO mapping
Index: 0x1A01	Index: 0x1A11
Flags: Alles deaktiviert	
Richtung: TxPdo (Eingang): aktiviert	
Ausschließen: 1A00, 1A02, 1A03: deaktiviert	1A10, 1A12, 1A13: deaktiviert
Bitte diese Einstellungen beachten, da ansonsten die PDO-Mappings nicht zeitgleich aktiviert werden können!	

Einträge	**Modul 1 (Achse 1)**	**Modul 2 (Achse 2)**	**Bitlänge**
Name	**Index**	**Index**	
Touch probe status	0x60B9:00	0x68B9:00	16Bit
Touch probe 1 position value	0x60BA:00	0x68BA:00	32Bit
Touch probe 2 position value	0x60BC:00	0x68BC:00	32Bit
Velocity actual value	0x606C:00	0x686C:00	32Bit

5. Führen Sie für die Receive PDOs folgende Einstellungen durch:

Ausgänge: 1st Receive PDO

Module 1 (SGD7). 1st Receive PDO	Module 2 (SGD7). 1st Receive PDO
Name: Module 1 (SGD7). 1st Receive PDO mapping	Name: Module 2 (SGD7). 1st Receive PDO mapping
Index: 0x1600	Index: 0x1610
Flags: Alles deaktiviert	
Richtung: RxPdo (Ausgang): aktiviert	
Ausschließen: 1601, 1602, 1603: deaktiviert	1611, 1612, 1613: deaktiviert
Bitte diese Einstellungen beachten, da ansonsten die PDO-Mappings nicht zeitgleich aktiviert werden können!	

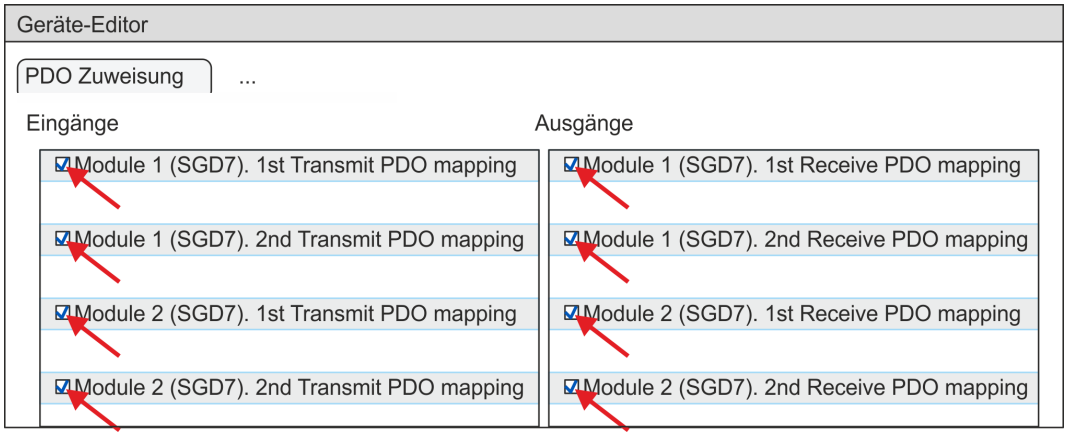
Einträge	Modul 1 (Achse 1)	Modul 2 (Achse 2)	Bitlänge
Name	**Index**	**Index**	
Control word	0x6040:00	0x6840:00	16Bit
Target position	0x607A:00	0x687A:00	32Bit
Target velocity	0x60FF:00	0x68FF:00	32Bit
Modes of operation	0x6060:00	0x6860:00	8Bit
---	---	---	8Bit
Touch probe function	0x60B8:00	0x68B8:00	16Bit

Ausgänge: 2nd Receive PDO

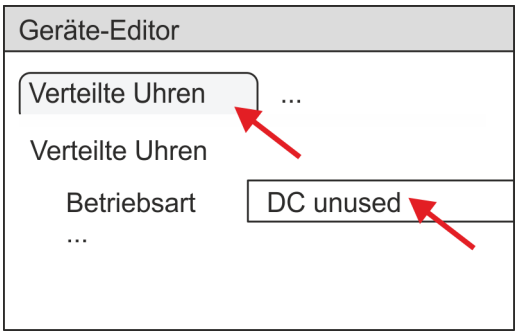
Module 1 (SGD7). 2nd Receive PDO	Module 2 (SGD7). 2nd Receive PDO
Name: Module 1 (SGD7). 2nd Receive PDO mapping	Name: Module 2 (SGD7). 2nd Receive PDO mapping
Index: 0x1601	Index: 0x1611
Flags: Alles deaktiviert	
Richtung: RxPdo (Ausgang): aktiviert	
Ausschließen: 1600, 1602, 1603: deaktiviert	1610, 1612, 1613: deaktiviert
Bitte diese Einstellungen beachten, da ansonsten die PDO-Mappings nicht zeitgleich aktiviert werden können!	

Einträge	Modul 1 (Achse 1)	Modul 2 (Achse 2)	Bitlänge
Name	**Index**	**Index**	
Profile velocity	0x6081:00	0x6881:00	32Bit
Profile acceleration	0x6083:00	0x6883:00	32Bit
Profile deceleration	0x6084:00	0x6884:00	32Bit

6. Aktivieren Sie für *"Module 1"* und *"Module 2"* in PDO-Zuweisung die PDOs 1 und 2 für die Ein- und Ausgänge. Alle nachfolgenden PDOs müssen deaktiviert bleiben. Sollte dies nicht möglich sein, überprüfen Sie bitte den jeweiligen PDO-Parameter *"Ausschließen"*.

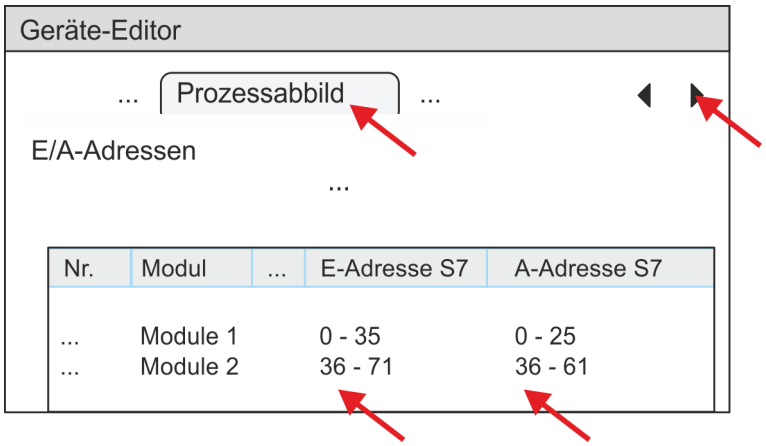


7. Wählen Sie im *"Geräte-Editor"* des *SPEED7 EtherCAT Manager* den Reiter *"Verteilte Uhren"* an und stellen Sie *"DC unused"* als *"Betriebsart"* ein.

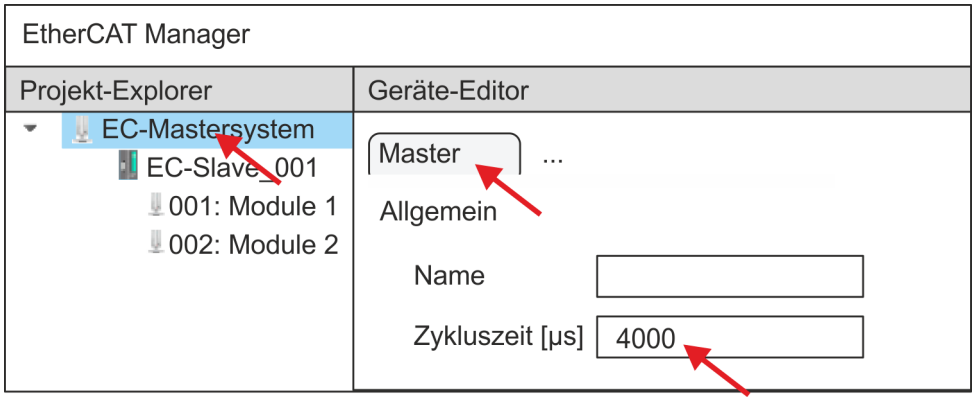


8. Wählen Sie im *"Geräte-Editor"* über die Pfeiltaste den Reiter *"Prozessabbild"* an und notieren Sie sich für die Parameter des Bausteins FB 874 - VMC_InitSigma7W_EC folgende PDO-Anfangsadressen:

- Module 1: *"E-Adresse S7"* → *"M1_PdoInputs"* (hier 0)
- Module 2: *"E-Adresse S7"* → *"M2_PdoInputs"* (hier 36)
- Module 1: *"A-Adresse S7"* → *"M1_PdoOutputs"* (hier 0)
- Module 2: *"A-Adresse S7"* → *"M2_PdoOutputs"* (hier 36)



9. Klicken Sie im *SPEED7 EtherCAT Manager* auf *"EC-Mastersystem"* und wählen Sie im *"Geräte-Editor"* den Reiter *"Master"* an.

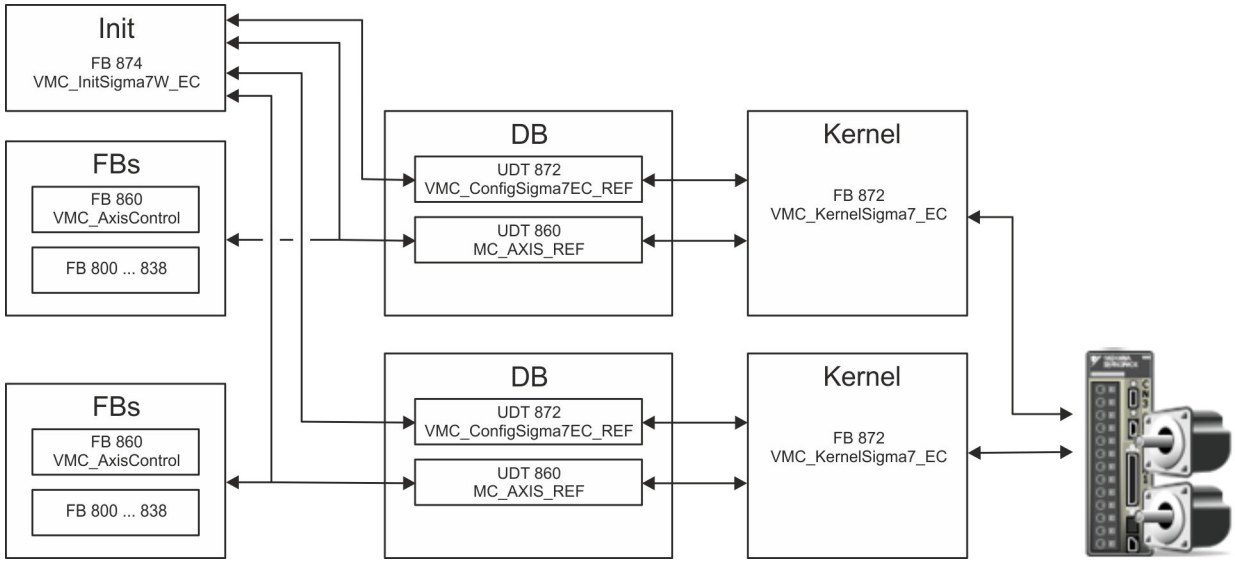


⇨ Stellen Sie für Sigma-7W (400V) Antriebe eine Zykluszeit von mindestens 4ms ein.

10. Indem Sie den Dialog des *SPEED7 EtherCAT Manager* mit [X] schließen, wird die Konfiguration in das *SPEED7 Studio* übernommen.

15.2.3.3.2 Anwender-Programm

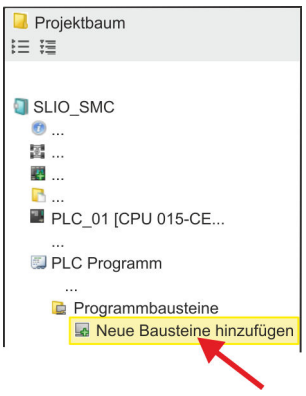
Programmstruktur



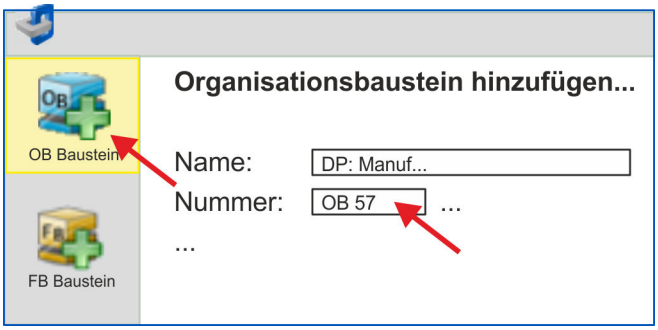
- DB
  Für jede Achse ist ein Datenbaustein (Achs-DB) für Konfiguration und Statusdaten anzulegen. Der Datenbaustein besteht aus folgenden Datenstrukturen:
  – UDT 872 - *VMC_Config*Sigma7EC_REF
    Die Datenstruktur beschreibt den Aufbau der Konfiguration des Antriebs.
    Spezifische Datenstruktur für *Sigma-7* EtherCAT.
  – UDT 860 - *MC_AXIS_REF*
    Die Datenstruktur beschreibt den Aufbau der Parameter und Statusinformationen von Antrieben.
    Allgemeine Datenstruktur für alle Antriebe und Bussysteme.
- FB 874 - *VMC_Init*Sigma7W_EC
  – Der *Init*-Baustein dient zur Konfiguration des Doppelachs-Antriebs.
  – Spezifischer Baustein für *Sigma-7W* EtherCAT.
  – Die Konfigurationsdaten für die Initialisierung sind im *Achs-DB* abzulegen.
- FB 872 - *VMC_Kernel*Sigma7_EC
  – Der *Kernel*-Baustein kommuniziert mit dem Antrieb über das entsprechende Bussystem, verarbeitet die Benutzeraufträge und liefert Statusmeldungen zurück.
  – Je Achse ist der FB 872 - *VMC_Kernel*Sigma7_EC aufzurufen.
  – Spezifischer Baustein für *Sigma-7* EtherCAT.
  – Der Austausch der Daten erfolgt mittels des *Achs-DB*.
- FB 860 - *VMC_AxisControl*
  – Universal-Baustein für alle Antriebe und Bussysteme.
  – Je Achse ist der FB 860 - *VMC_AxisControl* aufzurufen.
  – Unterstützt einfache Bewegungskommandos und liefert alle relevanten Statusmeldungen.
  – Der Austausch der Daten erfolgt mittels des *Achs-DB*.
  – Über die Instanzdaten des Bausteins können Sie zur Bewegungssteuerung und Statusabfrage eine Visualisierung anbinden.
  – Zusätzlich zum FB 860 - *VMC_AxisControl* haben Sie die Möglichkeit *PLCopen*-Bausteine zu nutzen.
- FB 800 ... FB 838 - PLCopen
  – Die PLCopen-Bausteine dienen zur Programmierung von Bewegungsabläufen und Statusabfragen.
  – Je Achse sind die PLCopen-Bausteine aufzurufen.

Programmierung

Bausteine in Projekt kopieren

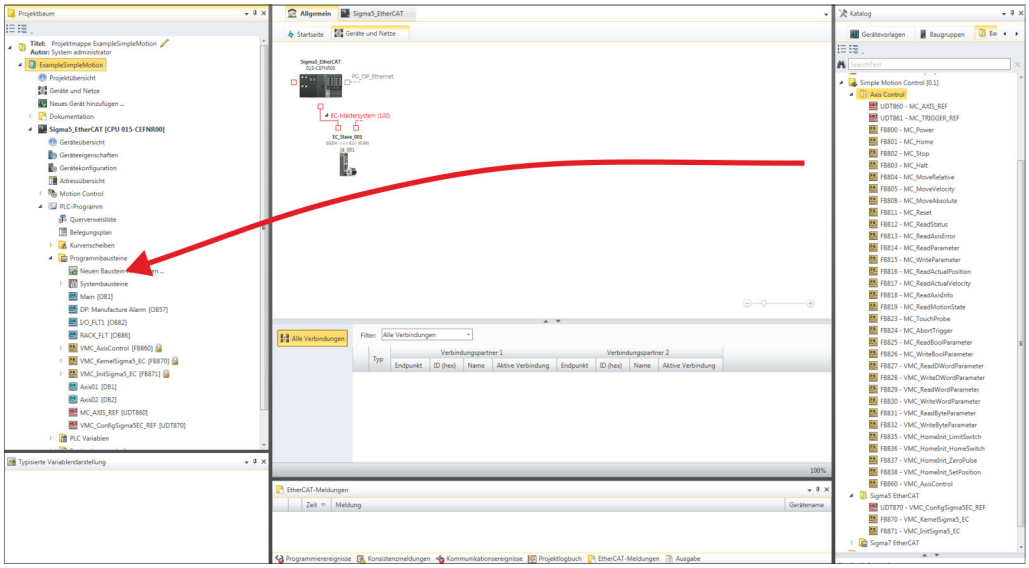


1. Klicken Sie im *Projektbaum* innerhalb der CPU unter *"PLC-Programm"*, *"Programmbausteine"* auf *"Neuen Baustein hinzufügen"*.



⇨ Das Dialogfenster *"Baustein hinzufügen"* öffnet sich.

2. Wählen Sie den Bausteintyp *"OB Baustein"* und fügen Sie nacheinander OB 57, OB 82 und OB 86 Ihrem Projekt hinzu.



3. Öffnen Sie im *"Katalog"* unter *"Bausteine"* *"Simple Motion Control"* und ziehen Sie per Drag&Drop folgende Bausteine in *"Programmbausteine"* des *Projektbaums*:

- *Sigma-7* EtherCAT:
  - UDT 872 - VMC_ConfigSigma7EC_REF
  - FB 872 - VMC_KernelSigma7_EC
  - FB 874 - VMC_InitSigma7W_EC
- Axis Control
  - UDT 860 - MC_AXIS_REF
  - Bausteine für die gewünschten Bewegungsabläufe

Achs-DB für *"Module 1"* anlegen

1. Fügen Sie Ihrem Projekt einen neuen DB als *Achs-DB* hinzu. Klicken Sie hierzu im *Projektbaum* innerhalb der CPU unter *"PLC-Programm"*, *"Programmbausteine"* auf *"Neuen Baustein hinzufügen"*, wählen Sie den Bausteintyp *"DB Baustein"* und vergeben Sie diesem den Namen "Axis01". Die DB-Nr. können Sie frei wählen wie z.B. DB 10.

⇨ Der Baustein wird angelegt und geöffnet.

2.
- Legen Sie in "Axis01" die Variable "Config" vom Typ UDT 872 an. Dies sind spezifische Achs-Konfigurationsdaten.
- Legen Sie in "Axis01" die Variable "Axis" vom Typ UDT 860 an. Während des Betriebs werden hier alle Betriebsdaten der Achse abgelegt.

Axis01 [DB10]

Bausteinstruktur

		Adr...	Name	Datentyp	...
		...	Config	UDT	[872]
		...	Axis	UDT	[860]

Achs-DB für *"Module 2"* anlegen

1. Fügen Sie Ihrem Projekt einen weiteren DB als *Achs-DB* hinzu und vergeben Sie diesem den Namen "Axis02". Die DB-Nr. können Sie frei wählen wie z.B. DB 11.

⇨ Der Baustein wird angelegt und geöffnet.

2. ■ Legen Sie in "Axis02" die Variable "Config" vom Typ UDT 872 an. Dies sind spezifische Achs-Konfigurationsdaten.

■ Legen Sie in "Axis02" die Variable "Axis" vom Typ UDT 860 an. Während des Betriebs werden hier alle Betriebsdaten der Achse abgelegt.

Axis02 [DB11]

Bausteinstruktur

		Adr...	Name	Datentyp	...
		...	Config	UDT	[872]
		...	Axis	UDT	[860]

OB 1

Konfiguration der Doppelachse

Öffnen Sie den OB 1 und programmieren Sie folgende FB-Aufrufe mit zugehörigen DBs:

FB 874 - VMC_InitSigma7W_EC, DB 874 *Kap. 15.2.3.4.3 "FB 874 - VMC_InitSigma7W_EC - Sigma-7W EtherCAT Initialisierung" Seite 639*

Geben Sie unter *M1/M2_PdoInputs* bzw. *M1/M2_PdoOutputs* die Adresse aus dem *SPEED7 EtherCAT Manager* für die entsprechende Achse an. *631*

```
⇨ CALL  "VMC_InitSigma7W_EC" , "DI_InitSgm7WETC01"
  Enable                    :=TRUE
  LogicalAddress            :=0
  M1_PdoInputs              :=0 (EtherCAT-Manager
                                 Module1: E-Adresse S7)

  M1_PdoOutputs             :=0 (EtherCAT-Manager
                                 Module1: A-Adresse S7)
  M1_EncoderType            :=2
  M1_EncoderResolutionBits  :=20
  M1_FactorPosition         :=1.048576e+006
  M1_FactorVelocity         :=1.048576e+006
  M1_FactorAcceleration     :=1.048576e+002
  M1_OffsetPosition         :=0.000000e+000
  M1_MaxVelocityApp         :=5.000000e+001
  M1_MaxAccelerationApp     :=1.000000e+002
  M1_MaxDecelerationApp     :=1.000000e+002
  M1_MaxVelocityDrive       :=6.000000e+001
  M1_MaxAccelerationDrive   :=1.500000e+002
  M1_MaxDecelerationDrive   :=1.500000e+002
  M1_MaxPosition            :=1.048500e+003
  M1_MinPosition            :=-1.048514e+003
  M1_EnableMaxPosition      :=TRUE
  M1_EnableMinPosition      :=TRUE
  M2_PdoInputs              :=36 (EtherCAT-Manager
                                 Module2: E-Adresse S7)

  M2_PdoOutputs             :=36 (EtherCAT-Manager
                                 Module2: A-Adresse S7)
  M2_EncoderType            :=2
  M2_EncoderResolutionBits  :=20
  M2_FactorPosition         :=1.048576e+006
  M2_FactorVelocity         :=1.048576e+006
  M2_FactorAcceleration     :=1.048576e+002
  M2_OffsetPosition         :=0.000000e+000
  M2_MaxVelocityApp         :=5.000000e+001
  M2_MaxAccelerationApp     :=1.000000e+002
  M2_MaxDecelerationApp     :=1.000000e+002
  M2_MaxVelocityDrive       :=6.000000e+001
  M2_MaxAccelerationDrive   :=1.500000e+002
  M2_MaxDecelerationDrive   :=1.500000e+002
  M2_MaxPosition            :=1.048500e+003
  M2_MinPosition            :=-1.048514e+003
  M2_EnableMaxPosition      :=TRUE
  M2_EnableMinPosition      :=TRUE
  M1_MinUserPosition        :=-1000.0
  M1_MaxUserPosition        :=1000.0
  M2_MinUserPosition        :=-1000.0
  M2_MaxUserPosition        :=1000.0
  Valid                     :="InitS7WEC1_Valid"
  Error                     :="InitS7WEC1_Error"
```

```
ErrorID                 :="InitS7WEC1_ErrorID"
M1_Config               :="Axis01".Config
M1_Axis                 :="Axis01".Axis
M2_Config               :="Axis02".Config
M2_Axis                 :="Axis02".Axis
```

Kernel für die jeweilige Achse beschalten

Der *Kernel* verarbeitet die Benutzerkommandos und gibt sie entsprechend aufbereitet an den Antrieb über das jeweilige Bussystem weiter.

FB 872 - VMC_KernelSigma7_EC, DB 872 für Achse 1

FB 872 - VMC_KernelSigma7_EC, DB 1872 für Achse 2 *Kap. 15.2.2.4.2 "FB 872 - VMC_KernelSigma7_EC - Sigma-7 EtherCAT Kernel" Seite 617*

```
⇨       CALL  "VMC_KernelSigma7_EC" , DB 872
         Init  :="KernelS7WEC1_Init"
         Config:="Axis01".Config
         Axis  :="Axis01".Axis

        CALL  "VMC_KernelSigma7_EC" , DB 1872
         Init  :="KernelS7WEC2_Init"
         Config:="Axis02".Config
         Axis  :="Axis02".Axis
```

Baustein für Bewegungsabläufe beschalten

Zur Vereinfachung soll hier die Beschaltung des FB 860 - VMC_AxisControl gezeigt werden. Dieser Universalbaustein unterstützt einfache Bewegungskommandos und liefert Statusmeldungen zurück. Die Ein- und Ausgänge können Sie individuell beschalten. Bitte geben Sie unter *"Axis"* die Referenz zu den entsprechenden Achsdaten im *Achs-DB* an.

FB 860 - VMC_AxisControl, DB 860 *Kap. 15.8.2.2 "FB 860 - VMC_AxisControl - Control-Baustein Achskontrolle" Seite 754*

```
⇨       CALL  "VMC_AxisControl" , "DI_AxisControl01"
         AxisEnable       :="AxCtrl1_AxisEnable"
         AxisReset        :="AxCtrl1_AxisReset"
         HomeExecute      :="AxCtrl1_HomeExecute"
         HomePosition     :="AxCtrl1_HomePosition"
         StopExecute      :="AxCtrl1_StopExecute"
         MvVelocityExecute:="AxCtrl1_MvVelExecute"
         MvRelativeExecute:="AxCtrl1_MvRelExecute"
         MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
         PositionDistance :="AxCtrl1_PositionDistance"
         Velocity         :="AxCtrl1_Velocity"
         Acceleration     :="AxCtrl1_Acceleration"
         Deceleration     :="AxCtrl1_Deceleration"
         JogPositive      :="AxCtrl1_JogPositive"
         JogNegative      :="AxCtrl1_JogNegative"
         JogVelocity      :="AxCtrl1_JogVelocity"
         JogAcceleration  :="AxCtrl1_JogAcceleration"
         JogDeceleration  :="AxCtrl1_JogDeceleration"
         AxisReady        :="AxCtrl1_AxisReady"
         AxisEnabled      :="AxCtrl1_AxisEnabled"
         AxisError        :="AxCtrl1_AxisError"
         AxisErrorID      :="AxCtrl1_AxisErrorID"
         DriveWarning     :="AxCtrl1_DriveWarning"
         DriveError       :="AxCtrl1_DriveError"
         DriveErrorID     :="AxCtrl1_DriveErrorID"
         IsHomed          :="AxCtrl1_IsHomed"
         ModeOfOperation  :="AxCtrl1_ModeOfOperation"
         PLCopenState     :="AxCtrl1_PLCopenState"
         ActualPosition   :="AxCtrl1_ActualPosition"
         ActualVelocity   :="AxCtrl1_ActualVelocity"
         CmdDone          :="AxCtrl1_CmdDone"
         CmdBusy          :="AxCtrl1_CmdBusy"
         CmdAborted       :="AxCtrl1_CmdAborted"
         CmdError         :="AxCtrl1_CmdError"
         CmdErrorID       :="AxCtrl1_CmdErrorID"
         DirectionPositive:="AxCtrl1_DirectionPos"
         DirectionNegative:="AxCtrl1_DirectionNeg"
         SWLimitMinActive :="AxCtrl1_SWLimitMinActive"
         SWLimitMaxActive :="AxCtrl1_SWLimitMaxActive"
         HWLimitMinActive :="AxCtrl1_HWLimitMinActive"
         HWLimitMaxActive :="AxCtrl1_HWLimitMaxActive"
         Axis             :="Axis...".Axis
```

Geben Sie unter *Axis* für die Achse 1 "Axis01" und für die Achse 2 "Axis02" an.

Für komplexe Bewegungsaufgaben können Sie die PLCopen-Bausteine verwenden. Hier müssen Sie ebenfalls unter Axis die Referenz zu den Achsdaten im Achs-DB angeben.

Ihr Projekt beinhaltet nun folgende Bausteine:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT

- FB 860 - VMC_AxisControl mit Instanz-DB
- FB 872 - VMC_KernelSigma7_EC mit Instanz-DB
- FB 874 - VMC_InitSigma7W_EC mit Instanz-DB
- UDT 860 - MC_Axis_REF
- UDT 872 - VMC_ConfigSigma7EC_REF

Zeitlicher Ablauf

1. Wählen Sie *"Projekt ➔ Alles übersetzen"* und übertragen Sie das Projekt in Ihre CPU.

   ⇨ Sie können jetzt Ihre Applikation in Betrieb nehmen.

VORSICHT!
Bitte beachten Sie immer die Sicherheitshinweise zu ihrem Antrieb, insbesondere bei der Inbetriebnahme!

2. Bevor der Doppelachs-Antrieb gesteuert werden kann, muss diese initialisiert werden. Rufen Sie hierzu den *Init*-Baustein FB 874 - VMC_InitSigma7W_EC mit *Enable* = TRUE auf.

   ⇨ Der Ausgang *Valid* meldet TRUE zurück. Im Fehlerfall können Sie durch Auswertung der *ErrorID* den Fehler ermitteln.

   Den *Init*-Baustein müssen Sie erneut aufrufen, wenn Sie einen neuen Achs-DB laden oder Parameter am *Init*-Baustein geändert wurden.

Fahren Sie erst fort, wenn der Init-Baustein keinen Fehler meldet!

3. Stellen Sie sicher, dass für jede Achse der *Kernel*-Baustein FB 872 - VMC_KernelSigma7_EC zyklisch aufgerufen wird. Auf diese Weise werden Steuersignale an den Antrieb übergeben und Statusmeldungen übermittelt.
4. Programmieren Sie für jede Achse Ihre Applikation mit dem FB 860 - VMC_AxisControl oder mit den PLCopen Bausteinen.

Steuerung des Antriebs über HMI

Sie haben die Möglichkeit über ein HMI Ihren Antrieb zu steuern. Hierzu gibt es für Movicon eine vorgefertigte Symbolbibliothek für den Zugriff auf den VMC_AxisControl Funktionsbaustein. *Kap. 15.9 "Antrieb über HMI steuern" Seite 825*

15.2.3.4 Antriebsspezifische Bausteine

Die PLCopen-Bausteine zur Achskontrolle finden Sie hier: Kap. 15.8 "Bausteine zur Achskontrolle" Seite 751

15.2.3.4.1 UDT 872 - VMC_ConfigSigma7EC_REF - *Sigma-7* EtherCAT Datenstruktur Achskonfiguration

Dies ist eine benutzerdefinierte Datenstruktur, die Informationen zu den Konfigurationsdaten beinhaltet. Die UDT ist speziell angepasst an die Verwendung eines *Sigma-7*-Antriebs, welcher über EtherCAT angebunden ist.

15.2.3.4.2 FB 872 - VMC_KernelSigma7_EC - *Sigma-7* EtherCAT Kernel

Beschreibung

Dieser Baustein setzt die Antriebskommandos für eine *Sigma-7* Achse über EtherCAT um und kommuniziert mit dem Antrieb. Je *Sigma-7* Achse ist eine Instanz dieses FBs zyklisch aufzurufen.

Bitte beachten Sie, dass dieser Baustein intern den SFB 238 aufruft.

Im SPEED7 Studio wird dieser Baustein automatisch in Ihr Projekt eingefügt.

Parameter	Deklaration	Datentyp	Beschreibung
Init	INPUT	BOOL	Mit einer Flanke 0-1 wird der Baustein intern zurückgesetzt. Hierbei werden bestehende Bewegungskommandos abgebrochen und der Baustein wird initialisiert.
Config	IN_OUT	UDT872	Datenstruktur zur Übergabe von achsabhängigen Konfigurationsdaten an den *AxisKernel*.
Axis	IN_OUT	MC_AXIS_REF	Datenstruktur zur Übergabe von achsabhängigen Informationen an *AxisKernel* und PLCopen-Bausteine.

15.2.3.4.3 FB 874 - VMC_InitSigma7W_EC - *Sigma-7W* EtherCAT Initialisierung

Beschreibung

Dieser Baustein dient zur Konfiguration der Doppelachse eines *Sigma-7W*-Antriebs. Der Baustein ist speziell angepasst an die Verwendung eines *Sigma-7W*-Antriebs, welcher über EtherCAT angebunden ist.

Parameter	Deklaration	Datentyp	Beschreibung
Enable	INPUT	BOOL	Freigabe der Initialisierung
LogicalAddress	INPUT	INT	Startadresse der PDO-Eingangsdaten
M1_PdoInputs	INPUT	INT	Startadresse der Eingabe-PDOs für Achse 1
M1_PdoOutputs	INPUT	INT	Startadresse der Ausgabe-PDOs für Achse 1

Parameter	Deklaration	Datentyp	Beschreibung
M1_EncoderType	INPUT	INT	Encoder-Typ von Achse 1 ■ 1: Absolut-Encoder ■ 2: Inkremental-Encoder
M1_EncoderResolutionBits	INPUT	INT	Anzahl der Bits, die einer Geber-Umdrehung von Achse 1 entsprechen. Default: 20
M1_FactorPosition	INPUT	REAL	Faktor zur Umrechnung der Position von Benutzereinheiten [u] in Antriebseinheiten [Inkremente] und zurück von Achse 1. Es gilt: $p_{[Inkremente]} = p_{[u]}$ x *FactorPosition* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2701:1 und 0x2701:2 vorgeben können. Dieser sollte 1 sein.
M1_FactorVelocity	INPUT	REAL	Faktor zur Umrechnung der Geschwindigkeit von Benutzereinheiten [u/s] in Antriebseinheiten [Inkremente/s] und zurück von Achse 1. Es gilt: $v_{[Inkremente/s]} = v_{[u/s]}$ x *FactorVelocity* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2702:1 und 0x2702:2 vorgeben können. Dieser sollte 1 sein.
M1_FactorAcceleration	INPUT	REAL	Faktor zur Umrechnung der Beschleunigung von Benutzereinheiten [u/s^2] in Antriebseinheiten [10^{-4} x Inkremente/s^2] und zurück von Achse 1. Es gilt: 10^{-4} x $a_{[Inkremente/s^2]} = a_{[u/s^2]}$ x *FactorAcceleration* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2703:1 und 0x2703:2 vorgeben können. Dieser sollte 1 sein.
M1_OffsetPosition	INPUT	REAL	Offset für die Nullposition von Achse 1 [u].
M1_MaxVelocityApp	INPUT	REAL	Maximale Geschwindigkeit der Applikation von Achse 1 [u/s]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.
M1_MaxAccelerationApp	INPUT	REAL	Maximale Beschleunigung der Applikation von Achse 1 [u/s^2]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.
M1_MaxDecelerationApp	INPUT	REAL	Maximale Verzögerung der Applikation von Achse 1 [u/s^2]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.
M1_MaxPosition	INPUT	REAL	Maximale Position für die Überwachung der Softwarelimits von Achse 1 [u].
M1_MinPosition	INPUT	REAL	Minimale Position für die Überwachung der Softwarelimits von Achse 1 [u].
M1_EnableMaxPosition	INPUT	BOOL	Überwachung maximale Position von Achse 1 ■ TRUE: Aktivierung der Überwachung der maximalen Position.

Parameter	Deklaration	Datentyp	Beschreibung
M1_EnableMinPosition	INPUT	BOOL	Überwachung minimale Position von Achse 1 ■ TRUE: Aktivierung der Überwachung der minimalen Position.
M2_PdoInputs	INPUT	INT	Startadresse der Eingabe-PDOs für Achse 2
M2_PdoOutputs	INPUT	INT	Startadresse der Ausgabe-PDOs für Achse 2
M2_EncoderType	INPUT	INT	Encoder-Typ von Achse 2 ■ 1: Absolut-Encoder ■ 2: Inkremental-Encoder
M2_EncoderResolutionBits	INPUT	INT	Anzahl der Bits, die einer Geber-Umdrehung von Achse 2 entsprechen. Default: 20
M2_FactorPosition	INPUT	REAL	Faktor zur Umrechnung der Position von Benutzereinheiten [u] in Antriebseinheiten [Inkremente] und zurück von Achse 2. Es gilt: $p_{[Inkremente]} = p_{[u]}$ x *FactorPosition* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2701:1 und 0x2701:2 vorgeben können. Dieser sollte 1 sein.
M2_FactorVelocity	INPUT	REAL	Faktor zur Umrechnung der Geschwindigkeit von Benutzereinheiten [u/s] in Antriebseinheiten [Inkremente/s] und zurück von Achse 2. Es gilt: $v_{[Inkremente/s]} = v_{[u/s]}$ x *FactorVelocity* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2702:1 und 0x2702:2 vorgeben können. Dieser sollte 1 sein.
M2_FactorAcceleration	INPUT	REAL	Faktor zur Umrechnung der Beschleunigung von Benutzereinheiten [u/s^2] in Antriebseinheiten [10^{-4} x Inkremente/s^2] und zurück von Achse 2. Es gilt: 10^{-4} x $a_{[Inkremente/s^2]} = a_{[u/s^2]}$ x *FactorAcceleration* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2703:1 und 0x2703:2 vorgeben können. Dieser sollte 1 sein.
M2_OffsetPosition	INPUT	REAL	Offset für die Nullposition von Achse 2 [u].
M2_MaxVelocityApp	INPUT	REAL	Maximale Geschwindigkeit der Applikation von Achse 2 [u/s]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.
M2_MaxAccelerationApp	INPUT	REAL	Maximale Beschleunigung der Applikation von Achse 2 [u/s^2]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.
M2_MaxDecelerationApp	INPUT	REAL	Maximale Verzögerung der Applikation von Achse 2 [u/s^2]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.
M2_MaxPosition	INPUT	REAL	Maximale Position für die Überwachung der Softwarelimits von Achse 2 [u].

Parameter	Deklaration	Datentyp	Beschreibung
M2_MinPosition	INPUT	REAL	Minimale Position für die Überwachung der Softwarelimits von Achse 2 [u].
M2_EnableMaxPosition	INPUT	BOOL	Überwachung maximale Position von Achse 2 ■ TRUE: Aktivierung der Überwachung der maximalen Position.
M2_EnableMinPosition	INPUT	BOOL	Überwachung minimale Position von Achse 2 ■ TRUE: Aktivierung der Überwachung der minimalen Position.
M1_MinUserPosition	OUTPUT	REAL	Minimale Benutzerposition für Achse 1 basierend auf dem minimalen Encoder Wert von 0x80000000 und dem *FactorPosition* [u].
M1_MaxUserPosition	OUTPUT	REAL	Maximale Benutzerposition für Achse 1 basierend auf dem maximalen Encoder Wert von 0x7FFFFFFF und dem *FactorPosition* [u].
M2_MinUserPosition	OUTPUT	REAL	Minimale Benutzerposition für Achse 2 basierend auf dem minimalen Encoder Wert von 0x80000000 und dem *FactorPosition* [u].
M2_MaxUserPosition	OUTPUT	REAL	Maximale Benutzerposition für Achse 2 basierend auf dem maximalen Encoder Wert von 0x7FFFFFFF und dem *FactorPosition* [u].
Valid	OUTPUT	BOOL	Initialisierung ■ TRUE: Initialisierung ist gültig.
Error	OUTPUT	BOOL	■ Fehler – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden. Die Achse wird gesperrt.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
M1_Config	IN_OUT	UDT872	Datenstruktur zur Übergabe von achsabhängigen Konfigurationsdaten an den *AxisKernel* für Achse 1.
M1_Axis	IN_OUT	MC_AXIS_REF	Datenstruktur zur Übergabe von achsabhängigen Informationen an *AxisKernel* und PLCopen-Bausteine für Achse 1.
M2_Config	IN_OUT	UDT872	Datenstruktur zur Übergabe von achsabhängigen Konfigurationsdaten an den *AxisKernel* für Achse 2.
M2_Axis	IN_OUT	MC_AXIS_REF	Datenstruktur zur Übergabe von achsabhängigen Informationen an *AxisKernel* und PLCopen-Bausteine für Achse 2.

15.3 Einsatz *Sigma-5/7* PROFINET

15.3.1 Einsatz *Sigma-5* PROFINET

15.3.1.1 Übersicht

Voraussetzung

- SPEED7 Studio ab V1.8
- CPU mit PROFINET-IO-Controller wie z.B. CPU 015-CEFPR01
- *Sigma-5*-Antrieb mit PROFINET-Optionskarte

Schritte der Projektierung

1. Parameter am Antrieb über Drehschalter der *Sigma-5*-Optionskarte einstellen.
2. Hardwarekonfiguration im VIPA *SPEED7 Studio.*
   - Projektierung einer CPU mit PROFINET-IO-Controller.
   - Projektierung eines *Sigma-5* PROFINET-Antriebs.
3. Programmierung im VIPA *SPEED7 Studio*
   - *Init*-Baustein zur Konfiguration der Achse beschalten.
   - *Kernel*-Baustein zur Kommunikation mit der Achse beschalten.
   - Bausteine für die Bewegungsabläufe beschalten.

15.3.1.2 Parameter am Antrieb einstellen

Parameter Sigma-5

Vor der Erstinbetriebnahme müssen Sie die PROFINET-Options-Karte des Sigma-5 Antriebs auf *"Telegramm 100 (all OP modes)"* einstellen. Hierzu befindet sich auf der Frontseite der Optionskarte der Drehschalter *"S12"*. Drehen Sie diesen auf die Position *"E"*. Weitere Einstellungen sind für die PROFINET-Kommunikation nicht erforderlich.

Bitte beachten Sie, dass Sie gemäß ihren Anforderungen die entsprechende Fahrtrichtung für Ihren Antrieb freigeben. Verwenden Sie hierzu die Parameter Pn50A (P-OT) bzw. Pn50B (N-OT) in Sigma Win+.

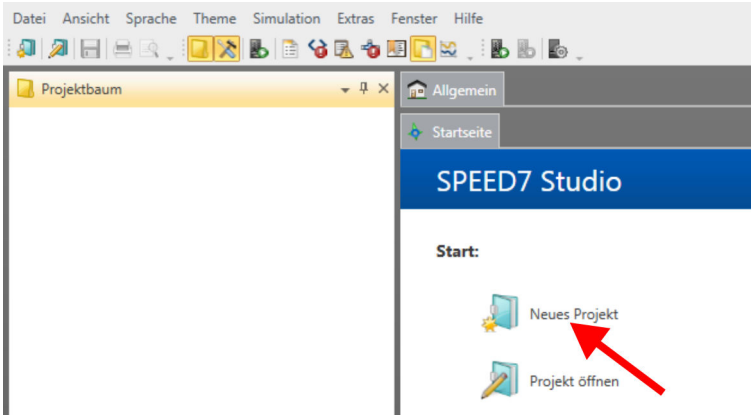
15.3.1.3 Einsatz im VIPA *SPEED7 Studio*

15.3.1.3.1 Hardware-Konfiguration System MICRO

CPU im Projekt anlegen

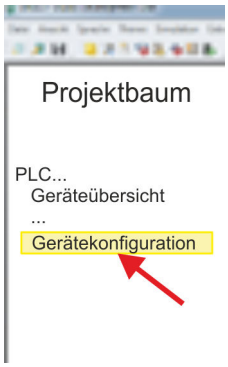
Bitte verwenden Sie für die Projektierung das *SPEED7 Studio* ab V1.8

1. Starten Sie das *SPEED7 Studio*.



2. Erstellen sie auf der Startseite mit *"Neues Projekt"* ein neues Projekt und vergeben Sie einen *"Projektnamen"*.

   ⇨ Ein neues Projekt wird angelegt und in die Sicht *"Geräte und Netze"* gewechselt.

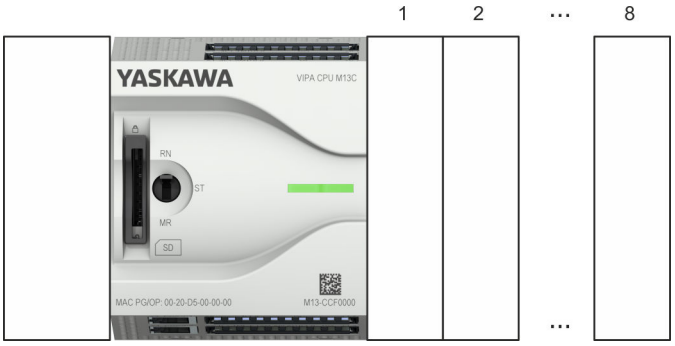


3. Klicken Sie im *Projektbaum* auf *"Neues Gerät hinzufügen ..."*.

   ⇨ Es öffnet sich ein Dialog für die Geräteauswahl.

4. Wählen Sie unter den *"Gerätevorlagen"* die System MICRO CPU M13-CCF0000 V2.4.... und klicken Sie auf [OK].

   ⇨ Die CPU wird in *"Geräte und Netze"* eingefügt und die *"Gerätekonfiguration"* geöffnet.

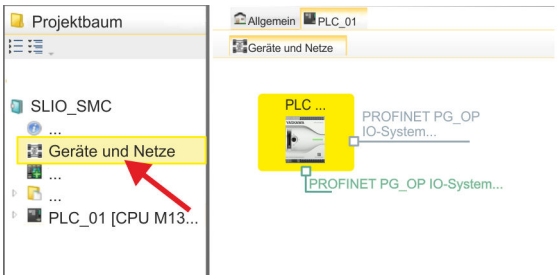


Gerätekonfiguration

Slot	Baugruppe	...	...	...	...
0	CPU M13-CCF0000				
-X2	MPI-Schnittstelle				
-X3	PROFINET PG_OP IO-System				
...	...			...	

Ethernet-PG/OP-Kanal parametrieren

1. Klicken Sie im *Projektbaum* auf *"Geräte und Netze"*.
   ⇨ Sie erhalten eine grafische Objekt-Ansicht Ihrer CPU. Hierbei werden beide Schnittstellen des PROFINET bzw. Ethernet-PG/OP Kanal Switch unter identischem Namen aufgeführt.



2. Klicken Sie auf eins der Netzwerke *"PROFINET PG_OP_Ethernet IO-System ..."*.
3. Wählen Sie *"Kontextmenü ➔ Eigenschaften der Schnittstelle"*.
   ⇨ Es öffnet sich ein Dialogfenster. Hier können Sie IP-Adressdaten für Ihren Ethernet-PG/OP-Kanal angeben. Gültige IP-Adress-Parameter erhalten Sie von Ihrem Systemadministrator.
4. Bestätigen Sie Ihre Eingabe mit [OK].
   ⇨ Die IP-Adressdaten werden in Ihr Projekt übernommen und in *"Geräte und Netze"* unter *"Lokale Baugruppen"* aufgelistet.

   Nach der Übertragung Ihres Projekts ist Ihre CPU über die angegebenen IP-Adressdaten via Ethernet-PG/OP-Kanal erreichbar.

GSDML-Datei installieren

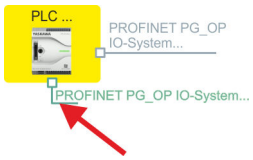
Damit der *Sigma-5* PROFINET Antrieb im *SPEED7 Studio* konfiguriert werden kann, muss die entsprechende GSDML-Datei installiert sein. In der Regel wird das *SPEED7 Studio* mit aktuellen GSDML-Dateien ausgeliefert und Sie können diesen Teil überspringen. Sollte Ihre GSDML-Datei veraltet sein, finden Sie die aktuellste GSDML-Datei für den *Sigma-5* PROFINET Antrieb unter *www.yaskawa.eu.com* unter *"Service ➔ Drives & Motion Software"*.

1. Laden Sie die zu Ihrem Antrieb passende GSDML-Datei herunter. Entpacken Sie diese falls erforderlich.
2. Gehen Sie in Ihr *SPEED7 Studio*.
3. Öffnen Sie mit *"Extras ➔ Gerätebeschreibungsdatei installieren (PROFINET - GSDML)"* das zugehörige Dialogfenster.
4. Geben Sie unter *"Quellpfad"* die GSDML-Datei an und installieren Sie diese mit [Installieren].
   ⇨ Die Geräte der GSDML-Datei steht Ihnen nun zur Verfügung.

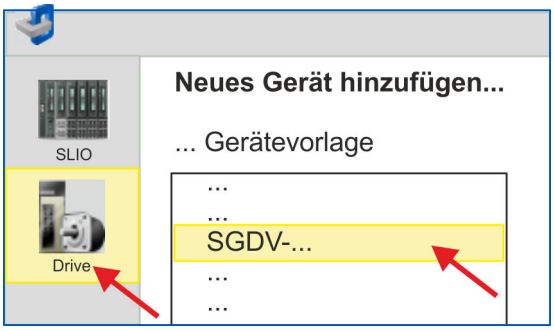
Sigma-5 Antrieb hinzufügen

Bei der Konfiguration ist für jede Achse ein *Sigma-5* PROFINET-IO-Device zu konfigurieren.

1. Klicken Sie im Projektbaum auf *"Geräte und Netze"*.
2. Klicken Sie auf *"PROFINET PG_OP_Ethernet IO-System ..."* und wählen sie *"Kontextmenü ➔ Neues Gerät hinzufügen"*.

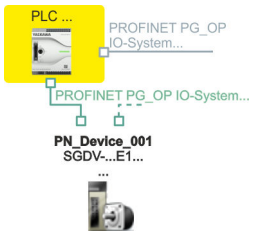


⇨ Es öffnet sich die Gerätevorlage zur Auswahl eines PROFINET-Devices.



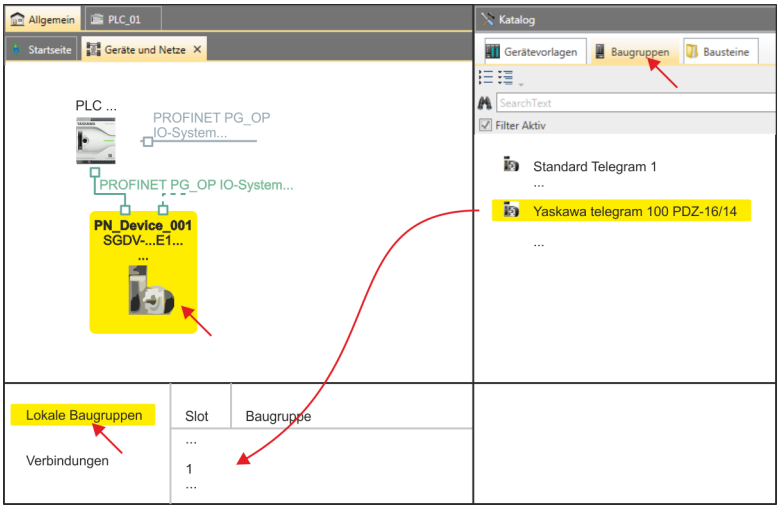
3. Wählen Sie Ihren *Sigma-5* Antrieb aus:
   - SGDV-xxxxE1...

   Bestätigen Sie Ihre Angaben mit [OK]. Sollte Ihr Antrieb nicht vorhanden sein, müssen Sie die entsprechende GSDML-Datei wie weiter oben beschrieben installieren.



⇨ Der *Sigma-5* Antrieb wird an Ihren PROFINET-IO-Controller angebunden.

4. Klicken Sie auf den *Sigma-5* Antrieb.



5. Wählen Sie unter *"Katalog"* den Reiter *"Baugruppen"* an.

⇨ Die Telegramme für den *Sigma-5* Antrieb werden aufgelistet.

6. Wählen Sie *"Yaskawa telegram 100 PZD..."* und ziehen Sie dieses unter *"Lokale Baugruppen"* auf *"Slot 1"*.

⇨ Telegram 100 wird mit den entsprechenden Untergruppen eingefügt.

Die Verbindung zwischen den Achsen in der Hardware-Konfiguration und Ihrem Anwenderprogramm erfolgt durch Angabe folgender Moduleigenschaften in den Aufrufparametern des FB 891 - VMC InitSigma_PN:

- *Moduleeigenschaft "Parameter Access Point": Diagnoseadresse von Steckplatz 1 der Steckplatzübersicht.*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress: Angabe der Diagnoseadresse von Steckplatz 1 der Steckplatzübersicht.*
- *Moduleeigenschaft "YASKAWA Telegram PZD...": Jeweilige Startadresse des Ein-/Ausgabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "InputsStartAddress": Angabe der Startadresse des Eingabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "OutputsStartAddress": Angabe der Startadresse des Ausgabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "LogicalAddress": Angabe des kleineren Werts der Startadressen des Ein-/Ausgabe-Adressbereichs.*

- Anwenderprogramm ↬ *654*
- FB 891 - VMC InitSigma_PN ↬ *676*

Beispiel Hardware-Konfiguration

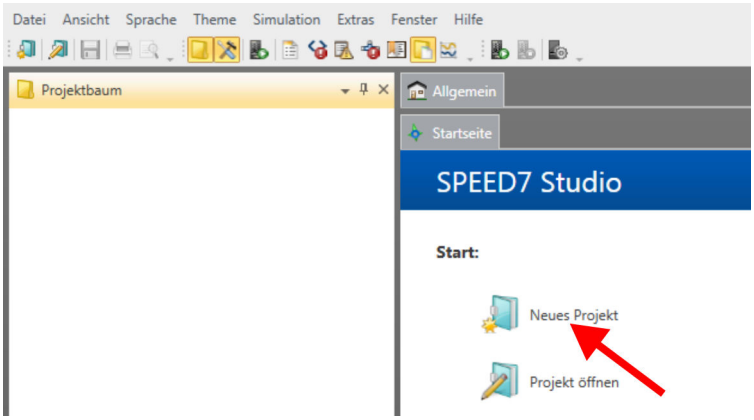
Slot	Baugruppe	...	E-Adr.	A-Adr.	Diagnoseadresse
0	SGDV-OCB03A		2045		2045
X1	PN-IO		2039		2039
X1 P1	Port 1		2038		2038
X1 P2	Port 2		2037		2037
1	DO with YASKAWA telegr.100, PZD-16/14		2036		2036
1.1	Parameter Access Point		2036		2036
1.2	YASKAWA telegram, PZD-16/14		0-27	0-31	2036

15.3.1.3.2 Hardware-Konfiguration System SLIO

CPU im Projekt anlegen

Bitte verwenden Sie für die Projektierung das *SPEED7 Studio* ab V1.8

1. Starten Sie das *SPEED7 Studio*.

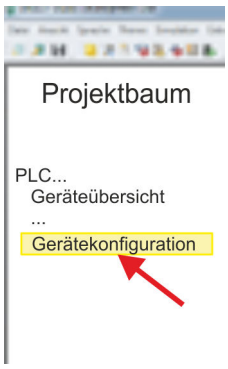


2. Erstellen sie auf der Startseite mit *"Neues Projekt"* ein neues Projekt und vergeben Sie einen *"Projektnamen"*.

⇨ Ein neues Projekt wird angelegt und in die Sicht *"Geräte und Netze"* gewechselt.

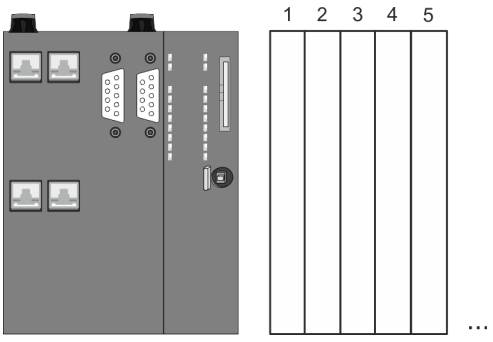
3. Klicken Sie im *Projektbaum* auf *"Neues Gerät hinzufügen ..."*.

⇨ Es öffnet sich ein Dialog für die Geräteauswahl.



4. Wählen Sie unter den *"Gerätevorlagen"* Ihre PROFINET-CPU wie z.B. die CPU 015-CEFPR01 und klicken Sie auf [OK].

⇨ Die CPU wird in *"Geräte und Netze"* eingefügt und die *"Gerätekonfiguration"* geöffnet.

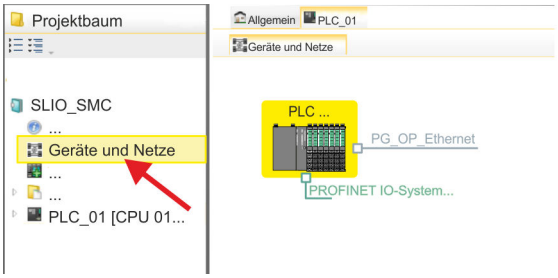


Gerätekonfiguration

Slot	Baugruppe	...	...	...	...
0	CPU 015-CEFPR01				
-X1	PG_OP_Ethernet				
-X3	MPI-Schnittstelle				
-X4	PROFINET-IO-System			...	
...					

Ethernet-PG/OP-Kanal parametrieren

1. Klicken Sie im *Projektbaum* auf *"Geräte und Netze"*.
   ⇨ Sie erhalten eine grafische Objekt-Ansicht Ihrer CPU.



2. Klicken Sie auf das Netzwerk *"PG_OP_Ethernet"*.
3. Wählen Sie *"Kontextmenü ➔ Eigenschaften der Schnittstelle"*.
   ⇨ Es öffnet sich ein Dialogfenster. Hier können Sie IP-Adressdaten für Ihren Ethernet-PG/OP-Kanal angeben. Gültige IP-Adress-Parameter erhalten Sie von Ihrem Systemadministrator.
4. Bestätigen Sie Ihre Eingabe mit [OK].
   ⇨ Die IP-Adressdaten werden in Ihr Projekt übernommen und in *"Geräte und Netze"* unter *"Lokale Baugruppen"* aufgelistet.

   Nach der Übertragung Ihres Projekts ist Ihre CPU über die angegebenen IP-Adressdaten via Ethernet-PG/OP-Kanal erreichbar.

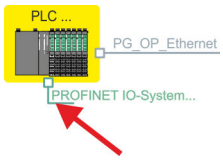
GSDML-Datei installieren

Damit der *Sigma-5* PROFINET Antrieb im *SPEED7 Studio* konfiguriert werden kann, muss die entsprechende GSDML-Datei installiert sein. In der Regel wird das *SPEED7 Studio* mit aktuellen GSDML-Dateien ausgeliefert und Sie können diesen Teil überspringen. Sollte Ihre GSDML-Datei veraltet sein, finden Sie die aktuellste GSDML-Datei für den *Sigma-5* PROFINET Antrieb unter *www.yaskawa.eu.com* unter *"Service ➔ Drives & Motion Software"*.

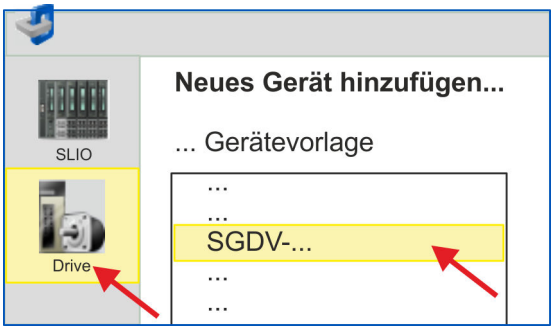
1. Laden Sie die zu Ihrem Antrieb passende GSDML-Datei herunter. Entpacken Sie diese falls erforderlich.
2. Gehen Sie in Ihr *SPEED7 Studio*.
3. Öffnen Sie mit *"Extras ➔ Gerätebeschreibungsdatei installieren (PROFINET - GSDML)"* das zugehörige Dialogfenster.
4. Geben Sie unter *"Quellpfad"* die GSDML-Datei an und installieren Sie diese mit [Installieren].
   ⇨ Die Geräte der GSDML-Datei steht Ihnen nun zur Verfügung.

Sigma-5 Antrieb hinzufügen

1. Klicken Sie im Projektbaum auf *"Geräte und Netze"*.
2. Klicken Sie auf *"PROFINET IO-System ..."* und wählen sie *"Kontextmenü ➔ Neues Gerät hinzufügen"*.

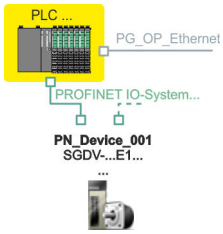


⇨ Es öffnet sich die Gerätevorlage zur Auswahl eines PROFINET-Devices.



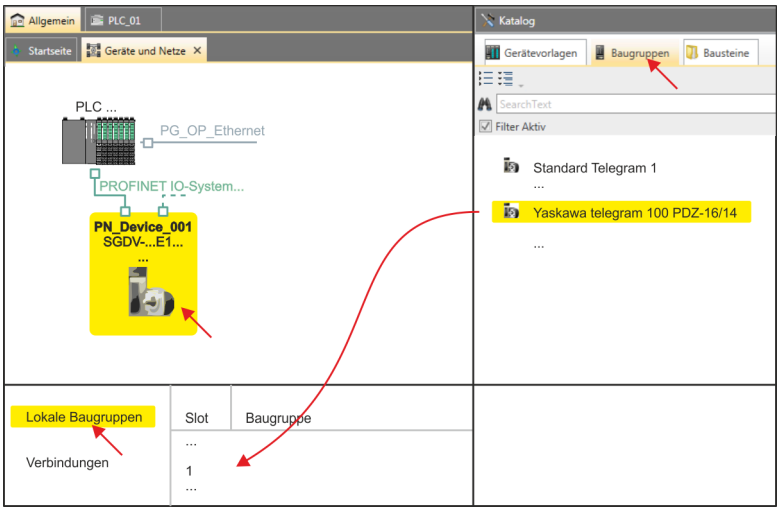
3. Wählen Sie Ihren *Sigma-5* Antrieb aus:
   - SGDV-xxxxE1...

   Bestätigen Sie Ihre Angaben mit [OK]. Sollte Ihr Antrieb nicht vorhanden sein, müssen Sie die entsprechende GSDML-Datei wie weiter oben beschrieben installieren.



⇨ Der *Sigma-5* Antrieb wird an Ihren PROFINET-IO-Controller angebunden.

4. Klicken Sie auf den *Sigma-5* Antrieb



5. Wählen Sie unter *"Katalog"* den Reiter *"Baugruppen"* an.

   ⇨ Die Telegramme für den *Sigma-5* Antrieb werden aufgelistet.

6. Wählen Sie *"Yaskawa telegram 100 PZD..."* und ziehen Sie dieses unter *"Lokale Baugruppen"* auf *"Slot 1"*.

⇨ Telegram 100 wird mit den entsprechenden Untergruppen eingefügt.

Die Verbindung zwischen den Achsen in der Hardware-Konfiguration und Ihrem Anwenderprogramm erfolgt durch Angabe folgender Moduleigenschaften in den Aufrufparametern des FB 891 - VMC InitSigma_PN:

- *Moduleeigenschaft "Parameter Access Point": Diagnoseadresse von Steckplatz 1 der Steckplatzübersicht.*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress: Angabe der Diagnoseadresse von Steckplatz 1 der Steckplatzübersicht.*
- *Moduleeigenschaft "YASKAWA Telegram PZD...": Jeweilige Startadresse des Ein-/Ausgabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "InputsStartAddress": Angabe der Startadresse des Eingabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "OutputsStartAddress": Angabe der Startadresse des Ausgabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "LogicalAddress": Angabe des kleineren Werts der Startadressen des Ein-/Ausgabe-Adressbereichs.*

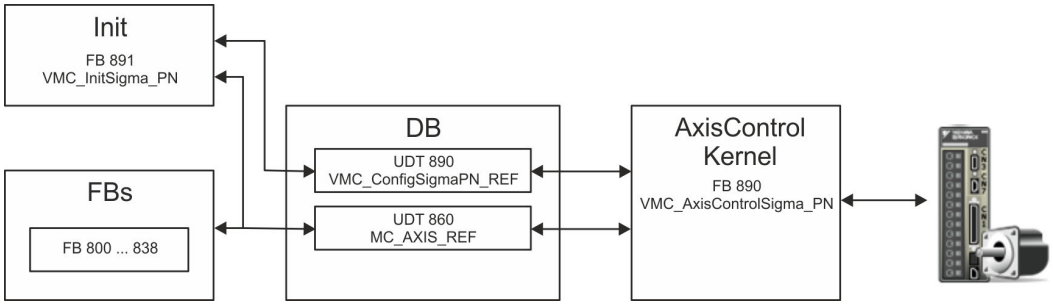
- Anwenderprogramm *654*
- FB 891 - VMC InitSigma_PN *676*

Beispiel Hardware-Konfiguration

Slot	Baugruppe	...	E-Adr.	A-Adr.	Diagnoseadresse
0	SGDV-OCB03A		2045		2045
X1	PN-IO		2039		2039
X1 P1	Port 1		2038		2038
X1 P2	Port 2		2037		2037
1	DO with YASKAWA telegr.100, PZD-16/14		2036		2036
1.1	Parameter Access Point		2036		2036
1.2	YASKAWA telegram, PZD-16/14		0-27	0-31	2036

15.3.1.3.3 Anwender-Programm

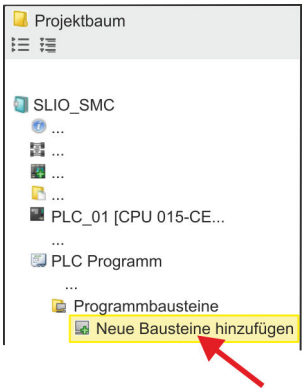
Programmstruktur



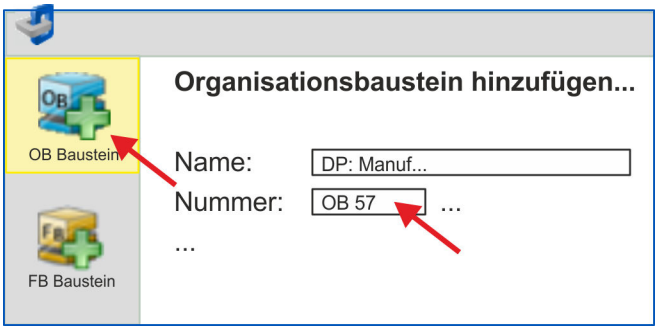
- DB
  Für jede Achse ist ein Datenbaustein (Achs-DB) für Konfiguration und Statusdaten anzulegen. Der Datenbaustein besteht aus folgenden Datenstrukturen:
  - UDT 890 - *VMC_Config*SigmaPN_REF
    Die Datenstruktur beschreibt den Aufbau der Konfiguration des Antriebs.
    Spezifische Datenstruktur für *Sigma-5/7* PROFINET.
  - UDT 860 - *MC_AXIS_REF*
    Die Datenstruktur beschreibt den Aufbau der Parameter und Statusinformationen von Antrieben.
    Allgemeine Datenstruktur für alle Antriebe und Bussysteme.
- FB 891 - *VMC_Init*Sigma_PN
  - Der *Init*-Baustein dient zur Konfiguration einer Achse.
  - Spezifischer Baustein für *Sigma-5/7* PROFINET.
  - Die Konfigurationsdaten für die Initialisierung sind im *Achs-DB* abzulegen.
- FB 890 - *VMC_AxisControl*Sigma_PN
  - Spezifischer Baustein für *Sigma-5/7* PROFINET.
  - Dieser Baustein ist eine Kombination aus einem *Kernel* und einem *AxisControl* Baustein und kommuniziert mit über PROFINET mit dem Antrieb, verarbeitet die Benutzeranforderungen und gibt Statusmeldungen zurück
  - Unterstützt einfache Bewegungskommandos und liefert alle relevanten Statusmeldungen.
  - Der Austausch der Daten erfolgt mittels des *Achs-DB*.
  - Über die Instanzdaten des Bausteins können Sie zur Bewegungssteuerung und Statusabfrage eine Visualisierung anbinden.
  - Zusätzlich zum FB 890 - *VMC_AxisControl*Sigma_PN, haben Sie die Möglichkeit *PLCopen*-Bausteine zu nutzen.
- FB 800 ... FB 838 - *PLCopen*
  - Die PLCopen-Bausteine dienen zur Programmierung von Bewegungsabläufen und Statusabfragen.
  - Allgemeine Bausteine für alle Antriebe und Bussysteme.

Programmierung

Alarm-OBs anlegen



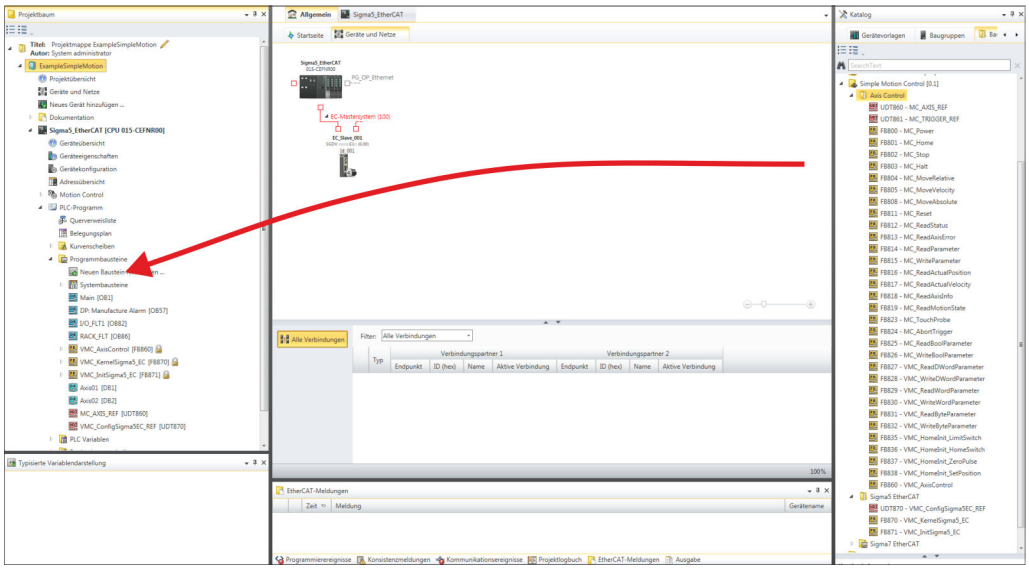
1. Klicken Sie im *Projektbaum* innerhalb der CPU unter *"PLC-Programm"*, *"Programmbausteine"* auf *"Neuen Baustein hinzufügen"*.



⇨ Das Dialogfenster *"Baustein hinzufügen"* öffnet sich.

2. Wählen Sie den Bausteintyp *"OB Baustein"* und fügen Sie nacheinander OB 57, OB 82 und OB 86 Ihrem Projekt hinzu.

Bausteine in Projekt kopieren



Öffnen Sie im *"Katalog"* unter *"Bausteine"* *"Simple Motion Control"* und ziehen Sie per Drag&Drop folgende Bausteine in *"Programmbausteine"* des *Projektbaums*:

- *Sigma* PROFINET:
  - UDT 890 - VMC_ConfigSigmaPN_REF ↳ *Kap. 15.3.3.1 "UDT 890 - VMC_ConfigSigmaPN_REF - Sigma-5/7 PROFINET Datenstruktur Achskonfiguration" Seite 672*
  - FB 890 - VMC_AxisControlSigma_PN ↳ *Kap. 15.3.3.2 "FB 890 - VMC_AxisControlSigma_PN - Control-Baustein Achskontrolle für Sigma-5/7 PROFINET" Seite 672*
  - FB 891 - VMC_InitSigma_PN ↳ *Kap. 15.3.3.3 "FB 891 - VMC_InitSigma_PN - Sigma-5/7 PROFINET Initialisierung" Seite 676*
- Axis Control
  - UDT 860 - MC_AXIS_REF ↳ *Kap. 15.8.2.1 "UDT 860 - MC_AXIS_REF - Datenstruktur Achsdaten" Seite 754*
  - FB 860 - VMC_AxisControl ↳ *Kap. 15.8.2.2 "FB 860 - VMC_AxisControl - Control-Baustein Achskontrolle" Seite 754*

Achs-DB anlegen

1. Fügen Sie Ihrem Projekt einen neuen DB als *Achs-DB* hinzu. Klicken Sie hierzu im *Projektbaum* innerhalb der CPU unter *"PLC-Programm"*, *"Programmbausteine"* auf *"Neuen Baustein hinzufügen"*, wählen Sie den Bausteintyp *"DB Baustein"* und vergeben Sie diesem den Namen "Axis01". Die DB-Nr. können Sie frei wählen wie z.B. DB 10.

   ⇨ Der Baustein wird angelegt und geöffnet.

2. - Legen Sie in "Axis01" die Variable "Config" vom Typ UDT 890 an. Dies sind spezifische Achs-Konfigurationsdaten.
   - Legen Sie in "Axis01" die Variable "Axis" vom Typ UDT 860 an. Während des Betriebs werden hier alle Betriebsdaten der Achse abgelegt.

Axis01 [DB10]

Bausteinstruktur

		Adr...	Name	Datentyp	...
		...	Config	UDT	[890]
		...	Axis	UDT	[860]

OB 1 - Konfiguration der Achsen

Öffnen Sie den OB 1 und programmieren Sie folgende FB-Aufrufe mit zugehörigen DBs:

FB 891 - VMC_InitSigma_PN, DB 891

Die Verbindung zwischen den Achsen in der Hardware-Konfiguration und Ihrem Anwenderprogramm erfolgt durch Angabe folgender Moduleigenschaften in den Aufrufparametern des FB 891 - VMC InitSigma_PN:

- *Moduleeigenschaft "Parameter Access Point": Diagnoseadresse von Steckplatz 1 der Steckplatzübersicht.*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress: Angabe der Diagnoseadresse von Steckplatz 1 der Steckplatzübersicht.*
- *Moduleeigenschaft "YASKAWA Telegram PZD...": Jeweilige Startadresse des Ein-/Ausgabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "InputsStartAddress": Angabe der Startadresse des Eingabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "OutputsStartAddress": Angabe der Startadresse des Ausgabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "LogicalAddress": Angabe des kleineren Werts der Startadressen des Ein-/Ausgabe-Adressbereichs.*

- Hardware-Konfiguration *644*
- FB 891 - VMC InitSigma_PN *676*

Beispiel Hardware-Konfiguration

Slot	Baugruppe	...	E-Adr.	A-Adr.	Diagnoseadresse
0	SGDV-OCB03A		2045		2045
X1	PN-IO		2039		2039
X1 P1	Port 1		2038		2038
X1 P2	Port 2		2037		2037
1	DO with YASKAWA telegr.100, PZD-16/14		2036		2036
1.1	Parameter Access Point		2036		2036
1.2	YASKAWA telegram, PZD-16/14		0-27	0-31	2036

Beispielaufruf

```
CALL  "VMC_InitSigma_PN" , "VMC_InitSigma_PN_1"
Enable                :="InitS5PN1_Enable"
LogicalAddress        :=0 //HW-Konfig: Smallest IO addr.
ParaAccessPointAdress :=2036 //HW-Konfig: Diag addr.
InputsStartAddress    :=0 //HW-Konfig: Telegr.100 start I addr.
OutputsStartAddress   :=0 //HW-Konfig: Telegr. 100 start O addr.
EncoderType           :=1
EncoderResolutionBits :=20
FactorPosition        :=1.048576e+006
FactorVelocity        :=1.048576e+006
FactorAcceleration    :=1.048576e+006
OffsetPosition        :=0.000000e+000
MaxVelocityApp        :=5.000000e+001
MaxAccelerationApp    :=1.000000e+002
MaxDecelerationApp    :=1.000000e+002
MaxVelocityDrive      :=6.000000e+001
MaxPosition           :=1.048500e+003
MinPosition           :=-1.048514e+003
EnableMaxPosition     :=TRUE
EnableMinPosition     :=TRUE
MinUserPosition       :="InitS5PN1_MinUserPos"
MaxUserPosition       :="InitS5PN1_MaxUserPos"
Valid                 :="InitS5PN1_Valid"
Error                 :="InitS5PN1_Error"
ErrorID               :="InitS5PN1_ErrorID"
Config                :="Axis01".Config
Axis                  :="Axis01".Axis
```

AxisControl verbinden

FB 890 - VMC_AxisControlSigma_PN, DB 890 ↳ *Kap. 15.3.3.2 "FB 890 - VMC_AxisControlSigma_PN - Control-Baustein Achskontrolle für Sigma-5/7 PROFINET" Seite 672*

Der Baustein verarbeitet die Benutzerkommandos und gibt sie entsprechend aufbereitet an den Antrieb über PROFINET weiter.

```
CALL  "VMC_AxisControlSigma_PN" , "DI_AxisControlSigmaPN01"
 AxisEnable       :="AxCtrl1_AxisEnable"
 AxisReset        :="AxCtrl1_AxisReset"
 HomeExecute      :="AxCtrl1_HomeExecute"
 HomePosition     :="AxCtrl1_HomePosition"
 StopExecute      :="AxCtrl1_StopExecute"
 MvVelocityExecute:="AxCtrl1_MvVelExecute"
 MvRelativeExecute:="AxCtrl1_MvRelExecute"
 MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
 PositionDistance :="AxCtrl1_PositionDistance"
 Direction        :="AxCtrl1_Direction"
 Velocity         :="AxCtrl1_Velocity"
```

```
Acceleration      :="AxCtrl1_Acceleration"
Deceleration      :="AxCtrl1_Deceleration"
JogPositive       :="AxCtrl1_JogPositive"
JogNegative       :="AxCtrl1_JogNegative"
JogVelocity       :="AxCtrl1_JogVelocity"
JogAcceleration   :="AxCtrl1_JogAcceleration"
JogDeceleration   :="AxCtrl1_JogDeceleration"
AxisReady         :="AxCtrl1_AxisReady"
AxisEnabled       :="AxCtrl1_AxisEnabled"
AxisError         :="AxCtrl1_AxisError"
AxisErrorID       :="AxCtrl1_AxisErrorID"
DriveWarning      :="AxCtrl1_DriveWarning"
DriveError        :="AxCtrl1_DriveError"
DriveErrorID      :="AxCtrl1_DriveErrorID"
IsHomed           :="AxCtrl1_IsHomed"
ModeOfOperation   :="AxCtrl1_ModeOfOperation"
PLCopenState      :="AxCtrl1_PLCopenState"
ActualPosition    :="AxCtrl1_ActualPosition"
ActualVelocity    :="AxCtrl1_ActualVelocity"
CmdDone           :="AxCtrl1_CmdDone"
CmdBusy           :="AxCtrl1_CmdBusy"
CmdAborted        :="AxCtrl1_CmdAborted"
CmdError          :="AxCtrl1_CmdError"
CmdErrorID        :="AxCtrl1_CmdErrorID"
DirectionPositive:="AxCtrl1_DirectionPos"
DirectionNegative:="AxCtrl1_DirectionNeg"
SWLimitMinActive  :="AxCtrl1_SWLimitMinActive"
SWLimitMaxActive  :="AxCtrl1_SWLimitMaxActive"
HWLimitMinActive  :="AxCtrl1_HWLimitMinActive"
HWLimitMaxActive  :="AxCtrl1_HWLimitMaxActive"
Axis              :="Axis01".Axis
```

Für komplexe Bewegungsaufgaben können Sie die PLCopen-Bausteine verwenden. Hier müssen Sie ebenfalls unter Axis die Referenz zu den Achsdaten im Achs-DB angeben.

Ihr Projekt beinhaltet nun folgende Bausteine:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 890 - VMC_AxisControlSigma_PN mit Instanz-DB
- FB 891 - VMC_InitSigma_PN mit Instanz-DB
- UDT 860 - MC_Axis_REF
- UDT 890 - VMC_ConfigSigmaPN_REF

Zeitlicher Ablauf

1. Wählen Sie *"Projekt ➔ Alles übersetzen"* und übertragen Sie das Projekt in Ihre CPU.

   ⇨ Sie können jetzt Ihre Applikation in Betrieb nehmen.

VORSICHT!
Bitte beachten Sie immer die Sicherheitshinweise zu ihrem Antrieb, insbesondere bei der Inbetriebnahme!

2. Bevor eine Achse gesteuert werden kann, muss diese initialisiert werden. Rufen Sie hierzu den *Init* Baustein FB 891 - VMC_InitSigma_PN mit *Enable* = TRUE auf.

⇨ Der Ausgang *Valid* meldet TRUE zurück. Im Fehlerfall können Sie durch Auswertung der *ErrorID* den Fehler ermitteln.

Den *Init*-Baustein müssen Sie erneut aufrufen, wenn Sie einen neuen Achs-DB laden oder Parameter am *Init*-Baustein geändert wurden.

Fahren Sie erst fort, wenn der Init-Baustein keinen Fehler meldet!

3. Programmieren Sie Ihre Applikation mit dem FB 890 - VMC_AxisControlSigma_PN oder mit den PLCopen Bausteinen.

15.3.2 Einsatz *Sigma-7* PROFINET

15.3.2.1 Übersicht

Voraussetzung

- SPEED7 Studio ab V1.8
- CPU mit PROFINET-Funktionalität wie z.B. CPU 015-CEFPR01
- *Sigma-7*-Antrieb mit PROFINET-Anbindung

Schritte der Projektierung

1. Parameter am Antrieb einstellen
   - Die Einstellung der Parameter hat mit dem Softwaretool *Sigma Win+* zu erfolgen.
2. Hardwarekonfiguration im VIPA *SPEED7 Studio.*
   - Projektierung einer CPU mit PROFINET-Funktionalität.
   - Projektierung eines *Sigma-7* PROFINET-Antriebs.
3. Programmierung im VIPA *SPEED7 Studio.*
   - *Init*-Baustein zur Konfiguration der Achse beschalten.
   - *Kernel*-Baustein zur Kommunikation mit der Achse beschalten.
   - Bausteine für die Bewegungsabläufe beschalten.

15.3.2.2 Parameter am Antrieb einstellen

Parameter Sigma-7

VORSICHT!
Vor der Erstinbetriebnahme müssen Sie Ihren Antrieb mit dem Softwaretool *Sigma Win+* an Ihre Applikation anpassen! Näheres hierzu finden Sie im Handbuch zu ihrem Antrieb.

Zur Abstimmung auf die *Simple Motion Control Library* sind folgende Parameter über *Sigma Win+* einzustellen:

Sigma-7S (24bit Encoder)

Servopack Parameter	Adresse	Name	Wert
PnC20	922h	Telegram Selection (100: General Telegram: All OP modes)	100

Bitte beachten Sie, dass Sie gemäß ihren Anforderungen die entsprechende Fahrtrichtung für Ihren Antrieb freigeben. Verwenden Sie hierzu die Parameter Pn50A (P-OT) bzw. Pn50B (N-OT) in Sigma Win+.

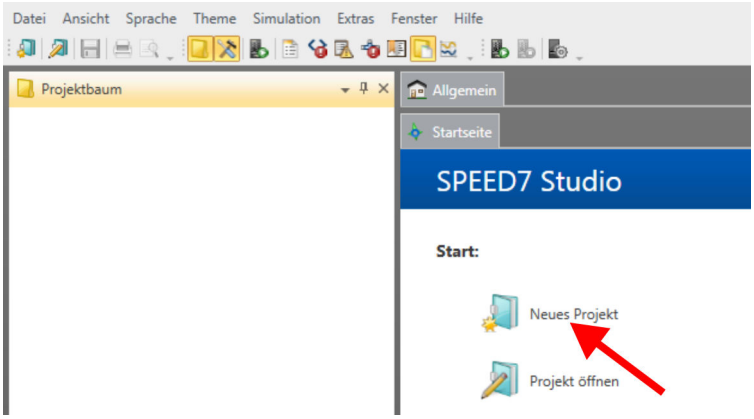
15.3.2.3 Einsatz im VIPA *SPEED7 Studio*

15.3.2.3.1 Hardware-Konfiguration System MICRO

CPU im Projekt anlegen

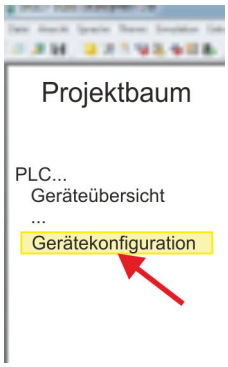
Bitte verwenden Sie für die Projektierung das *SPEED7 Studio* ab V1.8

1. Starten Sie das *SPEED7 Studio*.



2. Erstellen sie auf der Startseite mit *"Neues Projekt"* ein neues Projekt und vergeben Sie einen *"Projektnamen"*.

   ⇨ Ein neues Projekt wird angelegt und in die Sicht *"Geräte und Netze"* gewechselt.

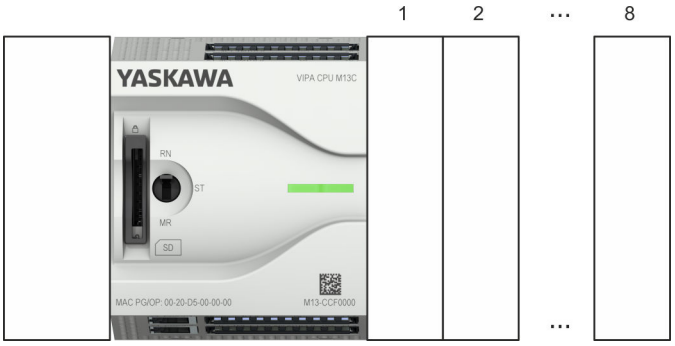


3. Klicken Sie im *Projektbaum* auf *"Neues Gerät hinzufügen ..."*.

   ⇨ Es öffnet sich ein Dialog für die Geräteauswahl.

4. Wählen Sie unter den *"Gerätevorlagen"* die System MICRO CPU M13-CCF0000 V2.4.... und klicken Sie auf [OK].

   ⇨ Die CPU wird in *"Geräte und Netze"* eingefügt und die *"Gerätekonfiguration"* geöffnet.



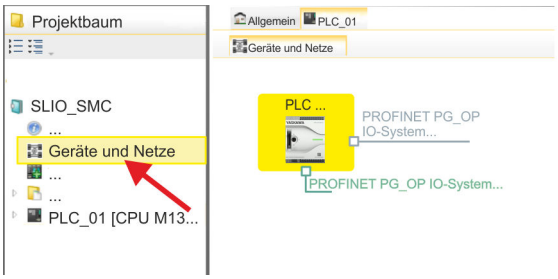
Gerätekonfiguration

Slot	Baugruppe	...	...	...	...
0	CPU M13-CCF0000				
-X2	MPI-Schnittstelle				
-X3	PROFINET PG_OP IO-System				
...	...			...	

Ethernet-PG/OP-Kanal parametrieren

1. Klicken Sie im *Projektbaum* auf *"Geräte und Netze"*.

   ⇨ Sie erhalten eine grafische Objekt-Ansicht Ihrer CPU. Hierbei werden beide Schnittstellen des PROFINET bzw. Ethernet-PG/OP Kanal Switch unter identischem Namen aufgeführt.



2. Klicken Sie auf eins der Netzwerke *"PROFINET PG_OP_Ethernet IO-System ..."*.
3. Wählen Sie *"Kontextmenü ➔ Eigenschaften der Schnittstelle"*.

   ⇨ Es öffnet sich ein Dialogfenster. Hier können Sie IP-Adressdaten für Ihren Ethernet-PG/OP-Kanal angeben. Gültige IP-Adress-Parameter erhalten Sie von Ihrem Systemadministrator.

4. Bestätigen Sie Ihre Eingabe mit [OK].

   ⇨ Die IP-Adressdaten werden in Ihr Projekt übernommen und in *"Geräte und Netze"* unter *"Lokale Baugruppen"* aufgelistet.

   Nach der Übertragung Ihres Projekts ist Ihre CPU über die angegebenen IP-Adressdaten via Ethernet-PG/OP-Kanal erreichbar.

GSDML-Datei installieren

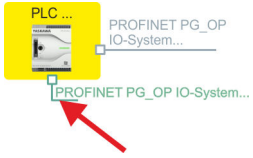
Damit der *Sigma-7* PROFINET Antrieb im *SPEED7 Studio* konfiguriert werden kann, muss die entsprechende GSDML-Datei installiert sein. In der Regel wird das *SPEED7 Studio* mit aktuellen GSDML-Dateien ausgeliefert und Sie können diesen Teil überspringen. Sollte Ihre GSDML-Datei veraltet sein, finden Sie die aktuellste GSDML-Datei für den *Sigma-7* PROFINET Antrieb unter *www.yaskawa.eu.com* unter *"Service ➔ Drives & Motion Software"*.

1. Laden Sie die zu Ihrem Antrieb passende GSDML-Datei herunter. Entpacken Sie diese falls erforderlich.
2. Gehen Sie in Ihr *SPEED7 Studio*.
3. Öffnen Sie mit *"Extras ➔ Gerätebeschreibungsdatei installieren (PROFINET - GSDML)"* das zugehörige Dialogfenster.
4. Geben Sie unter *"Quellpfad"* die GSDML-Datei an und installieren Sie diese mit [Installieren].

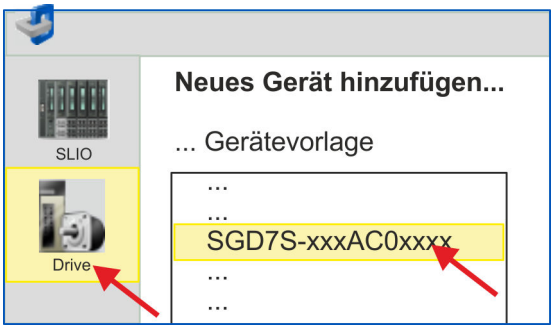
   ⇨ Die Geräte der GSDML-Datei steht Ihnen nun zur Verfügung.

***Sigma-7* Antrieb hinzufügen**

1. Klicken Sie im Projektbaum auf *"Geräte und Netze"*.
2. Klicken Sie auf *"PROFINET PG_OP_Ethernet IO-System ..."* und wählen sie *"Kontextmenü ➔ Neues Gerät hinzufügen"*.



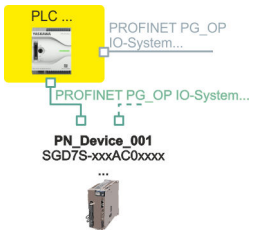
   ⇨ Es öffnet sich die Gerätevorlage zur Auswahl eines PROFINET-Devices.



3. Wählen Sie Ihren *Sigma-7* Antrieb aus:

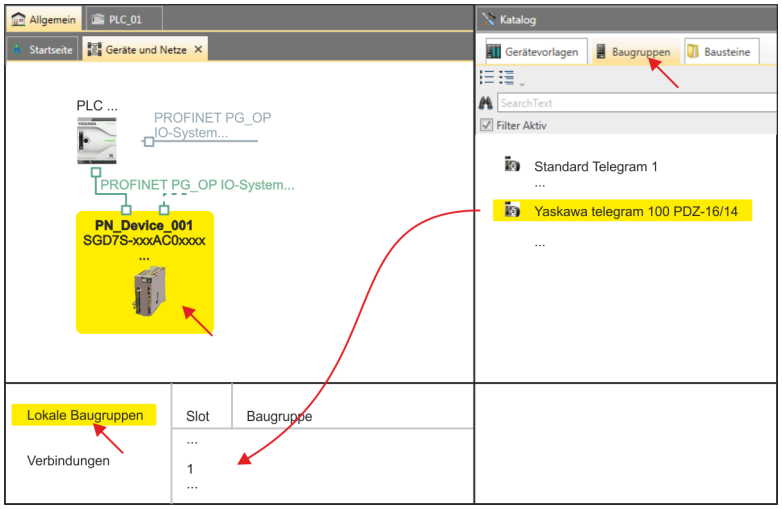
- SGD7S-xxxAC0xxxx

Bestätigen Sie Ihre Angaben mit [OK]. Sollte Ihr Antrieb nicht vorhanden sein, müssen Sie die entsprechende GSDML-Datei wie weiter oben beschrieben installieren.



⇨ Der *Sigma-7* Antrieb wird an Ihren PROFINET-IO-Controller angebunden.

4. Klicken Sie auf den *Sigma-7* Antrieb.



5. Wählen Sie unter *"Katalog"* den Reiter *"Baugruppen"* an.

⇨ Die Telegramme für den *Sigma-7* Antrieb werden aufgelistet.

6. Wählen Sie *"Yaskawa telegram 100 PZD..."* und ziehen Sie dieses unter *"Lokale Baugruppen"* auf *"Slot 1"*.

⇨ Telegram 100 wird mit den entsprechenden Untergruppen eingefügt.

Die Verbindung zwischen den Achsen in der Hardware-Konfiguration und Ihrem Anwenderprogramm erfolgt durch Angabe folgender Moduleigenschaften in den Aufrufparametern des FB 891 - VMC InitSigma_PN:

- *Moduleeigenschaft "Parameter Access Point": Diagnoseadresse von Steckplatz 1 der Steckplatzübersicht.*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress: Angabe der Diagnoseadresse von Steckplatz 1 der Steckplatzübersicht.*
- *Moduleeigenschaft "YASKAWA Telegram PZD...": Jeweilige Startadresse des Ein-/Ausgabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "InputsStartAddress": Angabe der Startadresse des Eingabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "OutputsStartAddress": Angabe der Startadresse des Ausgabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "LogicalAddress": Angabe des kleineren Werts der Startadressen des Ein-/Ausgabe-Adressbereichs.*

- Anwenderprogramm *669*
- FB 891 - VMC InitSigma_PN *676*

Beispiel Hardware-Konfiguration

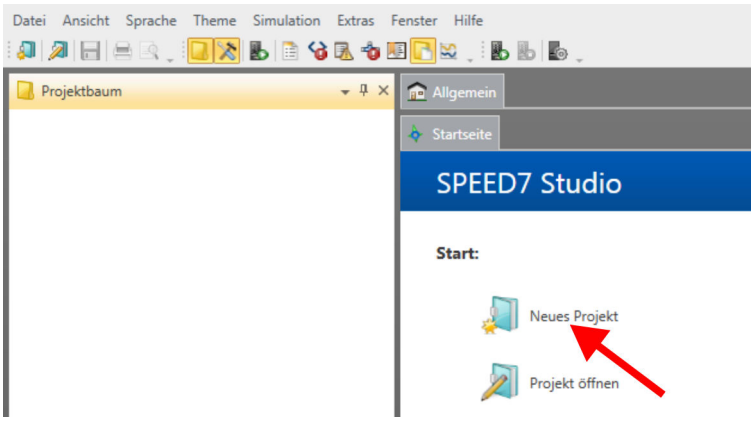
Slot	Baugruppe	...	E-Adr.	A-Adr.	Diagnoseadresse
0	SGD7S-xxxAC0xxxx		2035		2035
X1	PN-IO		2034		2034
X1 P1	Port 1		2033		2033
X1 P2	Port 2		2032		2032
1	DO with YASKAWA telegr.100, PZD-16/14		2044		2044
1.1	Parameter Access Point		2044		2044
1.2	YASKAWA telegram, PZD-16/14		28-55	32-63	2044

15.3.2.3.2 Hardware-Konfiguration System SLIO

CPU im Projekt anlegen

Bitte verwenden Sie für die Projektierung das *SPEED7 Studio* ab V1.8

1. Starten Sie das *SPEED7 Studio*.

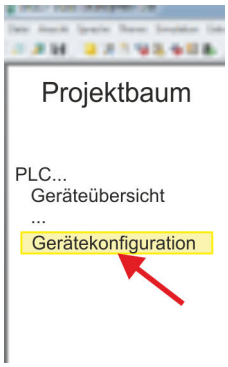


2. Erstellen sie auf der Startseite mit *"Neues Projekt"* ein neues Projekt und vergeben Sie einen *"Projektnamen"*.

⇨ Ein neues Projekt wird angelegt und in die Sicht *"Geräte und Netze"* gewechselt.

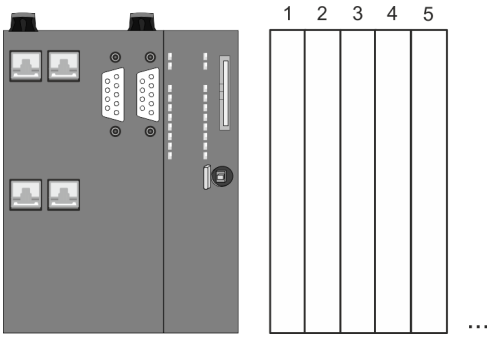
3. Klicken Sie im *Projektbaum* auf *"Neues Gerät hinzufügen ..."*.

⇨ Es öffnet sich ein Dialog für die Geräteauswahl.



4. Wählen Sie unter den *"Gerätevorlagen"* Ihre PROFINET-CPU wie z.B. die CPU 015-CEFPR01 und klicken Sie auf [OK].

⇨ Die CPU wird in *"Geräte und Netze"* eingefügt und die *"Gerätekonfiguration"* geöffnet.

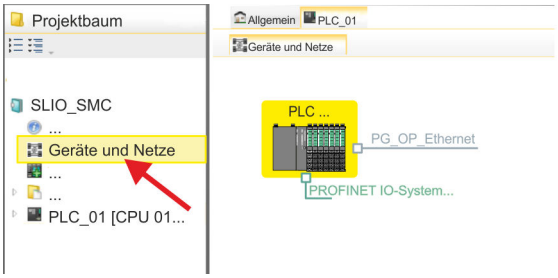


Gerätekonfiguration

Slot	Baugruppe	...	...	...	...
0	CPU 015-CEFPR01				
-X1	PG_OP_Ethernet				
-X3	MPI-Schnittstelle				
-X4	PROFINET-IO-System			...	
...					

Ethernet-PG/OP-Kanal parametrieren

1. Klicken Sie im *Projektbaum* auf *"Geräte und Netze"*.
   ⇨ Sie erhalten eine grafische Objekt-Ansicht Ihrer CPU.



2. Klicken Sie auf das Netzwerk *"PG_OP_Ethernet"*.
3. Wählen Sie *"Kontextmenü ➔ Eigenschaften der Schnittstelle"*.
   ⇨ Es öffnet sich ein Dialogfenster. Hier können Sie IP-Adressdaten für Ihren Ethernet-PG/OP-Kanal angeben. Gültige IP-Adress-Parameter erhalten Sie von Ihrem Systemadministrator.
4. Bestätigen Sie Ihre Eingabe mit [OK].
   ⇨ Die IP-Adressdaten werden in Ihr Projekt übernommen und in *"Geräte und Netze"* unter *"Lokale Baugruppen"* aufgelistet.

   Nach der Übertragung Ihres Projekts ist Ihre CPU über die angegebenen IP-Adressdaten via Ethernet-PG/OP-Kanal erreichbar.

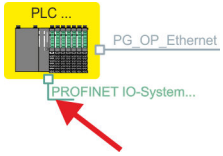
GSDML-Datei installieren

Damit der *Sigma-7* PROFINET Antrieb im *SPEED7 Studio* konfiguriert werden kann, muss die entsprechende GSDML-Datei installiert sein. In der Regel wird das *SPEED7 Studio* mit aktuellen GSDML-Dateien ausgeliefert und Sie können diesen Teil überspringen. Sollte Ihre GSDML-Datei veraltet sein, finden Sie die aktuellste GSDML-Datei für den *Sigma-7* PROFINET Antrieb unter *www.yaskawa.eu.com* unter *"Service ➔ Drives & Motion Software"*.

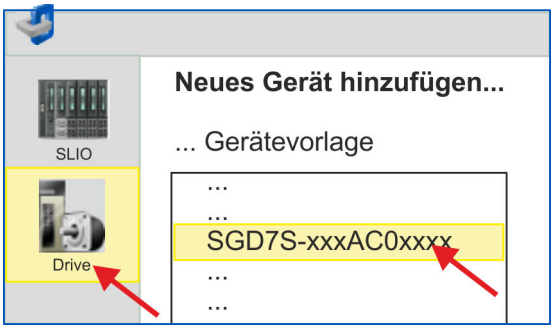
1. Laden Sie die zu Ihrem Antrieb passende GSDML-Datei herunter. Entpacken Sie diese falls erforderlich.
2. Gehen Sie in Ihr *SPEED7 Studio*.
3. Öffnen Sie mit *"Extras ➔ Gerätebeschreibungsdatei installieren (PROFINET - GSDML)"* das zugehörige Dialogfenster.
4. Geben Sie unter *"Quellpfad"* die GSDML-Datei an und installieren Sie diese mit [Installieren].
   ⇨ Die Geräte der GSDML-Datei steht Ihnen nun zur Verfügung.

***Sigma-7* Antrieb hinzufügen**

1. Klicken Sie im Projektbaum auf *"Geräte und Netze"*.
2. Klicken Sie auf *"PROFINET IO-System ..."* und wählen sie *"Kontextmenü ➔ Neues Gerät hinzufügen"*.



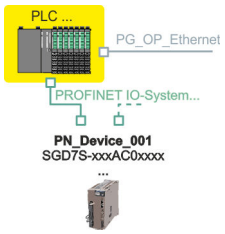
   ⇨ Es öffnet sich die Gerätevorlage zur Auswahl eines PROFINET-Devices.



3. Wählen Sie Ihren *Sigma-7* Antrieb aus:

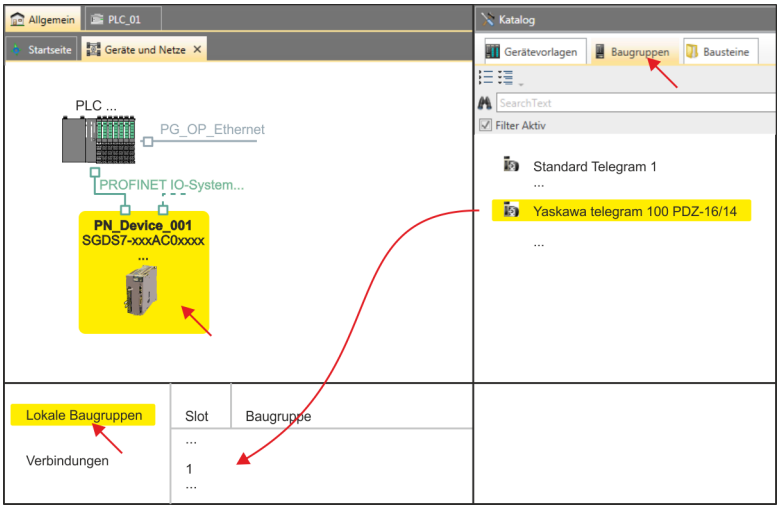
- SGD7S-xxxACxxxx

Bestätigen Sie Ihre Angaben mit [OK]. Sollte Ihr Antrieb nicht vorhanden sein, müssen Sie die entsprechende GSDML-Datei wie weiter oben beschrieben installieren.



⇨ Der *Sigma-7* Antrieb wird an Ihren PROFINET-IO-Controller angebunden.

4. Klicken Sie auf den *Sigma-7* Antrieb



5. Wählen Sie unter *"Katalog"* den Reiter *"Baugruppen"* an.

⇨ Die Telegramme für den *Sigma-7* Antrieb werden aufgelistet.

6. Wählen Sie *"Yaskawa telegram 100 PZD..."* und ziehen Sie dieses unter *"Lokale Baugruppen"* auf *"Slot 1"*.

⇨ Telegram 100 wird mit den entsprechenden Untergruppen eingefügt.

Die Verbindung zwischen den Achsen in der Hardware-Konfiguration und Ihrem Anwenderprogramm erfolgt durch Angabe folgender Moduleigenschaften in den Aufrufparametern des FB 891 - VMC InitSigma_PN:

- *Moduleeigenschaft "Parameter Access Point": Diagnoseadresse von Steckplatz 1 der Steckplatzübersicht.*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress: Angabe der Diagnoseadresse von Steckplatz 1 der Steckplatzübersicht.*
- *Moduleeigenschaft "YASKAWA Telegram PZD...": Jeweilige Startadresse des Ein-/Ausgabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "InputsStartAddress": Angabe der Startadresse des Eingabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "OutputsStartAddress": Angabe der Startadresse des Ausgabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "LogicalAddress": Angabe des kleineren Werts der Startadressen des Ein-/Ausgabe-Adressbereichs.*

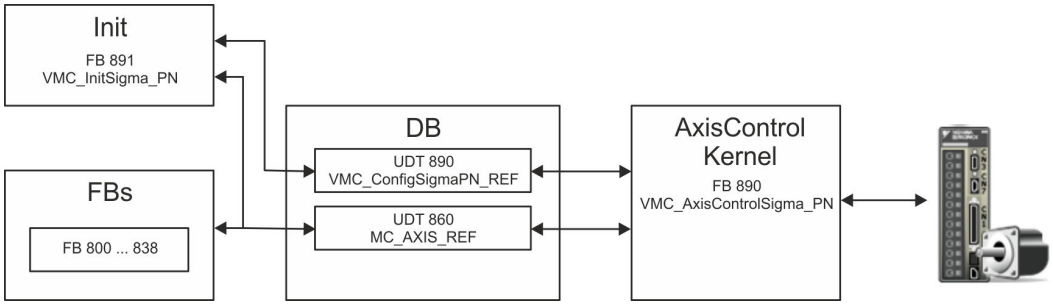
- Anwenderprogramm *669*
- FB 891 - VMC InitSigma_PN *676*

Beispiel Hardware-Konfiguration

Slot	Baugruppe	...	E-Adr.	A-Adr.	Diagnoseadresse
0	SGD7S-xxxAC0xxxx		2035		2035
X1	PN-IO		2034		2034
X1 P1	Port 1		2033		2033
X1 P2	Port 2		2032		2032
1	DO with YASKAWA telegr.100, PZD-16/14		2044		2044
1.1	Parameter Access Point		2044		2044
1.2	YASKAWA telegram, PZD-16/14		28-55	32-63	2044

15.3.2.3.3 Anwender-Programm

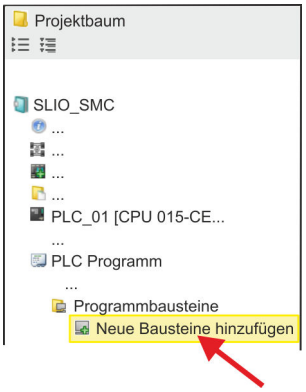
Programmstruktur



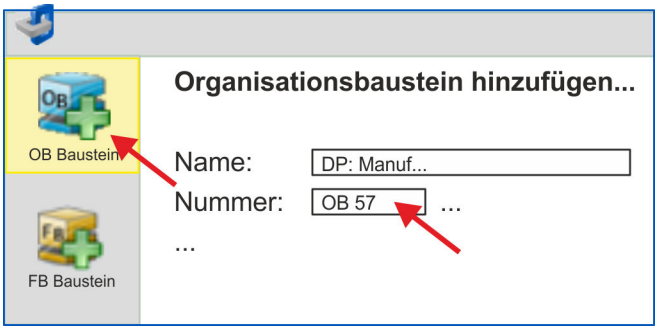
- DB
  Für jede Achse ist ein Datenbaustein (Achs-DB) für Konfiguration und Statusdaten anzulegen. Der Datenbaustein besteht aus folgenden Datenstrukturen:
  - UDT 890 - *VMC_Config*SigmaPN_REF
    Die Datenstruktur beschreibt den Aufbau der Konfiguration des Antriebs.
    Spezifische Datenstruktur für *Sigma-5/7* PROFINET.
  - UDT 860 - *MC_AXIS_REF*
    Die Datenstruktur beschreibt den Aufbau der Parameter und Statusinformationen von Antrieben.
    Allgemeine Datenstruktur für alle Antriebe und Bussysteme.
- FB 891 - *VMC_Init*Sigma_PN
  - Der *Init*-Baustein dient zur Konfiguration einer Achse.
  - Spezifischer Baustein für *Sigma-5/7* PROFINET.
  - Die Konfigurationsdaten für die Initialisierung sind im *Achs-DB* abzulegen.
- FB 890 - *VMC_AxisControl*Sigma_PN
  - Spezifischer Baustein für *Sigma-5/7* PROFINET.
  - Dieser Baustein ist eine Kombination aus einem *Kernel* und einem *AxisControl* Baustein und kommuniziert mit über PROFINET mit dem Antrieb, verarbeitet die Benutzeranforderungen und gibt Statusmeldungen zurück
  - Unterstützt einfache Bewegungskommandos und liefert alle relevanten Statusmeldungen.
  - Der Austausch der Daten erfolgt mittels des *Achs-DB*.
  - Über die Instanzdaten des Bausteins können Sie zur Bewegungssteuerung und Statusabfrage eine Visualisierung anbinden.
  - Zusätzlich zum FB 890 - *VMC_AxisControl*Sigma_PN, haben Sie die Möglichkeit *PLCopen*-Bausteine zu nutzen.
- FB 800 ... FB 838 - *PLCopen*
  - Die PLCopen-Bausteine dienen zur Programmierung von Bewegungsabläufen und Statusabfragen.
  - Allgemeine Bausteine für alle Antriebe und Bussysteme.

Programmierung

Alarm-OBs anlegen



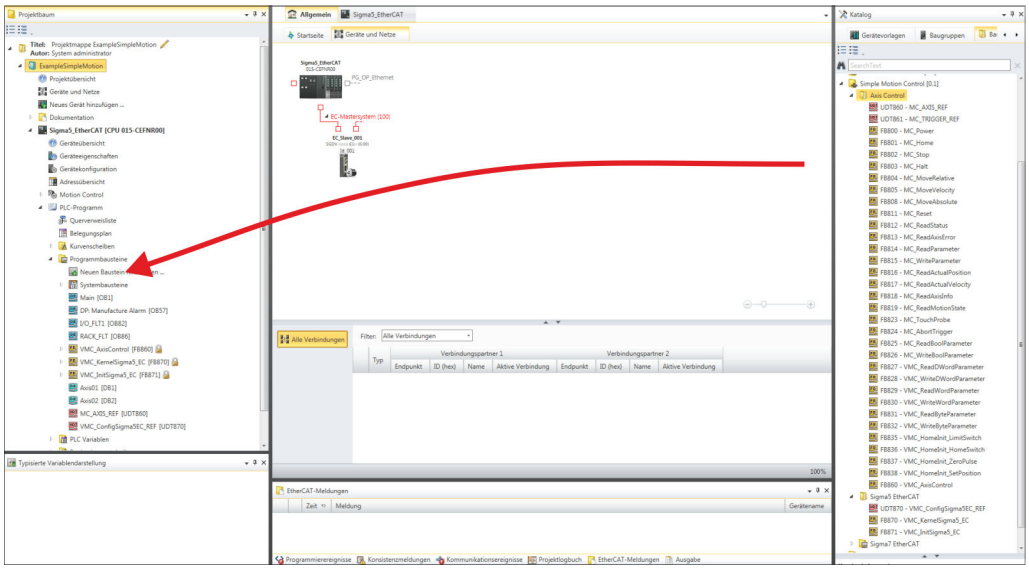
1. Klicken Sie im *Projektbaum* innerhalb der CPU unter *"PLC-Programm"*, *"Programmbausteine"* auf *"Neuen Baustein hinzufügen"*.



⇨ Das Dialogfenster *"Baustein hinzufügen"* öffnet sich.

2. Wählen Sie den Bausteintyp *"OB Baustein"* und fügen Sie nacheinander OB 57, OB 82 und OB 86 Ihrem Projekt hinzu.

Bausteine in Projekt kopieren



Öffnen Sie im *"Katalog"* unter *"Bausteine"* *"Simple Motion Control"* und ziehen Sie per Drag&Drop folgende Bausteine in *"Programmbausteine"* des *Projektbaums*:

- *Sigma* PROFINET:
  - UDT 890 - VMC_ConfigSigmaPN_REF ↳ *Kap. 15.3.3.1 "UDT 890 - VMC_ConfigSigmaPN_REF - Sigma-5/7 PROFINET Datenstruktur Achskonfiguration" Seite 672*
  - FB 890 - VMC_AxisControlSigma_PN ↳ *Kap. 15.3.3.2 "FB 890 - VMC_AxisControlSigma_PN - Control-Baustein Achskontrolle für Sigma-5/7 PROFINET" Seite 672*
  - FB 891 - VMC_InitSigma_PN ↳ *Kap. 15.3.3.3 "FB 891 - VMC_InitSigma_PN - Sigma-5/7 PROFINET Initialisierung" Seite 676*
- Axis Control
  - UDT 860 - MC_AXIS_REF ↳ *Kap. 15.8.2.1 "UDT 860 - MC_AXIS_REF - Datenstruktur Achsdaten" Seite 754*
  - FB 860 - VMC_AxisControl ↳ *Kap. 15.8.2.2 "FB 860 - VMC_AxisControl - Control-Baustein Achskontrolle" Seite 754*

Achs-DB anlegen

1. Fügen Sie Ihrem Projekt einen neuen DB als *Achs-DB* hinzu. Klicken Sie hierzu im *Projektbaum* innerhalb der CPU unter *"PLC-Programm"*, *"Programmbausteine"* auf *"Neuen Baustein hinzufügen"*, wählen Sie den Bausteintyp *"DB Baustein"* und vergeben Sie diesem den Namen "Axis01". Die DB-Nr. können Sie frei wählen wie z.B. DB 10.

   ⇨ Der Baustein wird angelegt und geöffnet.

2. - Legen Sie in "Axis01" die Variable "Config" vom Typ UDT 890 an. Dies sind spezifische Achs-Konfigurationsdaten.
   - Legen Sie in "Axis01" die Variable "Axis" vom Typ UDT 860 an. Während des Betriebs werden hier alle Betriebsdaten der Achse abgelegt.

Axis01 [DB10]

Bausteinstruktur

		Adr...	Name	Datentyp	...
		...	Config	UDT	[890]
		...	Axis	UDT	[860]

OB 1 - Konfiguration der Achsen

Öffnen Sie den OB 1 und programmieren Sie folgende FB-Aufrufe mit zugehörigen DBs:

FB 891 - VMC_InitSigma_PN, DB 891

Die Verbindung zwischen den Achsen in der Hardware-Konfiguration und Ihrem Anwenderprogramm erfolgt durch Angabe folgender Moduleigenschaften in den Aufrufparametern des FB 891 - VMC InitSigma_PN:

- *Moduleeigenschaft "Parameter Access Point": Diagnoseadresse von Steckplatz 1 der Steckplatzübersicht.*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress: Angabe der Diagnoseadresse von Steckplatz 1 der Steckplatzübersicht.*
- *Moduleeigenschaft "YASKAWA Telegram PZD...": Jeweilige Startadresse des Ein-/Ausgabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "InputsStartAddress": Angabe der Startadresse des Eingabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "OutputsStartAddress": Angabe der Startadresse des Ausgabe-Adressbereichs.*
  - *FB 891 - VMC InitSigma_PN: "LogicalAddress": Angabe des kleineren Werts der Startadressen des Ein-/Ausgabe-Adressbereichs.*

- Hardware-Konfiguration *659*
- FB 891 - VMC InitSigma_PN *676*

Beispiel Hardware-Konfiguration

Slot	Baugruppe	...	E-Adr.	A-Adr.	Diagnoseadresse
0	SGD7S-xxxAC0xxxx		2035		2035
X1	PN-IO		2034		2034
X1 P1	Port 1		2033		2033
X1 P2	Port 2		2032		2032
1	DO with YASKAWA telegr.100, PZD-16/14		2044		2044
1.1	Parameter Access Point		2044		2044
1.2	YASKAWA telegram, PZD-16/14		28-55	32-63	2044

Beispielaufruf

```
CALL  "VMC_InitSigma_PN" , "VMC_InitSigma_PN_1"
Enable                :="InitS7PN1_Enable"
LogicalAddress        :28 //HW-Konfig: Smallest IO addr.
ParaAccessPointAdress :=2044 //HW-Konfig: Diag addr.
InputsStartAddress    :=28 //HW-Konfig: Telegr.100 start I addr.
OutputsStartAddress   :=32 //HW-Konfig: Telegr. 100 start O addr.
EncoderType           :=1
EncoderResolutionBits :=20
FactorPosition        :=1.048576e+006
FactorVelocity        :=1.048576e+006
FactorAcceleration    :=1.048576e+006
OffsetPosition        :=0.000000e+000
MaxVelocityApp        :=5.000000e+001
MaxAccelerationApp    :=1.000000e+002
MaxDecelerationApp    :=1.000000e+002
MaxVelocityDrive      :=6.000000e+001
MaxPosition           :=1.048500e+003
MinPosition           :=-1.048514e+003
EnableMaxPosition     :=TRUE
EnableMinPosition     :=TRUE
MinUserPosition       :="InitS7PN1_MinUserPos"
MaxUserPosition       :="InitS7PN1_MaxUserPos"
Valid                 :="InitS7PN1_Valid"
Error                 :="InitS7PN1_Error"
ErrorID               :="InitS7PN1_ErrorID"
Config                :="Axis01".Config
Axis                  :="Axis01".Axis
```

AxisControl verbinden

FB 890 - VMC_AxisControlSigma_PN, DB 890 ↳ *Kap. 15.3.3.2 "FB 890 - VMC_AxisControlSigma_PN - Control-Baustein Achskontrolle für Sigma-5/7 PROFINET" Seite 672*

Der Baustein verarbeitet die Benutzerkommandos und gibt sie entsprechend aufbereitet an den Antrieb über PROFINET weiter.

```
CALL  "VMC_AxisControlSigma_PN" , "DI_AxisControlSigmaPN01"
 AxisEnable       :="AxCtrl1_AxisEnable"
 AxisReset        :="AxCtrl1_AxisReset"
 HomeExecute      :="AxCtrl1_HomeExecute"
 HomePosition     :="AxCtrl1_HomePosition"
 StopExecute      :="AxCtrl1_StopExecute"
 MvVelocityExecute:="AxCtrl1_MvVelExecute"
 MvRelativeExecute:="AxCtrl1_MvRelExecute"
 MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
 PositionDistance :="AxCtrl1_PositionDistance"
 Direction        :="AxCtrl1_Direction"
 Velocity         :="AxCtrl1_Velocity"
```

```
Acceleration      :="AxCtrl1_Acceleration"
Deceleration      :="AxCtrl1_Deceleration"
JogPositive       :="AxCtrl1_JogPositive"
JogNegative       :="AxCtrl1_JogNegative"
JogVelocity       :="AxCtrl1_JogVelocity"
JogAcceleration   :="AxCtrl1_JogAcceleration"
JogDeceleration   :="AxCtrl1_JogDeceleration"
AxisReady         :="AxCtrl1_AxisReady"
AxisEnabled       :="AxCtrl1_AxisEnabled"
AxisError         :="AxCtrl1_AxisError"
AxisErrorID       :="AxCtrl1_AxisErrorID"
DriveWarning      :="AxCtrl1_DriveWarning"
DriveError        :="AxCtrl1_DriveError"
DriveErrorID      :="AxCtrl1_DriveErrorID"
IsHomed           :="AxCtrl1_IsHomed"
ModeOfOperation   :="AxCtrl1_ModeOfOperation"
PLCopenState      :="AxCtrl1_PLCopenState"
ActualPosition    :="AxCtrl1_ActualPosition"
ActualVelocity    :="AxCtrl1_ActualVelocity"
CmdDone           :="AxCtrl1_CmdDone"
CmdBusy           :="AxCtrl1_CmdBusy"
CmdAborted        :="AxCtrl1_CmdAborted"
CmdError          :="AxCtrl1_CmdError"
CmdErrorID        :="AxCtrl1_CmdErrorID"
DirectionPositive:="AxCtrl1_DirectionPos"
DirectionNegative:="AxCtrl1_DirectionNeg"
SWLimitMinActive  :="AxCtrl1_SWLimitMinActive"
SWLimitMaxActive  :="AxCtrl1_SWLimitMaxActive"
HWLimitMinActive  :="AxCtrl1_HWLimitMinActive"
HWLimitMaxActive  :="AxCtrl1_HWLimitMaxActive"
Axis              :="Axis01".Axis
```

Für komplexe Bewegungsaufgaben können Sie die PLCopen-Bausteine verwenden. Hier müssen Sie ebenfalls unter Axis die Referenz zu den Achsdaten im Achs-DB angeben.

Ihr Projekt beinhaltet nun folgende Bausteine:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 890 - VMC_AxisControlSigma_PN mit Instanz-DB
- FB 891 - VMC_InitSigma_PN mit Instanz-DB
- UDT 860 - MC_Axis_REF
- UDT 890 - VMC_ConfigSigmaPN_REF

Zeitlicher Ablauf

1. Wählen Sie *"Projekt ➔ Alles übersetzen"* und übertragen Sie das Projekt in Ihre CPU.

   ⇨ Sie können jetzt Ihre Applikation in Betrieb nehmen.

VORSICHT!
Bitte beachten Sie immer die Sicherheitshinweise zu ihrem Antrieb, insbesondere bei der Inbetriebnahme!

2. Bevor eine Achse gesteuert werden kann, muss diese initialisiert werden. Rufen Sie hierzu den *Init* Baustein FB 891 - VMC_InitSigma_PN mit *Enable* = TRUE auf.

   ⇨ Der Ausgang *Valid* meldet TRUE zurück. Im Fehlerfall können Sie durch Auswertung der *ErrorID* den Fehler ermitteln.

   Den *Init*-Baustein müssen Sie erneut aufrufen, wenn Sie einen neuen Achs-DB laden oder Parameter am *Init*-Baustein geändert wurden.

   Fahren Sie erst fort, wenn der Init-Baustein keinen Fehler meldet!

3. Programmieren Sie Ihre Applikation mit dem FB 890 - VMC_AxisControlSigma_PN oder mit den PLCopen Bausteinen.

15.3.3 Antriebsspezifische Bausteine

Die PLCopen-Bausteine zur Achskontrolle finden Sie hier: Kap. 15.8 "Bausteine zur Achskontrolle" Seite 751

15.3.3.1 UDT 890 - VMC_ConfigSigmaPN_REF - *Sigma-5/7* PROFINET Datenstruktur Achskonfiguration

Dies ist eine benutzerdefinierte Datenstruktur, die Informationen zu den Konfigurationsdaten beinhaltet. Die UDT ist speziell angepasst an die Verwendung eines *Sigma-5/7*-Antriebs, welcher über PROFINET angebunden ist.

15.3.3.2 FB 890 - VMC_AxisControlSigma_PN - Control-Baustein Achskontrolle für Sigma-5/7 PROFINET

Beschreibung

Der FB *VMC_AxisControlSigma_PN* ist eine Kombination aus einem *Kernel* für Sigma-5/7 Achsen für PROFINET und einem *Axis_Control* Baustein zum Ansteuern der Motion Control Funktionen. Mit dem FB *VMC_AxisControlSigma_PN* können Sie die angebundene Achse steuern. Sie können den Status des Antriebs abrufen, den Antrieb ein- bzw. ausschalten oder verschiedene Bewegungskommandos ausführen.

Der Baustein VMC_AxisControlSigma_PN sollte nie gleichzeitig mit dem PLCopen-Baustein MC_Power verwendet werden. Da der VMC_AxisControlSigma_PN Funktionalitäten des MC_Power beinhaltet und immer der aktuellste Befehl vom Kerneln ausgeführt wird, kann dies zu einem Fehlverhalten des Antriebs führen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
AxisEnable	INPUT	BOOL	■ Achsenfreigabe – TRUE: Die Achse wird freigegeben. – FALSE: Die Achse wird gesperrt.
AxisReset	INPUT	BOOL	■ Reset Achse – Flanke 0-1: Reset der Achse wird durchgeführt.

Parameter	Deklaration	Datentyp	Beschreibung
HomeExecute	INPUT	BOOL	■ Referenzfahrt – Flanke 0-1: Referenzfahrt wird gestartet.
HomePosition	INPUT	REAL	Bei erfolgreicher Referenzierung wird die Istposition der Achse einmalig gleich Position gesetzt. Die Position ist in der verwendeten Anwendereinheit anzugeben.
StopExecute	INPUT	BOOL	■ Achse stoppen – Flanke 0-1: Stoppen der Achse wird gestartet.
MvVelocityExecute	INPUT	BOOL	■ Verfahren der Achse starten – Flanke 0-1: Die Achse wird auf die angegebene Geschwindigkeit beschleunigt / abgebremst.
MvRelativeExecute	INPUT	BOOL	■ Verfahren der Achse starten – Flanke 0-1: Die relative Positionierung der Achse wird gestartet.
MvAbsoluteExecute	INPUT	BOOL	■ Verfahren der Achse starten – Flanke 0-1: Die absolute Positionierung der Achse wird gestartet.
Direction*	INPUT	BYTE	Modus für absolute Positionierung: ■ 0: kürzester Weg ■ 1: positive Richtung ■ 2: negative Richtung ■ 3: aktuelle Richtung
PositionDistance	INPUT	REAL	Absolute Position bzw. relative Wegstrecke je nach Kommando in [Anwendereinheiten].
Velocity	INPUT	REAL	Geschwindigkeitsvorgabe (vorzeichenbehafteter Wert) in [Anwendereinheiten/s].
Acceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2].
Deceleration	INPUT	REAL	Verzögerung in [Anwendereinheiten/s^2].
JogPositive	INPUT	BOOL	■ Achse mit konstanter Geschwindigkeit in positive Richtung verfahren – Flanke 0-1: Das Verfahren der Achse mit konstanter Geschwindigkeit wird gestartet. – Flanke 1-0: Die Achse wird gestoppt.
JogNegative	INPUT	BOOL	■ Achse mit konstanter Geschwindigkeit in negative Richtung verfahren – Flanke 0-1: Das Verfahren der Achse mit konstanter Geschwindigkeit wird gestartet. – Flanke 1-0: Die Achse wird gestoppt.
JogVelocity	INPUT	REAL	Geschwindigkeitsvorgabe für Jogging (positiver Wert) in [Anwendereinheiten/s].
JogAcceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2].
JogDeceleration	INPUT	REAL	Verzögerung für Jogging in [Anwendereinheiten/s^2].
KernelInitReset	INPUT	BOOL	Rücksetzen der *Kernel*-Funktionen. Achtung, laufende Kommandos werden abgebrochen!

Parameter	Deklaration	Datentyp	Beschreibung
AxisReady	OUTPUT	BOOL	■ AxisReady – TRUE: Die Achse ist einschaltbereit. – FALSE: Die Achse ist nicht einschaltbereit. → Prüfe und behebe *AxisError* (siehe AxisErrorID). → Prüfe und behebe *DriveError* (siehe DriveErrorID). → Prüfe Initialisierungs FB (Input- und Output Addressen bzw. Diagnoseadresse richtig?)
AxisEnabled	OUTPUT	BOOL	■ Status Achse – TRUE: Achse ist eingeschaltet und nimmt Bewegungsaufträge an. – FALSE: Achse ist nicht eingeschaltet und nimmt keine Bewegungsaufträge an.
AxisError	OUTPUT	BOOL	■ Fehler bei Motion Achse – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *AxisErrorID* entnommen werden. → Die Achse wird gesperrt.
AxisErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
DriveWarning	OUTPUT	BOOL	■ Warnung – TRUE: Der Antrieb liefert eine Warnung. Zusätzliche Informationen sind aus dem entsprechenden Handbuch des Herstellers zu entnehmen.
DriveError	OUTPUT	BOOL	■ Fehler direkt am Antrieb – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *DriveErrorID* entnommen werden. → Die Achse wird gesperrt.
DriveErrorID	OUTPUT	WORD	■ Fehler – TRUE: Der Antrieb liefert einen Fehler. Zusätzliche Informationen sind aus dem entsprechenden Handbuch des Herstellers zu entnehmen.
IsHomed	OUTPUT	BOOL	■ Information Achse: referenziert – TRUE: Die Achse ist referenziert.

Parameter	Deklaration	Datentyp	Beschreibung
ModeOfOperation	OUTPUT	INT	Antriebsspezifischer Modus. Weitere Infos siehe Antriebsmanual. Beispiel *Sigma-5*: 0: No mode changed/no mode assigned 1: Profile Position mode 2: Reserved (keep last mode) 3: Profile Velocity mode 4: Torque Profile mode 6: Homing mode 7: Interpolated Position mode 8: Cyclic Sync Position mode 9: Cyclic Sync Velocity mode 10: Cyclic Sync Torque mode Other Reserved (keep last mode)
PLCopenState	OUTPUT	INT	Aktueller PLCopenState: 1: Disabled 2: Standstill 3: Homing 4: Discrete Motion 5: Continous Motion 7: Stopping 8: Errorstop
ActualPosition	OUTPUT	REAL	Position der Achse in [Anwendereinheit].
ActualVelocity	OUTPUT	REAL	Geschwindigkeit der Achse in [Anwendereinheit/s].
CmdDone	OUTPUT	BOOL	■ Status – TRUE: Auftrag wurde ohne Fehler beendet.
CmdBusy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung.
CmdAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen.
CmdError	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *CmdErrorID* entnommen werden.
CmdErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
DirectionPositive	OUTPUT	BOOL	■ Zustand Bewegungsauftrag: Position zunehmend – TRUE: Die Position der Achse nimmt zu.

Parameter	Deklaration	Datentyp	Beschreibung
DirectionNegative	OUTPUT	BOOL	■ Zustand Bewegungsauftrag: Position abnehmend – TRUE: Die Position der Achse nimmt ab.
SWLimitMinActive	OUTPUT	BOOL	■ Software Endschalter – TRUE: Software Endschalter Minimum aktiv (Minimale Position in negative Richtung überschritten).
SWLimitMaxActive	OUTPUT	BOOL	■ Software Endschalter – TRUE: Software Endschalter Maximum aktiv (Maximale Position in positive Richtung überschritten).
HWLimitMinActive	OUTPUT	BOOL	■ Hardware Endschalter – TRUE: Negativer Hardware Endschalter am Antrieb aktiv (NOT- Negative Overtravel).
HWLimitMaxActive	OUTPUT	BOOL	■ Hardware Endschalter – TRUE: Positiver Hardware Endschalter am Antrieb aktiv (POT- Positive Overtravel).
Config	IN_OUT	VMC_Config-SigmaPN_REF	Referenz zur Konfiguration der Achse.
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse.

*) Dieser Parameter wird aktuell nicht unterstützt! Es wird immer der kürzester Weg genommen. Die Prüfung erfolgt aber auf Werte von 0 bis 3.

15.3.3.3 FB 891 - VMC_InitSigma_PN - *Sigma-5/7* PROFINET Initialisierung

Beschreibung

Dieser Baustein dient zur Konfiguration der Achse. Der Baustein ist speziell angepasst an die Verwendung eines *Sigma-5/7*-Antriebs, welcher über PROFINET angebunden ist.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Enable	INPUT	BOOL	Freigabe der Initialisierung
LogicalAddress	INPUT	INT	Kleinste Adresse des Ein-/Ausgabe-Adressbereichs der Hardware-Konfiguration der Achse
ParaAccessPointAdress	INPUT	INT	Diagnoseadresse von Steckplatz 1 der Hardware-Konfiguration der Achse
InputsStartAddress	INPUT	INT	Startadresse des Eingabe-Adressbereichs der Hardware-Konfiguration der Achse
OutputsStartAddress	INPUT	INT	Startadresse des Ausgabe-Adressbereichs der Hardware-Konfiguration der Achse
EncoderType	INPUT	INT	Encoder-Typ ■ 1: Absolut-Encoder ■ 2: Inkremental-Encoder
EncoderResolutionBits	INPUT	INT	Anzahl der Bits, die einer Geber-Umdrehung entsprechen. Default: 20
FactorPosition	INPUT	REAL	Faktor zur Umrechnung der Position von Benutzereinheiten [u] in Antriebseinheiten [Inkremente] und zurück. Es gilt: $p_{[Inkremente]} = p_{[u]}$ x *FactorPosition* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2301:1 und 0x2301:2 vorgeben können. Dieser sollte 1 sein.
FactorVelocity	INPUT	REAL	Faktor zur Umrechnung der Geschwindigkeit von Benutzereinheiten [u/s] in Antriebseinheiten [Inkremente/s] und zurück. Es gilt: $v_{[Inkremente/s]} = v_{[u/s]}$ x *FactorVelocity* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2302:1 und 0x2302:2 vorgeben können. Dieser sollte 1 sein.
FactorAcceleration	INPUT	REAL	Faktor zur Umrechnung der Beschleunigung von Benutzereinheiten [u/s^2] in Antriebseinheiten [10^{-4} x Inkremente/s^2] und zurück. Es gilt: 10^{-4} x $a_{[Inkremente/s^2]} = a_{[u/s^2]}$ x *FactorAcceleration* Bitte berücksichten sie auch den Faktor, welchen Sie am Antrieb über die Objekte 0x2303:1 und 0x2303:2 vorgeben können. Dieser sollte 1 sein.
OffsetPosition	INPUT	REAL	Offset für die Nullposition [u].
MaxVelocityApp	INPUT	REAL	Maximale Geschwindigkeit der Applikation [u/s]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.
MaxAccelerationApp	INPUT	REAL	Maximale Beschleunigung der Applikation [u/s^2]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.
MaxDecelerationApp	INPUT	REAL	Maximale Verzögerung der Applikation [u/s^2]. Die Kommandoeingaben werden vor Ausführung auf den Maximalwert überprüft.

Parameter	Deklaration	Datentyp	Beschreibung
MaxPosition	INPUT	REAL	Maximale Position für die Überwachung der Softwarelimits [u].
MinPosition	INPUT	REAL	Minimale Position für die Überwachung der Softwarelimits [u].
EnableMaxPosition	INPUT	BOOL	Überwachung maximale Position ■ TRUE: Aktivierung der Überwachung der maximalen Position.
EnableMinPosition	INPUT	BOOL	Überwachung minimale Position ■ TRUE: Aktivierung der Überwachung der minimalen Position.
MinUserPosition	OUTPUT	REAL	Minimale Benutzerposition basierend auf dem minimalen Encoder Wert von 0x80000000 und dem *FactorPosition* [u].
MaxUserPosition	OUTPUT	REAL	Maximale Benutzerposition basierend auf dem maximalen Encoder Wert von 0x7FFFFFFF und dem *FactorPosition* [u].
Valid	OUTPUT	BOOL	Initialisierung ■ TRUE: Initialisierung ist gültig.
Error	OUTPUT	BOOL	■ Fehler – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden. Die Achse wird gesperrt.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Config	IN_OUT	VMC_Config-SigmaPN_REF	Datenstruktur zur Übergabe von achsabhängigen Konfigurationsdaten an den *AxisKernel*.
Axis	IN_OUT	MC_AXIS_REF	Datenstruktur zur Übergabe von achsabhängigen Informationen an *AxisKernel* und PLCopen-Bausteine.

15.4 Einsatz *Sigma-5/7* Pulse Train

15.4.1 Übersicht

Voraussetzung

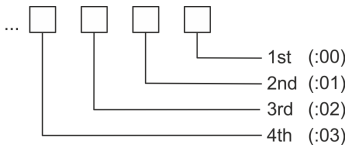
- SPEED7 Studio ab V1.7
- System MICRO bzw. System SLIO CPU mit Pulse Train Ausgabe wie z.B. CPU M13-CCF0000 bzw. CPU 013-CCF0R00.
- *Sigma-5*- bzw. *Sigma-7*-Antrieb mit Pulse Train Optionskarte

Schritte der Projektierung

1. Parameter am Antrieb einstellen
   - Die Einstellung der Parameter hat mit dem Softwaretool *Sigma Win+* zu erfolgen.
2. Hardwarekonfiguration im VIPA *SPEED7 Studio.*
   - Projektierung der CPU.
3. Programmierung im VIPA *SPEED7 Studio.*
   - *VMC_AxisControl_PT*-Baustein zur Konfiguration und zur Kommunikation mit der Achse, welche über Pulse Train angebunden ist.

15.4.2 Parameter am Antrieb einstellen

Parameter-Digits



VORSICHT!
Vor der Erstinbetriebnahme müssen Sie Ihren Antrieb mit dem Softwaretool *Sigma Win+* an Ihre Applikation anpassen! Näheres hierzu finden Sie im Handbuch zu ihrem Antrieb.

Die nachfolgende Tabelle zeigt alle Parameter auf, die nicht den Standardwerten entsprechen. Zur Abstimmung auf die *Simple Motion Control Library* sind diese über *Sigma Win+* einzustellen:

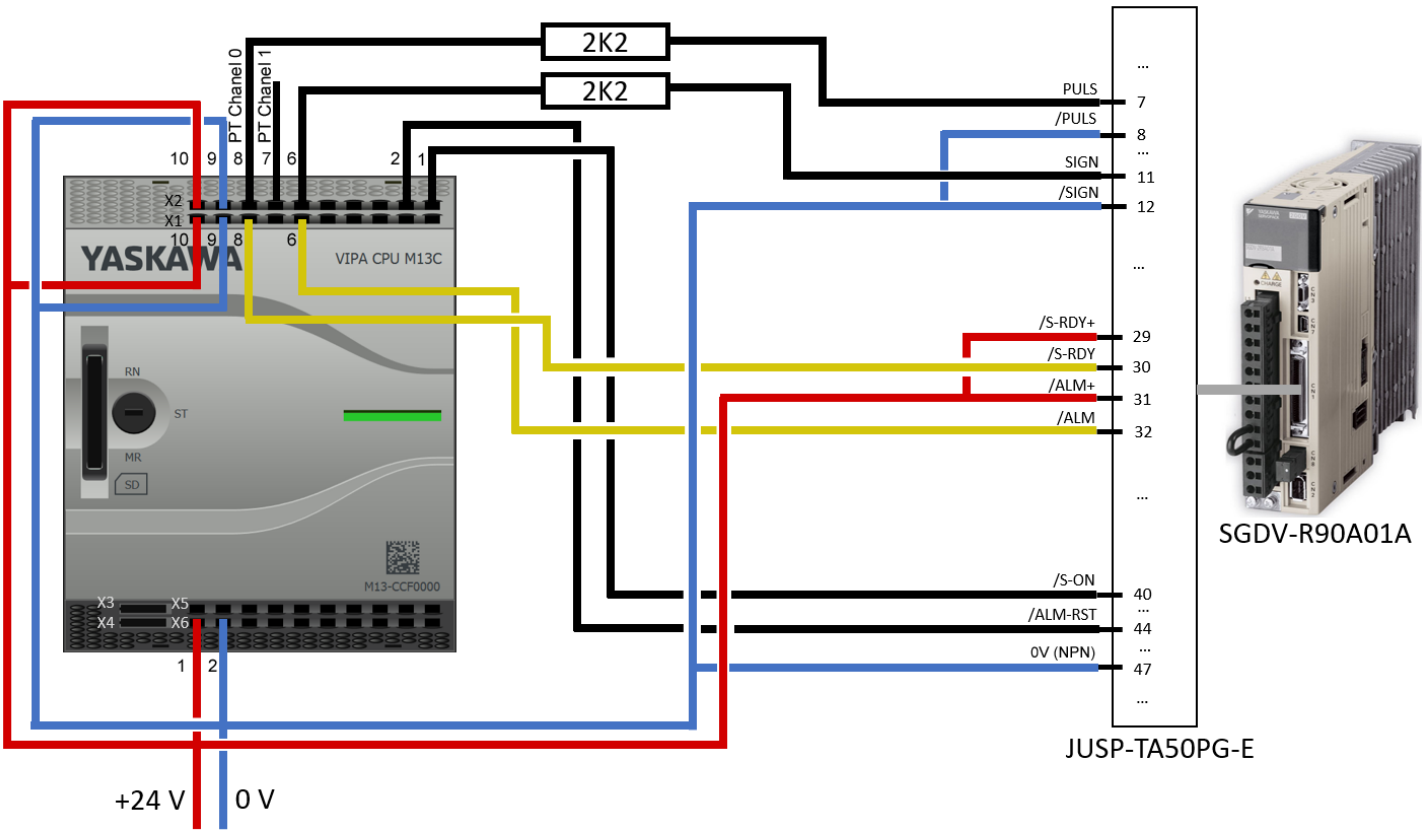
Sigma-5/7

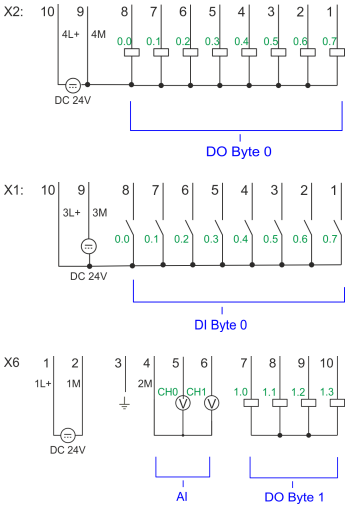
Servopack Parameter	Adresse:Digit	Name	Wert
Pn000	(2000h:01)	Basic Function Selection Switch 0	1: Position control (pulse train reference)
Pn002	(2002h:02)	Application Function Select Switch 2	1: Uses absolute encoder as incremental encoder
Pn200	(2200h:03)	Position Control Reference From Selection Switch	1: Uses reference input filter for open collector signal
Pn20E	(220Eh)	Electronic Gear Ratio (Numerator)	1024
Pn216	(2216h)	Position Reference Acceleration / Deceleration Time Constant	0
Pn217	(2217h)	Average Movement Time of Position Reference	0
Pn50A	(250Ah:02)	/P-CON Signal Mapping	8: Sets signal off
Pn50A	(250Ah:03)	P-OT Signal Mapping	8: Forward run allowed
Pn50B	(250Bh:00)	N-OT Signal Mapping	8: Reverse run allowed
Pn50B	(250Bh:02)	/P-CL Signal Mapping	8: Sets signal off
Pn50B	(250Bh:03)	/N-CL Signal Mapping	8: Sets signal off

15.4.3 Beschaltung

Beispielapplikation

Die nachfolgende Abbildung zeigt den Anschluss eines Sigma-5 Servo-Antriebs über Pulse Train an eine System MICRO CPU M13C. In diesem Beispiel ist der Pulse Train Kanal 0 (X2 - Pin 8) beschaltet. Zur Beschaltung von Kanal 1 ist X2 - Pin 7 zu verwenden.





X2	Funk-tion	Typ	LED grün rot	Beschreibung
1	DO 0.7	A		Digital Ausgang DO 7
2	DO 0.6	A		Digital Ausgang DO 6
6	DO 0.2	A		Digital Ausgang DO 2
7	DO 0.1	A		Pulse Train Kanal 1
8	DO 0.0	A		Pulse Train Kanal 0
9	0 V	E		4M: GND für Pulse Train LED leuchtet bei Fehler, Überlast bzw. Kurzschluss an den Ausgängen
10	DC 24V	E		4L+: DC 24V Leistungsversorgung für Pulse Train

X1	Funk-tion	Typ	LED grün	Beschreibung
6	DI 0.2	E		Digitaler Eingang DI 2
8	DI 0.0	E		Digitaler Eingang DI 0
9	0 V	E		3M: GND Leistungsversorgung für Onboard DI
10	DC 24V	E		3L+: DC 24V Leistungsversorgung für Onboard DI

X6	Funk-tion	Typ	LED grün	Beschreibung
1	Sys DC 24V	E		1L+: DC 24V für Elektronikversorgung
2	Sys 0V	E		1M: GND für Elektronikversorgung

15.4.4 Einsatz im VIPA *SPEED7 Studio*

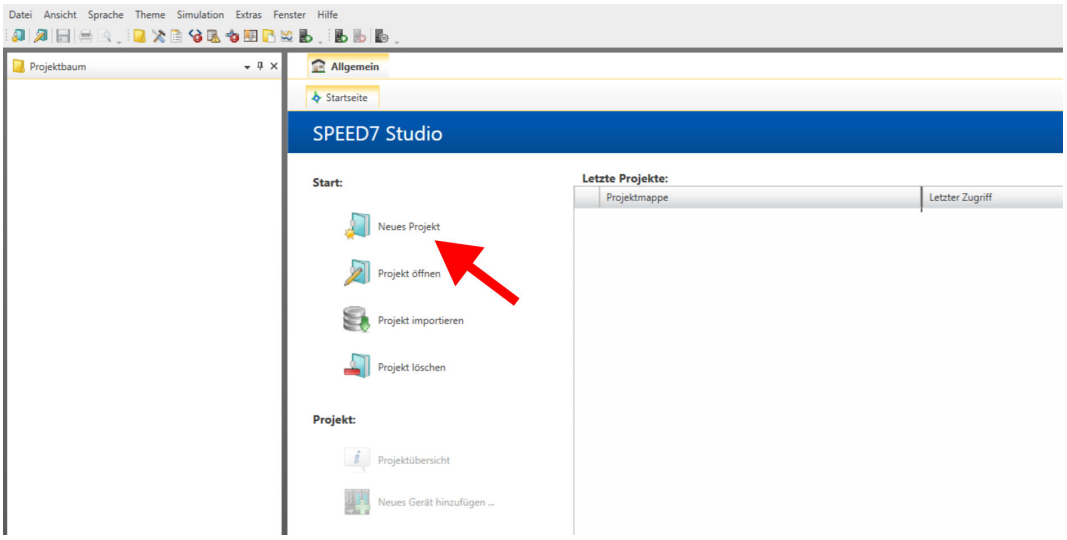
15.4.4.1 Hardware-Konfiguration

CPU im Projekt anlegen

Bitte verwenden Sie für die Projektierung das *SPEED7 Studio* ab V1.7.

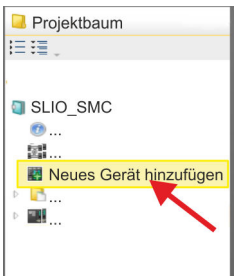
Sollten Sie einen anderen Kanal als Kanal 0 verwenden, müssen Sie diesen in der Hardware-Konfiguration und in Ihrem Anwenderprogramm entsprechend anpassen.

1. Starten Sie das *SPEED7 Studio*.

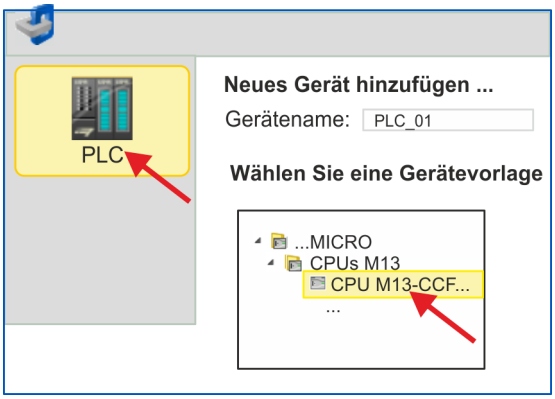


2. Erstellen sie auf der Startseite mit *"Neues Projekt"* ein neues Projekt und vergeben Sie einen *"Projektnamen"*.

⇨ Ein neues Projekt wird angelegt und in die Sicht *"Geräte und Netze"* gewechselt.



3. Klicken Sie im *Projektbaum* auf *"Neues Gerät hinzufügen ..."*.



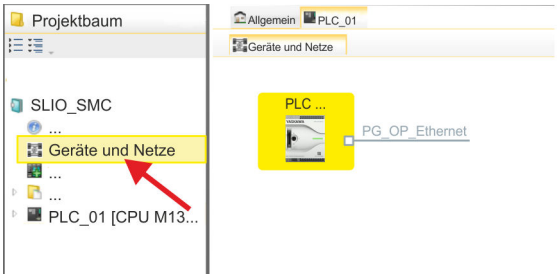
⇨ Es öffnet sich ein Dialog für die Geräteauswahl.

4. Wählen Sie unter den *"Gerätevorlagen"* Ihre CPU mit Pulse Train Funktionalität wie z.B. die System MICRO CPU M13-CCF0000 und klicken Sie auf [OK].

⇨ Die CPU wird in *"Geräte und Netze"* eingefügt und die *"Gerätekonfiguration"* geöffnet.

Ethernet-PG/OP-Kanal parametrieren

1. Klicken Sie im *Projektbaum* auf *"Geräte und Netze"*.
   ⇨ Sie erhalten eine grafische Objekt-Ansicht Ihrer CPU.



2. Klicken Sie auf das Netzwerk *"PG_OP_Ethernet"*.
3. Wählen Sie *"Kontextmenü ➔ Eigenschaften der Schnittstelle"*.
   ⇨ Es öffnet sich ein Dialogfenster. Hier können Sie IP-Adressdaten für Ihren Ethernet-PG/OP-Kanal angeben. Gültige IP-Adress-Parameter erhalten Sie von Ihrem Systemadministrator.
4. Bestätigen Sie Ihre Eingabe mit [OK].
   ⇨ Die IP-Adressdaten werden in Ihr Projekt übernommen und in *"Geräte und Netze"* unter *"Lokale Baugruppen"* aufgelistet.

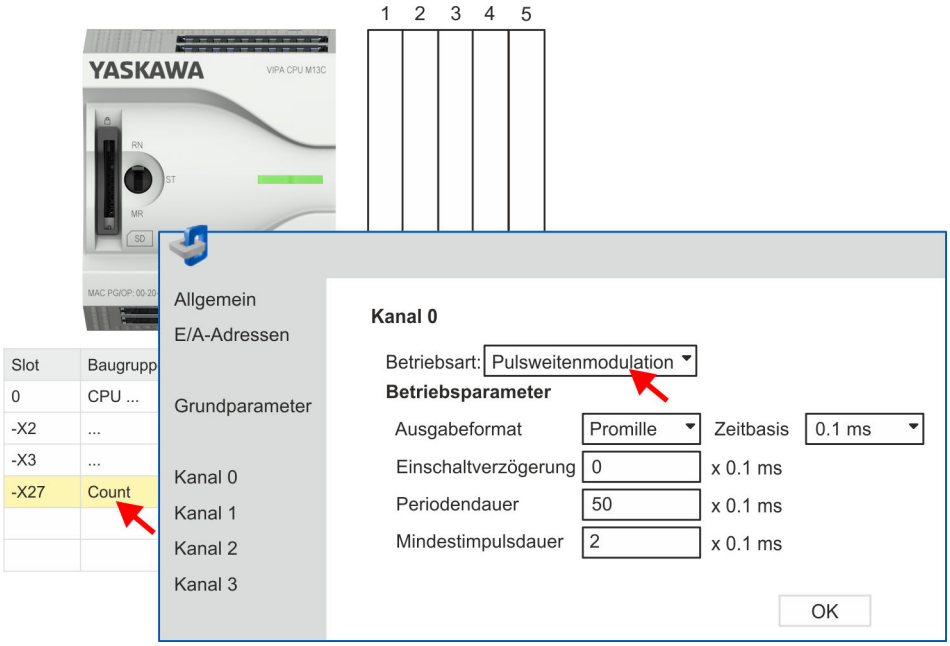
   Nach der Übertragung Ihres Projekts ist Ihre CPU über die angegebenen IP-Adressdaten via Ethernet-PG/OP-Kanal erreichbar.

E/A-Peripherie auf Pulse Train umschalten

Für die Parametrierung der Ein-/Ausgabeperipherie und der *Technologischen Funktionen* sind die entsprechenden Submodule der CPU zu verwenden. Für die Pulse Train Ausgabe ist das Submodul Count auf *"Pulsweitenmodulation"* umzuschalten.

1. Klicken Sie im *Projektbaum* auf *"PLC... > Gerätekonfiguration"*.
2. Klicken Sie in der *"Gerätekonfiguration"* auf *"-X27 Count"* und wählen Sie *"Kontextmenü ➔ Eigenschaften der Baugruppe"*.
   ⇨ Es öffnet sich der Eigenschaften-Dialog.
3. Klicken Sie auf den entsprechenden Kanal wie z.B. *"Kanal 0"* und stellen Sie unter *"Betriebsart"* die Funktion *"Pulsweitenmodulation"* ein.

4. Die für Pulse Train erforderlichen Betriebsparameter werden intern auf die entsprechenden Werte angepasst. Lassen Sie alle Werte unverändert.

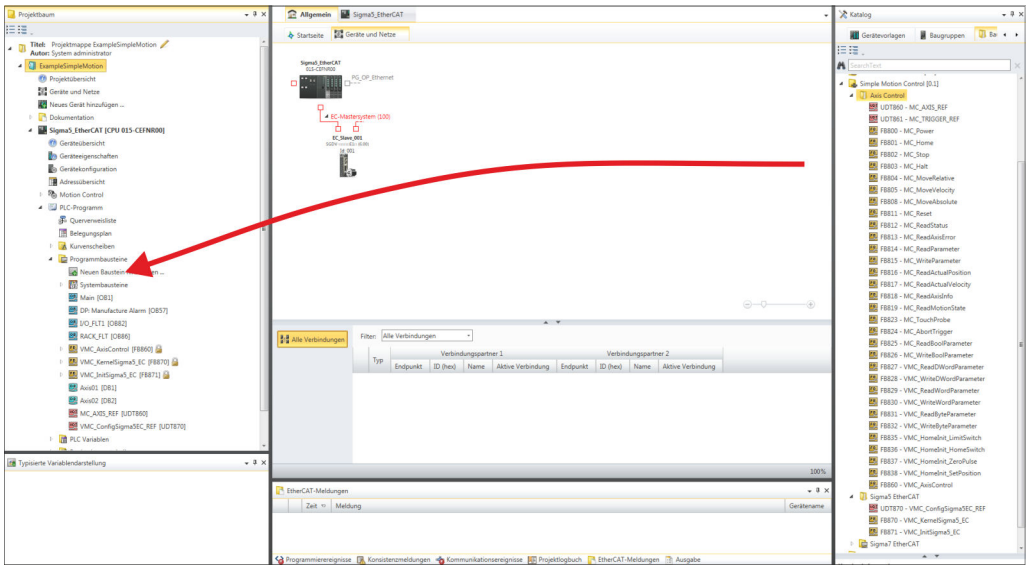


5. Schließen Sie den Dialog mit [OK].

6. Wählen Sie *"Projekt ➔ Alles übersetzen"*.

15.4.4.2 Anwender-Programm

Baustein in Projekt kopieren



Öffnen Sie im *"Katalog"* unter *"Bausteine"* *"Simple Motion Control"* und ziehen Sie per Drag&Drop folgenden Baustein in *"Programmbausteine"* des *Projektbaums*:

- *Sigma5+7* Pulse Train
  - FB 875 - VMC_AxisControl_PT ↳ *Kap. 15.4.5.1 "FB 875 - VMC_AxisControl_PT - Achskontrolle über Pulse Train" Seite 686*

OB 1

Konfiguration der Achse

Sollten Sie einen anderen Kanal als Kanal 0 verwenden, müssen Sie diesen in der Hardware-Konfiguration und in Ihrem Anwenderprogramm entsprechend anpassen.

1. Öffnen Sie im *Projektbaum* innerhalb der CPU unter *"PLC-Programm"*, *"Programmbausteine"* den OB 1 und programmieren Sie den Call FB 875, DB 875.

   ⇨ Es öffnet sich der Dialog *"Instanz-Datenbaustein hinzufügen"*.

2. Stellen Sie, wenn nicht schon geschehen, die Nummer für den Instanz-Datenbaustein ein und schließen Sie den Dialog mit [OK].

   ⇨ Der Bausteinaufruf wird angelegt und die Parameter werden aufgelistet

3. Vergeben Sie für das Beispielprojekt folgende Parameter. Beachten Sie hier insbesondere die beiden Umrechnungsfaktoren *FactorPosition* und *FactorVelocity*:

   ⇨
```
CALL  FB    "VMC_AxisControl_PT" , "DI_AxisControl_PT"
          S_ChannelNumberPWM      := 0
          S_Ready                 := E 136.0
          S_Alarm                 := E 136.2
          FactorPosition          := 1024.0
          FactorVelocity          := 976.5625
          AxisEnable              := M 100.1
          AxisReset               := M 100.2
          StopExecute             := M 100.3
          MvVelocityExecute       := M 100.4
          MvRelativeExecute       := M 100.5
          JogPositive             := M 100.6
          JogNegative             := M 100.7
          PositionDistance        := MD 102
          Velocity                := MD 106
          S_On                    := A 136.7
          S_Direction             := A 136.2
          S_AlarmReset            := A 136.6
          MinUserDistance         := MD 110
          MaxUserDistance         := MD 114
          MinUserVelocity         := MD 118
          MaxUserVelocity         := MD 122
          AxisReady               := M 101.3
          AxisEnabled             := M 101.4
          AxisError               := M 101.5
          AxisErrorID             := MW 126
          DriveError              := M 101.6
          CmdActive               := MB 128
          CmdDone                 := M 130.0
          CmdBusy                 := M 130.1
          CmdAborted              := M 130.2
          CmdError                := M 130.3
          CmdErrorID              := MW 132
```

   Die Adressen von *S_Ready* und *S_Alarm* ergeben sich aus den Adressen der Eingänge, welche mit den Digitalausgängen des Antrieb verbunden sind. Diese können über das Submodul *"-X25 DI/DIO"* der CPU ermittelt werden.

   Die Adressen von *S_On*, *S_Direction* und *S_AlarmReset* ergeben sich aus den Adressen der Ausgänge, welche mit den Digitaleingängen des Antriebs verbunden sind. Diese können über das Submodul *"-X25 DI/DIO"* der CPU ermittelt werden.

Zeitlicher Ablauf

1. Wählen Sie *"Projekt ➔ Alles übersetzen"* und übertragen Sie das Projekt in Ihre CPU.
   ⇨ Sie können jetzt Ihre Applikation in Betrieb nehmen.

VORSICHT!
Bitte beachten Sie immer die Sicherheitshinweise zu ihrem Antrieb, insbesondere bei der Inbetriebnahme!

2. Bringen Sie Ihre CPU in RUN und schalten Sie Ihren Antrieb ein.
   ⇨ Der FB 875 - VMC_AxisControl_PT wird zyklisch abgearbeitet.
3. Sobald *AxisReady* = TRUE meldet, können Sie mit *AxisEnable* den Antrieb frei geben.
4. Sie haben jetzt die Möglichkeit über die entsprechenden Parameter Ihren Antrieb zu steuern und dessen Status abzufragen. ↬ *Kap. 15.4.5.1 "FB 875 - VMC_Axis-Control_PT - Achskontrolle über Pulse Train" Seite 686*

Steuerung des Antriebs über HMI

Sie haben die Möglichkeit über ein HMI Ihren Antrieb zu steuern. Hierzu gibt es für Movicon eine vorgefertigte Symbolbibliothek für den Zugriff auf den VMC_AxisControl_PT Funktionsbaustein. ↬ *Kap. 15.9 "Antrieb über HMI steuern" Seite 825*

15.4.5 Antriebsspezifischer Baustein

15.4.5.1 FB 875 - VMC_AxisControl_PT - Achskontrolle über Pulse Train

15.4.5.1.1 Beschreibung

Mit dem FB *VMC_AxisControl_PT* können Sie eine über Pulse Train angebundene Achse steuern. Sie können den Status des Antriebs abrufen, den Antrieb ein- bzw. ausschalten oder verschiedene Bewegungskommandos ausführen. In den Instanzdaten des Bausteins befindet sich ein gesonderter Speicherbereich. Über diesen können Sie mittels eines HMI Ihre Achse steuern. ↬ *Kap. 15.9 "Antrieb über HMI steuern" Seite 825*

Die Ansteuerung eines Pulse Train Antriebs erfolgt ausschließlich mit dem FB 875 VMC_AxisControl_PT. PLCopen-Bausteine werden nicht unterstützt!

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
S_Channel-NumberPWM	INPUT	INT	Kanalnummer vom PWM-Ausgang, der für die Ansteuerung vom Pulse-Train-Eingang des Servos (Signal PULS) verwendet wird.
S_Ready	INPUT	BOOL	■ Digitaleingang zur Anbindung des S_Ready-Signals (S-RDY) – TRUE: Servo ist bereit für das S_On-Signal.
S_Alarm	INPUT	BOOL	■ Digitaleingang zur Anbindung des S_Alarm-Signals (ALM) – FALSE, wenn der Servo einen Fehler erkannt hat.
FactorPosition	INPUT	REAL	Faktor zur Umrechnung der Position von Benutzer- in Antriebseinheiten (Inkremente) und zurück. ↬ *"FactorPosition" Seite 689*
FactorVelocity	INPUT	REAL	Faktor zur Umrechnung der Geschwindigkeit von Benutzer- in Antriebseinheiten (Inkremente/s) und zurück. ↬ *"FactorVelocity" Seite 690*

Parameter	Deklaration	Datentyp	Beschreibung
AxisEnable	INPUT	BOOL	■ Achsenfreigabe – TRUE: Die Achse wird freigegeben. – FALSE: Die Achse wird gesperrt.
AxisReset	INPUT	BOOL	■ Reset Achse – Flanke 0-1: Reset der Achse wird durchgeführt. – Der Zustand eines mit *AxisReset* gestarteten Reset wird nicht an den Ausgängen *CmdActive*, *CmdDone*, *CmdBusy*, *CmdAborted*, *CmdError* und *CmdErrorID* ausgegeben.
StopExecute	INPUT	BOOL	■ Achse stoppen – Flanke 0-1: Stoppen der Achse wird gestartet. Hinweis: StopExecute = 1: Kein anderes Kommando kann gestartet werden!
MvVelocityExecute	INPUT	BOOL	■ Verfahren der Achse starten – Flanke 0-1: Die Achse wird auf die angegebene Geschwindigkeit beschleunigt / abgebremst.
MvRelativeExecute	INPUT	BOOL	■ Verfahren der Achse starten – Flanke 0-1: Die relative Positionierung der Achse wird gestartet.
JogPositive	INPUT	BOOL	Tipp-Betrieb positiv ■ Achse mit konstanter Geschwindigkeit in positive Richtung verfahren – Flanke 0-1: Das Verfahren der Achse mit konstanter Geschwindigkeit wird gestartet. – Flanke 1-0: Die Achse wird gestoppt.
JogNegative	INPUT	BOOL	Tipp-Betrieb negativ ■ Achse mit konstanter Geschwindigkeit in negative Richtung verfahren – Flanke 0-1: Das Verfahren der Achse mit konstanter Geschwindigkeit wird gestartet. – Flanke 1-0: Die Achse wird gestoppt.
PositionDistance	INPUT	REAL	Absolute Position bzw. relative Wegstrecke für *MvRelativeExecute* [Anwendereinheiten].
Velocity	INPUT	REAL	Geschwindigkeitsvorgabe (vorzeichenbehafteter Wert) in [Anwendereinheiten/s].
S_On	OUTPUT	BOOL	■ Digitalausgang zur Ansteuerung des S_On-Signals (S-ON) – TRUE: schaltet den Servo ein. – FALSE: schaltet den Servo aus.
S_Direction	OUTPUT	BOOL	■ Digitalausgang zur Ansteuerung des S_Direction-Signals (SIGN) – TRUE: Vorgabe der Drehrichtung positive Richtung für den Servo. – FALSE: Vorgabe der Drehrichtung negative Richtung für den Servo.
S_AlarmReset	OUTPUT	BOOL	■ Digitalausgang zur Ansteuerung des S_AlarmReset-Signals (ALM-RST) – TRUE: Alarme werden im Servo zurückgesetzt. – FALSE: Alarme im Servo bleiben bestehen.

Parameter	Deklaration	Datentyp	Beschreibung
MinUserDistance	OUTPUT	REAL	Minimaler Verfahrweg (1 Inkrement) des Servos [Anwendereinheiten].
MaxUserDistance	OUTPUT	REAL	Maximaler Verfahrweg (8388607 Inkremente = maximale Anzahl Impulse des PWM-Ausgangs) des Servos [Anwendereinheiten].
MinUserVelocity	OUTPUT	REAL	Minimale Geschwindigkeit (Periodendauer = 65535µs = maximale Periodendauer des PWM-Ausgangs) des Servos [Anwendereinheiten].
MaxUserVelocity	OUTPUT	REAL	Maximale Geschwindigkeit (Periodendauer = 20µs = minimale Periodendauer des PWM-Ausgang) des Servos [Anwendereinheiten].
AxisReady	OUTPUT	BOOL	■ AxisReady – TRUE: Die Achse ist einschaltbereit. – FALSE: Die Achse ist nicht einschaltbereit. → Prüfe und behebe *AxisError* (siehe *AxisErrorID*). → Prüfe und behebe *DriveError*.
AxisEnabled	OUTPUT	BOOL	■ Status Achse – TRUE: Achse ist eingeschaltet und nimmt Bewegungsaufträge an. – FALSE: Achse ist nicht eingeschaltet und nimmt keine Bewegungsaufträge an. ■ Bedingungen für *AxisEnabled* = TRUE – *AxisEnable* = TRUE – *S_Ready* = TRUE – *S_Alarm* = TRUE
AxisError	OUTPUT	BOOL	■ Fehler bei Motion Achse – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *AxisErrorID* entnommen werden. → Die Achse wird gesperrt (*S_On* = FALSE und *AxisEnabled* = FALSE). Kommando wird nicht ausgeführt.
AxisErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
DriveError	OUTPUT	BOOL	■ Fehler direkt am Antrieb – TRUE: Ein Fehler ist aufgetreten. – → Die Achse wird gesperrt.
CmdActive	OUTPUT	BYTE	■ Kommando – 0: kein Cmd aktiv – 1: STOP – 2: MvVelocity – 3: MvRelative – 4: JogPos – 5: JogNeg
CmdDone	OUTPUT	BOOL	■ Status Done – TRUE: Auftrag wurde ohne Fehler beendet.
CmdBusy	OUTPUT	BOOL	■ Status Busy – TRUE: Auftrag ist in Bearbeitung.

Parameter	Deklaration	Datentyp	Beschreibung
CmdAborted	OUTPUT	BOOL	■ Status Aborted – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen. Hinweis: *CmdAborted* wird beim Start eines Cmd zurückgesetzt
CmdError	OUTPUT	BOOL	■ Status Error – TRUE: Ein Fehler ist aufgetreten. Die Achse wird gesperrt Zusätzliche Fehlerinformationen können dem Parameter *CmdErrorID* entnommen werden.
CmdErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*

15.4.5.1.2 Umrechnungsfaktoren

FactorPosition

Die Berechnung von FactorPosition ist nur gültig, wenn der Servo-Parameter Reference Pulse Multiplier (Pn218) = 1 gesetzt ist.

$$FactorPosition = \frac{Resolution}{Numerator} \cdot Denominator$$

FactorPosition - Faktor zur Umrechnung der Position von Benutzer- in Antriebseinheiten (Inkremente) und zurück.

Resolution - Anzahl Inkremente pro Anwendereinheit

Numerator - Zähler: Electronic Gear Ratio (Pn20E) der Servo-Parameter

Denominator - Nenner: Electronic Gear Ratio (Pn210) der Servo-Parameter

Beispiel Anwendereinheit für Position = 1 Umdrehung

FactorPosition - Faktor zur Umrechnung der Position von Benutzer- in Antriebseinheiten (Inkremente) und zurück.

Resolution - Anzahl Inkremente pro Anwendereinheit

Resolution = 2^{20} = 1048576

Numerator - Zähler: Electronic Gear Ratio (Pn20E) der Servo-Parameter

Numerator = 1024

Denominator - Nenner: Electronic Gear Ratio (Pn210) der Servo-Parameter

Denominator = 1

$$FactorPosition = \frac{Resolution}{Numerator} \cdot Denominator$$

$$FactorPosition = \frac{1048576}{1024} \cdot 1 = 1024$$

Beispiel minimaler Verfahrweg

MinPos - Minimaler Verfahrweg in Umdehungen

Resolution - Anzahl Inkremente pro Anwendereinheit

Resolution = 2^{20} = 1048576

Numerator - Zähler: Electronic Gear Ratio (Pn20E) der Servo-Parameter

Numerator = 1024

Period - Minimale Periode

Period = 1

$$MinPos = Numerator \cdot \frac{Period}{Resolutioon}$$

$$MinPos = 1024 \cdot \frac{1}{1048576} = \frac{1}{1024}$$

Beispiel maximaler Verfahrweg

MaxPos - Maximaler Verfahrweg in Umdrehungen

Resolution - Anzahl Inkremente pro Anwendereinheit

Resolution = 2^{20} = 1048576

Numerator - Zähler: Electronic Gear Ratio (Pn20E) der Servo-Parameter

Numerator = 1024

Period - Maximale Periode

Period = 8388607

$$MaxPos = Numerator \cdot \frac{Period}{Resolution}$$

$$MaxPos = 1024 \cdot \frac{8388607}{1048576} = 8192$$

FactorVelocity

Die Berechnung von FactorVelocity ist nur gültig, wenn der Servo-Parameter Reference Pulse Multiplier (Pn218) = 1 gesetzt ist.

$$FactorVelocity = Time \cdot \frac{\frac{Numerator}{Denominator}}{Resolution}$$

Time - Zeit für 1 Umdrehung in µs

Numerator - Zähler: Electronic Gear Ratio (Pn20E) der Servo-Parameter

Denominator - Nenner: Electronic Gear Ratio (Pn210) der Servo-Parameter

Resolution - Anzahl Inkremente pro Anwendereinheit

Beispiel Anwendereinheit für Geschwindigkeit = Umdrehung/min

FactorVelocity - Faktor zur Umrechnung von Benutzer- in Antriebseinheiten (Inkremente/sec) und zurück.

Time - Zeit für 1 Umdrehung in µs

Time = 1min = $60 \cdot 10^6$µs

Numerator - Zähler: Electronic Gear Ratio (Pn20E) der Servo-Parameter

Numerator = 1024

Denominator - Nenner: Electronic Gear Ratio (Pn210) der Servo-Parameter

Denominator = 1

Resolution - Anzahl Inkremente pro Anwendereinheit

Resolution = 2^{20} = 1048576

$$FactorVelocity = Time \cdot \frac{\frac{Numerator}{Denominator}}{Resolution}$$

$$FactorVelocity = 60 \cdot 10^6 \frac{\frac{1024}{1}}{1048576} = \frac{60 \cdot 10^6}{1024} = 58593{,}75$$

Beispiel Anwendereinheit für Geschwindigkeit = Umdrehung/s

FactorVelocity - Faktor zur Umrechnung von Benutzer- in Antriebseinheiten (Inkremente/sec) und zurück.

Time - Zeit für 1 Umdrehung in µs

Time = 1s = 10^6µs

Numerator - Zähler: Electronic Gear Ratio (Pn20E) der Servo-Parameter

Numerator = 1024

Denominator - Nenner: Electronic Gear Ratio (Pn210) der Servo-Parameter

Denominator = 1

Resolution - Anzahl Inkremente pro Anwendereinheit

Resolution = 2^{20} = 1048576

$$FactorVelocity = Time \cdot \frac{\frac{Numerator}{Denominator}}{Resolution}$$

$$FactorVelocity = 10^6 \frac{\frac{1024}{1}}{1048576} = \frac{10^6}{1024} = 976{,}5625$$

Minimale Geschwindigkeit für Umdrehungen/min

MinVel - Minimale Geschwindigkeit in Umdrehungen/min

FactorVelocity - Faktor zur Umrechnung von Benutzer- in Antriebseinheiten (Inkremente/s) und zurück.

$$MinVel = \frac{FactorVelocity}{65535} = \frac{58593,75}{65535} = 0,89$$

Maximale Geschwindigkeit für Umdrehungen/min

MaxVel - Maximale Geschwindigkeit in Umdrehungen/min

FactorVelocity - Faktor zur Umrechnung von Benutzer- in Antriebseinheiten (Inkremente/s) und zurück.

$$MaxVel = \frac{FactorVelocity}{20} = \frac{58593,75}{20} = 2929,69$$

15.4.5.1.3 Funktionalität

Antrieb ein- bzw. ausschalten

- Der Eingang *AxisEnable* dient zum Ein- und Ausschalten einer Achse.
- Das Einschalten ist nur möglich, wenn *AxisReady* = TRUE meldet, d.h. die Achse einschaltbereit ist.
- Sobald die Achse eingeschaltet ist, wird dies durch die Statusinformation *AxisEnabled* angezeigt.
- Hat die Achse einen Fehler, wird dies durch die Statusinformation *AxisError* angezeigt. Weitere Informationen liefert *AxisErrorID*.

Bitte beachten Sie, dass Sie den Baustein immer innerhalb des OB 1 aufrufen, ansonsten erhalten Sie die Fehlermeldung 0x8317.

Verhalten der Ausgänge *CmdActive*, *CmdDone* und *CmdBusy*

Die Kommandoabarbeitung lässt sich in 3 Phasen gliedern. Abhängig von der Betriebsart zeigen die Ausgänge *CmdActive*, *CmdDone* und *CmdBusy* innerhalb dieser Phasen folgendes Verhalten:

Geschwindigkeitsregelung mit *Velocity* <> 0

- Phase 1: Mit einer Flanke 0-1 an *MvVelocityExecute* wird das Kommando gestartet.
  - *CmdActive* = 2, *CmdDone* = FALSE, *CmdBusy* = TRUE
- Phase 2: Die Geschwindigkeitsvorgabe wurde erreicht, *MvVelocityExecute* = TRUE
  - Kommando wird weiterhin ausgeführt.
  - *CmdActive* = 2, *CmdDone* = TRUE, *CmdBusy* = FALSE
- Phase 3: *MvVelocityExecute* = FALSE
  - Kommando wird weiterhin ausgeführt.
  - *CmdActive* = 2, *CmdDone* = FALSE, *CmdBusy* = FALSE

Geschwindigkeitsregelung mit *Velocity* = 0

- Phase 1: Mit einer Flanke 0-1 an *MvVelocityExecute* wird das Kommando gestartet.
  - *CmdActive* = 2, *CmdDone* = FALSE, *CmdBusy* = TRUE
- Phase 2: Die Geschwindigkeit 0 wurde erreicht, *MvVelocityExecute* = TRUE
  - Achse befindet sich im Stillstand und ist bereit für weitere Kommandos.
  - *CmdActive* = 0, *CmdDone* = TRUE, *CmdBusy* = FALSE
- Phase 3: *MvVelocityExecute* = FALSE
  - Achse befindet sich im Stillstand und ist bereit für weitere Kommandos.
  - *CmdActive* = 0, *CmdDone* = FALSE, *CmdBusy* = FALSE

Achse stoppen

- Phase 1: Mit einer Flanke 0-1 an *StopExecute* wird das Kommando gestartet.
  - *CmdActive* = 1, *CmdDone* = FALSE, *CmdBusy* = TRUE
- Phase 2: Die Geschwindigkeit 0 wurden erreicht, *StopExecute* = TRUE
  - Achse befindet sich im Stillstand und das Stop-Kommando blockiert die Ausführung weiterer Kommandos.
  - *CmdActive* = 1, *CmdDone* = TRUE, *CmdBusy* = FALSE
- Phase 3: *StopExecute* = FALSE
  - Achse befindet sich im Stillstand und ist bereit für weitere Kommandos.
  - *CmdActive* = 0, *CmdDone* = FALSE, *CmdBusy* = FALSE

Relative Positionierung

- Phase 1: Mit einer Flanke 0-1 an *MvRelativeExecute* wird das Kommando gestartet.
  - *CmdActive* = 3, *CmdDone* = FALSE, *CmdBusy* = TRUE
- Phase 2: Das Positionsziel wurden erreicht, *MvRelativeExecute* = TRUE
  - Es ist kein Kommando aktiv
  - *CmdActive* = 0, *CmdDone* = TRUE, *CmdBusy* = FALSE
- Phase 3: *MvRelativeExecute* = FALSE
  - *CmdActive* = 0, *CmdDone* = FALSE, *CmdBusy* = FALSE

Tipp-Betrieb

- Phase 1: Mit einer Flanke 0-1 an *JogPositive* bzw. *JogNegative* wird das Kommando gestartet.
  - *CmdActive* = 4 bzw. 5, *CmdDone* = FALSE, *CmdBusy* = TRUE
- Phase 2: Die Geschwindigkeitsvorgabe wurden erreicht, *JogPositive* = TRUE bzw. *JogNegative* = TRUE.
  - Kommando ist noch aktiv, Achse wird erst mit *JogPositive* = FALSE bzw. *JogNegative* = FALSE gestoppt.
  - *CmdActive* = 4 bzw. 5, *CmdDone* = TRUE, *CmdBusy* = FALSE
- Phase 3: *JogPositive* = FALSE bzw. *JogNegative* = FALSE
  - Achse befindet sich im Stillstand und ist bereit für weitere Kommandos.
  - *CmdActive* = 0, *CmdDone* = FALSE, *CmdBusy* = FALSE

Antriebsfehler quittieren

- Mit *AxisReset* können Sie Fehler am Antrieb quittieren.
- Anliegende Fehler werden über *DriveError* zurück gemeldet.

Achse stoppen - MC_Stop

- Eine Achse in Bewegung können Sie durch Setzen von *StopExecute* stoppen.
- Solange *StopExecute* gesetzt ist, werden keine weiteren Pulse generiert und alle Kommandos sind gesperrt.

Geschwindigkeitsmodus - MC_MoveVelocity

- Voraussetzung: Der Antrieb ist eingeschaltet und *AxisReady* = TRUE.
- Mit *MvVelocityExecute* können Sie die Achse zum Drehen mit konstanter Geschwindigkeit bringen.

- Die Geschwindigkeit geben Sie über *Velocity* vor.
- Durch Vorgabe von 0 stoppt die Achse ebenso wie mit *StopExecute*.
- Die Drehrichtung bestimmen Sie über das Vorzeichen von *Velocity*.
- Der Wert für *Velocity* darf 0 sein oder *MinUserVelocity* ≤ *Velocity* ≤ *MaxUserVelocity*.

Systembedingt kann die Istgeschwindigkeit von der Sollgeschwindigkeit abweichen. Die Differenz MaxVelError nimmt mit steigender Drehzahl zu und kann mit der nachfolgenden Formel ermittelt werden.

$$MaxVelError = \frac{FactorVelocity}{20} - \frac{FactorVelocity}{21}$$

Relative Positionierung - MC_MoveRelative

- Voraussetzung: Der Antrieb ist eingeschaltet und *AxisReady* = TRUE.
- Die relative Positionierung erfolgt über *MvRelativeExecute*.
- Die Distanz können Sie über *PositionDistance* in Anwendereinheiten vorgeben.
- Die Drehrichtung bestimmen Sie über das Vorzeichen von *PositionDistance*.
- Die Geschwindigkeit geben Sie über *Velocity* vor.
- Durch Setzen von *StopExecute* können Sie ein laufendes Kommando stoppen.

Tipp-Betrieb - Jogging

- Voraussetzung: Der Antrieb ist eingeschaltet und *AxisReady* = TRUE.
- Mit einer Flanke 0-1 an *JogPositive* oder *JogNegative* können Sie Ihren Antrieb im Tipp-Betrieb steuern. Hierbei wird ein Jogging Kommando in die entsprechende Drehrichtung ausgeführt.
- Die Geschwindigkeit geben Sie über *Velocity* vor. Das Vorzeichen ist nicht relevant.
- Mit einer Flanke 1-0 an *JogPositive* oder *JogNegative* bzw. durch Setzen von *StopExecute* wird die Achse gestoppt.

Bitte beachten Sie, dass Sie im Tipp-Betrieb bei Velocity = 0 eine Fehlermeldung (0x8003) erhalten!

15.5 Einsatz Frequenzumrichter über PWM

15.5.1 Übersicht

Voraussetzung

- SPEED7 Studio ab V1.7.1
- System MICRO bzw. System SLIO CPU mit PWM-Ausgabe wie z.B. CPU M13-CCF0000 bzw. CPU 013-CCF0R00.
- Frequenzumrichter mit PWM-Eingang z.B. *V1000*.

Schritte der Projektierung

1. Parameter am Frequenzumrichter einstellen
   - Die Einstellung der Parameter hat mit dem Softwaretool *Drive Wizard+* zu erfolgen.
2. Hardwarekonfiguration im VIPA *SPEED7 Studio.*
   - Projektierung der CPU.
3. Programmierung im VIPA *SPEED7 Studio.*
   - *VMC_AxisControlV1000PWM*-Baustein zur Konfiguration und zur Kommunikation mit der Achse, welche über PWM angebunden ist.

15.5.2 Parameter am Frequenzumrichter einstellen

VORSICHT!
Vor der Erstinbetriebnahme müssen Sie Ihren Frequenzumrichter mit dem Softwaretool *Drive Wizard+* an Ihre Applikation anpassen! Näheres hierzu finden Sie im Handbuch zu ihrem Frequenzumrichter.

Die nachfolgende Tabelle zeigt alle Parameter auf, die nicht den Standardwerten entsprechen. Zur Abstimmung auf die *Simple Motion Control Library* sind diese über *Drive Wizard+* einzustellen. Danach folgt eine Tabelle mit Parameter, welche in Abhängigkeit von der Applikation angepasst werden können.

Nr.	Parameter, die vom Standard abweichen	Einstellung für *Simple Motion Control Library*
B1-01	Auswahl Frequenzsollwert	■ 4: Pulse train input
B1-02	Auswahl Start-Befehl	■ 1: Control circuit terminal
H1-01	Funktionsauswahl für Digitaleingänge S1	■ 0040: Forward Run Command
H1-02	Funktionsauswahl für Digitaleingänge S2	■ 0041: Reverse Run Command
H2-01	Funktionswahl Klemme MA, MB und MC (Relais)	■ 000E: Fault
H2-02	Funktionswahl für Klemme P1 (Open-Collector-Ausgang)	■ 0006
H6-01	Funktionsauswahl für die Impulsfolgeeingangsklemmen RP	■ 0: Frequency reference
H6-02	Skalierung für Impulsfolgeeingang	■ 20000Hz
H6-03	Verstärkung für Impulsfolgeeingang	■ 100.0%
H6-04	Vorspannung für Impulsfolgeeingang	■ 0.0%
H6-05	Filterzeit für Impulsfolgeeingang	■ 0.10s
H6-06	Auswahl der Überwachung für Impulsfolgeausgangsklemme MP	■ 102: Output frequency
H6-07	Überwachungsskalierung für Impulsfolgeeingang	■ 20000Hz

Nr.	Parameter in Abhängigkeit von der Applikation	Beispiel
C1-01	Hochlaufzeit 1	■ 10.00s
C1-02	Tieflaufzeit 1	■ 10.00s
C1-10	Einstellschritte für Hochlauf-/Tieflaufzeit	■ 0: 0.01- second units
C1-11	Umschaltfrequenz für Hochlauf-/Tieflaufzeit	■ 0.0Hz
O1-02	Auswahl Anwender-Überwachungsparameter nach dem Einschalten	■ 1: Frequency reference
O1-03	Auswahl Anzeige am digitalen Bedienteil	■ 2: min-1 unit

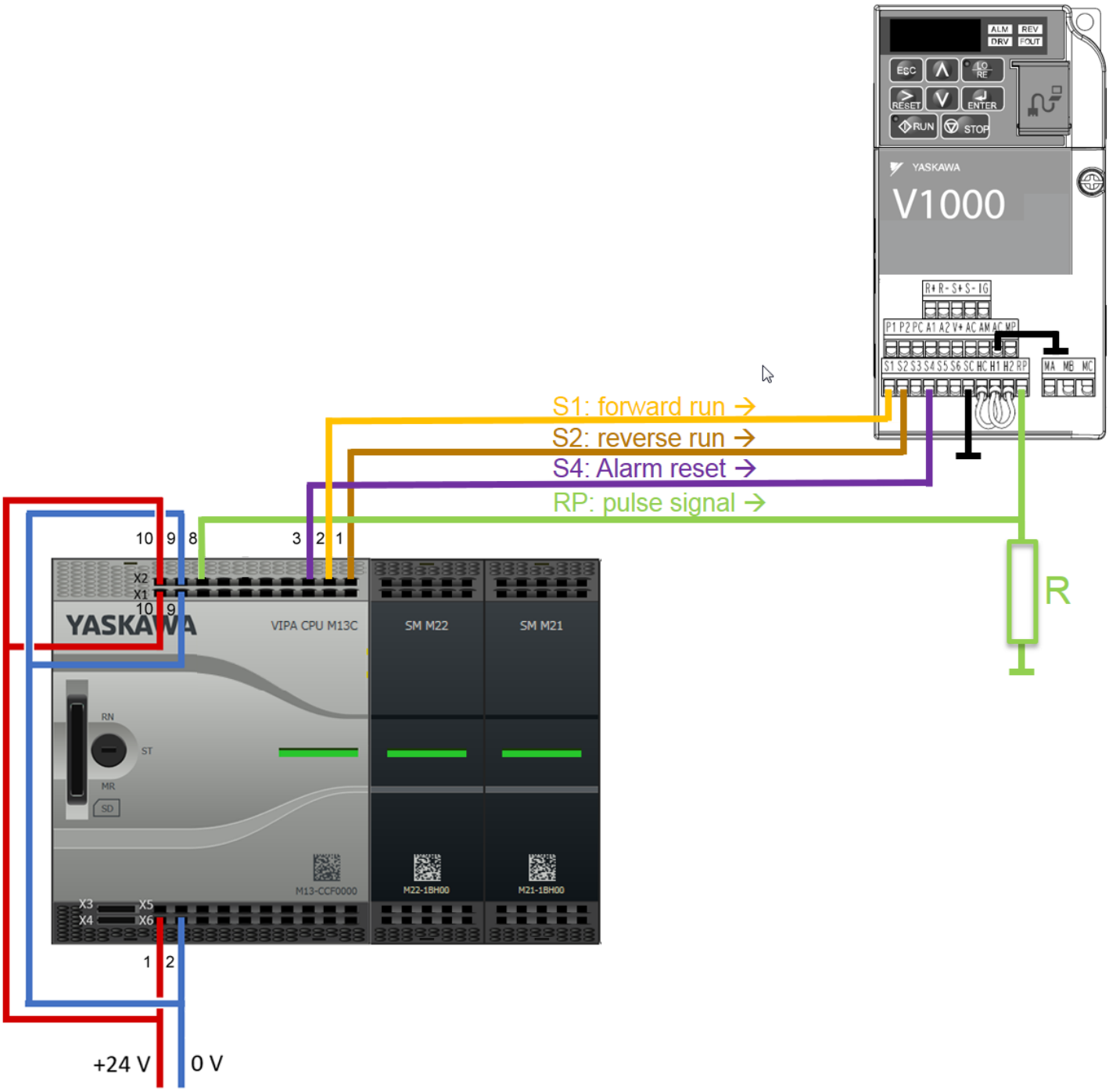
Damit alle Einstellungen übernommen werden, müssen Sie den Frequenzumrichter nach der Parametrierung neu starten!

15.5.3 Beschaltung

15.5.3.1 Ansteuerung *V1000* Eingänge

Beispielapplikation

Die nachfolgende Abbildung zeigt eine Beispielapplikation zum Anschluss der Eingänge eines V1000 Frequenzumrichters über PWM an eine System MICRO CPU M13C. In diesem Beispiel ist der PWM Kanal 0 (X2 - Pin 8) beschaltet. Zur Beschaltung von Kanal 1 ist X2 - Pin 7 zu verwenden.

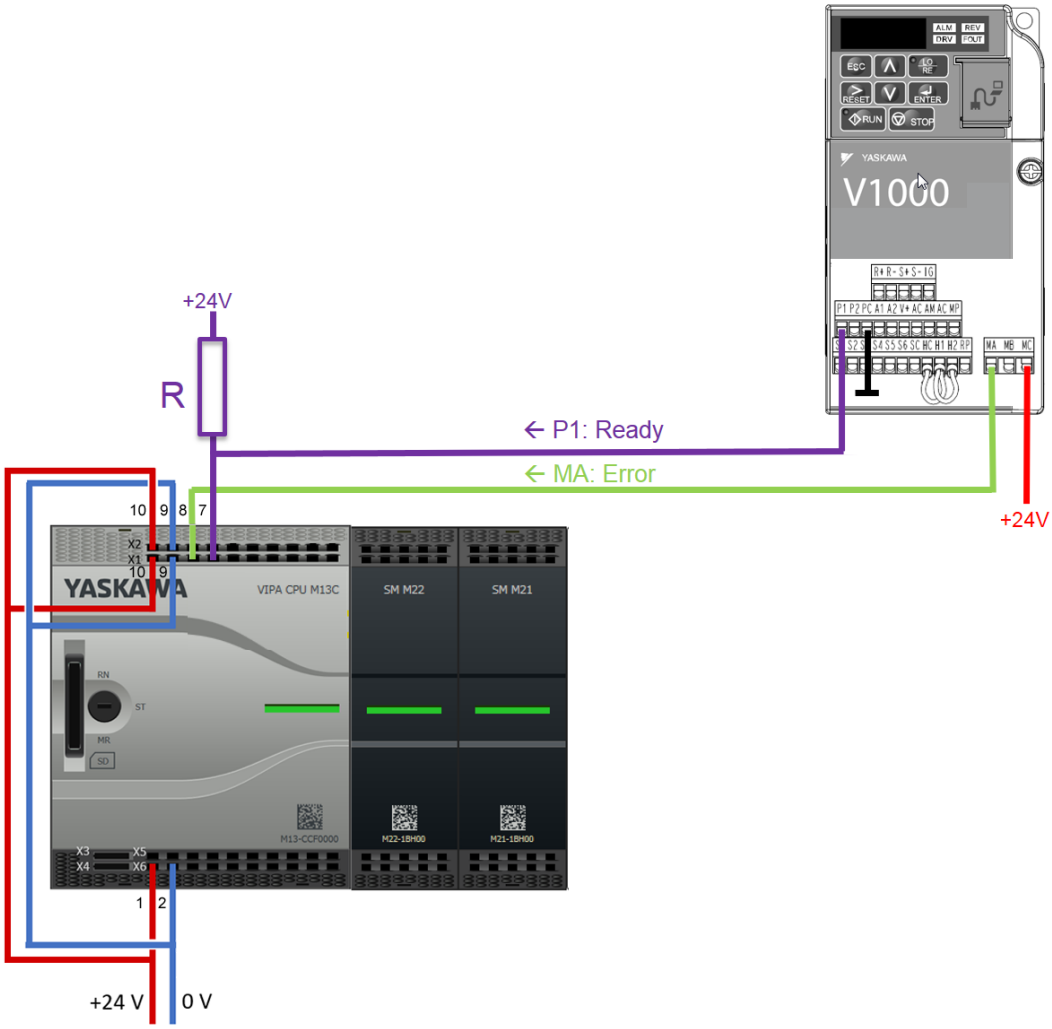


R Widerstand
Wert: max. 470Ω
Verlustleistung: min. 0,6W
Widerstandsbeispiel: Metallschichtwiderstand 0207
bedrahtet mit 0,6W Verlustleistung
Kabellänge max. 20m

15.5.3.2 Ansteuerung *V1000* Ausgänge

Beispielapplikation

Die nachfolgende Abbildung zeigt eine Beispielapplikation zum Anschluss der Ausgänge eines V1000 Frequenzumrichters an eine System MICRO CPU M13C.



R Widerstand
Wert: 4,7kΩ
Verlustleistung: min. 0,25W
Widerstandsbeispiel: Kohleschichtwiderstand 0207 bedrahtet mit 0,25W Verlustleistung

15.5.4 Einsatz im VIPA *SPEED7 Studio*

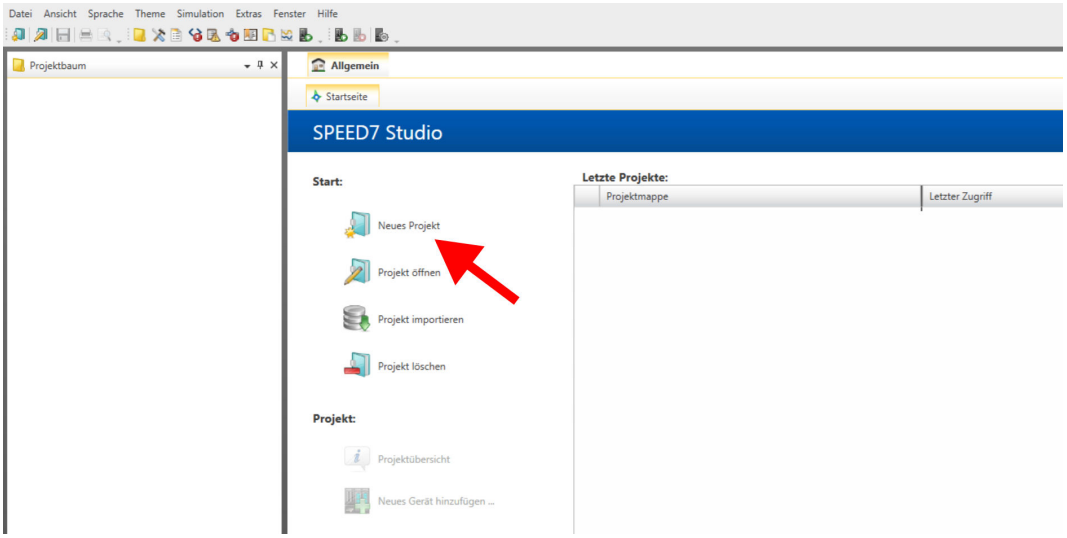
15.5.4.1 Hardware-Konfiguration

CPU im Projekt anlegen

Bitte verwenden Sie für die Projektierung das *SPEED7 Studio* ab V1.7.1

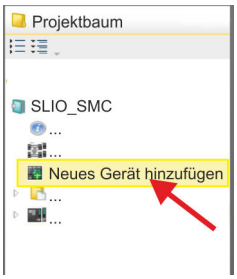
Sollten Sie einen anderen Kanal als Kanal 0 verwenden, müssen Sie diesen in der Hardware-Konfiguration und in Ihrem Anwenderprogramm entsprechend anpassen.

1. Starten Sie das *SPEED7 Studio*.

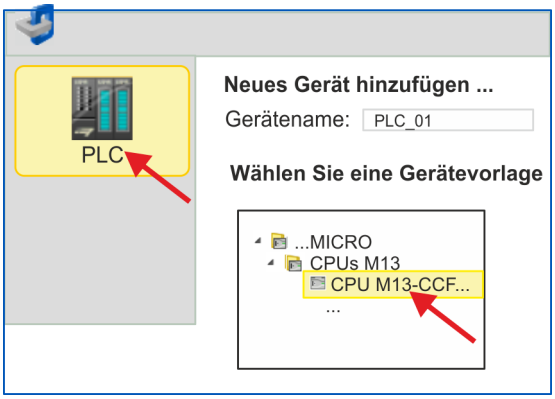


2. Erstellen sie auf der Startseite mit *"Neues Projekt"* ein neues Projekt und vergeben Sie einen *"Projektnamen"*.

⇨ Ein neues Projekt wird angelegt und in die Sicht *"Geräte und Netze"* gewechselt.



3. Klicken Sie im *Projektbaum* auf *"Neues Gerät hinzufügen ..."*.



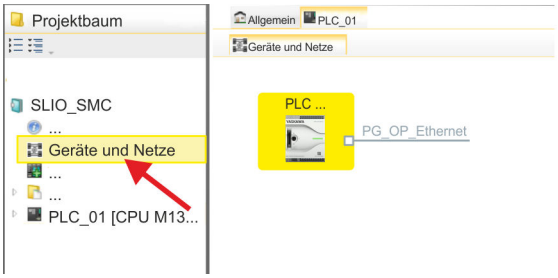
⇨ Es öffnet sich ein Dialog für die Geräteauswahl.

4. Wählen Sie unter den *"Gerätevorlagen"* Ihre CPU mit PWM Funktionalität wie z.B. die System MICRO CPU M13-CCF0000 und klicken Sie auf [OK].

⇨ Die CPU wird in *"Geräte und Netze"* eingefügt und die *"Gerätekonfiguration"* geöffnet.

Ethernet-PG/OP-Kanal parametrieren

1. Klicken Sie im *Projektbaum* auf *"Geräte und Netze"*.
   ⇨ Sie erhalten eine grafische Objekt-Ansicht Ihrer CPU.



2. Klicken Sie auf das Netzwerk *"PG_OP_Ethernet"*.
3. Wählen Sie *"Kontextmenü ➔ Eigenschaften der Schnittstelle"*.
   ⇨ Es öffnet sich ein Dialogfenster. Hier können Sie IP-Adressdaten für Ihren Ethernet-PG/OP-Kanal angeben. Gültige IP-Adress-Parameter erhalten Sie von Ihrem Systemadministrator.
4. Bestätigen Sie Ihre Eingabe mit [OK].
   ⇨ Die IP-Adressdaten werden in Ihr Projekt übernommen und in *"Geräte und Netze"* unter *"Lokale Baugruppen"* aufgelistet.

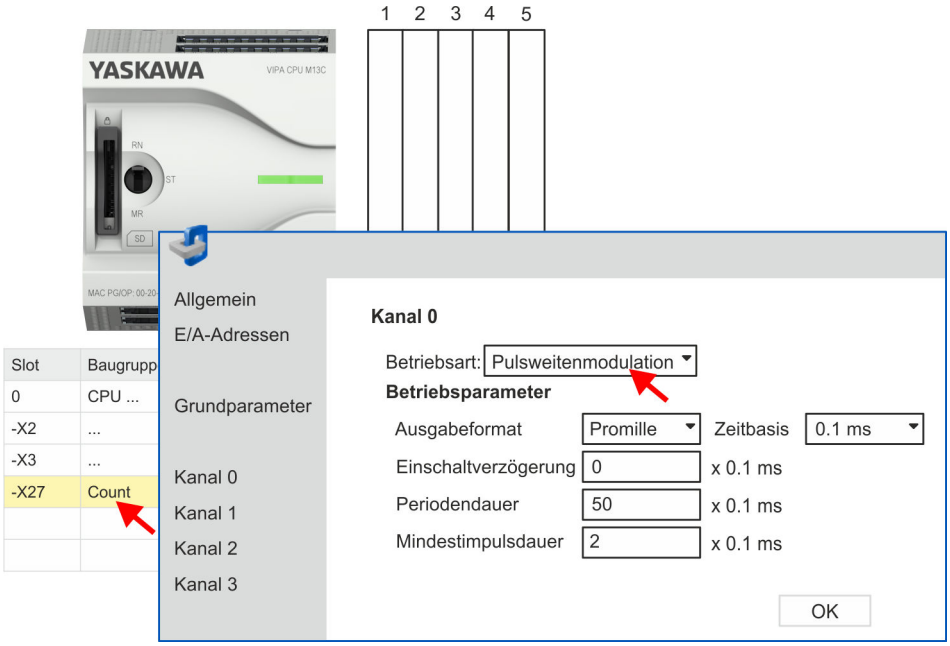
   Nach der Übertragung Ihres Projekts ist Ihre CPU über die angegebenen IP-Adressdaten via Ethernet-PG/OP-Kanal erreichbar.

E/A-Peripherie auf PWM umschalten

Für die Parametrierung der Ein-/Ausgabeperipherie und der *Technologischen Funktionen* sind die entsprechenden Submodule der CPU zu verwenden. Für die PWM Ausgabe ist das Submodul Count auf *"Pulsweitenmodulation"* umzuschalten.

1. Klicken Sie im *Projektbaum* auf *"PLC... > Gerätekonfiguration"*.
2. Klicken Sie in der *"Gerätekonfiguration"* auf *"-X27 Count"* und wählen Sie *"Kontextmenü ➔ Eigenschaften der Baugruppe"*.
   ⇨ Es öffnet sich der Eigenschaften-Dialog.
3. Klicken Sie auf den entsprechenden Kanal wie z.B. *"Kanal 0"* und stellen Sie unter *"Betriebsart"* die Funktion *"Pulsweitenmodulation"* ein.

4. Die für PWM erforderlichen Betriebsparameter werden intern auf die entsprechenden Werte angepasst. Lassen Sie alle Werte unverändert.

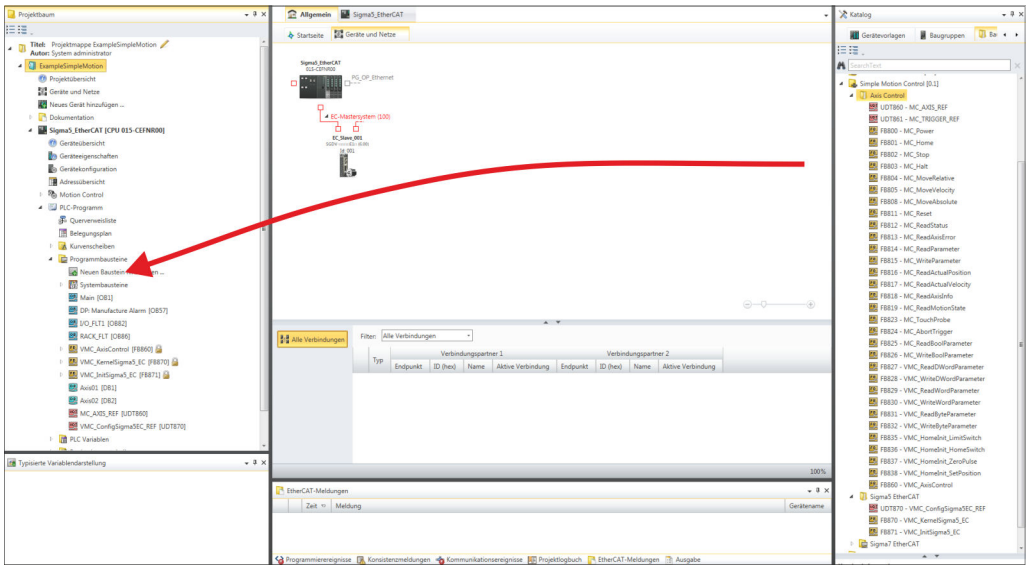


5. Schließen Sie den Dialog mit [OK].

6. Wählen Sie *"Projekt ➔ Alles übersetzen"*.

15.5.4.2 Anwender-Programm

Baustein in Projekt kopieren



Öffnen Sie im *"Katalog"* unter *"Bausteine"* *"Simple Motion Control"* und ziehen Sie per Drag&Drop folgenden Baustein in *"Programmbausteine"* des *Projektbaums*:

- V1000 PWM
  - FB885 – VMC_AxisControlV1000PWM ↳ *Kap. 15.5.5.1 "FB 885 - VMC_AxisControlV1000_PWM - Achskontrolle über PWM" Seite 702*

OB 1

Konfiguration der Achse

Sollten Sie einen anderen Kanal als Kanal 0 verwenden, müssen Sie diesen in der Hardware-Konfiguration und in Ihrem Anwenderprogramm entsprechend anpassen.

1. Öffnen Sie in *"Projektbaum ➔ ...CPU... ➔ PLC-Programm ➔ Programmbausteine"* den OB 1 und programmieren Sie den Call FB 885, DB 885.

   ⇨ Es öffnet sich der Dialog *"Instanz-Datenbaustein hinzufügen"*.

2. Stellen Sie, wenn nicht schon geschehen, die Nummer für den Instanz-Datenbaustein ein und schließen Sie den Dialog mit [OK].

   ⇨ Der Bausteinaufruf wird angelegt und die Parameter werden aufgelistet.

3. Vergeben Sie für das Beispielprojekt folgende Parameter:

   ⇨
```
CALL  FB   "VMC_AxisControlV1000PWM" ,
"VMC_AxisCtrlV1000PWM_885"
       I_ChannelNumberPWM  :="Ax1_I_ChannelNumberPWM"
       I_MA_Alarm          :="Ax1_MA_Alarm"
       I_P1_Ready          :="I_P1_Ready"
       MaxVelocityDrive    :=1.000000e+002
       AxisEnable          :="Ax1_AxisEnable"
       AxisReset           :="Ax1_AxisReset"
       StopExecute         :="Ax1_StopExecute"
       MvVelocityExecute   :="Ax1_MvVelExecute"
       JogPositive         :="Ax1_JogPositive"
       JogNegative         :="Ax1_JogNegative"
       Velocity            :="Ax1_Velocity"
       I_S1_ForwardRun     :="Ax1_S1_ForwardRun"
       I_S2_ReverseRun     :="Ax1_S2_ReverseRun"
       I_S4_AlarmReset     :="Ax1_S4_AlarmReset"
       MinUserVelocity     :="Ax1_MinUserVelocity"
       MaxUserVelocity     :="Ax1_MaxUserVelocity"
       AxisReady           :="Ax1_AxisReady"
       AxisEnabled         :="Ax1_AxisEnabled"
       AxisError           :="Ax1_AxisError"
       AxisErrorID         :="Ax1_AxisErrorID"
       DriveError          :="Ax1_DriveError"
       CmdActive           :="Ax1_CmdActive"
       CmdDone             :="Ax1_CmdDone"
       CmdBusy             :="Ax1_CmdBusy"
       CmdAborted          :="Ax1_CmdAborted"
       CmdError            :="Ax1_CmdError"
       CmdErrorID          :="Ax1_CmdErrorID"
```

   Die Adressen von *I_P1_Ready* und *I_MA_Alarm* ergeben sich aus den Adressen der Eingänge, welche mit den Digitalausgängen des Frequenzumrichters verbunden sind. Diese können über das Submodul *"-X25 DI/DIO"* der CPU ermittelt werden.

   Die Adressen von *I_S1_ForwardRun*, *I_S2_ReverseRun* und *I_S4_AlarmReset* ergeben sich aus den Adressen der Ausgänge, welche mit den Digitaleingängen des Frequenzumrichters verbunden sind. Diese können über das Submodul *"-X25 DI/DIO"* der CPU ermittelt werden.

Zeitlicher Ablauf

1. Wählen Sie *"Projekt ➔ Alles übersetzen"* und übertragen Sie das Projekt in Ihre CPU.
   ⇨ Sie können jetzt Ihre Applikation in Betrieb nehmen.

VORSICHT!
Bitte beachten Sie immer die Sicherheitshinweise zu ihrem Frequenzumrichter, insbesondere bei der Inbetriebnahme!

2. Bringen Sie Ihre CPU in RUN und schalten Sie Ihren Frequenzumrichter ein.
   ⇨ Der FB 885 - VMC_AxisControlV1000PWM wird zyklisch abgearbeitet.
3. Sobald *AxisReady* = TRUE meldet, können Sie mit *AxisEnable* die Achse frei geben.
4. Sie haben jetzt die Möglichkeit über die entsprechenden Parameter Ihre Achse zu steuern und deren Status abzufragen. ↯ *Kap. 15.5.5.1 "FB 885 - VMC_AxisControlV1000_PWM - Achskontrolle über PWM" Seite 702*

15.5.5 Antriebsspezifischer Baustein

15.5.5.1 FB 885 - VMC_AxisControlV1000_PWM - Achskontrolle über PWM

15.5.5.1.1 Beschreibung

Mit dem FB *VMC_AxisControlV1000_PWM* können Sie einen über PWM angebundenen Frequenzumrichter steuern und dessen Status abrufen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
I_Channel-NumberPWM	INPUT	INT	Kanalnummer des PWM-Ausgangs, der für die Ansteuerung des PWM-Eingangs des Frequenzumrichters verwendet wird.
I_MA_Alarm	INPUT	BOOL	■ Digitaleingang zur Anbindung des *I_MA_Alarm*-Signals (MA) – TRUE: Der Frequenzumrichter hat einen Fehler erkannt.
I_P1_Ready	INPUT	BOOL	■ Digitaleingang zur Anbindung des *I_P1_Ready*-Signals – FALSE: Der Frequenzumrichter ist bereit.
MaxVelocity-Drive	INPUT	REAL	■ Maximale Geschwindigkeit des Frequenzumrichters [Anwendereinheiten]. ↯ *Kap. 15.5.5.1.2 "Berechnung" Seite 704*
AxisEnable	INPUT	BOOL	■ Achsenfreigabe – Dieser Parameter dient zur bausteininternen Freigabe und hat keinen Einfluss auf den Frequenzumrichter. – TRUE: Die Achse wird freigegeben. – FALSE: Die Achse wird gesperrt.
AxisReset	INPUT	BOOL	■ Reset Achse – Flanke 0-1: Reset der Achse wird durchgeführt. – Der Zustand eines mit *AxisReset* gestarteten Reset wird nicht an den Ausgängen *CmdActive*, *CmdDone*, *CmdBusy*, *CmdAborted*, *CmdError* und *CmdErrorID* ausgegeben.
StopExecute	INPUT	BOOL	■ Achse stoppen – Flanke 0-1: Stoppen der Achse wird gestartet. Hinweis: *StopExecute* = 1: Kein anderes Kommando kann gestartet werden!

Parameter	Deklaration	Datentyp	Beschreibung
MvVelocityExecute	INPUT	BOOL	■ Verfahren der Achse starten – Flanke 0-1: Die Achse wird auf die angegebene Geschwindigkeit beschleunigt/abgebremst.
JogPositive	INPUT	BOOL	Tipp-Betrieb positiv ■ Achse mit konstanter Geschwindigkeit in positive Richtung verfahren – Flanke 0-1: Das Verfahren der Achse mit konstanter Geschwindigkeit wird gestartet. – Flanke 1-0: Die Achse wird gestoppt.
JogNegative	INPUT	BOOL	Tipp-Betrieb negativ ■ Achse mit konstanter Geschwindigkeit in negative Richtung verfahren – Flanke 0-1: Das Verfahren der Achse mit konstanter Geschwindigkeit wird gestartet. – Flanke 1-0: Die Achse wird gestoppt.
Velocity	INPUT	REAL	Geschwindigkeitsvorgabe (vorzeichenbehafteter Wert) in [Anwendereinheiten/s]. Hinweis: Bei *JogPositive*, *JogNegative* wird der absolute Wert der Geschwindkeit verwendet.
I_S1_ForwardRun	OUTPUT	BOOL	■ Digitalausgang zur Ansteuerung des Frequenzumrichter-Signals S1 – TRUE: Schaltet Frequenzumrichter in positive Richtung frei.
I_S2_ReverseRun	OUTPUT	BOOL	■ Digitalausgang zur Ansteuerung des Frequenzumrichter-Signals S2 – TRUE: Schaltet Frequenzumrichter in negative Richtung frei.
I_S4_Alarm-Reset	OUTPUT	BOOL	■ Digitalausgang zur Ansteuerung des Frequenzumrichter-Signals S4 – TRUE: Alarme werden im Frequenzumrichter zurückgesetzt. – FALSE: Alarme im Frequenzumrichter bleiben bestehen.
MinUserVelocity	OUTPUT	REAL	Minimale Geschwindigkeit (Periodendauer = 65535µs = maximale Periodendauer des PWM-Ausgangs) des Frequenzumrichters [Anwendereinheiten].
MaxUserVelocity	OUTPUT	REAL	Maximale Geschwindigkeit bei einer maximalen Frequenz von 20kHz des Frequenzumrichters [Anwendereinheiten].
AxisReady	OUTPUT	BOOL	■ AxisReady – TRUE: Die Achse ist einschaltbereit. – FALSE: Die Achse ist nicht einschaltbereit. → Prüfe und behebe *AxisError* (siehe *AxisErrorID*). → Prüfe und behebe *DriveError*.
AxisEnabled	OUTPUT	BOOL	■ Status Achse – TRUE: Achse ist eingeschaltet und nimmt Bewegungsaufträge an. – FALSE: Achse ist nicht eingeschaltet und nimmt keine Bewegungsaufträge an.

Parameter	Deklaration	Datentyp	Beschreibung
AxisError	OUTPUT	BOOL	■ Fehler an Achse – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *AxisErrorID* entnommen werden. → Die Achse wird gesperrt (*S_On* = FALSE und *AxisEnabled* = FALSE) Kommando wird nicht ausgeführt.
AxisErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
DriveError	OUTPUT	BOOL	■ Fehler direkt am Frequenzumrichter – TRUE: Ein Fehler ist aufgetreten. → Die Achse wird gesperrt.
CmdActive	OUTPUT	BYTE	■ Kommando – 0: kein Cmd aktiv – 1: STOP – 2: MvVelocity – 4: JogPos – 5: JogNeg
CmdDone	OUTPUT	BOOL	■ Status Done – TRUE: Auftrag wurde ohne Fehler beendet.
CmdBusy	OUTPUT	BOOL	■ Status Busy – TRUE: Auftrag ist in Bearbeitung.
CmdAborted	OUTPUT	BOOL	■ Status Aborted – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen. Hinweis: *CmdAborted* wird beim Start eines Cmd zurückgesetzt
CmdError	OUTPUT	BOOL	■ Status Error – TRUE: Ein Fehler ist aufgetreten. Die Achse wird gesperrt Zusätzliche Fehlerinformationen können dem Parameter *CmdErrorID* entnommen werden.
CmdErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*

VORSICHT!
Bitte beachten Sie, dass der Baustein keinen CPU-Neustart erkennt. Um zu verhindern, dass bei einem CPU-Neustart die Achse ungewollt anläuft, sollten mittels des Anlauf-OBs z.B. OB 100 die Werte an den Eingängen *AxisEnable*, *JogPositive* und *JogNegative* auf FALSE gesetzt werden!

15.5.5.1.2 Berechnung

MaxVelocityDrive Dieser Wert dient zur Normierung des Eingangswerts *Velocity*.

$$n = 2 \cdot 60 \cdot \frac{fmax,out}{poles} \frac{1}{min}$$

$f_{max, out}$ - Maximale Frequenz (Parameter E1-04)

poles - Anzahl der Motor-Pole (Parameter E5-04)

n - Maximale Geschwindigkeit des Frequenzumrichters [Anwendereinheiten] wie z.B 100.0 % oder 3000.0 Umdrehungen/min.

15.5.5.1.3 Funktionalität

Achse ein- bzw. ausschalten

- Der Eingang *AxisEnable* dient zum Ein- und Ausschalten einer Achse.
- Das Einschalten ist nur möglich, wenn *AxisReady* = TRUE meldet, d.h. die Achse einschaltbereit ist.
- Sobald die Achse eingeschaltet ist, wird dies durch die Statusinformation *AxisEnabled* angezeigt.
- Hat die Achse einen Fehler, wird dies durch die Statusinformation *AxisError* angezeigt. Weitere Informationen liefert *AxisErrorID*.

Achsfehler quittieren

- Mit *AxisReset* können Sie Achsfehler quittieren.
- Anliegende Fehler werden über *DriveError* zurück gemeldet.

Achse stoppen

- Eine Achse in Bewegung können Sie durch Setzen von *StopExecute* stoppen.
- Solange *StopExecute* gesetzt ist, werden keine weiteren Pulse generiert und alle Kommandos sind gesperrt.

Geschwindigkeitsmodus

- Voraussetzung: Die Achse ist eingeschaltet und *AxisReady* = TRUE.
- Mit *MvVelocityExecute* können Sie die Achse zum Drehen mit konstanter Geschwindigkeit bringen.
- Die Geschwindigkeit geben Sie über *Velocity* vor.
- Durch Vorgabe von 0 stoppt die Achse ebenso wie mit *StopExecute*.
- Die Drehrichtung bestimmen Sie über das Vorzeichen von *Velocity*.
- Der Wert für *Velocity* darf 0 sein oder *MinUserVelocity* ≤ *Velocity* ≤ *MaxUserVelocity*.

Tipp-Betrieb - Jogging

- Voraussetzung: Die Achse ist eingeschaltet und *AxisReady* = TRUE.
- Mit einer Flanke 0-1 an *JogPositive* oder *JogNegative* können Sie Ihre Achse im Tipp-Betrieb steuern. Hierbei wird ein Jogging Kommando in die entsprechende Drehrichtung ausgeführt.
- Die Geschwindigkeit geben Sie über *Velocity* vor. Das Vorzeichen ist nicht relevant.
- Mit einer Flanke 1-0 an *JogPositive* oder *JogNegative* bzw. durch Setzen von *StopExecute* wird die Achse gestoppt.

15.6 Einsatz Frequenzumrichter über Modbus RTU

15.6.1 Übersicht

Voraussetzung

- SPEED7 Studio ab V1.7.1
- System MICRO bzw. System SLIO CPU mit serieller Schnittstelle wie z.B. CPU M13-CCF0000 bzw. CPU 013-CCF0R00.
- *V1000*-Frequenzumrichter mit serieller Schnittstelle und zugehörigem Motor

Schritte der Projektierung

1. Parameter am Frequenzumrichter einstellen
   - Die Einstellung der Parameter hat mit dem Softwaretool *Drive Wizard+* zu erfolgen.
2. Hardwarekonfiguration im VIPA *SPEED7 Studio.*
   - Projektierung der CPU.
3. Programmierung im VIPA *SPEED7 Studio.*
   - Baustein zur seriellen Kommunikation beschalten.
   - Baustein für jeden Modbus-Slave beschalten.
   - Baustein für die Kommunikationsdaten aller Modbus-Slaves beschalten.
   - Baustein für den Kommunikations-Manager beschalten.
   - Baustein zur Initialisierung des Frequenzumrichters beschalten.
   - Bausteine für die Bewegungsabläufe beschalten.

15.6.2 Parameter am Frequenzumrichter einstellen

VORSICHT!
Vor der Erstinbetriebnahme müssen Sie Ihren Frequenzumrichter mit dem Softwaretool *Drive Wizard+* an Ihre Applikation anpassen! Näheres hierzu finden Sie im Handbuch zu ihrem Frequenzumrichter.

Die nachfolgende Tabelle zeigt alle Parameter auf, die nicht den Standardwerten entsprechen. Zur Abstimmung auf die *Simple Motion Control Library* sind diese über *Drive Wizard+* einzustellen:

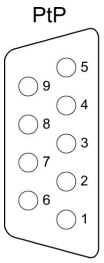
Nr.	Bezeichnung	Wertebereich	Einstellung für *Simple Motion Control Library*
H5-01	Slave-Adresse Frequenzumrichter	00h ... 20h	Standardmäßig ist die Slave-Adresse auf 1Fh eingestellt. Bitte beachten Sie, dass Adressen im Netzwerk nicht mehrfach vergeben sein dürfen!
H5-02	Kommunikationsgeschwindigkeit MEMOBUS/Modbus	0, 1, 2, ..., 8	■ 3: 9600Bit/s
H5-03	Übertragungsparität MEMOBUS/ Modbus	0, 1, 2	■ 0: keine Parität
H5-04	Stoppmethode nach Kommunikations-Fehler (CE-Fehler)	0, 1, 2, 3	■ 3: Betrieb wird fortgesetzt mit Alarm
H5-05	Erkennung Kommunikations-Fehler (CE-Fehler)	0, 1	■ 1: Aktiviert - Bei einem Abbruch der Verbindung für länger als 2s (einstellbar über *H2-09*) wird ein CE-Fehler ausgelöst.
H5-06	Wartezeit zwischen Empfang und Senden von Daten des Frequenzumrichters	5 ... 65ms	■ 5ms
H5-07	Request to send (RTS)-Steuerung	0, 1	■ 1: Aktiviert - RTS wird nur beim Senden aktiviert (bei RS485 oder RS422 und *multi-drop*)
H5-09	Zeit nach der ein Kommunikations-Fehler (CE-Fehler) erkannt wird.	0,0 ... 10,0s	■ 2s

Nr.	Bezeichnung	Wertebereich	Einstellung für *Simple Motion Control Library*
H5-10	Schrittgröße (Auflösung) für das MEMOBUS/Modbus-Register 0025h	0, 1	Standardmäßig ist die Auflösung auf 0,1V Inkremente (0) eingestellt. ■ 0: 0,1V Inkremente ■ 1: 1V Inkremente
H5-11	ENTER-Funktion für Verbindungen	0 , 1	■ 1: Enter-Befehl nicht erforderlich
H5-12	Auswahl Startbefehlmethode	0, 1	■ 1: Run/Stop
B1-01	Eingangsquelle Frequenzsollwert 1	0, 1, 2, 3, 4	■ 2: MEMOBUS/Modbus-Kommunikation
B1-02	Eingangsquelle Startbefehl 1	0, 1, 2, 3	■ 2: MEMOBUS/Modbus-Kommunikation
B1-15	Eingangsquelle Frequenzsollwert 2	0, 1, 2, 3, 4	■ 2: MEMOBUS/Modbus-Kommunikation
B1-16	Eingangsquelle Startbefehl 2	0, 1, 2, 3	■ 2: MEMOBUS/Modbus-Kommunikation

Damit alle Einstellungen übernommen werden, müssen Sie den Frequenzumrichter nach der Parametrierung neu starten!

15.6.3 Beschaltung

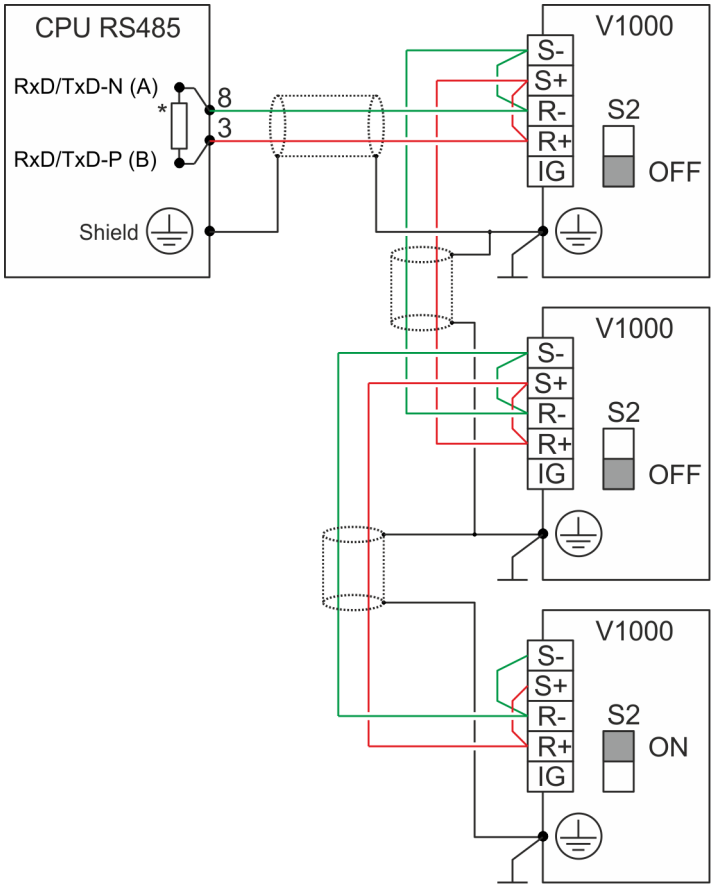
RS485-Verkabelung



① n.c.
② M24V
③ RxD/TxD-P (line B)
④ RTS
⑤ M5V
⑥ P5V
⑦ P24V
⑧ RxD/TxD-N (line A)
⑨ n.c.

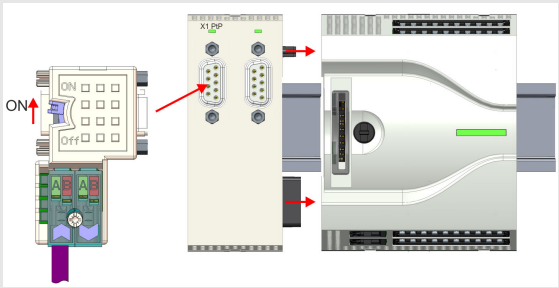
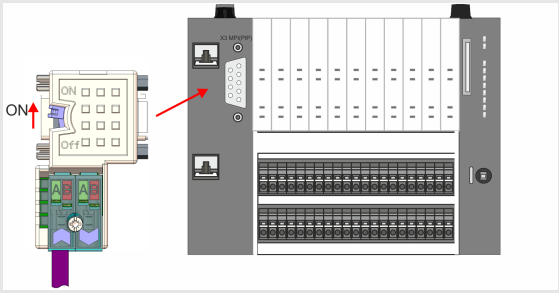
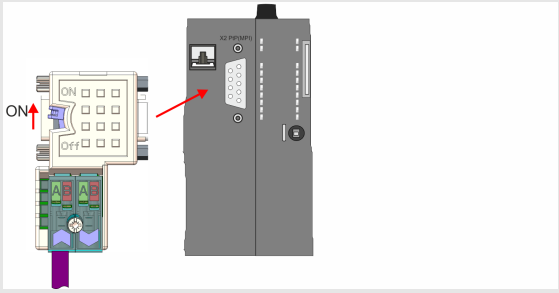
Die nachfolgende Abbildung zeigt den Anschluss von *V1000*-Frequenzumrichter über RS485. Hierbei werden die einzelnen Frequenzumrichter über PROFIBUS-Kabel verbunden und über einen PROFIBUS-Stecker an die PtP-Schnittstelle (**P**oint-**t**o-**P**oint) der CPU angebunden.

- Maximal 8 Frequenzumrichter können über Modbus RTU angebunden werden.
- Bei allen angebunden Frequenzumrichter ist der Parameter H5-07 auf 1 zu setzen.
- Die serielle Leitung ist an Ihrem Ende mit einem Abschlusswiderstand abzuschließen. Zur Aktivierung müssen Sie hierzu am entsprechenden Frequenzumrichter den Schalter S2 auf *"ON"* stellen.

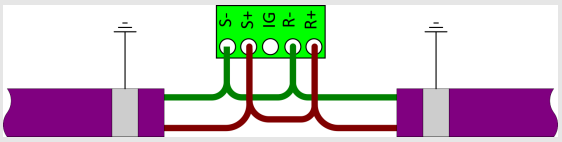
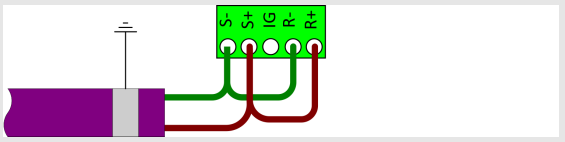
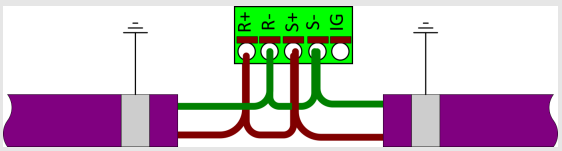
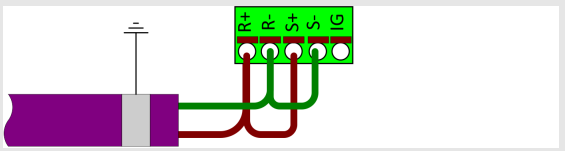
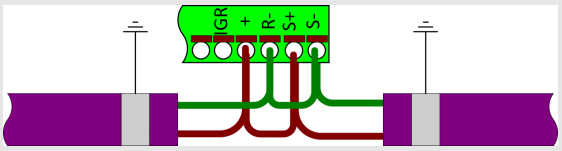
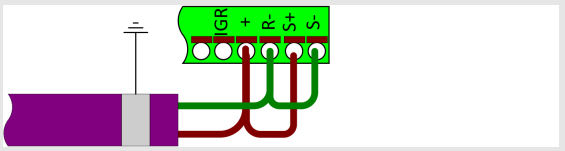
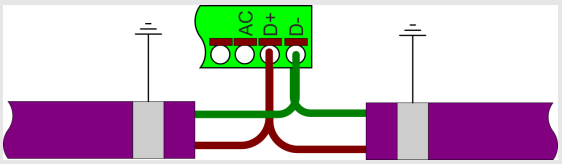
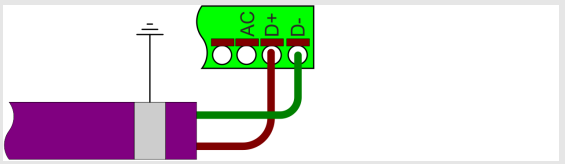


– **) Verwenden Sie für einen störungsfreien Datenverkehr einen Abschlusswiderstand von ca. 120Ω an der CPU, wie z.B. beim VIPA-PROFIBUS-Stecker.*
– *Verbinden Sie niemals Kabelschirm und M5V (Pin 5) miteinander, da aufgrund von Ausgleichsströmen die Schnittstellen zerstört werden könnten!*

Anschluss der CPU

CPU	Anschluss	Kommentar
MICRO CPU M13C		■ Für die PtP-Kommunikation ist der Einsatz des optional erhältlichen Erweiterungsmoduls EM M09 erforderlich. ■ Das Erweiterungsmodul stellt die Schnittstelle X1: PtP (RS422/485) mit fixer Pinbelegung zur Verfügung. ■ Verwenden Sie zur Anbindung an die CPU einen VIPA-PROFIBUS-Stecker. ■ Aktivieren Sie am PROFIBUS-Stecker den Abschlusswiderstand. ■ Nach Einschalten der Spannungsversorgung und kurzer Anlaufzeit ist die CPU bereit für die PtP-Kommunikation.
System SLIO CPU 013C		■ Die CPU besitzt die Schnittstelle X3 MPI(PtP) mit fixer Pinbelegung. ■ Verwenden Sie zur Anbindung an die CPU einen VIPA-PROFIBUS-Stecker. ■ Aktivieren Sie am PROFIBUS-Stecker den Abschlusswiderstand. ■ Nach Einschalten der Spannungsversorgung und kurzer Anlaufzeit bzw. nach dem Urlöschen hat die Schnittstelle MPI-Funktionalität. Über die Hardware-Konfiguration können Sie die PtP-Funktionalität aktivieren. *Kap. 15.6.4 "Einsatz im VIPA SPEED7 Studio" Seite 711*
System SLIO CPU 014 ... 017		■ Die CPU besitzt die Schnittstelle X2 PtP(MPI) die standardmäßig auf PtP-Kommunikation (point to point) eingestellt ist. ■ Verwenden Sie zur Anbindung an die CPU einen VIPA-PROFIBUS-Stecker. ■ Aktivieren Sie am PROFIBUS-Stecker den Abschlusswiderstand. ■ Nach Einschalten der Spannungsversorgung und kurzer Anlaufzeit ist die CPU bereit für die PtP-Kommunikation.

Anschluss der YASKAWA Frequenzumrichter

FU	Anschluss Fortlaufend	Anschluss Abschluss
J1000		
V1000		
A1000		
GA700		

Näheres hierzu finden Sie im entsprechenden Handbuch.

15.6.4 Einsatz im VIPA *SPEED7 Studio*

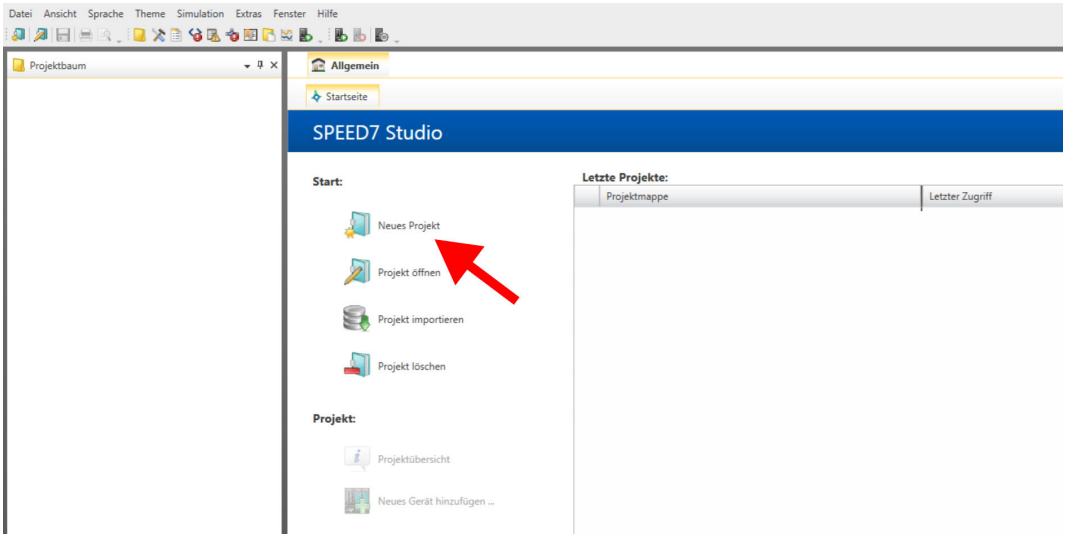
15.6.4.1 Hardware-Konfiguration

15.6.4.1.1 Hardware-Konfiguration System MICRO

CPU im Projekt anlegen

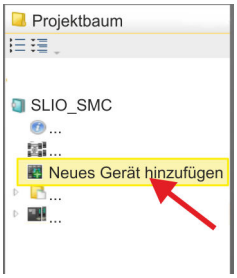
Bitte verwenden Sie für die Projektierung das *SPEED7 Studio* ab V1.7.1

1. Starten Sie das *SPEED7 Studio*.

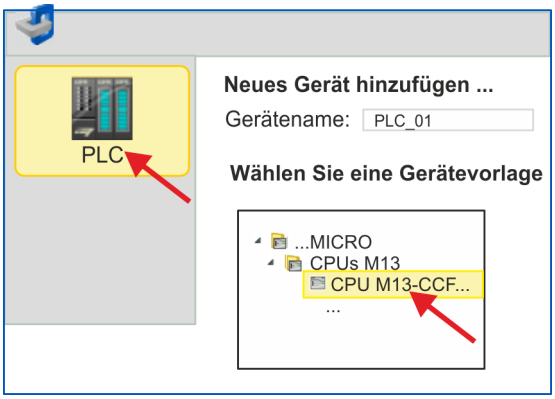


2. Erstellen sie auf der Startseite mit *"Neues Projekt"* ein neues Projekt und vergeben Sie einen *"Projektnamen"*.

⇨ Ein neues Projekt wird angelegt und in die Sicht *"Geräte und Netze"* gewechselt.



3. Klicken Sie im *Projektbaum* auf *"Neues Gerät hinzufügen ..."*.



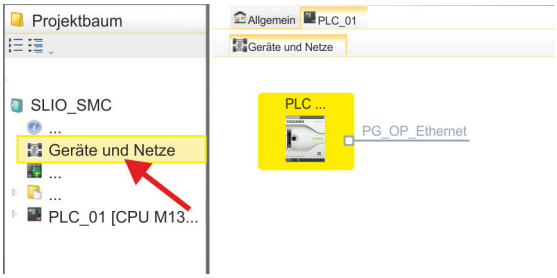
⇨ Es öffnet sich ein Dialog für die Geräteauswahl.

4. Wählen Sie unter den *"Gerätevorlagen"* die System MICRO CPU M13-CCF0000 und klicken Sie auf [OK].

⇨ Die CPU wird in *"Geräte und Netze"* eingefügt und die *"Gerätekonfiguration"* geöffnet.

Ethernet-PG/OP-Kanal parametrieren

1. Klicken Sie im *Projektbaum* auf *"Geräte und Netze"*.
   ⇨ Sie erhalten eine grafische Objekt-Ansicht Ihrer CPU.

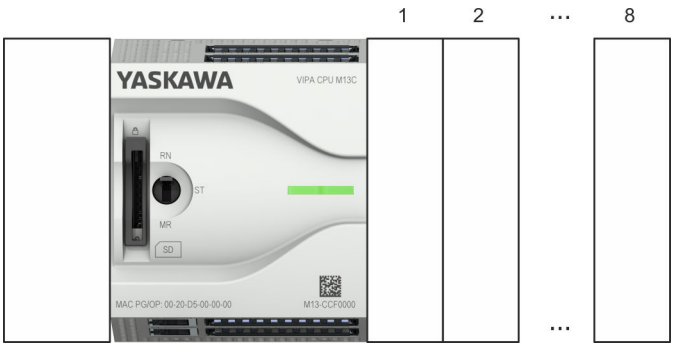


2. Klicken Sie auf das Netzwerk *"PG_OP_Ethernet"*.
3. Wählen Sie *"Kontextmenü ➔ Eigenschaften der Schnittstelle"*.
   ⇨ Es öffnet sich ein Dialogfenster. Hier können Sie IP-Adressdaten für Ihren Ethernet-PG/OP-Kanal angeben. Gültige IP-Adress-Parameter erhalten Sie von Ihrem Systemadministrator.
4. Bestätigen Sie Ihre Eingabe mit [OK].
   ⇨ Die IP-Adressdaten werden in Ihr Projekt übernommen und in *"Geräte und Netze"* unter *"Lokale Baugruppen"* aufgelistet.

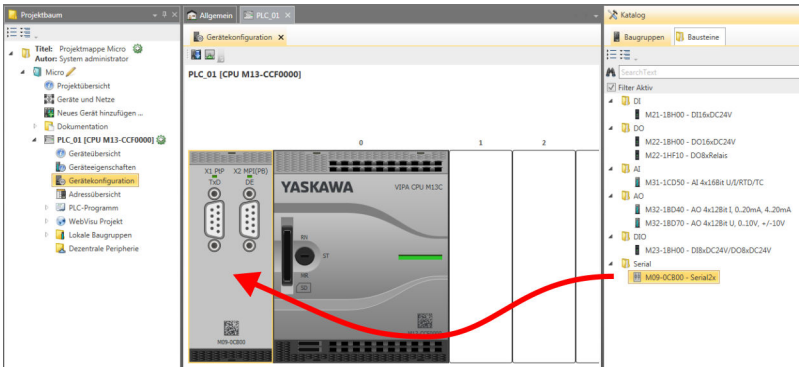
   Nach der Übertragung Ihres Projekts ist Ihre CPU über die angegebenen IP-Adressdaten via Ethernet-PG/OP-Kanal erreichbar.

PtP-Funktionalität aktivieren

1. Klicken Sie im *Projektbaum* auf *"PLC..CPU M13.... ➔ Gerätekonfiguration"*.
   ⇨ Die *"Gerätekonfiguration"* wird geöffnet.



2. Öffnen Sie im *"Katalog"* unter *"Baugruppen"* die Sammlung *"Seriell"* und ziehen Sie per Drag&Drop das serielle Modul *"M09-0CB00 - Serial2x"* auf den linken Steckplatz der CPU. Per Default ist die Schnittstelle X1 auf PtP-Funktionalität eingestellt.

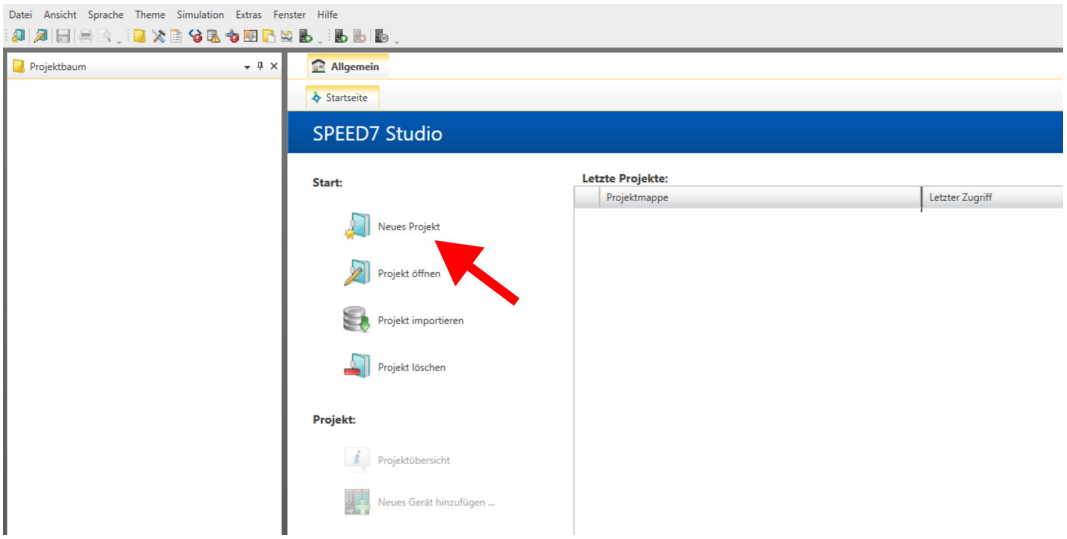


15.6.4.1.2 Hardware-Konfiguration System SLIO CPU 013C

CPU im Projekt anlegen

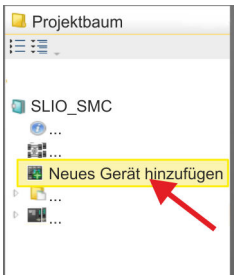
Bitte verwenden Sie für die Projektierung das *SPEED7 Studio* ab V1.7.1

1. Starten Sie das *SPEED7 Studio*.

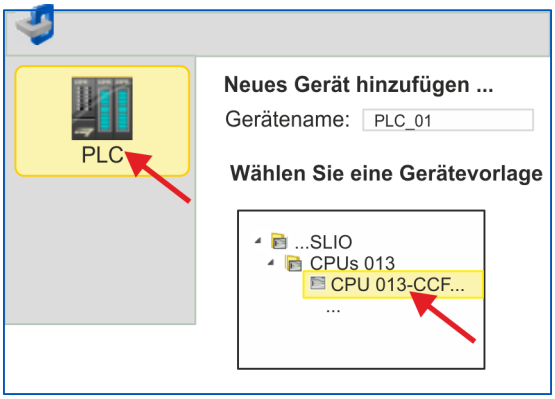


2. Erstellen sie auf der Startseite mit *"Neues Projekt"* ein neues Projekt und vergeben Sie einen *"Projektnamen"*.

⇨ Ein neues Projekt wird angelegt und in die Sicht *"Geräte und Netze"* gewechselt.



3. Klicken Sie im *Projektbaum* auf *"Neues Gerät hinzufügen ..."*.



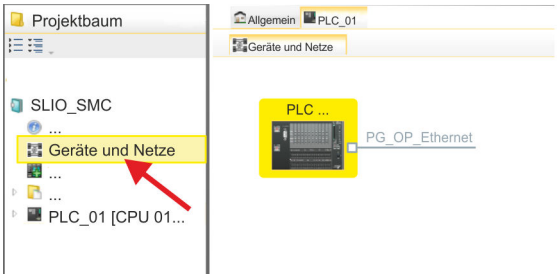
⇨ Es öffnet sich ein Dialog für die Geräteauswahl.

4. Wählen Sie unter den *"Gerätevorlagen"* die System SLIO CPU 013-CCF0R00 und klicken Sie auf [OK].

⇨ Die CPU wird in *"Geräte und Netze"* eingefügt und die *"Gerätekonfiguration"* geöffnet.

Ethernet-PG/OP-Kanal parametrieren

1. Klicken Sie im *Projektbaum* auf *"Geräte und Netze"*.
   ⇨ Sie erhalten eine grafische Objekt-Ansicht Ihrer CPU.

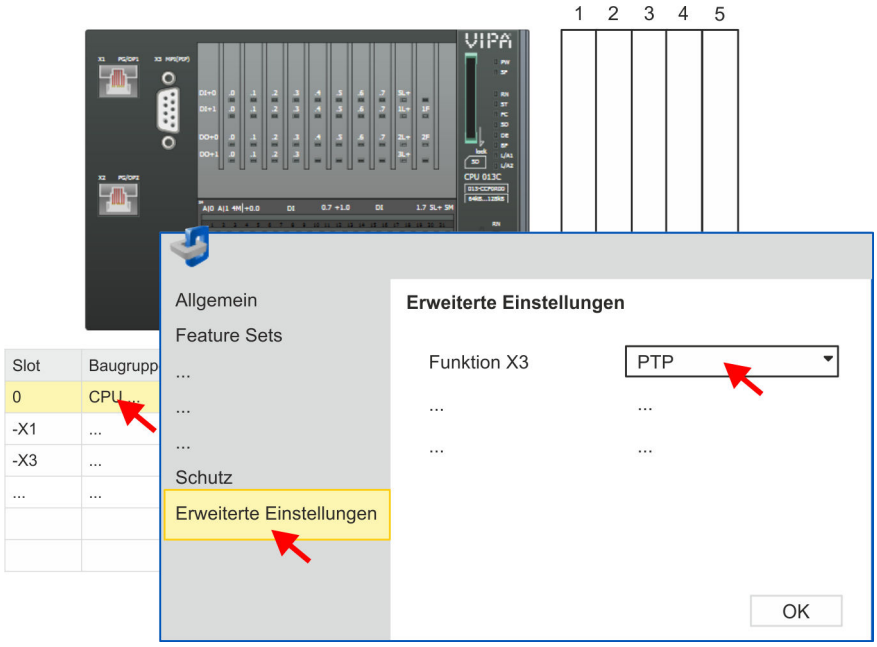


2. Klicken Sie auf das Netzwerk *"PG_OP_Ethernet"*.
3. Wählen Sie *"Kontextmenü ➔ Eigenschaften der Schnittstelle"*.
   ⇨ Es öffnet sich ein Dialogfenster. Hier können Sie IP-Adressdaten für Ihren Ethernet-PG/OP-Kanal angeben. Gültige IP-Adress-Parameter erhalten Sie von Ihrem Systemadministrator.
4. Bestätigen Sie Ihre Eingabe mit [OK].
   ⇨ Die IP-Adressdaten werden in Ihr Projekt übernommen und in *"Geräte und Netze"* unter *"Lokale Baugruppen"* aufgelistet.

   Nach der Übertragung Ihres Projekts ist Ihre CPU über die angegebenen IP-Adressdaten via Ethernet-PG/OP-Kanal erreichbar.

PtP-Funktionalität aktivieren

1. Klicken Sie im *Projektbaum* auf *"PLC... > Gerätekonfiguration"*.
2. Klicken Sie in der *"Gerätekonfiguration"* auf *"0 CPU 013..."* und wählen Sie *"Kontextmenü ➔ Eigenschaften der Baugruppe"*.
   ⇨ Es öffnet sich der Eigenschaften-Dialog.



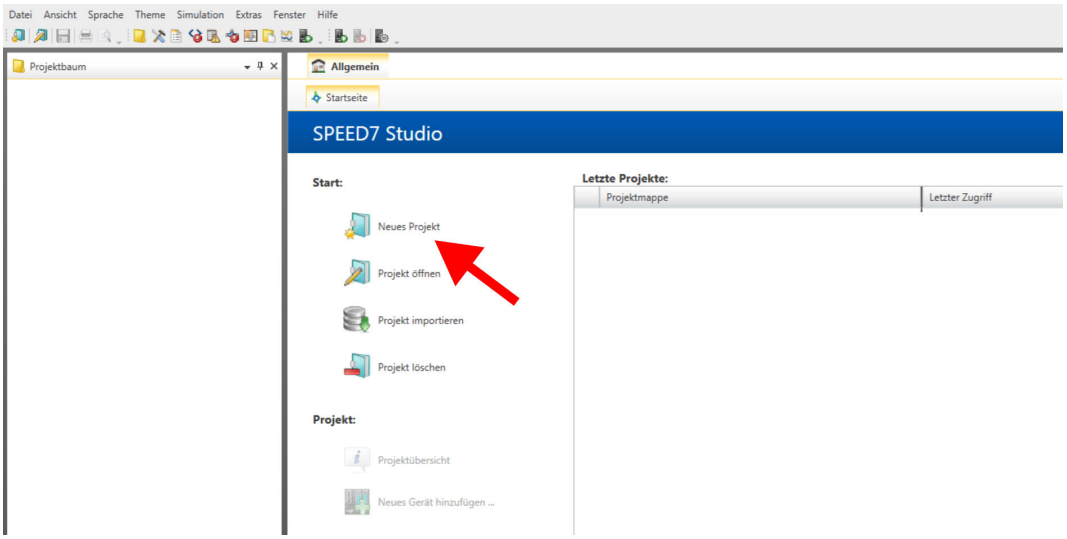
3. Klicken Sie auf *"Erweiterte Einstellungen"* und stellen Sie unter *"Funktion X3"* die Funktion *"PTP"* ein.

15.6.4.1.3 Hardware-Konfiguration System SLIO CPU 014 ... 017

CPU im Projekt anlegen

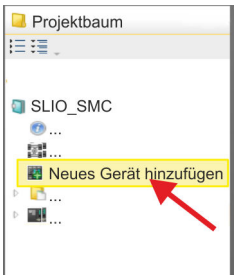
Bitte verwenden Sie für die Projektierung das *SPEED7 Studio* ab V1.7.1

1. Starten Sie das *SPEED7 Studio*.

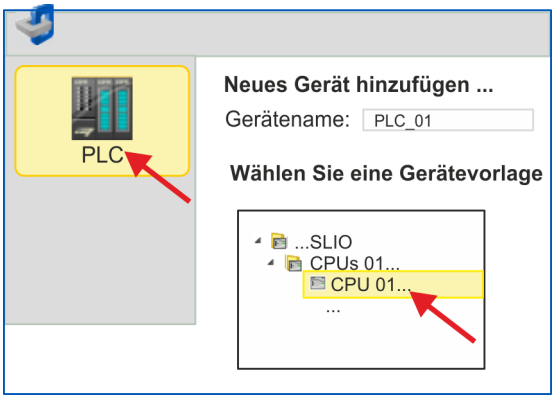


2. Erstellen sie auf der Startseite mit *"Neues Projekt"* ein neues Projekt und vergeben Sie einen *"Projektnamen"*.

⇨ Ein neues Projekt wird angelegt und in die Sicht *"Geräte und Netze"* gewechselt.



3. Klicken Sie im *Projektbaum* auf *"Neues Gerät hinzufügen ..."*.



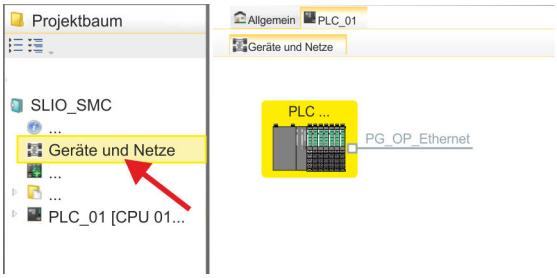
⇨ Es öffnet sich ein Dialog für die Geräteauswahl.

4. Wählen Sie unter den *"Gerätevorlagen"* die entsprechende System SLIO CPU aus und klicken Sie auf [OK].

⇨ Die CPU wird in *"Geräte und Netze"* eingefügt und die *"Gerätekonfiguration"* geöffnet.

Ethernet-PG/OP-Kanal parametrieren

1. Klicken Sie im *Projektbaum* auf *"Geräte und Netze"*.
   ⇨ Sie erhalten eine grafische Objekt-Ansicht Ihrer CPU.



2. Klicken Sie auf das Netzwerk *"PG_OP_Ethernet"*.
3. Wählen Sie *"Kontextmenü ➔ Eigenschaften der Schnittstelle"*.
   ⇨ Es öffnet sich ein Dialogfenster. Hier können Sie IP-Adressdaten für Ihren Ethernet-PG/OP-Kanal angeben. Gültige IP-Adress-Parameter erhalten Sie von Ihrem Systemadministrator.
4. Bestätigen Sie Ihre Eingabe mit [OK].
   ⇨ Die IP-Adressdaten werden in Ihr Projekt übernommen und in *"Geräte und Netze"* unter *"Lokale Baugruppen"* aufgelistet.

   Nach der Übertragung Ihres Projekts ist Ihre CPU über die angegebenen IP-Adressdaten via Ethernet-PG/OP-Kanal erreichbar.

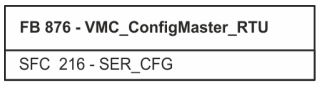
PtP-Funktionalität aktivieren

Bei den System SLIO CPUs 014 ... 017 ist die RS485-Schnittstelle standardmäßig auf PtP-Kommunikation eingestellt. Eine Hardware-Konfiguration zur Einstellung der PtP-Funktionalität ist nicht erforderlich.

15.6.4.2 Anwender-Programm

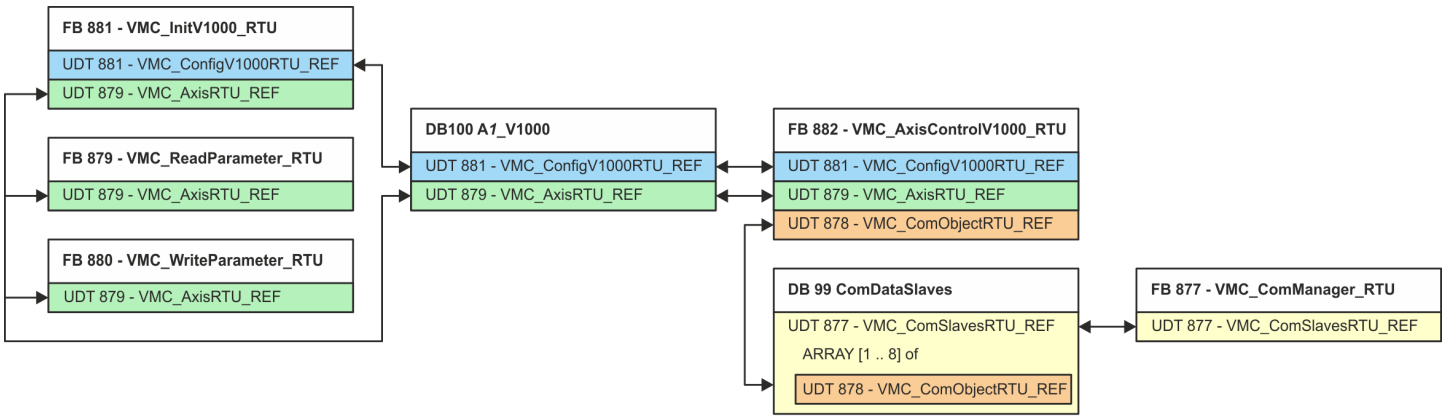
15.6.4.2.1 Programmstruktur

OB 100



- FB 876 - VMC_ConfigMaster_RTU ☞ *726*
  – Dieser Baustein dient zur Parametrierung der seriellen Schnittstelle der CPU für Modbus RTU Kommunikation.
  – Intern wird der Baustein SFC 216 - SER_CFG aufgerufen.

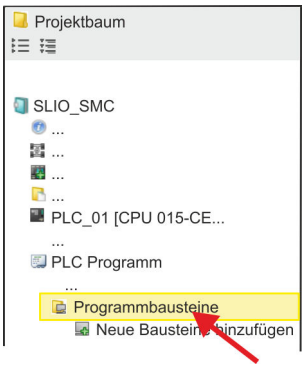
OB 1



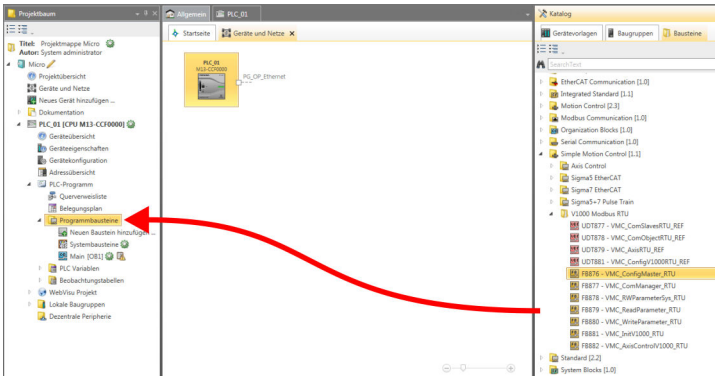
Mit Ausnahme der Bausteine DB 99 und FB 877 müssen Sie für jeden angebundenen Frequenzumrichter die nachfolgend aufgeführten Bausteine anlegen:

- FB 881 - VMC_InitV1000_RTU *730*
  - Der FB 881 - VMC_InitV1000_RTU initialisiert den entsprechenden Frequenzumrichter mit den Nutzerdaten.
  - Bevor ein Frequenzumrichter angesteuert werden kann, muss dieser initialisiert werden.
  - UDT 881 - VMC_ConfigV1000RTU_REF *726*
  - UDT 879 - VMC_AxisRTU_REF *726*
- FB 879 - VMC_ReadParameter_RTU *728*
  - Mit diesem FB haben Sie lesenden Zugriff auf die Parameter eines seriell über Modbus RTU angebundenen Frequenzumrichters.
  - Die gelesenen Daten werden in einem Datenbaustein erfasst.
  - UDT 879 - VMC_AxisRTU_REF *726*
- FB 880 - VMC_WriteParameter_RTU *729*
  - Mit diesem FB haben Sie schreibenden Zugriff auf die Parameter eines seriell über Modbus RTU angebundenen Frequenzumrichters.
  - Die zu schreibenden Daten sind in einem Datenbaustein abzulegen.
  - UDT 879 - VMC_AxisRTU_REF *726*
- DB 100 - A1_V1000
  - Für jeden Frequenzumrichter, welcher seriell über Modbus RTU angekoppelt ist, ist jeweils ein Datenbaustein anzulegen.
  - UDT 879 - VMC_AxisRTU_REF *726*
  - UDT 881 - VMC_ConfigV1000RTU_REF *726*
- FB 882 - VMC_AxisControlV1000_RTU *732*
  - Mit diesem Baustein können Sie einen seriell über Modbus RTU angebundenen Frequenzumrichter steuern und dessen Status abrufen.
  - UDT 881 - VMC_ConfigV1000RTU_REF *726*
  - UDT 879 - VMC_AxisRTU_REF *726*
  - UDT 878 - VMC_ComObjectRTU_REF *726*
- DB 99 - ComDataSlaves
  - Für die Erfassung der Kommunikationsdaten aller Frequenzumrichter (max. 8), welche seriell über Modbus RTU angekoppelt sind, ist ein gemeinsamer Datenbaustein anzulegen.
  - UDT 877 - VMC_ComSlavesRTU_REF *726*
  - UDT 878 - VMC_ComObjectRTU_REF *726*
- FB 877 - VMC_ComManager_RTU *728*
  - Der Baustein sorgt dafür, dass immer nur 1 Frequenzumrichter (Modbus-Slave) die serielle Schnittstelle benutzen kann. Bei Einsatz mehrerer Frequenzumrichter sendet dieser Baustein als Kommunikations-Manager die Aufträge an die jeweiligen Modbus-Slaves und wertet deren Antworten aus.
  - UDT 877 - VMC_ComSlavesRTU_REF *726*

15.6.4.2.2 Bausteine in Projekt kopieren



1. Klicken Sie auf *"Projektbaum ➔ ...CPU... ➔ PLC-Programm ➔ Programmbausteine"*.



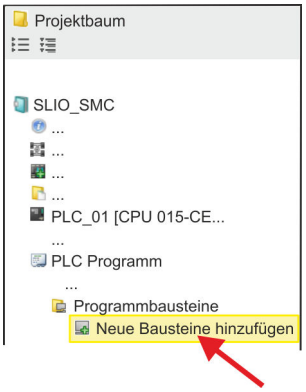
2. Öffnen Sie im *"Katalog"* unter *"Bausteine ➔ Simple Motion Control"* die Sammlung *"V1000 Modbus RTU"* und ziehen Sie per Drag&Drop folgende Bausteine in *"Programmbausteine"* des *Projektbaums*:

   - FB 876 - VMC_ConfigMaster_RTU
   - FB 877 - VMC_ComManager_RTU
   - FB 878 - VMC_RWParameterSys_RTU
   - FB 879 - VMC_ReadParameter_RTU
   - FB 880 - VMC_WriteParameter_RTU
   - FB 881 - VMC_InitV1000_RTU
   - FB 882 - VMC_AxisControlV1000_RTU

   Hierbei werden folgende Bausteine automatisch dem Projekt hinzugefügt:

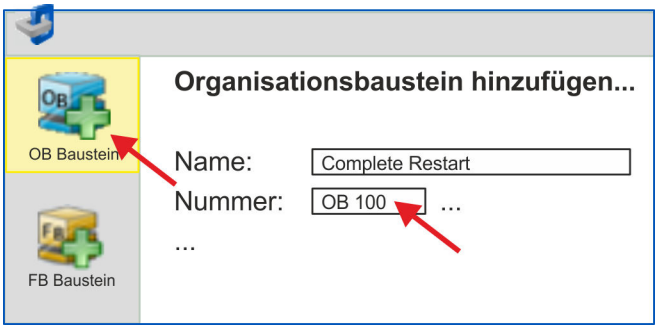
   - SEND (FB 60)
   - RECEIVE (FB 61)
   - RTU MB_MASTER (FB 72)
   - SER_CFG (FC 216)
   - SER_SND (FC 217)
   - SER_RCV (FC 218)
   - VMC_ComSlavesRTU_REF (UDT 877)
   - VMC_ComObjectRTU_REF (UDT 878)
   - VMC_AxisRTU_REF (UDT 879)
   - VMC_ConfigV1000RTU_REF (UDT 881)

15.6.4.2.3 OB 100 für serielle Kommunikation anlegen



1. Klicken Sie auf *"Projektbaum ➔ ...CPU... ➔ PLC-Programm ➔ Programmbausteine ➔ Neuen Baustein hinzufügen"*.

   ⇨ Das Dialogfenster *"Baustein hinzufügen"* öffnet sich.



2. Geben Sie OB 100 an und bestätigen Sie mit [OK].
   ⇨ Der OB 100 wird angelegt und geöffnet.
3. Fügen Sie dem OB 100 einen `Call FB876, DB876` zu.
   ⇨ Der Bausteinaufruf wird angelegt und es öffnet sich ein Dialog zur Angabe des Instanz-Datenbausteins *"VMC_ConfigMaster_RTU_876"*.
4. Bestätigen Sie die Abfrage des Instanz-Datenbausteins mit [OK].
5. Geben Sie folgende Parameter vor:

`Call FB876, DB876` *Kap. 15.6.5.5 "FB 876 - VMC_ConfigMaster_RTU - Modbus RTU CPU-Schnittstelle" Seite 726*

Baudrate	:= `B#16#09`	// Baudrate: 09h (9600Bit/s)	IN: BYTE
CharLen	:= `B#16#03`	// Anzahl Datenbits: 03h (8Bit)	IN: BYTE
Parity	:= `B#16#00`	// Parität: 0 (none = keine)	IN: BYTE
StopBits	:= `B#16#01`	// Stopbits: 1 (1Bit)	IN: BYTE
TimeOut	:= `W#16#1FFF`	// Fehler-Wartezeit: 1FFFh (hoch gewählt)	IN: WORD
Valid	:= `"ModbusConfigValid"`	// Konfiguration	OUT: BOOL
Error	:= `"ModbusConfigError"`	// Fehlerrückmeldung	OUT: BOOL
ErrorID	:= `"ModbusConfigErrorID"`	// Zusätzliche Fehlerinformationen	OUT: WORD

Symbolische Variable

Die symbolischen Variablen erstellen Sie über *"Kontextmenü ➔ Symbol erstellen/bearbeiten"*. Hier können Sie über einen Dialog den entsprechenden Operanden zuordnen.

15.6.4.2.4 Datenbaustein für Modbus-Slave anlegen

Für jeden Frequenzumrichter, welcher seriell über Modbus RTU angekoppelt ist, ist jeweils ein Datenbaustein anzulegen.

1. Klicken Sie hierzu auf *"Projektbaum ➔ ...CPU... ➔ PLC-Programm ➔ Programmbausteine ➔ Neuen Baustein hinzufügen"*.
   ⇨ Das Dialogfenster *"Baustein hinzufügen"* öffnet sich.
2. Wählen Sie den Bausteintyp *"DB Baustein"* und vergeben Sie diesem den Namen "A1_V1000". Die DB-Nr. können Sie frei wählen wie z.B. DB 100. Geben Sie DB 100 an und legen Sie diesen als globalen DB mit [OK] an.
   ⇨ Der Baustein wird angelegt und geöffnet.

3. Legen Sie in "A1_V1000" folgende Variablen an:
   - *"AxisData"* vom Typ UDT 879 - VMC_AxisRTU_REF
   - *"V1000Data"* vom Typ UDT 881 - VMC_ConfigV1000RTU_REF

15.6.4.2.5 Datenbaustein für alle Modbus-Slaves anlegen

Für die Erfassung der Kommunikationsdaten aller Frequenzumrichter, welche seriell über Modbus RTU angekoppelt sind, ist ein gemeinsamer Datenbaustein anzulegen.

1. Klicken Sie hierzu auf *"Projektbaum ➔ ...CPU... ➔ PLC-Programm ➔ Programmbausteine ➔ Neuen Baustein hinzufügen"*.
   ⇨ Das Dialogfenster *"Baustein hinzufügen"* öffnet sich.
2. Wählen Sie den Bausteintyp *"DB Baustein"* und vergeben Sie diesem den Namen "ComDataSlaves". Die DB-Nr. können Sie frei wählen wie z.B. DB 99. Geben Sie DB 99 an und legen Sie diesen als globalen DB mit [OK] an.
   ⇨ Der Baustein wird angelegt und geöffnet.
3. Legen Sie in "ComDataSlaves" folgende Variable an:
   - *"Slaves"* vom Typ UDT 877 - VMC_ComSlavesRTU_REF

15.6.4.2.6 OB 1 - Instanz des Kommunikations-Managers anlegen

Der FB 877 - VMC_ComManager_RTU sorgt dafür, dass immer nur 1 Frequenzumrichter (Modbus-Slave) die serielle Schnittstelle benutzen kann. Als Kommunikations-Manager sendet der Baustein die Aufträge an die jeweiligen Modbus-Slaves und wertet deren Antworten aus.

1. Doppelklicken Sie auf *"Projektbaum ➔ ...CPU... ➔ PLC-Programm ➔ Programmbausteine ➔ Main [OB1]"*.
   ⇨ Das Programmierfenster für den OB 1 wird geöffnet.
2. Fügen Sie dem OB 1 einen `Call FB877, DB877` zu.
   ⇨ Der Bausteinaufruf wird angelegt und es öffnet sich ein Dialog zur Angabe des Instanz-Datenbausteins *"VMC_ComManager_RTU_877"*.
3. Bestätigen Sie die Abfrage des Instanz-Datenbausteins mit [OK].
4. Geben Sie folgende Parameter vor:

`Call FB877, DB877` *Kap. 15.6.5.6 "FB 877 - VMC_ComManager_RTU - Modbus RTU Kommunikations-Manager" Seite 728*

NumberOfSlaves	:= `1`	// Anzahl angebundener Frequenzumrichter: 1	IN: INT
WaitCycles	:= `"ComWaitCycles"`	// Mindestanzahl Wartezyklen	IN: DINT
SlavesComData	:= `"ComDataSlaves.Slave"`	// Referenz zu allen Kommunikationsobjekten	IN-OUT: UDT 877

15.6.4.2.7 OB 1 - Instanz der V1000-Initialisierung anlegen

Der FB 881 - VMC_InitV1000_RTU initialisiert den entsprechenden Frequenzumrichter mit den Nutzerdaten. Bevor ein Frequenzumrichter angesteuert werden kann, muss dieser initialisiert werden.

1. Fügen Sie dem OB 1 einen `Call FB881, DB881` hinzu.
   ⇨ Der Bausteinaufruf wird angelegt und es öffnet sich ein Dialog zur Angabe des Instanz-Datenbausteins *"VMC_InitV1000_RTU _881"*.
2. Bestätigen Sie die Abfrage des Instanz-Datenbausteins mit [OK].
3. Geben Sie folgende Parameter vor:

`Call FB881, DB881` *Kap. 15.6.5.10 "FB 881 - VMC_InitV1000_RTU - Modbus RTU Initialisierung" Seite 730*

Execute	:= "A1_InitExecute"	// Mit Flanke 0-1 wird der Auftrag ausgeführt.	IN: BOOL
Hardware	:= "A1_InitHardware"	// Angabe der eingesetzten Hardware // 1: System SLIO CP040, 2: SPEED7 CPU	IN: BYTE
Laddr	:= "A1_InitLaddr"	// Logische Adresse bei Einsatz CP040	IN: INT
UnitId	:= "A1_InitUnitId"	// Modbus-Adresse des *V1000*	IN: BYTE
UserUnitsVelocity	:= "A1_InitUserUnitsVel"	// Benutzereinheit für Geschwindigkeiten // 0: Hz, 1: %, 2: U/min	IN: INT
UserUnitsAcceleration	:= "A1_InitUserUnitsAcc"	// Benutzereinheit Beschleunigung/Verzögerung // 0: 0,01s, 1: 0,1s	IN: INT
MaxVelocityApp	:= "A1_InitMaxVelocityApp"	// Max. Geschwindigkeit in Benutzereinheiten	IN: REAL
Done	:= "A1_InitDone"	// Status Auftrag fertig	OUT: BOOL
Busy	:= "A1_InitBusy"	// Status Auftrag in Bearbeitung	OUT: BOOL
Error	:= "A1_InitError"	// Fehlerrückmeldung	OUT: BOOL
ErrorID	:= "A1_InitErrorID"	// Zusätzliche Fehlerinformationen	OUT: WORD
Axis	:= "A1_V1000".AxisData	// Referenz zu den allgemeinen Achsdaten	IN-OUT: UDT 879
V1000	:= "A1_V1000".V1000Data	// Referenz zu den antriebsspezifischen Daten	IN-OUT: UDT 881

Eingabewerte

Alle Parameter sind mit den entsprechenden Variablen bzw. Operanden zu verschalten. Folgende Eingabe-Parameter sind entsprechend vorzubelegen:

- Hardware
  Geben Sie hier die Hardware an, über welche Sie Ihre Frequenzumrichter ansteuern:
  – 1: System SLIO CP040 dessen logische Adresse über *Laddr* vorzugeben ist.
  – 2: SPEED7 CPU
- Laddr
  – Logische Adresse für System SLIO CP040 (*Hardware* = 1). Ansonsten wird dieser Parameter ignoriert.
- UnitId
  – Modbus-Adresse des *V1000*.

- UserUnitsVelocity
  Benutzereinheit für Geschwindigkeiten:
  – 0: Hz
    Angabe in Hertz
  – 1: %
    Angabe als prozentualer Bezug auf die maximale Geschwindigkeit
    = $2*f_{max}/p$
    mit f_{max}: max. Ausgabefrequenz (Parameter E1-04)
    p: Anzahl der Motorpole (motorabhängiger Parameter E2-04, E4-04 oder E5-04)
  – 2: U/min
    Angabe in Umdrehungen pro Minute
- UserUnitsAcceleration
  Benutzereinheiten für die Beschleunigung und Verzögerung
  – 0: 0,01s (Wertebereich: 0,00s - 600,00s)
  – 1: 0,1s (Wertebereich: 0,0 - 6000,0s)
- MaxVelocityApp
  Max. Geschwindigkeit für die Applikation. Die Angabe hat in Benutzereinheiten zu erfolgen und wird bei Bewegungskommandos für den Abgleich verwendet.

15.6.4.2.8 OB 1 - Instanz Achskontrolle V1000 anlegen

Mit dem FB 882 - VMC_AxisControlV1000_RTU können Sie einen seriell über Modbus RTU angebundenen Frequenzumrichter steuern und dessen Status abrufen.

1. Fügen Sie dem OB 1 einen `Call FB882, DB882` hinzu.

⇨ Der Bausteinaufruf wird angelegt und es öffnet sich ein Dialog zur Angabe des Instanz-Datenbausteins *"VMC_AxisControlV1000_RTU _882"*.

2. Bestätigen Sie die Abfrage des Instanz-Datenbausteins mit [OK].

3. Geben Sie folgende Parameter vor:

`Call FB882, DB882` ↳ *Kap. 15.6.5.11 "FB 882 - VMC_AxisControlV1000_RTU - Modbus RTU Achskontrolle" Seite 732*

AxisEnable	:= "A1_AxisEnable"	// Aktivierung der Achse	IN: BOOL
AxisReset	:= "A1_AxisReset"	// Kommando: Fehler des *V1000* zurücksetzen.	IN: BOOL
StopExecute	:= "A1_StopExecute"	// Kommando: *Stop* - Achse stoppen	IN: BOOL
MvVelocityExecute	:= "A1_MvVelocityExecute"	// Kommando: *MoveVelocity* (Geschwindigkeitsregelung)	IN: BOOL
Velocity	:= "A1_Velocity"	// Parameter: Geschwindigkeitsvorgabe für *MoveVelocity*	IN: REAL
AccelerationTime	:= "A1_AccelerationTime"	// Parameter: Beschleunigungszeit	IN: REAL
DecelerationTime	:= "A1_DecelerationTime"	// Parameter: Verzögerungszeit	IN: REAL
JogPositive	:= "A1_JogPositive"	// Kommando: *JogPos*	IN: BOOL
JogNegative	:= "A1_JogNegative"	// Kommando: *JogNeg*	IN: BOOL
JogVelocity	:= "A1_JogVelocity"	// Parameter: Geschwindigkeitsvorgabe für Jogging	IN: REAL
JogAccelerationTime	:= "A1_JogAccelerationTime"	// Parameter: Beschleunigungszeit für Jogging	IN: REAL
JogDecelerationTime	:= "A1_JogDecelerationTime"	// Parameter: Verzögerungszeit für Jogging	IN: REAL
AxisReady	:= "A1_AxisReady"	// Status: Bereitschaft der Achse	OUT: BOOL
AxisEnabled	:= "A1_AxisEnabled"	// Status: Aktivierung der Achse	OUT: BOOL
AxisError	:= "A1_AxisError"	// Status: Achsfehler	OUT: BOOL

AxisErrorID	:= "A1_AxisErrorID"	// Status: Zusätzliche Fehlerinformationen für *AxisError*	OUT: WORD
DriveError	:= "A1_DriveError"	// Status: Fehler Frequenzumrichter	OUT: BOOL
ActualVelocity	:= "A1_ActualVelocity"	// Status: Aktuelle Geschwindigkeit	OUT: REAL
InVelocity	:= "A1_InVelocity"	// Status Zielgeschwindigkeit	OUT: BOOL
CmdDone	:= "A1_CmdDone"	// Status: Kommando fertig	OUT: BOOL
CmdBusy	:= "A1_CmdBusy"	// Status: Kommando in Bearbeitung	OUT: BOOL
CmdAborted	:= "A1_CmdAborted"	// Status: Kommando abgebrochen	OUT: BOOL
CmdError	:= "A1_CmdError"	// Status: Kommando Fehler	OUT: BOOL
CmdErrorID	:= "A1_CmdErrorID"	// Status: Zusätzliche Fehlerinformationen für *CmdError*	OUT: WORD
CmdActive	:= "A1_CmdActive"	// Status: Aktives Kommando	OUT: INT
DirectionPositive	:= "A1_DirectionPositive"	// Status: Drehrichtung positiv	OUT: BOOL
DirectionNegative	:= "A1_DirectionNegative"	// Status: Drehrichtung negativ	OUT: BOOL
Axis	:= "A1_V1000".AxisData	// Referenz zu den allgemeinen Achsdaten	IN-OUT: UDT 879
V1000	:= "A1_V1000".V1000Data	// Referenz zu den allgemeinen Daten des // Frequenzumrichters	IN-OUT: UDT 881
AxisComData	:= "ComDataSlaves".Slaves.Slave(1)	// Referenz zu den Kommunikationsdaten	IN-OUT: UDT 878

15.6.4.2.9 OB 1 - Instanz Parameter lesen anlegen

Mit dem FB 879 - VMC_ReadParameter_RTU haben Sie lesenden Zugriff auf die Parameter eines seriell über Modbus RTU angebundenen Frequenzumrichters. Für die Erfassung der Parameterdaten ist ein DB anzulegen.

1. Klicken Sie hierzu auf *"Projektbaum ➔ ...CPU... ➔ PLC-Programm ➔ Programmbausteine ➔ Neuen Baustein hinzufügen"*.
   ⇨ Das Dialogfenster *"Baustein hinzufügen"* öffnet sich.
2. Wählen Sie den Bausteintyp *"DB Baustein"* und vergeben Sie diesem den Namen "A1_TransferData". Die DB-Nr. können Sie frei wählen wie z.B. DB 98. Geben Sie DB 98 an und legen Sie diesen als globalen DB mit [OK] an.
   ⇨ Der Baustein wird angelegt und geöffnet.
3. Legen Sie in "A1_TransferData" folgende Variablen an:
   - *"Data_0"* vom Typ WORD
   - *"Data_1"* vom Typ WORD
   - *"Data_2"* vom Typ WORD
   - *"Data_3"* vom Typ WORD
4. Fügen Sie dem OB 1 einen `Call FB879, DB879` hinzu.
   ⇨ Der Bausteinaufruf wird angelegt und es öffnet sich ein Dialog zur Angabe des Instanz-Datenbausteins *"VMC_ReadParameter_RTU"*.
5. Bestätigen Sie die Abfrage des Instanz-Datenbausteins mit [OK].
6. Geben Sie folgende Parameter vor:

`Call FB879, DB879` ↻ *Kap. 15.6.5.8 "FB 879 - VMC_ReadParameter_RTU - Modbus RTU Parameter lesen" Seite 728*

Execute	:= "A1_RdParExecute"	// Mit Flanke 0-1 wird der Auftrag ausgeführt.	IN: BOOL
StartAddress	:= "A1_RdParStartAddress"	// Start-Adresse des 1. Registers	IN: INT

Quantity	:= "A1_RdParQuantity"	// Anzahl der zu lesenden Register	IN: INT
Done	:= "A1_RdParDone"	// Status Auftrag fertig	IN: REAL
Busy	:= "A1_RdParBusy"	// Status Auftrag in Bearbeitung	OUT: BOOL
Error	:= "A1_RdParError"	// Fehlerrückmeldung	OUT: BOOL
ErrorID	:= "A1_RdParErrorID"	// Zusätzliche Fehlerinformationen	OUT: BOOL
Data	:= P#DB98.DBX0.0 BYTES 8	// Ablageort der Parameterdaten	OUT: WORD
Axis	:= "A1_V1000".AxisData	// Referenz zu den allgemeinen Achsdaten	IN-OUT: UDT 879

Bitte beachten Sie, dass immer nur ganze Register als WORD gelesen werden können. Zur Auswertung einzelner Bits müssen Sie High- und Low-Byte vertauschen!

15.6.4.2.10 OB 1 - Instanz Parameter schreiben anlegen

Mit dem FB 880 - VMC_WriteParameter_RTU haben Sie schreibenden Zugriff auf die Parameter eines seriell über Modbus RTU angebundenen Frequenzumrichters. Für die Erfassung können Sie den für Lesezugriff angelegten DB verwenden - hier DB 98.

1. Fügen Sie dem OB 1 einen `Call FB880, DB880` hinzu.
   ⇨ Der Bausteinaufruf wird angelegt und es öffnet sich ein Dialog zur Angabe des Instanz-Datenbausteins *"VMC_WriteParameter_RTU"*.
2. Bestätigen Sie die Abfrage des Instanz-Datenbausteins mit [OK].
3. Geben Sie folgende Parameter vor:

`Call FB880, DB880` ↬ *Kap. 15.6.5.9 "FB 880 - VMC_WriteParameter_RTU - Modbus RTU Parameter schreiben" Seite 729*

Execute	:= "A1_WrParExecute"	// Mit Flanke 0-1 wird der Auftrag ausgeführt.	IN: BOOL
StartAddress	:= "A1_WrParStartAddress"	// Start-Adresse des 1. Registers	IN: INT
Quantity	:= "A1_WrParQuantity"	// Anzahl der zu schreibenden Register	IN: INT
Done	:= "A1_WrParDone"	// Status Auftrag fertig	IN: REAL
Busy	:= "A1_WrParBusy"	// Status Auftrag in Bearbeitung	OUT: BOOL
Error	:= "A1_WrParError"	// Fehlerrückmeldung	OUT: BOOL
ErrorID	:= "A1_WrParErrorID"	// Zusätzliche Fehlerinformationen	OUT: BOOL
Data	:= P#DB98.DBX0.0 BYTES 8	// Ablageort der Parameterdaten	OUT: WORD
Axis	:= "A1_V1000".AxisData	// Referenz zu den allgemeinen Achsdaten	IN-OUT: UDT 879

15.6.4.2.11 Zeitlicher Ablauf

1. Wählen Sie *"Projekt ➔ Alles übersetzen"* und übertragen Sie das Projekt in Ihre CPU.

   ⇨ Sie können jetzt Ihre Applikation über die bestehende Kommunikationsverbindung in Betrieb nehmen.

   VORSICHT!
   Bitte beachten Sie immer die Sicherheitshinweise zu ihrem Frequenzumrichter, insbesondere bei der Inbetriebnahme!

2. Mittels einer Beobachtungstabelle können Sie den Frequenzumrichter manuell steuern. Doppelklicken Sie auf *"Projektbaum ➔ ...CPU... ➔ PLC-Programm ➔ Beobachtungstabellen ➔ Neue Beobachtungstabelle hinzufügen"*.

3. Geben Sie einen Namen für die Beobachtungstabelle an wie z.B. *"V1000"* und bestätigen Sie mit [OK]

   ⇨ Die Beobachtungstabelle wird angelegt und für die Bearbeitung geöffnet.

4. Passen Sie zuerst die Wartezeit zwischen 2 Aufträgen an. Diese beträgt für einen V1000-Frequenzumrichter mindestens 200ms. Stellen Sie hierzu in der Beobachtungstabelle unter *"Namen"* den Bezeichner *"ComWaitCycles"* als *"Dezimal"* ein und geben Sie unter *"Steuerwert"* einen Wert zwischen 200 und 400 vor.

   Zur Performance-Steigerung können Sie diesen Wert später nach unten korrigieren, solange Sie keinen Timeout-Fehler (80C8h) erhalten. Bitte beachten Sie hierbei, dass manche Befehle, wie z.B. MoveVelocity aus mehreren Aufträgen bestehen können.

5. Bevor Sie einen Frequenzumrichter ansteuern können, muss dieser mit dem FB 881 - VMC_InitV1000_RTU initialisiert werden. ↳ *Kap. 15.6.5.10 "FB 881 - VMC_InitV1000_RTU - Modbus RTU Initialisierung" Seite 730*

   Stellen Sie hierzu in der Beobachtungstabelle unter *"Namen"* den Bezeichner *"A1_InitExecute"* als *"Boolean"* ein und geben Sie unter *"Steuerwert"* den Wert *"True"* vor. Aktivieren Sie *"Steuern"* und starten Sie die Übertragung der Steuerwerte.

   ⇨ Der Frequenzumrichter wird initialisiert. Nach Abarbeitung liefert der Ausgang *Done* TRUE zurück. Im Fehlerfall können Sie durch Auswertung der *ErrorID* den Fehler ermitteln.

   Fahren Sie erst fort, wenn der Init-Baustein keinen Fehler meldet!

6. Bei erfolgreicher Initialisierung erfolgt zyklisch die Abarbeitung der Register der angebundenen Frequenzumrichter d.h. diese erhalten zyklisch Aufträge. Zur manuellen Steuerung können Sie mit dem FB 882 - VMC_AxisControlV1000_RTU Steuerbefehle an den entsprechenden Frequenzumrichter senden. ↳ *Kap. 15.6.5.11 "FB 882 - VMC_AxisControlV1000_RTU - Modbus RTU Achskontrolle" Seite 732*

7. Legen Sie die Parameter des FB 882 - VMC_AxisControlV1000_RTU zur Steuerung und Abfrage in der Beobachtungstabelle an.

8. Aktivieren Sie durch Setzen von *AxisEnable* die entsprechende Achse. Sobald diese *AxisReady* = TRUE zurückmeldet, können Sie diese mit den entsprechenden Fahrbefehlen ansteuern.

15.6.5 Antriebsspezifische Bausteine

15.6.5.1 UDT 877 - VMC_ComSlavesRTU_REF - Modbus RTU Datenstruktur Kommunikationdaten aller Slaves

Dies ist eine benutzerdefinierte Datenstruktur, für die Kommunikationdaten der angebundenen Modbus RTU Slaves. Die UDT ist speziell angepasst an die Verwendung von Frequenzumrichter, welche über Modbus RTU angebunden sind.

15.6.5.2 UDT 878 - VMC_ComObjectRTU_REF - Modbus RTU Datenstruktur Kommunikationdaten Slave

Dies ist eine benutzerdefinierte Datenstruktur, für die Kommunikationdaten eines angebundenen Modbus RTU Slaves. Die UDT ist speziell angepasst an die Verwendung von Frequenzumrichter, welche über Modbus RTU angebunden sind.

15.6.5.3 UDT 879 - VMC_AxisRTU_REF - Modbus RTU Datenstruktur Achsdaten

Dies ist eine benutzerdefinierte Datenstruktur, die Statusinformationen des Frequenzumrichters beinhaltet. Diese Struktur dient als Referenz zu den allgemeinen Achsdaten des Frequenzumrichters.

15.6.5.4 UDT 881 - VMC_ConfigV1000RTU_REF - Modbus RTU Datenstruktur Konfiguration

Dies ist eine benutzerdefinierte Datenstruktur, die Informationen zu den Konfigurationsdaten eines Frequenzumrichters beinhaltet, welcher über Modbus RTU angebunden ist.

15.6.5.5 FB 876 - VMC_ConfigMaster_RTU - Modbus RTU CPU-Schnittstelle

Beschreibung

Dieser Baustein dient zur Parametrierung der seriellen Schnittstelle der CPU für Modbus RTU Kommunikation.

Bitte beachten Sie, dass dieser Baustein intern den SFC 216 aufruft.

Im SPEED7 Studio wird dieser Baustein automatisch in Ihr Projekt eingefügt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Baudrate	INPUT	BYTE	Geschwindigkeit der Datenübertragung in Bit/s (Baud). ■ 04h: 1200Baud ■ 05h: 1800Baud ■ 06h: 2400Baud ■ 07h: 4800Baud ■ 08h: 7200Baud ■ 09h: 9600Baud ■ 0Ah: 14400Baud ■ 0Bh: 19200Baud ■ 0Ch: 38400Baud ■ 0Dh: 57600Baud ■ 0Eh: 115200Baud
CharLen	INPUT	BYTE	Anzahl der Datenbits, auf die ein Zeichen abgebildet wird ■ 0: 5Bit ■ 1: 6Bit ■ 2: 7Bit ■ 3: 8Bit

Parameter	Deklaration	Datentyp	Beschreibung
Parity	INPUT	BYTE	Die Parität ist je nach Wert gerade oder ungerade. Zur Paritätskontrolle werden die Informationsbits um das Paritätsbit erweitert, das durch seinen Wert ("0" oder "1") den Wert aller Bits auf einen vereinbarten Zustand ergänzt. Ist keine Parität vereinbart, wird das Paritätsbit auf "1" gesetzt, aber nicht ausgewertet. ■ 0: None (keine) ■ 1: Odd (ungerade) ■ 2: Even (gerade)
StopBits	INPUT	BYTE	Die Stopbits werden jedem zu übertragenden Zeichen nachgesetzt und kennzeichnen das Ende eines Zeichens ■ 1: 1Bit ■ 2: 1.5Bit ■ 3: 2Bit
TimeOut	INPUT	WORD	Wartezeit, bis ein Fehler generiert wird, wenn ein Slave nicht antwortet. Die Zeitangabe für *TimeOut* ist als hexadezimaler Wert anzugeben. Den hexadezimalen Wert erhalten Sie, indem Sie die gewünschte Zeit in Sekunden mit der Baudrate multiplizieren. Beispiel: Gewünschte Zeit 8ms bei einer Baudrate von 19200Bit/s Berechnung: 19200Bit/s x 0,008s ≈ 154Bit >>>> (9Ah) Als Hex-Wert ist 9Ah vorzugeben.
Valid	OUTPUT	BOOL	Konfiguration ■ TRUE: Die Konfiguration ist gültig. ■ FALSE: Die Konfiguration ist nicht gültig.
Error	OUTPUT	BOOL	Fehlerrückmeldung ■ TRUE: Es ist ein Fehler aufgetreten - siehe *ErrorID*. ■ FALSE: Es liegt kein Fehler vor.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*

15.6.5.6 FB 877 - VMC_ComManager_RTU - Modbus RTU Kommunikations-Manager

Beschreibung

Dieser Baustein regelt, dass nacheinander immer nur ein Slave über die serielle Schnittstelle kommunizieren kann. Über die UDT 877 hat dieser Baustein Zugriff auf die Kommunikationsdaten aller Slaves.

Pro serielle Schnittstelle dürfen Sie immer nur einen FB 877 in Ihrem Projekt verwenden!

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
NumberOfSlaves	INPUT	INT	Anzahl der aktuell verwendeten Modbus Slaves
WaitCycles	INPUT	DINT	Mindestanzahl an Zyklen, die zwischen zwei Anfragen eines Slaves gewartet werden soll. Hiermit lassen sich Überläufe am Slave und hieraus resultierende Timeouts verhindern.
SlavesComData	IN_OUT	UDT 877	Referenz zum Datenbaustein mit allen Kommunikationobjekten

15.6.5.7 FB 878 - VMC_RWParameterSys_RTU - Modbus RTU Parameter System lesen/schreiben

Beschreibung

Dieser Baustein wird intern vom System für die Parameterübertragung verwendet.

Diesen Baustein dürfen Sie nicht aufrufen, da dies zu einem Fehlverhalten Ihres Systems führen kann!

15.6.5.8 FB 879 - VMC_ReadParameter_RTU - Modbus RTU Parameter lesen

Beschreibung

Mit diesem Baustein können Sie Parameter vom entsprechenden Slave lesen.

Bitte beachten Sie, dass immer nur ganze Register als WORD gelesen werden können. Zur Auswertung einzelner Bits müssen Sie High- und Low-Byte vertauschen!

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	IN	BOOL	Mit Flanke 0-1 wird der Auftrag ausgeführt.
StartAddress	IN	WORD	Start-Adresse des Registers, ab dem gelesen werden soll.
Quantity	IN	BYTE	Anzahl der Register, die gelesen werden sollen.
Done	OUT	BOOL	Status ■ TRUE: Auftrag erfolgreich durchgeführt

Parameter	Deklaration	Datentyp	Beschreibung
Busy	OUT	BOOL	Status ■ TRUE: Auftrag ist in Bearbeitung
Error	OUT	BOOL	Status ■ TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Data	IN-OUT	ANY	Referenz wohin die gelesenen Daten gespeichert werden sollen
Axis	IN-OUT	UDT 879	Referenz zu den allgemeinen Achsdaten des Frequenzumrichters

15.6.5.9 FB 880 - VMC_WriteParameter_RTU - Modbus RTU Parameter schreiben

Beschreibung

Mit diesem Baustein können Sie Parameter in die Register des entsprechenden Slave schreiben.

Bitte beachten Sie, dass immer nur ganze Register als WORD geschrieben werden können. Zum Setzen bzw. Rücksetzen einzelner Bits müssen Sie High- und Low-Byte vertauschen!

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	Mit Flanke 0-1 wird der Auftrag ausgeführt.
StartAddress	INPUT	WORD	Start-Adresse des Registers ab dem geschrieben werden soll.
Quantity	INPUT	BYTE	Anzahl der Register, die geschrieben werden sollen.
Done	OUTPUT	BOOL	Status ■ TRUE: Auftrag erfolgreich durchgeführt
Busy	OUTPUT	BOOL	Status ■ TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	Status ■ TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*

Parameter	Deklaration	Datentyp	Beschreibung
Data	IN_OUT	ANY	Referenz zu den Daten, die geschrieben werden sollen.
Axis	IN_OUT	UDT 879	Referenz zu den allgemeinen Achsdaten des Frequenzumrichters

15.6.5.10 FB 881 - VMC_InitV1000_RTU - Modbus RTU Initialisierung

Beschreibung

Dieser Baustein dient zur Initialisierung eines Frequenzumrichters mit den entsprechenden Nutzerdaten und muss vor der Übergabe von Kommandos abgearbeitet sein. Der Baustein ist speziell angepasst an die Verwendung eines Frequenzumrichters, welcher über Modbus RTU angebunden ist.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	Mit Flanke 0-1 wird der Auftrag ausgeführt.
Hardware	INPUT	BYTE	Angabe der eingesetzten Hardware ■ 1: System SLIO CP040 dessen logische Adresse über *Laddr* vorzugeben ist. ■ 2: SPEED7 CPU
Laddr	INPUT	INT	Logische Adresse für System SLIO CP040 (*Hardware* = 1). Ansonsten wird dieser Parameter ignoriert.
UnitId	INPUT	BYTE	Modbus-Adresse des *V1000*.
UserUnitsVelocity	INPUT	INT	Benutzereinheit für Geschwindigkeiten ■ 0: Hz – Angabe in Hertz ■ 1: % – Angabe als prozentualer Bezug auf die maximale Geschwindigkeit – = $2*f_{max}/p$ mit f_{max}: max. Ausgabefrequenz (Parameter E1-04) p: Anzahl der Motorpole (motorabhängiger Parameter E2-04, E4-04 oder E5-04) ■ 2: U/min – Angabe in Umdrehungen pro Minute
UserUnitsAcceleration	INPUT	INT	Benutzereinheiten für die Beschleunigung und Verzögerung ■ 0: 0,01s (Wertebereich: 0,00s - 600,00s) ■ 1: 0,1s (Wertebereich: 0,0 - 6000,0s)
MaxVelocityApp	INPUT	REAL	Max. Geschwindigkeit für die Applikation. Die Angabe hat in Benutzereinheiten zu erfolgen und wird bei Bewegungskommandos für den Abgleich verwendet.
Done	OUTPUT	BOOL	Status ■ TRUE: Auftrag erfolgreich durchgeführt
Busy	OUTPUT	BOOL	Status ■ TRUE: Auftrag ist in Bearbeitung

Parameter	Deklaration	Datentyp	Beschreibung
Error	OUTPUT	BOOL	Status ■ TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	UDT 879	Referenz zu den allgemeinen Achsdaten des Frequenzumrichters
V1000	IN_OUT	UDT 881	Referenz zu den Nutzerdaten des Frequenzumrichters

15.6.5.11 FB 882 - VMC_AxisControlV1000_RTU - Modbus RTU Achskontrolle

Beschreibung

Mit dem FB 882 *VMC_AxisControlV1000_RTU* können Sie einen über Modbus RTU angebundenen Frequenzumrichter steuern und dessen Status abrufen.

Die Ansteuerung eines V1000-Frequenzumrichters, welcher über Modbus RTU angebunden ist, erfolgt ausschließlich mit dem FB 882 VMC_AxisControlV1000_RTU. PLCopen-Bausteine werden nicht unterstützt!

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
AxisEnable	INPUT	BOOL	Aktivierung der Achse ■ TRUE: Achse einschalten → *AxisEnabled* = 1, Kommandos können ausgeführt werden ■ FALSE: Achse ausschalten → *AxisEnabled* = 0, es können keine Kommandos ausgeführt werden.
AxisReset	INPUT	BOOL	Kommando: Fehler des Frequenzumrichters zurücksetzen. → *CmdActive* = 1
StopExecute	INPUT	BOOL	Kommando: *Stop* - Achse stoppen → *CmdActive* = 1
MvVelocityExecute	INPUT	BOOL	Kommando: *MoveVelocity* (Geschwindigkeitsregelung) ausführen → *CmdActive* = 2
Velocity	INPUT	REAL	Parameter: Geschwindigkeitsvorgabe für MoveVelocity in Benutzereinheiten. Siehe Beispiel nach Tabelle
AccelerationTime	INPUT	REAL	Parameter: Beschleunigungszeit in Sekunden (Genauigkeit je nach *UserUnitsAcceleration* am Init Baustein). Ist immer bezogen auf die Zeit, vom Stillstand auf die maximal eingestellte Geschwindigkeit. Siehe Beispiel nach Tabelle. Dieser Parameter wird für das Kommando MoveVelocity (*MvVelocityExecute*) verwendet.
DecelerationTime	INPUT	REAL	Parameter: Verzögerungszeit in Sekunden (Genauigkeit je nach *UserUnitsAcceleration* am Init Baustein). Ist immer bezogen auf die Zeit, vom Stillstand auf die maximal eingestellte Geschwindigkeit. Siehe Beispiel unten. Dieser Parameter wird für die Kommandos Stop (*StopExecute*) und MoveVelocity (*MvVelocityExecute*) verwendet.
JogPositive	INPUT	BOOL	Kommando: *JogPos* ■ Flanke 0-1: Achse starten in positive Richtung (Jogging positiv) ■ Flanke 1-0: Achse stoppen
JogNegative	INPUT	BOOL	Kommando: *JogNeg* ■ Flanke 0-1: Achse starten in negative Richtung (Jogging negativ) ■ Flanke 1-0: Achse stoppen
JogVelocity	INPUT	REAL	Parameter: Geschwindigkeitsvorgabe für Jogging in Benutzereinheiten. Hinweis: Bei *JogPositive*, *JogNegative* wird der absolute Wert der Geschwindigkeit verwendet.

Parameter	Deklaration	Datentyp	Beschreibung
JogAcceleration-Time	INPUT	REAL	Parameter: Beschleunigungszeit für Jogging in Sekunden (Genauigkeit je nach *UserUnitsAcceleration* am Init-Baustein). Ist immer bezogen auf die Zeit, vom Stillstand auf die maximal eingestellte Geschwindigkeit. Siehe Beispiel nach Tabelle
JogDeceleration-Time	INPUT	REAL	Parameter: Verzögerungszeit für Jogging in Sekunden (Genauigkeit je nach *UserUnitsAcceleration* von FB 881). Parameter bezieht sich immer auf die Zeit vom Stillstand auf die maximal eingestellte Geschwindigkeit. Siehe Beispiel nach Tabelle
AxisReady	OUTPUT	BOOL	Status: Bereitschaft der Achse ■ TRUE: Achse ist einschaltbereit ■ FALSE: Achse ist nicht einschaltbereit
AxisEnabled	OUTPUT	BOOL	Status: Aktivierung der Achse ■ TRUE: Achse ist eingeschaltet ■ FALSE: Achse ist ausgeschaltet
AxisError	OUTPUT	BOOL	Status: Achsfehler ■ TRUE: Achse meldet einen Fehler und wird gesperrt. Nähere Fehlerinformationen befinden sich in *AxisErrorID*. ■ FALSE: Achse meldet keine Fehler.
AxisErrorID	OUTPUT	WORD	Status: Zusätzliche Fehlerinformationen für *AxisError* *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
DriveError	OUTPUT	BOOL	Status: Fehler Frequenzumrichter ■ TRUE: Frequenzumrichter meldet einen Fehler und wird gesperrt. ■ FALSE: Frequenzumrichter meldet keine Fehler.
ActualVelocity	OUTPUT	REAL	Status: Aktuelle Geschwindigkeit in Benutzereinheiten
InVelocity	OUTPUT	BOOL	Status Zielgeschwindigkeit ■ TRUE: Die Zielgeschwindigkeit *Velocity* wurde erreicht. ■ FALSE: Die Zielgeschwindigkeit *Velocity* wurde noch nicht erreicht.
CmdDone	OUTPUT	BOOL	Status: Kommando fertig ■ TRUE: Kommando wurde erfolgreich ausgeführt. ■ FALSE: Kommando wurde noch nicht ausgeführt bzw. befindet sich noch in der Bearbeitung.
CmdBusy	OUTPUT	BOOL	Status: Kommando in Bearbeitung ■ TRUE: Kommando befindet sich in der Bearbeitung ■ FALSE: Aktuell wird kein Kommando ausgeführt.
CmdAborted	OUTPUT	BOOL	Status: Kommando abgebrochen ■ TRUE: Kommando wurde abgebrochen. ■ FALSE: Kommando wurde nicht abgebrochen

Parameter	Deklaration	Datentyp	Beschreibung
CmdError	OUTPUT	BOOL	Status: Kommando Fehler ■ TRUE: Bei der Ausführung eines Kommandos ist ein Fehler aufgetreten ■ FALSE: Die Ausführung eines Kommandos verlief fehlerfrei.
CmdErrorID	OUTPUT	WORD	Status: Zusätzliche Fehlerinformationen für *CmdError* ↬ *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
CmdActive	OUTPUT	INT	Status: Aktives Kommando ■ 0: NoCmd - kein Kommando aktiv ■ 1: Stop ■ 2: MvVelocity ■ 3: MvRelative ■ 4: JogPos ■ 5: JogNeg
DirectionPositive	OUTPUT	BOOL	Status: Drehrichtung positiv ■ TRUE: Aktuelle Drehrichtung ist positiv ■ FALSE: Aktuelle Drehrichtung ist nicht positiv
DirectionNegative	OUTPUT	BOOL	Status: Drehrichtung negativ ■ TRUE: Aktuelle Drehrichtung ist negative ■ FALSE: Aktuelle Drehrichtung ist nicht negative
Axis	IN_OUT	UDT 879	Referenz zu den allgemeinen Achsdaten des Frequenzumrichters
V1000	IN_OUT	UDT 881	Referenz zu den Nutzerdaten des Frequenzumrichters
AxisComData	IN_OUT	UDT 878	Referenz zu den Kommunikationsdaten des aktuellen Slave

Beispiel *AccelerationTime*

Die Werte für *Velocity*, *AccelerationTime* und *DecelerationTime* sind in den unter FB 881 - VMC_InitV1000_RTU eingestellten Benutzereinheiten vorzugeben. *AccelerationTime* bzw. *DecelerationTime* beziehen sich immer auf die Zeit vom Stillstand bis zur maximal eingestellten Geschwindigkeit bzw. von der maximalen Geschwindigkeit bis zum Stillstand.

Die maximal Geschwindigkeit ergibt sich über die Formel

$$v_{max} = \frac{2 \cdot f}{p}$$

v_{max} max. Geschwindigkeit in 1/s
f max. Ausgabefrequenz (Parameter E1-04)
p Anzahl der Motorpole (motorabhängiger Parameter E2-04, E4-04 oder E5-04)

Zeitlicher Ablauf

1. Wählen Sie *"Projekt ➔ Alles übersetzen"* und übertragen Sie das Projekt in Ihre CPU.
   ⇨ Sie können jetzt Ihre Applikation in Betrieb nehmen.

VORSICHT!
Bitte beachten Sie immer die Sicherheitshinweise zu ihrem Frequenzumrichter, insbesondere bei der Inbetriebnahme!

2. Bringen Sie Ihre CPU in RUN und schalten Sie Ihren Frequenzumrichter ein.
   ⇨ Der FB 882 - VMC_AxisControlV1000_RTU wird zyklisch abgearbeitet.
3. Sobald *AxisReady* = TRUE meldet, können Sie mit *AxisEnable* die Achse frei geben.
4. Sie haben jetzt die Möglichkeit über die entsprechenden Parameter Ihre Achse zu steuern und deren Status abzufragen.

15.7 Einsatz Frequenzumrichter über EtherCAT

15.7.1 Übersicht

Voraussetzung

- SPEED7 Studio ab V1.8
- CPU mit EtherCAT-Master wie z.B. CPU 015-CEFNR00
- Frequenzumrichter mit EtherCAT-Optionskarte

Schritte der Projektierung

1. Parameter am Frequenzumrichter einstellen.
   - Die Einstellung der Parameter hat mit dem Softwaretool *Drive Wizard+* zu erfolgen.
2. Hardwarekonfiguration im VIPA *SPEED7 Studio.*
   - Projektierung der CPU.
3. Programmierung im VIPA *SPEED7 Studio.*
   - *Init*-Baustein zur Konfiguration der Achse beschalten.
   - *Kernel*-Baustein zur Kommunikation mit der Achse beschalten.
   - Bausteine für die Bewegungsabläufe beschalten.

15.7.2 Parameter am Frequenzumrichter einstellen

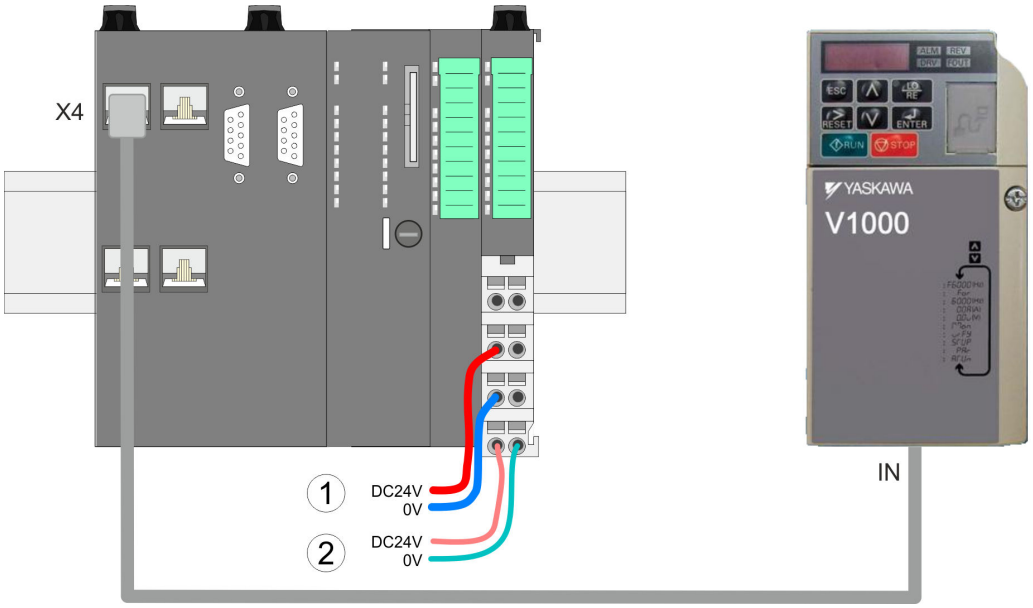
VORSICHT!
Vor der Erstinbetriebnahme müssen Sie Ihren Frequenzumrichter mit dem Softwaretool *Drive Wizard+* an Ihre Applikation anpassen! Näheres hierzu finden Sie im Handbuch zu ihrem Frequenzumrichter.

Die nachfolgende Tabelle zeigt alle Parameter auf, die nicht den Standardwerten entsprechen. Zur Abstimmung auf die *Simple Motion Control Library* sind diese über *Drive Wizard+* einzustellen:

Nr.	Bezeichnung	Wertebereich	Einstellung für *Simple Motion Control Library*
B1-01	Eingangsquelle Frequenzsollwert 1	0, 1, 2, 3, 4	■ 3: Option card
B1-02	Eingangsquelle Startbefehl 1	0, 1, 2, 3	■ 3: Option card
O1-03	Display Skalierung	0, 1, 2, 3, 4	■ 2: min-1 unit

Damit alle Einstellungen übernommen werden, müssen Sie den Frequenzumrichter nach der Parametrierung neu starten!

15.7.3 Beschaltung



(1) DC 24V für Leistungsversorgung I/O-Ebene (max. 10A)
(2) DC 24V für Elektronikversorgung CPU und I/O-Ebene

Vorgehensweise

1. Schalten Sie die Stromversorgung von CPU und Frequenzumrichter ab.
2. Montieren Sie, falls nicht schon vorhanden, die EtherCAT-Optionskarte in Ihrem Frequenzumrichter.
3. Verbinden Sie die Optionskarte und den Frequenzumrichter über das beiliegenden Massekabel.
4. Verbinden Sie die EtherCAT-Buchse *"X4"* der CPU mit der *"IN"*-Buchse der Optionskarte über ein EtherCAT-Kabel.

   ⇨ Ihr System ist nun bereit für die Inbetriebnahme.

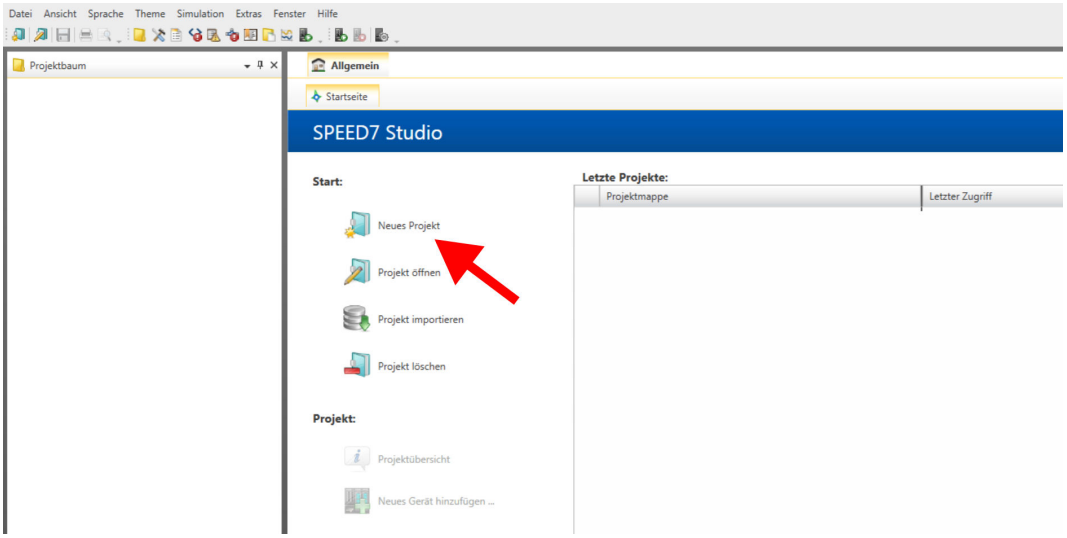
15.7.4 Einsatz im VIPA *SPEED7 Studio*

15.7.4.1 Hardware-Konfiguration

CPU im Projekt anlegen

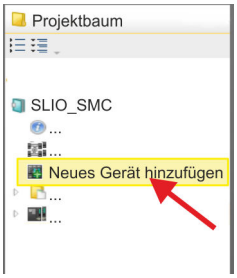
Bitte verwenden Sie für die Projektierung das *SPEED7 Studio* ab V1.8

1. Starten Sie das *SPEED7 Studio*.

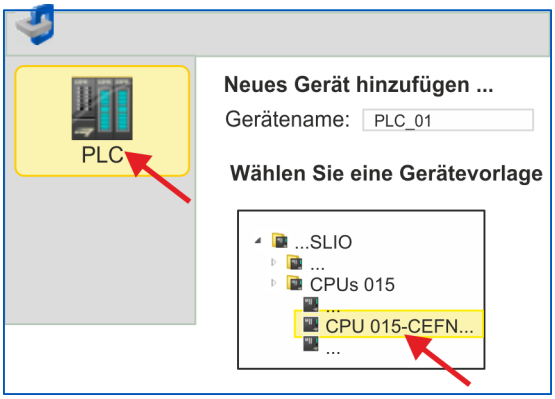


2. Erstellen sie auf der Startseite mit *"Neues Projekt"* ein neues Projekt und vergeben Sie einen *"Projektnamen"*.

⇨ Ein neues Projekt wird angelegt und in die Sicht *"Geräte und Netze"* gewechselt.



3. Klicken Sie im *Projektbaum* auf *"Neues Gerät hinzufügen ..."*.



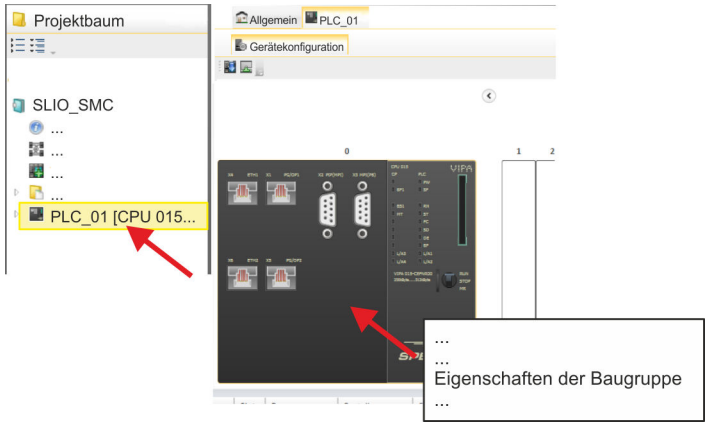
⇨ Es öffnet sich ein Dialog für die Geräteauswahl.

4. Wählen Sie unter den *"Gerätevorlagen"* eine CPU mit EtherCAT-Master-Funktionalität wie z.B. die CPU 015-CEFNR00 und klicken Sie auf [OK].

⇨ Die CPU wird in *"Geräte und Netze"* eingefügt und die *"Gerätekonfiguration"* geöffnet.

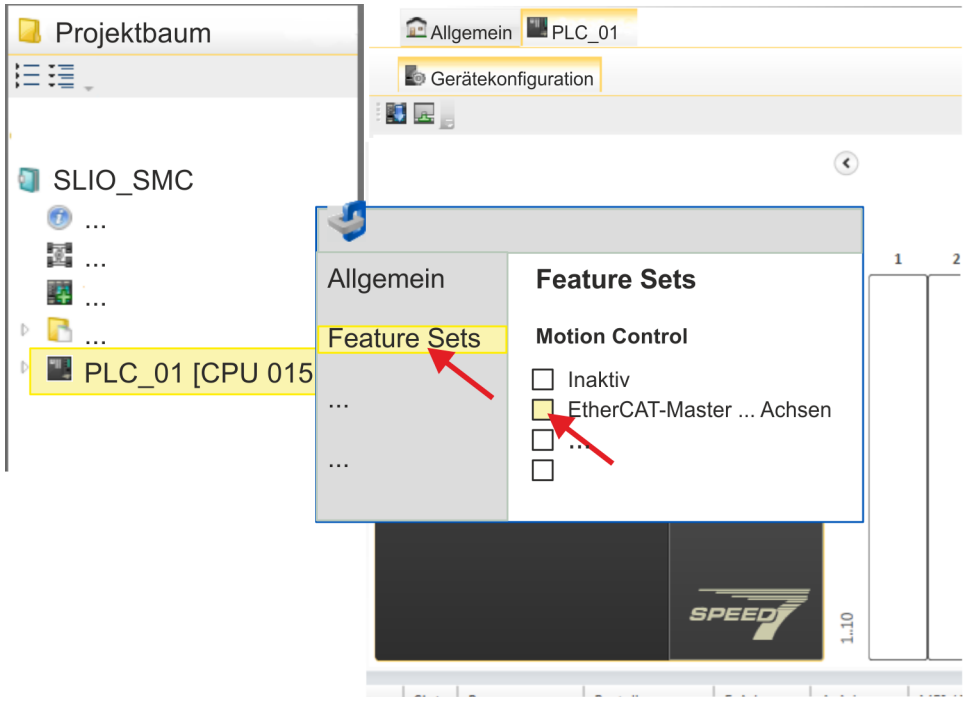
Motion-Control-Funktionen aktivieren

Sofern bei Ihrer CPU die EtherCAT-Master-Funktionalität noch nicht aktiviert ist, erfolgt die Aktivierung nach folgenden Vorgehensweise:



1. Klicken Sie in der *"Gerätekonfiguration"* auf die CPU und wählen Sie *"Kontextmenü ➔ Eigenschaften der Baugruppe"*.

   ⇨ Es öffnet sich der Eigenschaften-Dialog der CPU.



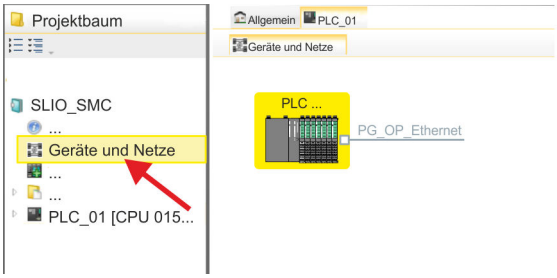
2. Klicken Sie auf *"Feature Sets"* und aktivieren Sie unter *"Motion Control"* einen der Parameter *"EtherCAT-Master ... Achsen"*. Die Anzahl der Achsen ist in diesem Beispiel nicht relevant.
3. Bestätigen Sie Ihre Angaben mit [OK].

   ⇨ Die Motion-Control-Funktionen steht Ihnen nun in Ihrem Projekt zur Verfügung.

VORSICHT!
Bitte beachten Sie, dass bei jeder Änderung der Feature-Set-Einstellungen systembedingt das EtherCAT-Feldbus-System zusammen mit der Motion-Control-Konfiguration aus Ihrem Projekt gelöscht werden!

Ethernet-PG/OP-Kanal parametrieren

1. Klicken Sie im *Projektbaum* auf *"Geräte und Netze"*.
   ⇨ Sie erhalten eine grafische Objekt-Ansicht Ihrer CPU.



2. Klicken Sie auf das Netzwerk *"PG_OP_Ethernet"*.
3. Wählen Sie *"Kontextmenü ➔ Eigenschaften der Schnittstelle"*.
   ⇨ Es öffnet sich ein Dialogfenster. Hier können Sie IP-Adressdaten für Ihren Ethernet-PG/OP-Kanal angeben. Gültige IP-Adress-Parameter erhalten Sie von Ihrem Systemadministrator.
4. Bestätigen Sie Ihre Eingabe mit [OK].
   ⇨ Die IP-Adressdaten werden in Ihr Projekt übernommen und in *"Geräte und Netze"* unter *"Lokale Baugruppen"* aufgelistet.

   Nach der Übertragung Ihres Projekts ist Ihre CPU über die angegebenen IP-Adressdaten via Ethernet-PG/OP-Kanal erreichbar.

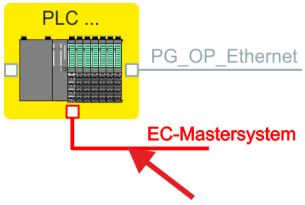
ESI-Datei installieren

Damit der Frequenzumrichter im *SPEED7 EtherCAT Manager* konfiguriert werden kann, muss die entsprechende ESI-Datei installiert sein. In der Regel wird das *SPEED7 Studio* mit aktuellen ESI-Dateien ausgeliefert und Sie können diesen Teil überspringen. Sollte Ihre ESI-Datei veraltet sein, finden Sie die aktuellste ESI-Datei für den Frequenzumrichter unter *www.yaskawa.eu.com* unter *"Service ➔ Drives & Motion Software"*.

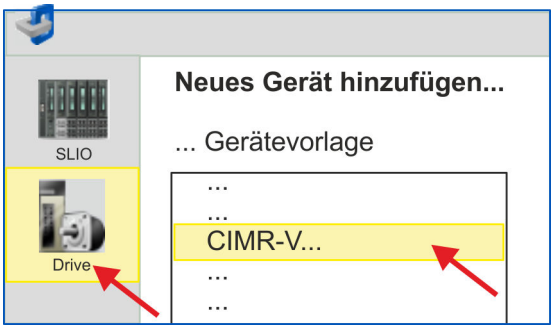
1. Laden Sie die zu Ihrem Frequenzumrichter passende ESI-Datei herunter. Entpacken Sie diese falls erforderlich.
2. Gehen Sie in Ihr *SPEED7 Studio*.
3. Öffnen Sie mit *"Extras ➔ Gerätebeschreibungsdatei installieren (EtherCAT - ESI)"* das zugehörige Dialogfenster.
4. Geben Sie unter *"Quellpfad"* die ESI-Datei an und installieren Sie diese mit [Installieren].
   ⇨ Die Geräte der ESI-Datei steht Ihnen nun zur Verfügung.

Frequenzumrichter hinzufügen

1. Klicken Sie im Projektbaum auf *"Geräte und Netze"*.
2. Klicken Sie hier auf *"EC-Mastersystem"* und wählen sie *"Kontextmenü ➔ Neues Gerät hinzufügen"*.



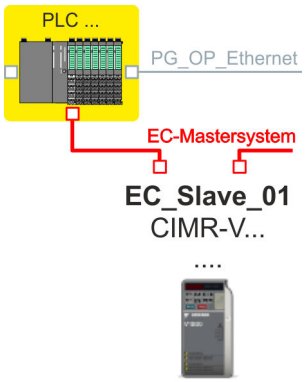
   ⇨ Es öffnet sich die Gerätevorlage zur Auswahl eines EtherCAT-Devices.



3. Wählen Sie Ihren Frequenzumrichter aus:

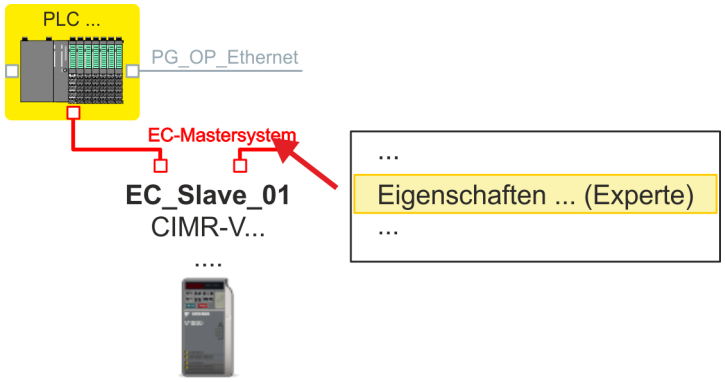
- CIMR-Vxxxx...
- CIPR-GA70xxxx...

Bestätigen Sie Ihre Angaben mit [OK]. Sollte Ihr Antrieb nicht vorhanden sein, müssen Sie die entsprechende ESI-Datei wie weiter oben beschrieben installieren.



⇨ Der Frequenzumrichter wird an Ihr EC-Mastersystem angebunden.

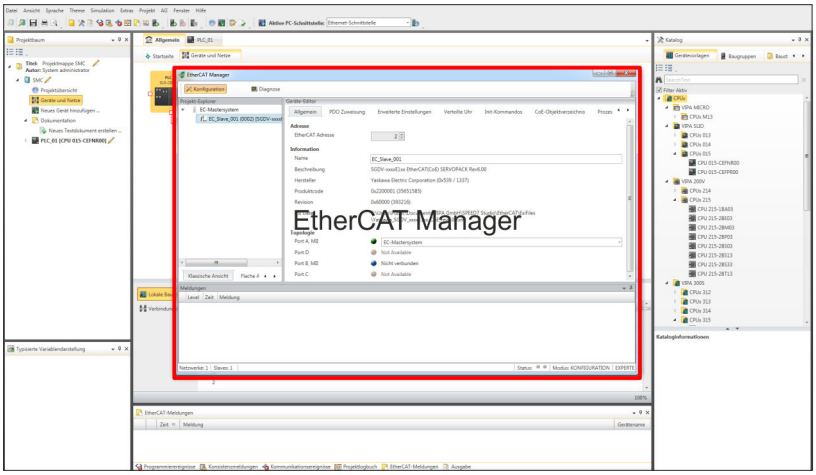
Frequenzumrichter konfigurieren



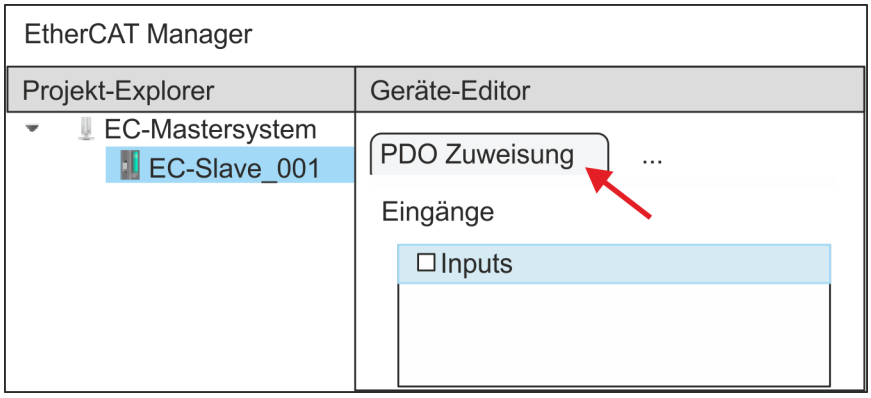
1. Klicken Sie auf *"EC-Mastersystem"* und wählen sie *"Kontextmenü ➔ Eigenschaft des Bussystems (Experte)"*.

PDOs können Sie nur im "Experten-Modus" bearbeiten! Ansonsten werden die Schaltflächen ausgeblendet.

⇨ Der *SPEED7 EtherCAT Manager* wird gestartet. Hier können Sie die EtherCAT-Kommunikation zu Ihrem Frequenzumrichter konfigurieren.



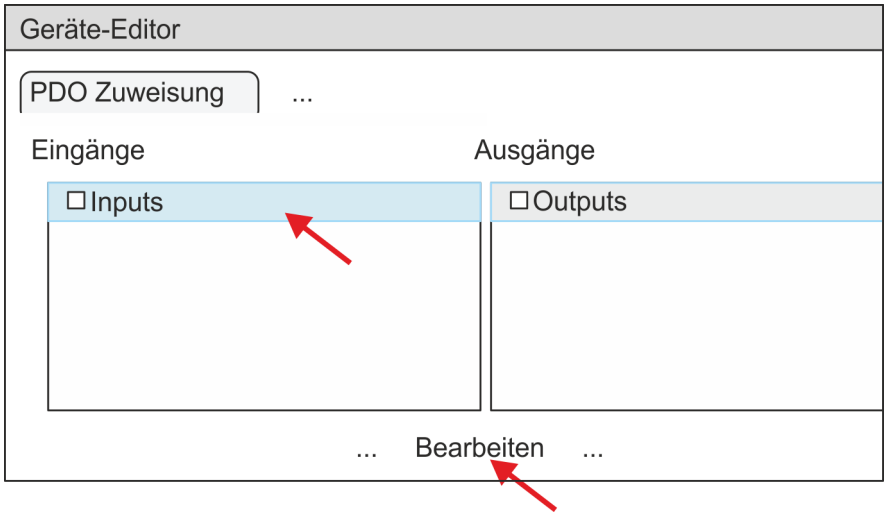
2. Klicken Sie im *SPEED7 EtherCAT Manager* auf den Slave und wählen Sie im *"Geräte-Editor"* den Reiter *"PDO-Zuweisung"* an.



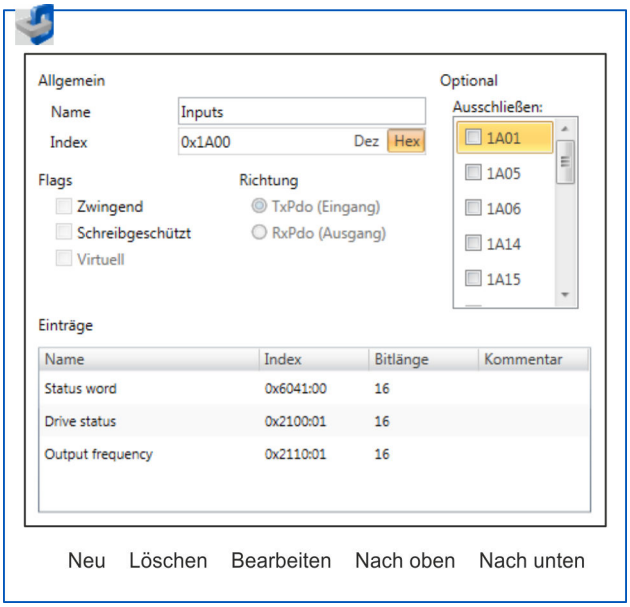
⇨ Dieser Dialog zeigt eine Auflistung aller PDOs.

3. Durch Anwahl des entsprechenden PDO-Mappings können Sie mit [Bearbeiten] die PDOs bearbeiten. Wählen Sie das Mapping *"Inputs"* an und klicken Sie auf [Bearbeiten].

> *Bitte beachten Sie, dass aufgrund der Voreinstellung manche PDOs nicht bearbeitet werden können. Durch Deaktivierung bereits aktivierter PDOs können Sie die Bearbeitung von gesperrten PDOs frei geben.*



⇨ Es öffnet sich der Dialog *"PDO bearbeiten"*. Bitte überprüfen Sie hier die aufgeführten PDO-Einstellungen und passen Sie diese ggf. an. Bitte berücksichtigen Sie hierbei auch die Reihenfolge der *"Einträge"* und ergänzen Sie diese entsprechend.



Für die Bearbeitung der *"Einträge"* stehen folgende Funktionen zur Verfügung:

- Neu
  - Hiermit können Sie in einem Dialogfenster einen neuen Eintrag anlegen, indem Sie aus dem *"CoE-Objektverzeichnis"* den entsprechenden Eintrag auswählen und Ihre Einstellungen vornehmen. Mit [OK] wird der Eintrag übernommen und in der Liste der Einträge aufgeführt.
- Löschen
  - Hiermit können Sie den angewählte Eintrag löschen.

- Bearbeiten
  - Hiermit können Sie allgemeinen Daten eines Eintrags bearbeiten.
- Nach oben/unten
  - Hiermit können Sie den angewählten Eintrag in der Liste nach oben bzw. nach unten bewegen.

4. Führen Sie folgende Einstellungen durch:

Inputs

- Allgemein
  - Name: Inputs
  - Index: 0x1A00
- Flags
  - Alles deaktiviert
- Richtung
  - TxPdo (Eingang): aktiviert
- Ausschließen
  Bitte diese Einstellungen beachten, da ansonsten die PDO-Mappings nicht zeitgleich aktiviert werden können!
  - Alles deaktiviert
- Einträge

Name	Index	Bitlänge
Status word	0x6041:00	16Bit
Drive status value	0x2100:01	16Bit
Output frequency value	0x2110:01	16Bit

Schließen Sie den Dialog *"PDO bearbeiten"* mit [OK].

5. Wählen Sie das Mapping *"Outputs"* an und klicken Sie auf [Bearbeiten]. Führen Sie folgende Einstellungen durch:

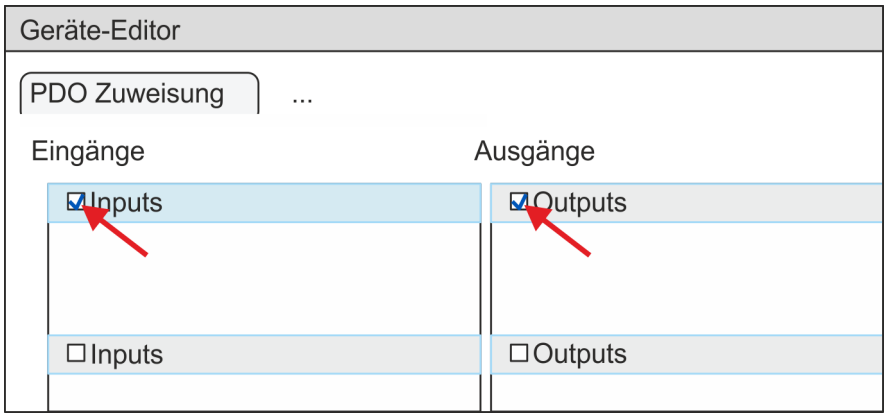
Outputs

- Allgemein
  - Name: Outputs
  - Index: 0x1600
- Flags
  - Alles deaktiviert
- Richtung
  - RxPdo (Ausgang): aktiviert
- Ausschließen
  Bitte diese Einstellungen beachten, da ansonsten die PDO-Mappings nicht zeitgleich aktiviert werden können!
  - Alles deaktiviert
- Einträge

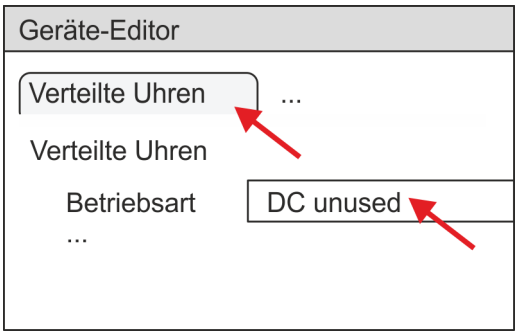
Name	Index	Bitlänge
Control word	0x6040:00	16Bit
vl target velocity	0x6042:00	16Bit
vl velocity acceleration: Delta speed	0x6048:01	32Bit
vl velocity acceleration: Delta time	0x6048:02	16Bit

Schließen Sie den Dialog *"PDO bearbeiten"* mit [OK].

6. Aktivieren Sie in *"PDO-Zuweisung"* die jeweils 1. PDOs "Inputs" und "Outputs". Alle nachfolgenden PDOs müssen deaktiviert bleiben. Sollte dies nicht möglich sein, überprüfen Sie bitte den jeweiligen PDO-Parameter *"Ausschließen"*.

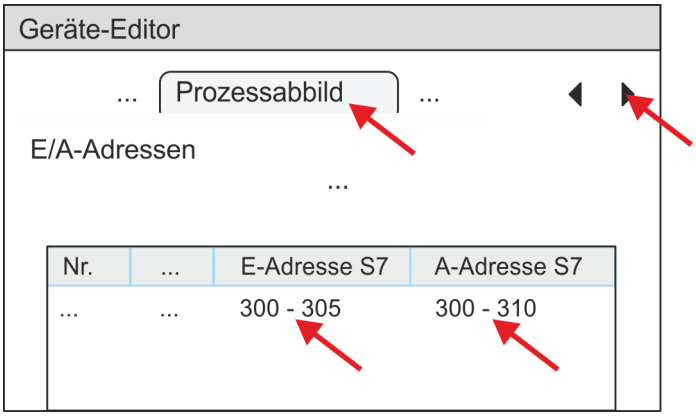


7. Wählen Sie im *"Geräte-Editor"* des *SPEED7 EtherCAT Manager* den Reiter *"Verteilte Uhren"* an und stellen Sie *"DC unused"* als *"Betriebsart"* ein.



8. Wählen Sie im *"Geräte-Editor"* über die Pfeiltaste den Reiter *"Prozessabbild"* an und notieren Sie sich für die Parameter des Bausteins FB 887 - VMC_InitInverter_EC folgende PDO-Anfangsadressen:

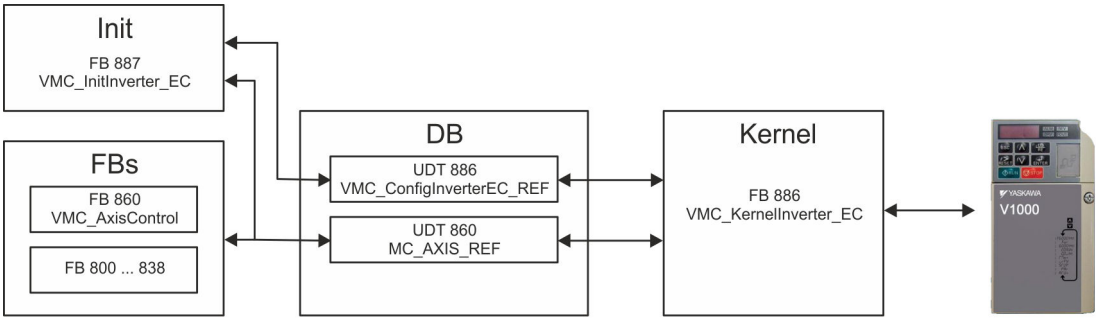
- *"E-Adresse S7"* → *"InputsStartAddressPDO"*
- *"A-Adresse S7"* → *"OutputsStartAddressPDO"*



9. Indem Sie den Dialog des *SPEED7 EtherCAT Manager* mit [X] schließen, wird die Konfiguration in das *SPEED7 Studio* übernommen.

15.7.4.2 Anwender-Programm

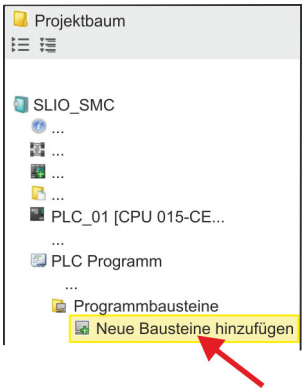
15.7.4.2.1 Programmstruktur



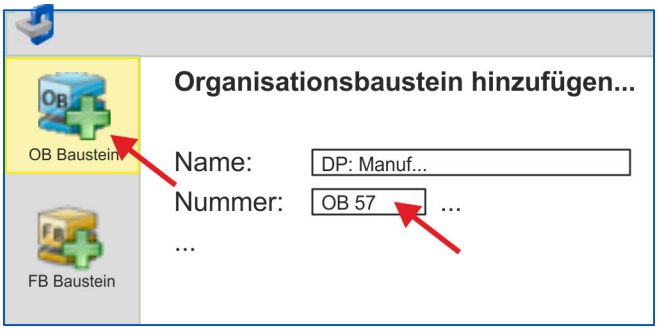
- DB
  Für jede Achse ist ein Datenbaustein (Achs-DB) für Konfiguration und Statusdaten anzulegen. Der Datenbaustein besteht aus folgenden Datenstrukturen:
  – UDT 886 - *VMC_Config*InverterEC_REF
    Die Datenstruktur beschreibt den Aufbau der Konfiguration des Antriebs.
    Spezifische Datenstruktur für Frequenzumrichter mit EtherCAT.
  – UDT 860 - *MC_AXIS_REF*
    Die Datenstruktur beschreibt den Aufbau der Parameter und Statusinformationen von Antrieben.
    Allgemeine Datenstruktur für alle Antriebe und Bussysteme.
- FB 887 - *VMC_Init*Inverter_EC
  – Der *Init*-Baustein dient zur Konfiguration einer Achse.
  – Spezifischer Baustein für Frequenzumrichter mit EtherCAT.
  – Die Konfigurationsdaten für die Initialisierung sind im *Achs-DB* abzulegen.
- FB 886 - *VMC_Kernel*Inverter_EC
  – Der *Kernel*-Baustein kommuniziert mit dem Antrieb über das entsprechende Bussystem, verarbeitet die Benutzeraufträge und liefert Statusmeldungen zurück.
  – Spezifischer Baustein für Frequenzumrichter mit EtherCAT.
  – Der Austausch der Daten erfolgt mittels des *Achs-DB*.
- FB 860 - *VMC_AxisControl*
  – Universal-Baustein für alle Antriebe und Bussysteme.
  – Unterstützt einfache Bewegungskommandos und liefert alle relevanten Statusmeldungen.
  – Der Austausch der Daten erfolgt mittels des *Achs-DB*.
  – Über die Instanzdaten des Bausteins können Sie zur Bewegungssteuerung und Statusabfrage eine Visualisierung anbinden.
  – Zusätzlich zum FB 860 - *VMC_AxisControl* haben Sie die Möglichkeit *PLCopen*-Bausteine zu nutzen.
- FB 800 ... FB 838 - *PLCopen*
  – Die PLCopen-Bausteine dienen zur Programmierung von Bewegungsabläufen und Statusabfragen.
  – Allgemeine Bausteine für alle Antriebe und Bussysteme.

15.7.4.2.2 Programmierung

Bausteine in Projekt kopieren

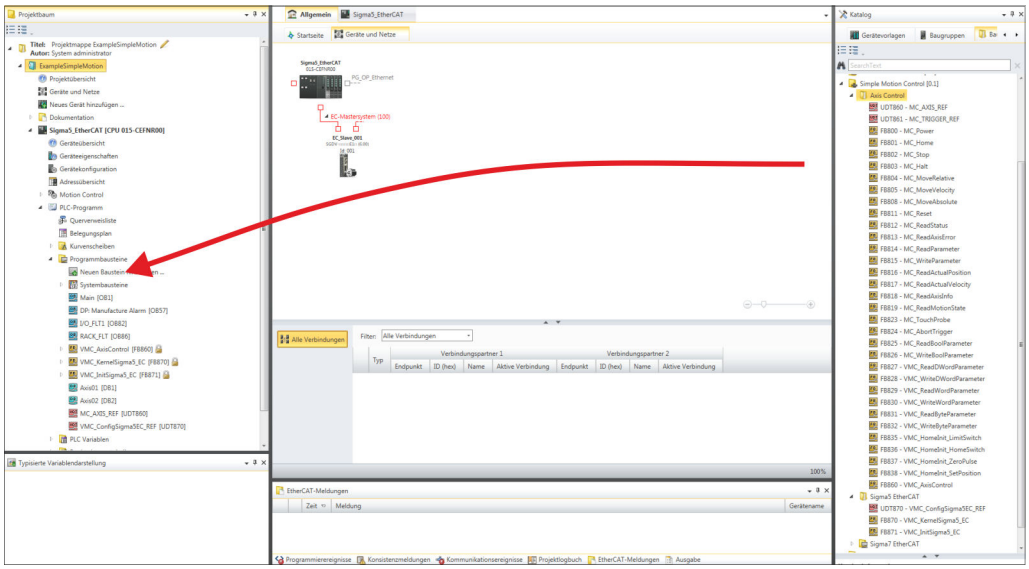


1. Klicken Sie im *Projektbaum* innerhalb der CPU unter *"PLC-Programm"*, *"Programmbausteine"* auf *"Neuen Baustein hinzufügen"*.



⇨ Das Dialogfenster *"Baustein hinzufügen"* öffnet sich.

2. Wählen Sie den Bausteintyp *"OB Baustein"* und fügen Sie nacheinander OB 57, OB 82 und OB 86 Ihrem Projekt hinzu.



3. Öffnen Sie im *"Katalog"* unter *"Bausteine"* *"Simple Motion Control"* und ziehen Sie per Drag&Drop folgende Bausteine in *"Programmbausteine"* des *Projektbaums*:

   - *Inverter* EtherCAT:
     - UDT 886 - VMC_ConfigInverterEC_REF
     - FB 886 - VMC_KernelInverter_EC
     - FB 887 - VMC_InitInverter_EC
   - Axis Control
     - UDT 860 - MC_AXIS_REF
     - Bausteine für die gewünschten Bewegungsabläufe

Achs-DB anlegen

1. Fügen Sie Ihrem Projekt einen neuen DB als *Achs-DB* hinzu. Klicken Sie hierzu im *Projektbaum* innerhalb der CPU unter *"PLC-Programm"*, *"Programmbausteine"* auf *"Neuen Baustein hinzufügen"*, wählen Sie den Bausteintyp *"DB Baustein"* und vergeben Sie diesem den Namen "Axis01". Die DB-Nr. können Sie frei wählen wie z.B. DB 10.

⇨ Der Baustein wird angelegt und geöffnet.

2. ■ Legen Sie in "Axis01" die Variable "Config" vom Typ UDT 886 an. Dies sind spezifische Achs-Konfigurationsdaten.
■ Legen Sie in "Axis01" die Variable "Axis" vom Typ UDT 860 an. Während des Betriebs werden hier alle Betriebsdaten der Achse abgelegt.

Axis01 [DB10]
Bausteinstruktur

		Adr...	Name	Datentyp	...
		...	Config	UDT	[886]
		...	Axis	UDT	[860]

OB 1

Konfiguration der Achse

Öffnen Sie den OB 1 und programmieren Sie folgende FB-Aufrufe mit zugehörigen DBs:

FB 887 - VMC_InitInverter_EC, DB 887 *Kap. 15.7.5.3 "FB 887 - VMC_InitInverter_EC - Frequenzumrichter EtherCAT Initialisierung" Seite 750*

Geben Sie unter *InputsStartAddressPDO* bzw. *OutputsStartAddressPDO* die Adresse aus dem *SPEED7 EtherCAT Manager* an. *744*

```
⇨ CALL  "VMC_InitInverter_EC" , "DI_InitInvEC01"
  Enable                :="InitInvEC1_Enable"
  LogicalAddress        :=300
  InputsStartAddressPDO :=300 (EtherCAT-Man.: E-Adresse S7)
  OutputsStartAddressPDO:=300 (EtherCAT-Man.: A-Adresse S7)
  MaxVelocityDrive      :=1.000000e+002
  MaxOutputFrequency    :=6.000000e+001
  NumberOfPoles         :=6
  Valid                 :="InitInvEC1_Valid"
  Error                 :="InitInvEC1_Error"
  ErrorID               :="InitInvEC1_ErrorID"
  MaxVelocity           :="InitInvEC1_MaxVelocityRPM"
  Config                :="Axis01".Config
  Axis                  :="Axis01".Axis
```

Kernel für Achse beschalten

Der *Kernel* verarbeitet die Benutzerkommandos und gibt sie entsprechend aufbereitet an den Antrieb über das jeweilige Bussystem weiter.

FB 886 - VMC_KernelInverter_EC, DB 886 *Kap. 15.7.5.2 "FB 886 - VMC_KernelInverter_EC - Frequenzumrichter EtherCAT Kernel" Seite 750*

```
⇨       CALL  "VMC_KernelInverter_EC" , "DI_KernelInvEC01"
         Init  :="KernelInvEC1_Init"
         Config:="Axis01".Config
         Axis  :="Axis01".Axis
```

Baustein für Bewegungsabläufe beschalten

Zur Vereinfachung soll hier die Beschaltung des FB 860 - VMC_AxisControl gezeigt werden. Dieser Universalbaustein unterstützt einfache Bewegungskommandos und liefert Statusmeldungen zurück. Die Ein- und Ausgänge können Sie individuell beschalten. Bitte geben Sie unter *"Axis"* die Referenz zu den entsprechenden Achsdaten im *Achs-DB* an.

FB 860 - VMC_AxisControl, DB 860 *Kap. 15.8.2.2 "FB 860 - VMC_AxisControl - Control-Baustein Achskontrolle" Seite 754*

```
⇨       CALL  "VMC_AxisControl" , "DI_AxisControl01"
         AxisEnable          :="AxCtrl1_AxisEnable"
         AxisReset           :="AxCtrl1_AxisReset"
         HomeExecute*        :="AxCtrl1_HomeExecute"
         HomePosition*       :="AxCtrl1_HomePosition"
         StopExecute         :="AxCtrl1_StopExecute"
         MvVelocityExecute   :="AxCtrl1_MvVelExecute"
         MvRelativeExecute*  :="AxCtrl1_MvRelExecute"
         MvAbsoluteExecute*  :="AxCtrl1_MvAbsExecute"
         PositionDistance*   :="AxCtrl1_PositionDistance"
         Velocity            :="AxCtrl1_Velocity"
         Acceleration        :="AxCtrl1_Acceleration"
         Deceleration        :="AxCtrl1_Deceleration"
         JogPositive         :="AxCtrl1_JogPositive"
         JogNegative         :="AxCtrl1_JogNegative"
         JogVelocity         :="AxCtrl1_JogVelocity"
         JogAcceleration     :="AxCtrl1_JogAcceleration"
         JogDeceleration     :="AxCtrl1_JogDeceleration"
         AxisReady           :="AxCtrl1_AxisReady"
         AxisEnabled         :="AxCtrl1_AxisEnabled"
         AxisError           :="AxCtrl1_AxisError"
         AxisErrorID         :="AxCtrl1_AxisErrorID"
         DriveWarning        :="AxCtrl1_DriveWarning"
         DriveError          :="AxCtrl1_DriveError"
         DriveErrorID        :="AxCtrl1_DriveErrorID"
         IsHomed*            :="AxCtrl1_IsHomed"
         ModeOfOperation     :="AxCtrl1_ModeOfOperation"
         PLCopenState        :="AxCtrl1_PLCopenState"
         ActualPosition*     :="AxCtrl1_ActualPosition"
         ActualVelocity      :="AxCtrl1_ActualVelocity"
         CmdDone             :="AxCtrl1_CmdDone"
         CmdBusy             :="AxCtrl1_CmdBusy"
         CmdAborted          :="AxCtrl1_CmdAborted"
         CmdError            :="AxCtrl1_CmdError"
         CmdErrorID          :="AxCtrl1_CmdErrorID"
         DirectionPositive   :="AxCtrl1_DirectionPos"
         DirectionNegative   :="AxCtrl1_DirectionNeg"
         SWLimitMinActive*   :="AxCtrl1_SWLimitMinActive"
         SWLimitMaxActive*   :="AxCtrl1_SWLimitMaxActive"
         HWLimitMinActive*   :="AxCtrl1_HWLimitMinActive"
         HWLimitMaxActive*   :="AxCtrl1_HWLimitMaxActive"
         Axis                :="Axis01".Axis
```

```
*) Dieser Parameter wird von einem Frequenzumrichter nicht
unterstützt.
```

Für komplexe Bewegungsaufgaben können Sie die PLCopen-Bausteine verwenden. Hier müssen Sie ebenfalls unter Axis die Referenz zu den Achsdaten im Achs-DB angeben.

Ihr Projekt beinhaltet nun folgende Bausteine:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1

- OB 86 - Rack_FLT
- FB 860 - VMC_AxisControl mit Instanz-DB
- FB 886 - VMC_KernelInverter_EC mit Instanz-DB
- FB 887 - VMC_InitInverter_EC mit Instanz-DB
- UDT 860 - MC_Axis_REF
- UDT 886 - VMC_ConfigInverterEC_REF

Zeitlicher Ablauf

1. Wählen Sie *"Projekt ➔ Alles übersetzen"* und übertragen Sie das Projekt in Ihre CPU.

   ⇨ Sie können jetzt Ihre Applikation in Betrieb nehmen.

   VORSICHT!
   Bitte beachten Sie immer die Sicherheitshinweise zu ihrem Antrieb, insbesondere bei der Inbetriebnahme!

2. Bevor eine Achse gesteuert werden kann, muss diese initialisiert werden. Rufen Sie hierzu den *Init*-Baustein FB 887 - VMC_InitInverter_EC mit *Enable* = TRUE auf.

   ⇨ Der Ausgang *Valid* meldet TRUE zurück. Im Fehlerfall können Sie durch Auswertung der *ErrorID* den Fehler ermitteln.

   Den *Init*-Baustein müssen Sie erneut aufrufen, wenn Sie einen neuen Achs-DB laden oder Parameter am *Init*-Baustein geändert wurden.

   Fahren Sie erst fort, wenn der Init-Baustein keinen Fehler meldet!

3. Stellen Sie sicher, dass der *Kernel*-Baustein FB 886 - VMC_KernelInverter_EC zyklisch aufgerufen wird. Auf diese Weise werden Steuersignale an den Antrieb übergeben und Statusmeldungen übermittelt.

4. Programmieren Sie Ihre Applikation mit dem FB 860 - VMC_AxisControl oder mit den PLCopen Bausteinen.

Steuerung des Antriebs über HMI

Sie haben die Möglichkeit über ein HMI Ihren Antrieb zu steuern. Hierzu gibt es für Movicon eine vorgefertigte Symbolbibliothek für den Zugriff auf den VMC_AxisControl Funktionsbaustein. ↬ *Kap. 15.9 "Antrieb über HMI steuern" Seite 825*

15.7.5 Antriebsspezifische Bausteine

Die PLCopen-Bausteine zur Achskontrolle finden Sie hier: Kap. 15.8 "Bausteine zur Achskontrolle" Seite 751

15.7.5.1 UDT 886 - VMC_ConfigInverterEC_REF - Frequenzumrichter EtherCAT Datenstruktur Achskonfiguration

Dies ist eine benutzerdefinierte Datenstruktur, die Informationen zu den Konfigurationsdaten beinhaltet. Die UDT ist speziell angepasst an die Verwendung eines Frequenzumrichters, welcher über EtherCAT angebunden ist.

15.7.5.2 FB 886 - VMC_KernelInverter_EC - Frequenzumrichter EtherCAT Kernel

Beschreibung

Dieser Baustein setzt die Antriebskommandos für einen Frequenzumrichter über EtherCAT um und kommuniziert mit dem Antrieb. Je Frequenzumrichter ist eine Instanz dieses FBs zyklisch aufzurufen.

Bitte beachten Sie, dass dieser Baustein intern den SFB 238 aufruft.

Im SPEED7 Studio wird dieser Baustein automatisch in Ihr Projekt eingefügt.

Parameter	Deklaration	Datentyp	Beschreibung
Init	INPUT	BOOL	Mit einer Flanke 0-1 wird der Baustein intern zurückgesetzt. Hierbei werden bestehende Bewegungskommandos abgebrochen und der Baustein wird initialisiert.
Config	IN_OUT	UDT 886	Datenstruktur zur Übergabe von achsabhängigen Konfigurationsdaten an den *AxisKernel*.
Axis	IN_OUT	UDT 860	Datenstruktur zur Übergabe von achsabhängigen Informationen an *AxisKernel* und PLCopen-Bausteine.

15.7.5.3 FB 887 - VMC_InitInverter_EC - Frequenzumrichter EtherCAT Initialisierung

Beschreibung

Dieser Baustein dient zur Konfiguration der Achse. Der Baustein ist speziell angepasst an die Verwendung eines Frequenzumrichters, welcher über EtherCAT angebunden ist.

Parameter	Deklaration	Datentyp	Beschreibung
Enable	INPUT	BOOL	Freigabe der Initialisierung
LogicalAddress	INPUT	INT	Startadresse der PDO-Eingangsdaten
InputsStartAddressPDO	INPUT	INT	Startadresse der Eingabe-PDOs
OutputsStartAddressPDO	INPUT	INT	Startadresse der Ausgabe-PDOs
MaxVelocityDrive	INPUT	REAL	Maximale Geschwindigkeit der Applikation [u].

Parameter	Deklaration	Datentyp	Beschreibung
MaxOutputFrequency	INPUT	REAL	Maximale Ausgabefrequenz [Hz]. Bitte hier den Wert aus dem Softwaretool *Drive Wizard+* übernehmen.
NumberOfPoles	INPUT	INT	Anzahl der Pole. Bitte hier den Wert aus dem Softwaretool *Drive Wizard+* übernehmen.
Valid	OUTPUT	BOOL	Initialisierung ■ TRUE: Initialisierung ist gültig.
Error	OUTPUT	BOOL	■ Fehler – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden. Die Achse wird gesperrt.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
MaxVelocity	OUTPUT	INT	Maximale Geschwindigkeit [rpm]. Dieser Wert wird automatisch ermittelt.
Config	IN_OUT	UDT 886	Datenstruktur zur Übergabe von achsabhängigen Konfigurationsdaten an den *AxisKernel*.
Axis	IN_OUT	UDT 860	Datenstruktur zur Übergabe von achsabhängigen Informationen an *AxisKernel* und PLCopen-Bausteine.

15.8 Bausteine zur Achskontrolle

15.8.1 Übersicht

Unter Axis Control finden Sie die Bausteine zur Programmierung von Bewegungsaufgaben und Statusabfragen. Die nachfolgend aufgeführten Bausteine können ausschließlich zur Ansteuerung folgender Antriebssysteme verwendet werden.

- *Sigma-5 EtherCAT*
- *Sigma-7S EtherCAT*
- *Sigma-7W EtherCAT*
- *Sigma-5/7 PROFINET*
- *Frequenzumrichter (Inverter) über EtherCAT*

Bitte beachten Sie, dass es hier ebenfalls zu Einschränkungen kommt. Die unterstützten Bausteine können Sie der nachfolgenden Tabelle entnehmen.

Einfache Bewegungsaufgaben

Unterstützte Bausteine	Sigma-5/7 PROFINET	Sigma-5/7 EtherCAT	Inverter EtherCAT	Seite
UDT 860 - MC_AXIS_REF - Datenstruktur für Achse	ja	ja	ja	*754*
FB 860 - VMC_AxisControl - Steuerung von Antriebfunktionen und Auslesen von Antriebszuständen	nein	ja	ja	*754*

Komplexe Bewegungsaufgaben - PLCopen-Bausteine

Unterstützte Bausteine	Sigma-5/7 PROFINET	Sigma-5/7 EtherCAT	Inverter EtherCAT	Seite
UDT 860 - MC_AXIS_REF - Datenstruktur für Achse	ja	ja	ja	*758*
UDT 861 - MC_TRIGGER_REF - Datenstruktur	nein	ja	nein	*758*
FB 800 - MC_Power - Achse freigeben bzw. sperren	nein	ja	ja	*759*
FB 801 - MC_Home - Achse referenzieren	nein	ja	nein	*761*
FB 802 - MC_Stop - Achse stoppen	nein	ja	ja	*763*
FB 803 - MC_Halt - Achse anhalten	nein	ja	ja	*765*
FB 804 - MC_MoveRelative - Achse relativ verfahren	nein	ja	nein	*767*
FB 805 - MC_MoveVelocity - Achse verfahren mit konstanter Geschwindigkeit	nein	ja	ja	*769*
FB 808 - MC_MoveAbsolute - Achse auf absolute Position verfahren	nein	ja	nein	*771*
FB 811 - MC_Reset - Achse zurücksetzen	nein	ja	ja	*774*
FB 812 - MC_ReadStatus - PLCopen-State der Achse lesen	nein	ja	ja	*776*
FB 813 - MC_ReadAxisError - Fehler von Achse lesen	nein	ja	ja	*778*
FB 814 - MC_ReadParameter - Parameter der Achse lesen	ja	ja	ja	*780*
FB 815 - MC_WriteParameter - Parameter an Achse schreiben	ja	ja	ja	*782*
FB 816 - MC_ReadActualPosition - Aktuelle Position der Achse lesen	nein	ja	nein	*784*
FB 817 - MC_ReadActualVelocity - Aktuelle Geschwindigkeit der Achse lesen	nein	ja	ja	*786*
FB 818 - MC_ReadAxisInfo - Zusatzinformationen der Achse lesen	nein	ja	ja	*788*
FB 819 - MC_ReadMotionState - Zustand Bewegungsauftrag lesen	nein	ja	ja	*790*
FB 823 - MC_TouchProbe - Achsposition erfassen	ja	ja	nein	*792*
FB 824 - MC_AbortTrigger - Achsposition erfassen abbrechen	ja	ja	nein	*794*
FB 825 - MC_ReadBoolParameter - Boolean-Parameter von Achse lesen	ja	ja	ja	*795*
FB 826 - MC_WriteBoolParameter - Boolean-Parameter an Achse schreiben	ja	ja	ja	*797*
FB 827 - VMC_ReadDWordParameter - Doppelwort-Parameter von Achse lesen	ja	ja	ja	*799*
FB 828 - VMC_WriteDWordParameter - Doppelwort-Parameter an Achse schreiben	ja	ja	ja	*801*
FB 829 - VMC_ReadWordParameter - Wort-Parameter von Achse lesen	ja	ja	ja	*803*
FB 830 - VMC_WriteWordParameter - Wort-Parameter an Achse schreiben	ja	ja	ja	*805*
FB 831 - VMC_ReadByteParameter - Byte-Parameter von Achse lesen	ja	ja	ja	*807*
FB 832 - VMC_WriteByteParameter - Byte-Parameter an Achse schreiben	ja	ja	ja	*809*
FB 833 - VMC_ReadDriveParameter - Antriebsparameter lesen	ja	ja	ja	*811*
FB 834 - VMC_WriteDriveParameter - Antriebsparameter schreiben	ja	ja	ja	*813*
FB 835 - VMC_HomeInit_LimitSwitch - Initialisierung Referenzfahrt auf Endschalter	ja	ja	nein	*815*
FB 836 - VMC_HomeInit_HomeSwitch - Initialisierung Referenzfahrt auf Referenzschalter	ja	ja	nein	*817*

Unterstützte Bausteine	Sigma-5/7 PROFINET	Sigma-5/7 EtherCAT	Inverter EtherCAT	Seite
FB 837 - VMC_HomeInit_ZeroPulse - Initialisierung Referenzfahrt auf Nullimpuls	ja	ja	nein	*820*
FB 838 - VMC_HomeInit_SetPosition - Initialisierung Referenzfahrt setze Position	ja	ja	nein	*822*

15.8.2 Einfache Bewegungsaufgaben

15.8.2.1 UDT 860 - MC_AXIS_REF - Datenstruktur Achsdaten

Dies ist eine benutzerdefinierte Datenstruktur, die Statusinformationen der Achse beinhaltet.

15.8.2.2 FB 860 - VMC_AxisControl - Control-Baustein Achskontrolle

Beschreibung

Mit dem FB *VMC_AxisControl* können Sie die angebundene Achse steuern. Sie können den Status des Antriebs abrufen, den Antrieb ein- bzw. ausschalten oder verschiedene Bewegungskommandos ausführen. In den Instanzdaten des Bausteins befindet sich ein gesonderter Speicherbereich. Über diesen können Sie mittels eines HMI Ihre Achse steuern. ↳ *Kap. 15.9 "Antrieb über HMI steuern" Seite 825*

Der Baustein VMC_AxisControl sollte nie gleichzeitig mit dem PLCopen-Baustein MC_Power verwendet werden. Da der VMC_AxisControl Funktionalitäten des MC_Power beinhaltet und immer der aktuellste Befehl vom VMC_Kernel-Baustein ausgeführt wird, kann dies zu einem Fehlverhalten des Antriebs führen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
AxisEnable	INPUT	BOOL	■ Achsenfreigabe – TRUE: Die Achse wird freigegeben. – FALSE: Die Achse wird gesperrt.
AxisReset	INPUT	BOOL	■ Reset Achse – Flanke 0-1: Reset der Achse wird durchgeführt.
HomeExecute	INPUT	BOOL	■ Referenzfahrt – Flanke 0-1: Referenzfahrt wird gestartet.
HomePosition	INPUT	REAL	Bei erfolgreicher Referenzierung wird die Istposition der Achse einmalig gleich Position gesetzt. Die Position ist in der verwendeten Anwendereinheit anzugeben.
StopExecute	INPUT	BOOL	■ Achse stoppen – Flanke 0-1: Stoppen der Achse wird gestartet.
MvVelocityExecute	INPUT	BOOL	■ Verfahren der Achse starten – Flanke 0-1: Die Achse wird auf die angegebene Geschwindigkeit beschleunigt / abgebremst.
MvRelativeExecute	INPUT	BOOL	■ Verfahren der Achse starten – Flanke 0-1: Die relative Positionierung der Achse wird gestartet.
MvAbsoluteExecute	INPUT	BOOL	■ Verfahren der Achse starten – Flanke 0-1: Die absolute Positionierung der Achse wird gestartet.
Direction*	INPUT	BYTE	Modus für absolute Positionierung: ■ 0: kürzester Weg ■ 1: positive Richtung ■ 2: negative Richtung ■ 3: aktuelle Richtung

Parameter	Deklaration	Datentyp	Beschreibung
PositionDistance	INPUT	REAL	Absolute Position bzw. relative Wegstrecke je nach Kommando in [Anwendereinheiten].
Velocity	INPUT	REAL	Geschwindigkeitsvorgabe (vorzeichenbehafteter Wert) in [Anwendereinheiten/s].
Acceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2].
Deceleration	INPUT	REAL	Verzögerung in [Anwendereinheiten/s^2].
JogPositive	INPUT	BOOL	■ Achse mit konstanter Geschwindigkeit in positive Richtung verfahren – Flanke 0-1: Das Verfahren der Achse mit konstanter Geschwindigkeit wird gestartet. – Flanke 1-0: Die Achse wird gestoppt.
JogNegative	INPUT	BOOL	■ Achse mit konstanter Geschwindigkeit in negative Richtung verfahren – Flanke 0-1: Das Verfahren der Achse mit konstanter Geschwindigkeit wird gestartet. – Flanke 1-0: Die Achse wird gestoppt.
JogVelocity	INPUT	REAL	Geschwindigkeitsvorgabe für Jogging (positiver Wert) in [Anwendereinheiten/s].
JogAcceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2].
JogDeceleration	INPUT	REAL	Verzögerung für Jogging in [Anwendereinheiten/s^2].
AxisReady	OUTPUT	BOOL	■ AxisReady – TRUE: Die Achse ist einschaltbereit. – FALSE: Die Achse ist nicht einschaltbereit. → Prüfe und behebe *AxisError* (siehe AxisErrorID). → Prüfe und behebe *DriveError* (siehe DriveErrorID). → Prüfe Initialisierungs FB (Input- und Output Addressen bzw. PDO Mapping richtig?)
AxisEnabled	OUTPUT	BOOL	■ Status Achse – TRUE: Achse ist eingeschaltet und nimmt Bewegungsaufträge an. – FALSE: Achse ist nicht eingeschaltet und nimmt keine Bewegungsaufträge an.
AxisError	OUTPUT	BOOL	■ Fehler bei Motion Achse – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *AxisErrorID* entnommen werden. → Die Achse wird gesperrt.
AxisErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
DriveWarning	OUTPUT	BOOL	■ Warnung – TRUE: Der Antrieb liefert eine Warnung. Zusätzliche Informationen sind aus dem entsprechenden Handbuch des Herstellers zu entnehmen.

Parameter	Deklaration	Datentyp	Beschreibung
DriveError	OUTPUT	BOOL	■ Fehler direkt am Antrieb – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *DriveErrorID* entnommen werden. → Die Achse wird gesperrt.
DriveErrorID	OUTPUT	WORD	■ Fehler – TRUE: Der Antrieb liefert einen Fehler. Zusätzliche Informationen sind aus dem entsprechenden Handbuch des Herstellers zu entnehmen.
IsHomed	OUTPUT	BOOL	■ Information Achse: referenziert – TRUE: Die Achse ist referenziert.
ModeOfOperation	OUTPUT	INT	Antriebsspezifischer Modus. Weitere Infos siehe Antriebsmanual. Beispiel *Sigma-5*: 0: No mode changed/no mode assigned 1: Profile Position mode 2: Reserved (keep last mode) 3: Profile Velocity mode 4: Torque Profile mode 6: Homing mode 7: Interpolated Position mode 8: Cyclic Sync Position mode 9: Cyclic Sync Velocity mode 10: Cyclic Sync Torque mode Other Reserved (keep last mode)
PLCopenState	OUTPUT	INT	Aktueller PLCopenState: 1: Disabled 2: Standstill 3: Homing 4: Discrete Motion 5: Continous Motion 7: Stopping 8: Errorstop
ActualPosition	OUTPUT	REAL	Position der Achse in [Anwendereinheit].
ActualVelocity	OUTPUT	REAL	Geschwindigkeit der Achse in [Anwendereinheit/s].
CmdDone	OUTPUT	BOOL	■ Status – TRUE: Auftrag wurde ohne Fehler beendet.
CmdBusy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung.

Parameter	Deklaration	Datentyp	Beschreibung
CmdAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen.
CmdError	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *CmdErrorID* entnommen werden.
CmdErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
DirectionPositive	OUTPUT	BOOL	■ Zustand Bewegungsauftrag: Position zunehmend – TRUE: Die Position der Achse nimmt zu.
DirectionNegative	OUTPUT	BOOL	■ Zustand Bewegungsauftrag: Position abnehmend – TRUE: Die Position der Achse nimmt ab.
SWLimitMinActive	OUTPUT	BOOL	■ Software Endschalter – TRUE: Software Endschalter Minimum aktiv (Minimale Position in negative Richtung überschritten).
SWLimitMaxActive	OUTPUT	BOOL	■ Software Endschalter – TRUE: Software Endschalter Maximum aktiv (Maximale Position in positive Richtung überschritten).
HWLimitMinActive	OUTPUT	BOOL	■ Hardware Endschalter – TRUE: Negativer Hardware Endschalter am Antrieb aktiv (NOT- Negative Overtravel).
HWLimitMaxActive	OUTPUT	BOOL	■ Hardware Endschalter – TRUE: Positiver Hardware Endschalter am Antrieb aktiv (POT- Positive Overtravel).
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse.

*) Dieser Parameter wird nicht von allen Antrieben unterstützt, z.B. *Sigma 5 über EtherCAT* unterstützt diesen Parameter nicht.

15.8.3 Komplexe Bewegungsaufgaben - PLCopen-Bausteine

15.8.3.1 UDT 860 - MC_AXIS_REF - Datenstruktur Achsdaten

Dies ist eine benutzerdefinierte Datenstruktur, die Statusinformationen der Achse beinhaltet.

15.8.3.2 UDT 861 - MC_TRIGGER_REF - Datenstruktur Triggersignal

Diese ist eine benutzerdefinierte Datenstruktur, die Informationen zum Triggersignal beinhaltet.

15.8.3.3 FB 800 - MC_Power - Achsenfreigabe

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: *Kap. 15.8.1 "Übersicht" Seite 751*

Mit MC_Power kann eine Achse freigegeben bzw. gesperrt werden.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Enable	INPUT	BOOL	■ Achsenfreigabe – TRUE: Die Achse wird freigegeben – FALSE: Die Achse wird gesperrt
EnablePositive	INPUT	BOOL	Parameter aktuell nicht unterstützt; Aufruf mit FALSE
EnableNegative	INPUT	BOOL	Parameter aktuell nicht unterstützt; Aufruf mit FALSE
Status	OUTPUT	BOOL	■ Status Achse – TRUE: Achse nimmt Bewegungsaufträge an – FALSE: Achse nimmt keine Bewegungsaufträge an
Valid	OUTPUT	BOOL	Immer FALSE
Error	OUTPUT	BOOL	■ Fehler – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden. Die Achse wird gesperrt.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

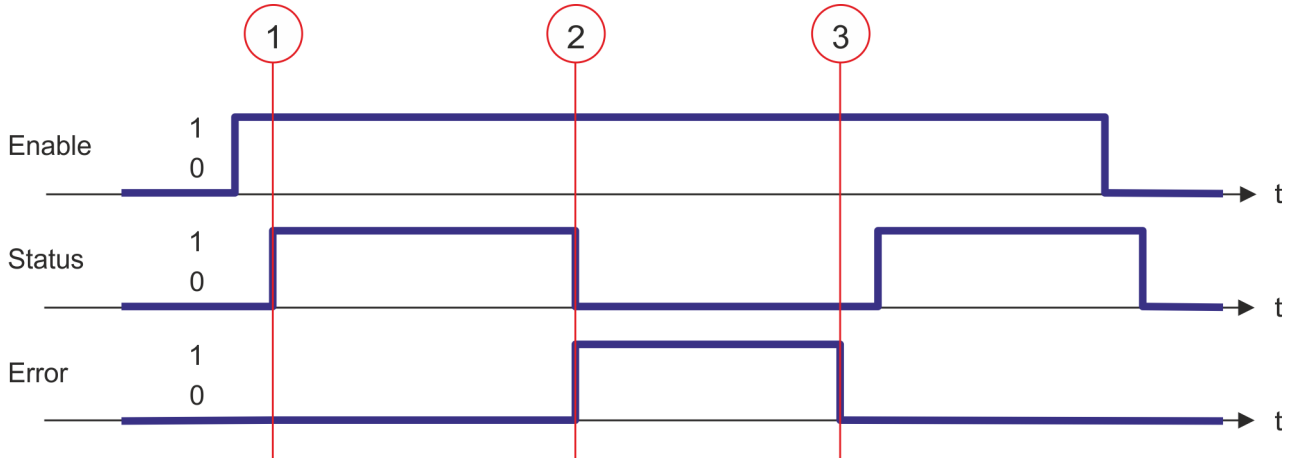
Achse freigeben

Aufruf von MC_Power mit *Enable* = TRUE. Sobald *Status* den Wert TRUE zeigt, ist die Achse freigegeben. In diesem Zustand können Bewegungsaufträge aktiviert werden.

Achse sperren

Aufruf von MC_Power mit *Enable* = FALSE. Sobald *Status* den Wert FALSE zeigt, ist die Achse gesperrt. Bei Sperren der Achse wird ein ggf. aktiver Bewegungsauftrag abgebrochen und die Achse gestoppt.

Zustandsdiagramm der Bausteinparameter



(1) Die Achse wird mit *Enable* = TRUE freigegeben. Zum Zeitpunkt (1) ist die Freigabe erfolgt. Anschließend können Bewegungsaufträge aktiviert werden.
(2) Zum Zeitpunkt (2) tritt ein Fehler auf, der das Sperren der Achse zur Folge hat. Ein ggf. aktiver Bewegungsauftrag wird abgebrochen und die Achse gestoppt.
(3) Der Fehler wird beseitigt und zum Zeitpunkt (3) quittiert. Da *Enable* weiterhin gesetzt ist, wird die Achse wieder freigegeben. Zuletzt wird die Achse mit *Enable* = FALSE gesperrt.

15.8.3.4 FB 801 - MC_Home - Achse referenzieren

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_Home kann eine Achse referenziert werden. Dadurch kann ein Bezug zwischen der Position der Achse und der mechanischen Stellung hergestellt werden. Die Referenzfahrt-Methode und die zugehörigen Parameter müssen Sie direkt am Antrieb konfigurieren. Verwenden Sie hierzu die VMC_HomeInit_... Bausteine.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Referenzfahrt – Flanke 0-1: Referenzfahrt wird gestartet
Position	INPUT	REAL	Bei erfolgreicher Referenzierung wird die Istposition der Achse einmalig gleich *Position* gesetzt. *Position* ist in der verwendeten Anwendereinheit anzugeben.
BufferMode	INPUT	BYTE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen.
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

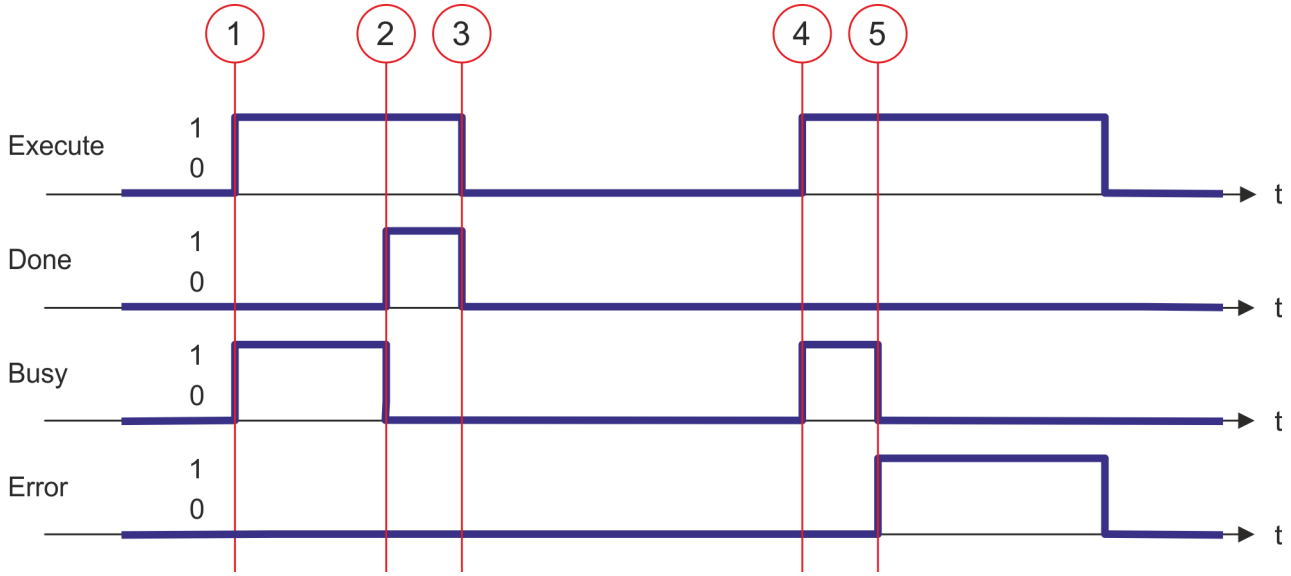
Start des Auftrags nur im PLCopen-State *Standstill* möglich.

Achse referenzieren

Mit einer Flanke 0-1 an *Execute* wird die Referenzierung gestartet. Solange die Referenzierung läuft zeigt *Busy* den Wert TRUE. Sobald *Done* den Wert TRUE hat, ist die Referenzierung erfolgreich abgeschlossen. Die Istposition der Achse wurde auf den Wert von *Position* gesetzt.

– *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.*
– *Ein laufender Auftrag kann durch einen Bewegungsauftrag (z.B. MC_MoveRelative) nicht abgebrochen werden.*

Zustandsdiagramm der Bausteinparameter



(1) Mit Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird die Referenzierung gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist die Referenzierung abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.
(4) Zum Zeitpunkt (4) wird erneut die Referenzierung mit einer Flanke 0-1 an *Execute* gestartet und *Busy* liefert den Wert TRUE.
(5) Zum Zeitpunkt (5) tritt ein Fehler bei der Referenzierung auf. *Busy* liefert den Wert FALSE und *Error* den Wert TRUE.

15.8.3.5 FB 802 - MC_Stop - Achse stoppen

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_Stop wird die Achse gestoppt. Mit dem Parameter *Deceleration* kann das dynamische Verhalten beim Stoppvorgang bestimmt werden.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Achse stoppen – Flanke 0-1: Stoppen der Achse wird gestartet
Deceleration	INPUT	REAL	Verzögerung beim Stoppen in [Anwendereinheiten/s^2]
Jerk	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

- Start des Auftrags in den PLCopen-States *Standstill*, *Homing*, *Discrete Motion* und *Continuous Motion* möglich.
- MC_Stop führt die Achse in den PLCopen-State *Stopping* über. In *Stopping* können keine Bewegungsaufträge gestartet werden. Solange *Execute* gleich TRUE ist, bleibt die Achse im PLCopen-State *Stopping*. Wird *Execute* gleich FALSE gesetzt, geht die Achse in den PLCopen-State *Standstill* über. In *Standstill* können Bewegungsaufträge gestartet werden.

Achse stoppen

Mit einer Flanke 0-1 an *Execute* wird das Stoppen der Achse gestartet. Solange das Stoppen der Achse läuft, zeigt *Busy* den Wert TRUE. Nachdem die Achse gestoppt wurde und somit die Geschwindigkeit 0 erreicht hat, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE bis zum Stopp der Achse ausgeführt.*
- *Ein laufender Auftrag kann durch einen Bewegungsauftrag (z.B. MC_MoveRelative) nicht abgebrochen werden.*

Zustandsdiagramm der Bausteinparameter

(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Stoppen der Achse gestartet und *Busy* liefert den Wert TRUE. Die Geschwindigkeit der Achse wird unter Berücksichtigung des Parameters *Deceleration* bis auf null verringert.
(2) Zum Zeitpunkt (2) ist das Stoppen der Achse abgeschlossen, die Achse ist gestoppt. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.6 FB 803 - MC_Halt - Achse anhalten

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_Halt wird die Achse bis zum Stillstand abgebremst. Mit dem Parameter *Deceleration* kann das dynamische Verhalten beim Bremsvorgang bestimmt werden.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Achse anhalten – Flanke 0-1: Anhalten der Achse wird gestartet
Deceleration	INPUT	REAL	Verzögerung beim Bremsen in [Anwendereinheiten/s^2]
Jerk	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
BufferMode	INPUT	BYTE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Active	OUTPUT	BOOL	■ Status – TRUE: Baustein hat die Kontrolle über die Achse
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

- Start des Auftrags in den PLCopen-States *Discrete Motion* und *Continuous Motion* möglich.
- MC_Halt führt die Achse in den PLCopen-State *Discrete Motion* über.

Achse anhalten

Mit einer Flanke 0-1 an *Execute* wird das Anhalten der Achse gestartet. Solange das Anhalten der Achse läuft, zeigt *Busy* den Wert TRUE. Nachdem die Achse angehalten wurde und somit die Geschwindigkeit 0 erreicht hat, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE bis zum Anhalten der Achse ausgeführt.*
- *Ein laufender Auftrag kann durch einen anderen Bewegungsauftrag (z.B. MC_MoveRelative) abgebrochen werden.*

Zustandsdiagramm der Bausteinparameter

(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Anhalten der Achse gestartet und *Busy* liefert den Wert TRUE. Die Geschwindigkeit der Achse wird unter Berücksichtigung des Parameters *Deceleration* bis auf 0 verringert.
(2) Zum Zeitpunkt (2) ist das Anhalten der Achse abgeschlossen, die Achse steht. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.7 FB 804 - MC_MoveRelative - Achse relativ verfahren

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_MoveRelative wird die Achse relativ zu der Position bei Auftragsstart um eine spezifizierte Distanz verfahren. Mit den Parametern *Velocity*, *Acceleration* und *Deceleration* wird das dynamische Verhalten beim Bewegungsvorgang bestimmt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Achse relativ verfahren – Flanke 0-1: Das relative Verfahren der Achse wird gestartet
ContinuousUpdate	INPUT	BOOL	Parameter aktuell nicht unterstützt; Aufruf mit FALSE
Distance	INPUT	REAL	Relative Wegstrecke in [Anwendereinheiten]
Velocity	INPUT	REAL	Max. Geschwindigkeit (muss nicht zwingend erreicht werden) in [Anwendereinheiten/s]
Acceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2]
Deceleration	INPUT	REAL	Verzögerung in [Anwendereinheiten/s^2]
Jerk	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
BufferMode	INPUT	BYTE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt; Zielposition erreicht
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Active	OUTPUT	BOOL	■ Status – TRUE: Baustein hat die Kontrolle über die Achse
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

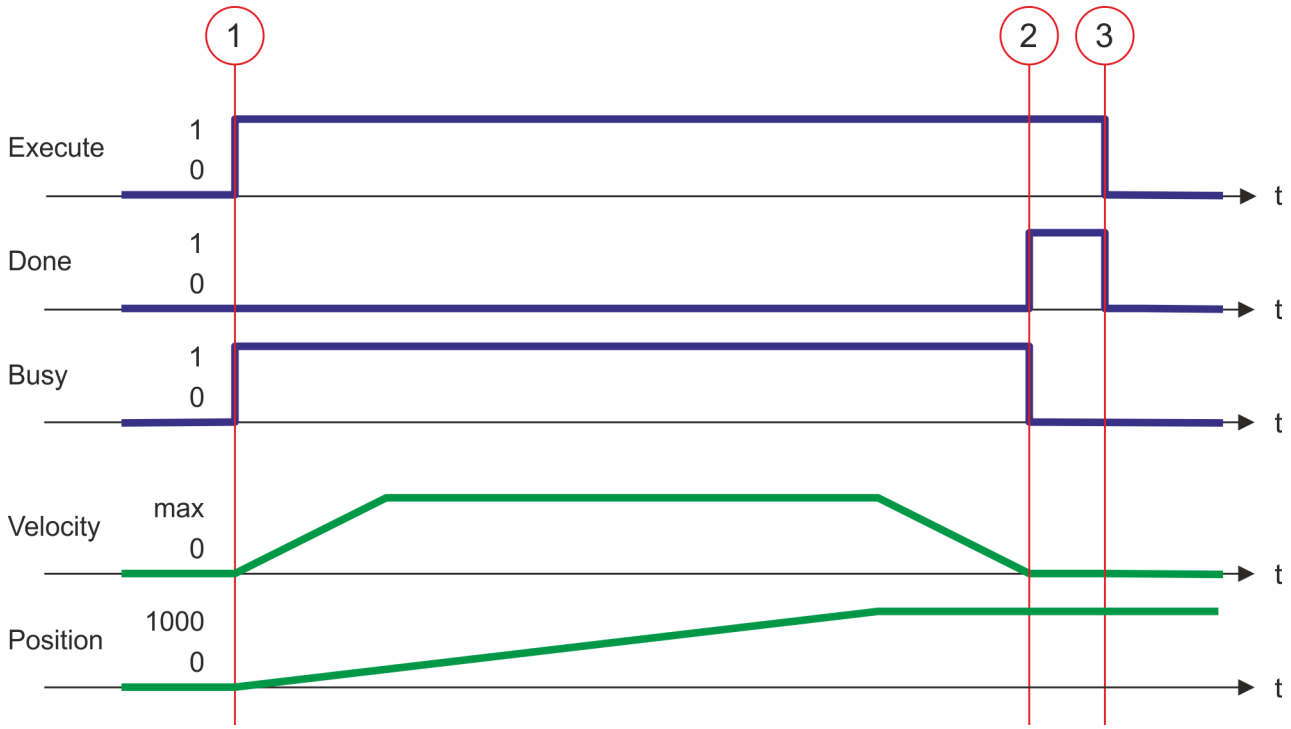
- Start des Auftrags in den PLCopen-States *Standstill*, *Discrete Motion* und *Continuous Motion* möglich.
- MC_MoveRelative führt die Achse in den PLCopen-State *Discrete Motion* über.

Achse relativ verfahren

Mit einer Flanke 0-1 an *Execute* wird das Verfahren der Achse gestartet. Solange das Verfahren der Achse läuft, zeigt *Busy* den Wert TRUE. Nachdem die Zielposition erreicht wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Die Geschwindigkeit der Achse ist dann gleich 0.

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE bis die Achse die Zielposition erreicht hat, ausgeführt.*
- *Ein laufender Auftrag kann durch einen anderen Bewegungsauftrag (z.B. MC_MoveAbsolute) abgebrochen werden.*

Zustandsdiagramm der Bausteinparameter



(1) Die Achse wird mit MC_MoveRelative um eine *Distance* = 1000.0 verfahren (Startposition bei Auftragsstart gleich 0.0). Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Verfahren der Achse gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) wurde die Achse um die *Distance* = 1000.0 verfahren, d.h. die Zielposition wurde erreicht. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.8 FB 805 - MC_MoveVelocity - Achse verfahren mit konstanter Geschwindigkeit

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_MoveVelocity wird die Achse mit einer konstanten Geschwindigkeit verfahren. Mit den Parametern *Velocity*, *Acceleration* und *Deceleration* wird das dynamische Verhalten beim Bewegungsvorgang bestimmt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Achse mit konstanter Geschwindigkeit verfahren – Flanke 0-1: Das Verfahren der Achse mit konstanter Geschwindigkeit wird gestartet
ContinuousUpdate	INPUT	BOOL	Parameter aktuell nicht unterstützt; Aufruf mit FALSE
Velocity	INPUT	REAL	Geschwindigkeitsvorgabe (vorzeichenbehafteter Wert) in [Anwendereinheiten/s]
Acceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2]
Deceleration	INPUT	REAL	Verzögerung in [Anwendereinheiten/s^2]
Jerk	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
BufferMode	INPUT	BYTE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0
InVelocity	OUTPUT	BOOL	■ Geschwindigkeitsvorgabe – TRUE: Geschwindigkeitsvorgabe erreicht
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Active	OUTPUT	BOOL	■ Status – TRUE: Baustein hat die Kontrolle über die Achse
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

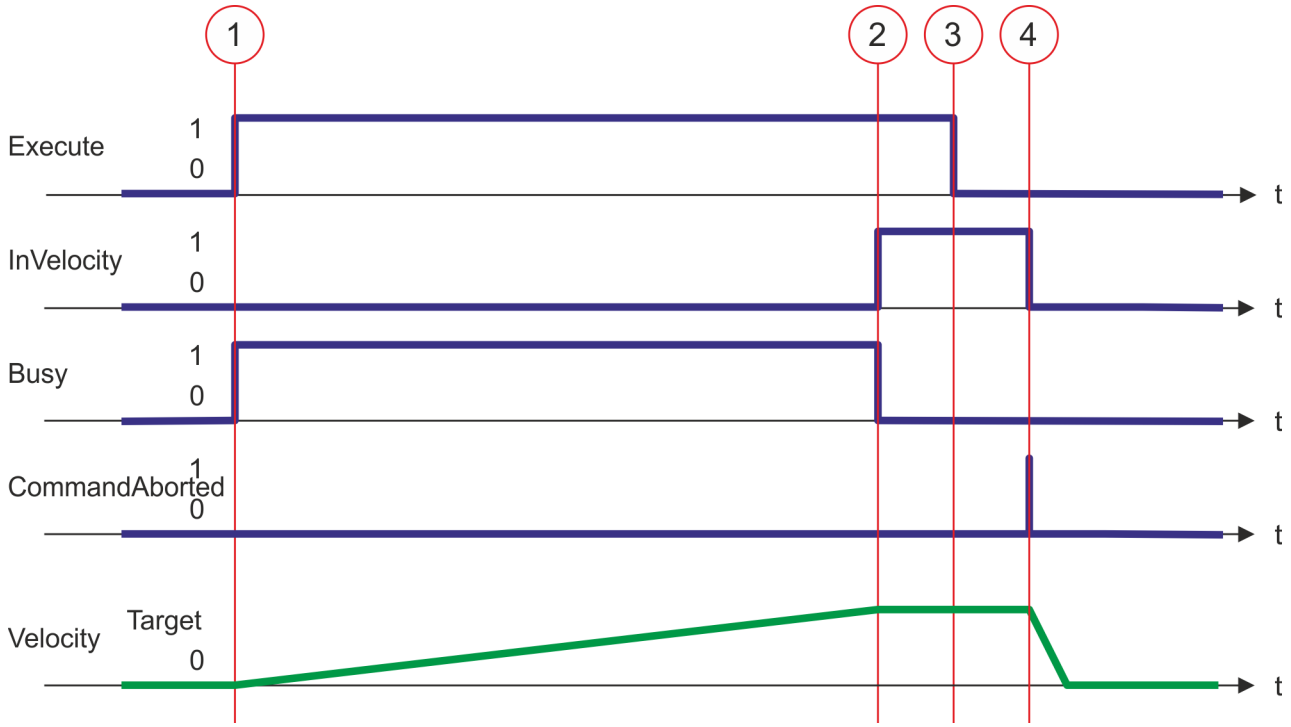
- Start des Auftrags in den PLCopen-States *Standstill*, *Discrete Motion* und *Continuous Motion* möglich.
- MC_MoveVelocity führt die Achse in den PLCopen-State *Continuous Motion* über.

Achse mit Geschwindigkeitsvorgabe verfahren

Mit einer Flanke 0-1 an *Execute* wird das Verfahren der Achse mit Geschwindigkeitsvorgabe gestartet. Solange die Geschwindigkeitsvorgabe nicht erreicht ist, zeigt *Busy* den Wert TRUE und *InVelocity* den Wert FALSE. Ist die Geschwindigkeitsvorgabe erreicht, wird *Busy* gleich FALSE und *InVelocity* gleich TRUE. Die Achse wird mit dieser Geschwindigkeit konstant weiter verfahren.

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt, auch wenn die Geschwindigkeitsvorgabe erreicht wurde.*
- *Ein laufender Auftrag kann durch einen anderen Bewegungsauftrag (z.B. MC_MoveAbsolute) abgebrochen werden.*

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Verfahren der Achse mit Geschwindigkeitsvorgabe gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) erreicht die Achse die Geschwindigkeitsvorgabe und *Busy* liefert den Wert FALSE und *InVelocity* den Wert TRUE.
(3) Das Rücksetzen von Execute auf FALSE zum Zeitpunkt (3) hat keine Auswirkung auf die Achse. Die Achse wird weiterhin konstant mit der Geschwindigkeitsvorgabe verfahren und *InVelocity* liefert weiterhin den Wert TRUE.
(4) Zum Zeitpunkt (4) wird der MC_Velocity-Auftrag durch einen MC_Halt-Auftrag abgebrochen. Die Achse wird bis zum Halt abgebremst.

15.8.3.9 FB 808 - MC_MoveAbsolute - Achse auf absolute Position verfahren

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_MoveAbsolute wird die Achse auf eine absolute Position verfahren. Mit den Parametern *Velocity*, *Acceleration* und *Deceleration* wird das dynamische Verhalten beim Bewegungsvorgang bestimmt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Verfahren der Achse starten – Flanke 0-1: Das Verfahren der Achse wird gestartet
ContinuousUpdate	INPUT	BOOL	Parameter aktuell nicht unterstützt; Aufruf mit FALSE
Position	INPUT	REAL	Absolute Position in [Anwendereinheiten]
Velocity	INPUT	REAL	Maximale Geschwindigkeit (muss nicht zwingend erreicht werden) vorzeichenbehafteter Wert in [Anwendereinheiten/s]
Acceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2]
Deceleration	INPUT	REAL	Verzögerung in [Anwendereinheiten/s^2]
Jerk	INPUT	REAL	Parameter aktuell nicht unterstützt; Aufruf mit 0.0
Direction	INPUT	Byte	■ Richtung – 0: Kürzeste Entfernung – 1: Positive Richtung – 2: Negative Richtung – 3: Aktuelle Richtung
BufferMode	INPUT	BYTE	Parameter aktuell nicht unterstützt; Aufruf mit B#16#0
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Die Zielposition wurde erreicht.
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Active	OUTPUT	BOOL	■ Status – TRUE: Baustein hat die Kontrolle über die Achse
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.

Parameter	Deklaration	Datentyp	Beschreibung
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

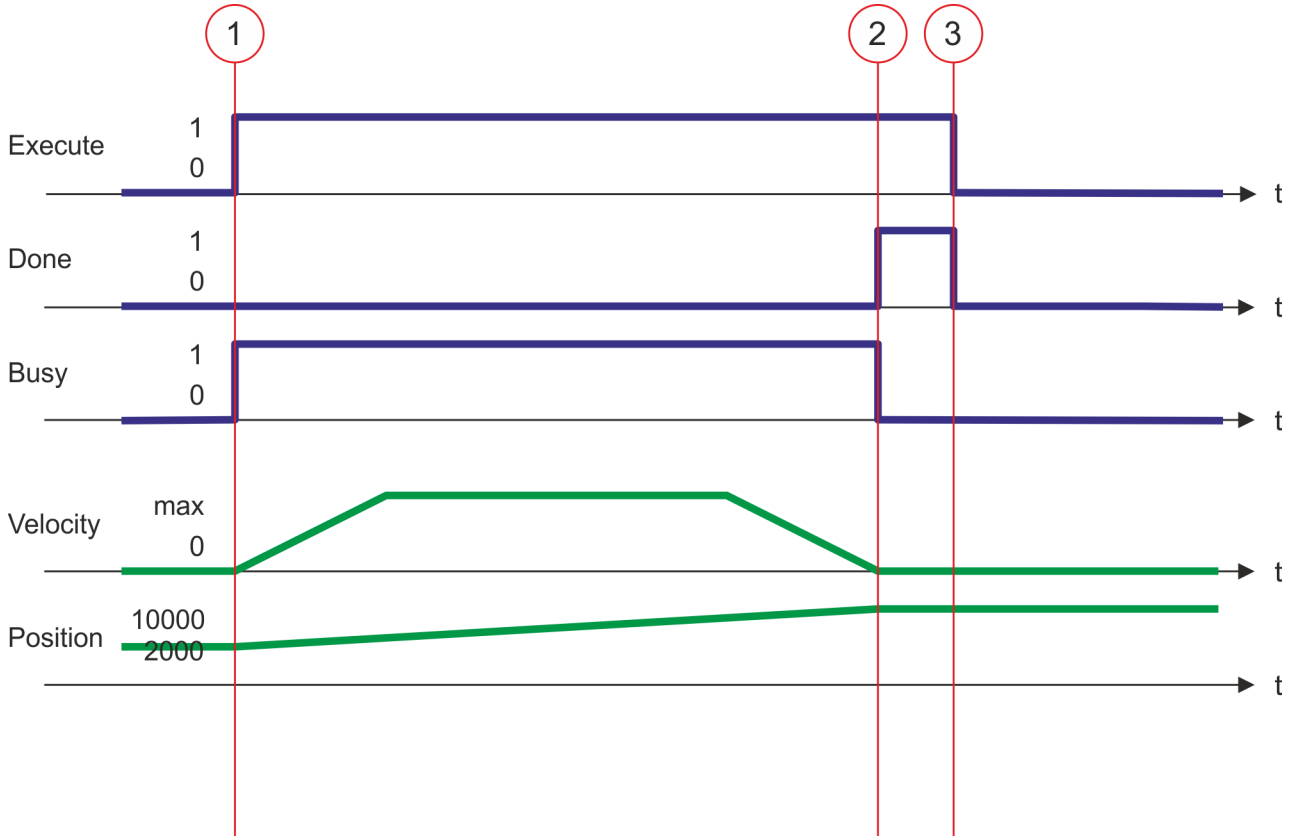
- Start des Auftrags in den PLCopen-States *Standstill*, *Discrete Motion* und *Continuous Motion* möglich.
- MC_MoveVelocity führt die Achse in den PLCopen-State *Discrete Motion* über.

Achse absolute verfahren

Mit einer Flanke 0-1 an *Execute* wird das Verfahren der Achse gestartet. Solange das Verfahren der Achse läuft, zeigt *Busy* den Wert TRUE. Nachdem die Zielposition erreicht wurde, wird *Busy* = FALSE und *Done* = TRUE geliefert. Die Geschwindigkeit der Achse ist dann gleich 0.

- *Mit Sigma-5 EtherCAT wird die Zielposition immer über den Weg angefahren, welcher am kürzesten ist.*
- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE, bis die Achse die Zielposition erreicht hat, ausgeführt.*
- *Ein laufender Auftrag kann durch einen anderen Bewegungsauftrag (z.B. MC_MoveVelocity) abgebrochen werden.*

Zustandsdiagramm der Bausteinparameter



(1) Die Achse wird mit MC_MoveAbsolute auf die absolute Position = 10000.0 verfahren (Startposition bei Auftragsstart gleich 2000.0). Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Verfahren der Achse gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) hat die Achse die Zielposition erreicht. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.10 FB 811 - MC_Reset - Reset Achse

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: ↯ Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_Reset wird ein Reset (Neuinitialisieren) der Achse durchgeführt. Dabei werden alle internen Fehler der Achse zurückgesetzt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Reset Achse – Flanke 0-1: Reset der Achse wird durchgeführt
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Reset wurde durchgeführt
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *↯ Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

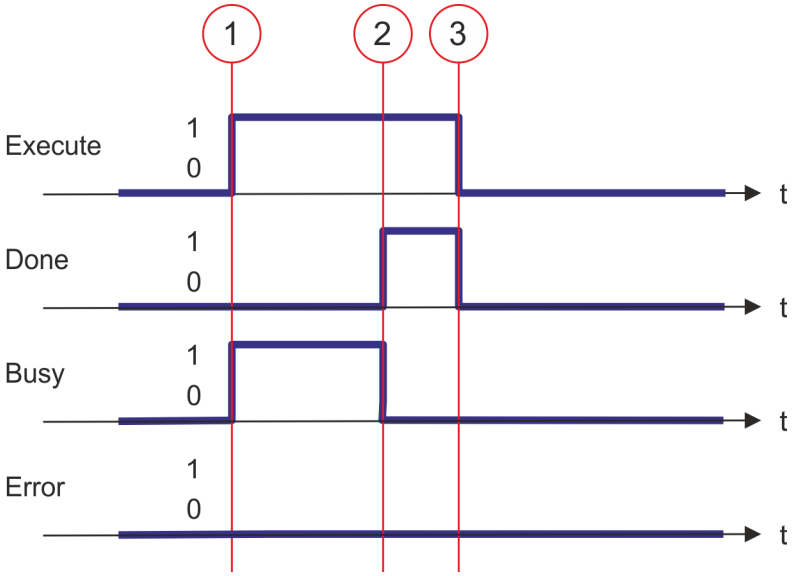
- Start des Auftrags im PLCopen-State *ErrorStop* möglich.
- MC_Reset führt die Achse in Abhängigkeit von MC_Power entweder in den PLCopen-State *Standstill* (Aufruf von MC_Power mit *Enable* = TRUE) oder *Disabled* (Aufruf von MC_Power mit *Enable* = FALSE) über.

Reset an Achse durchführen

Mit einer Flanke 0-1 an *Execute* wird der Reset der Achse gestartet. Solange der Reset der Achse läuft, zeigt *Busy* den Wert TRUE. Nachdem die Achse neu initialisiert wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE ausgeführt, bis der Auftrag abgeschlossen ist.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird der Reset der Achse gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist der Reset erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.11 FB 812 - MC_ReadStatus - PLCopen Status

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_ReadStatus kann der PLCopen-State der Achse ermittelt werden.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Enable	INPUT	BOOL	■ Statusanzeige – TRUE: Der Status wird an den Ausgängen permanent angezeigt – FALSE: Alle Ausgänge werden gleich FALSE bzw. 0 geliefert
Valid	OUTPUT	BOOL	■ Status gültig – TRUE: Der angezeigte Status ist gültig
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
ErrorStop	OUTPUT	BOOL	■ Achsfehler – TRUE: Achsfehler aufgetreten; ein Bewegungsauftrag kann nicht aktiviert werden
Disabled	OUTPUT	BOOL	■ Status Achse: Sperrung – TRUE: Achse ist gesperrt; ein Bewegungsauftrag kann nicht aktiviert werden
Stopping	OUTPUT	BOOL	■ Status Achse: Stop – TRUE: Achse wird gestoppt (MC_Stop ist aktiv)
Homing	OUTPUT	BOOL	■ Status Achse: Referenzierung – TRUE: Achse wird referenziert (MC_Homing ist aktiv)
Standstill	OUTPUT	BOOL	■ Status Bewegungsauftrag – TRUE: Kein Bewegungsauftrag aktiv; Bewegungsauftrag kann aktiviert werden
DiscreteMotion	OUTPUT	BOOL	■ Status Achsbewegung: Diskret – TRUE: Achse wird durch eine diskrete Bewegung verfahren (MC_MoveRelative, MC_MoveAbsolute oder MC_Halt ist aktiv)
ContinuousMotion	OUTPUT	BOOL	■ Status Achsbewegung: Kontinuierlich – TRUE: Achse wird durch eine kontinuierliche Bewegung verfahren (MC_MoveVelocity ist aktiv)
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Slave-Achse

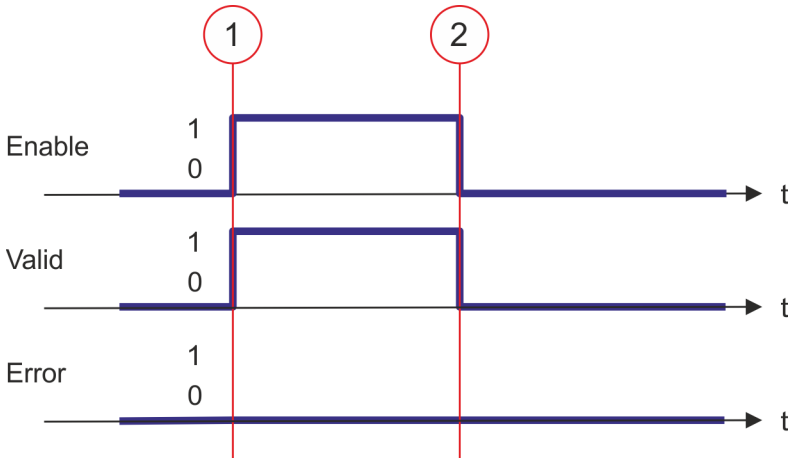
PLCopen-State

- Start des Auftrags in jedem PLCopen-State möglich.

Status der Achse ermitteln

Mit *Enable* = TRUE wird an den Ausgängen der Zustand der Achse entsprechend dem Zustandsdiagramm nach PLCopen geliefert.

Zustandsdiagramm der Bausteinparameter



(1) Zum Zeitpunkt (1) wird *Enable* = TRUE gesetzt. Damit liefert *Valid* den Wert TRUE und an den Ausgängen wird der Zustand entsprechend des PLCopen-Zustandsdiagramms angezeigt.
(2) Zum Zeitpunkt (2) wird *Enable* = FALSE gesetzt. Damit werden sämtliche Ausgänge gleich FALSE bzw. 0 gesetzt.

15.8.3.12 FB 813 - MC_ReadAxisError - Fehler von Achse lesen

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_ReadAxisError wird der aktuell anstehende Fehler direkt vom Antrieb gelesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Reset Achse – Flanke 0-1: Achsfehler wird gelesen.
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Achsfehler ausgelesen.
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung.
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
AxisErrorID	OUTPUT	WORD	Achsfehler-ID; der gelieferte Wert ist Hersteller-spezifisch kodiert.
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

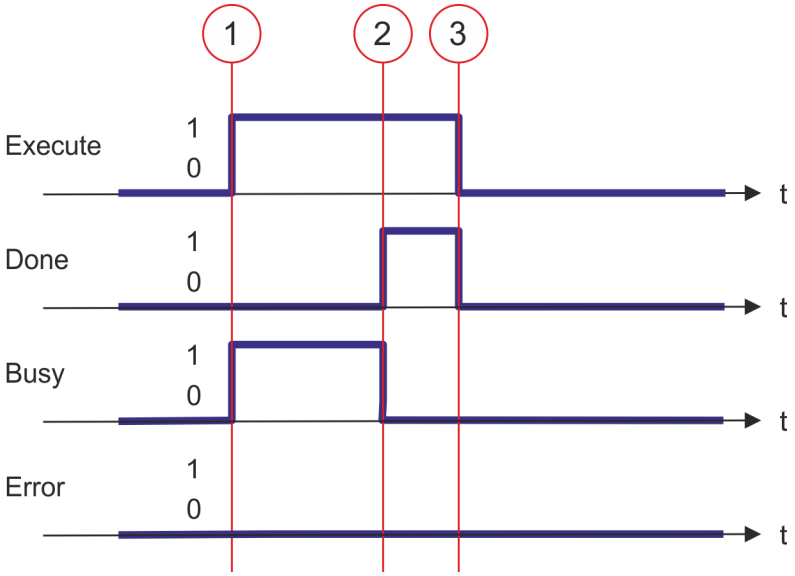
■ Start des Auftrags in jedem PLCopen-State möglich.

Fehler der Achse lesen

Mit einer Flanke 0-1 an *Execute* wird das Lesen des Achsfehlers gestartet. Solange das Lesen des Achsfehlers läuft, zeigt *Busy* den Wert TRUE. Nachdem der Achsfehler gelesen wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der Ausgang *AxisErrorID* zeigt den aktuell anstehenden Achsfehler an.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Lesen des Achsfehlers gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Lesen des Achsfehlers erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.13 FB 814 - MC_ReadParameter - Parameter der Achse lesen

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_ReadParameter wird der Parameter, der über die Parameter-Nummer festgelegt ist, von der Achse gelesen. *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Parameter Achse lesen – Flanke 0-1: Das Lesen des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher gelesen werden soll. *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde ausgelesen
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Value	OUTPUT	REAL	Wert des gelesenen Parameters
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

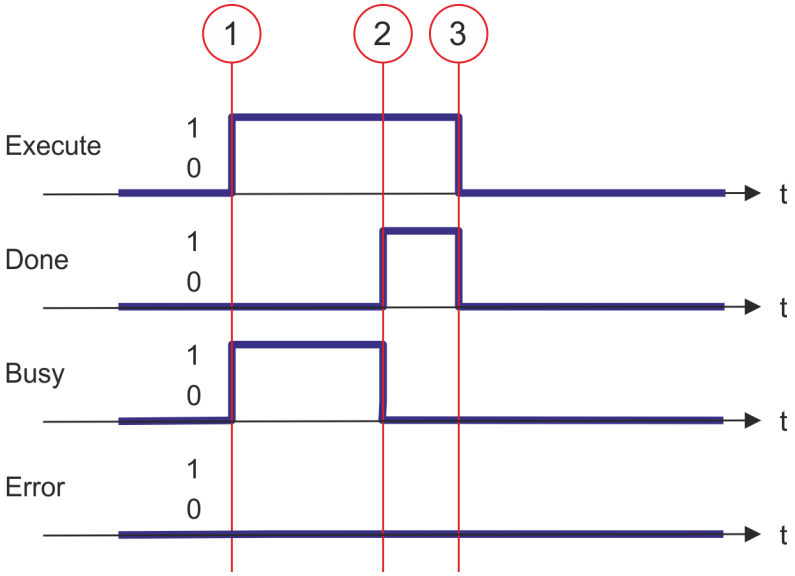
■ Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse lesen

Mit einer Flanke 0-1 an *Execute* wird das Lesen des Parameters gestartet. Solange das Lesen des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter gelesen wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der Ausgang *Value* zeigt den Wert des Parameters an.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Lesen des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Lesen des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.14 FB 815 - MC_WriteParameter - Parameter an Achse schreiben

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_WriteParameter wird der Wert des Parameters, der über die Parameter-Nummer festgelegt ist, zur Achse geschrieben. *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Parameter Achse schreiben – Flanke 0-1: Das Schreiben des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher geschrieben werden soll. *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*
Value	INPUT	REAL	Wert des geschriebenen Parameters
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde geschrieben
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

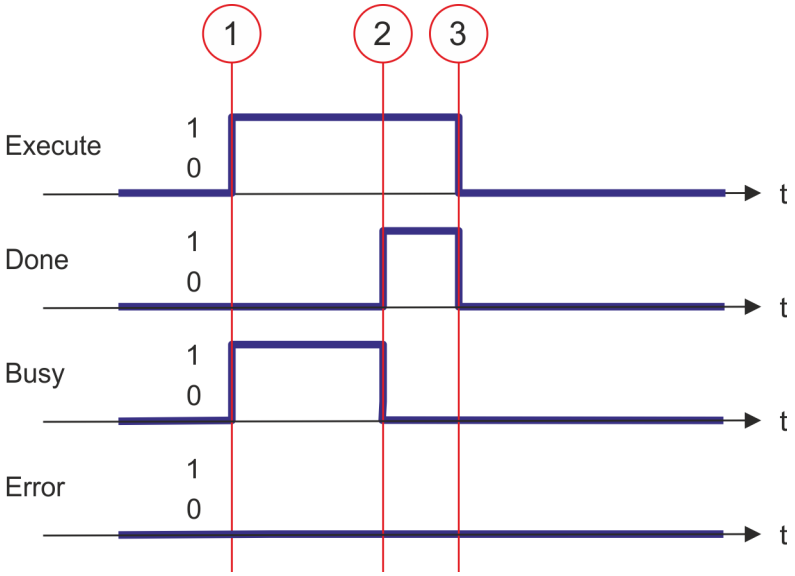
■ Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse schreiben

Mit einer Flanke 0-1 an *Execute* wird das Schreiben des Parameters gestartet. Solange das Schreiben des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter geschrieben wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Schreiben des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Schreiben des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.15 FB 816 - MC_ReadActualPosition - Aktuelle Position der Achse lesen

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_ReadActualPosition wird die aktuelle Position der Achse gelesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Enable	INPUT	BOOL	■ Position Achse lesen – TRUE: Die Position der Achse wird kontinuierlich gelesen – FALSE: Alle Ausgänge werden gleich FALSE bzw. 0 geliefert
Valid	OUTPUT	BOOL	■ Position gültig – TRUE: Die gelesene Position ist gültig
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Position	OUTPUT	REAL	Position der Achse in [Anwendereinheit]
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

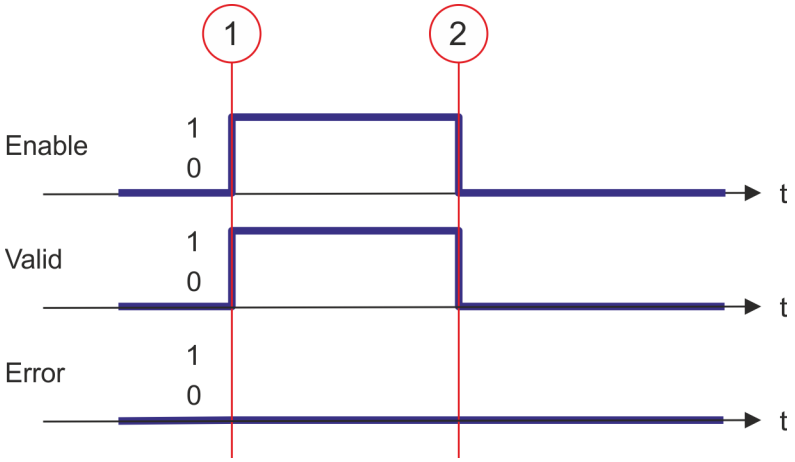
PLCopen-State

■ Start des Auftrags in jedem PLCopen-State möglich.

Position der Achse lesen

Mit *Enable* gleich TRUE wird die aktuelle Position der Achse ermittelt und unter *Position* abgelegt.

Zustandsdiagramm der Bausteinparameter



(1) Zum Zeitpunkt (1) wird *Enable* = TRUE gesetzt. Damit liefert *Valid* den Wert TRUE und am Ausgang *Position* wird die aktuelle Position der Achse angezeigt.
(2) Zum Zeitpunkt (2) wird *Enable* = FALSE gesetzt. Damit werden sämtliche Ausgänge gleich FALSE bzw. 0 gesetzt.

15.8.3.16 FB 817 - MC_ReadActualVelocity - Aktuelle Geschwindigkeit der Achse lesen

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_ReadActualVelocity wird die aktuelle Geschwindigkeit der Achse gelesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Enable	INPUT	BOOL	■ Geschwindigkeit Achse lesen – TRUE: Die Geschwindigkeit der Achse wird kontinuierlich gelesen – FALSE: Alle Ausgänge werden gleich FALSE bzw. 0 geliefert
Valid	OUTPUT	BOOL	■ Geschwindigkeit gültig – TRUE: Die gelesene Geschwindigkeit ist gültig
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Velocity	OUTPUT	REAL	Geschwindigkeit der Achse in [Anwendereinheit/s]
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

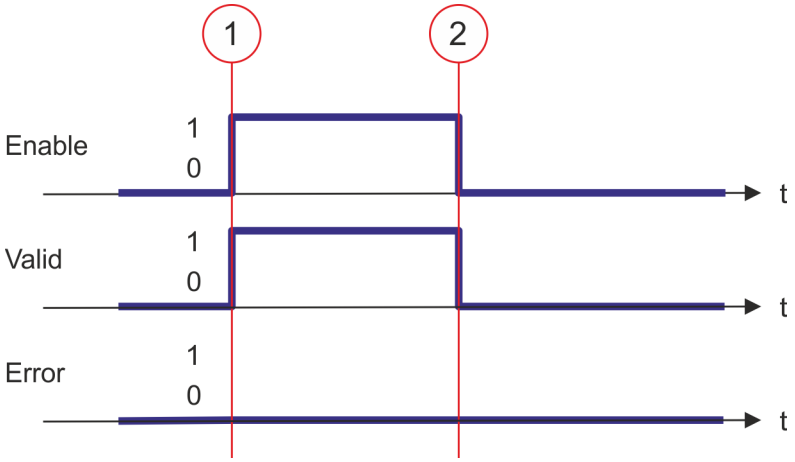
PLCopen-State

■ Start des Auftrags in jedem PLCopen-State möglich.

Geschwindigkeit der Achse lesen

Mit *Enable* gleich TRUE wird die aktuelle Geschwindigkeit der Achse ermittelt und unter *Velocity* abgelegt.

Zustandsdiagramm der Bausteinparameter



(1) Zum Zeitpunkt (1) wird *Enable* = TRUE gesetzt. Damit liefert *Valid* den Wert TRUE und am Ausgang *Velocity* wird die aktuelle Geschwindigkeit der Achse angezeigt.
(2) Zum Zeitpunkt (2) wird *Enable* = FALSE gesetzt. Damit werden sämtliche Ausgänge gleich FALSE bzw. 0 gesetzt.

15.8.3.17 FB 818 - MC_ReadAxisInfo - Zusatzinformationen der Achse lesen

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_ReadAxisInfo werden einige Zusatzinformationen der Achse angezeigt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Enable	INPUT	BOOL	■ Zusatzinformationen Achse lesen – TRUE: Die Zusatzinformationen der Achse werden kontinuierlich gelesen – FALSE: Alle Ausgänge werden gleich FALSE bzw. 0 geliefert
Valid	OUTPUT	BOOL	■ Zusatzinformationen gültig – TRUE: Die gelesene Zusatzinformationen sind gültig
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
HomeAbsSwitch	OUTPUT	BOOL	Referenzschalter ■ TRUE: Der Referenzschalter ist aktiviert
LimitSwitchPos	OUTPUT	BOOL	Endschalter positive Richtung ■ TRUE: Endschalter positive Richtung ist aktiviert
LimitSwitchNeg	OUTPUT	BOOL	Endschalter negative Richtung (NOT-Bit am Antrieb) ■ TRUE: Endschalter negative Richtung ist aktiviert
Simulation	OUTPUT	BOOL	Parameter aktuell nicht unterstützt; immer FALSE
Communication-Ready	OUTPUT	BOOL	■ Information Achse: Datenaustausch – TRUE: Datenaustausch mit der Achse initialisiert; Achse ist kommunikationsbereit
ReadyForPowerOn	OUTPUT	BOOL	■ Information Achse: Freigabe möglich – TRUE: Die Freigabe der Achse ist möglich
PowerOn	OUTPUT	BOOL	■ Information Achse: freigegeben – TRUE: Die Freigabe der Achse ist erfolgt
IsHomed	OUTPUT	BOOL	■ Information Achse: referenziert – TRUE: Die Achse ist referenziert
AxisWarning	OUTPUT	BOOL	■ Information Achse: Fehler – TRUE: Mindestens 1 Fehler wird von der Achse gemeldet
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

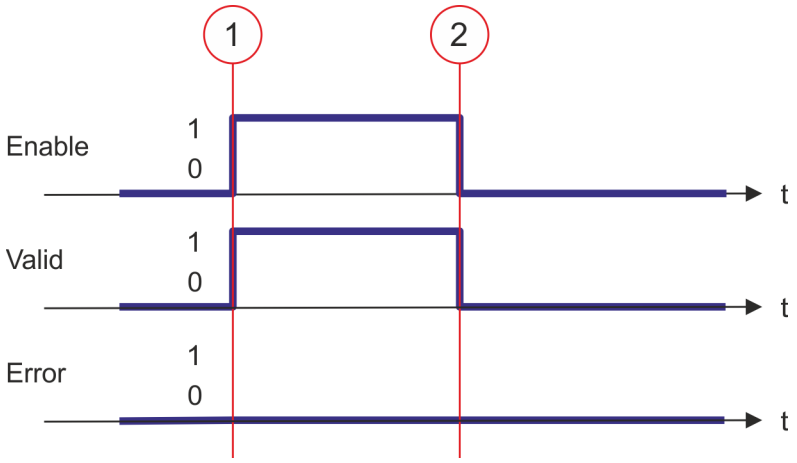
PLCopen-State

- Start des Auftrags in jedem PLCopen-State möglich.

Status der Achse ermitteln

Mit *Enable* gleich TRUE werden an den Ausgängen die Zusatzinformationen zur Achse geliefert.

Zustandsdiagramm der Bausteinparameter



(1) Zum Zeitpunkt (1) wird *Enable* = TRUE gesetzt. Damit liefert *Valid* den Wert TRUE und an den Ausgängen werden die Zusatzinformationen zur Achse angezeigt.
(2) Zum Zeitpunkt (2) wird *Enable* = FALSE gesetzt. Damit werden sämtliche Ausgänge gleich FALSE bzw. 0 gesetzt.

15.8.3.18 FB 819 - MC_ReadMotionState - Zustand Bewegungsauftrag lesen

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_ReadMotionState wird der aktuelle Zustand des Bewegungsauftrags angezeigt.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Enable	INPUT	BOOL	■ Zustand Bewegungsauftrag lesen – TRUE: Zustand Bewegungsauftrag wird kontinuierlich gelesen – FALSE: Alle Ausgänge werden gleich FALSE bzw. 0 geliefert
Source	INPUT	Byte	Nur Source = 0 wird unterstützt; an den Ausgängen werden die Istzustände des Bewegungsauftrags angezeigt.
Valid	OUTPUT	BOOL	■ Zustand gültig – TRUE: Die gelesene Zustand des Bewegungsauftrags ist gültig
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
ConstantVelocity	OUTPUT	BOOL	■ Zustand Bewegungsauftrag: Geschwindigkeit – TRUE: Geschwindigkeit ist konstant
Accelerating	OUTPUT	BOOL	Bitte beachten Sie, dass dieser Parameter bei Einsatz von Frequenzumrichter über EtherCAT nicht unterstützt wird! ■ Zustand Bewegungsauftrag: Beschleunigung – TRUE: Achse wird beschleunigt; die Geschwindigkeit der Achse erhöht sich.
Decelerating	OUTPUT	BOOL	Bitte beachten Sie, dass dieser Parameter bei Einsatz von Frequenzumrichter über EtherCAT nicht unterstützt wird! ■ Zustand Bewegungsauftrag: Bremsvorgang – TRUE: Achse wird gebremst; die Geschwindigkeit der Achse wird geringer.
DirectionPositive	OUTPUT	BOOL	■ Zustand Bewegungsauftrag: Position zunehmend – TRUE: Die Position der Achse nimmt zu
DirectionNegative	OUTPUT	BOOL	■ Zustand Bewegungsauftrag: Position abnehmend – TRUE: Die Position der Achse nimmt ab
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

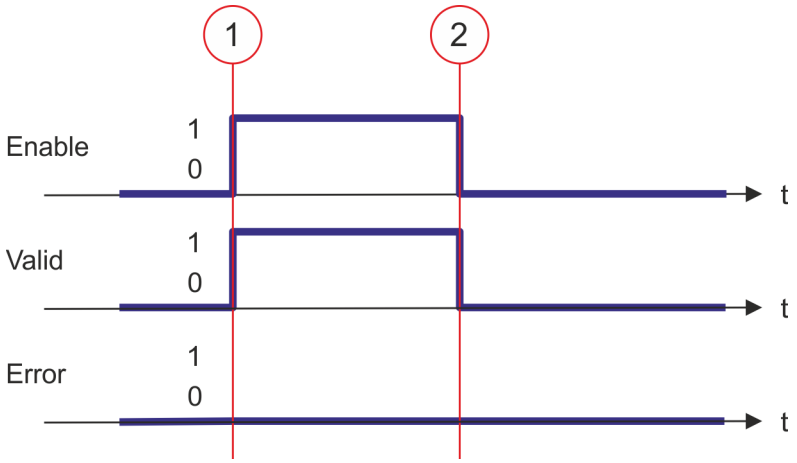
PLCopen-State

■ Start des Auftrags in jedem PLCopen-State möglich.

Zustand des Bewegungsauftrags lesen

Mit *Enable* gleich TRUE wird an den Ausgängen der Zustand des Bewegungsauftrags der Achse geliefert.

Zustandsdiagramm der Bausteinparameter



(1) Zum Zeitpunkt (1) wird *Enable* = TRUE gesetzt. Damit liefert *Valid* den Wert TRUE und an den Ausgängen wird der Zustand des Bewegungsauftrags angezeigt.
(2) Zum Zeitpunkt (2) wird *Enable* = FALSE gesetzt. Damit werden sämtliche Ausgänge gleich FALSE bzw. 0 gesetzt.

15.8.3.19 FB 823 - MC_TouchProbe - Achsposition erfassen

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Dieser Baustein erfasst einmalig die Achsposition in Abhängigkeit eines Trigger-Signals. Das Trigger-Signal kann über die am Eingang *TriggerInput* angegebene Variable konfiguriert werden. Als Trigger-Signal kann z.B. ein Digitaleingang oder die Gebernullspur dienen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	Mit einer Flanke 0-1 an *Execute* wird die Erfassung der Achsposition aktiviert.
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Die Achsposition wurde erfasst.
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung.
CommandAborted	OUTPUT	BOOL	■ Status – TRUE: Der Auftrag wurde während der Bearbeitung von einem anderen Auftrag abgebrochen.
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
RecordedPosition	OUTPUT	REAL	Erfasste Achsposition zum Zeitpunkt des Trigger-Signals [Anwendereinheiten].
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse.
TriggerInput	IN_OUT	MC_TRIGGER_REF	Referenz zum Trigger-Eingang. Struktur ■ .Probe – 01: TouchProbe-Register 1 – 02: TouchProbe-Register 2 ■ .TriggerSource – 00: Eingang – 00: Encoder Nullimpuls ■ .Triggermode – 00: SingleTrigger (fix) ■ .Reserved (0 fix)

- *Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE, solange ausgeführt, bis dieser abgearbeitet ist. Die erfasste Achsposition wird dann für einen Zyklus am Ausgang RecordedPosition ausgegeben. Kap. 15.10.3 "Verhalten der Ein- und Ausgänge" Seite 849*
- *Damit der Befehl ausgeführt werden kann, muss die Kommunikation mit der Achse OK und der PLCopen-State ungleich Homing sein.*
- *Ein laufender Auftrag kann durch einen neuen MC_TouchProbe auf der gleichen Achse abgebrochen werden.*
- *Ein laufender Auftrag kann durch den Befehl MC_AbortTrigger abgebrochen werden.*
- *Ein laufender Auftrag kann durch den Befehl MC_Home abgebrochen werden.*

Achsposition erfassen

Mit einer Flanke 0-1 an *Execute* wir die Erfassung der Achsposition aktiviert. Solange der Befehl abgearbeitet wird, zeigt *Busy* den Wert TRUE. Nach Abarbeitung des Befehls, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der erfasste Wert wird in *RecordedPosition* ausgegeben.

15.8.3.20 FB 824 - MC_AbortTrigger - Achsposition erfassen abbrechen

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Dieser Baustein bricht die durch MC_TouchProbe gestartete Erfassung der Achsposition ab.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	Mit einer Flanke 0-1 an *Execute* wird die Erfassung der Achsposition abgebrochen.
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Die Erfassung der Achsposition wurde abgebrochen.
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung.
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse.
TriggerInput	IN_OUT	MC_TRIGGER_REF	Referenz zum Trigger-Eingang. Struktur ■ .Probe – 01: TouchProbe-Register 1 – 02: TouchProbe-Register 2 ■ .TriggerSource – 00: Eingang – 00: Encoder Nullimpuls ■ .Triggermode – 00: SingleTrigger (fix) ■ .Reserved (0 fix)

Damit der Befehl ausgeführt werden kann, muss die Kommunikation mit der Achse OK sein.

Erfassung der Achsposition abbrechen

Mit einer Flanke 0-1 an *Execute* wir die Erfassung der Achsposition abgebrochen. Solange der Befehl abgearbeitet wird, zeigt *Busy* den Wert TRUE. Nach Abarbeitung des Befehls, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

15.8.3.21 FB 825 - MC_ReadBoolParameter - Boolean-Parameter von Achse lesen

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_ReadBoolParameter wird der Parameter, der über die Parameter-Nummer festgelegt ist, vom Datentyp BOOL von der Achse gelesen. *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Parameter Achse lesen – Flanke 0-1: Das Lesen des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher gelesen werden soll. *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde ausgelesen
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Value	OUTPUT	BOOL	Wert des gelesenen Parameters
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

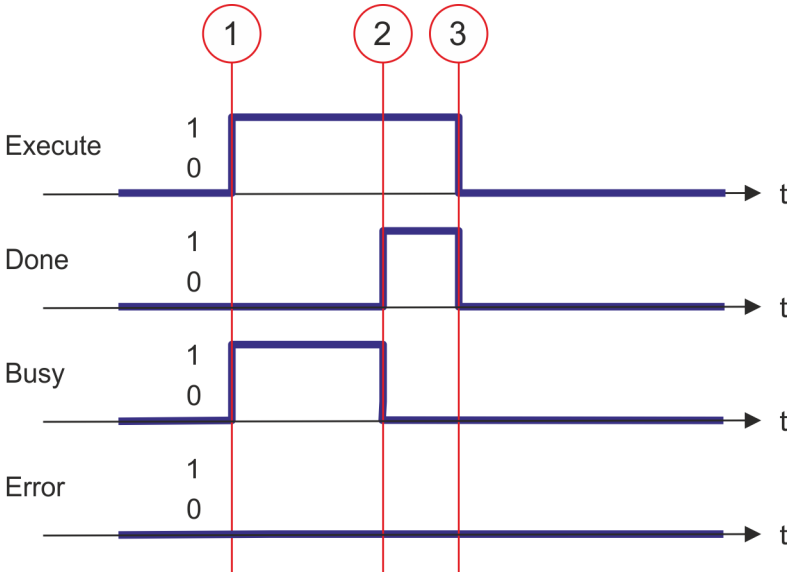
■ Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse lesen

Mit einer Flanke 0-1 an *Execute* wird das Lesen des Parameters gestartet. Solange das Lesen des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter gelesen wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der Ausgang *Value* zeigt den Wert des Parameters an.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Lesen des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Lesen des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.22 FB 826 - MC_WriteBoolParameter - Boolean-Parameter an Achse schreiben

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit MC_WriteBoolParameter wird der Wert des Parameters, der über die Parameter-Nummer festgelegt ist, vom Datentyp BOOL zur Achse geschrieben. *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Parameter Achse schreiben – Flanke 0-1: Das Schreiben des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher geschrieben werden soll. *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*
Value	INPUT	BOOL	Wert des geschriebenen Parameters
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde geschrieben
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

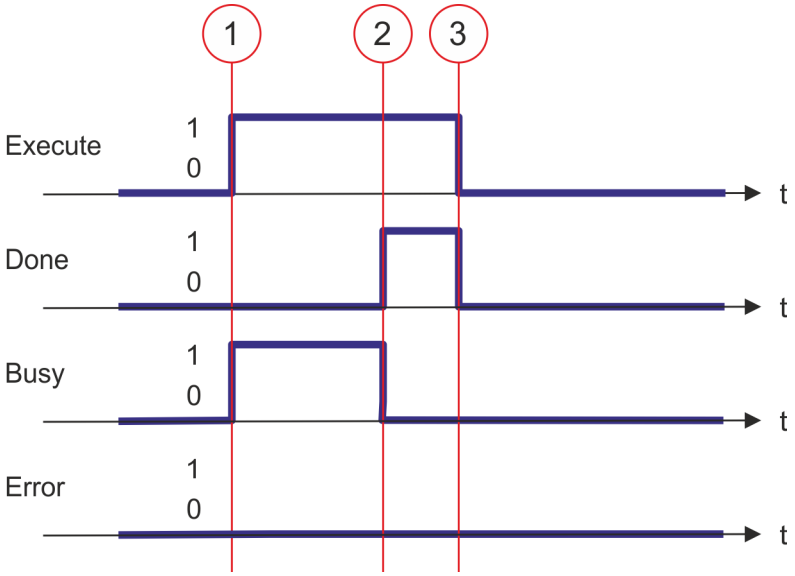
■ Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse schreiben

Mit einer Flanke 0-1 an *Execute* wird das Schreiben des Parameters gestartet. Solange das Schreiben des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter geschrieben wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Schreiben des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Schreiben des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.23 FB 827 - VMC_ReadDWordParameter - Doppelwort-Parameter von Achse lesen

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit VMC_ReadDWordParameter wird der Parameter, der über die Parameter-Nummer festgelegt ist, vom Datentyp DWORD von der Achse gelesen. *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Parameter Achse lesen – Flanke 0-1: Das Lesen des Parameters wird durchgeführt
ParameterNumber	INPUT	INT	Nummer des Parameters, welcher gelesen werden soll. *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde ausgelesen
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Value	OUTPUT	DWORD	Wert des gelesenen Parameters
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

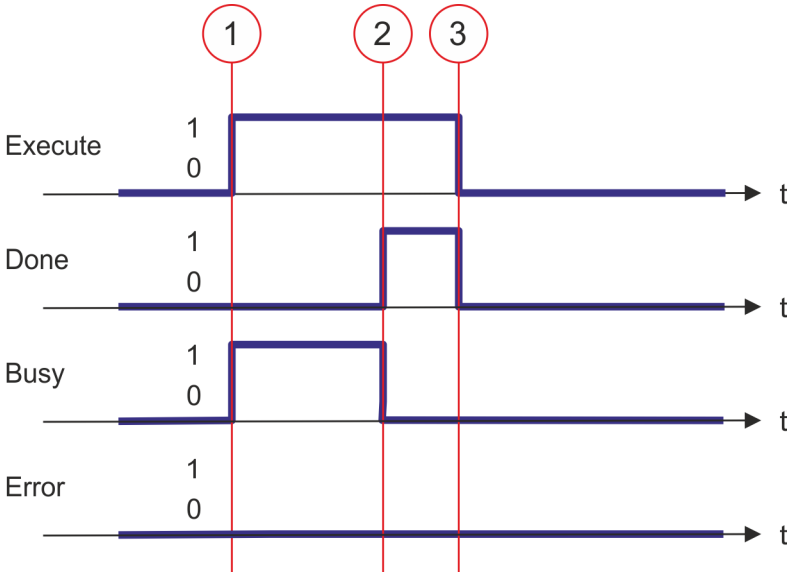
■ Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse lesen

Mit einer Flanke 0-1 an *Execute* wird das Lesen des Parameters gestartet. Solange das Lesen des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter gelesen wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der Ausgang *Value* zeigt den Wert des Parameters an.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Lesen des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Lesen des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.24 FB 828 - VMC_WriteDWordParameter - Doppelwort-Parameter an Achse schreiben

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: ↳ Kap. 15.8.1 "Übersicht" Seite 751

Mit VMC_WriteDWordParameter wird der Wert des Parameters, der über die Parameter-Nummer festgelegt ist, vom Datentyp DWORD zur Achse geschrieben. ↳ *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Parameter Achse schreiben – Flanke 0-1: Das Schreiben des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher geschrieben werden soll. ↳ *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*
Value	INPUT	DWORD	Wert des geschriebenen Parameters
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde geschrieben
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

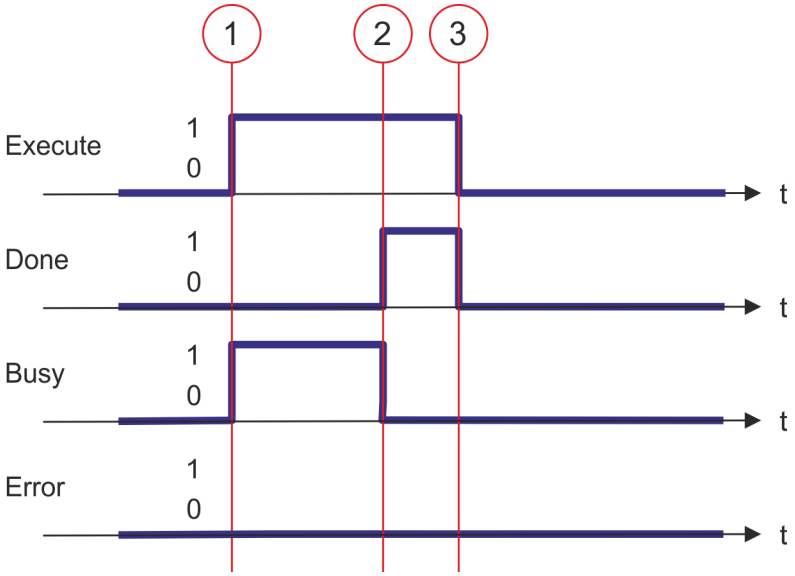
■ Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse schreiben

Mit einer Flanke 0-1 an *Execute* wird das Schreiben des Parameters gestartet. Solange das Schreiben des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter geschrieben wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Schreiben des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Schreiben des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.25 FB 829 - VMC_ReadWordParameter - Wort-Parameter von Achse lesen

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit VMC_ReadWordParameter wird der Parameter, der über die Parameter-Nummer festgelegt ist, vom Datentyp WORD von der Achse gelesen. *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Parameter Achse lesen – Flanke 0-1: Das Lesen des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher gelesen werden soll. *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde ausgelesen
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Value	OUTPUT	WORD	Wert des gelesenen Parameters
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

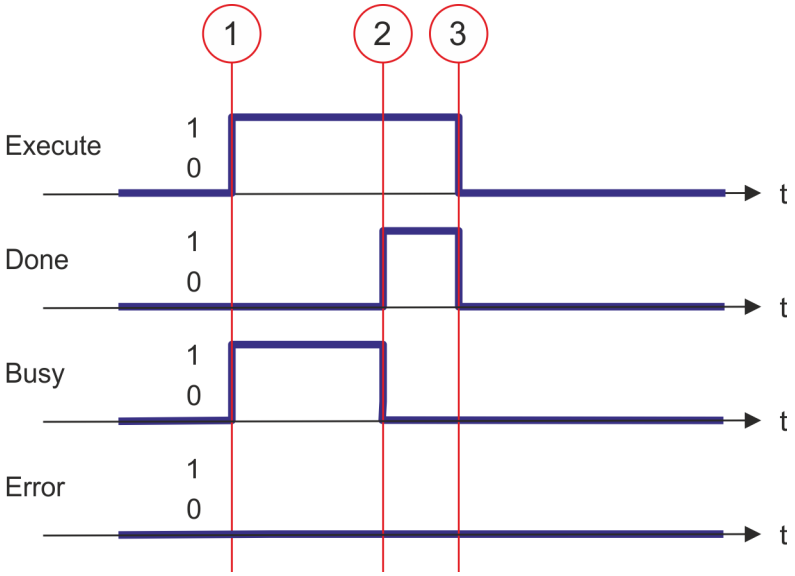
■ Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse lesen

Mit einer Flanke 0-1 an *Execute* wird das Lesen des Parameters gestartet. Solange das Lesen des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter gelesen wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der Ausgang *Value* zeigt den Wert des Parameters an.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Lesen des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Lesen des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.26 FB 830 - VMC_WriteWordParameter - Wort-Parameter an Achse schreiben

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: ↳ Kap. 15.8.1 "Übersicht" Seite 751

Mit VMC_WriteWordParameter wird der Wert des Parameters, der über die Parameter-Nummer festgelegt ist, vom Datentyp WORD zur Achse geschrieben. ↳ *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Parameter Achse schreiben – Flanke 0-1: Das Schreiben des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher geschrieben werden soll. ↳ *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*
Value	INPUT	WORD	Wert des geschriebenen Parameters
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde geschrieben
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

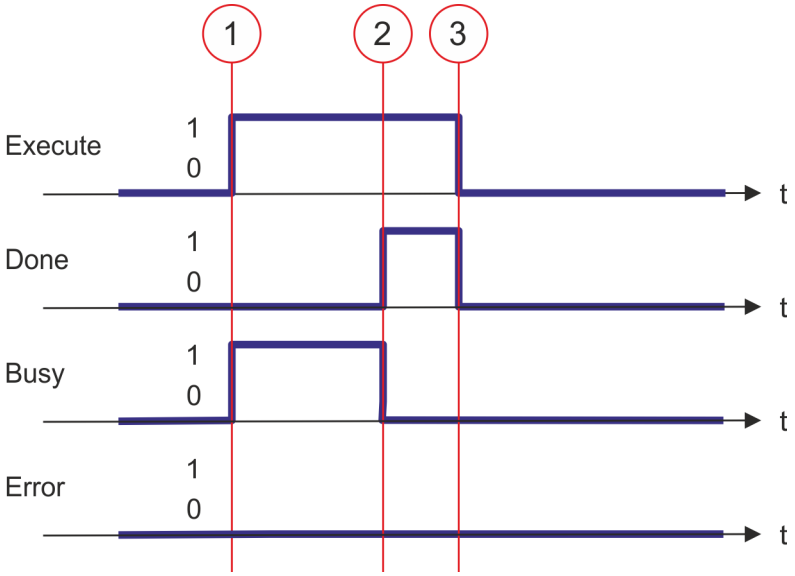
■ Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse schreiben

Mit einer Flanke 0-1 an *Execute* wird das Schreiben des Parameters gestartet. Solange das Schreiben des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter geschrieben wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Schreiben des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Schreiben des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.27 FB 831 - VMC_ReadByteParameter - Byte-Parameter von Achse lesen

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit VMC_ReadByteParameter wird der Parameter, der über die Parameter-Nummer festgelegt ist, vom Datentyp BYTE von der Achse gelesen. *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Parameter Achse lesen – Flanke 0-1: Das Lesen des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher gelesen werden soll. *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde ausgelesen
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Value	OUTPUT	BYTE	Wert des gelesenen Parameters
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

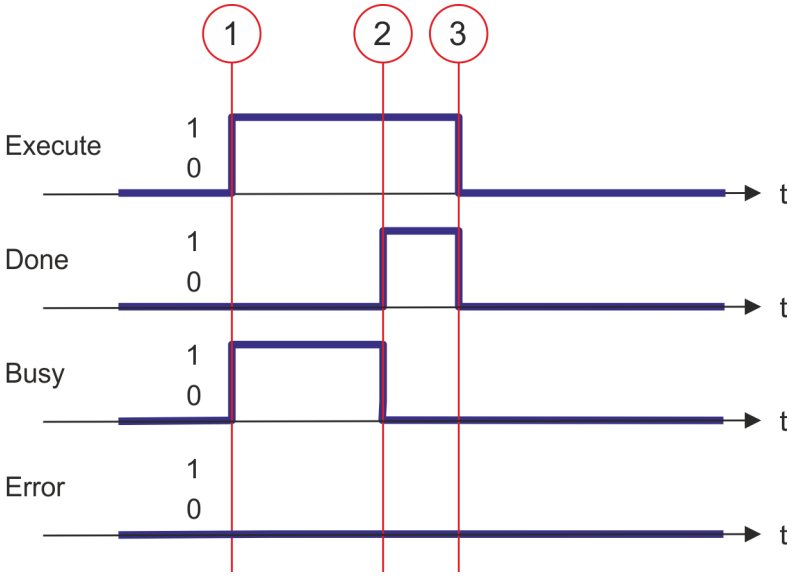
■ Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse lesen

Mit einer Flanke 0-1 an *Execute* wird das Lesen des Parameters gestartet. Solange das Lesen des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter gelesen wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der Ausgang *Value* zeigt den Wert des Parameters an.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Lesen des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Lesen des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.28 FB 832 - VMC_WriteByteParameter - Byte-Parameter an Achse schreiben

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: ↳ Kap. 15.8.1 "Übersicht" Seite 751

Mit VMC_WriteByteParameter wird der Wert des Parameters, der über die Parameter-Nummer festgelegt ist, vom Datentyp BYTE zur Achse geschrieben.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Parameter Achse schreiben – Flanke 0-1: Das Schreiben des Parameters wird durchgeführt
Parameter Number	INPUT	INT	Nummer des Parameters, welcher geschrieben werden soll. ↳ *Kap. 15.8.3.35 "PLCopen Parameter" Seite 823*
Value	INPUT	BYTE	Wert des geschriebenen Parameters
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde geschrieben
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen ↳ *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

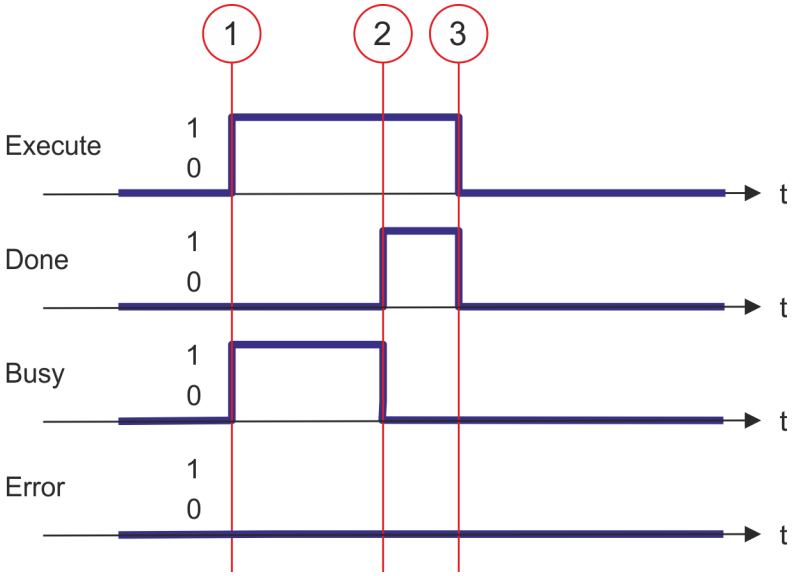
■ Start des Auftrags in jedem PLCopen-State möglich.

Parameter der Achse schreiben

Mit einer Flanke 0-1 an *Execute* wird das Schreiben des Parameters gestartet. Solange das Schreiben des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter geschrieben wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Schreiben des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Schreiben des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.29 FB 833 - VMC_ReadDriveParameter - Antriebsparameter lesen

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Mit dem VMC_ReadDriveParameter wird ein Parameterwert aus dem angebundenen Antrieb gelesen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Antriebsparameter lesen – Flanke 0-1: Das Lesen des Antriebsparameters wird durchgeführt.
Index	INPUT	WORD	Index des Antriebsparameters
Subindex	INPUT	BYTE	Subindex des Antriebsparameters
Length	INPUT	BYTE	Datenlänge ■ 1: BYTE ■ 2: WORD ■ 4: DWORD
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde ausgelesen
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Value	OUTPUT	DWORD	Wert des gelesenen Parameters
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

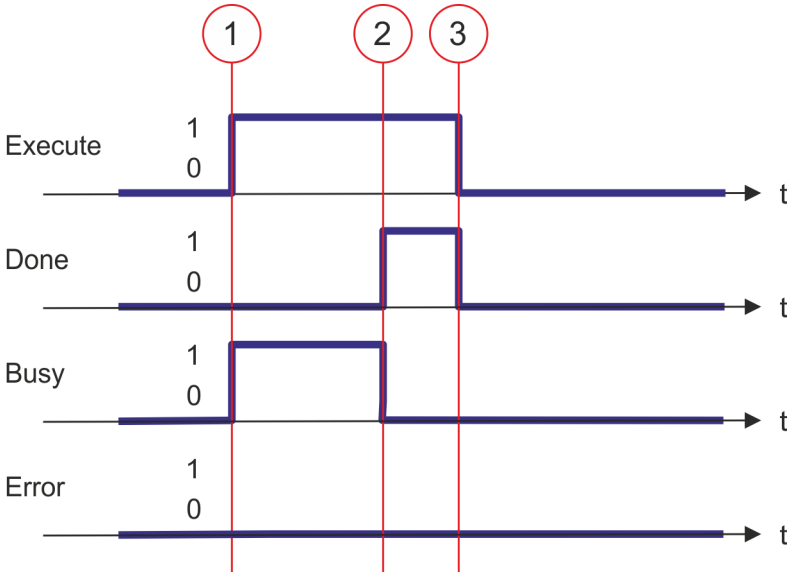
■ Start des Auftrags in jedem PLCopen-State möglich.

Antriebsparameter lesen

Mit einer Flanke 0-1 an *Execute* wird das Lesen des Antriebsparameters gestartet. Solange das Lesen des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter gelesen wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert. Der Ausgang *Value* zeigt den Wert des Parameters an.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Lesen des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Lesen des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.30 FB 834 - VMC_WriteDriveParameter - Antriebsparameter schreiben

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: ↳ Kap. 15.8.1 "Übersicht" Seite 751

Mit dem VMC_WriteDriveParameter wird ein Parameterwert in den angebundenen Antrieb geschrieben.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Antriebsparameter schreiben – Flanke 0-1: Das Schreiben des Antriebsarameters wird durchgeführt.
Index	INPUT	WORD	Index des Antriebsparameters
Subindex	INPUT	BYTE	Subindex des Antriebsparameters
Length	INPUT	BYTE	Datenlänge: 1=BYTE; 2=WORD; 4=DWORD
Value	INPUT	DWORD	Wert des geschriebenen Parameters
Done	OUTPUT	BOOL	■ Status – TRUE: Auftrag erfolgreich durchgeführt. Parameter wurde ausgelesen
Busy	OUTPUT	BOOL	■ Status – TRUE: Auftrag ist in Bearbeitung
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *↳ Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
Axis	IN_OUT	MC_AXIS_REF	Referenz zur Achse

PLCopen-State

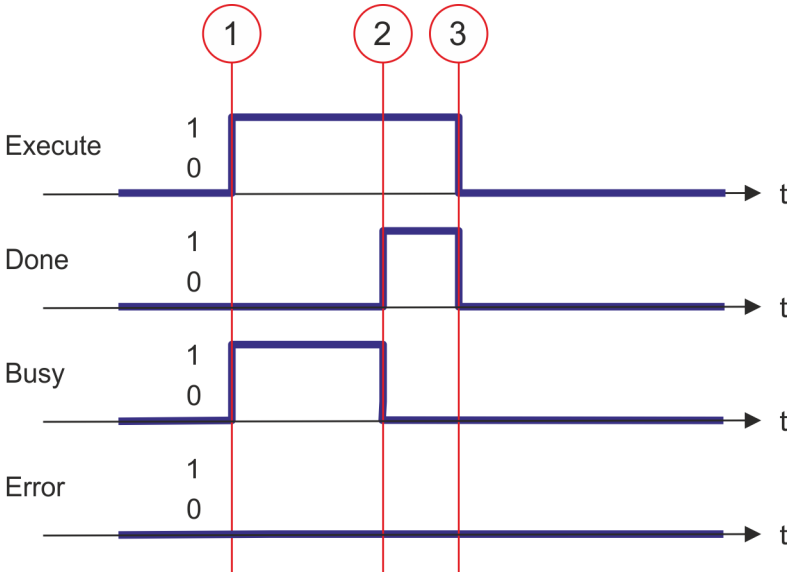
■ Start des Auftrags in jedem PLCopen-State möglich.

Antriebsparameter schreiben

Mit einer Flanke 0-1 an *Execute* wird das Schreiben des Parameters gestartet. Solange das Schreiben des Parameters läuft, zeigt *Busy* den Wert TRUE. Nachdem der Parameter geschrieben wurde, wird *Busy* gleich FALSE und *Done* gleich TRUE geliefert.

Ein laufender Auftrag wird auch beim Setzen von Execute gleich FALSE weiterhin ausgeführt.

Zustandsdiagramm der Bausteinparameter



(1) Mit der Flanke 0-1 an *Execute* zum Zeitpunkt (1) wird das Schreiben des Parameters gestartet und *Busy* liefert den Wert TRUE.
(2) Zum Zeitpunkt (2) ist das Schreiben des Parameters erfolgreich abgeschlossen. *Busy* liefert den Wert FALSE und *Done* den Wert TRUE.
(3) Zum Zeitpunkt (3) ist der Auftrag abgeschlossen und *Execute* wird gleich FALSE gesetzt und dadurch sämtliche Ausgangsparameter auf FALSE bzw. 0 gesetzt.

15.8.3.31 FB 835 - VMC_HomeInit_LimitSwitch - Initialisierung Referenzfahrt auf Endschalter

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Dieser Baustein initialisiert die Referenzfahrt (Homing) auf den Endschalter.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Initialisierung der Referenzfahrt Methode – Flanke 0-1: Werte der Eingangsparameter werden übernommen und die Initialisierung der Referenzfahrt Methode gestartet.
Direction	INPUT	BOOL	■ Richtung der Referenzfahrt – TRUE: auf positiven Endschalter – FALSE: auf negativen Endschalter
Velocity-SearchSwitch	INPUT	REAL	Geschwindigkeit für die Suche nach dem Schalter in [Anwendereinheiten/s]
VelocitySearch-Zero	INPUT	REAL	Geschwindigkeit für die Suche nach dem Index in [Anwendereinheiten/s]
Acceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2]
Done	OUTPUT	BOOL	■ Status – TRUE: Initialisierung wurde ohne Fehler beendet.
Busy	OUTPUT	BOOL	■ Status – TRUE: Initialisierung ist aktiv
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
AXIS	IN_OUT	MC_AXIS_REF	Referenz zur Achse.

Initialisierung Referenzfahrt auf Endschalter

Mit einer Flanke 0-1 an *Execute* werden die Werte der Eingangsparameter übernommen und die Initialisierung der Referenzfahrt Methode gestartet. So lange die Initialisierung aktiv ist, wird der Ausgang *Busy* auf True gesetzt. Ist die Initialisierung ohne Fehler beendet worden, wird der Ausgang *Done* auf True gesetzt. Tritt bei der Initialisierung ein Fehler auf, wird der Ausgang *Error* auf True gesetzt und am Ausgang *ErrorID* eine Fehlernummer ausgegeben.

Initialisierung der Referenzfahrt Methode

1. Überprüfen der Kommunikation zur Achse.
2. Prüfen auf erlaubte PLCopen Zustände.
3. Prüfung der Eingangswerte:
   - Eingang VelocitySearchSwitch [UserUnits] > 0.0
   - VelocitySearchSwitch [InternalUnits] > 0
   - VelocitySearchSwitch [InternalUnits] ≤ VelocityMax
   - Eingang VelocitySearchZero [UserUnits] > 0.0
   - VelocitySearchZero [InternalUnits] > 0
   - VelocitySearchZero [InternalUnits] ≤ VelocityMax
   - Eingang Acceleration [UserUnits] > 0.0
   - Acceleration [InternalUnits] > 0
   - Acceleration [InternalUnits] ≤ AccelerationMax
4. Übertragung der Antriebsparameter:
   - "Homing Method" Referenzfahrtmethode in Abhängigkeit vom Eingang "Direction"
     Siehe Tabelle unten!
   - "Homing Speed during search for switch" [Inc/s]
     Geschwindigkeit für die Schaltersuche
   - "Homing Speed during search for zero" [Inc/s]
     Geschwindigkeit für die Indexsuche
   - "Homing Acceleration" [Inc/s^2]
     Anfahr- und Bremsbeschleunigung für die Referenzfahrt

Homing Method	Direction
1	false
2	true

15.8.3.32 FB 836 - VMC_HomeInit_HomeSwitch - Initialisierung Referenzfahrt auf Referenzschalter

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Dieser Baustein initialisiert die Referenzfahrt (Homing) auf den Referenzschalter.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Initialisierung der Referenzfahrt Methode – Flanke 0-1: Werte der Eingangsparameter werden übernommen und die Initialisierung der Referenzfahrt Methode gestartet.
InitialDirection	INPUT	BOOL	■ Startrichtung der Referenzfahrt – TRUE: auf positiven Endschalter – FALSE: auf negativen Endschalter
WithIndexPulse	INPUT	BOOL	■ Referenzfahrt – TRUE: mit Nullimpuls – FALSE: ohne Nullimpuls
OnRisingEdge	INPUT	BOOL	■ Flanke Referenzschalter – TRUE: Flanke 0-1 – FALSE: Flanke 1-0
SameDirIndex-Pulse	INPUT	BOOL	■ Nullimpulssuche – TRUE: Nach dem Erkennen des Referenzschalters ohne Richtungswechsel den Nullimpuls suchen. – FALSE: Nach dem Erkennen des Referenzschalters einen Richtungswechsel zur Nullimpulssuche durchführen.
Velocity-SearchSwitch	INPUT	REAL	Geschwindigkeit für die Schaltersuche in [Anwendereinheiten/s]
VelocitySearch-Zero	INPUT	REAL	Geschwindigkeit für die Indexsuche in [Anwendereinheiten/s]
Acceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2]
Done	OUTPUT	BOOL	■ Status – TRUE: Initialisierung wurde ohne Fehler beendet.
Busy	OUTPUT	BOOL	■ Status – TRUE: Initialisierung ist aktiv
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
AXIS	IN_OUT	MC_AXIS_REF	Referenz zur Achse.

Initialisierung Referenzfahrt auf Referenzschalter

Mit einer Flanke 0-1 an *Execute* werden die Werte der Eingangsparameter übernommen und die Initialisierung der Referenzfahrt Methode gestartet. So lange die Initialisierung aktiv ist, wird der Ausgang *Busy* auf True gesetzt. Ist die Initialisierung ohne Fehler beendet worden, wird der Ausgang *Done* auf True gesetzt. Tritt bei der Initialisierung ein Fehler auf, wird der Ausgang *Error* auf True gesetzt und am Ausgang *ErrorID* eine Fehlernummer ausgegeben.

Initialisierung der Referenzfahrt Methode

1. Überprüfen der Kommunikation zur Achse.
2. Prüfen auf erlaubte PLCopen Zustände.
3. Prüfung der Eingangswerte:
   - Eingang VelocitySearchSwitch [UserUnits] > 0.0
   - VelocitySearchSwitch [InternalUnits] > 0
   - VelocitySearchSwitch [InternalUnits] ≤ VelocityMax
   - Eingang VelocitySearchZero [UserUnits] > 0.0
   - VelocitySearchZero [InternalUnits] > 0
   - VelocitySearchZero [InternalUnits] ≤ VelocityMax
   - Eingang Acceleration [UserUnits] > 0.0
   - Acceleration [InternalUnits] > 0
   - Acceleration [InternalUnits] ≤ AccelerationMax
4. Übertragung der Antriebsparameter:
   - "Homing Method" Referenzfahrtmethode in Abhängigkeit vom Eingang "Direction"
     Siehe Tabelle unten!
   - "Homing Speed during search for switch" [Inc/s]
     Geschwindigkeit für die Schaltersuche
   - "Homing Speed during search for zero" [Inc/s]
     Geschwindigkeit für die Indexsuche
   - "Homing Acceleration" [Inc/s^2]
     Anfahr- und Bremsbeschleunigung für die Referenzfahrt

Homing Method	InitialDirection	WithIndexPulse	OnRisingEdge	SameDirIndexPulse
7	positive	true	true	false
8	positive	true	true	true
9	positive	true	false	false
10	positive	true	false	true
11	negative	true	true	false
12	negative	true	true	true
13	negative	true	false	false
14	negative	true	false	true
24	positive	false	true	false
24	positive	false	true	true
24	positive	false	false	false
24	positive	false	false	true
28	negative	false	true	false

Homing Method	InitialDirection	WithIndexPulse	OnRisingEdge	SameDirIndexPulse
28	negative	false	true	true
28	negative	false	false	false
28	negative	false	false	true

15.8.3.33 FB 837 - VMC_HomeInit_ZeroPulse - Initialisierung Referenzfahrt auf Null Impuls

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: Kap. 15.8.1 "Übersicht" Seite 751

Dieser Baustein initialisiert die Referenzfahrt (Homing) auf Null Impuls.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Initialisierung der Referenzfahrt Methode – Flanke 0-1: Werte der Eingangsparameter werden übernommen und die Initialisierung der Referenzfahrt Methode gestartet.
Direction	INPUT	BOOL	■ Richtung der Referenzfahrt – TRUE: Positive Richtung – FALSE: Negative Richtung
VelocitySearch-Zero	INPUT	REAL	Geschwindigkeit für die Indexsuche in [Anwendereinheiten/s]
Acceleration	INPUT	REAL	Beschleunigung in [Anwendereinheiten/s^2]
Done	OUTPUT	BOOL	■ Status – TRUE: Initialisierung wurde ohne Fehler beendet.
Busy	OUTPUT	BOOL	■ Status – TRUE: Initialisierung ist aktiv
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
AXIS	IN_OUT	MC_AXIS_REF	Referenz zur Achse.

Initialisierung Referenzfahrt auf Nullimpuls

Mit einer Flanke 0-1 an *Execute* werden die Werte der Eingangsparameter übernommen und die Initialisierung der Referenzfahrt Methode gestartet. So lange die Initialisierung aktiv ist, wird der Ausgang *Busy* auf True gesetzt. Ist die Initialisierung ohne Fehler beendet worden, wird der Ausgang *Done* auf True gesetzt. Tritt bei der Initialisierung ein Fehler auf, wird der Ausgang *Error* auf True gesetzt und am Ausgang *ErrorID* eine Fehlernummer ausgegeben.

Initialisierung der Referenzfahrt Methode

1. Überprüfen der Kommunikation zur Achse.
2. Prüfen auf erlaubte PLCopen Zustände.

3. Prüfung der Eingangswerte:

- Eingang VelocitySearchZero [UserUnits] > 0.0
- VelocitySearchZero [InternalUnits] > 0
- VelocitySearchZero [InternalUnits] ≤ VelocityMax
- Eingang Acceleration [UserUnits] > 0.0
- Acceleration [InternalUnits] > 0
- Acceleration [InternalUnits] ≤ AccelerationMax

4. Übertragung der Antriebsparameter:

- "Homing Method" Referenzfahrtmethode in Abhängigkeit vom Eingang "Direction"
  Siehe Tabelle unten!
- "Homing Speed during search for switch" [Inc/s]
  Geschwindigkeit für die Schaltersuche
- "Homing Speed during search for zero" [Inc/s]
  Geschwindigkeit für die Indexsuche
- "Homing Acceleration" [Inc/s^2]
  Anfahr- und Bremsbeschleunigung für die Referenzfahrt

Homing Method	Direction
33	false
34	true

15.8.3.34 FB 838 - VMC_HomeInit_SetPosition - Initialisierung Referenzfahrt auf aktuelle Position

Beschreibung

Eine Übersicht der Antriebe, welche mit diesem Baustein angesteuert werden können, finden Sie hier: ↯ Kap. 15.8.1 "Übersicht" Seite 751

Dieser Baustein initialisiert die Referenzfahrt (Homing) auf die aktuelle Position.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
Execute	INPUT	BOOL	■ Initialisierung der Referenzfahrt Methode – Flanke 0-1: Werte der Eingangsparameter werden übernommen und die Initialisierung der Referenzfahrt Methode gestartet.
Done	OUTPUT	BOOL	■ Status – TRUE: Initialisierung wurde ohne Fehler beendet.
Busy	OUTPUT	BOOL	■ Status – TRUE: Initialisierung ist aktiv
Error	OUTPUT	BOOL	■ Status – TRUE: Ein Fehler ist aufgetreten. Zusätzliche Fehlerinformationen können dem Parameter *ErrorID* entnommen werden.
ErrorID	OUTPUT	WORD	Zusätzliche Fehlerinformationen *↯ Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
AXIS	IN_OUT	MC_AXIS_REF	Referenz zur Achse.

Initialisierung Referenzfahrt auf Endschalter

Mit einer Flanke 0-1 an *Execute* werden die Werte der Eingangsparameter übernommen und die Initialisierung der Referenzfahrt Methode gestartet. So lange die Initialisierung aktiv ist, wird der Ausgang *Busy* auf TRUE gesetzt. Ist die Initialisierung ohne Fehler beendet worden, wird der Ausgang *Done* auf TRUE gesetzt. Tritt bei der Initialisierung ein Fehler auf, wird der Ausgang *Error* auf TRUE gesetzt und am Ausgang *ErrorID* eine Fehlernummer ausgegeben.

Initialisierung der Referenzfahrt Methode

1. Überprüfen der Kommunikation zur Achse.
2. Prüfen auf erlaubte PLCopen Zustände.
3. Übertragung der Antriebsparameter:
   - "Homing Method" = 35

15.8.3.35 PLCopen Parameter

PN	Name	Datentyp	R/W	Beschreibung
1	CommandedPosition	REAL	R	Sollposition Zugriff auf: `#Axis.Status.Positioning.SetValues.CommandedPosition`
2	SWLimitPos	REAL	R/W	Positive Software Endschalter Position Zugriff auf: `"Axis".AxisConfiguration.PositionLimits.MaxPosition`
3	SWLimitNeg	REAL	R/W	Negative Software Endschalter Position Zugriff auf: `"Axis".AxisConfiguration.PositionLimits.MinPosition`
4	EnableLimitPos	BOOL	R/W	Positiven Endschalter aktivieren Zugriff auf: `"Axis".AxisConfiguration.PositionLimits.EnableMaxPos`
5	EnableLimitNeg	BOOL	R/W	Negativen Endschalter aktivieren Zugriff auf: `"Axis".AxisConfiguration.PositionLimits.EnableMinPos`
6	EnablePosLagMonitoring	BOOL	R/W	Überwachung Schleppabstand aktivieren Funktion wird nicht unterstützt
7	MaxPositionLag	REAL	R/W	Maximaler Schleppabstand Funktion wird nicht unterstützt
8	MaxVelocitySystem	REAL	R	Maximal zulässige Geschwindigkeit der Achse im Bewegungssystem Dieser Parameter wird aktuell nicht unterstützt
9	MaxVelocityAppl	REAL	R/W	Maximal zulässige Geschwindigkeit der Achse in der Anwendung Zugriff auf: `#Axis.AxisConfiguration.DynamicLimits.MaxVelocityApp`
10	ActualVelocity	REAL	R	Aktuelle Geschwindigkeit Zugriff auf: `#Axis.Status.Positioning.ActValues.Velocity`
11	CommandedVelocity	REAL	R	Sollgeschwindigkeit Zugriff auf: `#Axis.Status.Positioning.SetValues.Velocity`
12	MaxAccelerationSystem	REAL	R	Maximal zulässige Beschleunigung der Achse im Bewegungssystem Dieser Parameter wird aktuell nicht unterstützt

PN	Name	Datentyp	R/W	Beschreibung
13	MaxAccelerationAppl	REAL	R/W	Maximal zulässige Beschleunigung der Achse in der Anwendung Zugriff auf: `#Axis.AxisConfiguration.DynamicLimits.MaxAccelerationApp`
14	MaxDecelerationSystem	REAL	R	Maximal zulässige Verzögerung der Achse im Bewegungssystem Dieser Parameter wird aktuell nicht unterstützt
15	MaxDecelerationAppl	REAL	R/W	Maximal zulässige Verzögerung der Achse in der Anwendung Zugriff auf: `#Axis.AxisConfiguration.DynamicLimits.MaxDecelerationApp`
16	MaxJerkSystem	REAL	R	Maximal zulässiger Ruck der Achse im Bewegungssystem Dieser Parameter wird aktuell nicht unterstützt
17	MaxJerkAppl	REAL	R/W	Maximal zulässiger Ruck der Achse in der Anwendung Dieser Parameter wird aktuell nicht unterstützt

15.8.3.36 VIPA-spezifische Parameter

Positionierachse: Yaskawa *Sigma-5* / *Sigma-7* über EtherCAT

Nr.	Name	Datentyp	Index	Subindex	Zugriff
900	HomingDone	BOOL	-	-	R/W [1, 2]
901	PositiveTorqueLimit	BOOL	-	-	R/W [1, 2]
902	NegativeTorqueLimit	BOOL	-	-	R/W [1, 2]
1000	ErrorCode	WORD	603F	0	R [3]
1001	HomeOffset	DWORD	607C	0	R/W [5, 6]
1002	HomingMethod	WORD	6098	0	R/W [3, 4]
1003	SpeedSearchSwitch	DWORD	6099	1	R/W [5, 6]
1004	SpeedSearchZero	DWORD	6099	2	R/W [5, 6]
1005	HomingAcceleration	DWORD	609A	0	R/W [5, 6]
1006	PositiveTorqueLimit	WORD	60E0	0	R/W [3, 4]
1007	NegativeTorqueLimit	WORD	0x60E1	0	R/W [3, 4]
1008	MotorRatedTorque	DWORD	0x6076	0	R/W [5, 6]

1) Zugriff über ↯ *Kap. 15.8.3.21 "FB 825 - MC_ReadBoolParameter - Boolean-Parameter von Achse lesen" Seite 795*

2) Zugriff über ↯ *Kap. 15.8.3.22 "FB 826 - MC_WriteBoolParameter - Boolean-Parameter an Achse schreiben" Seite 797*

3) Zugriff über ↯ *Kap. 15.8.3.25 "FB 829 - VMC_ReadWordParameter - Wort-Parameter von Achse lesen" Seite 803*

4) Zugriff über ↯ *Kap. 15.8.3.26 "FB 830 - VMC_WriteWordParameter - Wort-Parameter an Achse schreiben" Seite 805*

5) Zugriff über ↯ *Kap. 15.8.3.23 "FB 827 - VMC_ReadDWordParameter - Doppelwort-Parameter von Achse lesen" Seite 799*

6) Zugriff über ↯ *Kap. 15.8.3.24 "FB 828 - VMC_WriteDWordParameter - Doppelwort-Parameter an Achse schreiben" Seite 801*

Nr.	Name	Datentyp	Index	Subindex	Zugriff
1009	FollowingErrorWindow	DWORD	0x6065	0	R/W [5, 6]
1010	FollowingErrorTimeOut	WORD	0x6066	0	R/W [3, 4]
1011	PositionWindow	DWORD	0x6067	0	R/W [5, 6]
1012	PositionTime	WORD	0x6068	0	R/W [3, 4]
1013	Min Position Limit	DWORD	0x607D	1	R/W [5, 6]
1014	Max Position Limit	DWORD	0x607D	2	R/W [5, 6]
1015	Digital outputs/ physical outputs	DWORD	0x60FE	1	R/W [5, 6]
1016	Digital outputs/ mask	DWORD	0x60FE	2	R/W [5, 6]
1017	Quick stop deceleration	DWORD	0x6085	0	R/W [5, 6]
1018	Forward external torque limit	WORD	0x2404	0	R/W [3, 4]
1019	Reverse external torque limit	WORD	0x2405	0	R/W [3, 4]

1) Zugriff über *Kap. 15.8.3.21 "FB 825 - MC_ReadBoolParameter - Boolean-Parameter von Achse lesen" Seite 795*

2) Zugriff über *Kap. 15.8.3.22 "FB 826 - MC_WriteBoolParameter - Boolean-Parameter an Achse schreiben" Seite 797*

3) Zugriff über *Kap. 15.8.3.25 "FB 829 - VMC_ReadWordParameter - Wort-Parameter von Achse lesen" Seite 803*

4) Zugriff über *Kap. 15.8.3.26 "FB 830 - VMC_WriteWordParameter - Wort-Parameter an Achse schreiben" Seite 805*

5) Zugriff über *Kap. 15.8.3.23 "FB 827 - VMC_ReadDWordParameter - Doppelwort-Parameter von Achse lesen" Seite 799*

6) Zugriff über *Kap. 15.8.3.24 "FB 828 - VMC_WriteDWordParameter - Doppelwort-Parameter an Achse schreiben" Seite 801*

15.9 Antrieb über HMI steuern

15.9.1 Übersicht

Die Antriebssteuerung über ein HMI ist bei folgenden Bibliotheks-Gruppen möglich:

- *Sigma-5* EtherCAT *578*
- *Sigma-7S* EtherCAT *598*
- *Sigma-7W* EtherCAT *619*
- *Sigma-5/7* Pulse Train *679*

Zur Steuerung des entsprechenden Antriebs über ein HMI wie Touch Panel bzw. Panel-PC gibt es für Movicon eine Symbolbibliothek. Hiermit können Sie mit Hilfe von Templates den entsprechenden VMC_AxisControl-Funktionsbaustein ansteuern. Die Symbolbibliothek bietet Ihnen folgende Templates:

- Numeric Touchpad
  - Dies ist ein an die VMC_AxisControl-Templates angepasstes Eingabefeld für verschiedene Displayauflösungen.
  - Das Touchpad können Sie anstelle des Default-Eingabefeldes verwenden.
- VMC_AxisControl
  - Template zur Ansteuerung des FB 860 - VMC_AxisControl-Funktionsbausteins in der CPU.
  - Das Template gibt es für verschiedene Displayauflösungen.

- VMC_AxisControl ... Trend
  - Template zur Ansteuerung des FB 860 - VMC_AxisControl-Funktionsbausteins in der CPU mit zusätzlicher Anzeige der Fahrkurve.
  - Der Einsatz dieses Templates kann sich auf die Performance des Panels auswirken.
  - Das Template gibt es für verschiedene Displayauflösungen.
- VMC_AxisControl_PT
  - Template zur Ansteuerung des FB 875 - VMC_AxisControl_PT-Funktionsbausteins in der CPU, dessen Antrieb über Pulse Train angebunden ist.
  - Das Template gibt es für verschiedene Displayauflösungen.

Bitte beachten Sie, dass aktuell keine ECO-Panels unterstützt werden!

Installation in Movicon

1. Gehen Sie in den Service-Bereich von www.vipa.com.
2. Laden Sie aus dem Downloadbereich unter *"VIPA Lib"* die *"Symbolbibliothek für Movicon"*.
3. Geben Sie ein Zielverzeichnis an, in dem die Bausteine abzulegen sind und starten Sie den Entpackvorgang mit [OK].
4. Öffnen Sie die Bibliothek nach dem Entpackvorgang und ziehen Sie per Drag&Drop die *Symbolbibliothek "vipa simple motion control VX.X.X.msxz"* und die *Sprachtabelle "vipa simple motion control VX.X.X.CSV"* in das Movicon-Benutzer-Verzeichnis ...\Public\Documents\Progea\Movicon\Symbols.

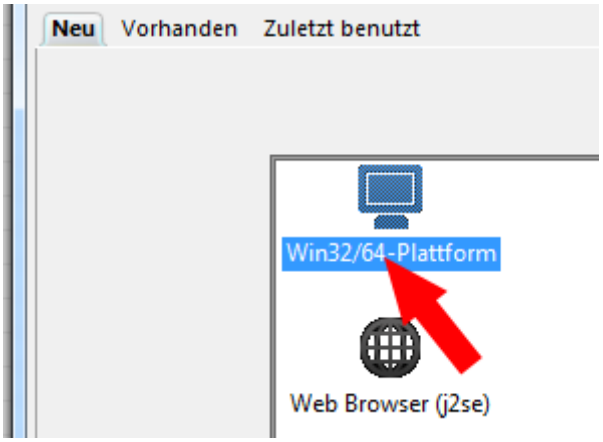
   ⇨ Nach einem Neustart von Movicon steht ihnen in Movicon die Symbolbibliothek über die *"Vorlagenbibliothek"* zur Verfügung.

   Damit die Texte der Templates richtig angezeigt werden, müssen Sie die Sprachtabelle in Ihr Projekt importieren. *"Sprachtabelle importieren" Seite 832*

15.9.2 Neues Projekt erstellen

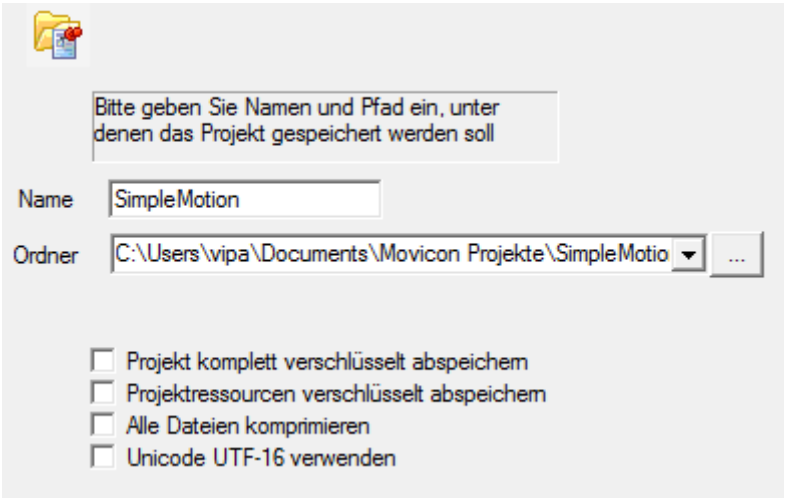
Projekt erstellen

1. Starten Sie Movicon und öffnen Sie den Projekt-Wizard über *"Datei ➔ Neu"*.
2. Wählen Sie als Zielplattform *"Win32/64-Plattform "* und klicken Sie auf [Öffnen].



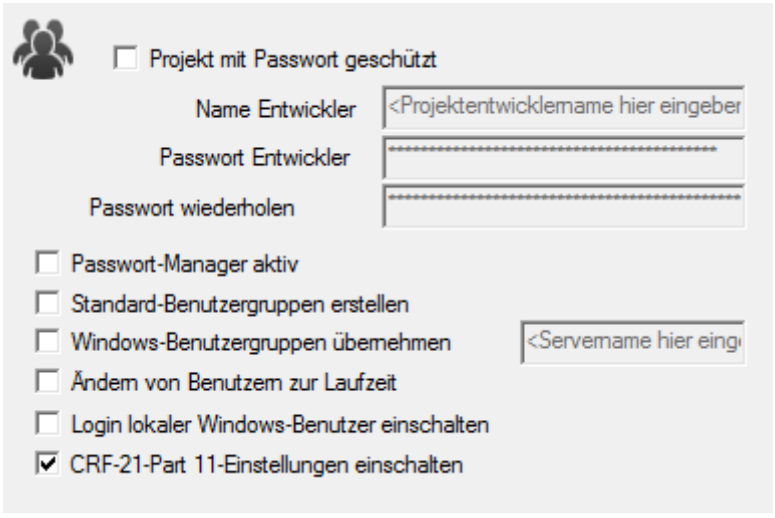
   ⇨ Es öffnet sich das Dialogfenster *"Projektname"*.

3. Geben Sie unter *"Name"* einen Namen für Ihr Projekt an.
   Geben Sie unter *"Ordner"* einen Speicherort an.
   Lassen Sie alle Einstellungen deaktiviert und klicken Sie auf [Weiter].



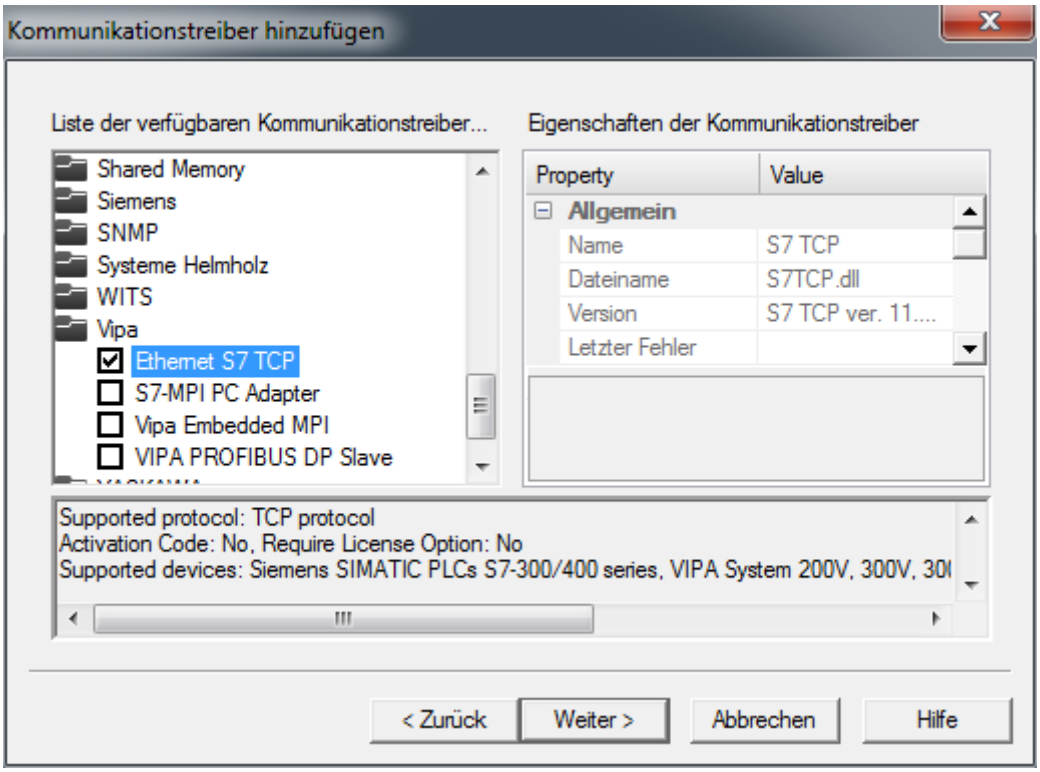
⇨ Es öffnet sich das Dialogfenster *"Benutzer"*.

4. Treffen Sie die passenden Benutzereinstellungen, sofern gewünscht, bzw. aktivieren Sie nur *"CRF-21-Part..."* und klicken Sie auf [Weiter].



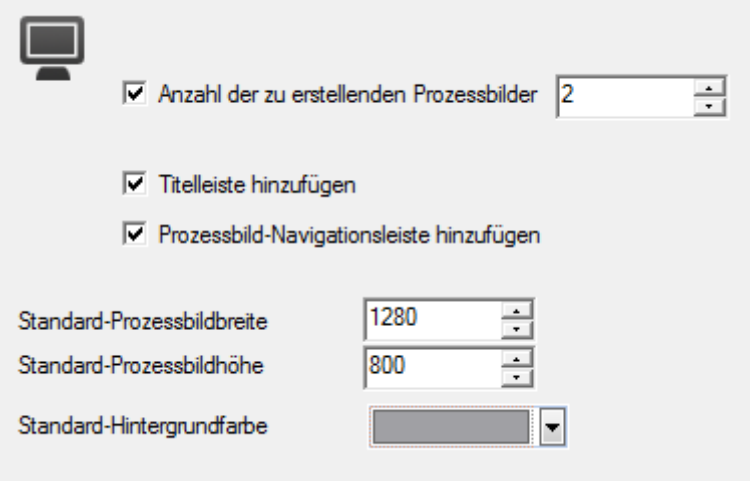
⇨ Es öffnet sich das Dialogfenster *"Kommunikationstreiber hinzufügen"*.

5. Da die Anbindung zur CPU über TCP/IP erfolgt, aktivieren Sie in der *"Liste der verfügbaren Kommunikationstreiber"* den Treiber *"VIPA"* > *"Ethernet S7 TCP"* und klicken Sie auf [Weiter].



⇨ Es öffnet sich das Dialogfenster *"Prozessbilder"*.

6. Geben Sie 2 Prozessbilder und deren Größe, welche zu ihrem Panel passt an und klicken Sie auf [Weiter].



⇨ Es öffnet sich das Dialogfenster *"Datenbankeinstellungen (ODBC)"*.

7. Sofern Sie eine Datenbankanbindung wünschen, können Sie hier die entsprechenden Einstellungen durchführen. Ansonsten klicken sie auf [Weiter].

⇨ Es öffnet sich das Dialogfenster *"Datenlogger und Rezept-Einstellungen (ODBC)"*.

8. Sofern Vorlagen generiert werden sollen, können Sie hier die entsprechenden Einstellungen durchführen. Ansonsten klicken sie auf [Weiter].

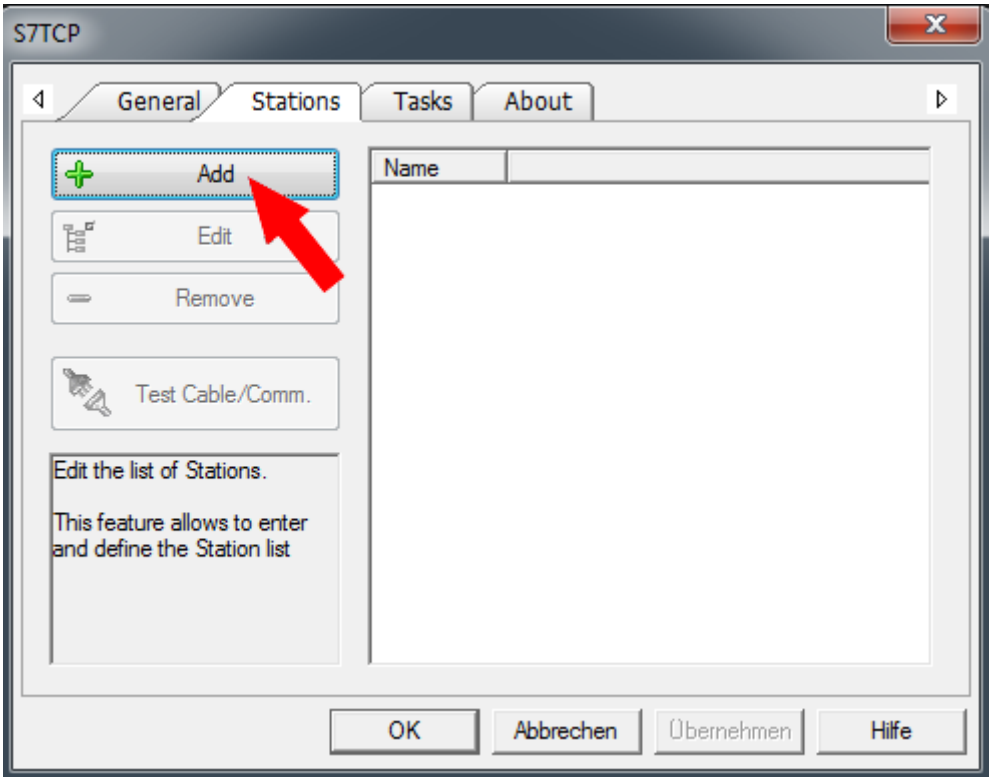
⇨ Es öffnet sich das Dialogfenster *"Alarm-Einstellungen"*.

9. Sofern Alarme generiert werden sollen, können Sie hier die entsprechenden Einstellungen durchführen. Ansonsten klicken sie auf [Fertig stellen].

⇨ Ihr Projekt wird mit den getroffenen Einstellungen erstellt und es öffnet sich automatisch der Einstelldialog für den Kommunikationstreiber *"S7TCP"*.

10. Wählen Sie den Reiter *"Stations"*.

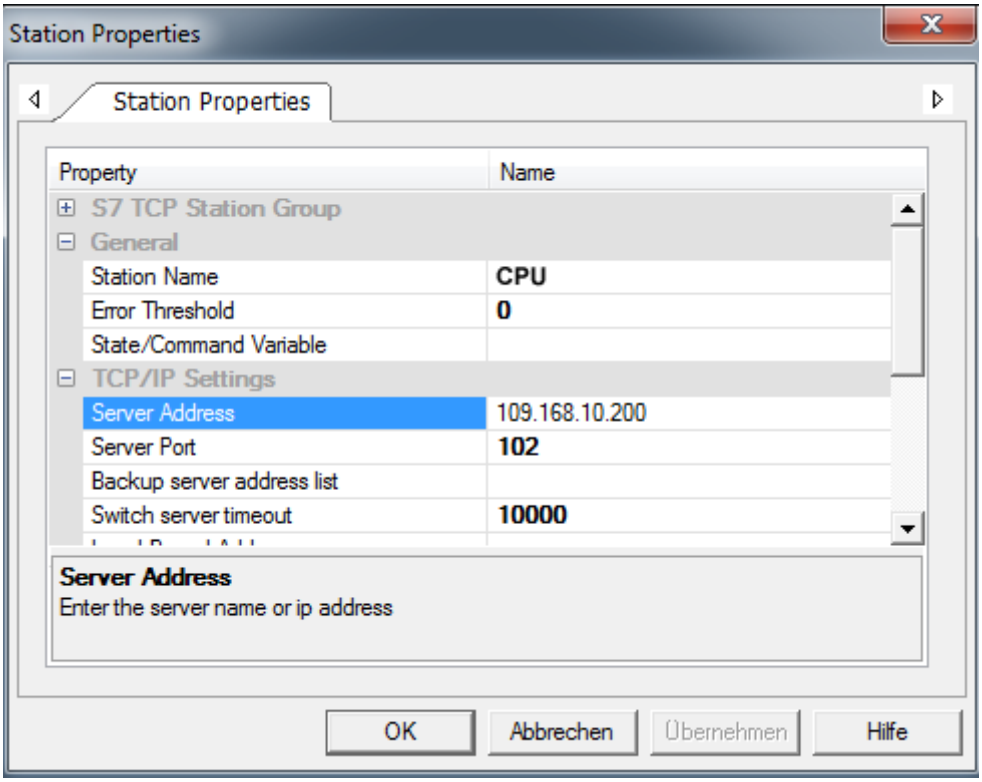
11. Zum Hinzufügen einer neuen Station klicken Sie auf [+ Add].



⇨ Es öffnet sich das Dialogfenster *"Station Properties"*.

12. Geben Sie unter *"Station Name"* einen Stationsnamen an. Dieser Name ist weiter unten für das Prozessbild im Initialisierungs-Dialog zu übernehmen. Erlaubte Zeichen: *A-Z*, *a-z*, *0-9*, Leerzeichen und die Trennzeichen "_" und "-"

Geben Sie unter *"Server Address"* die IP-Adresse der CPU an und klicken Sie auf [OK].



13. Verneinen Sie die Abfrage zum Import von Variablen aus der SPS-Datenbank und schließen Sie den *"S7TCP"*-Dialog mit [OK].

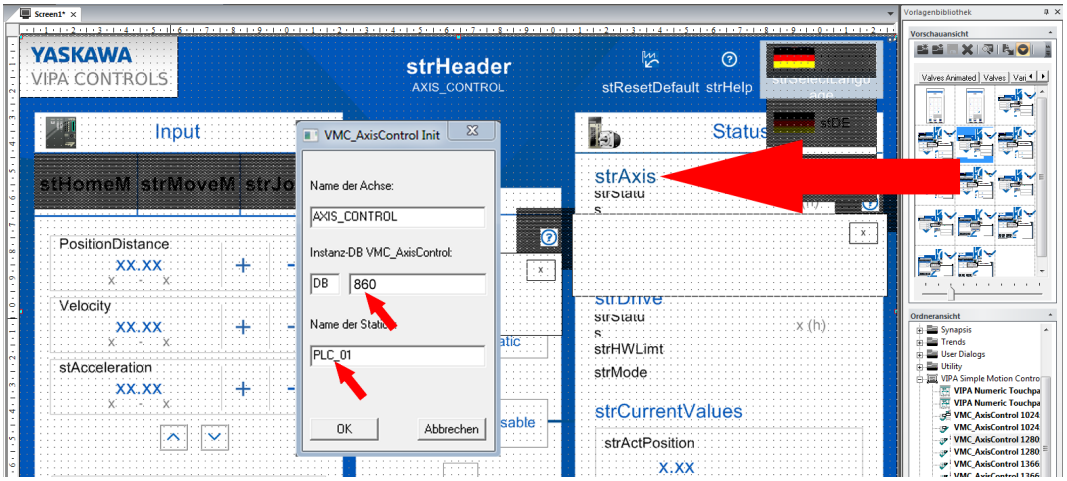
⇨ Das Projekt und der Arbeitsbereich sind nun für die Benutzung freigeben. Im Projektbaum wurde unter *"Ressourcen > SimpleMotion"* zu den Standardelementen folgendes angelegt:

- Kommunikationsobjekte
  - Kommunikationstreiber
    S7 TCP
- Prozessbilder
  - Prozessbild1
  - Prozessbild2
  - Schaltflächenleiste

15.9.3 Projekt in Movicon anpassen

Prozessbild konfigurieren

1. Öffnen Sie über *"Ressourcen > SimpleMotion > Prozessbilder"* das *"Prozessbild1"*.
2. Navigieren Sie in der *"Ordneransicht"* zu *"vipa simple motion control ..."* und ziehen Sie aus der *"Vorschauansicht"* das Template in das *"Prozessbild1"*, welches der Auflösung Ihres Panels entspricht.



⇨ Es öffnet sich der Initialisierungs-Dialog

3. Geben Sie einem Namen für die Achse an. Erlaubte Zeichen: *A-Z*, *a-z*, *0-9*, Leerzeichen und die Trennzeichen "_" und "-"

   Geben Sie die Nummer des Instanz-DB an, welchen Sie in ihrem SPS-Programm verwenden.

   Geben Sie den Stationsnamen an. Dieser muss mit dem *"Station Name"* aus den *"Station Properties"* der *"S7 TCP"*-Kommunikationseinstellungen übereinstimmen. Erlaubte Zeichen: *A-Z*, *a-z*, *0-9*, Leerzeichen und die Trennzeichen "_" und "-"

   ⇨ Mit [OK] werden alle Variablen, sowie deren Strukturen erzeugt und die Adressen werden auf die vorgegebene Zieladresse eingestellt.

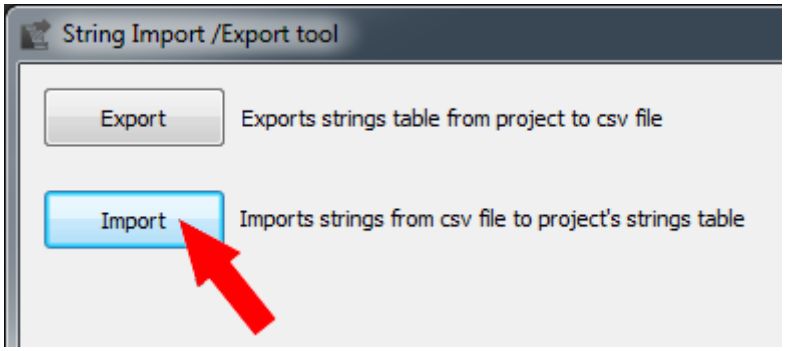
4. Platzieren Sie das Template und passen Sie ggf. dessen Größe an.

Für jedes Template werden Variablen unter dem entsprechenden Namen angelegt. Beim Löschen des Templates sind die entsprechenden Variablen wieder zu löschen. Diese können Sie sich unter "Ressourcen > SimpleMotion > Kommunikationsobjekte > Variablen" markieren lassen. Beim Löschen sind diese zusammen mit dem übergeordneten Verzeichnis zu löschen. Sofern keine weiteren Templates auf die "Strukturdefinitionen" für den Axis-Control zugreifen, sind diese ebenfalls zu löschen.

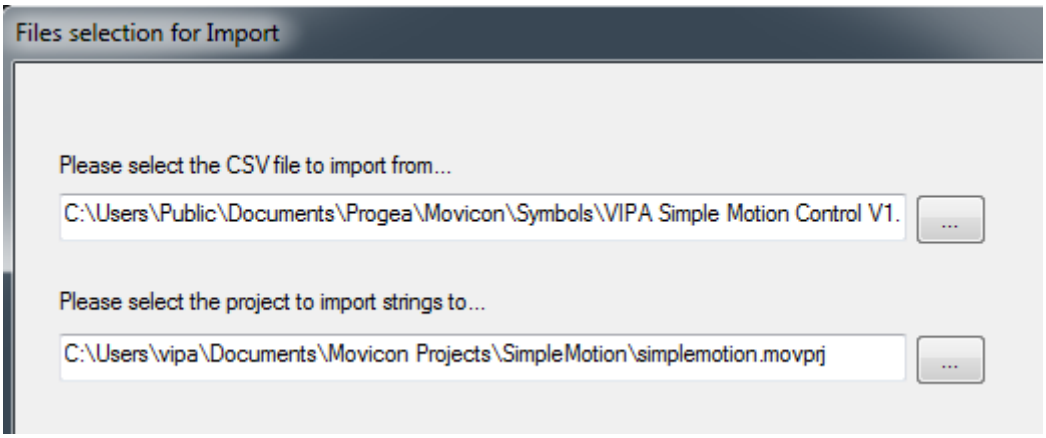
Sprachtabelle importieren

Die Templates beziehen die angezeigten Texte aus einer Sprachtabelle, welche aus dem Arbeitsverzeichnis in Ihr Projekt zu importieren ist.

1. Wählen Sie *"Extras ➔ Zeichenfolgen Import/Export"*.
   ⇨ Es öffnet sich das *"String Import/Export tool"*.



2. Klicken Sie auf [Import].
3. Für die CSV-Datei navigieren Sie mit [...] in Ihr Movicon-Benutzer-Verzeichnis ...\Public\Documents\Progea\Movicon\Symbols und wählen Sie die Datei *"vipa simple motion control VX.X.X.CSV"* aus.
4. Als Projektverzeichnis geben Sie die Projekt-Datei *"simplemotion.movprj"* an, welche sich im Benutzer-Verzeichnis wie z.B. ...\vipa\Documents\Movicon Projects \SimpleMotion befindet.



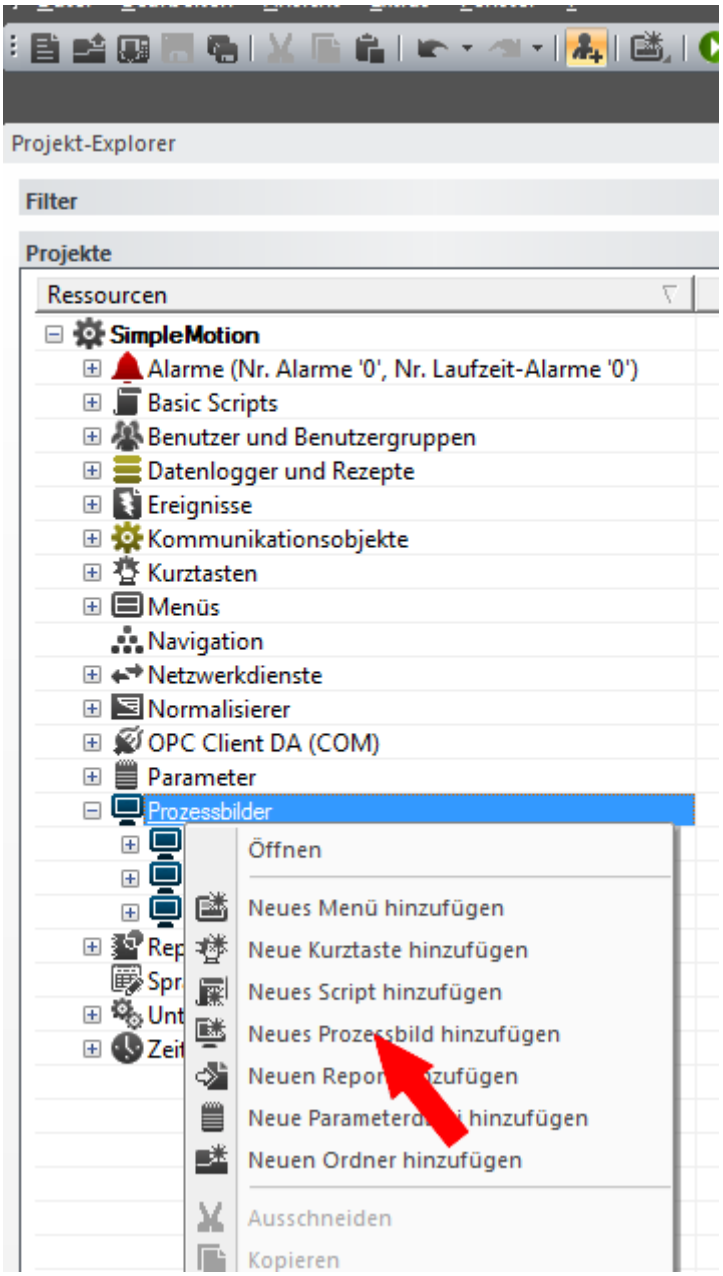
5. Klicken Sie auf [Weiter].
   ⇨ Es öffnet sich *"Language selection"*.
6. Wählen Sie [Select all languages] und klicken Sie auf [Fertig stellen].

   ⇨ Die Sprachtabelle wird in Ihr Projekt importiert.
7. Schließen Sie nach erfolgreichem Import das *"String Import/Export tool"*.

Numerisches Eingabefeld anpassen

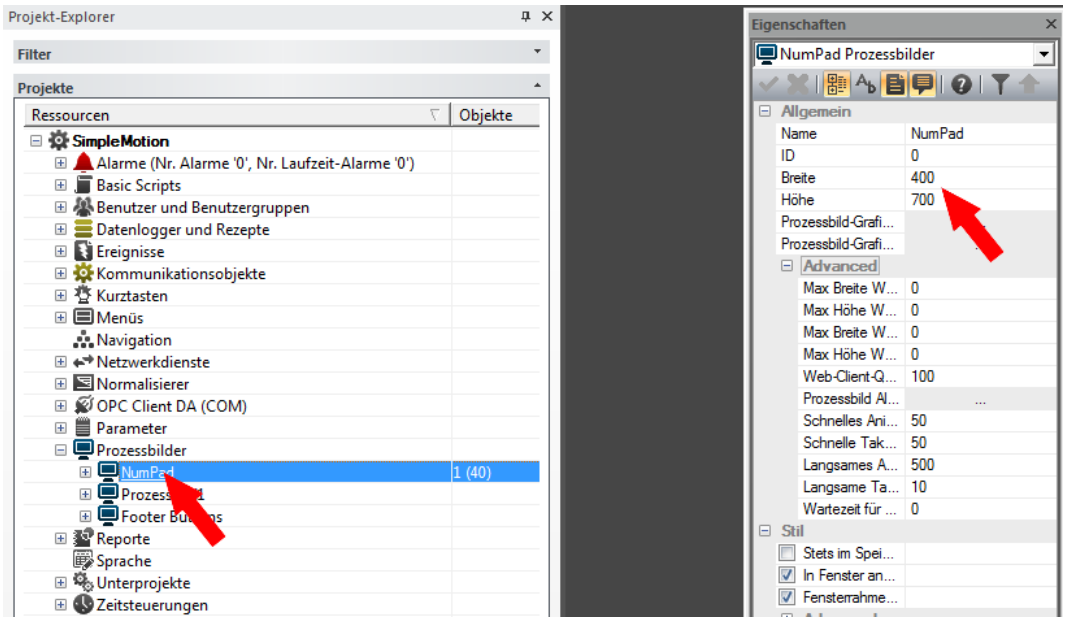
Unter den Templates befindet sich ein *"Numeric Touchpad"* in verschiedenen Auflösungen. Dies ist ein an die VMC_AxisControl-Templates angepasstes Eingabefeld für verschiedene Displayauflösungen. Mit folgender Vorgehensweise können Sie dieses Touchpad anstelle des Default-Eingabefeldes verwenden.

1. Klicken Sie auf *"Ressourcen > SimpleMotion > Prozessbilder"* und wählen Sie *"Kontextmenü ➔ Neues Prozessbild hinzufügen"*.

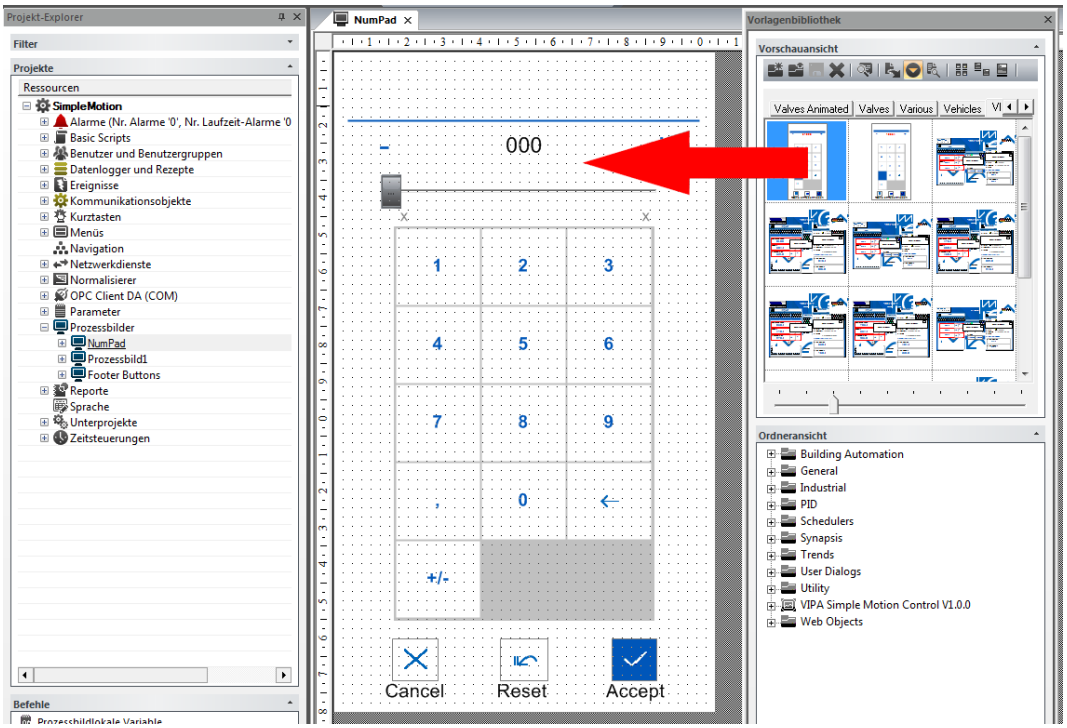


2. Vergeben Sie einem Namen wie z.B. *"NumPad"* und bestätigen Sie mit [OK].

3. Klicken Sie auf das Prozessbild *"NumPad"* und passen Sie über *"Kontextmenü ➔ Eigenschaften"* Breite und Höhe an wie z.B. *"Breite"* = 400 und *"Höhe"* = 700. Übernehmen Sie mit ✔ Ihre Einstellungen.



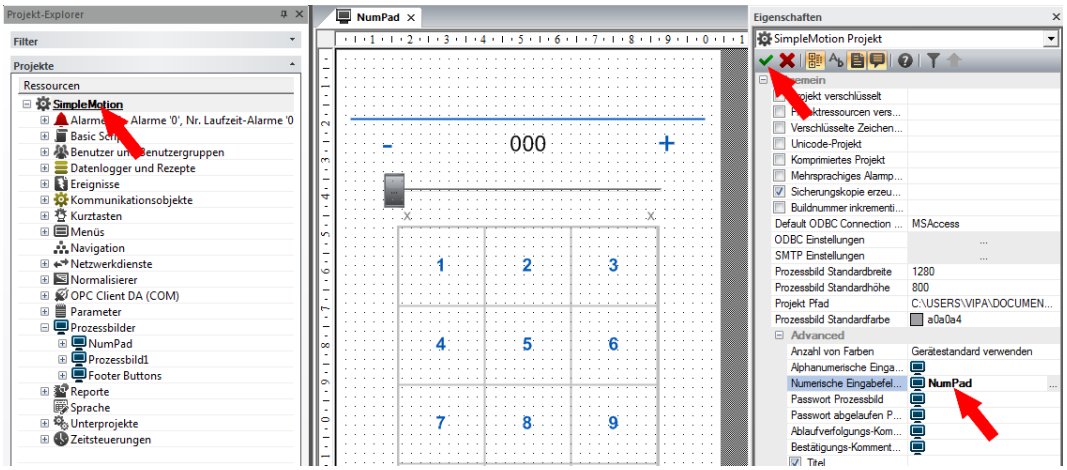
4. Wählen Sie *"Ansicht ➔ Vorlagenbibliothek"*. Navigieren Sie in der *"Ordneransicht"* zu *"vipa simple motion control ..."* und ziehen Sie aus der *"Vorschauansicht"* das *"Numeric Touchpad"*-Template in das Prozessbild *"NumPad"*, welches der Auflösung Ihres Panels entspricht.



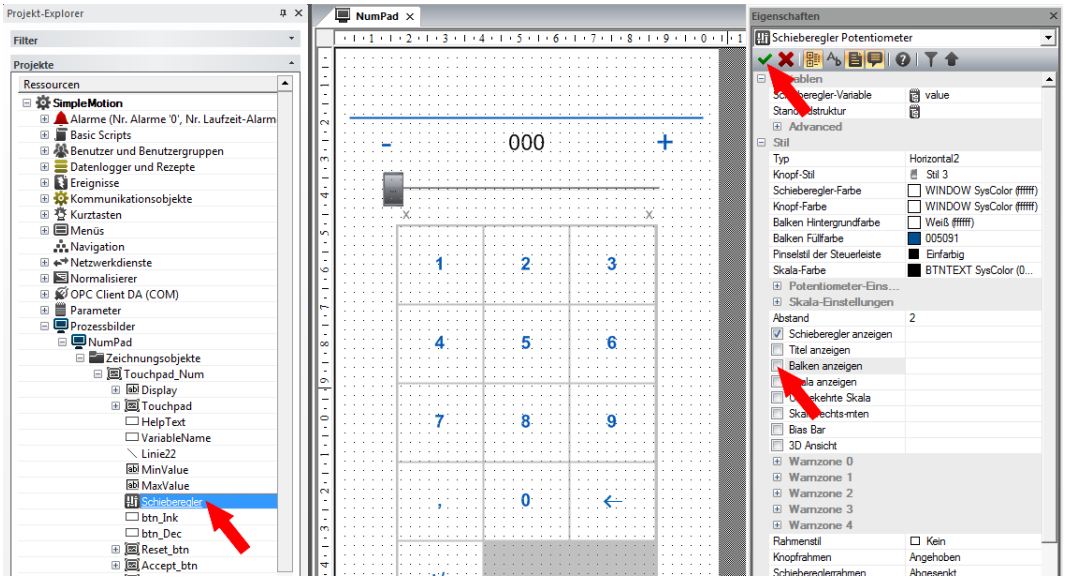
5. Passen Sie ggf. dessen Größe an.

6. Klicken Sie auf *"Ressourcen > SimpleMotion"* und wählen Sie *"Kontextmenü ➔ Eigenschaften"*.

7. Stellen Sie unter *"Allgemein > Advanced"* das numerische Eingabefeld *"NumPad"* ein. Übernehmen Sie mit ✔ Ihre Einstellungen.



8. Zur optischen Anpassung klicken Sie unter *"Ressourcen > SimpleMotion > Prozessbilder > NumPad > Zeichnungsobjekte > Touchpad_Num"* auf *"Schieberegler"* und wählen Sie *"Kontextmenü ➔ Eigenschaften"*. Erweitern Sie den Bereich *"Stil"* und deaktivieren Sie *"Balken anzeigen"*.



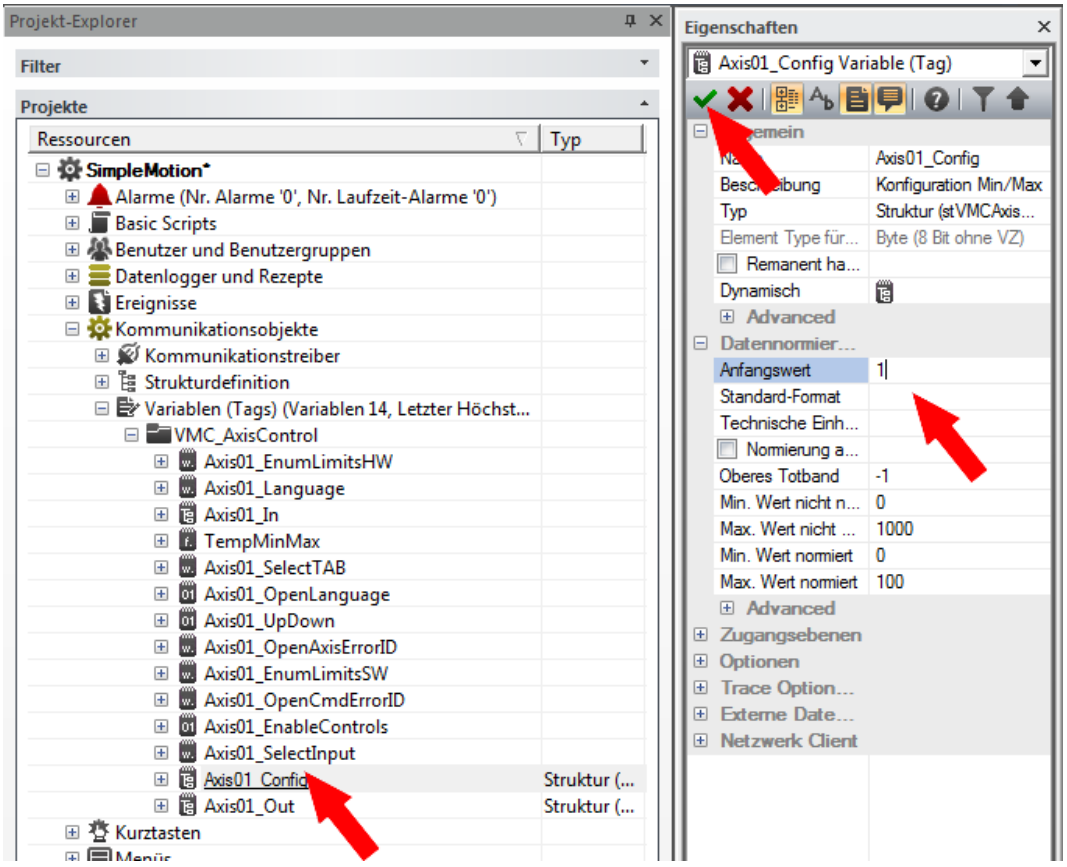
Grenzwerte und Standard-werte anpassen

Bei der Platzierung eines Templates in einem Prozessbild werden die zugehörigen Variablen und Strukturdefinitionen unter *"Ressourcen > SimpleMotion > Kommunikationsobjekte > Variablen > VMC_AxisControl > ..._Config"* automatisch erzeugt. Hierbei werden folgende Variablen angelegt und mit Initialwerten versehen:

AccelerationMaxValue	-	Maximaler Beschleunigungswert
AccelerationMinValue	-	Minimaler Beschleunigungswert
DecelerationMaxValue	-	Maximaler Verzögerungswert
DecelerationMinValue	-	Minimaler Verzögerungswert
HomePosMaxValue	-	Maximale Home-Position
HomePosMinValue	-	Minimale Home-Position
JogAccelerationMaxValue	-	Maximale Beschleunigungswert Jog-Modus
JogAccelerationMinValue	-	Minimaler Beschleunigungswert Jog-Modus
JogDecelerationMaxValue	-	Maximaler Verzögerungswert Jog-Modus
JogDecelerationMinValue	-	Minimaler Verzögerungswert Jog-Modus
PositionMaxValue	-	Maximaler Positionswert
PositionMinValue	-	Minimaler Positionswert
VelocityMaxValue	-	Maximaler Geschwindigkeitswert
VelocityMinValue	-	Minimaler Geschwindigkeitswert

→ Zur Anpassung der Grenz-und Standardwerte klicken Sie auf *"Ressourcen > SimpleMotion > Kommunikationsobjekte > Variablen > VMC_AxisControl > ..._Config"* und wählen Sie *"Kontextmenü ➔ Eigenschaften"*.

⇨ Unter *"Datennormierung"* können Sie die entsprechenden Werte anpassen. Übernehmen Sie mit ✔ Ihre Einstellungen.

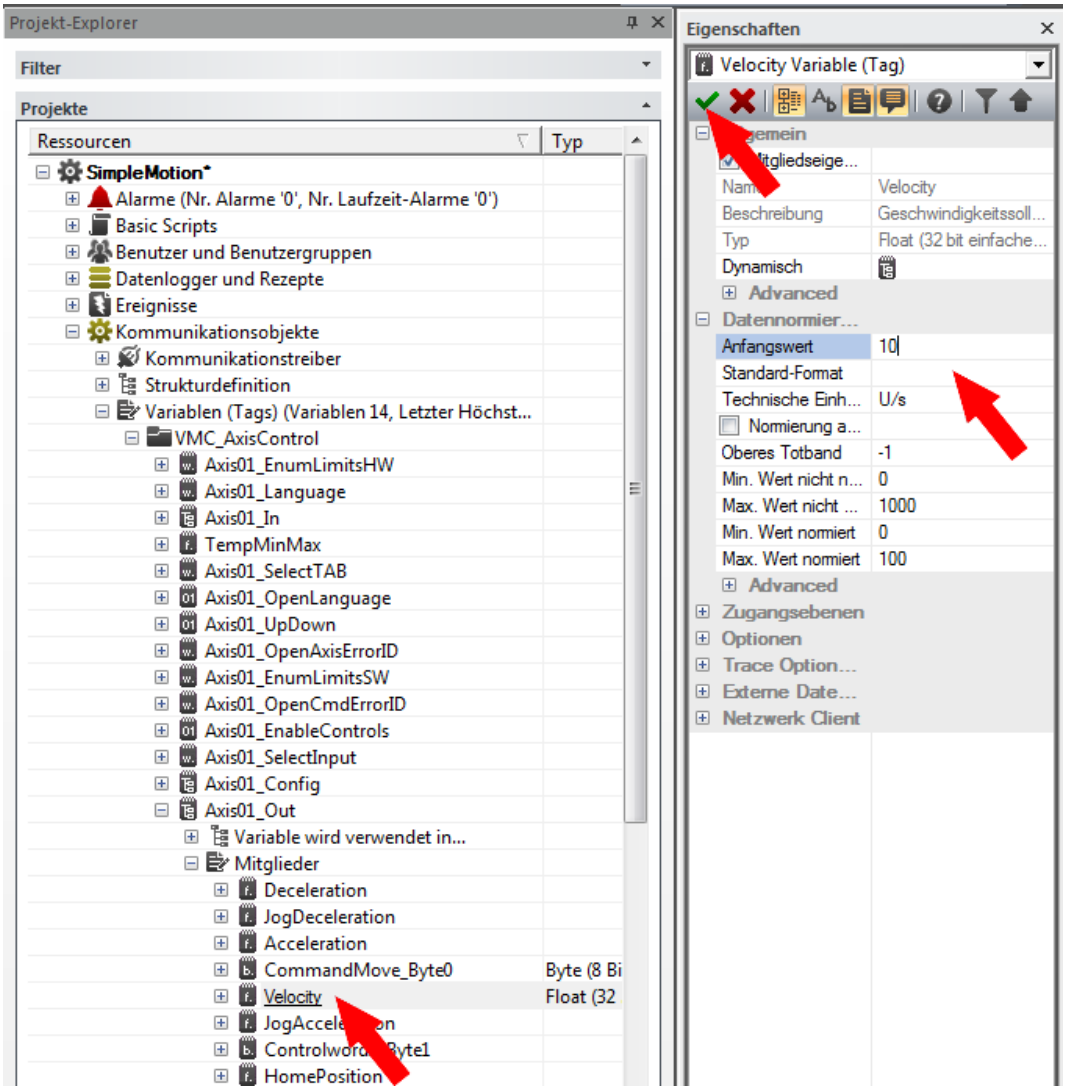


Technische Einheiten anpassen

Bei der Platzierung eines Templates in einem Prozessbild werden die zugehörigen Variablen mit ihren technischen Einheiten automatisch erzeugt. Diese können Sie über die Eigenschaften anpassen.

→ Zur Anpassung der Technischen Einheiten z.B. für die Geschwindigkeit klicken Sie auf *"Ressourcen > SimpleMotion > Kommunikationsobjekte > Variablen > VMC_AxisControl > ..._Out > Mitglieder > Velocity"* und wählen Sie *"Kontextmenü ➔ Eigenschaften"*.

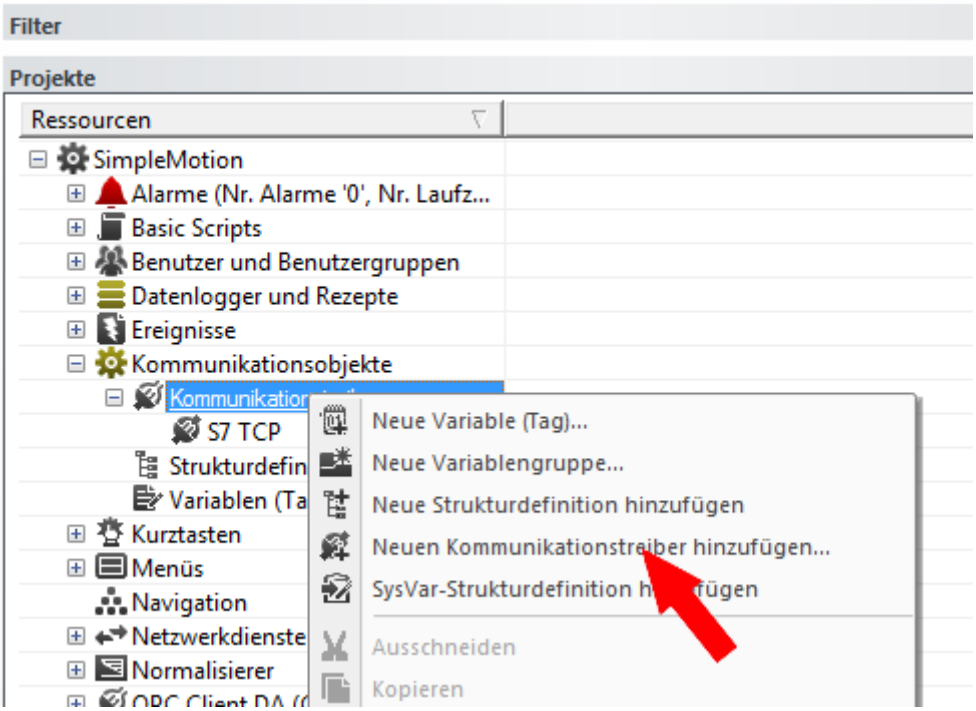
⇨ Unter *"Datennormierung"* können Sie die entsprechenden Werte anpassen. Übernehmen Sie mit ✔ Ihre Einstellungen.



Kommunikationstreiber manuell hinzufügen

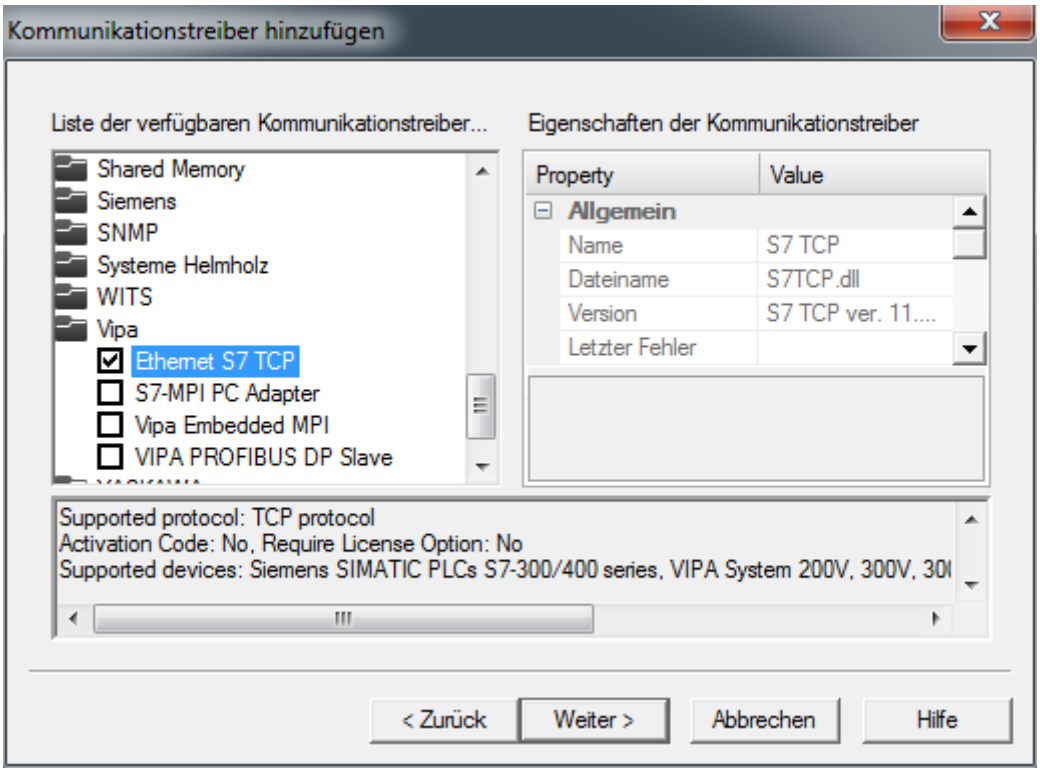
Anstelle über Wizard können Sie den Kommunikationstreiber auch manuell hinzufügen:

1. Klicken Sie unter *"Ressourcen > SimpleMotion > Kommunikationsobjekte"* auf *"Kommunikationstreiber"* und wählen Sie *"Kontexmenü*
   ➔ *Neuen Kommunikationstreiber hinzufügen"*.



⇨ Es öffnet sich das Dialogfenster *"Neuer Komm. E/A-Treiber"*.

2. Da die Anbindung zur CPU über TCP/IP erfolgt, aktivieren Sie in der *"Liste der verfügbaren Kommunikationstreiber"* den Treiber *"VIPA"* > *"Ethernet S7 TCP"* und klicken Sie auf [OK].



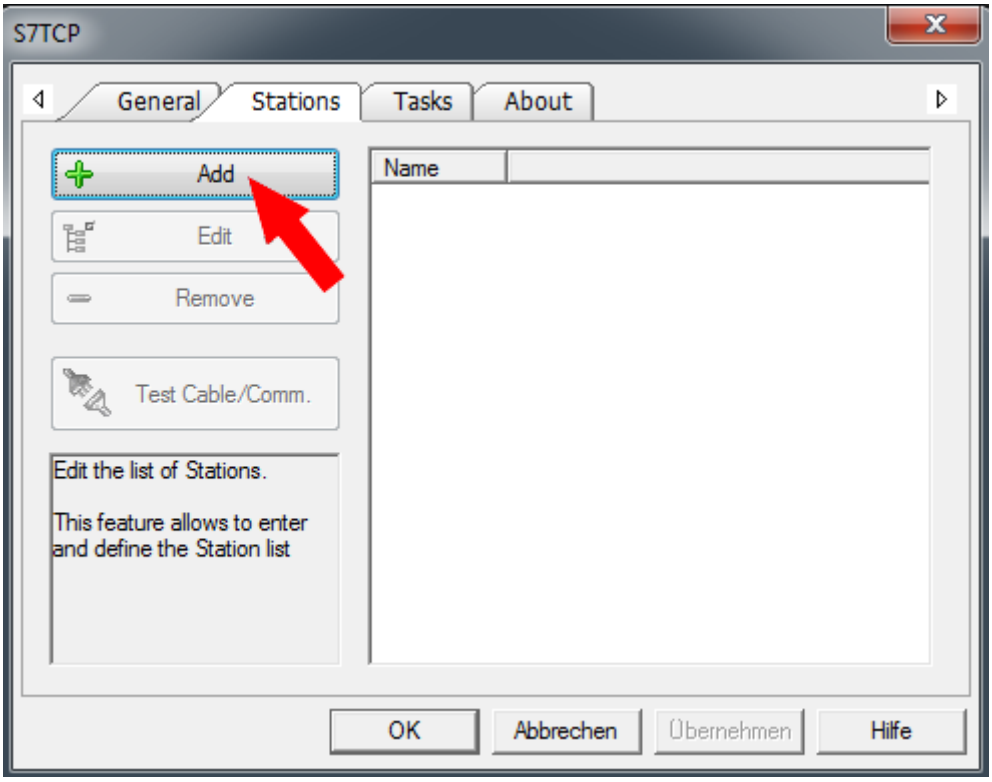
⇨ Der Kommunikationstreiber *"S7 TCP"* wird unter *"Ressourcen > SimpleMotion > Kommunikationsobjekte > Kommunikationstreiber"* aufgeführt.

3. Klicken Sie auf *"S7 TCP"* und wählen Sie *"Kontextmenü ➔ Kommunikationstreiber-Einstellungen"*.

⇨ Es öffnet sich das Dialogfenster *"S7 TCP"*.

4. Wählen Sie den Reiter *"Stations"*.

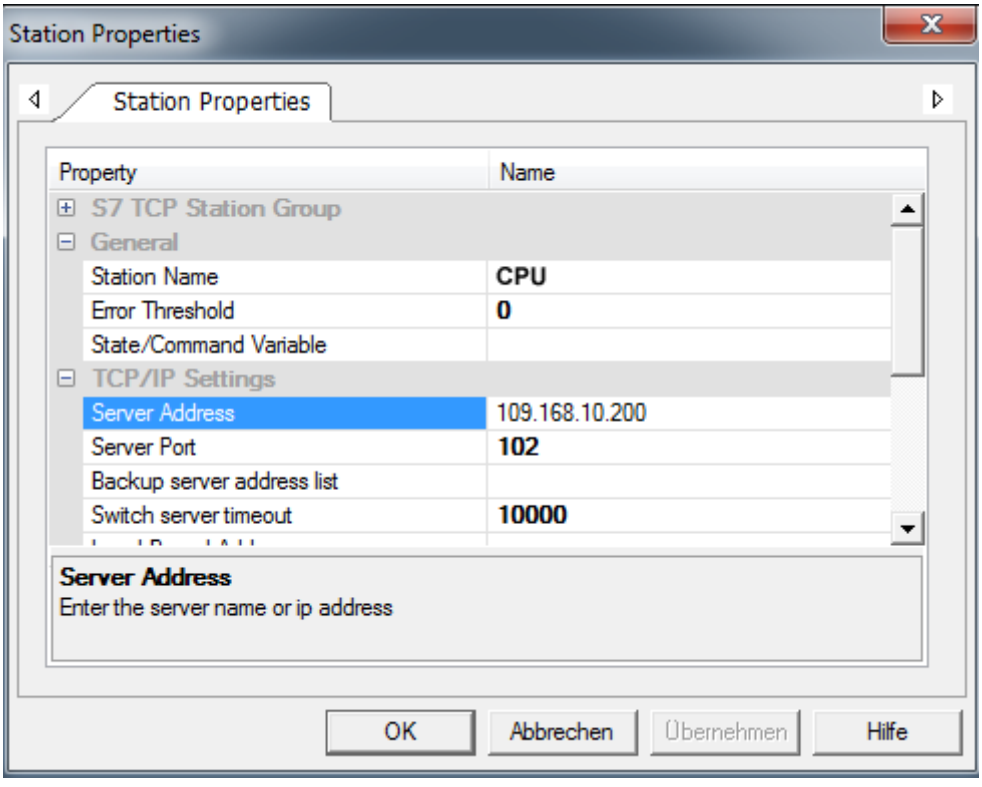
5. Zum Hinzufügen einer neuen Station klicken Sie auf [+ Add].



⇨ Es öffnet sich das Dialogfenster *"Station Properties"*.

6. Geben Sie unter *"Station Name"* einen Stationsnamen an. Erlaubte Zeichen: *A-Z*, *a-z*, *0-9*, Leerzeichen und die Trennzeichen "_" und "-"

Geben Sie unter *"Server Address"* die IP-Adresse der CPU an und klicken Sie auf [OK].



7. Verneinen Sie die Abfrage zum Import von Variablen aus der SPS-Datenbank und schließen Sie den *"S7 TCP"*-Dialog mit [OK].

15.9.4 Inbetriebnahme

15.9.4.1 Projekt auf Zielgerät übertragen

Ihr Projekt können Sie über Ethernet in Ihr Panel übertragen. Über die in Ihrem Panel vorinstallierte Movicon Runtime-Version wird Ihr Projekt ablauffähig.

1. Verbinden Sie Ihren PC und Ihr Panel über Ethernet.
2. Starten Sie Ihr Panel und ermitteln Sie im *"Startup-Manager"* die IP-Adresse Ihres Panels.
3. Rufen Sie im *"Startup-Manager"* den Menüpunkt *"Autostart"* auf.
4. Damit Sie mit Movicon über Ethernet ein Projekt in Ihr Panel übertragen können, müssen Sie unter *"Autostart"* die Option *"Movicon TCP Upload Server"* aktivieren.

⇨ Bestätigen Sie die Abfrage zur Aktivierung.

5. Sie können jetzt aus Movicon Ihr Projekt in Ihr Panel übertragen. Klicken Sie hierzu in Movicon in *"Ressourcen"* auf *"SimpleMotion"* und wählen Sie *"Kontextmenü ➔ Projekt zum Zielgerät transferieren"*.

   ⇨ Es öffnet sich der Transfer-Dialog.

6. Wählen Sie unter *"PlugIn-Typ"* *"TCP"*.

   Geben Sie unter *"Server"* die IP-Adresse des Panels an.

   Tragen Sie unter *"Benutzername"* und *"Passwort"* die Zugangsdaten zu Ihrem Panel ein.

   Standardmäßig werden folgende Zugangsdaten verwendet:

   - Benutzername: wince
   - Passwort: vipatp

   Geben Sie unter *"Zielpfad"* Ihre Speicherkarte an und legen Sie ein Projektverzeichnis an.

7. Starten Sie die Übertragung mit [Projekt transferieren].

8. Nach erfolgreichem Transfer können Sie Ihr Projekt auf dem Panel im Autostart-Verzeichnis eintragen und in Betrieb nehmen.

VORSICHT!
Bitte beachten Sie immer die Sicherheitshinweise zu ihrem Antrieb, insbesondere bei der Inbetriebnahme!

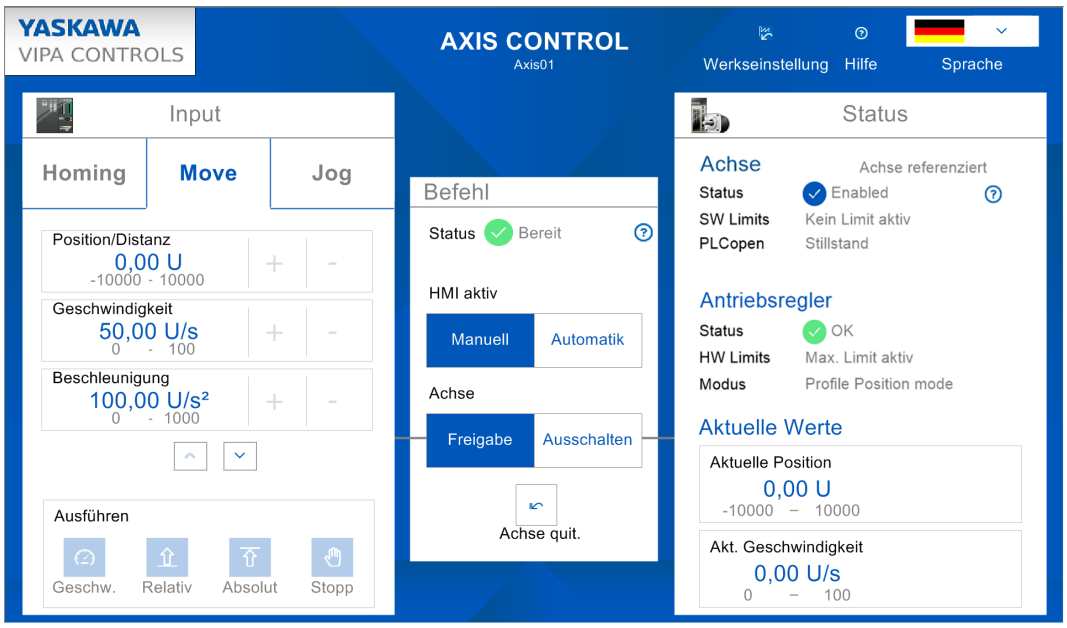
15.9.4.2 *VMC_AxisControl* über das Panel ansteuern

15.9.4.2.1 Inbetriebnahme

Es wird vorausgesetzt, dass Sie Ihre Applikation aufgebaut haben und Sie Ihren Antrieb mit dem VMC_AxisControl-Funktionsbaustein ansteuern können.

Binden Sie Ihre CPU an Ihr Panel an und schalten Sie Ihre Applikation ein.

⇨ Das Panel startet mit der Oberfläche zur Steuerung Ihres Antriebs.



Damit Sie Ihren Antrieb über das Panel steuern können, schalten Sie "HMI aktiv" auf [Manuell]. Sofern die Statusanzeige keine Fehler zurückmeldet, können Sie den Antrieb mit [Freigabe] für die Ansteuerung freigeben. Sie können jetzt Ihren Antrieb über die entsprechenden Schaltflächen steuern.

15.9.4.2.2 Bedienung

Oberfläche

"Werkseinstellung"

- Über *"Werkseinstellung"* werden folgende Werte auf die Defaultwerte der Applikation zurückgesetzt, welche Sie, wie weiter oben beschrieben, entsprechend anpassen können:
  - Geschwindigkeit: 50U/s
  - Beschleunigung/Verzögerung: 100U/s^2
  - Position/Home Position: 0U

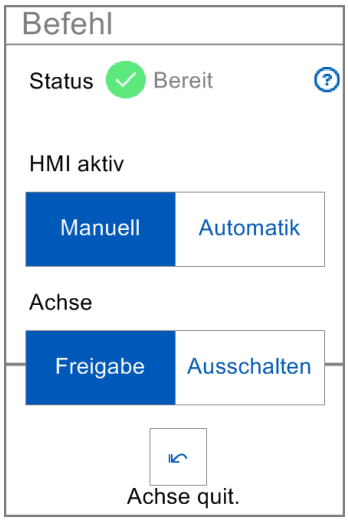
"Hilfe"

- Über *"Hilfe"* können Sie Ihre eigene Hilfedatei abrufen. Diese ist in Movicon entsprechend einzubinden.

"Sprache"

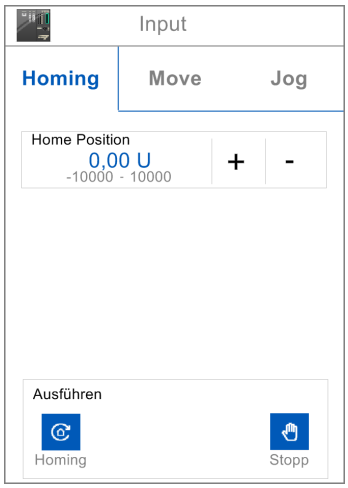
- Über *"Sprache"* können Sie die entsprechende Sprache für die Bedienoberfläche vorgeben.

"Befehl"



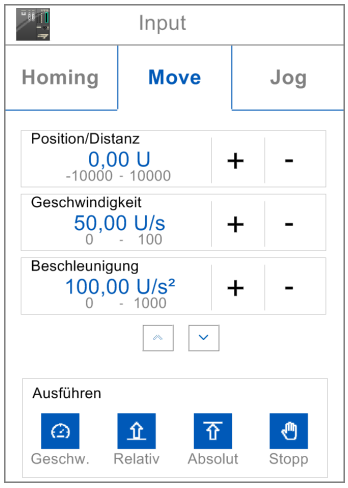
- *"Status"*
  - Hier bekommen Sie den aktuellen Status Ihres Fahrbefehls angezeigt.
- *"HMI aktiv"*
  - *"Manuell"*: Im aktivierten Zustand kann der Antrieb über das Panel gesteuert werden.
  - *"Automatik"*: Im aktivierten Zustand erfolgt die Steuerung des Antriebs über das SPS-Programm Ihrer CPU und kann vom Panel nicht beeinflusst werden.
- *"Achse"*
  - *"Freigabe"*: Im aktivierten Zustand und bei aktiviertem *"Manuell"* von *"HMI aktiv"* ist der Antrieb freigegeben und Sie können diesen über den Bereich *"Input "* steuern.
  - *"Ausschalten"*: Im aktivierten Zustand ist der Antrieb gesperrt und keine Steuerung möglich.
- *"Achse quit"*
  - Im Fehlerfall werden die Schaltflächen zur Steuerung inaktiv. Mit *"Achse quit."* können Sie Fehler quittieren und Schaltflächen wieder aktivieren.

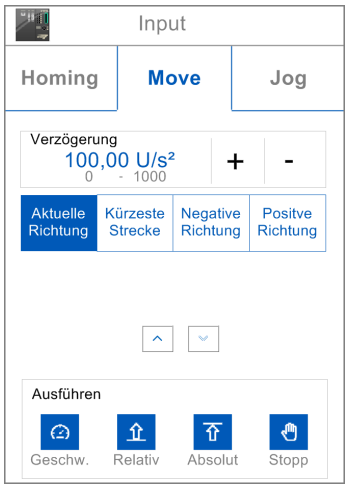
"Input"



"Homing"

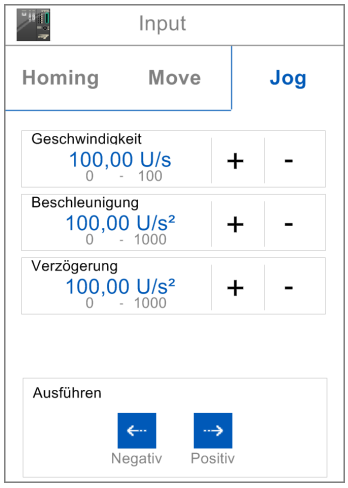
- Über das Eingabefeld bzw. [+] und [-] können Sie eine Homing-Position vorgeben und diese über *"Ausführen > Homing"* als Referenzpunkt anfahren
- Mit *"Ausführen > Stopp"* können Sie die Referenzfahrt stoppen.





"Move"

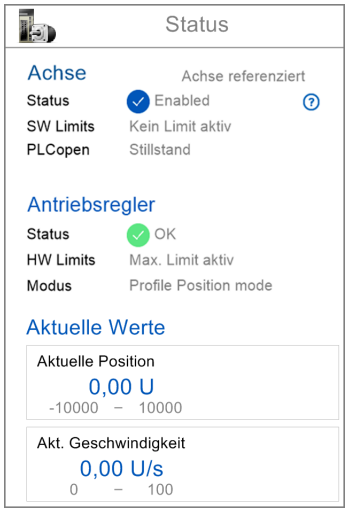
- Über das entsprechende Eingabefeld bzw. [+] und [-] können Sie *"Position/Distanz"*, *"Geschwindigkeit"*, *"Beschleunigung"* und *"Verzögerung"* vorgeben und über den entsprechenden Fahrbefehl unter *"Ausführen"* ausführen lassen. Mit [v] navigieren Sie nach unten.
  - *"Geschw"*: Bei Betätigung führt der Antrieb den Fahrbefehl mit konstanter Geschwindigkeit aus.
  - *"Relativ"*: Bei Betätigung fährt der Antrieb an die relative Position, welche Sie unter *"Position/Distanz"* vorgeben können.
  - *"Absolut"*: Bei Betätigung fährt der Antrieb an die absolute Position, welche Sie unter *"Position/Distanz"* vorgeben können.
  - *"Stopp"*: Bei Betätigung wird der Anrieb gestoppt.
  - *"Aktuelle Richtung"*: Im aktivierten Zustand wird die Fahrtrichtung beibehalten.
  - *"Kürzeste Strecke"*: Im aktivierten Zustand wird die kürzeste Entfernung zur vorgegebenen Position verwendet.
  - *"Negative Richtung"*: Im aktivierten Zustand wird die negative Fahrtrichtung verwendet.
  - *"Positive Richtung"*: Im aktivierten Zustand wird die positive Fahrtrichtung verwendet.



"Jog"

- Über das entsprechende Eingabefeld bzw. [+] und [-] können Sie *"Geschwindigkeit"*, *"Beschleunigung"* und *"Verzögerung"* vorgeben und über die Richtungstaste unter *"Ausführen"* den entsprechenden Fahrbefehl in negative bzw. positive Richtung ausführen lassen.
- Solange Sie eine der Richtungstasten betätigen, wird der Antrieb mit der eingestellten Beschleunigung auf die gewünschte Geschwindigkeit beschleunigt.
- Beim Loslassen der Richtungstaste wird der Antrieb mit der eingestellten Verzögerung gestoppt.

"Status"



"Achse"

- *"Status"*: Hier wird Ihnen der Status Ihrer Achse angezeigt.
  - *"Enabled"*: Die Achse ist eingeschaltet.
  - *"Ready"*: Die Achse ist einschaltbereit.
  - *"Disabled"*: Die Achse ist deaktiviert.
  - *"Achsfehler"*: Es liegt ein Achsfehler vor mit Angabe der Fehlernummer. ↬ *Kap. 15.11 "ErrorID - Zusätzliche Fehlerinformationen" Seite 851*
- *"SW Limits"*: Sobald SW-Grenzen bestehen, wird dies hier angezeigt.
- *"PLCopen"*: Hier wird Ihnen der PLCopen-Status angezeigt.

"Antriebsregler"

- *"Status"*: Hier wird Ihnen der Status des Antriebsreglers angezeigt.
- *"HW-Limits"*: Hier wird Ihnen eine eventuell eingestellte Begrenzung in Ihrem Antriebsregler angezeigt.
- *"Modus"*: Hier erhalten sie Informationen über das aktuell eingestellt Antriebsprofil.

"Aktuelle Werte"

- Hier werden die aktuellen Werte von *"Position"* und *"Geschwindigkeit"* angezeigt.
- Werte, welche außerhalb der definierten Grenzwerte liegen, werden rot eingerahmt.

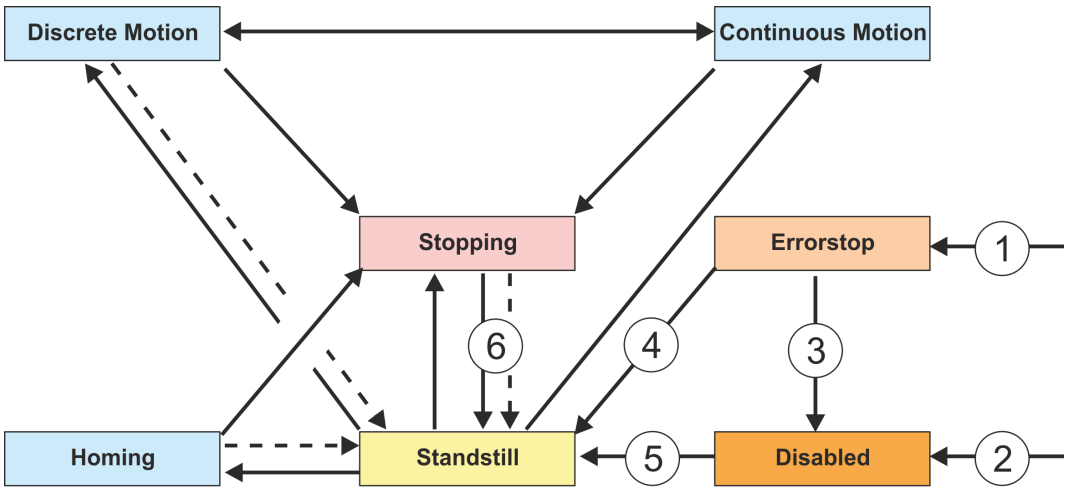
15.10 Zustände und Verhalten der Ausgänge

15.10.1 Zustände

Zustandsdiagramm

Im *Zustandsdiagramm* sind alle Zustände aufgeführt, die eine Achse annehmen kann. Eine Achse befindet sich immer in einem dieser Zustände. Je nach Ausgangszustand kann ein Zustandswechsel automatisch oder über die Bausteine der Achskontrolle erfolgen. Grundsätzlich werden Bewegungsaufgaben sequenziell abgearbeitet. Mit folgenden Funktionsbausteinen können Sie den Status abfragen:

- ↬ *Kap. 15.8.3.11 "FB 812 - MC_ReadStatus - PLCopen Status" Seite 776*
- Parameter *PLCopenState* von ↬ *Kap. 15.8.2.2 "FB 860 - VMC_AxisControl - Control-Baustein Achskontrolle" Seite 754*



- - - → Rücksprung wenn fertig
(1) Aus jedem Status: An der Achse ist ein Fehler aufgetreten
(2) Aus jedem Status: MC_Power.Enable = FALSE und es gibt keinen Fehler an der Achse
(3) MC_Reset und MC_Power.Status = FALSE

(4) MC_Reset und MC_Power.Status = TRUE und MC_Power.Enable = TRUE
(5) MC_Power.Enable = TRUE und MC_Power.Status = TRUE
(6) MC_Stop.Done = TRUE und MC_Stop.Execute = FALSE

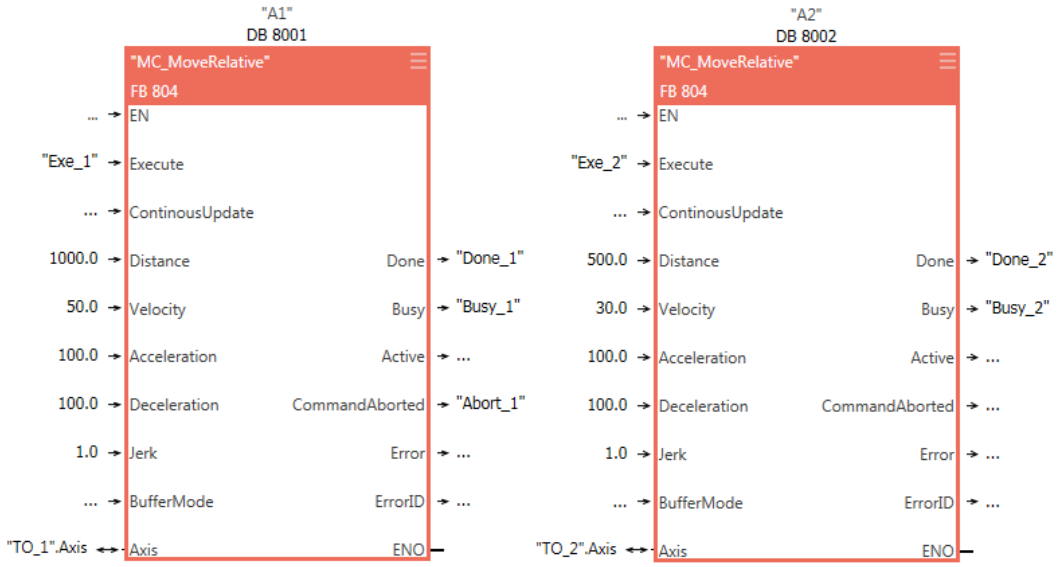
Es gibt folgende Zustände

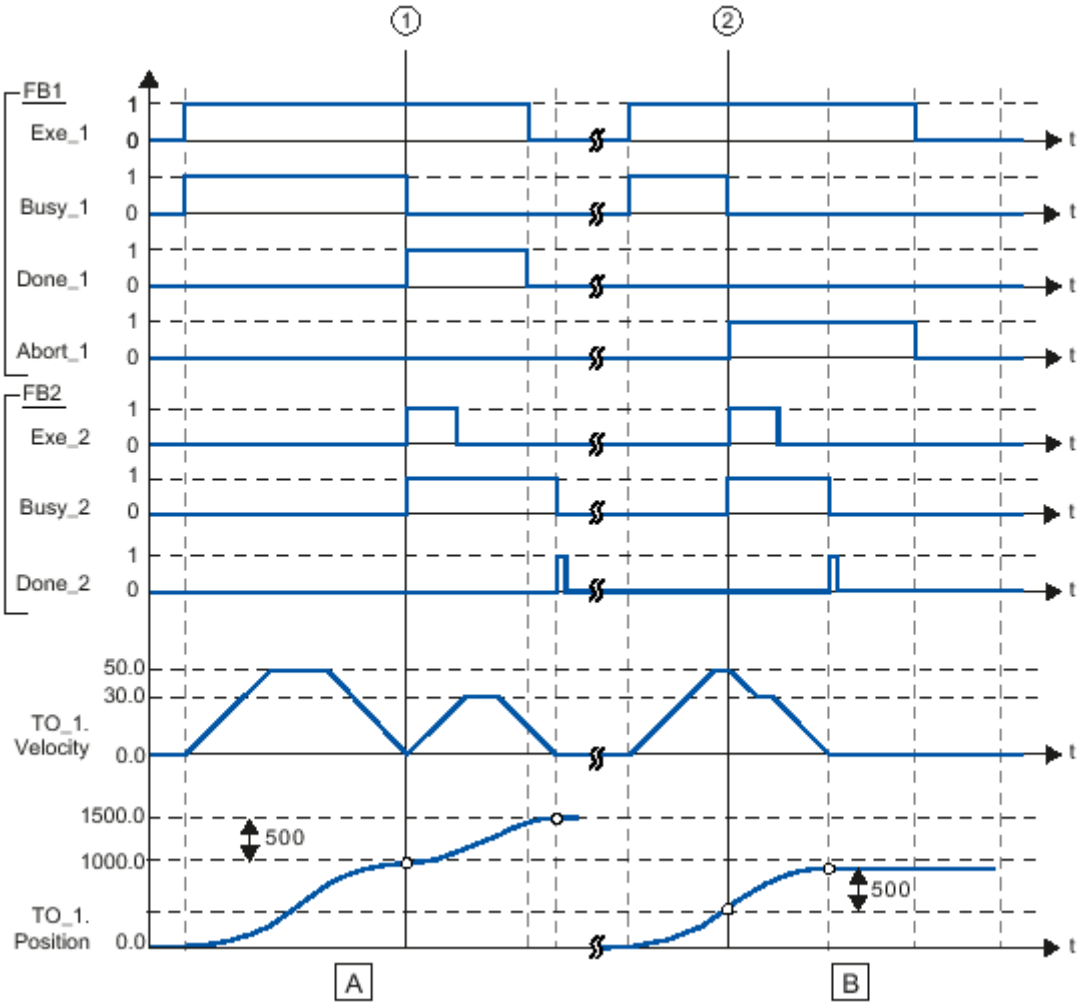
- Disabled
  - Grundzustand einer Achse.
  - Achse kann durch keinen Funktionsbaustein bewegt werden.
- Error Stop
  - Es ist ein Fehler an der Achse aufgetreten.
  - Achse wird gestoppt und ist für weitere Bewegungsaufgaben gesperrt.
  - Achse bleibt in diesem Zustand bis der Fehler behoben ist und ein RESET ausgelöst wird.
  - Fehler an einer Achse werden auch über den entsprechenden Funktionsbaustein zurück gemeldet.
  - Fehler an einem Funktionsbaustein führen nicht in diesen Zustand
- Stand Still
  - Bereit für Bewegungsaufgaben
  - Es liegt kein Fehler an der Achse vor
  - Es sind keine Bewegungsaufgaben an der Achse aktiv
  - Achse wird mit Spannung versorgt
- Stopping
  - Achse wird aktuell gestoppt:
    Kap. 15.8.3.5 "FB 802 - MC_Stop - Achse stoppen" Seite 763
    Kap. 15.8.2.2 "FB 860 - VMC_AxisControl - Control-Baustein Achskontrolle" Seite 754
  - Der Zustand *Stopping* ist aktiv solange ein Stop Kommando vom aktiv ist (*Execute* = 1). Auch wenn die Achse schon gestoppt ist. Danach wechselt der Zustand automatisch nach *Standstill*.
- Homing
  - Die Achse führt aktuell eine Referenzfahrt durch:
    Kap. 15.8.3.4 "FB 801 - MC_Home - Achse referenzieren" Seite 761
    Kap. 15.8.2.2 "FB 860 - VMC_AxisControl - Control-Baustein Achskontrolle" Seite 754
  - Sobald die Achse referenziert ist, wechselt der Zustand automatisch nach *Standstill*.
- Discrete Motion
  - Die Achse führt aktuell eine Bewegungsaufgabe durch:
    Kap. 15.8.3.9 "FB 808 - MC_MoveAbsolute - Achse auf absolute Position verfahren" Seite 771
    Kap. 15.8.3.7 "FB 804 - MC_MoveRelative - Achse relativ verfahren" Seite 767
    Kap. 15.8.3.6 "FB 803 - MC_Halt - Achse anhalten" Seite 765
    Kap. 15.8.2.2 "FB 860 - VMC_AxisControl - Control-Baustein Achskontrolle" Seite 754
  - Sobald das Ziel der Bewegungsaufgabe erreicht ist, wechselt der Zustand automatisch nach *Standstill*.
- Continuous Motion
  - Die Achse führt eine dauerhafte Bewegungsaufgabe durch:
    Kap. 15.8.3.8 "FB 805 - MC_MoveVelocity - Achse verfahren mit konstanter Geschwindigkeit" Seite 769
    Kap. 15.8.2.2 "FB 860 - VMC_AxisControl - Control-Baustein Achskontrolle" Seite 754

15.10.2 Ablöseverhalten von Bewegungsaufträgen

Beispiel

Das Ablöseverhalten von Bewegungsaufträgen wird nachfolgend am Beispiel von MC_MoveRelative erklärt. ↳ *Kap. 15.8.3.7 "FB 804 - MC_MoveRelative - Achse relativ verfahren" Seite 767*





(A) Die Achse wird durch einen "MC_MoveRelative"-Auftrag (A1) um die Wegstrecke *Distance* 1000.0 verfahren (Startposition ist hier die Position 0.0).

(1) Das Erreichen der Zielposition wird zum Zeitpunkt (1) über *Done_1* gemeldet. Zu diesem Zeitpunkt (1) wird ein weiterer MC_MoveRelative-Auftrag (A2) mit der Wegstrecke 500.0 gestartet. Das erfolgreiche Erreichen der neuen Zielposition wird über *Done_2* gemeldet. Da *Exe_2* vorher zurückgesetzt wurde, steht *Done_2* nur für einen Zyklus an.

(B) Ein laufender MC_MoveRelative-Auftrag (A1) wird durch einen weiteren MC_MoveRelative-Auftrag (A2) abgelöst.

(2) Der Abbruch wird zum Zeitpunkt (2) über *Abort_1* gemeldet. Die Achse wird anschließend mit der neuen Geschwindigkeit um die Wegstrecke *Distance* 500.0 verfahren. Das Erreichen der neuen Zielposition wird über *Done_2* gemeldet.

15.10.3 Verhalten der Ein- und Ausgänge

Ausschließlichkeit der Ausgänge

- Die Ausgänge *Busy*, *Done*, *Error* und *CommandAborted* schließen sich gegenseitig aus, es kann also an einem Funktionsbaustein nur einer dieser Ausgänge zu einer Zeit TRUE sein.
- Sobald der Eingang *Execute* TRUE wird, muss einer der Ausgänge TRUE werden. Ebenfalls kann nur einer der Ausgänge *Active*, *Error*, *Done* und *CommandAborted* zu einer Zeit TRUE sein.

Ausgangs-Zustand

- Die Ausgänge *Done*, *InVelocity*, *Error*, *ErrorID* und *CommandAborted* werden mit einer Flanke 1-0 am Eingang *Execute* zurückgesetzt, wenn der Funktionsbaustein nicht aktiv ist (*Busy* = FALSE).
- Die Kommandoausführung wird durch eine Flanke 1-0 an *Execute* nicht beeinflusst.
- Falls *Execute* bereits während der Kommandoausführung zurückgesetzt wird, so ist sichergestellt, dass einer der Ausgänge am Ende des Kommandos für einen SPS-Zyklus gesetzt wird. Erst danach werden die Ausgänge zurückgesetzt.

Eingangs-Parameter

- Die Eingangs-Parameter werden mit Flanke 0-1 an *Execute* übernommen.
- Zur Änderung der Parameter muss das Kommando neu getriggert werden.
- Falls ein Eingangs-Parameter nicht an den Funktionsbaustein übergeben wird, so bleibt der zuletzt an diesen Baustein übergebene Wert gültig.
- Beim ersten Aufruf muss ein sinnvoller Default-Wert übergeben werden.

Position und Distanz

- Der Eingang *Position* bezeichnet einen absoluten Positionswert.
- *Distance* bezeichnet ein relatives Maß als Abstand zweier Positionen.
- Sowohl *Position*, als auch *Distance* werden in technischen Einheiten, z.B. [mm] oder [°], entsprechend der Skalierung der Achse angegeben.

Parameter für das dynamische Verhalten

- Die Dynamikparameter für *Move*-Funktionen werden in technischen Einheiten mit der Zeitbasis Sekunde angegeben.
  Ist eine Achse beispielsweise in Millimetern skaliert, so sind die Einheiten für *Velocity* [mm/s], *Acceleration* [mm/s^2], und *Deceleration* [mm/s^2].

Fehlerbehandlung

- Alle Funktionsbausteine haben zwei Fehlerausgänge um Fehler während der Kommandoausführung anzuzeigen.
- *Error* zeigt den Fehler an und *ErrorID* gibt eine ergänzende Fehlernummer aus.
- Die Ausgänge *Done* und*InVelocity*, bezeichnen eine erfolgreiche Kommandoausführung und werden nicht gesetzt, wenn *Error* TRUE wird.

Fehlertypen

- Funktionsbausteinfehler
  - Funktionsbausteinfehler sind Fehler, die ausschließlich den Funktionsbaustein und nicht die Achse betreffen wie z.B. fehlerhafte Parametrierung.
  - Funktionsbausteinfehler müssen nicht explizit zurückgesetzt werden, sondern werden selbständig zurückgesetzt, wenn der Eingang *Execute* zurückgesetzt wird.
- Kommunikationsfehler
  - Kommunikationsfehler wie z.B. der Funktionsbaustein kann die Achse nicht adressieren.
  - Kommunikationsfehler deuten oft auf eine fehlerhafte Konfiguration oder Parametrierung hin.
  - Ein Reset ist nicht möglich, sondern der Funktionsbaustein kann neu getriggert werden, nachdem die Konfiguration korrigiert wurde.
- Achsfehler
  - Achsfehler treten üblicherweise während der Fahrt auf wie z.B. Schleppabstandsfehler.
  - Ein Achsfehler muss durch MC_Reset zurückgesetzt werden.

Verhalten des *Done*-Ausgangs

- Der *Done*-Ausgang wird gesetzt, wenn ein Kommando erfolgreich ausgeführt wurde.
- Wenn mit mehreren Funktionsbausteinen an einer Achse gearbeitet wird und das laufende Kommando durch einen weiteren Baustein unterbrochen wird, so wird der *Done*-Ausgang des ersten Bausteins nicht gesetzt.

Verhalten des *CommandAborted*-Ausgangs

- *CommandAborted* wird gesetzt, wenn ein Kommando durch einen anderen Baustein unterbrochen wird.

Verhalten des *Busy*-Ausgangs

- Der *Busy*-Ausgang zeigt an, dass der Funktionsbaustein aktiv ist.
- *Busy* wird sofort mit Flanke 0-1 an *Execute* gesetzt und wird erst zurückgesetzt, wenn das Kommando erfolgreich oder auch nicht erfolgreich beendet wurde.
- Solange *Busy* TRUE ist, muss der Funktionsbaustein zyklisch aufgerufen werden, um das Kommando ausführen zu können.

Verhalten des *Active*-Ausgangs

- Wenn die Bewegung einer Achse durch mehrere Funktionsbausteine gesteuert wird, so zeigt der *Active*-Ausgang jedes Bausteins an, dass das Kommando durch die Achse ausgeführt wird.

***Enable*-Eingang und *Valid*-Ausgang**

- Im Gegensatz zu *Execute* führt der *Enable*-Eingang dazu, dass eine Aktion permanent und wiederholt ausgeführt wird, solange *Enable* TRUE ist. MC_ReadStatus aktualisiert beispielsweise zyklisch den Zustand einer Achse solange *Enable* TRUE ist.
- Ein Funktionsbaustein mit einem *Enable*-Eingang zeigt durch den *Valid*-Ausgang an, dass die an den Ausgängen angezeigten Daten gültig sind. Die Daten können jedoch ständig aktualisiert werden während *Valid* TRUE ist.

BufferMode

- *BufferMode* wird nicht unterstützt.

15.11 ErrorID - Zusätzliche Fehlerinformationen

ErrorID	Beschreibung	Bemerkung
0x0000	Kein Fehler	
0x8y24	Fehler in Baustein-Parameter y, mit y: ■ 1: Fehler in PROTOKOLL ■ 2: Fehler in PARAMETER ■ 3: Fehler in BAUDRATE ■ 4: Fehler in CHARLENGTH ■ 5: Fehler in PARITY ■ 6: Fehler in STOPBITS ■ 7: Fehler in FLOWCONTROL (Parameter fehlt)	VMC_ConfigMaster_RTU
0x8001	Unzulässiger Wert beim Parameter *Position*.	
0x8002	Unzulässiger Wert beim Parameter *Distance*.	
0x8003	Unzulässiger Wert beim Parameter *Velocity*.	
0x8004	Unzulässiger Wert beim Parameter *Acceleration*.	
0x8005	Unzulässiger Wert beim Parameter *Deceleration*.	
0x8007	Unzulässiger Wert beim Parameter *ContinuousUpdate*.	
0x8008	Unzulässiger Wert beim Parameter *BufferMode*.	
0x8009	Unzulässiger Wert beim Parameter *EnablePositive*.	
0x800A	Unzulässiger Wert beim Parameter *EnableNegative*.	
0x800B	Unzulässiger Wert beim Parameter *MasterOffset*.	
0x800C	Unzulässiger Wert beim Parameter *SlaveOffset*.	
0x800D	Unzulässiger Wert beim Parameter *MasterScaling*.	
0x800E	Unzulässiger Wert beim Parameter *SlaveScaling*.	
0x800F	Unzulässiger Wert beim Parameter *StartMode*.	
0x8010	Unzulässiger Wert beim Parameter *ActivationMode*.	
0x8011	Unzulässiger Wert beim Parameter *Source*.	
0x8012	Unzulässiger Wert beim Parameter *Direction*.	
0x8014	Unzulässiger Parameter der physikalischen Achse.	MC_ReadParameter
0x8015	Unzulässiger Index oder Subindex.	MC_ReadParameter
0x8016	Unzulässige Parameterlänge.	MC_ReadParameter
0x8017	Unzulässige LADDR.	MC_ReadParameter
0x8018	Unzulässiger Wert beim Parameter *RatioDenominator*.	MC_GearIn
0x8019	Unzulässiger Wert beim Parameter *RatioNumerator*.	MC_GearIn
0x801A	Parameternummer nicht bekannt.	MC_ReadParameter, MC_Write-Parameter
0x801B	Parameter kann nicht geschrieben werden, Parameter ist schreib-geschützt.	MC_WriteParameter
0x801C	Parameter Kommunikation mit unbekanntem Mode.	MC_Home, MC_WriteParameter

ErrorID	Beschreibung	Bemerkung
0x801D	Parameterkommunikation mit allgemeinem Fehler. Die Fehlerursache ist nicht näher beschrieben.	
0x801E	SDO-Parameterwert außerhalb des zulässigen Bereichs.	MC_Home, MC_WriteParameter
0x801F	Der Typ in ANY ist nicht BYTE.	Parameter lesen/schreiben
0x8020	Unterschiedliche Konfiguration der Anwendereinheiten in Cam und Master-Achse.	
0x8021	Unterschiedliche Konfiguration der Anwendereinheiten in Cam und Slave-Achse.	
0x8022	Auf der über LADDR angegebenen logischen Adresse gibt es kein PROFIBUS/PROFINET-Device, von dem konsistente Daten gelesen werden können.	Parameter lesen/schreiben
0x8023	Es wurde ein Zugriffsfehler beim Zugriff auf ein I/O-Geräte erkannt.	Parameter lesen/schreiben
0x8024	Slave-Fehler an externem DP-Slave.	Parameter lesen/schreiben
0x8025	Systemfehler an externem DP-Slave.	Parameter lesen/schreiben
0x8026	Systemfehler an externem DP-Slave.	Parameter lesen/schreiben
0x8027	Die Daten wurden noch nicht vom Modul gelesen.	Parameter lesen/schreiben
0x8028	Systemfehler an externem DP-Slave.	Parameter lesen/schreiben
0x8029	Schreibversuch auf eine Objekt, welches nur gelesen werden kann.	Parameter lesen/schreiben
0x802A	Leseversuch auf ein Objekt, welches nur geschrieben werden kann.	Parameter lesen/schreiben
0x802B	Nicht unterstützte Zugriff auf ein Objekt.	Parameter lesen/schreiben
0x802C	Falscher Datentyp.	Parameter lesen/schreiben
0x802D	Fehler im Geräteprofil.	Parameter lesen/schreiben
0x802E	Fehler Kommando-Typ.	Parameter lesen/schreiben
0x802F	Keine Systemressourcen verfügbar.	Parameter lesen/schreiben
0x8030	Unzulässiger Wert beim Parameter *Hardware* (1 = SLIO CP; 2 = VIPA CPU).	Modbus; Init
0x8031	Unzulässiger Wert beim Parameter *UnitId*.	Modbus; Init
0x8032	Unzulässiger Wert beim Parameter *UserUnitsVelocity* (0 = Hz, 1 = %, 2 = U/min).	Modbus; Init
0x8033	Unzulässiger Wert beim Parameter *UserUnitsAcceleration* (0 = 0.00s, 1 = 0.0s).	Modbus; Init
0x8034	Unzulässiger Wert beim Parameter *MaxVelocityApp* (muss > 0 sein).	Modbus; Init
0x8035	Fehler beim Lesezugriff auf *MonitorData*.	Modbus; Init
0x8036	Fehler beim Lesezugriff auf *NumberOfPoles*.	Modbus; Init
0x8037	Fehler beim Schreibzugriff auf *UserUnitsVelocity*.	Modbus; Init
0x8038	Fehler beim Lesezugriff auf *MinOutputFrequency*.	Modbus; Init
0x8039	Fehler beim Lesezugriff auf *MaxOutputFrequency*.	Modbus; Init
0x803A	Fehler beim Schreibzugriff auf *StoppingMethodSelection*.	Modbus; Init
0x803B	Fehler beim Schreibzugriff auf *UserUnitsAcceleration*.	Modbus; Init

ErrorID	Beschreibung	Bemerkung
0x8041	Unzulässiger Wert beim Parameter *AccelerationTime*.	Modbus V1000
0x8042	Unzulässiger Wert beim Parameter *DecelerationTime*.	Modbus V1000
0x8043	Unzulässiger Wert beim Parameter *JogAccelerationTime*.	Modbus V1000
0x8044	Unzulässiger Wert beim Parameter *JogDecelerationTime*.	Modbus V1000
0x8045	Unzulässiger Wert beim Parameter *JogVelocity* (≤ *MaxVelocityApp*).	Modbus V1000
0x80C8	Modbus Übertragungsfehler: Keine Antwort des Servers im definierten Zeitraum (Timeout über Schnittstelle parametrierbar).	Modbus V1000
0x809y	Fehler in Wert des Baustein-Parameter y, mit y: ■ 1: Fehler in PROTOKOLL ■ 3: Fehler in BAUDRATE ■ 4: Fehler in CHARLENGTH ■ 5: Fehler in PARITY ■ 6: Fehler in STOPBITS	VMC_ConfigMaster_RTU
0x8092	Zugriffsfehler auf Parameter-DB (DB zu kurz).	VMC_ConfigMaster_RTU
0x809A	Schnittstelle nicht vorhanden bzw. wird unter PROFIBUS betrieben.	VMC_ConfigMaster_RTU
0x8101	Keine zyklische Kommunikation mit der Achse möglich.	
0x8102	Befehl ist im aktuellen PLCopen-State nicht zulässig.	
0x8103	Befehl wird von der Achse nicht unterstützt.	
0x8104	Achse ist nicht einschaltbereit, mögliche Gründe: ■ Kommunikation zur Achse nicht bereit. ■ Antrieb ist nicht im Zustand *"eingeschaltet"* → Antriebsfehler evtl. mit MC_Reset zurücksetzen. ■ Kommunikation wurde unterbrochen, z.B. durch Aus- Einschalten der CPU. Fehler mit MC_Reset zurücksetzen.	*PreOperational* muss auch im Status *Operational* gesetzt werden.
0x8105	Kommando wird von virtuellen Achsen nicht unterstützt.	
0x8106	PLCopen-State ist nicht definiert.	
0x8107	Befehl ist bei deaktiviertem Antrieb nicht zulässig.	VMC_AxisControl_PT, ModbusV1000
0x8188	Modbus Übertragungsfehler: Interner Fehler MB_FUNCTION ungültig.	Modbus V1000
0x8189	Modbus Übertragungsfehler: Interner Fehler MB_DATA_ADDR ungültig.	Modbus V1000
0x818A	Modbus Übertragungsfehler: Interner Fehler MB_DATA_LEN ungültig.	Modbus V1000
0x818B	Modbus Übertragungsfehler: Interner Fehler MB_DATA_PTR ungültig.	Modbus V1000
0x8201	Wegen Mangels an internen Ressourcen kann der Befehl aktuell nicht ausgeführt werden (kein freier Slot im CommandBuffer).	
0x8202	Fehler beim Schreiben des Offsets für Referenzfahrt (kein freier Slot im CommandBuffer).	DriveManager → Referenzfahrt (aktives Kommando)

ErrorID	Beschreibung	Bemerkung
0x8210	Modbus Übertragungsfehler: Die Hardware ist inkompatibel zur Baustein-Bibliothek Modbus RTU/TCP.	Modbus V1000
0x828y	Fehler in Parameter y von DB-Parameter, mit y: ■ 1: Fehler im 1. Parameter ■ 2: Fehler im 2. Parameter ■ ...	VMC_ConfigMaster_RTU
0x8301	Keine zyklische Kommunikation mit der Master-Achse möglich.	
0x8302	Befehl ist im aktuellen PLCopen-State der Master-Achse nicht zulässig.	
0x8303	Befehl wird von der Master-Achse nicht unterstützt.	
0x8304	Master-Achse befindet sich nicht im Status *Pre-Operational*.	
0x8305	Die Nummer des Datenbausteins der Master-Achse hat sich geändert.	
0x8306	Kommunikationsfehler an der Master Achse. Slave Achse wird mit Schnellhalt gestoppt.	
0x8311	Keine zyklische Kommunikation mit der Slave-Achse möglich.	
0x8312	Befehl ist im aktuellen PLCopen-State der Slave-Achse nicht zulässig.	
0x8313	Befehl wird von der Slave-Achse nicht unterstützt.	
0x8314	Slave-Achse befindet sich nicht im Status *Pre-Operational*.	
0x8315	Die Nummer des Datenbausteins der Slave-Achse hat sich geändert.	
0x8317	Baustein wurde nicht innerhalb des OB 1 aufgerufen	VMC_AxisControl_PT
0x8321	Koppeln mit *StartMode* = relative und *ActivationMode* = nextcycle ist nicht zulässig.	
0x8322	Koppeln oder schalten mit *StartMode* = absolute und *ActivationMode* = nextcycle ist nicht zulässig.	
0x8323	Schalten mit einem unterschiedlichen *StartMode* (*StartMode* der Kopplung ist zu verwenden).	
0x8331	MC_CamIn ist nicht aktiv.	
0x8332	MC_GearIn ist nicht aktiv.	
0x8340	Ungültiger Wert in TriggerInput.Probe.	MC_TouchProbe und MC_AbortTrigger
0x8341	Ungültiger Wert in TriggerInput.Source.	MC_TouchProbe und MC_AbortTrigger
0x8342	Ungültiger Wert in TriggerInput.TriggerMode.	MC_TouchProbe und MC_AbortTrigger
0x8350	Ungültiger Wert in VelocitySearchSwitch.	Referenzfahrt, Initialisierung
0x8351	Ungültiger Wert in VelocitySearchZero.	Referenzfahrt, Initialisierung
0x8352	Ungültige Kombination von Eingängen.	Referenzfahrt, Initialisierung

ErrorID	Beschreibung	Bemerkung
0x8360	Die CPU unterstützt kein Pulse Train.	VMC_AxisControl_PT
0x8361	Falscher Wert in *S_ChannelNumberPWM*.	VMC_AxisControl_PT
0x8362	Allgemeiner Fehler bei der Pulse Train Ausgabe.	VMC_AxisControl_PT
0x8363	Fahr-Kommando erhalten bei gesetztem *StopExecute*.	VMC_AxisControl_PT, ModbusV1000
0x8381	Modbus Übertragungsfehler: Server liefert Exception-Code 01h.	Modbus V1000
0x8382	Modbus Übertragungsfehler: Server liefert Exception-Code 03h oder falsche Startadresse.	Modbus V1000
0x8383	Modbus Übertragungsfehler: Server liefert Exception-Code 02h.	Modbus V1000
0x8384	Modbus Übertragungsfehler: Server liefert Exception-Code 04h.	Modbus V1000
0x8386	Modbus Übertragungsfehler: Server liefert falschen Funktions-Code.	Modbus V1000
0x8388	Modbus Übertragungsfehler: Server liefert falschen Wert oder falsche Anzahl.	Modbus V1000
0x8400	MC_Power: Unerwarteter Drive-State Drive-State <> Operation enabled	MC_Power
0x8401	MC_Power: Unerwarteter Drive-State Drive-State = Quick stop active	MC_Power
0x8402	MC_Power: Unerwarteter Drive-State Drive-State = Fault reaction active	MC_Power
0x8403	MC_Power: Unerwarteter Drive-State Drive-State = Fault	MC_Power
0x8410	Zeitüberschreitung beim Versuch den Antrieb zurückzusetzen.	Kernel FB --> MC_Reset
0x8500	Falscher Wert in *EncoderType* (1 oder 2).	Init-Baustein
0x8501	Falscher Wert in *EncoderResolutionBits* (>0 und ≤32).	Init-Baustein
0x8502	Falscher Wert in *LogicalAddress* (≥0).	Init-Baustein
0x8503	Falscher Wert in *StartInputAddress* (≥0).	Init-Baustein
0x8504	Falscher Wert in *StartOutputAddress* (≥0).	Init-Baustein
0x8505	Falscher Wert in *FactorPosition* (>0.0).	Init-Baustein
0x8506	Falscher Wert in *FactorVelocity* (>0.0).	Init-Baustein
0x8507	Falscher Wert in *FactorAcceleration* (>0.0).	Init-Baustein
0x8508	Falscher Wert in *MaxVelocityApp* (>0.0).	Init-Baustein
0x8509	Falscher Wert in *MaxAccelerationApp* (>0.0).	Init-Baustein
0x850A	Falscher Wert in *MaxDecelerationApp* (>0.0).	Init-Baustein
0x850B	Falscher Wert in *MaxVelocityDrive* (>0.0).	Init-Baustein
0x850C	Falscher Wert in *MaxAccelerationDrive* (>0.0).	Init-Baustein
0x850D	Falscher Wert in *MaxDecelerationDrive* (>0.0).	Init-Baustein
0x850E	Falscher Wert in *MinPosition* (≥MinUserPos).	Init-Baustein

ErrorID	Beschreibung	Bemerkung
0x850F	Falscher Wert in *MaxPosition* (≥MaxUserPos).	Init-Baustein
0x8510	Falscher Wert in *M2_EncoderType*.	VMC_InitSigma7W_EC
0x8511	Falscher Wert in *M2_EncoderResolutionBits*.	VMC_InitSigma7W_EC
0x8513	Falscher Wert in *M2_PdoInputs*.	VMC_InitSigma7W_EC
0x8514	Falscher Wert in *M2_PdoOutputs*.	VMC_InitSigma7W_EC
0x8515	Falscher Wert in *M2_FactorPosition*.	VMC_InitSigma7W_EC
0x8516	Falscher Wert in *M2_FactorVelocity*.	VMC_InitSigma7W_EC
0x8517	Falscher Wert in *M2_FactorAcceleration*.	VMC_InitSigma7W_EC
0x8518	Falscher Wert in *M2_MaxVelocityApp*.	VMC_InitSigma7W_EC
0x8519	Falscher Wert in *M2_MaxAccelerationApp*.	VMC_InitSigma7W_EC
0x851A	Falscher Wert in *M2_MaxDecelerationApp*.	VMC_InitSigma7W_EC
0x851D	Falscher Wert in *ParaAccessPointAddress*.	VMC_InitSigma_PN
0x8603	Fehler Referenzfahrt am Antrieb, Geschwindigkeit <> 0.	MC_Home
0x8604	Fehler Referenzfahrt am Antrieb, Geschwindigkeit = 0.	MC_Home
0x8700	Fehler: Unzulässige Größe.	
0x8710	SDO-Fehler: Toggle-Bit hat nicht gewechselt.	
0x8711	SDO-Fehler: SDO-Protokoll Timeout.	
0x8712	SDO-Fehler: Client/Server-Kommando nicht gültig oder unbekannt.	
0x8713	SDO-Fehler: Unzulässige Blockgröße (nur im Blockmodus).	
0x8714	SDO-Fehler: Unzulässige Sequenznummer (nur im Blockmodus).	
0x8715	SDO-Fehler: CRC-Fehler (nur im Block-Modus).	
0x8716	SDO-Fehler: Nicht genügend Arbeitsspeicher.	
0x8717	SDO-Fehler: Nicht unterstützter Zugriff auf ein Objekt.	
0x8718	SDO-Fehler: Leseversuch auf ein Objekt, welches nur geschrieben werden kann.	
0x8719	SDO-Fehler: Schreibversuch auf ein Objekt, welches nur gelesen werden kann.	
0x871A	SDO-Fehler: Objekt existiert nicht im Objektverzeichnis.	
0x871B	SDO-Fehler: Objekt kann nicht auf ein PDO gemappt werden.	
0x871C	SDO-Fehler: Anzahl und Länge der zu mappenden Objekte übersteigt die PDO-Länge.	
0x871D	SDO-Fehler: Allgemeine Parameter-Inkompatibilität.	
0x871E	SDO-Fehler: Allgemeine interne Inkompatibilität im Gerät.	
0x871F	SDO-Fehler: Zugriff fehlgeschlagen aufgrund eines Hardwarefehlers.	
0x8720	SDO-Fehler: Datentyp passt nicht, Länge Service- Parameter passt nicht.	
0x8721	SDO-Fehler: Datentyp passt nicht, Service-Parameter ist zu lang.	

ErrorID	Beschreibung	Bemerkung
0x8722	SDO-Fehler: Datentyp passt nicht, Service-Parameter ist zu kurz.	
0x8723	SDO-Fehler: Es existiert kein Subindex.	
0x8724	SDO-Fehler: Schreibzugriff - Parameterwert liegt außerhalb des zulässigen Bereichs.	
0x8725	SDO-Fehler: Schreibzugriff - Parameterwert überschreitet den zulässigen Bereich.	
0x8726	SDO-Fehler: Schreibzugriff - Parameterwert unterschreitet den zulässigen Bereich.	
0x8727	SDO-Fehler: Maximaler Wert < Minimaler Wert.	
0x8728	SDO-Fehler: Allgemeiner Fehler.	
0x8729	SDO-Fehler: Daten können nicht in die Anwendung übertragen oder dort gespeichert werden.	
0x872A	SDO-Fehler: Daten können nicht in die Anwendung übertragen oder dort gespeichert werden, da die lokale Steuerung aktiviert ist.	
0x872B	SDO-Fehler: Aufgrund des aktuellen Gerätezustands können keine Daten in die Anwendung übertragen oder dort gespeichert werden.	
0x872C	SDO-Fehler: Die dynamische Generierung des Objektverzeichnisses konnte nicht durchgeführt werden bzw. das Objektverzeichnis ist nicht vorhanden.	
0x872D	SDO-Fehler: Unbekannter Code.	
0x8750	Falscher Wert in *LADDR*.	
0x8751	Typ in ANY-Pointer weicht von BYTE ab.	
0x8752	Auf der über *LADDR* spezifizierten Adresse gibt es kein PROFIBUS-DP-Modul bzw. PROFINET-IO-Device, von dem konsistente Daten gelesen werden können.	
0x8753	Zugriffsfehler beim Zugriff auf ein PROFINET-IO-Device.	
0x8754	Slave-Fehler am externen PROFIBUS-DP-Slave.	
0x8755	Länge der SFB-Daten passt nicht zur Länge der Benutzerdaten.	
0x8756	Fehler auf externem PROFIBUS-DP-Slave.	
0x8757	Systemfehler auf externem PROFIBUS-DP-Slave.	
0x8758	Die Daten wurden vom Gerät noch nicht gelesen.	
0x8759	Systemfehler auf externem PROFIBUS-DP-Slave.	
0x875A	Es sind keine Systemressourcen verfügbar.	
0x8799	SDO-Fehler: Es ist ein anderer Fehler aufgetreten, nähere Informationen finden Sie in den Daten von *Info1* und *Info2*.	
0x8888	Intern: BufferIndex-Fehler	VMC_AxisControl
0x8A00	Zugriff auf einen nicht vorhandenen Parameter.	VMC_AxisControlSigma_PN
0x8A01	Zugriff auf einen Parameter, welcher nicht geändert werden kann.	VMC_AxisControlSigma_PN
0x8A02	Zugriff mit einem Wert, welcher außerhalb des Wertebereichs liegt.	VMC_AxisControlSigma_PN
0x8A03	Zugriff auf einen nicht vorhandenen Subindex.	VMC_AxisControlSigma_PN
0x8A04	Zugriff über Subindex auf einen nicht indizierten Parameter.	VMC_AxisControlSigma_PN

ErrorID	Beschreibung	Bemerkung
0x8A05	Unzulässiger Datentyp	VMC_AxisControlSigma_PN
0x8A06	Zugriff mit einem Wert ≠ 0, sofern dies nicht zulässig ist.	VMC_AxisControlSigma_PN
0x8A07	Zugriff auf eine beschreibendes Element, welches nicht geändert werden kann.	VMC_AxisControlSigma_PN
0x8A09	Zugriff auf Beschreibungsdaten, welche nicht vorhanden sind.	VMC_AxisControlSigma_PN
0x8A0B	Zugriff ohne das Recht Parameter zu ändern.	VMC_AxisControlSigma_PN
0x8A0F	Zugriff auf Textfeld, welches nicht verfügbar ist.	VMC_AxisControlSigma_PN
0x8A11	Zugriff ist aktuell nicht möglich.	VMC_AxisControlSigma_PN
0x8A14	Zugriff mit einem Wert, der innerhalb der Grenzen liegt, jedoch aktuell nicht möglich ist.	VMC_AxisControlSigma_PN
0x8A15	Die Länge der aktuellen Antwort überschreitet die maximal mögliche Länge.	VMC_AxisControlSigma_PN
0x8A16	Unzulässiger Wert bzw. Wert wird vom Paramater-Typ nicht unterstützt.	VMC_AxisControlSigma_PN
0x8A17	Fehler beim Schreibzugriff auf Parameter: Unzulässiges Format	VMC_AxisControlSigma_PN
0x8A18	Fehler beim Schreibzugriff auf Parameter: Parameteranzahl stimmt nicht mit der Anzahl der Elemente auf der Parameter-Adresse überein.	VMC_AxisControlSigma_PN
0x8A19	Fehler beim Schreibzugriff auf einen digitalen Ausgang, welcher nicht existiert.	VMC_AxisControlSigma_PN
0x8A20	Schreibzugriff auf einen Parametertext, welcher nicht geändert werden kann.	VMC_AxisControlSigma_PN
0x8A21	Unzulässige Auftrags-ID	VMC_AxisControlSigma_PN
0x8A22	Maximale Anzahl von Parameterabrufen ist erreicht	VMC_AxisControlSigma_PN
0xC000	Interner Fehler: Status Init ist undefiniert.	Modbus; Init
0xC001	Interner Fehler: Unzulässiger Wert beim Parameter *Cmd.ActiveType*.	Modbus V1000
0xC002	Internal Error: Unzulässiger Wert beim Parameter *Cmd.State*.	Modbus V1000

16 Integrierte Standardfunktionen - *"Integrated Standard"*

16.1 System-SFCs - *"System Functions"*

16.1.1 SFC 0 - SET_CLK - Uhrzeit stellen

Beschreibung

Mit dem SFC 0 SET_CLK (set system clock) werden Uhrzeit und Datum der CPU-Uhr gestellt, wobei die Uhr dann ab der eingestellten Uhrzeit und dem eingestellten Datum läuft.

Handelt es sich um eine Master-Uhr, dann wird beim Aufruf des SFC 0 zusätzlich die Synchronisation der Uhrzeit gestartet. Die Synchronisationsintervalle werden in der Hardwarekonfiguration eingestellt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
PDT	INPUT	DT	D, L	Über *PDT* geben Sie das Datum und die Uhrzeit ein, die Sie einstellen möchten.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

PDT

Datum und Uhrzeit sind als Datentyp DT einzugeben.

Beispiel:

Datum: 27.04.2006, Uhrzeit: 14:15:55 → DT#2006-04-27-14:15:55

Die Uhrzeit kann nur sekundengranular eingegeben werden. Der Wochentag wird vom SFC 0 automatisch aus dem Datum errechnet.

Beachten Sie, dass Sie den Datentyp DT mit dem FC 3 D_TOD_DT erst bilden müssen, bevor Sie ihn dem Eingangsparameter übergeben können

(siehe Uhrzeitfunktionen; FC 3, FC 6, FC 7, FC 8, FC 33, FC 40, FC 1, FC 35, FC 34).

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	kein Fehler
8080h	Fehler im Datum
8081h	Fehler in der Uhrzeit

16.1.2 SFC 1 - READ_CLK - Uhrzeit lesen

Beschreibung

Mit dem SFC 1 READ_CLK (read system clock) wird die Uhr in der CPU ausgelesen. Dadurch erhalten Sie das aktuelle Datum und die Uhrzeit.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
CDT	OUTPUT	DT	D, L	Am Ausgang CDT werden das aktuelle Datum und die aktuelle Uhrzeit ausgegeben.

RET_VAL (Rückgabewert) Der SFC 1 liefert keine spezifischen Fehlerinformationen aus.

CDT Am Ausgang CDT wird das aktuelle Datum und die aktuelle Uhrzeit im Format DT ausgegeben.

16.1.3 SFC 2 ... 4 - Betriebsstundenzähler

Beschreibung

VIPA-CPUs verfügen über 8 Betriebsstundenzähler.

Sie können über:

SFC 2	SET_RTM	Betriebsstundenzähler setzen
SFC 3	CTRL_RTM	Betriebsstundenzähler starten/stoppen
SFC 4	READ_RTM	Betriebsstundenzähler auslesen

Über einen Betriebsstundenzähler können Sie:

- die Betriebsdauer der CPU berechnen.
- die Betriebsdauer von angesteuerten Betriebsmitteln berechnen.

Eigenschaften

Mit dem Start beginnt der Betriebsstundenzähler immer ab dem letzten Zählerstand zu zählen. Soll er ab einem anderen Anfangswert beginnen, dann müssen Sie diesen Wert mit dem SFC 2 zuweisen.

Geht die CPU in STOP oder Sie stoppen den Betriebsstundenzähler, dann merkt sich die CPU den aktuellen Wert. Bei Neustart der CPU muss der Betriebsstundenzähler erneut mit dem SFC 3 gestartet werden.

Wertebereich Der Betriebsstundenzähler hat einen Wertebereich von 0 ... 32767 Stunden.

16.1.4 SFC 2 - SET_RTM - Betriebsstundenzähler setzen

Beschreibung

Mit dem SFC 2 SET_RTM (set run-time meter) wird ein Betriebsstundenzähler der CPU auf einen vorgegebenen Wert gestellt. Bei VIPA-CPUs sind maximal 8 Betriebsstundenzähler verfügbar.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
NR	INPUT	BYTE	E, A, M, D, L, Konstante	Eingang *NR* enthält die Nummer des Betriebsstundenzählers, den Sie einstellen möchten. Mögliche Werte: 0 ... 7
PV	INPUT	INT	E, A, M, D, L, Konstante	Eingang *PV* enthält die Einstellung für den Betriebsstundenzähler.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	kein Fehler
8080h	Falsche Nummer des Betriebsstundenzählers
8081h	Ein negativer Wert wurde dem Parameter *PV* übergeben.

16.1.5 SFC 3 - CTRL_RTM - Betriebsstundenzähler starten/stoppen

Beschreibung

Mit dem SFC 3 CTRL_RTM (control run-time meter) wird ein Betriebsstundenzähler in Abhängigkeit des Zustandes von Eingang *S* gestartet oder gestoppt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
NR	INPUT	BYTE	E, A, M, D, L, Konstante	Eingang *NR* enthält die Nummer des Betriebsstundenzählers, den Sie starten bzw. stoppen möchten. Mögliche Werte: 0 ... 7
S	INPUT	BOOL	E, A, M, D, L, Konstante	Eingang *S* startet bzw. stoppt den Betriebsstundenzähler. Setzen Sie den Signalzustand auf "0", wenn Sie den Zähler stoppen möchten. Setzen Sie den Signalzustand auf "1", wenn Sie den Zähler starten möchten.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	kein Fehler
8080h	Falsche Nummer des Betriebsstundenzählers

16.1.6 SFC 4 - READ_RTM - Betriebsstundenzähler auslesen

Beschreibung

Mit dem SFC 4 READ_RTM (read run-time meter) wird ein Betriebsstundenzähler ausgelesen. Als Ausgangsdaten werden die aktuelle Betriebsstundenzahl und der Status des Zählers ("gestoppt" bzw. "zählt") ausgegeben.

Wenn der Betriebsstundenzähler länger als 32767 Stunden läuft, dann bleibt er bei diesem Wert stehen und der Rückgabewert *RET_VAL* enthält die Fehlermeldung "8081h: Überlauf".

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
NR	INPUT	BYTE	E, A, M, D, L, Konstante	Eingang *NR* enthält die Nummer des Betriebsstundenzählers, den Sie auslesen möchten. Mögliche Werte: 0 ... 7
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
CQ	OUTPUT	BOOL	E, A, M, D, L	Ausgang *CQ* gibt an, ob der Betriebsstundenzähler läuft oder angehalten ist. ■ "0": Betriebsstundenzähler gestoppt ■ "1": Betriebsstundenzähler läuft
CV	OUTPUT	INT	E, A, M, D, L	Ausgang *CV* gibt den aktuellen Wert des Betriebsstundenzählers an.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	kein Fehler
8080h	Falsche Nummer des Betriebsstundenzählers
8081h	Überlauf des Betriebsstundenzählers

16.1.7 SFC 5 - GADR_LGC - Logische Adresse eines Kanals ermitteln

Beschreibung

Mit dem SFC 5 GADR_LGC (convert geographical address to logical address) ermitteln Sie die logische Adresse des Moduls.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
SUBNETID	INPUT	BYTE	E, A, M, D, L, Konstante	Bereichskennung
RACK	INPUT	WORD	E, A, M, D, L, Konstante	Nr. des Racks
SLOT	INPUT	WORD	E, A, M, D, L, Konstante	Steckplatz-Nummer
SUBSLOT	INPUT	BYTE	E, A, M, D, L, Konstante	Submodulsteckplatz

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
SUBADDR	INPUT	WORD	E, A, M, D, L, Konstante	Offset im Nutzdatenadressraum des Moduls
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
IOID	OUTPUT	BYTE	E, A, M, D, L	Bereichskennung
LADDR	OUTPUT	WORD	E, A, M, D, L	Logische Basisadresse des Moduls

SUBNETID

Bereichskennung:

- "0": falls das Modul lokal gesteckt ist (inklusive Zeilenanschaltung)
- DP-Mastersystem-ID des zugehörigen dezentralen Peripheriesystems, falls sich der Steckplatz in einem dezentralen Peripheriegerät befindet.

Rack

Nr. des Racks, falls Bereichskennung 0

Stations-Nr. des dezentralen Peripheriegeräts, falls Bereichskennung > 0.

SLOT

Steckplatz-Nummer

SUBSLOT

Submodulsteckplatz

(falls kein Submodul gesteckt werden kann, ist hier 0 anzugeben).

SUBADDR

Offset im Nutzdatenadressraum des Moduls.

RET_VAL (Rückgabewert)

Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Wert	Beschreibung
0000h	kein Fehler
8094h	Es wurde kein Subnetz mit der angegebenen *SUBNETID* konfiguriert.
8095h	Unzulässiger Wert beim Parameter *RACK*
8096h	Unzulässiger Wert beim Parameter *SLOT*
8097h	Unzulässiger Wert beim Parameter *SUBSLOT*
8098h	Unzulässiger Wert beim Parameter *SUBADDR*
8099h	Der Steckplatz ist nicht projektiert.
809Ah	Die Subadresse für den ausgewählten Steckplatz ist nicht projektiert.

IOID

Bereichskennung:

- 54h: Peripherie-Eingang (PE)
- 55h: Peripherie-Ausgang (PA)

Im Fall eines Mischmoduls liefert der SFC die Bereichskennung der niedrigeren Adresse. Bei gleichen Adressen liefert der SFC die Kennung 54h.

LADDR Logische Basisadresse des Moduls.

16.1.8 SFC 6 - RD_SINFO - Startinformation auslesen

Beschreibung

Mit dem SFC 6 RD_SINFO (read start information) werden die Startinformationen des zuletzt aufgerufenen OBs, der noch nicht vollständig abgearbeitet wurde, und des zuletzt gestarteten Anlauf-OBs ausgelesen. Beide Startinformationen enthalten keinen Zeitstempel. Erfolgt der Aufruf im OB 100, dann werden zwei identische Startinformationen zurückgeliefert.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
TOP_SI	OUTPUT	STRUCT	D, L	Startinformation des aktuellen OBs
START_UP_SI	OUTPUT	STRUCT	D, L	Startinformation des zuletzt gestarteten Anlauf-OBs

TOP_SI und START_UP_SI Hierbei handelt es sich um zwei identisch aufgebaute Strukturen, deren Aufbau nachfolgend dargestellt ist.

Strukturelement	Datentyp	Beschreibung
EV_CLASS	BYTE	Bits 3 ... 0: Ereigniskennung Bits 7 ... 4: Ereignisklasse: 1: Startereignisse von Standard-OBs 2: Startereignisse von Synchronfehler-OBs 3: Startereignisse von Asynchronfehler-OBs
EV_NUM	BYTE	Ereignisnummer
PRIORITY	BYTE	Das Strukturelement PRIORITY liefert die zum aktuellen OB gehörige Prioritätsklasse.
NUM	BYTE	OB-Nummer NUM enthält die Nummer des aktuellen OBs bzw. des zuletzt gestarteten Anlauf-OBs.
TYP2_3	BYTE	Datenkennung 2_3: kennzeichnet die in ZI2_3 eingetragene Information
TYP1	BYTE	Datenkennung 1: kennzeichnet die in ZI1 eingetragene Information
ZI1	WORD	Zusatzinformation 1
ZI2_3	DWORD	Zusatzinformation 2_3

Die Strukturelemente entsprechen inhaltlich genau den temporären Variablen eines OBs.

Die temporären Variablen können in den einzelnen OBs andere Namen und andere Datentypen haben. Beachten Sie, dass die Aufrufschnittstelle der OBs zusätzlich Datum und Uhrzeit enthält.

RET_VAL (Rückgabewert)

Der SFC 6 liefert keine spezifischen, sondern nur allgemeine Fehlerinformationen zurück.

Beispiel

Der zuletzt aufgerufene, noch nicht vollständig abgearbeitet OB, ist der OB 80, der zuletzt gestartete Anlauf-OB ist der OB 100.

Die folgende Tabelle zeigt die Zuordnung zwischen den Strukturelementen des Parameters *TOP_SI* des SFC 6 und den lokalen Variablen des OB 80.

TOP_SI Strukturelement	Datentyp	Lokale Variable	Datentyp
EV_CLASS	BYTE	OB100_EV_CLASS	BYTE
EV_NUM	BYTE	OB80_FLT_ID	BYTE
PRIORITY	BYTE	OB80_PRIORITY	BYTE
NUM	BYTE	OB80_OB_NUMBR	BYTE
TYP2_3	BYTE	OB80_RESERVED_1	BYTE
TYP1	BYTE	OB80_RESERVED_2	BYTE
ZI1	WORD	OB80_ERROR_INFO	WORD
ZI2_3	DWORD	OB80_ERR_EV_CLASS	BYTE
		OB80_ERR_EV_NUM	BYTE
		OB80_OB_PRIORITY	BYTE
		OB80_OB_NUM	BYTE

Die folgende Tabelle zeigt die Zuordnung zwischen den Strukturelementen des Parameters *START_UP_SI* des SFC 6 und den lokalen Variablen des OB 100.

START_UP_SI Strukturelement	Datentyp	Lokale Variable	Datentyp
EV_CLASS	BYTE	OB100_EV_CLASS	BYTE
EV_NUM	BYTE	OB100_STRTUP	BYTE
PRIORITY	BYTE	OB100_PRIORITY	BYTE
NUM	BYTE	OB100_OB_NUMBR	BYTE
TYP2_3	BYTE	OB100_RESERVED_1	BYTE
TYP1	BYTE	OB100_RESERVED_2	BYTE
ZI1	WORD	OB100_STOP	WORD
ZI2_3	DWORD	OB100_STRT_INFO	DWORD

16.1.9 SFC 7 - DP_PRAL - Prozessalarm beim DP-Master auslösen

Beschreibung

Mit dem SFC 7 DP_PRAL lösen Sie aus dem Anwenderprogramm eines intelligenten Slaves beim zugehörigen DP-Master einen Prozessalarm aus. Das führt zum Start des OB 40 beim DP-Master. Mit dem Eingangsparameter *AL_INFO* können Sie die Ursache für den von Ihnen gewünschten Prozessalarm kennzeichnen. Diese Alarmkennung wird an den DP-Master übertragen und kann von Ihnen im OB 40 (Variable OB40_POINT_ADDR) ausgewertet werden. Der angeforderte Prozessalarm wird durch die Eingangsparameter *IOID* und *LADDR* eindeutig festgelegt. Für jeden projektierten Adressbereich im Übergabespeicher können Sie zu einem beliebigen Zeitpunkt genau einen Prozessalarm auslösen.

Arbeitsweise

Der SFC 7 DP_PRAL ist ein asynchron arbeitender SFC, d. h. die Bearbeitung erstreckt sich über mehrere SFC-Aufrufe. Sie starten die Prozessalarmanforderung, indem Sie den SFC 7 mit *REQ* = 1 aufrufen. Über die Ausgangsparameter *RET_VAL* und *BUSY* wird der Zustand des Auftrags angezeigt, siehe auch Bedeutung von *REQ, RET_VAL* und *BUSY* bei asynchron arbeitenden SFCs. Der Auftrag ist abgeschlossen, wenn die Bearbeitung des OB 40 im DP-Master beendet ist.

Betreiben Sie den DP-Slave als Normslave, ist der Auftrag abgeschlossen, sobald das Diagnosetelegramm vom DP-Master abgeholt wurde.

Identifikation eines Auftrags

Die Eingangsparameter *IOID* und *LADDR* legen einen Auftrag eindeutig fest. Falls Sie den SFC 7 DP_PRAL auf einem DP-Slave aufgerufen haben und Sie diesen SFC erneut aufrufen, bevor der DP-Master den angeforderten Prozessalarm quittiert hat, dann hängt das weitere Verhalten des SFC entscheidend davon ab, ob es sich beim erneuten Aufruf um denselben Auftrag handelt: Stimmen die Parameter *IOID* und *LADDR* mit einem noch nicht abgeschlossenen Auftrag überein, so wird der SFC-Aufruf unabhängig vom Wert des Parameters *AL_INFO* als Folgeaufruf interpretiert, und in *RET_VAL* wird der Wert W#16#7002 eingetragen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	*REQ* = 1: Prozessalarm auf dem zugehörigen DP-Master auslösen.
IOID	INPUT	BYTE	E, A, M, D, L, Konstante	Kennung des Adressbereichs im Übergabespeicher (aus Sicht des DP-Slaves): ■ B#16#00: Bit15 von *LADDR* gibt an, ob Ein- (Bit15=0) oder Ausgangsadresse (Bit15=1) vorliegt. ■ B#16#54: Peripherie Eingang (PE) ■ B#16#55: Peripherie Ausgang (PA) Handelt es sich um eine Mischbaugruppe, ist die Bereichskennung der niedrigeren Adresse anzugeben. Bei gleichen Adressen ist B#16#54 anzugeben.
LAADR	INPUT	WORD	E, A, M, D, L, Konstante	Anfangsadresse des Adressbereich im Übergabespeicher (aus Sicht des DP-Slaves). Handelt es sich um einen Bereich, der zu einer Mischbaugruppe gehört, ist die kleinere der beiden Adressen anzugeben.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
AL_INFO	INPUT	DWORD	E, A, M, D, L, Konstante	Alarmkennung Diese wird dem OB 40, der auf dem zugehörigen DP-Master gestartet werden soll, mitgegeben (Variable OB40_POINT_ADDR). Falls Sie den intelligenten Slave an einem Fremdmaster betreiben, müssen Sie im Master das Diagnosetelegramm auswerten.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: Der ausgelöste Prozessalarm wurde vom DP-Master noch nicht quittiert.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Der Auftrag wurde fehlerfrei durchgeführt.
7000h	Erstaufruf mit *REQ* = 0. Es ist keine Prozessalarmanforderung aktiv; *BUSY* hat den Wert 0.
7001h	Erstaufruf mit *REQ* = 1. Eine Prozessalarmanforderung an den DP-Master wurde gestellt; *BUSY* hat den Wert 1.
7002h	Zwischenaufruf (*REQ* irrelevant). Der ausgelöste Prozessalarm wurde vom DP-Master noch nicht quittiert; *BUSY* hat den Wert 1.
8090h	Anfangsadresse des Adressbereichs im Übergabespeicher fehlerhaft
8091h	Alarm durch Projektierung gesperrt
8093h	Über das *IOID* und *LADDR* wird ein Modul angesprochen, von der aus eine Prozessalarmanforderung nicht möglich ist.
80B5h	Aufruf in DP-Master nicht zulässig
80C3h	Erforderliche Betriebsmittel (Speicher usw.) sind momentan belegt.
80C5h	Dezentrale Peripherie ist momentan nicht verfügbar (z.B. Stationsausfall).
80C8h	Die Funktion ist im aktuellen Betriebszustand des DP-Masters nicht erlaubt.
8xyyh	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

16.1.10 SFC 12 - D_ACT_DP - DP-Slave aktivieren und deaktivieren

Beschreibung

Mit dem SFC 12 D_ACT_DP können Sie projektierte DP-Slaves gezielt deaktivieren und wieder aktivieren. Darüber hinaus können Sie für jeden eingesetzten DP-Slave ermitteln, ob dieser momentan aktiviert oder deaktiviert ist.

Der SFC 12 ist nicht anwendbar auf PROFIBUS PA-Feldgeräte, die über DP/PA Link an ein DP-Mastersystem angeschlossen sind.

So lange ein oder mehrere SFC 12 Aufträge aktiv sind, können Sie keine geänderte Konfiguration vom PG in die CPU laden. Während des Ladens einer geänderten Konfiguration vom PG in die CPU weist die CPU die Aktivierung eines SFC 12-Auftrages ab.

Anwendung

Wenn Sie in einer CPU DP-Slaves konfigurieren, die real nicht vorhanden sind oder aktuell nicht benötigt werden, greift die CPU dennoch regelmäßig auf diese DP-Slaves zu. Nach deren Deaktivierung unterbleiben weitere CPU-Zugriffe. Dadurch kann der schnellstmögliche DP-Buszyklus erreicht werden, und die entsprechende Fehlerereignisse treten nicht mehr auf.

Beispiel

Sämtliche mögliche Maschinenoptionen sind vom Hersteller als DP-Slaves projektiert, um ein gemeinsames Anwenderprogramm über alle möglichen Optionen erstellen und pflegen zu können. Mit dem SFC 12 können Sie im Anlauf der Maschine alle nicht vorhandenen DP-Slaves deaktivieren.

Arbeitsweise

Der SFC 12 ist ein asynchron arbeitender SFC, d.h. die Bearbeitung erstreckt sich über mehrere SFC-Aufrufe. Sie starten den Auftrag, indem Sie den SFC 12 mit *REQ* = 1 aufrufen.

Über die Ausgangsparameter *RET_VAL* und *BUSY* wird der Zustand des Auftrages angezeigt.

Identifikation eines Auftrages

Falls Sie einen Deaktivierungs- bzw. Aktivierungsauftrag angestoßen haben und den SFC 12 erneut aufrufen, bevor dieser beendet wurde, hängt das weitere Verhalten des SFC entscheidend davon ab, ob es sich beim erneuten Aufruf um denselben Auftrag handelt. Stimmt der Eingangsparameter *LADDR* überein, so gilt der SFC-Aufruf als Folgeaufruf.

Deaktivieren von DP-Slaves

Wenn Sie einen DP-Slave mit dem SFC 12 deaktivieren, werden dadurch dessen Prozessausgänge auf die projektierten Ersatzwerte bzw. auf "0" gesetzt (sicherer Zustand).

Der zugehörige DP-Master spricht diesen DP-Slave im weiteren nicht mehr an. Deaktivierte DP-Slaves werden an den Fehler-LEDs des DP-Masters oder der CPU nicht als gestört oder fehlend gekennzeichnet.

Das Prozessabbild der Eingänge von deaktivierten DP-Slaves wird mit 0 aktualisiert, d.h. es wird wie bei ausgefallenen DP-Slaves behandelt.

Sie können nicht alle DP-Slaves deaktivieren. Mindestens 1 Slave muss am Bus aktiviert bleiben.

Falls Sie in Ihrem Programm mittels Direktzugriff auf die Nutzdaten eines zuvor deaktivierten DP-Slaves zugreifen, wird der Peripheriezugriffsfehler- OB (OB 122) aufgerufen und das zugehörige Startereignis in den Diagnosepuffer eingetragen.

Falls Sie per SFC (z.B. SFC 59 RD_REC) auf einen deaktivierten DP-Slave zugreifen, erhalten Sie in *RET_VAL* dieselbe Fehlerinformation wie bei einem nicht verfügbaren DP-Slave.

Das Deaktivieren eines DP-Slaves verursacht keinen Start des Programmablauffehler-OB 85, auch wenn dessen Ein- bzw. Ausgänge zum systemseitig zu aktualisierenden Prozessabbild gehören. Es erfolgt auch kein Eintrag in den Diagnosepuffer.

Das Deaktivieren eines DP-Slaves hat keinen Start des Slave-Ausfall-OB 86 zur Folge, und das Betriebssystem veranlasst auch keinen Eintrag in den Diagnosepuffer.

Fällt eine DP-Station aus, nachdem Sie diese mit dem SFC 12 deaktiviert haben, wird der Ausfall vom Betriebssystem nicht erkannt. Es erfolgt daher weder ein OB 86-Start noch ein Diagnosepuffereintrag. Der Stationsausfall wird erst beim erneuten Aktivieren der Station festgestellt und Ihnen über den zugehörigen *RET_VAL* mitgeteilt.

Falls Sie DP-Slaves deaktivieren wollen, die als Sender am Querverkehr beteiligt sind, wird empfohlen, zuerst die Empfänger (Mithörer) zu deaktivieren, die mithören, welche Eingangsdaten der Sender seinem DPMaster schickt. Erst im Anschluss daran deaktivieren Sie den Sender.

Aktivieren von DP-Slaves

Wenn Sie einen DP-Slave mit dem SFC 12 wieder aktivieren, wird dieser vom zugehörigen DP-Master konfiguriert und parametriert (wie bei der Wiederkehr einer ausgefallenen DP-Station). Die Aktivierung ist abgeschlossen, wenn der Slave Nutzdaten transferieren kann.

Das Aktivieren eines DP-Slaves verursacht keinen Start des Programmablauffehler- OB 85, auch wenn dessen Ein- bzw. Ausgänge zum systemseitig zu aktualisierenden Prozessabbild gehören. Es erfolgt auch kein Eintrag in den Diagnosepuffer.

Das Aktivieren eines DP-Slaves hat keinen Start des Slave-Ausfall-OB 86 zur Folge, und das Betriebssystem veranlasst auch keinen Eintrag in den Diagnosepuffer.

Wenn Sie versuchen, einen deaktivierten Slave, der physikalisch vom DP-Bus getrennt ist, mit dem SFC 12 zu aktivieren läuft eine Überwachungszeit von 10sec ab. Nach Ablauf dieser Überwachungszeit liefert der SFC den Fehlercode 80A2h, und der Slave bleibt deaktiviert. Falls er zu einem späteren Zeitpunkt wieder Verbindung zum DP-Bus hat, müssen Sie den Slave mit dem SFC 12 wieder aktivieren.

Das Aktivieren eines DP-Slaves kann geraume Zeit dauern. Falls Sie einen laufenden Aktivierungsauftrag abbrechen wollen, starten Sie den SFC 12 mit dem gleichen Wert für LADDR und MODE = 2. Sie wiederholen den Aufruf des SFC 12 mit MODE = 2 so lange, bis der erfolgreiche Abbruch des Aktivierungsauftrags mit RET_VAL = 0 angezeigt wird.

Falls Sie DP-Slaves aktivieren wollen, die am Querverkehr beteiligt sind, wird empfohlen, zuerst die Sender und anschließend die Empfänger (Mithörer) zu aktivieren.

CPU-Anlauf

Bei einem Neustart werden deaktivierte Slaves automatisch wieder aktiviert. Nach dem CPU-Anlauf versucht die CPU zyklisch zu allen projektierten und nicht deaktivierten Slaves, die nicht vorhanden oder nicht ansprechbar sind, Kontakt aufzunehmen.

Ein Aufruf des SFC 12 im Anlauf-OB 100 wird nicht unterstützt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	Pegelgetriggerter Steuerparameter *REQ* = 1: Aktivieren bzw. Deaktivieren durchführen
MODE	INPUT	BYTE	E, A, M, D, L, Konstante	Auftragskennung Mögliche Werte: 0: Auskunft einholen, ob der angesprochene DP-Slave aktiviert oder deaktiviert ist. 1: DP-Slave aktivieren 2: DP-Slave deaktivieren
LAADR	INPUT	WORD	E, A, M, D, L, Konstante	Beliebige logische Adresse des DP-Slaves
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
BUSY	OUTPUT	BOOL	E, A, M, D, L	Aktivkennung: *BUSY* = 1: Der Auftrag ist noch aktiv. *BUSY* = 0: Der Auftrag wurde beendet.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Der Auftrag wurde fehlerfrei durchgeführt.
0001h	DP-Slave aktiviert (Dieser Fehlercode ist nur möglich bei *MODE* = 0.)
0002h	DP-Slave deaktiviert (Dieser Fehlercode ist nur möglich bei *MODE* = 0.)
7000h	Erstaufruf mit *REQ* = 0. Der über *LADDR* festgelegte Auftrag ist nicht aktiv; BUSY hat den Wert 0.
7001h	Erstaufruf mit *REQ* = 1. Der über *LADDR* festgelegte Auftrag wurde angestoßen; BUSY hat den Wert 1.
7002h	Zwischenaufruf (*REQ* irrelevant). Der aktivierte Auftrag ist noch in Bearbeitung; BUSY hat den Wert 1.
8090h	Sie haben kein Modul mit der in *LADDR* angegebenen Adresse projektiert. Sie betreiben Ihre CPU als I-Slave und haben in *LADDR* eine Adresse dieses Slaves angegeben.
8092h	Der laufende Deaktivierungsvorgang eines DP-Slaves (*MODE* = 2) kann nicht durch seine Aktivierung (*MODE* = 1) abgebrochen werden. Aktivieren Sie den DP-Slave zu einem späteren Zeitpunkt.
8093h	Zu der in *LADDR* angegebenen Adresse gehört kein DP-Slave (es liegt keine Projektierung vor), oder der Parameter *MODE* ist nicht bekannt.
80A1h	Der angesprochene DP-Slave konnte nicht parametriert werden. (Dieser Fehlercode ist nur möglich bei *MODE* = 1.) **Hinweis!** Diese Fehlerinformation liefert der SFC nur dann, wenn während der Parametrierung des aktivierten Slaves dieser wieder ausfällt. Wenn nur die Parametrierung eines einzelnen Moduls nicht erfolgreich war, liefert der SFC die Fehlerinformation 0000h.
80A2h	Der angesprochene DP-Slave gibt keine Rückmeldung.

Wert	Beschreibung
80A3h	Der betroffene DP-Master unterstützt diese Funktion nicht.
80A4h	Die CPU unterstützt diese Funktion bei externen DP-Mastern nicht.
80A6h	Steckplatzfehler im DP-Slave; es kann nicht auf alle Nutzdaten zugegriffen werden. (Dieser Fehlercode ist nur möglich bei *MODE* = 1.) **Hinweis!** Diese Fehlerinformation liefert der SFC nur dann, wenn nach der Parametrierung des aktivierten Slaves und vor dem Ende des SFC der Slave wieder ausfällt. Wenn nur ein einzelnes Modul nicht verfügbar ist, liefert der SFC die Fehlerinformation 0000h.
80C1h	Der SFC 12 wurde gestartet und wird mit einer anderen logischen Adresse fortgesetzt. (Dieser Fehlercode ist nur möglich bei *MODE* = 1.)
80C3h	■ Temporärer Ressourcenfehler: die CPU bearbeitet momentan das mögliche Maximum an Aktivierungs-/Deaktivierungsaufträgen. (Dieser Fehlercode ist nur möglich bei *MODE* = 1 und *MODE* = 2.) ■ Die CPU erhält gerade eine geänderte Konfiguration. Das Aktivieren und Deaktivieren von DP-Slaves ist daher momentan nicht möglich.
F001h	Es dürfen nicht alle Slaves deaktiviert werden. Mindestens 1 Slave muss noch aktiv bleiben.
F002h	Unbekannte Slaveadresse

16.1.11 SFC 13 - DPNRM_DG - Slave-Diagnosedaten lesen

Beschreibung

Mit dem SFC 13 DPNRM_DG (read diagnosis data of a DP slave) werden die aktuellen Diagnosedaten eines DP-Slaves gelesen. Die Diagnosedaten eines jeden DP-Slave sind durch EN 50 170 Volume 2, PROFIBUS festgelegt.

Durch den Eingangsparameter *RECORD* wird der Zielbereich festgelegt, in den die gelesenen Daten nach fehlerfreier Datenübertragung eingetragen werden. Der Lesevorgang wird gestartet, wenn am Eingangsparameter *REQ* der Wert 1 anliegt.

Die nachfolgende Tabelle enthält Informationen zum prinzipiellen Aufbau der Slave-Diagnose.

Weitergehende Informationen entnehmen Sie bitte den Handbüchern zu den von Ihnen verwendeten DP-Slaves.

Byte	Bedeutung
0	Stationsstatus 1
1	Stationsstatus 2
2	Stationsstatus 3
3	Master-Stationsnummer
4	Herstellerkennung (High-Byte)
5	Herstellerkennung (Low-Byte)
6 ...	Weitere slavespezifische Diagnose

Arbeitsweise

Beim SFC 13 handelt es sich um einen asynchron arbeitenden SFC, d.h. dass sich die Bearbeitung über mehrere SFC-Aufrufe erstreckt. Die Ausgangsparameter *RET_VAL* und *BUSY* zeigen den Zustand des Auftrags an.

Zusammenhang zwischen Aufruf, *REQ, RET_VAL* und *BUSY*:

Lfd. Nr. des Aufrufs	Aufrufart	REQ	RET_VAL	BUSY
1	erster Aufruf	1	7001h bzw. Fehlercode	1 0
2 ... (n-1)	Zwischenaufruf	irrelevant	7002h	1
n	letzter Aufruf	irrelevant	Falls keine Fehler aufgetreten sind, wird die Anzahl gelieferter Daten in Bytes als positive Zahl eingetragen bzw. Fehlercode, falls Fehler aufgetreten sind.	0

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	*REQ* = 1: Anforderung zum Lesen
LADDR	INPUT	WORD	E, A, M, D, L, Konstante	Projektierte Diagnoseadresse des DP-Slaves
RET_VAL	OUTPUT	INT	E, A, M, D, L	Rückgabewert
RECORD	OUTPUT	ANY	E, A, M, D, L	Zielbereich für die gelesenen Diagnosedaten. Es ist nur der Datentyp BYTE zulässig. Die Mindestlänge des zu lesenden Datensatzes bzw. des Zielbereichs beträgt 6. Die Maximallänge des zu lesenden Datensatzes beträgt 240. Bei Normslaves, bei denen die Anzahl der Normdiagnosedaten größer als 240 Bytes ist und maximal 244Bytes beträgt, werden die ersten 240Bytes in den Zielbereich übertragen und das entsprechende Overflow-Bit in den Daten gesetzt.
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: Der Lesevorgang ist noch nicht beendet.

RECORD

Von der CPU wird die tatsächliche Länge der gelesenen Diagnosedaten ausgewertet:

Wenn die Längenangabe von *RECORD*

- kleiner als die Anzahl der gelieferten Daten ist, werden die Daten verworfen, und in *RET_VAL* wird die zugehörige Fehlerinformation eingetragen.
- größer oder gleich der Anzahl der gelieferten Daten ist, werden die Daten in den Zielbereich übernommen, und in *RET_VAL* wird die tatsächliche Länge als positiver Wert eingetragen.

Sie müssen darauf achten, dass die Aktualparameter von RECORD bei allen Aufrufen, die zu einem Auftrag gehören, übereinstimmen. Ein Auftrag ist eindeutig festgelegt durch den Eingangsparameter LADDR und RECORD.

Normslaves

Bei Normslaves, bei denen die Anzahl der Normdiagnosedaten zwischen 241 und 244Bytes liegt, ist folgendes beachten:

Falls die Längenangabe von *RECORD*

- kleiner als 240Bytes ist, werden die Daten verworfen und in *RET_VAL* wird die zugehörige Fehlerinformation eingetragen.
- größer oder gleich 240Bytes ist, werden die ersten Bytes der Normdiagnosedaten in den Zielbereich übertragen und das entsprechende Overflow-Bit in den Daten gesetzt.

RET_VAL (Rückgabewert)

Trat während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Falls kein Fehler auftrat, steht in *RET_VAL* die Länge der tatsächlich übertragenen Daten.

Die Anzahl der gelesenen Daten ist bei einem DP-Slave von seinem Diagnosezustand abhängig.

Fehlerinformation

Nähere Informationen zu allgemeinen Fehlerinformationen finden Sie am Anfang dieses Kapitels.

Die für den SFC 13 spezifischen Fehlerinformationen sind eine Teilmenge der Fehlerinformationen für den SFC 59 RD_REC.

Nähere Informationen finden Sie im SFC 59.

16.1.12 SFC 14 - DPRD_DAT - Konsistente Nutzdaten lesen

Beschreibung

Mit dem SFC 14 DPRD_DAT (read consistent data of a DP-normslave) werden die konsistenten Daten eines DP-Normslaves ausgelesen. Dabei muss die Länge der konsistenten Daten drei oder mehr als vier Bytes betragen, wobei die Maximallänge 128Byte beträgt. Der Eingangsparameter *RECORD* legt den Zielbereich fest, in den die gelesenen Daten nach einer fehlerfreien Datenübertragung eingetragen werden. Dabei muss der Zielbereich dieselbe Länge aufweisen wie von Ihnen für das selektierte Modul projektiert worden ist.

Handelt es sich um einen DP-Normslave mit modularem Aufbau bzw. mit mehreren DP-Kennungen, so kann mit einem SFC 14-Aufruf jeweils nur auf die Daten eines Moduls / DP-Kennung unter der projektierten Anfangsadresse zugegriffen werden.

Der SFC 14 wird verwendet, da mit den Ladebefehlen, die auf die Peripherie bzw. auf das Prozessabbild der Eingänge zugreifen, maximal vier Bytes zusammenhängend ausgelesen werden können.

Definition

Konsistente Daten

Als konsistente Daten werden Daten bezeichnet, die inhaltlich zusammengehören und nicht getrennt werden dürfen. Z.B. ist es wichtig die Werte von Analogmodulen immer konsistent zu behandeln, denn der Wert eines Analogmoduls darf durch das Auslesen zu zwei verschiedenen Zeitpunkten nicht verfälscht werden.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
LADDR	INPUT	WORD	E, A, M, D, L, Konstante	Projektierte Anfangsadresse aus dem E-Bereich des Moduls, aus dem gelesen werden soll
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
RECORD	OUTPUT	ANY	E, A, M, D, L	Zielbereich für die gelesenen Nutzdaten. Er muss genauso lang sein, wie für das selektierte Modul projektiert wurde. Es ist nur der Datentyp BYTE zulässig.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Es ist kein Fehler aufgetreten.
8090h	Für die angegebene logische Basisadresse haben sie kein Modul projektiert, oder Sie haben die Einschränkung über die Länge der konsistenten Daten nicht beachtet.
8092h	In ANY-Referenz ist eine Typangabe ungleich BYTE angegeben.
8093h	Für die unter *LADDR* angegebene logische Adresse existiert kein DP-Modul, von der Sie konsistente Daten lesen können.
80A0h	Anfangsadresse des Adressbereichs im Übergabespeicher fehlerhaft.
80B0h	Slaveausfall an externer DP-Anschaltung
80B1h	Die Länge des angegebenen Zielbereichs ist ungleich der projektierten Nutzdatenlänge.
80B2h	Systemfehler bei externer DP-Anschaltung
80B3h	Systemfehler bei externer DP-Anschaltung
80C0h	Systemfehler bei externer DP-Anschaltung
80C2h	Systemfehler bei externer DP-Anschaltung
80Fxh	Systemfehler bei externer DP-Anschaltung
87xyh	Systemfehler bei externer DP-Anschaltung
808xh	Systemfehler bei externer DP-Anschaltung

16.1.13 SFC 15 - DPWR_DAT - Konsistente Nutzdaten schreiben

Beschreibung

Mit dem SFC 15 DPWR_DAT (write consistent data to a DP-normslave) werden die Daten, die im Eingangsparameter *RECORD* stehen, konsistent zum adressierten DP-Normslave übertragen. Dabei muss die Länge der konsistenten Daten drei oder mehr als vier Bytes betragen, wobei die Maximallänge 128Byte beträgt. Die Datenübertragung erfolgt synchron, d.h. nach Beendigung des SFC ist der Schreibvorgang abgeschlossen. Es ist darauf zu achten, dass der Quellbereich dieselbe Länge aufweist, wie von Ihnen für das selektierte Modul projektiert worden ist.

Handelt es sich um einen DP-Normslave mit modularem Aufbau, kann nur auf ein Modul des DP-Slaves zugegriffen werden.

Der SFC 15 wird verwendet, da mit den Transferbefehlen, die auf die Peripherie bzw. auf das Prozessabbild der Ausgänge zugreifen, maximal vier Bytes zusammenhängend geschrieben werden können.

Definition

Konsistente Daten

Als konsistente Daten werden Daten bezeichnet, die inhaltlich zusammengehören und nicht getrennt werden dürfen. Beispielsweise ist es wichtig die Werte von Analogmodulen immer konsistent zu behandeln, denn der Wert eines Analogmoduls darf durch das Auslesen zu zwei verschiedenen Zeitpunkten nicht verfälscht werden.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
LADDR	INPUT	WORD	E, A, M, D, L, Konstante	Projektierte Anfangsadresse aus dem A-Bereich des Moduls, auf die geschrieben werden soll
RECORD	INPUT	ANY	E, A, M, D, L	Quellbereich für die zu schreibenden Nutzdaten. Er muss genauso lang sein, wie für das selektierte Modul projektiert wurde. Es ist nur der Datentyp BYTE zulässig.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Es ist kein Fehler aufgetreten.
8090h	Für die angegebene logische Basisadresse haben sie kein Modul projektiert, oder Sie haben die Einschränkung über die Länge der konsistenten Daten nicht beachtet.
8092h	In ANY-Referenz ist eine Typangabe ungleich BYTE angegeben.
8093h	Für die unter *LADDR* angegebene logische Adresse existiert kein DP-Modul, auf die Sie konsistente Daten schreiben können.
80A1h	Das selektierte Modul ist fehlerhaft.
80B0h	Slaveausfall an externer DP-Anschaltung
80B1h	Die Länge des angegebenen Quellbereichs ist ungleich der projektierten Nutzdatenlänge.
80B2h	Systemfehler bei externer DP-Anschaltung
80B3h	Systemfehler bei externer DP-Anschaltung
80C1h	Die Daten des auf dem Modul vorangegangenen Schreibauftrags sind von dem Modul noch nicht bearbeitet.
80C2h	Systemfehler bei externer DP-Anschaltung
80Fxh	Systemfehler bei externer DP-Anschaltung
85xyh	Systemfehler bei externer DP-Anschaltung
808xh	Systemfehler bei externer DP-Anschaltung

16.1.14 SFC 17 - ALARM_SQ und SFC 18 - ALARM_S

Beschreibung

Der SFC 17 ALARM_SQ (Erzeugung quittierbarer bausteinbezogener Meldungen) und der SFC 18 ALARM_S (Erzeugung stets quittierbarer bausteinbezogener Meldungen) generieren bei jedem Aufruf eine Meldung, an die Sie einen Begleitwert anhängen können. Die Meldung wird an alle dafür angemeldeten Teilnehmer gesendet. Der SFC 17 und SFC 18 stellen Ihnen also einen einfachen Meldemechanismus zur Verfügung. Bitte beachten Sie hierbei, dass Sie den SFC 17 und SFC 18 nur dann aufrufen, wenn der Wert des meldeauslösenden Signals SIG gegenüber dem letzten Aufruf invertiert ist. Ist dies nicht der Fall, so wird Ihnen dies über *RET_VAL* mitgeteilt, und es wird keine Meldung gesendet. Beim ersten Aufruf des SFC 17 und SFC 18 müssen Sie dafür Sorge tragen, dass am Eingang SIG "1" anliegt. Sonst bekommen Sie über *RET-VAL* eine Fehlerinformation, und es wird keine Meldung gesendet.

Rufen Sie den SFC 17 und den SFC 18 aus einem FB heraus auf, dem Sie zuvor die entsprechenden Systemattribute zugewiesen haben!

Systemressourcen

Bei der Meldungserzeugung mit dem SFC 17 und dem SFC 18 belegt das Betriebssystem für die Dauer eines Signalzyklus eine Systemressource.

Der Signalzyklus dauert beim SFC 18 vom SFC-Aufruf mit *SIG* = "1" bis zum erneuten Aufruf mit *SIG* = "0".

Beim SFC 17 kommt zu dieser Zeitspanne ggf. noch die Zeit bis zur Quittierung des kommenden Signals durch eines der angemeldeten Anzeigegeräte hinzu.

Falls innerhalb des Signalzyklus ein Überladen oder Löschen des meldungserzeugenden Bausteins erfolgt, bleibt die zugehörige Systemressource bis zum nächsten Neustart belegt.

Meldungs-Quittierung

Sie können die vom SFC 17 gesendeten Meldungen mit Signalzustand "1" an einem angemeldeten Anzeigegerät quittieren. Den Quittierzustand der letzten "Gekommen-Meldung" und den Signalzustand beim letzten SFC 17-Aufruf können Sie mit Hilfe des SFC 19 ALARM_SC ermitteln.

Meldungen, die Sie mit dem SFC 18 gesendet haben, sind immer implizit quittiert. Den Signalzustand beim letzten SFC 18-Aufruf können Sie mit Hilfe des SFC 19 ALARM_SC ermitteln.

Zwischenspeicherung

Die SFCs 17 und 18 belegen Systemressourcen. Darin werden unter anderem die letzten beiden Signalzustände inklusiv Zeitstempel und Begleitwert zwischengespeichert. Erfolgt ein Aufruf des SFC 17 oder SFC 18 zu einem Zeitpunkt, an dem die Signalzustände der beiden letzten "gültigen" SFC-Aufrufe noch nicht gesendet sind (Signaloverflow), werden der aktuelle und der letzte Signalzustand verworfen und eine Overflow- Kennung im Zwischenspeicher gesetzt. Zum nächstmöglichen Zeitpunkt wird das vorletzte Signal samt Overflow-Kennung gesendet.

Instanzoverflow

Falls die Anzahl von SFC 17- und SFC 18-Aufrufen größer ist als die maximale Anzahl von Systemressourcen Ihrer CPU, kann es zu einem Ressourcenengpass (Instanzoverflow) kommen. Dies wird Ihnen sowohl durch eine Fehlerinformation in *RET_VAL* als auch an den angemeldeten Anzeigegeräten mitgeteilt.

Die maximale Anzahl von SFC 17- oder SFC 18-Aufrufen ist CPUabhängig.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
SIG	INPUT	BOOL	E, A, M, D, L	Das meldungsauslösende Signal
ID	INPUT	WORD	E, A, M, D, L	Datenkanal für Meldungen: EEEEh
EV_ID	INPUT	DWORD	Konstante (E, A, M, D, L)	Meldungsnummer (nicht erlaubt: 0)
SD	INPUT	ANY	E, A, M, D, T, Z	Begleitwert
RET_VAL	OUTPUT	INT	E, A, M, D, L	Rückgabewert

SD

Begleitwert

Maximale Länge: 12Byte

Zulässig sind nur die Datentypen:

BOOL (nicht erlaubt Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME

RET_VAL (Rückgabewert)

Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Wert	Beschreibung
0000h	kein Fehler
0001h	■ Der Begleitwert ist länger als die maximal zulässige Länge oder ■ Der Zugriff auf den Anwenderspeicher ist nicht möglich (z.B. Zugriff auf gelöschten DB). Die Meldung wird gesendet. ■ Der Begleitwert zeigt auf einen Wert im Lokaldatenbereich. Die Meldung wird gesendet.
0002h	Warnung: Der letzte freie Meldequittierspeicher wurde belegt.
8081h	Der angegebene *EV_ID* liegt außerhalb des zulässigen Bereiches.
8082h	Meldungsverlust, da Ihre CPU keine Ressourcen für die Erzeugung bausteinbezogener Meldungen durch SFCs mehr hat.
8083h	Meldungsverlust, da derselbe Signalwechsel bereits vorliegt, aber noch nicht gesendet werden konnte (Signaloverflow).
8084h	Beim aktuellen und beim vorangegangenen SFC 17- / SFC 18-Aufruf hat das meldungsauslösende Signal *SIG* denselben Wert.
8085h	Für die angegebene *EV_ID* liegt keine Anmeldung vor.
8086h	Ein SFC-Aufruf für die angegebene EV_ID ist bereits in einer Prioritätsklasse niedriger Priorität in Bearbeitung.
8087h	Beim ersten Aufruf des SFC 17, SFC 18 hatte das meldungsauslösende Signal den Wert "0".
8088h	Die angegebene *EV_ID* wird bereits von einer anderen Systemressource (zu SFC 17, SFC 18) belegt.
8xyyh	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

16.1.15 SFC 19 - ALARM_SC - Quittierzustand der letzten Meldung

Beschreibung

Mit Hilfe des SFC 19 ALARM_SC können Sie:

- Den Quittierzustand der letzten ALARM_SQ-Gekommen-Meldung und den Zustand des meldungsauslösenden Signals beim letzten Aufruf des SFC 17 ermitteln
- Den Zustand des meldungsauslösenden Signals beim letzten Aufruf des SFC 18 ermitteln

Die Meldung bzw. das Signal ist über die von Ihnen vorgegebene Meldungsnummer eindeutig referenziert, falls Sie die Meldungsnummern mit Hilfe der Meldungsprojektierung vergeben haben.

Der SFC 19 greift auf die temporär belegten Speicher des SFC 17 und SFC 18 zu.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EV_ID	INPUT	DWORD	E, A, M, D, L, Konstante	Meldungsnummer, zu der Sie den Signalzustand beim letzten SFC-Aufruf bzw. den Quittierzustand der letzten Gekommen-Meldung (nur bei SFC 17) ermitteln möchten
RET_VAL	OUTPUT	INT	E, A, M, D, L	Rückgabewert
STATE	OUTPUT	BOOL	E, A, M, D, L	Zustand des meldungsauslösendenSignals beim letzten SFC-Aufruf
Q_STATE	OUTPUT	BOOL	E, A, M, D, L	Falls der angegebene Parameter *EV_ID* zu einem SFC 18-Aufruf gehört: "1"
				Falls der angegebene Parameter *EV_ID* zu einem SFC 17-Aufruf gehört: Quittierzustand der letzten Gekommen-Meldung: "0": nicht quittiert "1": quittiert

RET_VAL (Rückgabewert)

Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Wert	Beschreibung
0000h	kein Fehler
8081h	Der angegebene *EV_ID* liegt außerhalb des zulässigen Bereiches.
8082h	Zu dieser *EV_ID* ist momentan kein Speicherplatz belegt (Mögliche Ursache: Das zugehörige Signal hatte noch nie den Zustand "1", oder es hat bereits wieder den Zustand "0" angenommen).
8xyyh	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

16.1.16 SFC 20 - BLKMOV - Variable kopieren

Beschreibung

Mit dem SFC 20 BLKMOV (block move) wird der Inhalt eines Speicherbereiches (Quellfeld) in einen anderen Speicherbereich (Zielfeld) kopiert.

Es können alle Speicherbereiche kopiert werden, außer:

- folgende Bausteine: FC, SFC, FB, SFB, OB, SDB
- Zähler
- Zeiten
- Speicherbereiche des Peripheriebereiches.

Es besteht die Möglichkeit, das der Quellparameter auch in einem nichtablaufrelevanten Datenbaustein (DB, der mit dem Schlüsselwort UNLINKED kompiliert wurde) im Ladespeicher liegen kann.

Unterbrechbarkeit

Die Schachtelungstiefe ist nicht begrenzt, solange das Quellfeld nicht Teil eines Datenbausteins ist, der nur im Ladespeicher vorhanden ist. Wird jedoch eine SFC 20-Bearbeitung unterbrochen, bei der aus einem nicht ablaufrelevanten DB kopiert wird, kann eine solche SFC 20-Bearbeitung nicht mehr eingeschachtelt werden.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
SRCBLK	INPUT	ANY	E, A, M, D, L	Angabe des Speicherbereichs, der kopiert werden soll (Quellfeld). Arrays vom Datentyp STRING sind nicht erlaubt.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
DSTBLK	OUTPUT	ANY	E, A, M, D, L	Angabe des Speicherbereichs, in den kopiert werden soll (Zielfeld). Arrays vom Datentyp STRING sind nicht erlaubt.

Quell- und Zielfeld dürfen sich nicht überlappen. Wenn das angegebene Zielfeld größer als das Quellfeld ist, dann werden auch nur so viele Daten in das Zielfeld kopiert, wie im Quellfeld stehen. Wenn das angegebene Zielfeld jedoch kleiner als das Quellfeld ist, dann werden nur so viele Daten kopiert, wie das Zielfeld aufnehmen kann.

Wenn der ANY-Pointer (Quelle oder Ziel) vom Typ BOOL ist, so muss die angegebene Länge durch 8 teilbar sein, da ansonsten der SFC nicht ausgeführt wird.

Wenn der ANY-Pointer vom Typ STRING ist, so muss die angegebene Länge 1 sein.

RET_VAL (Rückgabewert)

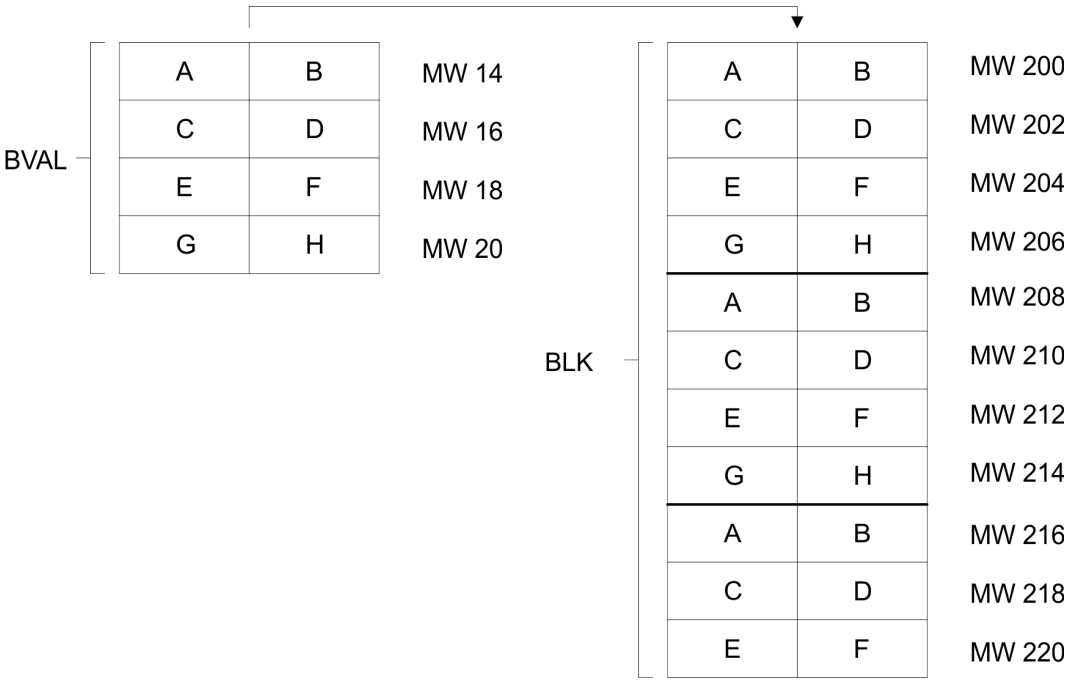
Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Wert	Beschreibung
0000h	kein Fehler
8091h	Die Schachtelungstiefe wurde überschritten.

16.1.17 SFC 21 - FILL - Feld vorbesetzen

Beschreibung

Mit dem SFC 21 FILL können Sie einen Speicherbereich (Zielfeld) mit dem Inhalt eines anderen Speicherbereiches (Quellfeld) vorbesetzen. In das angegebene Zielfeld kopiert der SFC 21 solange den Inhalt, bis der Speicherbereich komplett beschrieben ist.



Quell- und Zielfeld dürfen sich nicht überlappen.

Wenn das vorzubelegende Zielfeld kein ganzzahliges Vielfaches der Länge des Eingangsparameters BVAL ist, wird das Zielfeld trotzdem bis zum letzten Byte beschrieben.

Wenn das vorzubelegende Zielfeld kleiner als das Quellfeld ist, dann werden nur so viele Daten kopiert, wie das Zielfeld aufnehmen kann.

Mit dem SFC 21 können Sie keine Werte schreiben in:

- folgende Bausteine: FB, SFB; FC, SFC, OB, SDB
- Zähler
- Zeiten
- Speicherbereiche des Peripheriebereiches

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
BVAL	INPUT	ANY	E, A, M, D, L	Enthält den Wert bzw. die Beschreibung des Feldes, mit dessen Inhalt das Zielfeld vorbesetzt werden soll (Quellfeld). Arrays vom Datentyp STRING sind nicht erlaubt.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
BLK	OUTPUT	ANY	E, A, M, D, L	Enthält die Beschreibung des Feldes, das vorbesetzt werden soll (Zielfeld). Arrays vom Datentyp STRING sind nicht erlaubt.

Parameter ist eine Struktur

Wenn als Eingangsparameter eine Struktur übergeben wird, muss folgende Besonderheit berücksichtigt werden: Die Länge einer Struktur wird immer auf eine gerade Anzahl von Bytes ausgerichtet. Wenn also eine Struktur mit einer ungeraden Anzahl Bytes deklariert wird, benötigt die Struktur ein Byte zusätzlichen Speicherplatz.

Beispiel:

Die Struktur wird folgendermaßen deklariert:

STRUKTUR_7_BYTE: STRUCT

BYTE_1_2 : WORD

BYTE_3_4 : WORD

BYTE_5_6 : WORD

BYTE_7: BYTE

END_STRUCT

Die deklarierte Struktur "STRUKTUR_7_BYTE" benötigt 8Bytes Speicherplatz.

RET_VAL (Rückgabewert)

Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Der SFC 21 gibt keine spezifischen Fehlerinformationen aus.

16.1.18 SFC 22 - CREAT_DB - Datenbaustein erzeugen

Beschreibung

Mit dem SFC 22 CREAT_DB (create data block) wird im Anwenderprogramm ein Datenbaustein erzeugt, der keine vorbesetzten Werte enthält. Es wird ein Datenbaustein mit einer Nummer aus einem angegebenen Bereich und mit vorgegebener Größe erzeugt, wobei aus dem angegebenen Bereich immer die kleinste mögliche Nummer an den DB vergeben wird. Um einen DB mit bestimmter Nummer zu erzeugen wird der Ober- und Untergrenze des vorzugebenden Bereichs die gleiche Nummer gegeben. Sind im Anwenderprogramm bereits DBs enthalten, können diese Nummern nicht mehr vergeben werden. Die Länge des DBs muss in einer geraden Zahl angegeben werden.

Unterbrechbarkeit

Der SFC 22 kann durch höherpriore OBs unterbrochen werden. Wird ein SFC 22 in einem höherprioren OB aufgerufen, wird dieser Aufruf mit dem Fehlercode 8091h abgewiesen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
LOW_LIMIT	INPUT	WORD	E, A, M, D, L, Konstante	Der untere Grenzwert ist die kleinste Nummer in dem Bereich der Nummern, die Sie Ihrem Datenbaustein zuordnen können.
UP_LIMIT	INPUT	WORD	E, A, M, D, L, Konstante	Der obere Grenzwert ist die größte Nummer in dem Bereich der Nummern, die Sie Ihrem Datenbaustein zuordnen können.
COUNT	INPUT	WORD	E, A, M, D, L, Konstante	Der Zählwert gibt die Anzahl der Datenbytes an, die Sie für ihren Datenbaustein reservieren möchten. Sie müssen hier eine gerade Zahl an Bytes (maximal 65534) angeben.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
DB_NUMBER	OUTPUT	WORD	E, A, M, D, L	Die Datenbausteinnummer ist die Nummer des erstellten Datenbausteins. Im Fehlerfall (Bit 15 von *RET_VAL* wurde gesetzt) wird in *DB_NUMBER* der Wert 0 eingetragen.

RET_VAL (Rückgabewert) Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Wert	Beschreibung
0000h	kein Fehler
8091h	Sie haben den SFC 22 geschachtelt aufgerufen.
8092h	Die Funktion "Erzeugen eines DB" ist momentan nicht durchführbar, weil ■ die Funktion "Komprimieren des Anwenderspeichers" gerade aktiv ist.
80A1h	Fehler in der Nummer des DBs: ■ die Nummer ist 0 ■ die Nummer überschreitet die CPU-spezifische DB-Anzahl ■ Untergrenze > Obergrenze
80A2h	Fehler in der Länge des DBs: ■ die Länge ist 0 ■ die Länge wurde als ungerade Zahl angegeben ■ die Länge ist größer als die CPU zulässt
80B1h	Es ist keine DB-Nummer frei.
80B2h	Es steht nicht genügend freier Speicherplatz zur Verfügung.
80B3h	Es steht nicht genügend zusammenhängender Speicher zur Verfügung. (Komprimieren durchführen!)

16.1.19 SFC 23 - DEL_DB - Datenbaustein löschen

Beschreibung

Mit dem SFC 23 DEL_DB (delete data block) wird ein Datenbaustein im Arbeitsspeicher und gegebenenfalls im Ladespeicher der CPU gelöscht. Der angegebene DB darf weder in der aktuellen Ablaufebene noch in einer Ablaufebene mit niederer Priorität aufgeschlagen sein, also weder in einem der beiden DB-Register noch im B-Stack eingetragen sein, ansonsten wechselt die CPU beim Aufruf des SFC 23 in den Betriebszustand STOP.

In der folgenden Tabelle ist aufgeführt, wann ein DB mit dem SFC 23 gelöscht werden kann.

Wenn der DB ...	dann ist er mit dem SFC 23 ...
durch Aufruf des SFC 22 "CREAT_DB" erzeugt wurde,	löschbar.
nicht mit dem Schlüsselwort UNLINKED erzeugt wurde,	löschbar.

Unterbrechbarkeit

Der SFC 23 kann von höher prioren Ablaufebenen unterbrochen werden. Wird dort der SFC erneut aufgerufen, dann wird dieser zweite Aufruf abgebrochen, und in *RET_VAL* wird der Fehlercode 8091h eingetragen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
DB_NUMBER	INPUT	WORD	E, A, M, D, L, Konstante	Nummer des zu löschenden DB.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

RET_VAL (Rückgabewert)

Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Wert	Beschreibung
0000h	kein Fehler
8091h	Bei ineinandergeschachtelten SFC 23-Aufrufen wurde die maximale Schachtelungstiefe überschritten.
8092h	Die Funktion "Löschen eines DB" ist momentan nicht durchführbar, weil die Funktion: ■ "Komprimieren des Anwenderspeichers" oder ■ "Anwenderprogramm sichern" gerade aktiv ist.
80A1h	Fehler in der Nummer des DBs: ■ die Nummer ist 0 ■ die Nummer ist größer als die maximal mögliche DB-Anzahl
80B1h	Der DB mit der angegebenen Nummer ist auf der CPU nicht vorhanden.
80B2h	Der DB mit der angegebenen Nummer wurde mit dem Schlüsselwort UNLINKED erzeugt.
80B3h	Der DB befindet sich auf der Flashcard.

16.1.20 SFC 24 - TEST_DB - Datenbaustein testen

Beschreibung

Mit dem SFC 24 TEST_DB (test data block) können Informationen über einen im Arbeitsspeicher der CPU liegenden Datenbaustein abgerufen werden. Der SFC ermittelt die Anzahl der Datenbytes und stellt fest, ob der ausgewählte DB schreibgeschützt ist.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
DB_NUMBER	INPUT	WORD	E, A, M, D, L, Konstante	Nummer des zu überprüfenden DBs
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
DB_LENGTH	OUTPUT	WORD	E, A, M, D, L	Anzahl der Datenbytes, die der ausgewählte DB enthält.
WRITE_PROT	OUTPUT	BOOL	E, A, M, D, L	Information über die Schreibschutzkennung des ausgewählten DB (1 bedeutet schreibgeschützt).

RET_VAL (Rückgabewert)

Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Wert	Beschreibung
0000h	kein Fehler
80A1h	Fehler beim Eingangsparameter *DB_NUMBER*: ■ die Nummer ist 0 ■ die Nummer ist größer als die maximal mögliche DB-Anzahl
80B1h	Der DB mit der angegebenen Nummer ist auf der CPU nicht vorhanden.
80B2h	Der DB mit der angegebenen Nummer wurde mit dem Schlüsselwort UNLINKED erzeugt.

16.1.21 FC/SFC 25 - COMPRESS - Komprimieren Anwenderspeicher

Entstehen von Speicherlücken

Durch mehrfaches Löschen und Nachladen von Bausteinen können sowohl im Lade- als auch im Arbeitsspeicher Lücken entstehen, die den nutzbaren Speicherbereich verringern.

Beschreibung

Mit der FC/SFC 25 COMPRESS stoßen Sie die Komprimierung sowohl des RAM-Anteils des Ladespeichers als auch des Arbeitsspeichers an. Der Komprimiervorgang ist derselbe wie nach einem externen Anstoß im Betriebszustand RUN (Stellung des Betriebsartenschalters).

Ist die Komprimierung aufgrund eines externen Anstoßes (über Baugruppenzustand) bereits aktiv, führt der Aufruf der FC/SFC 25 zur Fehleranzeige.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
RET_VAL	OUTPUT	INT	E, A, M, D, L	Fehlerinformation
BUSY	OUTPUT	BOOL	E, A, M, D, L	Information, ob Komprimierung aufgrund der FC/SFC 25 aktiv ist. (1 bedeutet aktiv)
DONE	OUTPUT	BOOL	E, A, M, D, L	Information, ob der Komprimiervorgang, der durch die FC/SFC 25 angestoßen wurde, erfolgreich beendet wurde. (1 bedeutet erfolgreich beendet)

Kontrolle über Komprimiervorgang

Bei einem einmaligen Aufruf der FC/SFC 25 COMPRESS stoßen Sie den Komprimiervorgang an.

Der FC/SFC 25 ist im Zyklus aufzurufen. Nach jedem Aufruf ist zunächst der Parameter *RET_VAL* zu bewerten. Für den Fall, dass er den Wert 0 hat, sind die Parameter *BUSY* und *DONE* zu bewerten. Ist *BUSY* = 1 und *DONE* = 0, so weist dies darauf hin, dass der Komprimiervorgang noch aktiv ist. Erst wenn *BUSY* den Wert 0 und *DONE* den Wert 1 annimmt, wurde der Komprimiervorgang erfolgreich beendet.

Falls danach die FC/SFC 25 wieder aufgerufen wird, wird erneut ein Komprimieren angestoßen.

16.1.22 SFC 28 ... SFC 31 - Uhrzeitalarm

16.1.22.1 Übersicht

Voraussetzungen

Ein Uhrzeitalarm ist die Ursache für den uhrzeitgesteuerten Aufruf des Uhrzeitalarm-OB 10. Hierzu müssen folgende Voraussetzungen erfüllt sein:

- Der Uhrzeitalarm-OB ist parametriert über Hardware-Konfiguration oder mittels des SFC 28 (SET_TINT) im Anwenderprogramm.
- Der Uhrzeitalarm-OB ist aktiviert über Hardware-Konfiguration oder mittels des SFC 30 (ACT_TINT) im Anwenderprogramm.
- Der Uhrzeitalarm-OB darf nicht abgewählt worden sein.
- Der Uhrzeitalarm-OB muss in der CPU geladen sein.
- Falls das Stellen über den SFC 30 erfolgt, dürfen Startdatum und Startuhrzeit bei einmaliger Ausführung nicht abgelaufen sein, bei periodischer Ausführung wird der Uhrzeitalarm-OB zur nächsten abgelaufenen Periode aufgerufen (Startzeitpunkt + Vielfaches der Periodendauer).

SFCs 28 ... 31

Folgende SFCs kommen bei Uhrzeitalarmen zum Einsatz:

- Stellen: SFC 28
- Stornieren: SFC 29
- Aktivieren: SFC 30
- Abfragen: SFC 31

16.1.22.2 SFC 28 - SET_TINT - Uhrzeitalarm stellen

Mit dem SFC 28 SET_TINT (set time-of-day interrupt) können Startdatum und -uhrzeit der Uhrzeitalarm-Organisationsbausteine eingestellt werden. Bei der Startuhrzeit werden die Angaben für Sekunden und Millisekunden ignoriert und auf 0 gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
OB_NR	INPUT	INT	E, A, M, D, L, Konstante	Nummer des OBs, der zum Zeitpunkt *SDT* + Vielfaches von *PERIOD* gestartet wird (OB 10, OB 11).
SDT	INPUT	DT	D, L	Startdatum und Startuhrzeit
PERIOD	INPUT	WORD	E, A, M, D, L, Konstante	Periode vom Ausgangspunkt SDT an: 0000h = Einmal 0201h = Minütlich 0401h = Stündlich 1001h = Täglich 1201h = Wöchentlich 1401h = Monatlich 1801h = Jährlich 2001h = Monatsende
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

RET_VAL (Rückgabewert) Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Wert	Beschreibung
0000h	kein Fehler
8090h	Fehlerhafter Parameter *OB_NR*
8091h	Fehlerhafter Parameter *SDT*
8092h	Fehlerhafter Parameter *PERIOD*
80A1h	Der eingestellte Startzeitpunkt liegt in der Vergangenheit.

16.1.22.3 SFC 29 - CAN_TINT - Uhrzeitalarm stornieren

Mit dem SFC 29 CAN_TINT (cancel time-of-day interrupt) können Startdatum und -uhrzeit eines angegebenen Uhrzeitalarm - Organisationsbausteins gelöscht werden.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
OB_NR	INPUT	INT	E, A, M, D, L, Konstante	Nummer des OBs, dessen Startdatum und -uhrzeit gelöscht werden soll (OB 10, OB 11).
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Es ist kein Fehler aufgetreten.
8090h	Fehlerhafter Parameter *OB_NR*
80A0h	Kein Startdatum/-uhrzeit festgelegt für den betreffenden Uhrzeitalarm-OB

16.1.22.4 SFC 30 - ACT_TINT - Uhrzeitalarm aktivieren

Mit dem SFC 30 ACT_TINT (activate time-of-day interrupt) kann ein angegebener Uhrzeitalarm-Organisationsbaustein aktiviert werden.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
OB_NR	INPUT	INT	E, A, M, D, L, Konstante	Nummer des OBs, der aktiviert werden soll (OB 10, OB 11)
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Es ist kein Fehler aufgetreten.
8090h	Fehlerhafter Parameter *OB_NR*
80A0h	Kein Startdatum/-uhrzeit festgelegt für den betreffendenUhrzeitalarm-OB
80A1h	Die aktivierte Zeit liegt in der Vergangenheit; Fehler tritt nur bei einmaliger Ausführung auf.

16.1.22.5 SFC 31 - QRY_TINT - Uhrzeitalarm abfragen

Mit dem SFC 31 QRY_TINT (query time-of-day interrupt) kann der Status des angegebenen Uhrzeitalarm-Organisationsbausteins am Ausgangsparameter STATUS angezeigt werden.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
OB_NR	INPUT	INT	E, A, M, D, L, Konstante	Nummer des OBs, dessen Status abgefragt wird (OB 10, OB 11)
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
STATUS	OUTPUT	WORD	E, A, M, D, L	Status des Uhrzeitalarms

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Es ist kein Fehler aufgetreten.
8090h	Fehlerhafter Parameter *OB_NR*

STATUS

Bit	Wert	Beschreibung
0	0	Uhrzeitalarm ist vom Betriebssystem freigegeben.
1	0	Neue Uhrzeitalarme werden nicht verworfen.
2	0	Uhrzeitalarm ist nicht aktiviert oder abgelaufen.
3	-	reserviert
4	0	Uhrzeitalarm-OB ist nicht geladen.
5	0	Die Ausführung des Uhrzeitalarm-OBs ist durch eine laufende Testfunktion gesperrt.

16.1.23 SFC 32 - SRT_DINT - Verzögerungsalarm starten

Beschreibung

Mit dem SFC 32 SRT_DINT (start time-delay interrupt) kann ein Verzögerungsalarm gestartet werden, der nach Ablauf der parametrierten Verzögerungszeit (Parameter *DTIME*) einen Verzögerungsalarm-OB aufruft. Am Parameter *SIGN* kann ein anwenderspezifisches Kennzeichen angeben werden, um den Start des Verzögerungsalarms zu kennzeichnen. Während der Ausführung erscheinen die Werte von *DTIME* und *SIGN* in der Startereignisinformation des angegebenen OB.

Voraussetzungen

Damit ein Verzögerungsalarm aufgerufen werden kann, müssen folgende Voraussetzungen erfüllt werden:

- Der Verzögerungsalarm-OB muss gestartet sein (mit dem SFC 32).
- Der Verzögerungsalarm-OB darf nicht abgewählt worden sein.
- Der Verzögerungsalarm-OB muss in der CPU vorhanden sein.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
OB_NR	INPUT	INT	E, A, M, D, L, Konstante	Nummer des OB, der nach einer Verzögerungszeit gestartet wird (OB 20, OB 21)
DTIME	INPUT	TIME	E, A, M, D, L, Konstante	Zeitwert der Verzögerung (1 ... 60 000 ms)
SIGN	INPUT	WORD	E, A, M, D, L, Konstante	Kennzeichen, das beim Aufruf des Verzögerungsalarm-OBs in der Startereignisinformation des OBs erscheint
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Genauigkeit

Die Zeit zwischen dem Aufruf des SFC 32 und dem Start des Verzögerungsalarm-OBs ist maximal eine Millisekunde kleiner als die parametrierte Zeit, sofern keine Unterbrechungsereignisse den Aufruf verzögern.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Es ist kein Fehler aufgetreten.
8090h	Fehlerhafter Parameter *OB_NR*
8091h	Fehlerhafter Parameter *DTIME*

16.1.24 SFC 33 - CAN_DINT - Verzögerungsalarm stornieren

Beschreibung

Mit dem SFC 33 CAN_DINT (cancel time-delay interrupt) kann ein bereits gestarteter Verzögerungsalarm storniert werden. In diesem Fall wird der Verzögerungsalarm-OB nicht aufgerufen.

Voraussetzungen

Damit ein Verzögerungsalarm aufgerufen werden kann, müssen folgende Voraussetzungen erfüllt werden:

- Der Verzögerungsalarm-OB muss gestartet sein (mit dem SFC 32).
- Der Verzögerungsalarm-OB darf nicht abgewählt worden sein.
- Der Verzögerungsalarm-OB muss in der CPU vorhanden sein.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
OB_NR	INPUT	INT	E, A, M, D, L, Konstante	Nummer des OBs, der storniert werden soll (OB 20, OB 21).
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Es ist kein Fehler aufgetreten.
8090h	Fehlerhafter Parameter *OB_NR*
80A0h	Verzögerungsalarm ist nicht gestartet.

16.1.25 SFC 34 - QRY_DINT - Verzögerungsalarm Status abfragen

Beschreibung

Mit dem SFC 34 QRY_DINT (query time-delay interrupt) kann der Zustand eines Verzögerungsalarms am Ausgangsparameter *STATUS* angezeigt werden.

Voraussetzungen

Damit ein Verzögerungsalarm aufgerufen werden kann, müssen folgende Voraussetzungen erfüllt werden:

- Der Verzögerungsalarm-OB muss gestartet sein (mit dem SFC 32).
- Der Verzögerungsalarm-OB darf nicht abgewählt worden sein.
- Der Verzögerungsalarm-OB muss in der CPU vorhanden sein.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
OB_NR	INPUT	INT	E, A, M, D, L, Konstante	Nummer des OB, dessen Zustand abgefragt wird (OB 20, OB 21)
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustand des Verzögerungsalarms

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Es ist kein Fehler aufgetreten.
8090h	Fehlerhafter Parameter *OB_NR*

STATUS

Bit	Wert	Beschreibung
0	0	Verzögerungsalarm ist vom Betriebssystem freigegeben.
1	0	Neue Verzögerungsalarm werden nicht verworfen.
2	0	Verzögerungsalarm ist nicht aktiviert oder abgelaufen.
3	-	-
4	0	Verzögerungsalarm -OB ist nicht geladen.
5	0	Die Ausführung des Verzögerungsalarm-OBs ist durch eine laufende Testfunktion gesperrt.

16.1.26 SFC 36 - MSK_FLT - Synchronfehlerereignisse maskieren

Beschreibung

Mit dem SFC 36 MSK_FLT (mask synchronous faults) können Sie die Reaktion der CPU auf Synchronfehlerereignisse steuern, indem die jeweiligen Synchronfehlerereignisse maskiert werden.

Mit dem Aufruf des SFC 36 werden die Synchronfehlerereignisse in der aktuellen Prioritätsklasse maskiert. Wenn in den Eingangsparametern einzelne Bits der Synchronfehlermasken auf "1" gesetzt werden, dann behalten andere, vorher bereits gesetzte Bits ihre Wertigkeit "1". Dadurch erhält man neue Fehlermasken, die über die Ausgangsparameter ausgelesen werden können. Die maskierten Synchronfehlerereignisse werden in einem Ereignisstatusregister eingetragen und rufen keinen OB auf. Das Auslesen des Ereignisstatusregisters erfolgt mit dem SFC 38 READ_ERR.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
PRGFLT_SET_MASK	INPUT	DWORD	E, A, M, D, L, Konstante	Zu maskierende Programmierfehler
ACCFLT_SET_MASK	INPUT	DWORD	E, A, M, D, L, Konstante	Zu maskierende Zugriffsfehler
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
PRGFLT_MASKED	OUTPUT	DWORD	E, A, M, D, L	Maskierte Programmierfehler
ACCFLT_MASKED	OUTPUT	DWORD	E, A, M, D, L	Maskierte Zugriffsfehler

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Keiner der Fehler war bereits maskiert.
0001h	Mindestens einer der Fehler war bereits maskiert, trotzdem werden die anderen Fehler maskiert.

16.1.27 SFC 37 - DMSK_FLT - Synchronfehlerereignisse demaskieren

Beschreibung

Mit dem SFC 37 DMSK_FLT (unmask synchronous faults) werden bereits maskierte Synchronfehlerereignisse demaskiert. Mit dem Aufruf des SFC 37 werden die Synchronfehlerereignisse in der aktuellen Prioritätsklasse demaskiert. Hierzu werden in den Eingangsparametern die entsprechenden Bits der Fehlermasken auf "1" gesetzt. Dadurch erhält man neue Fehlermasken, die über die Ausgangsparameter ausgelesen werden können. Die abgefragten Einträge im Ereignisstatusregister werden gelöscht.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
PRGFLT_RESET_MASK	INPUT	DWORD	E, A, M, D, L, Konstante	Zu demaskierende Programmierfehler
ACCFLT_RESET_MASK	INPUT	DWORD	E, A, M, D, L, Konstante	Zu demaskierende Zugriffsfehler
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
PRGFLT_MASKED	OUTPUT	DWORD	E, A, M, D, L	Noch maskierte Programmierfehler
ACCFLT_MASKED	OUTPUT	DWORD	E, A, M, D, L	Noch maskierte Zugriffsfehler

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Alle angegebenen Fehler wurden demaskiert.
0001h	Mindestens einer der Fehler war nicht maskiert, trotzdem werden die anderen Fehler demaskiert.

16.1.28 SFC 38 - READ_ERR - Ereignisstatusregister lesen

Beschreibung

Mit dem SFC 38 READ_ERR (read error registers) wird das Ereignisstatusregister ausgelesen. Im Aufbau entspricht das Ereignisstatusregister den Programmier- und Zugriffsfehlermasken, die mit dem SFC 36 und 37 als Eingangsparameter programmiert wurden. Mit dem Aufruf des SFC 38 werden die gewünschten Einträge aus dem Ereignisstatusregister gelesen und gleichzeitig gelöscht. Über die Eingangsparameter wird festgelegt, welche Synchronfehlerereignisse aus dem Ereignisstatusregister abgefragt werden sollen. Sie erhalten die Information, welche der maskierten Synchronfehler in der aktuellen Prioritätsklasse mindestens einmal aufgetreten sind. Ein gesetztes Bit bedeutet, dass der zugehörige maskierte Synchronfehler mindestens einmal aufgetreten ist.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
PRGFLT_QUERY	INPUT	DWORD	E, A, M, D, L, Konstante	Abfrage der Programmierfehler
ACCFLT_QUERY	INPUT	DWORD	E, A, M, D, L, Konstante	Abfrage der Zugriffsfehler

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
PRGFLT_ESR	OUTPUT	DWORD	E, A, M, D, L	Aufgetretene Programmierfehler
ACCFLT_ESR	OUTPUT	DWORD	E, A, M, D, L	Aufgetretene Zugriffsfehler

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Alle abgefragten Fehler sind maskiert.
0001h	Mindestens einer der abgefragten Fehler ist nicht maskiert.

16.1.29 SFC 39 - DIS_IRT - Alarmereignisse sperren

Beschreibung

Mit dem SFC 39 DIS_IRT (disable interrupt) sperren Sie die Bearbeitung neuer Alarm- und Asynchronfehlerereignisse. Sperren heißt, dass das Betriebssystem der CPU bei einem Unterbrechungsereignis

- weder einen Alarm-OB bzw. einen Asynchronfehler-OB aufruft,
- noch die festgelegte Reaktion bei nicht programmiertem Alarm-OB bzw. Asynchronfehler-OB auslöst.

Die Sperre der Bearbeitung von Alarm- und Asynchronfehlerereignissen bleibt über alle Prioritätsklassen gültig. Die Sperre kann nur mit dem SFC 40 aufgehoben werden bzw. auch bei Neustart.

Ob das Betriebssystem das aufgetretene Alarm- bzw. Asynchronfehlerereignis in den Diagnosepuffer einträgt, ist abhängig von Ihrer Wahl des Eingangsparameters *MODE*.

Bitte beachten Sie, dass bei der Programmierung des SFC 39 alle eintreffenden Alarme verworfen werden.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
MODE	INPUT	BYTE	E, A, M, D, L, Konstante	Gibt an, welche Alarm- und Asynchronfehlerereignisse gesperrt werden.
OB_NR	INPUT	INT	E, A, M, D, L, Konstante	OB-Nummer
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

MODE

MODE	Beschreibung
00	Alle neu aufgetretenen Alarm- und Asynchronfehlerereignisse werden gesperrt (nicht gesperrt werden Synchronfehlerereignisse).
01	Alle neu auftretenden Ereignisse einer angegebenen Alarmklasse werden gesperrt. Die Alarmklasse kennzeichnen Sie wie folgt: ■ Uhrzeitalarme: 10 ■ Verzögerungsalarme: 20 ■ Weckalarme: 30 ■ Prozessalarme: 40 ■ Alarme für DP-V1: 50 ■ Asynchrone Fehleralarme: 80 Die Einträge in den Diagnosepuffer erfolgen weiterhin.
02	Alle neu auftretenden Ereignisse eines angegebenen Alarms werden gesperrt. Den Alarm kennzeichnen Sie durch die OB-Nummer. Die Einträge in den Diagnosepuffer erfolgen weiterhin.
80	Alle neu auftretenden Alarm- und Asynchronfehlerereignisse werden gesperrt und sie werden auch nicht mehr in den Diagnosepuffer eingetragen. Das Betriebssystem trägt das Ereignis 5380h in den Diagnosepuffer ein.
81	Alle neu auftretenden Ereignisse einer angegebenen Alarmklasse werden gesperrt und sie werden auch nicht mehr in den Diagnosepuffer eingetragen. Das Betriebssystem trägt das Ereignis 5380h in den Diagnosepuffer ein.
82	Alle neu auftretenden Ereignisse einer angegebenen Alarms werden gesperrt und sie werden auch nicht mehr in den Diagnosepuffer eingetragen. Das Betriebssystem trägt das Ereignis 5380h in den Diagnosepuffer ein.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Es ist kein Fehler aufgetreten.
8090h	Der Eingangsparameter *OB_NR* enthält einen unzulässigen Wert.
8091h	Der Eingangsparameter *MODE* enthält einen unzulässigen Wert.
8xyyh	Allgemeine Fehlerinformation, *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

16.1.30 SFC 40 - EN_IRT - Gesperrte Alarmereignisse freigeben

Beschreibung

Mit dem SFC 40 EN_IRT (enable interrupt) geben Sie die mit dem SFC 39 gesperrte Bearbeitung neuer Alarm- und Asynchronfehlerereignisse wieder frei. Freigeben heißt, dass das Betriebssystem der CPU bei einem Unterbrechungsereignis:

- einen Alarm-OB bzw. einen Asynchronfehler-OB aufruft,
  oder
- die festgelegte Reaktion bei nicht programmiertem Alarm-OB bzw. Asynchronfehler-OB auslöst.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
MODE	INPUT	BYTE	E, A, M, D, L, Konstante	Gibt an, welche Alarm- und Asynchronfehlerereignisse freigegeben werden.
OB_NR	INPUT	INT	E, A, M, D, L, Konstante	OB-Nummer
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

MODE

MODE	Beschreibung
00	Alle neu aufgetretenen Alarm- und Asynchronfehlerereignisse werden freigegeben.
01	Alle neu auftretenden Ereignisse einer angegebenen Alarmklasse werden freigegeben. Die Alarmklasse kennzeichnen Sie wie folgt: ■ Uhrzeitalarme: 10 ■ Verzögerungsalarme: 20 ■ Weckalarme: 30 ■ Prozessalarme: 40 ■ Alarme für DP-V1: 50 ■ Asynchrone Fehleralarme: 80
02	Alle neu auftretenden Ereignisse eines angegebenen Alarms werden freigegeben. Den Alarm kennzeichnen Sie durch die OB-Nummer.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Es ist kein Fehler aufgetreten.
8090h	Der Eingangsparameter *OB_NR* enthält einen unzulässigen Wert.
8091h	Der Eingangsparameter *MODE* enthält einen unzulässigen Wert.
8xyyh	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

16.1.31 SFC 41 - DIS_AIRT - Alarmereignisse verzögern

Beschreibung

Mit dem SFC 41 DIS_AIRT (disable alarm interrupts) wird die Bearbeitung von Alarm-OBs und Asynchronfehler-OBs, deren Priorität höher ist als die des aktuellen OBs verzögert. Innerhalb eines OBs kann der SFC 41 mehrmals aufgerufen werden, wobei die Aufrufe vom Betriebssystem gezählt werden. Die Bearbeitungsverzögerung ist solange gültig, bis mit dem SFC 42 EN_AIRT jede mit einem SFC 41 verzögerte Bearbeitung von Alarm-OBs und Asynchronfehler-OBs aufgehoben wird oder der aktuelle OB abgearbeitet ist.

Die Bearbeitung der anstehenden Alarm- oder Asynchronfehlerereignisse findet statt, sobald die Bearbeitungsverzögerung mit dem SFC 42 EN_AIRT aufgehoben wurde oder die aktuelle OB-Bearbeitung beendet ist.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
RET_VAL	OUTPUT	INT	E, A, M, D, L	Anzahl der Verzögerungen (= Anzahl der Aufrufe des SFC 41)

RET_VAL (Rückgabewert) Der Rückgabewert *RET_VAL* zeigt nach Ablauf des SFC die Anzahl der Bearbeitungsverzögerungen, also der Aufrufe des SFC 41 an. (Die Alarmbearbeitung ist erst wieder freigegeben, wenn *RET_VAL* = 0 ist.)

16.1.32 SFC 42 - EN_AIRT - Verzögerte Alarmereignissen freigeben

Beschreibung Mit dem SFC 42 EN_AIRT (enable alarm interrupts) wird die mit dem SFC 41 verzögerte Bearbeitung von höherprioren Alarm- bzw. Asynchronfehlerereignissen wieder freigegeben.

Dabei muss jede einzelne Bearbeitungsverzögerung mit dem SFC 42 beendet werden. Wenn also z.B. mit 5 SFC 41-Aufrufen 5 verschiedene Alarme verzögert wurden, so muss mit 5 SFC 42-Aufrufen jede dieser Alarmverzögerungen auch wieder aufgehoben werden.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
RET_VAL	OUTPUT	INT	E, A, M, D, L	Anzahl der noch programmierten Verzögerungen nach Ablauf des SFC 42 bzw. wenn während der Bearbeitung der Funktion ein Fehler auftritt, enthält der Rückgabewert einen Fehlercode.

RET_VAL (Rückgabewert) Der Rückgabewert *RET_VAL* zeigt nach Ablauf des SFC die Anzahl der Bearbeitungsverzögerungen, also der Aufrufe des SFC 41 an. (Die Alarmbearbeitung ist erst wieder freigegeben, wenn *RET_VAL* = 0 ist.)

Wert	Beschreibung
8080h	Obwohl die Alarmbearbeitung bereits freigegeben war, wurde die Funktion aufgerufen.

16.1.33 SFC 43 - RE_TRIGR - Zykluszeitüberwachung neu starten

Beschreibung Mit dem SFC 43 RE_TRIGR (retrigger watchdog) wird die Zykluszeitüberwachung der CPU neu gestartet.

Parameter und Rückmeldungen Der SFC 43 besitzt weder Parameter noch werden Fehlerinformationen zurückgeliefert.

16.1.34 SFC 44 - REPL_VAL - Ersatzwert in AKKU1 übertragen

Beschreibung

Mit dem SFC 44 REPL_VAL (replace value) wird ein Wert in den AKKU1 der fehlerverursachenden Programmebene übertragen. Der SFC 44 darf nur in Synchronfehler-OBs (OB 121, OB 122) aufgerufen werden.

Anwendungsbeispiel des SFC 44:

Wenn an einem Eingabemodul eine Störung vorhanden ist, die dazu führt, dass von dem Modul keine Werte mehr gelesen werden können, so wird nach jedem Zugriffsversuch auf dieses Modul der OB 122 gestartet. Mit Hilfe des SFC 44 REPL_VAL kann im OB 122 ein geeigneter Ersatzwert in den AKKU1 der unterbrochenen Programmebene übertragen werden, wobei die Programmbearbeitung dann mit diesem Ersatzwert fortgesetzt wird. Die Informationen, die für die Auswahl des Ersatzwertes notwendig sind (z.B. Baustein, in dem der Fehler auftrat, betroffene Adresse), können den lokalen Variablen des OB 122 entnommen werden.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
VAL	INPUT	DWORD	E, A, M, D, L, Konstante	Ersatzwert
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Es ist kein Fehler aufgetreten. Ein Ersatzwert wurde eingegeben.
8080h	Der SFC 44 wurde nicht von einem Synchronfehler-OB (OB 121, OB 122) aus aufgerufen.

16.1.35 SFC 46 - STP - CPU in STOP überführen

Beschreibung

Mit dem SFC 46 STP wird die CPU in den Betriebszustand STOP überführt.

Parameter und Rückmeldungen

Der SFC 46 besitzt weder Parameter noch werden Fehlerinformationen zurückgeliefert.

16.1.36 SFC 47 - WAIT - Verzögern des Anwenderprogramms

Beschreibung

Mit dem SFC 47 WAIT werden Zeitverzögerungen bzw. Wartezeiten von 1 bis zu 32767µs in Ihrem Anwenderprogramm programmiert.

Unterbrechbarkeit

Der SFC 47 kann durch höherpriore OBs unterbrochen werden.

Bei der mit dem SFC 47 programmierten Verzögerungszeit handelt es sich um eine Mindestzeit, die um die Ausführungszeit der eingeschachtelten Prioritätsklassen sowie um Systemlasten verlängert werden kann!

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
WT	INPUT	INT	E, A, M, D, L, Konstante	Der Parameter *WT* enthält die Verzögerungszeit in µs.

Fehlerinformationen Der SFC 47 gibt keine spezifischen Fehlerinformationen aus.

16.1.37 SFC 49 - LGC_GADR - Steckplatz ermitteln

Beschreibung Mit dem SFC 49 LGC_GADR (convert logical address to geographical address) wird der zu einer logischen Adresse gehörende Modulsteckplatz sowie der Offset im Nutzdatenadressraum des Moduls ermittelt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IOID	INPUT	BYTE	E, A, M, D, L, Konstante	Kennung des Adressbereichs: 54h = Peripherie-Eingang (PE) 55h = Peripherie-Ausgang (PA) Handelt es sich um ein Mischmodul, ist die Bereichskennung der niedrigeren Adresse anzugeben. Bei gleichen Adressen ist 54h anzugeben.
LADDR	INPUT	WORD	E, A, M, D, L, Konstante	Logische Adresse. Bei einem Mischmodul ist die kleinere der beiden Adressen anzugeben.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
AREA	OUTPUT	BYTE	E, A, M, D, L	Bereichskennung: Sie gibt an, wie die restlichen Ausgangsparameter zu interpretieren sind.
RACK	OUTPUT	WORD	E, A, M, D, L	Siehe *AREA* unten
SLOT	OUTPUT	WORD	E, A, M, D, L	
SUBADDR	OUTPUT	WORD	E, A, M, D, L	

AREA *AREA* gibt an, wie die Ausgangsparameter *RACK*, *SLOT* und *SUBADDR* zu interpretieren sind. Diese Abhängigkeit ist nachfolgend aufgeführt.

Wert von *AREA*	System	Bedeutung von *RACK*, *SLOT* und *SUBADDR*
0	-	reserviert
1	Siemens S7-300	*RACK*: Nummer Baugruppenträger *SLOT*: Nummer Steckplatz *SUBADDR*: Adressoffset zur Anfangsadresse

Wert von *AREA*	System	Bedeutung von *RACK*, *SLOT* und *SUBADDR*
2	Dezentrale Peripherie	*RACK* (Low-Byte): Stationsnummer *RACK* (High-Byte): DP-Mastersystem-ID *SLOT*: Nummer Steckplatz in Station *SUBADDR*: Adressoffset zur Anfangsadresse
3 ... 6	-	reserviert

RET_VAL (Rückgabewert) Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Wert	Beschreibung
0000h	Es ist kein Fehler aufgetreten.
8090h	Angegebene logische Adresse ungültig oder unzulässiger Wert beim Parameter *IOID*.

16.1.38 SFC 50 - RD_LGADR - Alle Adressen eines Moduls lesen

Beschreibung

Mit dem SFC 50 RD_LGADR (read module logical addresses) werden alle vereinbarten logischen Adressen eines Moduls ermittelt, ausgehend von einer logischen Adresse dieses Moduls.

Die Zuordnung von logischen Adressen zum Modul muss vorher von Ihnen projektiert worden sein. Die ermittelten logischen Adressen werden in das Feld *PEADDR* bzw. in das Feld *PAADDR* in aufsteigender Reihenfolge eingetragen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IOID	INPUT	BYTE	E, A, M, D, L, Konstante	Bereichskennung: 54h = Peripherie-Eingang (PE) 55h = Peripherie-Ausgang (PA)
LADDR	INPUT	WORD	E, A, M, D, L, Konstante	Eine logische Adresse
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
PEADDR	OUTPUT	ANY	E, A, M, D, L	Feld für die PE-Adressen, Feldelemente müssen vom Datentyp WORD sein.
PECOUNT	OUTPUT	INT	E, A, M, D, L	Anzahl der zurückgelieferten PE-Adressen
PAADDR	OUTPUT	ANY	E, A, M, D, L	Feld für die PA-Adressen, Feldelemente müssen vom Datentyp WORD sein.
PACOUNT	OUTPUT	INT	E, A, M, D, L	Anzahl der zurückgelieferten PA-Adressen

RET_VAL (Rückgabewert) Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Wert	Beschreibung
0000h	Es ist kein Fehler aufgetreten.
8090h	Angegebene logische Adresse ungültig oder unzulässiger Wert beim Parameter *IOID*.
80A0h	Fehler beim Ausgangsparameter *PEADDR*: Der Datentyp der Feldelemente ist nicht WORD.
80A1h	Fehler beim Ausgangsparameter *PAADDR*: Der Datentyp der Feldelemente ist nicht WORD.
80A2h	Fehler beim Ausgangsparameter *PEADDR*: Das angegebene Feld konnte nicht alle logischen Adressen aufnehmen.
80A3h	Fehler beim Ausgangsparameter *PAADDR*: Das angegebene Feld konnte nicht alle logischen Adressen aufnehmen.

16.1.39 SFC 51 - RDSYSST - Auslesen der Informationen der SZL

Beschreibung

Mit dem SFC 51 RDSYSST (read system status) können Sie eine Teilliste bzw. einen Teillistenauszug der SZL (**S**ystem**z**ustands**l**iste) anfordern. Hierbei bestimmen Sie mit den Parametern *SZL_ID* und *INDEX* was Sie auslesen möchten.

Der *INDEX* ist nicht immer erforderlich. Er dient der Bestimmung eines Objekts innerhalb einer Teilliste.

Durch Setzen von *REQ* starten Sie die Abfrage. Sobald *BUSY* = 0 zurückgemeldet wird, liegen die Daten im Zielbereich DR ab.

Informationen zur SZL finden Sie im Teil "Systemzustandsliste SZL".

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	*REQ* = 1: Anstoß der Bearbeitung
SZL_ID	INPUT	WORD	E, A, M, D, L, Konstante	*SZL_ID* der Teilliste oder des Teillistenauszugs
INDEX	INPUT	WORD	E, A, M, D, L, Konstante	Typ oder Nummer eines Objekts in einer Teilliste
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: Lesevorgang noch nicht abgeschlossen

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
SZL_HEADER	OUTPUT	STRUCT	D, L	Struktur mit 2 WORD-Typen: LENGTHDR: Länge Datensatz WORDN_DR: Anzahl vorhandener zugehöriger Datensätze (bei Zugriff auf Teillistenkopfinfo) oder Anzahl der in DR übertragenen Datensätze.
DR	OUTPUT	ANY	E, A, M, D, L	Zielbereich für die gelesene SZL-Teilliste bzw. den gelesenen SZL-Teillistenauszug: Falls Sie nur die SZL-Teillistenkopfinfo einer SZL-Teilliste ausgelesen haben, dürfen Sie DR nicht auswerten, sondern nur *SZL_HEADER*. Andernfalls gibt das Produkt aus LENGTHDR und N_DR an, wie viele Bytes in DR eingetragen wurden.

RET_VAL (Rückgabewert) Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Wert	Beschreibung
0000h	kein Fehler
0081h	Länge des Ergebnisfeldes zu klein. Es werden trotzdem so viele Datensätze wie möglich geliefert. Der SZL-Header zeigt diese Anzahl an.
7000h	Erstaufruf mit *REQ* = 0: keine Datenübertragung aktiv; *BUSY* = 0.
7001h	Erstaufruf mit *REQ* = 1: Datenübertragung angestoßen; *BUSY* = 1.
7002h	Zwischenaufruf (*REQ* irrelevant): Datenübertragung bereits aktiv; *BUSY* = 1.
8081h	Länge des Ergebnisfeldes zu klein. Platz reicht nicht für einen Datensatz.
8082h	*SZL_ID* ist falsch oder in der CPU bzw. in dem SFC unbekannt.
8083h	*INDEX* ist falsch oder nicht erlaubt.
8085h	Die Information ist systembedingt momentan nicht verfügbar, z.B. wegen Ressourcenmangels.
8086h	Datensatz ist nicht lesbar wegen eines Systemfehlers.
8087h	Datensatz ist nicht lesbar, weil das Modul nicht vorhanden ist oder nicht quittiert.
8088h	Datensatz ist nicht lesbar, weil die tatsächliche Typkennung von der Solltypkennung abweicht.
8089h	Datensatz ist nicht lesbar, weil das Modul nicht diagnosefähig ist.
80A2h	DP-Protokollfehler - Layer-2-Fehler (temporärer Fehler).
80A3h	DP-Protokollfehler bei User-Interface/User (temporärer Fehler)
80A4h	Kommunikation am Bus ist gestört. Fehler tritt auf zwischen CPU und externer DP-Anschaltung (temporärer Fehler).
80C5h	Dezentrale Peripherie nicht verfügbar (temporärer Fehler).

16.1.40 SFC 52 - WR_USMSG - Eintrag in Diagnosepuffer schreiben

Beschreibung

Mit dem SFC 52 WR_USMSG (write user element in diagnosis buffer) wird ein anwenderdefiniertes Diagnoseereignis in den Diagnosepuffer geschrieben.

Diagnosemeldung senden

Um zu prüfen, ob das Senden von anwenderdefinierten Diagnosemeldungen momentan möglich ist, rufen Sie den SFC 51 "RDSYSST" mit den Parametern *SZL_ID* = 0132h und *INDEX* = 0005h auf. Enthält das vierte Wort des dabei gelieferten Datensatzes den Wert 1 ist das Senden möglich, enthält es den Wert 0 ist das Senden nicht möglich.

Sendepuffer voll

Die Diagnosemeldung kann nur dann in den Sendepuffer eingetragen werden, wenn der Sendepuffer nicht voll ist. Maximal können 50 Einträge im Sendepuffer abgelegt werden.

Ist der Sendepuffer voll, dann:

- erfolgt trotzdem der Eintrag des Diagnoseereignisses in den Diagnosepuffer.
- wird die entsprechende Fehlermeldung (8092h) im Parameter RET_VAL angezeigt.

Teilnehmer nicht angemeldet

Wenn eine anwenderdefinierte Diagnosemeldung gesendet werden soll und kein Teilnehmer angemeldet ist, dann

- erfolgt trotzdem der Eintrag des anwenderdefinierten Diagnoseereignisses in den Diagnosepuffer,
- wird die entsprechende Fehlermeldung (0091h oder 8091h) im Parameter *RET_VAL* angezeigt.

Aufbau eines Eintrags

Ein Eintrag in den Diagnosepuffer hat intern den folgenden Aufbau:

Byte	Inhalt
1, 2	Ereignis-ID
3	Prioritätsklasse
4	OB-Nummer
5, 6	reserviert
7, 8	Zusatzinformation 1
9, 10, 11, 12	Zusatzinformation 2
13 ... 20	Zeitstempel: Der Zeitstempel ist vom Datentyp Date_and_Time.

Ereignis-ID

Jedem Ereignis ist eine Ereignis-ID zugeordnet.

Zusatzinformationen

Die Zusatzinformationen beinhalten zusätzliche Informationen zum Ereignis. Diese Zusatzinformationen können für jedes Ereignis unterschiedlich sein. Wenn ein Diagnoseereignis erzeugt wird, dann kann der Inhalt dieser Einträge selbst bestimmt werden.

Wenn eine anwenderdefinierte Diagnosemeldung versendet wird, können die Zusatzinformationen als Begleitwerte in den (Ereignis-ID-spezifischen) Meldetext integriert werden.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
SEND	INPUT	BOOL	E, A, M, D, L, Konstante	Senden der anwenderdefinierten Diagnosemeldung an alle angemeldeten Teilnehmer freigeben
EVENTN	INPUT	WORD	E, A, M, D, L, Konstante	Ereignis-ID. Sie vergeben die Ereignis-ID. Die Vorgabe erfolgt nicht durch den Meldeserver.
INFO1	INPUT	ANY	E, A, M, D, L	Zusatzinformation 1 Wort lang
INFO2	INPUT	ANY	E, A, M, D, L	Zusatzinformation 2 Worte lang
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

SEND

Wenn *SEND* den Wert 1 hat, wird die anwenderdefinierte Diagnosemeldung an alle dafür angemeldeten Teilnehmer gesendet. Das Senden erfolgt nur unter der Voraussetzung, dass mindestens ein Teilnehmer angemeldet und der Sendepuffer nicht voll ist. Das Senden erfolgt asynchron zum Anwenderprogramm.

EVENTN

In *EVENTN* wird die Ereignis-ID des Anwenderereignisses eingetragen. Die Ereignis-IDs dürfen nur in der Form 8xyzh , 9xyzh, Axyzh und Bxyzh eingetragen werden, wobei die IDs der Form 8xyzh und 9xyzh zu den vordefinierten Ereignissen und die IDs der Form Axyzh und Bxyzh zu den frei definierten Ereignissen gehören.

Ein kommendes Ereignis wird durch x = 1,

ein gehendes Ereignis durch x = 0 gekennzeichnet.

Bei den Ereignissen der Klasse A und B ist yz die in der Meldungsprojektierung für die zugehörige Meldung vergebene Meldungsnummer in hexadezimaler Darstellung.

INFO1

INFO1 enthält eine Information, die ein Wort lang ist. Folgende Datentypen sind zulässig:

- WORD
- INT
- ARRAY [0...1] OF CHAR

INFO1 kann als Begleitwert in den Meldetext integriert werden und damit können der Meldung aktuelle Informationen hinzugefügt werden.

INFO2

INFO2 enthält eine Information, die zwei Worte lang ist. Folgende Datentypen sind zulässig:

- DWORD
- DINT
- REAL
- TIME
- ARRAY [0...3] OF CHAR

INFO2 kann als Begleitwert in den Meldetext integriert werden und damit können der Meldung aktuelle Informationen hinzugefügt werden.

RET_VAL (Rückgabewert)

Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Wert	Beschreibung
0000h	kein Fehler
0091h	kein Teilnehmer angemeldet (Eintrag des Diagnoseereignisses in den Diagnosepuffer erfolgt)
8083h	Datentyp *INFO1* nicht zulässig
8084h	Datentyp *INFO2* nicht zulässig
8085h	*EVENTN* nicht zulässig
8086h	Länge von *INFO1* nicht zulässig
8087h	Länge von *INFO2* nicht zulässig
8091h	Fehlermeldung identisch mit Fehlercode 0091h
8092h	Senden derzeit nicht möglich, Sendepuffer ist voll (Eintrag des Diagnoseereignisses in den Diagnosepuffer erfolgt)

16.1.41 FC/SFC 54 - RD_DPARM - Vordefinierte Parameter lesen

Beschreibung

Mit dem SFC 54 RD_DPARM (read defined parameter) wird der Datensatz mit der Nummer *RECNUM* des adressierten Moduls aus dem zugehörigen SDB1xy gelesen.

Durch den Parameter *RECORD* wird der Zielbereich festgelegt, in den der gelesene Datensatz eingetragen wird.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IOID	INPUT	BYTE	E, A, M, D, L, Konstante	Kennung des Adressbereichs: 54h = Peripherie Eingang (PE) 55h = Peripherie Ausgang (PA) Handelt es sich um ein Mischmodul, ist die Bereichskennung der niedrigeren Adresse anzugeben. Bei gleichen Adressen ist 54h anzugeben.
LADDR	INPUT	WORD	E, A, M, D, L, Konstante	Logische Basisadresse des Moduls. Bei einem Mischmodul ist die kleinere der beiden Adressen anzugeben.
RECNUM	INPUT	BYTE	E, A, M, D, L, Konstante	Datensatznummer (zulässige Werte: 0 ... 240)
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.Zusätzlich: Länge des gelesenen Datensatzes in Bytes, falls der gelesene Datensatz in den Zielbereich passt und bei der Übertragung kein Fehler auftrat.
RECORD	OUTPUT	ANY	E, A, M, D, L	Zielbereich für den gelesenen Datensatz. Es ist nur der Datentyp BYTE zulässig.

RET_VAL (Rückgabewert)

Bei *RET_VAL* = 8xxxh sind zwei Fälle zu unterscheiden:

- Temporäre Fehler (Fehlercodes 80A2h ... 80A4h, 80Cxh):
  Bei dieser Fehlerart besteht die Möglichkeit, dass sich der Fehler ohne Ihr Zutun behebt, es wäre also sinnvoll, den SFC erneut (ggf. mehrfach) aufzurufen.
  Beispiel für temporäre Fehler: Benötigte Betriebsmittel sind momentan belegt (80C3h).
- Permanente Fehler (Fehlercodes 809xh, 80A1h, 80Bxh, 80Dxh):
  Bei dieser Fehlerart kann der Fehler nicht ohne Ihr Zutun behoben werden. Ein erneuter Aufruf des SFC ist erst wieder sinnvoll, wenn der Fehler beseitigt wurde.
  Beispiel für permanente Fehler: Falsche Länge des zu übertragenden Datensatzes (80B1h).

Wert	Beschreibung
7000h	Erstaufruf mit *REQ* = 0: keine Datenübertragung aktiv; *BUSY* hat den Wert 0.
7001h	Erstaufruf mit *REQ* = 1: Datenübertragung angestoßen; *BUSY* hat den Wert 1.
7002h	Zwischenaufruf (*REQ* irrelevant): Datenübertragung bereits aktiv; *BUSY* hat den Wert 1.
8090h	Angegebene logische Basisadresse ungültig: Es ist keine Zuordnung im SDB1/SDB2x vorhanden, oder es ist keine Basisadresse.
8092h	In ANY-Referenz ist eine Typangabe ungleich BYTE angegeben.
8093h	Für das über *LADDR* und *IOID* ausgewählte Modul ist dieser SFC nicht zulässig.
80B1h	Die Länge des durch *RECORD* festgelegten Zielbereichs ist zu klein.
80D0h	Im zugehörigen SDB ist kein Eintrag für das Modul vorhanden.
80D1h	Die Datensatznummer ist im zugehörigen SDB für das Modul nicht projektiert.
80D2h	Das Modul ist laut Typkennung nicht parametrierbar.
80D3h	Auf den SDB kann nicht zugegriffen werden, da er nicht vorhanden ist.
80D4h	SDB-Strukturfehler: SDB-interner Zeiger zeigt außerhalb SDB.

16.1.42 SFC 55 - WR_PARM - Dynamische Parameter schreiben

Beschreibung

Mit dem SFC 55 WR_PARM (write parameter) wird der Datensatz *RECORD* zum adressierten Modul übertragen. Die Parameter dieses Moduls im zugehörigen SDB werden, falls diese dort vorhanden sind, nicht durch die Parameter, die zum Modul übertragen werden, überschrieben.

Dieser SFC kann für Digital-, Analogmodule, FMs, CPs und über PROFIBUS DP-V1 eingesetzt werden.

Voraussetzungen

Es ist darauf zu achten das der zu übertragende Datensatz nicht statisch ist:

- Datensatz 0 darf nicht verwendet werden, da Datensatz 0 systemweit statisch ist.
- Ist der Datensatz in den SDBs 100 ... 129 vermerkt, dann darf das Statisch-Bit nicht gesetzt sein.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	*REQ* = 1: Anforderung zum Schreiben
IOID	INPUT	BYTE	E, A, M, D, L, Konstante	Kennung des Adressbereichs: 54h = Peripherie Eingang (PE) 55h = Peripherie Ausgang (PA) Handelt es sich um ein Mischmodul, ist die Bereichskennung der niedrigeren Adresse anzugeben. Bei gleichen Adressen ist 54h anzugeben.
LADDR	INPUT	WORD	E, A, M, D, L, Konstante	Logische Basisadresse des Moduls. Bei einem Mischmodul ist die kleinere der beiden Adressen anzugeben.
RECNUM	INPUT	BYTE	E, A, M, D, L, Konstante	Datensatznummer (zulässige Werte: 0 ... 240)
RECORD	INPUT	ANY	E, A, M, D, L	Datensatz
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: Der Schreibvorgang ist noch nicht beendet.

RECORD

Beim ersten Aufruf des SFC werden die zu übertragenden Daten aus dem Parameter *RECORD* gelesen. Sollte die Übertragung des Datensatzes jedoch länger als einen Aufruf dauern, dann ist bei den Folgeaufrufen des SFC (zum gleichen Auftrag) der Inhalt des Parameters *RECORD* nicht mehr relevant.

RET_VAL (Rückgabewert)

Bei RET_VAL = 8xxxh sind zwei Fälle zu unterscheiden:

- Temporäre Fehler (Fehlercodes 80A2h ... 80A4h, 80Cxh):
  Bei dieser Fehlerart besteht die Möglichkeit, dass sich der Fehler ohne Ihr Zutun behebt, es wäre also sinnvoll, den SFC erneut (ggf. mehrfach) aufzurufen.
  Beispiel für temporäre Fehler: Benötigte Betriebsmittel sind momentan belegt (80C3h).
- Permanente Fehler (Fehlercodes 809xh, 80A1h, 80Bxh, 80Dxh):
  Bei dieser Fehlerart kann der Fehler nicht ohne Ihr Zutun behoben werden. Ein erneuter Aufruf des SFC ist erst wieder sinnvoll, wenn der Fehler beseitigt wurde.
  Beispiel für permanente Fehler: Falsche Länge des zu übertragenden Datensatzes (80B1h).

Wert	Beschreibung
7000h	Erstaufruf mit *REQ* = 0: keine Datenübertragung aktiv; *BUSY* hat den Wert 0.
7001h	Erstaufruf mit *REQ* = 1: Datenübertragung angestoßen; *BUSY* hat den Wert 1.

Wert	Beschreibung
7002h	Zwischenaufruf (*REQ* irrelevant): Datenübertragung bereits aktiv; *BUSY* hat den Wert 1.
8090h	Angegebene logische Basisadresse ungültig: Es ist keine Zuordnung im SDB1/SDB2x vorhanden, oder es ist keine Basisadresse.
8092h	In ANY-Referenz ist eine Typangabe ungleich BYTE angegeben.
8093h	Für die über *LADDR* und *IOID* ausgewählte Modul ist dieser SFC nicht zulässig.
80A1h	Negative Quittung beim Senden des Datensatzes zum Modul (Modul während des Sendens gezogen oder Modul defekt)
80A2h	DP-Protokollfehler bei Layer 2, evtl. Hardware- / Schnittstellenfehler im DP-Slave
80A3h	DP-Protokollfehler bei User Interface/User
80A4h	Kommunikation gestört (Fehler tritt auf zwischen CPU und externer DP-Anschaltung)
80B0h	SFC für Modultyp nicht möglich oder Modul kennt den Datensatz nicht
80B1h	Die Länge des durch *RECORD* festgelegten Zielbereichs ist zu klein.
80B2h	Der projektierte Steckplatz ist nicht belegt.
80B3h	Ist-Modultyp ungleich Soll-Modultyp im SDB1.
80C1h	Die Daten des auf dem Modul vorangegangenen Schreibauftrags für denselben Datensatz sind von dem Modul noch nicht verarbeitet.
80C2h	Das Modul bearbeitet momentan das mögliche Maximum an Aufträgen für eine CPU.
80C3h	Benötigte Betriebsmittel (Speicher etc.) sind momentan belegt.
80C4h	Kommunikationsfehler
80C5h	Dezentrale Peripherie nicht verfügbar.
80C6h	Datensatzübertragung wurde abgebrochen wegen Prioritätsklassenabbruchs
80D0h	Im zugehörigen SDB ist kein Eintrag für das Modul vorhanden.
80D1h	Die Datensatznummer ist im zugehörigen SDB für das Modul nicht projektiert.
80D2h	Das Modul ist laut Typkennung nicht parametrierbar.
80D3h	Auf den SDB kann nicht zugegriffen werden, da er nicht vorhanden ist.
80D4h	SDB-Strukturfehler: SDB-interner Zeiger zeigt außerhalb SDB.
80D5h	Der Datensatz ist statisch.

16.1.43 SFC 56 - WR_DPARM - Vordefinierte Parameter schreiben

Beschreibung

Mit dem SFC 56 WR_DPARM (write default parameter) wird der Datensatz mit der Nummer *RECNUM* aus dem zuständigen SDB zum adressierten Modul übertragen. Ob es sich dabei um einen statischen oder dynamischen Datensatz handelt ist ohne Bedeutung.

Dieser SFC kann für Digital-, Analogmodule, FMs, CPs und über PROFIBUS DP-V1 eingesetzt werden.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	*REQ* = 1: Anforderung zum Schreiben
IOID	INPUT	BYTE	E, A, M, D, L, Konstante	Kennung des Adressbereichs: 54h = Peripherie Eingang (PE) 55h = Peripherie Ausgang (PA) Handelt es sich um ein Mischmodul, ist die Bereichskennung der niedrigeren Adresse anzugeben. Bei gleichen Adressen ist 54h anzugeben.
LADDR	INPUT	WORD	E, A, M, D, L, Konstante	Logische Basisadresse des Moduls. Bei einem Mischmodul ist die kleinere der beiden Adressen anzugeben.
RECNUM	INPUT	BYTE	E, A, M, D, L, Konstante	Datensatznummer (zulässige Werte: 0 ... 240)
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
BUSY	OUTPUT	BOOL	E, A ,M, D, L	*BUSY* = 1: Der Schreibvorgang ist noch nicht beendet.

RET_VAL (Rückgabewert)

Bei *RET_VAL* = 8xxxh sind zwei Fälle zu unterscheiden:

- Temporäre Fehler (Fehlercodes 80A2h ... 80A4h, 80Cxh):
  Bei dieser Fehlerart besteht die Möglichkeit, dass sich der Fehler ohne Ihr Zutun behebt, es wäre also sinnvoll den SFC erneut (ggf. mehrfach) aufzurufen.
  Beispiel für temporäre Fehler: Benötigte Betriebsmittel sind momentan belegt (80C3h).
- Permanente Fehler (Fehlercodes 809xh, 80A1h, 80Bxh, 80Dxh):
  Bei dieser Fehlerart kann der Fehler nicht ohne Ihr Zutun behoben werden. Ein erneuter Aufruf des SFC ist erst wieder sinnvoll, wenn der Fehler beseitigt wurde.
  Beispiel für permanente Fehler: Falsche Länge des zu übertragenden Datensatzes (80B1h).

Wert	Beschreibung
7000h	Erstaufruf mit *REQ* = 0: keine Datenübertragung aktiv; *BUSY* hat den Wert 0.
7001h	Erstaufruf mit *REQ* = 1: Datenübertragung angestoßen; *BUSY* hat den Wert 1.
7002h	Zwischenaufruf (*REQ* irrelevant): Datenübertragung bereits aktiv; *BUSY* hat den Wert 1.
8090h	Angegebene logische Basisadresse ungültig: Es ist keine Zuordnung im SDB1/SDB2x vorhanden, oder es ist keine Basisadresse.
8093h	Für das über *LADDR* und *IOID* ausgewählte Modul ist dieser SFC nicht zulässig.

Wert	Beschreibung
80A1h	Negative Quittung beim Senden des Datensatzes zum Modul (Modul während des Sendens gezogen oder Modul defekt).
80A2h	DP-Protokollfehler bei Layer 2, evtl. Hardware- / Schnittstellenfehler im DP-Slave
80A3h	DP-Protokollfehler bei User Interface/User
80A4h	Kommunikation gestört (Fehler tritt auf zwischen CPU und externer DP-Anschaltung).
80B0h	SFC für Modultyp nicht möglich, oder das Modul kennt den Datensatz nicht.
80B1h	Die Länge des durch RECORD festgelegten Zielbereichs ist zu klein.
80B2h	Der projektierte Steckplatz ist nicht belegt.
80B3h	Ist-Modultyp ungleich Soll-Modultyp im SDB1.
80C1h	Die Daten des auf dem Modul vorangegangenen Schreibauftrags für denselben Datensatz sind von dem Modul noch nicht verarbeitet.
80C2h	Das Modul bearbeitet momentan das mögliche Maximum an Aufträgen für eine CPU.
80C3h	Benötigte Betriebsmittel (Speicher etc.) sind momentan belegt.
80C4h	Kommunikationsfehler
80C5h	Dezentrale Peripherie nicht verfügbar.
80C6h	Datensatzübertragung wurde abgebrochen wegen Prioritätsklassenabbruchs.
80D0h	Im zugehörigen SDB ist kein Eintrag für das Modul vorhanden.
80D1h	Die Datensatznummer ist im zugehörigen SDB für das Modul nicht projektiert.
80D2h	Das Modul ist laut Typkennung nicht parametrierbar.
80D3h	Auf den SDB kann nicht zugegriffen werden, da er nicht vorhanden ist.
80D4h	SDB-Strukturfehler: SDB-interner Zeiger zeigt außerhalb SDB.

16.1.44 SFC 57 - PARM_MOD - Modul parametrieren

Beschreibung

Mit dem SFC 57 PARM_MOD (parameterize module) werden alle Datensätze eines Moduls, die im zugehörigen SDB projektiert wurden, zum Modul übertragen. Ob es sich dabei um statische oder dynamische Datensätze handelt ist ohne Bedeutung.

Dieser SFC kann für Digital-, Analogmodule, FMs, CPs und über PROFIBUS DP-V1 eingesetzt werden.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	*REQ* = 1: Anforderung zum Schreiben
IOID	INPUT	BYTE	E, A, M, D, L, Konstante	Kennung des Adressbereichs: 54h = Peripherie Eingang (PE) 55h = Peripherie Ausgang (PA) Handelt es sich um eine Mischmodul, ist die Bereichskennung der niedrigeren Adresse anzugeben. Bei gleichen Adressen ist 54h anzugeben.
LADDR	INPUT	WORD	E, A, M, D, L, Konstante	Logische Basisadresse des Moduls. Bei einem Mischmodul ist die kleinere der beiden Adressen anzugeben.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: Der Schreibvorgang ist noch nicht beendet.

RET_VAL (Rückgabewert)

Bei *RET_VAL* = 8xxxh sind zwei Fälle zu unterscheiden:

- Temporäre Fehler (Fehlercodes 80A2h ... 80A4h, 80Cxh):
  Bei dieser Fehlerart besteht die Möglichkeit, dass sich der Fehler ohne Ihr Zutun behebt, es wäre also sinnvoll den SFC erneut (ggf. mehrfach) aufzurufen.
  Beispiel für temporäre Fehler: Benötigte Betriebsmittel sind momentan belegt (80C3h).
- Permanente Fehler (Fehlercodes 809xh, 80A1h, 80Bxh, 80Dxh):
  Bei dieser Fehlerart kann der Fehler nicht ohne Ihr Zutun behoben werden. Ein erneuter Aufruf des SFC ist erst wieder sinnvoll, wenn der Fehler beseitigt wurde.
  Beispiel für permanente Fehler: Falsche Länge des zu übertragenden Datensatzes (80B1h).

Wert	Beschreibung
7000h	Erstaufruf mit *REQ* = 0: keine Datenübertragung aktiv; *BUSY* hat Wert 0.
7001h	Erstaufruf mit *REQ* = 1: Datenübertragung angestoßen; *BUSY* hat den Wert 1.
7002h	Zwischenaufruf (*REQ* irrelevant): Datenübertragung bereits aktiv; *BUSY* hat den Wert 1.
8090h	Angegebene logische Basisadresse ungültig: Es ist keine Zuordnung im SDB1/SDB2x vorhanden, oder es ist keine Basisadresse.
8093h	Für das über *LADDR* und *IOID* ausgewählte Modul ist dieser SFC nicht zulässig.
80A1h	Negative Quittung beim Senden des Datensatzes zum Modul (Modul während des Sendens gezogen oder Modul defekt)
80A2h	DP-Protokollfehler bei Layer 2, evtl. Hardware- / Schnittstellenfehler im DP-Slave
80A3h	DP-Protokollfehler bei User Interface/User
80A4h	Kommunikation gestört (Fehler tritt auf zwischen CPU und externer DP-Anschaltung)
80B0h	SFC für Modultyp nicht möglich, oder das Modul kennt Datensatz nicht.
80B1h	Die Länge des durch *RECORD* festgelegten Zielbereichs ist zu klein.
80B2h	Der projektierte Steckplatz ist nicht belegt.
80B3h	Ist-Modultyp ungleich Soll-Modultyp im SDB1
80C1h	Die Daten des auf dem Modul vorangegangenen Schreibauftrags für denselben Datensatz sind von dem Modul noch nicht verarbeitet.
80C2h	Das Modul bearbeitet momentan das mögliche Maximum an Aufträgen für eine CPU.
80C3h	Benötigte Betriebsmittel (Speicher etc.) sind momentan belegt.
80C4h	Kommunikationsfehler
80C5h	Dezentrale Peripherie nicht verfügbar
80C6h	Datensatzübertragung wurde abgebrochen wegen Prioritätsklassenabbruchs.
80D0h	Im zugehörigen SDB ist kein Eintrag für das Modul vorhanden.
80D1h	Die Datensatznummer ist im zugehörigen SDB für das Modul nicht projektiert.
80D2h	Das Modul ist laut Typkennung nicht parametrierbar.

Wert	Beschreibung
80D3h	Auf den SDB kann nicht zugegriffen werden, da er nicht vorhanden ist.
80D4h	SDB-Strukturfehler: SDB-interner Zeiger zeigt außerhalb SDB.

16.1.45 SFC 58 - WR_REC - Datensatz schreiben

Beschreibung

Mit dem SFC 58 WR_REC (write record) wird der Datensatz *RECORD* zum adressierten Modul übertragen.

Der Schreibvorgang wird gestartet, wenn beim Aufruf des SFC 58 der Eingangsparameter *REQ* mit 1 belegt wird. Der Ausgangsparameter *BUSY* liefert den Wert 0, wenn der Schreibvorgang sofort ausgeführt werden konnte. Wenn der Schreibvorgang noch nicht abgeschlossen ist, liefert *BUSY* den Wert 1.

Dieser SFC kann für Digital-, Analogmodule, FMs, CPs und über PROFIBUS DP-V1 eingesetzt werden.

Systembedingt ist dieser Baustein nicht unterbrechbar!

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	*REQ* = 1: Anforderung zum Schreiben
IOID	INPUT	BYTE	E, A, M, D, L, Konstante	Kennung des Adressbereichs: 54h = Peripherie Eingang (PE) 55h = Peripherie Ausgang (PA) Handelt es sich um eine Mischmodul, ist die Bereichskennung der niedrigeren Adresse anzugeben. Bei gleichen Adressen ist 54h anzugeben.
LADDR	INPUT	WORD	E, A, M, D, L, Konstante	Logische Basisadresse des Moduls. Bei einem Mischmodul ist die kleinere der beiden Adressen anzugeben.
RECNUM	INPUT	BYTE	E, A, M, D, L, Konstante	Datensatznummer (zulässige Werte: 2 ... 240)
RECORD	INPUT	ANY	E, A, M, D, L	Datensatz Es ist nur der Datentyp BYTE zulässig.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: Der Schreibvorgang ist noch nicht beendet.

RECORD

Beim ersten Aufruf des SFC werden die zu übertragenden Daten aus dem Parameter *RECORD* gelesen. Sollte die Übertragung des Datensatzes jedoch länger als einen Aufruf dauern, dann ist bei den Folgeaufrufen des SFC (zum gleichen Auftrag) der Inhalt des Parameters *RECORD* nicht mehr relevant.

RET_VAL (Rückgabewert)

Bei *RET_VAL* = 8xxxh sind zwei Fälle zu unterscheiden:

- Temporäre Fehler (Fehlercodes 80A2h ... 80A4h, 80Cxh):
  Bei dieser Fehlerart besteht die Möglichkeit, dass sich der Fehler ohne Ihr Zutun behebt, es wäre also sinnvoll den SFC erneut (ggf. mehrfach) aufzurufen.
  Beispiel für temporäre Fehler: Benötigte Betriebsmittel sind momentan belegt (80C3h).
- Permanente Fehler (Fehlercodes 809xh, 80A0h, 80A1h, 80Bxh):
  Bei dieser Fehlerart kann der Fehler nicht ohne Ihr Zutun behoben werden. Ein erneuter Aufruf des SFC ist erst wieder sinnvoll, wenn der Fehler beseitigt wurde.
  Beispiel für permanente Fehler: Falsche Länge des zu übertragenden Datensatzes (80B1h).

Wert	Beschreibung
7000h	Erstaufruf mit *REQ* = 0: keine Datenübertragung aktiv; *BUSY* hat den Wert 0.
7001h	Erstaufruf mit *REQ* = 1: Datenübertragung angestoßen; *BUSY* hat den Wert 1.
7002h	Zwischenaufruf (REQ irrelevant): Datenübertragung bereits aktiv; BUSY hat den Wert 1.
8090h	Angegebene logische Basisadresse ungültig: Es ist keine Zuordnung im SDB1/SDB2x vorhanden, oder es ist keine Basisadresse.
8092h	In ANY-Referenz ist eine Typangabe ungleich BYTE angegeben.
8093h	Für das über *LADDR* und *IOID* ausgewählte Modul ist dieser SFC nicht zulässig.
80A1h	Negative Quittung beim Schreiben zum Modul (Modul während des Schreibvorgangs gezogen oder Modul defekt).
80A2h	DP-Protokollfehler bei Layer 2, evtl. Hardware- / Schnittstellenfehler im DP-Slave
80A3h	DP-Protokollfehler bei User Interface/User
80A4h	Kommunikation gestört (Fehler tritt auf zwischen CPU und externer DP-Anschaltung)
80B0h	■ SFC für Modultyp nicht möglich. ■ Modul kennt den Datensatz nicht. ■ Datensatznummer ≥ 241 ist unzulässig. ■ Die Datensätze 0 und 1 sind nicht erlaubt.
80B1h	Die Längenangabe im Parameter *RECORD* ist falsch.
80B2h	Der projektierte Steckplatz ist nicht belegt.
80B3h	Ist-Modultyp ungleich Soll-Modultyp im SDB1
80C1h	Die Daten des auf dem Modul vorangegangenen Schreibauftrags für denselben Datensatz sind von dem Modul noch nicht verarbeitet.
80C2h	Das Modul bearbeitet momentan das mögliche Maximum an Aufträgen für eine CPU.
80C3h	Benötigte Betriebsmittel (Speicher etc.) sind momentan belegt.
80C4h	Kommunikationsfehler
80C5h	Dezentrale Peripherie nicht verfügbar.
80C6h	Datensatzübertragung wurde abgebrochen wegen Prioritätsklassenabbruchs.

Sollte der allgemeine Fehler 8544h auftreten, wird dadurch nur angezeigt, dass der Zugriff auf mindestens ein Byte des den Datensatz enthaltenden E/A-Speicherbereichs gesperrt war. Die Datenübertragung wurde jedoch fortgesetzt.

16.1.46 SFC 59 - RD_REC - Datensatz lesen

Beschreibung

Mit dem SFC 59 RD_REC (read record) wird der Datensatz mit der Nummer *RECNUM* von dem adressierten Modul gelesen.

Dieser SFC kann für Digital-, Analogmodule, FMs, CPs und über PROFIBUS DP-V1 eingesetzt werden.

Der Lesevorgang wird gestartet, wenn beim Aufruf des SFC 59 der Eingangsparameter *REQ* mit 1 belegt wird. Der Ausgangsparameter *BUSY* liefert den Wert 0, wenn der Lesevorgang sofort ausgeführt werden konnte. Wenn der Lesevorgang noch nicht abgeschlossen ist, liefert *BUSY* den Wert 1. Durch den Parameter *RECORD* wird der Zielbereich festgelegt, in den der gelesene Datensatz nach fehlerfreier Datenübertragung eingetragen wird.

Systembedingt ist dieser Baustein nicht unterbrechbar!

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	*REQ* = 1: Anforderung zum Lesen
IOID	INPUT	BYTE	E, A, M, D, L, Konstante	Kennung des Adressbereichs: 54h = Peripherie Eingang (PE) 55h = Peripherie Ausgang (PA) Handelt es sich um ein Mischmodul, ist die Bereichskennung der niedrigeren Adresse anzugeben. Bei gleichen Adressen ist 54h anzugeben.
LADDR	INPUT	WORD	E, A, M, D, L, Konstante	Logische Basisadresse des Moduls. Bei einem Mischmodul ist die kleinere der beiden Adressen anzugeben.
RECNUM	INPUT	BYTE	E, A, M, D, L, Konstante	Datensatznummer (zulässige Werte: 0 ... 240)
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode. Zusätzlich: Länge des tatsächlich übertragenen Datensatzes in Bytes (mögliche Werte: +1 ... +240), falls der Zielbereich größer ist als der übertragene Datensatz und bei der Übertragung kein Fehler auftrat.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: Der Schreibvorgang ist noch nicht beendet.
RECORD	OUTPUT	ANY	E, A, M, D, L	Zielbereich für den gelesenen Datensatz. Sie müssen bei asynchroner Bearbeitung des SFC 59 darauf achten, dass die Aktualparameter von *RECORD* bei allen Aufrufen dieselbe Längeninformation haben. Es ist nur der Datentyp BYTE zulässig.

Geeignete Wahl von RECORD

Wenn Sie sicherstellen wollen, dass immer der gesamte Datensatz gelesen wird, wählen Sie einen Zielbereich mit der Länge 241Bytes. Bei einer fehlerfreien Datenübertragung steht dann in *RET_VAL* die tatsächliche Datensatzlänge.

RET_VAL (Rückgabewert)

RET_VAL enthält einen Fehlercode, wenn während der Bearbeitung der Funktion ein Fehler auftrat.

Bei einer fehlerfreien Übertragung enthält *RET_VAL*:

- den Wert 0 wenn der gesamte Zielbereich mit Daten aus dem selektierten Datensatz gefüllt wurde (Der Datensatz kann aber unvollständig sein).
- die Länge des tatsächlich übertragenen Datensatzes in Bytes (mögliche Werte: 1 ... 240), wenn der Zielbereich größer ist als der übertragene Datensatz.

Fehlerinformationen

Bei *RET_VAL* = 8xxxh sind zwei Fälle zu unterscheiden:

- Temporäre Fehler (Fehlercodes 80A2h ... 80A4h, 80Cxh): Bei dieser Fehlerart besteht die Möglichkeit, dass sich der Fehler ohne Ihr Zutun behebt. Es wäre also sinnvoll, den SFC erneut (ggf. mehrfach) aufzurufen. Beispiel für temporäre Fehler: Benötigte Betriebsmittel sind momentan belegt (80C3h).
- Permanente Fehler (Fehlercodes 809xh, 80A0h, 80A1h, 80Bxh): Bei dieser Fehlerart kann der Fehler nicht ohne Ihr Zutun behoben werden. Ein erneuter Aufruf des SFC ist erst wieder sinnvoll, wenn der Fehler beseitigt wurde. Beispiel für permanente Fehler: Falsche Länge des zu übertragenden Datensatzes (80B1h).

Fehlerinformationen

Wert	Beschreibung
7000h	Erstaufruf mit *REQ* = 0: keine Datenübertragung aktiv; *BUSY* hat den Wert 0.
7001h	Erstaufruf mit *REQ* = 1: Datenübertragung angestoßen; *BUSY* hat den Wert 1.
7002h	Zwischenaufruf (*REQ* irrelevant): Datenübertragung bereits aktiv; *BUSY* hat den Wert 1.
8090h	Angegebene logische Basisadresse ungültig: Es ist keine Zuordnung im SDB1/SDB2x vorhanden, oder es ist keine Basisadresse.
8092h	In ANY-Referenz ist eine Typangabe ungleich BYTE angegeben.
8093h	Für das über *LADDR* und *IOID* ausgewählte Modul ist dieser SFC nicht zulässig.
80A0h	Negative Quittung beim Lesen vom Modul (Modul während des Lesevorgangs gezogen oder Modul defekt)

Wert	Beschreibung
80A2h	DP-Protokollfehler bei Layer 2, evtl. Hardware- / Schnittstellenfehler im DP-Slave
80A3h	DP-Protokollfehler bei User Interface/User
80A4h	Kommunikation gestört (Fehler tritt auf zwischen CPU und externer DP-Anschaltung)
80B0h	■ SFC für Modultyp nicht möglich ■ Modul kennt den Datensatz nicht. ■ Datensatznummer ≥ 241 ist unzulässig.
80B1h	Die Längenangabe im Parameter *RECORD* ist falsch.
80B2h	Der projektierte Steckplatz ist nicht belegt.
80B3h	Ist-Modultyp ungleich Soll-Modultyp im SDB1
80C0h	Das Modul führt den Datensatz, aber es sind noch keine Lesedaten da.
80C1h	Die Daten des auf dem Modul vorangegangenen Schreibauftrags für denselben Datensatz sind von dem Modul noch nicht verarbeitet.
80C2h	Das Modul bearbeitet momentan das mögliche Maximum an Aufträgen für eine CPU.
80C3h	Benötigte Betriebsmittel (Speicher etc.) sind momentan belegt.
80C4h	Kommunikationsfehler
80C5h	Dezentrale Peripherie nicht verfügbar.
80C6h	Datensatzübertragung wurde abgebrochen wegen Prioritätsklassenabbruchs.

Sollte der allgemeine Fehler 8745h auftreten, wird dadurch nur angezeigt, dass beim Schreibvorgang auf das Prozessabbild der Zugriff auf mindestens ein Byte nicht möglich war. Der Datensatz wurde jedoch ordnungsgemäß von dem Modul gelesen und in den E/A-Speicherbereich geschrieben.

16.1.47 SFC 64 - TIME_TCK - Systemzeit lesen

Beschreibung

Mit dem SFC 64 TIME_TCK (time tick) wird die Systemzeit der CPU ausgelesen. Das kann zum Beispiel dazu genutzt werden, die Dauer von Vorgängen zu messen, indem eine Differenzbildung der Ergebnisse von zwei SFC 64-Aufrufen vorgenommen wird. Bei der Systemzeit handelt es sich um einen "Zeitzähler", der von 0 bis max. 2147483647ms zählt und bei einem Überlauf der Systemzeit wieder von 0 beginnt. Das Zeitraster und die Genauigkeit der Systemzeit hängen von der jeweiligen CPU ab. Die Systemzeit kann nur von den Betriebszuständen der CPU beeinflusst werden.

Systemzeit und Betriebszustände

Betriebzustand	Systemzeit ...
Anlauf RUN	... wird ständig aktualisiert.
STOP	... wird angehalten und behält den aktuellen Wert.
Neustart	... wird gelöscht und beginnt wieder von "0" an zu laufen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
RET_VAL	OUTPUT	TIME	E, A, M, D, L	Der Parameter *RET_VAL* enthält die gelesene Systemzeit im Bereich von 0 ... 2^{31} -1ms.

RET_VAL (Rückgabewert) Der SFC 64 gibt keine spezifischen Fehlerinformationen aus.

16.1.48 SFC 65 - X_SEND - Daten senden

Beschreibung

Mit dem SFC 65 X_SEND können Daten an einen außerhalb der eigenen Station liegenden Kommunikationspartner gesendet werden. Der Kommunikationspartner empfängt die Daten mit Hilfe des SFC 66 X_RCV. Die Kennzeichnung der Sendedaten erfolgt über den Eingangsparameter *REQ_ID*. Diese Auftragskennung wird mit übertragen und kann beim Kommunikationspartner ausgewertet werden, um die Herkunft der Daten zu ermitteln. Der Auftrag wird gestartet, wenn am Eingangsparameter *REQ* der Wert 1 anliegt. Der über den Parameter *SD* definierte Sendebereich (auf der sendenden CPU) muss kleiner oder gleich dem über den Parameter *RD* definierten Empfangsbereich (beim Kommunikationspartner) sein, des Weiteren müssen die Datentypen von Sendebereich und Empfangsbereich übereinstimmen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter "request to activate", dient dazu, den Auftrag anzustoßen
CONT	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter "continue", legt fest, ob die Verbindung zum Kommunikationspartner nach Abschluss des Auftrags bestehen bleibt
DEST_ID	INPUT	WORD	E, A, M, D, L, Konstante	Adressierungsparameter "destination ID". Er enthält die MPI-Adresse des Kommunikationspartners.
REQ_ID	INPUT	DWORD	E, A, M, D, L, Konstante	Auftragskennung. Sie dient zur Identifizierung der Daten beim Kommunikationspartner.
SD	INPUT	ANY	E, A, M, D	Referenz auf den Sendebereich. Folgende Datentypen sind erlaubt: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME sowie Arrays der genannten Datentypen mit Ausnahme von BOOL.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: Der Sendevorgang ist noch nicht abgeschlossen. *BUSY* = 0: Der Sendevorgang ist abgeschlossen, bzw. es ist kein Sendevorgang aktiv.

REQ_ID

Der Eingangsparameter *REQ_ID* dient zur Kennzeichnung der Sendedaten.

Auf Empfängerseite wird der Parameter *REQ_ID* benötigt, wenn

- auf einer Sende-CPU mehrere SFC 65 mit unterschiedlichen Parametern *REQ_ID* aufgerufen und die Daten an einen Kommunikationspartner übertragen werden.
- von mehreren Sende-CPUs Daten mit Hilfe des SFC 65 an einen Kommunikationspartner übertragen werden.

Die Empfangsdaten können mit Hilfe der Auswertung des Parameters *REQ_ID* in unterschiedlichen Speicherbereichen abgelegt werden.

Datenkonsistenz

Da die Sendedaten beim ersten Aufruf des SFC in einen internen Puffer des Betriebssystems umkopiert werden, ist darauf zu achten, dass der Sendebereich nicht beschrieben wird, bevor der Erstaufruf abgeschlossen ist, da ansonsten inkonsistente Daten übertragen werden können.

Bei einem schreibenden Zugriff auf die Sendedaten nach dem Erstaufruf wird die Datenkonsistenz nicht beeinträchtigt.

RET_VAL (Rückgabewert)

Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.

Die in der Tabelle "Spezifische Fehlerinformationen" unter anderem angegebenen "echten" Fehlerinformationen können wie folgt klassifiziert werden:

Wert	Beschreibung
809xh	Fehler der CPU, in der SFC abläuft
80Axh	Permanenter Kommunikationsfehler
80Bxh	Fehler beim Kommunikationspartner
80Cxh	Temporärer Fehler

Spezifische Fehlerinformationen:

Wert	Beschreibung
0000h	Die Bearbeitung wurde fehlerfrei abgeschlossen.
7000h	Erstaufruf mit *REQ* = 0: keine Datenübertragung aktiv; *BUSY* hat Wert 0.
7001h	Erstaufruf mit *REQ* = 1: Datenübertragung angestoßen; *BUSY* hat den Wert 1.
7002h	Zwischenaufruf (REQ irrelevant): Datenübertragung bereits aktiv; *BUSY* hat den Wert 1.
8090h	Angegebene Zieladresse des Kommunikationspartners ist ungültig, z.B. ■ falsche *IOID* ■ falsche Basisadresse vorhanden ■ falsche MPI-Adresse (> 126)
8092h	Fehler bei *SD* oder *RD*, z.B.: ■ unzulässige Länge von *SD* ■ *SD* = NIL ist unzulässig
8095h	Baustein wird bereits in einer Prioritätsklasse mit niedrigerer Priorität bearbeitet.
80A0h	Fehler in empfangener Quittung
80A1h	Kommunikationsprobleme: SFC-Aufruf nach Abbruch einer bestehenden Verbindung.

Wert	Beschreibung
80B1h	Fehler im ANY-Pointer. Die Länge des zu übertragenden Datenbereichs ist falsch.
80B4h	Datentypfehler bei ANY-Pointer, oder ARRAY des angegebenen Datentyps nicht erlaubt.
80B5h	Bearbeitung wegen unzulässigem Betriebszustand abgelehnt.
80B6h	In der empfangenen Quittung steht ein unbekannter Fehlercode.
80B8h	Der SFC 66 "X_RCV" des Kommunikationspartners hat die Datenübernahme abgelehnt (*RD* = NIL).
80B9h	Der Datenblock wurde vom Kommunikationspartner identifiziert (Aufruf des SFC 66 "X_RCV" mit *EN_DT* = 0), jedoch wegen Betriebszustand STOP noch nicht ins Anwenderprogramm übernommen.
80BAh	Antwort des Kommunikationspartners passt nicht ins Kommunikationstelegramm.
80C0h	Die angegebene Verbindung ist durch einen anderen Auftrag bereits belegt.
80C1h	Ressourcenengpass bei der CPU, in der SFC abläuft, z.B.: ■ Die maximale Anzahl verschiedener Sendeaufträge des Moduls wird bereits bearbeitet. ■ Die Verbindungsressource ist z.B. durch Empfang belegt.
80C2h	Temporärer Ressourcenmangel des Kommunikationspartners, z.B.: ■ Der Kommunikationspartner bearbeitet momentan das Maximum an Aufträgen. ■ Die benötigten Betriebsmittel (Speicher etc.) sind belegt. ■ Zu wenig Arbeitsspeicher vorhanden. (Stoßen Sie Komprimiervorgang an.)
80C3h	Fehler beim Verbindungsaufbau, z.B.: ■ Die eigene Station befindet sich nicht am MPI-Subnetz. ■ Sie haben die eigene Station am MPI-Subnetz adressiert. ■ Der Kommunikationspartner ist nicht mehr erreichbar. ■ Temporärer Ressourcenmangel des Kommunikationspartners.

16.1.49 SFC 66 - X_RCV - Daten empfangen

Beschreibung

Mit dem SFC 66 X_RCV können Daten empfangen werden, die ein oder mehrere außerhalb der eigenen Station liegende Kommunikationspartner mit Hilfe des SFC 65 X_SEND gesendet haben.

Mit dem SFC 66 kann festgestellt werden, ob zum aktuellen Zeitpunkt gesendete Daten bereitstehen. Diese Daten wurden vom Betriebssystem gegebenenfalls in eine interne Warteschlange gestellt. Sind in dieser Warteschlange Daten vorhanden, so kann der älteste Datenblock in der Warteschlange in einen vorgegebenen Empfangsbereich kopiert werden.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN_DT	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter "enable data transfer". Mit dem Wert 0 überprüfen Sie, ob mindestens ein Datenblock bereitsteht. Der Wert 1 bewirkt das Umkopieren des ältesten in der Warteschlange vorhandenen Datenblocks in den Bereich des Arbeitsspeichers, den Sie durch *RD* vorgegeben haben.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
REQ_ID	OUTPUT	DWORD	E, A, M, D, L	Auftragskennung desjenigen SFC 65 "X_SEND", dessen gesendete Daten in der Warteschlange an erster Stelle stehen, d.h. die ältesten Daten in der Warteschlange. Falls kein Datenblock in der Warteschlange steht, enthält *REQ_ID* den Wert 0.
NDA	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter "new data arrived". *NDA* = 0: ■ In der Warteschlange ist kein Datenblock vorhanden. *NDA* = 1: ■ In der Warteschlange ist mindestens ein Datenblock vorhanden. (Aufruf des SFC 66 mit *EN_DT* = 0). ■ Der älteste Datenblock in der Warteschlange wurde ins Anwenderprogramm kopiert. (Aufruf des SFC 66 mit *EN_DT* = 1).
RD	OUTPUT	ANY	E, A, M, D	Referenz auf den Empfangsbereich (receive data area). Folgende Datentypen sind erlaubt: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME sowie Arrays der genannten Datentypen mit Ausnahme von BOOL. Falls Sie den ältesten in der Warteschlange vorhandenen Datenblock verwerfen wollen, belegen Sie *RD* mit dem Wert NIL.

Datenempfang anzeigen

mit EN_DT = 0

Die empfangenen Daten eines Kommunikationspartners werden vom Betriebssystem in der Reihenfolge ihres Empfangs in die Warteschlange eingereiht.

Zur Überprüfung, ob mindestens ein Datenblock bereitsteht, wird der SFC 66 mit *EN_DT* = 0 aufgerufen und der Ausgangsparameter NDA ausgewertet.

- *NDA* = 0 bedeutet, in der Warteschlange ist kein Datenblock enthalten. *REQ_ID* ist irrelevant, *RET_VAL* enthält den Wert 7000h.
- *NDA* = 1 bedeutet, in der Warteschlange ist mindestens ein Datenblock enthalten.

Ist ein Datenblock in der Warteschlange enthalten, sollten zusätzlich die Ausgangsparameter *RET_VAL* und *REQ_ID* ausgewertet werden. *RET_VAL* enthält die Länge des Datenblocks in Byte, *REQ_ID* die Auftragskennung des Sendebausteins. Wenn sich in der Warteschlange mehrere Datenblöcke befinden, so gehören *REQ_ID* und *RET_VAL* zum ältesten in der Warteschlange liegenden Datenblock.

Daten in Empfangsbereich übernehmen

mit EN_DT = 1

Wenn der Eingangsparameter *EN_DT* = 1 ist, wird der älteste in der Warteschlange vorhandene Datenblock in den durch *RD* vorgegebenen Bereich kopiert. Es ist darauf zu achten, das *RD* größer oder gleich dem über den Parameter SD definierten Sendebereich des zugehörigen SFC 65 X_SEND ist und das die Datentypen übereinstimmen. Wenn die empfangenen Daten in unterschiedlichen Bereichen abgelegt werden sollen, kann im Erstaufruf *REQ_ID* ermittelt (SFC-Aufruf mit *EN_DT* = 0) und im Folgeaufruf (mit *EN_DT* = 1) RD geeignet gewählt werden. Wurde der Auftrag fehlerfrei bearbeitet, enthält *RET_VAL* die Länge des kopierten Datenblocks in Byte und der Sender erhält eine positive Quittung.

Daten verwerfen

Wenn die empfangenen Daten nicht übernommen werden sollen, belegen Sie *RD* mit dem Wert NIL. Der zugehörige Kommunikationspartner erhält dann eine negative Quittung (*RET_VAL* des zugehörigen SFC 65 X_SEND hat den Wert 80B8h) und der Parameter *RET_VAL* hat den Wert 0.

Datenkonsistenz

Es ist darauf zu achten, das der Empfangsbereich nicht ausgelesen wird, bevor der Auftrag nicht abgeschlossen ist, da ansonsten inkonsistente Daten ausgelesen werden können.

Betriebszustandsübergang nach STOP

Wenn die CPU in den Betriebszustand STOP übergeht,

- werden alle neu ankommenden Aufträge negativ quittiert.
- gilt für bereits angekommene Aufträge: Alle in der Empfangswarteschlange eingetragenen Aufträge werden negativ quittiert.
- werden bei einem anschließenden Neustart alle Datenblöcke verworfen.

Verbindungsabbruch

Sollte es zu einem Abbruch der Verbindung kommen, wird ein in der Empfangswarteschlange bereits eingetragener Auftrag, der zu dieser Verbindung gehört, verworfen.

Ausnahme: Falls es sich bei diesem Auftrag um den ältesten Auftrag in der Warteschlange handelt und er durch einen SFC-Aufruf mit *EN_DT* = 0 bereits erkannt wurde, kann er mit *EN_DT* = 1 in den Empfangsbereich übernommen werden.

RET_VAL (Rückgabewert)

Tritt kein Fehler auf, enthält RET_VAL:

- bei *EN_DT* = 0/1 und *NDA* = 0: 7000h. In diesem Fall steht kein Datenblock in der Warteschlange.
- bei *EN_DT* = 0 und *NDA* = 1 die Länge des ältesten in der Warteschlange eingetragenen Datenblocks in Byte als positive Zahl.
- bei *EN_DT* = 1 und *NDA* = 1 die Länge des in den Empfangsbereich *RD* kopierten Datenblocks in Byte als positive Zahl.

Fehlerinformationen

Die in der Tabelle "Spezifische Fehlerinformationen" u.a. angegebenen "echten" Fehlerinformationen können wie folgt klassifiziert werden:

Wert	Beschreibung
809xh	Fehler auf der CPU, auf der der SFC abläuft
80Axh	Permanenter Kommunikationsfehler
80Bxh	Fehler beim Kommunikationspartner
80Cxh	Temporärer Fehler

Spezifische Fehlerinformationen:

Wert	Beschreibung
0000h	Die Bearbeitung wurde fehlerfrei abgeschlossen.
00xyh	Bei NDA = 1 und *RD* <> NIL: *RET_VAL* enthält die Länge des empfangenen (bei *EN_DT* = 0) bzw. des in *RD* kopierten Datenblocks (bei *EN_DT* = 1).
7000h	*EN_DT* = 0/1 und *NDA* = 0
7001h	Erstaufruf mit *REQ* = 1: Datenübertragung angestoßen; *BUSY* hat den Wert 1.
7002h	Zwischenaufruf (*REQ* irrelevant): Datenübertragung bereits aktiv; *BUSY* = Wert 1.
8090h	Angegebene Zieladresse des Kommunikationspartners ist ungültig, z.B.: ■ falsche *IOID* ■ falsche Basisadresse vorhanden ■ falsche MPI-Adresse (> 126)
8092h	Fehler bei *SD* oder *RD*, z.B.: ■ Es wurden mehr Daten empfangen als in den durch *RD* vorgegebenen Bereich hineinpassen. ■ *RD* ist vom Datentyp BOOL, die Länge der empfangenen Daten ist jedoch größer als ein Byte.
8095h	Der Baustein wird bereits in einer Prioritätsklasse mit niedrigerer Priorität bearbeitet.
80A0h	Fehler in empfangener Quittung.
80A1h	Kommunikationsprobleme: SFC-Aufruf nach Abbruch einer bestehenden Verbindung.
80B1h	Fehler im ANY-Pointer. Die Länge des zu übertragenden Datenbereichs falsch.
80B4h	Datentypfehler bei ANY-Pointer, oder ARRAY des angegebenen Datentyps nicht erlaubt.
80B6h	In der empfangenen Quittung steht ein unbekannter Fehlercode.
80BAh	Die Antwort des Kommunikationspartners passt nicht in das Kommunikationstelegramm.
80C0h	Die angegebene Verbindung ist durch einen anderen Auftrag bereits belegt.
80C1h	Ressourcenengpass bei der CPU, auf der der SFC abläuft, z.B.: ■ Die maximale Anzahl verschiedener Sendeaufträge des Moduls wird bereits bearbeitet. ■ Die Verbindungsressource ist z.B. durch Empfang belegt.

Wert	Beschreibung
80C2h	Temporärer Ressourcenmangel des Kommunikationspartners, z.B.: ■ Der Kommunikationspartner bearbeitet momentan das Maximum an Aufträgen. ■ Die benötigten Betriebsmittel (Speicher etc.) sind belegt. ■ Zu wenig Arbeitsspeicher vorhanden. (Stoßen Sie Komprimiervorgang an.)
80C3h	Fehler beim Verbindungsaufbau, z.B.: ■ Die eigene Station befindet sich nicht am MPI-Subnetz. ■ Sie haben die eigene Station am MPI-Subnetz adressiert. ■ Der Kommunikationspartner ist nicht mehr erreichbar. ■ Temporärer Ressourcenmangel des Kommunikationspartners.

16.1.50 SFC 67 - X_GET - Daten lesen

Beschreibung

Mit dem SFC 67 X_GET können Daten aus einem Kommunikationspartner, der außerhalb der eigenen Station liegt, gelesen werden, wobei auf dem Kommunikationspartner kein zugehöriger SFC existiert. Der Auftrag wird gestartet, wenn am Eingangsparameter *REQ* der Wert 1 anliegt. Danach wird der SFC 67 so lange aufgerufen, bis am Ausgangsparameter BUSY der Wert 0 anliegt. Der Ausgangsparameter *RET_VAL* enthält die Länge des empfangenen Datenblocks in Byte.

Der über den Parameter *RD* definierte Empfangsbereich (auf der empfangenden CPU) muss mindestens so lang wie der über den Parameter *VAR_ADDR* definierte Lesebereich (beim Kommunikationspartner) sein, des Weiteren müssen die Datentypen von *RD* und *VAR_ADDR* übereinstimmen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter "request to activate", dient dazu, den Auftrag anzustoßen.
CONT	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter "continue", legt fest, ob die Verbindung zum Kommunikationspartner nach Abschluss des Auftrags bestehen bleibt.
DEST_ID	INPUT	WORD	E, A, M, D, L, Konstante	Adressierungsparameter "destination ID". Er enthält die MPI-Adresse des Kommunikationspartners.
VAR_ADDR	INPUT	ANY	E, A, M, D	Referenz auf den Bereich in der Partner-CPU, der gelesen werden soll. Sie müssen einen Datentyp wählen, der vom Kommunikationspartner unterstützt wird.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.Tritt kein Fehler auf, enthält *RET_VAL* die Länge des in den Empfangsbereich *RD* kopierten Datenblocks in Byte als positive Zahl.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: Der Empfangsvorgang ist noch nicht abgeschlossen. *BUSY* = 0: Der Empfangsvorgang ist abgeschlossen, bzw. es ist kein Empfangsvorgang aktiv.
RD	OUTPUT	ANY	E, A, M, D	Referenz auf den Empfangsbereich (receive data area). Folgende Datentypen sind erlaubt: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME sowie Arrays der genannten Datentypen mit Ausnahme von BOOL

Datenkonsistenz

Um sicherzustellen, dass die Datenkonsistenz nicht beeinträchtigt wird sollten folgende Konsistenzregeln beachtet werden:

- Aktive CPU (Datenempfänger):
  Der Empfangsbereich sollte in dem OB ausgelesen werden, in dem der entsprechende SFC aufgerufen wird. Sollte dies nicht möglich sein, so sollte der Empfangsbereich erst dann ausgelesen werden, wenn die Bearbeitung des zugehörigen SFC abgeschlossen ist.
- Passive CPU (Datensender):
  Es dürfen höchstens so viele Daten in den Sendebereich geschrieben werden, wie die Blockgröße der passiven CPU (Datensender) angibt.
- Passive CPU (Datensender):
  Die zu sendenden Daten sollten unter Interrupt-Sperre in den Sendebereich geschrieben werden.

Betriebszustandsübergang nach STOP

Wenn die CPU in den Betriebszustand STOP übergeht, wird die vom SFC 67 aufgebaute Verbindung abgebaut. Von der Art des anschließenden Anlaufs hängt ab, ob die bereits empfangenen Daten, die in einem Zwischenspeicher des Betriebssystems stehen, verloren gehen.

Bei einem anschließenden Neustart werden die Daten verworfen.

Betriebszustandsübergang des Partners nach STOP

Da die Daten auch im Betriebszustand STOP gelesen werden können, hat ein Übergang der CPU des Kommunikationspartners in den Betriebszustand STOP keine Auswirkung auf die Datenübertragung.

RET_VAL (Rückgabewert)

Die in der Tabelle "Spezifische Fehlerinformationen" u.a. angegebenen "echten Fehlerinformationen" können wie folgt klassifiziert werden:

Wert	Beschreibung
809xh	Fehler auf der CPU, auf der der SFC abläuft
80Axh	Permanenter Kommunikationsfehler
80Bxh	Fehler beim Kommunikationspartner
80Cxh	Temporärer Fehler

Spezifische Fehlerinformationen:

Wert	Beschreibung
0000h	Die Bearbeitung wurde fehlerfrei abgeschlossen.
00xyh	*RET_VAL* enthält die Länge des empfangenen Datenblocks.
7000h	Aufruf mit *REQ* = 0 (Aufruf ohne Bearbeitung), *BUSY* hat den Wert 0, es ist keine Datenübertragung aktiv.
7001h	Erstaufruf mit *REQ* = 1: Datenübertragung angestoßen; *BUSY* hat den Wert 1.
7002h	Zwischenaufruf (*REQ* irrelevant): Datenübertragung bereits aktiv; *BUSY* hat den Wert 1.
8090h	Angegebene Zieladresse des Kommunikationspartners ist ungültig, z.B.: ■ falsche *IOID* ■ falsche Basisadresse vorhanden ■ falsche MPI-Adresse (> 126)
8092h	Fehler bei *SD* oder *RD*, z.B.: ■ unzulässige Länge bei *RD* ■ Die Länge oder der Datentyp von *RD* stimmt mit den empfangenen Daten nicht überein. ■ *RD* = NIL ist unzulässig
8095h	Der Baustein wird bereits in einer Prioritätsklasse mit niedrigerer Priorität bearbeitet.
80A0h	Fehler in empfangener Quittung
80A1h	Kommunikationsprobleme: SFC-Aufruf nach Abbruch einer bestehenden Verbindung.
80B0h	Objekt ist nicht erreichbar, z.B. DB nicht geladen.
80B1h	Fehler im ANY-Pointer. Die Länge des zu übertragenden Datenbereichs ist falsch.
80B2h	HW-Fehler: Modul nicht vorhanden ■ Der projektierte Steckplatz ist nicht belegt. ■ Ist-Modultyp ungleich Soll-Modultyp ■ Dezentrale Peripherie ist nicht verfügbar. ■ Im zugehörigen SDB ist kein Eintrag für das Modul vorhanden.
80B3h	Daten dürfen entweder nur gelesen oder nur geschrieben werden, z.B. schreibgeschützter DB
80B4h	Der in *VAR_ADDR* angegebene Datentyp wird vom Kommunikationspartner nicht unterstützt.
80B6h	In der empfangenen Quittung steht ein unbekannter Fehlercode.
80BAh	Die Antwort des Kommunikationspartners passt nicht in das Kommunikationstelegramm.
80C0h	Die angegebene Verbindung ist durch einen anderen Auftrag bereits belegt.
80C1h	Ressourcenengpass bei der CPU, auf der der SFC abläuft, z.B.: ■ Die maximale Anzahl verschiedener Sendeaufträge des Moduls wird bereits bearbeitet. ■ Die Verbindungsressource ist z.B. durch Empfang belegt.

Wert	Beschreibung
80C2h	Temporärer Ressourcenmangel des Kommunikationspartners, z.B.: ■ Der Kommunikationspartner bearbeitet momentan das Maximum an Aufträgen. ■ Die benötigten Betriebsmittel (Speicher etc.) sind belegt. ■ Zu wenig Arbeitsspeicher vorhanden. (Stoßen Sie einen Komprimiervorgang an.)
80C3h	Fehler beim Verbindungsaufbau, z.B.: ■ Die eigene Station befindet sich nicht am MPI-Subnetz. ■ Sie haben die eigene Station am MPI-Subnetz adressiert. ■ Der Kommunikationspartner ist nicht mehr erreichbar. ■ Temporärer Ressourcenmangel des Kommunikationspartners.

16.1.51 SFC 68 - X_PUT - Daten schreiben

Beschreibung

Mit dem SFC 68 X_PUT können Daten in einen Kommunikationspartner, der außerhalb der eigenen Station liegt, geschrieben werden, wobei auf dem Kommunikationspartner kein zugehöriger SFC existiert. Der Auftrag wird gestartet, wenn am Eingangsparameter *REQ* der Wert 1 anliegt. Danach wird der SFC 68 so lange aufgerufen, bis am Ausgangsparameter *BUSY* der Wert 0 anliegt. Der über den Parameter *SD* definierte Sendebereich (auf der sendenden CPU) muss mindestens so lang wie der über den Parameter *VAR_ADDR* definierte Empfangsbereich (beim Kommunikationspartner) sein, des Weiteren müssen die Datentypen von *SD* und *VAR_ADDR* übereinstimmen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter "request to activate", dient dazu, den Auftrag anzustoßen.
CONT	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter "continue", legt fest, ob die Verbindung zum Kommunikationspartner nach Abschluss des Auftrags bestehen bleibt
DEST_ID	INPUT	WORD	E, A, M, D, L, Konstante	Adressierungsparameter "destination ID". Er enthält die MPI-Adresse des Kommunikationspartners.
VAR_ADDR	INPUT	ANY	E, A, M, D	Referenz auf den Bereich in der Partner-CPU, in den geschrieben werden soll. Sie müssen einen Datentyp wählen, der vom Kommunikationspartner unterstützt wird.
SD	INPUT	ANY	E, A, M, D	Referenz auf den Bereich in der eigenen CPU, der die zu versendenden Daten enthält. Folgende Datentypen sind erlaubt: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME sowie Arrays der genannten Datentypen mit Ausnahme von BOOL.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: Der Sendevorgang ist noch nicht abgeschlossen. *BUSY* = 0: Der Sendevorgang ist abgeschlossen, bzw. es ist kein Sendevorgang aktiv.

Datenkonsistenz

Um sicherzustellen das die Datenkonsistenz nicht beeinträchtigt wird sollten folgende Konsistenzregeln beachtet werden:

- Aktive CPU (Datensender):
  Der Sendebereich sollte von dem OB aus beschrieben werden, in dem der entsprechende SFC aufgerufen wird. Sollte dies nicht möglich sein, so sollte der Sendebereich erst dann beschrieben werden, wenn der erste Aufruf des zugehörigen SFC abgeschlossen ist.
- Aktive CPU (Datensender):
  Es dürfen höchstens so viele Daten in den Sendebereich geschrieben werden, wie die Blockgröße der passiven CPU (Datenempfänger) angibt.
- Passive CPU (Datenempfänger):
  Die zu empfangenden Daten sollten unter Interrupt-Sperre aus dem Empfangsbereich ausgelesen werden.

Betriebszustandsübergang nach STOP

Wenn die CPU in den Betriebszustand STOP übergeht, wird die vom SFC 68 aufgebaute Verbindung abgebaut und die Daten können nicht mehr gesendet werden. Wenn die Sendedaten zum Zeitpunkt des Betriebszustandsübergangs nach STOP bereits in den internen Puffer kopiert waren, wird der Pufferinhalt verworfen.

Betriebszustandsübergang des Partners nach STOP

Da die Daten auch im Betriebszustand STOP geschrieben werden können, hat ein Übergang der CPU des Kommunikationspartners in den Betriebszustand STOP keine Auswirkung auf die Datenübertragung.

RET_VAL (Rückgabewert)

Die in der Tabelle "Spezifische Fehlerinformationen" u.a. angegebenen "echten Fehlerinformationen" können wie folgt klassifiziert werden:

Wert	Beschreibung
809xh	Fehler auf der CPU, auf der der SFC abläuft
80Axh	Permanenter Kommunikationsfehler
80Bxh	Fehler beim Kommunikationspartner
80Cxh	Temporärer Fehler

Spezifische Fehlerinformationen:

Wert	Beschreibung
0000h	Die Bearbeitung wurde fehlerfrei abgeschlossen.
7000h	Aufruf mit *REQ* = 0 (Aufruf ohne Bearbeitung), *BUSY* hat den Wert 0, es ist keine Datenübertragung aktiv.
7001h	Erstaufruf mit *REQ* = 1: Datenübertragung angestoßen; *BUSY* hat den Wert 1.
7002h	Zwischenaufruf (*REQ* irrelevant): Datenübertragung bereits aktiv; BUSY hat den Wert 1.
8090h	Angegebene Zieladresse des Kommunikationspartners ist ungültig, z.B.: ■ falsche *IOID* ■ falsche Basisadresse vorhanden ■ falsche MPI-Adresse (> 126)
8092h	Fehler bei *SD* oder *RD*, z.B.: ■ unzulässige Länge bei *SD* ■ *SD* = NIL ist unzulässig
8095h	Der Baustein wird bereits in einer Prioritätsklasse mit niedrigerer Priorität bearbeitet.
80A0h	Der in *SD* der sendenden CPU angegebene Datentyp wird vom Kommunikationspartner nicht unterstützt.
80A1h	Kommunikationsprobleme: SFC-Aufruf nach Abbruch einer bestehenden Verbindung.
80B0h	Objekt ist nicht erreichbar, z.B. DB nicht geladen.
80B1h	Fehler im ANY-Pointer. Die Länge des zu übertragenden Datenbereichs ist falsch.
80B2h	HW-Fehler: Modul nicht vorhanden ■ Der projektierte Steckplatz ist nicht belegt. ■ Ist-Modultyp ungleich Soll-Modultyp ■ Dezentrale Peripherie ist nicht verfügbar. ■ Im zugehörigen SDB ist kein Eintrag für das Modul vorhanden.
80B3h	Daten dürfen entweder nur gelesen oder nur geschrieben werden, z.B. schreibgeschützter DB
80B4h	Der in *VAR_ADDR* angegebene Datentyp wird vom Kommunikationspartner nicht unterstützt.
80B6h	In der empfangenen Quittung steht ein unbekannter Fehlercode.
80B7h	Datentyp und/oder Länge der übertragenen Daten passen nicht zum Bereich in der Partner-CPU, in den geschrieben werden soll.
80BAh	Die Antwort des Kommunikationspartners passt nicht in das Kommunikationstelegramm.
80C0h	Die angegebene Verbindung ist durch einen anderen Auftrag bereits belegt.
80C1h	Ressourcenengpass bei der CPU, auf der der SFC abläuft, z.B.: ■ Die maximale Anzahl verschiedener Sendeaufträge des Moduls wird bereits bearbeitet. ■ Die Verbindungsressource ist z.B. durch Empfang belegt.

Wert	Beschreibung
80C2h	Temporärer Ressourcenmangel des Kommunikationspartners, z.B.: ■ Der Kommunikationspartner bearbeitet momentan das Maximum an Aufträgen. ■ Die benötigten Betriebsmittel (Speicher etc.) sind belegt. ■ Zu wenig Arbeitsspeicher vorhanden (Stoßen Sie einen Komprimiervorgang an).
80C3h	Fehler beim Verbindungsaufbau, z.B.: ■ Die eigene Station befindet sich nicht am MPI-Subnetz. ■ Sie haben die eigene Station am MPI-Subnetz adressiert. ■ Der Kommunikationspartner ist nicht mehr erreichbar. ■ Temporärer Ressourcenmangel des Kommunikationspartners.

16.1.52 SFC 69 - X_ABORT - Verbindung abbrechen

Beschreibung

Mit dem SFC 69 X_ABORT kann die Verbindung zu einem Kommunikationspartner, der außerhalb der eigenen Station liegt, abgebrochen werden, wenn die Verbindung mit einem der SFCs 65, 67 oder 68 aufgebaut wurde. Der Auftrag wird gestartet, wenn am Eingangsparameter *REQ* der Wert 1 anliegt.

Wenn der zu den SFCs 65, 67 oder 68 gehörende Auftrag bereits abgeschlossen wurde (*BUSY* = 0), dann sind nach dem Aufruf des SFC 69 die dafür belegten Verbindungsressourcen auf beiden Seiten wieder freigegeben. Ist der zugehörige Auftrag jedoch noch nicht abgeschlossen (*BUSY* = 1), muss nach Abschluss des Verbindungsabbruchs der zugehörige SFC 65, 67 oder 68 erneut mit *REQ* = 0 und *CONT* = 0 aufgerufen und *BUSY* = 0 abgewartet werden, denn erst dann sind alle belegten Verbindungsressourcen wieder freigegeben. Der SFC 69 kann nur auf derjenigen Seite aufgerufen werden, auf der der SFC 65, 67 oder 68 abläuft.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter "request to activate", dient dazu, den Auftrag anzustoßen.
DEST_ID	INPUT	WORD	E, A, M, D, L, Konstante	Adressierungsparameter "destination ID". Er enthält die MPI-Adresse des Kommunikationspartners.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: Der Verbindungsabbruch ist noch nicht abgeschlossen. *BUSY* = 0: Der Verbindungsabbruch ist abgeschlossen.

Betriebszustandsübergang nach STOP

Wenn die CPU in den Betriebszustand STOP übergeht, wird der vom SFC 69 angestoßene Verbindungsabbruch dennoch zu Ende bearbeitet.

Betriebszustandsübergang des Partners nach STOP

Ein Übergang der CPU des Kommunikationspartners in den Betriebszustand STOP hat keine Auswirkung auf den Verbindungsabbruch, die Verbindung wird ungeachtet des Betriebszustandsübergangs abgebrochen.

RET_VAL (Rückgabewert) Die in der Tabelle "Spezifische Fehlerinformationen" u.a. angegebenen "echten Fehlerinformationen" können wie folgt klassifiziert werden:

Wert	Beschreibung
809xh	Fehler auf der CPU, auf der der SFC abläuft
80Axh	Permanenter Kommunikationsfehler
80Bxh	Fehler beim Kommunikationspartner
80Cxh	Temporärer Fehler

Spezifische Fehlerinformationen:

Wert	Beschreibung
0000h	*REQ* = 1, und die angegebene Verbindung ist nicht aufgebaut.
7000h	Aufruf mit *REQ* = 0 (Aufruf ohne Bearbeitung), BUSY hat den Wert 0, es ist keine Datenübertragung aktiv.
7001h	Erstaufruf mit *REQ* = 1: Datenübertragung angestoßen; BUSY hat den Wert 1.
7002h	Zwischenaufruf mit *REQ* = 1.
8090h	Angegebene Zieladresse des Kommunikationspartners ist ungültig, z.B.: ■ falsche *IOID* ■ falsche Basisadresse vorhanden ■ falsche MPI-Adresse (> 126)
8095h	Der Baustein wird bereits in einer Prioritätsklasse mit niedrigerer Priorität bearbeitet.
80A0h	Fehler in empfangener Quittung.
80A1h	Kommunikationsprobleme: SFC-Aufruf nach Abbruch einer bestehenden Verbindung.
80B1h	Fehler im ANY-Pointer. Die Länge des zu übertragenden Datenbereichs ist falsch.
80B4h	Datentypfehler bei ANY-Pointer, oder ARRAY des angegebenen Datentyps nicht erlaubt.
80B6h	In der empfangenen Quittung steht ein unbekannter Fehlercode.
80BAh	Die Antwort des Kommunikationspartners passt nicht in das Kommunikationstelegramm.
80C0h	Die angegebene Verbindung ist durch einen anderen Auftrag bereits belegt.
80C1h	Ressourcenengpass bei der CPU, auf der der SFC abläuft, z.B.: ■ Die maximale Anzahl verschiedener Sendeaufträge des Moduls wird bereits bearbeitet. ■ Die Verbindungsressource ist z.B. durch Empfang belegt.

Wert	Beschreibung
80C2h	Temporärer Ressourcenmangel des Kommunikationspartners, z.B.: ■ Der Kommunikationspartner bearbeitet momentan das Maximum an Aufträgen. ■ Die benötigten Betriebsmittel (Speicher etc.) sind belegt. ■ Zu wenig Arbeitsspeicher vorhanden (Stoßen Sie einen Komprimiervorgang an).
80C3h	Fehler beim Verbindungsaufbau, z.B.: ■ Die eigene Station befindet sich nicht am MPI-Subnetz. ■ Sie haben die eigene Station am MPI-Subnetz adressiert. ■ Der Kommunikationspartner ist nicht mehr erreichbar. ■ Temporärer Ressourcenmangel des Kommunikationspartners.

16.1.53 SFC 70 - GEO_LOG - Anfangsadresse einer Baugruppe ermitteln

Beschreibung

Vom Kanal einer Signalbaugruppe seien der zugehörige Baugruppensteckplatz der Baugruppe bekannt. Mit der SFC 70 GEO_LOG (convert geographical address to logical address) ermitteln Sie daraus die zugehörige Anfangsadresse der Baugruppe, d.h. die kleinste E- oder A-Adresse. Wenn Sie die SFC 70 auf Power-Module oder Module mit gepackten Adressen anwenden, wird die Diagnoseadresse zurückgeliefert.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
MASTER	INPUT	INT	E, A, M, D, L, Konstante	Bereichskennung: ■ 0, falls sich der Steckplatz im zentralen Aufbau befindet: Baugruppenträger 0 bis 3 (S7-300) bzw. 0 bis 21 (S7-400) ■ 1 bis 32: DP-Mastersystem-ID des zugehörigen Feldgeräts, falls sich der Steckplatz in einem Feldgerät am PROFIBUS befindet ■ 100 bis 115: PROFINET IO-System- ID des zugehörigen Feldgeräts, falls sich der Steckplatz in einem Feldgerät am PROFINET befindet
STATION	INPUT	INT	E, A, M, D, L, Konstante	■ Nr. des Baugruppenträgers, falls Bereichskennung = 0 ■ Stationsnummer des Feldgeräts, falls Bereichskennung > 0
SLOT	INPUT	INT	E, A, M, D, L, Konstante	Steckplatz-Nr.
SUBSLOT	INPUT	INT	E, A, M, D, L Konstante	Submodulsteckplatz (falls kein Submodul gesteckt werden kann, ist hier 0 anzugeben)
RET_VAL	OUTPUT	INT	E, A, M, D, L	Fehlerinformation
LADDR	OUTPUT	WORD	E, A, M, D, L	Anfangsadresse der Baugruppe Bit 15 von *LADDR* gibt an, ob eine Ein- (Bit 15 = 0) oder eine Ausgangsadresse (Bit 15 = 1) vorliegt.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Der Auftrag wurde fehlerfrei durchgeführt.
8094h	Es wurde kein Subnetz mit der angegebenen *SUBNETID* konfiguriert.
8095h	Unzulässiger Wert beim Parameter *STATION*
8096h	Unzulässiger Wert beim Parameter *SLOT*
8097h	Unzulässiger Wert beim Parameter *SUBSLOT*
8099h	Der Steckplatz ist nicht projektiert.
809Ah	Die Submoduladresse für den ausgewählten Steckplatz ist nicht projektiert.
8xyyh	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

16.1.54 SFC 71 - LOG_GEO - zu logischer Adresse gehörenden Slot ermitteln

Beschreibung

Mit der SFC 71 LOG_GEO (convert logical address to geographical address) ermitteln Sie den zu einer logischen Adresse gehörenden Modulsteckplatz sowie den Offset im Nutzdatenadressraum des Moduls.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
LADDR	INPUT	WORD	E, A, M, D, L, Konstante	Irgendeine logische Adresse des Moduls In Bit15 geben Sie an, ob es sich um eine Ein- (Bit 15 = 0) oder eine Ausgangsadresse (Bit 15 = 1) handelt.
RET_VAL	OUTPUT	INT	E, A, M, D, L,	Fehlerinformation
AREA	OUTPUT	INT	E, A, M, D, L,	Bereichskennung: Sie gibt an, wie die restlichen Ausgangsparameter zu interpretieren sind.
MASTER	OUTPUT	INT	E, A, M, D, L Konstante	Bereichskennung: ■ 0, falls sich der Steckplatz im zentralen Aufbau befindet: Modulträger 0 bis 3 (S7-300) bzw. 0 bis 21 (S7-400) ■ 1 bis 32: DP-Mastersystem-ID des zugehörigen Feldgeräts, falls sich der Steckplatz in einem Feldgerät am PROFIBUS befindet ■ 100 bis 115: PROFINET IO-System- ID des zugehörigen Feldgeräts, falls sich der Steckplatz in einem Feldgerät am PROFINET befindet
STATION	OUTPUT	INT	E, A, M, D, L	■ Nr. des Modulträgers, falls Bereichskennung = 0 ■ Stationsnummer des Feldgeräts, falls Bereichskennung > 0
SLOT	OUTPUT	INT	E, A, M, D, L	Steckplatz-Nr.
SUBSLOT	OUTPUT	INT	E, A, M, D, L	Submodulnummer
OFFSET	OUTPUT	INT	E, A, M, D, L	Offset im Nutzdatenadressraum des zugehörigen Moduls

Ausgangsparameter AREA

Wert von AREA	System	Bedeutung von RACK, SLOT und SUBADDR
0	S7-400	■ *MASTER*: 0 ■ *STATION*: Modulträger-Nr. ■ *SLOT*: Steckplatz-Nr. ■ *SUBSLOT*: 0 ■ *OFFSET*: Differenz zwischen logischer Adresse und logischer Basisadresse
1	S7-300	■ *MASTER*: 0 ■ *STATION*: Modulträger-Nr. ■ *SLOT*: Steckplatz-Nr. ■ *SUBSLOT*: 0 ■ *OFFSET*: Differenz zwischen logischer Adresse und logischer Basisadresse
2	PROFIBUS DP	■ *MASTER*: DP-Mastersystem-ID ■ *STATION*: Stationsnummer ■ *SLOT*: Steckplatz-Nr. in der Station ■ *SUBSLOT*: 0 ■ *OFFSET*: Offset im Nutzdatenadressraum der zugehörigen Baugruppe
	PROFINET IO	■ *MASTER*: PROFINET IO-System-ID ■ *STATION*: Stationsnummer ■ *SLOT*: Steckplatz-Nr. in der Station ■ *SUBSLOT*: Submodulnummer ■ *OFFSET*: Offset im Nutzdatenadressraum der zugehörigen Baugruppe
3	S5-P-Bereich	■ *MASTER*: 0 ■ *STATION*: Modulträger-Nr. ■ *SLOT*: Steckplatz-Nr. in der Adaptionskapsel ■ *SUBSLOT*: 0 ■ *OFFSET*: Adresse im S5-x-Bereich
4	S5-Q-Bereich	■ *MASTER*: 0 ■ *STATION*: Modulträger-Nr. ■ *SLOT*: Steckplatz-Nr. der Adaptionskapsel ■ *SUBSLOT*: 0 ■ *OFFSET*: Adresse im S5-x-Bereich
5	S5-IM3-Bereich	■ *MASTER*: 0 ■ *STATION*: Modulträger-Nr. ■ *SLOT*: Steckplatz-Nr. der Adaptionskapsel ■ *OFFSET*: Adresse im S5-x-Bereich
6	S5-IM4-Bereich	■ *MASTER*: 0 ■ *STATION*: Modulträger-Nr. ■ *SLOT*: Steckplatz-Nr. der Adaptionskapsel ■ *SUBSLOT*: 0 ■ *OFFSET*: Adresse im S5-x-Bereich

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Der Auftrag wurde fehlerfrei durchgeführt.
8090h	Angegebene logische Adresse ungültig
8xyyh	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

16.1.55 SFC 81 - UBLKMOV - Variable ununterbrechbar kopieren

Beschreibung

Mit dem SFC 81 UBLKMOV (uninterruptable move) kopieren Sie den Inhalt eines Speicherbereiches (= Quellbereich) konsistent in einen anderen Speicherbereich (=Zielbereich). Der Kopiervorgang kann nicht durch andere Tätigkeiten des Betriebssystems unterbrochen werden.

Mit dem SFC 81 UBLKMOV können Sie alle Speicherbereiche kopieren außer:

- folgende Bausteine: FB, SFB, FC, SFC, OB, SDB
- Zähler
- Zeiten
- Speicherbereiche des Peripheriebereiches
- nicht ablaufrelevante Datenbausteine.

Die maximale Datenmenge, die Sie kopieren können, beträgt 512Byte.

Unterbrechbarkeit

Der Kopiervorgang ist nicht unterbrechbar. Sie müssen daher beachten, dass sich die Alarmreaktionszeit Ihrer CPU bei Einsatz des SFC 81 erhöhen kann.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
SRCBLK	INPUT	ANY	E, A, M, D, L	Angabe des Speicherbereiches, der kopiert werden soll (Quellfeld). Arrays vom Datentyp STRING sind nicht erlaubt.
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
DSTBLK	OUTPUT	ANY	E, A, M, D, L	Angabe des Speicherbereiches, in den kopiert werden soll (Zielfeld). Arrays vom Datentyp STRING sind nicht erlaubt.

Quell- und Zielbereich dürfen sich nicht überlappen.

Ist der angegebene Zielbereich größer als der Quellbereich, dann werden auch nur so viele Daten in den Zielbereich kopiert, wie im Quellbereich stehen.

Ist der angegebenen Zielbereich kleiner als der Quellbereich, dann werden auch nur so viele Daten kopiert, wie der Zielbereich aufnehmen kann.

Ist der ANY-Pointer (Quelle oder Ziel) vom Typ BOOL, so muss die angegebene Länge durch 8 teilbar sein, da sonst der SFC nicht ausgeführt wird.

Ist der ANY-Pointer vom Typ STRING, so muss die angegebene Länge 1 sein.

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	kein Fehler
8091h	Der Quellbereich liegt in einem nicht ablaufrelevanten Datenbaustein.

16.1.56 SFC 101 - RTM - Hantierung Betriebsstundenzähler

Beschreibung

Mit der SFC 101 RTM (run-time meter) können Sie einen 32-Bit-Betriebsstundenzähler Ihrer CPU setzen, starten, stoppen und auslesen. Falls Sie alle 32-Bit-Betriebsstundenzähler Ihrer CPU auslesen möchten, verwenden Sie die SFC 51 RDSYSST mit SZL_ID=W#16#0132 und INDEX=W#16#000B (für die Betriebsstundenzähler 0 bis 7) bzw. INDEX=W#16#000C (für die Betriebsstundenzähler 8 bis 15).

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
NR	INPUT	BYTE	E, A, M, D, L, Konstante	Nummer des Betriebsstundenzählers Die Nummerierung beginnt mit 0. Die Anzahl der Betriebsstundenzähler Ihrer CPU entnehmen Sie den Technischen Daten.
MODE	INPUT	BYTE	E, A, M, D, L, Konstante	Auftragskennung: ■ 0: auslesen (Status wird dann in *CQ*, aktueller Wert wird in *CV* eingetragen). Läuft der Betriebsstundenzähler länger als (2 hoch 31) - 1 Stunden, dann bleibt er beim letzten darstellbaren Wert stehen und liefert die Fehlermeldung "Überlauf". ■ 1: starten (mit dem letzten Zählerstand) ■ 2: stoppen ■ 4: setzen (auf den in *PV* angegebenen Wert) ■ 5: setzen (auf den in *PV* angegebenen Wert) und starten ■ 6: setzen (auf den in *PV* angegebenen Wert) und stoppen

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
PV	INPUT	DINT	E, A, M, D, L, Konstante	Neuer Wert für den Betriebsstundenzähler
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode.
CQ	OUTPUT	BOOL	E, A, M, D, L	Status des Betriebsstundenzählers (1: läuft)
CV	OUTPUT	DINT	E, A, M, D, L	Aktueller Wert des Betriebsstundenzählers

Kompatibilität zu Programmen, für CPU mit 16-Bit-Betriebsstundenzählern

Sie können die 32-Bit-Betriebsstundenzähler auch mit den SFCs 2 SET_RTM, SFC 3 CTRL_RTM und SFC 4 READ_RTM verwenden. In diesem Fall verhalten sich die 32-Bit-Betriebsstundenzähler jedoch wie 16-Bit-Betriebsstundenzähler (Wertebereich 0 bis 32767 Stunden). Im Teillistenauszug mit der SZL-ID W#16#0132 und dem Index W#16#0008 werden Ihnen die 32-Bit- Betriebsstundenzähler 0 bis 7 als 16-Bit-Betriebsstundenzähler angezeigt. Damit können Sie Programme, die für eine CPU mit 16-Bit-Betriebsstundenzählern entwickelt wurden, und die den Teillistenauszug mit der SZL-ID W#16#0132 und dem Index W#16#0008 benutzen, weiterhin einsetzen.

RET_VAL (Rückgabewert)

Fehlercode	Erläuterung
0000h	Es ist kein Fehler aufgetreten.
8080h	Falsche Nummer des Betriebsstundenzählers
8081h	Ein negativer Wert wurde dem Parameter *PV* übergeben.
8082h	Überlauf des Betriebsstundenzählers
8091h	Der Eingangsparameter *MODE* enthält einen unzulässigen Wert.
8xyyh	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

16.1.57 SFC 102 - RD_DPARA - Vordefinierte Parameter lesen

Beschreibung

Mit der SFC 102 RD_DPARA lesen Sie den Datensatz mit der Nummer *RECNUM* einer ausgewählten Baugruppe aus den projektierten Systemdaten. Der gelesene Datensatz wird in den durch den Parameter *RECORD* aufgespannten Zielbereich eingetragen.

Arbeitsweise

Der SFC 102 RD_DPARA ist ein asynchron arbeitender SFC, d. h. die Bearbeitung erstreckt sich über mehrere SFC-Aufrufe. Sie starten den Auftrag, indem Sie die SFC 102 mit *REQ* = 1 aufrufen.

Über die Ausgangsparameter *RET_VAL* und *BUSY* wird der Zustand des Auftrags angezeigt. Siehe auch Bedeutung von *REQ*, *RET_VAL* und *BUSY* bei asynchron arbeitenden SFCs.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L	*REQ* = 1: Anforderung zum Lesen
LADDR	INPUT	WORD	E, A, M, D, L, Konstante	Adresse der Baugruppe. Bei einer Ausgabeadresse muss das höchstwertige Bit gesetzt sein.
RECNUM	INPUT	BYTE	E, A, M, D, L, Konstante	Datensatznummer (zulässige Werte: 0 ... 240)
RET_VAL	OUTPUT	INT	E, A, M, D, L	Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen Fehlercode. Falls bei der Übertragung kein Fehler auftrat, sind die folgenden Fälle zu unterscheiden: ■ *RET_VAL* enthält die Länge des tatsächlich gelesenen Datensatzes in Bytes, falls der Zielbereich größer ist als der gelesene Datensatz. ■ *RET_VAL* enthält 0, falls die Länge des gelesenen Datensatzes gleich der Länge des Zielbereichs ist.
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: Der Auftrag ist noch nicht beendet.
RECORD	OUTPUT	ANY	E, A, M, D, L	Zielbereich für den gelesenen Datensatz. Es ist nur der Datentyp BYTE zulässig. Hinweis: Beachten Sie, dass der Parameter *RECORD* bei CPUs immer die vollständige Angabe der DB-Parameter erfordert (Bsp.: P#DB13.DBX0.0 Byte 100). Das Weglassen einer expliziten DB-Nr. ist für CPUs unzulässig und führt zu einer Fehlermeldung im Anwenderprogramm.

Fehlerinformationen *Kap. 16.1.44 "SFC 57 - PARM_MOD - Modul parametrieren" Seite 909*

16.1.58 SFC 105 - READ_SI - Auslesen dyn. Systemressourcen

Übersicht

Bei der Meldungserzeugung mit der SFC 107 ALARM_DQ und SFC108 ALARM_D belegt das Betriebssystem temporär Speicherplatz im Systemspeicher.

Wenn Sie z.B. einen in der CPU vorhandenen FB mit SFC 107- oder SFC 108-Aufrufen löschen, kann es vorkommen, dass die zugehörigen Systemressourcen dauerhaft belegt bleiben. Wenn Sie den FB mit SFC 107- oder SFC 108-Aufrufen erneut laden, kann es vorkommen, dass die SFCs 107 und 108 nicht mehr ordnungsgemäß bearbeitet werden.

Beschreibung

Mit der SFC 105 READ_SI können Sie momentan verwendete Systemressourcen auslesen, die bei der Meldungserzeugung mit der SFC 107 und SFC 108 belegt wurden. Dies geschieht über die dort verwendeten Werte von *EV_ID* und *CMP_ID*. Diese werden der SFC 105 READ_SI im Parameter *SI_ID* übergeben.

Die SFC 105 READ_SI hat 4 mögliche Betriebsarten, die in der folgenden Tabelle erläutert werden. Die gewünschte Betriebsart stellen Sie über den Parameter *MODE* ein.

MODE	Welche der von SFC 107/SFC 108 belegten Systemressourcen werden ausgelesen?
1	Alle (Aufruf der SFC 105 erfolgt mit SI_ID: =0)
2	Diejenige Systemressource, die beim SFC 107-/SFC 108-Aufruf mit *EV_ID*: = ev_id belegt wurde (Aufruf der SFC 105 erfolgt mit *SI_ID*: = ev_id)
3	Alle Systemressourcen, die beim SFC 107-/SFC 108-Aufruf mit *CMP_ID*: = cmp_id belegt wurden (Aufruf der SFC 105 erfolgt mit *SI_ID*: = cmp_id)
0	Weitere Systemressourcen, die beim vorherigen Aufruf mit *MODE* =1 oder *MODE* =3 nicht ausgelesen werden konnten, weil Sie das Zielfeld *SYS_INST* zu klein gewählt hatten.

Arbeitsweise

Wenn Sie beim Aufruf der SFC 105 mit *MODE* =1 oder *MODE* =3 den Zielbereich *SYS_INST* ausreichend groß gewählt haben, enthält er nach dem Aufruf den Inhalt aller über den Parameter MODE ausgewählten Systemressourcen, die aktuell belegt sind.

Falls aktuell sehr viele Systemressourcen belegt sind, ist die SFC-Laufzeit entsprechend hoch. Bei hoher Auslastung Ihrer CPU kann dann die projektierte maximale Zyklusüberwachungszeit überschritten werden.

Dieses Laufzeitproblem können Sie wie folgt umgehen: Sie wählen den Zielbereich *SYS_INST* relativ klein. Falls die SFC nicht alle auszulesenden Systemressourcen in *SYS_INST* eintragen kann, wird Ihnen dies über *RET_VAL* = 0001h mitgeteilt. Dann rufen Sie die SFC 105 mit *MODE* =0 und gleicher *SI_ID* wie beim vorherigen Aufruf so lange auf, bis *RET_VAL* den Wert 0000h annimmt.

Da das Betriebssystem keine Koordinierung der zu einem Leseauftrag gehörenden SFC 105-Aufrufe vornimmt, sollten Sie alle SFC 105-Aufrufe in ein und derselben Prioritätsklasse durchführen.

Zielbereich SYS_INST

Der Zielbereich für die gelesenen belegten Systemressourcen muss in einem DB liegen. Sinnvollerweise definieren Sie den Zielbereich als Feld von Strukturen, wobei eine Struktur wie folgt aufgebaut ist:

Strukturelement	Datentyp	Beschreibung
SFC_NO	WORD	Nr. der SFC, die die Systemressource belegt hat
LEN	BYTE	Länge der Struktur in Bytes, inkl. *SFC_NO* und *LEN*: 0Ch
SIG_STAT	BOOL	Signalzustand
ACK_STAT	BOOL	Quittierungszustand des kommenden Ereignisses (steigende Flanke)
EV_ID	DWORD	Meldungsnummer
CMP_ID	DWORD	Kennung zur Identifikation des Teilsystems

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
MODE	INPUT	INT	E, A, M, D, L, Konstante	Auftragskennung Zulässige Werte: ■ 1: Alle Systemressourcen lesen ■ 2: Diejenige Systemressource lesen, die beim SFC 107-/SFC 108-Aufruf mit *EV_ID* = ev_id belegt wurde ■ 3: Diejenigen Systemressourcen lesen, die beim SFC 107-/SFC 108-Aufruf mit *CMP_ID* = cmp_id belegt wurden ■ 0: Folgeaufruf
SI_ID	INPUT	DWORD	E, A, M, D, L, Konstante	Kennung für die auszulesende(n) Systemressource(n) Zulässige Werte: ■ 0, falls *MODE* = 1 ■ Meldungsnummer ev_id, falls MODE = 2 ■ Kennung cmp_id zur Identifikation des Teilsystems, falls *MODE* = 3
RET_VAL	OUTPUT	INT	E, A, M, D, L	Rückgabewert (Fehlerinformation bzw.Auftragszustand)
N_SI	OUTPUT	INT	E, A, M, D, L	Anzahl der in *SYS_INT* ausgegebenen Systemressourcen
SYS_INST	OUTPUT	ANY	D	Zielbereich für die gelesenen Systemressourcen

RET_VAL (Rückgabewert)

Fehlercode	Erläuterung
0000h	Es ist kein Fehler aufgetreten.
0001h	Es konnten nicht alle Systemressourcen gelesen werden, da Sie den Zielbereich *SYS_INST* zu kurz gewählt haben.
8081h	(nur bei *MODE* =2 oder 3) Sie haben SI_ID den Wert 0 zugewiesen.
8082h	(nur bei *MODE* =1) Sie haben SI_ID einen von 0 verschiedenen Wert zugewiesen.
8083h	(nur bei *MODE* =0) Sie haben *SI_ID* einen anderen Wert zugewiesen als beim vorangegangenen SFC-Aufruf mit *MODE* =1 oder 3.
8084h	Sie haben *MODE* einen unzulässigen Wert zugewiesen.
8085h	Die SFC 105 wird bereits in einem anderen OB bearbeitet.
8086h	Der Zielbereich *SYS_INST* ist zu klein für eine Systemressource.

Fehlercode	Erläuterung
8087h oder 8092h	Der Zielbereich *SYS_INST* liegt nicht in einem DB oder Fehler im ANY-Pointer.
8xyyh	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

16.1.59 SFC 106 - DEL_SI - Freigeben dyn. belegter Systemressourcen

Übersicht

Bei der Meldungserzeugung mit der SFC 107 ALARM_DQ und SFC108 ALARM_D belegt das Betriebssystem temporär Speicherplatz im Systemspeicher.

Wenn Sie z.B. einen in der CPU vorhandenen FB mit SFC 107- oder SFC 108-Aufrufen löschen, kann es vorkommen, dass die zugehörigen Systemressourcen dauerhaft belegt bleiben. Wenn Sie den FB mit SFC 107- oder SFC 108-Aufrufen erneut laden, kann es vorkommen, dass die SFCs 107 und 108 nicht mehr ordnungsgemäß bearbeitet werden.

Beschreibung

Mit der SFC 106 DEL_SI können Sie momentan verwendete Systemressourcen löschen.

Die SFC 106 DEL_SI hat 3 mögliche Betriebsarten, die in der folgenden Tabelle erläutert werden. Die gewünschte Betriebsart stellen Sie über den Parameter *MODE* ein.

MODE	Welche der von SFC 107/SFC 108 belegten Systemressourcen werden gelöscht?
1	Alle (Aufruf der SFC 106 erfolgt mit *SI_ID*: = 0)
2	Diejenige Systemressource, die beim SFC 107-/SFC 108-Aufruf mit *EV_ID*: = ev_id belegt wurde (Aufruf der SFC 106 erfolgt mit *SI_ID*: = ev_id)
3	Alle Systemressourcen, die beim SFC 107-/SFC 108-Aufruf mit *CMP_ID*: = cmp_id belegt wurden (Aufruf der SFC 106 erfolgt mit *SI_ID*: = cmp_id)

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
MODE	INPUT	INT	E, A, M, D, L, Konstante	Auftragskennung Zulässige Werte: ■ 1: Alle Systemressourcen löschen ■ 2: Diejenige Systemressource löschen, die beim SFC 107-/SFC 108-Aufruf mit *EV_ID* = ev_id belegt wurde ■ 3: Diejenigen Systemressourcen löschen, die beim SFC 107-/SFC 108-Aufruf mit *CMP_ID* = cmp_id belegt wurden.
SI_ID	INPUT	DWORD	E, A, M, D, L, Konstante	Kennung für die zu löschende(n) System ressource(n) Zulässige Werte: ■ 0, falls *MODE* = 1 ■ Meldungsnummer ev_id, falls *MODE* = 2 ■ Kennung cmp_id zur Identifikation desTeilsystems, falls *MODE* = 3
RET_VAL	OUTPUT	INT	E, A, M, D, L	Fehlerinformation

RET_VAL (Rückgabewert)

Fehlercode	Erläuterung
0000h	Es ist kein Fehler aufgetreten.
8081h	(nur bei *MODE* = 2 oder 3) Sie haben SI_ID den Wert 0 zugewiesen.
8082h	(nur bei *MODE* = 1) Sie haben SI_ID einen von 0 verschiedenen Wert zugewiesen.
8084h	Sie haben MODE einen unzulässigen Wert zugewiesen.
8085h	Die SFC 106 wird momentan bereits bearbeitet.
8086h	Es konnten nicht alle ausgewählten Systemressourcen gelöscht werden, da mindestens eine zum Aufrufzeitpunkt der SFC 106 in Bearbeitung war.
8xyyh	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

16.1.60 SFC 107 - ALARM_DQ und SFC 108 - ALARM_D

Beschreibung

Die SFCs 107 ALARM_DQ (Erzeugung quittierbarer bausteinbezogener Meldungen) und SFC 108 ALARM_D (Erzeugung stets quittierbarer bausteinbezogener Meldungen) generieren bei jedem Aufruf eine Meldung, an die Sie einen Begleitwert anhängen können. Darin stimmen Sie mit den SFCs 17 ALARM_SQ und 18 ALARM_S überein.

Bei der Meldungserzeugung mit den SFCs 107 ALARM_DQ und 108 ALARM_D belegt das Betriebssystem für die Dauer eines Signalzyklus eine Systemressource.

Der Signalzyklus dauert bei der SFC 108 ALARM_D vom SFC-Aufruf mit *SIG* = 1 bis zum erneuten Aufruf mit *SIG* = 0. Bei der SFC 107 ALARM_DQ kommt zu dieser Zeitspanne ggf. noch die Zeit bis zur Quittierung des kommenden Signals durch eines der angemeldeten Anzeigegeräte hinzu.

Falls innerhalb des Signalzyklus ein Überladen oder Löschen des meldungserzeugenden Bausteins erfolgt, bleibt die zugehörige Systemressource bis zum nächsten Neustart belegt.

Die zusätzliche Funktionalität der SFCs 107 ALARM_DQ und 108 ALARM_D gegenüber den SFCs 17 und 18 besteht nun darin, dass Sie diese belegten Systemressourcen verwalten können.

- Mit Hilfe der SFC 105 READ_SI können Sie Informationen über belegte Systemressourcen auslesen.
- Mit der SFC 106 DEL_SI können Sie belegte Systemressourcen wieder freigeben. Dies ist insbesondere bei dauerhaft belegten Systemressourcen von Bedeutung. Eine aktuell belegte Systemressource bleibt z.B. dann bis zum nächsten Neustart belegt, wenn Sie bei einer Programmänderung einen FB-Aufruf löschen und dieser FB SFC107- oder SFC108-Aufrufe enthält. Wenn Sie bei einer Programmänderung einen FB mit SFC 107- oder SFC 108-Aufrufen erneut laden, kann es vorkommen, dass die SFCs 107 und 108 keine Meldungen mehr erzeugen.

Beschreibung Parameter

Die SFCs 107 und 108 haben einen Parameter mehr als die SFCs 17 und 18, nämlich den Eingang *CMP_ID*. Mit ihm ordnen Sie die durch die SFCs 107 und 108 erzeugten Meldungen logischen Bereichen zu, z.B. Teilanlagen. Falls Sie die SFC 107/SFC 108 in einem FB aufrufen, bietet es sich an, *CMP_ID* mit der Nummer des zugehörigen Instanz-DB zu belegen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
SIG	INPUT	BOOL	E, A, M, D, L	Das meldungsauslösende Signal
ID	INPUT	WORD	E, A, M, D, L, Konstante	Datenkanal für Meldungen: EEEEh
EV_ID	INPUT	DWORD	E, A, M, D, L, Konstante	Meldungsnummer (nicht erlaubt: 0)
CMP_ID	INPUT	DWORD	E, A, M, D, L, Konstante	Component identifier (nicht erlaubt: 0) Kennung zur Identifikation des Teilsystems, dem die zugehörige Meldung zugeordnet ist. Empfohlene Werte: ■ Low-Word: 1 ... 65535 ■ High-Word: 0 Wenn Sie sich an diese Empfehlung halten, treten keine Konflikte auf.
SD	INPUT	ANY	E, A, M, D, T, Z	Begleitwert Maximale Länge: 12 Byte Zulässig sind nur die Datentypen: BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT,DWORD, DINT, REAL, DATE, TOD, TIME,S5TIME, DATE_AND_TIME
RET_VAL	OUTPUT	INT	E, A, M, D, L	Fehlerinformation

RET_VAL (Rückgabewert)

Fehlercode	Erläuterung
0000h	Es ist kein Fehler aufgetreten.
0001h	■ Der Begleitwert ist länger als die maximal zulässige Länge, oder ■ Der Zugriff auf den Anwenderspeicher ist nicht möglich (z.B. Zugriff auf gelöschten DB). Die Meldung wird gesendet. ■ Der Begleitwert zeigt auf einen Wert im Lokaldatenbereich. Die Meldung wird gesendet. (nur S7-400)
0002h	Warnung: Der letzte freie Meldequittierspeicher wurde belegt. (nur S7-400)
8081h	Die angegebene *EV_ID* liegt außerhalb des zulässigen Bereichs.
8082h	Meldungsverlust, da Ihre CPU keine Ressourcen für die Erzeugung bausteinbezogener Meldungen durch SFCs mehr frei hat.
8083h	Meldungsverlust, da derselbe Signalwechsel bereits vorliegt, aber noch nicht gesendet werden konnte (Signaloverflow).
8084h	Beim aktuellen und beim vorangegangenen SFC 107-/SFC-108-Aufruf hat das meldungsauslösende Signal SIG denselben Wert.
8085h	Für die angegebene *EV_ID* liegt keine Anmeldung vor.
8086h	Ein SFC-Aufruf für die angegebene *EV_ID* ist bereits in einer Prioritätsklasse niedrigerer Priorität in Bearbeitung.
8087h	Beim ersten Aufruf der SFC 107/SFC 108 hatte das meldungsauslösende Signal den Wert 0.
8088h	Die angegebene *EV_ID* wird bereits von einer anderen Systemressource (zu SFC 17, 18, 107, 108) belegt.
8089h	Sie haben *CMP_ID* den Wert 0 zugewiesen.
808Ah	*CMP_ID* passt nicht zu *EV_ID*
8xyyh	Allgemeine Fehlerinformation ↳ *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

16.2 System-SFBs - *"System Function Blocks"*

16.2.1 SFB 0 - CTU - Vorwärtszählen

Beschreibung

Mit dem SFB 0 CTU können Sie vorwärts zählen. Hierbei haben Sie folgendes Verhalten:

- Wechselt der Signalzustand am Vorwärtszählereingang CU von "0" auf "1" (positive Flanke), wird der aktuelle Zählerstand um 1 erhöht und am Ausgang CV angezeigt.
- Beim ersten Mal Aufrufen und bei R="0" entspricht der Zählwert dem Vorbesetzwert am Eingang PV.
- Erreicht der Zählwert die obere Grenze 32767, wird er nicht mehr erhöht und jede weitere steigende Flanke am Eingang CU bleibt ohne Wirkung.
- Der Zähler wird auf Null zurückgesetzt bzw. gehalten, solange R Signalzustand "1" führt.
- Der Ausgang Q führt Signalzustand "1", wenn CV ≥ PV ist.
- Falls die Instanzen des SFB 0 nach Neustart initialisiert werden sollen, müssen Sie im OB 100 die zu initialisierenden Instanzen mit R = 1 aufrufen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
CU	INPUT	BOOL	E, A, M, D, L, Konstante	Zähleingang
R	INPUT	BOOL	E, A, M, D, L, Konstante	Rücksetzeingang. R dominiert gegenüber CU.
PV	INPUT	INT	E, A, M, D, L, Konstante	Vorbesetzwert
Q	OUTPUT	BOOL	E, A, M, D, L	Status des Zählers
CV	OUTPUT	INT	E, A, M, D, L	aktueller Zählwert

CU

Zähleingang:

Mit einer steigenden Flanke (gegenüber dem letzten SFB-Aufruf) am Zähleingang CU wird der Zählwert um 1 erhöht.

R

Rücksetzeingang:

Signal "1" am Eingang R bewirkt das Rücksetzen des Zählers auf den Wert 0, unabhängig davon welcher Wert an CU anliegt.

PV

Vorbesetzwert (Vergleichswert):

Ist der aktuelle Zählerstand größer oder gleich diesem Vorbesetzwert wird der Ausgang Q gesetzt.

Q

Status des Zählers:

- *Q* ist "1" wenn *CV* ≥ *PV* (aktueller Zählwert ≥ Vorbesetzwert)
- *Q* ist sonst "0"

CV

aktueller Zählerwert

- Mögliche Zählerwerte: 0 ... 32 767

16.2.2 SFB 1 - CTD - Rückwärtszählen

Beschreibung

Mit dem SFB 1 können Sie rückwärts zählen. Hierbei haben Sie folgendes Verhalten:

- Wechselt der Signalzustand am Rückwärtszählereingang *CD* von "0" auf "1" (positive Flanke), wird der aktuelle Zählerstand um 1 erniedrigt und am Ausgang *CV* angezeigt.
- Beim ersten Mal Aufrufen und bei *LOAD* = "0" entspricht der Zählwert dem Vorbesetzwert am Eingang *PV*.
- Erreicht der Zählwert die untere Grenze -32768, wird er nicht mehr erniedrigt und jede weitere steigende Flanke am Eingang *CU* bleibt ohne Wirkung.
- Der Zähler wird auf den Vorbesetzwert *PV* gesetzt bzw. gehalten, solange *LOAD* Signalzustand "1" führt.

- Der Ausgang *Q* führt Signalzustand "1", wenn *CV* ≤ 0 ist.
- Falls die Instanzen des SFB 1 nach Neustart initialisiert werden sollen, müssen Sie im OB 100 die zu initialisierenden Instanzen mit *LOAD* = 1 und *PV* = Anfangswert aufrufen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
CD	INPUT	BOOL	E, A, M, D, L, Konstante	Zähleingang
LOAD	INPUT	BOOL	E, A, M, D, L, Konstante	Ladeeingang *LOAD* dominiert gegenüber *CD*
PV	INPUT	INT	E, A, M, D, L, Konstante	Vorbesetzwert
Q	OUTPUT	BOOL	E, A, M, D, L	Status des Zählers
CV	OUTPUT	INT	E, A, M, D, L	aktueller Zählwert

CD

Zähleingang:

Mit einer steigenden Flanke (gegenüber dem letzten SFB-Aufruf) am Zähleingang *CD* wird der Zählwert um 1 erniedrigt.

LOAD

Ladeeingang:

Signal "1" am Eingang *LOAD* bewirkt, dass der Zähler auf den Vorbesetzwert PV voreingestellt wird, unabhängig davon welcher Wert an *CD* anliegt.

PV

Vorbesetzwert:

Der Zähler wird auf *PV* voreingestellt, wenn am Eingang *LOAD* Signal "1" ansteht.

Q

Status des Zählers:

- *Q* ist "1" wenn *CV* ≤ 0
- *Q* ist sonst "0"

CV

aktueller Zählerwert:

- Mögliche Zählerwerte: -32 768 ... 32 767

16.2.3 SFB 2 - CTUD - Vorwärts-/Rückwärtszählen

Beschreibung

Mit dem SFB 2 können Sie vorwärts und rückwärts zählen. Hierbei haben Sie folgendes Verhalten:

- Wechselt der Signalzustand am Vorwärtszählereingang *CU* von "0" auf "1" (positive Flanke), wird der aktuelle Zählerstand um 1 erhöht und am Ausgang *CV* angezeigt.
- Wechselt der Signalzustand am Rückwärtszählereingang *CD* von "0" auf "1" (positive Flanke), wird der aktuelle Zählerstand um 1 erniedrigt und am Ausgang *CV* angezeigt.

- Zeigen beide Zähleingänge eine positive Flanke, ändert sich der aktuelle Zählwert nicht.
- Erreicht der Zählwert die obere Grenze 32767, wird er nicht mehr erhöht und jede weitere steigende Flanke am Eingang *CU* bleibt ohne Wirkung.
- Erreicht der Zählwert die untere Grenze -32768, wird er nicht mehr erniedrigt und jede weitere steigende Flanke am Eingang *CU* bleibt ohne Wirkung.
- Der Zähler wird auf den Vorbesetzwert *PV* gesetzt bzw. gehalten, solange *LOAD* Signalzustand "1" führt.
- Der Zähler wird auf Null zurückgesetzt bzw. gehalten, solange *R* Signalzustand "1" führt. Solange der Eingang *R* Signalzustand "1" führt, bleiben positive Signalflanken und den Zählereingängen und Signalzustand "1" am Eingang *LOAD* ohne Wirkung.
- Der Ausgang *QU* führt Signalzustand "1", wenn *CV* ≥ *PV* ist.
- Der Ausgang *QD* führt Signalzustand "1", wenn *CV* ≤ 0 ist.
- Falls die Instanzen des SFB 2 nach Neustart initialisiert werden sollen, müssen Sie im OB 100 die zu initialisierenden Instanzen aufrufen:
  - bei Verwendung als Vorwärtszähler mit *R* = "1"
  - bei Verwendung als Rückwärtszähler mit *R* = "0", *LOAD* = "1" und *PV* = Anfangswert aufrufen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
CU	INPUT	BOOL	E, A, M, D, L, Konstante	Vorwärtszähleingang
CD	INPUT	BOOL	E, A, M, D, L, Konstante	Rückwärtszähleingang
R	INPUT	BOOL	E, A, M, D, L, Konstante	Rücksetzeingang, *R* dominiert gegenüber *LOAD*
LOAD	INPUT	BOOL	E, A, M, D, L, Konstante	Ladeeingang, *LOAD* dominiert gegenüber *CU* und *CD*
PV	INPUT	INT	E, A, M, D, L, Konstante	Vorbesetzwert
QU	OUTPUT	BOOL	E, A, M, D, L	Status des Vorwärtszählers
QD	OUTPUT	BOOL	E, A, M, D, L	Status des Rückwärtszählers
CV	OUTPUT	INT	E, A, M, D, L	aktueller Zählwert

CU

Vorwärts Zähleingang:

Mit einer steigenden Flanke (gegenüber dem letzten SFB-Aufruf) am Zähleingang *CU* wird der Zählwert um 1 erhöht.

CD

Rückwärts Zähleingang:

Mit einer steigenden Flanke (gegenüber dem letzten SFB-Aufruf) am Zähleingang *CD* wird der Zählwert um 1 erniedrigt.

R

Rücksetzeingang:

Signal "1" am Eingang R bewirkt das Rücksetzen des Zählers auf den Wert 0, unabhängig davon welcher Wert an *CU*, *CD* oder *LOAD* anliegt.

LOAD

Ladeeingang:

Signal "1" am Eingang *LOAD* bewirkt, dass der Zähler auf den Vorbesetzwert *PV* voreingestellt wird, unabhängig davon welche Werte an *CU* und *CD* anliegen.

PV

Vorbesetzwert:

Der Zähler wird auf *PV* voreingestellt, wenn am Eingang *LOAD* Signal "1" ansteht.

QU

Status des Vorwärtszählers:

- *QU* ist "1" wenn *CV* ≥ *PV* (aktueller Zählwert ≥ Vorbesetzwert)
- *QU* ist sonst "0"

QD

Status des Rückwärtszählers:

- *QD* ist "1" wenn 0 ≥ *CV* (aktueller Zählwert kleiner/gleich 0)
- *QD* ist sonst "0"

CV

aktueller Zählerwert

- Mögliche Zählerwerte: -32 768 ... 32 767

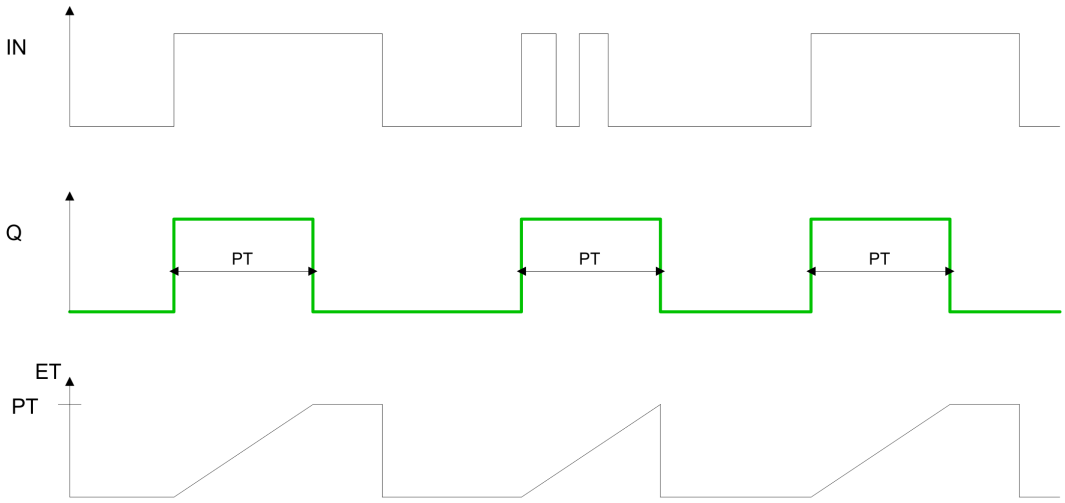
16.2.4 SFB 3 - TP - Impuls erzeugen

Beschreibung

Der SFB 3 erzeugt einen Impuls mit der Länge *PT*. Hierbei haben Sie folgendes Verhalten:

- Die Zeit läuft nur in den Betriebszuständen ANLAUF und RUN.
- Eine steigende Flanke am Eingang IN bewirkt den Start des Impulses.
- Der Ausgang *Q* bleibt für die Zeitdauer *PT* gesetzt unabhängig vom Eingangssignal.
- Der Ausgang *ET* liefert die Zeit während der der Ausgang *Q* bereits gesetzt ist. Er kann maximal den Wert des Eingangs *PT* annehmen. Er wird zurückgesetzt, wenn der Eingang *IN* nach "0" wechselt, jedoch frühestens nach Ablauf der Zeit *PT*.
- Falls die Instanzen des SFB 3 nach Neustart initialisiert werden sollen, müssen Sie im OB 100 die zu initialisierenden Instanzen mit *PT* = 0ms aufrufen.

Zeitdiagramm



Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN	INPUT	BOOL	E, A, M, D, L, Konstante	Starteingang
PT	INPUT	TIME	E, A, M, D, L, Konstante	Zeitdauer des Impulses
Q	OUTPUT	BOOL	E, A, M, D, L	Status der Zeit
ET	OUTPUT	TIME	E, A, M, D, L	abgelaufene Zeit

IN

Starteingang:

Eine steigende Flanke am Eingang *IN* bewirkt den Start des Impulses.

PT

Zeitdauer:

Die Zeitdauer muss positiv sein. Der Wertebereich ist durch den Datentyp TIME festgelegt.

Q

Ausgang *Q*:

Der Ausgang *Q* bleibt für die Zeitdauer von *PT* gesetzt, unabhängig vom weiteren Verlauf des Eingangssignals.

ET

abgelaufene Zeit:

Der Ausgang *ET* liefert die Zeit während der Ausgang *Q* bereits gesetzt ist.

Er kann maximal den Wert des Eingangs *PT* annehmen. Er wird zurückgesetzt, wenn der Eingang *IN* nach 0 wechselt, jedoch frühestens nach Ablauf der Zeit *PT*.

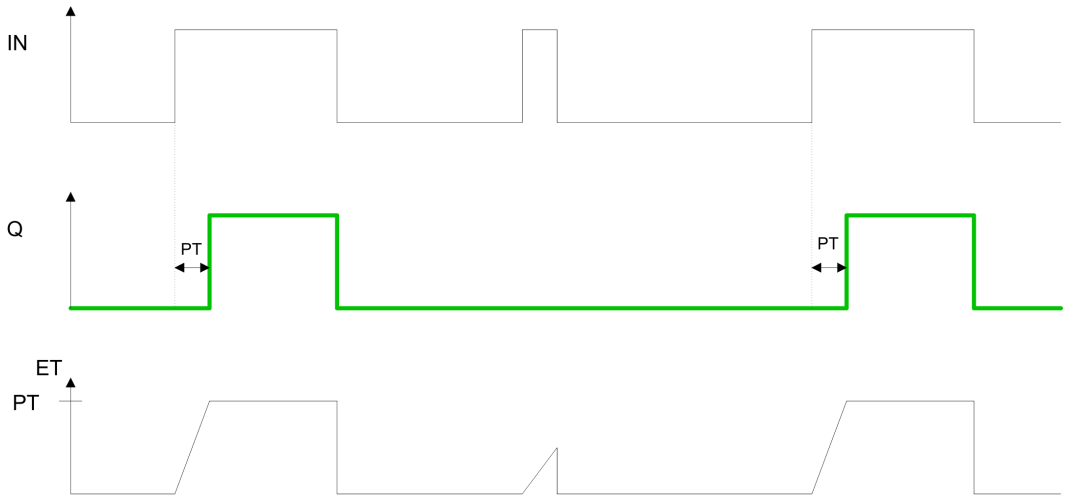
16.2.5 SFB 4 - TON - Einschaltverzögerung

Beschreibung

Der SFB 4 verzögert eine steigende Flanke um die Zeit *PT*. Hierbei haben Sie folgendes Verhalten:

- Die Zeit läuft nur in den Betriebszuständen ANLAUF und RUN.
- Eine steigende Flanke am Eingang *IN* hat nach Ablauf der Zeitdauer *PT* eine steigende Flanke am Ausgang *Q* zur Folge. *Q* bleibt so lange gesetzt, bis der Eingang *IN* nach "0" wechselt. Falls der Eingang *IN* nach "0" wechselt, bevor die Zeit *PT* abgelaufen ist, bleibt der Ausgang *Q* auf "0".
- Der Ausgang *ET* liefert die Zeit, die seit der letzten steigenden Flanke am Eingang *IN* vergangen ist, jedoch höchstens bis zum Wert des Eingangs *PT*. *ET* wird zurückgesetzt, wenn der Eingang *IN* nach "0" wechselt.
- Falls Instanzen dieses SFB nach Neustart initialisiert sein sollen, müssen Sie im OB 100 die zu initialisierenden Instanzen mit *PT* = 0 ms aufrufen.

Zeitdiagramm



Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN	INPUT	BOOL	E, A, M, D, L, Konstante	Starteingang
PT	INPUT	TIME	E, A, M, D, L, Konstante	Zeitdauer
Q	OUTPUT	BOOL	E, A, M, D, L	Status der Zeit
ET	OUTPUT	TIME	E, A, M, D, L	abgelaufene Zeit

IN

Starteingang:

Eine steigende Flanke am Eingang *IN* hat nach Ablauf der Zeitdauer *PT* eine steigende Flanke am Ausgang *Q* zur Folge.

PT

Zeitdauer:

Zeitdauer, um die die steigende Flanke am Eingang *IN* verzögert wird. *PT* muss positiv sein. Der Wertebereich ist durch den Datentyp TIME festgelegt.

Q

Ausgang Q:

Eine steigende Flanke am Eingang *IN* hat nach Ablauf der Zeitdauer *PT* eine steigende Flanke am Ausgang *Q* zur Folge. Falls der Eingang *IN* nach "0" wechselt bevor die Zeit *PT* abgelaufen ist, bleibt der Ausgang *Q* auf "0".

ET

abgelaufene Zeit:

Der Ausgang *ET* liefert die Zeit, die seit der letzten steigenden Flanke am Eingang *IN* vergangen ist, jedoch höchstens bis zum Wert des Eingangs *PT*. *ET* wird zurückgesetzt, wenn der Eingang *IN* nach "0" wechselt.

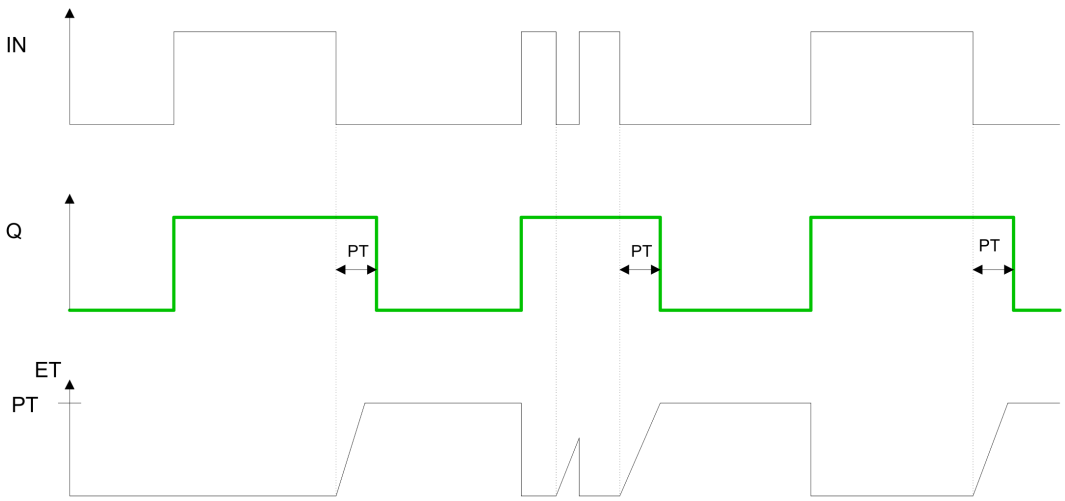
16.2.6 SFB 5 - TOF - Ausschaltverzögerung

Beschreibung

Der SFB 5 verzögert eine fallende Flanke um die Zeit *PT*. Hierbei haben Sie folgendes Verhalten:

- Die Zeit läuft nur in den Betriebszuständen ANLAUF und RUN.
- Eine steigende Flanke am Eingang *IN* bewirkt eine steigende Flanke am Ausgang *Q*. Eine fallende Flanke am Eingang *IN* hat nach Ablauf der Zeitdauer *PT* eine fallende Flanke am Ausgang *Q* zur Folge. Falls der Eingang *IN* wieder nach "1" wechselt, bevor die Zeit *PT* abgelaufen ist, bleibt der Ausgang *Q* auf "1".
- Der Ausgang *ET* liefert die Zeit, die seit der letzten fallenden Flanke am Eingang *IN* vergangen ist, jedoch höchstens bis zum Wert des Eingangs *PT*. *ET* wird zurückgesetzt, wenn der Eingang *IN* nach "1" wechselt.
- Falls die Instanzen des SFB 5 nach Neustart initialisiert werden sollen, müssen Sie im OB 100 die zu initialisierenden Instanzen mit *PT* = 0ms aufrufen.

Zeitdiagramm



Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN	INPUT	BOOL	E, A, M, D, L, Konstante	Starteingang
PT	INPUT	TIME	E, A, M, D, L, Konstante	Zeitdauer
Q	OUTPUT	BOOL	E, A, M, D, L	Status der Zeit
ET	OUTPUT	TIME	E, A, M, D, L	abgelaufene Zeit

IN

Starteingang:

Eine steigende Flanke am Eingang *IN* bewirkt eine steigende Flanke am Ausgang *Q* Eine fallende Flanke am Eingang *IN* hat nach Ablauf der Zeitdauer *PT* eine fallende Flanke am Ausgang *Q* zur Folge. Falls der Eingang *IN* wieder nach "1" wechselt, bevor die Zeit *PT* abgelaufen ist, bleibt der Ausgang *Q* auf "1".

PT

Zeitdauer:

Zeitdauer, um die die fallende Flanke am Eingang *IN* verzögert wird. *PT* muss positiv sein. Der Wertebereich ist durch den Datentyp TIME festgelegt.

Q

Ausgang *Q*:

Eine steigende Flanke am Eingang *IN* bewirkt eine steigende Flanke am Ausgang *Q*. Eine fallende Flanke am Eingang *IN* hat nach Ablauf der Zeitdauer *PT* eine fallende Flanke am Ausgang *Q* zur Folge. Falls der Eingang *IN* wieder nach "1" wechselt, bevor die Zeit *PT* abgelaufen ist, bleibt der Ausgang *Q* auf "1".

ET

abgelaufene Zeit:

Der Ausgang *ET* liefert die Zeit, die seit der letzten fallenden Flanke am Eingang *IN* vergangen ist, jedoch höchstens bis zum Wert des Eingangs *PT*. *ET* wird zurückgesetzt, wenn der Eingang *IN* nach "1" wechselt.

16.2.7 FB/SFB 12 - BSEND - Blockorientiertes Senden

Beschreibung

Mit dem FB/SFB 12 BSEND können Daten an einen remoten Partner- FB/SFB vom Typ BRCV (FB/SFB 13) gesendet werden. Der zu sendende Datenbereich wird segmentiert. Jedes Segment wird einzeln an den Partner gesendet. Das letzte Segment wird vom Partner bereits bei seiner Ankunft quittiert, unabhängig vom zugehörigen Aufruf des FB/SFB BRCV. Aufgrund der Segmentierung können Sie mit einem Sendeauftrag bis zu 65534Byte große Daten übertragen.

Bitte beachten Sie, dass dieser Baustein intern den FC bzw. SFC 202 AG_BSEND aufruft. Dieser darf nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB!

Je nach Kommunikationsfunktion haben Sie folgendes Verhalten:

- *Siemens S7-300-Kommunikation (FB 12)*
  - Der Sendevorgang erfolgt nach einer positiven Flanke an *REQ*. Mit jeder positiven Flanke an *REQ* werden die Parameter *R_ID, ID, SD_1* und *LEN* übernommen. Nach Abschluss eines Auftrags können Sie den Parametern *R_ID, ID, SD_1* und *LEN* neue Werte zuweisen. Zur Übertragung von segmentierten Daten ist der Baustein zyklisch im Anwenderprogramm aufzurufen. Die Anfangsadresse und die maximale Länge der zu sendenden Daten werden durch *SD_1* vorgegeben. Die Länge des Datenblocks legen Sie auftragsbezogen durch *LEN* fest.
- *Siemens S7-400-Kommunikation (SFB 12)*
  - Die Aktivierung des Sendevorgangs erfolgt nach Aufruf des Bausteins und positiver Flanke an *REQ*. Das Senden der Daten aus dem Anwenderspeicher erfolgt asynchron zur Bearbeitung des Anwenderprogramms. Die Anfangsadresse der zu sendenden Daten wird durch *SD_1* vorgegeben. Die Länge der Sendedaten legen Sie auftragsbezogen durch *LEN* fest. *LEN* ersetzt damit den Längenanteil von *SD_1*.

Funktion

- Bei einer positiven Flanke am Steuereingang *R* wird ein laufender Sendevorgang abgebrochen.
- Der erfolgreiche Abschluss des Sendevorgangs wird am Zustandsparameter *DONE* mit 1 angezeigt.

- Nach Abschluss eines Sendevorganges, kann erst wieder ein neuer Sendeauftrag bearbeitet werden, wenn die Zustandsparameter *DONE* oder *ERROR* den Wert 1 angenommen haben.
- Aufgrund der asynchronen Datenübertragung kann ein erneutes Senden von Daten erst gestartet werden, wenn die vorhergehenden Daten durch Aufruf des Partner-FB/SFB abgeholt wurden. Bis die Daten abgeholt werden, wird beim Aufruf des FB/SFB BSEND der Statuswert 7 ausgegeben.

Der Parameter R_ID muss bei den zusammengehörenden FB/SFBs identisch sein.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L	Steuerparameter request, aktiviert den Datenaustausch bei steigender Flanke (gegenüber letztem FB/SFB-Aufruf)
R	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter reset: Abbruch des aktuellen Auftrags
ID	INPUT	WORD	E, A, M, D, Konstante	Referenz auf die Verbindung. *ID* muss in der Form W#16#xxxx angegeben werden.
R_ID	INPUT	DWORD	E, A, M, D, L, Konstante	Adressierungsparameter *R_ID*. Er muss in der Form DW#16#wxyzWXYZ angegeben werden.
DONE	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *DONE*: ■ 0: Auftrag wurde noch nicht gestartet oder wird noch ausgeführt. ■ 1: Auftrag wurde fehlerfrei ausgeführt.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – Weder Warnung noch Fehler. ■ *ERROR* = 0 + *STATUS* ungleich 0000h – Es liegt eine Warnung vor. *STATUS* liefert detaillierte Auskunft. ■ *ERROR* = 1 – Es liegt ein Fehler vor.
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter *STATUS*, liefert detaillierte Auskunft über die Art des Fehlers.
SD_1	IN_OUT	ANY	E, A, M, D, T, Z	Zeiger auf Sendebereich. Die Längenangabe wird nur beim ersten Bausteinaufruf nach dem Start ausgewertet. Sie gibt die maximale Länge des Sendebereichs vor. Zulässig sind nur die Datentypen: BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.
LEN	IN_OUT	WORD	E, A, M, D, L	Länge des zu sendenden Datenblocks in Byte.

Fehlerinformationen

ERROR	STATUS (dezimal)	Bedeutung
0	11	Warnung: Neuer Auftrag ist unwirksam, da vorangegangener Auftrag noch nicht abgeschlossen ist.
0	25	Die Kommunikation wurde angestoßen. Der Auftrag ist in Bearbeitung.
1	1	Kommunikationsprobleme, z.B.: ■ Verbindungsbeschreibung nicht geladen (lokal oder remote) ■ Verbindung unterbrochen (z.B. Kabel, CPU ausgeschaltet, CP in STOP)
1	2	Negative Quittung vom Partner-FB/SFB. Die Funktion ist nicht ausführbar.
1	3	*R_ID* ist auf der durch *ID* vorgegebenen Kommunikationsverbindung nicht bekannt, oder der Empfangsbaustein wurde noch nie aufgerufen.
1	4	Fehler im Sendebereichszeiger *SD_1* bezüglich der Datenlänge oder des Datentyps, oder am Parameter *LEN* wurde der Wert 0 übergeben oder Fehler im Empfangsbereichszeiger *RD_1* des zugehörigen FB/SFB 13 BRCV
1	5	Resetanforderung wurde ausgeführt.
1	6	Partner-FB/SFB befindet sich im Zustand DISABLED (*EN_R* hat den Wert 0).
1	7	Partner-FB/SFB befindet sich in falschem Zustand. (Der Empfangsbaustein wurde nach der letzten Datenübertragung nicht mehr aufgerufen).
1	8	Zugriff auf remotes Objekt im Anwenderspeicher wurde abgelehnt.
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z.B. Zugriff auf gelöschten DB).
1	12	Beim Aufruf des FB/SFB wurde ■ ein Instanz-DB, der nicht zum FB/SFB 12 gehört, angegeben ■ kein Instanz-DB, sondern ein Global-DB angegeben ■ kein Instanz-DB gefunden (Laden eines neuen Instanz-DB vom PG)
1	18	*R_ID* existiert bereits in der Verbindung *ID*.
1	20	Zu wenig Arbeitsspeicher vorhanden.

Datenkonsistenz

Um sicherzustellen das die Datenkonsistenz nicht beeinträchtigt wird, darf der aktuell benutzte Teil des Sendebereichs *SD_1* erst dann wieder beschrieben werden, wenn der aktuelle Sendevorgang abgeschlossen ist. Dazu muss der Parameter *DONE* ausgewertet werden.

16.2.8 FB/SFB 13 - BRCV - Blockorientiertes Empfangen

Beschreibung

Mit dem FB/SFB 13 BRCV können Daten von einem remoten Partner- FB/SFB vom Typ BSEND (FB/SFB 12) empfangen werden, wobei darauf zu achten ist, dass der Parameter *R_ID* bei beiden FB/SFBs identisch ist. Nach jedem empfangenen Datensegment wird eine Quittung an den Partner-FB/SFB geschickt, und der Parameter LEN aktualisiert.

Bitte beachten Sie, dass dieser Baustein intern den FC bzw. SFC 203 AG_BRCV aufruft. Dieser darf nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB!

Je nach Kommunikationsfunktion haben Sie folgendes Verhalten:

- *Siemens S7-300-Kommunikation (FB 13)*
  - Mit jeder positiven Flanke an *EN_R* werden die Parameter *R_ID, ID* und *RD_1* übernommen. Nach Abschluss eines Auftrags können Sie den Parametern *R_ID, ID* und *RD_1* neue Werte zuweisen. Zur Übertragung von segmentierten Daten muss der Baustein zyklisch im Anwenderprogramm aufgerufen werden.
- *Siemens S7-400-Kommunikation (SFB 13)*
  - Der Empfang der Daten aus dem Anwenderspeicher erfolgt asynchron zur Bearbeitung des Anwenderprogramms.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN_R	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter enabled to receive, signalisiert Empfangsbereitschaft
ID	INPUT	WORD	E, A, M, D, Konstante	Referenz auf die Verbindung. *ID* muss in der Form W#16#xxxx angegeben werden.
R_ID	INPUT	DWORD	E, A, M, D, L, Konstante	Adressierungsparameter *R_ID*. Er muss in der Form DW#16#wxyzWXYZ angegeben werden.
NDR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *NDR*: neue Daten übernommen.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – Weder Warnung noch Fehler. ■ *ERROR* = 0 + *STATUS* ungleich 0000h – Es liegt eine Warnung vor. *STATUS* liefert detaillierte Auskunft. ■ *ERROR* = 1 – Es liegt ein Fehler vor.
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter *STATUS*, liefert detaillierte Auskunft über die Art des Fehlers.
RD_1	IN_OUT	ANY	E, A, M, D, T, Z	Zeiger auf Empfangsbereich. Die Längenangabe gibt die maximale Länge des zu empfangenden Blocks vor. Zulässig sind nur die Datentypen: BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.
LEN	IN_OUT	WORD	E, A, M, D, L	Länge der bisher empfangenen Daten in Byte.

Funktion

- Der FB/SFB 13 ist empfangsbereit, wenn am Steuereingang *EN_R* der Wert 1 anliegt. Durch den Parameter *RD_1* wird die Anfangsadresse des Empfangsbereichs angegeben. Der Partner-FB/SFB erhält nach jedem empfangenen Datensegment eine Quittung und der Parameter *LEN* des FB/SFB 13 wird aktualisiert. Wenn der Baustein während des asynchronen Empfangs aufgerufen wird, erfolgt die Ausgabe einer Warnung im Zustandsparameter *STATUS*.
- Wenn dieser Aufruf mit dem Wert 0 am Steuereingang *EN_R* erfolgt, wird der Empfangsvorgang abgebrochen, und der FB/SFB in seinen Grundzustand zurückversetzt. Wird der fehlerfreie Empfang sämtlicher Datensegmente erfolgreich abgeschlossen, hat der Parameter *NDR* den Wert 1. Die empfangenen Daten bleiben solange unverändert, bis der FB/SFB 13 erneut mit dem Parameter *EN_R* = 1 aufgerufen wird.

Fehlerinformationen

ERROR	STATUS (dezimal)	Bedeutung
0	11	Warnung: Neuer Auftrag ist unwirksam, da vorangegangener Auftrag noch nicht abgeschlossen ist.
0	17	Warnung: Baustein empfängt asynchron Daten.
0	25	Die Kommunikation wurde angestoßen. Der Auftrag ist in Bearbeitung.
1	1	Kommunikationsprobleme, z.B. ■ Verbindungsbeschreibung nicht geladen (lokal oder remote) ■ Verbindung unterbrochen (z.B. Kabel, CPU ausgeschaltet, CP in STOP)
1	2	Funktion nicht ausführbar.
1	4	Fehler im Empfangsbereichszeiger *RD_1* bezüglich der Datenlänge oder des Datentyps (gesendeter Datenblock ist länger als der Empfangsbereich).
1	5	Resetanforderung eingetroffen, unvollständige Übertragung.
1	8	Zugriff auf remotes Objekt im Anwenderspeicher wurde abgelehnt.
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z.B. Zugriff auf gelöschten DB).
1	12	Beim Aufruf des FB/SFB wurde ■ ein Instanz-DB, der nicht zum FB/SFB 13 gehört, angegeben ■ kein Instanz-DB, sondern ein Global-DB angegeben ■ kein Instanz-DB gefunden (Laden eines neuen Instanz-DB vom PG)
1	18	*R_ID* existiert bereits in der Verbindung *ID*.
1	20	Zu wenig Arbeitsspeicher vorhanden.

Datenkonsistenz

Die Daten werden konsistent empfangen, wenn Sie folgendes beachten:

- Nach Abschluss des Kopiervorganges (der Parameter *NDR* hat den Wert 1), muss der FB/SFB 13 erneut mit dem Wert 0 am Parameter *EN_R* aufgerufen werden, um zu gewährleisten, dass der Empfangsbereich nicht bereits wieder überschrieben wird, bevor er ausgewertet wurde.
- Der zuletzt benutzte Empfangsbereich *RD_1* muss vollständig ausgewertet werden, bevor der Baustein wieder empfangsbereit gemacht werden kann (Aufruf mit dem Wert 1 am Parameter *EN_R*).

Datenempfang Siemens S7-400

- Falls die Empfänger-CPU mit einem empfangsbereiten BRCV-Baustein (d.h. ein Aufruf mit *EN_R* =1 ist bereits erfolgt) in STOP geht, bevor der zugehörige Sendebaustein das erste Datensegment eines Auftrags abgeschickt hat, geschieht folgendes:
- Die Daten des ersten Auftrags nach Übergang der Empfänger-CPU in STOP werden vollständig in den Empfangsbereich eingetragen,
- Der Partner-SFB BSEND erhält darüber eine positive Quittung.
- Weitere BSEND-Aufträge können von der Empfänger-CPU im STOP-Zustand nicht mehr angenommen werden.
- Solange sie sich im STOP-Zustand befindet, haben *NDR* und *LEN* den Wert 0.
- Damit Ihnen die Information über die empfangenen Daten nicht verloren geht, müssen Sie bei der Empfänger-CPU einen Wiederanlauf durchführen und die SFB 13 BRCV mit *EN_R* = 1 aufrufen.

16.2.9 FB/SFB 14 - GET - Remote CPU lesen

Beschreibung

Mit dem FB/SFB 14 GET können Daten aus einer remoten CPU ausgelesen werden, wobei sich die CPU im Betriebszustand RUN oder STOP befinden kann.

Bitte beachten Sie, dass dieser Baustein intern den FC bzw. SFC 200 AG_GET aufruft. Dieser darf nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB!

Je nach Kommunikationsfunktion haben Sie folgendes Verhalten:

- *Siemens S7-300-Kommunikation (FB 14)*
  - Der Lesevorgang erfolgt nach einer positiven Flanke an *REQ*. Mit jeder positiven Flanke an *REQ* werden die Parameter *ID, ADDR_1* und *RD_1* übernommen. Nach Abschluss eines Auftrags können Sie den Parametern *ID, ADDR_1* und *RD_1* neue Werte zuweisen.
- *Siemens S7-400-Kommunikation (SFB 14)*
  - Bei einer positiven Flanke an *REQ* wird der SFB gestartet. Dabei werden die relevanten Zeiger auf die auszulesenden Bereiche (*ADDR_i*) an die Partner-CPU gesendet.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L	Steuerparameter request, aktiviert den Datenaustausch bei steigender Flanke (gegenüber letztem FB/SFB-Aufruf)
ID	INPUT	WORD	E, A, M, D, Konstante	Referenz auf die Verbindung. *ID* muss in der Form W#16#xxxx angegeben werden.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
NDR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *NDR*: Daten aus Partner-CPU übernommen.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – Weder Warnung noch Fehler. ■ *ERROR* = 0 + *STATUS* ungleich 0000h – Es liegt eine Warnung vor. *STATUS* liefert detaillierte Auskunft. ■ *ERROR* = 1 – Es liegt ein Fehler vor.
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter *STATUS*, liefert detaillierte Auskunft über die Art des Fehlers.
ADDR_1	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, die gelesen werden sollen
ADDR_2	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, die gelesen werden sollen
ADDR_3	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, die gelesen werden sollen
ADDR_4	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, die gelesen werden sollen
RD_i,1≤ i ≤4	IN_OUT	ANY	E, A, M, D, T, Z	Zeiger auf diejenigen Bereiche in der eigenen CPU, in der die gelesenen Daten abgelegt werden. Zulässig sind nur die Datentypen: BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

Funktion

- Die Remote-CPU sendet die Dateninhalte zurück und die Antwort wird daraufhin auf Zugriffsprobleme beim Lesen der Daten ausgewertet, des weiteren wird eine Datentypprüfung vorgenommen.
- Bei einer fehlerfreien Datenübertragung werden die empfangenen Daten beim nächsten Aufruf des FB/SFB 14 in die projektierten Empfangsbereiche (*RD_i*) kopiert und der Parameter *NDR* erhält den Wert 1.
- Ein erneuter Lesevorgang kann erst dann wieder aktiviert werden, wenn der letzte Lesevorgang abgeschlossen wurde. Sie müssen darauf achten, dass die über die Parameter *ADDR_i* und *RD_i* definierten Bereiche in der Anzahl, in der Länge und im Datentyp zueinander passen.

Fehlerinformationen

ERROR	STATUS (dezimal)	Bedeutung
0	11	Warnung: Neuer Auftrag ist unwirksam, da voran-gegangener Auftrag noch nicht abgeschlossen ist.
0	25	Die Kommunikation wurde angestoßen. Der Auftrag ist in Bearbeitung.

ERROR	STATUS (dezimal)	Bedeutung
1	1	Kommunikationsprobleme, z.B. ■ Verbindungsbeschreibung nicht geladen (lokal oder remote) ■ Verbindung unterbrochen (z.B.: Kabel, CPU ausgeschaltet, CP in STOP)
1	2	Negative Quittung vom Partnergerät. Die Funktion ist nicht ausführbar.
1	4	Fehler in den Empfangsbereichszeigern *RD_i* bezüglich der Datenlänge oder des Datentyps.
1	8	Zugriffsfehler bei der Partner-CPU
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z.B. Zugriff auf gelöschten DB).
1	12	Beim Aufruf des FB/SFB wurde ■ ein Instanz-DB, der nicht zum FB/SFB 14 gehört, angegeben ■ kein Instanz-DB, sondern ein Global-DB angegeben ■ kein Instanz-DB gefunden (Laden eines neuen Instanz-DB vom PG)
1	20	Zu wenig Arbeitsspeicher vorhanden.

Datenkonsistenz

Die Daten werden konsistent empfangen, wenn Sie den aktuell verwendeten des Empfangsbereichs *RD_i* vollständig auswerten, bevor Sie einen erneuten Auftrag aktivieren.

16.2.10 FB/SFB 15 - PUT - Remote CPU schreiben

Beschreibung

Mit dem FB/SFB 15 PUT können Daten in eine remote CPU geschrieben werden, wobei sich die CPU im Betriebszustand RUN oder STOP befinden kann.

Bitte beachten Sie, dass dieser Baustein intern den FC bzw. SFC 201 AG_PUT aufruft. Dieser darf nicht überschrieben werden! Der direkte Aufruf eines internen Bausteins führt zu Fehler im entsprechenden Instanz-DB!

Je nach Kommunikationsfunktion haben Sie folgendes Verhalten:

- *Siemens S7-300-Kommunikation (FB 15)*
  - Der Sendevorgang erfolgt nach einer positiven Flanke an *REQ*. Mit jeder positiven Flanke an *REQ* werden die Parameter *ID, ADDR_1* und *SD_1* übernommen. Nach Abschluss eines Auftrags können Sie den Parametern *ID, ADDR_1* und *SD_1* neue Werte zuweisen.
- *Siemens S7-400-Kommunikation (SFB 15)*
  - Bei einer positiven Flanke an *REQ* wird der SFB gestartet. Dabei werden die Zeiger auf die zu schreibenden Bereiche (*ADDR_i*) und die Daten (*SD_i*) an die Partner-CPU gesendet.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L	Steuerparameter request, aktiviert den Datenaustausch bei steigender Flanke (gegenüber letztem FB/SFB-Aufruf)
ID	INPUT	WORD	E, A, M, D, Konstante	Referenz auf die Verbindung. *ID* muss in der Form W#16#xxxx angegeben werden.
DONE	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *DONE*: Funktion ausgeführt.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – Weder Warnung noch Fehler. ■ *ERROR* = 0 + *STATUS* ungleich 0000h – Es liegt eine Warnung vor. *STATUS* liefert detaillierte Auskunft. ■ *ERROR* = 1 – Es liegt ein Fehler vor.
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter STATUS, liefert detaillierte Auskunft über die Art des Fehlers.
ADDR_1	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, in die geschrieben werden soll
ADDR_2	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, in die geschrieben werden soll
ADDR_3	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, in die geschrieben werden soll
ADDR_4	IN_OUT	ANY	z.B. E, A, M, D	Zeiger auf diejenigen Bereiche in der Partner-CPU, in die geschrieben werden soll
SD_i,1≤i ≤4	IN_OUT	ANY	E, A, M, D, T, Z	Zeiger auf diejenigen Bereiche in der eigenen CPU, die die zu versendenden Daten enthalten. Zulässig sind nur die Datentypen BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

Funktion

- Die Partner-CPU legt die gesendeten Daten unter den mitgeführten Adressen ab und sendet eine Ausführungsquittung zurück.
- Diese Ausführungsquittung wird ausgewertet und bei einer fehlerfreien Datenübertragung erhält der Parameter *DONE* beim nächsten Aufruf des FB/SFB 15 den Wert 1.
- Ein erneuter Schreibvorgang kann erst dann wieder aktiviert werden, wenn der letzte Schreibvorgang abgeschlossen wurde. Die über die Parameter *ADDR_i* und *SD_i,* 1 ≤ i ≤ 4, definierten Bereiche müssen in Anzahl, Länge und im Datentyp übereinstimmen.

Fehlerinformationen

ERROR	STATUS (dezimal)	Bedeutung
0	11	Warnung: Neuer Auftrag ist unwirksam, da vorangegangener Auftrag noch nicht abgeschlossen ist.
0	25	Die Kommunikation wurde angestoßen. Der Auftrag ist in Bearbeitung.

ERROR	STATUS (dezimal)	Bedeutung
1	1	Kommunikationsprobleme, z.B. ■ Verbindungsbeschreibung nicht geladen (lokal oder remote) ■ Verbindung unterbrochen (z.B.: Kabel, CPU ausgeschaltet, CP in STOP)
1	2	Negative Quittung vom Partnergerät. Die Funktion ist nicht ausführbar.
1	4	Fehler in den Sendebereichszeigern *SD_i* bezüglich der Datenlänge oder des Datentyps.
1	8	Zugriffsfehler bei der Partner-CPU
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z.B. Zugriff auf gelöschten DB).
1	12	Beim Aufruf des FB/SFB wurde ein Instanz-DB, der nicht zum FB/SFB 15 gehört, angegeben. kein Instanz-DB, sondern ein Global-DB angegeben. kein Instanz-DB gefunden (laden eines neuen Instanz-DB vom PG).
1	20	Zu wenig Arbeitsspeicher vorhanden.

Datenkonsistenz

- *Siemens S7-300-Kommunikation*
  - Um Datenkonsistenz zu gewährleisten, dürfen Sie den Sendebereich *SD_1* erst dann wieder beschreiben, wenn der aktuelle Sendevorgang abgeschlossen ist. Dies ist der Fall, wenn der Zustandsparameter *DONE* den Wert "1" annimmt.
- *Siemens S7-400-Kommunikation*
  - Mit dem Aktivieren eines Sendevorgangs (steigende Flanke an *REQ*) sind die zu sendenden Daten der Sendebereiche *SD_i* aus dem Anwenderprogramm kopiert. Sie können diese Bereiche nach dem Bausteinaufruf neu beschreiben, ohne die aktuellen Sendedaten zu verfälschen.

16.2.11 SFB 31 - NOTIFY_8P - Meldung ohne Quittierungsanzeige (8x)

Beschreibung

Erzeugung bausteinbezogener Meldungen ohne Quittierungsanzeige für 8 Signale.

- Der SFB 31 NOTIFY_8P ist die Erweiterung des SFB 36 NOTIFY auf 8 Signale.
- Eine Meldung wird erzeugt, wenn bei mindestens einem Signal ein Signalwechsel erkannt wurde. Beim Erstaufruf des SFB 31 wird stets eine Meldung erzeugt. Für alle 8 Signale gibt es eine gemeinsame Meldungsnummer, die am Anzeigegerät in 8 Teilmeldungen aufgesplittet wird.
- Pro Instanz des SFB 31 NOTIFY_8P steht ein Meldespeicher mit 2 Speicherplätzen zur Verfügung.
- Trotz Meldeverlust werden dem Anzeigegerät die letzten beiden Signalwechsel jedes Signals übergeben.

Bevor Sie den SFB 31 NOTIFY_8P in einem Automatisierungssystem aufrufen, müssen Sie sicherstellen, dass alle angeschlossenen Anzeigegeräte diesen Baustein kennen. Nähere Informationen hierzu finden Sie im Handbuch zu Ihren eingesetzten Komponenten.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
SIG_i	INPUT	BOOL	E, A, M, D, L	i-tes zu überwachendes Signal
ID	INPUT	WORD	Konstante (E, A, M, D, L)	Datenkanal für Meldungen: EEEEh: ID wird nur beim Erstaufruf ausgewertet.
EV_ID	INPUT	DWORD	Konstante (E, A, M, D, L)	Meldungsnummer (nicht erlaubt: 0)
SEVERITY	INPUT	WORD	Konstante (E, A, M, D, L)	Gewicht des Ereignisses
DONE	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter DONE: Meldungsgenerierung abgeschlossen.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter ERROR
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter STATUS: Anzeige einer Fehlerinformation
SD_i	IN_OUT	ANY	E, A, M, D, T, Z	i-ter Begleitwert

SIG_i i-tes zu überwachendes Signal. Es gilt 1 ≤ i ≤ 8.

ID Datenkanal für Meldungen: EEEEh. *ID* wird nur beim Erstaufruf ausgewertet.

EV_ID EV_ID wird nur beim Erstaufruf ausgewertet. Danach gilt bei jedem Aufruf des SFB mit dem zugehörigen Instanz-DB die beim Erstaufruf verwendete Meldungsnummer.

SEVERITY Gewicht des Ereignisses. Hierbei bedeutet der Wert 0 höchstes Gewicht. Dieser Parameter ist für die Bearbeitung der Meldung nicht relevant. Mögliche Werte: 0 ... 127 (Defaultwert: 64)

DONE Zustandsparameter *DONE*, Meldungsgenerierung abgeschlossen.

SD_i i-ter Begleitwert. Es gilt 1 ≤ i ≤ maxAnzahl. Die maximale Anzahl an Begleitwerten finden Sie in den technischen Daten Ihrer CPU. Zulässig sind nur die Datentypen BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND _TIME.

Wenn der ANY-Pointer auf einen DB zugreift, ist der DB immer zu spezifizieren.

(z.B.: P# DB10.DBX5.0 Byte 10).

Fehlerinformation ERROR / STATUS

ERROR = TRUE bedeutet, dass bei der Bearbeitung ein Fehler aufgetreten ist. Details hierzu finden Sie im Parameter *STATUS*. Die folgende Tabelle enthält alle spezifischen Fehlerinformationen für den SFB 31, die über die Parameter *ERROR* und *STATUS* ausgegeben werden können.

ERROR	STATUS (dezimal)	Beschreibung
0	11	Meldungsverlust: Der vorangegangene Signalwechsel bzw. die vorangegangene Meldung konnte nicht gesendet werden und wird durch die aktuelle Meldung ersetzt.
0	22	■ Fehler im Zeiger auf die Begleitwerte *SD_i*: – bezüglich der Datenlänge oder des Datentyps – Begleitwerte im Anwenderspeicher nicht erreichbar, z.B. wegen gelöschtem DB oder Bereichslängenfehler. Die aktivierte Meldung wird ohne oder ggf. mit der gerade noch möglichen Anzahl von Begleitwerten gesendet. ■ Der von Ihnen gewählte Aktualparameter von *SEVERITY* liegt oberhalb des zulässigen Bereichs. Die aktivierte Meldung wird mit *SEVERITY* = 127 gesendet.
0	25	Die Kommunikation wurde angestoßen. Die Meldung ist in Bearbeitung.
1	1	Kommunikationsprobleme: Verbindungsabbruch oder keine Anmeldung vorhanden
1	4	Beim Erstaufruf liegt die angegebene *EV_ID* außerhalb des zulässigen Bereichs oder es liegt ein formaler Fehler des ANY-Pointers *SD_i* vor oder es wurde der für die CPU pro SFB 31 maximal versendbare Speicherbereich überschritten.
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z.B. Zugriff auf gelöschten DB).
1	12	Beim Aufruf des SFB wurde ein Instanz-DB, der nicht zum SFB 31 gehört oder kein Instanz-DB, sondern ein Global-DB angegeben.
1	18	*EV_ID* wurde bereits von einem der SFBs 31 oder 33 ... 36 verwendet.
1	20	Zu wenig Arbeitsspeicher vorhanden.
1	21	Die Meldung mit der angegebenen *EV_ID* ist gesperrt.

16.2.12 SFB 32 - DRUM - Schrittschaltwerk

Beschreibung

Der SFB 32 realisiert ein Schrittschaltwerk mit maximal 16 Schritten.

- Die Nummer des ersten Schrittes geben Sie mit dem Parameter *DSP* und die Nummer des letzten Schrittes mit dem Parameter *LST_STEP* vor.
- In jedem Schritt werden alle 16 Ausgangsbits *OUT0 ... OUT15* und der Ausgangsparameter *OUT_WORD* (Ausgangsbits zusammengefasst) beschrieben.
- Der SFB 32 schaltet in den nächsten Schritt, wenn am Eingang *JOG* gegenüber dem vorangegangenen SFB-Aufruf eine positive Flanke aufgetreten ist. Falls sich der SFB bereits im letzten Schritt befindet, werden bei positiver Flanke an JOG die Variablen *Q* und *EOD* gesetzt, *DCC* erhält den Wert 0, und der SFB verbleibt im letzten Schritt, bis Sie den Eingang *RESET* mit "1" belegen.

Zeitabhängiges Weiterschalten

- Wenn Sie ein zeitabhängiges Weiterschalten in den nächsten Schritt möchten, belegen Sie den Parameter *DRUM_EN* mit "1". Der Übergang in den nächsten Schritt erfolgt dann, wenn
  - das zum aktuellen Schritt zugehörige Ereignisbit EVENTi gesetzt ist und
  - die für den aktuellen Schritt vorgegebene Zeit abgelaufen ist.
- Die Zeit ergibt sich aus dem Produkt von der Zeitbasis *DTBP* und dem für den aktuellen Schritt gültigen Zeitfaktor (Parameter *S_PRESET*).
- Wenn beim Aufruf des SFB am Eingang *RESET* "1" anliegt, geht das Schrittschaltwerk in den Schritt, dessen Nummer Sie dem Eingang *DSP* zugewiesen haben.
- Beim Erstaufruf des Bausteins müssen Sie den Eingang *RESET* mit "1" belegen.
- Wenn sich das Schaltwerk im letzten Schritt befindet (*DSC* hat den Wert *LST_STEP*) und die für diesen Schritt vorgegebene Bearbeitungszeit abgelaufen ist, werden die Ausgänge *Q* und *EOD* gesetzt, und der SFB verbleibt im letzten Schritt, bis Sie den Eingang *RESET* mit "1" belegen.
- Der SFB 32 läuft nur in den Betriebszuständen ANLAUF und RUN.
- Falls der SFB 32 nach Neustart initialisiert werden soll, müssen Sie ihn im OB 100 mit *RESET* = "1" aufrufen.

Die im aktuellen Schritt noch verbleibende Bearbeitungszeit DCC wird nur dann reduziert, wenn das zugehörige Ereignisbit EVENTi gesetzt ist.

Falls Sie DRUM_EN zu "1" gewählt haben, erreichen Sie den Sonderfall:

- *einer rein zeitgesteuerten Weiterschaltung der Schritte, indem Sie EVENTi = "1" wählen für DSP ≤ i ≤ LST_STEP.*
- *einer rein ereignisgesteuerten Weiterschaltung der Schritte über die Ereignisbits EVENTi, indem Sie Zeitbasis DTBP = "0" wählen.*

Zusätzlich können Sie das Schrittschaltwerk jederzeit (auch bei DRUM_EN = "1") über den Eingang JOG weiterschalten.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
RESET	INPUT	BOOL	E, A, M, D, L, Konstante	Rücksetzen
JOG	INPUT	BOOL	E, A, M, D, L, Konstante	Weiterschalten in den nächsten Schritt
DRUM_EN	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter
LST_STEP	INPUT	BYTE	E, A, M, D, L, Konstante	Nummer des letzten Schritts
EVENTi,1 ≤ i ≤ 16	INPUT	BOOL	E, A, M, D, L, Konstante	Ereignisbit Nr. i (gehört zum Schritt i)
OUTj,0 ≤ j ≤ 15	OUTPUT	BOOL	E, A, M, D, L	Ausgangsbit Nr. j
Q	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
OUT_WORD	OUTPUT	WORD	E, A, M, D, L, P	Ausgangsbits
ERR_CODE	OUTPUT	WORD	E, A, M, D, L, P	Tritt während der Bearbeitung des SFB ein Fehler auf, enthält *ERR_CODE* die Fehlerinformation.
JOG_HIS	VAR	BOOL	E, A, M, D, L, Konstante	Nicht anwenderrelevant
EOD	VAR	BOOL	E, A, M, D, L, Konstante	Identisch zum Ausgangsparameter *Q*
DSP	VAR	BYTE	E, A, M, D, L, P, Konstante	Nummer des ersten Schritts
DSC	VAR	BYTE	E, A ,M, D, L, P, Konstante	Nummer des aktuellen Schritts
DCC	VAR	DWORD	E, A, M, D, L, P, Konstante	noch verbleibende Bearbeitungszeit in ms
DTBP	VAR	WORD	E, A, M, D, L, P, Konstante	Zeitbasis in ms
PREV_TIME	VAR	DWORD	E, A, M, D, L, Konstante	Nicht anwenderrelevant
S_PRESET	VAR	ARRAY of WORD	E, A, M, D, L, Konstante	Zeitfaktoren für jeden Schritt.
OUT_VAL	VAR	ARRAY of BOOL	E, A, M, D, L, Konstante	in jedem Schritt auszugebenden Werte
S_MASK	VAR	ARRAY of BOOL	E, A, M, D, L, Konstante	Maskenbits für jeden Schritt

RESET

Rücksetzen:

- Der Pegel 1 bewirkt das Rücksetzen des Schrittschaltwerks.
- Beim Erstaufruf des Bausteins ist *RESET* mit "1" zu belegen.

JOG

Eine steigende Flanke (gegenüber dem letzten SFB-Aufruf) bewirkt das Weiterschalten in den nächsten Schritt, falls sich das Schaltwerk noch nicht im letzten Schritt befindet. Die Weiterschaltung erfolgt unabhängig davon, welchen Wert Sie *DRUM_EN* zugewiesen haben.

DRUM_EN

Steuerparameter, der festlegt, ob ein zeitabhängiges Weiterschalten in den nächsten Schritt möglich sein soll

("1": zeitabhängiges Weiterschalten möglich).

LST_STEP

Nummer des letzten Schrittes:

- mögliche Werte 1 ... 16

EVENTi, 1≤i≤16 Ereignisbit Nr. i (gehört zum Schritt i)

OUTj, 0≤j≤15 Ausgangsbit NR. j (identisch mit dem Bit Nr. j von *OUT_WORD*)

Q Zustandsparameter, der angibt, ob die von Ihnen vorgegebene Bearbeitungszeit des letzten Schritts abgelaufen ist.

OUT_WORD Ausgangsbits zusammengefasst in einer Variablen.

ERR_CODE Tritt während der Bearbeitung des SFB ein Fehler auf, enthält *ERR_CODE* die Fehlerinformation. ↳ *"Fehlerinformation" Seite 967*

JOG_HIS Nicht anwenderrelevant: Eingangsparameter *JOG* des vorhergehenden SFB-Aufruf.

EOD Identisch zum Ausgangsparameter *Q*.

DSP Nummer des ersten Schrittes:

- mögliche Werte 1 ... 16

DSC Nummer des aktuellen Schrittes

DCC Die im aktuellen Schritt noch verbleibende Bearbeitungszeit in ms (nur relevant, falls DRUM_EN ="1" und das zugehörige Ereignisbit = "1").

DTBP Die für alle Schritte gültige Zeitbasis in ms.

PREV_TIME Nicht anwenderrelevant: Systemzeit des vorhergehenden SFB-Aufrufs.

S_PRESET Eindimensionales Feld mit den Zeitfaktoren für alle Schritte.

- Sinnvolle Wahl der Indizes: [1 ... 16].
  In diesem Fall enthält *S_PRESET*[x] den Zeitfaktor des Schrittes x.

OUT_VAL

Zweidimensionales Feld mit den in jedem Schritt auszugebenden Werten, falls Sie diese nicht über *S_MASK* ausgeblendet haben.

- Sinnvolle Wahl der Indizes: [1 ... 16, 0 ... 15].
  In diesem Fall enthält *S_MASK* [x, y] das Maskenbit für den y-ten auszugebenden Wert im Schritt x.

S_MASK

Zweidimensionales Feld mit den Maskenbits für jeden Schritt.

- Sinnvolle Wahl der Indizes: [1 ... 16, 0 ... 15].
  In diesem Fall enthält *S_MASK* [x, y] das Maskenbit für den y-ten auszugebenden Wert im Schritt x.
- Bedeutung der Maskenbits:
  – 0: der entsprechende Wert des vorangegangenen Schrittes wird dem zugehörigen Ausgangsbit zugewiesen
  – 1: der entsprechende Wert aus *OUT_VAL* wird dem zugehörigen Ausgangsbit zugewiesen.

Fehlerinformation

ERR_CODE

- Im Fehlerfall verbleibt der SFB 32 im aktuellen Zustand und folgender Fehlercode wird ausgegeben:

ERR_CODE	Beschreibung
0000h	kein Fehler
8081h	unzulässiger Wert für *LST_STEP*
8082h	unzulässiger Wert für *DSC*
8083h	unzulässiger Wert für *DSP*
8084h	Das Produkt *DCC* = *DTBP* x *S_PRESET* [DSC] überschreitet den Wert 2^{31-1} (ca. 24,86 Tage)

16.2.13 SFB 33 - ALARM - Meldungen mit Quittierungsanzeige

Beschreibung

Erzeugung bausteinbezogener Meldungen mit Quittierungsanzeige:

- Der SFB 33 ALARM überwacht ein Signal:
  – Quittierungsgetriggertes Melden ausgeschaltet (Standardbetrieb): Der Baustein generiert sowohl bei steigender Flanke (kommendes Ereignis) als auch bei fallender Flanke (gehendes Ereignis) eine Meldung, an die Sie Begleitwerte anhängen können.
  – Quittierungsgetriggertes Melden eingeschaltet: Der Baustein erzeugt nach generierter Kommend-Meldung für das Signal so lange keine weiteren Meldungen, bis Sie diese Kommend-Meldung an einem Anzeigegerät quittiert haben.
- Beim Erstaufruf wird eine Meldung mit dem aktuellen Signalzustand gesendet. Die Meldung wird an alle dafür angemeldeten Teilnehmer gesendet.
- Nach dem Eintreffen Ihrer Quittierung von einem angemeldeten Anzeigegerät wird die Quittierinformation an alle dafür angemeldeten Teilnehmer weitergeleitet.
- Pro Instanz des SFB 33 ALARM steht ein Meldespeicher mit 2 Speicherplätzen zur Verfügung.
- Der SFB 33 ALARM entspricht der Norm IEC 1131-5.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN_R	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter
SIG	INPUT	BOOL	E, A, M, D, L	Das zu überwachende Signal

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
ID	INPUT	WORD	Konstante (E, A, M, D, L)	Datenkanal für Meldungen: EEEEh *ID* wird nur beim Erstaufruf ausgewertet.
EV_ID	INPUT	DWORD	Konstante (E, A, M, D, L)	Meldungsnummer (nicht erlaubt: 0)
SEVERITY	INPUT	WORD	Konstante (E, A, M, D, L)	Gewicht des Ereignisses
DONE	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *DONE*: Meldungsgenerierung abgeschlossen.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter *STATUS*: Anzeige einer Fehlerinformation
ACK_DN	OUTPUT	BOOL	E, A, M, D, L	Gegangenes Ereignis quittiert.
ACK_UP	OUTPUT	BOOL	E, A, M, D, L	Gekommenes Ereignis quittiert.
SD_i	IN_OUT	ANY	E, A, M, D, T, Z	i-ter Begleitwert

EN_R

Steuerparameter (enabled to receive), der bewirkt, dass die Ausgänge *ACK_UP* und *ACK_DN* beim Bausteinaufruf aktualisiert werden (*EN_R* = 1) bzw. nicht aktualisiert werden (*EN_R* = 0). Bei *EN_R* = 0 bleiben die Ausgangsparameter *ACK_UP* und *ACK_DN* unverändert.

SIG

Das zu überwachende Signal.

ID

Datenkanal für Meldungen: EEEEh. *ID* wird nur beim Erstaufruf ausgewertet.

EV_ID

EV_ID wird nur beim Erstaufruf ausgewertet. Danach gilt bei jedem Aufruf des SFB mit dem zugehörigen Instanz-DB die beim Erstaufruf verwendete Meldungsnummer. Die Meldungsnummer muss innerhalb Ihres Anwenderprogramms eindeutig sein.

SEVERITY

Gewicht des Ereignisses. Hierbei bedeutet der Wert 0 höchstes Gewicht. Dieser Parameter ist für die Bearbeitung der Meldung nicht relevant. Mögliche Werte: 0 ... 127 (Defaultwert: 64)

DONE

Zustandsparameter *DONE*, Meldungsgenerierung abgeschlossen.

ACK_DN

Gegangenes Ereignis wurde auf einem Anzeigegerät quittiert. Initialisierungszustand: 1. Der Ausgang *ACK_DN* wird rückgesetzt, wenn eine fallende Flanke vorliegt. Er wird gesetzt, wenn Ihre Quittierung des gegangenen Ereignisses von einem angemeldeten Anzeigegerät eingetroffen ist.

ACK_UP

Gekommenes Ereignis wurde auf einem Anzeigegerät quittiert. Initialisierungszustand: 1 Der Ausgang *ACK_UP* wird rückgesetzt, wenn eine steigende Flanke vorliegt. Er wird gesetzt, wenn Ihre Quittierung des gekommenen Ereignisses von einem angemeldeten Anzeigegerät eingetroffen ist.

SD_i

i-ter Begleitwert. Es gilt 1 ≤ i ≤ maxAnzahl. Die maximale Anzahl an Begleitwerten finden Sie in den technischen Daten Ihrer CPU. Zulässig sind nur die Datentypen BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND _TIME.

Wenn der ANY-Pointer auf einen DB zugreift, ist der DB immer zu spezifizieren (z.B.: P# DB10.DBX5.0 Byte 10).

Fehlerinformation *ERROR / STATUS*

ERROR = TRUE bedeutet, dass bei der Bearbeitung ein Fehler aufgetreten ist. Details hierzu finden Sie im Parameter *STATUS*. Die folgende Tabelle enthält alle spezifischen Fehlerinformationen für den SFB 33, die über die Parameter *ERROR* und *STATUS* ausgegeben werden können.

ERROR	*STATUS* (dezimal)	Beschreibung
0	11	Meldungsverlust: Der vorangegangene Signalwechsel bzw. die vorangegangene Meldung konnte nicht gesendet werden und wird durch die aktuelle Meldung ersetzt.
0	22	■ Fehler im Zeiger auf die Begleitwerte *SD_i*: – bezüglich der Datenlänge oder des Datentyps – Begleitwerte im Anwenderspeicher nicht erreichbar, z.B. wegen gelöschtem DB oder Bereichslängenfehler. Die aktivierte Meldung wird ohne oder ggf. mit der gerade noch möglichen Anzahl von Begleitwerten gesendet. ■ Der von Ihnen gewählte Aktualparameter von *SEVERITY* liegt oberhalb des zulässigen Bereichs. Die aktivierte Meldung wird mit *SEVERITY* = 127 gesendet.
0	25	Die Kommunikation wurde angestoßen. Die Meldung ist in Bearbeitung.
1	1	Kommunikationsprobleme: Verbindungsabbruch oder keine Anmeldung vorhanden. Bei aktiviertem quittierungsgetriggerten Melden: temporäre Anzeige, falls keine Anzeigegeräte das quittierungsgetriggerte Melden beherrschen.
1	4	Beim Erstaufruf liegt die angegebene *EV_ID* außerhalb des zulässigen Bereichs oder liegt ein formaler Fehler des ANY-Pointers *SD_i* vor oder wurde der für die CPU pro SFB 31 maximal versendbare Speicherbereich überschritten.
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z. B. Zugriff auf gelöschten DB).
1	12	Beim Aufruf des SFB wurde ein Instanz-DB, der nicht zum SFB 31 gehört, angegeben oder kein Instanz-DB, sondern ein Global-DB angegeben.
1	18	*EV_ID* wurde bereits von einem der SFBs 31 oder 33 ... 36 verwendet.
1	20	Zu wenig Arbeitsspeicher vorhanden.
1	21	Die Meldung mit der angegebenen *EV_ID* ist gesperrt.

Nach dem ersten Bausteinaufruf haben die Ausgänge ACK_UP und ACK_DN den Wert 1 und der Vergangenheitswert des Eingangs SIG wird zu 0 angenommen.

16.2.14 SFB 34 - ALARM_8 - Meldungen ohne Begleitwerte (8x)

Beschreibung

Erzeugung bausteinbezogener Meldungen ohne Begleitwerte für 8 Signale.

- Der SFB 34 ALARM_8 ist identisch zum SFB 35 ALARM_8P.
- Es werden nur keine Begleitwerte übertragen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN_R	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter
SIG_i	INPUT	BOOL	E, A, M, D, L	i-tes zu überwachendes Signal
ID	INPUT	WORD	Konstante (E, A, M, D, L)	Datenkanal für Meldungen: EEEEh *ID* wird nur beim Erstaufruf ausgewertet.
EV_ID	INPUT	DWORD	Konstante (E, A, M, D, L)	Meldungsnummer (nicht erlaubt: 0)
SEVERITY	INPUT	WORD	Konstante (E, A, M, D, L)	Gewicht des Ereignisses
DONE	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *DONE*: Meldungsgenerierung abgeschlossen.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter *STATUS*: Anzeige einer Fehlerinformation
ACK_STATE	OUTPUT	WORD	E, A, M, D, L	Bitfeld Quittierzustand aller 8 Meldungen

EN_R

Steuerparameter (enabled to receive), der bewirkt, dass der Ausgang *ACK_STATE* beim Bausteinaufruf aktualisiert wird (*EN_R* = 1) bzw. nicht aktualisiert wird (*EN_R* = 0).

SIG_i

i-tes zu überwachendes Signal. Es gilt $1 \leq i \leq 8$.

ID

Datenkanal für Meldungen: EEEEh. *ID* wird nur beim Erstaufruf ausgewertet.

EV_ID

EV_ID wird nur beim Erstaufruf ausgewertet. Danach gilt bei jedem Aufruf des SFB mit dem zugehörigen Instanz-DB die beim Erstaufruf verwendete Meldungsnummer. Die Meldungsnummer muss innerhalb Ihres Anwenderprogramms eindeutig sein.

SEVERITY

Gewicht des Ereignisses. Hierbei bedeutet der Wert 0 höchstes Gewicht. Dieser Parameter ist für die Bearbeitung der Meldung nicht relevant. Mögliche Werte: 0 ... 127 (Defaultwert: 64)

DONE

Zustandsparameter *DONE*: Meldungsgenerierung abgeschlossen.

ACK_STATE

Bitfeld mit dem aktuellen Quittierzustand aller 8 Meldungen

- Bit 7 ... 0: kommendes Ereignis von SIG_1 ... SIG_8
- Bit 15 ... 8: gegangene Ereignis von SIG_1 ... SIG_8

(1: Ereignis quittiert, 0: Ereignis nicht quittiert):

Initialisierungszustand: FFFFh, d.h. alle kommenden und gehenden Ereignisse sind quittiert.

Fehlerinformation *ERROR / STATUS*

ERROR = TRUE bedeutet, dass bei der Bearbeitung ein Fehler aufgetreten ist. Details hierzu finden Sie im Parameter *STATUS*. Die folgende Tabelle enthält alle spezifischen Fehlerinformationen für den SFB 34, die über die Parameter *ERROR* und *STATUS* ausgegeben werden können.

ERROR	*STATUS* (dezimal)	Beschreibung
0	11	Meldungsverlust: Der vorangegangene Signalwechsel bzw. die vorangegangene Meldung konnte nicht gesendet werden und wird durch die aktuelle Meldung ersetzt.
0	22	Der von Ihnen gewählte Aktualparameter von *SEVERITY* liegt oberhalb des zulässigen Bereichs. Die aktivierte Meldung wird mit *SEVERITY* = 127 gesendet.
0	25	Die Kommunikation wurde angestoßen. Die Meldung ist in Bearbeitung.
1	1	Kommunikationsprobleme: Verbindungsabbruch oder keine Anmeldung vorhanden Bei aktiviertem quittierungsgetriggerten Melden: temporäre Anzeige, falls keine Anzeigegeräte das quittierungsgetriggerte Melden beherrschen.
1	4	Beim Erstaufruf liegt die angegebene *EV_ID* außerhalb des zulässigen Bereichs.
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z.B. Zugriff auf gelöschten DB).
1	12	Beim Aufruf des SFB wurde ein Instanz-DB, der nicht zum SFB 34 gehört, angegeben oder kein Instanz-DB, sondern ein Global-DB angegeben.
1	18	EV_ID wurde bereits von einem der SFBs 31 oder 33 ... 36 verwendet.
1	20	Zu wenig Arbeitsspeicher vorhanden.
1	21	Die Meldung mit der angegebenen *EV_ID* ist gesperrt.

Nach dem ersten Bausteinaufruf sind alle Bits des Ausgangs ACK_STATE gesetzt, und die Vergangenheitswerte der Eingänge SIG_i, 1≤ i ≤ 8, werden zu 0 angenommen.

16.2.15 SFB 35 - ALARM_8P - Meldungen mit Begleitwerten (8x)

Beschreibung

Erzeugung bausteinbezogener Meldungen mit Begleitwerten für 8 Signale.

- Der SFB 35 ALARM_8P ist die Erweiterung des SFB 33 ALARM auf 8 Signale.
- Falls Sie das Verfahren des Quittierungsgetriggerten Meldens nicht aktiviert haben, wird stets eine Meldung generiert, wenn bei mindestens einem Signal ein Signalwechsel erkannt wurde. (Ausnahme: Beim Erstaufruf wird stets eine Meldung gesendet.) Für alle 8 Signale gibt es eine gemeinsame Meldungsnummer, die am Anzeigegerät in 8 Teilmeldungen aufgesplittet wird. Sie können jede Teilmeldung einzeln quittieren oder auch mehrere Teilmeldungen auf einmal.

- Über den Ausgangsparameter *ACK_STATE* können Sie den Quittierungszustand der einzelnen Meldungen in Ihrem Programm weiterverarbeiten. Falls Sie eine Meldung eines ALARM_8P-Bausteins sperren oder freigeben, so betrifft das immer den gesamten ALARM_8PBaustein. Das Sperren und Freigeben einzelner Signale ist nicht möglich.
- Pro Instanz des SFB 35 ALARM_8P steht ein Meldespeicher mit 2 Speicherplätzen zur Verfügung.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN_R	INPUT	BOOL	E, A, M, D, L, Konstante	Steuerparameter
SIG_i,	INPUT	BOOL	E, A, M, D, L	i-tes zu überwachendes Signal
ID	INPUT	WORD	Konstante (E, A, M, D, L)	Datenkanal für Meldungen: EEEEh ID wird nur beim Erstaufruf ausgewertet.
EV_ID	INPUT	DWORD	Konstante (E, A, M, D, L)	Meldungsnummer (nicht erlaubt: 0)
SEVERITY	INPUT	WORD	Konstante (E, A, M, D, L)	Gewicht des Ereignisses
DONE	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *DONE*: Meldungsgenerierung abgeschlossen.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter *STATUS*: Anzeige einer Fehlerinformation
ACK_STATE	OUTPUT	WORD	E, A, M, D, L	Bitfeld Quittierzustand aller 8 Meldungen
SD_j	IN_OUT	ANY	E, A, M, D, T, Z	j-ter Begleitwert

EN_R

Steuerparameter (enabled to receive), der bewirkt, dass der Ausgang *ACK_STATE* beim Bausteinaufruf aktualisiert wird (*EN_R* = 1) bzw. nicht aktualisiert wird (*EN_R* = 0).

SIG_i

i-tes zu überwachendes Signal. Es gilt 1 ≤ i ≤ 8.

ID

Datenkanal für Meldungen: EEEEh. ID wird nur beim Erstaufruf ausgewertet.

EV_ID

EV_ID wird nur beim Erstaufruf ausgewertet. Danach gilt bei jedem Aufruf des SFB mit dem zugehörigen Instanz-DB die beim Erstaufruf verwendete Meldungsnummer. Die Meldungsnummer muss innerhalb Ihres Anwenderprogramms eindeutig sein.

SEVERITY

Gewicht des Ereignisses. Hierbei bedeutet der Wert 0 höchstes Gewicht. Dieser Parameter ist für die Bearbeitung der Meldung nicht relevant. Mögliche Werte: 0 ... 127 (Defaultwert: 64)

DONE

Zustandsparameter *DONE*, Meldungsgenerierung abgeschlossen.

ACK_STATE

Bitfeld mit dem aktuellen Quittierzustand aller 8 Meldungen.

- Bit 7 ... 0: kommendes Ereignis von SIG_1 ... SIG_8
- Bit 15 ... 8: gegangene Ereignis von SIG_1 ... SIG_8

(1: Ereignis quittiert, 0: Ereignis nicht quittiert):

Initialisierungszustand: FFFFh, d.h. alle kommenden und gehenden Ereignisse sind quittiert.

SD_i

i-ter Begleitwert. Es gilt 1 ≤ i ≤ maxAnzahl. Die maximale Anzahl an Begleitwerten finden Sie in den technischen Daten Ihrer CPU. Zulässig sind nur die Datentypen BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND _TIME.

Wenn der ANY-Pointer auf einen DB zugreift, ist der DB immer zu spezifizieren (z.B.: P# DB10.DBX5.0 Byte 10).

Fehlerinformation *ERROR / STATUS*

ERROR = TRUE bedeutet, dass bei der Bearbeitung ein Fehler aufgetreten ist. Details hierzu finden Sie im Parameter *STATUS*. Die folgende Tabelle enthält alle spezifischen Fehlerinformationen für den SFB 35, die über die Parameter *ERROR* und *STATUS* ausgegeben werden können.

ERROR	*STATUS* (dezimal)	Beschreibung
0	11	Meldungsverlust: Der vorangegangene Signalwechsel bzw. die vorangegangene Meldung konnte nicht gesendet werden und wird durch die aktuelle Meldung ersetzt.
0	22	■ Fehler im Zeiger auf die Begleitwerte SD_i: – bezüglich der Datenlänge oder des Datentyps – Begleitwerte im Anwenderspeicher nicht erreichbar, z. B. wegen gelöschtem DB oder Bereichslängenfehler. Die aktivierte Meldung wird ohne Begleitwerte gesendet. ■ Der von Ihnen gewählte Aktualparameter von *SEVERITY* liegt oberhalb des zulässigen Bereichs. Die aktivierte Meldung wird mit *SEVERITY* = 127 gesendet.
0	25	Die Kommunikation wurde angestoßen. Die Meldung ist in Bearbeitung.
1	1	Kommunikationsprobleme: Verbindungsabbruch oder keine Anmeldung vorhanden. Bei aktiviertem quittierungsgetriggerten Melden: temporäre Anzeige, falls keine Anzeigegeräte das quittierungsgetriggerte Melden beherrschen.
1	4	Beim Erstaufruf liegt die angegebene *EV_ID* außerhalb des zulässigen Bereichs oder es liegt ein formaler Fehler des ANY-Pointers SD_i vor oder es wurde der für die CPU pro SFB 35 maximal versendbare Speicherbereich überschritten
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z.B. Zugriff auf gelöschten DB)
1	12	Beim Aufruf des SFB wurde ein Instanz-DB, der nicht zum SFB 34 gehört, angegeben oder kein Instanz-DB, sondern ein Global-DB angegeben.
1	18	EV_ID wurde bereits von einem der SFBs 31 oder 33 ... 36 verwendet.

ERROR	*STATUS* (dezimal)	Beschreibung
1	20	Zu wenig Arbeitsspeicher vorhanden.
1	21	Die Meldung mit der angegebenen *EV_ID* ist gesperrt.

Nach dem ersten Bausteinaufruf sind alle Bits des Ausgangs ACK_STATE gesetzt, und die Vergangenheitswerte der Eingänge SIG_i, 1≤ i ≤ 8, werden zu 0 angenommen.

16.2.16 SFB 36 - NOTIFY - Meldungen ohne Quittierungsanzeige

Beschreibung

Erzeugung bausteinbezogener Meldungen ohne Quittierungsanzeige

- Der SFB 36 NOTIFY überwacht ein Signal. Er generiert sowohl bei steigender Flanke (kommendes Ereignis) als auch bei fallender Flanke (gehendes Ereignis) eine Meldung, an die Sie Begleitwerte anhängen können.
- Beim Erstaufruf wird eine Meldung mit dem aktuellen Signalzustand gesendet. Die Meldung wird an alle dafür angemeldeten Teilnehmer gesendet.
- Die Begleitwerte werden zum Zeitpunkt der Flankenauswertung erfasst und der Meldung zugeordnet.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
SIG	INPUT	BOOL	E, A, M, D, L.	Das zu überwachende Signal.
ID	INPUT	WORD	Konstante (E, A, M, D, L)	Datenkanal für Meldungen: EEEEh *ID* wird nur beim Erstaufruf ausgewertet.
EV_ID	INPUT	DWORD	Konstante (E, A, M, D, L)	Meldungsnummer (nicht erlaubt: 0)
SEVERITY	INPUT	WORD	Konstante (E, A, M, D, L)	Gewicht des Ereignisses
DONE	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *DONE*: Meldungsgenerierung abgeschlossen.
ERROR	OUTPUT	BOOL	E, A, M, D, L	Zustandsparameter *ERROR*
STATUS	OUTPUT	WORD	E, A, M, D, L	Zustandsparameter *STATUS*: Anzeige einer Fehlerinformation
SD_i,	IN_OUT	ANY	E, A, M, D, T, Z	i-ter Begleitwert

SIG

Das zu überwachende Signal.

ID

Datenkanal für Meldungen: EEEEh. *ID* wird nur beim Erstaufruf ausgewertet.

EV_ID

EV_ID wird nur beim Erstaufruf ausgewertet. Danach gilt bei jedem Aufruf des SFB mit dem zugehörigen Instanz-DB die beim Erstaufruf verwendete Meldungsnummer. Die Meldungsnummer muss innerhalb Ihres Anwenderprogramms eindeutig sein.

SEVERITY

Gewicht des Ereignisses. Hierbei bedeutet der Wert 0 höchstes Gewicht. Dieser Parameter ist für die Bearbeitung der Meldung nicht relevant. Mögliche Werte: 0 ... 127 (Defaultwert: 64)

DONE

Zustandsparameter *DONE*: Meldungsgenerierung abgeschlossen.

SD_i

i-ter Begleitwert. Es gilt 1 ≤ i ≤ maxAnzahl. Die maximale Anzahl an Begleitwerten finden Sie in den technischen Daten Ihrer CPU. Zulässig sind nur die Datentypen BOOL (nicht erlaubt: Bitfeld), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND _TIME.

Wenn der ANY-Pointer auf einen DB zugreift, ist der DB immer zu spezifizieren (z.B.: P# DB10.DBX5.0 Byte 10).

Fehlerinformation *ERROR / STATUS*

Die folgende Tabelle enthält alle spezifischen Fehlerinformationen für den SFB 36, die über die Parameter *ERROR* und *STATUS* ausgegeben werden können.

ERROR	*STATUS* (dezimal)	Beschreibung
0	11	Meldungsverlust: Der vorangegangene Signalwechsel bzw. die vorangegangene Meldung konnte nicht gesendet werden und wird durch die aktuelle Meldung ersetzt.
0	22	■ Fehler im Zeiger auf die Begleitwerte SD_i: – bezüglich der Datenlänge oder des Datentyps – Begleitwerte im Anwenderspeicher nicht erreichbar, z.B. wegen gelöschtem DB oder Bereichslängenfehler. Die aktivierte Meldung wird ohne oder ggf. mit der gerade noch möglichen Anzahl von Begleitwerten gesendet. ■ Der von Ihnen gewählte Aktualparameter von *SEVERITY* liegt oberhalb des zulässigen Bereichs. Die aktivierte Meldung wird mit *SEVERITY* = 127 gesendet.
0	25	Die Kommunikation wurde angestoßen. Die Meldung ist in Bearbeitung.
1	1	Kommunikationsprobleme: Verbindungsabbruch oder keine Anmeldung vorhanden
1	4	Beim Erstaufruf liegt die angegebene *EV_ID* außerhalb des zulässigen Bereichs oder es liegt ein formaler Fehler des ANY-Pointers *SD_i* vor oder es wurde der für die CPU pro SFB 36 maximal versendbare Speicherbereich überschritten.
1	10	Zugriff auf lokalen Anwenderspeicher nicht möglich (z.B. Zugriff auf gelöschten DB).
1	12	Beim Aufruf des SFB wurde: ein Instanz-DB, der nicht zum SFB 36 gehört oder kein Instanz-DB, sondern ein Global-DB angegeben.
1	18	*EV_ID* wurde bereits von einem der SFBs 31 oder 33 ... 36 verwendet.

ERROR	*STATUS* (dezimal)	Beschreibung
1	20	Zu wenig Arbeitsspeicher vorhanden.
1	21	Die Meldung mit der angegebenen *EV_ID* ist gesperrt.

16.2.17 SFB 47 - COUNT - Zähler steuern

Beschreibung

Bei dem SFB 47 handelt es sich um einen speziell für Kompakt-CPUs entwickelten Baustein, der zur Ansteuerung der Zähler dient. Der SFB ist immer zyklisch mit zugehörigem Instanz-DB aufzurufen. Hier liegen die Parameter für den SFB ab. Mit dem SFB COUNT (SFB 47) stehen Ihnen folgende Funktionalitäten zur Verfügung:

- Zähler Starten/Stoppen mit dem Softwaretor *SW_GATE*
- Freigabe/Steuern des digitalen Ausgang
- Auslesen von Statusbits
- Auslesen von Zähler- und Latchwert
- Aufträge zum Lesen und Schreiben der internen Zählregister

Parameter

Name	Datentyp	Adresse (Instanz-DB)	Defaultwert	Kommentar
LADDR	WORD	0.0	300h	Dieser Parameter wird nicht ausgewertet. Es wird immer die interne Ein-/ Ausgabe-Peripherie angesprochen.
CHANNEL	INT	2.0	0	Kanalnummer
SW_GATE	BOOL	4.0	FALSE	Softwaretor freigegeben
CTRL_DO	BOOL	4.1	FALSE	Ausgang DO freigegeben False: Standard Digitaler Ausgang
SET_DO	BOOL	4.2	FALSE	Parameter wird nicht ausgewertet
JOB_REQ	BOOL	4.3	FALSE	Auftragsanstoß (Flanke 0-1)
JOB_ID	WORD	6.0	0	Auftragsnummer
JOB_VAL	DINT	8.0	0	Wert für schreibende Aufträge
STS_GATE	BOOL	12.0	FALSE	Status internes Tor
STS_STRT	BOOL	12.1	FALSE	Status Hardwaretor
STS_LTCH	BOOL	12.2	FALSE	Status Latch-Eingang
STS_DO	BOOL	12.3	FALSE	Status Ausgang
STS_C_DN	BOOL	12.4	FALSE	Status Richtung rückwärts. Es wird immer die letzte Zählrichtung angezeigt. Nach dem ersten Aufruf des SFB hat *STS_C_DN* den Wert FALSE.
STS_C_UP	BOOL	12.5	FALSE	Status Richtung vorwärts. Es wird immer die letzte Zählrichtung angezeigt. Nach dem ersten Aufruf des SFB hat *STS_C_UP* den Wert TRUE.
COUNTVAL	DINT	14.0	0	Aktueller Zählwert
LATCHVAL	DINT	18.0	0	Aktueller Latchwert
JOB_DONE	BOOL	22.0	TRUE	Neuer Auftrag kann gestartet werden
JOB_ERR	BOOL	22.1	FALSE	Auftrag fehlerhaft
JOB_STAT	WORD	24.0	0	Auftragsfehler-Nummer

Lokaldaten nur im Instanz-DB

Name	Datentyp	Adresse (Instanz-DB)	Defaultwert	Kommentar
RES00	BOOL	26.0	FALSE	reserviert
RES01	BOOL	26.1	FALSE	reserviert
RES02	BOOL	26.2	FALSE	reserviert
STS_CMP	BOOL	26.3	FALSE	Status Vergleicher * Das Statusbit STS_CMP zeigt an, dass die Vergleichsbedingung des Komparators erfüllt ist oder erfüllt war. Mit STS_CMP wird auch angezeigt, dass der Ausgang gesetzt war (*STS_DO* = TRUE).
RES04	BOOL	26.4	FALSE	reserviert
STS_OFLW	BOOL	26.5	FALSE	Status Überlauf *
STS_UFLW	BOOL	26.6	FALSE	Status Unterlauf *
STS_ZP	BOOL	26.7	FALSE	Status Nulldurchgang * Wird nur gesetzt beim Zählen ohne Hauptzählrichtung. Zeigt Nulldurchgang an. Wird auch gesetzt, wenn der Zähler auf 0 gesetzt wird oder der Zähler ab *Ladewert* = 0 zählt.
JOB_OVAL	DINT	28.0		Ausgabewert für Leseaufträge
RES10	BOOL	32.0	FALSE	reserviert
RES11	BOOL	32.1	FALSE	reserviert
RES_STS	BOOL	32.2	FALSE	Status-Bits rücksetzen: Setzt die Status-Bits: STS_CMP, STS_OFLW, STS_ZP zurück. Zum Zurücksetzen der Statusbits werden zwei Aufrufe des SFB benötigt.

*) wird mit RES_STS zurückgesetzt

Pro Kanal dürfen Sie den SFB immer nur mit dem gleichen Instanz-DB aufrufen, da hier die für den internen Ablauf erforderlichen Daten abgelegt werden. Ein schreibender Zugriff auf Ausgänge des Instanz-DB ist nicht zulässig.

Zähler Auftragsschnittstelle

Zum Beschreiben bzw. Auslesen der Zählregister steht ihnen mit dem SFB 47 eine Auftragsschnittstelle zur Verfügung. Damit ein neuer Auftrag ausgeführt werden kann, muss immer der letzte Auftrag mit *JOB_DONE* = TRUE abgeschlossen sein.

Vorgehensweise Der Einsatz der Auftragsschnittstelle erfolgt nach folgendem Ablauf:

1. Versorgen Sie folgende Eingangsparameter:

Name	Datentyp	Adresse (DB)	Default	Kommentar
JOB_REQ	BOOL	4.3	FALSE	Auftragsanstoß (Flanke 0-1) *
JOB_ID	WORD	6.0	0	Auftragsnummer: 00h Auftrag ohne Funktion 01h *Zählwert* schreiben 02h *Ladewert* schreiben 04h *Vergleichswert* schreiben 08h *Hysterese* schreiben 10h *Impulsdauer* schreiben 20h *Endwert* schreiben 82h *Ladewert* lesen 84h *Vergleichswert* lesen 88h *Hysterese* lesen 90h *Impulsdauer* lesen A0h *Endwert* lesen
JOB_VAL	DINT	8.0	0	Wert für schreibende Aufträge

*) Zustand bleibt auch nach einem CPU STOP-RUN-Übergang gesetzt.

2. Rufen Sie den SFB auf. Der Auftrag wird sofort bearbeitet. *JOB_DONE* geht für den Durchlauf des SFB auf FALSE. Im Fehlerfall wird *JOB_ERR* = TRUE gesetzt und die Fehlerursache in *JOB_STAT* zurückgeliefert:

Name	Datentyp	Adresse (DB)	Default	Kommentar
JOB_DONE	BOOL	22.0	TRUE	Neuer Auftrag kann gestartet werden
JOB_ERR	BOOL	22.1	FALSE	Auftrag fehlerhaft
JOB_STAT	WORD	24.0	0000h	Auftragsfehler-Nummer 0000h kein Fehler 0121h *Vergleichswert* zu klein 0122h *Vergleichswert* zu groß 0131h *Hysterese* zu klein 0132h *Hysterese* zu groß 0141h *Impulsdauer* zu klein 0142h *Impulsdauer* zu groß 0151h *Ladewert* zu klein 0152h *Ladewert* zu groß 0161h *Zählerstand* zu klein 0162h *Zählerstand* zu groß 01FFh *Auftrags-Nr.* ungültig

3. Mit *JOB_DONE* = TRUE kann ein neuer Auftrag gestartet werden.

4. Bei Leseaufträgen finden Sie den zu lesenden Wert im Parameter *JOB_OVAL* im Instanz-DB auf Adresse 28.

Zulässiger Wertebereich für JOB_VAL

Endlos Zählen:

Auftrag	Gültiger Wertebereich
Zähler direkt schreiben	-2147483647 (-2^{31}+1) ... +2147483646 (2^{31}-2)
Ladewert schreiben	-2147483647 (-2^{31}+1) ... +2147483646 (2^{31}-2)
Vergleichswert schreiben	-2147483648 (-2^{31}) ... +2147483647 (2^{31}-1)
Hysterese schreiben	0 ... 255
Impulsdauer schreiben*	0 ... 510ms

Einmalig/Periodisch Zählen, keine Hauptzählrichtung:

Auftrag	Gültiger Wertebereich
Zähler direkt schreiben	-2147483647 (-2^{31}+1) ... +2147483646 (2^{31}-2)
Ladewert schreiben	-2147483647 (-2^{31}+1) ... +2147483646 (2^{31}-2)
Vergleichswert schreiben	-2147483648 (-2^{31}) ... +2147483647 (2^{31}-1)
Hysterese schreiben	0 ... 255
Impulsdauer schreiben*	0 ... 510ms

Einmalig/Periodisch Zählen, Hauptzählrichtung vorwärts:

Auftrag	Gültiger Wertebereich
Endwert	2 ... +2147483646 (2^{31}-1)
Zähler direkt schreiben	-2147483648 (-2^{31}) ... *Endwert* -2
Ladewert schreiben	-2147483648 (-2^{31}) ... *Endwert* -2
Vergleichswert schreiben	-2147483648 (-2^{31}) ... *Endwert* -1
Hysterese schreiben	0 ... 255
Impulsdauer schreiben*	0 ... 510ms

Einmalig/Periodisch Zählen, Hauptzählrichtung rückwärts:

Auftrag	Gültiger Wertebereich
Zähler direkt schreiben	2 ... +2147483647 (2^{31}-1)
Ladewert schreiben	2 ... +2147483647 (2^{31}-1)
Vergleichswert schreiben	1 ... +2147483647 (2^{31}-1)
Hysterese schreiben	0 ... 255

Auftrag	Gültiger Wertebereich
Impulsdauer schreiben*	0 ... 510ms

*) Es sind nur gerade Werte erlaubt. Ungerade Werte werden automatisch abgerundet.

Latch-Funktion

Sobald während eines Zählvorgangs am "Latch"-Eingang eines Zählers eine Flanke 0-1 auftritt, wird der aktuelle Zählerwert im entsprechenden Latch-Register gespeichert.

Mit dem Parameter *LATCHVAL* des SFB 47 haben Sie Zugriff auf den Latch-Wert.

Nach einem STOP-RUN-Übergang der CPU bleibt ein zuvor in *LATCHVAL* geladener Wert erhalten.

16.2.18 SFB 48 - FREQUENC - Frequenzmessung steuern

Beschreibung

Bei dem SFB 48 handelt es sich um einen speziell für Kompakt-CPUs entwickelten Baustein, welcher der Frequenzmessung dient.

- Zur Steuerung des Frequenzmessers ist der SFB FREQUENC zyklisch (z.B. OB 1) zu verwenden.
- Der SFB ist mit zugehörigem Instanz-DB aufzurufen. Hier liegen die Parameter für den SFB ab.
- Unter anderem bietet der SFB 48 eine Auftragsschnittstelle. Hiermit haben Sie lesenden und schreibenden Zugriff auf die Register des Frequenzmessers.
- Damit ein neuer Auftrag ausgeführt werden kann, muss immer der letzte Auftrag mit *JOB_DONE* = TRUE abgeschlossen sein.
- Pro Kanal dürfen Sie den SFB immer nur mit dem gleichen Instanz-DB aufrufen, da hier die für den internen Ablauf erforderlichen Daten abgelegt werden. Ein schreibender Zugriff auf Ausgänge des Instanz-DB ist nicht zulässig.
- Mit dem SFB FREQUENC (SFB 48) stehen Ihnen folgende Funktionalitäten zur Verfügung:
  - Frequenzmesser Starten/Stoppen mit dem Softwaretor SW_GATE
  - Auslesen von Statusbits
  - Auslesen der berechneten Frequenz
  - Aufträge zum Lesen und Schreiben der internen Register des Frequenzmessers.

Parameter

Name	Deklaration	Datentyp	Adresse (Inst.-DB)	Default Wert	Kommentar
LADDR	INPUT	WORD	0.0	300h	Dieser Parameter wird nicht ausgewertet. Es wird immer die interne Ein-/ Ausgabe-Peripherie angesprochen.
CHANNEL	INPUT	INT	2.0	0	Kanalnummer
SW_GATE	INPUT	BOOL	4.0	FALSE	Softwaretor freigegeben
JOB_REQ	INPUT	BOOL	4.3	FALSE	Auftragsanstoß (Flanke 0-1)
JOB_ID	INPUT	WORD	6.0	0	Auftragsnummer
JOB_VAL	INPUT	DINT	8.0	0	Wert für schreibende Aufträge
STS_GATE	OUTPUT	BOOL	12.0	FALSE	Status internes Tor
MEAS_VAL	OUTPUT	DINT	14.0	0	Berechnete Frequenz

Name	Deklaration	Datentyp	Adresse (Inst.-DB)	Default Wert	Kommentar
JOB_DONE	OUTPUT	BOOL	22.0	TRUE	Neuer Auftrag kann gestartet werden.
JOB_ERR	OUTPUT	BOOL	22.1	FALSE	Auftrag fehlerhaft
JOB_STAT	OUTPUT	WORD	24.0	0	Auftragsfehler-Nummer

Lokaldaten nur im Instanz-DB

Name	Datentyp	Adresse (Instanz-DB)	Default	Kommentar
JOB_OVAL	DINT	28.0	-	Ausgabewert für Leseaufträge

Pro Kanal dürfen Sie den SFB immer nur mit dem gleichen Instanz-DB aufrufen, da hier die für den internen Ablauf erforderlichen Daten abgelegt werden. Ein schreibender Zugriff auf Ausgänge des Instanz-DB ist nicht zulässig.

Frequenzmesser Auftragsschnittstelle

Zum Beschreiben bzw. Auslesen der Register des Frequenzmessers steht ihnen mit dem SFB 48 eine Auftragsschnittstelle zur Verfügung.

Damit ein neuer Auftrag ausgeführt werden kann, muss immer der letzte Auftrag mit *JOB_DONE* = TRUE abgeschlossen sein.

Vorgehensweise

Der Einsatz der Auftragsschnittstelle erfolgt nach folgendem Ablauf:

→ Versorgen Sie folgende Eingangsparameter:

Name	Datentyp	Adresse (DB)	Default	Kommentar
JOB_REQ	BOOL	4.3	FALSE	Auftragsanstoß (Flanke 0-1)
JOB_ID	WORD	6.0	0	Auftragsnummer: 00h Auftrag ohne Funktion 04h Integrationszeit schreiben 84h Integrationszeit lesen
JOB_VAL	DINT	8.0	0	Wert für schreibende Aufträge. Zulässiger Wertebereich für Integrationszeit: 10 ... 10000ms

→ Rufen Sie den SFB auf. Der Auftrag wird sofort bearbeitet. *JOB_DONE* geht für den Durchlauf des SFB auf FALSE. Im Fehlerfall wird *JOB_ERR* = TRUE gesetzt und die Fehlerursache in *JOB_STAT* zurückgeliefert.

Name	Datentyp	Adresse (DB)	Default	Kommentar
JOB_DONE	BOOL	22.0	TRUE	Neuer Auftrag kann gestartet werden
JOB_ERR	BOOL	22.1	FALSE	Auftrag fehlerhaft
JOB_STAT	WORD	24.0	0000h	Auftragsfehler-Nummer 0000h kein Fehler 0221h Integrationszeit zu klein 0222h Integrationszeit zu groß 02FFh Auftrags-Nr. ungültig 8001h Parametrierfehler 8009h Kanal-Nr. ungültig

1. Mit *JOB_DONE* = TRUE kann ein neuer Auftrag gestartet werden.
2. Bei Leseaufträgen finden Sie den zu lesenden Wert im Parameter *JOB_OVAL* im Instanz-DB auf Adresse 28.

Kanal-Nr. ungültig

(8009h und Parametrierfehler 8001h)

Haben Sie mit CHANNEL eine Kanal-Nr. größer 3 übergeben, bekommen Sie die Rückmeldung Kanal-Nr. (8009h) ungültig. Sofern Sie mit CHANNEL eine Kanal-Nr. übergeben haben, die größer ist als die maximal verfügbare Kanalanzahl der CPU, bekommen Sie einen Parametrierfehler (8001h).

Frequenzmesser steuern

Gesteuert wird der Frequenzmesser über das interne Tor (I-Tor). Das I-Tor ist identisch mit dem Software-Tor (SW-Tor).

SW-Tor:

öffnen (aktivieren): Im Anwenderprogramm durch Setzen von *SW_GATE* des SFB 48

schließen (deaktivieren): Im Anwenderprogramm durch Rücksetzen von *SW_GATE* des SFB 48

16.2.19 SFB 49 - PULSE - Pulsweitenmodulation

Beschreibung

Bei dem SFB 49 handelt es sich um einen speziell für Kompakt-CPUs entwickelten Baustein, der zur *PWM*-Ausgabe und *Pulse Train* Ausgabe dient. Mit dem SFB PULSE (SFB 49) stehen Ihnen folgende Funktionalitäten zur Verfügung:

- PWM (**P**uls**w**eiten**m**odulation)
  - Starten/Stoppen mit dem Softwaretor *SW_EN*
  - Freigabe/Steuern des PWM-Ausgangs
  - Auslesen von Status-Bits
  - Aufträge zum Lesen und Schreiben der internen PWM-Register
- Konfigurierbare Pulse Train Ausgabe mit maximal 2 Fahraufträgen
  - Starten/Stoppen mit dem Softwaretor *SW_EN*
  - Freigabe/Steuern des Pulse-Train-Ausgangs
  - Auslesen von Status-Bits
  - Aufträge zum Lesen und Schreiben der internen Pulse-Train-Register
- Konfigurierbare Zeitbasis (1µs ... 1ms)

Beim Einsatz des Bausteins ist folgendes zu beachten:

- Der SFB ist mit zugehörigem Instanz-DB zyklisch z.B. im OB 1 aufzurufen.
- Über die Auftragsschnittstelle des SFB 49 haben Sie lesenden und schreibenden Zugriff auf die entsprechenden Register.
- Pro Kanal dürfen Sie den SFB immer nur mit dem gleichen Instanz-DB aufrufen. Ein schreibender Zugriff auf Ausgänge des Instanz-DB ist nicht zulässig.
- Damit ein neuer Auftrag ausgeführt werden kann, muss immer der letzte Auftrag mit *JOB_DONE* = TRUE abgeschlossen sein.
- Die Umschaltung zwischen den Modi erfolgt durch die Vorgabe der Pulsanzahl (JOB_ID = 08h/09h). Sobald Sie eine Pulsanzahl > 0 angeben, erfolgt eine Umschaltung in die Pulse Train Betriebsart, ansonsten ist PWM aktiv.

Bitte beachten Sie, dass manche Funktionalitäten dieses Bausteins nicht in allen CPUs verfügbar sind. Bei Aufruf einer Funktionalität, welche nicht unterstützt wird, erhalten Sie die Fehlermeldung 04FFh "Auftrags-Nr. ungültig" als Rückgabewert zurück. Näheres zu den unterstützten Funktionalitäten finden Sie auch in den "Leistungsmerkmalen" Ihrer CPU.

Parameter

Parameter	Deklaration	Datentyp	Adresse (Inst.-DB)	Default Wert	Kommentar
LADDR	INPUT	WORD	0.0	300h	Dieser Parameter wird nicht ausgewertet. Es wird immer die interne Ein-/Ausgabe-Peripherie angesprochen.
CHANNEL	INPUT	INT	2.0	0	Kanalnummer
SW_EN	INPUT	BOOL	4.0	FALSE	Softwaretor freigegeben
MAN_DO	INPUT	BOOL	4.1	FALSE	Dieser Parameter wird nicht ausgewertet.
SET_DO	INPUT	BOOL	4.2	FALSE	Dieser Parameter wird nicht ausgewertet.
OUTP_VAL	INPUT	INT	6.0	0	Ausgabewert ↳ *"OUTP_VAL" Seite 985*
JOB_REQ	INPUT	BOOL	8.0	FALSE	Auftragsanstoß (Flanke 0-1)
JOB_ID	INPUT	WORD	10.0	0	Auftragsnummer ↳ *"JOB_ID" Seite 985*
JOB_VAL	INPUT	DINT	12.0	0	Wert für schreibende Aufträge
STS_EN	OUTPUT	BOOL	16.0	FALSE	Status internes Tor
STS_STRT	OUTPUT	BOOL	16.1	FALSE	Dieser Parameter ist reserviert.
STS_DO	OUTPUT	BOOL	16.2	FALSE	Dieser Parameter ist reserviert.
JOB_DONE	OUTPUT	BOOL	16.3	TRUE	Zustandsparameter ■ 0: Auftrag wurde noch nicht gestartet oder wird noch ausgeführt. ■ 1: Auftrag wurde ausgeführt. Ein neuer Auftrag kann gestartet werden.

Parameter	Deklaration	Datentyp	Adresse (Inst.-DB)	Default Wert	Kommentar
JOB_ERR	OUTPUT	BOOL	16.4	FALSE	Zustandsparameter ■ 0: kein Fehler ■ 1: Fehler (siehe *JOB_STAT*)
JOB_STAT	OUTPUT	WORD	18.0	0	↬ *"Rückgabewert JOB_STAT" Seite 990*

OUTP_VAL

Über die Hardwarekonfiguration können Sie für PWM und Pulse Train das *"Ausgabeformat"* einstellen. Je nach Ausgabeformat ergeben sich folgende Wertebereiche für den *Ausgabewert*:

- Ausgabe in ‰
  - Wertebereich: 0 ... 1000
  - *Impulsdauer* = (*OUTP_VAL* / 1000) x *Periodendauer*
- Ausgabeformat: S7 Analogwert
  - *Impulsdauer* = (*OUTP_VAL* / 27648) x *Periodendauer*
  - Wertebereich: 0 ... 27648

JOB_ID

Auftragsnummer

- 00h: Auftrag ohne Funktion
- 01h: *Periodendauer* schreiben für PWM und 1. Pulse Train Auftrag
  Wertebereich in Abhängigkeit von der Zeitbasis:
  - 1ms: 1 ... 87
  - 0,1ms: 1 ... 870
  - 10µs: 2 ... 8700
  - 1µs: 20 ... 65535
- 02h: *Einschaltverzögerung* schreiben
  Wertebereich in Abhängigkeit von der Zeitbasis:
  - 1ms: 0 ... 65535
  - 0,1ms: 0 ... 65535
  - 10µs: 0 ... 65535
  - 1µs: 0 ... 65535
- 04h: *Mindestimpulsdauer* schreiben
  Wertebereich in Abhängigkeit von der Zeitbasis:
  - 1ms: 0 ... Periodendauer/2
  - 0,1ms: 0 ... Periodendauer/2
  - 10µs: 0 ... Periodendauer/2
  - 1µs: 5 ... Periodendauer/2
- 08h: *Anzahl der Pulse* schreiben für den 1. Pulse Train Auftrag
  Wertebereich:
  - 0 ... 8.388.607
- 09h: *Anzahl der Pulse* schreiben für den 2. Pulse Train Auftrag
  Wertebereich:
  - 0 ... 8.388.607
- 0Ah: *Periodendauer* schreiben für 2. Pulse Train Auftrag
- 0Bh: *Zeitbasis* schreiben
  - 00h: 0,1ms
  - 01h: 1ms
  - 02h: 1µs
  - 03h: 10µs

- 0Ch: 2. Pulse Train Auftrag an den 1. Pulse Train Auftrag anhängen
  - Bei Einsatz dieser Auftragsnummer ist zusätzlich über *OUTP_VAL* das Tastverhältnis für den 2. Pulse Train Auftrag anzugeben.
- 81h: *Periodendauer* von PWM und 1. Pulse Train Auftrag lesen
- 82h: *Einschaltverzögerung* lesen
- 84h: *Mindestimpulsdauer* lesen
- 88h: *Anzahl der Impulse* von 1. Pulse Train Auftrag lesen
- 89h: *Anzahl der Impulse* von 2. Pulse Train Auftrag lesen
- 8Ah: *Periodendauer* von 2. Pulse Train Auftrag lesen
- 8Bh: *Zeitbasis* lesen
  - 00h: 0,1ms
  - 01h: 1ms
  - 02h: 1µs
  - 03h: 10µs

JOB_VAL

Wert für schreibende Aufträge, dessen Wertebereich vom jeweiligen Auftrag abhängt:

-2147483648 (-2^{31}) ... +2147483647 (2^{31}-1)

Lokaldaten nur im Instanz-DB

Name	Datentyp	Adresse (Instanz-DB)	Default	Kommentar
JOB_OVAL	DINT	20.0	-	Ausgabewert für Leseaufträge

Pro Kanal dürfen Sie den SFB immer nur mit dem gleichen Instanz-DB aufrufen, da hier die für den internen Ablauf erforderlichen Daten abgelegt werden. Ein schreibender Zugriff auf Ausgänge des Instanz-DB ist nicht zulässig.

Auftragsschnittstelle

- Zum Beschreiben bzw. Auslesen der Register steht ihnen mit dem SFB 49 eine Auftragsschnittstelle zur Verfügung.
- Damit ein neuer Auftrag ausgeführt werden kann, muss immer der letzte Auftrag mit *JOB_DONE* = TRUE abgeschlossen sein.
- Mit einer Flanke 0-1 an *JOB_REQ* können Sie jederzeit einen Auftrag übergeben, unabhängig vom Zustand von *SW_EN* und *STS_EN*.
- Änderungen der *Periodendauer* und der *Mindestimpulsdauer* werden sofort wirksam.
- Änderungen der *Einschaltverzögerung* werden mit der nächsten Flanke 0-1 von *SW_EN* wirksam.
- Eine laufende PWM-Ausgabe wird durch Einstellen von Pulse Train spezifischen Werten wie *Pulsanzahl* und *Periodendauer* für den 2. Pulse Train Auftrag nicht beeinflusst.

Ausgabe steuern

PWM-Ausgabe steuern

Der Einsatz der Auftragsschnittstelle erfolgt nach folgendem Ablauf:

1. Rufen Sie den SFB 49 auf:
   - *SW_EN* = FALSE
   - *JOB_VAL* = Geben Sie hier einen Wert für die *Periodendauer* an
   - *JOB_ID* = 01h: *Periodendauer* schreiben für PWM-Ausgabe.
   - *JOB_REQ* = TRUE (Flanke 0-1)

   ⇨
   - Aus *JOB_VAL* wird die Periodendauer für die PWM-Ausgabe übermittelt.
   - *JOB_DONE* geht für den Durchlauf des SFB auf FALSE.
   - Im Fehlerfall wird *JOB_ERR* = TRUE gesetzt und die Fehlerursache in *JOB_STAT* zurückgeliefert.
2. Zum Zurücksetzen von *JOB_REQ* rufen Sie den SFB 49 mit den gleichen Parametern und *JOB_REQ* = FALSE erneut auf.
3. Rufen Sie den SFB 49 auf:
   - *SW_EN* = FALSE
   - *JOB_VAL* = Geben Sie hier einen Wert für die *Einschaltverzögerung* an
   - *JOB_ID* = 02h: *Einschaltverzögerung* schreiben für PWM-Ausgabe.
   - *JOB_REQ* = TRUE (Flanke 0-1)

   ⇨
   - Aus *JOB_VAL* wird die *Einschaltverzögerung* für die PWM-Ausgabe übermittelt.
   - *JOB_DONE* geht für den Durchlauf des SFB auf FALSE.
   - Im Fehlerfall wird *JOB_ERR* = TRUE gesetzt und die Fehlerursache in *JOB_STAT* zurückgeliefert.
4. Zum Zurücksetzen von *JOB_REQ* rufen Sie den SFB 49 mit den gleichen Parametern und *JOB_REQ* = FALSE erneut auf.
5. Rufen Sie den SFB 49 auf:
   - *SW_EN* = FALSE
   - *JOB_VAL* = Geben Sie hier einen Wert für die *Mindestimpulsdauer* an
   - *JOB_ID* = 04h: *Mindestimpulsdauer* schreiben für PWM-Ausgabe.
   - *JOB_REQ* = TRUE (Flanke 0-1)

   ⇨
   - Aus *JOB_VAL* wird die *Mindestimpulsdauer* für die PWM-Ausgabe übermittelt.
   - *JOB_DONE* geht für den Durchlauf des SFB auf FALSE.
   - Im Fehlerfall wird *JOB_ERR* = TRUE gesetzt und die Fehlerursache in *JOB_STAT* zurückgeliefert.
6. Zum Zurücksetzen von *JOB_REQ* rufen Sie den SFB 49 mit den gleichen Parametern und *JOB_REQ* = FALSE erneut auf.
7. Rufen Sie den SFB 49 auf:
   - *SW_EN* = TRUE (Flanke 0-1)
   - *JOB_REQ* = TRUE (Flanke 0-1)
   - *OUTP_VAL*: Geben Sie hier das Tastverhältnis vor.

   ⇨
   - Die PWM-Ausgabe wird gestartet.
   - *STS_EN* geht auf TRUE und bleibt in diesem Zustand, bis SFB 49 mit *SW_EN* = FALSE aufgerufen wird.
   - Im Fehlerfall wird *JOB_ERR* = TRUE gesetzt und die Fehlerursache in *JOB_STAT* zurückgeliefert.
8. Rufen Sie zyklisch den SFB 49 auf:
   - *SW_EN* = TRUE
   - Über *STS_EN* erhalten Sie den aktuellen Status der PWM-Ausgabe. Mit *OUTP_VAL* können Sie jederzeit das Tastverhältnis ändern.

9. Sobald *JOB_DONE* TRUE zurück liefert, können Sie durch Wiederholen der Schritte 1 bis 5 die PWM-Parameter ändern.

> *Werden Werte während der PWM-Ausgabe geändert, so werden die neuen Werte erst mit dem Anfang einer neuen Periode ausgegeben. Eine gestartete Periode wird immer zu Ende geführt!*

10. Durch Rücksetzen von *SW_EN* (*SW_EN* = FALSE) wird die Ausgabe sofort gestoppt.

11. Bei Leseaufträgen finden Sie den zu lesenden Wert im Parameter *JOB_OVAL* im Instanz-DB auf Adresse 20.

Pulse Train Ausgabe steuern

Der Einsatz der Auftragsschnittstelle erfolgt nach folgendem Ablauf:

1. Rufen Sie den SFB 49 auf:

- *SW_EN* = FALSE
- *JOB_VAL* = Geben Sie hier einen Wert für die *Anzahl der Pulse* an.
- *JOB_ID* = 08h: *Anzahl der Pulse* schreiben für den 1. Pulse Train Auftrag.
- *JOB_REQ* = TRUE (Flanke 0-1)

⇨
- Aus *JOB_VAL* wird die *Anzahl der Pulse* für den 1. Pulse Train Auftrag übermittelt.
- *JOB_DONE* geht für den Durchlauf des SFB auf FALSE.
- Im Fehlerfall wird *JOB_ERR* = TRUE gesetzt und die Fehlerursache in *JOB_STAT* zurückgeliefert.

2. Zum Zurücksetzen von *JOB_REQ* rufen Sie den SFB 49 mit den gleichen Parametern und *JOB_REQ* = FALSE erneut auf.

3. Rufen Sie den SFB 49 auf:

- *SW_EN* = FALSE
- *JOB_VAL* = Geben Sie hier einen Wert für die *Periodendauer* an.
- *JOB_ID* = 01h: *Periodendauer* schreiben für den 1. Pulse Train Auftrag.
- *JOB_REQ* = TRUE (Flanke 0-1)

⇨
- Aus *JOB_VAL* wird die *Periodendauer* für den 1. Pulse Train Auftrag übermittelt.
- *JOB_DONE* geht für den Durchlauf des SFB auf FALSE.
- Im Fehlerfall wird *JOB_ERR* = TRUE gesetzt und die Fehlerursache in *JOB_STAT* zurückgeliefert.

4. Zum Zurücksetzen von *JOB_REQ* rufen Sie den SFB 49 mit den gleichen Parametern und *JOB_REQ* = FALSE erneut auf.

5. Optional für den 2. Pulse Train Auftrag: Rufen Sie den SFB 49 auf:

- *SW_EN* = FALSE
- *JOB_VAL* = Geben Sie hier einen Wert für die *Anzahl der Pulse* an.
- *JOB_ID* = 09h: *Anzahl der Pulse* schreiben für den 2. Pulse Train Auftrag.
- *JOB_REQ* = TRUE (Flanke 0-1)

⇨
- Die *Anzahl der Pulse* wird für den 2. Pulse Train Auftrag übermittelt.
- *JOB_DONE* geht für den Durchlauf des SFB auf FALSE.
- Im Fehlerfall wird *JOB_ERR* = TRUE gesetzt und die Fehlerursache in *JOB_STAT* zurückgeliefert.

6. Zum Zurücksetzen von *JOB_REQ* rufen Sie den SFB 49 mit den gleichen Parametern und *JOB_REQ* = FALSE erneut auf.

7. Optional für den 2. Puls Train Auftrag: Rufen Sie den SFB 49 auf:

- *SW_EN* = FALSE
- *JOB_VAL* = Geben Sie hier einen Wert für die *Periodendauer* an.
- *JOB_ID* = 0Ah: *Periodendauer* schreiben für den 2. Pulse Train Auftrag.
- *JOB_REQ* = TRUE (Flanke 0-1)

⇨
- Aus *JOB_VAL* wird die *Periodendauer* für den 2. Pulse Train Auftrag übermittelt.
- *JOB_DONE* geht für den Durchlauf des SFB auf FALSE.
- Im Fehlerfall wird *JOB_ERR* = TRUE gesetzt und die Fehlerursache in *JOB_STAT* zurückgeliefert.

8. Zum Zurücksetzen von *JOB_REQ* rufen Sie den SFB 49 mit den gleichen Parametern und *JOB_REQ* = FALSE erneut auf.

9. Rufen Sie den SFB 49 auf:

- *SW_EN* = TRUE (Flanke 0-1)
- *JOB_REQ* = TRUE (Flanke 0-1)
- *OUTP_VAL*: Geben Sie hier das Tastverhältnis vor wie z.B. 50%.

⇨
- Der 1. Puls Train Auftrag wird gestartet und danach falls vorhanden der 2. Puls Train Auftrag.
- Über *STS_EN* erhalten Sie den aktuellen Status der Pulse Train Ausgabe. Solange die geforderte Anzahl an Pulsen ausgegeben wird, liefert *STS_EN* TRUE zurück. *STS_EN* liefert FALSE zurück, wenn entweder die angeforderte Anzahl an Pulsen ausgegeben wurde oder die Ausgabe mit *SW_EN* = FALSE frühzeitig beendet wurde.
- Im Fehlerfall wird *JOB_ERR* = TRUE gesetzt und die Fehlerursache in *JOB_STAT* zurückgeliefert.

10. Zum Zurücksetzen von *JOB_REQ* rufen Sie den SFB 49 mit den gleichen Parametern und *JOB_REQ* = FALSE erneut auf.

11. Rufen Sie zyklisch den SFB 49 auf:

- *SW_EN* = TRUE
- Über *STS_EN* erhalten Sie den aktuellen Status der Pulse Train Ausgabe.

12. Sobald *JOB_DONE* TRUE zurück liefert, können Sie durch Wiederholen der Schritte 1 bis 6 weitere Puls Train Aufträge ausgeben.

13. Durch Rücksetzen von *SW_EN* (*SW_EN* = FALSE) wird die Ausgabe sofort gestoppt.

14. Bei Leseaufträgen finden Sie den zu lesenden Wert im Parameter *JOB_OVAL* im Instanz-DB auf Adresse 20.

Laufenden Pulse Train Aufrag erweitern

Solange nur ein Puls Train Auftrag definiert ist, und dieser aktuell abgearbeitet wird, haben Sie die Möglichkeit einen 2. Puls Train Auftrag an den 1. Puls Train Auftrag anzuhängen.

1. Rufen Sie den SFB 49 auf:

- *SW_EN* = TRUE (Flanke 0-1)
- *JOB_VAL* = Geben Sie hier einen Wert für die *Anzahl der Pulse* an.
- *JOB_ID* = 09h: *Anzahl der Pulse* schreiben für den 2. Pulse Train Auftrag.
- *JOB_REQ* = TRUE (Flanke 0-1)

⇨
- Aus *JOB_VAL* wird die *Anzahl der Pulse* für den 2. Pulse Train Auftrag übermittelt.
- *JOB_DONE* geht für den Durchlauf des SFB auf FALSE.
- Im Fehlerfall wird *JOB_ERR* = TRUE gesetzt und die Fehlerursache in *JOB_STAT* zurückgeliefert.

2. Zum Zurücksetzen von *JOB_REQ* rufen Sie den SFB 49 mit den gleichen Parametern und *JOB_REQ* = FALSE erneut auf.
3. Rufen Sie den SFB 49 auf:
   - *SW_EN* = TRUE
   - *JOB_VAL* = Geben Sie hier einen Wert für die *Periodendauer* an.
   - *JOB_ID* = 0Ah: *Periodendauer* schreiben für den 2. Pulse Train Auftrag.
   - *JOB_REQ* = TRUE (Flanke 0-1)

   ⇨
   - Aus *JOB_VAL* wird die *Periodendauer* für den 2. Pulse Train Auftrag übermittelt.
   - *JOB_DONE* geht für den Durchlauf des SFB auf FALSE.
   - Im Fehlerfall wird *JOB_ERR* = TRUE gesetzt und die Fehlerursache in *JOB_STAT* zurückgeliefert.
4. Zum Zurücksetzen von *JOB_REQ* rufen Sie den SFB 49 mit den gleichen Parametern und *JOB_REQ* = FALSE erneut auf.
5. Rufen Sie den SFB 49 auf:
   - *SW_EN* = TRUE (Flanke 0-1)
   - *JOB_ID* = 0Ch: 2. Pulse Train Auftrag an den 1. Pulse Train Auftrag anhängen
   - *JOB_REQ* = TRUE (Flanke 0-1)
   - *OUTP_VAL*: Geben Sie hier das Tastverhältnis vor wie z.B. 50%.

   ⇨
   - Solange der 1. Puls Train Auftrag noch läuft, wird der 2. Puls Train Auftrag angehängt. Ansonsten erhalten Sie die Fehlermeldung 0461h als *Rückgabewert*.
   - Über *STS_EN* erhalten Sie den aktuellen Status der Pulse Train Ausgabe. Solange die geforderte Anzahl an Pulsen ausgegeben wird, liefert *STS_EN* TRUE zurück. *STS_EN* liefert FALSE zurück, wenn entweder die angeforderte Anzahl an Pulsen ausgegeben wurde oder die Ausgabe mit *SW_EN* = FALSE frühzeitig beendet wurde.
   - Im Fehlerfall wird *JOB_ERR* = TRUE gesetzt und die Fehlerursache in *JOB_STAT* zurückgeliefert.

Bitte beachten Sie, dass maximal 2 Pulse Train Aufträge direkt hintereinander ausgeführt werden können!

Rückgabewert *JOB_STAT* Über den Rückgabewert JOB_STAT erhalten Sie detaillierte Informationen im Fehlerfall.

Wert	Beschreibung
0000h	kein Fehler
0411h	*Periodendauer* zu klein
0412h	*Periodendauer* zu groß
0421h	*Einschaltverzögerung* zu klein
0422h	*Einschaltverzögerung* zu groß
0431h	*Mindestimpulsdauer* zu klein
0432h	*Mindestimpulsdauer* zu groß
0441h	*Anzahl der Pulse* zu niedrig
0442h	*Anzahl der Pulse* zu hoch

Wert	Beschreibung
0451h	Unzulässige *Zeitbasis*
0461h	Pulse Train Auftrag konnte nicht angehängt werden
04FFh	Auftrags-Nr. ungültig Diese Fehlermeldung erhalten Sie, wenn beispielsweise die entsprechende Funktionalität von Ihrer CPU nicht unterstützt wird.
8001h	Parametrierfehler Haben Sie mit *CHANNEL* eine Kanal-Nr. übergeben, die größer ist als die maximal verfügbare Kanalanzahl der CPU, bekommen Sie einen Parametrierfehler (8001h).
8009h	Kanal-Nr. ungültig Haben Sie mit *CHANNEL* eine Kanal-Nr. größer 3 übergeben, bekommen Sie die Rückmeldung Kanal-Nr. ungültig (8009h).

16.2.20 SFB 52 - RDREC - Datensatz lesen

Die Schnittstelle des SFB 52 ist identisch mit der des in der Norm "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3" definierten FB RDREC.

Beschreibung

Mit dem SFB 52 RDREC (read record) lesen Sie den Datensatz mit der Nummer *INDEX* von dem mittels ID adressierten Modul. Mit *MLEN* geben Sie vor, wie viele Bytes Sie maximal lesen möchten. Den Zielbereich *RECORD* sollten Sie daher mindestens *MLEN* Bytes lang wählen. Den Wert TRUE des Ausgangsparameters *VALID* zeigt an, dass der Datensatz erfolgreich in den Zielbereich RECORD übertragen wurde. In diesem Fall erhält der Ausgangsparameter LEN die Länge der gelesenen Daten in Bytes. Falls bei der Datensatzübertragung ein Fehler auftrat, wird dies über den Ausgangsparameter *ERROR* angezeigt. Der Ausgangsparameter *STATUS* enthält in diesem Fall die Fehlerinformation. Systembedingt ist dieser Baustein nicht unterbrechbar!

Funktion

Der SFB 52 RDREC ist ein asynchron arbeitender SFB, d.h. die Bearbeitung erstreckt sich über mehrere SFB-Aufrufe. Sie starten die Datensatzübertragung, indem Sie den SFB 52 mit *REQ* = 1 aufrufen. Über die Ausgangsparameter *BUSY* und die Bytes 2 und 3 des Ausgangsparameters *STATUS* wird der Zustand des Auftrages angezeigt. Dabei entsprechen die Bytes 2 und 3 von *STATUS* dem Ausgangsparameter *RET_VAL* der asynchron arbeitenden SFCs (siehe auch Bedeutung von *REQ*, *RET_VAL* und *BUSY* bei asynchron arbeitenden SFCs). Die Datensatzübertragung ist abgeschlossen, wenn der Ausgangsparameter *BUSY* den Wert FALSE angenommen hat.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	REQ = 1: Datensatzübertragung durchführen
ID	INPUT	DWORD	E, A, M, D, L, Konstante	Logische Adresse des Moduls Bei einem Ausgabemodul muss Bit 15 gesetzt werden (z.B. für Adresse 5: *ID*: = 8005h). Bei einem Mischmodul ist die kleinere der beiden Adressen anzugeben.
INDEX	INPUT	INT	E, A, M, D, L, Konstante	Datensatznummer
MLEN	INPUT	INT	E, A, M, D, L, Konstante	Maximale Länge der zu lesenden Datensatzinformation in Bytes
VALID	OUTPUT	BOOL	E, A, M, D, L	Neuer Datensatz wurde empfangen und ist gültig
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: Der Lesevorgang ist noch nicht beendet.
ERROR	OUTPUT	BOOL	E, A, M, D, L	*ERROR* = 1: Beim Lesevorgang trat ein Fehler auf.
STATUS	OUTPUT	DWORD	E, A, M, D, L	Aufrufkennung (Bytes 2 und 3) bzw. Fehlercode
LEN	OUTPUT	INT	E, A, M, D, L	Länge der gelesenen Datensatzinformation
RECORD	IN_OUT	ANY	E, A, M, D, L	Zielbereich für den gelesenen Datensatz.

Fehlerinformationen

Kap. 16.2.22 "SFB 54 - RALRM - Alarm von einer Peripheriebaugruppe empfangen" Seite 993

16.2.21 SFB 53 - WRREC - Datensatz schreiben

Die Schnittstelle des SFB 53 ist identisch mit der des in der Norm "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3" definierten FB WRREC.

Beschreibung

Mit dem SFB 53 WRREC (write record) übertragen Sie den Datensatz *RECORD* zu dem mittels ID adressierten Modul. Mit *LEN* geben Sie die Länge des zu übertragenden Datensatzes in Bytes vor. Den Wert TRUE des Ausgangsparameters *DONE* zeigt an, dass der Datensatz erfolgreich zum DP-Slave übertragen wurde. Falls bei der Datensatzübertragung ein Fehler auftrat, wird dies über den Ausgangsparameter *ERROR* angezeigt. Der Ausgangsparameter *STATUS* enthält in diesem Fall die Fehlerinformation. Systembedingt ist dieser Baustein nicht unterbrechbar!

Funktion

Der SFB 53 WRREC ist ein asynchron arbeitender SFB, d.h. die Bearbeitung erstreckt sich über mehrere SFB-Aufrufe. Sie starten die Datensatzübertragung, indem Sie den SFB 53 mit *REQ* = 1 aufrufen. Über die Ausgangsparameter *BUSY* und die Bytes 2 und 3 des Ausgangsparameters *STATUS* wird der Zustand des Auftrages angezeigt. Dabei ent-

sprechen die Bytes 2 und 3 von *STATUS* dem Ausgangsparameter *RET_VAL* der asynchron arbeitenden SFCs (siehe auch Bedeutung von *REQ*, *RET_VAL* und *BUSY* bei asynchron arbeitenden SFCs). Beachten Sie, dass Sie dem Aktualparameter von *RECORD* bei allen zu ein und demselben Auftrag gehörenden Aufrufen des SFB 53 denselben Wert zuweisen. Dasselbe gilt für die Aktualparameter von *LEN*. Die Datensatzübertragung ist abgeschlossen, wenn der Ausgangsparameter *BUSY* den Wert FALSE angenommen hat.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L, Konstante	*REQ* = 1: Datensatzübertragung durchführen
ID	INPUT	DWORD	E, A, M, D, L, Konstante	Logische Adresse des Moduls. Bei einem Ausgabemodul muss Bit 15 gesetzt werden (z.B. für Adresse 5: *ID*: = 8005h). Bei einem Mischmodul ist die kleinere der beiden Adressen anzugeben.
INDEX	INPUT	INT	E, A, M, D, L, Konstante	Datensatznummer
LEN	INPUT	INT	E, A, M, D, L, Konstante	Maximale Länge des zu übertragenden Datensatzes in Bytes
DONE	OUTPUT	BOOL	E, A, M, D, L	Datensatz wurde übertragen
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: Der Schreibvorgang ist noch nicht beendet.
ERROR	OUTPUT	BOOL	E, A, M, D, L	*ERROR* = 1: Beim Schreibvorgang trat ein Fehler auf.
STATUS	OUTPUT	DWORD	E, A, M, D, L	Aufrufkennung (Bytes 2 und 3) bzw. Fehlercode
RECORD	IN_OUT	ANY	E, A, M, D, L	Datensatz

Fehlerinformationen

Kap. 16.2.22 "SFB 54 - RALRM - Alarm von einer Peripheriebaugruppe empfangen" Seite 993

16.2.22 SFB 54 - RALRM - Alarm von einer Peripheriebaugruppe empfangen

16.2.22.1 Parameter

Die Schnittstelle des SFB 54 ist identisch mit der des in der Norm "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3" definierten FB RALRM.

Beschreibung

Der SFB 54 RALRM empfängt einen Alarm samt der zugehörigen Information von einem Peripheriemodul bzw. einer Komponente des entsprechenden Bus-Slave und stellt diese Information an seinen Ausgangsparametern zur Verfügung. Die Information in den Ausgangsparametern enthält sowohl die Startinformation des aufgerufenen OB als auch Informationen aus der Alarmquelle. Rufen Sie den SFB 54 nur innerhalb des Alarm-OBs auf, den das Betriebssystem der CPU aufgrund des zu untersuchenden Alarms aus der Peripherie gestartet hat.

Wenn Sie den SFB 54 RALRM in einem OB aufrufen, dessen Starterereignis kein Alarm aus der Peripherie ist, stellt Ihnen der SFB an seinen Ausgängen entsprechend weniger Informationen zur Verfügung.

Achten Sie darauf, dass Sie beim Aufruf des SFB 54 in verschiedenen OBs unterschiedliche Instanz-DBs verwenden. Falls Sie die aus einem SBF 54 Aufruf resultierenden Daten außerhalb des zugehörigen Alarm-OBs auswerten, sollten Sie pro OB-Startereignis einen eigenen Instanz-DB benutzen.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
MODE	INPUT	INT	E, A, M, D, L, Konstante	Betriebsart
F_ID	INPUT	DWORD	E, A, M, D, L, Konstante	Logische Anfangsadresse der Komponente (Modul), von der Alarme empfangen werden sollen
MLEN	INPUT	INT	E, A, M, D, L, Konstante	Maximale Länge der zu empfangenen Alarminformation in Bytes
NEW	OUTPUT	BOOL	E, A, M, D, L	TRUE: Ein neuer Alarm wurde empfangen FALSE: Kein neuer Alarm wurde empfangen
STATUS	OUTPUT	DWORD	E, A, M, D, L	C0000000h: Kein Fehler C080C300h: Betriebsmittel sind momentan belegt C0809000h: Ungültige logische Anfangsadresse **Nur für PROFINET-IO:** C080A000h: Lesefehler C080B700h: Invalider Bereich
ID	OUTPUT	DWORD	E, A, M, D, L	Logische Anfangsadresse der Komponente (Modul), von der ein Alarm empfangen wurde Bit 15 enthält die E/A-Kennung: 0: bei einer Eingangsadresse 1: bei einer Ausgangsadresse
LEN	OUTPUT	INT	E, A, M, D, L	Länge der empfangenen Alarminformation
TINFO	IN_OUT	ANY	E, A, M, D, L	(Task Information) Zielbereich für OB-Startinformation und Verwaltungsinformation
AINFO	IN_OUT	ANY	E, A, M, D, L	(Alarm Information) Zielbereich für Kopfinformation und Alarmzusatzinformation. Für *AINFO* sollten Sie eine Länge von mindestens *MLEN* Bytes vorsehen.

MODE

Den SFB 54 können Sie in drei verschiedenen Betriebsarten (*MODE*) aufrufen:

- 0: zeigt die alarmauslösende Komponente im Ausgangsparameter *ID* an und beschreibt den Ausgangsparameter *NEW* mit TRUE.
- 1: beschreibt sämtliche Ausgangsparameter unabhängig von der alarmauslösenden Komponente.
- 2: prüft, ob die im Eingangsparameter *F_ID* angegebene Komponente den Alarm ausgelöst hat.
  - falls nein, erhält *NEW* den Wert FALSE
  - falls ja, erhält *NEW* den Wert TRUE, und alle anderen Ausgangsparameter werden beschrieben.

Falls Sie den Zielbereich TINFO bzw. AINFO zu kurz wählen, kann der SFC 54 nicht die gesamte Information eintragen.

TINFO

TINFO PROFIBUS: Datenstruktur des Zielbereichs (Task-Information)						
Byte	**Datentyp**	**Beschreibung**				
0 ... 19		Startinformation des OB, in dem der SFC 54 aktuell aufgerufen wurde: Byte 0 ... 11: strukturiert wie der Parameter TOP_SI der SFC 6 RD_SINFO Byte 12 ... 19: Datum und Uhrzeit, zu denen der OB angefordert wurde				
20 ... 27		Verwaltungsinformation:				
20	Byte	zentral: 0 dezentral: DP-Mastersystem-ID (mögliche Werte: 1 ... 255)				
21	Byte	zentral: Baugruppenträgernummer (mögliche Werte: 0 ... 31) dezentral: Nummer der DP-Station (mögliche Werte: 0 ... 127)				
22	Byte	zentral: 0				
		dezentral:	Bit 3 ... 0	Slavetyp	0000:	DP
					0001:	DPS7
					0010:	DPS7 V1
					0011:	DP-V1
					ab 0100:	reserviert
			Bit 7 ... 4	Profiltyp	0000:	DP
					ab 0001:	reserviert
23	Byte	zentral: 0				
		dezentral:	Bit 3 ... 0	Alarminfotyp	0000:	Transparent (Alarm kommt von einem projektierten dezentralen Modul)
					0001:	Stellvertreter (Alarm eines nicht-DP-V1 Slaves oder eines nicht projektierten Steckplatzes)
					0010:	Erzeugter Alarm (in der CPU)
					ab 0011:	reserviert
			Bit 7 ... 4	Strukturversion	0000:	Initial
					ab 0001:	reserviert
24	Byte	zentral: 0				
		dezentral: Flags der DP-Master-Anschaltung				
		Bit 0 = 0:	Alarm von einer integrierten DP-Anschaltung			
		Bit 0 = 1:	Alarm von einer externen DP-Anschaltung			
		Bit 7 ... 1:	reserviert			
25	Byte	zentral: 0				
		dezentral: Flags der DP-Slave-Anschaltung				

TINFO PROFIBUS: Datenstruktur des Zielbereichs (Task-Information)			
Byte	**Datentyp**	**Beschreibung**	
		Bit 0:	EXT_DIAG_Bit aus dem Diagnosetelegramm bzw. 0, falls dieses Bit beim Alarm nicht vorliegt
		Bit 7 ... 1:	reserviert
26, 27	WORD	zentral: 0	
		dezentral: PROFIBUS Identnummer	

TINFO PROFINET-IO: Datenstruktur des Zielbereichs (Task-Information)			
Byte	**Bezeichner**	**Datentyp**	**Beschreibung**
0 ... 19	OB Startinfo	BYTE	Startinformation des OB, in dem der SFB 54 aktuell aufgerufen wurde.
20 ... 21	Addressinfo	WORD	Bit 0 ... 10: Stationsnummer (0-2047) Bit 11 ... 14: die letzten beiden Stellen der PROFINET-IO-System-ID (0-15), um die vollständige PROFINET-IO-System-ID zu erhalten, müssen Sie 100 (dezimal) dazu addieren. Bit 15: 1
22	Slavetyp	BYTE	Bit 0 ... 3: 1000: Festwert für PROFINET-IO Bit 4 ... 7: reserviert
23	Alarminfo	BYTE	Bit 0 ... 3: 0000: Transparent, Alarm kommt von einer projektierten dezentralen Baugruppe Bit 4 ... 7: reserviert
24	PROFINET-IO-Controller-Anschaltung	BYTE	Flags der PROFINET-IO-Controller-Anschaltung Bit 0: 0: Alarm von einer integrierten Anschaltung 1: Alarm von einer externen Anschaltung Bit 1 ... 7: reserviert
25	Flags der PROFINET-IO-Controller-Anschaltung	BYTE	Bit 0: AR Diagnosisstate bzw. 0, falls keine Information beim Alarm vorliegt. Bit 0: 1: IO-Device gestört Bit 1 ... 7: reserviert
26 ... 27	PROFINET-IO-Device Ident-nummer	WORD	Eindeutige Identifikation des PROFINET-IO-Devices
28 ... 29		WORD	Herstellerkennung
30 ... 31	ID	WORD	Identnummer der Instanz

TINFO EtherCAT: Datenstruktur des Zielbereichs (Task-Information)			
Byte	**Bezeichner**	**Datentyp**	**Beschreibung**
0 ... 19	OB Startinfo	BYTE	Startinformation des OB, in dem der SFB 54 aktuell aufgerufen wurde.
20 ... 21	Addressinfo	WORD	Bit 0 ... 10: Master/Slave Bit 11 ... 14: System-ID des EtherCAT Netzwerks - 100 Bit 15: 1: Kennbit für EtherCAT (PROFINET "look and feel")
22	Slavetyp	BYTE	Bit 0 ... 3: 1000: 0b1111 EtherCAT[1] Bit 4 ... 7: reserviert
23	Alarminfo	BYTE	Bit 0 ... 3: 0000: Transparent, Alarm kommt von einer projektierten dezentralen Baugruppe Bit 4 ... 7: reserviert
24	EC Flags I	BYTE	Flags der EtherCAT IO-Controller-Anschaltung Bit 0: 0: Alarm von einer integrierten Anschaltung 1: Alarm von einer externen Anschaltung Bit 1 ... 7: reserviert
25 ... 31			reserviert

1) Ab 0b1001 PROFINET-IO

AINFO

AINFO PROFIBUS: Datenstruktur des Zielbereiches (Alarm-Information)				
Byte	**Datentyp**	**Beschreibung**		
0 ... 3		Kopfinformation		
0	Byte	Länge der empfangenen Alarminformation in Bytes		
		zentral: 4 ... 224		
		dezentral: 4 ... 63		
1	Byte	zentral: reserviert		
		dezentral:	Kennung für den Alarmtyp	
			1:	Diagnosealarm
			2:	Prozessalarm
			3:	Ziehen-Alarm
			4:	Stecken-Alarm
			5:	Status-Alarm
			6:	Update-Alarm
			31:	Ausfall eines Erweiterungsgeräts, eines DP-Mastersystems oder einer DP-Station
			32 ... 126	herstellerspezifischer Alarm
2	**Byte**	**Steckplatznummer der alarmauslösenden Komponente**		
3	Byte	zentral: reserviert		
		dezentral:	Specifier	
			Bit 1, 0:	
			00	keine weitere Information
			01	kommendes Ereignis, Steckplatz gestört
			10	gehendes Ereignis, Steckplatz nicht mehr gestört
			11	gehendes Ereignis, Steckplatz weiterhin gestört
			Bit 2:	Add_Ack
			Bit 7 ... 3	Sequenznummer
4 ... 223		Alarmzusatzinformation: Modulspezifische Daten zum jeweiligen Alarm:		
		zentral:	ARRAY[0] ... ARRAY[220]	
		dezentral:	ARRAY[0] ... ARRAY[59]	

AINFO PROFINET-IO: Datenstruktur des Zielbereiches (Alarm-Information)			
Byte	**Bezeichner**	**Datentyp**	**Beschreibung**
0 ... 1	Bausteintyp	WORD	Bit 0 ... 7: Bausteintyp Bit 8 ... 15: reserviert
2 ... 3	Bausteinlänge	WORD	Länge der empfangenen Alarminformation in Bytes MIN: 0 MAX: 1536 (1,5kByte)
4 ... 5	Version	WORD	Bits 0 ... 7: low byte Bits 8 ... 15: high byte

AINFO PROFINET-IO: Datenstruktur des Zielbereiches (Alarm-Information)			
Byte	**Bezeichner**	**Datentyp**	**Beschreibung**
6 ... 7	Alarmtyp	WORD	Kennung für den Alarmtyp: 1: Diagnosealarm (kommend) 2: Prozessalarm 3: Ziehen-Alarm 4: Stecken-Alarm 5: Status-Alarm 6: Update-Alarm 7: Redundanz-Alarm 8: Vom Supervisor gesteuert (Controlled_by_Supervisor) 9: Vom Supervisor freigegeben (Released_by_Supervisor) 10: Nicht die projektierte Baugruppe gesteckt 11: Wiederkehr des Submoduls 12: Diagnosealarm (gehend) 13: Querverkehr-Verbindungsmeldung 14: Nachbarschaftsänderungsmeldung 15: Taktsynchronisationsmeldung (busseitig) 16: Taktsynchronisationsmeldung (geräteseitig) 17: Netzwerkkomponentenmeldung 18: Uhrzeitsynchronisationsmeldung (busseitig) 19 ... 29: reserviert 30: Hochladenzubringen-Alarm 31: reserviert 32 ... 127: Herstellerspezifischer Alarm 128 ... 65535: Reserviert außer die folgenden VIPA-spezifischen Alarmtypen: 38CAh: Wiederkehr des Controllers 48CAh: Konfiguration des Controllers akzeptiert 39CAh: Ausfall des Controllers 49CAh: Ausfall des Controllers wegen Watchdog 38CBh: Wiederkehr des Geräts 38CCh: Störung am Wiederkehr des Geräts 38CDh: Ein anderes Gerät wird erkannt während Wiederkehr des Geräts 38CEh: Parameterfehler während Wiederkehr des Geräts 39CBh: Ausfall des Geräts
8 ... 11	API	DWORD	API (Application Process Identifier)
12 ... 13	Steckplatz-nummer	WORD	Steckplatznummer der Alarm auslösenden Komponente (Wertebereich 0 ... 65535)

AINFO PROFINET-IO: Datenstruktur des Zielbereiches (Alarm-Information)			
Byte	**Bezeichner**	**Datentyp**	**Beschreibung**
14 ... 15	Submodulsteck-platznummer	WORD	Submodul-Steckplatznummer der Alarm auslösenden Komponente (Wertebereich 0 ... 65535)
16 ... 19	Modul-Identifika-tion	DWORD	Eindeutige Information der Alarmquelle: Modul-ID
20 ... 23	Submodul-Identi-fikation	DWORD	Eindeutige Information der Alarmquelle: Submodul-ID
24 ... 25	Alarm Specifier	WORD	Bit 0 ... 10: Sequenznummer (Wertebereich 0 bis 2047) Bit 11: Kanaldiagnose 0: keine Kanaldiagnose vorhanden 1: Kanaldiagnose vorhanden Bit 12: Status der herstellerspezifischen Diagnose 0: keine herstellerspezifische Statusinformation vorhanden 1: Herstellerspezifische Statusinformation vorhanden Bit 13: Status der Diagnose beim Submodul: 0: keine Statusinformation vorhanden, alle Fehler wurden beseitigt 1: mindestens eine Kanaldiagnose und/oder Statusinformation vor-handen Bit 14: reserviert Bit 15: Application Relationship Diagnosis State 0: keine der innerhalb dieser AR projektierten Baugruppen meldet eine Diagnose 1: mindestens eine innerhalb dieser AR projektierten Baugruppen meldet eine Diagnose
26 ... 1535	Alarmzusatz-information	WORD	**Hinweis:** Die Alarmzusatzinformation kann auch entfallen.

AINFO EtherCAT: Datenstruktur des Zielbereiches (Alarm-Information)			
Byte	**Bezeichner**	**Datentyp**	**Beschreibung**
0, 1	Länge	WORD	Länge der empfangenen Alarminformation in Bytes: MIN: 0 MAX: 1535 (1,5kByte)

2, 3	AlarmType	WORD	Kennung für den Alarmtyp:
			0001h: DIAGNOSE_ALARM_KOMMEND
			0002h: PROZESS_ALARM
			000Ch: DIAGNOSE_ALARM_GEHEND
			0020h: MANUFACTOR_SPECIFIC_ALARM_MIN
			// VIPA specific:
			39CAh: CONTROLLER_AUSFALL
			49CAh: CONTROLLER_AUSFALL_WATCHDOG
			// EtherCAT specific:
			8001h: BUS_STATE_CHANGED
			8002h: SLAVE_STATE_CHANGED
			8003h: TOPOLOGY_OK
			8004h: TOPOLOGY_MISMATCH

4, 5	RackSlot	WORD	Steckplatznummer des EtherCAT-Masters
6, 7	Master/Slave ID	WORD	EtherCAT Master-/Slave Adresse
8, 9	AlarmSpecifier	WORD	Wert ist abhängig von AlarmType: AlarmType: Wert BUS_STATE_CHANGED: neuer Busstatus [1] DIAGNOSE_ALARM_GEHEND: reserved DIANGOSE_ALARM_KOMMEND: reserved PROZESS_ALARM: reserved MANUFACTOR_SPECIFIC_ALARM_MIN: reserved SLAVE_STATE_CHANGED: neuer Busstatus CONTROLLER_AUSFALL: reserved CONTROLLER_AUSFALL_WATCHDOG: reserved TOPOLOGY_OK: reserved TOPOLOGY_MISMATCH: reserved

AINFO EtherCAT: Datenstruktur des Zielbereiches (Alarm-Information)			
Byte	**Bezeichner**	**Datentyp**	**Beschreibung**
10 ... n	Data	BYTE	Inhalt ist abhängig vom AlarmType: AlarmType: Inhalt BUS_STATE_CHANGED: Datenstruktur [2] DIAGNOSE_ALARM_GEHEND: CoE-Emergency [3] DIAGNOSE_ALARM_KOMMEND: CoE-Emergency PROZESS_ALARM: CoE-Emergency MANUFACTOR_SPECIFIC_ALARM_MIN: CoE-Emergency SLAVE_STATE_CHANGED: AL Status Code [4] CONTROLLER_AUSFALL: Ausfall-Code [5] CONTROLLER_AUSFALL_WATCHDOG: reserved TOPOLOGY_OK: reserved TOPOLOGY_MISMATCH: reserved

1) EtherCAT-States *1005*

2) Datenstruktur BUS_STATE_CHANGED *1006*

3) CoE-Emergency *1006*

4) AL Status Code *1006*

5) Ausfall-Code *1006*

16.2.22.2 EtherCAT-States

Die Buszustände bei EtherCAT sind wie folgt codiert

Name	Code	Beschreibung
Undefined/Unknown	0x00	Diesen Status hat ein Slave, bevor er seine Initialisierungsroutinen durchführen konnte. Für den VIPA EtherCAT-Master besitzt ein Slave ebenfalls den Zustand Undefined, wenn ein Slave-Ausfall vorliegt (Kabel ziehen).
Init	0x01	Es findet keine direkte Kommunikation zwischen Master und Slaves statt. Der Master initialisiert in diesem Zustand die Konfigurations-Register des ESC. Es findet kein Prozessdaten-Austausch oder Mailbox-Kommunikation statt.
PreOp	0x02	In diesem Zustand ist Mailbox-Kommunikation möglich, es findet jedoch kein Prozessdaten-Austausch statt.
BootStrap	0x03	Spezieller Zustand des EtherCAT-Slaves in dem nur Mailbox-Kommunikation stattfindet. Um ein Firmware-Update des Slave durchzuführen, muss der Slave in diesen Zustand versetzt werden.
SafeOp	0x04	Im Zustand SafeOp ist Mailbox-Kommunikation möglich und es können Prozesseingangsdaten ausgetauscht werden. Es findet jedoch kein Austausch von Prozessausgangsdaten statt.
Op	0x08	In diesem Zustand können Mailbox-Daten und Prozessdaten ausgetauscht werden.

16.2.22.3 Ursache für Controller-Ausfall

Bei einem Controller-Ausfall gibt der Alarm-Specifier Auskunft über die Ursache des Ausfalls

Name	Code	Beschreibung
REASON_UNKNOWN	0	Ursache unbekannt
ALARM_OVERFLOW	1	Alarm-Überlast
MESSAGE_QUEUE_OVERFLOW	2	Überlast an EtherCAT-Events
CYCLIC_FRAMES_NOT_IN_BUSCYCLE	3	EtherCAT-Empfangs-Telegramm wurde nicht innerhalb der Buszykluszeit empfangen
APPL_BUSCYCLE_ERROR	4	Buszyklus-Zeit konnte nicht eingehalten werden (z.B. aufgrund hoher Systemlast)

16.2.22.4 CoE-Emergency

Eine CoE-Emergency ist eine besondere Form der Mailbox-Kommunikation bei welcher der EtherCAT-Slave dem EtherCAT-Master signalisieren kann, dass ein Fehler aufgetreten ist. Sie hat folgenden Aufbau:

Name	Datentyp	Beschreibung
Error Code	WORD	Error Code
Error Register	BYTE	EtherCAT-State als der Fehler des Slaves aufgetreten ist
Data	BYTE[5]	Manufacturer Specific Error Field (MEF), enthält weitere Diagnosedaten

16.2.22.5 AL Status Code

AL ist die Abkürzung für Application Layer. Der AL Status Code ist ein Error Code aus der Slave Applikation.

16.2.22.6 Datenstruktur BUS_STATE_CHANGED

Header

NrOfSlavesTotal - Anzahl aller Slaves, die nicht im Master-State sind.

NrOfSlavesUndefined - Anzahl der Slaves im *Undefined*-Zustand.

NrOfSlavesInit - Anzahl der Slaves im *Init*-Zustand.

NrOfSlavesPreop - Anzahl der Slaves im *PreOp*-Zustand.

NrOfSlavesBoostrap - Anzahl der Slaves im *Bootstrap*-Zustand.

NrOfSlavesSafeop - Anzahl der Slaves im *SafeOp*-Zustand.

NrOfSlavesOp - Anzahl der Slaves im *Op*-Zustand.

DeviceId

DeviceId[0] ... - EtherCAT-Adresse des Slave, wie in der Projektierung vergeben.

DeviceId[NrOfSlaves-Total-1] - EtherCAT-Adresse des Slave, wie in der Projektierung vergeben.

TINFO und AINFO

Abhängig von dem jeweiligen OB, in dem der SFB 54 aufgerufen wird, werden die Zielbereiche TINFO und AINFO nur teilweise beschrieben. Welche Information jeweils eingetragen wird, entnehmen Sie der folgenden Tabelle.

Zielbereich					
Alarmtyp	**OB**	**TINFO** **OB-Status-Information**	**TINFO** **Verwaltungs-Information**	**AINFO** **Kopf-Information**	**AINFO** **Alarmzusatz-Information**
Prozessalarm	4x	ja	ja	ja	zentral: nein dezentral: wie vom DP-Slave geliefert
Statusalarm	55	ja	ja	ja	ja
Update-Alarm	56	ja	ja	ja	ja
Hersteller-spezifischer Alarm	57	ja	ja	ja	ja
Peripherie-Redundanz-fehler	70	ja	ja	nein	nein
Diagnosealarm	82	ja	ja	ja	zentral: Datensatz 1 dezentral: wie vom DP-Slave geliefert
Ziehen-Stecken-Alarm	83	ja	ja	ja	zentral: nein dezentral: wie vom DP-Slave geliefert
Baugruppen-trägerausfall/Stationsausfall	86	ja	ja	nein	nein
...	alle anderen OBs	ja	nein	nein	nein

Fehlerinformationen

Der Ausgangsparameter *STATUS* enthält Fehlerinformationen. Wird er als ARRAY[1...4] OF BYTE interpretiert, hat die Fehlerinformation folgende Struktur:

Feldelement	Name	Beschreibung
STATUS[1]	Function_Num	00h: falls kein Fehler Funktionskennung aus DP-V1-CPU: im Fehlerfall ist 80h aufgeodert. Falls kein DP-V1-Protokollelement benutzt wird: C0h
STATUS[2]	Error_Decode	Ort der Fehlerkennung
STATUS[3]	Error_1	Fehlerkennung
STATUS[4]	Error_2	herstellerspezifische Erweiterung der Fehlerkennung *STATUS*[4] wird bei DP-V1-Fehlern vom DP-Master an die CPU und den SFB durchgereicht. Ohne DP-V1-Fehler wird der Wert auf "0" gesetzt mit folgenden Ausnahmen beim SFB 52: ■ *STATUS*[4] enthält die Länge des Zielbereichs aus *RECORD*, falls *MLEN* > Länge des Zielbereichs aus RECORD ■ *STATUS*[4]=*MLEN*, falls die tatsächliche Datensatzlänge< *MLEN* < Länge des Zielbereichs aus *RECORD*

STATUS[2] (Ort der Fehlerkennung) kann folgende Werte annehmen:		
Error_Decode	**Quelle**	**Beschreibung**
00 ... 7Fh	CPU	kein Fehler oder keine Warnung
80h	DP-V1	Fehler nach IEC 61158-6
81h ... 8Fh	CPU	8xh zeigt einen Fehler im x-ten Aufrufparameter des SFB an
FEh, FFh	DP Profile	profilspezifischer Fehler

***STATUS*[3] (Fehlerkennung) kann folgende Werte annehmen:**			
Error_Decode	**Error_Code_1**	**Erläuterung laut DP-V1**	**Beschreibung**
00h	00h		kein Fehler, keine Warnung
70h	00h	reserved, reject	Erstaufruf; keine Datensatzübertragung aktiv
	01h	reserved, reject	Erstaufruf; Datensatzübertragung angestoßen
	02h	reserved, reject	Zwischenaufruf; Datensatzübertragung ist bereits aktiv
80h	90h	reserved, pass	logische Anfangsadresse ungültig
	92h	reserved, pass	unzulässiger Typ bei ANY-Pointer
	93h	reserved, pass	Die mittels *ID* bzw. *F_ID* adressierte DP-Komponente ist nicht konfiguriert.
	A0h	read error	negative Quittung beim Lesen vom Modul

***STATUS*[3] (Fehlerkennung) kann folgende Werte annehmen:**			
Error_Decode	**Error_Code_1**	**Erläuterung laut DP-V1**	**Beschreibung**
	A1h	write error	negative Quittung beim Schreiben zum Modul
	A2h	module failure	DP-Protokollfehler bei Layer 2
	A3h	reserved, pass	DP-Protokollfehler bei Direct-Data-Link-Mapper oder User-Interface/User
	A4h	reserved, pass	Kommunikation am Bus gestört
	A5h	reserved, pass	-
	A7h	reserved, pass	DP-Slave oder Modul ist beschäftigt (temporärer Fehler)
	A8h	version conflict	DP-Slave oder Modul meldet nicht passende Versionen
	A9h	feature not supported	Funktion wird vom DP-Slave oder Modul nicht unterstützt
	AA ... AFh	user specific	DP-Slave oder Modul meldet einen herstellerspezifischen Fehler seiner Anwendung. Bitte sehen Sie in der Dokumentation des Herstellers des DP-Slaves bzw. des Moduls nach.
	B0h	invalid index	Modul kennt den Datensatz nicht Datensatznummer ≥256 ist unzulässig
	B1h	write length error	Die Längenangabe im Parameter *RECORD* ist falsch; bei SFB 54: Längenfehler in *AINFO*
	B2h	invalid slot	Der projektierte Steckplatz ist nicht belegt.
	B3h	type conflict	Ist-Modultyp ungleich Soll-Modultyp
	B4h	invalid area	DP-Slave oder Modul meldet einen Zugriff auf einen unzulässigen Bereich
	B5h	state conflict	DP-Slave oder Modul ist nicht bereit
	B6h	access denied	DP-Slave oder Modul verweigert den Zugriff
	B7h	invalid range	DP-Slave oder Modul meldet einen unzulässigen Bereich eines Parameters oder eines Werts
	B8h	invalid parameter	DP-Slave oder Modul meldet einen unzulässigen Parameter
	B9h	invalid type	DP-Slave oder Modul meldet einen unzulässigen Typ
	BAh ... BFh	user specific	DP-Slave oder Modul meldet einen herstellerspezifischen Fehler beim Zugriff. Bitte sehen Sie in der Dokumentation des Herstellers des DP-Slaves oder Moduls nach.

***STATUS*[3] (Fehlerkennung) kann folgende Werte annehmen:**			
Error_Decode	**Error_Code_1**	**Erläuterung laut DP-V1**	**Beschreibung**
	C0h	read constrain conflict	Das Modul führt den Datensatz, aber es sind noch keine Lesedaten da.
	C1h	write constrain conflict	Die Daten des auf dem Modul vorangegangenen Schreibauftrags für denselben Datensatz sind von dem Modul noch nicht verarbeitet.
	C2h	resource busy	Das Modul bearbeitet momentan das mögliche Maximum an Aufträgen für eine CPU.
	C3h	resource unavailable	Die benötigten Betriebsmittel sind momentan belegt.
	C4h		Interner temporärer Fehler. Auftrag konnte nicht ausgeführt werden. Wiederholen Sie den Auftrag. Bei häufigem Auftreten dieses Fehlers überprüfen Sie bitte Ihren Aufbau auf elektrische Störquellen.
	C5h		DP-Slave oder Modul nicht verfügbar
	C6h		Datensatzübertragung wurde abgebrochen wegen Prioritätsklassenabbruchs.
	C7h		Auftragsabbruch wegen Neustart des DP-Masters
	C8h ... CFh		DP-Slave oder Modul meldet einen herstellerspezifischen Fehler zu seinen Ressourcen. Bitte sehen Sie in der Dokumentation des Herstellers des DP-Slaves oder Moduls nach.
	Dxh	user specific	DP-Slave spezifisch
81h	00h ... FFh		Fehler im 1. Aufrufparameter (bei SFB 54: *MODE*)
	00h		Unzulässige Betriebsart
82h	00h ... FFh		Fehler im 2. Aufrufparameter.
...	...		...
88h	00h ... FFh		Fehler im 8. Aufrufparameter (bei SFB 54: *TINFO*)
	01h		Syntaxkennung falsch
	23h		Überschreitung des Mengengerüsts oder Zielbereich zu klein
	24h		Bereichskennung falsch
	32h		DB/DI-Nr. außerhalb des Anwenderbereichs
	3Ah		DB/DI-Nr. ist NULL bei Bereichskennung DB/DI oder angegebener DB/DI nicht vorhanden.

***STATUS*[3] (Fehlerkennung) kann folgende Werte annehmen:**			
Error_Decode	**Error_Code_1**	**Erläuterung laut DP-V1**	**Beschreibung**
89h	00h ... FFh		Fehler im 9. Aufrufparameter (bei SFB 54: *AINFO*)
	01h		Syntaxkennung falsch
	23h		Überschreitung des Mengengerüsts oder Zielbereich zu klein
	24h		Bereichskennung falsch
	32h		DB/DI-Nr. außerhalb des Anwenderbereichs
	3Ah		DB/DI-Nr. ist Null bei Bereichskennung DB/DI oder angegebener DB/DI nicht vorhanden
8Ah	00h ... FFh		Fehler im 10. Aufrufparameter
...	...		...
8Fh	00h ... FFh		Fehler im 15. Aufrufparameter
FEh, FFh			Profilspezifischer Fehler

17 Standard-Bausteine - *"Standard"*

17.1 Konvertierung - *"Converting"*

17.1.1 FB 80 - LEAD_LAG - Lead/Lag Algorithmus

Beschreibung

Mit dem Funktionsbaustein Lead/Lag Algorithmus LEAD_LAG können mit einer analogen Variable Signale bearbeitet werden. Ein Ausgang *OUT* wird aufgrund eines Eingangs *IN* und der angegebenen Werte für Verstärken *GAIN*, Voreilen *LD_TIME* und Nacheilen *LG_TIME* berechnet. Der Wert für die Verstärkung muss größer sein als Null. Der LEAD_LAG Algorithmus arbeitet mit der folgenden Gleichung:

$$und\ OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] PREV_IN$$

Typischerweise wird der FB LEAD_LAG zusammen mit Schleifen zur Kompensation bei der dynamischen Vorwärtsregelung eingesetzt. LEAD_LAG besteht aus zwei Teilen. Die Phase LEAD verschiebt die Phase des Ausgangs des Funktionsbausteins, so dass der Ausgang dem Eingang vor eilt. Die Phase LAG hingegen verschiebt den Ausgang, so dass der Ausgang dem Eingang nacheilt. Da die Operation LAG mit einer Integration gleichzusetzen ist, kann sie als Entstörelement oder als Tiefpassfilter eingesetzt werden. Die Operation LEAD entspricht einer Differentiation und entspricht deshalb einem Hochpassfilter. Beide Operationen zusammen (LEAD_LAG) führen dazu, dass die Ausgangsphase dem Eingang bei niederen Frequenzen nacheilt und ihm bei hohen Frequenzen vor eilt. Deshalb kann LEAD_LAG als Bandpassfilter eingesetzt werden.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	Eingang	BOOL	E, A, M, D, L	Ein Signalzustand von 1 am Freigabeeingang aktiviert die Box.
ENO	Ausgang	BOOL	E, A, M, D, L	Der Freigabeausgang hat den Signalzustand 1, wenn der Funktionsbaustein fehlerfrei ausgeführt wird.
IN	Eingang	REAL	E, A, M, D, L, P, Konstante	Eingabewert des aktuellen Beispielzeitraums, der bearbeitet werden soll
SAMPLE_T	Eingang	INT	E, A, M, D, L, P, Konstante	Beispielzeit
OUT	Ausgang	REAL	E, A, M, D, L, P, Konstante	Ergebnis der Operation LEAD_LAG
ERR_CODE	Ausgang	WORD	E, A, M, D, L, P	Gibt den Wert W#16#0000 aus, wenn die Operation fehlerfrei ausgeführt wird. Wird ein anderer Wert ausgegeben, entnehmen Sie der Fehlerinformation nähere Angaben hierzu.
LD_TIME	Statisch	REAL	E, A, M, D, L, P, Konstante	Voreilzeit in Minuten
LG_TIME	Statisch	REAL	E, A, M, D, L, P, Konstante	Nacheilzeit in Minuten
GAIN	Statisch	REAL	E, A, M, D, L, P, Konstante	Verstärkung in % / % (Verhältnis von Ausgabeveränderung zu Eingabeveränderung als stetiger Zustand)

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
PREV_IN	Statisch	REAL	E, A, M, D, L, P, Konstante	Vorheriger Eingang
PREV_OUT	Statisch	REAL	E, A, M, D, L, P, Konstante	Vorheriger Ausgang

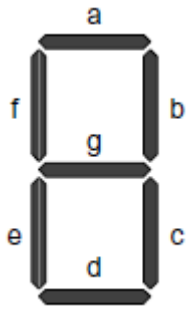
Fehlerinformation

Ist *GAIN* kleiner als oder gleich 0, dann wird der Funktionsbaustein nicht ausgeführt. Der Signalzustand von *ENO* wird auf 0 und *ERR_CODE* wird auf W#16#0009 gesetzt.

17.1.2 FC 93 - SEG - Bitmuster für 7-Segment-Anzeige erzeugen

Beschreibung

Die Funktion Bitmuster für 7-Segment-Anzeige erzeugen SEG wandelt jede der vier Hexadezimalziffern des angegebenen Quellworts *IN* in vier equivalente Codes für eine 7-Segment-Anzeige um und schreibt diese in das Doppelwort des Ausgangs *OUT*. Das Bild unten zeigt die Beziehung zwischen den Hexadezimalziffern des Eingangs und den Bitmustern des Ausgangs.



Parameter

Ziffer	– g f e d c b a	Anzeige
0 0 0 0	0 0 1 1 1 1 1 1	0
0 0 0 1	0 0 0 0 0 1 1 0	1
0 0 1 0	0 1 0 1 1 0 1 1	2
0 0 1 1	0 1 0 0 1 1 1 1	3
0 1 0 0	0 1 1 0 0 1 1 0	4
0 1 0 1	0 1 1 0 1 1 0 1	5
0 1 1 0	0 1 1 1 1 1 0 1	6
0 1 1 1	0 0 0 0 0 1 1 1	7
1 0 0 0	0 1 1 1 1 1 1 1	8
1 0 0 1	0 1 1 0 0 1 1 1	9
1 0 1 0	0 1 1 1 0 1 1 1	A
1 0 1 1	0 1 1 1 1 1 0 0	b
1 1 0 0	0 0 1 1 1 0 0 1	C
1 1 0 1	0 1 0 1 1 1 1 0	d
1 1 1 0	0 1 1 1 1 0 0 1	E
1 1 1 1	0 1 1 1 0 0 0 1	F

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	Eingang	BOOL	E, A, M, D, L	Ein Signalzustand von 1 am Freigabeeingang aktiviert die Box.
ENO	Ausgang	BOOL	E, A, M, D, L	Der Freigabeausgang hat den Signalzustand 1, wenn die Funktion fehlerfrei ausgeführt wird.
IN	Eingang	WORD	E, M, D, P, oder Konstante	Quellwort mit vier Hexadezimalziffern.
OUT	Ausgang	DWORD	A, M, D, L, P	Bitmuster des Ziels mit vier Bytes.

Fehlerinformation Diese Funktion erkennt keine Fehlerbedingungen.

17.1.3 FC 94 - ATH - ASCII-Zeichenkette in Hexadezimalzahl wandeln

Beschreibung

Die Funktion ASCII-Zeichenkette in Hexadezimalzahl wandeln ATH wandelt die ASCII-Zeichenkette, auf die der Parameter *IN* zeigt, in Hexadezimalziffern um und speichert diese in der Zieltabelle, auf die der Parameter *OUT* zeigt. Da ein ASCII-Zeichen 8 Bits benötigt und eine Hexadezimalziffer nur 4 Bits, ist das Ausgabewort nur halb so lang wie das Eingabewort. Die ASCII-Zeichen werden nach dem Umwandeln in dem Ausgang in der gleichen Reihenfolge angeordnet, in der sie eingelesen wurden. Handelt es sich um eine ungerade Anzahl an ASCII-Zeichen, wird die Hexadezimalziffer in dem Halbbyte ganz rechts der zuletzt umgewandelten Hexadezimalziffer mit Nullen aufgefüllt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	Eingang	BOOL	E, A, M, D, L	Ein Signalzustand von 1 am Freigabeeingang aktiviert die Box.
ENO	Ausgang	BOOL	E, A, M, D, L	Der Freigabeausgang hat den Signalzustand 1, wenn die Funktion fehlerfrei ausgeführt wird.
IN	Eingang	Pointer*	E, A, M, D, L	Zeigt auf die Anfangsadresse einer ASCII-Zeichenkette.
N	Eingang	INT	E, A, M, L, P	Anzahl der ASCII-Zeichen, die umgewandelt werden sollen.
RET_VAL	Ausgang	WORD	E, A, M, D, L, P	Gibt den Wert W#16#0000 aus, wenn die Operation fehlerfrei ausgeführt wird. Wird ein anderer Wert ausgegeben, entnehmen Sie der Fehlerinformation nähere Angaben hierzu.
OUT	Ausgang	Pointer*	A, M, D, L	Zeigt auf die Anfangsadresse der Tabelle.

*) Pointer im Doppelwortformat bei der bereichsübergreifenden, registerindirekten Adressierung

Fehlerinformation Wird ein ungültiges ASCII-Zeichen erkannt, dann wird es als "0" umgewandelt. Der Signalzustand von *ENO* wird auf 0 gesetzt und *RET_VAL* ist gleich W#16#0007.

17.1.4 FC 95 - HTA - Hexadezimalzahl in ASCII-Zeichenkette wandeln

Beschreibung

Die Funktion Hexadezimalzahl in ASCII-Zeichenkette wandeln HTA wandelt Hexadezimalziffern, auf die der Parameter *IN* zeigt, um und speichert sie in der Zeichenkette, auf die der Parameter *OUT* zeigt. Da ein ASCIIZeichen 8 Bits benötigt und eine Hexadezimalziffer nur 4 Bits, ist das Ausgabewort doppelt so lang wie das Eingabewort. Jedes Halbbyte der Hexadezimalziffer wird in ein Zeichen umgewandelt, und zwar in der gleichen Reihenfolge, in der sie eingelesen werden (das Halbbyte ganz links der Hexadezimalziffer wird zuerst umgewandelt, danach das Halbbyte ganz rechts der gleichen Ziffer).

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	Eingang	BOOL	E, A, M, D, L	Ein Signalzustand von 1 am Freigabeeingang aktiviert die Box.
ENO	Ausgang	BOOL	E, A, M, D, L	Der Freigabeausgang hat den Signalzustand 1, wenn die Funktion fehlerfrei ausgeführt wird.
IN	Eingang	Pointer*	E, A, M, D	Zeigt auf die Anfangsadresse der Hexadezimalziffern.
N	Eingang	WORD	E, A, M, L, P	Anzahl der Hexadezimalbytes, die umgewandelt werden sollen.
OUT	Ausgang	Pointer*	A, M, D, L	Zeigt auf die Anfangsadresse der Zieltabelle.

*) Pointer im Doppelwortformat bei der bereichsübergreifenden, registerindirekten Adressierung

Fehlerinformation

Diese Funktion erkennt keine Fehlerbedingungen.

17.1.5 FC 96 - ENCO - Bitnummer des niederwertigsten gesetzten Bits lesen

Beschreibung

Die Funktion Bitnummer des niederwertigsten gesetzten Bits lesen ENCO wandelt den Inhalt des Parameters *IN* in eine Binärziffer mit 5 Bits um, die der Bitposition des Bits ganz rechts in dem Parameter *IN* entspricht, und gibt das Ergebnis als Funktionswert aus. Hat der Parameter *IN* den Wert 0000 0001 oder 0000 0000, dann wird der Wert 0 ausgegeben.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	Eingang	BOOL	E, A, M, D, L	Ein Signalzustand von 1 am Freigabeeingang aktiviert die Box.
ENO	Ausgang	BOOL	E, A, M, D, L	Der Freigabeausgang hat den Signalzustand 1, wenn die Funktion fehlerfrei ausgeführt wird.
IN	Eingang	DWORD	E, M, D, L, P, Konstante	Wert, der codiert werden soll.
RET_VAL	Ausgang	INT	A, M, D, L, P	Wert, der ausgegeben wird (enthält eine Binärziffer mit 5 Bits).

Fehlerinformation Diese Funktion erkennt keine Fehlerbedingungen.

17.1.6 FC 97 - DECO - Vorgegebenes Bit im Wort setzen

Beschreibung Die Funktion Vorgegebenes Bit im Wort setzen DECO wandelt eine Binärziffer mit 5 Bits (0 bis 31), die von Eingang *IN* angegeben wird, um, indem sie die entsprechende Bitposition in dem Rückgabewert der Funktion setzt. Ist der Parameter *IN* größer als 31, wird eine Modulo-32-Operation ausgeführt, damit eine Binärziffer mit 5 Bits ausgegeben werden kann.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	Eingang	BOOL	E, A, M, D, L	Ein Signalzustand von 1 am Freigabeeingang aktiviert die Box.
ENO	Ausgang	BOOL	E, A, M, D, L	Der Freigabeausgang hat den Signalzustand 1, wenn die Funktion fehlerfrei ausgeführt wird.
IN	Eingang	DWORD	E, M, D, L, P, Konstante	Variable, die decodiert werden soll.
RET_VAL	Ausgang	INT	A, M, D, L, P	Ausgegebener Wert

Fehlerinformation Diese Funktion erkennt keine Fehlerbedingungen.

17.1.7 FC 98 - BCDCPL - Zehnerkomplement erzeugen

Beschreibung Die Funktion Zehnerkomplement erzeugen BCDCPL gibt das Zehnerkomplement einer siebenstelligen BCD-Zahl aus, die von dem Parameter *IN* angegeben wird. Diese Operation rechnet mit folgender mathematischer Formel:

10000000 (als BCD) - 7stelliger BCD-Wert = Zehnerkomplement (als BCD)

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	Eingang	BOOL	E, A, M, D, L	Ein Signalzustand von 1 am Freigabeeingang aktiviert die Box.
ENO	Ausgang	BOOL	E, A, M, D, L	Der Freigabeausgang hat den Signalzustand 1, wenn die Funktion fehlerfrei ausgeführt wird.
IN	Eingang	DWORD	E, M, D, L, P, Konstante	7stellige BCD-Zahl
RET_VAL	Ausgang	DWORD	A, M, D, L, P	Ausgegebener Wert

Fehlerinformation Die Funktion erkennt keine Fehlerbedingungen.

17.1.8 FC 99 - BITSUM - Anzahl der gesetzten Bits zählen

Beschreibung

Die Funktion Anzahl der gesetzten Bits zählen BITSUM zählt die Anzahl der Bits, die am Eingang *IN* auf 1 gesetzt werden, und gibt diesen Wert als Funktionswert aus.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	Eingang	BOOL	E, A, M, D, L	Ein Signalzustand von 1 am Freigabeeingang aktiviert die Box.
ENO	Ausgang	BOOL	E, A, M, D, L	Der Freigabeausgang hat den Signalzustand 1, wenn die Funktion fehlerfrei ausgeführt wird.
IN	Eingang	DWORD	E, M, D, L, P, Konstante	Variable, in der die Bits gezählt werden sollen.
RET_VAL	Ausgang	INT	A, M, D, L, P	Ausgegebener Wert

Fehlerinformation

Diese Funktion erkennt keine Fehlerbedingungen.

17.1.9 FC 105 - SCALE - Werte skalieren

Beschreibung

Die Funktion Werte skalieren SCALE wandelt einen ganzzahligen Wert *IN* in einen Realzahlenwert um, der in physikalischen Einheiten zwischen einem unteren und einem oberen Grenzwert *LO_LIM* und *HI_LIM* skaliert wird. Das Ergebnis wird in den Parameter *OUT* geschrieben. Die Funktion SCALE arbeitet mit der folgenden Gleichung:

$$OUT = [((FLOAT\ (IN)\ -\ K1)\ /\ (K2\ -\ K1))\ \cdot\ (HI_LIM\ -\ LO_LIM)] + LO_LIM$$

Die Konstanten K1 und K2 werden unterschiedlich gesetzt, je nachdem, ob der Eingabewert *BIPOLAR* oder *UNIPOLAR* ist.

- *BIPOLAR*:
  - Es wird angenommen, dass der ganzzahlige Eingabewert zwischen -27648 und 27648 liegt, deshalb sind
    K1 = -27648,0 und K2 = +27648,0.
- *UNIPOLAR*:
  - Es wird angenommen, dass der ganzzahlige Eingabewert zwischen 0 und 27648 liegt, deshalb sind
    K1 = 0,0 und K2 = +27648,0.

Ist der ganzzahlige Eingabewert größer als K2, dann wird der Ausgang *OUT* an *HI_LIM* gebunden und ein Fehler ausgegeben. Ist der ganzzahlige Eingabewert kleiner als K1, dann wird der Ausgang an *LO_LIM* gebunden und ein Fehler ausgegeben. Zum umgekehrten Skalieren wird *LO_LIM* > *HI_LIM* programmiert. Beim umgekehrten Skalieren verringert sich der Ausgabewert, während der Eingabewert zunimmt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Funktion fehlerfrei ausgeführt
IN	INPUT	INT	E, M, D, L, P, Konstante	Eingabewert, der in einen Wert vom Datentyp REAL in physikalischen Einheiten skaliert werden soll.
HI_LIM	INPUT	REAL	E, M, D, L, P, Konstante	Oberer Grenzwert in physikalischen Einheiten
LO_LIM	INPUT	REAL	E, M, D, L, P, Konstante	Unterer Grenzwert in physikalischen Einheiten
BIPOLAR	INPUT	BOOL	E, A, M, D, L	Bei dem Signalzustand 1 handelt es sich um einen bipolaren Eingabewert. Bei dem Signalzustand 0 handelt es sich um einen unipolaren Eingabewert.
OUT	OUTPUT	REAL	E, A, M, D, L, P,	Ergebnis der Skalierung
RET_VAL	INPUT	WORD	E, A, M, D, L, P	Gibt den Wert W#16#0000 aus, wenn die Operation fehlerfrei ausgeführt wird. Wird ein anderer Wert ausgegeben, entnehmen Sie der Fehlerinformation nähere Angaben hierzu.

Fehlerinformation

- Ist der ganzzahlige Eingabewert größer als K2, dann wird der Ausgang *OUT* an *HI_LIM* gebunden und ein Fehler ausgegeben.
- Ist der ganzzahlige Eingabewert kleiner als K1, dann wird der Ausgang an *LO_LIM* gebunden und ein Fehler ausgegeben.
- Der Signalzustand von *ENO* wird auf FALSE und *RET_VAL* wird auf W#16#0008 gesetzt.

17.1.10 FC 106 - UNSCALE - Werte deskalieren

Beschreibung

Die Funktion Werte deskalieren UNSCALE wandelt einen Realzahlenwert *IN*, der in physikalischen Einheiten zwischen einem unteren und einem oberen Grenzwert *LO_LIM* und *HI_LIM* skaliert ist, in einen ganzzahligen Wert um. Das Ergebnis wird in den Parameter *OUT* geschrieben. Die Funktion UNSCALE arbeitet mit der folgenden Gleichung:

$$OUT = [((IN - LO_LIM) / (HI_LIM - LO_LIM)) \cdot (K2 - K1)] + K1$$

Die Konstanten K1 und K2 werden unterschiedlich gesetzt, je nachdem, ob der Eingabewert *BIPOLAR* oder *UNIPOLAR* ist.

- *BIPOLAR*:
  – Es wird angenommen, dass der ganzzahlige Eingabewert zwischen -27648 und 27648 liegt, deshalb sind
    K1 = -27648,0 und K2 = +27648,0.
- *UNIPOLAR*:
  – Es wird angenommen, dass der ganzzahlige Eingabewert zwischen 0 und 27648 liegt, deshalb sind
    K1 = 0,0 und K2 = +27648,0.

Liegt der Eingabewert nicht in dem Bereich zwischen *LO_LIM* und *HI_LIM*, dann wird der Ausgang *OUT* an den nächsten Grenzwert (den oberen oder den unteren) für den angegebenen Bereich des jeweiligen Typs (*BIPOLAR* oder *UNIPOLAR*) gebunden und ein Fehler ausgegeben.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	Eingang	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	Ausgang	BOOL	E, A, M, D, L	■ Status – TRUE: Funktion fehlerfrei ausgeführt
IN	Eingang	REAL	E, M, D, L, P, Konstante	Eingabewert, der in einen ganzzahligen Wert deskaliert werden soll.
HI_LIM	Eingang	REAL	E, M, D, L, P, Konstante	Oberer Grenzwert in physikalischen Einheiten
LO_LIM	Eingang	REAL	E, M, D, L, P, Konstante	Unterer Grenzwert in physikalischen Einheiten
BIPOLAR	Eingang	BOOL	E, A, M, D, L	Bei dem Signalzustand 1 handelt es sich um einen bipolaren Eingabewert, bei dem Signalzustand "0" handelt es sich um einen unipolaren Eingabewert.
OUT	Ausgang	INT	E, A, M, D, L, P,	Ergebnis der Skalierung
RET_VAL	Ausgang	WORD	E, A, M, D, L, P	Gibt den Wert W#16#0000 aus, wenn die Operation fehlerfrei ausgeführt wird. Wird ein anderer Wert ausgegeben, entnehmen Sie der Fehlerinformation nähere Angaben hierzu.

Fehlerinformation

Liegt der Eingabewert nicht in dem Bereich zwischen *LO_LIM* und *HI_LIM*, dann wird der Ausgang *OUT* an den nächsten Grenzwert (den oberen oder den unteren) für den angegebenen Bereich des jeweiligen Typs (*BIPOLAR* oder *UNIPOLAR*) gebunden und ein Fehler ausgegeben. Der Signalzustand von *ENO* wird auf "0" und *RET_VAL* wird auf W#16#0008 gesetzt.

17.1.11 FC 108 - RLG_AA1 - Analogwert ausgeben

Beschreibung

Die Funktion RLG_AA1 (Analogwert ausgeben) formt einen Eingangswert *XE* (Festpunktzahl) in einen Ausgabewert für eine Analogausgabebaugruppe entsprechend dem Nennbereich zwischen *OGR* und *UGR* um. Bei Überschreiten des Nennbereichs wird eine Fehlermeldung ausgegeben.

Parameter	Datentyp	Speicherbereich	Beschreibung
XE	INT	E, A, M, L, D, Konstante	Eingangswert *XE* als Festpunktzahl
BG	INT	E, A, M, L, D, Konstante	Angabe der Baugruppenadresse
KNKT	WORD	E, A, M, L, D, Konstante	Kanalnummer KN Kanaltyp KT

Parameter	Datentyp	Speicherbereich	Beschreibung
OGR	INT	E, A, M, L, D, Konstante	Obergrenze des Eingangswerts *XE*
UGR	INT	E, A, M, L, D, Konstante	Untergrenze des Eingangswerts *XE*
FEH	BOOL	E, A, M, L, D	Fehlerbit
BU	BOOL	E, A, M, L, D	Bereichsüberschreitung

Abweichungen von S5

- Parameter BG:
  - Eine Adressprüfung findet nicht statt. Der Wertebereich ist der gesamte P-Bereich.

Diese FC wird nur dazu verwendet, den FB 251 eines bestehenden S5-Programms einer S5-CPU 941 bis 944 in eine FC eines S7-Programms für das Automatisierungssystem S7-400 umzuwandeln.

17.1.12 FC 109 - RLG_AA2 - Write Analog Value 2

Beschreibung

Die Funktion RLG_AA2 (Analogwert ausgeben) formt einen Eingangswert *XE* (Gleitpunktzahl) in einen Ausgabewert für eine Analogausgabebaugruppe entsprechend dem Nennbereich zwischen *OGR* und *UGR* um. Bei Überschreiten des Nennbereichs wird eine Fehlermeldung ausgegeben.

Parameter	Datentyp	Speicherbereich	Beschreibung
XE	REAL	E, A, M, L, D, Konstante	Eingangswert *XE* als Gleitpunktzahl
BG	INT	E, A, M, L, D, Konstante	Angabe der Baugruppenadresse
P_Q	WORD	E, A, M, L, D, Konstante	Peripheriebereich normal/erweitert
KNKT	WORD	E, A, M, L, D, Konstante	Kanalnummer KN Kanaltyp KT
OGR	REAL	E, A, M, L, D, Konstante	Obergrenze des Eingangswerts *XE*
UGR	REAL	E, A, M, L, D, Konstante	Untergrenze des Eingangswerts *XE*
FEH	BOOL	E, A, M, L, D	Fehlerbit
BU	BOOL	E, A, M, L, D	Bereichsüberschreitung

Abweichungen von S5

- Parameter BG:
  - Eine Adressprüfung findet nicht statt. Der Wertebereich ist der gesamte P-Bereich.
- In S7 wird der Parameter *P_Q* nicht ausgewertet.
- Die S5 Peripheriebereiche P/Q/IM3/IM4 werden auf den S7-Peripheriebereich abgebildet. Die Zuordnung des Peripheriebereichs müssen Sie in der Konfigurationstabelle durchführen.

Diese FC wird nur dazu verwendet, den FB 41 eines bestehenden S5-Programms einer S5-CPU 928B, 945 oder 948 in eine FC eines S7-Programms für das Automatisierungssystem S7-400 umzuwandeln.

17.1.13 FC 110 - PER_ET1 - Read/Write Ext. Per. 1

Beschreibung

Die Funktion PER_ET1 (Lesen und Schreiben für Erweiterte Peripherie) überträgt wahlweise (je nach Parametrierung) einen Peripheriebereich in einen CPU-internen Bereich oder umgekehrt. Dadurch können Eingangsbytes von der erweiterten Peripherie gelesen und Ausgangsbytes zur erweiterten Peripherie geschrieben werden. Falls als interner Bereich ein Datenbaustein gewählt wird, so muss der Baustein vom Anwender vor dem Aufruf der Funktion mit der erforderlichen Länge angelegt worden sein.

Parameter	Datentyp	Speicherbereich	Beschreibung
PBIB	WORD	E, A, M, L, D, Konstante	Angabe der zu bearbeitenden Bereiche
ANF	INT	E, A, M, L, D, Konstante	Anfang des internen Bereichs
ANEN	WORD	E, A, M, L, D, Konstante	Anfang und Ende des Blocks auf der Anschaltung
E_A	BOOL	E, A, M, L, D, Konstante	Transferrichtung
PAFE	BOOL	E, A, M, L, D	Parametrierfehler

Abweichungen von S5

- Parameter *PBIB*:
  - In S7 wird der Peripheriebereich wie folgt ausgewertet:

	S5		S7
P-Bereich	0 bis 255	P-Bereich	0 bis 255
Q-Bereich	0 bis 255	P-Bereich	256 bis 511
IM3-Bereich	0 bis 255	P-Bereich	512 bis 767
IM4-Bereich	0 bis 255	P-Bereich	768 bis 1023
DB	0 bis 255	DB	0 bis 255
DX	0 bis 255	DB	256 bis 511
M	0 bis 199	M	0 bis 199
S		Fehlermeldung: "Ungültiger Wertebereich"	

- Die S5-Peripheriebereiche P/Q/IM3/IM4 werden auf den S7-Peripheriebereich abgebildet. Die Zuordnung des Peripheriebereichs müssen Sie in der Konfigurationstabelle durchführen.

Diese FC wird nur dazu verwendet, den FB 196 eines bestehenden S5-Programms einer S5-CPU 95U, 103, 941 bis 944, 945, 928B, 948 in eine FC eines S7-Programms für das Automatisierungssystem S7-300/400 umzuwandeln.

17.1.14 FC 111 - PER_ET2 - Read/Write Ext. Per. 2

Beschreibung

Die Funktion PER_ET2 (Lesen und Schreiben für Erweiterte Peripherie) überträgt wahlweise (je nach Parametrierung) einen Peripheriebereich in einen CPU-internen Bereich oder umgekehrt. Dadurch können Eingangsbytes von der erweiterten Peripherie gelesen und Ausgangsbytes zur erweiterten Peripherie geschrieben werden. Falls als interner Bereich ein Datenbaustein gewählt wird, so muss der Baustein vom Anwender vor dem Aufruf der Funktion mit der erforderlichen Länge angelegt worden sein.

Abweichungen von S5

- Parameter *PBIB* (in DB definiert):
  – In S7 wird der Peripheriebereich wie folgt ausgewertet:

	S5		S7
P-Bereich	0 bis 255	P-Bereich	0 bis 255
Q-Bereich	0 bis 255	P-Bereich	256 bis 511
IM3-Bereich	0 bis 255	P-Bereich	512 bis 767
IM4-Bereich	0 bis 255	P-Bereich	768 bis 1023
DB	0 bis 255	DB	0 bis 255
DX	0 bis 255	DB	256 bis 511
M	0 bis 199	M	0 bis 199
S		Fehlermeldung: "Ungültiger Wertebereich"	

- Die S5-Peripheriebereiche P/Q/IM3/IM4 werden auf den S7-Peripheriebereich abgebildet. Die Zuordnung des Peripheriebereichs müssen Sie in der Konfigurationstabelle durchführen.

Diese FC wird nur dazu verwendet, den FB 197 eines bestehenden S5-Programms einer S5-CPU 95U, 103, 941 bis 944, 945, 928B, 948 in eine FC eines S7-Programms für das Automatisierungssystem S7-300/400 umzuwandeln.

17.2 IEC-Funktionen - *"IEC"*

17.2.1 Datum und Uhrzeit als zusammengesetzte Datentypen

Aktualparameter für DATE_AND_TIME

Der Datentyp DATE_AND_TIME ist ein zusammengesetzter Datentyp wie auch ARRAY, STRING und STRUCT. Die zulässigen Speicherbereiche für zusammengesetzte Datentypen sind der Datenbaustein (DB) und der Speicherbereich für Lokaldaten (L-Stack). Wenn Sie den Datentyp DATE_AND_TIME als Formalparameter in einer Anweisung verwenden, können Sie aufgrund des zusammengesetzten Datentyps die Aktualparameter nur in einem der folgenden Formate angeben:

- Als bausteinlokales Symbol aus der Variablendeklarationstabelle für einen bestimmten Baustein
- Als symbolischen Namen für einen Datenbaustein, z.B. "DB_sys_info.Zeit", der aus den beiden folgenden Teilen besteht:
  – Ein Name, der in der Symboltabelle für die Nummer des Datenbausteins definiert ist (z.B. "DB_sys_info" für DB 5)
  – Ein Name, der in dem Datenbaustein für das Element DATE_AND_TIME definiert ist (z.B. "Zeit" für eine Variable vom Datentyp DATE_AND_TIME, die in DB 5 enthalten ist)

Sie können keine Konstanten als Aktualparameter für Formalparameter von zusammengesetzten Datentypen, einschließlich DATE_AND_TIME, verwenden. Auch können Sie keine absoluten Adressen als Aktualparameter an DATE_AND_TIME übergeben.

17.2.2 FC 1 - AD_DT_TM - Zeitdauer auf einen Zeitpunkt addieren

Beschreibung

Die Funktion FC 1 addiert eine Zeitdauer *D* (Uhrzeit) einen Zeitpunkt *T* (Datum und Uhrzeit) und liefert als Ergebnis einen neuen Zeitpunkt (Datum und Uhrzeit). Der Zeitpunkt *T* muss im Bereich DT#1990-01-01-00:00:00.000 ... DT#2089-12-31-23:59:59.999 liegen. Die Funktion führt keine Eingangsprüfung durch. Liegt das Ergebnis der Addition nicht im oben angegebenen Bereich, wird das Ergebnis auf den entsprechenden Wert begrenzt und das Binärergebnis BIE auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
T*	INPUT	DATE_AND_TIME	D, L	Zeitpunkt im Format DT
D	INPUT	TIME	E, A, M, D, L, Konstante	Zeitdauer im Format TIME
RET_VAL*	OUTPUT	DATE_AND_TIME	D, L	Summe im Format DT

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.3 FC 2 - CONCAT - Zusammenfassen zweier STRING-Variablen

Beschreibung

Die Funktion FC 2 fasst zwei STRING-Variablen zu einer Zeichenkette zusammen. Ist die Ergebniszeichenkette länger als die am Ausgangsparameter angelegte Variable, wird die Ergebniszeichenkette auf die maximal eingerichtete Länge begrenzt und das BIE-Bit auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN1*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
IN2*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
RET_VAL*	OUTPUT	STRING	D, L	Zusammengefasste Zeichenkette

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.4 FC 3 - D_TOD_DT - Zusammenfassen DATE und TIME_OF_DAY

Beschreibung

Die Funktion FC 3 fasst die Datenformate DATE und TIME_OF_DAY (TOD) zusammen und wandelt diese Formate in das Datenformat DATE_AND_TIME (DT) um. Der Eingangswert *IN1* muss im Bereich DATE#1990-01-01 ... DATE#2089-12-31 liegen. Die Funktion führt keine Eingangsprüfung durch und meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN1	INPUT	DATE	E, A, M, D, L, Konstante	Eingangsvariable im Format DATE
IN2	INPUT	TIME_OF_DAY	E, A, M, D, L, Konstante	Eingangsvariable im Format TOD

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
RET_VAL*	OUTPUT	DATE_AND_TIME	D, L	Rückgabewert im Format DT

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.5 FC 4 - DELETE - Löschen in einer STRING-Variable

Beschreibung

Die Funktion FC 4 löscht in einer Zeichenkette *L* Zeichen ab dem *P*. Zeichen (einschließlich). Die Funktion meldet keine Fehler.

- Ist *L* und/oder *P* gleich Null oder ist *P* größer als die aktuelle Länge der Eingangszeichenkette, wird die Eingangszeichenkette zurückgeliefert.
- Ist die Summe aus *L* und *P* größer als die Eingangszeichenkette, wird bis zum Ende der Zeichenkette gelöscht.
- Ist *L* und/oder *P* negativ wird ein Leerstring ausgegeben und das BIE-Bit auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN*	INPUT	STRING	D, L	STRING-Variable, in der gelöscht wird
L	INPUT	INT	E, A, M, D, L, Konstante	Anzahl der zu löschenden Zeichen
P	INPUT	INT	E, A, M, D, L, Konstante	Position des 1. zu löschenden Zeichens
RET_VAL*	OUTPUT	STRING	D, L	Ergebniszeichenkette

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.6 FC 5 - DI_STRNG - Formatwandlung DINT nach STRING

Beschreibung

Die Funktion FC 5 wandelt eine Variable im DINT-Format in eine Zeichenkette um. Die Zeichenkette wird mit einem führenden Vorzeichen dargestellt. Ist die am Rückgabeparameter angegebene Variable zu kurz, findet keine Wandlung statt und das BIE-Bit wird auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
I	INPUT	DINT	E, A, M, D, L, Konstante	Eingangswert
RET_VAL*	OUTPUT	STRING	D, L	Ergebniszeichenkette

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.7 FC 6 - DT_DATE - Extrahieren DATE aus DT

Beschreibung

Die Funktion FC 6 extrahiert das Datenformat DATE aus dem Format DATE_AND_TIME. DATE liegt zwischen den Grenzen DATE#1990-1-1 ... DATE#2089-12-31. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im DT-Format
RET_VAL	OUTPUT	DATE	E, A, M, D, L	Rückgabewert im Format DATE

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.8 FC 7 - DT_DAY - Extrahieren des Wochentags aus DT

Beschreibung

Die Funktion FC 7 extrahiert den Wochentag aus dem Format DATE_AND_TIME. Die Funktion meldet keine Fehler. Der Wochentag wird als INTEGER-Wert zurückgeliefert:

- 1: Sonntag
- 2: Montag
- 3: Dienstag
- 4: Mittwoch
- 5: Donnerstag
- 6: Freitag
- 7: Samstag

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im Format DT
RET_VAL	OUTPUT	INT	E, A, M, D, L	Rückgabewert im Format INT

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.9 FC 8 - DT_TOD - Extrahieren TIME_OF_DAY aus DT

Beschreibung

Die Funktion FC 8 extrahiert das Datenformat TIME_OF_DAY aus dem Format DATE_AND_TIME. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im Format DT
RET_VAL	OUTPUT	TIME_OF_DAY	E, A, M, D, L	Rückgabewert im Format TOD

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.10 FC 9 - EQ_DT - Vergleich DT auf gleich

Beschreibung

Die Funktion FC 9 vergleicht die Inhalte zweier Variablen im Format DATE_AND_TIME auf gleich und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn der Zeitpunkt am Parameter *DT1* gleich dem Zeitpunkt am Parameter *DT2* ist. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
DT1*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im Format DT
DT2*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im Format DT
RET_VAL	OUTPUT	BOOL	E, A, M, D, L	Vergleichsergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.11 FC 10 - EQ_STRNG - Vergleich STRING auf gleich

Beschreibung

Die Funktion FC 10 vergleicht die Inhalte zweier Variablen im Format STRING auf gleich und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn die Zeichenkette am Parameter *S1* gleich der Zeichenkette am Parameter *S2* ist. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
S1*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
S2*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
RET_VAL	OUTPUT	BOOL	E, A, M, D, L	Vergleichsergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.12 FC 11 - FIND - Suchen in einer STRING-Variable

Beschreibung

Die Funktion FC 11 liefert die Position der zweiten Zeichenkette *IN2* innerhalb der ersten Zeichenkette *IN1*. Die Suche beginnt links; es wird das erste Auftreten der Zeichenkette gemeldet. Ist die zweite Zeichenkette in der ersten nicht vorhanden, wird Null zurückgemeldet. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN1*	INPUT	STRING	D, L	STRING-Variable, in der gesucht wird
IN2*	INPUT	STRING	D, L	zu suchende STRING-Variable

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
RET_VAL	OUTPUT	INT	E, A, M, D, L	Position der gefundenen Zeichenkette

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.13 FC 12 - GE_DT - Vergleich DT auf größer oder gleich

Beschreibung

Die Funktion FC 12 vergleicht die Inhalte zweier Variablen im Format DATE_AND_TIME auf größer oder gleich und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn der Zeitpunkt am Parameter *DT1* größer (jünger) ist als der Zeitpunkt am Parameter *DT2*, oder wenn beide Zeitpunkte gleich sind. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
DT1*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im Format DT
DT2*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im Format DT
RET_VAL	OUTPUT	BOOL	E, A, M, D, L	Vergleichsergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.14 FC 13 - GE_STRNG - Vergleich STRING auf größer oder gleich

Beschreibung

Die Funktion FC 13 vergleicht die Inhalte zweier Variablen im Format STRING auf größer oder gleich und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn die Zeichenkette am Parameter *S1* größer oder gleich der Zeichenkette am Parameter *S2* ist. Die Zeichen werden beginnend von links über ihre ASCII-Codierung verglichen (z.B. ist 'a' größer als 'A'). Das erste unterschiedliche Zeichen entscheidet über das Vergleichsergebnis. Ist der linke Teil der längeren Zeichenkette identisch mit der kürzeren Zeichenkette, gilt die längere Zeichenkette als größer. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
S1*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
S2*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
RET_VAL	OUTPUT	BOOL	E, A, M, D, L	Vergleichsergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.15 FC 14 - GT_DT - Vergleich DT auf größer

Beschreibung

Die Funktion FC 14 vergleicht die Inhalte zweier Variablen im Format DATE_AND_TIME auf größer und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn der Zeitpunkt am Parameter *DT1* größer (jünger) ist als der Zeitpunkt am Parameter *DT2*. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
DT1*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im Format DT
DT2*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im Format DT
RET_VAL	OUTPUT	BOOL	E, A, M, D, L	Vergleichsergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.16 FC 15 - GT_STRNG - Vergleich STRING auf größer

Beschreibung

Die Funktion FC 15 vergleicht die Inhalte zweier Variablen im STRING-Format auf größer und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn die Zeichenkette am Parameter *S1* größer als die Zeichenkette am Parameter *S2* ist. Die Zeichen werden beginnend von links über ihre ASCII-Codierung verglichen (z.B. ist 'a' größer als 'A'). Das erste unterschiedliche Zeichen entscheidet über das Vergleichsergebnis. Ist der linke Teil der längeren Zeichenkette identisch mit der kürzeren Zeichenkette, gilt die längere Zeichenkette als größer. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
S1*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
S2*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
RET_VAL	OUTPUT	BOOL	E, A, M, D, L	Vergleichsergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.17 FC 16 - I_STRNG - Formatwandlung INT nach STRING

Beschreibung

Die Funktion FC 16 wandelt eine Variable im INT-Format in eine Zeichenkette um. Die Zeichenkette wird mit einem führenden Vorzeichen dargestellt. Ist die am Rückgabeparameter angegebene Variable zu kurz, findet keine Wandlung statt und das BIE-Bit wird auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
I	INPUT	INT	E, A, M, D, L, Konstante	Eingangswert
RET_VAL*	OUTPUT	STRING	D, L	Ergebniszeichenkette

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.18 FC 17 - INSERT - Einfügen in eine STRING-Variable

Beschreibung

Die Funktion FC 17 fügt die Zeichenkette am Parameter *IN2* in die Zeichenkette am Parameter *IN1* nach dem *P*. Zeichen ein.

- Ist *P* gleich Null, wird die zweite Zeichenkette vor der ersten Zeichenkette eingefügt.
- Ist *P* größer als die aktuelle Länge der ersten Zeichenkette, wird die zweite Zeichenkette an die erste angehängt.
- Ist *P* negativ, wird ein Leerstring ausgegeben und das BIE-Bit auf "0" gesetzt. Das BIE-Bit wird auch auf "0" gesetzt, wenn die Ergebniszeichenkette länger ist als die am Ausgangsparameter angegebene Variable; in diesem Fall wird die Ergebniszeichenkette auf die maximal eingerichtete Länge begrenzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN1*	INPUT	STRING	D, L	STRING-Variable, in die eingefügt wird
IN2*	INPUT	STRING	D, L	einzufügende STRING-Variable
P	INPUT	INT	E, A, M, D, L, Konstante	Einfügeposition
RET_VAL*	OUTPUT	STRING	D, L	Ergebniszeichenkette

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.19 FC 18 - LE_DT - Vergleich DT auf kleiner oder gleich

Beschreibung

Die Funktion FC 18 vergleicht die Inhalte zweier Variablen im Format DATE_AND_TIME auf kleiner oder gleich und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn der Zeitpunkt am Parameter *DT1* kleiner (älter) ist als der Zeitpunkt am Parameter *DT2*, oder wenn beide Zeitpunkte gleich sind. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
DT1*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im Format DT
DT2*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im Format DT
RET_VAL*	OUTPUT	BOOL	E, A, M, D, L	Vergleichsergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.20 FC 19 - LE_STRNG - Vergleich STRING auf kleiner oder gleich

Beschreibung

Die Funktion FC 19 vergleicht die Inhalte zweier Variablen im STRING-Format auf kleiner oder gleich und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn die Zeichenkette am Parameter *S1* kleiner oder gleich der Zeichenkette am Parameter *S2* ist. Die Zeichen werden beginnend von links über ihre ASCII-Codierung verglichen (z.B. ist 'A' kleiner als 'a'). Das erste unterschiedliche Zeichen entscheidet über das Vergleichsergebnis. Ist der linke Teil der längeren Zeichenkette identisch mit der kürzeren Zeichenkette, gilt die kürzere Zeichenkette als kleiner. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
S1*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
S2*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
RET_VAL	OUTPUT	BOOL	E, A, M, D, L	Vergleichsergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.21 FC 20 - LEFT - Linker Teil einer STRING-Variable

Beschreibung

Die Funktion FC 20 liefert die ersten *L* Zeichen einer Zeichenkette.

- Ist *L* größer als die aktuelle Länge der STRING-Variable, wird der Eingangswert zurückgeliefert.
- Bei *L* = 0 und bei einem Leerstring als Eingangswert wird ein Leerstring zurückgeliefert.
- Ist *L* negativ wird ein Leerstring ausgegeben und das BIE-Bit auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
L	INPUT	INT	E, A, M, D, L, Konstante	Länge der linken Zeichenkette
RET_VAL*	OUTPUT	STRING	D, L	Ausgangsvariable im Format STRING

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.22 FC 21 - LEN - Länge einer STRING-Variablen

Beschreibung

Eine STRING-Variable enthält zwei Längen:

- Maximale Länge
  - Sie wird bei der Definition der Variablen in eckigen Klammern vorgegeben.
- Aktuelle Länge
  - Das ist die Anzahl der momentan gültigen Zeichen.

Die aktuelle Länge ist kleiner oder gleich der maximalen Länge. Die Anzahl der durch eine Zeichenkette belegten Bytes ist um 2 größer als die maximale Länge. Die Funktion FC 21 gibt die aktuelle Länge einer Zeichenkette (Anzahl der gültigen Zeichen) als Rückgabewert aus. Ein Leerstring (' ') hat die Länge Null. Die maximale Länge beträgt 254. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
S*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
RET_VAL	OUTPUT	INT	E, A, M, D, L	Anzahl der aktuellen Zeichen

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.23 FC 22 - LIMIT - Begrenzer

Beschreibung

Die Funktion FC 22 begrenzt den Zahlenwert einer Variablen auf parametrierbare Grenzwerte.

- Als Eingangswerte sind Variablen vom Datentyp INT, DINT und REAL zugelassen.
- Alle parametrierten Variablen müssen vom gleichen Datentyp sein.
- Die Art der Variable wird über den ANY-Pointer erkannt.
- *MN* darf nicht größer sein als *MX*.
- Der Ausgangswert bleibt unverändert, und das BIE-Bit wird auf "0" gesetzt, wenn:
  - eine parametrierte Variable einen unzulässigen Datentyp hat.
  - alle parametrierten Variablen untereinander nicht den gleichen Datentyp haben.
  - der untere Grenzwert größer ist als der obere Grenzwert.
  - eine REAL-Variable keine gültige Gleitpunktzahl darstellt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
MN	INPUT	ANY	E, A, M, D, L	Untergrenze
IN	INPUT	ANY	E, A, M, D, L	Eingangsvariable
MX	INPUT	ANY	E, A, M, D, L	Obergrenze
RET_VAL	OUTPUT	ANY	E, A, M, D, L	Begrenzte Ausgangsvariable

17.2.24 FC 23 - LT_DT - Vergleich DT auf kleiner

Beschreibung

Die Funktion FC 23 vergleicht die Inhalte zweier Variablen im Format DATE_AND_TIME auf kleiner und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn der Zeitpunkt am Parameter *DT1* kleiner (älter) ist als der Zeitpunkt am Parameter *DT2*. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
DT1*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im Format DT
DT2*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im Format DT
RET_VAL	OUTPUT	BOOL	E, A, M, D, L	Vergleichsergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.25 FC 24 - LT_STRNG - Vergleich STRING auf kleiner

Beschreibung

Die Funktion FC 24 vergleicht die Inhalte zweier Variablen im STRING-Format auf kleiner und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn die Zeichenkette am Parameter *S1* kleiner als die Zeichenkette am Parameter *S2* ist. Die Zeichen werden beginnend von links über ihre ASCII-Codierung verglichen (z.B. ist 'A' kleiner als 'a'). Das erste unterschiedliche Zeichen entscheidet über das Vergleichsergebnis. Ist der linke Teil der längeren Zeichenkette identisch mit der kürzeren Zeichenkette, gilt die kürzere Zeichenkette als kleiner. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
S1*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
S2*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
RET_VAL	OUTPUT	BOOL	E, A, M, D, L	Vergleichsergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.26 FC 25 - MAX - Maximumauswahl

Beschreibung

Die Funktion FC 25 wählt aus drei numerischen Variablenwerten den größten aus.

- Als Eingangswerte sind Variablen vom Datentyp INT, DINT und REAL zugelassen.
- Alle parametrierten Variablen müssen vom gleichen Datentyp sein.
- Die Art der Variable wird über den ANY-Pointer erkannt.
- Der Ausgangswert bleibt unverändert, und das BIE-Bit wird auf "0" gesetzt, wenn:
  - eine parametrierte Variable einen unzulässigen Datentyp hat.
  - alle parametrierten Variablen untereinander nicht den gleichen Datentyp haben.
  - eine REAL-Variable keine gültige Gleitpunktzahl darstellt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN1	INPUT	ANY	E, A, M, D, L	1. Eingangswert
IN2	INPUT	ANY	E, A, M, D, L	2. Eingangswert

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN3	INPUT	ANY	E, A, M, D, L	3. Eingangswert
RET_VAL	OUTPUT	ANY	E, A, M, D, L	Größter der Eingangswerte

Die zugelassenen Datentypen INT, DINT und REAL müssen im ANY-Pointer angegeben werden. Parameter wie "MD20" sind ebenfalls zulässig; dazu muss "MD20" aber in "Symbol" mit dem entsprechenden Datentyp definiert werden.

Beispiel in AWL:

```
CALL FC 25
IN1 := P#M 10.0 DINT 1
IN2 := MD20
IN3 := P#DB1.DBX 0.0 DINT 1
RET_VAL := P#M 40.0 DINT 1
= M 0.0
```

17.2.27 FC 26 - MID - Mittlerer Teil einer STRING-Variablen

Beschreibung

Die Funktion FC 26 liefert den mittleren Teil einer Zeichenkette (*L* Zeichen ab dem *P*. Zeichen einschließlich).

- Geht die Summe aus *L* und (*P*-1) über die aktuelle Länge der STRING-Variable hinaus, wird eine Zeichenkette ab dem *P*. Zeichen bis zum Ende des Eingangswerts geliefert.
- In allen anderen Fällen (*P* liegt außerhalb der aktuellen Länge, *P* und/oder *L* gleich Null oder negativ) wird ein Leerstring ausgegeben und das BIE-Bit auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
L	INPUT	INT	E, A, M, D, L, Konstante	Länge der mittleren Zeichenkette
P	INPUT	INT	E, A, M, D, L, Konstante	Position des ersten Zeichens
RET_VAL*	OUTPUT	STRING	D, L	Ausgangsvariable im Format STRING

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.28 FC 27 - MIN - Minimumauswahl

Beschreibung

Die Funktion FC 27 wählt aus drei numerischen Variablenwerten den kleinsten aus.

- Als Eingangswerte sind Variablen vom Datentyp INT, DINT und REAL zugelassen.
- Alle parametrierten Variablen müssen vom gleichen Datentyp sein.

- Die Art der Variable wird über den ANY-Pointer erkannt.
- Der Ausgangswert bleibt unverändert, und das BIE-Bit wird auf "0" gesetzt, wenn:
  - eine parametrierte Variable einen unzulässigen Datentyp hat.
  - alle parametrierten Variablen untereinander nicht den gleichen Datentyp haben.
  - eine REAL-Variable keine gültige Gleitpunktzahl darstellt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN1	INPUT	ANY	E, A, M, D, L	1. Eingangswert
IN2	INPUT	ANY	E, A, M, D, L	2. Eingangswert
IN3	INPUT	ANY	E, A, M, D, L	3. Eingangswert
RET_VAL	OUTPUT	ANY	E, A, M, D, L	Kleinster der Eingangswerte

Die zugelassenen Datentypen INT, DINT und REAL müssen im ANY-Pointer angegeben werden. Parameter wie "MD20" sind ebenfalls zulässig; dazu muss "MD20" aber in "Symbol" mit dem entsprechenden Datentyp definiert werden.

Beispiel in AWL:

```
CALL FC 27
IN1 := P#M 10.0 DINT 1
IN2 := MD20
IN3 := P#DB1.DBX 0.0 DINT 1
RET_VAL := P#M 40.0 DINT 1
= M 0.0
```

17.2.29 FC 28 - NE_DT - Vergleich DT auf ungleich

Beschreibung

Die Funktion FC 28 vergleicht die Inhalte zweier Variablen im Format DATE_AND_TIME auf ungleich und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn der Zeitpunkt am Parameter *DT1* ungleich dem Zeitpunkt am Parameter *DT2* ist. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
DT1*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im Format DT
DT2*	INPUT	DATE_AND_TIME	D, L	Eingangsvariable im Format DT
RET_VAL	OUTPUT	BOOL	E, A, M, D, L	Vergleichsergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.30 FC 29 - NE_STRNG - Vergleich STRING auf ungleich

Beschreibung

Die Funktion FC 29 vergleicht die Inhalte zweier Variablen im STRING-Format auf ungleich und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand "1", wenn die Zeichenkette am Parameter *S1* ungleich der Zeichenkette am Parameter *S2* ist. Die Funktion meldet keine Fehler.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
S1*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
S2*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
RET_VAL	OUTPUT	BOOL	E, A, M, D, L	Vergleichsergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.31 FC 30 - R_STRNG - Formatwandlung REAL nach STRING

Beschreibung

Die Funktion FC 30 wandelt eine Variable im REAL-Format in eine Zeichenkette um.

- Die Zeichenkette wird mit 14 Stellen dargestellt:
  ±v.nnnnnnnE±xx
  - ±: Vorzeichen
  - v: 1 Vorkommastelle
  - n: 7 Nachkommastellen
  - x: 2 Exponentenstellen
- Ist die am Rückgabeparameter angegebene Variable zu kurz oder liegt am Parameter IN keine gültige Gleitpunktzahl an, findet keine Wandlung statt und das BIE-Bit wird auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN	INPUT	REAL	E, A, M, D, L, Konstante	Eingangswert
RET_VAL*	OUTPUT	STRING	D, L	Ergebniszeichenkette

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.32 FC 31 - REPLACE - Ersetzen in einer STRING-Variable

Beschreibung

Die Funktion FC 31 ersetzt die Anzahl von *L* Zeichen der ersten Zeichenkette *IN1* ab dem *P*. Zeichen (einschließlich) durch die vollständige zweite Zeichenkette *IN2*.

- Ist *L* gleich Null und *P* ungleich Null, wird die erste Zeichenkette zurückgeliefert.
- Ist *L* gleich Null und *P* gleich Null, wird die zweite Zeichenkette der ersten Zeichenkette vorangestellt.
- Ist *L* ungleich Null und *P* gleich Null oder Eins, wird ab dem 1. Zeichen (einschließlich) ersetzt.

- Liegt *P* außerhalb der ersten Zeichenkette, wird die zweite Zeichenkette an die erste Zeichenkette angehängt.
- Ist *L* und/oder *P* negativ wird ein Leerstring ausgegeben und das BIE-Bit auf "0" gesetzt. Das BIE-Bit wird auch auf "0" gesetzt, wenn die Ergebniszeichenkette länger ist als die am Ausgangsparameter angegebene Variable ist; in diesem Fall wird die Ergebniszeichenkette auf die maximal eingerichtete Länge begrenzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN1*	INPUT	STRING	D, L	STRING-Variable, in die eingesetzt wird
IN2*	INPUT	STRING	D, L	einzusetzende STRING-Variable
L	INPUT	INT	E, A, M, D, L, Konstante	Anzahl der zu ersetzenden Zeichen
P	INPUT	INT	E, A, M, D, L, Konstante	Position des 1. ersetzten Zeichens
RET_VAL*	OUTPUT	STRING	D, L	Ergebniszeichenkette

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.33 FC 32 - RIGHT - Rechter Teil einer STRING-Variable

Beschreibung

Die Funktion FC 32 liefert die letzten *L* Zeichen einer Zeichenkette.

- Ist *L* größer als die aktuelle Länge der STRING-Variable, wird der Eingangswert zurückgeliefert.
- Bei *L* = 0 und bei einem Leerstring als Eingangswert wird ein Leerstring zurückgeliefert.
- Ist *L* negativ, wird ein Leerstring ausgegeben und das Binärergebnis BIE auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN*	INPUT	STRING	D, L	Eingangsvariable im Format STRING
L	INPUT	INT	E, A, M, D, L, Konstante	Länge der rechten Zeichenkette
RET_VAL*	OUTPUT	STRING	D, L	Ausgangsvariable im Format STRING

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.34 FC 33 - S5TI_TIM - Formatwandlung S5TIME nach TIME

Beschreibung

Die Funktion FC 33 wandelt das Datenformat S5TIME in das Datenformat TIME um. Liegt das Ergebnis der Wandlung außerhalb des TIME-Zahlenbereichs, wird das Ergebnis auf den entsprechenden Wert begrenzt und das Binärergebnis BIE auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN	INPUT	S5TIME	E, A, M, D, L, Konstante	Eingangsvariable im Format S5TIME
RET_VAL	OUTPUT	TIME	E, A, M, D, L	Rückgabewert im Format TIME

17.2.35 FC 34 - SB_DT_DT - Zwei Zeitpunkte subtrahieren

Beschreibung

Die Funktion FC 34 subtrahiert zwei Zeitpunkte *DTx* (Datum und Uhrzeit) und liefert als Ergebnis eine Zeitdauer (Uhrzeit). Die Zeitpunkte *DTx* müssen in Bereich DT#1990-01-01-00:00:00.000 ... DT#2089-12-31-23:59:59.999 liegen. Die Funktion führt keine Eingangsprüfung durch. Es gilt:

- Ist *DT1* > *DT2*, so ist das Ergebnis positiv.
- Ist *DT1* < *DT2*, so ist das Ergebnis negativ.
- Liegt das Ergebnis der Subtraktion außerhalb des TIME-Zahlenbereichs, wird das Ergebnis auf den entsprechenden Wert begrenzt und das Binärergebnis BIE auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
DT1*	INPUT	DATE_AND_TIME	D, L	1. Zeitpunkt im Format DT
DT2*	INPUT	DATE_AND_TIME	D, L	2. Zeitpunkt im Format DT
RET_VAL	OUTPUT	TIME	E, A, M, D, L	Differenz im Format TIME

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.36 FC 35 - SB_DT_TM - Zeitdauer von einem Zeitpunkt subtrahieren

Beschreibung

Die Funktion FC 35 subtrahiert eine Zeitdauer *D* (Uhrzeit) von einem Zeitpunkt (Datum und Uhrzeit) und liefert als Ergebnis einen neuen Zeitpunkt (Datum und Uhrzeit). Der Zeitpunkt *T* muss im Bereich DT#1990-01-01-00:00:00.000 ... DT#2089-12-31-23:59:59.999 liegen. Die Funktion führt keine Eingangsprüfung durch. Liegt das Ergebnis der Subtraktion nicht im oben angegebenen Bereich, wird das Ergebnis auf den entsprechenden Wert begrenzt und das Binärergebnis BIE auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
T*	INPUT	DATE_AND_TIME	D, L	Zeitpunkt im Format DT
D	INPUT	TIME	E, A, M, D, L, Konstante	Zeitdauer im Format TIME
RET_VAL*	OUTPUT	DATE_AND_TIME	D, L	Differenz im Format DT

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.37 FC 36 - SEL - Binärauswahl

Beschreibung

Die Funktion FC 36 wählt abhängig von einem Schalter *G* einen aus zwei Variablenwerten aus.

- Als Eingangswerte an den Parametern *IN0* und *IN1* sind Variablen mit allen Datentypen zugelassen, die der Datenbreite Bit, Byte, Wort und Doppelwort entsprechen (nicht Datentyp DT und STRING).
- *IN0*, *IN1* und *RET_VAL* müssen vom gleichen Datentyp sein.
- Der Ausgangswert bleibt unverändert, und das BIE-Bit wird auf "0" gesetzt, wenn:
  - eine parametrierte Variable einen unzulässigen Datentyp hat.
  - alle parametrierten Variablen untereinander nicht den gleichen Datentyp haben.
  - eine REAL-Variable keine gültige Gleitpunktzahl darstellt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
G	INPUT	BOOL	E, A, M, D, L, Konstante	Auswahlschalter
IN0	INPUT	ANY	E, A, M, D, L	1. Eingangswert
IN1	INPUT	ANY	E, A, M, D, L	2. Eingangswert
RET_VAL	OUTPUT	ANY	E, A, M, D, L	Ausgewählter Eingangswert

17.2.38 FC 37 - STRNG_DI - Formatwandlung STRING nach DINT

Beschreibung

Die Funktion FC 37 wandelt eine Zeichenkette in eine Variable im Format DINT um

- Das erste Zeichen in der Zeichenkette darf ein Vorzeichen oder eine Ziffer sein, die dann folgenden Zeichen müssen aus Ziffern bestehen.
- Ist die Länge der Zeichenkette Null oder größer als 11 oder befinden sich unerlaubte Zeichen in der Zeichenkette, findet keine Wandlung statt und das BIE-Bit wird auf "0" gesetzt.
- Liegt das Ergebnis der Wandlung außerhalb des DINT-Zahlenbereichs, wird das Ergebnis auf den entsprechenden Wert begrenzt und das Binärergebnis BIE auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
S*	INPUT	STRING	D, L	Eingangszeichenkette
RET_VAL	OUTPUT	DINT	E, A, M, D, L	Ergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.39 FC 38 - STRNG_I - Formatwandlung STRING nach INT

Beschreibung

Die Funktion FC 38 wandelt eine Zeichenkette in eine Variable im Format INT um.

- Das erste Zeichen in der Zeichenkette darf ein Vorzeichen oder eine Ziffer sein, die dann folgenden Zeichen müssen aus Ziffern bestehen.
- Ist die Länge der Zeichenkette Null oder größer als 6 oder befinden sich unerlaubte Zeichen in der Zeichenkette, findet keine Wandlung statt und das BIE-Bit wird auf "0" gesetzt.
- Liegt das Ergebnis der Wandlung außerhalb des INT-Zahlenbereichs, wird das Ergebnis auf den entsprechenden Wert begrenzt und das Binärergebnis BIE auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
S*	INPUT	STRING	D, L	Eingangszeichenkette
RET_VAL	OUTPUT	INT	E, A, M, D, L	Ergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.40 FC 39 - STRNG_R - Formatwandlung STRING nach REAL

Beschreibung

Die Funktion FC 39 wandelt eine Zeichenkette in eine Variable im Format REAL um.

- Die Zeichenkette muss in folgendem Format vorliegen:
  ±v.nnnnnnnE±xx
  - ±: Vorzeichen
  - v: 1 Vorkommastellen
  - n: 7 Nachkommastellen
  - x: 2 Exponentenstellen
- Ist die Länge der Zeichenkette kleiner als 14 oder ist sie nicht wie oben gezeigt aufgebaut, findet keine Wandlung statt und das BIE-Bit wird auf "0" gesetzt.
- Liegt das Ergebnis der Wandlung außerhalb des REAL-Zahlenbereichs, wird das Ergebnis auf den entsprechenden Wert begrenzt und das Binärergebnis BIE auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
S*	INPUT	STRING	D, L	Eingangszeichenkette
RET_VAL	OUTPUT	REAL	E, A, M, D, L	Ergebnis

*) Parameter kann nur mit einer symbolisch definierten Variablen belegt werden.

17.2.41 FC 40 - TIM_S5TI - Formatwandlung TIME nach S5TIME

Beschreibung

Die Funktion FC 40 wandelt das Datenformat TIME in das Format S5TIME um. Hierbei wird immer abgerundet. Ist der Eingangsparameter größer als das darstellbare S5TIME-Format (TIME#02:46:30.000), wird als Ergebnis S5TIME#999.3 ausgegeben und das Binärergebnis BIE auf "0" gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
IN	INPUT	TIME	E, A, M, D, L, Konstante	Eingangsvariable im Format TIME
RET_VAL	OUTPUT	S5TIME	E, A, M, D, L	Rückgabewert im Format S5TIME

17.3 Ein-/Ausgabe - *"IO"*

17.3.1 FB 20 - GETIO - PROFIBUS/PROFINET alle Eingänge lesen

Beschreibung

Mit dem FB 20 GETIO lesen Sie alle Eingänge eines PROFIBUS-DP-Slave / PROFINET IO-Devices konsistent aus. Der FB 20 ruft dabei die SFC 14 DPRD_DAT auf. Falls bei der Datenübertragung kein Fehler auftrat, werden die gelesenen Daten in den durch *INPUTS* aufgespannten Zielbereich eingetragen. Der Zielbereich muss dieselbe Länge aufweisen, die Sie für die selektierte Komponente projektiert haben. Bei einem PROFIBUS-DP-Slave mit modularem Aufbau bzw. mit mehreren DP-Kennungen können Sie mit einem FB 20-Aufruf jeweils nur auf die Daten einer Komponente / DP-Kennung unter der projektierten Anfangsadresse zugreifen.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
ID	INPUT	DWORD	E, A, M, D, L, Konstante	■ low word: logische Adresse der DP-Slave- / PROFINET-IO-Komponente (Baugruppe bzw. Modul) ■ high word: irrelevant
STATUS	OUTPUT	DWORD	E, A, M, D, L	enthält die Fehlerinformation der SFC 14 DPRD_DAT in der Form DW#16#40xxxx00
LEN	OUTPUT	INT	E, A, M, D, L	Anzahl der gelesenen Daten in Bytes
INPUTS	IN_OUT	ANY	E, A, M, D	Zielbereich für die gelesenen Daten. Er muss genauso lang sein, wie Sie für die selektierte DP-Slave- / PROFINET-IO-Komponente projektiert haben. Es ist nur der Datentyp BYTE zulässig.

Fehlerinformationen

Siehe SFC 14 - DPRD_DAT - Konsistente Nutzdaten lesen. *873*

17.3.2 FB 21 - SETIO - PROFIBUS/PROFINET alle Ausgänge schreiben

Beschreibung

Mit dem FB 21 SETIO übertragen Sie die Daten aus dem durch *OUTPUTS* aufgespannten Quellbereich konsistent zum adressierten PROFIBUS-DP-Slave / PROFINET IO-Device und ggf. ins Prozessabbild (nämlich dann, wenn Sie den betroffenen Adressbereich des DP-Normslaves als Konsistenzbereich in einem Prozessabbild projektiert haben). Der FB 21 ruft dabei die SFC 15 DPWR_DAT auf. Der Quellbereich muss dieselbe Länge aufweisen, die Sie für die selektierte Komponente projektiert haben. Bei einem DP-Normslave mit modularem Aufbau bzw. mit mehreren DP-Kennungen können Sie mit einem FB 20-Aufruf nur auf eine Komponente / DP-Kennung des DP-Slaves zugreifen.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
ID	INPUT	DWORD	E, A, M, D, L, Konstante	■ Low word: logische Adresse der DPSlave- / PROFINET-IO-Komponente (Baugruppe bzw. Modul) ■ High word: irrelevant
LEN	INPUT	INT	E, A, M, D, L	Irrelevant
STATUS	OUTPUT	DWORD	E, A, M, D, L	enthält die Fehlerinformation der SFC 15 DPWR_DAT in der Form DW#16#40xxxx00
OUTPUTS	IN_OUT	ANY	E, A, M, D	Quellbereich für die zu schreibenden Daten. Er muss genauso lang sein, wie Sie für die selektierte DP-Slave- / PROFINET-IO-Komponente projektiert haben. Es ist nur der Datentyp BYTE zulässig.

Fehlerinformationen Siehe SFC 15 - DPWR_DAT - Konsistente Nutzdaten schreiben. ↳ *874*

17.3.3 FB 22 - GETIO_PART - PROFIBUS/PROFINET Teil-Eingänge lesen

Beschreibung

Mit dem FB 22 GETIO_PART lesen Sie konsistent einen Teil des zu einem PROFIBUS-DP-Slave / PROFINET IO-Device gehörenden Prozessabbildbereichs. Der FB 22 ruft dabei die SFC 81 UBLKMOV auf.

Sie müssen dem OB, in dem der FB 22 GETIO_PART aufgerufen wird, ein Teilprozessabbild der Eingänge zuordnen. Sie müssen weiterhin vor Aufruf des FB 22 den zugehörigen PROFIBUS-DP-Slave bzw. das zugehörige PROFINET IO-Device in dieses Teilprozessabbild der Eingänge aufnehmen. Falls Ihre CPU keine Teilprozessabbilder kennt oder Sie den FB 22 im OB 1 aufrufen wollen, müssen Sie vor Aufruf des FB 22 den zugehörigen PROFIBUS-DP-Slave bzw. das zugehörige PROFINET IO-Device in das Prozessabbild der Eingänge aufnehmen. Über die Parameter OFFSET und LEN legen Sie die auszulesende Teilmenge des Prozessabbildbereichs der über ID adressierten Komponente fest. Falls bei der Datenübertragung kein Fehler auftrat, erhält ERROR den Wert FALSE, und die gelesenen Daten werden in den durch INPUTS aufgespannten Zielbereich eingetragen. Falls bei der Datenübertragung ein Fehler auftrat, erhält ERROR den Wert TRUE, und STATUS erhält die Fehlerinformation der SFC 81 UBLKMOV. Falls der Zielbereich (Parameter INPUTS) kleiner ist als LEN, werden so viele Bytes übertragen, wie INPUTS aufnehmen kann. ERROR erhält den Wert FALSE. Falls der Zielbereich größer ist als LEN, werden die ersten LEN Bytes des Zielbereichs beschrieben. ERROR erhält den Wert FALSE.

Der FB 22 GETIO_PART überprüft im Prozessabbild der Eingänge keine Grenzen zwischen Daten, die zu verschiedenen PROFIBUS-DP- bzw. PROFINET IO- Komponenten gehören. Sie müssen daher selbst darauf achten, dass der über OFFSET und LEN festgelegte Prozessabbildbereich zu einer Komponente gehört. Das Komponenten übergreifende Lesen kann für zukünftige Systeme nämlich nicht garantiert werden und gefährdet die Übertragbarkeit auf Systeme anderer Hersteller.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
ID	INPUT	DWORD	E, A, M, D, L, Konstante	■ Low word: logische Adresse der DP-Slave- / PROFINET-IO-Komponente (Baugruppe bzw. Modul) ■ High word: irrelevant
OFFSET	INPUT	INT	E, A, M, D, L, Konstante	Nummer des ersten zu lesenden Bytes im Prozessabbild der Komponente (kleinstmöglicher Wert: 0)
LEN	INPUT	INT	E, A, M, D, L, Konstante	Anzahl der zu lesenden Bytes
STATUS	OUTPUT	DWORD	E, A, M, D, L	enthält die Fehlerinformation der SFC 81 UBLKMOV in der Form DW#16#40xxxx00, falls *ERROR* = TRUE
ERROR	OUTPUT	BOOL	E, A, M, D, L	Fehleranzeige: *ERROR* = TRUE, falls beim Aufruf der SFC 81 UBLKMOV ein Fehler auftritt.
INPUTS	IN_OUT	ANY	E, A, M, D	Zielbereich für die gelesenen Daten: ■ Falls der Zielbereich kleiner ist als *LEN*, werden so viele Bytes übertragen, wie INPUTS aufnehmen kann. *ERROR* erhält den Wert FALSE. ■ Falls der Zielbereich größer ist als *LEN*, werden die ersten *LEN* Bytes des Zielbereichs beschrieben. *ERROR* erhält den Wert FALSE.

Fehlerinformationen Siehe SFC 81 - UBLKMOV - Variable ununterbrechbar kopieren. ↳ *935*

17.3.4 FB 23 - SETIO_PART - PROFIBUS/PROFINET Teil-Ausgänge schreiben

Beschreibung Mit dem FB 23 SETIO_PART übertragen Sie die Daten aus dem durch *OUTPUTS* aufgespannten Quellbereich konsistent in einen Teil des zu einem PROFIBUS-DP-Slave / PROFINET IO-Device gehörenden Prozessabbildbereichs. Der FB 23 ruft dabei die SFC 81 UBLKMOV auf.

Sie müssen dem OB, in dem der FB 23 SETIO_PART aufgerufen wird, ein Teilprozessabbild der Ausgänge zuordnen. Sie müssen weiterhin vor Aufruf des FB 23 den zugehörigen PROFIBUS-DP-Slave bzw. das zugehörige PROFINET IO-Device in dieses Teilprozessabbild der Ausgänge aufnehmen. Falls Ihre CPU keine Teilprozessabbilder kennt oder Sie den FB 23 im OB 1 aufrufen wollen, müssen Sie vor Aufruf des FB 23 den zugehörigen PROFIBUS-DP-Slave bzw. das zugehörige PROFINET IO-Device in das Prozessabbild der Ausgänge aufnehmen. Über die Parameter OFFSET und LEN legen Sie die zu beschreibende Teilmenge des Prozessabbildbereichs der über ID adressierten Komponente fest. Falls bei der Datenübertragung kein Fehler auftrat, erhält ERROR den Wert FALSE. Falls bei der Datenübertragung ein Fehler auftrat, erhält ERROR den Wert TRUE, und STATUS erhält die Fehlerinformation der SFC 81 UBLKMOV. Falls der Quellbereich (Parameter OUTPUTS) kleiner ist als LEN, werden so viele Bytes übertragen, wie OUTPUTS enthält. ERROR erhält den Wert FALSE. Falls der Quellbereich größer ist als LEN, werden die ersten LEN Bytes aus OUTPUTS übertragen. ERROR erhält den Wert FALSE.

Der FB 23 SETIO_PART überprüft im Prozessabbild der Ausgänge keine Grenzen zwischen Daten, die zu verschiedenen PROFIBUS-DP- bzw. PROFINET IO- Komponenten gehören. Sie müssen daher selbst darauf achten, dass der über OFFSET und LEN festgelegte Prozessabbildbereich zu einer Komponente gehört. Das Komponenten übergreifende Schreiben kann für zukünftige Systeme nämlich nicht garantiert werden und gefährdet die Übertragbarkeit auf Systeme anderer Hersteller.

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
ID	INPUT	DWORD	E, A, M, D, L, Konstante	■ Low word: logische Adresse der DP-Slave- / PROFINET-IO-Komponente (Baugruppe bzw. Modul) ■ High word: irrelevant
OFFSET	INPUT	INT	E, A, M, D, L, Konstante	Nummer des ersten zu beschreibenden Bytes im Prozessabbild der Komponente (kleinstmöglicher Wert: 0)
LEN	INPUT	INT	E, A, M, D, L, Konstante	Anzahl der zu schreibenden Bytes
STATUS	OUTPUT	DWORD	E, A, M, D	enthält die Fehlerinformation der SFC 81 UBLKMOV in der Form DW#16#40xxxx00, falls *ERROR* = TRUE

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
ERROR	OUTPUT	BOOL	E, A, M, D	Fehleranzeige: *ERROR* = TRUE, falls beim Aufruf der SFC 81 UBLKMOV ein Fehler auftritt.
OUTPUTS	IN_OUT	ANY	E, A, M, D, L	Quellbereich für die zu schreibenden Daten: ■ Falls der Quellbereich kleiner ist als *LEN*, werden so viele Bytes übertragen, wie *OUTPUTS* enthält. *ERROR* erhält den Wert FALSE. ■ Falls der Quellbereich größer ist als *LEN*, werden die ersten *LEN* Bytes aus *OUTPUTS* übertragen. *ERROR* erhält den Wert FALSE.

Fehlerinformationen Siehe SFC 81 - UBLKMOV - Variable ununterbrechbar kopieren. ↳ *935*

17.4 S5-Konvertierung - *"S5 Converting"*

17.4.1 FC 112 - Sine(x) - Sinus

Beschreibung

Die Funktion FC 112 erwartet den Eingangswert im AKKU 1 als Gleitpunktzahl.

1. Der Eingangswert muss im Bereich von Null
   (REAL = +0.0000000e+00) ... 2 x π (REAL = +0.6283185e+01) liegen.
2. Das Ergebnis legt die Funktion im AKKU 1 ebenfalls als Gleitpunktzahl ab.
3. Der Eingangswert DWORD = DW#16#0000 0000 wird wie der Gleitpunktwert Null (REAL = +0.0000000e+00 entsprechend DWORD = DW#16#8000 0000) behandelt.
   ⇨ Bei ordnungsgemäß ausgeführter Berechnung ist das Verknüpfungsergebnis *ENO* nach dem Aufruf der Funktion gleich FALSE.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Ein Fehler ist aufgetreten

Fehlerinformation Im Fehlerfall, wenn der Eingangswert nicht im Bereich von 0 ... 2 x π liegt, setzt die Funktion das Verknüpfungsergebnis *ENO* auf Status gleich TRUE. Dann bleibt der Inhalt von AKKU 1 unverändert. Die Belegung der restlichen Register und die der Hilfsmerker werden nicht verändert.

Diese FC wird nur dazu verwendet, den FB 101 eines bestehenden S5-Programms in eine FC eines S7-Programms umzuwandeln.

17.4.2 FC 113 - Cosine(x) - Cosinus

Beschreibung

Die Funktion FC 113 erwartet den Eingangswert im AKKU 1 als Gleitpunktzahl.

1. Der Eingangswert muss im Bereich von Null
   (REAL = +0.0000000e+00) ... 2 x π (REAL = +0.6283185e+01) liegen.
2. Das Ergebnis legt die Funktion im AKKU 1 ebenfalls als Gleitpunktzahl ab.
3. Der Eingangswert DWORD = DW#16#0000 0000 wird wie der Gleitpunktwert Null (REAL = +0.0000000e+00 entsprechend DWORD = DW#16#8000 0000) behandelt.
   ⇨ Bei ordnungsgemäß ausgeführter Berechnung ist das Verknüpfungsergebnis *ENO* nach dem Aufruf der Funktion gleich FALSE.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Ein Fehler ist aufgetreten

Fehlerinformation

Im Fehlerfall, wenn der Eingangswert nicht im Bereich von 0 ... 2 x π liegt, setzt die Funktion das Verknüpfungsergebnis *ENO* auf Status gleich TRUE. Dann bleibt der Inhalt von AKKU 1 unverändert. Die Belegung der restlichen Register und die der Hilfsmerker werden nicht verändert.

Diese FC wird nur dazu verwendet, den FB 102 eines bestehenden S5-Programms in eine FC eines S7-Programms umzuwandeln.

17.4.3 FC 114 - Tangent(x) - Tangens

Beschreibung

Die Funktion FC 114 erwartet den Eingangswert im AKKU 1 als Gleitpunktzahl.

1. Der Eingangswert muss im Bereich von Null
   (REAL = +0.0000000e+00) ... 2 x π (REAL = +0.6283185e+01) liegen.
2. Das Ergebnis legt die Funktion im AKKU 1 ebenfalls als Gleitpunktzahl ab.

3. Der Eingangswert DWORD = DW#16#0000 0000 wird wie der Gleitpunktwert Null (REAL = +0.0000000e+00 entsprechend DWORD = DW#16#8000 0000) behandelt.

   ⇨ Bei ordnungsgemäß ausgeführter Berechnung ist das Verknüpfungsergebnis *ENO* nach dem Aufruf der Funktion gleich FALSE.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Ein Fehler ist aufgetreten

Fehlerinformation

Im Fehlerfall setzt die Funktion das Verknüpfungsergebnis *ENO* auf Status TRUE. Dann bleibt der Inhalt von AKKU 1 unverändert. Es liegt einer der folgenden Fehler vor:

- Der Eingangswert liegt nicht im Bereich 0 ... 2 x π.
- Während der Berechnung der Funktion ist ein Zahlenbereichsüberlauf aufgetreten.
- Der Eingangswert beträgt π/2 oder 3 x π/2. Dann ist der Funktionswert unendlich.

Die Belegung der restlichen Register und die der Hilfsmerker werden nicht verändert.

Diese FC wird nur dazu verwendet, den FB 103 eines bestehenden S5-Programms in eine FC eines S7-Programms umzuwandeln.

17.4.4 FC 115 - Cotangent(x) - Cotangens

Beschreibung

Die Funktion FC 115 erwartet den Eingangswert im AKKU 1 als Gleitpunktzahl.

1. Der Eingangswert muss im Bereich von

   REAL = +0.2938734e-34 ...

   (fast) 2 x π (REAL = +0.6283184e+01) liegen.

2. Das Ergebnis legt die Funktion im AKKU 1 ebenfalls als Gleitpunktzahl ab.

3. Der Eingangswert DWORD = DW#16#0000 0000 wird wie der Gleitpunktwert Null (REAL = +0.0000000e+00 entsprechend DWORD = DW#16#8000 0000) behandelt.

   ⇨ Bei ordnungsgemäß ausgeführter Berechnung ist das Verknüpfungsergebnis *ENO* nach dem Aufruf der Funktion gleich FALSE.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Ein Fehler ist aufgetreten

Fehlerinformation

Im Fehlerfall setzt die Funktion das Verknüpfungsergebnis *ENO* auf Status TRUE. Dann bleibt der Inhalt von AKKU 1 unverändert. Es liegt einer der folgenden Fehler vor:

- Der Eingangswert liegt nicht im Bereich von
  REAL = +0.2938734e-34 und REAL = +0.6283184e+01
- Während der Berechnung der Funktion ist ein Zahlenbereichsüberlauf aufgetreten.
- Der Eingangswert beträgt 0 oder π oder 2 x π. Dann ist der Funktionswert unendlich.

Die Belegung der restlichen Register und die der Hilfsmerker werden nicht verändert.

Diese FC wird nur dazu verwendet, den FB 104 eines bestehenden S5-Programms in eine FC eines S7-Programms umzuwandeln.

17.4.5 FC 116 - Arc Sine(x) - Arcussinus

Beschreibung

Die Funktion FC 116 erwartet den Eingangswert im AKKU 1 als Gleitpunktzahl.

1. Der Eingangswert muss im Bereich von
   -1 (REAL = -0.1000000e+01) ... +1 (REAL = +0.1000000e+01) liegen.
2. Das Ergebnis legt die Funktion im AKKU 1 ebenfalls als Gleitpunktzahl ab.
3. Der Eingangswert DWORD = DW#16#0000 0000 wird wie der Gleitpunktwert Null (REAL = +0.0000000e+00 entsprechend DWORD = DW#16#8000 0000) behandelt.

   ⇨ Bei ordnungsgemäß ausgeführter Berechnung ist das Verknüpfungsergebnis *ENO* nach dem Aufruf der Funktion gleich FALSE.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Ein Fehler ist aufgetreten

Fehlerinformation

Im Fehlerfall, wenn der Eingangswert nicht im Bereich von -1 ... +1 liegt, setzt die Funktion das Verknüpfungsergebnis *ENO* auf Status gleich TRUE. Dann bleibt der Inhalt von AKKU 1 unverändert. Die Belegung der restlichen Register und die der Hilfsmerker werden nicht verändert.

Diese FC wird nur dazu verwendet, den FB 105 eines bestehenden S5-Programms in eine FC eines S7-Programms umzuwandeln.

17.4.6 FC 117 - Arc Cosine(x) - Arcuscosinus

Beschreibung

Die Funktion FC 117 erwartet den Eingangswert im AKKU 1 als Gleitpunktzahl.

1. Der Eingangswert muss im Bereich von

   -1 (REAL = -0.1000000e+01) ... +1 (REAL = +0.1000000e+01) liegen.
2. Das Ergebnis legt die Funktion im AKKU 1 ebenfalls als Gleitpunktzahl ab.
3. Der Eingangswert DWORD = DW#16#0000 0000 wird wie der Gleitpunktwert Null (REAL = +0.0000000e+00 entsprechend DWORD = DW#16#8000 0000) behandelt.

   ⇨ Bei ordnungsgemäß ausgeführter Berechnung ist das Verknüpfungsergebnis *ENO* nach dem Aufruf der Funktion gleich FALSE.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Ein Fehler ist aufgetreten

Fehlerinformation

Im Fehlerfall, wenn der Eingangswert nicht im Bereich von -1 ... +1 liegt, setzt die Funktion das Verknüpfungsergebnis *ENO* auf Status gleich TRUE. Dann bleibt der Inhalt von AKKU 1 unverändert. Die Belegung der restlichen Register und die der Hilfsmerker werden nicht verändert.

Diese FC wird nur dazu verwendet, den FB 106 eines bestehenden S5-Programms in eine FC eines S7-Programms umzuwandeln.

17.4.7 FC 118 - Arc Tangent(x) - Arcustangens

Beschreibung

Die Funktion FC 118 erwartet den Eingangswert im AKKU 1 als Gleitpunktzahl.

1. Der Eingangswert muss im Bereich von

   -1 (REAL = -0.1000000e+01) ... +1 (REAL = +0.1000000e+01) liegen.

2. Das Ergebnis legt die Funktion im AKKU 1 ebenfalls als Gleitpunktzahl ab.
3. Der Eingangswert DWORD = DW#16#0000 0000 wird wie der Gleitpunktwert Null (REAL = +0.0000000e+00 entsprechend DWORD = DW#16#8000 0000) behandelt.
4. Bei einem Eingangswert größer als REAL = +0.1209486e+07 wird als Ergebnis +π/2 ausgegeben.

   Bei einem Eingangswert kleiner als REAL = -0.5773456e+07 wird als Ergebnis -π/2 ausgegeben.

   ⇨ Das Verknüpfungsergebnis *ENO* wird auf Signalzustand FALSE gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Ein Fehler ist aufgetreten

Fehlerinformation

Im Fehlerfall, wenn der Eingangswert nicht im Bereich von -1 ... +1 liegt, setzt die Funktion das Verknüpfungsergebnis *ENO* auf Status gleich TRUE. Dann bleibt der Inhalt von AKKU 1 unverändert. Die Belegung der restlichen Register und die der Hilfsmerker werden nicht verändert.

Diese FC wird nur dazu verwendet, den FB 107 eines bestehenden S5-Programms in eine FC eines S7-Programms umzuwandeln.

17.4.8 FC 119 - Arc Cotangent(x) - Arcuscotangens

Beschreibung

Die Funktion FC 119 erwartet den Eingangswert im AKKU 1 als Gleitpunktzahl.

1. Der Eingangswert muss im Bereich von

   -1 (REAL = -0.1000000e+01) ... +1 (REAL = +0.1000000e+01) liegen.
2. Das Ergebnis legt die Funktion im AKKU 1 ebenfalls als Gleitpunktzahl ab.
3. Der Eingangswert DWORD = DW#16#0000 0000 wird wie der Gleitpunktwert Null (REAL = +0.0000000e+00 entsprechend DWORD = DW#16#8000 0000) behandelt.
4. Bei einem Eingangswert größer als REAL = +1.209486e+07 wird als Ergebnis +π/2 ausgegeben.

   Bei einem Eingangswert kleiner als REAL = -0.5773456e+07 wird als Ergebnis -π/2 ausgegeben.

   ⇨ Das Verknüpfungsergebnis *ENO* wird auf Signalzustand FALSE gesetzt.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Ein Fehler ist aufgetreten

Fehlerinformation

Im Fehlerfall, wenn der Eingangswert nicht im Bereich von -1 ... +1 liegt, setzt die Funktion das Verknüpfungsergebnis *ENO* auf Status gleich TRUE. Dann bleibt der Inhalt von AKKU 1 unverändert. Die Belegung der restlichen Register und die der Hilfsmerker werden nicht verändert.

Diese FC wird nur dazu verwendet, den FB 108 eines bestehenden S5-Programms in eine FC eines S7-Programms umzuwandeln.

17.4.9 FC 120 - Naperian Logarithm ln(x) - Natürlicher Logarithmus

Beschreibung

Die Funktion FC 120 erwartet den Eingangswert im AKKU 1 als Gleitpunktzahl.

1. Der Eingangswert muss im Bereich von
   -1 (REAL = -0.1000000e+01) bis + 1 (REAL = +0.1000000e+01) liegen.
2. Das Ergebnis legt die Funktion im AKKU 1 ebenfalls als Gleitpunktzahl ab.
3. Bei ordnungsgemäß ausgeführter Berechnung ist das Verknüpfungsergebnis nach dem Aufruf der Funktion FALSE.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Ein Fehler ist aufgetreten

Fehlerinformation

Im Fehlerfall, wenn der Eingangswert kleiner oder gleich Null ist, setzt die Funktion das Verknüpfungsergebnis *ENO* von Status gleich TRUE. Dann bleibt der Inhalt von AKKU 1 unverändert. Die Belegung der restlichen Register und die der Hilfsmerker wird nicht verändert.

Diese FC wird nur dazu verwendet, den FB 109 eines bestehenden S5-Programms in eine FC eines S7-Programms umzuwandeln.

17.4.10 FC 121 - Decimal Logarithm lg(x) - Dezimaler Logarithmus

Beschreibung

Die Funktion FC 121 erwartet den Eingangswert im AKKU 1 als Bit-Gleitpunktzahl.

1. Der Eingangswert muss im Bereich von
   -1 (REAL = -0.1000000e+01) bis + 1 (REAL = +0.1000000e+01) liegen.
2. Das Ergebnis legt die Funktion im AKKU 1 ebenfalls als Gleitpunktzahl ab.
3. Bei ordnungsgemäß ausgeführter Berechnung ist das Verknüpfungsergebnis nach dem Aufruf der Funktion FALSE.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Ein Fehler ist aufgetreten

Fehlerinformation

Im Fehlerfall, wenn der Eingangswert kleiner oder gleich Null ist, setzt die Funktion das Verknüpfungsergebnis *ENO* auf Status gleich TRUE. Dann bleibt der Inhalt von AKKU 1 unverändert. Die Belegung der restlichen Register und die der Hilfsmerker wird nicht verändert.

Diese FC wird nur dazu verwendet, den FB 110 eines bestehenden S5-Programms in eine FC eines S7-Programms umzuwandeln.

17.4.11 FC 122 - Gen. Logarithm to Base b - Allgemeiner Logarithmus log (x) zur Basis b

Beschreibung

Die Funktion FC 122 erwartet den Eingangswert für die Basis (b) im AKKU 2 und den Eingangswert für den Numerus (x) im AKKU 1 als Gleitpunktzahlen.

1. Beide Eingangswerte müssen größer als Null sein; zusätzlich darf die Basis nicht den Wert +1 haben.
2. Bei ordnungsgemäß ausgeführter Berechnung steht im AKKU 1 das Ergebnis als Gleitpunktzahl , im AKKU 2 der vorherige Inhalt von AKKU 3 und im AKKU 3 der vorherige Inhalt des AKKU 4. Der Inhalt von AKKU 4 wird nicht verändert. Die Belegung der Schmiermerker wird nicht verändert.
3. Bei einer Berechnung ohne Fehler ist das Verknüpfungsergebnis *ENO* nach dem Aufruf der Funktion FALSE.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Ein Fehler ist aufgetreten

Fehlerinformation

Im Falle eines Fehlers, wenn einer der Eingangswerte kleiner oder gleich Null ist oder wenn die Basis den Wert +1 hat, setzt die Funktion das Verknüpfungsergebnis *ENO* auf den Signalzustand TRUE. Dann bleibt der Inhalt der AKKUs unverändert.

Diese FC wird nur dazu verwendet, den FB 111 eines bestehenden S5-Programms in eine FC eines S7-Programms umzuwandeln.

17.4.12 FC 123 - E to Power n - E hoch n

Beschreibung

Die Funktion FC 123 erwartet den Eingangswert im AKKU 1 als Gleitpunktzahl.

1. Der Eingangswert DWORD = DW#16#0000 0000 wird wie der Gleitpunktwert Null (REAL = +0.0000000e+00 entsprechend DWORD = DW#16#8000 0000) behandelt.
2. Das Ergebnis legt die Funktion im AKKU 1 ebenfalls als Gleitpunktzahl ab.
3. Bei ordnungsgemäß ausgeführter Berechnung ist das Verknüpfungsergebnis *ENO* nach dem Aufruf der Funktion FALSE.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Ein Fehler ist aufgetreten

Fehlerinformation

Im Fehlerfall, wenn der Eingangswert nicht im Bereich von REAL = - 0.8802962e+02 ... REAL = +0.8802966e+02 liegt (dann würde der Funktionswert außerhalb des Zahlenbereichs liegen), setzt die Funktion das Verknüpfungsergebnis *ENO* auf den Signalzustand TRUE. Dann bleibt der Inhalt von AKKU 1 unverändert. Die Belegung der Hilfsmerker wird nicht verändert.

Diese FC wird nur dazu verwendet, den FB 112 eines bestehenden S5-Programms in eine FC eines S7-Programms umzuwandeln.

17.4.13 FC 124 - 10 to Power n - 10 hoch n

Beschreibung

Die Funktion FC 124 erwartet den Eingangswert im AKKU 1 als Gleitpunktzahl.

1. Der Eingangswert DWORD = DW#16#0000 0000 wird wie der Gleitpunktwert Null (REAL = +0.0000000e+00 entsprechend DWORD = DW#16#8000 0000) behandelt.
2. Das Ergebnis legt die Funktion im AKKU 1 ebenfalls als Gleitpunktzahl ab.
3. Bei ordnungsgemäß ausgeführter Berechnung ist das Verknüpfungsergebnis *ENO* nach dem Aufruf der Funktion FALSE.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Ein Fehler ist aufgetreten

Fehlerinformation

Im Fehlerfall, wenn der Eingangswert nicht im Bereich von REAL = -0.3823079e+02 ... REAL = + 0.3823080e+02 liegt (dann würde der Funktionswert außerhalb des Zahlenbereichs liegen), setzt die Funktion das Verknüpfungsergebnis auf den Signalzustand TRUE. Dann bleibt der Inhalt von AKKU 1 unverändert. Die Belegung der Hilfsmerker wird nicht verändert.

Diese FC wird nur dazu verwendet, den FB 113 eines bestehenden S5-Programms in eine FC eines S7-Programms umzuwandeln.

17.4.14 FC 125 - ACCU 2 to Power ACCU 1 - AKKU 2 hoch AKKU 1

Beschreibung

Die Funktion FC 125 erwartet den Eingangswert für die Basis im AKKU 2 und den Eingangswert für den Exponenten im AKKU 1, beide als Gleitpunktzahlen.

1. Der Eingangswert für die Basis muss positiv sein.

   Ein Eingangswert DWORD = DW#16#0000 0000 wird wie der Gleitpunktwert Null (REAL = +0.0000000e+00 entsprechend DWORD = DW#16#8000 0000) behandelt.

   Für Null hoch Null wird als Ergebnis Null ausgegeben.
2. Das Ergebnis der Berechnung steht als Gleitpunktzahl im AKKU 1.
3. Bei ordnungsgemäß ausgeführter Berechnung ist das Verknüpfungsergebnis *ENO* nach dem Aufruf der Funktion FALSE.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	INPUT	BOOL	E, A, M, D, L	■ Freigabeeingang – TRUE: aktiviert die Funktion – FALSE: deaktiviert die Funktion
ENO	OUTPUT	BOOL	E, A, M, D, L	■ Status – TRUE: Ein Fehler ist aufgetreten

Fehlerinformation

Im Fehlerfall wird *ENO* TRUE wenn:

- der Eingangswert für die Basis kleiner als Null ist
- während der Berechnung der Funktion ein Zahlenbereichsüberlauf aufgetreten ist

Im Fehlerfall bleibt der Inhalt von AKKU 1 und 2 unverändert.

Diese FC wird nur dazu verwendet, den FB 114 eines bestehenden S5-Programms in eine FC eines S7-Programms umzuwandeln.

17.5 PID-Steuerung - *"PID Control"*

17.5.1 FB 41 - CONT_C - Kontinuierliches Regeln

Beschreibung

Der FB 41 CONT_C dient zum Regeln von technischen Prozessen mit kontinuierlichen Ein- und Ausgangsgrößen. Über die Parametrierung können Sie Teilfunktionen des PID Reglers zu- oder abschalten und damit diesen an die Regelstrecke anpassen.

Bitte beachten Sie, dass der Regelungsbaustein zyklisch in regelmäßigen Abständen aufzurufen ist, ansonsten kann dies zu Fehlberechnungen führen. Rufen Sie hierzu den Regelungsbaustein in einem Weckalarm-OB (OB 30 ... OB 38) auf und geben Sie am Eingang CYCLE das Zeitraster des Weckalarm-OBs an.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
COM_RST	INPUT	BOOL	COMPLETE RESTART / Neustart ■ Der Baustein hat eine Neustartroutine, die bearbeitet wird, wenn der Eingang *COM_RST* gesetzt ist. ■ Default: FALSE
MAN_ON	INPUT	BOOL	MANUAL VALUE ON / Handbetrieb einschalten ■ Ist der Eingang *MAN_ON* gesetzt, ist der Regelkreis unterbrochen. Als Stellwert wird ein Handwert vorgegeben. ■ Default: TRUE

Parameter	Deklaration	Datentyp	Beschreibung
PVPER_ON	INPUT	BOOL	PROCESS VARIABLE PERIPHERY ON / Istwert Peripherie einschalten ■ Soll der Istwert von der Peripherie eingelesen werden, so muss der Eingang *PV_PER* mit der Peripherie verschaltet werden und der Eingang *PVPER_ON* gesetzt werden. ■ Default: FALSE
P_SEL	INPUT	BOOL	PROPORTIONAL ACTION ON / P-Anteil einschalten ■ Im PID-Algorithmus lassen sich die PID-Anteile einzeln zu- und abschalten. Der P-Anteil ist eingeschaltet, wenn der Eingang *P_SEL* gesetzt ist. ■ Default: TRUE
I_SEL	INPUT	BOOL	INTEGRAL ACTION ON / I-Anteil einschalten ■ Im PID-Algorithmus lassen sich die PID-Anteile einzeln zu- und abschalten. Der I-Anteil ist eingeschaltet, wenn der Eingang *I_SEL* gesetzt ist. ■ Default: TRUE
INT_HOLD	INPUT	BOOL	INTEGRAL ACTION HOLD / I-Anteil einfrieren ■ Der Ausgang des Integrierers kann eingefroren werden. Hierzu muss der Eingang *INT_HOLD* gesetzt werden. ■ Default: FALSE
I_ITL_ON	INPUT	BOOL	INITIALIZATION OF THE INTEGRAL ACTION / I-Anteil setzen ■ Der Ausgang des Integrierers kann auf den Eingang *I_ITLVAL* gesetzt werden. Hierzu muss der Eingang *I_ITL_ON* gesetzt werden. ■ Default: FALSE
D_SEL	INPUT	BOOL	DERIVATIVE ACTION ON / D-Anteil einschalten ■ Im PID-Algorithmus lassen sich die PID-Anteile einzeln zu- und abschalten. Der D-Anteil ist eingeschaltet, wenn der Eingang *D_SEL* gesetzt ist. ■ Default: FALSE
CYCLE	INPUT	TIME	SAMPLE TIME / Abtastzeit ■ Die Zeit zwischen den Bausteinaufrufen muss konstant sein. Der Eingang *CYCLE* gibt die Zeit zwischen den Bausteinaufrufen an. ■ Default: T#1s ■ Wertebereich: ≥ 1ms
SP_INT	INPUT	REAL	INTERNAL SETPOINT / Interner Sollwert ■ Der Eingang *SP_INT* dient zur Vorgabe eines Sollwertes. ■ Default: 0.0 ■ Wertebereich: -100.0...100. 0 (%) oder phys. Größe[1]

Parameter	Deklaration	Datentyp	Beschreibung
PV_IN	INPUT	REAL	PROCESS VARIABLE IN / Istwert Eingang ■ Am Eingang *PV_IN* kann ein Inbetriebsetzungswert parametriert oder ein externer Istwert im Gleitpunktformat verschaltet werden. ■ Default: 0.0 ■ Wertebereich: -100.0...100. 0 (%) oder phys. Größe[1]
PV_PER	INPUT	WORD	PROCESS VARIABLE PERIPHERY / Istwert Peripherie ■ Der Istwert in Peripherieformat wird am Eingang *PV_PER* mit dem Regler verschaltet. ■ Default: W#16#0000
MAN	INPUT	REAL	MANUAL VALUE / Handwert ■ Der Eingang *MAN* dient zur Vorgabe eines Handwertes mittels Bedien-Beobachterfunktion. ■ Default: 0.0 ■ Wertebereich: -100.0...100. 0 (%) oder phys. Größe[2]
GAIN	INPUT	REAL	PROPORTIONAL GAIN / Proportionalbeiwert ■ Der Eingang *GAIN* gibt die Reglerverstärkung an. ■ Default: 2.0 ■ Wertebereich: ≥ CYCLE
TI	INPUT	TIME	RESET TIME / Integrationszeit ■ Der Eingang *TI* bestimmt das Zeitverhalten des Integrierers. ■ Default: T#20s ■ Wertebereich: ≥ CYCLE
TD	INPUT	TIME	DERIVATIVE TIME / Differenzierzeit ■ Der Eingang *TD* bestimmt das Zeitverhalten des Differenzierers. ■ Default: T#10s ■ Wertebereich: ≥ CYCLE
TM_LAG	INPUT	TIME	TIME LAG OF THE DERIVATIVE ACTION / Verzögerungszeit des D-Anteils ■ Der Algorithmus des D-Anteils beinhaltet eine Verzögerung, die am Eingang *TM_LAG* parametriert werden kann. ■ Default: T#2s ■ Wertebereich: ≥ CYCLE/2
DEADB_W	INPUT	REAL	DEAD BAND WIDTH / Totzonenbreite ■ Die Regeldifferenz wird über eine Totzone geführt. Der Eingang *DEADB_W* bestimmt die Größe der Totzone. ■ Default: 0.0 ■ Wertebereich: ≥ 0.0 (%) oder phys. Größe[1]

Parameter	Deklaration	Datentyp	Beschreibung
LMN_HLM	INPUT	REAL	MANIPULATED VALUE HIGH LIMIT / Stellwert obere Begrenzung ■ Der Stellwert wird immer auf eine obere und untere Grenze begrenzt. Der Eingang *LMN_HLM* gibt die obere Begrenzung an. ■ Default: 100.0 ■ Wertebereich: *LMN_LLM* ...100.0 (%) oder phys. Größe[2]
LMN_LLM	INPUT	REAL	MANIPULATED VALUE LOW LIMIT / Stellwert untere Begrenzung ■ Der Stellwert wird immer auf eine obere und untere Grenze begrenzt. Der Eingang *LMN_LLM* gibt die untere Begrenzung an. ■ Default: 0.0 ■ Wertebereich: -100.0... *LMN_HLM* (%) oder phys. Größe[2]
PV_FAC	INPUT	REAL	PROCESS VARIABLE FACTOR / Istwertfaktor ■ Der Eingang *PV_FAC* wird mit dem Istwert multipliziert. Der Eingang dient zur Anpassung des Istwertbereiches. ■ Default: 1.0
PV_OFF	INPUT	REAL	PROCESS VARIABLE OFFSET / Istwertoffset ■ Der Eingang *PV_OFF* wird mit dem Istwert addiert. Der Eingang dient zur Anpassung des Istwertbereiches. ■ Default: 0.0
LMN_FAC	INPUT	REAL	MANIPULATED VALUE FACTOR / Stellwertfaktor ■ Der Eingang *LMN_FAC* wird mit dem Stellwert multipliziert. Der Eingang dient zur Anpassung des Stellwertbereiches. ■ Default: 1.0
LMN_OFF	INPUT	REAL	MANIPULATED VALUE OFFSET / Stellwertoffset ■ Der Eingang *LMN_OFF* wird mit dem Stellwert addiert. Der Eingang dient zur Anpassung des Stellwertbereiches. ■ Default: 0.0
I_ITLVAL	INPUT	REAL	INITIALIZATION VALUE OF THE INTEGRAL ACTION / Initialisierungswert für I-Anteil ■ Der Ausgang des Integrierers kann am Eingang *I_ITL_ON* gesetzt werden. Am Eingang *I_ITLVAL* steht der Initialisierungwert. ■ Default: 0.0 ■ Wertebereich: -100.0...100. 0 (%) oder phys. Größe[2]

Parameter	Deklaration	Datentyp	Beschreibung
DISV	INPUT	REAL	DISTURBANCE VARIABLE / Störgröße ■ Für eine Störgrößenaufschaltung wird die Störgröße am Eingang *DISV* verschaltet. ■ Default: 0.0 ■ Wertebereich: -100.0...100. 0 (%) oder phys. Größe[2]
LMN	OUTPUT	REAL	MANIPULATED VALUE / Stellwert ■ Am Ausgang *LMN* wird der effektiv wirkende Stellwert in Gleitpunktformat ausgegeben. ■ Default: 0.0
LMN_PER	OUTPUT	WORD	MANIPULATED VALUE PERIPHERY / Stellwert Peripherie ■ Der Stellwert in Peripherieformat wird am Ausgang *LMN_PER* mit dem Regler verschaltet. ■ Default: W#16#0000
QLMN_HLM	OUTPUT	BOOL	HIGH LIMIT OF MANIPULATED VALUE REACHED / Obere Begrenzung des Stellwertes angesprochen ■ Der Stellwert wird immer auf eine obere und untere Grenze begrenzt. Der Ausgang *QLMN_HLM* meldet die Überschreitung der oberen Begrenzung. ■ Default: FALSE
QLMN_LLM	OUTPUT	BOOL	LOW LIMIT OF MANIPULATED VALUE REACHED / Untere Begrenzung des Stellwertes angesprochen ■ Der Stellwert wird immer auf eine obere und untere Grenze begrenzt. Der Ausgang *QLMN_LLM* meldet die Überschreitung der unteren Begrenzung. ■ Default: FALSE
LMN_P	OUTPUT	REAL	PROPORTIONALITY COMPONENT / P-Anteil ■ Der Ausgang *LMN_P* beinhaltet den Proportionalanteil der Stellgröße. ■ Default: 0.0
LMN_I	OUTPUT	REAL	INTEGRAL COMPONENT / I-Anteil ■ Der Ausgang *LMN_I* beinhaltet den Integralanteil der Stellgröße. ■ Default: 0.0
LMN_D	OUTPUT	REAL	DERIVATIVE COMPONENT / D-Anteil ■ Der Ausgang *LMN_D* beinhaltet den Differentialanteil der Stellgröße. ■ Default: 0.0
PV	OUTPUT	REAL	PROCESS VARIABLE / Istwert ■ Am Ausgang *PV* wird der effektiv wirkende Istwert ausgegeben. ■ Default: 0.0

Parameter	Deklaration	Datentyp	Beschreibung
ER	OUTPUT	REAL	ERROR SIGNAL / Regeldifferenz ■ Am Ausgang *ER* wird die effektiv wirkende Regeldifferenz ausgegeben. ■ Default: 0.0

1) Parameter im Soll-, Istwertzweig mit gleicher Einheit

2) Parameter im Stellwertzweig mit gleicher Einheit

Anwendung

Den Regler können Sie als PID-Festwertregler einzeln oder auch in mehrschleifigen Regelungen als Kaskaden-, Mischungs- oder Verhältnisregler einsetzen. Die Arbeitsweise basiert auf dem PID-Regelalgorithmus des Abtastreglers mit analogem Ausgangssignal, gegebenenfalls um eine Impulsformerstufe zur Bildung von pulsbreitenmodulierten Ausgangssignalen für Zwei- oder Dreipunktregelungen mit proportionalen Stellgliedern ergänzt.

Neben den Funktionen im Soll- und Istwertzweig realisiert der FB einen fertigen PID-Regler mit kontinuierlichem Stellgrößen-Ausgang und Beeinflussungsmöglichkeit des Stellwertes von Hand.

Sollwertzweig

Der Sollwert wird am Eingang *SP_INT* im Gleitpunktformat eingegeben.

Istwertzweig

Der Istwert kann im Peripherie- und im Gleitpunktformat eingelesen werden. Die Funktion CRP_IN wandelt den Peripheriewert *PV_PER* in ein Gleitpunktformat von -100 .... +100 % nach folgender Vorschrift um:

$$Ausgang\ von\ CPR_IN = PV_PER * \frac{100}{27648}$$

Die Funktion *PV_NORM* normiert den Ausgang von *CRP_IN* nach folgender Vorschrift:

$$Ausgang\ von\ PV_NORM = (Ausgang\ von\ CPR_IN) * PV_FAC + PV_OFF$$

PV_FAC ist mit 1 und *PV_OFF* ist mit 0 vorbelegt.

Regeldifferenzbildung

Die Differenz von Soll- und Istwert ergibt die Regeldifferenz. Zur Unterdrükkung einer kleinen Dauerschwingung aufgrund der Stellgrößen-Quantisierung (z.B. bei einer Puls-Breitenmodulation mit PULSEGEN) wird die Regeldifferenz über eine Totzone (DEAD-BAND) geleitet. Bei *DEADB_W* = 0 ist die Totzone ausgeschaltet.

PID-Algorithmus

Der PID-Algorithmus arbeitet im Stellungsalgorithmus. Der Proportional-, Integral (*INT*) und Differentialanteil (*DIF*) sind parallel geschaltet und einzeln zu- und abschaltbar. Damit lassen sich P-, PI-, PD- und PID-Regler parametrieren. Aber auch reine I-Regler sind möglich.

Handwertverarbeitung

Es kann zwischen Hand- und Automatikbetrieb umgeschaltet werden. Bei Handbetrieb wird die Stellgröße einem Handwert nachgeführt. Der Integrierer (*INT*) wird intern auf *LMN - LMN_P - DISV* und der Differenzierer (*DIF*) auf 0 gesetzt und intern abgeglichen. Das Umschalten in den Automatikbetrieb ist damit stoßfrei.

Stellwertverarbeitung

Der Stellwert wird mit der Funktion *LMNLIMIT* auf vorgebbare Werte begrenzt. Das Überschreiten der Grenzen durch die Eingangsgröße wird durch Meldebits angezeigt. Die Funktion *LMN_NORM* normiert den Ausgang von *LMNLIMIT* nach folgender Vorschrift:

$$LMN = (Ausgang\ von\ LMNLIMIT) * LMN_FAC + LMN_OFF$$

LMN_FAC ist mit 1 und *LMN_OFF* mit 0 vorbelegt.

Der Stellwert steht auch im Peripherieformat zur Verfügung. Die Funktion *CRP_OUT* wandelt den Gleitpunktwert *LMN* in einen Peripheriewert nach folgender Vorschrift um:

$$LMN_PER = LMN * \frac{27648}{100}$$

Störgrößenaufschaltung

Am Eingang *DISV* kann eine Störgröße additiv aufgeschaltet werden.

Betriebszustände

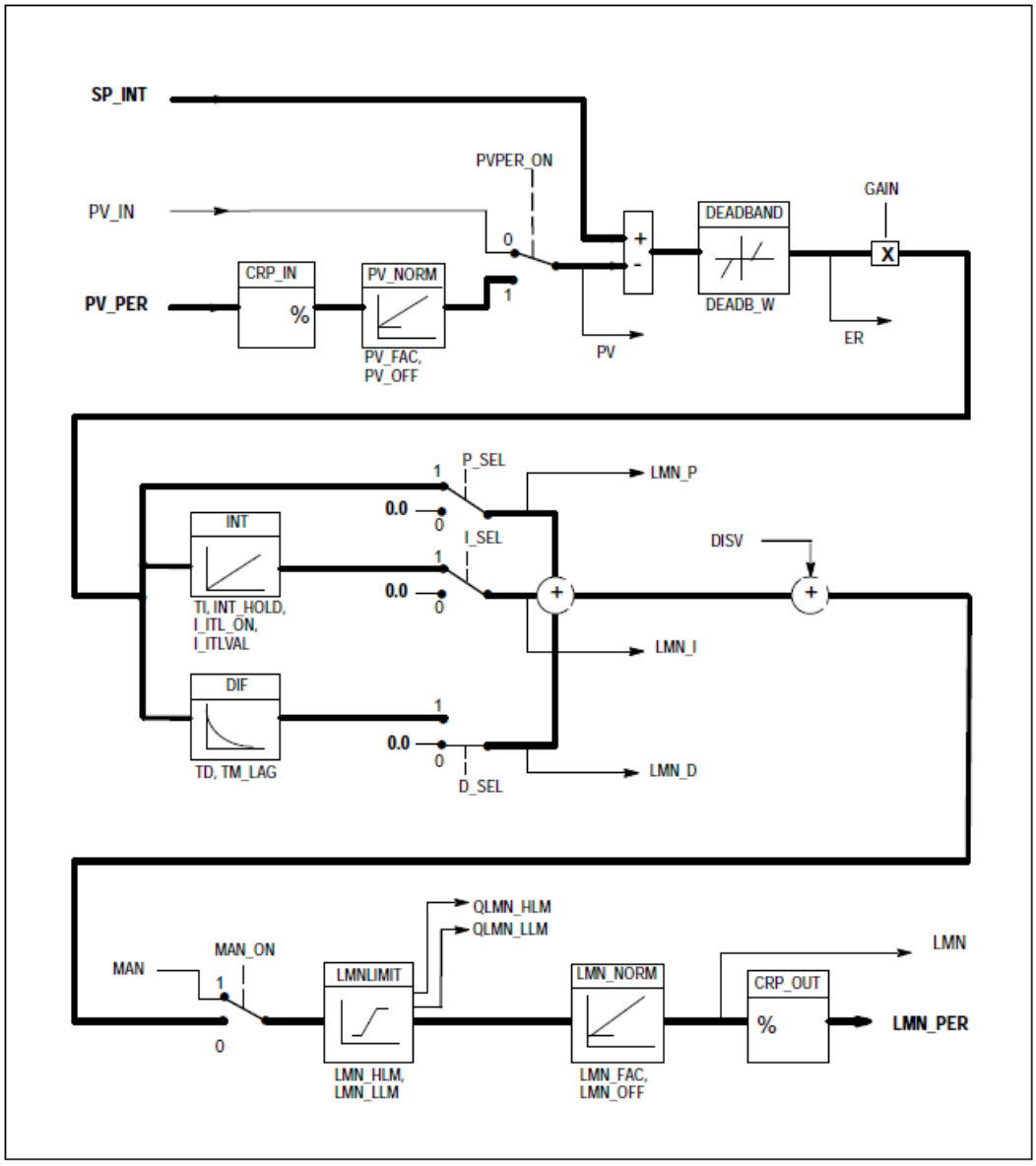
Neustart/Wiederanlauf

- Der FB 41 CONT_C verfügt über eine Neustartroutine, die durchlaufen wird, wenn der Eingangs-Parameter *COM_RST* = TRUE gesetzt ist.
- Der Integrierer wird beim Anlauf intern auf den Initialisierungswert *I_ITVAL* gesetzt. Beim Aufruf in einer Weckalarmebene arbeitet er von diesem Wert aus weiter.
- Alle anderen Ausgänge werden auf ihre Vorbelegungswerte gesetzt.

Fehlerinformationen

Der Baustein enthält keine Fehlerprüfung, deshalb wird keine Fehlerinformation ausgegeben.

Blockschaltbild



17.5.2 FB 42 - CONT_S - Schrittregeln

Beschreibung

Der FB 42 CONT_S dient zum Regeln von technischen Prozessen mit binären Stellwertausgangssignalen für integrierende Stellglieder. Über die Parametrierung lassen sich Teilfunktionen des PI-Schrittreglers zu- oder abschalten und damit an die Regelstrecke anpassen.

Bitte beachten Sie, dass der Regelungsbaustein zyklisch in regelmäßigen Abständen aufzurufen ist, ansonsten kann dies zu Fehlberechnungen führen. Rufen Sie hierzu den Regelungsbaustein in einem Weckalarm-OB (OB 30 ... OB 38) auf und geben Sie am Eingang CYCLE das Zeitraster des Weckalarm-OBs an.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
COM_RST	INPUT	BOOL	COMPLETE RESTART / Neustart ■ Der Baustein hat eine Neustartroutine, die bearbeitet wird, wenn der Eingang *COM_RST* gesetzt ist. ■ Default: FALSE
LMNR_HS	INPUT	BOOL	HIGH LIMIT SIGNAL OF REPEATED MANIPULATED VALUE / Oberes Anschlagsignal der Stellungsrückmeldung ■ Das Signal "Stellventil am oberen Anschlag" wird am Eingang *LMNR_HS* verschaltet. – *LMNR_HS* = TRUE heißt: Das Stellventil befindet sich am oberen Anschlag. ■ Default: FALSE
LMNR_LS	INPUT	BOOL	LOW LIMIT SIGNAL OF REPEATED MANIPULATED VALUE / Unteres Anschlagsignal der Stellungsrückmeldung ■ Das Signal "Stellventil am unteren Anschlag" wird am Eingang *LMNR_LS* verschaltet. – *LMNR_LS* = TRUE heißt: Das Stellventil befindet sich am unteren Anschlag. ■ Default: FALSE
LMNS_ON	INPUT	BOOL	MANIPULATED SIGNALS ON / Handbetrieb der Stellwertsignale einschalten ■ Am Eingang *LMNS_ON* wird die Stellwertsignalverarbeitung auf Hand geschaltet. ■ Default: FALSE
LMNUP	INPUT	BOOL	MANIPULATED SIGNALS UP / Stellwertsignal Hoch ■ Bei Handbetrieb der Stellwertsignale wird am Eingang *LMNUP* das Ausgangssignal *QLMNUP* bedient. ■ Default: FALSE
LMNDN	INPUT	BOOL	MANIPULATED SIGNALS DOWN / Stellwertsignal Tief ■ Bei Handbetrieb der Stellwertsignale wird am Eingang *LMNDN* das Ausgangssignal *QLMNDN* bedient. ■ Default: FALSE
PVPER_ON	INPUT	BOOL	PROCESS VARIABLE PERIPHERY ON / Istwert Peripherie einschalten ■ Soll der Istwert von der Peripherie eingelesen werden, so muss der Eingang *PV_PER* mit der Peripherie verschaltet werden und der Eingang *PVPER_ON* gesetzt werden. ■ Default: FALSE

Parameter	Deklaration	Datentyp	Beschreibung
CYCLE	INPUT	TIME	SAMPLE TIME / Abtastzeit ■ Die Zeit zwischen den Bausteinaufrufen muss konstant sein. Der Eingang *CYCLE* gibt die Zeit zwischen den Bausteinaufrufen an. ■ Default: T#1s ■ Wertebereich: ≥ 1ms
SP_INT	INPUT	REAL	INTERNAL SETPOINT / Interner Sollwert ■ Der Eingang *SP_INT* dient zur Vorgabe eines Sollwertes. ■ Default: 0.0 ■ Wertebereich: -100.0...100. 0 (%) oder phys. Größe[1]
PV_IN	INPUT	REAL	PROCESS VARIABLE IN / Istwert Eingang ■ Am Eingang *PV_IN* kann ein Inbetriebsetzungswert parametriert oder ein externer Istwert im Gleitpunktformat verschaltet werden. ■ Default: 0.0 ■ Wertebereich: -100.0...100. 0 (%) oder phys. Größe[1]
PV_PER	INPUT	WORD	PROCESS VARIABLE PERIPHERY / Istwert Peripherie ■ Der Istwert in Peripherieformat wird am Eingang *PV_PER* mit dem Regler verschaltet. ■ Default: W#16#0000
GAIN	INPUT	REAL	PROPORTIONAL GAIN / Proportionalbeiwert ■ Der Eingang *GAIN* gibt die Reglerverstärkung an. ■ Default: 2.0 ■ Wertebereich: ≥ *CYCLE*
TI	INPUT	TIME	RESET TIME / Integrationszeit ■ Der Eingang *TI* bestimmt das Zeitverhalten des Integrierers. ■ Default: T#20s ■ Wertebereich: ≥ *CYCLE*
DEADB_W	INPUT	REAL	DEAD BAND WIDTH / Totzonenbreite ■ Die Regeldifferenz wird über eine Totzone geführt. Der Eingang *DEADB_W* bestimmt die Größe der Totzone. ■ Default: 1.0 ■ Wertebereich: 0.0...100.0 (%) oder phys. Größe[1]
PV_FAC	INPUT	REAL	PROCESS VARIABLE FACTOR / Istwertfaktor ■ Der Eingang *PV_FAC* wird mit dem Istwert multipliziert. Der Eingang dient zur Anpassung des Istwertbereiches. ■ Default: 1.0

Parameter	Deklaration	Datentyp	Beschreibung
PV_OFF	INPUT	REAL	PROCESS VARIABLE OFFSET / Istwertoffset ■ Der Eingang *PV_OFF* wird mit dem Istwert addiert. Der Eingang dient zur Anpassung des Istwertbereiches. ■ Default: 0.0
PULSE_TM	INPUT	TIME	MINIMUM PULSE TIME / Mindestimpulsdauer ■ Am Parameter *PULSE_TM* kann eine minimale Impulslänge parametriert werden. ■ Default: 1.0 ■ Wertebereich: ≥ *CYCLE*
BREAK_TM	INPUT	TIME	MINIMUM BREAK TIME / Mindestpausendauer ■ Am Parameter *BREAK_TM* kann eine minimale Pausenlänge parametriert werden. ■ Default: 0.0 ■ Wertebereich: ≥ *CYCLE*
MTR_TM	INPUT	TIME	MOTOR MANIPULATED VALUE / Motorstellzeit ■ Am Parameter *MTR_TM* wird die Laufzeit des Stellventils vom Anschlag zu Anschlag eingetragen. ■ Default: 0.0 ■ Wertebereich: -100.0...100. 0 (%) oder phys. Größe[2] ■ Wertebereich: ≥ *CYCLE*
DISV	INPUT	REAL	DISTURBANCE VARIABLE / Störgröße ■ Für eine Störgrößenaufschaltung wird die Störgröße am Eingang *DISV* verschaltet. ■ Default: 0.0 ■ Wertebereich: -100.0...100. 0 (%) oder phys. Größe[2]
QLMNUP	OUTPUT	BOOL	MANIPULATED SIGNAL UP / Stellwertsignal Hoch ■ Ist der Ausgang *QLMNUP* gesetzt, soll das Stellventil geöffnet werden. ■ Default: FALSE
QLMNDN	OUTPUT	BOOL	MANIPULATED SIGNAL DOWN / Stellwertsignal Tief ■ Ist der Ausgang *QLMNDN* gesetzt, soll das Stellventil geschlossen werden. ■ Default: FALSE
PV	OUTPUT	REAL	PROCESS VARIABLE / Istwert ■ Am Ausgang *PV* wird der effektiv wirkende Istwert ausgegeben. ■ Default: 0.0
ER	OUTPUT	REAL	ERROR SIGNAL /Regeldifferenz ■ Am Ausgang *ER* wird die effektiv wirkende Regeldifferenz ausgegeben. ■ Default: 0.0

1) Parameter im Soll-, Istwertzweig mit gleicher Einheit

2) Parameter im Stellwertzweig mit gleicher Einheit

Anwendung

Der Regler kann als PI-Festwertregler einzeln oder in unterlagerten Regelkreisen bei Kaskaden-, Mischungs- oder Verhältnisregelungen eingesetzt werden, jedoch nicht als Führungsregler. Die Arbeitsweise basiert auf dem PI Regelalgorithmus des Abtastreglers und wird um die Funktionsglieder zur Erzeugung des binären Ausgangssignals aus dem analogen Stellsignal ergänzt.

Neben den Funktionen im Istwertzweig realisiert der FB einen fertigen PI Regler mit binärem Stellwertausgang und Beeinflussungsmöglichkeit des Stellwertes von Hand. Der Schrittregler arbeitet ohne Stellungsrückmeldung.

Sollwertzweig

Der Sollwert wird am Eingang *SP_INT* im Gleitpunktformat eingegeben.

Istwertzweig

Der Istwert kann im Peripherie- und im Gleitpunktformat eingelesen werden. Die Funktion *CRP_IN* wandelt den Peripheriewert *PV_PER* in ein Gleitpunktformat von -100 .... +100 % nach folgender Vorschrift um:

$$Ausgang\ von\ CPR_IN = PV_PER * \frac{100}{27648}$$

Die Funktion *PV_NORM* normiert den Ausgang von *CRP_IN* nach folgender Vorschrift:

$$Ausgang\ von\ PV_NORM = (Ausgang\ von\ CPR_IN) * PV_FAC + PV_OFF$$

PV_FAC ist mit 1 und *PV_OFF* ist mit 0 vorbelegt.

Regeldifferenzbildung

Die Differenz von Soll- und Istwert ergibt die Regeldifferenz. Zur Unterdrückung einer kleinen Dauerschwingung aufgrund der Stellgrößen-Quantisierung (z.B. bei einer Puls-Breitenmodulation mit PULSEGEN) wird die Regeldifferenz über eine Totzone (DEAD-BAND) geleitet. Bei *DEADB_W* = 0 ist die Totzone ausgeschaltet.

PI-Schrittalgorithmus

Der FB arbeitet ohne Stellungsrückmeldung. Der I-Anteil des PI-Algorithmus und die gedachte Stellungsrückmeldung werden in einem Integrator (INT) berechnet und als Rückführungswert mit dem verbliebenen P-Anteil verglichen. Die Differenz geht auf ein Dreipunktglied (THREE_ST) und einen Impulsformer (PULSEOUT), der die Impulse für das Stellventil bildet. Über eine Adaption der Ansprechschwelle des Dreipuntgliedes wird die Schalthäufigkeit des Reglers reduziert.

Störgrößenaufschaltung

Am Eingang *DISV* kann eine Störgröße additiv aufgeschaltet werden.

Betriebszustände

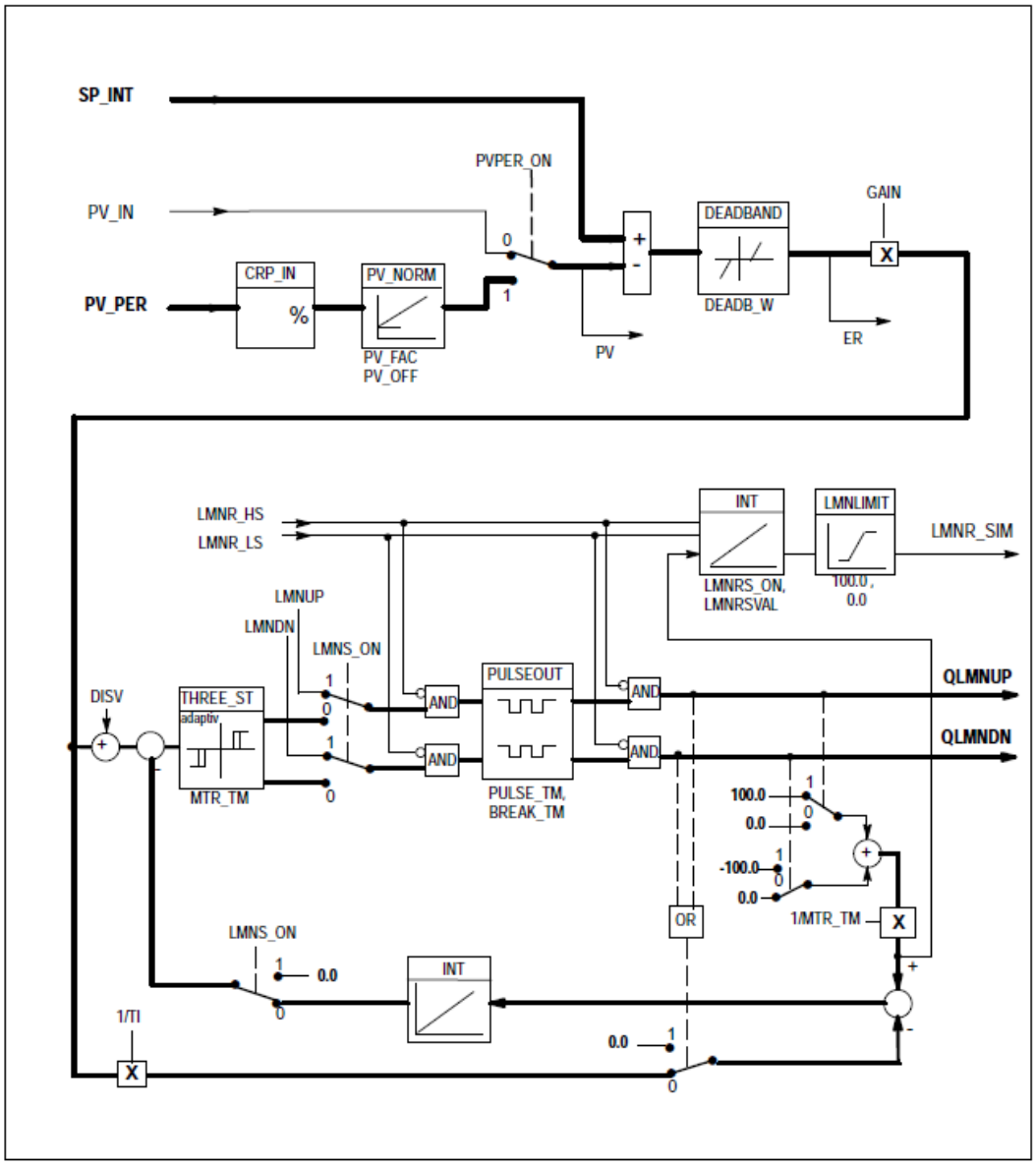
Neustart/Wiederanlauf

- Der FB 42 CONT_S verfügt über eine Neustartroutine, die durchlaufen wird, wenn der Eingangs-Parameter *COM_RST* = TRUE gesetzt ist.
- Alle anderen Ausgänge werden auf ihre Vorbelegungswerte gesetzt.

Fehlerinformationen

Der Baustein enthält keine Fehlerprüfung, deshalb wird keine Fehlerinformation ausgegeben.

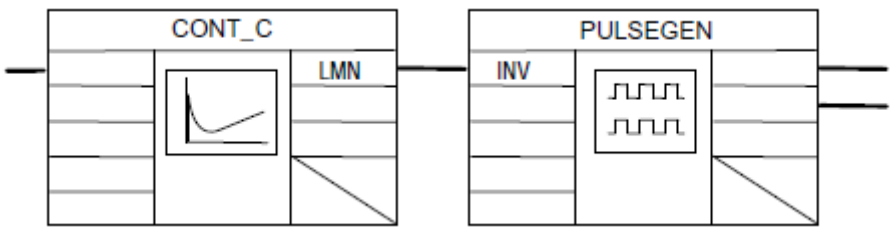
Blockschaltbild



17.5.3 FB 43 - PULSGEN - Impulsformen

Beschreibung

Der FB 43 PULSGEN dient zum Aufbau eines PID-Reglers mit Impulsausgang für proportionale Stellglieder. Mit dem FB 43 lassen sich PID-Zwei- oder Dreipunktregler mit Pulsbreitenmodulation aufbauen. Die Funktion wird meistens in Verbindung mit dem kontinuierlichen Regler CONT_C angewendet.



Bitte beachten Sie, dass der Regelungsbaustein zyklisch in regelmäßigen Abständen aufzurufen ist, ansonsten kann dies zu Fehlberechnungen führen. Rufen Sie hierzu den Regelungsbaustein in einem Weckalarm-OB (OB 30 ... OB 38) auf und geben Sie am Eingang CYCLE das Zeitraster des Weckalarm-OBs an.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
INV	INPUT	REAL	INPUT VARIABLE / Eingangsvariable ■ Am Eingangsparameter *INV* wird eine analoge Stellwertgröße aufgeschaltet. ■ Default: 0.0 ■ Wertebereich: -100.0...100.0 (%)
PER_TM	INPUT	TIME	PERIOD TIME / Periodendauer ■ Am Parameter *PER_TM* wird die konstante Periodendauer der Pulsbreitenmodulation eingegeben. Sie entspricht der Abtastzeit des Reglers. Das Verhältnis Abtastzeit des Impulsformers zu Abtastzeit des Reglers bestimmt die Genauigkeit der Pulsbreitenmodulation. ■ Default: T#1s ■ Wertebereich: ≥20**CYCLE*
P_B_TM	INPUT	TIME	MINIMUM PULSE/BREAK TIME / Mindestimpuls- bzw. Mindestpausendauer ■ Am Parameter *P_B_TM* kann eine minimale Impuls- bzw. Pausenlänge parametriert werden. ■ Default: T#50ms ■ Wertebereich: ≥ *CYCLE*
RATIOFAC	INPUT	REAL	RATIO FACTOR / Verhältnisfaktor ■ Durch den Eingangsparameter *RATIOFAC* kann das Verhältnis der Dauer von negativen zu positiven Impulsen verändert werden. Bei einem thermischen Prozess können damit unterschiedliche Zeitkonstanten für Heizen und Kühlen (z.B. Prozess mit elektrischer Heizung und Wasserkühlung) kompensiert werden. ■ Default: 1.0 ■ Wertebereich: 0.1 ...10.0
STEP3_ON	INPUT	BOOL	THREE STEP CONTROL ON / Dreipunktregelung einschalten ■ Am Eingangsparameter *STEP3_ON* wird die entsprechende Betriebsart aktiviert. Bei Dreipunktregelung arbeiten beide Ausgangssignale. ■ Default: TRUE

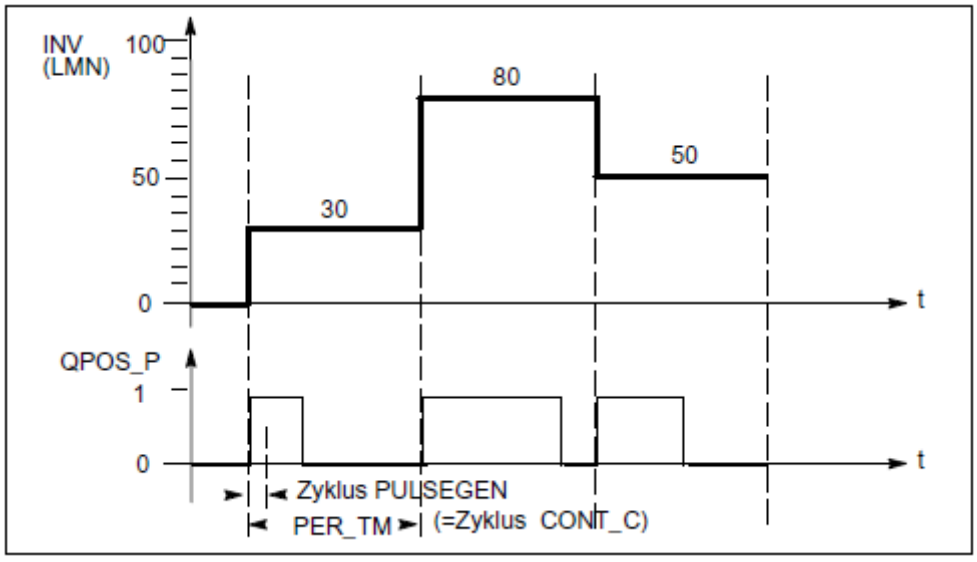
Parameter	Deklaration	Datentyp	Beschreibung
ST2BI_ON	INPUT	BOOL	TWO STEP CONTROL FOR BIPOLAR MANIPULATED VALUE RANGE ON / Zweipunktregelung für bipolaren Stellwertbereich einschalten ■ Am Eingangsparameter *ST2BI_ON* kann zwischen den Betriebsarten "Zweipunktregelung für bipolaren Stellwertbereich" und "Zweipunktregelung für unipolaren Stellwertbereich" gewählt werden. Hierbei muss *STEP3_ON* = FALSE sein. ■ Default: FALSE
MAN_ON	INPUT	BOOL	MANUAL MODE ON / Handbetrieb einschalten ■ Durch Setzen des Eingangsparameter *MAN_ON* können die Ausgangssignale von Hand gesetzt werden. ■ Default: FALSE
POS_P_ON	INPUT	BOOL	POSITIVE MODE ON / positiver Impuls ein ■ Bei Handbetrieb Dreipunktregelung kann am Eingangsparameter *POS_P_ON* das Ausgangssignal *QPOS_P* bedient werden. Bei Handbetrieb Zweipunktregelung wird *QNEG_P* immer invertiert zu *QPOS_P* gesetzt. ■ Default: FALSE
NEG_P_ON	INPUT	BOOL	NEGATIVE PULSE ON / negativer Impuls ein ■ Bei Handbetrieb Dreipunktregelung kann am Eingangsparameter *NEG_P_ON* das Ausgangssignal *QNEG_P* bedient werden. Bei Handbetrieb Zweipunkt– regelung wird *QNEG_P* immer invertiert zu *QPOS_P* gesetzt. ■ Default: FALSE
SYN_ON	INPUT	BOOL	SYNCHRONISATION ON / Synchronisation einschalten ■ Es besteht die Möglichkeit durch Setzen des Eingangsparameters *SYN_ON* die Impulsausgabe mit dem Baustein, der die Eingangsgröße *INV* aktualisiert, automatisch zu synchronisieren. Damit ist gewährleistet, dass eine sich ändernde Eingangsgröße auch schnellstmöglich als Impuls ausgegeben wird. ■ Default: TRUE
COM_RST	INPUT	BOOL	COMPLETE RESTART / Neustart ■ Der Baustein hat eine Neustartroutine, die bearbeitet wird, wenn der Eingang *COM_RST* gesetzt ist. ■ Default: FALSE
CYCLE	INPUT	TIME	SAMPLE TIME / Abtastzeit ■ Die Zeit zwischen den Bausteinaufrufen muss konstant sein. Der Eingang *CYCLE* gibt die Zeit zwischen den Bausteinaufrufen an. ■ Default: T#10ms ■ Wertebereich: ≥ 1ms

Parameter	Deklaration	Datentyp	Beschreibung
QPOS_P	OUTPUT	BOOL	OUTPUT POSITIVE PULSE / Ausgangssignal positiver Impuls ■ Der Ausgangsparameter *QPOS_P* ist gesetzt, wenn ein Impuls ausgegeben werden soll. Bei Dreipunktregelung ist es der positive Impuls. Bei Zweipunktregelung wird *QNEG_P* immer invertiert zu *QPOS_P* gesetzt. ■ Default: FALSE
QNEG_P	OUTPUT	BOOL	OUTPUT NEGATIVE PULSE / Ausgangssignal negativer Impuls ■ Der Ausgangsparameter *QNEG_P* ist gesetzt, wenn ein Impuls ausgegeben werden soll. Bei Dreipunktregelung ist es der negative Impuls. Bei Zweipunktregelung wird QNEG_P immer invertiert zu QPOS_P gesetzt. ■ Default: FALSE

Die Werte der Eingangsparameter werden im Baustein nicht begrenzt; eine Prüfung der Parameter findet nicht statt.

Anwendung

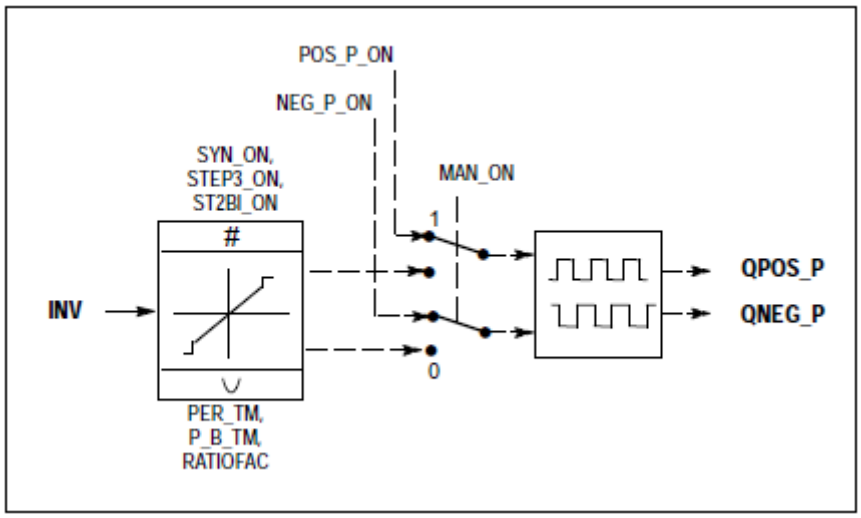
Die Funktion PULSEGEN transformiert die Eingangsgröße *INV* (= LMN des PID-Reglers) durch Modulation der Impulsbreite in eine Impulsfolge mit konstanter Periodendauer, welche der Zykluszeit, mit der die Eingangsgröße aktualisiert wird, entspricht und in *PER_TM* parametriert werden muss. Die Dauer eines Impulses pro Periodendauer ist proportional der Eingangsgröße. Dabei ist der durch *PER_TM* parametrierte Zyklus nicht identisch mit dem Bearbeitungszyklus des FB PULSEGEN. Vielmehr setzt sich ein Zyklus *PER_TM* aus mehreren Bearbeitungszyklen des FB PULSEGEN zusammen, wobei die Anzahl der FB PULSEGEN-Aufrufe pro *PER_TM* Zyklus ein Maß für die Genauigkeit der Impulsbreite darstellt.



Eine Eingangsgröße 30% und 10 FB PULSEGEN-Aufrufe pro *PER_TM* bedeuten also:

- "1" am Ausgang *QPOS* für die ersten drei Aufrufe des FB PULSEGEN (30% von 10 Aufrufen)
- "0" am Ausgang *QPOS* für sieben weitere Aufrufe des FB PULSEGEN (70% von 10 Aufrufen)

Blockschaltbild



Stellwertgenauigkeit

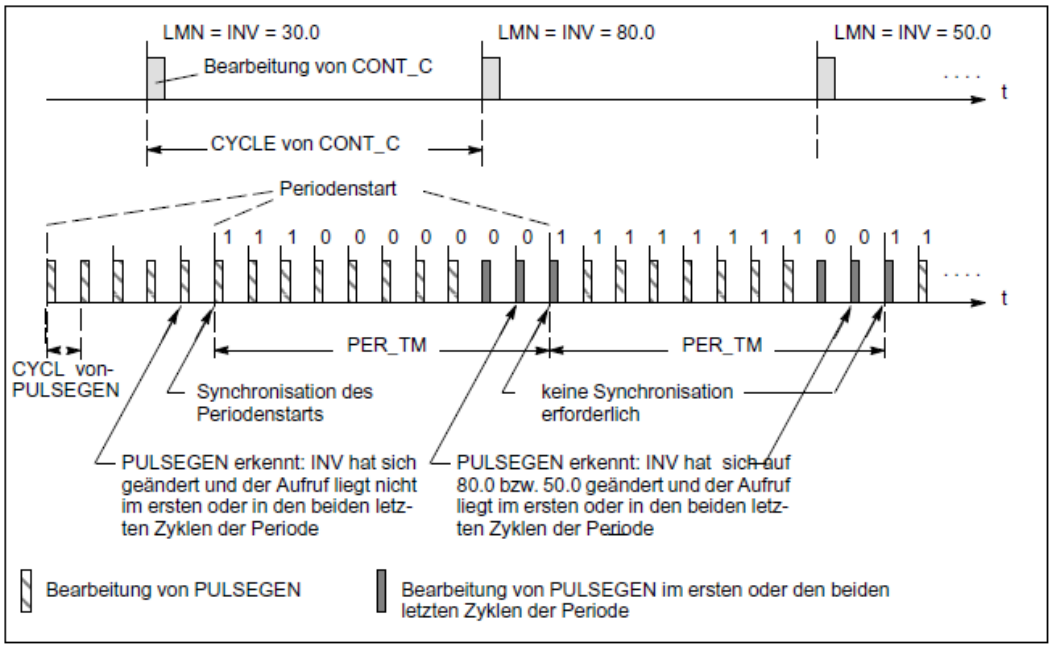
Durch ein "Abtastverhältnis" von 1:10 (CONT_C-Aufrufe zu PULSEGEN-Aufrufe) ist die Stellwertgenauigkeit in diesem Beispiel auf 10% beschränkt, d.h. vorgegebene Eingangswerte *INV* können nur im Raster von 10% auf eine Impulslänge am Ausgang *QPOS* abgebildet werden. Entsprechend erhöht sich die Genauigkeit mit der Anzahl der FB PULSEGEN-Aufrufe pro CONT_C-Aufruf. Wird z.B. PULSEGEN 100 mal häufiger aufgerufen als CONT_C, so erreicht man eine Auflösung von 1% des Stellwertbereiches.

Die Untersetzung der Aufrufhäufigkeit müssen Sie selbst programmieren.

Automatische Synchronisation

Es besteht die Möglichkeit die Impulsausgabe mit dem Baustein, der die Eingangsgröße *INV* (z.B. CONT_C) aktualisiert, automatisch zu synchronisieren. Damit ist gewährleistet, dass eine ändernde Eingangsgröße auch schnellstmöglich als Impuls ausgegeben wird. Der Impulsformer wertet immer im Zeitabstand der Periodendauer *PER_TM* die Eingangsgröße INV aus und wandelt den Wert in ein Impulssignal der entsprechenden Länge. Da aber *INV* meistens in einer langsameren Weckalarmebene berechnet wird, sollte der Impulsformer möglichst schnell nach der Aktualisierung von *INV* mit der Umwandlung des diskreten Wertes in ein Impulssignal beginnen. Dazu kann der Baustein den Start der Periode nach folgendem Verfahren selbst synchronisieren:

- Hat sich *INV* geändert und befindet sich der Bausteinaufruf nicht im ersten oder in den letzten zwei Aufrufzyklen einer Periode, so wird eine Synchronisation durchgeführt. Die Impulsdauer wird neu berechnet und beim nächsten Zyklus mit einer neuen Periode wird mit der Ausgabe begonnen.



Die automatische Synchronisation lässt sich am Eingang *SYN_ON* (= FALSE) abschalten.

> *Durch den Beginn der neuen Periode wird der Altwert von INV (d.h. von LMN) nach erfolgter Synchronisation mehr oder weniger ungenau auf das Impulssignal abgebildet.*

Betriebsarten

Je nach Parametrierung des Impulsformers können PID-Regler mit Dreipunktverhalten oder mit bipolarem bzw. unipolarem Zweipunktausgang konfiguriert werden. Nachstehende Tabelle zeigt die Einstellung der Schalterkombinationen für die möglichen Betriebsarten.

Betriebsart	Schalter		
	MAN_ON	STEP3_ON	ST2BI_ON
Dreipunktregelung	FALSE	TRUE	beliebig
Zweipunktregelung mit bipolarem Stellbereich (-100 % ... 100 %)	FALSE	FALSE	TRUE
Zweipunktregelung mit unipolarem Stellbereich (0 % ... 100 %)	FALSE	FALSE	FALSE
Handbetrieb	TRUE	beliebig	beliebig

Dreipunktregelung

In der Betriebsart Dreipunktregelung können drei Zustände des Stellsignals erzeugt werden. Dazu werden die Zustandswerte der binären Ausgangssignale *QPOS_P* und *QNEG_P* den jeweiligen Betriebszuständen des Stellgliedes zugeordnet. Die Tabelle zeigt das Beispiel einer Temperaturregelung:

Ausgangssignale	Stellglied		
	heizen	aus	kühlen
QPOS_P	TRUE	FALSE	FALSE
QNEG_P	FALSE	FALSE	TRUE

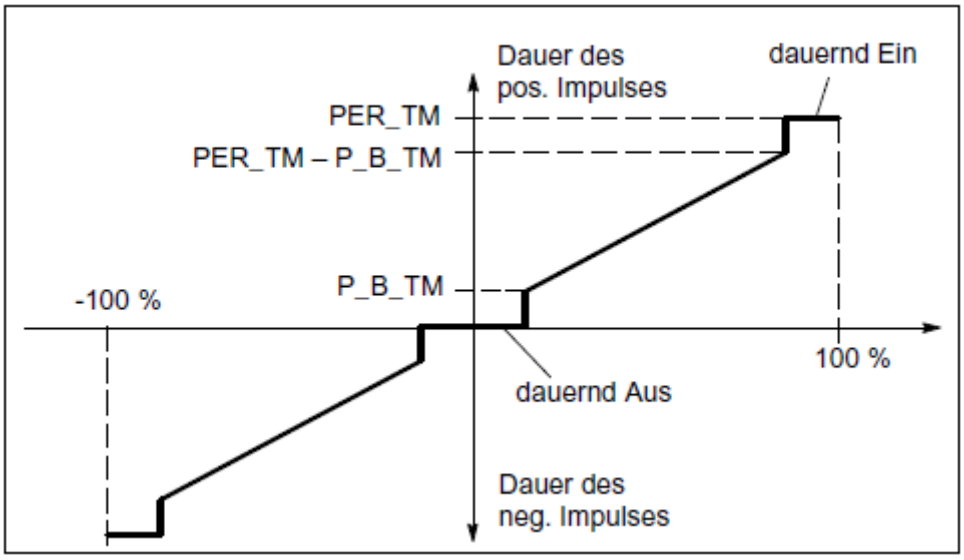
Aus der Eingangsgröße wird über eine Kennlinie die Impulsdauer berechnet. Die Form dieser Kennlinie wird durch die Mindestimpuls- bzw. Mindestpausendauer und dem Verhältnisfaktor definiert. Der normale Wert für den Verhältnisfaktor ist 1. Die Knickpunkte an den Kennlinien werden durch die Mindestimpuls- bzw. Mindestpausendauer verursacht.

- *Mindestimpuls- bzw. Mindestpausendauer*
  Eine richtig parametrierte Mindestimpuls- bzw. Mindestpausendauer *P_B_TM* kann kurze Ein- oder Ausschaltzeiten, die die Lebensdauer von Schaltgliedern und Stelleinrichtungen beeinträchtigen, verhindern.

Kleine Absolutwerte der Eingangsgröße LMN, die eine Impulsdauer kleiner als P_B_TM erzeugen würden, werden unterdrückt. Große Eingangswerte, die eine Impulsdauer größer als (PER_TM - P_B_TM) erzeugen würden, werden auf 100 % bzw. -100 % gesetzt.

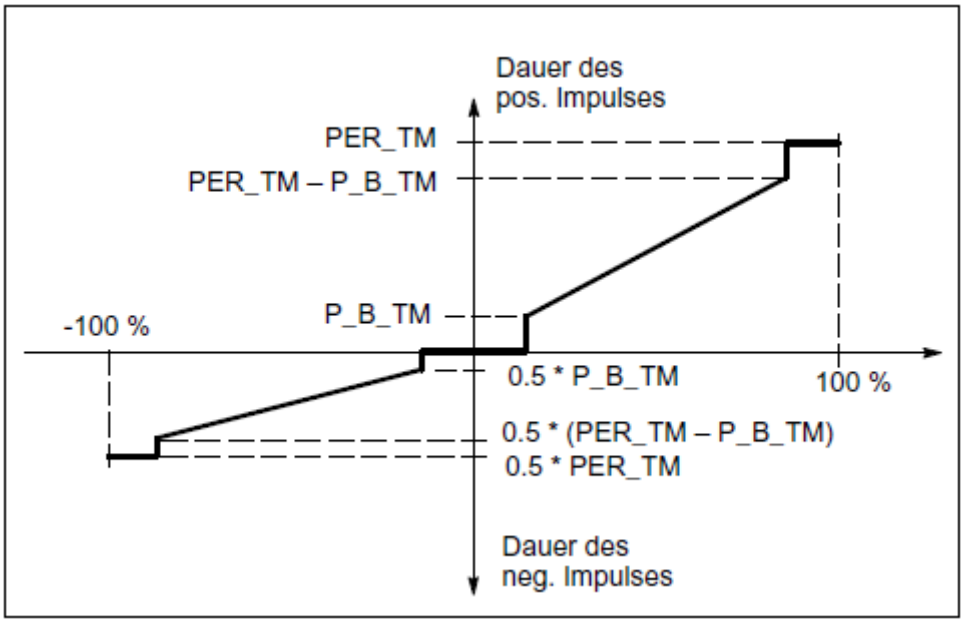
Die Dauer der positiven oder negativen Impulse errechnet sich aus Eingangsgröße (in %) mal Periodendauer:

$$Impulsdauer = \frac{INV}{100} * PER_TM$$



Dreipunktregelung unsymmetrisch

Über den Verhältnisfaktor *RATIOFAC* kann das Verhältnis der Dauer von positiven zu negativen Impulsen verändert werden. Bei einem thermischen Prozess lassen sich damit z.B. unterschiedliche Streckenzeitkonstanten für Heizen und Kühlen berücksichtigen. Der Verhältnisfaktor beeinflusst auch die Mindestimpuls- bzw. Mindestpausendauer. Verhältnisfaktor < 1 bedeutet, der Ansprechwert für negative Impulse wird mit dem Verhältnisfaktor multipliziert.



- *Verhältnisfaktor < 1*
  Die aus Eingangsgröße mal Periodendauer berechnete Impulsdauer am negativen Impulsausgang wird um den Verhältnisfaktor verkürzt.

$$positive\ \ Impulsdauer = \frac{INV}{100} * PER_TM$$

$$negative\ \ Impulsdauer = \frac{INV}{100} * PER_TM * RATIOFAC$$

- *Verhältnisfaktor > 1*
  Die aus Eingangsgröße mal Periodendauer berechnete Impulsdauer am positiven Impulsausgang wird um den Verhältnisfaktor verkürzt.

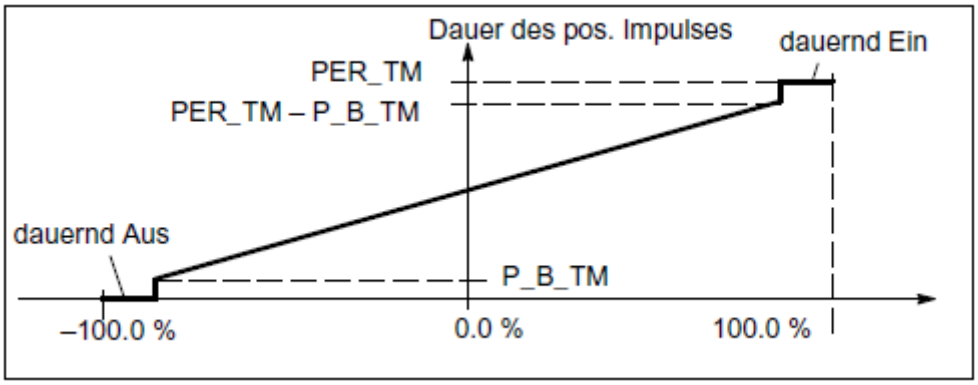
$$negative\ \ Impulsdauer = \frac{INV}{100} * PER_TM$$

$$positive\ \ Impulsdauer = \frac{INV}{100} * \frac{PER_TM}{RATIOFAC}$$

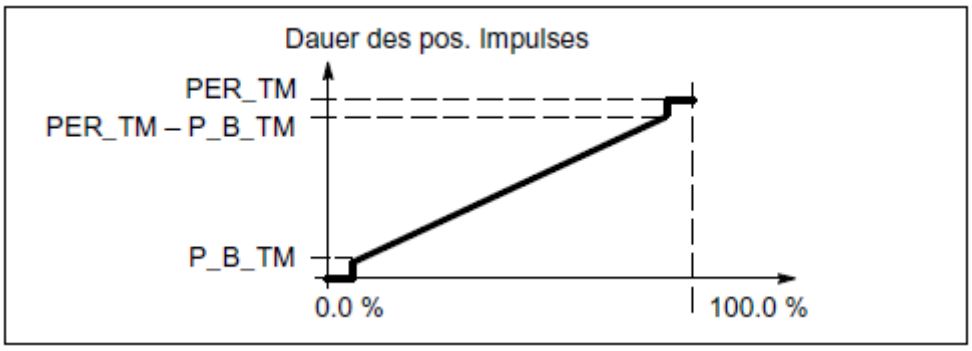
Zweipunktregelung

Bei der Zweipunktregelung wird nur der positive Impulsausgang *QPOS_P* von PULSEGEN mit dem betreffenden Ein/Aus-Stellglied verbunden. Je nach genutztem Stellwertbereich hat der Zweipunktregler einen bipolaren oder einen unipolaren Stellwertbereich.

- *Zweipunktregelung mit bipolarem Stellwertbereich (-100 % ... 100 %)*



- *Zweipunktregelung mit unipolarem Stellwertbereich (0 % ... 100 %)*



An *QNEG_P* steht das negierte Ausgangssignal zur Verfügung, falls die Verschaltung des Zweipunktreglers im Regelkreis ein logisch invertiertes Binärsignal für die Stellimpulse erfordert.

Impuls	Stellglied	
	ein	aus
QPOS_P	TRUE	FALSE
QNEG_P	FALSE	TRUE

Handbetrieb bei Zwei- bzw. Dreipunkt- Regelung

Im Handbetrieb (*MAN_ON* = TRUE) können die Binärausgänge des Dreipunkt- bzw. Zweipunktreglers über die Signale *POS_P_ON* und *NEG_P_ON* unabhängig von *INV* gesetzt werden.

	POS_P_ON	NEG_P_ON	QPOS_P	QNEG_P
Dreipunktrege-lung	FALSE	FALSE	FALSE	FALSE
	TRUE	FALSE	TRUE	FALSE
	FALSE	TRUE	FALSE	TRUE
	TRUE	TRUE	FALSE	FALSE
Zweipunktrege-lung	FALSE	beliebig	FALSE	TRUE
	TRUE	beliebig	TRUE	FALSE

Betriebszustände

Neustart/Wiederanlauf

- Bei Neustart werden alle Signalausgänge auf Null gesetzt.

Fehlerinformationen

Der Baustein enthält keine Fehlerprüfung, deshalb wird keine Fehlerinformation ausgegeben.

17.5.4 FB 58 - TCONT_CP - Kontinuierliches Temperaturregeln

Beschreibung

Der FB 58 TCONT_CP dient zum Regeln von Temperaturprozessen mit kontinuierlicher oder impulsförmiger Ansteuerung. Über die Parametrierung können Sie Teilfunktionen des PID-Reglers zu- oder abschalten und damit diesen an die Regelstrecke anpassen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
PV_IN	INPUT	REAL	PROCESS VARIABLE IN / Istwert Eingang ■ Am Eingang *PV_IN* kann ein Inbetriebsetzungswert parametriert oder ein externer Istwert im Gleitpunktformat verschaltet werden. ■ Default: 0.0 ■ Wertebereich: Abhängig von den eingesetzten Sensoren
PV_PER	INPUT	WORD	PROCESS VARIABLE PERIPHERY / Istwert Peripherie ■ Der Istwert in Peripherieformat wird am Eingang *PV_PER* mit dem Regler verschaltet. ■ Default: 0
DISV	INPUT	REAL	DISTURBANCE VARIABLE / Störgröße ■ Für eine Störgrößenaufschaltung wird die Störgröße am Eingang *DISV* verschaltet. ■ Default: 0.0
INT_HPOS	INPUT	BOOL	INTEGRAL ACTION HOLD IN POSITIVE DIRECTION / I-Anteil blockieren in positive Richtung ■ Der Ausgang des Integrierers kann in positive Richtung blockiert werden. Hierzu muss der Eingang *INT_HPOS* auf TRUE gesetzt werden. Bei einer Kaskaderegelung wird *INT_HPOS* des Führungsregler mit *QLMN_HLM* des Folgereglers verschaltet. ■ Default: FALSE
INT_HNEG	INPUT	BOOL	INTEGRAL ACTION HOLD IN NEGATIVE DIRECTION / I-Anteil blockieren in negative Richtung ■ Der Ausgang des Integrierers kann in negative Richtung blockiert werden. Hierzu muss der Eingang *INT_HNEG* auf TRUE gesetzt werden. Bei einer Kaskaderegelung wird *INT_HNEG* des Führungsreglers mit *QLMN_LLM* des Folgereglers verschaltet. ■ Default: FALSE

Parameter	Deklaration	Datentyp	Beschreibung
SELECT	INPUT	BOOL	SELECTION OF CALL PID AND PULSE GENERATOR / Auswahl des Aufrufverhaltens von PID und Impulsformer ■ Ist der Impulsformer eingeschaltet, gibt es mehrere Möglichkeiten den PID-Algorithmus und Impulsformer aufzurufen: – *SELECT* = 0: Der Regler wird in einer schnellen Weckalarmebene aufgerufen und es werden PID-Algorithmus und Impulsformer bearbeitet. – *SELECT* = 1: Der Regler wird im OB 1 aufgerufen und es wird nur der PID-Algorithmus bearbeitet. – *SELECT* = 2: Der Regler wird in einer schnellen Weckalarmebene aufgerufen und es wird nur der Impulsformer bearbeitet. – *SELECT* = 3: Der Regler wird in einer langsamen Weckalarmebene aufgerufen und es wird nur der PID-Algorithmus bearbeitet. ■ Default: 0 ■ Wertebereich: 0 ... 3
PV	OUTPUT	REAL	PROCESS VARIABLE / Istwert ■ Am Ausgang *PV* wird der effektiv wirkende Istwert ausgegeben. ■ Default: 0.0 ■ Wertebereich: abhängig von den eingesetzten Sensoren
LMN	OUTPUT	REAL	MANIPULATED VALUE / Stellwert ■ Am Ausgang *LMN* wird der effektiv wirkende Stellwert in Gleitpunktformat ausgegeben. ■ Default: 0.0
LMN_PER	OUTPUT	WORD	MANIPULATED VALUE PERIPHERY / Stellwert Peripherie ■ Der Stellwert in Peripherieformat wird am Ausgang *LMN_PER* mit dem Regler verschaltet. ■ Default: 0
QPULSE	OUTPUT	BOOL	QUTPUT PULSE SIGNAL / Pulsausgang ■ Der Stellwert wird pulsweitenmoduliert am Ausgang *QPULSE* ausgegeben. ■ Default: FALSE
QLMN_HLM	OUTPUT	BOOL	HIGH LIMIT OF MANIPULATED VALUE REACHED / Obere Begrenzung des Stellwertes angesprochen ■ Der Stellwert wird immer auf eine obere und untere Grenze begrenzt. Der Ausgang *QLMN_HLM* meldet die Überschreitung der oberen Begrenzung. ■ Default: FALSE

Parameter	Deklaration	Datentyp	Beschreibung
QLMN_LLM	OUTPUT	BOOL	LOW LIMIT OF MANIPULATED VALUE REACHED / Untere Begrenzung des Stellwertes angesprochen ■ Der Stellwert wird immer auf eine obere und untere Grenze begrenzt. Der Ausgang *QLMN_LLM* meldet die Überschreitung der unteren Begrenzung. ■ Default: FALSE
QC_ACT	OUTPUT	BOOL	NEXT CYCLE, THE CONTINUOUS CONTROLLER IS WORKING / Kontinuierlicher Regleranteil wird beim nächsten Aufruf bearbeitet ■ Der Parameter zeigt an, ob beim nächsten Bausteinaufruf der kontinuierliche Regleranteil bearbeitet wird (nur relevant, wenn *SELECT* den Wert 0 oder 1 hat). ■ Default: TRUE
CYCLE	INPUT/ OUTPUT	REAL	SAMPLE TIME OF CONTINUOUS CONTROLLER [s] / Abtastzeit des kontinuierlichen Reglers [s] ■ Vorgabe der Abtastzeit für den PID-Algorithmus. Der Optimierer berechnet in Phase 1 die Abtastzeit und trägt sie in *CYCLE* ein. ■ Default: 0.1s ■ Wertebereich: ≥ 1ms
CYCLE_P	INPUT/ OUTPUT	REAL	SAMPLE TIME OF PULSE GENERATOR [s] / Abtastzeit des Impulsformers [s] ■ An diesem Eingang geben Sie die Abtastzeit für den Impulsformer-Anteil ein. Der FB 58 "TCONT_CP" berechnet in Phase 1 die Abtastzeit und trägt sie in *CYCLE_P* ein. ■ Default: 0.2s ■ Wertebereich: ≥ 1ms
SP_INT	INPUT/ OUTPUT	REAL	INTERNAL SETPOINT / Interner Sollwert ■ Der Eingang *SP_INT* dient zur Vorgabe eines Sollwertes. ■ Default: 0.0 ■ Wertebereich des Istwertes
MAN	INPUT/ OUTPUT	REAL	MANUAL VALUE / Handwert ■ Der Eingang *MAN* dient zur Vorgabe eines Handwertes. Im Automatikbetrieb wird er dem Stellwert nachgeführt. ■ Default: 0.0
COM_RST	INPUT/ OUTPUT	REAL	COMPLETE RESTART / Neustart ■ Der Baustein hat eine Initialisierungsroutine, die bearbeitet wird, wenn der Eingang *COM_RST* gesetzt ist. ■ Default: FALSE
MAN_ON	INPUT/ OUTPUT	REAL	MANUAL OPERATION ON / Handbetrieb einschalten ■ Ist der Eingang *MAN_ON* gesetzt, ist der Regelkreis unterbrochen. Als Stellwert wird der Handwert MAN vorgegeben. ■ Default: TRUE

Interne Parameter

Parameter	Deklaration	Datentyp	Beschreibung
DEADB_W	INPUT	REAL	DEAD BAND WIDTH / Totzonenbreite ■ Die Regeldifferenz wird über eine Totzone geführt. Der Eingang *DEADB_W* bestimmt die Größe der Totzone. ■ Default: 0.0 ■ Wertebereich: Abhängig von den eingesetzten Sensoren
I_ITLVAL	INPUT	REAL	INITIALIZATION VALUE OF THE INTEGRAL ACTION / Initialisierungswert für I-Anteil ■ Der Ausgang des Integrierers kann am Eingang *I_ITL_ON* gesetzt werden. Am Eingang *I_ITLVAL* steht der Initialisierungswert. ■ Bei Neustart *COM_RST* = TRUE wird der I-Anteil auf den Initialisierungswert gesetzt. ■ Default: 0.0 ■ Wertebereich: 0 bis 100 %
LMN_HLM	INPUT	REAL	MANIPULATED VARIABLE HIGH LIMIT / Stellwert obere Begrenzung ■ Der Stellwert wird immer auf eine obere und untere Grenze begrenzt. Der Eingang *LMN_HLM* gibt die obere Begrenzung an. ■ Default: 100.0 ■ Wertebereich: > *LMN_ LLM*
LMN_LLM	INPUT	REAL	MANIPULATED VARIABLE LOW LIMIT / Stellwert untere Begrenzung ■ Der Stellwert wird immer auf eine obere und untere Grenze begrenzt. Der Eingang *LMN_LLM* gibt die untere Begrenzung an. ■ Default: 0.0 ■ Wertebereich: < *LMN_HLM*
PV_FAC	INPUT	REAL	PROCESS VARIABLE FACTOR / Istwertfaktor ■ Der Eingang *PV_FAC* wird mit dem *PV_PER* multipliziert. Der Eingang dient zur Anpassung des Istwertbereiches. ■ Default: 1.0
PV_OFFS	INPUT	REAL	PROCESS VARIABLE OFFSET / Istwertoffset ■ Der Eingang *PV_OFFS* wird zum *PV_PER* addiert. Der Eingang dient zur Anpassung des Istwertbereiches. ■ Default: 0.0
LMN_FAC	INPUT	REAL	MANIPULATED VARIABLE FACTOR / Stellwertfaktor ■ Der Eingang *LMN_FAC* wird mit dem Stellwert multipliziert. Der Eingang dient zur Anpassung des Stellwertbereiches. ■ Default: 1.0

Parameter	Deklaration	Datentyp	Beschreibung
LMN_OFFS	INPUT	REAL	MANIPULATED VARIABLE OFFSET / Stellwertoffset ■ Der Eingang *LMN_OFFS* wird zum Stellwert addiert. Der Eingang dient zur Anpassung des Stellwertbereiches. ■ Default: 0.0
PER_TM	INPUT	REAL	PERIOD TIME [s] / Periodendauer [s] ■ Am Parameter *PER_TM* wird die Periodendauer der Pulsbreitenmodulation eingegeben. Das Verhältnis Periodendauer zu Abtastzeit des Impulsformers bestimmt die Genauigkeit der Pulsbreitenmodulation. ■ Default: 1.0 s ■ Wertebereich: ≥ *CYCLE*
P_B_TM	INPUT	REAL	MINIMUM PULSE/BREAK TIME [s] / Mindestimpuls- bzw. Mindestpausendauer [s] ■ Am Parameter *P_B_TM* kann eine minimale Impulsbzw. Pausenlänge parametriert werden. *P_B_TM* wird intern auf > *CYCLE_P* begrenzt. ■ Default: 0.02 s ■ Wertebereich: ≥ 0.0
TUN_DLMN	INPUT	REAL	DELTA MANIPULATED VARIABLE FOR PROCESS EXCITATION / Delta-Stellwert für Prozessanregung ■ Die Prozessanregung für die Regleroptimierung erfolgt durch einen Stellwertsprung von *TUN_DLMN*. ■ Default: 20.0 ■ Wertebereich: -100.0 ... 100.0 %
PER_MODE	INPUT	INT	PERIPHERY MODE / Peripherie Betriebsart ■ An diesem Schalter können Sie den Typ der AE-Baugruppe eingeben. Der Istwert am Eingang *PV_PER* wird dadurch am Ausgang *PV* in °C normiert. – *PER_MODE* = 0: Standard – *PER_MODE* = 1: Klima – *PER_MODE* = 2: Strom/Spannung ■ Default: 0 ■ Wertebereich: 0, 1, 2
PVPER_ON	INPUT	BOOL	PROCESS VARIABLE PERIPHERY ON / Istwert Peripherie einschalten ■ Soll der Istwert von der Peripherie eingelesen werden, so muss der Eingang *PV_PER* mit der Peripherie verschaltet werden und der Eingang *PVPER_ON* gesetzt werden. ■ Default: FALSE
I_ITL_ON	INPUT	BOOL	INITIALIZATION OF THE INTEGRAL ACTION ON / I-Anteil setzen ■ Der Ausgang des Integrierers kann auf den Eingang *I_ITLVAL* gesetzt werden. Hierzu muss der Eingang *I_ITL_ON* gesetzt werden. ■ Default: FALSE

Parameter	Deklaration	Datentyp	Beschreibung
PULSE_ON	INPUT	BOOL	PULSE GENERATOR ON / Impulsformer einschalten ■ Mit *PULSE_ON* = TRUE wird der Impulsformer eingeschaltet. ■ Default: FALSE
TUN_KEEP	INPUT	BOOL	KEEP TUNING ON / Optimierbetrieb halten ■ Ein Übergang in den Automatikbetrieb erfolgt erst, wenn *TUN_KEEP* = FALSE. ■ Default: FALSE
ER	OUTPUT	REAL	ERROR SIGNAL / Regeldifferenz ■ Am Ausgang *ER* wird die effektiv wirkende Regeldifferenz ausgegeben. ■ Default: 0.0 ■ Wertebereich: Abhängig von den eingesetzten Sensoren
LMN_P	OUTPUT	REAL	PROPORTIONALITY COMPONENT / P-Anteil ■ Der Ausgang *LMN_P* beinhaltet den Proportionalanteil der Stellgröße. ■ Default: 0.0
LMN_I	OUTPUT	REAL	INTEGRAL COMPONENT / I-Anteil ■ Der Ausgang *LMN_I* beinhaltet den Integralanteil der Stellgröße. ■ Default: 0.0
LMN_D	OUTPUT	REAL	DERIVATIVE COMPONENT / D-Anteil ■ Der Ausgang *LMN_D* beinhaltet den Differentialanteil der Stellgröße. ■ Default: 0.0
PHASE	OUTPUT	INT	PHASE OF SELF TUNING / Phasenanzeige der Regleroptimierung ■ Am Ausgang *PHASE* wird die aktuelle Ablaufphase der Regleroptimierung angezeigt (0...7). ■ Default: 0 ■ Wertebereich: 0, 1, 2, 3, 4, 5, 7
STATUS_H	OUTPUT	INT	STATUS HEATING OF SELF TUNING / Status Heizen der Regleroptimierung ■ *STATUS_H* zeigt einen Diagnosewert über die Suche des Wendepunktes beim Heizvorgang an. ■ Default: 0
STATUS_D	OUTPUT	INT	STATUS CONTROLLER DESIGN OF SELF TUNING / Status Reglerentwurf der Regleroptimierung ■ *STATUS_D* zeigt einen Diagnosewert über den Reglerentwurf beim Heizvorgang an. ■ Default: 0

Parameter	Deklaration	Datentyp	Beschreibung
QTUN_RUN	OUTPUT	BOOL	TUNING IS ACTIVE (PHASE 2) / Optimierung läuft (Phase 2) ■ Die Optimierung wurde durch Aufschalten der Optimierungsstellgröße begonnen und befindet sich noch in Phase 2 (Wendepunktsuche). ■ Default: 0
PI_CON	OUTPUT	STRUCT	PI CONTROLLER PARAMETERS / PI Reglerparameter
GAIN	OUTPUT	REAL	PI PROPORTIONAL GAIN / PI Reglerverstärkung ■ Default: 0.0 ■ Wertebereich: % / phys. Einheit
TI	OUTPUT	REAL	PI RESET TIME [s] / PI Integrationszeit [s] ■ Default: 0.0 s ■ Wertebereich: ≥ 0.0 s
PID_CON	OUTPUT	STRUCT	PID CONTROLLER PARAMETERS / PID Reglerparameter
GAIN	OUTPUT	REAL	PID PROPORTIONAL GAIN / PID Reglerverstärkung ■ Default: 0.0
TI	OUTPUT	REAL	PID RESET TIME [s] / PID Integrationszeit [s] ■ Default: 0.0 s ■ Wertebereich: ≥ 0.0 s
TD	OUTPUT	REAL	PID DERIVATIVE TIME [s] / PID Differenzierzeit [s] ■ Default: 0.0 s ■ Wertebereich: ≥ 0.0 s
PAR_SAVE	OUTPUT	STRUCT	SAVED CONTROLLER PARAMETERS / Gespeicherte PID Reglerparameter ■ In dieser Struktur werden die PID-Parameter gespeichert.
PFAC_SP	INPUT/ OUTPUT	REAL	PROPORTIONAL FACTOR FOR SETPOINT CHANGES / Proportionalfaktor bei Sollwertänderungen ■ Default: 1.0 ■ Wertebereich: 0.0 ... 1.0
GAIN	OUTPUT	REAL	PROPORTIONAL GAIN / Reglerverstärkung ■ Default: 0.0 ■ Wertebereich: % / phys. Einheit
TI	INPUT/ OUTPUT	REAL	RESET TIME [s] / Integrationszeit [s] ■ Default: 40.0 s ■ Wertebereich: ≥ 0.0 s
TD	INPUT/ OUTPUT	REAL	DERIVATIVE TIME [s] / Differenzierzeit [s] ■ Default: 10.0 s ■ Wertebereich: ≥ 0.0 s

Parameter	Deklaration	Datentyp	Beschreibung
D_F	OUTPUT	REAL	DERIVATIVE FACTOR / Differenzierfaktor ■ Default: 5.0 ■ Wertebereich: 5.0 ... 10.0
CON_ZONE	OUTPUT	REAL	CONTROL ZONE ON / Regelzone einschalten ■ Default: 100.0 ■ Wertebereich: ≥ 0.0
CONZ_ON	OUTPUT	REAL	CONTROL ZONE / Regelzonenbreite ■ Default: FALSE
PFAC_SP	INPUT/ OUTPUT	REAL	PROPORTIONAL FACTOR FOR SETPOINT CHANGES / Proportionalfaktor bei Sollwertänderungen ■ *PFAC_SP* gibt den wirksamen P-Anteil bei Sollwertänderung an. Er wird zwischen 0 und 1 eingestellt. – 1: P-Anteil ist auch bei Sollwertänderungen voll wirksam. – 0: P-Anteil ist bei Sollwertänderungen nicht wirksam. ■ Default: 1.0 ■ Wertebereich: 0.0 ... 1.0
GAIN	INPUT/ OUTPUT	REAL	PROPORTIONAL GAIN / Reglerverstärkung ■ Der Eingang *GAIN* gibt die Reglerverstärkung an. Eine Invertierung des Regelsinns wird durch das negative Vorzeichen von GAIN erreicht. ■ Default: 0.0 ■ Wertebereich: % / phys. Einheit
TI	INPUT/ OUTPUT	REAL	RESET TIME [s] / Integrationszeit [s] ■ Der Eingang *TI* (Nachstellzeit) bestimmt das Zeitverhalten des Integrierers. ■ Default: 40.0 s ■ Wertebereich: ≥ 0.0 s
TD	INPUT/ OUTPUT	REAL	DERIVATIVE TIME [s] / Differenzierzeit [s] ■ Der Eingang *TD* (Vorhaltezeit) bestimmt das Zeitverhalten des Differenzierers. ■ Default: 10.0 s ■ Wertebereich: ≥ 0.0 s
D_F	INPUT/ OUTPUT	REAL	DERIVATIVE FACTOR / Differenzierfaktor ■ Der Differenzierfaktor *D_F* bestimmt die Verzögerung des D-Anteils. ■ *D_F* = Differenzierzeit / "Verzögerung des D-Anteils" ■ Default: 5.0 ■ Wertebereich: 5.0 ... 10.0

Parameter	Deklaration	Datentyp	Beschreibung
CON_ZONE	INPUT/ OUTPUT	REAL	CONTROL ZONE ON / Regelzonenbreite ■ Ist die Regeldifferenz größer als die Regelzonenbreite *CON_ZONE*, so wird die obere Stellwertbegrenzung als Stellwert ausgegeben. ■ Ist die Regeldifferenz kleiner als die negative Regelzonenbreite, so wird die untere Stellwertbegrenzung als Stellwert ausgegeben. ■ Default: 100.0 ■ Abhängig von den eingesetzten Sensoren
CONZ_ON	INPUT/ OUTPUT	BOOL	CONTROL ZONE / Regelzone einschalten ■ Mit *CONZ_ON* = TRUE können Sie die Regelzone einschalten. ■ Default: FALSE
TUN_ON	INPUT/ OUTPUT	BOOL	SELF TUNING ON / Regleroptimierung einschalten ■ Wenn *TUN_ON* = TRUE wird der Stellwert gemittelt bis entweder durch einen Sollwertsprung oder durch *TUN_ST* = TRUE die Stellwertanregung *TUN_DLMN* aufgeschaltet wird. ■ Default: FALSE
TUN_ST	INPUT/ OUTPUT	BOOL	START SELF TUNING / Regleroptimierung starten ■ Soll bei der Regleroptimierung am Arbeitspunkt der Sollwert konstant bleiben, wird durch *TUN_ST* = TRUE ein Stellwertsprung um *TUN_DLMN* aufgeschaltet. ■ Default: FALSE
UNDO_PAR	INPUT/ OUTPUT	BOOL	UNDO CHANGE OF CONTROLLER PARAMETERS / Rückgängigmachen der Reglerparameteränderung ■ Lädt die Reglerparameter *PFAC_SP*, *GAIN*, *TI*, *TD*, *D_F*, *CONZ_ON* und *CON_ZONE* aus der Datenstruktur *PAR_SAVE* (nur im Handbetrieb). ■ Default: FALSE
SAVE_PAR	INPUT/ OUTPUT	BOOL	SAVE CURRENT CONTROLLER PARAMETERS / Aktuelle Reglerparameter sichern ■ Sichert die Reglerparameter *PFAC_SP*, *GAIN*, *TI*, *TD*, *D_F*, *CONZ_ON* und *CON_ZONE* in die Datenstruktur *PAR_SAVE*. ■ Default: FALSE
LOAD_PID	INPUT/ OUTPUT	BOOL	LOAD OPTIMIZED PI/PID PARAMETERS / Optimierte PID-Parameter laden ■ Lädt die Reglerparameter *GAIN*, *TI*, *TD* in Abhängigkeit von *PID_ON* aus der Datenstruktur *PI_CON* bzw. *PID_CON* (nur im Handbetrieb) ■ Default: FALSE

Parameter	Deklaration	Datentyp	Beschreibung
PID_ON	INPUT/ OUTPUT	BOOL	PID MODE ON / PID Betriebsart einschalten ■ Am Eingang *PID_ON* können Sie festlegen, ob der optimierte Regler als PI- oder als PID-Regler arbeiten soll. – PID-Regler: *PID_ON* = TRUE – PI-Regler: *PID_ON* = FALSE ■ Es kann jedoch sein, dass bei manchen Streckentypen trotz *PID_ON* = TRUE nur ein PI-Regler entworfen wird. ■ Default: TRUE
GAIN_P	OUTPUT	REAL	PROZESS PROPORTIONAL GAIN / Prozessverstärkung ■ Identifizierte Prozessverstärkung. Beim Streckentyp I wird *GAIN_P* tendenziell zu klein geschätzt. ■ Default: 0.0
TU	OUTPUT	REAL	DELAY TIME [s] / Verzugszeit [s] ■ Identifizierte Verzugszeit vom Prozess. ■ Default: 0.0 ■ Wertebereich: ≥ 3**CYCLE*
TA	OUTPUT	REAL	RECOVERY TIME [s] / Ausgleichszeit [s] ■ Identifizierte Ausgleichszeit vom Prozess. Beim Streckentyp I wird *TA* tendenziell zu klein geschätzt. ■ Default: 0.0
KIG	OUTPUT	REAL	MAXIMAL ASCENT RATIO OF PV WITH 100 % LMN CHANGE / Maximaler Istwertanstieg bei einer Stellgrößenanregung von 0 nach 100 % [1/s] ■ *GAIN_P* = 0.01 * *KIG* * *TA* ■ Default: 0.0
N_PTN	OUTPUT	REAL	PROCESS ORDER / Prozessordnung ■ Der Parameter gibt die Ordnung der Strecke an. Auch "nicht ganzzahlige Werte" sind möglich. ■ Default: 0.0 ■ Wertebereich: 1.01 ... 10.0
TM_LAG_P	OUTPUT	REAL	TIME LAG OF PTN MODEL [s] / Zeitkonstante eines PTN-Modells [s] ■ Zeitkonstante eines PTN-Modells (sinnvolle Werte nur für *N_PTN* ≥ 2). ■ Default: 0.0
T_P_INF	OUTPUT	REAL	TIME TO POINT OF INFLECTION [s] / Zeit bis zum Wendepunkt [s] ■ Zeit von der Prozessanregung bis zum Wendepunkt. ■ Default: 0.0

Parameter	Deklaration	Datentyp	Beschreibung
P_INF	OUTPUT	REAL	PV AT POINT OF INFLECTION - PV0 / Istwert am Wendepunkt-PV0 ■ Istwertänderung von der Prozessanregung bis zum Wendepunkt. ■ Default: 0.0 ■ Wertebereich: Wertebereich des Istwertes
LMN0	OUTPUT	REAL	MANIPULATED VAR. AT BEGIN OF TUNING / Stellwert zu Beginn der Optimierung ■ Wird in Phase 1 ermittelt (Mittelwert). ■ Default: 0.0 ■ Wertebereich: 0 ... 100 %
PV0	OUTPUT	REAL	PROCESS VALUE AT BEGIN OF TUNING / Istwert zu Beginn der Optimierung ■ Default: 0.0 ■ Wertebereich: Wertebereich des Istwertes
PVDT0	OUTPUT	REAL	RATE OF CHANGE OF PV AT BEGIN OF TUNING [1/s] / Istwertsteigung zu Beginn der Optimierung [1/s] ■ Vorzeichen angepasst ■ Default: 0.0
PVDT	OUTPUT	REAL	CURRENT RATE OF CHANGE OF PV [1/s] / Momentane Istwertsteigung [1/s] ■ Vorzeichen angepasst ■ Default: 0.0
PVDT_MAX	OUTPUT	REAL	MAX. RATE OF CHANGE OF PV PER SECOND [1/s] / Max. Änderung des Istwertes pro Sekunde [1/s] ■ Maximale Ableitung des Istwertes am Wendepunkt (Vorzeichen angepasst, immer > 0), wird verwendet zur Berechnung von *TU* und *KIG*. ■ Default: 0.0
NOI_PVDT	OUTPUT	REAL	RATIO OF NOISE IN PVDT_MAX IN % / Rauschanteil in PVDT_MAX in % ■ Je größer der Rauschanteil, desto ungenauer (sanfter) die Reglerparameter. ■ Default: 0.0
NOISE_PV	OUTPUT	REAL	ABSOLUTE NOISE IN PV / Absolutes Rauschen im Istwert ■ Differenz zwischen maximalem und minimalem Istwert in Phase 1. ■ Default: 0.0
FIL_CYC	OUTPUT	INT	NO OF CYCLES FOR MEAN-VALUE FILTER / Anzahl der Zyklen des Mittelwertfilters ■ Der Istwert wird über *FIL_CYC* Zyklen gemittelt. *FIL_CYC* wird bei Bedarf automatisch von 1 bis max. 1024 erhöht. ■ Default: 1 ■ Wertebereich: 1 ... 1024

Parameter	Deklaration	Datentyp	Beschreibung
POI_CMAX	OUTPUT	INT	MAX NO OF CYCLES AFTER POINT OF INFLECTION / Maximale Anz. Zyklen nach Wendepunkt ■ Diese Zeit wird genutzt, um bei Messrauschen einen weiteren (d. h. besseren) Wendepunkt zu finden. Erst dann wird die Optimierung beendet. ■ Default: 2
POI_CYCL	OUTPUT	INT	NUMBER OF CYCLES AFTER POINT OF INFLECTION / Anzahl Zyklen nach Wendepunkt ■ Default: 0

Anwendung

- Die Arbeitsweise basiert auf dem PID-Regelalgorithmus, der mit zusätzlichen Funktionen für Temperaturprozesse ausgestattet ist. Der Regler liefert analoge Stellwerte und pulsbreitenmodulierte Stellsignale. Der Regler versorgt ein Stellglied, d. h. Sie können mit einem Regler entweder nur heizen oder nur kühlen.
- Den FB 58 TCONT_CP können Sie sowohl für reine Heizstrecken als auch für reine Kühlstrecken einsetzen. Bei Einsatz für einen Kühlprozess müssen Sie *GAIN* mit einem negativen Wert parametrieren. Die so parametrierte Reglerinvertierung bewirkt, dass nun z.B. bei einem Anstieg der Temperatur sich die Stellgröße *LMN* und damit die Kühlleistung erhöht.
- Neben den Funktionen im Soll- und Istwertzweig realisiert der FB einen fertigen PID-Temperaturregler mit kontinuierlichem und binärem Stellgrößen-Ausgang. Zur Verbesserung des Regelverhaltens bei Temperaturstrecken hat der Baustein eine Regelzone und eine Reduzierung des P-Anteils bei Sollwertsprüngen. Die PI/PID-Parameter kann der Baustein mittels Regleroptimierung selbst einstellen.

Die Werte in den Regelbausteinen werden nur dann korrekt berechnet, wenn der Baustein in regelmäßigen Abständen aufgerufen wird. Deshalb sollten die Regelbausteine in einem Weckalarm-OB (OB 30 ... 38) aufgerufen werden. Die Abtastzeit wird am Parameter CYCLE vorgegeben.

Sollwertzweig

Der Sollwert wird am Eingang *SP_INT* im Gleitpunktformat physikalisch oder in Prozent eingegeben. Sollwert und Istwert müssen an der Regeldifferenzbildung die gleiche Einheit besitzen.

Istwertauswahl (*PVPER_ON*)

Der Istwert kann abhängig von *PVPER_ON* im Peripherie- oder im Gleitpunktformat eingelesen werden.

PVPER_ON	Istwerteingabe
TRUE	Der Istwert wird über die Analogperipherie (PEW xxx) am Eingang *PV_PER* eingelesen.
FALSE	Der Istwert wird im Gleitpunktformat am Eingang PV_IN eingelesen.

Istwertformatumwandlung *CRP_IN* (*PER_MODE*)

Die Funktion *CRP_IN* wandelt den Peripheriewert *PV_PER* abhängig vom Schalter *PER_MODE* in ein Gleitpunktformat nach folgender Vorschrift um:

PER_MODE	Ausgang von *CRP_IN*	Analogeingabe-Typ	Einheit
0	*PV_PER* * 0.1	Thermoelemente; PT100/ NI100; Standard	°C; °F
1	*PV_PER* * 0.01	PT100/NI100; Klima	°C; °F
2	*PV_PER* * 100/27648	Spannung/Strom	%

Istwertnormierung *PV_NORM* (*PV_FAC, PV_OFFS*)

Die Funktion *PV_NORM* berechnet den Ausgang von *CRP_IN* nach folgender Vorschrift:

*Ausgang von PV_NORM = Ausgang von CPR_IN * PV_FAC + PV_OFFS*

Sie kann für folgende Zwecke eingesetzt werden:

- Istwert-Anpassung mit *PV_FAC* als Istwertfaktor und *PV_OFFS* als Istwertoffset
- Normierung von Temperatur nach Prozent
  Sie wollen den Sollwert in Prozent eingeben und müssen nun den gemessenen Temperaturwert in Prozent umrechnen.
- Normierung von Prozent nach Temperatur
  Sie wollen den Sollwert in der physikalischen Größe Temperatur eingeben und müssen nun den gemessenen Spannungs/Strom-Wert in eine Temperatur umrechnen.

Berechnung der Parameter:

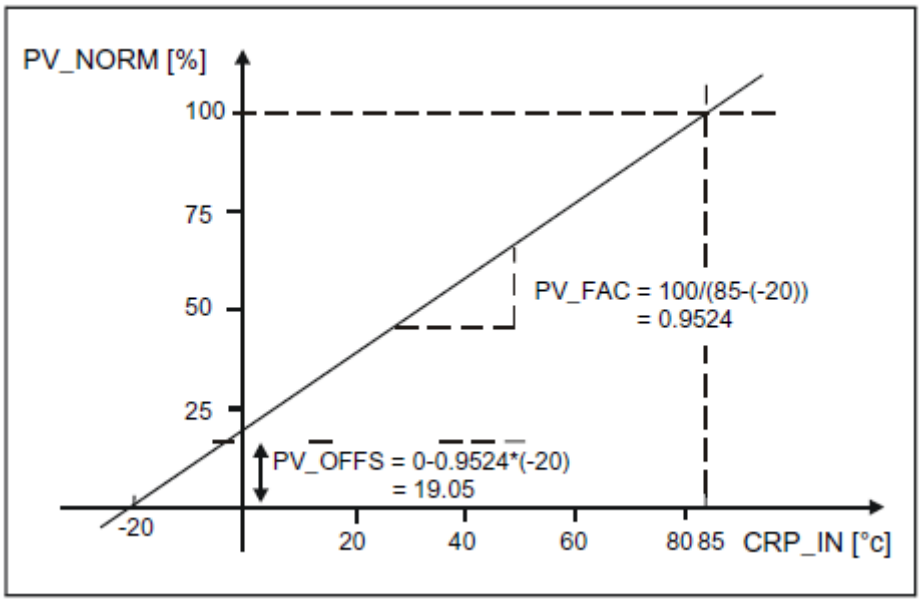
- *PV_FAC = Bereich von PV_NORM/Bereich von CRP_IN*
- *PV_OFFS = UG(PV_NORM) - PV_FAC * UG(CRP_IN)*; mit *UG*: Untergrenze

Mit den Defaultwerten (*PV_FAC* = 1.0 und *PV_OFFS* = 0.0) ist die Normierung abgeschaltet. Der effektiv wirksame Istwert wird am Ausgang *PV* ausgegeben.

Bei Impulsregelung muss der Istwert im schnellen Impulsaufruf dem Baustein übergeben werden (Grund: Mittelwertfilterung). Sonst kann sich die Regelqualität verschlechtern.

Beispiel zur Istwertnormierung

Wenn Sie den Sollwert in Prozent vorgeben wollen und Sie einen Temperaturbereich von -20 ... 85 °C an *CRP_IN* anliegen haben, müssen Sie den Temperaturbereich in Prozent umnormieren. Das folgende Bild stellt ein Beispiel für die Anpassung des Temperaturbereichs -20 ... 85 °C auf intern 0 ... 100 % dar:

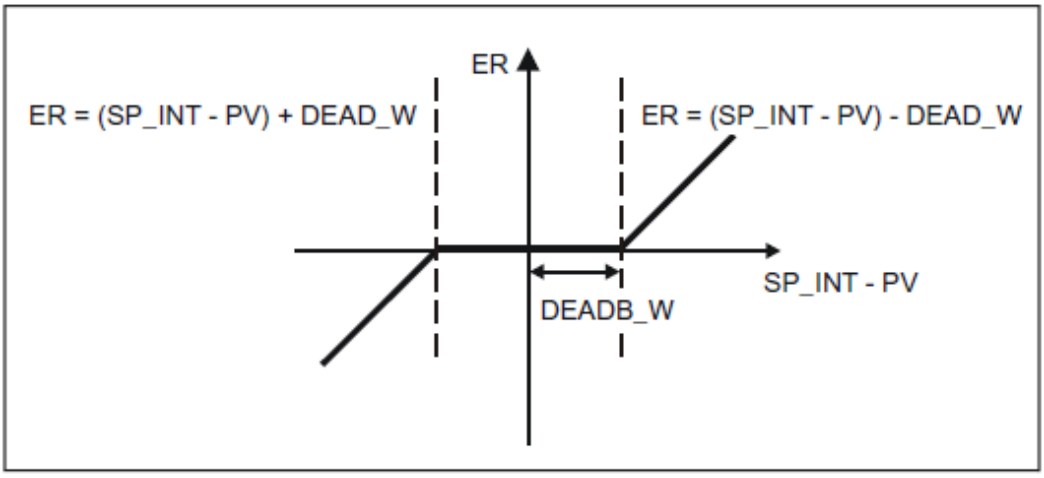


Regeldifferenzbildung

Die Differenz von Soll- und Istwert ergibt die Regeldifferenz vor der Totzone. Soll- und Istwert müssen in der gleichen Einheit vorliegen.

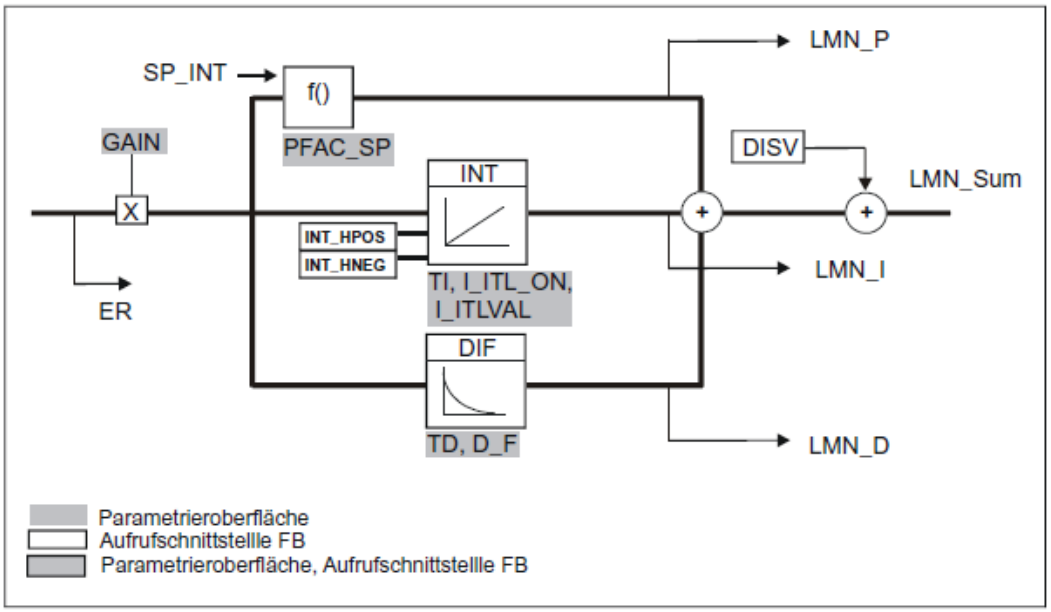
Totzone (*DEADB_W*)

Zur Unterdrückung einer kleinen Dauerschwingung aufgrund der Stellgrößen-Quantisierung (z. B. bei einer Pulsbreitenmodulation mit PULSEGEN) wird die Regeldifferenz über eine Totzone (DEADBAND) geleitet. Bei *DEADB_W* = 0.0 ist die Totzone ausgeschaltet. Die wirksame Regeldifferenz wird am Parameter *ER* angezeigt.



PID-Algorithmus

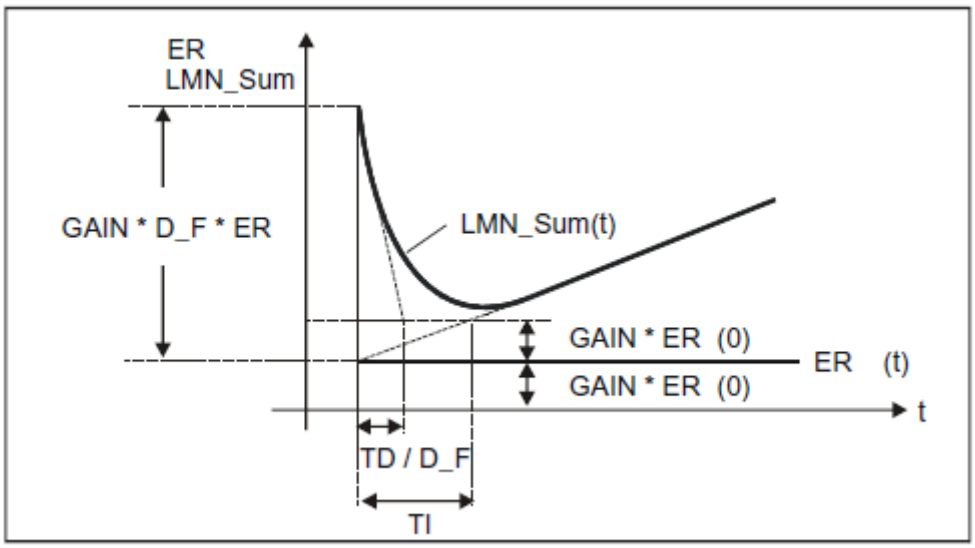
In folgendem Bild ist das Blockschaltbild des PID-Algorithmus dargestellt:



PID-Algorithmus (*GAIN, TI, TD, D_F*)

- Der PID-Algorithmus arbeitet im Stellungsalgorithmus. Der Proportional-, Integral (*INT*) und Differentialanteil (*DIF*) sind parallel geschaltet und lassen sich einzeln zu- und abschalten. Damit können Sie P-, PI-, PD- und PID-Regler parametrieren.
- Die Regleroptimierung unterstützt PI- und PID-Regler. Reglerinvertierung erfolgt über ein negatives *GAIN* (Kühlregler).
- Wenn Sie *TI* und *TD* auf 0.0 setzen erhalten Sie einen reinen P-Regler am Arbeitspunkt.

$$LMN_Sum(t) = GAIN * ER(0) \left(1 + \frac{1}{TI} * t + D_F * e^{\frac{-t}{TD/D_F}}\right)$$



LMN_Sum(t)	Stellgröße bei Automatikbetrieb des Reglers
ER (0)	Sprunghöhe der normierten Regeldifferenz
GAIN	Reglerverstärkung
TI	Integrationszeit
TD	Differenzierzeit
D_F	Differenzierfaktor

Integrierer (*TI, I_ITL_ON, I_ITLVAL*)

Bei Handbetrieb wird er wie folgt nachgeführt: *LMN_I = LMN - LMN_P - DISV*

Bei Begrenzung des Stellwerts wird der I-Anteil angehalten. Bei einer Regeldifferenz, die den I-Anteil in Richtung innerer Stellbereich bewegt, wird der I-Anteil wieder freigegeben.

Weitere Modifikationen des I-Anteils ergeben sich durch folgende Maßnahmen:

- Abschalten des I-Anteils des Reglers erfolgt mit *TI* = 0.0
- Abschwächung des P-Anteils bei Sollwertänderungen
- Regelzone
- Die Stellwertgrenzen lassen sich online ändern

Abschwächung des P-Anteils bei Sollwertänderungen (*PFAC_SP*)

Um Überschwingen zu vermeiden können Sie den P-Anteil über den Parameter "Proportionalfaktor bei Sollwertänderungen" (*PFAC_SP*) abschwächen. Über *PFAC_SP* können Sie zwischen 0.0 und 1.0 kontinuierlich wählen, wie stark der P-Anteil bei Sollwertänderungen wirken soll:

- *PFAC_SP* = 1.0: P-Anteil bei Sollwertänderung voll wirksam
- *PFAC_SP* = 0.0: P-Anteil bei Sollwertänderung nicht wirksam

Die Abschwächung des P-Anteils wird durch eine Kompensation am I-Anteil erreicht.

Differenzierer (*TD, D_F*)

- Abschalten des D-Anteils des Reglers erfolgt mit *TD* = 0.0.
- Bei zugeschaltetem D-Anteil sollte folgende Gleichung eingehalten werden:
  TD = 0.5 * *CYCLE* * *D_F*

Parametrierung eines P- oder PD-Reglers mit Arbeitspunkt

Schalten Sie in der Parametrieroberfläche den I-Anteil (TI = 0.0) und evtl. den D-Anteil (TD = 0.0) ab. Führen Sie außerdem folgende Parametrierung durch:

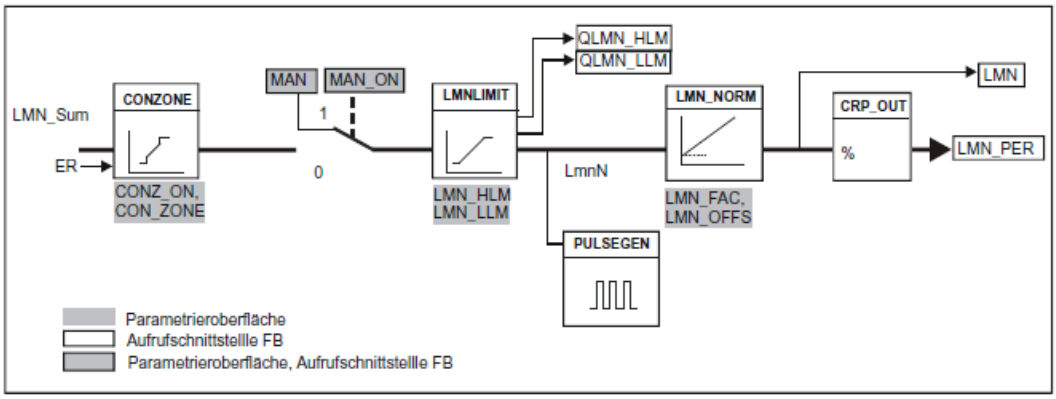
- *I_ITL_ON* = TRUE
- *I_ITLVAL* = Arbeitspunkt

Störgrößenaufschaltung (*DISV*)

Am Eingang *DISV* können Sie eine Störgröße additiv aufschalten.

Stellwertberechnung

In folgendem Bild ist das Blockschaltbild der Stellwertberechnung dargestellt:

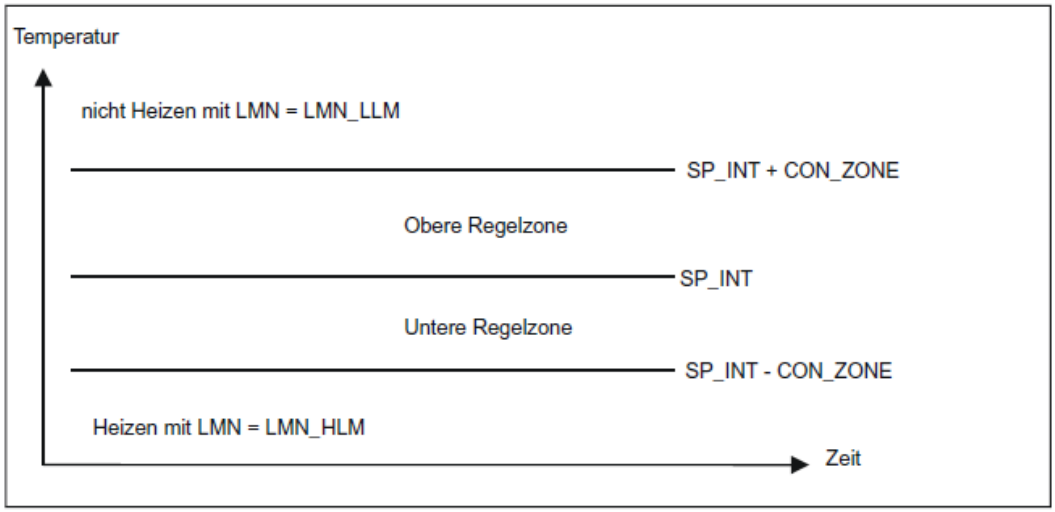


Regelzone (*CONZ_ON, CON_ZONE*)

Wenn *CONZ_ON* = TRUE arbeitet der Regler mit einer Regelzone. Das bedeutet, dass der Regler nach folgendem Algorithmus angesteuert wird:

- Übersteigt der Istwert *PV* den Sollwert *SP_INT* um mehr als *CON_ZONE*, wird als Stellgröße der Wert *LMN_LLM* ausgegeben (gesteuerter Regelbetrieb).
- Unterschreitet der Istwert *PV* den Sollwert *SP_INT* um mehr als *CON_ZONE*, wird *LMN_HLM* ausgegeben (gesteuerter Regelbetrieb).
- Bewegt sich der Istwert *PV* innerhalb der Regelzone (*CON_ZONE*), nimmt der Stellwert den Wert vom PID-Algorithmus *LMN_Sum* an (automatischer Regelbetrieb).

Der Übergang vom gesteuerten Regelbetrieb zum automatischen Regelbetrieb erfolgt unter Einhaltung einer Hysterese von 20% der Regelzone.



Bevor Sie die Regelzone von Hand einschalten müssen Sie sicherstellen, dass die Regelzonenbreite nicht zu klein eingestellt ist. Bei zu klein eingestellter Regelzonenbreite treten Schwingungen im Verlauf der Stellgröße und des Istwertverlaufs auf.

Vorteil der Regelzone

Beim Eintritt in die Regelzone führt der zugeschaltete D-Anteil zu einem sehr schnellen Reduzieren der Stellgröße. Daher ist die Regelzone nur bei eingeschaltetem D-Anteil sinnvoll. Ohne Regelzone würde im wesentlichen nur der sich reduzierende P-Anteil die Stellgröße reduzieren. Die Regelzone führt zu einem schnelleren Einschwingen ohne Über-/Unterschwingen, wenn die ausgegebene minimale oder maximale Stellgröße weit von der für den neuen Arbeitspunkt stationär notwendigen Stellgröße entfernt ist.

Handwertverarbeitung (*MAN_ON, MAN*)

Sie können zwischen Hand- und Automatikbetrieb umschalten. Bei Handbetrieb wird die Stellgröße einem Handwert nachgeführt. Der Integrierer (*INT*) wird intern auf *LMN - LMN_P - DISV* und der Differenzierer (*DIF*) auf 0 gesetzt und intern abgeglichen. Das Umschalten in den Automatikbetrieb ist damit stoßfrei.

Während einer Optimierung ist der Parameter MAN_ON nicht wirksam.

Stellwertbegrenzung *LMNLIMIT* (*LMN_HLM, LMN_LLM*)

Der Stellwert wird mit der Funktion *LMNLIMIT* auf die Stellwertgrenzen *LMN_HLM* und *LMN_LLM* begrenzt. Das Erreichen der Grenzen wird durch die Meldebits *QLMN_HLM* und *QLMN_LLM* angezeigt. Bei Begrenzung des Stellwerts wird der I-Anteil angehalten. Bei einer Regeldifferenz, die den I-Anteil in Richtung innerer Stellbereich bewegt, wird der I-Anteil wieder freigegeben.

Stellwertgrenzen online ändern

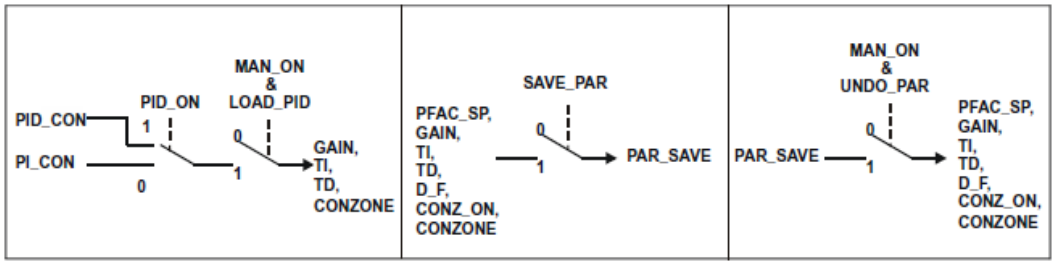
Wird der Stellwertbereich verringert und der neue unbegrenzte Stellwert liegt außerhalb der Grenzen, wird der I-Anteil und damit der Stellwert verschoben. Der Stellwert wird um die Änderung der Stellwertgrenze verringert. War der Stellwert vor der Änderung unbegrenzt, wird er genau auf die neue Grenze gesetzt (hier beschrieben für die obere Stellwertgrenze).

Stellwertnormierung *LMN_NORM* (*LMN_FAC, LMN_OFFS*)

- Die Funktion *LMN_NORM* normiert den Stellwert nach folgender Vorschrift:
  *LMN = LmnN * LMN_FAC + LMN_OFFS*
- Sie kann für folgende Zwecke eingesetzt werden:
  Stellwert-Anpassung mit *LMN_FAC* als Stellwertfaktor und *LMN_OFFS* als Stellwertoffset
- Der Stellwert steht auch im Peripherieformat zur Verfügung. Die Funktion CRP_OUT wandelt den Gleitpunktwert *LMN* in einen Peripheriewert nach folgender Vorschrift um:
  *LMN_PER = LMN * 27648/100*
  Mit den Defaultwerten (*LMN_FAC* = 1.0 und *LMN_OFFS* = 0.0) ist die Normierung abgeschaltet. Der effektiv wirksame Stellwert wird am Ausgang *LMN* ausgegeben.

Reglerparameter speichern und zurückladen

In folgendem Bild ist das Blockschaltbild dargestellt:



Reglerparameter speichern *SAVE_PAR*

Wenn Sie die aktuellen Reglerparameter als brauchbar einstufen, können Sie diese vor einer manuellen Änderung in eigens dafür vorgesehenen Strukturparametern im Instanz-DB des FB 58 speichern. Bei einer Regleroptimierung werden die gespeicherten Parameter durch die vor der Optimierung gültigen Werte überschrieben. *PFAC_SP, GAIN, TI, TD, D_F, CONZ_ON* und *CON_ZONE* werden in die Struktur PAR_SAVE geschrieben.

Gespeicherte Reglerparameter zurückladen *UNDO_PAR*

Die zuletzt gespeicherten Reglerparameter können mit dieser Funktion wieder für den Regler aktiviert werden (nur im Handbetrieb).

Wechsel zwischen PI- und PID-Parametern *LOAD_PID* (*PID_ON*)

Nach einer Optimierung werden die PI- und PID-Parameter in den Strukturen *PI_CON* und *PID_CON* hinterlegt. Mit *LOAD_PID* in Abhängigkeit von *PID_ON* können Sie im Handbetrieb die PI- bzw. PID-Parameter auf die wirksamen Reglerparameter schreiben.

PID-Parameter *PID_ON* = TRUE		PI-Parameter PID_ON = FALSE	
GAIN	= PID_CON.GAIN	GAIN	= PI_CON.GAIN
TI	= PID_CON.TI	TI	= PI_CON.TI
TD	= PID_CON.TD		

- *Die Reglerparameter werden mit UNDO_PAR oder LOAD_PID nur dann zurückgeschrieben, wenn die Reglerverstärkung ungleich Null ist:*
  Bei LOAD_PID werden die Parameter nur kopiert, falls das jeweiligen GAIN <> 0 ist (entweder vom PI- oder PID-Parametersatz). Damit ist der Fall berücksichtigt, dass noch keine Optimierung durchgeführt wurde bzw. PID-Parameter fehlen. War PID_ON = TRUE und PID.GAIN = FALSE, wird PID_ON auf FALSE gesetzt und die PI-Parameter kopiert.
- *D_F, PFAC_SP werden durch die Optimierung voreingestellt. Sie können anschließend vom Anwender modifiziert werden. LOAD_PID verändert diese Parameter nicht.*
- *Die Regelzone wird bei LOAD_PID immer neu berechnet (CON_ZONE = 250/GAIN), auch wenn CONZ_ON = FALSE.*

17.5.5 FB 59 - TCONT_S - Temperatur-Schrittregeln

Beschreibung

Der FB 59 TCONT_S dient zum Regeln von technischen Temperatur-Prozessen mit binären Stellwertausgangssignalen für integrierende Stellglieder. Über die Parametrierung lassen sich Teilfunktionen des PI-Schrittreglers zu- oder abschalten und damit an die Regelstrecke anpassen.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
CYCLE	INPUT	REAL	SAMPLE TIME OF STEP CONTROLLER [s] / Abtastzeit des Schrittreglers [s] ■ An diesem Eingang *CYCLE* geben Sie die Abtastzeit für den Regler ein. ■ Default: 0.0 ■ Wertebereich: ≥ 0.001
SP_INT	INPUT	REAL	INTERNAL SETPOINT / Interner Sollwert ■ Der Eingang *SP_INT* dient zur Vorgabe eines Sollwertes. ■ Default: 0.0 ■ Wertebereich: Abhängig von den eingesetzten Sensoren
PV_IN	INPUT	REAL	PROCESS VARIABLE IN / Istwert Eingang ■ Am Eingang *PV_IN* kann ein Inbetriebsetzungswert parametriert oder ein externer Istwert im Gleitpunktformat verschaltet werden. ■ Default: 0.0 ■ Wertebereich: Abhängig von den eingesetzten Sensoren

Parameter	Deklaration	Datentyp	Beschreibung
PV_PER	INPUT	WORD	PROCESS VARIABLE PERIPHERY / Istwert Peripherie ■ Der Istwert in Peripherieformat wird am Eingang *PV_PER* mit dem Regler verschaltet. ■ Default: 0
DISV	INPUT	REAL	DISTURBANCE VARIABLE / Störgröße ■ Für eine Störgrößenaufschaltung wird die Störgröße am Eingang *DISV* verschaltet. ■ Default: 0.0
LMNR_HS	INPUT	BOOL	HIGH LIMIT SIGNAL OF REPEATED MANIPULATED VALUE / Oberes Anschlagsignal der Stellungsrückmeldung ■ Das Signal "Stellventil am oberen Anschlag" wird am Eingang *LMNR_HS* verschaltet. ■ *LMNR_HS* = TRUE: Das Stellventil befindet sich am oberen Anschlag. ■ Default: FALSE
LMNR_LS	INPUT	BOOL	LOW LIMIT SIGNAL OF REPEATED MANIPULATED VALUE / Unteres Anschlagsignal der Stellungsrückmeldung ■ Das Signal "Stellventil am unteren Anschlag" wird am Eingang *LMNR_LS* verschaltet. ■ *LMNR_LS* = TRUE: Das Stellventil befindet sich am unteren Anschlag. ■ Default: FALSE
LMNS_ON	INPUT	BOOL	MANIPULATED SIGNALS ON / Handbetrieb der Stellwertsignale einschalten ■ Am Eingang *LMNS_ON* wird die Stellwertsignalverarbeitung auf Hand geschaltet. ■ Default: TRUE
LMNUP	INPUT	BOOL	MANIPULATED SIGNALS UP / Stellwertsignal Hoch ■ Bei Handbetrieb der Stellwertsignale wird am Eingang *LMNUP* das Ausgangssignal *QLMNUP* bedient. ■ Default: FALSE
LMNDN	INPUT	BOOL	MANIPULATED SIGNALS DOWN / Stellwertsignal Tief ■ Bei Handbetrieb der Stellwertsignale wird am Eingang *LMNDN* das Ausgangssignal *QLMNDN* bedient. ■ Default: FALSE
QLMNUP	OUTPUT	BOOL	MANIPULATED SIGNAL UP / Stellwertsignal Hoch ■ Ist der Ausgang *QLMNUP* gesetzt, soll das Stellventil geöffnet werden. ■ Default: FALSE
QLMNDN	OUTPUT	BOOL	MANIPULATED SIGNAL DOWN / Stellwertsignal Tief ■ Ist der Ausgang *QLMNDN* gesetzt, soll das Stellventil geschlossen werden. ■ Default: FALSE

Parameter	Deklaration	Datentyp	Beschreibung
PV	OUTPUT	REAL	PROCESS VARIABLE / Istwert ■ Am Ausgang *PV* wird der effektiv wirkende Istwert ausgegeben. ■ Default: 0.0
PE	OUTPUT	REAL	ERROR SIGNAL / Regeldifferenz ■ Am Ausgang *PE* wird die effektiv wirkende Regeldifferenz ausgegeben. ■ Default: 0.0
COM_RST	INPUT/ OUTPUT	BOOL	COMPLETE RESTART / Neustart ■ Der Baustein hat eine Initialisierungsroutine, die bearbeitet wird, wenn der Eingang *COM_RST* gesetzt ist. ■ Default: FALSE

Interne Parameter

Parameter	Deklaration	Datentyp	Beschreibung
PV_FAC	INPUT	REAL	PROCESS VARIABLE FACTOR / Istwertfaktor ■ Der Eingang *PV_FAC* wird mit dem Istwert multipliziert. Der Eingang dient zur Anpassung des Istwertbereiches. ■ Default: 1.0
PV_OFFS	INPUT	REAL	PROCESS VARIABLE OFFSET / Istwertoffset ■ Der Eingang *PV_OFFS* wird mit dem Istwert addiert. Der Eingang dient zur Anpassung des Istwertbereiches. ■ Default: 0.0 ■ Wertebereich: Abhängig von den eingesetzten Sensoren
DEADB_W	INPUT	REAL	DEAD BAND WIDTH / Totzonenbreite ■ Die Regeldifferenz wird über eine Totzone geführt. Der Eingang *DEADB_W* bestimmt die Größe der Totzone. ■ Default: 0.0 ■ Wertebereich: Abhängig von den eingesetzten Sensoren
PFAC_SP	INPUT	REAL	PROPORTIONAL FACTOR FOR SETPOINT CHANGES [0..1] / Proportionalfaktor bei Sollwertänderungen ■ *PFAC_SP* gibt den wirksamen P-Anteil bei Sollwertänderung an. Er wird zwischen 0 und 1 eingestellt. – 1: P-Anteil ist bei Sollwertänderungen voll wirksam – 0: P-Anteil ist bei Sollwertänderungen nicht wirksam ■ Default: 1.0 ■ Wertebereich: 0.0 ... 1.0

Parameter	Deklaration	Datentyp	Beschreibung
GAIN	INPUT	REAL	PROPORTIONAL GAIN / Reglerverstärkung ■ Der Eingang *GAIN* gibt die Reglerverstärkung an. Eine Invertierung des Regelsinns wird durch das negative Vorzeichen von GAIN erreicht. ■ Default: 2.0 ■ Wertebereich: %/phys. Einheit
TI	INPUT	REAL	RESET TIME [s] / Integrationszeit [s] ■ Der Eingang *TI* (Nachstellzeit) bestimmt das Zeitverhalten des Integrierers. ■ Default: 40.0 s ■ Wertebereich: ≥ 0.0 s
MTR_TM	INPUT	REAL	MOTOR ACTUATING TIME / Motorstellzeit [s] ■ Am Parameter *MTR_TM* wird die Laufzeit des Stellventils vom Anschlag zu Anschlag eingetragen. ■ Default: 30 s ■ Wertebereich: ≥ *CYCLE*
PULSE_TM	INPUT	REAL	MINIMUM PULSE TIME / Mindestimpulsdauer [s] ■ Am Parameter *PULSE_TM* kann eine minimale Impulslänge parametriert werden. ■ Default: 0.1s ■ Wertebereich: ≥ 0.0 s
BREAK_TM	INPUT	REAL	MINIMUM BREAK TIME / Mindestpausendauer [s] ■ Am Parameter *BREAK_TM* kann eine minimale Pausenlänge parametriert werden. ■ 0.1s ■ Wertebereich: ≥ 0.0 s
PER_MODE	INPUT	INT	PERIPHERIE MODE / Peripherie Betriebsart ■ An diesem Schalter können Sie den Typ der AE-Baugruppe eingeben. Der Istwert am Eingang *PV_PER* wird dadurch am Ausgang *PV* in °C normiert. – *PER_MODE* = 0: Standard – *PER_MODE* = 1: Klima – *PER_MODE* = 2: Strom/Spannung ■ Default: 0 ■ Wertebereich: 0, 1, 2
PVPER_ON	INPUT	BOOL	PROCESS VARIABLE PERIPHERY ON / Istwert Peripherie einschalten ■ Soll der Istwert von der Peripherie eingelesen werden, so muss der Eingang *PV_PER* mit der Peripherie verschaltet werden und der Eingang *PVPER_ON* gesetzt werden. ■ Default: FALSE

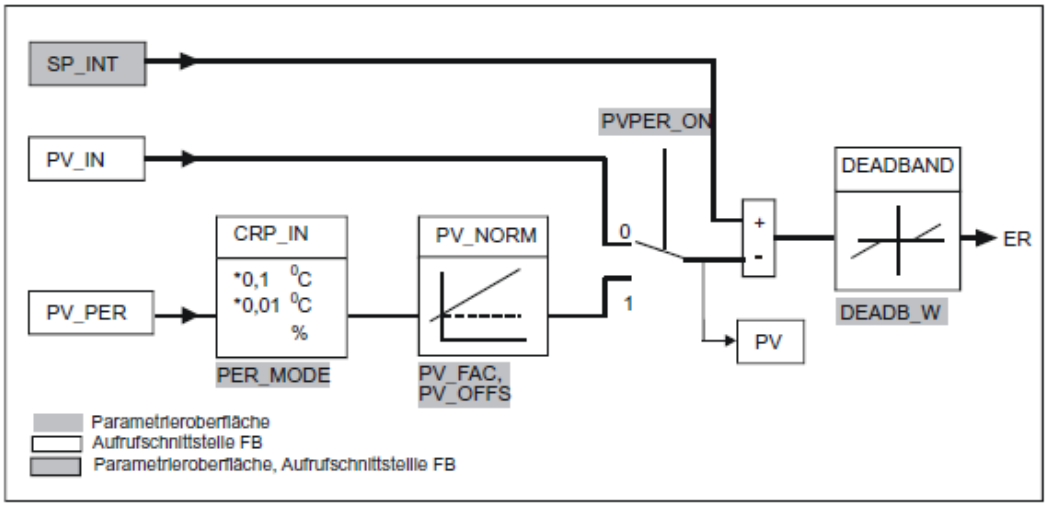
Anwendung

- Die Arbeitsweise basiert auf dem PI-Regelalgorithmus des Abtastreglers. Dieser wird um die Funktionsglieder zur Erzeugung des binären Ausgangssignals aus dem analogen Stellsignal ergänzt.
- Sie können den Regler auch in einer Reglerkaskade als unterlagerten Stellungsregler einsetzen. Über den Sollwerteingang *SP_INT* geben Sie die Stellgliedposition vor. In diesem Fall müssen Sie den Istwerteingang und den Parameter *TI* (Integrationszeit) auf Null setzen. Anwendungsfall ist z. B. eine Temperaturregelung mit Heizleistungsregelung über Puls-Pause-Ansteuerung und Kühlleistungsregelung über eine Ventilklappe. Um die Klappe ganz zu schließen, sollte die Stellgröße (*ER * GAIN*) negativ werden.
- Neben den Funktionen im Istwertzweig realisiert der FB 59 TCONT_S einen fertigen PI-Regler mit binärem Stellwertausgang und Beeinflussungsmöglichkeit der Stellwertsignale von Hand. Der Schrittregler arbeitet ohne Stellungsrückmeldung.

Die Werte in den Regelbausteinen werden nur dann korrekt berechnet, wenn der Baustein in regelmäßigen Abständen aufgerufen wird. Deshalb sollten die Regelbausteine in einem Weckalarm-OB (OB 30 ... 38) aufgerufen werden. Die Abtastzeit wird am Parameter CYCLE vorgegeben.

Regeldifferenzbildung

Blockschaltbild



Sollwertzweig

Der Sollwert wird am Eingang *SP_INT* im Gleitpunktformat physikalisch oder in Prozent eingegeben. Sollwert und Istwert müssen an der Regeldifferenzbildung die gleiche Einheit besitzen.

Istwertauswahl (*PVPER_ON*)

Der Istwert kann abhängig von *PVPER_ON* im Peripherie- oder im Gleitpunktformat eingelesen werden.

PVPER_ON	**Istwerteingabe**
TRUE	Der Istwert wird über die Analogperipherie (PEW xxx) am Eingang *PV_PER* eingelesen.
FALSE	Der Istwert wird im Gleitpunktformat am Eingang *PV_IN* eingelesen.

Istwertformatumwandlung *CRP_IN* (*PER_MODE*)

Die Funktion *CRP_IN* wandelt den Peripheriewert *PV_PER* abhängig vom Schalter *PER_MODE* in ein Gleitpunktformat nach folgender Vorschrift um:

PER_MODE	Ausgang von *CRP_IN*	Analogeingabe-Typ	Einheit
0	*PV_PER* * 0.1	Thermoelemente; PT100/ NI100; Standard	°C; °F
1	*PV_PER* * 0.01	PT100/NI100; Klima	°C; °F
2	*PV_PER* * 100/27648	Spannung/Strom	%

Istwertnormierung *PV_NORM* (*PF_FAC, PV_OFFS*)

Die Funktion *PV_NORM* berechnet den Ausgang von *CRP_IN* nach folgender Vorschrift:

*Ausgang von PV_NORM = Ausgang von CPR_IN * PV_FAC + PV_OFFS*

Sie kann für folgende Zwecke eingesetzt werden:

- Istwert-Anpassung mit *PV_FAC* als Istwertfaktor und *PV_OFFS* als Istwertoffset
- Normierung von Temperatur nach Prozent
  Sie wollen den Sollwert in Prozent eingeben und müssen nun den gemessenen Temperaturwert in Prozent umrechnen.
- Normierung von Prozent nach Temperatur
  Sie wollen den Sollwert in der physikalischen Größe Temperatur eingeben und müssen nun den gemessenen Spannungs/Strom-Wert in eine Temperatur umrechnen.

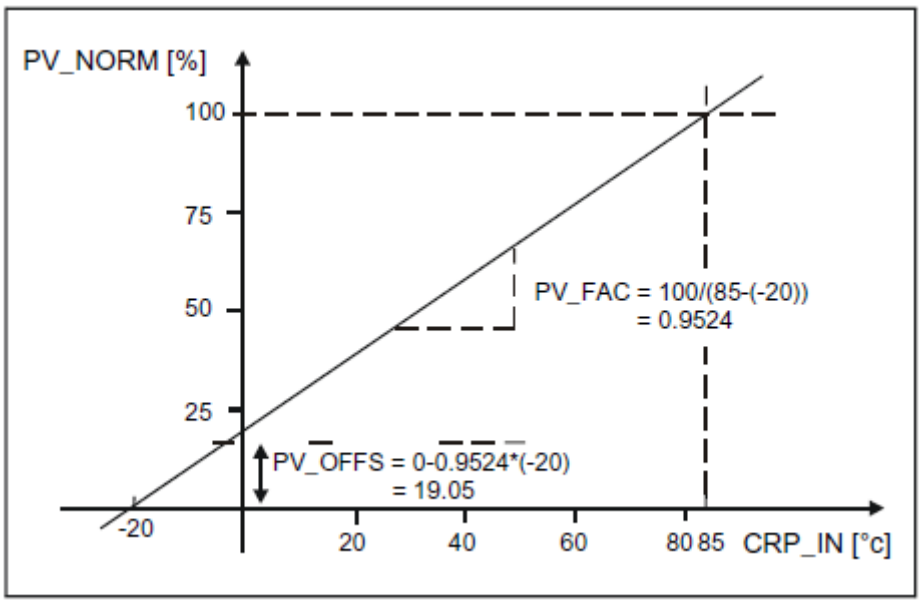
Berechnung der Parameter:

- *PV_FAC = Bereich von PV_NORM / Bereich von CRP_IN*
- *PV_OFFS = UG(PV_NORM) - PV_FAC * UG(CRP_IN)*; mit UG: Untergrenze

Mit den Defaultwerten (*PV_FAC* = 1.0 und *PV_OFFS* = 0.0) ist die Normierung abgeschaltet. Der effektiv wirksame Istwert wird am Ausgang PV ausgegeben.

Beispiel zur Istwertnormierung

Wenn Sie den Sollwert in Prozent vorgeben wollen und Sie einen Temperaturbereich von -20 bis 85 °C an *CRP_IN* anliegen haben, müssen Sie den Temperaturbereich in Prozent umnormieren. In folgendem Bild ist die Anpassung des Temperaturbereichs von -20 ... 85 °C auf intern 0 ... 100 % dargestellt:

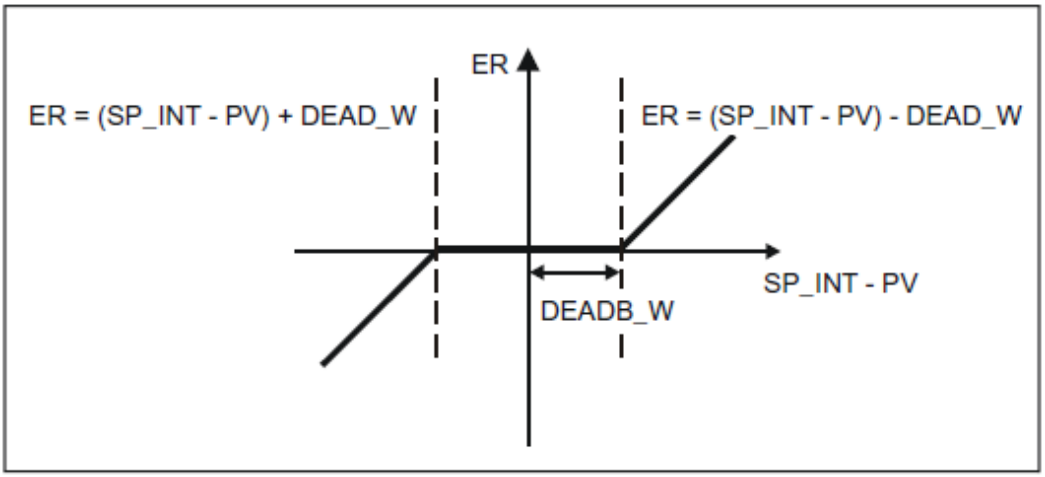


Regeldifferenzbildung

Die Differenz von Soll- und Istwert ergibt die Regeldifferenz vor der Totzone. Soll- und Istwert müssen in der gleichen Einheit vorliegen.

Totzone (*DEADB_W*)

Zur Unterdrückung einer kleinen Dauerschwingung aufgrund der Stellgrößen-Quantisierung (z. B. bei einer Pulsbreitenmodulation mit PULSEGEN) wird die Regeldifferenz über eine Totzone (DEADBAND) geleitet. Bei *DEADB_W* = 0.0 ist die Totzone ausgeschaltet. Die wirksame Regeldifferenz wird am Parameter *ER* angezeigt.



PI-Schrittregler-Algorithmus

Der FB 59 TCONT_S arbeitet ohne Stellungsrückmeldung (siehe folgendes Blockschaltbild). Der I-Anteil des PI-Algorithmus und die gedachte Stellungsrückmeldung werden in einem Integrator (INT) berechnet und als Rückführungswert mit dem verbliebenen P-Anteil verglichen. Die Differenz geht auf ein Dreipunktglied (THREE_ST) und einen Impulsformer (PULSEOUT), der die Impulse für das Stellventil bildet. Über eine Adaption der Ansprechschwelle des Dreipunktgliedes wird die Schalthäufigkeit des Reglers reduziert.

Abschwächung des P-Anteils bei Sollwertänderungen

Um Überschwingen zu vermeiden können Sie den P-Anteil über den Parameter "Proportionalfaktor bei Sollwertänderungen" (*PFAC_SP*) abschwächen. Über *PFAC_SP* können Sie nun zwischen 0.0 und 1.0 kontinuierlich wählen, wie stark der P-Anteil bei Sollwertänderungen wirken soll:

- *PFAC_SP* = 1.0: P-Anteil bei Sollwertänderung voll wirksam
- *PFAC_SP* = 0.0: Kein P-Anteil bei Sollwertänderung

Ein Wert von *PFAC_SP* < 1.0 kann wie beim kontinuierlichen Regler das Überschwingen reduzieren, falls die Motorlaufzeit *MTR_TM* klein gegenüber der Ausgleichszeit TA ist und das Verhältnis *TU/TA* < 0.2 ist. Erreicht *MTR_TM* 20 % von *TA*, ist nur noch eine geringe Verbesserung zu erzielen.

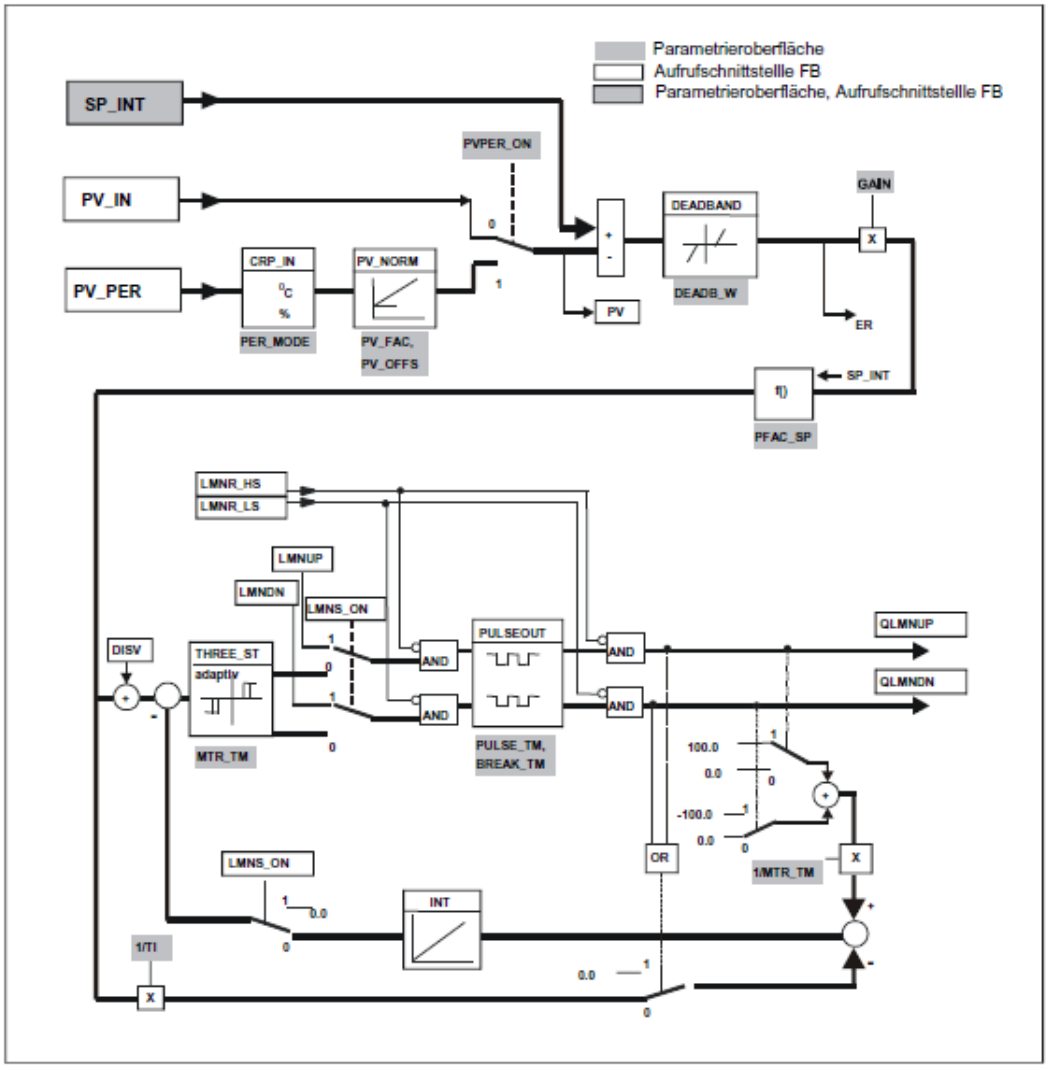
Störgrößenaufschaltung

Am Eingang *DISV* kann eine Störgröße additiv aufgeschaltet werden.

Handwertverarbeitung (*LMNS_ON*)

Mit *LMNS_ON* kann zwischen Hand- und Automatikbetrieb umgeschaltet werden. Bei Handbetrieb wird das Stellglied angehalten und der Integrierer (INT) wird intern auf 0 gesetzt. Über *LMNUP* und *LMNDN* kann das Stellglied AUF und ZU gefahren werden. Das Umschalten in den Automatikbetrieb erfolgt stoßbehaftet. Die anstehende Regeldifferenz führt über *GAIN* zu einer sprungförmigen Änderung der internen Stellgröße. Durch das integral wirkende Stellglied wird jedoch nur eine rampenförmige Ansteuerung des Prozesses bewirkt.

Blockschaltbild



17.6 Zeitfunktionen - *"Time Functions"*

17.6.1 UDT 60 - WS_RULES - Regel DB

Beschreibung

Ihr System muss in einem DB einige Informationen bereitstellen, die von verschiedenen Bausteinen ausgewertet werden. Diesen Datenbaustein erstellen Sie als DB vom Typ der UDT 60 und tragen die für Ihren Ort geltenden Werte (in Lokalzeit!) ein.

Umrechnung Basiszeit < - > Lokalzeit und "Alarm stellen nach Lokalzeit"

Name	Typ	Anfangswert	Kommentar
B2L	STRUCT		Basiszeit < - > Lokalzeit
S	INT	2	Offset Basiszeit -> Lokalzeit [30 min] im Winter zulässig: -24 .. +24
T	INT	2	Differenz Winter- und Sommerzeit [30 min] zulässig: 2

Regel für: Winterzeit -> Sommerzeit; Default: letzter Sonntag im März; 2:00 Uhr

Name	Typ	Anfangswert	Kommentar
W2S	STRUCT		In WINTERZEIT angeben!
M	BYTE	B#16#3	Monat der Umstellung
W	BYTE	B#16#9	n-tes Auftreten des Wochentags (1 = erstes, 2 = zweites,. , 9 = letztes)
D	BYTE	B#16#1	Wochentag (Sonntag = 1)
H	BYTE	B#16#2	Stunde

Regel für: Sommerzeit -> Winterzeit; Default: letzter Sonntag im Oktober, 3:00 Uhr

Name	Typ	Anfangswert	Kommentar
S2W	STRUCT		In SOMMERZEIT angeben!
M	BYTE	B#16#10	Monat der Umstellung
W	BYTE	B#16#9	n-tes Auftreten des Wochentags (1 = erstes, 2 = zweites,. , 9 = letztes)
D	BYTE	B#16#1	Wochentag (Sonntag = 1)
H	BYTE	B#16#3	Stunde

Alle Parameter die das Format BYTE haben, werden als BCD-Werte interpretiert!

Die Festlegung der Sommer-/Winterzeit-Umschaltpunkte durch eine Regel ist in der EU ab dem Jahr 2002 vorgeschrieben.

17.6.2 FC 61 - BT_LT - Umrechnung Basiszeit in Lokalzeit

Beschreibung Der FC 61 errechnet die Lokalzeit zu der am Eingang vorgegebenen Basiszeit.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
BT	INPUT	DATE_AND_TIME	Basiszeit
WS_DAT	INPUT	BLOCK_DB	Information zur Zeitzone sowie zur Sommer-/Winterzeit Umschaltung (Regel-DB)
RET_VAL	OUTPUT	INT	Fehlercode
LT	OUTPUT	DATE_AND_TIME	Lokalzeit

Arbeitsweise

Die am Eingang *BT* eingegebene Basiszeit wird mittels der in einem DB hinterlegten Daten in die Lokalzeit umgerechnet und am Ausgang *LT* ausgegeben. Der DB enthält die Anzahl an 30 Minuten-Einheiten, durch die sich Basis- und Lokalzeit unterscheiden sowie die Differenz zwischen Sommer- und Winterzeit, ebenfalls in Einheiten von 30 Minuten. (Regel-DB) Ergibt sich bei der Berechnung ein Datumsüberlauf wird dies durch einen speziellen Rückgabewert gekennzeichnet.

Aufrufende OBs

Der FC 61 BT_LT kann in jeder Ablaufebene aufgerufen werden.

Aufrufumgebung

Der FC 61 benutzt intern folgende Funktionen. Diese Funktionen müssen Sie mit den hier angegebenen Nummern in ihr Projekt laden. FC 1 (AD_DT_TM), FC 7 (DT_DAY), FC 35 (SB_DT_TM)

Ausgabewerte / Fehler

RET_VAL	LT	Beschreibung
0	Lokalzeit	Baustein fehlerfrei gelaufen
1	Lokalzeit	kein Fehler, aber Datums-sprung
8082	DT#90-01-01-0:0:0	Ungültige Daten im Regel-Datenbaustein

17.6.3 FC 62 - LT_BT - Umrechnung Lokalzeit in Basiszeit

Beschreibung

Der FC 62 errechnet die Basiszeit zu der am Eingang vorgegebenen Lokalzeit.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
LT	INPUT	DATE_AND_TIME	Lokalzeit
WS_DAT	INPUT	BLOCK_DB	Information zur Zeitzone sowie zur Sommer-/Winterzeit Umschaltung (Regel-DB)
RET_VAL	OUTPUT	INT	Fehlercode
LT	OUTPUT	DATE_AND_TIME	Lokalzeit

Arbeitsweise

Die am Eingang *LT* eingegebene Lokalzeit wird mittels der in einem DB hinterlegten Daten in die Basiszeit umgerechnet und am Ausgang *BT* ausgegeben. Der DB enthält die Anzahl an 30 Minuten-Einheiten, durch die sich Basis- und Lokalzeit unterscheiden sowie die Differenz zwischen Sommer- und Winterzeit, ebenfalls in Einheiten von 30 Minuten. (Regel-DB) Ergibt sich bei der Berechnung ein Datumsüberlauf wird dies durch einen speziellen Rückgabewert gekennzeichnet.

"Verbotene Stunde"

Bei der Umschaltung von Winter- nach Sommerzeit wird die Lokalzeit um eine Stunde vorgestellt. Das bedeutet aber, dass die dazwischen liegende Stunde nicht durchlaufen wird. Für einen Zeitpunkt *LT* innerhalb dieser Stunde wird von der FC 62 LT_BT in Sommerzeit "gedacht". Dies wird mit dem Returnwert 4 bzw. 5 gemeldet.

"Doppelte Stunde"

Bei der Umschaltung von Sommer- nach Winterzeit wird die Lokalzeit um eine Stunde zurückgestellt. Das bedeutet aber, dass diese eine Stunde zweimal durchlaufen wird. (Für ME(S)Z gilt dafür die Bezeichnung 2A und 2B). Für einen Zeitpunkt *LT* innerhalb dieser Stunde ist also eine eindeutige Abbildung auf eine Basiszeit nicht möglich. Die FC LT_BT erhält als Eingangsparameter eine *LT* und muss vor der Umrechnung in *BT* entscheiden, ob der Wert im Sommer oder im Winter liegt. Für den Fall, dass die *LT* innerhalb der doppelten Stunde liegt, wird die *LT* als Winterzeit interpretiert. Dies wird mit dem Returnwert 2 bzw. 3 gemeldet.

Aufrufende OBs

Die FC 62 LT_BT kann in jeder Ablaufebene aufgerufen werden.

Aufrufumgebung

Der FC 62 benutzt intern folgende Funktionen. Diese Funktionen müssen Sie mit den hier angegebenen Nummern in ihr Projekt laden. FC 1 (AD_DT_TM), FC 7 (DT_DAY), FC 35 (SB_DT_TM)

Ausgabewerte / Fehler

RET_VAL	LT	Beschreibung
0	Basiszeit	Baustein fehlerfrei gelaufen
1	Basiszeit	Kein Fehler, aber Datumssprung
2	Basiszeit	Die LT am Eingang liegt innerhalb der "doppelten" Stunde
3	Basiszeit	Wie 2, zusätzlich Datumssprung
4	Basiszeit	Die LT am Eingang liegt innerhalb der "verbotenen" Stunde
5	Basiszeit	Wie 4, zusätzlich Datumssprung
8082	DT#90-01-01-0:0:0	Ungültige Daten im Regel-Datenbaustein

17.6.4 FC 63 - S_LTINT - Einstellen Uhrzeitalarm in Lokalzeit

Beschreibung

Der FC 63 stellt den gewünschten Uhrzeitalarm zum vorgegebenen Zeitpunkt. Dieser Zeitpunkt wird in Lokalzeit angegeben.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
OB_NR	INPUT	INT	Nr. des zu startenden OB (zul. 10 – 17)
SDT	INPUT	BLOCK_DB	Startdatum und Uhrzeit in Lokalzeit (siehe SFC 28)

Parameter	Deklaration	Datentyp	Beschreibung
PERIOD	INPUT	INT	Periode vom Ausgangspunkt SDT an: ■ W#16#0000 = Einmal ■ W#16#0201 = Minütlich ■ W#16#0401 = Stündlich ■ W#16#1001 = Täglich ■ W#16#1201 = Wöchentlich ■ W#16#1401 = Monatlich ■ W#16#1801 = Jährlich ■ W#16#2001 = Monatsende
WS_DAT	INPUT	DATE_AND_TIME	Information zur Zeitzone sowie zur Winter-/ Sommerzeit Umschaltung (s.o.)
RET_VAL	OUTPUT	INT	Fehlercode

Arbeitsweise

Die am Eingang *LT* eingegebene Lokalzeit wird mittels der in einem DB hinterlegten Regel in die Basiszeit umgerechnet. Der DB enthält die Anzahl an 30 Minuten-Einheiten, durch die sich Basis- und Lokalzeit unterscheiden sowie die Differenz zwischen Sommer- und Winterzeit, ebenfalls in Einheiten von 30 Minuten. (s.u.) Mit der errechneten Basiszeit wird der angegebene Uhrzeit-Alarm-OB parametriert und aktiviert. Ergibt sich bei der Berechnung ein Datumsüberlauf wird dies durch einen speziellen Rückgabewert gekennzeichnet.

"Verbotene Stunde"

Bei der Umschaltung von Winter- nach Sommerzeit wird die Lokalzeit um eine Stunde vorgestellt. Das bedeutet aber, dass die dazwischen liegende Stunde nicht durchlaufen wird. Für einen Zeitpunkt *LT* innerhalb dieser Stunde wird von der FC S_LTINT in Sommerzeit "gedacht". Dies wird mit dem Returnwert (RET_VAL) 4 bzw. 5 gemeldet.

"Doppelte Stunde"

Bei der Umschaltung von Sommer- nach Winterzeit wird die Lokalzeit um eine Stunde zurückgestellt. Das bedeutet aber, dass diese eine Stunde zweimal durchlaufen wird. (Für ME(S)Z gilt dafür die Bezeichnung 2A und 2B). Für einen Zeitpunkt *LT* innerhalb dieser Stunde ist also eine eindeutige Abbildung auf eine Basiszeit nicht möglich. Die FC S_LTINT erhält als Eingangsparameter eine LT und muss vor der Umrechnung in *BT* entscheiden, ob der Wert im Sommer oder im Winter liegt. Für den Fall, dass die *LT* innerhalb der doppelten Stunde liegt, wird die *LT* als Winterzeit interpretiert. Dies wird mit dem Returnwert (RET_VAL) 2 bzw. 3 gemeldet.

Aufrufende OBs

Der FC 63 S_LTINT kann in jeder Ablaufebene aufgerufen werden.

Aufrufumgebung

Der FC 63 S_LTINT benutzt intern folgende Funktionen. Diese Funktionen müssen Sie mit den hier angegebenen Nummern in ihr Projekt laden. FC 7 (DT_DAY), FC 35 (SB_DT_TM)

Ausgabewerte / Fehler

RET_VAL	Beschreibung
0	Baustein fehlerfrei gelaufen
1	Kein Fehler, aber Datumssprung
2	Die LT am Eingang war innerhalb der "doppelten" Stunde
3	Wie 2, zusätzlich Datumssprung

RET_VAL	Beschreibung
4	Die LT am Eingang liegt innerhalb der "verbotenen" Stunde
5	Wie 4, zusätzlich Datumssprung
8082	Ungültige Daten im Regel-Datenbaustein
8090	Fehlerhafter Parameter OB_NR
8091	Fehlerhafter Parameter SDT
8092	Fehlerhafter Parameter PERIOD
80A1	Der eingestellte Startzeitpunkt liegt in der Vergangenheit
80A2	OB ist nicht geladen
80A3	OB kann nicht gestartet werden

18 Systembausteine - *"System Blocks"*

18.1 Fetch/Write - *"Fetch/Write Communication"*

18.1.1 SFC 228 - RW_KACHEL - Kacheldirektzugriff

Beschreibung

Über diesen SFC haben Sie direkten Zugriff auf den 4kByte großen Kachelbereich der CPU. Der Kachelbereich verteilt sich auf 4 Kacheln mit einer Größe von jeweils 1kByte. Durch Angabe von Kachel-Nr., -Offset und Datenbreite haben Sie über den SFC 228 schreibenden und lesenden Zugriff auf einen gewünschten Kachelbereich.

Dieser SFC wurde zu Testzwecken und zum Aufbau proprietärer Kommunikationssysteme entwickelt, und steht dem Anwender uneingeschränkt zur Verfügung. Bitte beachten, dass Sie durch einen schreibenden Zugriff auf einen Kachelbereich direkt in eine Kommunikation eingreifen können!

Parameter

Name	Deklaration	Typ	Beschreibung
K_NR	IN	INT	Kachelnummer
OFFSET	IN	INT	Kacheloffset
R_W	IN	INT	Zugriff
SIZE	IN	INT	Datenbreite
RET_VAL	OUT	BYTE	Rückgabewert (0 = OK)
VALUE	IN_ OUT	ANY	Zeiger auf Bereich für Datentransfer

K_NR

Kachel-Nr.

- Geben Sie hier die Kachel-Nr. an, auf die Sie zugreifen möchten.
  - Wertebereich: 0 ... 3

OFFSET

Kachel-Offset

- Geben Sie hier einen Offset innerhalb der spezifizierten Kachel an.
  - Wertebereich: 0 ... 1023

R_W

Read/Write

- Über diesen Parameter spezifizieren Sie einen Lese- bzw. Schreibzugriff.
  - 0 = Lesezugriff
  - 1 = Schreibzugriff

SIZE

Größe

- Hiermit bestimmen Sie die Breite des Datenfelds, das Sie über *K_NR* und *OFFSET* definiert haben. Sie können die Werte 1, 2 und 4Byte einstellen.

RET_VAL (Rückgabewert)

Byte, in das eine Fehlermeldung zurückgeliefert wird.

VALUE

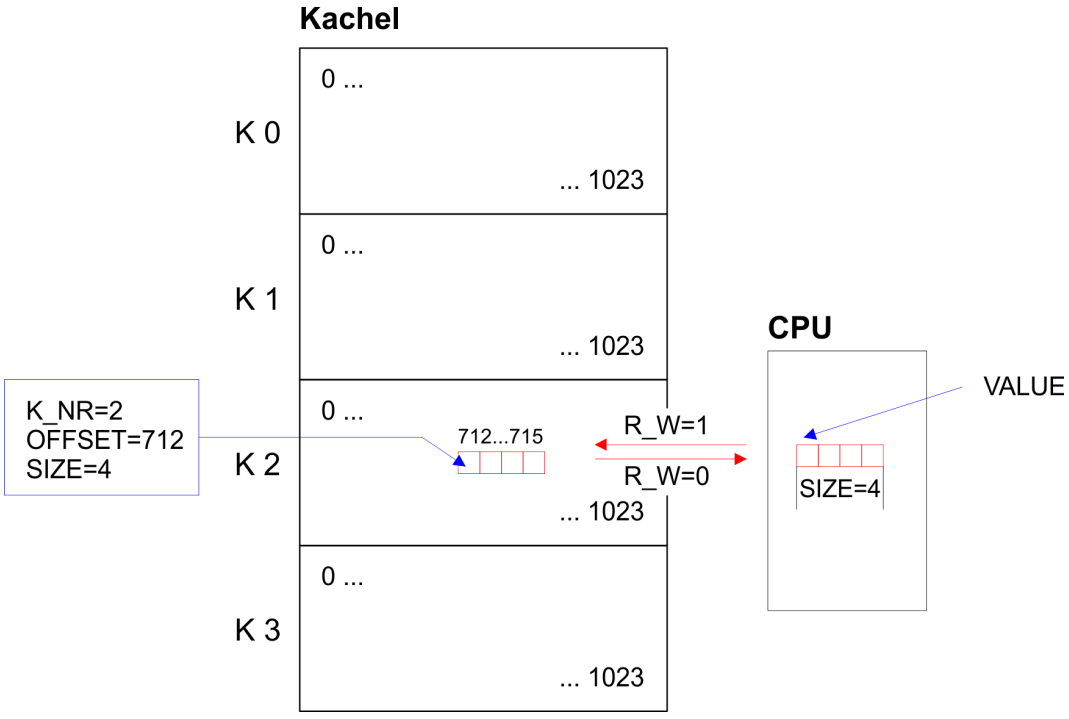
Ein-/Ausgabe-Bereich

- Mit diesem Parameter spezifizieren Sie den Ein- bzw. Ausgabebereich für den Datentransfer.
- Bei einem Lesezugriff finden in dem bis zu 4Byte breiten Bereich die Daten, die aus dem Kachelbereich gelesen werden.
- Bei einem Schreibzugriff werden aus diesem Bereich die bis zu 4Byte breiten Daten in den Kachelbereich übertragen.
  – Parameterart: Zeiger

Beispiel

Das nachfolgende Beispiel zeigt den lesenden Zugriff auf 4Bytes ab Byte 712 in Kachel 2. Die gelesenen 4Byte werden in DB10 ab Byte 2 abgelegt. Hierzu ist folgender Aufruf erforderlich:

```
CALL SFC 228
K_NR    :=2
OFFSET  :=712
R_W     :=0
SIZE    :=4
RET_VAL :=MB10
VALUE   :=P#DB10.DBX 2.0 Byte 4
```



Fehlermeldungen

Wert	Bedeutung
00h	kein Fehler aufgetreten
01h ... 05h	Interner Fehler: Für einen Parameter konnte keine gültige Adresse zugeordnet werden.
06h	die angegebene Kachel ist nicht vorhanden
07h	der Parameter SIZE ≠ 1, 2 oder 4 beim Lesezugriff

Wert	Bedeutung
08h	der Parameter SIZE ≠ 1, 2 oder 4 beim Schreibzugriff
09h	der Parameter R_W ist ≠ 0 oder 1

18.1.2 SFC 230 ... 238 - Kachelkommunikation

18.1.2.1 Paramenterbeschreibung

Übersicht

Durch die Hantierungsbausteine wird der Einsatz von Kommunikationsprozessoren in den VIPA-CPUs ermöglicht. Die Hantierungsbausteine steuern den gesamten Datenaustausch zwischen der CPU und den CPs. Vorteile der Hantierungsbausteine:

- wenig Anwenderprogrammspeicherplatz geht verloren
- kurze Laufzeiten der Bausteine

Die Hantierungsbausteine benötigen keine:

- Merkerbereiche
- Zeitbereiche
- Zählerbereiche

Alle nachfolgend behandelten Hantierungsbausteine haben eine einheitliche Schnittstelle zum Anwenderprogramm die folgende Parameter verwenden:

SSNR - Schnittstellennummer

ANR - Auftragsnummer

ANZW - Anzeigenwort (Doppelwort)

IND - Indirekte Angabe der relativen Anfangsadresse der Datenquelle bzw. des Datenziels

QANF/ZANF - Relative Anfangsadresse innerhalb des Typs

PAFE - Parametrierungsfehler

BLGR - Blockgröße

SSNR

Schnittstellen-Nummer

- Nummer der logischen Schnittstelle (Kacheladresse) auf die sich der betreffende Auftrag bezieht.
  - Parameterart: Integer
  - Sinnvoller Bereich: 0 ... 255

ANR

Auftragsnummer

- Angesprochene Auftragsnummer für die logische Schnittstelle.
  - Parameterart: Integer
  - Sinnvoller Bereich: 1 ... 223

ANZW

Anzeigenwort (Doppelwort)

- Adresse des Anzeigendoppelwortes im Anwenderspeicher, in dem die Abarbeitung des unter ANR angegebenen Auftrages angezeigt wird.
  - Parameterart: Doppelwort
  - Erlaubter Bereich: DW oder MW; belegt wird jeweils DW und DW+1 oder MW und MW+2
    Die Angabe DW bezieht sich auf den vor dem Aufruf aufgeschlagenen Datenbaustein oder auf den direkt angegebenen DB.

IND

Art der Parametrierung (direkt, indirekt)

- Über diesen Parameter bestimmen Sie die Art der Daten, auf die der Zeiger *QANF* zeigt.
  - 0: *QANF* zeigt direkt auf den Datenanfang der Quell- bzw. Zieldaten.
  - 1: der Zeiger *QANF/ZANF* zeigt auf eine Speicherzelle, ab der die Quell- bzw. Zieldaten definiert sind (indirekt).
  - 2: der Zeiger *QANF/ZANF* zeigt auf einen Speicherbereich in dem sich die Quell- und Zielangaben befinden (indirekt).
  - 5: der Zeiger *QANF/ZANF* zeigt auf eine Speicherzelle, ab der die Quell- bzw. Zielparameter und Parameter zum Anzeigenwort definiert sind (indirekt).
  - 6: der Zeiger *QANF/ZANF* zeigt auf einen Speicherbereich in dem die Quell- bzw. Zielparameter und Parameter zum Anzeigenwort definiert sind (indirekt).
  - Parameterart: Integer
  - Erlaubte Zahlen: 0, 1, 2, 5, 6

Bitte beachten Sie, dass bei IND = 5 bzw. IND = 6 der Parameter ANZW ignoriert wird!

QANF/ZANF

Relative Anfangsadresse der Datenquelle bzw. des Datenziels und bei *IND* = 5 bzw. *IND* = 6 des Anzeigenworts.

- Über diesen Parameter vom Typ Zeiger (Any-Pointer) können Sie die Relative Anfangsadresse und den Typ der Datenquelle (bei SEND) bzw. des Datenziels (bei RECEIVE) angeben.
- Bei *IND* = 5 bzw. *IND* = 6 befinden sich in der Datenquelle auch die Parameter zum Anzeigenwort.
  - Parameterart: Zeiger
  - Sinnvoller Bereich: DB, M, A, E

Beispiel:

```
P#DB10.DBX0.0 BYTE 16
P#M0.0 BYTE 10
P#E 0.0 BYTE 8
P#A 0.0 BYTE 10
```

BLGR

Blockgröße

- Bei Neustart wird mit Hilfe von "SYNCHRON" die Blockgröße (Größe der Datenblöcke) zwischen den Stationen ausgehandelt.
- Hierbei bedeutet große Bockgröße = hoher Datendurchsatz aber auch lange Laufzeit und damit hohe Zykluszeitbelastung.
- Kleine Blockgröße = kleiner Datendurchsatz aber auch kurze Laufzeiten der Bausteine.

Als Blockgröße kann eingestellt werden:

Wert	Blockgröße	Wert	Blockgröße
0	Default (64Byte)	4	128Byte
1	16Byte	5	256Byte
2	32Byte	6	512Byte
3	64Byte	255	512Byte

- Parameterart: Integer
- Möglicher Bereich: 0 ... 255

PAFE

Fehleranzeige bei Parametrierungsfehler

- Das hier angegebene "BYTE" (Ausgang, Merker) wird gesetzt, wenn der Baustein einen "Parametrierungsfehler" erkennt z.B. Schnittstelle (Anschaltung) nicht vorhanden oder unzulässige Parametrierung von QANF/ZANF erfolgte.
  - Parameterart: Byte
  - Sinnvoller Bereich: AB 0 ... AB127, MB 0...MB 255

18.1.2.2 Parameterübergabe

direkte/indirekte Parametrierung

Ein Hantierungsbaustein kann direkt oder indirekt parametriert werden. Nur der Parameter "*PAFE*" muss immer direkt angegeben werden. Bei der direkten Parametrierung verarbeitet der Hantierungsbaustein die beim Bausteinaufruf angegebenen Parameter unmittelbar. Bei der indirekten Parametrierung werden dem Hantierungsbaustein per Bausteinparameter Zeiger, die auf Parameterfelder (Datenbausteine bzw. Datenworte) zeigen, übergeben. Die Parameter *SSNR*, *ANR*, *IND* und *BLGR* sind von Typ "Integer" und können somit indirekt parametriert werden.

Beispiel

Direkte Parameterübergabe

```
CALL  SFC  230
       SSNR:=0
       ANR :=3
       IND :=0
       QANF:=P#A 0.0 BYTE 16
       PAFE:=MB79
       ANZW:=MD44
```

Indirekte Parameterübergabe

```
CALL  SFC  230
       SSNR:=MW10
       ANR :=MW12
       IND :=MW14
       QANF:=P#DB10.DBX0.0 BYTE 16
       PAFE:=MB80
       ANZW:=MD48
```

Bitte beachten Sie, dass die Merkerworte zuvor mit entsprechenden Werten zu laden sind.

18.1.2.3 Quell- bzw. Zielangaben

Übersicht

Sie haben die Möglichkeit die Angaben für Quelle, Ziel und für *ANZW* direkt anzugeben oder indirekt in einem Baustein abzulegen, auf den der Zeiger *QANF / ZANF* bzw. *ANZW* zeigt. Der Parameter *IND* dient als Umschaltkriterium für die direkte und indirekte Parametrierung.

Direkte Parametrierung der Quell- und Zielangaben (*IND* = 0)

Mit *IND* = 0 geben Sie an, dass der Zeiger *QANF / ZANF* direkt auf die Quell- bzw. Zieldaten zeigt. Nachfolgend sehen Sie eine Tabelle über mögliche *QANF / ZANF*-Parameter bei der direkten Parametrierung:

QTYP/ZTYP	Daten in DB	Daten in MB	Daten in AB Prozessabbild der Ausgänge	Daten in EB Prozessabbild der Eingänge
Zeiger: Beispiel:	`P#DBa.DBX b.0 BYTE CP#DB10.DBX 0.0 BYTE 8`	`P#M b.0 BYTE cP#M 5.0 BYTE 10`	`P#A b.0 BYTE cP#A 0.0 BYTE 2`	`P#E b.0 BYTE cP#E 20.0 BYTE 1`
DB, MB, AB, EB Bedeutung	`P#DBa` "a" steht für die DB-Nr., aus dem die Quelldaten entnommen werden oder in den die Zieldaten transferiert werden.	`P#M` kennzeichnet, dass die Daten in einem MB abgelegt sind.	`P#A` kennzeichnet, dass die Daten im Ausgangsbyte abgelegt sind.	`P#E` kennzeichnet, dass die Daten im Eingangsbyte abgelegt sind.
erlaubter Bereich für "a"	0 ... 32767	irrelevant	irrelevant	irrelevant
Daten-/Merker-Byte, AB, EB Bedeutung	DW-Nr., ab der die Daten entnommen oder geschrieben werden.	Merkerbyte-Nr., ab der die Daten entnommen oder geschrieben werden.	Ausgangs-Byte-Nr., ab der die Daten entnommen oder geschrieben werden.	Eingangsbyte-Nr., ab der die Daten entnommen oder geschrieben werden.
erlaubter Bereich für "b"	0.0 ... 2047.0	0 ... 255	0 ... 127	0 ... 127
`BYTE c` Bedeutung erlaubter Bereich für "c"	Länge des Quell-/ Ziel-Datenblocks in Worten. 1 ... 2048	Länge des Quell-/ Ziel-Datenblocks in Bytes. 1 ... 255	Länge des Quell-/ Ziel-Datenblocks in Bytes. 1 ... 128	Länge des Quell-/ Ziel-Datenblocks in Bytes. 1 ... 128

Indirekte Parametrierung der Quell- und Zielangaben (*IND* = 1 oder *IND* = 2)

Bei der indirekten Adressierung zeigt *QANF / ZANF* auf einen Speicherbereich, in dem die Adressen der Quell- bzw. Ziel-Bereiche hinterlegt sind. Hierbei können Sie entweder für Datenquelle und Datenziel einen Bereich angeben (*IND* = 1) oder für Datenquelle und Datenziel jeweils einen Bereich bestimmen (*IND* = 2). In der nachfolgenden Tabelle finden Sie mögliche QANF / ZANF-Parameter bei der indirekten Parametrierung:

QTYP/ZTYP	IND = 1		IND = 2		
Bedeutung	Indirekte Adressierung für Quell- **oder** Zielparameter. Die Quell- oder Zielparameter werden in einem DB hinterlegt. *QANF/ZANF:*		Indirekte Adressierung für Quell- **und** Zielparameter. Die Quell- und Zielparameter werden hintereinander in einem DB hinterlegt. *QANF/ZANF:*		
	DW +0	Datentyp Quelle	DW +0	Datentyp Quelle	Beschreibung Datenquelle
	+2	DB-Nr. bei Typ "DB" ansonsten irrelevant	+2	DB-Nr. bei Typ "DB" ansonsten irrelevant	
	+4	Anfangsadresse	+4	Anfangsadresse	
	+6	Länge in Byte	+6	Länge in Byte	
			+8	Datentyp Ziel	Beschreibung Datenziel
			+10	DB-Nr. bei Typ "DB" ansonsten irrelevant	
			+12	Anfangsadresse	
			+14	Länge in Byte	
erlaubte DB-Nr.	0 ... 32767		0 ... 32767		

QTYP/ZTYP	IND = 1	IND = 2
Daten-Wort Bedeutung	DW-Nr., ab der die Daten hinterlegt sind	DW-Nr., ab der die Daten hinterlegt sind
erlaubter Bereich	0.0 ... 2047.0	0.0 ... 2047.0
Länge Bedeutung	Länge des DBs in Byte	Länge des DBs in Byte
erlaubter Bereich	8 fix	16 fix

Indirekte Parametrierung von Quell- Zielangaben und *ANZW* (*IND* = 5 oder *IND* = 6)

Bei der indirekten Adressierung zeigt *QANF / ZANF* auf einen Speicherbereich, in dem die Adressen der Quell- bzw. Ziel-Bereiche und des Anzeigeworts hinterlegt sind. Hierbei können Sie entweder für Datenquelle oder -ziel und Anzeigenwort einen Bereich angeben (*IND* = 5) oder für Datenquelle, Datenziel und Anzeigenwort jeweils getrennte Bereiche bestimmen (*IND* = 6). In der nachfolgenden Tabelle finden Sie mögliche *QANF / ZANF*-Parameter bei der indirekten Parametrierung:

QTYP/ZTYP	IND = 5			IND = 6		
Bedeutung	Indirekte Adressierung für Quell- oder Zielparameter und Anzeigenwort.Die Quell- oder Ziel und *ANZW*-Parameter werden hintereinander in einem DB hinterlegt. *QANF/ZANF*			Indirekte Adressierung für Quell- und Zielparameter und Anzeigenwort.Die Quell-, Ziel- und *ANZW*-Parameter werden hintereinander in einem DB hinterlegt. *QANF/ZANF*		
	DW +0	Datentyp Quelle	Beschreibung Datenquelle/- ziel	DW +0	Datentyp Quelle	Beschreibung Datenquelle
	+2	DB-Nr. bei Typ "DB" ansonsten irrelevant		+2	DB-Nr. bei Typ "DB" ansonsten irrelevant	
	+4	Anfangsadresse		+4	Anfangsadresse	
	+6	Länge in Byte		+6	Länge in Byte	
	+8	Datentyp Quelle	Beschreibung Anzeigenwort	+8	Datentyp Ziel	Beschreibung Datenziel
	+10	DB-Nr. bei Typ "DB" ansonsten irrelevant		+10	DB-Nr. bei Typ "DB" ansonsten irrelevant	
	+12	Anfangsadresse		+12	Anfangsadresse	
				+14	Länge in Byte	
				+16	Datentyp Quelle	Beschreibung Anzeigenwort
				+18	DB-Nr. bei Typ "DB" ansonsten irrelevant	
				+20	Anfangsadresse	
erlaubte DB-Nr.	0 ... 32767			0 ... 32767		
Daten-Wort Bedeutung	DW-Nr., ab der die Daten hinterlegt sind			DW-Nr., ab der die Daten hinterlegt sind		
erlaubter Bereich	0.0 ... 2047.0			0.0 ... 2047.0		

QTYP/ZTYP	IND = 5	IND = 6
Länge Bedeutung	Länge des DBs in Byte	Länge des DBs in Byte
erlaubter Bereich	14 fix	22 fix

18.1.2.4 Anzeigenwort *ANZW*

Status- und Fehleranzeigen

Status und Fehleranzeigen liefern die Hantierungsbausteine:

- über das Anzeigenwort *ANZW* (Informationen zur Auftragsbearbeitung).
- über das Parametrierfehlerbyte *PAFE* (Anzeige einer fehlerhaften Auftragsparametrierung).

Inhalt und Aufbau Anzeigenwort *ANZW*

Das "Anzeigenwort" zeigt den Zustand für einen bestimmten Auftrag auf einem CP an. Im SPS-Programm sollte für jeden Auftrag ein eigenes "Anzeigenwort" für jeden definierten Auftrag bereitgestellt werden. Das Anzeigenwort hat den folgenden prinzipiellen Aufbau:

Byte	Bit 7 ... Bit 0
0	■ Bit 3 ... Bit 0: Fehlerverwaltung CPU – 0: kein Fehler – 1 ... 5: CPU-Fehler – 6 ... 15: CP-Fehler ■ Bit 7 ... Bit 4: reserviert
1	Statusverwaltung CPU ■ Bit 0: Handshake sinnvoll (Daten vorhanden) – 0: RECEIVE gesperrt – 1: RECEIVE freigegeben ■ Bit 1: Auftrag läuft – 0: SEND/FETCH freigegeben – 1: SEND/FETCH gesperrt ■ Bit 2: Auftrag fertig ohne Fehler ■ Bit 3: Auftrag fertig mit Fehler Datenverwaltung Hantierungsbaustein ■ Bit 4: Datenübernahme/-übergabe läuft ■ Bit 5: Datenübergabe erfolgt ■ Bit 6: Datenübernahme erfolgt ■ Bit 7: Disable/Enable Datenblock – 1: gesperrt – 0: freigegeben
2 ... 3	Längenwort Hantierungsbaustein

Im "Längenwort" hinterlegen die Hantierungsbausteine (SEND, RECEIVE) die für den entsprechenden Auftrag bereits transferierten Daten; empfangene Daten in Empfangsaufträgen; bereits gesendete Daten in Sendeaufträgen. Die Anzeige im "Längenwort" erfolgt immer in Bytes und absolut.

Fehlerverwaltung Byte 0, Bit 0 ... Bit 3

In diesen Bits werden die Fehleranzeigen des Auftrags angezeigt. Diese Fehleranzeigen sind nur gültig, wenn auch gleichzeitig das Bit "Auftrag fertig mit Fehler" im Statusbit gesetzt ist. Folgende Fehlermeldungen können ausgegeben werden:

0 - **kein Fehler**

Sollte das Bit "Auftrag fertig mit Fehler" gesetzt sein, so hat der CP die Verbindung neu aufbauen müssen, wie z.B. nach einem Neustart oder RESET.

1 - **falscher Q/ZTYP am HTB**

Auftrag wurde mit falscher TYP-Kennung parametriert.

2 - **Bereich im AG nicht vorhanden**

Beim Anstoß des Auftrags wurde eine falsche DB-NR parametriert.

3 - **Bereich im AG zu klein**

Die Summe aus Q/ZANF und Q/ZLAE überschreitet die Bereichsgrenzen. Die Bereichsgrenze wird bei Datenbausteinen durch die Bausteingröße bestimmt. Bei Merkern, Zeiten, Zählern usw. ist die Bereichsgröße AG-abhängig.

4 - **QVZ-Fehler im AG**

Mit dem Quell- bzw. Zielparameter wurde ein Bereich im AG angegeben, dessen Speicher defekt oder nicht bestückt ist. Der QVZ-Fehler kann nur bei Q/ZTYP AS, PB, QB oder bei Speicherdefekten auftreten.

5 - **Fehler beim Anzeigenwort**

Das parametrierte Anzeigenwort kann nicht bearbeitet werden. Dieser Fehler tritt auf, wenn mit *ANZW* ein Datenwort bzw. Doppelwort angegeben wurde, das sich nicht oder nicht mehr in dem spezifizierten Datenbaustein befindet d.h. DB zu klein oder nicht vorhanden.

6 - **kein gültiges ORG-Format**

Das Datenziel bzw. die Datenquelle ist weder beim Hantierungsbaustein (Q/TYP="NN") noch im Verbindungsbaustein angegeben.

7 - **Reserviert**

8 - **keine freien Transportverbindungen**

Die Transportverbindungskapazitäten sind überschritten. Löschen Sie unnötige Verbindungen.

9 - **Remote-Fehler**

Bei einem RAD/WRITE-Auftrag ist ein Fehler im Kommunikationspartner aufgetreten.

A - **Verbindungsfehler**

Die Verbindung für einen Auftrag ist nicht bzw. noch nicht aufgebaut. Der Fehler verschwindet, sobald eine Verbindung aufgebaut werden kann. Sind alle Verbindungen des CPs unterbrochen, so deutet dies auf einen Defekt der Baugruppe oder des Buskabel hin. Der Fehler kann auch durch eine fehlerhafte Parametrierung ausgelöst werden, wie z.B. fehlerhafte Adressierung.

B - **Handshakefehler**

Dies kann ein Systemfehler sein oder die Datenblockgröße ist zu groß gewählt.

C - **Anstoßfehler**

Zum Anstoß des Auftrags wurde ein falscher Hantierungsbaustein benutzt oder ein zu großer Datenblock übergeben.

D - **Abbruch nach RESET**

Hier handelt es sich um eine Betriebsmeldung. Bei Priorität 1 und 2 ist die Verbindung unterbrochen und wird neu aufgebaut, sobald sich der Kommunikationspartner auf eine neue Verbindung eingestellt hat. Bei Priorität 3 Verbindungen ist die Verbindung gelöscht, ein neuer Anstoß ist möglich.

E - **Auftrag mit Urladefunktion**

Dies ist eine Betriebsmeldung. Der Auftrag ist ein RAD/WRITE-PASSIV und kann vom AG aus nicht gestartet werden.

F - **Auftrag nicht vorhanden**Der angesprochene Auftrag ist nicht auf dem CP parametriert. Dieser Fehler kann auftreten, wenn SSNR/A-NR Kombination im Hantierungsbaustein falsch oder kein Verbindungsbaustein eingetragen ist.

Die Bits 4 bis 7 von Byte 2 sind für Erweiterungen reserviert.

Statusverwaltung Byte 1, Bit 0 ... Bit 3

Hier können Sie erkennen, ob ein Auftrag bereits gestartet ist, ob hierbei Fehler aufgetreten sind oder ob der Auftrag gesperrt ist, dass beispielsweise eine virtuelle Verbindung nicht mehr besteht.

- **Bit 0 - Handshake sinnvoll**
  - Setzen:
    Durch die Anschaltung entsprechend der "Löschen"-Anzeige im Auftragsstatus-Bit. Handshake sinnvoll (= 1) wird beim RECEIVE-Baustein genutzt. (Telegramm vorhanden bei PRO 1 oder RECEIVE-Anstoß möglich bei PRO 2/3).
  - Auswerten:
    Durch den RECEIVE-Baustein: Nur wenn das Bit gesetzt ist, leitet der RECEIVE den Handshake mit dem CP ein. Durch die Anwendung: Für RECEIVE-Anfrage (Abfrage, ob Telegramm vorhanden bei PRO 1).
- **Bit 1 - Auftrag läuft**
  - Setzen:
    Durch die Anschaltung, wenn Auftrag an CP erteilt ist.
  - Löschen:
    Durch die Anschaltung, wenn ein Auftrag abgearbeitet ist (z.B. Quittung eingetroffen).
  - Auswerten:
    Durch die Hantierungsbausteine: Ein neuer Auftrag wird nur erteilt, wenn der "alte" Auftrag abgearbeitet ist. Durch den Anwender: um zu erfahren, ob das Triggern eines neuen Auftrags sinnvoll ist.
- **Bit 2 - Auftrag fertig ohne Fehler**
  - Setzen:
    Durch die Anschaltung, wenn der entsprechende Auftrag ohne Fehler abgeschlossen wurde.
  - Löschen:
    Durch die Anschaltung, wenn der Auftrag erneut ausgelöst wird.
  - Auswerten:
    Durch den Anwender zur Prüfung, ob der Auftrag fehlerlos abgeschlossen wurde.
- **Bit 3 - Auftrag fertig mit Fehler**
  - Setzen:
    Durch die Anschaltung, wenn der entsprechende Auftrag mit Fehler abgeschlossen wurde. Die Fehlerursache ist dann im High-Teil des Anzeigenwortes verschlüsselt.
  - Löschen:
    Durch die Anschaltung, wenn der Auftrag erneut ausgelöst wird.
  - Auswerten:
    Durch den Anwender: Zur Prüfung, ob der Auftrag mit Fehler abgeschlossen, wurde. Ist die Kennung "Auftrag fertig mit Fehler" gesetzt, steht im High-Byte des Anzeigenwortes die Fehlerursache.

Datenverwaltung Byte 1, Bit 4 ... Bit 7

Hier ist verschlüsselt, ob der Datentransfer für den Auftrag noch läuft oder ob die Datenübergabe bzw. Datenübernahme bereits abgeschlossen ist. Mit dem Bit "Enable / Disable" kann der Datentransfer für den Auftrag gesperrt werden. (Disable = 1; Enable = 0).

- **Bit 4 - Datenübernahme / Datenübergabe läuft**
  - Setzen:
    Durch die Hantierungsbausteine SEND, RECEIVE, wenn die Übergabe/Übernahme für einen Auftrag begonnen wurde, z.B. wenn Daten über die ALL-Funktion (DBA-Ersatz) ausgetauscht werden, der Anstoß jedoch mit SEND-DIREKT erfolgte.
  - Löschen:
    Durch die Hantierungsbausteine SEND, RECEIVE, wenn der Datenaustausch für einen Auftrag beendet ist (letzter Teilblock übertragen).
  - Auswerten:
    Durch den Anwender: Während der Datenübertragung CP <<->> AG darf der Anwender den Datensatz eines Auftrags nicht mehr verändern. Bei PRO 0/1 Aufträgen ist dies unkritisch, da hierbei der Datenaustausch in einem Baustein-Durchlauf erledigt werden kann. Größere Datenmengen können jedoch nur in Blöcken übertragen werden, wobei diese Blockung über mehrere AG-Zyklen verteilt wird. Zur Wahrung der Datenkonsistenz ist zu prüfen ob der Datenblock gerade übertragen wird, bevor dessen Inhalt geändert wird.
- **Bit 5 - Datenübergabe erfolgt**
  - Setzen:
    Durch den Hantierungsbaustein SEND, wenn die Datenübergabe für einen Auftrag erfolgt ist.
  - Löschen:
    Durch den Hantierungsbaustein SEND, wenn für einen neuen Auftrag (neue TRIGGERN) mit dem Transfer der Daten begonnen wurde. Durch den Anwender: Wenn die Auswertung erfolgte (Flankenbildung).
  - Auswerten:
    Durch den Anwender: Mit diesem Bit ist zu ermitteln, ob der Datensatz für einen Auftrag schon auf den CP übertragen wurde bzw. wann ein neuer Datensatz für einen laufenden Auftrag (z.B. zyklische Übertragung) bereitgestellt werden kann.
- **Bit 6 - Datenübernahme erfolgt**
  - Setzen:
    Durch RECEIVE, wenn die Übernahme von Daten für einen Auftrag abgeschlossen wurde.
  - Löschen:
    Durch RECEIVE, wenn für einen neuen Auftrag (neue TRIGGERN) mit dem Transfer der Daten ins AG begonnen wurde. Durch den Anwender, wenn die Auswertung erfolgt (Flankenbildung).
  - Auswerten:
    Durch den Anwender: Mit diesem Bit kann der Anwender ermitteln, ob der Datensatz eines Auftrags schon auf das AG übertragen wurde bzw. wann ein neuer Datensatz für einen laufenden Auftrag ins AG transferiert wurde.
- **Bit 7 - Disable / Enable Datenblock**
  - Setzen:
    Durch den Anwender, um das Beschreiben eines Bereichs durch den RECEIVE-Baustein bzw. das Auslesen aus einem Bereich durch den SEND-Baustein zu verhindern (nur beim 1. Datenblock).
  - Löschen:
    Durch den Anwender, um den zugehörigen Datenbereich freizugeben.
  - Auswerten:
    Durch die Hantierungsbausteine SEND und RECEIVE. Ist das Bit 7 gesetzt, führen die Bausteine keinen Daten-verkehr durch, sondern melden dem CP den Fehler.

Längenwort Byte 2 und Byte 3

Im Längenwort hinterlegen die Hantierungsbausteine (SEND, RECEIVE) die Menge für den entsprechenden Auftrag bereits transferierten Daten, d.h. bei Empfangsaufträgen die bereits empfangene Datenmenge, bei Sendeaufträgen die bereits gesendete Datenmenge.

Beschreiben: - Durch SEND, RECEIVE während des Datenaustausches. Das "Längen-Wort" wird errechnet aus: aktuelle Übertragungsanzahl + Anzahl bereits ausgetauschter Daten

Löschen: - Durch Überschreiben bzw. mit jedem neuen SEND, RECEIVE, FETCH. Wenn das Bit "Auftrag fertig ohne Fehler" bzw. "Datenübergabe/-Übernahme erfolgt" gesetzt ist, steht im "Längen-Wort" die aktuelle Quell- bzw. Ziellänge. Wenn das Bit "Auftrag fertig mit Fehler" gesetzt ist, beinhaltet das Längenwort die bis zum Fehlerfall übertragene Datenanzahl.

Status- und Fehleranzeigen

Im Folgenden sind wichtige Status- und Fehlermeldungen der CPU aufgeführt, die im "Anzeigenwort" erscheinen können. Die Darstellung hierbei erfolgt in "HEX"-Mustern. Das Zeichen X steht für "nicht bestimmt" bzw. für "irrelevant"; Nr. ist die Fehlernummer.

X F X A - Die Fehlerkennung "F" besagt, dass der entsprechende Auftrag auf dem CP nicht definiert ist. Die Statuskennung A bewirkt, dass der Auftrag gesperrt ist (für SEND / FETCH und RECEIVE).

X A X A - Die Fehlerkennung "A" zeigt an, dass die Verbindung des Kommuni-kations-auftrags nicht bzw. noch nicht aufgebaut ist. Mit der Statuskennung "A" ist sowohl der SEND als auch der RECEIVE und FETCH gesperrt.

X 0 X 8 - Die Verbindung ist neu aufgebaut (z.B. nach einem CP-Neuanlauf), der SEND ist freigegeben (SEND-Kommunikationsauftrag).

X 0 X 9 - Die Verbindung ist neu aufgebaut, der RECEIVE ist freigegeben (RECEIVE-Kommunikationsauftrag).

X 0 2 4 - Der SEND ist ohne Fehler abgearbeitet worden, die Daten wurden übertragen.

X 0 4 5 - Der RECEIVE ist ohne Fehler abgearbeitet worden, die Daten sind auf dem AG angekommen.

X 0 X 2 - Der SEND-, RECEIVE-, READ- bzw. WRITE-Auftrag läuft. Bei SEND hat sich der Partner noch nicht auf den RECEIVE eingestellt. Bei RECEIVE hat der Partner noch kein SEND abgesetzt.

Wichtige Anzeigenwortzustände

Anzeigen bei SEND

Zustand unter H1	**Prio 0/1**	**Prio 2**	**Prio 3/4**
Zustand unter TCP/IP	**Prio 1**	**Prio 2**	**Prio 3**
nach Neustart	0 A 0 A	0 A 0 A	0 0 0 8
nach Verbindungsaufbau	X 0 X 8	X 0 X 8	.....
nach Anstoß	X 0 X 2	X 0 X 2	X 0 X 2
fertig ohne Fehler	X 0 2 4	X 0 2 4	X 0 2 4
fertig mit Fehler	X Nr X 8	X Nr X 8	X Nr X 8
nach RESET	X D X A	X D X A	X D X 8

Anzeigen bei RECEIVE

Zustand unter H1	Prio 0/1	Prio 2	Prio 3/4
Zustand unter TCP/IP	**Prio 1**	**Prio 2**	**Prio 3**
nach Neustart	0 A 0 A	0 A 0 A	0 0 0 1
nach Verbindungsaufbau	X 0 X 4	X 0 0 9	.....
nach Anstoß	X 0 X 2	X 0 X 2	X 0 X 2
Telegramm da	X 0 X 1	.....	.....
fertig ohne Fehler	X 0 4 1	X 0 4 5	X 0 4 5
fertig mit Fehler	X Nr X 8	X Nr X 9	X Nr X 9
nach RESET	X D X A	X D X A	X D X 9

Anzeigen bei READ/WRITE-AKTIV

Zustand unter H1	Prio 0/1	Prio 2	Prio 3/4
Zustand unter TCP/IP	**Prio 1**	**Prio 2**	**Prio 3**
nach Neustart		0 A 0 A	
nach Verbindungsaufbau		X 0 0 8	
nach Anstoß		X 0 X 2	
READ fertig		X 0 4 4	
WRITE fertig		X 0 2 4	
fertig mit Fehler		X Nr X 8	
nach RESET		X D X A	

18.1.2.5 Parametrierfehler *PAFE*

PAFE wird gesetzt (Ausgang oder Merker), wenn der Baustein einen "Parametrierungsfehler" erkennt, z.B. Schnittstelle nicht vorhanden oder unzulässige Parametrierung von *QANF* / *ZANF* erfolgte. *PAFE* hat folgenden Aufbau:

Byte	Bit 7 ... Bit 0
0	■ Bit 0: Fehler – 0: kein Fehler – 1: Fehler vorhanden, Fehler-Nr. in Bit 4 ... Bit 7 ■ Bit 3 ... Bit 1: reserviert ■ Bit 7 ... Bit 4: Fehler-Nr. – 0: kein Fehler – 1: falsches ORG-Format – 2: Bereich nicht vorhanden (DB nicht vorhanden) – 3: Bereich zu klein – 4: QVZ-Fehler – 5: falsches Anzeigenwort – 6: keine Quell-/Zielparameter bei SEND/RECEIVE ALL – 7: Schnittstelle nicht vorhanden – 8: Schnittstelle unklar – 9: Schnittstelle überlastet – A: reserviert – B: unzulässige Auftrags-Nr. – C: Schnittstelle des CPs quittiert nicht oder negativ – D: Parameter *BLGR* nicht zulässig – E: reserviert – F: reserviert

18.1.3 SFC 230 - SEND - Senden an Kachel

Beschreibung

Der SEND-Baustein dient zum Auslösen eines Sende-Auftrags zu einem CP. SEND wird im Normalfall im zyklischen Teil des Anwenderprogramms aufgerufen. Die Einbindung des Bausteins im Interrupt oder Weck-Programmteil ist zwar möglich, das Anzeigenwort (*ANZW*) kann hierbei jedoch nicht zyklisch aktualisiert werden, dies sollte durch den CONTROL-Baustein übernommen werden.

Der Verbindungsaufbau mit dem CP wird für die Datenübergabe und für die Aktivierung eines Send-Anstoßes nur dann aufgenommen, wenn:

- dem FB VKE (Verknüpfungsergebnis) "1" übergeben wurde.
- der CP den Auftrag freigegeben hat.
  (Bit "Auftrag läuft" im *ANZW* = 0).

Im Leerlauf des Bausteins wird nur das Anzeigenwort aktualisiert.

Parameter

Name	Deklaration	Typ	Beschreibung
SSNR	IN	INT	Schnittstellennummer
ANR	IN	INT	Auftragsnummer
IND	IN	INT	Adressierungsmodus
QANF	IN	ANY	Zeiger auf Datenquelle
PAFE	OUT	BYTE	Parametrierungsfehler
ANZW	IN_OUT	DWORD	Anzeigenwort

SEND_ALL zur Datenübergabe

Kann der CP die Daten direkt übernehmen, überträgt der SEND-Baustein die angeforderten Daten in einem Zug zum CP. Signalisiert der CP jedoch, dass er nur die Parameter des Auftrages wünscht oder ist die Anzahl der zu übergebenden Daten zu groß, werden dem CP nur die Sende-Parameter bzw. die Parameter mit dem ersten Datenblock übergeben. Die Daten oder der Folgeblock zu diesen Aufträgen fordert der CP über SEND_ALL bei der CPU an. Hierzu ist es jedoch erforderlich, dass mindestens einmal im Zyklus der Baustein SEND_ALL aufgerufen wird. Die Bedienoberfläche ist in allen "Anstoßarten" für den Anwender der Bausteine gleich, nur der Zeitpunkt der Datenübergabe ist bei den zuletzt genannten Fällen um mindestens einen CPU-Zyklus verschoben.

18.1.4 SFC 231 - RECEIVE - Empfangen von Kachel

Beschreibung

Der RECEIVE-Baustein dient zum Empfangen von Daten von einem CP. Im Normalfall wird der RECEIVE-Baustein im zyklischen Teil des Anwenderprogramms aufgerufen. Die Einbindung des Bausteins im Interrupt oder Weck-Programmteil ist ebenso möglich, dabei wird jedoch das Anzeigenwort nicht zyklisch aktualisiert. Diese Funktion muss dann der CONTROL-Baustein übernehmen.

Der Quittungsverkehr mit dem CP (Auftragsanstoß) wird vom RECEIVE-Baustein nur aufgenommen wenn:

- dem FB VKE "1" übergeben wurde und
- der CP den Auftrag freigeben hat (Bit "Handshake sinnvoll" = 1).

Parameter

Name	Deklaration	Typ	Beschreibung
SSNR	IN	INT	Schnittstellenummer
ANR	IN	INT	Auftragsnummer
IND	IN	INT	Adressierungsmodus
ZANF	IN	ANY	Zeiger auf Datenziel
PAFE	OUT	BYTE	Parametrierungsfehler
ANZW	IN_OUT	DWORD	Anzeigenwort

Im "Leerlauf" des Bausteins wird nur das Anzeigenwort aktualisiert. Der RECEIVE-Baustein verhält sich unterschiedlich je nach Art der Versorgung und der CP-Reaktion:

- Wird vom CP ein Parametersatz geliefert, obwohl der RECEIVE-Baustein selbst mit den Zielparametern versorgt wurde, haben die Parameterangaben am Baustein Priorität gegenüber dem Parametersatz vom CP.
- Große Datenmengen können nur in Blöcken übernommen werden. Hierzu ist es erforderlich, solche Folgeblöcke mit RECEIVE_ALL in die CPU zu übertragen. Der Aufruf des RECEIVE_ALL mindestens einmal im zyklischen Programmablauf pro CP-Schnittstelle ist daher immer dann erforderlich, wenn mit einem CP größere Datenblöcke ausgetauscht werden sollen. Ebenso ist die zyklische Einbindung des RECEIVE_ALL erforderlich, wenn der CP den RECEIVE nur zur Freigabe eines Empfangtelegramms benutzt und die Daten über die "Hintergrundkommunikation" der CPU übergibt.

18.1.5 SFC 232 - FETCH - Anfordern von Kachel

Beschreibung

Der FETCH-Baustein dient dem Auslösen eines "Holauftrags" auf einer Gegenstation. Mit dem FETCH-Auftrag werden Daten-Quelle und -Ziel definiert und die Datenquelle an die Gegenstation übertragen. Bei der VIPA-CPU erfolgt die Angabe von Quelle und Ziel über einen Zeiger-Parameter. Die Gegenstation stellt die Daten aus der Quelle bereit und schickt diese über SEND_ALL an die anfordernde Station zurück. Über RECEIVE_ALL werden die Daten empfangen und im Ziel abgelegt. Die Aktualisierung des Anzeigenworts erfolgt über FETCH bzw. CONTROL.

Der Quittungsverkehr für den Anstoß des FETCH wird nur aufgenommen, wenn:

- dem Baustein VKE "1" übergeben
- im entsprechenden CP-Anzeigenwort die Funktion freigegeben wurde (Auftrag läuft = 0).

Parameter

Name	Deklaration	Typ	Beschreibung
SSNR	IN	INT	Schnittstellennummer
ANR	IN	INT	Auftragsnummer
IND	IN	INT	Adressierungsmodus
ZANF	IN	ANY	Zeiger auf Datenziel
PAFE	OUT	BYTE	Parametrierungsfehler
ANZW	IN_OUT	DWORD	Anzeigenwort

Angaben zur indirekten Parametrierung ↬ Kap. 18.1.2.3 "Quell- bzw. Zielangaben" Seite 1110

18.1.6 SFC 233 - CONTROL - Control Kachel

Beschreibung

Der CONTROL-Baustein hat folgende Aufgaben:

- Aktualisierung des Anzeigenworts
- Abfrage, ob ein bestimmter Auftrag des CP zur Zeit "tätig" ist, z.B. Nachfrage nach einem Empfangstelegramm
- Abfrage des CP, welcher Auftrag zur Zeit bearbeitet wird

Der CONTROL-Baustein nimmt keinen Quittungsverkehr mit dem CP auf, sondern überträgt nur die Anzeigen aus dem "Auftragsstatus" zum parametrierten Anzeigenwort. Der Baustein ist nicht VKE abhängig und sollte im zyklischen Teil des Programms aufgerufen werden.

Parameter

Name	Deklaration	Typ	Beschreibung
SSNR	IN	INT	Schnittstellennummer
ANR	IN	INT	Auftragsnummer
PAFE	OUT	BYTE	Parametrierungsfehler
ANZW	IN_OUT	DWORD	Anzeigenwort

ANR

Bei einer *ANR* ≠ 0 wird das Anzeigenwort in der gleichen Weise aufgebaut und bearbeitet wie bei allen anderen Hantierungsbausteinen. Wird der Parameter *ANR* mit 0 versorgt, überträgt der CONTROL-Befehl den Inhalt der Auftragsstatuszelle 0 zum LOW-Teil des Anzeigenworts. In die Auftragsstatuszelle 0 schreibt der CP die Nummer des aktuellen Auftrags, d.h. des Auftrags, der gerade bearbeitet wird, wie z.B. die Auftragsnummer eines Telegramms.

18.1.7 SFC 234 - RESET - Rücksetzen Kachel

Beschreibung

Die RESET ALL-Funktion wird mit der Auftragsnummer 0 angewählt. Sie setzt alle Aufträge dieser logischen Schnittstelle zurück; z.B. löscht sie alle Auftragsdaten und bricht alle laufenden Aufträge ab. Mit einer "direkten" Funktion ($ANR \neq 0$) wird nur der angegebene Auftrag auf der logischen Schnittstelle zurückgesetzt. Der Baustein arbeitet VKE-abhängig und kann von zyklischen, zeitgesteuerten oder alarmgesteuerten Programmteilen aus aufgerufen werden.

Parameter

Name	Deklaration	Typ	Beschreibung
SSNR	IN	INT	Schnittstellennummer
ANR	IN	INT	Auftragsnummer
PAFE	OUT	BYTE	Parametrierfehler

Betriebsarten

Der Baustein kennt folgende beiden Betriebsarten:

- RESET ALL
- RESET DIREKT

18.1.8 SFC 235 - SYNCHRON - Synchronisieren Kachel

Beschreibung

Der Baustein stellt im CPU-Anlauf die Synchronisation zwischen CPU und CP her und ist daher in den Anlauf-OBs aufzurufen. Gleichzeitig wird der Übergabebereich der Schnittstelle gelöscht und voreingestellt, sowie die Blockgröße zwischen CP und CPU ausgehandelt.

Parameter

Name	Deklaration	Typ	Beschreibung
SSNR	IN	INT	Schnittstellenummer
BLGR	IN	INT	Blockgröße
PAFE	OUT	BYTE	Parametrierfehler

Blockgröße

Zur Vermeidung von langen Zykluszeiten ist es sinnvoll große Datenmengen in kleinen Blöcken zwischen CPU und CP zu übertragen. Die Größe dieser Blöcke stellen Sie über die "Blockgröße" ein. Hierbei bedeutet große Bockgröße = hoher Datendurchsatz aber auch lange Laufzeit und damit hohe Zykluszeitbelastung. Kleine Blockgröße = kleiner Datendurchsatz aber auch kleine Laufzeiten der Bausteine. Als Blockgröße kann eingestellt werden:

Wert	Blockgröße	Wert	Blockgröße
0	Default (64Byte)	4	128Byte
1	16Byte	5	256Byte
2	32Byte	6	512Byte
3	64Byte	255	512Byte

Parameterart:	Integer
Möglicher Bereich:	0 ... 255

18.1.9 SFC 236 - SEND_ALL - Alles senden an Kachel

Beschreibung

Mit dem SEND_ALL-Baustein werden die Daten von der CPU an den CP unter Verwendung der eingestellten Blockgröße übermittelt. Die Lage und Größe des Datenbereichs, der mit SEND_ALL zu übermitteln ist, muss zuvor über einen SEND bzw. FETCH-Aufruf definiert werden. Im Anzeigenwort, das dem betreffenden Auftrag zugeordnet ist, werden die Bits "Enable/Disable", "Datenübergabe erfolgt" sowie "Datenübergabe läuft" ausgewertet oder beeinflusst.

Parameter

Name	Deklaration	Typ	Beschreibung
SSNR	IN	INT	Schnittstellennummer
PAFE	OUT	BYTE	Parametrierfehler
ANZW	IN_OUT	DWORD	Anzeigenwort

ANZW

Im Baustein-Anzeigenwort, das im SEND_ALL-Baustein parametriert ist, wird die aktuelle Auftragsnummer hinterlegt (0 bedeutet Leerdurchlauf). Die Anzahl der übertragenen Daten zu einem Auftrag zeigt SEND_ALL in dem Datenwort an, das dem Anzeigenwort folgt.

In folgenden Fällen ist mindestens einmal SEND_ALL im Zyklus-Baustein OB 1 aufzurufen:

- *wenn der CP selbständig Daten von der CPU anfordern kann.*
- *wenn ein CP-Auftrag mit einem SEND angestoßen wird, der CP die Daten zu diesem Auftrag jedoch erst über die "Hintergrundkommunikation" bei der CPU anfordert.*
- *wenn die Anzahl der Daten, die mit einem SEND dem CP übergeben werden sollen, größer als die eingestellte Blockgröße ist.*

18.1.10 SFC 237 - RECEIVE_ALL - Alles empfangen von Kachel

Beschreibung

Mit dem RECEIVE_ALL-Baustein werden die Daten, die vom CP empfangen werden, vom CP an die CPU unter Verwendung der eingestellten Blockgröße übermittelt. Die Lage und Größe des Datenbereichs, der mit RECEIVE_ALL zu übermitteln ist, muss zuvor über einen RECEIVE-Aufruf definiert werden. Im Anzeigenwort, das dem zu bearbeitenden Auftrag zugeordnet ist, werden die Bits "Enable/Disable", "Datenübernahme erfolgt" sowie "Datenübernahme/-übergabe läuft" ausgewertet oder beeinflusst und im Folgewort die "Empfangslänge" angezeigt.

Parameter

Name	Deklaration	Typ	Beschreibung
SSNR	IN	INT	Schnittstellennummer
PAFE	OUT	BYTE	Parametrierfehler
ANZW	IN_OUT	DWORD	Anzeigenwort

ANZW

Im Baustein-Anzeigenwort, das im RECEIVE_ALL-Baustein parametriert ist, wird die aktuelle Auftragsnummer hinterlegt, für den RECEIVE_ALL aktiv war. Im Leerlauf des RECEIVE_ALL ist das Baustein-Anzeigenwort gelöscht.

In folgenden Fällen ist mindestens einmal RECEIVE_ALL im Zyklus-Baustein OB 1 aufzurufen:

- *wenn der CP selbständig Daten an die CPU senden soll.*
- *wenn ein CP-Auftrag mit RECEIVE angestoßen wird, der CP die Daten zu diesem Auftrag jedoch erst über die "Hintergrundkommunikation" an die CPU weitergeben kann.*
- *wenn die Anzahl der Daten, die mit einem RECEIVE an die CPU übergeben werden sollen, größer als die eingestellte Blockgröße ist.*

18.1.11 SFC 238 - CTRL1 - Control1 Kachel

Beschreibung

Dieser Baustein ist identisch mit dem CONTROL-Baustein SFC 233 mit der Ausnahme, dass das Anzeigenwort vom Typ Pointer ist und noch *IND* als weiterer Parameter eingefügt wurde. Der Parameter *IND* ist für zukünftige Erweiterungen reserviert. Der CONTROL-Baustein hat folgende Aufgaben:

- Aktualisierung des Anzeigenworts
- Abfrage, ob ein bestimmter Auftrag des CP zur Zeit "tätig" ist, z.B. Nachfrage nach einem Empfangstelegramm
- Abfrage des CP, welcher Auftrag zur Zeit bearbeitet wird

Der CONTROL-Baustein nimmt keinen Quittungsverkehr mit dem CP auf, sondern überträgt nur die Anzeigen aus dem "Auftragsstatus" zum parametrierten Anzeigenwort. Der Baustein ist nicht VKE abhängig und sollte im zyklischen Teil des Programms aufgerufen werden.

Parameter

Name	Deklaration	Typ	Beschreibung
SSNR	IN	INT	Schnittstellennummer
ANR	IN	INT	Auftragsnummer
IND	IN	INT	reserviert
PAFE	OUT	BYTE	Parametrierfehler
ANZW	IN_OUT	DWORD	Anzeigenwort

ANR

Bei einer *ANR* ≠ 0 wird das Anzeigenwort in der gleichen Weise aufgebaut und bearbeitet wie bei allen anderen "Hantierungsbausteinen". Wird der Parameter *ANR* mit 0 versorgt, überträgt der CTRL1-Befehl den Inhalt der Auftragsstatuszelle 0 zum LOW-Teil des Anzeigenworts. In die Auftragsstatuszelle 0 schreibt der CP die Nummer des aktuellen Auftrags, d.h. des Auftrags, der gerade bearbeitet wird, wie z.B. die Auftragsnummer eines Telegramms.

IND

Der Parameter *IND* hat zur Zeit keine Funktion und ist für zukünftige Erweiterungen reserviert.

ANZW

Das Anzeigenwort *ANZW* ist vom Typ Pointer. Somit haben Sie auch die Möglichkeit das Anzeigenwort in einem Datenbaustein abzulegen.

18.2 Datei-Funktionen SPEED7-CPUs - *"File Functions SPEED7 CPUs"*

18.2.1 FC/SFC 195 und FC/SFC 208...215 - Speicherkarten-Zugriff

Übersicht

Mit den FC/SFC 195 und FC/SFC 208 ... FC/SFC 215 haben Sie die Möglichkeit den Speicherkarten-Zugriff in Ihr Anwenderprogramm einzubinden. Folgende Parameter sind für den Einsatz der FC/SFCs erforderlich:

HANDLE, FILENAME

Der Zugriff erfolgt über eine *HANDLE*-Nr., die Sie durch Aufruf des FC/SFC 208 FILE_OPN bzw. FC/SFC 209 FILE_CRE einem *FILENAME* zuordnen können. Gleichzeitig dürfen maximal 4 *HANDLE* belegt sein (0 ... 3). Durch Schließen mit FC/SFC 210 FILE_CLO wird eine geöffnete Datei geschlossen und der *HANDLE* wieder freigegeben.

MEDIA

Geben Sie als Media-Format für die MMC eine 0 an. Andere Formate werden zur Zeit nicht unterstützt.

ORIGIN, OFFSET

Das Lesen und Schreiben erfolgt ab der Position einer Schreib/Lesemarke. Nach dem Öffnen bzw. neu Anlegen einer Datei befindet sich die Schreib/Lesemarke auf Position 0. Mit dem FC/SFC 213 FILE_SEK können Sie die Schreib/Lesemarke ab einer *ORIGIN*-Position um einen *OFFSET* (Anzahl Bytes) verschieben.

REQ, BUSY

- Mit *REQ* = 1 aktivieren Sie die entsprechende Funktion.
- Ist *REQ* = 0 erhalten Sie den aktuellen Status einer Funktion über *RETVAL* zurückgeliefert.
- *BUSY* = 1 zeigt an, dass die entsprechende Funktion bearbeitet wird.

RETVAL

Nach Abarbeitung einer Funktion liefert *RETVAL* einen Zahlencode zurück:

RETVAL = 0:	Funktion wurde fehlerfrei ausgeführt.
0 < RETVAL < 7000h:	*RETVAL* = Länge der transferierten Daten (nur FC/SFC 211 und FC/SFC 212).
7000h ≤ RETVAL < 8000h:	Zeigt den Bearbeitungs-Status der Funktion.
RETVAL ≥ 8000h:	Kennzeichnet einen Fehler, der bei dem entsprechenden FC/SFC näher beschrieben ist.

VORSICHT!

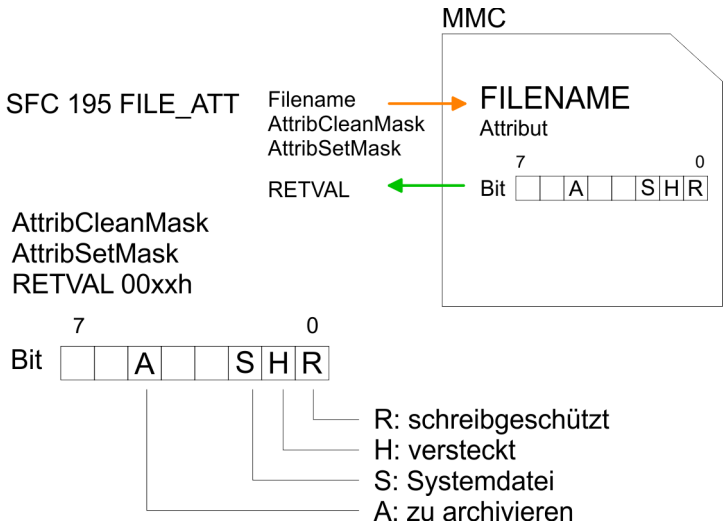
Für den Zugriff auf Speicherkarte sind folgende Hinweise zu berücksichtigen, deren Nichtbeachtung zu Datenverlust auf der Speicherkarte führen kann:

– Es dürfen maximal 4 Handle (0 ... 3) gleichzeitig belegt sein!
– Dateinamen müssen dem 8.3 Format ohne Sonderzeichen entsprechen!
– Mit diesen FC/SFCs haben Sie ausschließlich Zugriff auf die oberste Verzeichnis-Ebene (Root-Verzeichnis) der Speicherkarte!
– Sie dürfen ausschließlich Dateien umbenennen bzw. löschen, die Sie zuvor mit FC/SFCs 210 FILE_CLO geschlossen haben!

18.2.2 FC/SFC 195 - FILE_ATT - Datei-Attribute ändern

Beschreibung

Unter Einsatz von FILE_ATT können Sie die Datei-Attribute einer Datei im Root-Verzeichnis der Speicherkarte ändern. Geben Sie hierzu einen Dateinamen an. Durch Vorgabe eines Bitmusters können Sie mit *ATTRIBCLEANMASK* das entsprechende Attribut rücksetzen bzw. mit *ATTRIBSETMASK* setzen. Bitte beachten Sie, dass hierbei das Setzen Vorrang gegenüber dem Rücksetzen hat. Über *RETVAL* 00xxh bekommen Sie den aktuellen Zustand der Dateiattribute nach Befehlsausführung zurückgeliefert. Wenn Sie *ATTRIBCLEANMASK* und *ATTRIBSETMASK* den Wert 00h übergeben, können Sie über *RETVAL* den aktuellen Status der Dateiattribute ermitteln.



Parameter

Parameter	Deklaration	Datentyp	Beschreibung
REQ	IN	BOOL	Funktion aktivieren
MEDIA	IN	INT	0 = MMC
FILENAME	IN	STRING[254]	Dateiname (muss im 8.3-Format sein)
ATTRIBCLEANMASK	IN	BYTE	Bit-Maske Datei-Attribute zurücksetzen
ATTRIBSETMASK	IN	BYTE	Bit-Maske Datei-Attribute setzen
RETVAL	OUT	WORD	Rückgabewert (00xxh=OK mit xx: Attribute)
BUSY	OUT	BOOL	Funktion wird bearbeitet

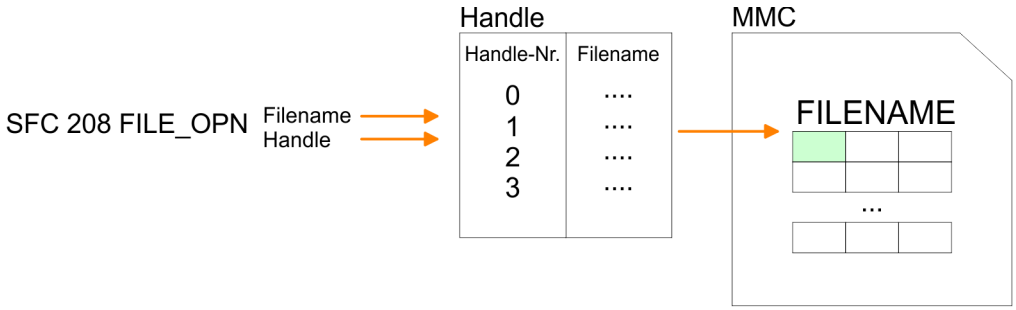
RETVAL (Rückgabewert) Codes, die *RETVAL* zurück liefert:

Code	Beschreibung
00xxh	OK, Attribute wurden geändert mit xx: Attribute
7000h	*REQ* = 0, *BUSY* = 0 (nichts ist zu tun)
7001h	*REQ* = 1, 1. Aufruf
7002h	Baustein wird bearbeitet
A001h	Der angegebene *MEDIA*-Typ ist falsch
A002h	Fehler im Parameter *ATTRIBSETMASK*
A004h	Datei *FILENAME* existiert nicht
A005h	*FILENAME* ist ein Verzeichnis
A006h	Datei ist geöffnet
A007h	Speicherkarte schreibgeschützt
A010h	Dateifehler FILENAME
A100h	Allgemeiner Filesystem-Fehler (z.B. keine Speicherkarte gesteckt)

18.2.3 FC/SFC 208 - FILE_OPN - Datei öffnen

Beschreibung

Eine Datei auf der Speicherkarte können Sie mit dem FC/SFC 208 öffnen. Hierbei wird ein *HANDLE* mit dem entsprechenden *FILENAME* verknüpft. Durch Angabe des *HANDLE* haben Sie jetzt solange lesenden und schreibenden Zugriff auf die Datei, bis die Datei mit FC/SFC 210 FILE_CLO wieder geschlossen wird. *REQ* = 1 löst die Funktion aus. Nach dem Öffnen steht die Schreib/Lesemarke auf 0.



Parameter

Parameter	Deklaration	Datentyp	Beschreibung
REQ	IN	BOOL	Funktion aktivieren
MEDIA	IN	INT	0 = MMC
FILENAME	IN	STRING[254]	Dateiname (muss im 8.3-Format sein)
HANDLE	IN	INT	Index der Datei 0 ... 3
RETVAL	OUT	WORD	Rückgabewert (0 = OK)
BUSY	OUT	BOOL	Funktion wird bearbeitet

RETVAL (Rückgabewert) Codes, die RETVAL zurück liefert:

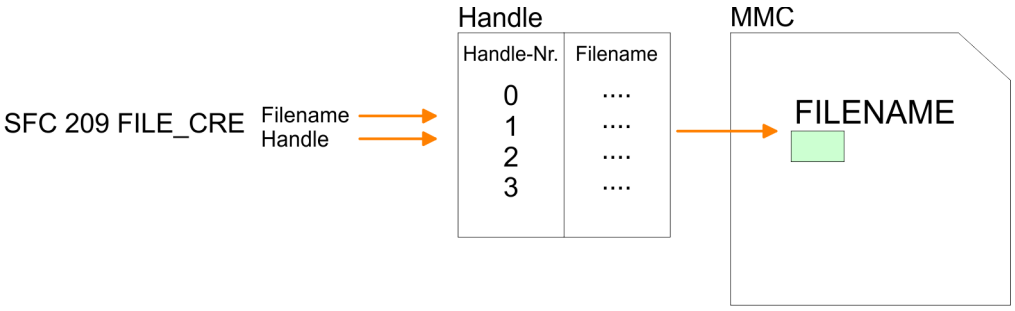
Code	Beschreibung
0000h	OK
7000h	*REQ* = 0, *BUSY* = 0 (nichts ist zu tun)
7001h	*REQ* = 1, 1. Aufruf
7002h	Baustein wird bearbeitet
8010h	Parameter *FILENAME* ist nicht verfügbar (z.B. DB nicht geladen)
8011h	*FILENAME* fehlerhaft (entspricht nicht dem Format 8.3 oder Sonderzeichen)
8100h	Der angegebene *HANDLE* ist ungültig
9001h	*HANDLE* ist bereits anderer Datei zugeordnet
9002h	Eine andere Funktion wurde über den *HANDLE* aufgerufen und ist fertig
9003h	Eine andere Funktion wurde über den *HANDLE* aufgerufen und ist nicht fertig
A000h	Systeminterner Fehler aufgetreten
A001h	Der angegebene *MEDIA*-Typ ist falsch

Code	Beschreibung
A003h	Es ist ein allgemeiner Fehler im Filesystem aufgetreten
A004h	Die unter *FILENAME* angegebene Datei existiert nicht bzw. ist ein Verzeichnis
A100h	Allgemeiner Filesystem-Fehler (z.B. keine Speicherkarte gesteckt)

18.2.4 FC/SFC 209 - FILE_CRE - Datei anlegen

Beschreibung

Durch Einsatz dieses Bausteins können Sie bei einer gesteckten Speicherkarte eine neue Datei mit dem entsprechenden Dateinamen anlegen und für den Lese-/Schreib-Zugriff öffnen. Bitte beachten Sie, dass ausschließlich Dateien auf der obersten Verzeichnis-Ebene erzeugt werden können. *REQ* = 1 löst die Funktion aus. Nach dem Öffnen steht die Schreib/Lesemarke auf 0.



Parameter

Parameter	Deklaration	Datentyp	Beschreibung
REQ	IN	BOOL	Funktion aktivieren
MEDIA	IN	INT	0 = MMC
FILENAME	IN	STRING[254]	Dateiname (muss im 8.3-Format sein)
HANDLE	IN	INT	Index der Datei 0 ... 3
RETVAL	OUT	WORD	Rückgabewert (0 = OK)
BUSY	OUT	BOOL	Funktion wird bearbeitet

RETVAL (Rückgabewert) Codes, die RETVAL zurück liefert:

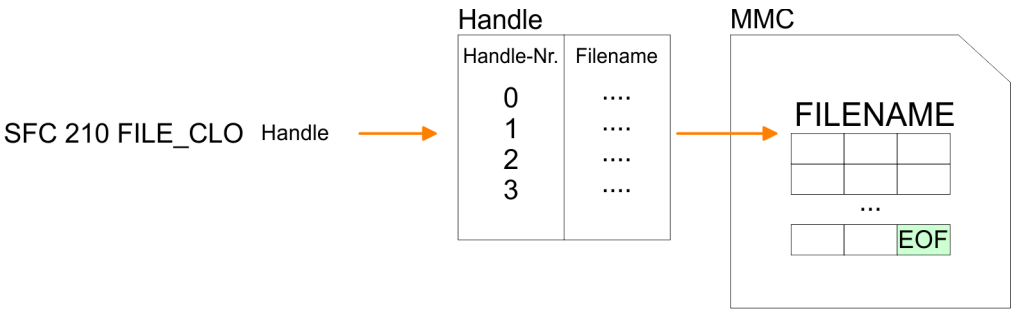
Code	Beschreibung
0000h	OK
7000h	*REQ* = 0, *BUSY* = 0 (nichts ist zu tun)
7001h	*REQ* = 1, 1. Aufruf
7002h	Baustein wird bearbeitet
8010h	Parameter *FILENAME* ist nicht verfügbar (z.B. DB nicht geladen)
8011h	*FILENAME* fehlerhaft (entspricht nicht dem Format 8.3 oder Sonderzeichen)
8100h	Der angegebene *HANDLE* ist ungültig

Code	Beschreibung
9001h	HANDLE ist bereits anderer Datei zugeordnet
9002h	Eine andere Funktion wurde über den *HANDLE* aufgerufen und ist fertig
9003h	Eine andere Funktion wurde über den *HANDLE* aufgerufen und ist nicht fertig
A000h	Systeminterner Fehler aufgetreten
A001h	Der angegebene *MEDIA*-Typ ist falsch
A003h	Es ist ein allgemeiner Fehler im Filesystem aufgetreten
A004h	Es ist kein Root-Eintrag im Verzeichnis verfügbar
A005h	Speicherkarte ist schreibgeschützt
A100h	Allgemeiner Filesystem-Fehler (z.B. keine Speicherkarte gesteckt)

18.2.5 FC/SFC 210 - FILE_CLO - Datei schließen

Beschreibung

Mit diesem Baustein können Sie eine geöffnete Datei schließen. Hierbei wird ein EOF (End of File) angefügt, die Datei geschlossen und der *HANDLE* wieder freigegeben. *REQ* = 1 löst die Funktion aus.



Parameter

Parameter	Deklaration	Datentyp	Beschreibung
REQ	IN	BOOL	Funktion aktivieren
HANDLE	IN	INT	Index der Datei 0 ... 3
RETVAL	OUT	WORD	Rückgabewert (0 = OK)
BUSY	OUT	BOOL	Funktion wird bearbeitet

RETVAL (Rückgabewert) Codes, die RETVAL zurück liefert:

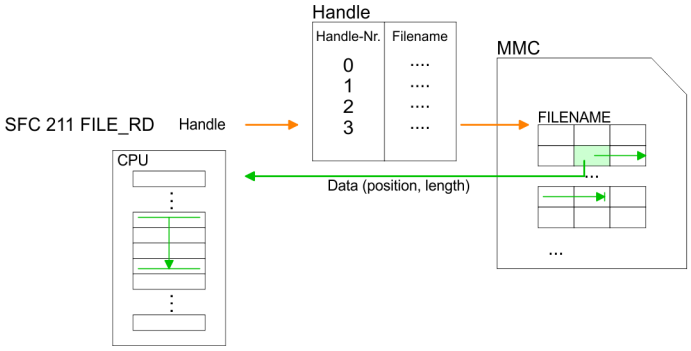
Code	Beschreibung
0000h	OK
7000h	*REQ* = 0, *BUSY* = 0 (nichts ist zu tun)
7001h	*REQ* = 1, 1. Aufruf
7002h	Baustein wird bearbeitet

Code	Beschreibung
8100h	Der angegebene *HANDLE* ist ungültig
9001h	Dem *HANDLE* ist kein Dateiname zugeordnet
9002h	Eine andere Funktion wurde über den *HANDLE* aufgerufen und ist fertig
9003h	Eine andere Funktion wurde über den *HANDLE* aufgerufen und ist nicht fertig
A000h	Systeminterner Fehler aufgetreten
A100h	Allgemeiner Filesystem-Fehler (z.B. keine Speicherkarte gesteckt)

18.2.6 FC/SFC 211 - FILE_RD - Datei lesen

Beschreibung

Hiermit können Sie ab einer ORIGIN-Position (Position der Schreib-/Lesemarke) von der Speicherkarte über den geöffneten Handle Daten in die CPU übertragen. Pro Aufruf können maximal 512Byte übertragen werden. Durch Angabe von *DATA* bestimmen Sie Speicherort und Länge des Schreib-Bereichs in Ihrer CPU. *REQ* = 1 löst die Funktion aus.



Parameter

Parameter	Deklaration	Datentyp	Beschreibung
REQ	IN	BOOL	Funktion aktivieren
HANDLE	IN	INT	Index der Datei 0 ... 3
DATA	IN	ANY	Zeiger auf Speicherort und Länge des Schreibbereichs
RETVAL	OUT	WORD	Rückgabewert (0 = OK)
BUSY	OUT	BOOL	Funktion wird bearbeitet

RETVAL (Rückgabewert) Codes, die RETVAL zurück liefert:

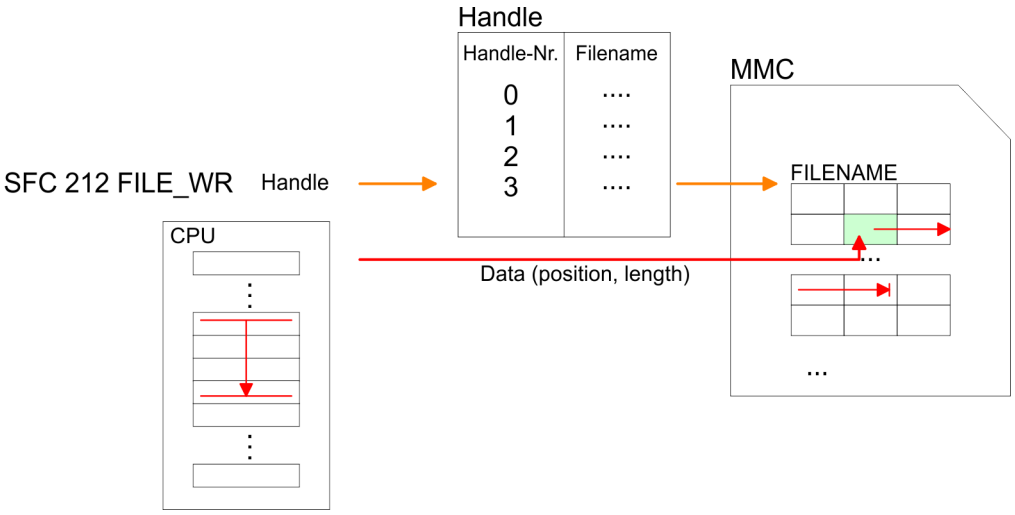
Code	Beschreibung
0xxxh	0 = OK, 0xxx = Länge gelesener Daten
7000h	*REQ* = 0, *BUSY* = 0 (nichts ist zu tun)
7001h	*REQ* = 1, 1. Aufruf
7002h	Baustein wird bearbeitet

Code	Beschreibung
8010h	Pointer in *DATA* ist vom Typ BOOL
8011h	Pointer in *DATA* kann nicht dekodiert werden (z.B. DB nicht geladen)
8012h	Datenlänge ist größer als 512Byte
8013h	Es wurde versucht auf einen schreibgeschützten DB zuzugreifen
8100h	Der angegebene *HANDLE* ist ungültig
9001h	Für diesen *HANDLE* ist keine Datei geöffnet
9002h	Eine andere Funktion wurde über den *HANDLE* aufgerufen und ist fertig
9003h	Eine andere Funktion wurde über den *HANDLE* aufgerufen und ist nicht fertig
A000h	Systeminterner Fehler aufgetreten
A003h	interner Fehler
A100h	Allgemeiner Filesystem-Fehler (z.B. keine Speicherkarte gesteckt)

18.2.7 FC/SFC 212 - FILE_WR - Datei schreiben

Beschreibung

Für Schreibzugriffe auf die Speicherkarte ist dieser Baustein zu verwenden. Hierbei werden Daten von der unter *DATA* angegebenen Position und Länge in der CPU über den entsprechenden *HANDLE* ab der Schreib-/Lese-Position auf die Speicherkarte geschrieben. Pro Aufruf können maximal 512Byte übertragen werden. *REQ* = 1 löst die Funktion aus.



Parameter

Parameter	Deklaration	Datentyp	Beschreibung
REQ	IN	BOOL	Funktion aktivieren
HANDLE	IN	INT	Index der Datei 0 ... 3
DATA	IN	ANY	Zeiger auf Speicherort und Länge des Schreibbereichs
RETVAL	OUT	WORD	Rückgabewert
BUSY	OUT	BOOL	Funktion wird bearbeitet

Der Parameter *RETVAL* liefert die Länge der geschriebenen Daten zurück. Der Baustein liefert keine Fehlermeldung, wenn die Speicherkarte voll ist. Der Anwender muss überprüfen, dass die Anzahl der geforderten zu schreibenden Bytes der in *RETVAL* zurück gelieferten geschriebenen Bytes entspricht.

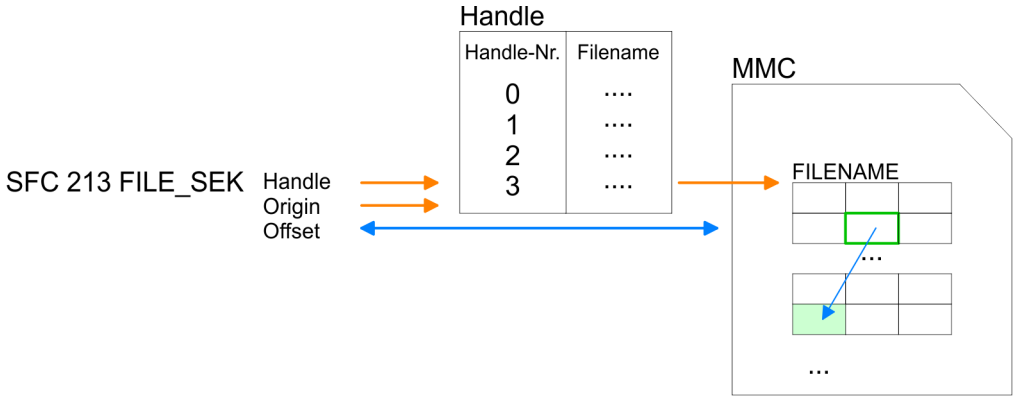
RETVAL (Rückgabewert) Codes, die RETVAL zurück liefert:

Code	Beschreibung
0xxxh	0 = OK, 0xxx = Länge geschriebener Daten
7000h	*REQ* = 0, *BUSY* = 0 (nichts ist zu tun)
7001h	*REQ* = 1, 1. Aufruf
7002h	Baustein wird bearbeitet
8010h	Pointer in *DATA* ist vom Typ BOOL
8011h	Pointer in *DATA* kann nicht dekodiert werden (z.B. DB nicht geladen)
8012h	Datenlänge ist größer als 512Byte
8100h	Der angegebene *HANDLE* ist ungültig
9001h	Für diesen *HANDLE* ist keine Datei geöffnet
9002h	Eine andere Funktion wurde über den *HANDLE* aufgerufen und ist fertig
9003h	Eine andere Funktion wurde über den *HANDLE* aufgerufen und ist nicht fertig
A000h	Systeminterner Fehler aufgetreten
A002h	Die Datei ist schreibgeschützt
A003h	Interner Fehler
A004h	Speicherkarte ist schreibgeschützt
A100h	Allgemeiner Filesystem-Fehler (z.B. keine Speicherkarte gesteckt)

18.2.8 FC/SFC 213 - FILE_SEK - Position Schreib-/Lesemarke

Beschreibung

Mit FILE_SEK können Sie die Position der Schreib-/Lesemarke für den entsprechenden *HANDLE* ändern bzw. ermitteln. Durch Angabe von *ORIGIN* als Startposition und einem *OFFSET* können Sie für den entsprechenden *HANDLE* die Schreib-/Lesemarke platzieren. *REQ* = 1 startet Funktion.



Parameter

Parameter	Deklaration	Datentyp	Beschreibung
REQ	IN	BOOL	Funktion aktivieren
HANDLE	IN	INT	Index der Datei 0 ... 3
ORIGIN	IN	INT	0 = Datei-Anfang, 1 = aktuelle Position, 2 = Dateiende
RETVAL	OUT	WORD	Rückgabewert (0 = OK)
BUSY	OUT	BOOL	Funktion wird bearbeitet
OFFSET	INOUT	DINT	Offset Schreib-/Lesemarke

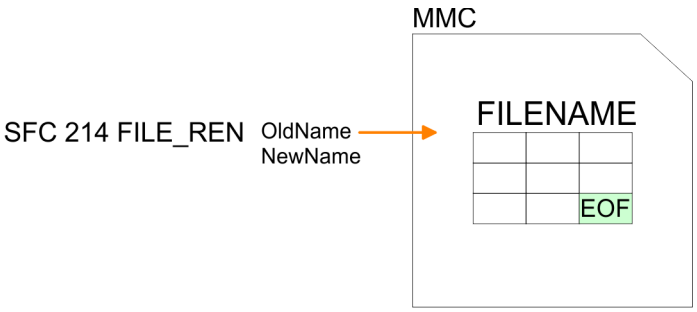
RETVAL (Rückgabewert) Codes, die *RETVAL* zurück liefert:

Code	Beschreibung
0000h	OK, OFFSET beinhaltet die Aktuelle Schreib-/Lese-Position
7000h	*REQ* = 0, *BUSY* = 0 (nichts ist zu tun)
7001h	*REQ* = 1, 1. Aufruf
7002h	Baustein wird bearbeitet
8100h	Der angegebene HANDLE ist ungültig
9001h	Für diesen HANDLE ist keine Datei geöffnet
9002h	Eine andere Funktion wurde über den *HANDLE* aufgerufen und ist fertig
9003h	Eine andere Funktion wurde über den *HANDLE* aufgerufen und ist nicht fertig
A000h	Systeminterner Fehler aufgetreten
A004h	*ORIGIN*-Parameter ist fehlerhaft
A100h	Allgemeiner Filesystem-Fehler (z.B. keine Speicherkarte gesteckt)

18.2.9 FC/SFC 214 - FILE_REN - Datei umbenennen

Beschreibung Unter Einsatz von FILE_REN können Sie den unter *OLDNAME* angegebenen Dateinamen ändern in *NEWNAME*.

VORSICHT!
Bitte beachten Sie, dass Sie nur Dateien umbenennen dürfen, die zuvor mit File_CLO geschlossen wurden. Ansonsten könnte dies zu Datenverlust auf der Speicherkarte führen!



Parameter

Parameter	Deklaration	Datentyp	Beschreibung
REQ	IN	BOOL	Funktion aktivieren
MEDIA	IN	INT	0 = MMC
OLDNAME	IN	STRING[254]	Alter Dateiname (muss im 8.3-Format sein)
NEWNAME	IN	STRING[254]	Neuer Dateiname (muss im 8.3-Format sein)
RETVAL	OUT	WORD	Rückgabewert (0 = OK)
BUSY	OUT	BOOL	Funktion wird bearbeitet

RETVAL (Rückgabewert) Codes, die RETVAL zurück liefert:

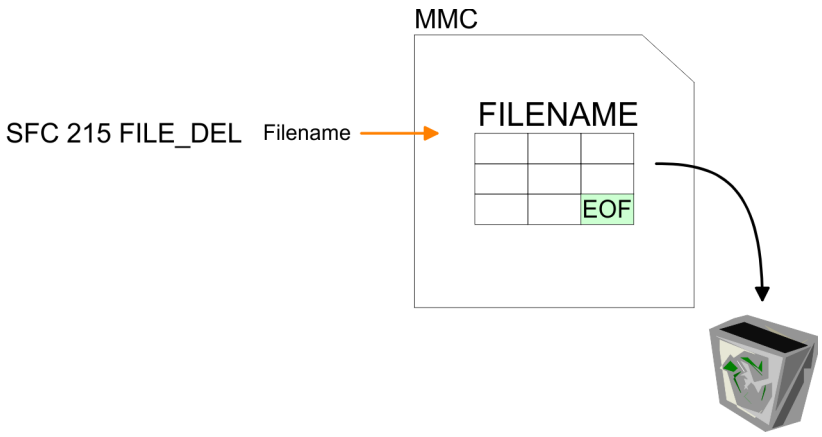
Code	Beschreibung
0000h	OK, Datei wurde umbenannt
7000h	*REQ* = 0, *BUSY* = 0 (nichts ist zu tun)
7001h	*REQ* = 1, 1. Aufruf
7002h	Baustein wird bearbeitet
8010h	Parameter *OLDNAME* ist nicht verfügbar (z.B. DB nicht geladen)
8011h	*OLDNAME* fehlerhaft (entspricht nicht dem 8.3 Format oder Sonderzeichen)
8020h	Parameter *NEWNAME* ist nicht verfügbar (z.B. DB nicht geladen)
8021h	*NEWNAME* fehlerhaft (entspricht nicht dem 8.3 Format oder Sonderzeichen)
A000h	Systeminterner Fehler aufgetreten
A001h	Der angegebene MEDIA-Typ ist falsch
A003h	Der neue Dateiname *NEWNAME* existiert schon
A004h	Datei *OLDNAME* existiert nicht
A006h	Datei *OLDNAME* ist geöffnet
A007h	Speicherkarte schreibgeschützt
A100h	Das Filesystem liefert einen Fehler beim Anlegen der Datei (z.B. keine Speicherkarte gesteckt)

18.2.10 FC/SFC 215 - FILE_DEL - Datei löschen

Beschreibung

Mit diesem Baustein können Sie eine Datei auf der Speicherkarte löschen. Geben Sie hierzu unter *FILENAME* den Namen der zu löschenden Datei an.

VORSICHT!
Bitte beachten Sie, dass Sie nur Dateien löschen dürfen, die zuvor mit File_CLO geschlossen wurden. Ansonsten könnte dies zu Datenverlust auf der Speicherkarte führen!



Parameter

Parameter	Deklaration	Datentyp	Beschreibung
REQ	IN	BOOL	Funktion aktivieren
MEDIA	IN	INT	0 = MMC
FILENAME	IN	STRING[254]	Dateiname (muss im 8.3-Format sein)
RETVAL	OUT	WORD	Rückgabewert (0 = OK)
BUSY	OUT	BOOL	Funktion wird bearbeitet.

RETVAL (Rückgabewert) Codes, die *RETVAL* zurück liefert:

Code	Beschreibung
0000h	OK, Datei wurde gelöscht
7000h	*REQ* = 0, *BUSY* = 0 (nichts ist zu tun)
7001h	*REQ* = 1, 1. Aufruf
7002h	Baustein wird bearbeitet
8010h	Parameter *FILENAME* ist nicht verfügbar (z.B. DB nicht geladen)
8011h	*FILENAME* ist fehlerhaft (z.B. entspricht nicht dem 8.3 Format oder Sonderzeichen)
A000h	Systeminterner Fehler aufgetreten
A001h	Der angegebene *MEDIA*-Typ ist falsch

Code	Beschreibung
A002h	Die Datei ist schreibgeschützt
A004h	Datei *FILENAME* existiert nicht
A005h	*FILENAME* ist ein Verzeichnis - nicht löschbar
A006h	Datei ist geöffnet
A007h	Speicherkarte schreibgeschützt.
A100h	Allgemeiner Filesystem-Fehler (z.B. keine Speicherkarte gesteckt)

18.3 Datei-Funktionen Standard-CPUs - *"File Functions Standard CPUs"*

18.3.1 SFC 220 ... 222 - MMC-Zugriff

Übersicht

Mit den hier aufgeführten SFCs haben Sie die Möglichkeit den Zugriff auf eine MMC in Ihr Anwenderprogramm einzubinden. Hierbei können Sie bei einer gesteckten MMC eine neue Datei anlegen bzw. eine bestehende Datei für den Zugriff öffnen. Solange Sie keine neue Datei öffnen haben Sie über Lese-/Schreib-Befehle Zugriff auf diese Datei.

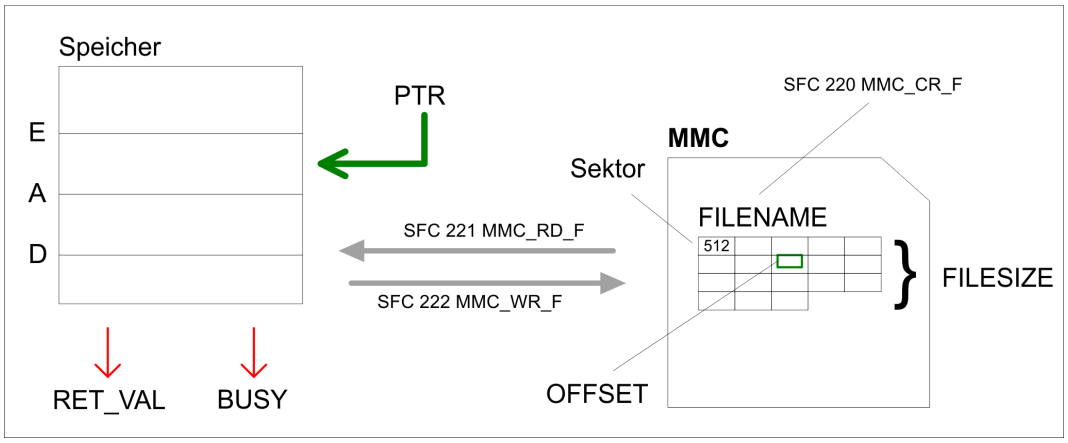
Einschränkungen

Für den Einsatz der SFCs 220, 221 und 222 sind folgende Einschränkungen zu beachten:

- Ein lesender bzw. schreibender Zugriff auf die MMC kann nur dann erfolgen, wenn die Datei zuvor mit dem SFC 220 angelegt bzw. geöffnet wurde.
- Es ist darauf zu achten, dass die Daten immer unfragmentiert auf der MMC abliegen, da nur zusammenhängende Datenblöcke gelesen bzw. geschrieben werden können.
- Werden Daten auf die MMC mit einem externen MMC-Kartenleser übertragen, so können diese fragmentiert sein d.h. die Daten werden in Blöcke aufgeteilt. Dies können Sie vermeiden, indem Sie die MMC vor dem Schreibzugriff formatieren.
- Bei einem Schreibzugriff von der CPU auf die MMC werden die Daten immer unfragmentiert auf der MMC abgelegt.
- Beim Öffnen einer schon bestehenden Datei sind für *FILENAME* und *FILESIZE* immer die Angaben zu verwenden, die Sie beim Anlegen der Datei verwendet haben.
- Eine MMC ist eingeteilt in Sektoren. Jeder Sektor hat eine Größe von 512Byte. Sektorübergreifendes Lesen bzw. Schreiben ist nicht möglich. Ein Zugriff auf sektorübergreifende Daten kann nur dann erfolgen, wenn Sie für jeden Sektor einen Schreib- bzw. Lesebefehl verwenden. Mit der Offset-Angabe bestimmen Sie den jeweiligen Sektor.

Die nachfolgende Abbildung zeigt die Verwendung der einzelnen SFCs und deren Variablen:

CPU



Für Lese- und Schreibzugriffe auf die MMC muss zuvor mit dem SFC 220 die Datei geöffnet werden!

18.3.2 SFC 220 - MMC_CR_F - MMC-Datei erstellen oder öffnen

Beschreibung

Unter Einsatz dieses Bausteins können Sie bei einer gesteckten MMC eine neue Datei anlegen bzw. eine bestehende Datei für den Zugriff öffnen. Solange Sie keine neue Datei öffnen, können Sie über Lese-/Schreib-Befehle auf diese Datei zugreifen. Näheres hierzu und zu den Einschränkungen *Kap. 18.3.1 "SFC 220 ... 222 - MMC-Zugriff" Seite 1141*.

Da der Aufruf des SFC im OB 1 zur Zykluszeit-Überschreitung führen kann, ist der SFC stattdessen im OB 100 aufzurufen.

Parameter

Name	Deklaration	Typ	Beschreibung
FILENAME	IN	STRING[254]	Dateiname
FILESIZE	IN	DWORD	Dateigröße
RET_VAL	OUT	WORD	Rückgabewert (0 = OK)

FILENAME

Geben Sie hier den Dateinamen an, unter dem Ihre Daten auf der MMC abzulegen sind bzw. abliegen. Der Dateiname mit Endekennung 00h darf eine maximale Länge von 13 Zeichen nicht überschreiten:

- 8 Zeichen für Name
- 1 Zeichen für "."
- 3 Zeichen für Dateierweiterung
- 1 Zeichen 00h als Endekennung

Aus softwaretechnischen Gründen müssen Sie das nächste Byte hinter dem Dateinamen mit 00h beschreiben (Endekennung Dateiname).

FILESIZE

Unter *FILESIZE* bestimmen Sie die Größe der Nutzdaten in Byte. Bei Zugriff auf eine schon bestehende Datei ist neben dem *FILENAME* die Angabe der vorgegebenen *FILESIZE* zwingend erforderlich. Die Angabe einer "Joker"-Länge wird zur Zeit nicht unterstützt.

Struktur

Byte 0	Byte 1	Byte 2	Byte 3	...	Byte 255
Max. Länge	belegte Länge	ASCII-Wert 1	ASCII-Wert 2	...	ASCII-Wert 254

RET_VAL (Rückgabewert)

Wort, in das eine Diagnose-/Fehlermeldung zurückgeliefert wird. 0 bedeutet, dass alles OK ist.

Wert	Beschreibung
Diagnosemeldungen	
0000h	keine Fehler (tritt nur beim Erzeugen einer neuen Datei auf).
0001h	Datei existiert schon, ist unfragmentiert und die Längenangabe *FILESIZE* ist identisch oder kleiner als die reale Dateigröße.
8001h	Es ist keine oder eine vom Typ unbekannte MMC gesteckt.

Wert	Beschreibung
Fehlermeldungen	
8002h	Es befindet sich keine FAT auf der MMC.
A001h	Es wurde kein Dateiname angegeben. Diese Meldung kommt nur dann, wenn der Dateiname sich beispielsweise in einem nicht geladenen DB befindet.
A002h	Der angegebene Dateiname ist falsch (nicht 8.3 oder leer).
A003h	Die Datei existiert schon, aber die unter *FILESIZE* angegebene Größe ist größer als die existierende Datei.
A004h	Die Datei existiert schon, ist aber fragmentiert und kann nicht geöffnet werden.
A005h	Es ist kein ausreichender Speicherplatz auf der MMC vorhanden.
A006h	Es existiert kein freier Eintrag im Root-Verzeichnis. Abhängig von der eingesetzten MMC dürfen sich mindestens 16 bis maximal 512 Einträge im Root-Verzeichnis befinden.
B000h	Es ist ein interner Fehler aufgetreten.

18.3.3 SFC 221 - MMC_RD_F - MMC-Datei lesen

Beschreibung

Über den SFC 221 können Sie von einer gesteckten MMC lesen. Bitte beachten Sie, dass die Datei zuvor mit dem SFC 220 für den Zugriff zu öffnen ist und die Datei unfragmentiert vorzuliegen hat. Näheres hierzu und zu den Einschränkungen ↯ *Kap. 18.3.1 "SFC 220 ... 222 - MMC-Zugriff" Seite 1141*.

Parameter

Name	Deklaration	Typ	Beschreibung
PTR	IN	ANY	Zeiger auf Datenbereich für Lesedaten
OFFSET	IN	DWORD	Offset der Daten innerhalb der Datei
BUSY	OUT	BOOL	Auftragsstatus
RET_VAL	OUT	WORD	Rückgabewert (0 = OK)

PTR

Diese Variable vom Typ Pointer zeigt auf einen Datenbereich in der CPU, der mit dem Inhalt der MMC zu beschreiben ist.

OFFSET

Hiermit bestimmen Sie auf der MMC innerhalb des Files den Anfang der Daten, die in die CPU zu übertragen sind.

BUSY

Während der Datenübertragung bleibt dieses Bit gesetzt. Ist der Datentransfer abgeschlossen wird das Bit zurückgesetzt.

RET_VAL (Rückgabewert)

Wort, in das eine Diagnose-/Fehlermeldung zurückgeliefert wird. 0 bedeutet, dass alles OK ist.

Wert	Bedeutung
0000h	keine Fehler (Daten wurden gelesen)
8001h	Es ist keine oder eine vom Typ unbekannte MMC gesteckt.
8002h	Es befindet sich keine FAT auf der MMC.
9000h	Es wurde versucht ein Bit zu lesen (Boolean-Variable). Das bitweise Lesen ist nicht möglich.
9001h	Pointerangabe ist fehlerhaft (zeigt z.B. außerhalb eines DBs)
9002h	Die Dateilänge wurde überschritten.
9003h	Es wurde versucht die Sektorgrenze von 512 zu überschreiten. Sektorübergreifendes Lesen ist nicht möglich.
B000h	Es ist ein interner Fehler aufgetreten.

18.3.4 SFC 222 - MMC_WR_F - MMC-Datei schreiben

Beschreibung

Über den SFC 222 können Sie auf eine gesteckte MMC schreiben. Bitte beachten Sie, dass die Datei zuvor mit dem SFC 220 für den Zugriff zu öffnen ist und die Datei unfragmentiert vorzuliegen hat. Näheres hierzu und zu den Einschränkungen *Kap. 18.3.1 "SFC 220 ... 222 - MMC-Zugriff" Seite 1141*.

Parameter

Name	Deklaration	Typ	Beschreibung
PTR	IN	ANY	Zeiger auf Datenbereich für Schreibdaten
OFFSET	IN	DWORD	Offset der Daten innerhalb der Datei
BUSY	OUT	BOOL	Auftragsstatus
RET_VAL	OUT	WORD	Rückgabewert (0 = OK)

PTR

Diese Variable vom Typ Pointer zeigt auf einen Datenbereich in der CPU, der die Daten beinhaltet, die auf die MMC zu schreiben sind.

OFFSET

Hiermit bestimmen Sie auf der MMC innerhalb der Datei den Anfang der Daten, ab dem die Daten geschrieben werden.

BUSY

Während der Datenübertragung bleibt dieses Bit gesetzt. Ist der Datentransfer abgeschlossen wird das Bit zurückgesetzt.

RET_VAL (Rückgabewert)

Wort, in das eine Diagnose-/Fehlermeldung zurückgeliefert wird. 0 bedeutet, dass alles OK ist.

Wert	Bedeutung
0000h	keine Fehler
8001h	Es ist keine oder eine falsche MMC gesteckt.
8002h	Es befindet sich keine FAT auf der MMC.

Wert	Bedeutung
9000h	Es wurde versucht ein Bit zu schreiben (Boolean-Variable). Das bitweise Schreiben ist nicht möglich.
9001h	Pointerangabe ist fehlerhaft (zeigt z.B. außerhalb eines DBs).
9002h	Die Dateilänge wurde überschritten.
9003h	Es wurde versucht die Sektorgrenze von 512 zu überschreiten. Sektorübergreifendes Lesen ist nicht möglich.
B000h	Es ist ein interner Fehler aufgetreten.

18.4 Systemfunktions-Blöcke - *"System Function Blocks"*

18.4.1 FB/SFB 7 - TIMEMESS - Zeitmessung

Im Gegensatz zum FC/SFC 53 liefert der FB/SFB 7 die Differenz zwischen zwei Aufrufen in µs zurück. Mit *RESET* = 1 wird der aktuelle µs Zählerstand im InstDB gespeichert. Ein erneuter Aufruf mit *RESET* = 0 liefert über *VALUE* den Differenzwert zum ersten Aufruf in µs.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
RESET	IN	BOOL	*RESET* = 1 startet Zähler
VALUE	OUT	DWORD	Differenz in µs

RESET

Mit *RESET* = 1 wird der aktuelle Zählerstand im InstDB gespeichert. Der Wert in *VALUE* wird hierbei nicht beeinflusst.

VALUE

Nach einem Aufruf mit *RESET* = 0 liefert *VALUE* die zeitliche Differenz zwischen den zwei FB/SFB 7 Aufrufen zurück.

18.5 Systemfunktionen - *"System Functions"*

18.5.1 FC/SFC 53 - uS_Tick - Zeitmessung

Mit diesem Baustein können Sie den in der SPEED7-CPU integrierten µs-Ticker auslesen. Der µs-Ticker ist ein 32Bit µs Zeitzähler, der bei jedem Neustart mit 0 beginnt und bis 2^{32-1}µs zählt. Bei einem Überlauf startet der Zähler wieder bei 0. Mittels Differenzbildung der *RETVAL*-Ergebnisse von 2 FC/SFC 53 Aufrufen vor und nach einer Anwendung können Sie auf diese Weise die Laufzeit der Anwendung in µs ermitteln.

Laufzeit in Abhängigkeit vom Betriebszustand

Zustand	µs-Systemzeit
Anlauf	beginnt bei 0 und wird ständig aktualisiert
RUN	wird ständig aktualisiert
STOP	wird angehalten (Zeit kann nicht ausgelesen werden)
Neustart	beginnt wieder bei 0

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
RETVAL	OUT	DINT	Systemzeit in µs

RETVAL

Der Parameter *RETVAL* enthält die gelesene Systemzeit im Bereich von 0 ... 2^{32}-1µs.

Bitte beachten Sie für weitere Berechnungen, dass die Systemzeit in einem vorzeichenbehafteten Datentyp zurückgegeben wird.

18.5.2 SFC 75 - SET_ADDR - PROFIBUS MAC-Adresse setzen

Beschreibung

Mit diesem SFC können Sie die MAC-Adresse der integrierten PROFIBUS-Schnittstelle einer CPU ändern. Die Funktion ist nur in der Betriebart passiver DP-Slave möglich. Zur Identifikation dient die Diagnoseadresse. Der SFC arbeitet asynchron und kann nur auf eine Schnittstelle angewendet werden. Bei STOP und anschließendem Warmstart bleibt die eingestellte Netzadresse erhalten. Bei PowerOFF-PowerON und bei Urlöschen erhält die Schnittstelle wieder ihre projektierte Teilnehmernummer. Der DP-Slave nimmt konsequent die Identität des DP-Slaves mit der neuen Adresse an. Gegenüber dem DP-Master fällt der DP-Slave mit der alten Adresse aus und ein DP-Slave mit der neuen Adresse kehrt wieder. Wird eine Adresse gewählt, die schon ein anderer Teilnehmer am DP-Strang besitzt, fallen beide Slaves bezogen auf die DP-Kommunikation aus.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
REQ	INPUT	BOOL	E, A, M, D, L	Funktionsanstoß mit *REQ* = 1
LADDR	INPUT	WORD	E, A, M, D, L	Identifikation der Schnittstelle
ADDR	INPUT	BYTE	E, A, M, D, L	Neue Teilnehmeradresse
RET_VAL	OUTPUT	INT	E, A, M, D, L	Fehlercode
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: In Bearbeitung

RET_VAL (Rückgabewert)

Wert	Beschreibung
0000h	Der Auftrag wurde fehlerfrei durchgeführt
7000h	Aufruf mit *REQ* = 0 (Aufruf ohne Bearbeitung), *BUSY* hat den Wert 0, es ist keine Datenübertragung aktiv
7001h	Erstaufruf mit *REQ* = 1: Datenübertragung angestoßen; *BUSY* hat den Wert 1
7002h	Zwischenaufruf (*REQ* irrelevant): Datenübertragung bereits aktiv; *BUSY* hat den Wert 1
8xyyh	Allgemeine Fehlerinformation *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*
8090h	Identifikation der Schnittstellen: Logische Adresse ist ungültig
8091h	Neue Teilnehmeradresse ist ungültig
8093h	Identifikation der Schnittstellen: Logische Adresse ist keine Schnittstelle
809Bh	Funktion nicht durchführbar (z.B. Schnittstelle ist kein DP-Slave oder aktiv)
80C3h	Ressourcenmangel (z.B. Mehrfachaufruf des SFC)

18.5.3 FC/SFC 193 - AI_OSZI - Oszilloskop-/FIFO-Funktion

Beschreibung

Der FC/SFC 193 dient der Ansteuerung der Oszilloskop-/FIFO-Funktion von analogen Eingabe-Kanälen, welche diese Funktionalität besitzen. Er ermöglicht das Starten der Aufzeichnung und das Auslesen der aufgezeichneten Daten. Je nach Parametrierung ergeben sich folgende Möglichkeiten:

Oszilloskop-Betrieb

- Je nach Trigger-Bedingung können Sie bei Flankenauswertung die Überwachung des eingestellten Kanals starten bzw. im manuellen Betrieb die Aufzeichnung starten.
- Sobald der Speicher voll ist haben Sie mit dem FC/SFC 193 Zugriff auf die aufgezeichneten Messwerte.

FIFO-Betrieb

- Die Aufzeichnung starten.
- Jederzeit den Puffer lesen.

Hinweis!

Der Aufruf des FC/SFC darf nur aus einer Prioritätsebene erfolgen, zum Beispiel nur aus OB 1 oder nur aus OB 35.

Das Modul muss zuvor parametriert werden.

Zum Starten und zum Auslesen ist jeweils ein Aufruf des FC/SFC 193 erforderlich. Die Unterscheidung der beiden Aufruf-Varianten erfolgt im Parameter MODE.

Parameter

Parameter	Deklaration	Datentyp	Funktion in Abhängig von MODE
REQ	IN	BOOL	Funktion ausführen (Starten/Auslesen)
LADR	IN	WORD	Basisadresse des Moduls
MODE	IN	WORD	Modus (Starten/Auslesen)
CHANNEL	IN	BYTE	Kanal, der ausgelesen werden soll
OFFSET	IN	DWORD	Adress-Offset beim Auslesen (nicht im FIFO-Betrieb)
RECORD	IN	ANY	Bereich für die ausgelesenen Daten
RETVAL	OUT	WORD	Rückgabewert (0 = OK)
BUSY	OUT	BOOL	Funktion wird bearbeitet
TIMESTAMP	OUT	DWORD	Zeitstempel (nur bei Flankenauswertung)
LEN	INOUT	DWORD	Anzahl der Werte, die pro Kanal zu bearbeiten sind

REQ

- Abhängig vom eingestellten *MODE* lässt sich durch Setzen dieses Bits die Aufzeichnung starten bzw. das Auslesen beginnen.
- Je nach Trigger-Bedingung wird bei Flankenauswertung die Überwachung des eingestellten Kanals oder im manuellen Betrieb die Aufzeichnung gestartet.
- Ist unter *MODE* der Befehl "Auslesen" eingestellt, werden die Daten aus dem Modul gelesen.

LADR

Logische Basisadresse des Moduls.

MODE

Den FC/SFC 193 können Sie in 3 verschiedenen Modi aufrufen. Den entsprechenden Modus geben Sie über *MODE* vor. Durch Setzen von *REQ* wird der entsprechende Modus ausgeführt. Folgende Werte werden unterstützt:

- 01h: Je nach Parametrierung Aufzeichnung starten bzw. Flankenüberwachung starten
- 00h: Daten über mehrere Zyklen lesen bis *BUSY* = 0 erfolgt.
- 80h: Daten in einem Zugriff lesen

CHANNEL

Hier wird der Kanal angegeben, der ausgelesen werden soll. Mit jedem Aufruf kann nur jeweils ein Kanal ausgelesen werden. Für Start-Aufrufe mit *MODE* = 01h ist dieser Parameter irrelevant.

OFFSET

- Der Adress-Offset gibt einen Offset-Wert der Adresse beim Auslesen an. Dies ermöglicht den Zugriff auf Teilbereiche der aufgezeichneten Daten.
- Der Wert für den maximale Offset-Wert hängt von der Anzahl der pro Kanal aufgezeichneten Werte ab.
- Im FIFO-Betrieb wird *OFFSET* nicht unterstützt und deshalb dieser Parameter ignoriert.

RECORD

- Hier können Sie einen Bereich definieren, in dem die gelesenen Werte zu speichern sind.
- Im FIFO-Betrieb werden hier alle Werte des eingestellten Kanals ausgelesen, die zum Zeitpunkt des Auslesens aufgezeichnet wurden.
- Bitte tragen Sie hierfür Sorge, dass der Puffer eine ausreichende Größe zur Aufnahme der Daten besitzt, ansonsten erhalten Sie eine Fehlermeldung.

BUSY

- *BUSY* = 1 zeigt an, dass die entsprechende Funktion bearbeitet wird.
- Mit *BUSY* = 0 ist die Bearbeitung der Funktion abgeschlossen.

TIMESTAMP

- In jedem SPEED-Bus-Modul läuft eine interne Uhr mit der Auflösung von 1µs mit.
- Der Rückgabewert entspricht der Uhrzeit auf dem SPEED-Bus- Modul, bei der das Trigger-Ereignis eingetreten ist.
- *TIMESTAMP* ist ausschließlich im flankengesteuerten Oszilloskop-Betrieb verfügbar.
- Er ist gültig solange der Auftrag läuft (*RETVAL* = 7xxxh) bzw. wenn dieser ohne Fehler beendet wurde (*RETVAL* = 0000h).

LEN

Der als IN/OUT realisierte Längenparameter wird beim Funktionsaufruf in den unterschiedlichen Modi verschieden interpretiert.

Modus: starten (*MODE*: = 01h)

Unter *MODE* = 01h kommt dieser Parameter ausschließlich bei manuellem Oszilloskop-Start zum Einsatz. Hier übergeben Sie die gewünschte Anzahl der Werte, die pro Kanal aufzuzeichnen sind. In diesem Modus liefert *LEN* keinen Wert zurück.

Modus: auslesen (*MODE*: = 00h oder 80h)

Bei *MODE* = 00h bzw. 80h geben Sie hier die Anzahl der Werte an, die auszulesen sind. Im FIFO-Betrieb wird dieser Parameter beim Aufruf nicht berücksichtigt. *LEN* liefert als Rückgabewert die Anzahl der Werte, die ausgelesen wurden.

RETVAL (Rückgabewert)

Zusätzlich zu den hier aufgeführten modulspezifischen Fehlercodes sind auch noch die allgemeingültigen Fehlercodes für FC/SFCs als Rückgabewert möglich.

RETVAL	Beschreibung in Abhängigkeit vom *BUSY*-Bit	BUSY
Byte		
0	Bit 1, 0:	
	00: Aufruf mit Request: = 0 (Leerlauf, warte auf *REQ* = 1).	0
	01: Erstaufruf mit *REQ*: = 1	1
	10: Folgeaufruf mit *REQ*: = 1	1
	11: Oszilloskop zeichnet gerade auf.	1
	Bit 2: Request: = 1, aber Aufzeichnung wurde noch nicht gestartet (*MODE*: = 00h oder *MODE*: = 80h)	0
	Bit 3: reserviert	-
	Bit 4: Trigger-Ereignis eingetreten und Aufzeichnung läuft	1
	Bit 5: Warte auf Trigger-Ereignis	1
	Bit 7…6: reserviert	-
1	Bit 0: reserviert	-
	Bit 1: Die Anzahl der aufgezeichneten Werte ist größer als die Länge des durch *RECORD* aufgespannten Zielbereichs (in Worten).	0
	Bit 2: Die Anzahl der aufgezeichneten Werte ist größer als die übergebene Länge am Parameter *LEN* und dem *OFFSET*.	0
	Bit 3: Im FIFO-Betrieb ist der Puffer übergelaufen.	0
	Bit 7...4:	
	0000: Auftrag beendet ohne Fehler	0
	0111: Auftrag läuft	1
	1000: Auftrag beendet mit Fehler	0

Auftrag beendet ohne Fehler

RETVAL	Beschreibung in Abhängigkeit vom *BUSY*-Bit	BUSY
0000h	Auftrag wurde ohne Fehler durchgeführt	0

Auftrag beendet mit Fehler

RETVAL	Beschreibung in Abhängigkeit vom *BUSY*-Bit	BUSY
8002h:	Die Oszilloskop-/FIFO-Funktion ist nicht projektiert.	0
8003h:	Es ist ein interner Fehler aufgetreten - kontaktieren Sie YASKAWA.	0
8005h:	Der angegebene Kanal kann nicht ausgelesen werden - falsche Kanal-Nummer.	0
8007h:	Der Wert unter *OFFSET* ist größer als die Anzahl der aufgezeichneten Werte.	0
8090h:	Es ist kein SPEED-Bus-Modul unter dieser Adresse verfügbar.	0
80D2h:	*LADR* liegt außerhalb des Peripherieadressbereichs.	0

18.5.4 FC/SFC 194 - DP_EXCH - Datenaustausch mit CP342S

Beschreibung

Mit dem FC/SFC 194 können Sie Daten zwischen Ihrer CPU und einem über SPEED-Bus angebunden PROFIBUS-DP-Master austauschen. Normalerweise blendet jeder PROFIBUS-DP-Master seinen E/A-Bereich im Peripherie-Bereich der CPU ein. Hierbei können Sie über die Hardware- Konfiguration einen Peripherie-Bereich von 0 ... 2047 adressieren. Da dies die maximale Anzahl an PROFIBUS-DP-Master-Modulen am SPEED-Bus einschränkt, haben Sie die Möglichkeit das Mapping an dem entsprechenden DP-Master zu deaktivieren und statt dessen den Zugriff über Hantierungsbaustein zu aktivieren. Hierbei können Sie mit dem FC/SFC 194 Daten von der CPU in einen definierten Bereich des DP-Master schreiben und Daten aus einem definierten Bereich des DP Master lesen.

Parameter

Parameter	Deklaration	Datentyp	Funktion in Abhängig von MODE
LADR	IN	WORD	Basisadresse des DP-Master-Moduls am SPEED-Bus
MODE	IN	WORD	Modus (0 = lesen / 1 = schreiben)
LEN	IN	WORD	Länge des Datenbereichs im DP-Master
OFFSET	IN	DWORD	Beginn des Datenbereichs im DP-Master
RETVAL	OUT	WORD	Rückgabewert (0 = OK)
DATA	IN OUT	ANY	Zeiger auf Datenbereich in der CPU

LADR

Logische Basisadresse des Moduls.

MODE

Den FC/SFC 194 können Sie mit folgenden Modi aufrufen:

- 0000 = Daten transferieren von DP-Master in die CPU.
- 0001 = Daten transferieren von der CPU in den DP-Master.

LEN

Hier definieren Sie die Länge des Datenbereichs im DP-Master.

OFFSET

Definieren Sie hier den Beginn des Datenbereichs im DP-Master. Bitte beachten Sie, dass der über *OFFSET* und *LEN* definierte Bereich den über die Hardware-Konfiguration parametrierten Bereich im DP-Master nicht überschreitet.

RETVAL (Rückgabewert)

Zusätzlich zu den hier aufgeführten modulspezifischen Fehlercodes sind auch noch die allgemeingültigen Fehlercodes für FC/SFCs als Rückgabewert möglich. ↳ *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

RETVAL	Beschreibung
0000h	Kein Fehler
8001h	*LADR* konnte keinem DP-Master am SPEED-Bus zugeordnet werden.
8002h	Wert des Parameters *MODE* ist außerhalb der Grenzen.
8003h	Wert des Parameters *LEN* ist 0.
8004h	Wert des Parameters *LEN* ist größer als der unter DATA definierte Datenbereich.

RETVAL	Beschreibung
8005h	Der über *OFFSET* und *LEN* definierte Bereich liegt außerhalb 0 …2047.
8006h	Der über *LADR* definierte DP-Master ist nicht für den Zugriff über Hantierungsbaustein parametriert. Aktivieren Sie in den Eigenschaften des DP-Master "IO-Mode HTB".
8008h	Lücke(n) im Eingangsbereich vorhanden.
8009h	Lücke(n) im Ausgangsbereich vorhanden.
8010h	Fehler beim Zugriff auf Eingabebereich (z.B. DP-Master ist nicht erreichbar)
8011h	Fehler beim Zugriff auf Ausgabebereich (z.B. DP-Master ist nicht erreichbar)
8Fxxh	DATA fehlerhaft (xx) ↳ *Kap. 6.1 "Allgemeine und spezifische Fehlercodes RET_VAL" Seite 269*

18.5.5 FC/SFC 219 - CAN_TLGR - CANopen-Kommunikation

FC/SFC 219 CAN_TLGR SDO-Anforderung an CAN-Master

Jede SPEED7-CPU hat den FC/SFC 219 integriert. Hiermit können Sie von Ihrem SPS-Programm auf Ihrem CAN-Master einen SDO- Lese- oder Schreibzugriff auslösen. Hierbei adressieren Sie den Master über die Steckplatz-Nr. und den Ziel- Slave über seine CAN-Adresse. Die Prozessdaten bestimmen Sie durch Angabe von *INDEX* und *SUBINDEX*. Über SDO kann pro Zugriff maximal ein Datenwort Prozessdaten übertragen werden.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
REQUEST	IN	BOOL	Funktion aktivieren
SLOT_MASTER	IN	BYTE	SPEED-Bus Steckplatz (101 ... 116)
NODEID	IN	BYTE	CAN-Adresse (1 ... 127)
TRANSFERTYP	IN	BYTE	Transfertyp
INDEX	IN	DWORD	CANopen Index
SUBINDEX	IN	DWORD	CANopen Subindex
CANOPENERROR	OUT	DWORD	CANopen Fehler
RETVAL	OUT	WORD	Rückgabewert (0 = OK)
BUSY	OUT	BOOL	Funktion wird bearbeitet
DATABUFFER	INOUT	ANY	Datenpuffer für FC/SFC-Kommunikation

REQUEST Steuerparameter: 1: Anstoß des Auftrags

SLOT_MASTER 101...116: Steckplatz 1 ... 16 von Master auf SPEED-Bus

NODELD Adresse des CANopen Knotens (1...127)

TRANSFERTYPE

40h: Lesen SDO	23h: Schreiben SDO (1 DWORD)
	2Bh: Schreiben SDO (1 WORD)
	2Fh: Schreiben SDO (1 BYTE)

INDEX CANopen Index

SUBINDEX CANopen Subindex

SLOT_MASTER

0:	System 200 CPU 21xCAN
1...32:	System 200 IM 208CAN
101...115:	System 300S 342-1CA70

CANOPENERROR Liegt kein Fehler vor, so liefert *CANOPENERROR* eine 0 zurück. Im Fehlerfall beinhaltet *CANOPENERROR* eine der nachfolgend aufgeführten Fehlermeldungen, die vom CAN-Master generiert wird:

Code	Beschreibung
0503 0000h	Toggle-Bit nicht geändert
0504 0000h	SDO Protokoll Time-out
0504 0001h	Client/Server Befehlsspezifizierung nicht gültig oder unbekannt
0504 0002h	Ungültige Blockgröße (nur Block-Modus)
0504 0003h	Ungültige Sequenznummer (nur Block-Modus)
0504 0004h	CRC Fehler (nur Block-Modus)
0504 0005h	Unzureichender Speicher
0601 0000h	Lesezugriff auf ein Nur-Schreiben-Objekt
0601 0001h	Schreibzugriff auf ein Nur-Lesen-Objekt
0602 0000h	Objekt nicht im Objektverzeichnis vorhanden
0604 0041h	Objekt kann nicht ins PDO gemappt werden
0604 0042h	Anzahl und Länge der zu mappenden Objekte überschreitet PDO-Länge
0604 0043h	Generelle Parameterinkompatibilität
0604 0047h	Generelle interne Inkompatibilität im Gerät
0606 0000h	Zugriffsfehler wegen Hardwareausfall
0607 0010h	Datentyp nicht korrekt, Länge der Serviceparameter nicht korrekt
0607 0012h	Datentyp nicht korrekt, Serviceparameter zu lang
0607 0013h	Datentyp nicht korrekt, Serviceparameter zu kurz
0609 0011h	Subindex existiert nicht
0609 0030h	Wertebereich der Parameter überschritten (nur für Schreibzugriff)
0609 0031h	Zu schreibender Parameterwert ist zu hoch

Code	Beschreibung
0609 0032h	Zu schreibender Parameterwert ist zu niedrig
0609 0036h	Maximumwert ist kleiner als Minimumwert
0800 0000h	Genereller Fehler
0800 0020h	Die Daten können entweder nicht transferiert oder nicht in der SPS gespeichert werden.
0800 0021h	Die Daten können wegen lokaler Kontrollen entweder nicht transferiert oder nicht in der SPS gespeichert werden.
0800 0022h	Die Daten können wegen aktuellem Modulstatus entweder nicht transferiert oder nicht in der SPS gespeichert werden.
0800 0023h	Dynamische Objektverzeichnisgenerierung fehlgeschlagen oder kein Objektverzeichnis gefunden (z.B. Objektverzeichnis wird aus Datei generiert und ein Dateifehler ist aufgetreten).

RETVAL

Wird die Funktion fehlerfrei ausgeführt, enthält der Rückgabewert die gültige Länge der Antwortdaten: 1: Byte, 2: Wort, 4: Doppelwort Tritt während der Bearbeitung der Funktion ein Fehler auf, enthält der Rückgabewert einen der nachfolgend aufgeführten Fehler-codes.

Code	Beschreibung
F021h	Ungültige Slave-Adresse (Aufrufparameter gleich 0 oder größer 127)
F022h	Ungültiger Transfertyp (Wert ungleich 40h, 23h, 2Bh, 2Fh)
F023h	Ungültige Datenlänge (der Datenpuffer ist zu klein, beim SDO-Lesezugriff sollte dieser mindestens 4Byte groß sein, beim SDO-Schreibzugriff sollte dieser 1Byte, 2Byte oder 4Byte groß sein)
F024h	Der FC/SFC wird nicht unterstützt.
F025h	Schreibpuffer im CANopen-Master ist voll, Service kann zur Zeit nicht bearbeitet werden.
F026h	Lesepuffer im CANopen-Master ist voll, Service kann zur Zeit nicht bearbeitet werden.
F027h	Der SDO-Lese- oder Schreibzugriff wurde fehlerhaft beantwortet ↳ *"CANOPENERROR" Seite 1153*.
F028h	SDO-Timeout (es wurde kein CANopen-Teilnehmer mit der Node-ID gefunden).

BUSY

Solange *BUSY* = 1 ist der aktuelle Auftrag ist noch nicht beendet.

DATABUFFER

- Datenbereich, über den der FC/SFC kommuniziert. Geben Sie hier einen ANYPointer vom Typ Byte an.
- SDO-Lesezugriff: Zielbereich für die gelesenen Nutzdaten.
- SDO-Schreibzugriff: Quellbereich für die zu schreibenden Nutzdaten.

Sofern eine SDO-Anforderung fehlerfrei abgearbeitet wurde, enthält RETVAL die Länge der gültigen Antwortdaten in (1, 2 oder 4Byte) und CANOPENERROR den Wert 0.

18.5.6 FC/SFC 254 - RW_SBUS - IBS-Kommunikation

Beschreibung

Dieser Baustein dient den INTERBUS-FCs 20x als Kommunikationsbaustein zwischen INTERBUS-Master und CPU. Für den Einsatz der INTERBUS-FCs 20x ist der FC/SFC 254 als Baustein in Ihr Projekt einzubinden.

Parameter

Parameter	Deklaration	Datentyp	Beschreibung
READ/WRITE	IN	Byte	0 = Lesen, 1 = Schreiben
LADDR	IN	WORD	Logical Adresse INTERBUS-Master
IBS_ADDR	IN	WORD	Adresse INTERBUS-Master
DATAPOINTER	IN	ANY	Zeiger auf Datenbereich in der CPU
RETVAL	OUT	WORD	Rückgabewert (0 = OK)

READ/WRITE

Hiermit bestimmen Sie die Transferrichtung aus CPU-Sicht. Mit *READ* lesen Sie Daten aus dem Dual-port-memory des INTERBUS-Master.

LADDR

Geben Sie hier die Adresse (**L**ogical **Addr**ess) an, ab der das Register des Masters in der CPU eingeblendet wird. Beim Hochlauf der CPU werden, sofern keine Hardware-Konfiguration vorliegt, die INTERBUS-Master nach folgender Formel im E/A-Adress-Bereich der CPU abgelegt:

Anfangsadresse = 256 = (Steckplatz-101)+2048

Die Steckplatz-Nummerierung am SPEED-Bus beginnt bei 101 links der CPU und geht von rechts nach links. Beispielsweise hat der 1. Steckplatz die Adresse 2048, der 2. den Steckplatz 2304 usw.

IBS_ADDR

Adresse im Adressraum des INTERBUS-Master.

DATAPOINTER

Zeiger auf Datenbereich in der CPU.

RETVAL

Wert, den die Funktion zurück liefert. Bei 0 ist alles OK.

19 Index

A

Abkürzungen 29
Achskontrolle 751
ErrorID 851
HMI 825
Zustände 845
Adressierungsbeispiele 36
Anlauf 273
Arithmetische Operationen 39
Asynchrone Fehleralarme 284
AWL-Operationen 25

B

Baustein-Operationen 44
Bausteinparameter 269
Befehlsliste 25
Beispiele 22
Bildaufbau-Operationen 45

C

CP040 418
CP240 427

D

Darstellung 22
Datei-Funktionen SPEED7-CPUs 1128
Datei-Funktionen Standard-CPUs 1141
Datentyp 459
DATE_AND_TIME 1022
DS-ID 459

E

Ein-/Ausgabe 1040
Energiemessung 458
ErrorID 851
EtherCAT-Kommunikation 434
Ethernet-Kommunikation 335

F

FB 1 414
FB 7 . . . . . 415, 1146
FB 8 . . . . . 352, 416
FB 9 354
FB 12 . . . . . 356, 952
FB 13 . . . . . 359, 955
FB 14 . . . . . 361, 957
FB 15 . . . . . 363, 959
FB 20 1040
FB 21 1040
FB 22 1041
FB 23 1042
FB 41 1054
FB 42 1061
FB 43 1066
FB 45 306
FB 46 307
FB 47 309
FB 48 310
FB 49 313
FB 50 315
FB 52 434
FB 53 438
FB 55 365
FB 58 1075
FB 59 1093
FB 60 419
FB 61 421
FB 63 319
FB 64 322
FB 65 . . . . . 326, 424
FB 66 333
FB 70 380
FB 71 383
FB 72 387
FB 73 390
FB 80 1012
FB 240 475
FB 241 475
FB 320 467
FB 321 470
FB 325 461
FB 700 486
FB 701 487
FB 702 489
FB 703 491
FB 704 493

FB 705 495
FB 706 549
FB 707 559
FB 708 497
FB 709 562
FB 710 564
FB 711 500
FB 712 502
FB 713 504
FB 714 506
FB 715 507
FB 716 509
FB 717 510
FB 718 512
FB 719 513
FB 720 566
FB 721 568
FB 722 515
FB 723 517
FB 724 518
FB 725 519
FB 726 521
FB 727 523
FB 728 525
FB 729 527
FB 730 529
FB 731 531
FB 732 533
FB 733 535
FB 734 537
FB 735 539
FB 736 541
FB 737 543
FB 738 544
FB 739 545
FB 740 547
FB 800 759
FB 801 761
FB 802 763
FB 803 765
FB 804 767
FB 805 769
FB 808 771
FB 811 774
FB 812 776
FB 813 778
FB 814 780
FB 815 782
FB 816 784
FB 817 786
FB 818 788
FB 819 790
FB 823 792
FB 824 794
FB 825 795
FB 826 797
FB 827 799
FB 828 801
FB 829 803
FB 830 805
FB 831 807
FB 832 809
FB 833 811
FB 834 813
FB 835 815
FB 836 817
FB 837 820
FB 838 822
FB 860 754
FB 870 596
FB 871 596
FB 872 . . . . . 617, 639
FB 873 617
FB 874 639
FB 875 686
FB 876 726
FB 877 728
FB 878 728
FB 879 728
FB 880 729
FB 881 730
FB 882 732
FB 885 702
FB 886 750
FB 887 750
FB 890 672
FB 891 676
FC 0 427

FC 1 . . . . . 429, 1023
FC 2 1023
FC 3 1023
FC 4 1024
FC 5 . . . . . 337, 1024
FC 6 . . . . . 340, 1025
FC 7 1025
FC 8 . . . . . 430, 1025
FC 9 . . . . . 431, 1026
FC 10 . . . . . 342, 1026
FC 11 . . . . . 432, 1026
FC 12 1027
FC 13 1027
FC 14 1028
FC 15 1028
FC 16 1028
FC 17 1029
FC 18 1029
FC 19 1030
FC 20 1030
FC 21 1030
FC 22 1031
FC 23 1031
FC 24 1032
FC 25 . . . . . 884, 1032
FC 26 1033
FC 27 1033
FC 28 1034
FC 29 1035
FC 30 1035
FC 31 1035
FC 32 1036
FC 33 1036
FC 34 1037
FC 35 1037
FC 36 1038
FC 37 1038
FC 38 1039
FC 39 1039
FC 40 1039
FC 53 1146
FC 54 904
FC 61 1101
FC 62 . . . . . 350, 1102
FC 63 1103
FC 93 1013
FC 94 1014
FC 95 1015
FC 96 1015
FC 97 1016
FC 98 1016
FC 99 1017
FC 105 1017
FC 106 1018
FC 108 1019
FC 109 1020
FC 110 1021
FC 111 1021
FC 112 1044
FC 113 1045
FC 114 1045
FC 115 1046
FC 116 1047
FC 117 1048
FC 118 1048
FC 119 1049
FC 120 1050
FC 121 1051
FC 122 1051
FC 123 1052
FC 124 1053
FC 125 1053
FC 193 1147
FC 194 1151
FC 195 . . . . . 1128, 1129
FC 208 . . . . . 1128, 1131
FC 209 . . . . . 1128, 1132
FC 210 . . . . . 1128, 1133
FC 211 . . . . . 1128, 1134
FC 212 . . . . . 1128, 1135
FC 213 . . . . . 1128, 1136
FC 214 . . . . . 1128, 1137
FC 215 . . . . . 1128, 1139
FC 216 403
FC 217 406
FC 218 412
FC 219 1152
FC 254 1155

FC 300 . . . 442
FC 301 . . . 444
FC 302 . . . 446
FC 303 . . . 448
FC 310 . . . 450
FC 311 . . . 452
FC 312 . . . 454
FC 313 . . . 456
Fetch/Write . . . 1106
FKT Modbus-Funktionscodes . . . 396
Flanken-Operationen . . . 46
FR-ID . . . 459
Frame . . . 459
Frequenzmessung . . . 442
Frequenzumrichter
EtherCAT . . . 735
Modbus RTU . . . 705
PWM . . . 694
FUP-Operationen . . . 75

G

Gebäude-Automatisierung . . . 305
Gegenüberstellung der Syntaxsprachen . . . 32

H

Handlungsanweisungen . . . 22
Hauptprogramm . . . 272
Hinweise . . . 22

I

ID . . . 458
IEC-Funktionen . . . 1022
Integrierte Standardfunktionen . . . 859
Interne Bausteine . . . 24

K

Kennzeichnungen . . . 23
Kommunikationsalarme . . . 275
Konvertierung . . . 1012
KOP-Operationen . . . 172

L

Lade-Operationen . . . 47

M

Messgröße . . . 458
Messwert . . . 458
Modbus Exception Codes . . . 395
Modbus-Kommunikation . . . 380
Modulspezifisch . . . 442
Motion-Module . . . 465

N

Netzwerkkommunikation . . . 318
Null-Operationen . . . 45

O

OB 1 . . . 272
OB 10 . . . 279
OB 11 . . . 279
OB 20 . . . 278
OB 21 . . . 278
OB 28 . . . 281
OB 29 . . . 281
OB 32 . . . 281
OB 33 . . . 281
OB 34 . . . 281
OB 35 . . . 281
OB 40 . . . 283
OB 41 . . . 283
OB 55 . . . 275
OB 56 . . . 276
OB 57 . . . 277
OB 60 . . . 303
OB 61 . . . 303
OB 80 . . . 284
OB 81 . . . 287
OB 82 . . . 288
OB 83 . . . 290
OB 85 . . . 293
OB 86 . . . 297
OB 100 . . . 273
OB 102 . . . 273
OB 121 . . . 299
OB 122 . . . 302
Offene Kommunikation . . . 318
Operationen
0 . . . 170, 268
ABS . . . 122, 221
ACOS . . . 129, 228

ADD_DI 114, 211
ADD_I 110, 208
ADD_R 119, 217
ASIN 129, 227
ATAN 130, 229
Ausgang rücksetzen 81, 177
Ausgang setzen 81, 178
BCD_DI 93, 190
BCD_I 91, 188
BIE 169, 267
CALL 132, 231
CEIL 102, 199
CMP 88, 89, 185, 186
COS 127, 226
DI_BCD 94, 191
DI_R 95, 192
DIV_DI 116, 214
DIV_I 113, 210
DIV_R 122, 220
Eingang einfügen 78
Eingang negieren 79
Ergebnisbit 170, 268
EXP 126, 224
Flankenwechsel 84, 85, 86, 87, 180, 181, 183
FLOOR 102, 200
I_BCD 92, 189
I_DI 93, 189
INV_DI 97, 194
INV_I 96, 193
JMP 135, 233
JMPN 136, 234
Konnektor 80, 176
LABEL 136, 235
LN 125, 223
MOD_DI 117, 215
MOVE 131, 230
MUL_DI 115, 213
MUL_I 112, 210
MUL_R 121, 219
N 84, 180
NEG 86, 183
NEG_DI 98, 195
NEG_I 97, 194
NEG_R 99, 196
ODER 76
Öffnerkontakt 174
OPN 133, 134, 232
OS 167, 266
OV 166, 265
P 85, 181
POS 87, 183
RET 133, 232
ROL_DW 143, 242
ROR_DW 144, 243
ROUND 100, 197
RS 82, 178
S_AVERZ 153, 251
S_EVERZ 149, 248
S_IMPULS 146, 244
S_SEVERZ 151, 249
S_VIMP 147, 246
SA 158, 257
SAVE 85, 182
Schließerkontakt 173
SE 156, 255
SHL_DW 141, 240
SHL_W 140, 238
SHR_DI 139, 237
SHR_DW 142, 241
SHR_I 138, 236
SHR_W 140, 239
SI 154, 253
SIN 126, 225
SQR 124, 222
SQRT 123, 222
SR 83, 179
SS 157, 256
SUB_DI 114, 212
SUB_I 111, 209
SUB_R 120, 218
SV 155, 254
SZ 108, 205
TAN 128, 226
TRUNC 101, 198
UND 77
UO 168, 266
Verknüpfungsergebnis invertieren 175
VKE 79, 176

WAND_DW 163, 261
WAND_W 160, 258
WOR_DW 164, 262
WOR_W 161, 259
WXOR_DW 165, 263
WXOR_W 162, 260
XOR 78, 174
Z_RUECK 106, 204
Z_VORW 105, 202
ZAEHLER 104, 201
ZR 109, 207
Zuweisung 79, 176
ZV 108, 206
Organisationsbausteine 272

P

PID 1054
PLCopen Parameter 823
Prozessalarme 283

R

RAM nach WLD 475
Raumsteuerung 306
Register 35
RET_VAL 269
RTU 387
Rücksetzen von Bitoperanden 52

S

S5 Converting 1044
Schiebe-Operationen 50
SDO-Kommunikation 434
Serielle Kommunikation 402
Setzen von Bitoperanden 52
SFB 0 944
SFB 1 945
SFB 2 946
SFB 3 948
SFB 4 949
SFB 5 951
SFB 7 1146
SFB 8 352
SFB 9 354
SFB 12 . . . . . 356, 952
SFB 13 . . . . . 359, 955
SFB 14 . . . . . 361, 957
SFB 15 . . . . . 363, 959
SFB 31 961
SFB 32 963
SFB 33 967
SFB 34 970
SFB 35 971
SFB 36 974
SFB 47 976
SFB 48 981
SFB 49 983
SFB 52 991
SFB 53 992
SFB 54 993
SFC 0 859
SFC 1 859
SFC 2 860
SFC 3 . . . . . 860, 861
SFC 4 . . . . . 860, 862
SFC 5 862
SFC 6 864
SFC 7 866
SFC 12 867
SFC 13 871
SFC 14 873
SFC 15 874
SFC 17 876
SFC 18 876
SFC 19 878
SFC 20 879
SFC 21 880
SFC 22 881
SFC 23 883
SFC 24 884
SFC 25 884
SFC 28 885
SFC 29 . . . . . 885, 886
SFC 30 . . . . . 885, 887
SFC 31 . . . . . 885, 887
SFC 32 888
SFC 33 889
SFC 34 890
SFC 36 891
SFC 37 892

SFC 38 892
SFC 39 893
SFC 40 894
SFC 41 895
SFC 42 896
SFC 43 896
SFC 44 897
SFC 46 897
SFC 47 897
SFC 49 898
SFC 50 899
SFC 51 900
SFC 52 902
SFC 53 1146
SFC 54 904
SFC 55 905
SFC 56 907
SFC 57 909
SFC 58 912
SFC 59 914
SFC 64 916
SFC 65 917
SFC 66 919
SFC 67 923
SFC 68 926
SFC 69 929
SFC 70 931
SFC 71 932
SFC 75 1147
SFC 81 935
SFC 101 936
SFC 102 937
SFC 105 938
SFC 106 941
SFC 107 942
SFC 108 942
SFC 193 1147
SFC 194 1151
SFC 195 . . . . . 1128, 1129
SFC 207 402
SFC 208 . . . . . 1128, 1131
SFC 209 . . . . . 1128, 1132
SFC 210 . . . . . 1128, 1133
SFC 211 . . . . . 1128, 1134
SFC 212 . . . . . 1128, 1135
SFC 213 . . . . . 1128, 1136
SFC 214 . . . . . 1128, 1137
SFC 215 . . . . . 1128, 1139
SFC 216 403
SFC 217 406
SFC 218 412
SFC 219 1152
SFC 220 1141
SFC 221 . . . . . 1141, 1143
SFC 222 . . . . . 1141, 1144
SFC 228 1106
SFC 230 . . . . . 1108, 1120
SFC 231 . . . . . 1108, 1121
SFC 232 . . . . . 1108, 1122
SFC 233 . . . . . 1108, 1123
SFC 234 . . . . . 1108, 1124
SFC 235 . . . . . 1108, 1125
SFC 236 . . . . . 1108, 1126
SFC 237 . . . . . 1108, 1127
SFC 238 . . . . . 1108, 1128
SFC 254 1155
Sicherheitshinweise 22
Sigma-5 EtherCAT 578
Sigma-5 PROFINET 643
Sigma-5/7 Pulse Train 679
Sigma-7 PROFINET 658
Sigma-7S EtherCAT 598
Sigma-7W EtherCAT 619
Software-Version 20
SPEED7 Programmierung 33
Sprung-Operationen 53
Standard-Bausteine 1012
Symbole
in der Anleitung 22
Synchronalarme 299
System-SFBs 944
System-SFCs 859
Systembausteine 1106
Systemfunktionen 1146
Systemfunktions-Blöcke 1146

T

Taktsynchronalarme 303

TCP 380
TimeFunctions 1100
Tipps 22
Transfer-Operationen 55

U

UDT 3 311
UDT 4 312
UDT 60 1100
UDT 65 328
UDT 321 474
UDT 325 462
UDT 860 . . . . . 754, 758
UDT 861 758
UDT 870 596
UDT 872 . . . . . 617, 639
UDT 877 726
UDT 878 726
UDT 879 726
UDT 881 726
UDT 886 750
UDT 890 672
UDT 8189 485
UDT 8190 485
UDT 8191 485
UDT 8192 485
Uhrzeitalarme 279
Umwandlungs-Operationen 60

V

Vergleichs-Operationen 62
Verknüpfungs-Operationen
    Bit 64
    Wort 72
Version (SPEED7 Studio) 20
Verzögerungsalarme 278

Z

Zähl-Operationen 74
Zeit-Operationen 73
Zugangskontrolle 310
Zyklusalarme 281