SEW EURODRIVE

Handbuch

Positionierung und Ablaufsteuerung IPOS$^{plus®}$

Ausgabe 11/2009

11645407 / DE

KKFU 144794 3
42G1

Inhaltsverzeichnis

1 Allgemeine Hinweise ... 14
1.1 Aufbau der Sicherheitshinweise ... 14
1.2 Mängelhaftung ... 15
1.3 Haftungsausschluss ... 15
1.4 Urheberrechtsvermerk ... 15
2 Sicherheitshinweise ... 16
2.1 Allgemein ... 16
2.2 Bestimmungsgemäße Verwendung ... 17
2.3 Zielgruppe ... 17
2.4 Fehlprogrammierung ... 17
3 Systembeschreibung ... 18
3.1 Einführung ... 18
3.1.1 Umfang dieser Dokumentation ... 18
3.1.2 Programmerstellung ... 19
3.2 Eigenschaften von IPOSplus® ... 19
3.3 Steuerung von IPOSplus®-Geräten ... 22
3.3.1 Aktive Steuerquelle ... 22
3.4 Technologie-Optionen / Applikationsmodule ... 22
3.4.1 Technologie-Optionen ... 22
3.4.2 Applikationsmodule ... 23
3.5 Technische Daten ... 25
3.5.1 MOVIDRIVE® B ... 25
3.5.2 MOVITRAC® B ... 25
3.5.3 MQx ... 26
3.6 Referenzdokumente ... 27
3.6.1 Allgemeine Handbücher ... 27
3.6.2 Handbücher Serielle Schnittstellen / Feldbusse ... 27
3.6.3 Handbücher für synchronisierte Achsbewegungen ... 27
3.6.4 Handbücher zu den Applikationsmodulen ... 27
3.6.5 Handbücher zu Feldbus-Schnittstellen MQx ... 27
4 IPOS-Variablen ... 28
4.1 Einführung ... 28
4.2 Übersicht über die Systemvariablen ... 29
5 Task-Verwaltung und Interrupts ... 40
5.1 Einführung ... 40
5.2 Task-Verwaltung bei MOVIDRIVE® A und B ... 40
5.3 Tasks bei MOVIDRIVE® A ... 43
5.4 Tasks bei MOVIDRIVE® B ... 43
5.4.1 Abarbeitungszeit Task 1 / Task 2 ... 43
5.4.2 Task 3 ... 44
5.4.3 Implementierungshinweise ... 44
5.4.4 Beispiel ... 45

5.5 Interrupts ... 45
5.5.1 Beispiel ... 46
5.6 Interrupts bei MOVIDRIVE® A und B ... 46
5.6.1 Aktivierung des Interrupts ... 46
5.6.2 Fehler-Interrupt ... 46
5.6.3 Touch-Probe-DI02 Interrupt ... 47
5.6.4 Timer0-Interrupt ... 48
5.7 Variablen-Interrupts bei MOVIDRIVE® B ... 49
5.7.1 Aufruf des Variablen-Interrupts ... 49
5.7.2 IPOS-Zugriff auf die interne Interrupt-Steuerung ... 50
6 Wegerfassung und Positionierung ... 53
6.1 Geberauswertung ... 53
6.2 Motorgeber (X15) ... 54
6.3 Geberkombinationen ... 54
6.4 Externer Geber (X14) ... 57
6.4.1 Positionierung auf externen Geber (X14) ... 57
6.4.2 Einsatzfall Schlupfausgleich mit externem Geber ... 57
6.5 SSI-Absolutwertgeber (DIP) ... 60
6.5.1 Inbetriebnahme ... 60
6.5.2 1. Gebertyp P950 wählen ... 60
6.5.3 2. Motordrehsinn P35_ einstellen ... 61
6.5.4 3. Zählrichtung P951 des SSI-Absolutwertgebers einstellen ... 61
6.5.5 4. Geberskalierung P955 einstellen ... 61
6.5.6 5. Positionsoffset P953 einstellen ... 61
6.5.7 6. Nullpunktoffset P954 einstellen ... 62
6.5.8 7. Geberfaktoren P942 und P943 einstellen ... 62
6.5.9 8. Quelle Ist-Position P941 einstellen ... 62
6.6 Referenzieren ... 63
6.6.1 Typ 0: Referenzfahrt auf Nullimpuls ... 67
6.6.2 Typ 1: Linkes Ende des Referenznockens ... 67
6.6.3 Typ 2: Rechtes Ende des Referenznockens ... 68
6.6.4 Typ 3: Endschalter rechts ... 69
6.6.5 Typ 4: Endschalter Links ... 69
6.6.6 Typ 5: Keine Referenzfahrt ... 70
6.6.7 Typ 6: Referenznocken bündig zum rechten Endschalter ... 70
6.6.8 Typ 7: Referenznocken bündig zum linken Endschalter ... 71
6.6.9 Typ 8: Ohne Freigabe ... 72
6.7 Modulo-Funktion ... 73
6.7.1 Einleitung ... 73
6.7.2 Funktionsweise ... 74
6.7.3 Verfahrstrategien ... 77
6.7.4 Projektierung ... 80
6.7.5 Projektierungsbeispiele ... 80
6.7.6 Häufig gestellte Fragen ... 83

6.8 Nockenschaltwerke ... 84
6.8.1 Standardnockenschaltwerk ... 85
6.8.2 Erweitertes Nockenschaltwerk ... 89
7 Wegerfassung über binäre Eingänge ... 96
7.1 Arten von Einbaugebern ... 96
7.2 Prinzip der Wegerfassung ... 96
7.3 Wegerfassung mit MOVIDRIVE® B ... 97
7.4 Wegerfassung mit MOVITRAC® B ... 98
7.5 Wegerfassung mit MQx ... 99
7.5.1 Näherungsgeberauswertung ... 99
7.5.2 Klemmenbelegung DI0 und DI1 ... 99
7.5.3 Wegerfassung mit Einbaugeber ... 100
7.5.4 Geberüberwachung ... 100
7.5.5 Abspeichern der Ist-Position ... 100
7.5.6 Counter ... 101
7.5.7 Anschluss der Einbaugeber ... 101
8 IPOSplus® und Feldbus ... 102
8.1 Einführung ... 102
8.2 Binäre Eingänge und Ausgänge ... 103
8.2.1 Feldbus-Schnittstelle, DIO und DIP ... 103
8.3 Zyklische Prozessdaten ... 103
8.3.1 Zyklische voreingestellte Prozessdaten ... 103
8.3.2 Zyklische anwenderspezifische Prozessdaten ... 104
8.4 Azyklische Kommunikation ... 105
8.5 Besonderheiten bei der Kommunikation über SBus ... 105
8.6 Besonderheiten bei der Kommunikation über RS-485 ... 106
8.7 Feldbus-Steuerworte und Feldbus-Statusworte ... 106
9 IPOSplus® und synchronisierte Bewegungen ... 107
9.1 Einführung ... 107
9.2 Drehzahlgleichlauf über Master-Slave-Funktion ... 107
9.3 Synchronlauf mit DRS-Optionskarte ... 107
9.3.1 Aktivieren und Deaktivieren der Funktion Freilauf ... 108
9.3.2 DRS11B Nullpunkt setzen ... 109
9.3.3 Aktivieren und Deaktivieren der Funktion Offset ... 111
9.3.4 Umschaltung zwischen Positionierung und Synchronlauf ... 113
9.4 Synchronlauf mit Technologie-Option "Interner Synchronlauf" ... 115
9.4.1 Voraussetzungen ... 115
9.5 Synchronlauf mit Technologie-Option "Kurvenscheibe" ... 116
9.5.1 Voraussetzungen ... 117
10 Besonderheiten bei IPOSplus® für MOVITRAC® B ... 118
10.1 Voraussetzungen ... 118
10.2 Funktionalität ... 119
11 Besonderheiten bei IPOSplus® für MQx ... 120
11.1 Einleitung ... 120
11.2 Start der Programmier-Tools ... 121

11.3 Ablaufsteuerung ... 121
11.4 Digitale Ein- und Ausgänge ... 121
11.5 Werte der Variablen DIAG11 beim Fehler IPOS ILLOP ... 122
12 P9xx IPOS-Parameter ... 123
12.1 P90x IPOS Referenzfahrt ... 123
12.1.1 P900 Referenzoffset ... 123
12.1.2 P901 Referenzdrehzahl 1 ... 124
12.1.3 P902 Referenzdrehzahl 2 ... 124
12.1.4 P903 Referenzfahrttyp ... 124
12.1.5 P904 Referenzierung auf Nullimpuls ... 126
12.1.6 P905 Hiperface Offset X15 ... 126
12.1.7 P906 Nockenabstand ... 126
12.2 P91x IPOSplus® Verfahrparameter ... 127
12.2.1 P910 Verstärkung X-Regler ... 127
12.2.2 P911 / 912 Positionier-Rampe 1 / 2 ... 127
12.2.3 P913 / P914 Verfahrdrehzahl RECHTS / LINKS ... 127
12.2.4 P915 Geschwindigkeitsvorsteuerung ... 127
12.2.5 P916 Rampenform ... 128
12.2.6 P917 Rampenmode ... 130
12.2.7 P918 Quelle Bussollwert ... 130
12.3 P92x IPOS-Überwachungen ... 131
12.3.1 P920 / P921 SW-Endschalter RECHTS / LINKS ... 131
12.3.2 P922 Positionsfenster ... 131
12.3.3 P923 Schleppfehlerfenster ... 131
12.3.4 P924 Erkennung Positionierunterbrechung ... 131
12.4 P93x IPOSplus®-Sonderfunktionen ... 132
12.4.1 P930 Override ... 132
12.4.2 P931 IPOS-STW. Task 1 ... 132
12.4.3 P932 IPOS-STW. Task 2 ... 132
12.4.4 P933 Ruckzeit ... 132
12.4.5 P938 Geschwindigkeit Task 1 ... 132
12.4.6 P939 Geschwindigkeit Task 2 ... 133
12.5 P94x IPOSplus®-Geber ... 134
12.5.1 P940 IPOS-Variablen Edit ... 134
12.5.2 P941 Quelle Ist-Position ... 134
12.5.3 P942 / P943 Geberfaktor Zähler / Nenner ... 134
12.5.4 P944 Geberskalierung Ext. Geber ... 135
12.5.5 P945 Streckengeber Typ (X14) ... 135
12.5.6 P946 Streckengeber Zählrichtung (X14) ... 136
12.5.7 P947 Hiperface Offset X14 ... 136
12.5.8 P948 Automatische Gebertauscherkennung ... 137

12.6 P95x Absolutwertgeber (SSI) ... 138
12.6.1 P950 Gebertyp ... 138
12.6.2 P951 Zählrichtung ... 138
12.6.3 P952 Taktfrequenz ... 139
12.6.4 P953 Positionsoffset ... 139
12.6.5 P954 Nullpunktoffset ... 139
12.6.6 P955 Geberskalierung ... 139
12.6.7 P956 Baudrate CAN-Geber ... 140
12.7 P96x IPOSplus® Modulofunktion ... 140
12.7.1 P960 Modulofunktion ... 140
12.7.2 P961 Modulo - Zähler ... 140
12.7.3 P962 Modulo - Nenner ... 140
12.7.4 P963 Modulo - Geberauflösung ... 140
12.8 P97x IPOS-Synchronisation ... 141
12.8.1 P970 DPRAM-Synchronisierung ... 141
12.8.2 P971 Synchronisierungsphase ... 141
13 Compiler – Editor ... 142
13.1 Technische Merkmale ... 142
13.2 Erste Schritte ... 143
13.2.1 Schritt 1: IPOSplus®-Compiler mit MOVITOOLS® MotionStudio starten ... 143
13.2.2 Schritt 2: Erstellung eines neuen Projekts ... 145
13.2.3 Schritt 3: Das erste IPOSplus®-Programm ... 148
13.2.4 Schritt 4: Compilieren und Starten des IPOSplus®-Programms ... 151
13.3 Einstellungen für den IPOSplus-®Compiler ... 154
13.4 Suchfunktion ... 157
13.5 Erstellen eines neuen Projekts ... 158
13.5.1 Projekteigenschaften festlegen ... 158
13.5.2 Programmstruktur festlegen ... 159
13.6 Projekt speichern ... 161
13.7 Aufbau einer Projektverwaltung ... 162
13.8 Projekt öffnen ... 164
13.9 Projekte handhaben mit MOVIDRIVE® B ... 164
13.9.1 Projekt in Umrichter speichern ... 164
13.9.2 Projekt aus Umrichter laden ... 165
13.9.3 Projekt aus Umrichter abrufen ... 165
13.10 Kompilieren eines Projekts ... 166
13.11 Kompilieren und Download ... 167
13.12 Starten eines Programms ... 167
13.13 Stoppen eines Programms ... 167
13.14 Vergleich mit Gerät ... 167
13.15 Debugger ... 168
13.16 Variablenfenster ... 169
13.17 Programminformationen ... 171
13.18 Anweisungseingabe ... 172

13.19 Kommentare ... 173
13.20 Symbolübersicht ... 174
14 Compiler - Programmierung ... 175
14.1 Präprozessor ... 176
14.2 Präprozessor-Anweisungen ... 176
14.3 #include ... 178
14.4 Verzeichnisse von include ... 179
14.5 #define ... 179
14.6 #undef ... 180
14.7 #declare ... 181
14.8 SEW-Standardstrukturen ... 182
14.9 Anwenderdefinierte Strukturen ... 184
14.10 long ... 186
14.11 initial long ... 186
14.12 #pragma ... 187
14.13 Erläuterung zur const.h und io.h / constb.h und iob.h ... 188
14.14 Bezeichner ... 190
14.15 Konstanten ... 190
14.16 IPOSplus®-Variablen im Compiler ... 191
14.16.1 Beispiel ... 191
14.17 Die Vereinbarung globaler Variablen ... 191
14.18 Indirekte Adressierung – Pointer ... 192
14.19 numof() ... 193
15 Compiler – Operatoren ... 194
15.1 Rangfolge der Operatoren ... 194
15.2 Unäre Operatoren ... 195
15.3 Binäre Operatoren ... 196
15.3.1 Beispiel ... 196
15.4 Ternäre Operatoren ... 196
15.4.1 Beispiel ... 196
16 Compiler – Konstrukte ... 197
16.1 if...else ... 197
16.1.1 Syntax ... 197
16.2 for ... 198
16.2.1 Syntax ... 198
16.3 while ... 199
16.3.1 Syntax ... 199
16.4 do...while ... 201
16.4.1 Syntax ... 201
16.5 switch...case...default ... 202
16.5.1 Syntax ... 202
16.6 return ... 203

17 Compiler – Funktionen ... 204
17.1 Anwenderdefinierte Funktionen ... 204
17.2 Befehlsübersicht Standardfunktionen ... 205
17.2.1 Bit-Standardfunktionen ... 205
17.2.2 Kommunikations-Standardfunktionen ... 206
17.2.3 Positionier-Standardfunktionen ... 206
17.2.4 Programm-Standardfunktionen ... 206
17.2.5 Setz-Standardfunktionen ... 207
17.2.6 Spezielle Geräte-Standardfunktionen ... 207
17.3 Standardfunktionen ... 208
17.3.1 _AxisStop ... 208
17.3.2 _BitClear ... 209
17.3.3 _BitMove ... 209
17.3.4 _BitMoveNeg ... 209
17.3.5 _BitSet ... 210
17.3.6 _Copy ... 210
17.3.7 _FaultReaction ... 210
17.3.8 _GetSys ... 211
17.3.9 _Go0 ... 217
17.3.10 _GoAbs ... 218
17.3.11 _GoRel ... 219
17.3.12 _InputCall ... 220
17.3.13 _Memorize ... 221
17.3.14 _MoviLink ... 222
17.3.15 _MovCommDef ... 227
17.3.16 _MovCommOn ... 229
17.3.17 _Nop ... 229
17.3.18 _SBusCommDef ... 230
17.3.19 _SBusCommOn ... 234
17.3.20 _SBusCommState ... 234
17.3.21 _SetInterrupt ... 235
17.3.22 _SetSys ... 236
17.3.23 _SetTask ... 238
17.3.24 _SetTask2 ... 239
17.3.25 _SetVarInterrupt ... 240
17.3.26 _SystemCall ... 241
17.3.27 _TouchProbe ... 242
17.3.28 _Wait ... 243
17.3.29 _WaitInput ... 243
17.3.30 _WaitSystem ... 244
17.3.31 _WdOff ... 244
17.3.32 _WdOn ... 245
18 Compiler – Beispiele ... 246
18.1 Setzen von Bits und Ausgangsklemmen ... 246
18.2 Löschen von Bits und Ausgangsklemmen ... 247

18.3 Abfrage von Bits und Eingangsklemmen ... 248
18.3.1 Testen von Einzelbits ... 248
18.3.2 Testen mehrerer Bits ... 248
18.4 Flankenabfrage ... 249
18.4.1 Beispiel 1 ... 249
18.4.2 Beispiel 2 ... 251
18.5 Betrag einer Zahl ... 252
18.6 MoviLink-Befehl ... 253
18.6.1 Lesen eines geräteeigenen Parameters ... 253
18.6.2 Beschreiben einer Variablen über SBus ... 254
18.6.3 Lesen eines Parameters über SBus ... 255
18.7 SCOM-Kommunikation ... 256
18.7.1 Empfänger ... 256
18.7.2 Sender ... 257
18.8 Touch-Probe Interrupt-Verarbeitung ... 258
18.9 Zustandsautomat, Feldbussteuerung mit Notbetrieb ... 261
18.9.1 Mode 0 ... 265
18.9.2 Mode 1 ... 265
18.9.3 Mode 2 ... 265
18.9.4 Mode 3 ... 265
18.10 Compiler Programmgerüst ... 266
19 Compiler – Fehlermeldungen ... 275
20 Assembler – Einführung ... 276
20.1 Einstellen der Anwenderverfahreinheiten ... 276
20.1.1 Wegfaktoren ZÄHLER /NENNER ... 276
20.1.2 EINHEIT ... 278
20.2 Erste Schritte ... 279
20.2.1 IPOSplus® Assembler starten ... 279
20.2.2 Erstellen eines neuen Programms ... 280
20.2.3 Compilieren und Starten des Programms ... 281
21 Assembler – Editor ... 282
21.1 Beispiel ... 283
21.2 Programme erstellen ... 283
21.2.1 Befehlszeilen einfügen ... 283
21.3 Kompilieren und Download ... 284
21.4 Programme Starten / Stoppen ... 285
21.4.1 Variablenfenster ... 285
21.5 Vergleich Datei mit Gerät ... 285
21.6 Debugger ... 286
21.6.1 Ausführen bis Cursor ... 286
21.6.2 Einzelschritt ... 286
21.7 Programm aus Umrichter laden ... 287
21.8 Symbolübersicht ... 287

22 Assembler – Programmierung ... 288
22.1 Grundlagen ... 288
22.1.1 Programmkopf ... 288
22.1.2 Task 1 / Task 2 / Task 3 ... 288
22.1.3 Kommentare ... 288
22.1.4 Programmverzweigungen ... 288
22.1.5 Unterprogrammtechnik ... 288
22.1.6 Programmschleifen ... 289
22.1.7 Positionierbefehle ... 289
22.1.8 Binäre / analoge Ein-/Ausgänge ... 289
22.1.9 Zugriff auf Systemgrößen / Parameter ... 289
22.1.10 Variablen ... 290
22.1.11 Programmzeile ... 290
22.2 Binäre Ein-/Ausgänge ... 291
22.2.1 Binäre Eingänge ... 291
22.2.2 Binäre Ausgänge ... 293
22.3 Analoge Ein-/Ausgänge ... 296
22.3.1 Lesen analoger Ein-/Ausgänge ... 296
22.3.2 Schreiben analoger Ausgänge ... 296
23 Assembler – Befehle ... 297
23.1 Allgemeine Hinweise ... 297
23.2 Befehlsübersicht ... 297
23.2.1 Arithmetische Befehle ... 297
23.2.2 Bit-Befehle ... 298
23.2.3 Kommunikationsbefehle ... 298
23.2.4 Positionierbefehle ... 299
23.2.5 Programmbefehle ... 299
23.2.6 Setzbefehle ... 300
23.2.7 Spezielle Gerätebefehle ... 300
23.2.8 Vergleichsbefehle ... 301
23.3 Arithmetische Befehle ... 302
23.3.1 Grundrechenarten ADD / SUB / MUL / DIV ... 302
23.3.2 Arithmetische Hilfsoperationen NOT / MOD ... 303
23.3.3 Logische Verknüpfungen AND / OR / XOR ... 304
23.3.4 SHIFT-Befehle SHL / SHR / ASHR ... 305
23.4 Bitbefehle ... 307
23.4.1 Bitbefehle BSET / BCLR / BMOV / BMOVN ... 307
23.5 Kommunikationsbefehle ... 309
23.5.1 MOVLNK ... 309
23.5.2 MOVCOM ... 314
23.5.3 MOVON ... 315
23.5.4 SCOM ... 316
23.5.5 SCOMON ... 321
23.5.6 SCOMST ... 322

23.6 Positionierbefehle ... 323
23.6.1 Referenzfahrt GO0 ... 323
23.6.2 Absolutpositionierung GOA / Relativpositionierung GOR ... 325
23.7 Programmbefehle ... 329
23.7.1 Programmende END ... 329
23.7.2 Unterprogrammaufruf CALL ... 329
23.7.3 Sprungbefehle JMP ... 330
23.7.4 Schleifenbefehle LOOP ... 332
23.7.5 Keine Operation NOP / Kommentar REM / Rücksprung RET / TASK / TASK2 / Warten WAIT ... 333
23.8 Setzbefehle ... 336
23.8.1 Variablen kopieren COPY ... 336
23.8.2 Systemgrößen lesen GETSYS ... 336
23.8.3 Setzbefehle Variable SET / Fehlerreaktion SETFR / Indirekte Adressierung SETI / Interrupt SETINT / Systemgrößen SETSYS. 339
23.8.4 SETSYS ... 344
23.8.5 VARINT ... 347
23.9 Spezielle Gerätebefehle ... 349
23.9.1 ASTOP / MEM / TOUCHP / WDOFF / WDON ... 349
23.10 Vergleichsbefehle ... 355
23.10.1 Vergleichsoperationen CPEQ / CPGE / CPGT / CPLE / CPLT / CPNE ... 355
23.10.2 Logische Operationen ANDL / ORL / NOTL ... 358
24 Assembler – Beispiele ... 360
24.1 Beispielprogramm "Blinklicht" ... 360
24.1.1 Beispiel "Steuerung" ... 360
24.1.2 Beispiel "Positionierung" ... 361
24.2 Beispielprogramm "Hubwerk" ... 362
24.2.1 Eigenschaften ... 362
24.2.2 Einstellungen ... 362
24.2.3 Schematischer Aufbau ... 363
24.2.4 Beschaltung der Klemmen ... 364
24.2.5 Einstellung der für das Beispiel relevanten Parameter ... 365
24.2.6 Berechnung der IPOSplus®-Parameter ... 365
24.2.7 Eingangsklemmen ... 366
24.2.8 Ausgangsklemmen ... 366
24.2.9 Programmquellcode (mit Kommentaren) ... 367
24.3 Beispielprogramm "Tippbetrieb" ... 368
24.3.1 Eigenschaften ... 368
24.3.2 Einstellungen ... 368
24.3.3 Eingangsklemmen ... 369
24.3.4 Ausgangsklemmen ... 369
24.3.5 Programmquellcode (mit Kommentaren) ... 370

24.4 Beispielprogramm "Tabellenpositionierung" 372
24.4.1 Eigenschaften 372
24.4.2 Einstellungen 373
24.4.3 Eingangsklemmen 374
24.4.4 Ausgangsklemmen 374
24.4.5 Programmquellcode (mit Kommentaren)..... 375
Stichwortverzeichnis 378

1 Allgemeine Hinweise

1.1 Aufbau der Sicherheitshinweise

Die Sicherheitshinweise dieser Betriebsanleitung sind folgendermaßen aufgebaut:

Piktogramm	SIGNALWORT!
	Art der Gefahr und ihre Quelle. Mögliche Folge(n) der Missachtung. • Maßnahme(n) zur Abwendung der Gefahr.

Piktogramm	Signalwort	Bedeutung	Folgen bei Missachtung
Beispiel: Allgemeine Gefahr Spezifische Gefahr, z. B. Stromschlag	**GEFAHR!**	Unmittelbar drohende Gefahr	Tod oder schwere Körperverletzungen
	WARNUNG!	Mögliche, gefährliche Situation	Tod oder schwere Körperverletzungen
	VORSICHT!	Mögliche, gefährliche Situation	Leichte Körperverletzungen
	VORSICHT!	Mögliche Sachschäden	Beschädigung des Antriebssystems oder seiner Umgebung
	HINWEIS	Nützlicher Hinweis oder Tipp. Erleichtert die Handhabung des Antriebssystems.	

1.2 Mängelhaftung

Die Einhaltung des vorliegenden Handbuchs und die Betriebsanleitungen der Geräte ist die Voraussetzung für einen störungsfreien Betrieb und die Erfüllung eventueller Mängelhaftungsansprüche. Lesen Sie deshalb zuerst das Handbuch und die Betriebsanleitungen der Geräte, bevor Sie mit der Software und den Geräten arbeiten!

Stellen Sie sicher, dass die Betriebsanleitungen den Anlagen- und Betriebsverantwortlichen, sowie Personen, die unter eigener Verantwortung an den Geräten arbeiten, in einem leserlichen Zustand zugänglich gemacht werden.

1.3 Haftungsausschluss

Die Beachtung des vorliegenden Handbuchs und der Betriebsanleitungen der Geräte ist Grundvoraussetzung für einen sicheren Betrieb und für die Erreichung der angegebenen Produkteigenschaften und Leistungsmerkmale. Für Personen-, Sach- oder Vermögensschäden, die wegen Nichtbeachtung der Betriebsanleitungen entstehen, übernimmt SEW-EURODRIVE keine Haftung. Die Sachmängelhaftung ist in solchen Fällen ausgeschlossen.

1.4 Urheberrechtsvermerk

© 2009 – SEW-EURODRIVE. Alle Rechte vorbehalten.

Jegliche – auch auszugsweise – Vervielfältigung, Bearbeitung, Verbreitung und sonstige Verwertung verboten.

2 Sicherheitshinweise

Die folgenden grundsätzlichen Sicherheitshinweise dienen dazu, Personen- und Sachschäden zu vermeiden. Der Betreiber muss sicherstellen, dass die grundsätzlichen Sicherheitshinweise beachtet und eingehalten werden. Vergewissern Sie sich, dass Anlagen- und Betriebsverantwortliche, sowie Personen, die unter eigener Verantwortung an den Geräten arbeiten, das Handbuch vollständig gelesen und verstanden haben. Bei Unklarheiten oder weiterem Informationsbedarf wenden Sie sich bitte an SEW-EURODRIVE.

Die folgenden Sicherheitshinweise beziehen sich auf den Einsatz der IPOS$^{plus®}$ Positionierung und Ablaufsteuerung. Berücksichtigen Sie auch die ergänzenden Sicherheitshinweise in den einzelnen Kapiteln dieser Dokumentation und in den Dokumentationen der Geräte.

2.1 Allgemein

Lesen Sie dieses Handbuch sorgfältig durch, bevor Sie mit IPOS$^{plus®}$ arbeiten.

Diese Druckschrift ersetzt nicht die ausführliche Betriebsanleitung zu den Geräten! Das vorliegende Handbuch setzt das Vorhandensein und die Kenntnis der Dokumentationen zu den Geräten voraus.

Installieren Sie niemals beschädigte Produkte und nehmen Sie diese nicht in Betrieb. Reklamieren Sie Beschädigungen bitte umgehend beim Transportunternehmen. Nur qualifiziertes Elektrofachpersonal darf unter Beachtung der gültigen Unfallverhütungsvorschriften und der jeweiligen Betriebsanleitung Geräte installieren und in Betrieb nehmen.

Während des Betriebs können die Geräte ihrer Schutzart entsprechend spannungsführende, blanke gegebenenfalls auch bewegliche oder rotierende Teile sowie heiße Oberflächen besitzen.

Bei unzulässigem Entfernen von erforderlichen Abdeckungen, unsachgemäßem Einsatz, falscher Installation oder Bedienung besteht die Gefahr von schweren Personen- oder Sachschäden.

Weitere Informationen entnehmen Sie der Dokumentation.

2.2 Bestimmungsgemäße Verwendung

Verwenden Sie die Positionierung und Ablaufsteuerung mit IPOS$^{plus®}$ für die folgenden Geräte der SEW-EURODRIVE GmbH & Co KG:

- MOVIDRIVE® B
- MOVITRAC® B
- MQx

Zusätzlich wird IPOS$^{plus®}$ von den folgenden, bereits abgekündigten Geräten unterstützt:

- MOVIDRIVE® A
- MOVIDRIVE® *compact*
- MOVITRAC® 07

2.3 Zielgruppe

Der Anwender von IPOS$^{plus®}$ ist eine qualifizierte Fachkraft, die in geeigneter Weise unterwiesen wurde.

SEW-EURODRIVE empfiehlt dem Anwender zusätzlich Produktschulungen zu den Geräten und Motoren, die mit IPOS$^{plus®}$ programmiert werden.

Alle Arbeiten zur Installation, Inbetriebnahme und Instandhaltung der Geräte sowie Störungsbehebung an den Geräten sind von einer Elektrofachkraft auszuführen. Dabei sind die IEC 60364 bzw. CENELEC HD 384 oder DIN VDE 0100 und IEC 60664 oder DIN VDE 0110 und nationale Unfallverhütungsvorschriften zu beachten.

Elektrofachkräfte im Sinne dieser grundsätzlichen Sicherheitshinweise sind Personen, die mit Aufstellung, Montage, Inbetriebsetzung und Betrieb des Produktes vertraut sind und die über ihre Tätigkeit entsprechende Qualifikation verfügen.

Alle Arbeiten in den übrigen Bereichen Transport, Lagerung, Betrieb und Entsorgung müssen von qualifiziertem Fachpersonal durchgeführt werden, das in geeigneter Weise unterwiesen wurden.

2.4 Fehlprogrammierung

Sie verfügen mit IPOS$^{plus®}$ über ein Positionier- und Ablaufsteuerung, die es ermöglicht, in weiten Grenzen die IPOS$^{plus®}$-Geräte an Anlagengegebenheiten anzupassen. Wie bei allen programmierbaren Systemen besteht aber die Gefahr einer Fehlprogrammierung, die zu unerwartetem (nicht unkontrolliertem) Systemverhalten führt.

3 Systembeschreibung

3.1 Einführung

IPOS$^{plus®}$-Geräte stellen mit ihren Grundfunktionen und Optionen nicht mehr nur einen Drehzahlsteller dar.

Vielmehr kann mit der im MOVIDRIVE® B integrierten Positionierung und Ablaufsteuerung die übergeordnete SPS in vielen Fällen erheblich entlastet, wenn nicht sogar ersetzt werden.

Durch die Reduzierung der zentralen Steuerung bietet sich für die Kunden der SEW ein erhebliches Sparpotenzial an Hardware und an Aufwand für die elektrische Installation.

Der Programmieraufwand wird auf SPS und Umrichtersteuerung aufgeteilt, allerdings müssen sich die Nutzer in das System einarbeiten. Hierzu gehört die Einarbeitung in IPOS$^{plus®}$, wenn man die Vorteile effektiv nutzen möchte.

3.1.1 Umfang dieser Dokumentation

Die vorliegende Dokumentation behandelt die Positionierung und Ablaufsteuerung mit IPOS$^{plus®}$ für MOVIDRIVE® B.

IPOS$^{plus®}$ kann mit reduziertem Befehlssatz auch in Verbindung mit dem Schaltschrankumrichter MOVITRAC® B und den MQx-Modulen der dezentralen Technik eingesetzt werden. Auf Abweichungen im Funktionsumfang im Vergleich zu MOVIDRIVE® B wird in den betreffenden Technischen Daten und in den folgenden Kapiteln hingewiesen:

- IPOS$^{plus®}$ für MOVITRAC® B, siehe (Seite 118)
- IPOS$^{plus®}$ für MQx, siehe (Seite 120)

Das Handbuch beschreibt zunächst die sprachunabhängigen Funktionen von IPOS$^{plus®}$. Dies sind die dem "Drehzahlsteller" überlagerten Eigenschaften:

- Positionsregelung
- Positionsverarbeitung
- Task-Verwaltung
- Interrupt-Verwaltung
- IPOS-Parameter
- IPOS-Variablen.

Danach wird die Programmierung in Compiler-Sprache beschrieben. SEW-EURODRIVE empfiehlt, neue Programme in der Compiler-Sprache zu erstellen. Alle MOVIDRIVE® B lassen sich in dieser Sprache programmieren.

Nach der Beschreibung der Programmierung in Compiler-Sprache folgt die Beschreibung der Programmierung in Assembler-Sprache.

Im letzten Abschnitt werden Programmbeispiele beschrieben. Dort finden Sie auch ein Einsteigerbeispiel mit dem Grundgerüst der Zustandsmaschine eines Ablaufprogramms. Es wird empfohlen, zunächst mit diesem Grundgerüst zu beginnen und das Anwenderprogramm daraus weiterzuentwickeln.

3.1.2 Programmerstellung

Sie können IPOS$^{plus®}$-Programme mit dem Assembler oder mit dem Compiler erstellen. Beide Programmierwerkzeuge sind im Software-Paket MOVITOOLS® MotionStudio enthalten.

	HINWEIS Applikationsmodule lösen eine typische Antriebsaufgabe, ohne dass Sie als Anwender ein Programm erstellen müssen. Anstelle zu programmieren, parametrieren Sie ein von SEW erstelltes und bewährtes Programm (Applikationsmodul). Sie sparen dadurch Zeit und benötigen nicht die in diesem Handbuch beschriebenen Programmierkenntnisse.

3.2 Eigenschaften von IPOS$^{plus®}$

- Mit Geberrückführung bietet die IPOS$^{plus®}$-**Positionierung** eine leistungsfähige Punkt-zu-Punkt-Positionierung.
- Ausführung des Programms unabhängig von Geberrückführung und Betriebsart.
- Das Gerät führt das Anwenderprogramm auch bei einer Störung weiter aus (Fehlerbehandlung im Anwenderprogramm möglich).
- IPOS$^{plus®}$ kann mehrere Anwenderprogramme / Tasks parallel und unabhängig voneinander ausführen. Die `Tasks` können durch Interrupts unterbrochen werden.
- Die Anwenderprogramme können mehrere 100 Programmzeilen haben (siehe Technische Daten (Seite 25)).
- Komfortable und umfassende Steuermöglichkeiten für IPOS$^{plus®}$-Geräte.
- Zugriffsmöglichkeit auf vorhandene Optionen:
  - Ein-/Ausgabekarte
  - Feldbus-Schnittstellen
  - Synchronlaufkarte
- Umfangreiche Möglichkeiten zur Kommunikation:
  - Systembus (S-Bus)
  - RS-485 (RS-232 mit Schnittstellenumsetzer USS21A, UWS11A, UWS21A)
  - Feldbus-Schnittstellen
- Verarbeiten von binären und analogen Ein-/Ausgangssignalen.
- Positionierung mit wählbarer Verfahrdrehzahl und Positionierrampe.
- Vorsteuerung für Lage-, Drehzahl- und Drehmoment-Regelkreise für minimierten Schleppfehler.

- Absolutgeberverarbeitung.
- Im IPOS$^{plus®}$-Programm stehen 1024 32-Bit-Variablen zur Verfügung (siehe Technische Daten (Seite 25)).
- Mit IPOS$^{plus®}$ hat man die Möglichkeit über Kommunikationsbefehle alle Umrichterparameter zu lesen und zu schreiben.
- 2 Touch-Probe-Eingänge.
- Rampenformen:
  - Linear
  - Sinus
  - Quadratisch
  - Busrampe
  - Ruckbegrenzt
  - Kurvenscheibe
  - I-Synchronlauf
- Status- und Überwachungsfunktionen:
  - Schleppfehlerüberwachung
  - Positionsmeldung
  - Software- und Hardware-Endschalter
  - Geberfunktion
- 9 Referenzfahrttypen.
- Veränderung folgender Funktionen während des Verfahrens möglich:
  - Zielposition
  - Verfahrdrehzahl
  - Positionierrampe
  - Drehmoment
- "Endlos-Positionieren" möglich.
- Override-Funktion.
- Ansteuerung der folgenden Technologiefunktionen mit virtuellem Geber:
  - Kurvenscheibe
  - interner Synchronlauf

- Die Programmierung im Compiler bietet zusätzlich:
  - Programmerstellung in einer Hochsprache
  - Symbolische Variablennamen
  - Möglichkeit zur Erstellung von Programmmodulen, die Sie in anderen Projekten erneut verwenden können
  - Übersichtliche modulare und strukturierte Programmierung ist möglich
  - Verschiedene Programmiertechniken von Schleifen stehen zur Verfügung
  - Compilersteuerung durch Präprozessorbefehle
  - Standardstrukturen
  - Anwenderdefinierte Strukturen
  - Standardfunktionen stehen zur Verfügung
  - Debugger zur Fehlersuche
  - Umfangreiche Möglichkeiten zum Kommentieren
- Die Programmierung im Assembler bietet:
  - Kommentarzeilen
  - Programmierung in Anwenderverfahreinheiten (Eingabe der Einheiten im Programmkopf)
- Sollwertvorgabe. Abhängig von der vorhandenen Hardware und dem gewünschten Sollwert bestehen für die Vorgabe folgende Möglichkeiten:
  - Analog-Sollwerte
  - Festsollwerte
  - Festsollwerte + Analog-Sollwerte
  - Motorpotenziometer
  - Master-Slave-Betrieb SBus
  - Master-Slave-Betrieb RS-485
  - DRS-Sollwert (nur mit Option DRS11)
  - Feldbus-/Feldbusmonitor-Sollwert (nur mit Option Feldbus-Schnittstelle)
  - IPOS$^{plus®}$ Positions-Sollwert

  Ob Sie für die Sollwert-Verarbeitung eine Geberrückführung benötigen, hängt von der gewählten Betriebsart ab. Welcher Sollwert tatsächlich aktiv ist, ist abhängig von:
  - Betriebsart P700
  - Sollwertquelle P100
  - Parametrierung der Eingangsklemmen P600 ... P619
  - Feldbus PA-Datenbelegung/Monitorbetrieb
  - Anwahl von Handbetrieb

3.3 Steuerung von IPOSplus®-Geräten

IPOSplus®-Geräte sind folgendermaßen steuerbar:

- Steuerung über Geräte-Eingangsklemmen
- IPOSplus®-Steuerwort auf "Systemvariable" H484 CTRL. WORD
- RS-485-Schnittstelle
- Feldbus-Schnittstelle
- SBus (Systembus)

3.3.1 Aktive Steuerquelle

Die Steuerung über Eingangsklemmen und das IPOSplus®-Steuerwort H484 sind immer wirksam.

Sie können weitere Steuerquellen mit den folgenden Parametern festlegen:

- Sollwertquelle P100
- Steuerquelle P101
- Prozessdatenbeschreibung P870 ... P872

3.4 Technologie-Optionen / Applikationsmodule

3.4.1 Technologie-Optionen

MOVIDRIVE® mit Technologie-Option (Endung -0T in der Typenbezeichnung) bieten Zusatzfunktionen wie:

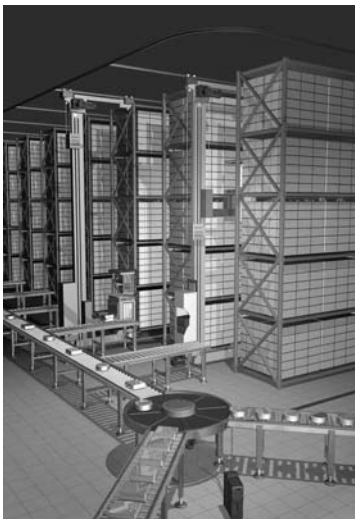
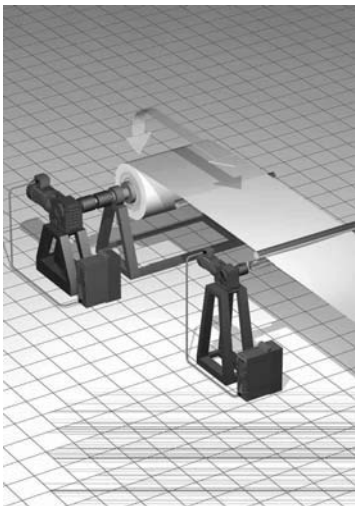
- Interner Synchronlauf (ISYNC)
- Kurvenscheibe
- Applikationsmodule
- Auto ASR (Anti-Schlupf-Regelung, z. Z. nur bei MOVIDRIVE® A)
- SBus TP (SBus Touch-Probe, nur bei MOVIDRIVE® A, bei MOVIDRIVE® B im Standardgerät lösbar mit Variablen-Interrupt)

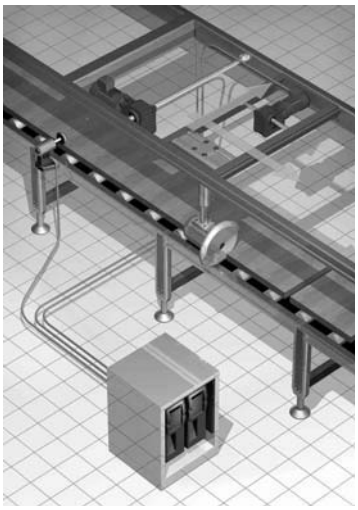
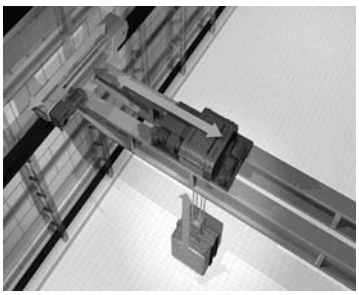
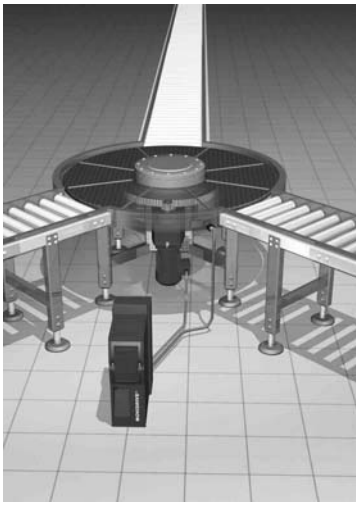
Die Funktionen "interner Synchronlauf (ISYNC)" und "Kurvenscheibe" werden in dem Kapitel "IPOSplus® und synchronisierte Bewegungen" erläutert und in separaten Handbüchern beschrieben. In diesen Fällen sind weitere IPOSplus®-Variablen mit Systemfunktionen belegt, die Sie im IPOSplus®-Anwenderprogramm ansprechen können.

3.4.2 Applikationsmodule

Bei einem Applikationsmodul handelt es sich um ein von SEW entwickeltes und geschütztes Anwendenderprogramm, welches in den Umrichter geladen wird.

Ein umfangreiches Paket aufeinander abgestimmter Funktionen, komfortablen Eingabemasken und eine einfache Benutzerführung erleichtern die Inbetriebnahme. Eine Anpassung des IPOS$^{plus®}$-Programms durch den Anwender ist nicht möglich.

Übersicht über verfügbare Applikationsmodule	
Positionieren	 473361163
Wickeln	 473365515

Übersicht über verfügbare Applikationsmodule	
Fliegende Säge	 473369867
Interner Synchronlauf ISYNC (nur MOVIDRIVE® B und MCH)	 473374219
Rotatorisch Positionieren	 473378571

Die intelligenten Applikationsmodule der Technologie-Option bieten Ihnen eine bisher nicht erreichte Funktionalität. Alle wichtigen Maschinendaten sind bestens zugänglich. Fehlerquellen sind praktisch ausgeschlossen, da nur die für die Anwendung erforderlichen Parameter eingegeben werden müssen. Mit Hilfe eines Diagnose-Tools können alle relevanten Daten, wie zum Beispiel Klemmenzustände oder Positionswerte während des laufenden Arbeitsprozesses beobachtet werden.

Die Funktionsweise wird in separaten Handbüchern beschrieben.

3.5 Technische Daten

3.5.1 MOVIDRIVE® B

Geberauflösung: X15, Motorgeber X14, externer Geber X62, Absolutwert-Geber (auch Absolutwert-Geber aus HIPERFACE®)	IPOS$^{plus®}$ arbeitet immer mit 4096 Inkrementen / Motorumdrehung (Voraussetzung: Geberauflösung von 128, 512, 1024 oder 2048 Impulse / Motorumdrehung (andere Geberauflösung ist nicht zulässig) oder Resolver)
maximale Programmlänge / Programmspeicher:	16 kByte entspricht ca. 200 ... 250 Assembler-Befehlen
Befehlsabarbeitungszeit:	Die Summe in Task 1 und Task 2 ≤ 12 Assembler-Befehle / ms): Task 1: 1 ... 10 Assembler-Befehle / ms Task 2: 2 ... 11 Assembler-Befehle / ms Task 3: freie Rechenzeit
Interrupts:	1 Interrupt ausgelöst durch Timer, Fehler oder Touch-Probe unterbricht Task 1 4 Variablen-Interrupts, die Task 2 und Task 3 unterbrechen.
Variablen:	1024, davon 128 (0 ... 127) netzausfallsicher speicherbar. Wertebereich: - 2^{31} ... +(2^{31} - 1)
Bereich der Systemvariablen	Die IPOS-Variablen H453 bis H560
Touch-Probe-Eingänge:	2 Eingänge, Verarbeitungszeit 200 µs
Abtastzeit analoger Eingänge:	1 ms
Abtastzeit binärer Eingänge:	1 ms
Binäre Ein-/Ausgänge:	MOVIDRIVE® B: 8 Eingänge / 6 Ausgänge Option DIO: 8 Eingänge / 8 Ausgänge Option DIP: 8 Eingänge / 8 Ausgänge
Analoge Ein-/Ausgänge:	MOVIDRIVE® B: 1 Eingang (0 ... 10 V, ± 10 V, 0 ... 20 mA, 4 ... 20 mA) Option DIO: 1 Eingang (0 ... 10 V, ± 10 V, 0 ... 20 mA) 2 Ausgänge (± 10 V, 0 ... 20 mA, 4 ... 20 mA)

3.5.2 MOVITRAC® B

Geberauflösung:	MOVITRAC® B hat keine Gebereingänge, unterstützt aber die Wegerfassung über binäre Eingänge (Zählereingang). Die Technischen Daten der binären Eingänge entnehmen Sie dem Kapitel "Wegerfassung über binäre Eingänge" (Seite 96)	
maximale Programmlänge / Programmspeicher:	8 kByte	
Befehlsabarbeitungszeit:	Task 1: 1 Assembler-Befehl / ms Task 2: 2 Assembler-Befehle / ms	
Interrupts:	-	
Variablen:	1024, davon 128 (0 ... 127) netzausfallsicher speicherbar.	
Bereich der Systemvariablen	IPOS-Variablen H453 bis H560	
Touch-Probe-Eingänge:	-	
Abtastzeit analoger Eingänge:	1 ms	
Abtastzeit binärer Eingänge:	5 ms	
Binäre Ein-/Ausgänge:	MOVITRAC® B Option FIO21B	6 Eingänge / 3 Ausgänge 7 Eingänge
Analoge Ein-/Ausgänge:	MOVITRAC® B Option FIO11B	1 Eingang (0 ... 10 V) 1 Eingang (± 10 V) 1 Ausgang (0 ... 20 mA, 4 ... 20 mA)

3.5.3 MQx

Geberauflösung:	Das MQx-Modul hat keine Gebereingänge, unterstützt aber die Wegerfassung über binäre Eingänge (Zählereingang). Die Technischen Daten der binären Eingänge entnehmen Sie dem Kapitel "Wegerfassung über binäre Eingänge" (Seite 96)
maximale Programmlänge / Programmspeicher:	4 kByte
Befehlsabarbeitungszeit:	Task 1: 1 Assembler-Befehl / ms Task 2: 2 Assembler-Befehle / ms
Interrupts:	-
Variablen:	512, davon 128 (0 ... 127) netzausfallsicher speicherbar.
Bereich der Systemvariablen	IPOS-Variablen H453 bis H511
Touch-Probe-Eingänge:	-
Abtastzeit analoger Eingänge:	-
Abtastzeit binärer Eingänge:	4 ms / Eingangsfrequenz bei den Zählereingängen: max. 4 KHz
Binäre Ein-/Ausgänge:	MQ.21./MQ.22.: 4 Eingänge und 2 Ausgänge MQ.32.: 6 Eingänge
Analoge Ein-/Ausgänge:	-

3.6 Referenzdokumente

In diesem Dokument wird die im MOVIDRIVE® integrierte Positionierung und Ablaufsteuerung IPOS$^{plus®}$ beschrieben.

Die nachfolgende Referenzliste gibt eine Übersicht über die Dokumente, auf die an den einzelnen Stellen verwiesen wird. Diese Dokumente sind nicht notwendig, um mit IPOS$^{plus®}$ zu programmieren, bieten aber zusätzliche Informationen.

Sie finden alle Dokumente unter http://www.sew-eurodrive.de

3.6.1 Allgemeine Handbücher

- MOVIDRIVE® MDX60B/61B Systemhandbuch
- MOVITRAC® B Systemhandbuch
- Handbücher zu den Feldverteilern MQx

3.6.2 Handbücher Serielle Schnittstellen / Feldbusse

- MOVIDRIVE® MDX60B/61B Kommunikation und Feldbus-Geräteprofil
- MOVIDRIVE® Feldbus-Schnittstelle DFx

3.6.3 Handbücher für synchronisierte Achsbewegungen

- MOVIDRIVE® Kurvenscheibe, Zusatz zum Systemhandbuch
- MOVIDRIVE® Synchronlaufkarte Typ DRS11
- MOVIDRIVE® Interner Synchronlauf

3.6.4 Handbücher zu den Applikationsmodulen

- MOVIDRIVE® Positionieren mit Absolutgeberkarte DIP11
- MOVIDRIVE® Erweiterte Buspositionierung
- MOVIDRIVE® Buspositionierung
- MOVIDRIVE® Tabellenpositionierung mit Bussteuerung
- MOVIDRIVE® Modulo-Positionierung

3.6.5 Handbücher zu Feldbus-Schnittstellen MQx

- Antriebssystem für Dezentrale Installation: PROFIBUS-Schnittstellen, -Feldverteiler
- Antriebssystem für Dezentrale Installation: INTERBUS-Schnittstellen, -Feldverteiler
- Antriebssystem für Dezentrale Installation: DeviceNet/CANopen-Schnittstellen, -Feldverteiler

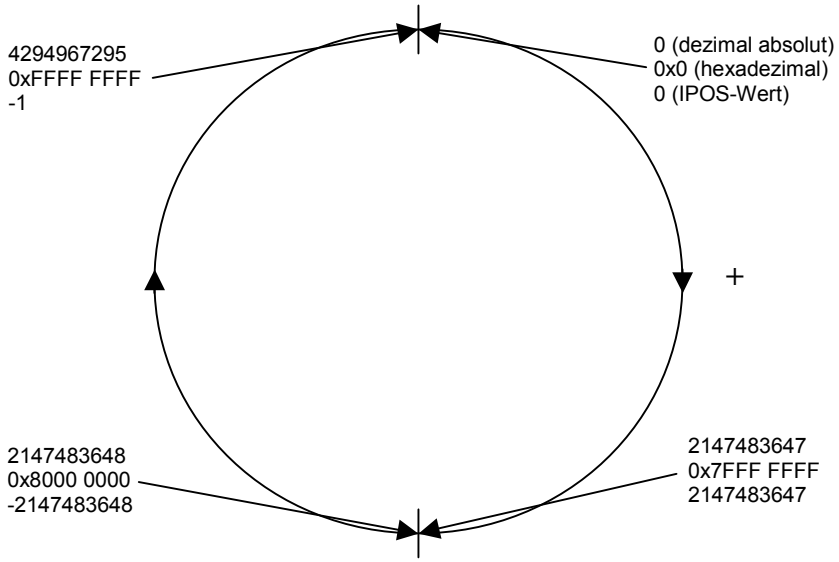
4 IPOS-Variablen

4.1 Einführung

Die integrierte Positionierung und Ablaufsteuerung benutzt globale Variablen, die von allen Tasks und den Interrupts gemeinsam genutzt werden. Es gibt **keine** lokalen Variablen, die nur in einer Task oder in einer Funktion deklariert sind.

Alle Variablen (Seite 25) sind 32-Bit-Variablen, die bei Rechenoperationen und Vergleichen als Ganzzahl mit Vorzeichen (Signed Integer) behandelt werden. Es ist im Anwenderprogramm darauf zu achten, dass das Endergebnis einer Rechenoperation im Zahlenbereich liegt.

Der Zahlenbereich kann wie folgt in einem Zahlenkreis dargestellt werden:



473666955

Beispiel:

$$H0 = 4, \quad H1 = 7, \quad H3 = \frac{H0}{H1} = 0$$

$$H0 = 2147483647, \quad H1 = 1, \quad H3 = H0 + H1 = -2147483648$$

Jede Variable hat einen Index über den z. B. mit dem Movilink-Befehl (_MoviLink/ `MOVLNK`) die Variable gelesen und geschrieben werden kann. Der Index berechnet sich aus:

Index = VarNr. + 11000

Beispiel: H371 hat den Index 11371.

4.2 Übersicht über die Systemvariablen

Bestimmte IPOS-Variablen sind fest mit bestimmten Funktionen vorbelegt und werden als Systemvariablen (Seite 25) bezeichnet.

Die symbolischen Namen sind im Compiler verfügbar, wenn am Programmanfang eine der folgenden Zeilen eingefügt ist:

```
#include <constb.h>   //symb. Namen Systemvariablen MOVIDRIVE B
```

Die nachfolgende Tabelle beschreibt die Funktion der Systemvariablen und deren Name im Compiler und im Assembler.

Variablen im angegebenen Bereich, die nicht belegt sind, sind für interne Funktionen reserviert und dürfen nicht für Anwendervariablen benutzt werden.

Nr.	Name Compiler / Assembler	Beschreibung
128		Diese Variable ist frei verfügbar in einem anwenderspezifischen IPOS-Programm. Die Variable wird bei den Applikationsmodulen benutzt, um die Programm-Identifikation abzulegen.
360 ... 452	Variablenbereich für internen Synchronlauf oder Kurvenscheibe	Dieser Variablenbereich ist durch weitere Systemvariablen belegt, wenn die Technologie-Optionen interner Synchronlauf oder Kurvenscheibe verwendet werden. In allen anderen Fällen können diese vom Anwender beliebig benutzt werden.
453	ModuloCtrl / MODULOCTRL	Steuerwort für die Modulo-Funktion (siehe auch Modulo-Funktion und IPOS-Parameter). **Bit 0 TargetReset_Off** Bit 0 = 0: Der aktuelle Positionierauftrag wird gelöscht (ModTagPos wird = ModActPos gesetzt), wenn der Positionierbetrieb (z.B: durch Wegnahme der Freigabe oder Setzen der Reglersperre oder des Halt-Bits) unterbrochen wird. Bit 0 = 1: Die Zielposition bleibt auch erhalten, wenn die Freigabe weggenommen, die Reglersperre oder das Halt-Bit gesetzt wurde. Wird der Antrieb wieder freigegeben, setzt der Antrieb die Positionierung fort. **Bit 1 TargetGAZ_Select** Bit 1 = 0: Standardeinstellung, 360° am Abtrieb entsprechen 2^{16} Inkr. Bit 1 = 1: Einstellung um Auflösung zu erhöhen: 360° entsprechen dem Produkt aus Modulo-Zähler P961 x Modulo-Geberauflösung P963. Es kann nicht über mehrere Umdrehungen positioniert werden.
454	ModTagPos / MOD.TAGPOS	Modulo Target Position Wird die Modulo Target Position bei freigegebenem Umrichter mit einem geänderten Wert beschrieben, beginnt die Positionierung in Abtriebseinheiten. Die Positionsvorgabe hat (bei H453.1 = 0) in der 16-Bit-Auflösung die Einheit H454 MODTAGPOS = k x 360° + 0 ... 360°= k x 2^{16} + 0 ... (2^{16} -1) zu erfolgen (k = Anzahl der ganzen Umdrehungen). Nachdem die Variable mit einem neuen Wert beschrieben wurde, ist in der Variable H454 nur die Zielposition innerhalb einer Umdrehung sichtbar. Es wird daher empfohlen, zur besseren Diagnose den neuen Wert zusätzlich auch auf eine andere Zwischenvariable zu schreiben. Nach Beschreiben der Position 454 berechnet die Firmware ein inkrementelles Ziel H492. Dadurch ist H473 Bit 19 "In Position" noch bis zu 1 ms lang gesetzt.
455	ModActPos / MOD.ACTPOS	Modulo Actual Position Die aktuelle Modulo-Istposition bewegt sich (in der 16-Bit-Auflösung bei H453.1 = 0) zwischen 0 und 2^{16} Inkremente (0° und 360°).
456	ModCount / MOD COUNT	Inkremente innerhalb einer Modulo-Umdrehung vor der Skalierung auf den Abtrieb. Anzeigewert des internen Zwischenergebnisses bei der Umrechnung vom inkrementellen Geberwert H509 / H510 / H511 (IPOS-Geber-Wert) auf die Modulo Actual Position H455. Mit H456 = (IPOS-Geber-Wert) MOD (P961 x P963) wird H455 = H456 / (P961 x P963) x 2^{16} (Voraussetzung: H453, Bit 1 = 0) Siehe Kapitel "Modulo-Positionierung". Wird H456 mit 0 beschrieben, wird automatisch H455 = 0 gesetzt.

Nr.	Name Compiler / Assembler	Beschreibung
473	StatusWord / STAT.WORD	**Mit dem Statuswort kann der Betriebszustand des Umrichters abgefragt werden.** **Bit** — **Funktion bei Pegel "1"** 0 Keine Funktion 1 /Störung 2 Betriebsbereit 3 Endstufe ein 4 Drehfeld ein 5 Bremse auf 6 Bremse zu 7 Motor-Stillstand (ab n < 20 min^{-1}) 8 Parametersatz 9 Drehzahl-Referenz (P400) 10 Drehzahl-Fenster (P410) 11 Soll-Ist-Vergleich (P420) 12 Stromreferenz (P430) **Bit** — **Funktion bei Pegel "1"** 13 Imax-Meldung (P442) 14 /Motorauslastung 1 15 /Motorauslastung 2 16 /DRS Vorwarnung 17 /DRS Schleppfehler 18 DRS Slave in Position 19 IPOS in Position (siehe auch H493) 20 IPOS referenziert 21 reserviert 22 /IPOS Störung 23 reserviert 24 Stromgrenze erreicht 25 LSM kommutiert 26 S-Verschliffprofil wird erzeugt 27 Umrichter ist im sicheren Halt 28..31 reserviert Befindet sich die Ist-Position des Antriebs innerhalb des Positionsfensters um die Zielposition, wird das Bit "IPOS in Position" auch bei Wegnahme der Freigabe oder Setzen der Reglersperre gesetzt.
474	Scope474 / SCOPE 474	Mit diesen beiden Variablen können mit der in MOVITOOLS® MotionStudio integrierten Oszilloskop-Funktion SCOPE Messwerte aufgezeichnet werden. Beispiel: Messung des Lage-Istwerts einer Modulo-Achse. Im IPOS-Programm wird der Befehl H474 = H455; zyklisch aufgerufen und im SCOPE Kanal 1 = IPOS-Variable H474 Low und Kanal 2 = IPOS-Variable H474 High eingestellt.
475	Scope475 / SCOPE 475	
476	DRS_Ctrl / DRS CTRL.	**Signalpegel der binären Ausgänge der Synchronlaufkarte DRS11 LESEN und SETZEN.** **Bit** — **Klemmenpegel** 0 X40.9 AUSG0 1 X40.10 AUSG1 2..14 reserviert 15 Hardware-Fehler DRS (Fehler 48) setzen 16..31 reserviert
477	DRS_Status / DRS STATUS	**Signalpegel der binären Eingänge und Statusmeldungen der Synchronlaufkarte DRS11 LESEN.** **Bit** — **Klemmenpegel / Statusmeldungen** 0 X40.5 EING4 Freier Eingang 1 1 X40.6 EING5 Freier Eingang 2 2 /DRS Vorwarnung 3 /DRS Schleppfehler 4 DRS Slave in Position 5 Master Stillstand 6..31 reserviert
478	AnaOutpIPOS2 / ANA.OUT IP2	**Analogausgänge der Ein-/Ausgabekarte DIO11 nur SETZEN.** Der Wert der Variable H478 wird auf einen Analogausgang ausgegeben, wenn die zugehörige Klemme auf "IPOS-AUSGABE 2" programmiert ist. Bei MOVIDRIVE® A und B wird dazu eine DIO11 benötigt, bei MCH und MCS / MCV / MCV 40A ist ein Ausgang als Binärausgang oder Analogausgang programmierbar. **Variablenwert** — **Physikalischer Ausgang** — **Ausgangsklemmenprogrammierung** - 10000..0..10000 — AOV1/AOC1/AO01 — P640 Analogausgang AO1 = IPOS-AUSGABE 2 - 10000..0..10000 — AOV2/AOC2/AO01 — P643 Analogausgang AO2 = IPOS-AUSGABE 2

Nr.	Name Compiler / Assembler	Beschreibung
479	AnaOutpIPOS / ANA.OUT IP	**Analogausgänge der Ein-/Ausgabekarte DIO11 nur SETZEN.** Der Wert der Variable H479 wird auf einen Analogausgang ausgegeben, wenn die zugehörige Klemme auf "IPOS-AUSGABE" programmiert ist. Bei MOVIDRIVE® A und B wird dazu eine DIO11 benötigt, bei MCH und MCS / MCV / MCV 40A ist ein Ausgang als Binärausgang oder Analogausgang programmierbar. **Variablenwert** / **Physikalischer Ausgang** / **Ausgangsklemmenprogrammierung** - 10000..0..10000 / AOV1/AOC1/AO01 / P640 Analogausgang AO1 = IPOS-AUSGABE - 10000..0..10000 / AOV2/AOC2/AO01 / P643 Analogausgang AO2 = IPOS-AUSGABE
480	OptOutpIPOS / OPT.OUT IP	**Binärausgänge der Ein-/Ausgabekarte DIO11 / DIP11 nur SETZEN.** LESEN ist bei MOVIDRIVE® A mit H482 und bei MOVIDRIVE® B mit H521 möglich. Ist weder eine DIO11 noch eine DIP11 gesteckt, so können damit auch die virtuellen Klemmen über Feldbus im Statuswort 2 gesetzt werden, wenn z. B. P873 = STATUSWORT 2. Die Bits der Variable H480 werden auf die Binärausgänge des Grundgeräts abgebildet, wenn die zugehörige Klemme auf IPOS-AUSGANG programmiert ist. Wird in IPOS ein Binärausgang getoggelt, so wird der physikalische Ausgang an der Klemme 1 ms später getoggelt, wenn er als IPOS-Ausgang parametriert ist.

Bit	IPOS-Name	DIO (+ DIO)	DIO + DIP	DIO + Feldbus	DIP	DIP + Feldbus	Feldbus
		P63x wirkt auf DIO			P63x wirkt auf DIP		P873 Statuswort 2
0	DO10	X23:1	X23:1	X23:1	X61:1	X61:1	Bit 8
1	DO11	X23:2	X23:2	X23:2	X61:2	X61:2	Bit 9
2	DO12	X23:3	X23:3	X23:3	X61:3	X61:3	Bit 10
3	DO13	X23:4	X23:4	X23:4	X61:4	X61:4	Bit 11
4	DO14	X23:5	X23:5	X23:5	X61:5	X61:5	Bit 12
5	DO15	X23:6	X23:6	X23:6	X61:6	X61:6	Bit 13
6	DO16	X23:7	X23:7	X23:7	X61:7	X61:7	Bit 14
7	DO17	X23:8	X23:8	X23:8	X61:8	X61:8	Bit 15
8			X61:1				
9		(X23:1)	X61:2				
10		...	X61:3				
11		(X23:8)	X61:4				
12			X61:5				
13			X61:6				
14			X61:7				
15			X61:8				

481	StdOutpIPOS / STD.OUT IP	**Binärausgänge des Grundgeräts nur SETZEN.** Wird in IPOS ein Binärausgang getoggelt, so wird der physikalische Ausgang an der Klemme 1 ms später getoggelt, wenn er als IPOS-Ausgang parametriert ist.

Bit	IPOS-Name	
0	DB00	nicht programmierbar, fest belegt mit "/Bremse"
(0)	DO00 MQx	falls P628 = IPOS-AUSGANG (nur MQx)
1	DO01	falls P620 = IPOS-AUSGANG
2	DO02	falls P621 = IPOS-AUSGANG
3	DO03	nur bei MOVIDRIVE® B, falls P622 = IPOS-AUSGANG
4	DO04	nur bei MOVIDRIVE® B, falls P623 = IPOS-AUSGANG
5	DO05	nur bei MOVIDRIVE® B, falls P624 = IPOS-AUSGANG

Nr.	Name Compiler / Assembler	Beschreibung							
482	OutputLevel / OUTPUT LVL MOVIDRIVE® A (MOVIDRIVE® B: H521)	**Signalpegel der binären Ausgänge nur LESEN.**							
		Bit	**IPOS-Name**	**DIO (+ DIO)**	**DIO + DIP**	**DIO + Feldbus**	**DIP**	**DIP + Feldbus**	**Feldbus**
				P63x wirkt auf DIO			P63x wirkt auf DIP		P873 Status-wort 2
		0	DB00	DB00	DB00	DB00	DB00	DB00	DB00
		1	DO01	DO01	DO01	DO01	DO01	DO01	DO01
		2	DO02	DO02	DO02	DO02	DO02	DO02	DO02
		3	DO10	X23:1	X23:1	X23:1	X61:1	X61:1	Bit 8
		4	DO11	X23:2	X23:2	X23:2	X61:2	X61:2	Bit 9
		5	DO12	X23:3	X23:3	X23:3	X61:3	X61:3	Bit 10
		6	DO13	X23:4	X23:4	X23:4	X61:4	X61:4	Bit 11
		7	DO14	X23:5	X23:5	X23:5	X61:5	X61:5	Bit 12
		8	DO15	X23:6	X23:6	X23:6	X61:6	X61:6	Bit 13
		9	DO16	X23:7	X23:7	X23:7	X61:7	X61:7	Bit 14
		10	DO17	X23:8	X23:8	X23:8	X61:8	X61:8	Bit 15
		11			X61:1				
		12		(X23:1)	X61:2				
		13		...	X61:3				
		14		(X23:8)	X61:4				
		15			X61:5				
		16			X61:6				
					X61:7				
					X61:8				
483	InputLevel / INPUT LVL MOVIDRIVE® A (MOVIDRIVE® B: H520)	**Signalpegel der binären Eingänge nur LESEN.**							
		Bit	**IPOS-Name**	**DIO (+ DIO)**	**DIO + DIP**	**DIO + Feldbus**	**DIP**	**DIP + Feldbus**	**Feldbus** P870 = Statuswort 2
		0	DI00	Je nach Grundgeräte, z. B. X13:1					
		1	DI01	X13:2					
		2	DI02	X13:3					
		3	DI03	X13:4					
		4	DI04	X13:5					
		5	DI05	X13:6					
		6	DI10	X22:1	X22:1	X22:1	X60:1	X60:1	Bit 8
		7	DI11	X22:2	X22:2	X22:2	X60:2	X60:2	Bit 9
		8	DI12	X22:3	X22:3	X22:3	X60:3	X60:3	Bit 10
		9	DI13	X22:4	X22:4	X22:4	X60:4	X60:4	Bit 11
		10	DI14	X22:5	X22:5	X22:5	X60:5	X60:5	Bit 12
		11	DI15	X22:6	X22:6	X22:6	X60:6	X60:6	Bit 13
		12	DI16	X22:7	X22:7	X22:7	X60:7	X60:7	Bit 14
		13	DI17	X22:8	X22:8	X22:8	X60:8	X60:8	Bit 15
		14			X60:1				
		15		(X22:1)	X60:2				
		16		...	X60:3				
		17		(X22:8)	X60:4				
		18			X60:5				
		19			X60:6				
		20			X60:7				
		21			X60:8				

Nr.	Name Compiler / Assembler	Beschreibung
484	ControlWord / CTRL.WORD	**IPOS$^{plus®}$-Steuerwort (Gerätefunktionen LESEN und SETZEN).** Das IPOS$^{plus®}$-Steuerwort kann unabhängig von der Betriebsart, Steuer- und Sollwertquelle immer verwendet werden. Das IPOS$^{plus®}$-Steuerwort wird im Gerät mit den Klemmenfunktionen, dem Feldbus-Steuerwort und dem Steuerwort über die RS-485/RS-232 und dem SBus logisch ODER verknüpft.

Bit	Funktion bei Pegel "1"	Bit	Funktion bei Pegel "1"
0	Keine Funktion	16	reserviert
1	Keine Freigabe	17	Referenznocken
2	Rechts	18	Referenzfahrt Start
3	Links	19	Slave Freilauf
4	n11/n21 (Festsollwert 1)	20	Sollwertsperre
5	n12/n22 (Festsollwert 2)	21	reserviert
6	Festsollwert Umschaltung	22	DRS-Nullpunkt setzen
7	Param.-Umschaltung (Param.-Satz 2)	23	DRS Slave Start
8	Rampen-Umschaltung (Rampen-Satz 2)	24	DRS Teach in
9	Motorpotenziometer auf	25	reserviert
10	Motorpotenziometer ab	26	reserviert
11	Externer Fehler	27	Condition Monitoring Schaltsignal: Antriebsschwingung Warnung
12	Fehler-Reset	28	Condition Monitoring Schaltsignal: Antriebsschwingung Fehler
13	Halteregelung	29	Condition Monitoring Schaltsignal: Bremsenverschleiß Fehler
14	Endschalter Rechts	30	Reglersperre
15	Endschalter Links	31	reserviert

485	T0_Reload / T0 RELOAD	Ladewert für Zykluszeit Anwender-Timer 0 LESEN und SETZEN.Wenn mit dem Befehl SET INTERRUPT (SETINT) ein Anwender-Timer (TIMER0 (H489)) zyklisch benutzt werden soll, so kann mit H485 die Zykluszeit festgelegt werden. Der in H485 eingetragene Zeitwert wird automatisch nach jedem Ablauf des Timers 0 (H489 = 0) wieder mit diesem Zeitwert neu geladen.Wertebereich: 0 ... 2^{31} -1 ms.
486	reserved	
487	Timer_2 / TIMER 2	**Zeit für Anwender-Timer 2 LESEN und SETZEN.** Der Anwender-Timer 2 zählt aufwärts. Wertebereich: 0 ... 2^{31} -1 ms.
488	Timer_1 / TIMER 1	**Zeit für Anwender-Timer 1 LESEN und SETZEN.** Der Anwender-Timer 1 zählt abwärts bis 0. Wertebereich: 0 ... 2^{31} -1 ms.
489	Timer_0 / TIMER 0	**Zeit für Anwender-Timer 0 LESEN und SETZEN.** Der Anwender-Timer 0 zählt abwärts bis 0. Bei Verwendung des Befehls SET INTERRUPT (SETINT) wird bei Erreichen des Timerwertes = 0 eine Interrupt-Verzweigung ausgeführt. Wenn mit dem Befehl SET INTERRUPT (SETINT) ein Anwender-Timer zyklisch benutzt werden soll, so kann mit der Variablen T0 RELOAD (H485) die Zykluszeit festgelegt werden. Siehe Kapitel „Task-Verwaltung und Interrupts". Wertebereich: 0 ... 2^{31} -1 ms.

Nr.	Name Compiler / Assembler	Beschreibung
490	WdogTimer / WD.TIMER	**Zeit für Anwender-Watchdog LESEN und SETZEN.** Der Watchdog-Timer zählt abwärts bis 0. Der Befehl WATCHDOG ON (WDON) aktiviert den Timer und legt die Zykluszeit fest. Wertebereich: 0 ... 2^{31} -1 ms.
491	SetpointPos / SETP.POS.	**Aktuelle Sollposition LESEN.** **ACHTUNG: Systemregelgröße! Wert darf nicht überschrieben werden!** Unabhängig davon, wie viel Impulse der Geber je Umdrehung liefert, hat die Sollposition immer die Einheit: 4096 Ink. / Motorumdrehung (Geberauflösung ≥ 512). Die aktuelle Sollposition repräsentiert die **momentan** für die Lageregelung gültige **absolute** Position des laufenden Verfahrauftrags. Der zeitliche Verlauf der Sollposition ergibt sich aus dem berechneten Verfahrprofil unter Berücksichtigung von Positionierrampe, Verfahrdrehzahl, Rampenform usw. Nach abgeschlossenem Verfahrauftrag und stillstehendem Antrieb ist der Wert von H491 gleich H492. Wertebereich: -2^{31} ... 0 ... 2^{31} -1 Ink.
492	TargetPos / TARGET POS	**Aktuelle Zielposition LESEN und SETZEN.** Unabhängig davon, wie viel Impulse der Geber je Umdrehung liefert, hat die Zielposition immer die Einheit: 4096 Ink. / Motorumdrehung (Geberauflösung ≥ 512). Diese Variable repräsentiert die aktuelle Zielposition des gerade anstehenden Verfahrauftrags. Die Position ist in H492 in absoluter Form dargestellt. Beispiel: 1. aktuelle Antriebsposition: 50000 Ink. 2. GOR NOWAIT #-8000 Ink. 3. aktuelle Zielposition: 42000 Ink. Wertebereich: -2^{31} ... 0 ... 2^{31} -1 Ink. Wird H492 direkt (nicht über einen GO-Befehl) beschrieben, so ist H473, Bit 19 "In Position" noch bis zu 1 ms lang gesetzt.
493	PosWindow / POS.WINDOW	**Positionsfenster LESEN und SETZEN.** H493 ist identisch mit P922. Das Positionsfenster definiert einen Entfernungsbereich um die Zielposition (H492) eines Verfahr- oder Stopp-Befehls (GOx oder ASTOP TARGET POSITION). Sobald der Antrieb das Positionsfenster erreicht, wird die Meldung “IPOS IN POSITION" generiert. Diese Meldung steht über einen binären Ausgang zur Verfügung, welcher auf die Funktion "IPOS IN POSITION" zu parametrieren ist und in der Systemvariablen H473, Bit 19. Die Meldung "IPOS IN POSITION" wird sofort beim Absetzen eines GO-Befehls zurückgesetzt. Die Überwachung des Positionsfensters findet immer statt, solange eine Betriebsart mit IPOS aktiv ist (P700). Die Größe des Positionierfensters hat keinen Einfluss auf die Positioniergenauigkeit. Einstellbereich: 0 ... **50** ... 2^{15} - 1 Inkremente
494	LagWindow / LAG WINDOW	**Schleppfehlerfenster LESEN und SETZEN.** H494 ist identisch mit P923. Das Schleppfehlerfenster definiert die maximal zulässige Differenz zwischen der augenblicklichen Sollposition, die der Rampengenerator alle 1 ms neu vorgibt und der Ist-Position. Wird der eingestellte Wert überschritten, so wird F42 (Schleppfehler) ausgelöst. Die Reaktion auf F42 muss über Parameter P834 "Reaktion SCHLEPPFEHLER" eingestellt werden. **Deaktivierung**: Mit P923 Schleppfehlerfenster = 0 ist die Schleppfehlerüberwachung deaktiviert. Einstellbereich: 0 ... **5000** ... 2^{31} - 1 Inkremente
495	LagDistance / LAG DISTAN	**Schleppabstand LESEN.** Betrag des aktuellen Schleppabstands der Positionierung (Differenz zwischen Soll- und Ist-Position). Wertebereich: 0 ... 2^{31} - 1 Inkremente
496	SLS_right / SLS RIGHT	**Software-Endschalter Rechts LESEN und SETZEN.** H496 ist identisch mit P920. Begrenzt die rechte Verfahrdrehrichtung. Die Angabe erfolgt in Anwenderverfahreinheiten. Einstellbereich: - 2^{31} ... **0 Anwendereinheiten** ... 2^{31} - 1 Inkremente
497	SLS_left / SLS LEFT	**Software-Endschalter Links LESEN und SETZEN.** H497 ist identisch mit P921. Begrenzt die linke Verfahrdrehrichtung. Die Angabe erfolgt in Anwenderverfahreinheiten. Einstellbereich: - 2^{31} ... **0 Anwendereinheiten** ... 2^{31} - 1 Inkremente
498	RefOffset / REF.OFFSET	**Referenz-Offset LESEN und SETZEN.** P498 ist identisch mit P900. Der Referenz-Offset ermöglicht eine Verschiebung des Maschinennullpunktes ohne den Referenzpunkt physikalisch zu verschieben. Es gilt die Gleichung: **Maschinennullpunkt = Referenzpunkt + Referenz-Offset** Der Antrieb fährt bei der Referenzfahrt auf den Referenzpunkt und bleibt dort stehen. Der Maschinennullpunkt wird nach der Referenzfahrt mit Referenzpunkt und Referenz-Offset lediglich errechnet. Die Angabe des Referenz-Offset erfolgt in Anwenderverfahreinheiten. Einstellbereich: - 2^{31} ... **0** ... + 2^{31} - 1

Nr.	Name Compiler / Assembler	Beschreibung
499	SetpPosBus / SP.POS.BUS	**Sollposition Bus LESEN.** Enthält die Sollposition, die über die Feldbusprozessdaten gesendet wird. Die Sollposition wird nur dann übernommen, falls "POSITION LO" und "POSITION HI" in der PA-Datenbeschreibung programmiert ist (Parametergruppe P87_).
500	TpPos2_VE / TP.POS2VE	Nur bei MOVIDRIVE® B verwendet, bei MOVIDRIVE® A reserviert. Der Wert des Virtuellen Encoders H376 wird in H500 abgespeichert, wenn der Eingang DI03 betätigt wurde (siehe auch _TouchProbe() / TOUCHP).
501	TpPos1_VE / TP.POS1VE	Nur bei MOVIDRIVE® B verwendet, bei MOVIDRIVE® A reserviert. Der Wert des Virtuellen Encoders H376 wird in H501 abgespeichert, wenn der Eingang DI02 betätigt wurde (siehe auch _TouchProbe() / TOUCHP).
502	TpPos2_Abs / TP.POS2ABS	Die Touch Probe-Positionen werden in den nachstehenden Variablen abgelegt.
503	TpPos1_Abs / TP.POS1ABS	
504	TpPos2_Ext / TP.POS2EXT	
505	TpPos2_Mot / TP.POS2MOT	
506	TpPos1_Ext / TP.POS1EXT	
507	TpPos1_Mot / TP.POS1MOT	

Geber	Geber-Position	Touch Probe 1 DI02	Touch Probe 2 DI03
Motorgeber (X15)	H511 ACTPOS.MOT	H507 TP.POS1MOT	H505 TP.POS2MOT
Externer Geber (X14)	H510 ACTPOS.EXT	H506 TP.POS1EXT	H504 TP.POS2EXT
Absolutwertgeber (X62)	H509 ACTPOS.ABS	H503 TP.POS1ABS	H502 TP.POS2ABS

Nr.	Name Compiler / Assembler	Beschreibung
508	IPOS-Zähler	• Zählerwert für den Frequenzeingang (falls dieser über DIP-Schalter S14 aktiviert ist) • Hochauflösende Motorposition (falls Interpolierter Lagesollwert über P916 eingestellt wird)
509	ActPos_Abs / ACTPOS ABS	**Aktuelle Ist-Position des DIP-Absolutwertgebers (SSI) LESEN.** **ACHTUNG: Systemregelgröße! Wert darf nicht überschrieben werden!** Diese Ist-Position wird durch die Signale, die am Stecker X62 (Option DIP11A) anliegen, ermittelt. Einheit: Inkremente je nach Geberauflösung.
510	ActPos_Ext / ACTPOS EXT	**Aktuelle Ist-Position externer Geber LESEN.** **ACHTUNG: Systemregelgröße! Wert darf nicht überschrieben werden!** Diese Ist-Position wird durch die Spursignale, die am Stecker X14 anliegen, ermittelt. Die Positionsermittlung wird nur durchgeführt, wenn der Stecker X14 als Gebereingang verwendet wird. Einheit: Inkremente je nach Geberauflösung.
511	ActPos_Mot / ACTPOS.MOT	**Aktuelle Ist-Position Motorgeber LESEN.** **ACHTUNG: Systemregelgröße! Wert darf nicht überschrieben werden!** Unabhängig davon, wie viel Impulse der Geber je Umdrehung liefert, hat diese Ist-Position immer die Einheit: 4096 Inkremente pro Motorumdrehung (Geberauflösung 512 Ink., Ausnahme: MQx mit NV26 hat 24 Inkremente pro Motorumdrehung).

Weiterhin sind bei MOVIDRIVE® B folgende Variablen mit Funktionen belegt oder reserviert:

Nr.	Name Compiler / Assembler	Beschreibung
512 ... 515	reserviert	
516	CCount_MotOn	C-Spur (Nullimpuls) Zählen für Motorgeber X15 aktivieren. 0 = Zähler ausgeschaltet / 1 = Zähler aktiviert
517	CCount_ExtOn	C-Spur (Nullimpuls) Zähler für externen Geber X14 aktivieren. 0 = Zähler ausgeschaltet / 1 = Zähler aktiviert
518	CCount_Mot	Zählen der Nullimpulse an X15. Low-Word wird bei jedem Nullimpuls am Motorgeber X15 inkrementiert.
519	CCount_Ext	Zählen der Nullimpulse an X14. Low-Word wird bei jedem Nullimpuls am externen Geber X14 inkrementiert.
520	InpLevelB / INPUTLVLB MOVIDRIVE® B (MOVIDRIVE® A: H483)	**Signalpegel der binären Eingänge nur LESEN.**

Bit	IPOS-Name	DIO (+ DIO)	DIO + DIP	DIO + Feldbus	DIP	DIP + Feldbus	Feldbus P870 = Steuerwort 2
0	DI00	Je nach Grundgeräte, z. B. X13:1					
1	DI01	X13:2					
2	DI02	X13:3					
3	DI03	X13:4					
4	DI04	X13:5					
5	DI05	X13:6					
6	DI06	X16:1					
7	DI07	X16:2					
8	DI10	X22:1	X22:1	X22:1	X60:1	X60:1	Bit 8
9	DI11	X22:2	X22:2	X22:2	X60:2	X60:2	Bit 9
10	DI12	X22:3	X22:3	X22:3	X60:3	X60:3	Bit 10
11	DI13	X22:4	X22:4	X22:4	X60:4	X60:4	Bit 11
12	DI14	X22:5	X22:5	X22:5	X60:5	X60:5	Bit 12
13	DI15	X22:6	X22:6	X22:6	X60:6	X60:6	Bit 13
14	DI16	X22:7	X22:7	X22:7	X60:7	X60:7	Bit 14
15	DI17	X22:8	X22:8	X22:8	X60:8	X60:8	Bit 15
16			X60:1				
17		(X22:1)	X60:2				
18		...	X60:3				
19		(X22:8)	X60:4				
20			X60:5				
21			X60:6				
22			X60:7				
23			X60:8				

Nr.	Name Compiler / Assembler	Beschreibung
521	OutpLevelB / OUTPUTLVLB MOVIDRIVE® B (MOVIDRIVE® A: H482)	**Signalpegel der binären Ausgänge nur LESEN.**

Bit	IPOS-Name	DIO (+ DIO)	DIO + DIP	DIO + Feldbus	DIP	DIP + Feldbus	Feldbus P873 = Statuswort 2
0	DB00	DB00	DB00	DB00	DB00	DB00	DB00
1	DO01	DO01	DO01	DO01	DO01	DO01	DO01
2	DO02	DO02	DO02	DO02	DO02	DO02	DO02
3	DO03	DO03	DO03	DO03	DO03	DO03	DO03
4	DO04	DO04	DO04	DO04	DO04	DO04	DO04
5	DO05	DO05	DO05	DO05	DO05	DO05	DO05
6	DO10	X23:1	X23:1	X23:1	X61:1	X61:1	Bit 8
7	DO11	X23:2	X23:2	X23:2	X61:2	X61:2	Bit 9
8	DO12	X23:3	X23:3	X23:3	X61:3	X61:3	Bit 10
9	DO13	X23:4	X23:4	X23:4	X61:4	X61:4	Bit 11
10	DO14	X23:5	X23:5	X23:5	X61:5	X61:5	Bit 12
11	DO15	X23:6	X23:6	X23:6	X61:6	X61:6	Bit 13
12	DO16	X23:7	X23:7	X23:7	X61:7	X61:7	Bit 14
13	DO17	X23:8	X23:8	X23:8	X61:8	X61:8	Bit 15
14			X61:1				
15		(X23:1)	X61:2				
16		...	X61:3				
17		(X23:8)	X61:4				
18			X61:5				
19			X61:6				
20			X61:7				
21			X61:8				

Nr.	Name Compiler / Assembler	Beschreibung
522	RecStatS1 / SBUS1REC	**Statuswort für den Empfang von SCOM-Datenobjekten (Doppelworten) über den Systembus** Für jedes mit _SBusCommDef / SCOM eingerichtete Datenobjekt wird ein Empfangs-Bit reserviert. Dem im IPOS-Programm zuerst initialisierten Receive-Objekt ist Bit 0 zugeordnet, dem Zweiten ist Bit 1 zugeordnet, usw. Empfängt das MOVIDRIVE® ein Telegram eines initialisierten Receive-Objekts, wird das zugehörige Bit gesetzt. Das Bit kann nur im Anwenderprogramm zurückgesetzt werden. Über den SBus können ereignis-orientiert Telegramme gesendet und empfangen werden, wenn auf das entsprechende Bit aus H522 ein Variablen-Interrupt eingerichtet wird. Das Rücksetzen des Bits muss dabei der letzte Befehl in der Interrupt-Routine sein. Mit dem Aufbau eines Prozessabbildes muss der Anwender sicherstellen, das keine Neben-effekte durch den Empfang des gleichen Objekts während der Verarbeitung entstehen (zyklischer Empfang eines Objekts). Verwenden Sie zum Rücksetzen des Bits den BITCLEAR-Befehl, damit Receive-Bits anderer Übertragungen nicht verloren gehen.
523	RecStatS2 / SBUS2REC	**Nur bei CAN-Bus über DFC11B: Statuswort für den Empfang von SCOM-Datenobjekten (Doppel-worten) über den CAN-Bus** Für jedes mit _SBusCommDef / SCOM eingerichtete Datenobjekt wird ein Empfangs-Bit reserviert. Dem im IPOS-Programm zuerst initialisierten Receive-Objekt ist Bit 0 zugeordnet, dem Zweiten ist Bit 1 zugeordnet, usw. Empfängt das MOVIDRIVE® ein Telegram eines initialisierten Receive-Objekts, wird das zugehörige Bit gesetzt. Das Bit kann nur im Anwenderprogramm zurückgesetzt werden. Über den SBUS können ereignis-orientiert Telegramme gesendet und empfangen werden, wenn auf das entsprechende Bit aus H523 ein Variablen-Interrupt eingerichtet wird. Das Rücksetzen des Bits muss dabei der letzte Befehl in der Interrupt-Routine sein. Mit dem Aufbau eines Prozessabbildes muss der Anwender sicherstellen, dass keine Neben-effekte durch den Empfang des gleichen Objekts während der Verarbeitung entstehen (zyklischer Empfang eines Objekts). Verwenden Sie zum Rücksetzen des Bits den BITCLEAR-Befehl, damit Receive-Bits anderer Übertragungen nicht verloren gehen.

Nr.	Name Compiler / Assembler	Beschreibung
524	IPOS_Setp / IPOS_SETP	**IPOS-Sollwert, Stellgröße des PID-Reglers, wenn H540 =1.** Wenn H540 = 0 oder 2 auch direkt vom Anwenderprogramm beschreibbar. H524 kann als Momentensollwert bzw. Drehzahl-Sollwert verwendet werden, wenn P100 Sollwertquelle = IPOS und P700 Betriebsart 1 = xxx&M-Regelung bzw. CFC oder SERVO gesetzt sind. 1 Inkrement in H524 entsprechen dann 0,2 1/min Drehzahl-Sollwert bzw. 0,01 % I_N Momentensollwert
525 ... 529	reserviert	
530	VarIntReq / VARINTREQ	Wird das entsprechende Request-Bit eines Variablen-Interrupts gesetzt, wird unabhängig von der tatsächlichen Interrupt-Bedingung ein Variablen-Interrupt ausgelöst. Der entsprechende Variablen-Interrupt muss zuvor aktiviert worden sein. Bit 0: Anforderung für Variablen-Interrupt 0 Bit 1: Anforderung für Variablen-Interrupt 1 Bit 2: Anforderung für Variablen-Interrupt 2 Bit 3: Anforderung für Variablen-Interrupt 3
540	PID_Mode / PID.MODE	Betriebsart des PID-Reglers, H540 und P260 sind identisch. 0 = Regler ausgeschaltet (default) 1 = Regelung aktiv 2 = Sprungantwort (geöffneter Regelkreis)
541	PID_K_p / PID.KP	PID-Regler: Faktor des Proportionalteils, H541 und P263 sind identisch, 3 Nachkommastellen; 0 <= K_P <= 32000 (= 32,000); Default: 1000 (= 1,0)
542	PID_Outp_P / PID.OUTPP	PID-Regler: Aktueller Wert des Regler P-Anteils
543	PID_Outp_I / PID.OUTPI	PID-Regler: Aktueller Wert des Regler I-Anteils Der Wert für den I-Anteil steht im High-Wort, im Low-Wort stehen interne Nachkomma-Anteile, z. B. H543 = 0x30000 ? I-Anteil = 3.
544	PID_Outp_D / PID.OUTPD	PID-Regler: Aktueller Wert des Regler D-Anteils
545	PID_Feedf / PID.FEEDF	PID-Regler Vorsteuerwert; H545 und P266 sind identisch -32000 <= Vorsteuerung <= 32000; Default: 0
546	PID_Command / PIDCOMMAND	PID-Regler: Sollwert, H546 und P271 sind identisch Wenn P270 = 0 (= "Parameter"), dann enthält P271 / H546 den gewünschten Sollwert des Prozesses -32000 <= Sollwert <= 32000 (bei Drehzahlregelung entspricht 1 Inkrement 0,2/min); Default: 0
547	PID_CmdAdr / PID.CMDADR	PID-Regler: Adresse des Sollwerts, H547 und P272 sind identisch Wenn P270 = 1 (= "IPOS Variable"), dann enthält P272 / H547 die Adresse der IPOS-Variable mit dem Sollwert; Default: 0
548	PID_CmdScale / PID.CMDSCA	PID-Regler: Faktor für die Skalierung der Sollwerte; H548 und P274 sind identisch, gewichtet mit 3 Nachkommastellen -32000 (-32,000) <= $K_{Sollwert}$ <= 32000 (32,000); Default: 1000 (1,0)
549	PID_ActAdr / PID.ACTADR	PID-Regler: Adresse des Istwertes, H549 und P276 sind identisch Wenn P275 = "IPOS Variable", dann enthält P276 / H549 die Adresse der IPOS-Variable; Default: 0
550	PID_ActScale / PID.ACTSCA	PID-Regler, Skalierungsfaktor des gefilterten Istwertes, H550 und P277 sind identisch gewichtet mit 3 Nachkommastellen -32,000 <= $K_{Istwert}$ <= 32,000; Default: 1000 (1,0)
551	PID_ActNorm / PID.ACTNOR	PID-Regler: Gefilterter und skalierter Istwert, Diagnosewert
552	PID_ActOffset / PID.ACTOFF	PID-Regler: Ganzzahliger, permanenter Offset des Istwertes, H552 und P278 sind identisch -32000 (-32000) <= Offset <= 32000 (32000); Default: 0
553	PID_ActMin / PID.ACTMIN	PID-Regler: Minimalwert für Istwert nach Glättung, Skalierung und Offset, H553 und P280 sind identisch -32000 (-32000) <= $x_{e,min}$ <= 32000 (32000); Default: 0
554	PID_ActMax / PID.ACTMAX	PID-Regler: Maximalwert für Istwert nach Glättung, Skalierung und Offset, H554 und P281 sind identisch -32000 (-32000) <= $x_{e,max}$ <= 32000 (32000); Default: 10000 (10,0)
555	PID_LimitMin / PID.LMTMIN	PID-Regler: Minimaler Ausgangswert, H555 und P282 sind identisch -32000 (-32000) <= $x_{Regler,min}$ <= 32000 (32000); Default: -1000 (-1,0)
556	PID_Limit_Max / PID.LMTMAX	PID-Regler: Maximaler Ausgangswert, H556 und P283 sind identisch -32000 (-32000) <= $x_{Regler,max}$ <= 32000 (32000); Default: 10000 (1,0)

Nr.	Name Compiler / Assembler	Beschreibung
557	PID_SetpMin / PID.SETMIN	PID-Regler: Minimaler Ausgangswert der Stellgröße, H557 und P284 sind identisch -32000 (-32000) <= $x_{a,min}$ <= 32000 (32000); Default: 0
558	PID_SetpMax / PID.SETMAX	PID-Regler: Maximaler Ausgangswert der Stellgröße, H558 und P285 sind identisch -32000 (-32000) <= $x_{a,max}$ <= 32000 (32000); Default: 7500 (7,5)
559	PID_Status / PID.STATUS	PID-Regler Statuswort Bit 0 = Summe aus Istwert und Offset überschreitet Grenze xe_{min} Bit 1 = Summe aus Istwert und Offset überschreitet Grenze xe_{max} Bit 2 = Wert des Regler P-Anteils wird begrenzt Bit 3 = I-Anteil des Reglers abgeschaltet Bit 4 = Wert des Regler I-Anteils wird begrenzt Bit 5 = Wert des Regler D-Anteils wird begrenzt Bit 6 = PID-Regler Stellgröße wird begrenzt Bit 7 = Summe aus PID-Regler Stellgröße und Vorsteuerung wird begrenzt
560	reserviert	

5 Task-Verwaltung und Interrupts

5.1 Einführung

IPOS$^{plus®}$ kann mehrere Teilprogramme parallel bearbeiten. Ein Teilprogramm entspricht einer Task. Die folgenden Funktionen können Interrupts für Task 1 auslösen:

- Timer0-Überlauf
- Systemfehler / Gerätefehler
- Touch-Probe DI02

MOVIDRIVE® B kann Task 2 und Task 3 durch 4 weitere Interrupts unterbrechen, die durch den Vergleich mit einem Variablenwert ausgelöst werden.

MOVIDRIVE® A kann 2 unabhängig voneinander laufende Teilprogramme - Task1 und Task 2 - ausführen.

MOVIDRIVE® B kann 3 unabhängig voneinander laufende Teilprogramme - Task1, Task 2 und Task 3 - ausführen.

Sie können ein MOVIDRIVE® B wie ein MOVIDRIVE® A betreiben.

Die folgenden Variablen sind global. Alle Tasks und Interrupts nutzen die Variablen gemeinsam:

- H0 - H511 bei MOVIDRIVE® A
- H0 - H1023 bei MOVIDRIVE® B

Es gibt somit keine lokalen Variablen, die nur in einer Task oder in einer Funktion deklariert sind. Eine Übersicht über den für die Systemvariablen reservierten Bereich finden Sie im Kapitel "IPOS-Variablen / Übersicht über die Systemvariablen“.

Sie können die Befehle in einem Programmfenster eingeben. Sie müssen alle Programmteile mit derselben Sprache erstellen: Assembler oder Compiler.

5.2 Task-Verwaltung bei MOVIDRIVE® A und B

Task 1 ist das Hauptprogramm. Sie können Task 1 über das Blitzsymbol in der Symbolleiste oder über das Handbediengerät (P931) starten. Initialisieren und starten Sie Task 2 und Task 3 durch Programmbefehle. Task 3 gibt es nur bei MOVIDRIVE® B. Der Interrupt wird durch Programmbefehle initialisiert und durch ein Interrupt-Ereignis ausgelöst. Wenn Sie Task 1 über das STOP-Symbol in der Symbolleiste stoppen, so stoppt die gesamte IPOS-Bearbeitung. Mit dem Handbediengerät DBG können alle Tasks über den Parameter P931 = HALT gestoppt werden. Mit P931 = STOP wird nur der Task 1 gestoppt. Nach dem Neustart wird das Programm da fortgesetzt, wo es unterbrochen wurde.

Symbolleiste bei MOVIDRIVE® A

473926155

[1] = Status von Task 1: START = gestartet
[2] = Status von Task 2: PSTOP = gestoppt
[3] = Blitzsymbol zum Starten von Task 1
[4] = STOP-Symbol um die gesamte IPOS-Bearbeitung zu stoppen

Symbolleiste bei MOVIDRIVE® B

474186507

[1] = Status von Task 1: PSTOP = gestoppt
[2] = Status von Task 2: PSTOP = gestoppt
[3] = Status von Task 3: PSTOP = gestoppt
[4] = Blitzsymbol zum Starten von Task 1
[5] = STOP-Symbol um die gesamte IPOS-Bearbeitung zu stoppen

IPOS$^{plus®}$ bearbeitet einen Task zyklisch. Ein Task beginnt nach dem letzten Befehl wieder mit dem ersten Befehl. Wenn Sie eine Initialisierungs-Routine in Task 1 nur einmalig ausführen wollen, so können Sie diesen Programmteil durch eine Endlosschleife ausgrenzen:

```
main()
{
   // Programmcode nur Initialisierung
   while(1)
   {
   // zyklischer Programmcode
   }
}
```

Die folgende Tabelle gibt einen Überblick über die Funktionen und Eigenschaften der Tasks und der Interrupts

	Task 1	Task 2	Task 3 (nur MOVIDRIVE® B)	"Task1"-Interrupt	Variablen-Interrupt (nur MOVIDRIVE® B)
Starten MOVIDRIVE® A	Blitzsymbol oder P931 im Handbediengerät	_SetTask2(ST2_START, Task2Name); bzw TASK2 START Mxx	Nicht vorhanden	Durch das definierte Interrupt-Ereignis	Nicht vorhanden
Starten MOVIDRIVE® B		_SetTask(ST2_START, Task2Name); bzw TASK TASK2, START Mxx	_SetTask(ST3_START, Task3Name); bzw TASK TASK3, START Mxx		Durch das definierte Interrupt-Ereignis
Stoppen MOVIDRIVE® A	STOP-Symbol oder P931 im Handbediengerät	STOP-Symbol oder _SetTask2(ST2_STOP, Task2Name); bzw TASK2 STOP Mxx	Nicht vorhanden	Wie Task 1	Nicht vorhanden
Stoppen MOVIDRIVE® B		STOP-Symbol oder _SetTask(ST2_STOP, Task2Name); bzw TASK TASK2, STOP Mxx	STOP-Symbol oder _SetTask(ST3_STOP, Task3Name); bzw TASK TASK3, STOP Mxx		Wie Task 1 oder wie zugeordnete Task 2 oder 3
Unterbrechung MOVIDRIVE® A	Durch "Task1"-Interrupt	Kann nicht unterbrochen werden	Nicht vorhanden	Durch anderen Task 1-Interrupt höherer Priorität	Nicht vorhanden
Unterbrechung MOVIDRIVE® B	Durch "Task1"-Interrupt	Durch Variablen-Interrupt	Durch Variablen-Interrupt	Durch anderen Task 1-Interrupt höherer Priorität	Durch Variablen-Interrupt in derselben Task mit höherer Priorität
Debuggen mit Breakpoint und Einzelschritt	Ja	Nein[1)]	Nein[1)]	Ja	Nein[1)]
Befehlsabarbeitungszeit MOVIDRIVE® A	1 Assemblerbefehl / ms	2 Assemblerbefehle / ms	Nicht vorhanden	Assemblerbefehl / ms	Nicht vorhanden
Befehlsabarbeitungszeit MOVIDRIVE® B	1 ... 10 Assembler-Befehle / ms, Werkseinstellung: 1 Befehl / ms	2 ... 11 Assembler-Befehle / ms, Werkeinstellung: 2 Befehle / ms	mind. 1 Befehl pro ms, je nach Prozessor-Auslastung werden weitere Befehle bearbeitet	Wie Task 1	Wie der Task, zu der der Interrupt zugeordnet ist (Task 2 oder 3)

1) Zum Debuggen Befehle in Task 1 kopieren.

5.3 Tasks bei MOVIDRIVE® A

Hier finden Sie zusätzlich zu dem allgemeinen Teil im Kapitel "Task-Verwaltung bei MOVIDRIVE® A und B" spezifische Implementierungshinweise für das MOVIDRIVE® A: Der Bewegungsablauf mit den Positionierbefehlen wird typischerweise in Task 1 programmiert.

Programmieren Sie die folgenden Funktionen in Task 2:

- Schnelle, zeitkritische Vorgänge
- Berechnungen
- Beobachtung von Systemgrößen
- Kommunikation mit den SEW-Bedienterminals
- Zyklisches Kopieren von Variablen auf die Oszilloskop-Variablen H474, H475
- Aufbereitung der Prozessdaten des Feldbusses / SBus mit einer übergeordneten Steuerung oder anderen MOVIDRIVE®

Damit führt IPOS$^{plus®}$ diese Funktionen auch durch, wenn in Task 1 die Interrupt-Routine aktiv ist.

5.4 Tasks bei MOVIDRIVE® B

Hier finden Sie zusätzlich zu dem allgemeinen Teil im Kapitel "Task-Verwaltung bei MOVIDRIVE® A und B" spezifische Implementierungshinweise für das MOVIDRIVE® B:

5.4.1 Abarbeitungszeit Task 1 / Task 2

Die Werkseinstellung für die Abarbeitungszeit der Tasks ist:

- Task 1: 1 Befehl / ms (P938 = 0)
- Task 2: 2 Befehle / ms

Sie können die Abarbeitungszeit in beiden Tasks zusammen um bis zu 9 zusätzliche Befehle pro ms beschleunigen. Sie können die zusätzlichen Befehle für Task 1 über Parameter P938 (Index 8888) und für Task 2 über Parameter P939 (Index 8962) zuteilen. Demnach können in Task 1 maximal 1 + 9 = 10 und Task 2 maximal 2 + 9 = 11 Befehle / ms ausgeführt werden.

Wenn Sie die maximale Anzahl von 9 zusätzlichen Befehlen / ms auf Task 1 und Task 2 aufteilen, so ergeben sich die folgenden möglichen Kombinationen:

Task 1		Task 2	
P938	**Befehle / ms**	**P939**	**Befehle / ms**
0	1	9	11
1	2	8	10
2	3	7	9
3	4	6	8
4	5	5	7
5	6	4	6
6	7	3	5
7	8	2	4
8	9	1	3
9	10	0	2

Beispiel: P938 = 2, P939 = 3 => Task 1 bearbeitet 3 Befehle / ms, Task 2 bearbeitet 5 Befehle / ms

5.4.2 Task 3

Task 3 ist ab den Geräten der B-Reihe verfügbar. Task 3 bearbeitet mindestens 1 Befehl je ms. Abhängig von der Gerätekonfiguration und von P938 / P939 führt Task 3 weitere Befehle aus. Typisch sind 20 ... 40 Befehle / ms. Der absolute Ressourcenbedarf zur Befehlsbearbeitung ist in Task 3 ca. 20 ... 40 % geringer als in Task 1 oder Task 2. Applikationsteile, bei denen die garantierte Ausführungszeit einzelner Programmzeilen nicht wichtig ist, werden in Task 3 schneller bearbeitet.

5.4.3 Implementierungshinweise

Programmieren Sie im Unterschied zu MOVIDRIVE® A den Bewegungsablauf in Task 1 oder in Task 3.

Programmieren Sie die folgenden Funktionen in Task 2 oder Task 3:

- Schnelle, zeitkritische Vorgänge
- Berechnungen
- Beobachtung von Systemgrößen
- Kommunikation mit den SEW-Bedienterminals
- Zyklisches Kopieren von Variablen auf die Oszilloskop-Variablen H474, H475
- Aufbereitung der Prozessdaten des Feldbusses / SBus mit einer übergeordneten Steuerung oder anderen MOVIDRIVE®

Damit führt IPOS$^{plus®}$ diese Funktionen auch durch, wenn in Task 1 die Interrupt-Routine aktiv ist.

Beachten Sie dabei, dass in Task 2 die Abarbeitungszeit pro Befehl im Gegensatz zu Task 3 deterministisch ist.

5.4.4 Beispiel

Ein MOVIDRIVE® B positioniert einen Fahrantrieb. Eine SPS steuert das MOVIDRIVE® über einen Feldbus. Ändern Sie einzelne Parameter mit SEW-Bediengeräten direkt am MOVIDRIVE®.

Lösungsvorschlag:

Task 1: Programmierung des Bewegungsablaufs

Task 2: HMI-Kommunikation mit dem Bedienterminal

Task 3: Feldbuskommunikation mit der SPS

Je nach Anwendungsfall ist es wichtig, die zusätzlichen Befehle richtig zu verteilen:

- **Interrupt-orientierte Programme:** Wenn ein Anwenderprogramm Interrupt-orientiert arbeitet und die Task 1-Interrupts schnell bearbeitet werden sollen, so muss Task 1 eine hohe Rechenpriorität in Form von zusätzlichen Befehlen in P938 zugewiesen werden.
- **Laufzeit-optimierte Programme:** Soll in IPOS z. B. eine Prozessdatenkonvertierung vorgenommen werden, kommt es darauf an, diese möglichst schnell abzuschließen. Die Task 3 kann dazu verwendet werden, die Konvertierungsroutinen möglichst schnell abzuarbeiten. In diesem Fall sollten Task 1 und Task 2 möglichst wenig zusätzliche Befehle/ms erhalten. Die schnellste Gesamt-Applikationsleistung wird somit erreicht, falls Task1 und 2 mit Minimalgeschwindigkeit laufen.

5.5 Interrupts

Ein Interrupt unterbricht - ausgelöst durch ein Ereignis - die Bearbeitung der Task, der er zugeordnet ist. Die Interrupt-Routine wird ein Mal vollständig durchlaufen, solange sie nicht durch einen höher prioren Interrupt desselben Task unterbrochen wird.

Ein Interrupt, der mit _SetInterrupt() bzw. SETINT aktiviert wird, kann durch einen Timer0-Überlauf, einen System- / Gerätefehler oder Touch-Probe DI02 ausgelöst werden und unterbricht Task 1.

Bei MOVIDRIVE® B können zusätzlich bis zu 4 Variablen-Interrupts mit _SetVarInterrupt() bzw. VARINT aktiviert werden, die wahlweise Task 2 oder Task 3 unterbrechen.

Bei Task 1-Interrupts (Gerätefehler, DI02-Touchprobe, oder T0_Überlauf) ist die Reaktionszeit (Latenzzeit) abhängig von der Anzahl der aktivierten Interrupts (1 Interrupt <= 1 ms, 2 Interrupt <= 2 ms, 3 Interrupt <= 3 ms). Bei Variablen-Interrupts ist die Reaktionszeit unabhängig von der Anzahl der aktivierten Interrupts <= 1ms.

Wird ein Interrupt ausgelöst während eines wartenden Befehls, läuft die Wartezeit des Befehls im Hintergrund weiter. Nach dem Rücksprung in den Task wird nur noch die Restzeit gewartet.

5.5.1 Beispiel

Ein WAIT-1000-ms-Befehl in Task 1 wird nach 500 ms durch einen Interrupt unterbrochen. Wird die Bearbeitung von Task 1 nach 175 ms fortgesetzt, beträgt die Restlaufzeit 325 ms.

5.6 Interrupts bei MOVIDRIVE® A und B

In den MOVIDRIVE®-A-Geräten können folgende Interrupts verwendet werden:

- Timer0-Überlauf (H489)-Interrupt Priorität = 1 (niedrigste Priorität)
- Touch-Probe-DI02-Interrupt Priorität = 2
- Fehler-Interrupt Priorität = 3 (höchste Priorität)

Es wird dabei jeweils Task 1 unterbrochen. Theoretisch kann gleichzeitig je ein Timer0, ein Touch-Probe und ein Fehler-Interrupt aktiv sein. Ein Interrupt mit höherer Priorität kann die Abarbeitung eines anderen Interrupt unterbrechen. Mit dem Argument DISABLE werden alle Interrupts deaktiviert. (siehe _SetInterrupt bzw. SETINT)

5.6.1 Aktivierung des Interrupts

Compiler: `_SetInterrupt(event ,myfunction );`

Aktiviert einen Interrupt. Wenn das Ereignis `event` eintritt, wird anstelle von Task 1 die Funktion `myfunction` ausgeführt.

Assembler: `SETINT event, Mxx`

Aktiviert einen Interrupt. Wenn das Ereignis `event` eintritt, werden anstelle von Task 1 die Befehle ab Marke Mxx ausgeführt.

5.6.2 Fehler-Interrupt

Die Interrupt-Routine wird ein Mal vollständig durchlaufen, wenn ein Fehler ansteht. Nach einem Durchlauf wird ein Assembler-Befehl von Task1 bearbeitet, bevor geprüft wird, ob der Fehler immer noch ansteht. Steht der Fehler immer noch an, wird wieder die Interrupt-Routine bearbeitet. Um die Interrupt-Routine erst dann zu verlassen, wenn der Fehler nicht mehr ansteht, muss sie also eine Schleife auf diese Bedingung enthalten.

Je nachdem welche Fehlerreaktion in Parametergruppe 83x oder mit dem Befehl _FaultReaction bzw. SETFR eingestellt wurde, ergibt sich folgendes Verhalten:

- Es wird auf einen bestimmten Fehler kein Interrupt ausgelöst, wenn die Fehlerreaktion dieses Fehlers auf KEINE REAKTION eingestellt ist.
- Ist die Fehlerreaktion eines Fehlers xy auf "... und Warnung" eingestellt, wird Task 1 nach Fehler-Reset an der unterbrochenen Stelle fortgesetzt.
- Ist die Fehlerreaktion eines Fehlers xy auf "... und Fehler" eingestellt, wird IPOS nach Fehler-Reset neu gestartet und die Variablen mit den Werten aus dem EPROM neu initialisiert. Hinweis: Sie können in diesem Fall vor dem Fehler-Reset wichtige Variablenwerte netzausfallsicher mit dem MEM- oder dem MOVLNK-Befehl speichern. Beachten Sie dabei, dass die Anzahl der zulässigen Schreibzugriffe nicht überschritten wird (siehe MEM bzw. _Memorize()).

Beispiel

```
fnErrorInterrupt()
{
   H2++;
   while( !(StatusWord & 0b10))
   {
      //while-Schleife erst verlassen, wenn Antrieb fehlerfrei
      H1++;
   }
}

main()
{
   _SetInterrupt(SI_ERROR, fnErrorInterrupt);
   while(1)
   {
      H0++;
   }
}
```

H0 wird inkrementiert, so lange der Umrichter fehlerfrei ist. Im Fehlerfall wird H2 um eins erhöht und H1 so lange inkrementiert bis der Umrichter fehlerfrei ist. Danach wird abhängig von der eingestellten Fehlerreaktion mit den aktuellen Werten in Task1 weitergearbeitet oder IPOS mit den Werten aus dem EPROM neu gestartet.

5.6.3 Touch-Probe-DI02 Interrupt

Die Interrupt-Routine wird ein Mal vollständig durchlaufen, wenn der Touch-Probe mit dem Befehl _TouchProbe (condition) bzw. TOUCHP condition freigegeben wurde und die Flankenbedingung eintritt. Danach wird Task1 weiter abgearbeitet. Die Interrupt-Routine wird erst dann ein zweites Mal aufgerufen, wenn der Touch-Probe erneut freigegeben wurde und danach die Flankenbedingung eintritt (siehe auch Befehl _TouchProbe bzw. TOUCHP).

Beispiel

```
fnTouchInterrupt()
{
H0++;
}
/*=============================================
Hauptfunktion (IPOS-Eintrittsfunktion)
================================================*/
main()
{
   _SetInterrupt( SI_TOUCHP1,fnTouchInterrupt); //Interrupt-Rout. akt.
   TouchProbe( TP_EN1 );
   while (1) { H1 = H1 +1; }
}
```

H0 wird ein einziges Mal um 1 erhöht.

Würde der Touch-Probe-Befehl zusätzlich in der while-Schleife oder in fnTouchInterrupt aufgerufen, würde der Interrupt auf alle Flankenwechsel an DI02 reagieren.

Typische Anwendungsfälle für den Touch-Probe sind die Positionierung um ein relatives Wegstück bei Abstapeleinrichtungen oder die Markenregelung bei Prozessen mit kontinuierlichem Materialfluss.

5.6.4 Timer0-Interrupt

Die Interrupt-Routine wird ein Mal vollständig durchlaufen, wenn der Timer abgelaufen (=0) ist. Nach einem Durchlauf wird ein Assembler-Befehl von Task1 bearbeitet, bevor geprüft wird, ob der Timer = 0 ist. Ist die Bedingung erfüllt, wird wieder in die Interrupt-Routine verzweigt.

Um in äquidistanten Abständen einen Timer0-Interrupt auszulösen, kann in Variable H485 T0_Reload die Zykluszeit eingestellt werden, mit der der Timer0 beim Eintritt in die Interrupt-Routine automatisch neu geladen wird.

	HINWEIS Da der Timer0 abwärts zählt, wäre die Interrupt-Bedingung ständig erfüllt, wenn T0_Relaod = 0 ist und der Wert des Timers in der Interrupt-Routine nicht verändert wird.

Damit ergeben sich folgende Einsatzmöglichkeiten:

- Soll mit dem Timer0-Interrupt ein Programmteil in äquidistanten Abständen bearbeitet werden, muss der Timer0 z. B. mit T0_Reload neu geladen werden.
- Soll mit dem Timer0-Interrupt ein Programmteil eine definierte Zeit nach dem Start von IPOS einmalig durchlaufen werden, muss der Timer0 im Interrupt auf -1 gesetzt werden.

Beispiel

```
fnTimerInterrupt()
{
   H0 = H0 +1;
   T0_Reload = 10000; //Timer 0 automatisch neu laden mit 10s
   _SetInterrupt( SI_TIMER0,fnTimerInterrupt);  // Interrupt aktivieren
}
main()
{
   while (1) {H1 = H1 +1;}
}
```

H0 wird alle 10 s um 1 erhöht.

5.7 Variablen-Interrupts bei MOVIDRIVE® B

In den MOVIDRIVE®-B-Geräten können alle Interrupts von MOVIDRIVE® A (siehe Kapitel "Task-Verwaltung bei MOVIDRIVE® A und B") in identischer Form verwendet werden und zusätzlich bis zu 4 Variablen-Interrupts.

Die Interrupts auf bestimmte Variablenwerte können z. B. reagieren auf

- einen Stückzahlwert,
- alle Timer 0, 1 und 2,
- das Erreichen einer Achsposition der eigenen oder einer anderen Achse
- die Änderung eines I/O-Signals
- einen bestimmten Umrichterzustand (H473) oder
- wenn neue Daten über den S-Bus empfangen werden oder gesendet werden sollen.

5.7.1 Aufruf des Variablen-Interrupts

Compiler: _SetVarInterrupt(pData ,myfunction);

aktiviert einen Variablen-Interrupt mit der Datenstruktur ab der Variable pData, der die Funktion myfunction ausführt, wenn das Interrupt-Ereignis eintritt.

Assembler: VARINT Hxx, Mxx

aktiviert einen Variablen-Interrupt mit der Datenstruktur ab der Variable Hxx, der die Befehle ab Marke Mxx ausführt, wenn das Interrupt-Ereignis eintritt.

In der Datenstruktur können folgende Eigenschaften und Funktionen des Variablen-Interrupts definiert werden (siehe auch Befehl _SetVarInterrupt bzw. VARINT):

- welche Task unterbrochen wird - Task 2 oder Task 3
- fortlaufende Nummer des Interrupts (0... 3)
- die Bezugsvariable und der Wert, die miteinander verglichen werden
- die Art des mathematischen Vergleichs (==, <, Flanke...)
- die Abarbeitungszeit: entweder so lange, wie die Bedingung erfüllt ist oder einmalig jedes Mal, wenn die Bedingung erfüllt wird (flankengetriggert)
- die Priorität des Interrupts
- der Wert der Bezugsvariable mit der der Interrupt getriggert wurde.

	HINWEIS Die Datenstruktur des Befehls ist bei der Systemfunktion beschrieben (Compiler – Funktionen / Assembler – Befehle).

Das Interrupt-Verhalten kann zur Laufzeit dynamisch angepasst werden, indem entweder

- die Datenstruktur geändert wird und danach der Befehl erneut aufgerufen wird (notwendig, wenn sich z. B. der Wert CompVar, mit dem verglichen wird, ändert) oder
- der Befehl mit einer anderen Datenstruktur aber dem identischen Wert in der Variablen H+1 (IntNum) aufgerufen wird.

5.7.2 IPOS-Zugriff auf die interne Interrupt-Steuerung

Die Information, ob ein Variablen-Interrupt angefordert wurde, ist im IPOS$^{plus®}$-Programm in der Variable uVarEventRequest (H530 Bit 0 bis 3) verfügbar. Diese "Request"-Bits können auch im IPOS$^{plus®}$-Programm beschrieben werden.

uVarEventRequest H530.0	Request für Variablen-Interrupt 0
uVarEventRequest H530.1	Request für Variablen-Interrupt 1
uVarEventRequest H530.2	Request für Variablen-Interrupt 2
uVarEventRequest H530.3	Request für Variablen-Interrupt 3

Damit kann z. B. während der Erstinbetriebnahme für Testzwecke unabhängig von der tatsächlichen Interrupt-Bedingung das Request-Bit gesetzt und ein Variablen-Interrupt ausgelöst werden (sofern zuvor der entsprechende Variablen-Interrupt aktiviert wurde).

Mit einem Variablen-Interrupt kann eine zeitgesteuerte Programmabarbeitung realisiert werden, z. B. zyklische Berechnung der Beschleunigung aus einer Geschwindigkeit.

Weiterhin kann während der Bearbeitung eines hochprioren Variablen-Interrupts durch Löschen des entsprechenden Request-Bits ein anstehender, niederpriorer Interrupt gelöscht werden.

Beispiel

Die Transportachse einer Abfüllmaschine soll über DO01 eine Dosiereinheit ansteuern, wenn sie die Position 5° an der Last überfährt. Der Ausgang soll unabhängig von der Taktzahl und der Achsposition 200 ms später wieder ausgeschaltet werden.

Beispiel gelöst im Compiler

Notwendige Parametereinstellungen

P620 = IPOS-Ausgang, P960 =z. B. KURZ

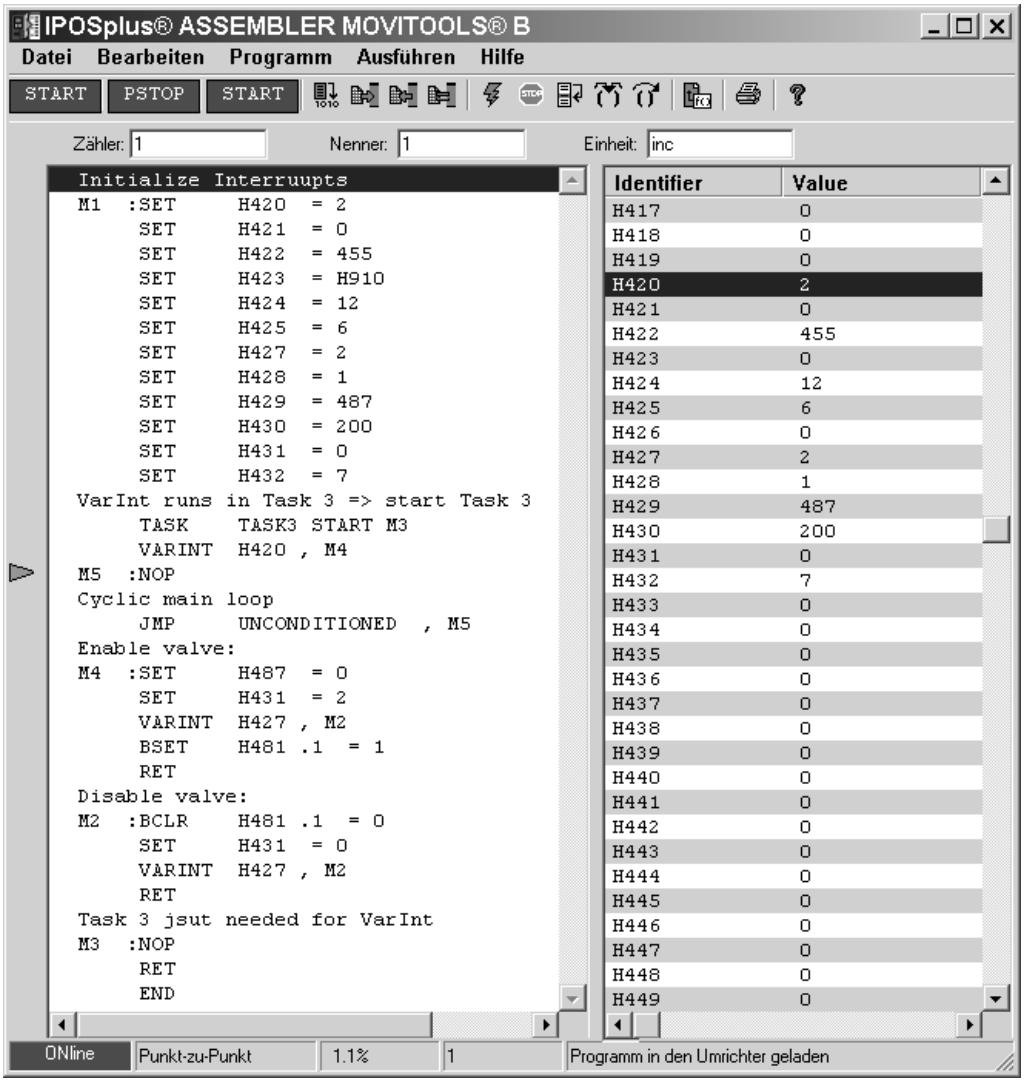
```
/*=============================================
IPOS-Quelldatei
=============================================*/
#include <constb.h>
#include <iob.h>

// notwendige Parametereinstellungen:
// P620 = IPOS-Ausgang, P960 =z.B. KURZ

VARINT hOPENvalve, hCLOSEvalve;

fnTask3()
{ //Task 3 wird nur benötigt, damit VarInt. aktiviert werden
   H1 = H1;
   //dummy-Befehl
}

fnOPENvalve()
{ //Dosiereinheit einschalten
   Timer_2 = 0;
   //Timer 2 auf 0 rücksetzen
   hCLOSEvalve.Mode = 2;
   //Stop-IRQ aktivieren
   _SetVarInterrupt( hCLOSEvalve,fnCLOSEvalve );
   _BitSet( StdOutpIPOS, 1);
   //DO01 setzen

}
fnCLOSEvalve()
{ //Dosiereinheit ausschalten
   _BitClear( StdOutpIPOS, 1);
   //DO01 löschen
   hCLOSEvalve.Mode = 0;
   //Stop-IRQ deaktivieren
   _SetVarInterrupt( hCLOSEvalve,fnCLOSEvalve );
}

/*==============================================
Hauptfunktion (IPOS-Eintrittsfunktion)
================================================*/
main()
{ //Initialisierungsteil
   hOPENvalve.Control = 2;        //Task3 unterbrechen
   hOPENvalve.IntNum =0;
   //fortlaufende Nr.
   hOPENvalve.SrcVar = numof ( ModActPos );   //Motorgeber Modulo
   hOPENvalve.CompVar = 910;

   // 5° an der Last = 5° x 910/65536
   hOPENvalve.Mode = 12;          // einmalig bei > = 5°
   hOPENvalve.Priority = 6;       // mittlere Priorität

   hCLOSEvalve.Control = 2;
   //Task3 unterbrechen
   hCLOSEvalve.IntNum = 1; //fortlaufende Nr.
   hCLOSEvalve.pSrcVar = numof ( Timer_2 ) ;    //Timer 2
   hCLOSEvalve.CompVar = 200;  //nach 200ms ausschalten
   hCLOSEvalve.Mode = 0;            //Timer-IR zuerst deaktiviert
   hCLOSEvalve.Priority = 7;
```

```
   // Interrupt-Routine und Task3 aktivieren
   _SetTask(ST3_START, fnTask3);
   _SetVarInterrupt( hOPENvalve,fnOPENvalve );
   /*-------------------------------------
   Hauptprogramm-Schleife
   -------------------------------------*/
   while(1)
   { //zyklischer Programmteil ...
   } //Ende while (1)
} //Ende main()
```

Beispiel gelöst im Assembler

Notwendige Parametereinstellungen:

P620 = IPOS-Ausgang, P960 =z. B. KURZ



474256139

6 Wegerfassung und Positionierung

6.1 Geberauswertung

Das MOVIDRIVE® stellt verschiedene Möglichkeiten der Positionierung zur Verfügung:

- externer Geber
- Motorgeber (Inkrementalgeber/Resolver)
- Hiperface®-Geber (Absolutwertgeber)
- SSI-Absolutwertgeber

Zur Verarbeitung werden die Werte in Systemvariablen zur Verfügung gestellt.

Die Anschlüsse für Motorgeber (X15) und externer Geber (X14) befinden sich auf der Steuerelektronik Mx**V**..., Mx**S**... und MC**H**... . Die Steuerelektronik Mx**F**... besitzt diese Anschlüsse nicht. Der Anschluss für den SSI-Absolutwertgeber befindet sich auf der Optionskarte DIP11 (X62).

Alle angeschlossenen Geber werden unabhängig von der Betriebsart (P700) immer ausgewertet. Betriebsarten mit Positionierbetrieb (VFC-n-Reg. & IPOS, CFC & IPOS, SERVO & IPOS) erfordern immer einen Motorgeber an X15.

Gebertyp		**Absolutwertgeber an DIP11** **P941: Absolutwertgeber (DIP)**	**Hiperface®-Geber / Inkrementalgebernachbildung / Inkrementalgeber** **P941: Externer Geber (X14)**	**Inkrementalgeber / Resolver / Hiperface®-Geber** **P941: Motorgeber (X15)**
Anschluss		X62 / DIP11	X14 / Grundgerät	X15 / Grundgerät
Istwert auf Variable		H509 / ACTPOS. ABS / ActPos_Abs	H510 / ACTPOS. EXT / ActPos_Ext	H511 / ACTPOS. MOT / ActPos_Mot
Auflösung		Absolutposition **nach** Umrechnung mit: Geberskalierung (P955), Nullpunktoffset (P954), Positionsoffset (P953), Zählrichtung (P951)	**Tatsächliche** Geberstrichzahl (mit 4fach-Auswertung) **nach** Umrechnung mit: Geberskalierung Ext. Geber (P944)	**immer** 4096 Inc./Motorumdrehung, unabhängig von der tatsächlichen Geberauflösung
Touch-Probe	**Flanke an DI02**	H503 / TP. POS1ABS / TpPos1_Abs	H506 / TP. POS1EXT / TpPos1_Ext	H507 / TP. POS1MOT / TpPos1_Mot
	Flanke an DI03	H502 / TP. POS2ABS / TpPos2_Abs	H504 / TP. POS2EXT / TpPos2_Ext	H505 / TP. POS2MOT / TpPos2_Mot
	max. Verzögerungszeit	1 ms	< 100 µs	< 100 µs

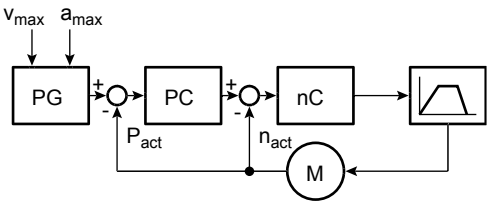
Die Positionswerte stehen der IPOS$^{plus®}$-Steuerung in den Variablen H509 bis H511 immer zur Verfügung. Auch ohne eine Positionierung mit IPOS$^{plus®}$ können an X14 und X15 angeschlossene Impulsgeber erfasst und im IPOS$^{plus®}$-Programm weiter verarbeitet werden. Über die Systemvariablen H570 ... H573 können für Motorgeber X15 und externer Geber X14 Nullimpulszähler aktiviert werden, um die Anzahl aufgetretener Nullimpulse zu erfassen. Zur Positionierung mit den IPOS$^{plus®}$-Befehlen (GO...) ist immer ein Motorgeber notwendig. Der Motorgeber stellt dem MOVIDRIVE® das Drehzahlsignal in hoher Güte bereit.

6.2 Motorgeber (X15)

Die Nutzung der IPOS$^{plus®}$-Positionierung mit Motorgeber setzt voraus, dass der Motorgeber an X15 angeschlossen ist und in P700 Betriebsart 1 eine Betriebsart "... & IPOS" eingestellt ist. P941 Quelle Ist-Position bestimmt, welche Wegmessung zur Positionierung herangezogen wird. Die Verfahrbefehle der IPOS$^{plus®}$-Steuerung (GO-Befehle) beziehen sich auf die Positionsinformation des in P941 Quelle Ist-Position angegebenen Gebers.

6.3 Geberkombinationen

Direkte Lageregelung mit Motorgeber

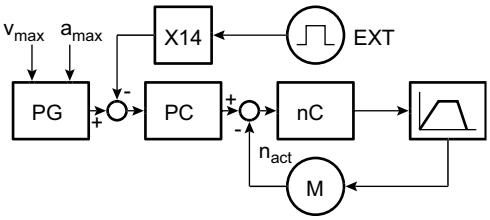


474539019

- Am Motor ist ein Inkrementalgeber / Resolver / Hiperface®-Geber (X15) notwendig.
- In IPOS$^{plus®}$ werden Positionierbefehle z.B. "GOA ..." mit Bezug auf die Quelle Ist-Position (hier: Motorgeber X15) ausgeführt.

v_{max} = maximale Geschwindigkeit
a_{max} = maximale Beschleunigung
PG = Profilgenerator
P_{act} = Ist-Position Motorgeber
PC = Lageregler
n_{act} = Ist-Drehzahl
nC = Drehzahlregler

Direkte Lageregelung mit externem Geber und Motorgeber

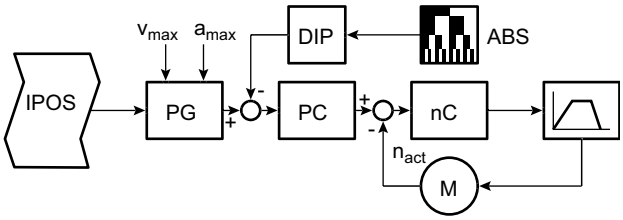


474540555

- Am Motor ist auf jeden Fall ein Inkrementalgeber / Resolver / Hiperface®-Geber (X15) zur Drehzahlrückführung notwendig.
- Es erfolgt automatisch ein Ausgleich eines Schlupfs oder mechanischen Loses (Getriebespiel) zwischen Motorgeber und dem externen Geber.
- In IPOS$^{plus®}$ werden Positionierbefehle z.B. "GOA ..." mit Bezug auf P941 Quelle Ist-Position (hier: externer Geber X14) ausgeführt.
- Die erzielbare Regeldynamik hängt von den Eigenschaften und der mechanischen Anbringung des externen Gebers sowie der Wegauflösung ab.
- siehe Kapitel "IPOS$^{plus®}$ mit Optionen" / "Externer Geber"

v_{max} = maximale Geschwindigkeit
a_{max} = maximale Beschleunigung
PG = Profilgenerator
PC = Lageregler
n_{act} = Ist-Drehzahl
nC = Drehzahlregler
EXT = externer Geber

Direkte Lageregelung mit Absolutwertgeber und Motorgeber

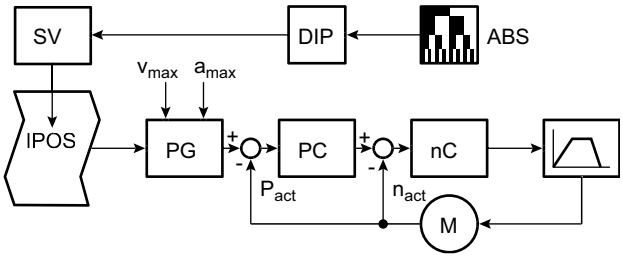


474542091

- Es erfolgt in IPOS$^{plus®}$ eine direkte Lageregelung mit dem über DIP11 angeschlossenen SSI-Absolutwertgeber.
- Am Motor ist auf jeden Fall ein Inkrementalgeber / Resolver / Hiperface® (X15) zur Drehzahlrückführung notwendig.
- Es erfolgt automatisch ein Ausgleich eines Schlupfs oder mechanischen Loses (Getriebespiel) zwischen Inkrementalgeber / Resolver / Hiperface® des Motors und dem Absolutwertgeber.
- In IPOS$^{plus®}$ werden Positionierbefehle z.B. "GOA ..." mit Bezug auf die Quelle Ist-Position (hier: Absolutwertgeber DIP) ausgeführt.
- Die erzielbare Regeldynamik hängt von den Eigenschaften und der mechanischen Anbringung des Absolutwertgebers sowie der Wegauflösung ab.
- siehe Handbuch "Positionieren mit Absolutwertgeber DIP11A"

v_{max} = maximale Geschwindigkeit
a_{max} = maximale Beschleunigung
PG = Profilgenerator
PC = Lageregler
n_{act} = Ist-Drehzahl
nC = Drehzahlregler
ABS = Absolutwertgeber
IPOS = IPOS$^{plus®}$-Programm

Lageregelung mit inkrementellem Geber am Motor,
Verarbeitung der Absolutwertgeberposition im IPOS$^{plus®}$-Programm

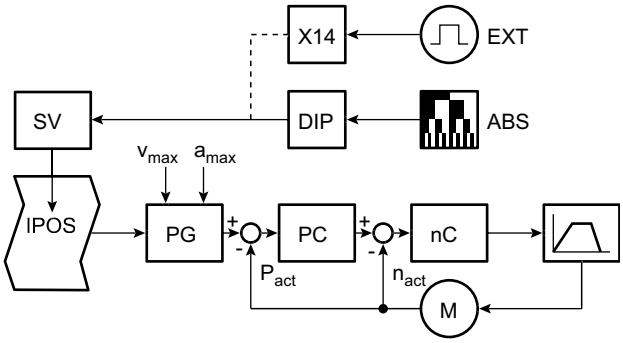


474543627

- Es erfolgt in IPOS$^{plus®}$ eine Lageregelung mit dem an X15 angeschlossenen Motorgeber.
- Am Motor ist auf jeden Fall ein Inkrementalgeber / Resolver zur Drehzahlrückführung notwendig.
- Die hohe Regeldynamik des Umrichters kann direkt für die Positionierung genutzt werden.
- Die Lageinformation des Absolutwertgebers wird automatisch auf einer IPOS$^{plus®}$-Variablen abgebildet und kann programmgesteuert verarbeitet werden.
- Diese Verwendung der DIP11 dient der Vermeidung der Referenzfahrt.
- siehe Handbuch "Positionieren mit Absolutwertgeber DIP11"

v_{max} = maximale Geschwindigkeit
a_{max} = maximale Beschleunigung
PG = Profilgenerator
P_{act} = Ist-Position Motorgeber
PC = Lageregler
n_{act} = Ist-Drehzahl
nC = Drehzahlregler
ABS = Absolutwertgeber
SV = Systemvariable
IPOS = IPOS$^{plus®}$-Programm

Lageregelung mit Motorgeber,
Verarbeitung des zweiten Gebers im IPOS$^{plus®}$-Programm als Leitgeber

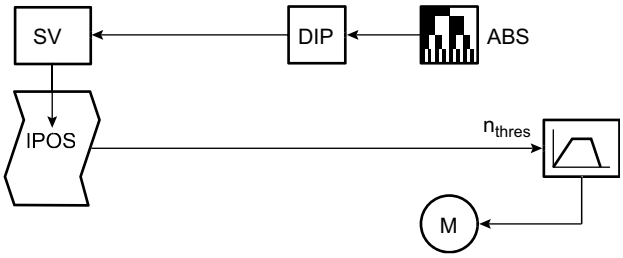


474583563

- Es erfolgt in IPOS$^{plus®}$ eine Lageregelung mit dem an X15 angeschlossenen Motorgeber.
- Am Motor ist auf jeden Fall ein Geber zur Drehzahlrückführung notwendig.
- Die hohe Regeldynamik des Umrichters kann direkt für die Positionierung genutzt werden.
- Die Lageinformation des zweiten Gebers wird automatisch auf einer IPOS$^{plus®}$-Variablen abgebildet und kann programmgesteuert verarbeitet werden.
- Dieser Aufbau wird dort verwendet, wo der Umrichter in Abhängigkeit eines zweiten Gebers verfahren soll (z. B. winkelsynchron oder Kurvenscheibe)

v_{max} = maximale Geschwindigkeit
a_{max} = maximale Beschleunigung
PG = Profilgenerator
P_{act} = Ist-Position Motorgeber
PC = Lageregler
n_{act} = Ist-Drehzahl
nC = Drehzahlregler
ABS = Absolutwertgeber
EXT = externer Geber
SV = Systemvariable
IPOS = IPOS$^{plus®}$-Programm

Verarbeitung der Absolutwertgeberposition im IPOS$^{plus®}$-Programm



474585099

- Die Lageinformation des Absolutwertgebers wird automatisch auf einer IPOS$^{plus®}$-Variablen abgebildet und kann programmgesteuert verarbeitet werden.
- Diese Verwendung der DIP11 oder eines HIPERFACE®-Gebers an X14 kann insbesondere zum Ersatz von Applikationen verwendet werden, bei denen sonst über mehrere Initiatoren mit Eilgang-Schleichgang positioniert wird.
- Am Motor ist kein Inkrementalgeber / Resolver zur Drehzahlrückführung notwendig, es kann ein Standard-Asynchronmotor verwendet werden. Es ist keine Betriebsart "... & IPOS" nötig.
- Siehe Handbuch "Positionieren mit Absolutwertgeber DIP11". Zur Realisierung einer solchen Positionieraufgabe steht in der SHELL das Applikationsmodul "Absolutwertpositionierung" zur Verfügung.

ABS = Absolutwertgeber
SV = Systemvariable
IPOS = IPOS$^{plus®}$-Programm
n_{thres} = Solldrehzahl

6.4 *Externer Geber (X14)*

An Antriebsumrichter MOVIDRIVE® A und B können an X14 folgende Geber angeschlossen werden. Welchen Gebertyp ihr Gerät unterstützt, entnehmen Sie bitte dem Systemhandbuch oder der Betriebsanleitung.

- Hiperface®-Geber Typ AS1H, ES1H oder AV1H (nur MOVIDRIVE® B oder MCH)
- sin/cos-Geber Typ ES1S, ES2S oder EV1S (nur MOVIDRIVE® B oder MCH)
- 5 V-TTL-Geber mit DC 24-V-Spannungsversorgung Typ ES1R, ES2R oder EV1R
- 5 V-TTL-Geber mit DC 5-V-Spannungsversorgung Typ ES1T, ES2T oder EV1T über Option DWI11

6.4.1 Positionierung auf externen Geber (X14)

Die Positionierung auf externen Geber ist sinnvoll, um eine schlupf- oder spielbehaftete Verbindung zwischen Antrieb und Wegstrecke (z. B. durch rutschende Räder oder Zahnflankenspiel) oder mechanische Lose im Getriebespiel zu kompensieren.

Tabelle 1: Bewertung der Impulse des externen Gebers

ankommende Impulse (Beispiel)	**2048**	**2048**	**1024**	**1024**
Vierfachauswertung (fest)	8192	8192	4096	4096
Skalierung externer Geber P944 (einstellbar)	x 1	x 8	x 1	x 2
Veränderung Zählerstand H510 ACTPOS. EXT pro Umdrehung des Gebers	**8192**	**65536**	**4096**	**8192**

Nach Einschalten der Steuerspannung wird bei Einsatz eines AS1H oder AV1H der absolute Wert dieses Gebers in den Positions-Istwert des externen Gebers H510 übernommen, bei allen anderen Gebertypen ist H510 = 0 Inkremente. Der externe Geber kann wie der Motorgeber referenziert werden (siehe Kapitel Referenzfahrt).

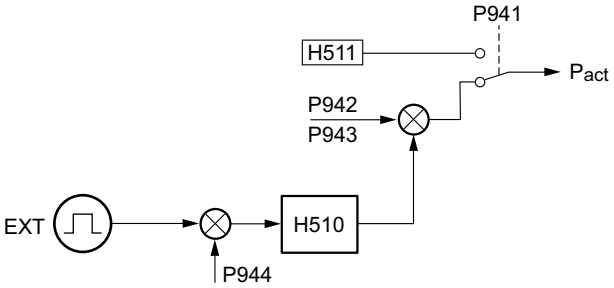
6.4.2 Einsatzfall Schlupfausgleich mit externem Geber

Ein Fahrwagen auf Rädern wird in einer Schiene geführt. Der Vorschub erfolgt durch Antreiben der Räder über einen Getriebemotor. Die Verbindung zwischen den Rädern und der Schiene ist nicht formschlüssig. Daraus folgt ein Schlupf, eine Verschiebung, zwischen der rotatorischen Bewegung des Rades und der translatorischen Bewegung des Fahrwagens.

Für eine Positionierung über die Steuerung des Motors ist es deshalb zwingend notwendig, die Position des Fahrwagens zu erfassen.

Dazu ist bei der Inbetriebnahme das Übersetzungsverhältnis der Inkremente des Motorgebers zu den Inkrementen des Streckengebers einzustellen. Die Beschreibung dieser Parameter, die Vorgehensweise zur Einstellung und Beispiele können Sie der Beschreibung der Parameter P944, P943 und P942 im Kapitel IPOS-Parameter entnehmen.

Es gilt folgendes Blockschaltbild:



476710667

P_x = Positions-Rohwert des Gebers
P_{act} = Positions-Istwert für Rampengenerator und Lageregler
P941 = Quelle Ist-Position
P942 = Geberfaktor Zähler
P943 = Geberfaktor Nenner
P944 = Geberskalierung externer Geber
H510 = Externer Geber mit Istwert auf Variable
H511 = Motorgeber mit Istwert auf Variable

Folgende Parameter sind für den externen Geber einzustellen:

Tabelle 2: Parametereinstellung für Fahrwagen

Nummer	Bezeichnung	Funktion	Einstellung	Bereich
P944	Geberskalierung ext. Geber	Multiplikation der Gebersignale mit eingestelltem Wert	Größter Wert, der noch kleiner ist als das Verhältnis der Auflösung von Motorgeber zu ext. Geber. Beispiel: Motorgeber: 4096 Ink. / ext. Geber 800 Inc. = 5.12. Wert: 4.	fest definiert: 1, 2, 4, 8, 16, 32, 64
P943	Geberfaktor Nenner	Nenner zur Bestimmung des Verhältnisses von Motorgeber zu ext. Geber	Anzahl der Zählinkremente (in H511 ACTPOS. MOT zu lesen), für einen bestimmten Weg s.	max. 32767
P942	Geberfaktor Zähler	Zähler zur Bestimmung des Verhältnisses von Motorgeber zu ext. Geber	Anzahl der Zählinkremente (in H510 ACTPOS. EXT zu lesen) für einen bestimmten, gleichen Weg s wie für P943.	max. 32767
P941	Quelle Ist-Position	Positions-Istwert für IPOS$^{plus®}$-Lageregler	Ext. Geber X14.	(Auswahl)
P945	Streckengeber Typ (X14)	Auswahl des Gebertyps	Entsprechend dem angeschlossenen Geber.	TTL SIN/COS HIPER-FACE
P946	Streckengeber Zählrichtung (X14)	Invertierung der Drehrichtung des Gebers	So, dass die Zählrichtung Motorgeber = Zählrichtung externer Geber ist.	NORMAL INVER-TIERT

	HINWEIS Die Berechnung von P210 (P-Verstärkung Halteregler) bei der Inbetriebnahme ist für P941 = Motorgeber optimiert. Bei Verwendung von externem Geber oder Absolutwertgeber muss der Wert evtl. niedriger eingestellt werden.

Für die Wegerfassung mit externem Geber an X14 gilt:

- Variable H510 zeigt die Ist-Position der Lageregelung ACTPOS. EXT
- Variable H506 zeigt die Touch-Probe Position 1 TP. POS1EXT
- Variable H504 zeigt die Touch-Probe Position 2 TP. POS2EXT

Die Variablen sind immer mit dem Parameter P944 bewertet.

6.5 SSI-Absolutwertgeber (DIP)

6.5.1 Inbetriebnahme

Der Antrieb muss in Verbindung mit dem MOVIDRIVE®-Antriebsumrichter wie im Systemhandbuch MOVIDRIVE® beschrieben in Betrieb genommen sein. Über eine geeignete Sollwert- und Steuerquelle muss es möglich sein, den Antrieb zu verfahren.

Weiterhin ist sicherzustellen, dass folgende Installationen korrekt und der Anwendung entsprechend ausgeführt sind:

- die Installation der DIP11A / DIP11B
- die Verdrahtung
- die Klemmenbelegung
- die Sicherheitsabschaltungen

Beziehen Sie sich dazu und für das weitere Vorgehen auf das Handbuch MOVIDRIVE ® MDX 61B Absolutwertgeberkarten DIP11B/DEH21B"

Die Ausführung der Werkseinstellung ist nicht notwendig. Wird eine Werkseinstellung aufgerufen, so werden die Parameter des MOVIDRIVE® auf eine Grundeinstellung zurückgestellt. Davon ist auch die Klemmenbelegung betroffen und muss ggf. auf die gewünschten Einstellungen gebracht werden.

	HINWEIS Mit MOVITOOLS® MotionStudio ist eine geführte Inbetriebnahme der Absolutwertgeberoption möglich. Sie werden dort über Dialoge zu den notwendigen Eingaben und Aktionen aufgefordert. • Starten Sie dazu MOVITOOLS® MotionStudio und markieren Sie das Gerät. • Öffnen sie das Kontextmenü und wählen Sie den Menüpunkt [Inbetriebnahme] / [DIP-Inbetriebnahme]. • Folgen sie den Anweisungen des Inbetriebnahme-Assistenten. Nach der DIP-Inbetriebnahme mit MOVITOOLS® MotionStudio muss nur noch der Parameter *"Quelle Ist-Position"* angegeben werden.

Nach der DIP-Inbetriebnahme werden bei der Referenzfahrt sowohl Motor- als auch DIP-Geber referenziert, - unabhängig von der Einstellung des Parameters *"Quelle Ist-position" P941*.

Falls der DIP-Geber von Hand eingestellt wurde, führt eine Referenzfahrt nur zur Referenzierung des Motorgebers, auch unabhängig von der Einstellung des P941

Alternativ kann die Inbetriebnahme der Absolutwertgeberoption DIP11 schrittweise wie nachfolgend beschrieben durchgeführt werden.

6.5.2 1. Gebertyp P950 wählen

Wählen Sie mit P950 Gebertyp den benutzten Geber aus. Die unterstützen Gebersysteme entnehmen Sie der Beschreibung von P950.

Der angeschlossene Typ wird aus der Liste der möglichen Geber ausgewählt. Andere Geber müssen auf ihre Einsetzbarkeit geprüft und von SEW-EURODRIVE freigegeben werden.

6.5.3 2. Motordrehsinn P35_ einstellen

Verfahren Sie den Antrieb mit geringer Drehzahl in positive Bewegungsrichtung (Definition laut Anwendung). Zählt die Ist-Position P003 oder H511 aufwärts, dann kann der Parameter P350 "Drehrichtungsumkehr" unverändert bleiben (Anzeige der Ist-Position mit MOVITOOLS® MotionStudio oder DBG11B). Zählt die Ist-Position abwärts, so muss P350 umgestellt werden. Somit ist die Zählrichtung des Motorgebers der Anwendung angepasst.

6.5.4 3. Zählrichtung P951 des SSI-Absolutwertgebers einstellen

Verfahren Sie den Antrieb mit geringer Drehzahl in positive Bewegungsrichtung (Definition laut Anwendung). Zählt die Absolutwertgeberposition (H509 ACTPOS. ABS) aufwärts, dann kann der Parameter P951 "Zählrichtung" unverändert bleiben. Zählt die Absolutwertgeberposition abwärts, so muss P951 invertiert werden.

6.5.5 4. Geberskalierung P955 einstellen

Ist kein Motorgeber vorhanden (keine Drehzahlregelung), so ist der Parameter nicht relevant. Mit dem eingestellten Wert wird die Positionsinformation des Absolutwertgebers multipliziert. Der Parameter wird so eingestellt, dass das Verhältnis der Weginformationen zwischen dem Motorgeber und dem Absolutwertgeber möglichst nahe "1" ist.

Zur Messung stellen Sie den Parameter zunächst auf 1. Notieren Sie sich die Werte der Variablen H509 (ACTPOS. ABS) und H511 (ACTPOS. MOT). Verfahren Sie den Antrieb um mindestens 1 Motorumdrehung. Ermitteln Sie die Differenz zwischen den notierten und den aktuellen Werten der Variablen und bilden den Quotienten.

ACTPOS. ABS	H509 alt	H509 neu	H509 Differenz
notierte Werte			
ACTPOS. MOT	H511 alt	H511 neu	H511 Differenz
notierte Werte			

Der Quotient Q ergibt sich aus H511 Differenz geteilt durch H509 Differenz.

Q = (H511 alt - H511 neu) / (H509 alt - H509 neu)

Den Parameter GEBERSKALIERUNG (P955) stellen Sie auf den Wert ein, der dem ermittelten Quotienten Q am nächsten kommt, vorzugsweise den kleineren Wert.

Ist der Quotient größer als 80, so wird die Positionierung auf den Absolutwertgeber nur mit reduzierter Dynamik realisiert werden können.

HINWEIS

Bei der Projektierung ist darauf zu achten, dass das Geberverhältnis 1:10 nicht überschreitet.

6.5.6 5. Positionsoffset P953 einstellen

Der Positionsoffset (P953) muss nur bei Drehgebern eingestellt werden. Bei anderen Gebern sollte er auf 0 gestellt werden.

Gehen Sie weiter vor wie im Kapitel IPOS-Parameter bei P953 beschrieben.

6.5.7 6. Nullpunktoffset P954 einstellen

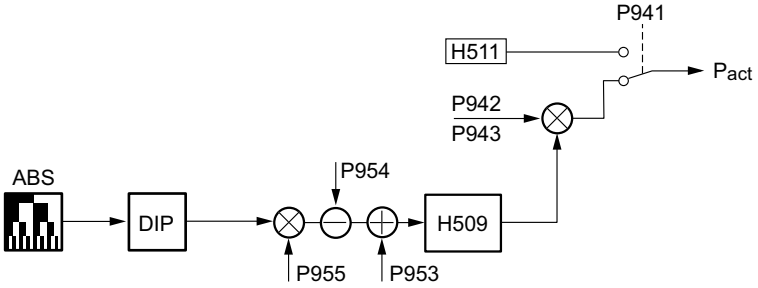
Mit dem Nullpunktoffset wird einer bestimmten Position ein gewünschter Wert zugeordnet. Stellen Sie den Parameter ein wie im Kapitel IPOS-Parameter bei P954 beschrieben.

6.5.8 7. Geberfaktoren P942 und P943 einstellen

Die Parameter werden zur internen Anpassung der Drehzahlregelung und für Überwachungsfunktionen in der DIP11 herangezogen.

Praktisch wird damit die physikalische Stückzahl, ein mechanisches Übersetzungsverhältnis zwischen Motorgeber und externem Geber und die mechanische Vorschubkonstante (z. B. bei externem inkrementellen Lineargeber) angepasst.

Die folgende Grafik zeigt den Zusammenhang zwischen den Parametern und Variablen.



476717707

P_x = Positions-Rohwert des Gebers
P_{abs}= Positions-Istwert für Rampengenerator und Lageregler
P941 = Quelle Ist-Position
P942 = Geberfaktor Zähler
P943 = Geberfaktor Nenner
P953 = Positionsoffset
P954 = Nullpunktoffset
P955 = Geberskalierung
H509 = Externer Geber mit Istwert auf Variable
H511 = Motorgeber mit Istwert auf Variable

Wie Sie die Geberfaktoren praktisch ermitteln, finden Sie bei der Parameterbeschreibung von P942 / P943.

6.5.9 8. Quelle Ist-Position P941 einstellen

Der Parameter bestimmt, welcher Positionsgeber zur Lageregelung benutzt wird, sofern eine Betriebsart "... & IPOS" in Parameter P700 "Betriebsart" eingestellt ist.

IPOS$^{plus®}$ verfügt über Positionierbefehle zum Steuern des am MOVIDRIVE® angeschlossenen Motors. Soll die Positionierung des Motors auf den Absolutwertgeber erfolgen, ist die "Quelle Ist-Position" auf "Absolutgeb. DIP" zu stellen.

i	**HINWEIS** Die Kreisverstärkung für die Lageregelung der IPOS$^{plus®}$, Parameter P910 "Verstärkung X-Regler", wurde bei der Inbetriebnahme des Drehzahl-Regelkreis voreingestellt. Die Voreinstellung setzt die Lageregelung auf den Motorgeber voraus. Der Unterschied der Geberauflösung oder das Zeitverhalten des Absolutwertgebers (z. B. Laserentfernungsmesser) können einen geringeren Einstellwert erfordern.

Stellen Sie maximal den halben Wert der berechneten Voreinstellung ein. Wenn P955 ≥ 32 eingestellt ist, sogar nur ein Viertel der berechneten Voreinstellung. Starten Sie ein IPOS$^{plus®}$-Programm mit einer Positionierung zwischen zwei gültigen Positionen mit mäßiger Geschwindigkeit. Verringern oder erhöhen Sie den Parameter P910 "Verstärkung X-Regler" schrittweise, bis das beste Fahr- und Positionierverhalten eingestellt ist. Bei großen Werten von P955 kann es notwendig sein, dass P910 Werte < 1 erhält.

Der vom Absolutwertgeber gelieferte Positionswert steht in der Variablen H509 (ACT-POS. ABS) zur Verfügung. Der Positionswert kann mit der internen Steuerung IPOS$^{plus®}$ auch ohne direkte Positionierung verarbeitet werden.

6.6 Referenzieren

Für Anwendungen mit absoluten Positionierbefehlen ist es notwendig den Bezugspunkt (Maschinen-Nullpunkt) zu definieren. Abhängig vom Gebertyp ist dies einmalig bei der Erstinbetriebnahme (Absolutgeber) oder jedes Mal, wenn die Maschine neu eingeschaltet wird, notwendig (alle anderen Geber).

Das MOVIDRIVE® unterstützt 9 verschiedene Referenzfahrtypen, die über P903 Referenzfahrtyp, P904 Referenzierung auf Nullimpuls und die Argumente des IPOS$^{plus®}$-Befehls `_Go0 ( ... )`; bzw `GO0 ...` eingestellt werden und sich aus der sinnvollen Kombination folgender Eigenschaften ergeben:

- Setzen des Gebers ohne Referenzfahrt (keine Referenzfahrt)
- Suchrichtung (= Bewegungsrichtung bei Start der Referenzfahrt)
- Referenzieren auf Hardware-Endschalter
- Referenzieren auf Nullimpuls (bei externen Gebern nur möglich, wenn Auflösung < 5000 Ink./Umdrehung)
- Referenzieren auf Referenznocke

Wird auf Hardware-Endschalter und / oder Referenznocke referenziert sind diese als Binäreingänge einzustellen.

Über den Parameter P941 Quelle Ist-Position wird definiert, auf welchen Geber sich die Referenzfahrt bezieht.

Die Referenzfahrt wird bei freigegebenem Antrieb mit einer der beiden folgenden Methoden gestartet:

- über eine positive Flanke am Binäreingang P600 ... P606 oder P610 ... P617, der auf die Funktion REF.-FAHRT START eingestellt wurde
- über den IPOS$^{plus®}$-Befehl `_Go0 ( ... )`; bzw. `GO0 ...`

Die Anzeige wechselt auf "c" - REFERENZBETRIEB.

Während der Referenzfahrt wird immer mit der Stopprampe P136 beschleunigt. Bei Referenzfahrtypen mit Referenznocke oder Endschalter wird zunächst auf die Geschwindigkeit P901 Referenzdrehzahl 1 beschleunigt und danach die Bedingung für das Ende der Referenzfahrt mit P902 Referenzdrehzahl 2 gesucht. Bei Referenztyp 0 wird sofort P902 Referenzdrehzahl 2 verwendet.

Wird während der Referenzfahrt mit Typ 1 oder Typ 2 ein Hardware-Endschalter angefahren und wurde der Referenzpunkt noch nicht gefunden, wendet der Antrieb und setzt die Referenzfahrt in die andere Richtung fort.

Hat der Antrieb den Referenzpunkt erreicht, so werden die folgenden Funktionen ausgeführt:

- Der Antrieb bleibt stehen und schaltet intern von Drehzahl- in Lageregelung um.
- Das Bit 20 "IPOS-Referenziert" in H473 StatusWord wird gesetzt und ein auf "IPOS-REFERENZ" parametrierter Binärausgang wird gesetzt.
- Der Referenz-Offset P900 wird in den Lage-Istwert übernommen. Bei abgeschalteter Modulo-Funktion (P960 = AUS) ist dies die Variable des mit P941 ausgewählten Gebers H509 - H511, bei aktivierter Modulo-Funktion die Variable H455 ModActPos.

Ab diesem Zeitpunkt gilt für den Maschinen-Nullpunkt die Gleichung:

Maschinen-Nullpunkt = Referenzpunkt + Referenz-Offset

Der Status "referenziert" wird zurückgesetzt, wenn der Umrichter ausgeschaltet wird oder bei Fehlermeldungen, die das Wegmess-System betreffen (Ausnahme: bei Hiperface®-Gebern siehe Hinweis unten).

Für fortgeschrittene Anwender bietet das MOVIDRIVE® auch die Möglichkeit Absolutgeber zu setzen, in dem am Referenzpunkt ein neuer Geberoffset berechnet und beschrieben wird, ohne dass der Antrieb freigegeben ist und während er sich z. B. im Zustand Sicherer Halt befindet. Durch geschickte IPOS$^{plus®}$-Programmierung ist es außerdem möglich abstandskodierte Gebersysteme auszuwerten. Für Unterstützung in diesen beiden Fälle wenden Sie sich bitte an das SEW-Servicepersonal.

	HINWEIS Bei Hiperface®-Absolutwertgebern und SSI-Absolutwertgebern ist der Status "referenziert" immer gesetzt und wird nur während einer Referenzfahrt zurückgesetzt.

	HINWEIS Je nachdem wie die Referenzfahrt gestartet wurde, ergibt sich ein unterschiedliches Verhalten, wenn bei laufender Referenzfahrt die Reglersperre gesetzt ist. • Wurde die Referenzfahrt über eine positive Flanke am Binäreingang REF.-FAHRT START gestartet oder über den nicht-wartenden IPOS-Befehl `_Go0( ._NW_.. )`; bzw `GO0...,NW,...` gestartet, wird die Referenzfahrt nicht fortgesetzt, wenn die Reglersperre zurückgenommen wird. Zum erneuten Start ist eine positive Flanke an REF.-FAHRT START nötig. • Wurde die Referenzfahrt über den wartenden IPOS-Befehl `_Go0( ._W_.. )`; bzw `GO0...,W,...` gestartet, wird die Fehlermeldung F39 Referenzfahrt generiert.

Bei der Entscheidung, ob auf Referenznocke oder Nullimpuls referenziert werden soll, sind folgende Punkte zu beachten:

- Der Nullimpuls verschiebt sich, wenn der Motor getauscht wird.
- Der Referenznocken könnte durch Alterung und Verschleiß oder Schalthysterese ungenau werden.
- Wenn der Referenzpunkt mit Nullimpuls und Referenznocken ermittelt wird und der Nullimpuls genau am Ende des Referenznockens liegt, so kann die Schaltflanke des Referenznockens vor oder nach dem Nullimpuls erfasst werden (Schalthysterese). Daraus kann sich ein Referenzpunkt ergeben, der von einem zum anderen Mal um eine Motorumdrehung variiert. Abhilfe wird durch Verschieben des Referenznockens (ca. um die Länge einer halben Motorumdrehung) geschaffen.
- Endlosantriebe in eine Richtung können nur mit einer Referenznocke referenziert werden. Zusätzlich ist zu beachten, dass es bei nicht ganzzahligen Übersetzungen keinen festen Abstand zwischen Referenznocken und Nullimpuls des Gebers gibt, so dass hier als Referenzpunkt nur das Ende des Referenznockens gewählt werden sollte.
- Die Referenznockenlänge und die Referenzdrehzahlen müssen so gewählt werden, dass der Antrieb auf dem Referenznocken sicher auf die niedrige Referenzdrehzahl (Referenzdrehzahl 2) herunterfahren kann. Das Ende des Referenznockens oder der nächstliegende Nullimpuls des Gebersystems können als Referenzpunkt benutzt werden.
- Auf den Nullimpuls kann nur referenziert werden, wenn der Geber einen Nullimpuls hat, die Nullspur am Umrichter angeschlossen ist und die Geberstrichzahl < 5000 Ink./Umdrehung ist.

	HINWEIS Bei einer Referenzfahrt eines Antriebssystems mit Absolutwertgeber (Hiperface® oder DIP) werden, je nach eingestellter Quelle Ist-Position durch die Referenzfahrt P905 Hiperface® Offset X14 / P947 Hiperface® Offset X15 oder DIP Offset P953 Positions-Offset neu berechnet und überschrieben.

	HINWEIS Wenn Sie P350 / P351 Drehrichtungsumkehr ändern, dann ändern Sie diesen Parameter vor dem Referenz-Offset oder führen Sie nach dem Ändern der Parameter den Referenz-Offset erneut durch, denn diese Änderung wirkt auf die Ist-Position der Achse.

Nachfolgend werden die verschiedenen Referenzfahrttypen mit verschiedenen Startpunkten des Antriebs mit Verfahrdiagrammen erläutert.

Erklärung zu den Referenzfahrttyp-Bildern

- nRef1 =Referenzdrehzahl 1
- nRef2 =Referenzdrehzahl 2
- Startpunkt des Antriebs
  - [1] zwischen Referenznocken und rechtem Hardware-Endschalter
  - [2] auf dem Referenznocken
  - [3] zwischen Referenznocken und linkem Hardware-Endschalter
- LHWLS = Linker Hardware-Endschalter
- RHWLS = Rechter Hardware-Endschalter
- CAM = Referenznocken
- RefCAM = Referenzpunkt Nocken: diese Position wird angefahren, wenn im Argument des Referenzfahrtbefehls GO0 **CAM** steht.
- RefZP = Referenzpunkt Nullimpuls: diese Position wird angefahren, wenn im Argument des Referenzfahrtbefehls GO0 **ZP** (Zero Pulse) steht.
- RefOffCAM = Referenz-Offset bei Referenzfahrt mit Referenzpunkt Nocken CAM
- RefOffZP = Referenz-Offset bei Referenzfahrt mit Nullimpuls ZP
- MZP = Maschinennullpunkt

6.6.1 Typ 0: Referenzfahrt auf Nullimpuls

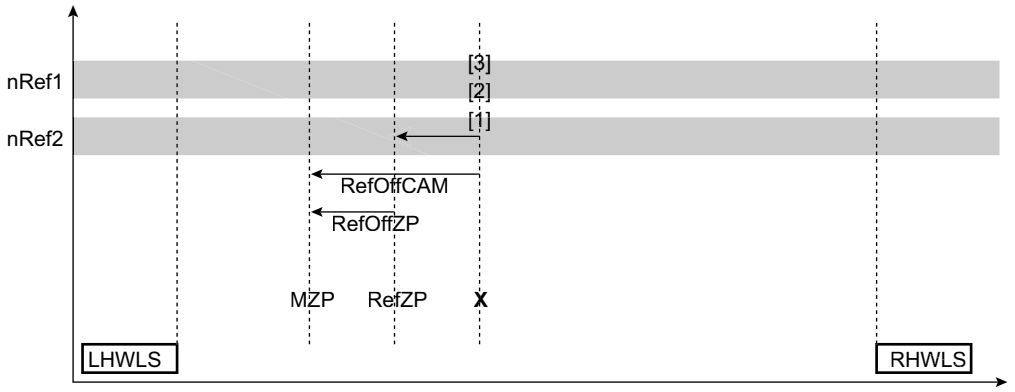
Referenzpunkt ist der erste Nullimpuls links von der Startposition der Referenzfahrt.

Ein Referenznocken ist nicht erforderlich. Für die Referenzfahrt wird ausschließlich P902 Referenzdrehzahl 2 verwendet.

Wird die Referenzfahrt über die positive Flanke am Eingang "REF.-FAHRT START" gestartet, sollte P904 Referenzierung auf Nullimpuls = JA eingestellt sein.

Wird die Referenzfahrt über den IPOS$^{plus®}$-Befehl `Go0` gestartet, sollte das Argument `"ZP"` eingestellt sein, P904 hat keine Bedeutung.

Ist im ersten Fall P904 = NEIN oder wird im zweiten Fall das Argument `"CAM"` verwendet, verhält sich der Antrieb wie bei Typ 5 und setzt die aktuelle Position auf den Referenzpunkt.



476740875

6.6.2 Typ 1: Linkes Ende des Referenznockens

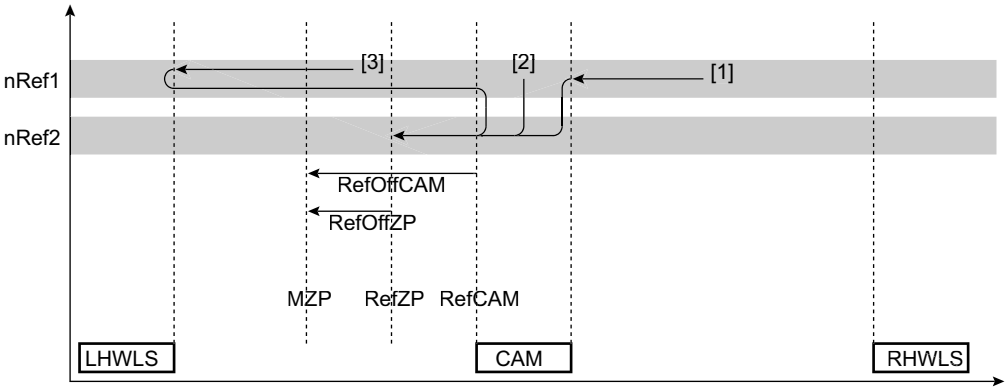
Referenzpunkt ist das linke Ende des Referenznockens oder der erste Nullimpuls links nach dem Ende des Referenznockens.

Einer der Binäreingänge P600 ... 606 muss auf "REFERENZNOCKEN" eingestellt sein.

Die Referenzfahrt startet in Drehrichtung links, bis zur ersten positiven Flanke des Referenznockens wird P901 Referenzdrehzahl 1 verwendet, danach P902 Referenzdrehzahl 2.

Wird die Referenzfahrt über die positive Flanke am Eingang "REF.-FAHRT START" gestartet, wird je nach Einstellung in P904 Referenzierung auf Nullimpuls auf die fallende Flanke des Referenznockens oder auf den Nullimpuls nach der fallenden Flanke des Referenznockens referenziert.

Wird die Referenzfahrt über den IPOS$^{plus®}$-Befehl `Go0` gestartet, wird je nach Argument `"ZP"` oder `"CAM"` auf die fallende Flanke des Referenznockens oder auf den Nullimpuls nach der fallenden Flanke des Referenznockens referenziert.



476742795

6.6.3 Typ 2: Rechtes Ende des Referenznockens

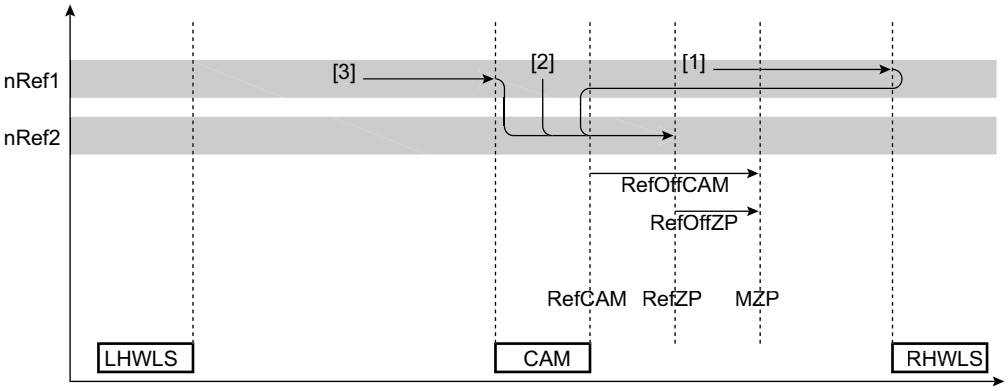
Referenzpunkt ist das rechte Ende des Referenznockens oder der erste Nullimpuls rechts nach dem Ende des Referenznockens.

Einer der Binäreingänge P600 ... 606 muss auf "REFERENZNOCKEN" eingestellt sein.

Die Referenzfahrt startet in Drehrichtung rechts, bis zur ersten positiven Flanke des Referenznockens wird P901 Referenzdrehzahl 1 verwendet, danach P902 Referenzdrehzahl 2.

Wird die Referenzfahrt über die positive Flanke am Eingang "REF.-FAHRT START" gestartet, wird je nach Einstellung in P904 Referenzierung auf Nullimpuls auf die fallende Flanke des Referenznockens oder auf den Nullimpuls nach der fallenden Flanke des Referenznockens referenziert.

Wird die Referenzfahrt über den IPOS$^{plus®}$-Befehl `Go0` gestartet, wird je nach Argument `"ZP"` oder `"CAM"` auf die fallende Flanke des Referenznockens oder auf den Nullimpuls nach der fallenden Flanke des Referenznockens referenziert.



476744331

6.6.4 Typ 3: Endschalter rechts

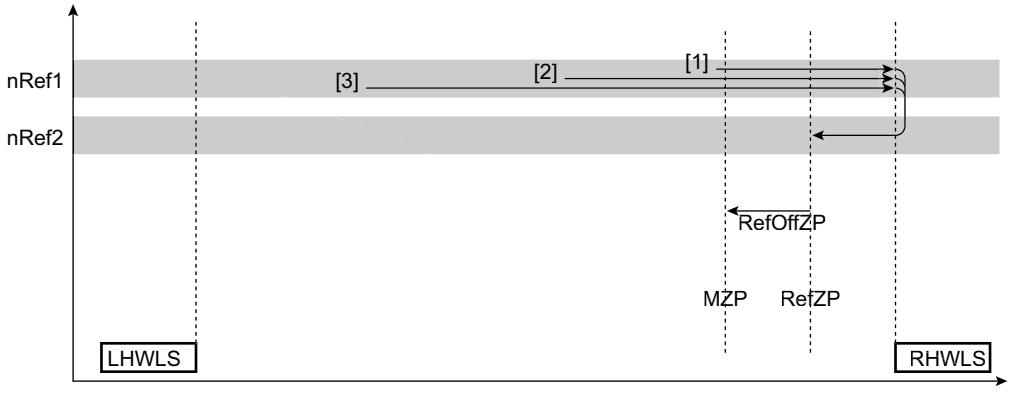
Referenzpunkt ist der erste Nullimpuls links vom rechten Endschalter.

Die Einstellung "linkes Ende des rechten Endschalters" hat keine praktische Bedeutung, da sich der Antrieb nach der Referenzfahrt in der Schalthysterese des Endschalters befinden könnte und nach Ende der Referenzfahrt sporadisch der Fehler "29 Endschalter erreicht" auftreten könnte. Ein Referenznocken ist nicht erforderlich.

Die Referenzfahrt startet in Drehrichtung rechts. Bis zur fallenden Flanke des rechten Endschalters wird P901 Referenzdrehzahl 1 verwendet, danach P902 Referenzdrehzahl 2.

Wird die Referenzfahrt über die positive Flanke am Eingang "REF.-FAHRT START" gestartet, ist P904 Referenzierung auf Nullimpuls = JA einzustellen.

Wird die Referenzfahrt über den IPOS$^{plus®}$-Befehl `Go0` gestartet, ist das Argument `"ZP"` einzustellen.



476758667

6.6.5 Typ 4: Endschalter Links

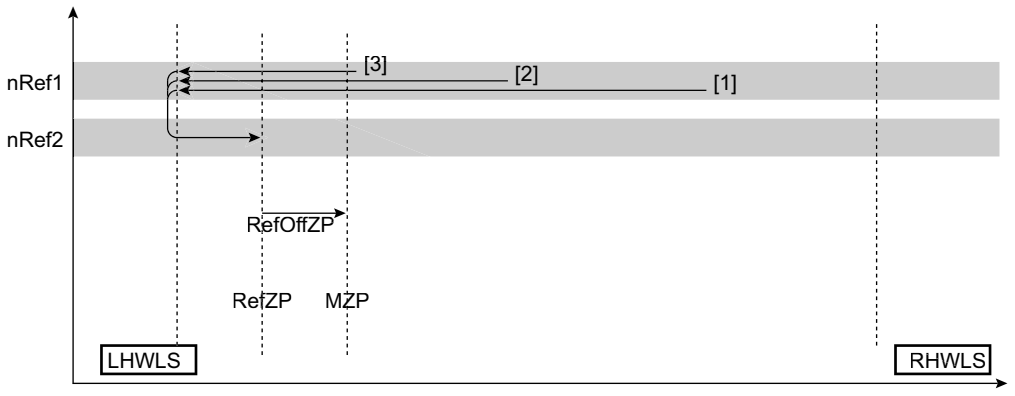
Referenzpunkt ist der erste Nullimpuls rechts vom linken Endschalter.

Die Einstellung "rechtes Ende des linken Endschalters" hat keine praktische Bedeutung, da sich der Antrieb nach der Referenzfahrt in der Schalthysterese des Endschalters befinden könnte und nach Ende der Referenzfahrt sporadisch der Fehler "29 Endschalter erreicht" auftreten könnte. Ein Referenznocken ist nicht erforderlich.

Die Referenzfahrt startet in Drehrichtung links, bis zur fallenden Flanke des linken Endschalters wird P901 Referenzdrehzahl 1 verwendet, danach P902 Referenzdrehzahl 2.

Wird die Referenzfahrt über die positive Flanke am Eingang "REF.-FAHRT START" gestartet, ist P904 Referenzierung auf Nullimpuls = JA einzustellen.

Wird die Referenzfahrt über den IPOS$^{plus®}$-Befehl `Go0` gestartet, ist das Argument `"ZP"` einzustellen.

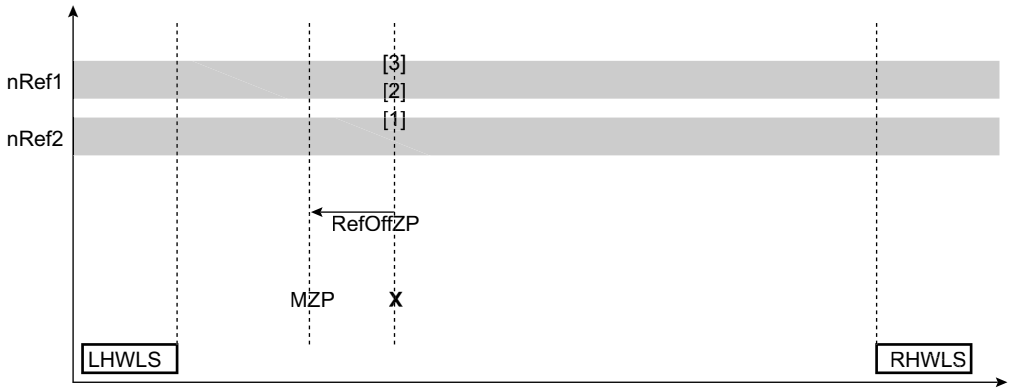


476760203

6.6.6 Typ 5: Keine Referenzfahrt

Referenzpunkt ist die aktuelle Position. Die Argumente im IPOS$^{plus®}$-Befehl `Go0"ZP"` bzw. `"CAM"` und P904 sind wirkungslos.

Dieser Referenzfahrtyp ist sinnvoll bei Absolutgebern und bei Antrieben, die im Stillstand referenziert werden sollen. So kann z. B. die Position einer Vorschubachse während dem Stillstand auf Null gesetzt werden. Damit kann der Maschinenbetreiber erkennen, wo der Antrieb sich innerhalb eines jeden Vorschubs befindet.



476761739

6.6.7 Typ 6: Referenznocken bündig zum rechten Endschalter

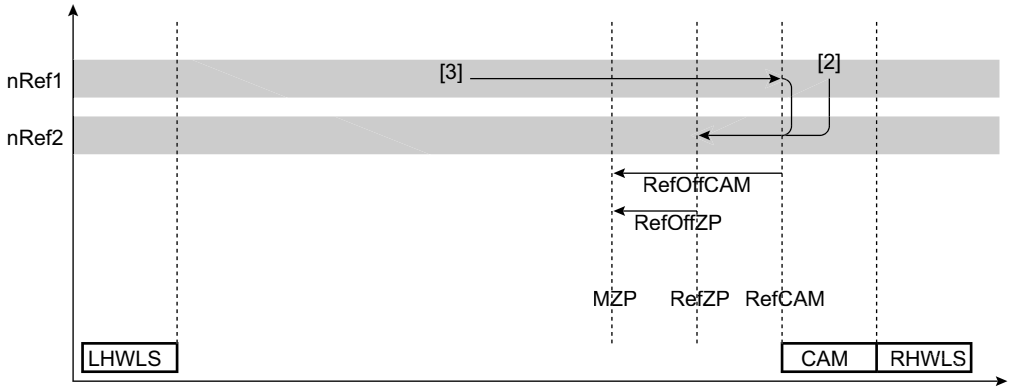
Referenzpunkt ist das linke Ende des Referenznockens oder der erste Nullimpuls links nach dem Ende des Referenznockens.

Einer der Binäreingänge P600 ... 606 muss auf "REFERENZNOCKEN" eingestellt sein. Die Referenzfahrt startet in Drehrichtung rechts, bis zur ersten positiven Flanke des Referenznockens wird P901 Referenzdrehzahl 1 verwendet, danach P902 Referenzdrehzahl 2. Im Unterschied zu Typ 1 startet der Antrieb nach rechts und wendet auf dem Referenznocken.

Wird die Referenzfahrt über die positive Flanke am Eingang "REF.-FAHRT START" gestartet, wird je nach Einstellung in P904 Referenzierung auf Nullimpuls auf die fallende Flanke des Referenznockens oder auf den Nullimpuls nach der fallenden Flanke des Referenznockens referenziert.

Wird die Referenzfahrt über den IPOS$^{plus®}$-Befehl `Go0` gestartet, wird je nach Argument `"ZP"` oder `"CAM"` auf die fallende Flanke des Referenznockens oder auf den Nullimpuls nach der fallenden Flanke des Referenznockens referenziert.

Der Referenznocken muss kurz vor oder genau mit dem rechten Hardware-Endschalter beginnen und muss in den Endschalter hineinragen. Damit ist gewährleistet, dass während der Referenzfahrt kein Hardware-Endschalter angefahren wird.



476763275

6.6.8 Typ 7: Referenznocken bündig zum linken Endschalter

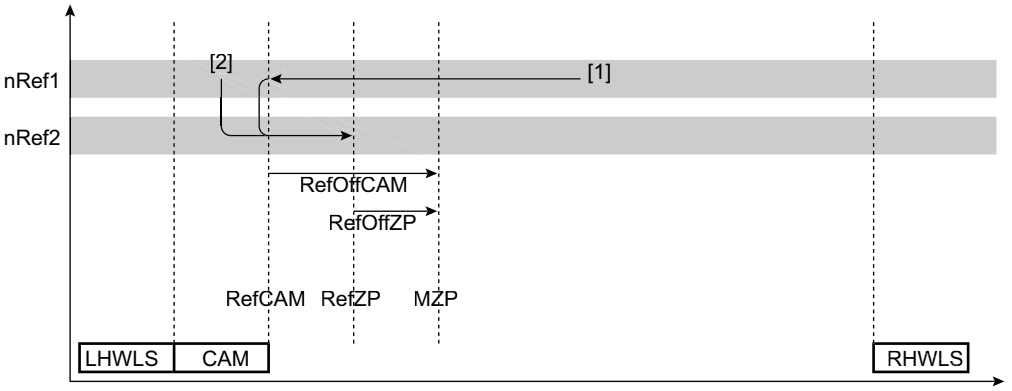
Referenzpunkt ist das rechte Ende des Referenznockens oder der erste Nullimpuls rechts nach dem Ende des Referenznockens.

Einer der Binäreingänge P600 ... 606 muss auf "REFERENZNOCKEN" eingestellt sein. Die Referenzfahrt startet in Drehrichtung links, bis zur ersten positiven Flanke des Referenznockens wird P901 Referenzdrehzahl 1 verwendet, danach P902 Referenzdrehzahl 2. Im Unterschied zu Typ 2 startet der Antrieb nach links und wendet auf dem Referenznocken.

Wird die Referenzfahrt über die positive Flanke am Eingang "REF.-FAHRT START" gestartet, wird je nach Einstellung in P904 Referenzierung auf Nullimpuls auf die fallende Flanke des Referenznockens oder auf den Nullimpuls nach der fallenden Flanke des Referenznockens referenziert.

Wird die Referenzfahrt über den IPOS$^{plus®}$-Befehl `Go0` gestartet, wird je nach Argument `"ZP"` oder `"CAM"` auf die fallende Flanke des Referenznockens oder auf den Nullimpuls nach der fallenden Flanke des Referenznockens referenziert.

Der Referenznocken muss kurz vor oder genau mit dem linken Hardware-Endschalter beginnen und muss in den Endschalter hineinragen. Damit ist gewährleistet, dass während der Referenzfahrt kein Hardware-Endschalter angefahren wird.

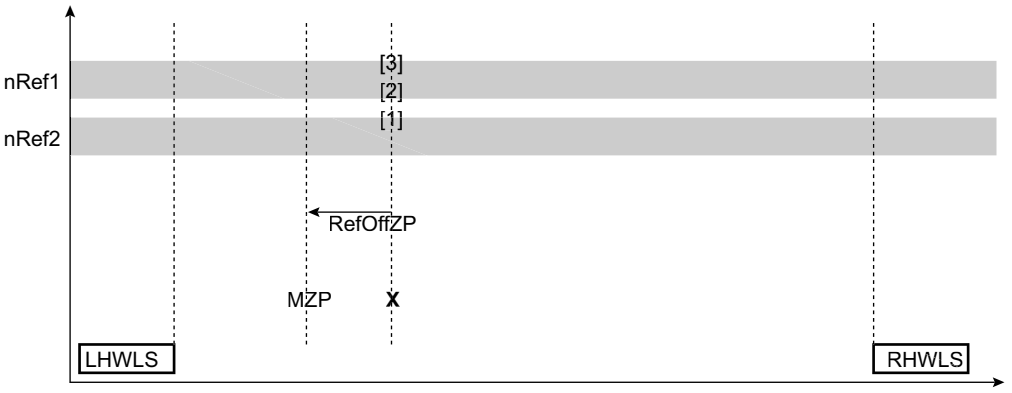


476764811

6.6.9 Typ 8: Ohne Freigabe

Referenzpunkt ist die aktuelle Position. Die Argumente im IPOS$^{plus®}$-Befehl `Go0"ZP"` bzw. `"CAM"` und P904 sind wirkungslos.

Dieser Referenzfahrtyp ist sinnvoll bei Absolutgebern und bei Antrieben, die im Stillstand referenziert werden sollen. So kann z. B. die Position einer Vorschubachse während dem Stillstand auf Null gesetzt werden. Damit kann der Maschinenbetreiber erkennen, wo der Antrieb sich innerhalb eines jeden Vorschubs befindet.



476761739

Im Gegensatz zu Typ 5 kann Typ 8 auch bei Systemzustand ungleich "A" ausgeführt werden.

6.7 Modulo-Funktion

6.7.1 Einleitung

Bei endlos in eine Richtung drehenden Anwendungen wie z. B. Rundtischen oder Transportketten kann die Modulo-Funktion aktiviert werden. Damit werden alle Positionsdaten zusätzlich im Bereich 0 bis (Modulo-Wert - 1 Inkr.) dargestellt. Ähnlich wie beim Zifferblatt einer Uhr springt, wenn der vom Anwender definierte Modulo-Wert (z. B. 100 mm oder 360°) überfahren wird, der Modulo-Positionswert wieder auf Null.

Es ist auch möglich mit den inkrementellen Werten des mit P941 ausgewählten Gebers zu positionieren und die Modulo-Funktion lediglich im Hintergrund zu aktivieren z.B. um Umdrehungen am Abtrieb zu zählen.

Die Modulo-Funktion hat folgende Eigenschaften:

- Positionsvorgabe in Abtriebseinheiten. Damit kann eine 360° Rundtischumdrehung direkt vorgegeben werden ohne wie bisher auf den IPOS$^{plus®}$-Geber inkrementell umrechnen zu müssen. Beispiele hierfür sind:
  - Rundtischumdrehung 360° = Modulo-Wert = 2^{16} Inkremente.
  - Wegstrecke bei einem Maschinentakt = Modulo-Wert = 2^{16} Inkremente (Rundtisch mit 4 Stationen: 1 Takt = 90° = Modulo-Wert).
- Dauerhaft korrekte Positionierung ohne Langzeitdrift und Fehlpositionierung ("Position läuft nicht davon") auch bei ungeradzahligen Getriebeübersetzungen, wenn die Projektierungsrichtlinien eingehalten sind.
- Bisherige Lösungen waren der Anbau eines externen Streckengebers oder die Erfassung des Nulldurchganges durch einen digitalen Eingang mit zusätzlichem Programmieraufwand in IPOS$^{plus®}$.
- Absolute Positionsvorgabe über mehrere Umdrehungen möglich.
- Vorgabe einer Verfahrstrategie: der Positions-Sollwert kann auf dem kürzesten Weg, links herum oder rechts herum angefahren werden.

Die Endlospositionierung in Kombination mit Absolutwertgeberauswertung über DIP11 ist ab der Firmwareversion 14 (822 890 6.14) möglich (Fehler F92 DIP Arbeitsbereich tritt bei aktivierter Modulo-Funktion nicht mehr auf).

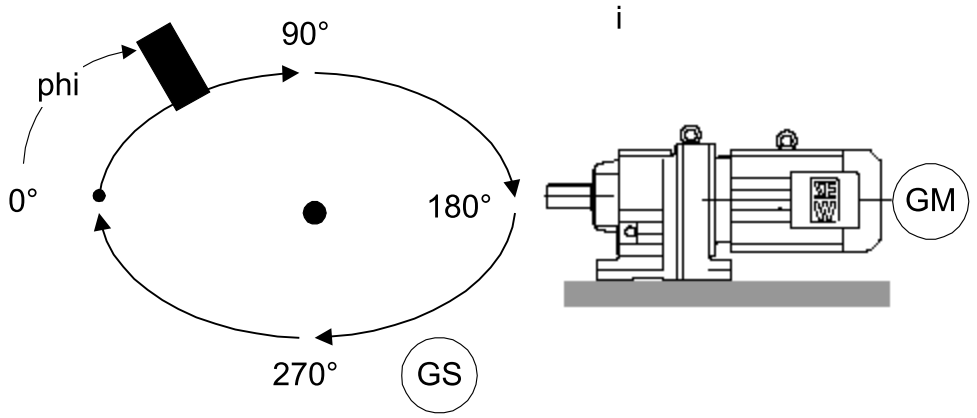
	HINWEIS Systembedingt muss das Getriebe sowie evtl. vorhandene Vorgelege über die Zähnezahlen nachgebildet werden. Erfragen Sie die exakten Zähnezahlen beim Hersteller des Getriebes. Übernehmen Sie nicht das Übersetzungsverhältnis vom Typenschild. Außerdem wird bei der Auswahl des Getriebes eine maximal darstellbare Zielposition errechnet, die nicht überschritten werden darf. Dies ist bereits während der Projektierungsphase zu berücksichtigen (Kapitel Projektierung).

6.7.2 Funktionsweise

Bei aktivierter Modulo-Funktion werden Positionsvorgaben anstelle von Inkrementen an der Motorwelle in Abtriebseinheiten ausgedrückt.

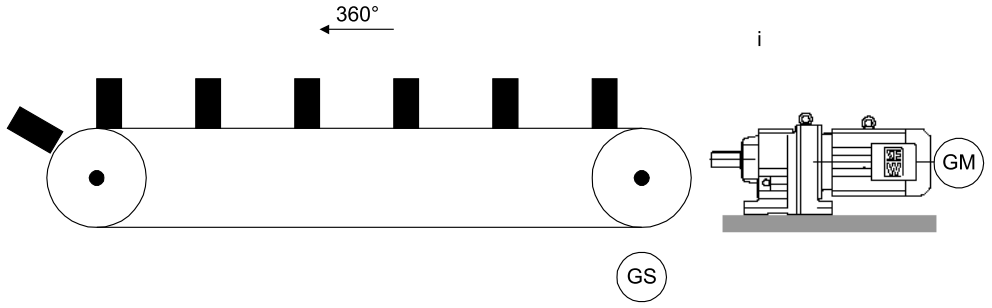
Beispiele für Abtriebseinheiten (GM = Motorgeber, GS = Streckengeber):

- Rundtischanwendungen mit der Abtriebseinheit 360° entsprechen einer Rundtischumdrehung.



476886155

- Transportkette mit einem Trägerabstand als Abtriebseinheit .



476887691

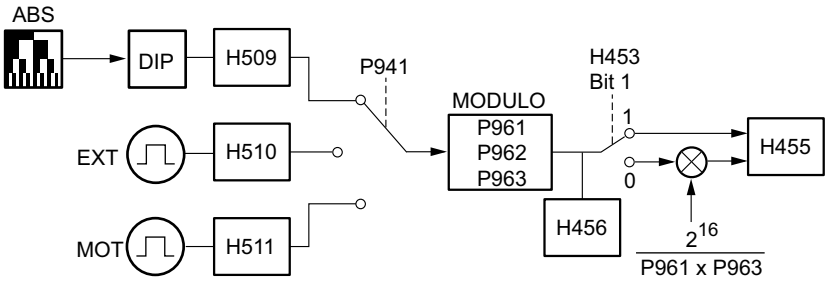
Während der Inbetriebnahme wird die Mechanik der Anwendung nachgebildet. Dazu ist die Angabe der exakten Zahnzahlen des Getriebes sowie evtl. des Vorgeleges erforderlich. Dargestellt werden diese Angaben in folgenden SHELL-Parametern:

- Modulo Zähler- und Nennerfaktor P961 / P962
- Modulo Geberauflösung P963

Um der Systemsoftware des MOVIDRIVE® Zielpositionen in Abtriebseinheiten vorgeben zu können stehen dem Anwender die folgenden IPOS$^{plus®}$-Systemvariablen zur Verfügung:

- Modulo Target Position H454 zur Beschreibung der Zielposition
- Modulo Ist-Position Abtrieb H455 zum Lesen der Ist-Position

Die Ist-Position H455 wird nach folgendem Blockschaltbild berechnet:



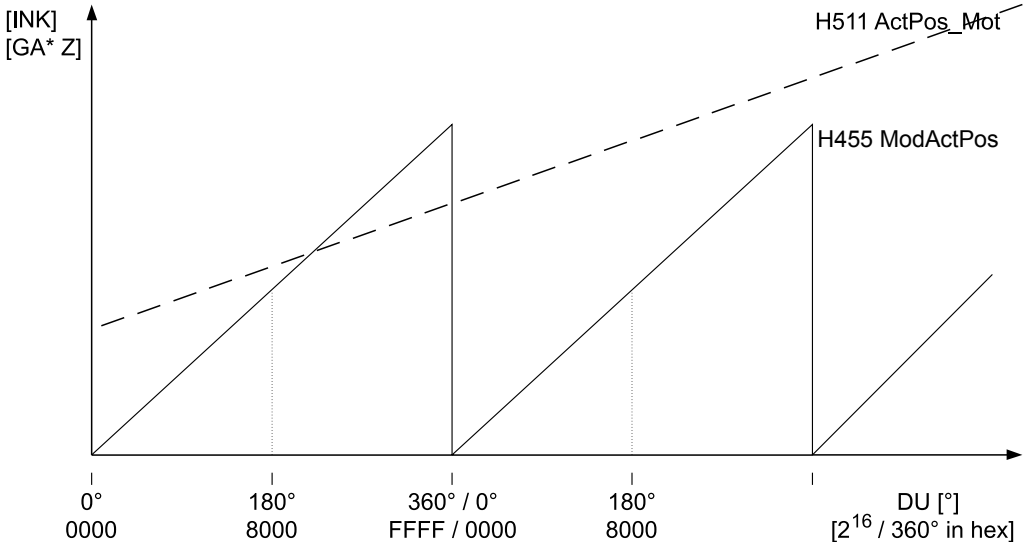
476895243

P941 = Quelle Ist-Position
P961 = Modulo Zähler
P962 = Modulo Nenner
P963 = Modulo Geberauflösung
H509 = Aktuelle Ist-Position Absolutwertgeber
H510 = Aktuelle Ist-Position externer Geber
H511 = Aktuelle Ist-Position Motorgeber
H453 = Modulo Steuerung
H455 = Modulo Ist-Position Abtrieb
H456 = Modulo Zähler

	HINWEIS Soll auf die Zielposition im Modulo-Bereich positioniert werden, beginnt der Antrieb zu positionieren, wenn die Zielposition in die Variable H454 MOD.TAGPOS geschrieben wird. GO-Befehle beziehen sich auf den IPOS$^{plus®}$-Geber und können nicht verwendet werden, um mit Modulo zu positionieren.

	HINWEIS Die nachfolgenden Beispiele arbeiten mit der Einstellung Modulo-Wert = 360°. Es ist prinzipiell auch möglich, auf einen anderen physikalischen Wert als Modulo-Wert zu skalieren.

In der untenstehenden Abbildung ist der Zusammenhang zwischen der aktuellen Position des IPOS$^{plus®}$-Gebers, z. B. des Motorgebers H511 sowie der Ist-Position in der Modulo-Darstellung dargestellt. Die Modulo-Istposition bewegt sich immer innerhalb der Abtriebseinheit z. B. von 0° (= 0 Inkremente) bis 360° (= 2^{16} Inkremente).



477219595

Die Vorgabe einer neuen Zielposition erfolgt durch Beschreibung der IPOS$^{plus®}$-Variablen H454 MODTAGPOS im 32-Bit-Format.

Die Systemsoftware unterscheidet dabei 2 einstellbare Darstellungsformen (einstellbar über H453, Bit 1):

- 360° = 16 Bit (im Folgenden als "2^{16} / 360°" bezeichnet) - Standardeinstellung: Dabei kann der höherwertige Bitbereich für die Vorgabe ganzzahliger 360° Umdrehungen genutzt werden.
- 360° = 32 Bit (im Folgenden als "2^{32} / 360°" bezeichnet): Diese Darstellung sollte wegen der Einschränkung des maximalen Darstellungsbereiches vermieden werden. Bei Nutzung entspricht das Produkt aus Modulo Zähler sowie Modulo Geberauflösung einer 360° Umdrehung.

Beispiel für Positionsvorgabe in Abtriebseinheiten (in hexadezimaler Darstellung):

Darstellung mehrerer ganzzahliger Umdrehungen **H454 MODTAGPOS = k × 360° + 0 ... 360° = k × 2^{16} + 0 ... (2^{16} -1)** **Darstellung einer ganzzahligen Umdrehung** **H454 MODTAGPOS = 0 ... 360°**	
Zielposition in Abtriebseinheit [°]	Umsetzung über IPOS$^{plus®}$ Variable H454 MOD.TAGPOS
360°	0001 0000
3 × 360°	0003 0000
180°	0000 8000
270°	0000 C000

6.7.3 Verfahrstrategien

Bei aktivierter Modulo-Funktion können für das Positionieren verschiedene Verfahrstrategien eingestellt werden. Die Verfahrstrategie für das Referenzieren ist davon unabhängig.

Referenzieren

Gestartet wird die Referenzfahrt wie bei ausgeschalteter Modulo-Funktion. Bei aktivierter Modulo-Funktion, wird die Variable H455 MOD.ACTPOS referenziert.

Ein angegebener Referenz-Offset P900 wird in der Skalierung Abtriebseinheit (2^{16} = 360°) interpretiert.

Nach Abschluss der Referenzfahrt wird die aktuelle Zielposition H454 MOD.TAGPOS auf den Istwert MOD.ACTPOS gesetzt (siehe Kapitel "Referenzieren").

ACTPOS.MOT H511 wird nicht genullt (referenziert).

Positionieren

Die Verfahrstrategie für das Positionieren wird über die SHELL Parameter 960 Modulo-Funktion angewählt. Die Modi können durch ein IPOS$^{plus®}$-Programm mit dem MOVILINK-Befehl umgeschaltet werden (Kapitel Anwenderschnittstelle). Die Beispiele beziehen sich auf die gewählte Auflösung (2^{16} / 360°).

Als IPOS$^{plus®}$-Geber wurde ein Motorgeber eingetragen (P941 Quelle Ist-Position = MOTORGEBER(X15)).

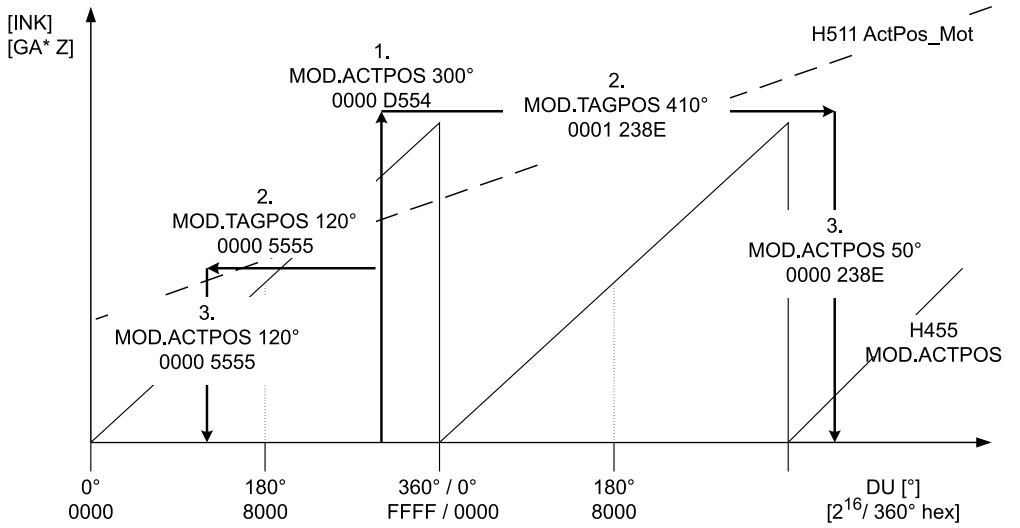
"kurzer Weg"

ModuloMode "kurzer Weg" (P960 = KURZ) - Standardeinstellung

Ausgehend von der aktuellen Ist-Position H455 MOD.ACTPOS wird der kürzeste Verfahrweg zur angeforderten Zielposition H454 MOD.TAGPOS ermittelt. In Abhängigkeit vom kürzesten Weg wird die Drehrichtung gewählt.

Darstellbare Zielposition:

H454 MOD.TAGPOS = k × 360° + 0 ... 360° = k × 2^{16} + 0 ... (2^{16} - 1)



477225739

1. Ist-Position vor Positionierung	Modulo Actual Position
2. Absetzen der Zielposition	Modulo Target Position
3. Ist-Position nach Positionierung	Modulo Actual Position

Modulo kurzer Weg (Anmerkung: die Achse fährt bei MOD.TAGPOS = 120° nur dann nach links, wenn der Antrieb mindestens 1 Inkr. links von 300° steht, da 300° + 180° = 120° und 300° - 180° = 120°).

Um also eine Achse, die bei 0° steht, 1 Umdrehung nach rechts zu positionieren, muss H454 = 0x 10000 gesetzt werden, um dieselbe Achse 1 Umdrehung nach links zu positionieren, muss H454 = 0x FFFF0000 gesetzt werden.

"rechts"

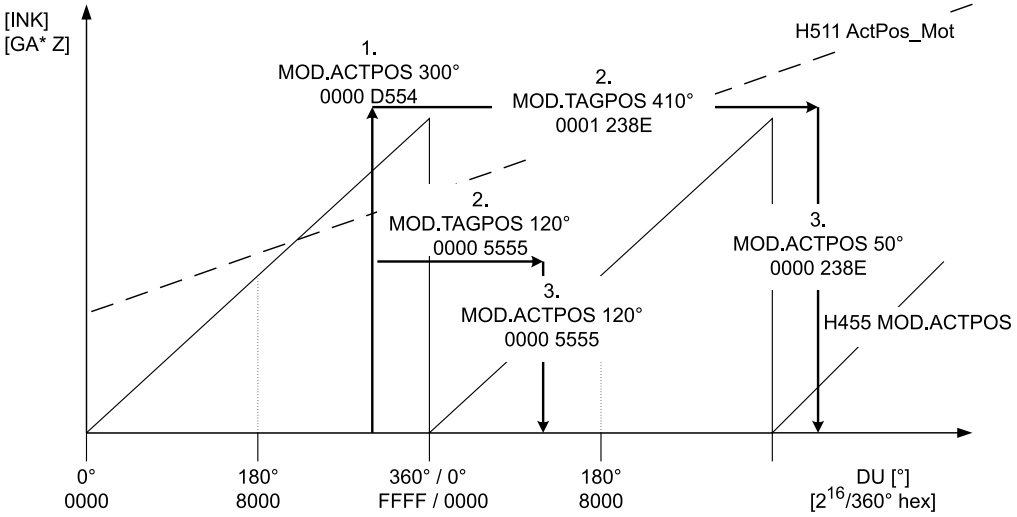
ModuloMode "rechts" (P960 = RECHTS)

Der Antrieb bewegt sich von der aktuellen Ist-Position H455 MOD.ACTPOS in Rechtsrichtung auf die Zielposition H454 MOD.TAGPOS.

Darstellbare Zielposition:

H454 MOD.TAGPOS = k × 360° + 0 ... 360° = k × 2^{16} + 0 ... (2^{16} - 1)

Im High-Bereich sind dabei nur positive Werte erlaubt. Wird diese Bedingung verletzt und das Vorzeichenbit 2^{32} gesetzt, geht der Antriebsumrichter in den Fehlerzustand (IPOS$^{plus®}$-Programmfehler).



477230091

1. Ist-Position vor Positionierung — Modulo Actual Position
2. Absetzen der Zielposition — Modulo Target Position
3. Ist-Position nach Positionierung — Modulo Actual Position

"links"

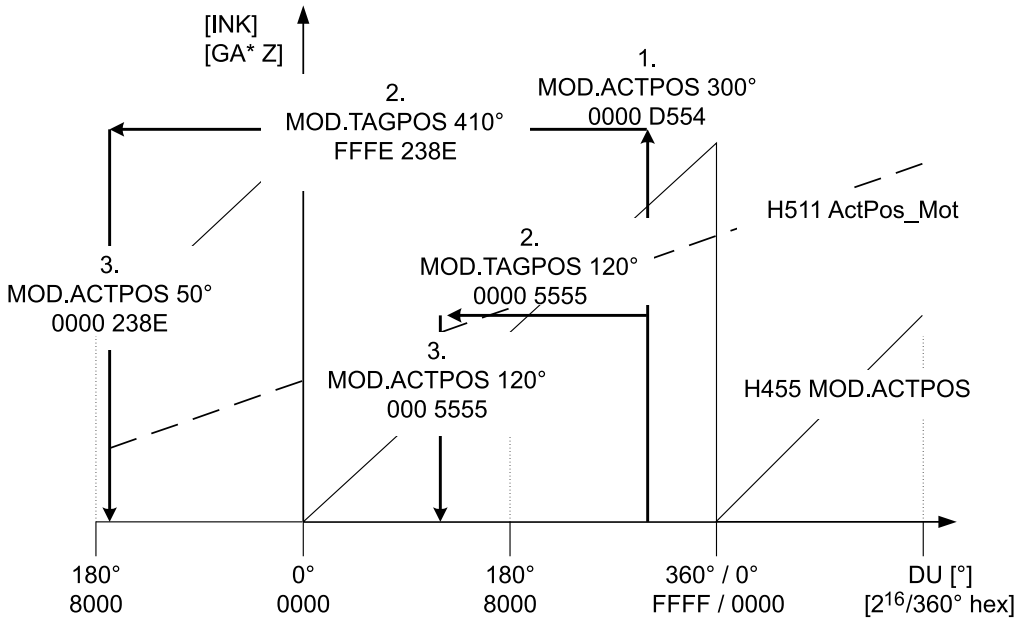
ModuloMode "links" (P960=LINKS)

Der Antrieb bewegt sich von der aktuellen Ist-Position H455 MOD.ACTPOS in Linksrichtung auf die Zielposition H454 MOD.TAGPOS.

Darstellbare Zielposition:

H454 MOD.TAGPOS = - k × 360° + 0 ... 360° = - k × 2^{16} + 0 ... (2^{16} - 1)

Im High-Teil sind dabei nur negative Werte erlaubt. Wird diese Bedingung verletzt und das Vorzeichenbit 2^{32} nicht gesetzt, geht der Antriebsumrichter in den Fehlerzustand (IPOS$^{plus®}$-Programmfehler).



Parameter und Variablen

Parameter und Variablen der Modulo-Funktion

Siehe auch Kapitel "IPOS$^{plus®}$-Parameter" und "Systemvariablen".

Parameter Nr.	Name
P 960, Index 8835	Modulo Control
P 961, Index 8836	Modulo Zähler
P 962, Index 8837	Modulo Nenner
P 963, Index 8838	Modulo Geberauflösung

HINWEIS

Um die Modulo-Funktion nutzen zu können muss die Randbedingung Produkt aus Modulo Geberauflösung und Modulo Zähler < 2^{31} erfüllt sein.

Variable Nr.	Name
H453	MODULOCTRL
H454	MOD.TAGPOS
H455	MOD.ACTPOS
H456	MODCOUNT

6.7.4 Projektierung

Definition der Antriebseinheit
- Getriebe und Vorgelege bestimmen Abtriebseinheit 360°
- Maximale Zielposition in "Anzahl von Antriebseinheiten" festlegen
- 16 Bit oder Auflösung (Geber × Modulo Zähler) für 360° festlegen

↓

Bestimmung der SHELL-Modulo-Parameter
- Modulo Funktion P960 (Auswahl der Verfahrstrategie)
- Modulo Zähler P961
- Modulo Nenner P962
- Modulo Geberauflösung P963

Hilfsmittel zur Bestimmung der Getriebezahnzahlen:
- SEW-Projektierungsmappe
- SEW-Wingear-Programm um Zähler-/Nennerfaktoren zu kürzen

↓

Modulo-Darstellungsbereich und maximale Abtriebsposition
- Bedingung für den Darstellungsbereich: Modulo Geberauflösung × Modulo Zähler < 2^{31}
- Bedingung für maximale Abtriebsposition: = 2^{31} / (Modulo Geberauflösung × Modulo Zähler)

Bei Verletzung dieser Bedingungen kommt es zu Fehlpositionierungen!

↓

Umsetzung im IPOS$^{plus®}$-Programm
- Soll-Positionierung mit Variablen H454 ModTagPos vorgeben: MOD.TAGPOS = k × 360° + 0 ... 360° = k × 2^{16} + 0 ... (2^{16} - 1)
- Ist-Position in Variable H455 ModActPos ablesen: MOD.ACTPOS = ... + 0 ... 360° = ... + 0 ... (2^{16} - 1)

Die angeforderte Zielposition in der ModActPos wird von der System-Software eingelesen und danach das High-Wort auf 0 gesetzt!

Die Ist-Position ModuloAbsIst bewegt sich immer zwischen 0° und 360°!

6.7.5 Projektierungsbeispiele

Kettenförderer

1. Schritt: Definition der Abtriebseinheit

Die Positionen für einen Kettenförderer sollen in Abtriebseinheiten vorgegeben werden. Eine 360° Umdrehung am Getriebeabtrieb entspricht der Modulo-Abtriebseinheit von 360°.

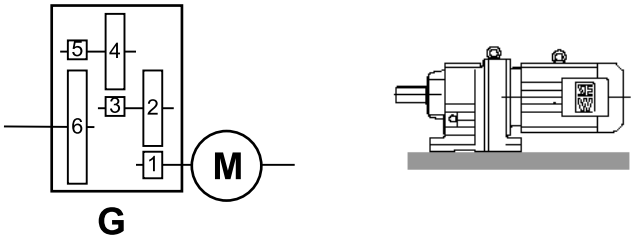
2. Schritt: Bestimmung der SHELL-Parameter

Technische Daten	
Getriebetyp	KA47B
Abtriebsdrehzahl [min^{-1}]	19
Motordrehzahl [min^{-1}]	2000
Getriebeübersetzung i	104.37
Motortyp	DY71S

Die Zahnzahlen des Getriebes können SEW-Mitarbeiter für Sie aus der SEW-Projektierungsmappe (DriveNet) ablesen oder aus dem elektronischen Typenschild (nur bei Hiperface®) ausgelesen werden.

In diesem Beispiel wurden folgende Zahnzahlen entnommen:

Z1 = 17 / Z2 = 74 / Z3 = 8 / Z4 = 33 / Z5 = 16 / Z6 = 93



477251979

Zur Ermittlung der SHELL-Parameter Modulo Zähler, Modulo Nenner sowie Modulo Geberauflösung muss folgende Berechnung durchgeführt werden:

$$\frac{M_N}{M_D} = i_G \times i_{AG}$$

$$\frac{M_N}{M_D} = \frac{Z_2 + Z_4 + Z_6}{Z_1 + Z_3 + Z_5}$$

$$\frac{M_N}{M_D} = \frac{227106}{2176} = \frac{113553}{1088}$$

M_N = Modulo Zähler (Nominator)
M_D = Modulo Nenner (Denominator)
i_G = i Getriebe
i_{AG} = i Vorgelege

Im obigen Beispiel wurden die Zähler- und Nennerfaktoren gekürzt (erfolgt beim Wingear-Programm automatisch).

Somit ergeben folgende Eingabewerte für die SHELL-Parameter:

- Modulo Zähler = 113553
- Modulo Nenner = 1088
- Modulo Geberauflösung = 4096

3. Schritt: Modulo-Darstellungsbereich und maximale Zielposition

Überprüfung des Modulo-Darstellungsbereiches:

Produkt aus Modulo Geberauflösung und Modulo Zähler muss < 2^{31} (dezimal 2147483648) sein.

Modulo Zähler × Modulo Geberauflösung = 113553 × 4096 = 465113088

=> Die Bedingung ist erfüllt, die Zielposition lässt sich darstellen.

Überprüfung der maximalen Zielposition:

$$TP_{\max} = \frac{2^{31}}{M\,360^\circ} = \frac{2^{31}}{M_N \times M_{ER}} = \frac{2^{31}}{113553 \times 4096} = 4.6$$

TP_{max} =Maximale Zielposition
M = Modulo
M_N = Modulo Zähler (Nominator)
M_{ER} = Modulo Geberauflösung

Die maximale Zielposition entspricht 4,6 Abtriebsumdrehungen.

Kettenförderer mit Träger

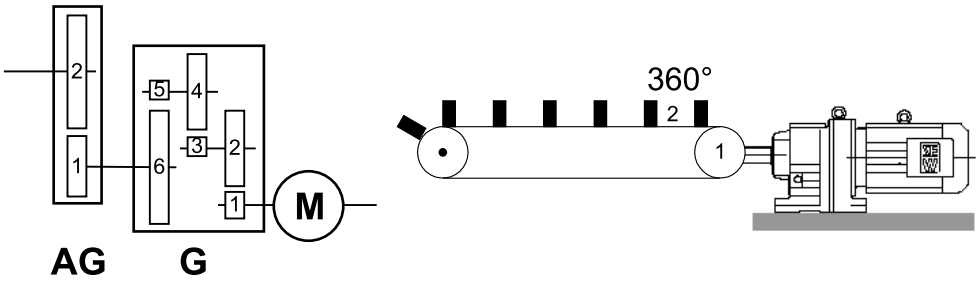
1. Schritt: Definition der Abtriebseinheit

Das vorherige Beispiel soll nun erweitert werden:

An dem Getriebe ist ein Zahnrad befestigt, das eine Kette antreibt. An jedem 36. Kettenglied sind Materialträger montiert.

2. Schritt: Bestimmung der SHELL-Parameter

Technische Daten	
Getriebetyp	KA47B
Abtriebsdrehzahl [min^{-1}]	19
Motordrehzahl [min^{-1}]	2000
Getriebeübersetzung i	104.37
Motortyp	DY71S



477253515

Anzahl der Zähne des Kettenrads des Vorgeleges: $Z_{AG}1 = 5$

Kettengliederabstand zum Träger: $Z_{AG}2 = 36$

i Vorgelege = 5/36

$$\frac{M_N}{M_D} = i_G \times i_{AG}$$

$$\frac{M_N}{M_D} = \frac{Z_2 + Z_4 + Z_6}{Z_1 + Z_3 + Z_5} \times \frac{Z_{AG}2}{Z_{AG}1}$$

$$\frac{M_N}{M_D} = \frac{1021977}{1360}$$

Modulo Zähler = 1021977

Modulo Nenner = 1360

Modulo Geberauflösung = 4096

3. Schritt: Modulo-Darstellungsbereiches und maximalen Zielposition

Überprüfung des Modulo-Darstellungsbereiches:

Produkt aus Modulo Geberauflösung und Modulo Zähler muss < 2^{31} (dezimal 2 147 483 648) sein.

Modulo Zähler × Modulo Geberauflösung = 1021977 × 4096 = 4 186 017 792

Die Bedingung wird verletzt, die gewünschte Zielposition lässt sich nicht darstellen. Würde der Anwender diese Applikation umsetzen, kommt es zu Fehlpositionierungen. Systemintern wird bei Vorgabe z. B. einer Zielposition von 180° das Vorzeichenbit gesetzt, der Antrieb positioniert falsch.

Lösung: Auswahl eines Getriebes mit geänderter Übersetzung.

Durch Auswahl eines anderen Getriebes mit anderen Teilungsfaktoren der Zahnzahlen (d. h. Teile der Zahnzahlen kürzen sich bei der Berechnung weg) wird die darstellbare Zielposition größer.

6.7.6 Häufig gestellte Fragen

- Gibt es fertige Programme zur Nutzung der Modulo-Funktion?
  - Ja, das Applikationsmodul "Modulo Positionierung" mit wahlweiser Ansteuerung über Feldbus oder Hardware-Klemme.
- Warum wird die Einstellung P960 = KURZ empfohlen?
  - Beim Beschreiben der Zielposition im Stillstand wird die Verfahrstrategie strikt eingehalten. Steht der Antrieb z. B. bei P960 = RECHTS, wenn die Zielposition beschrieben wird, nur ein Inkrement rechts davon, fährt der Antrieb eine komplette Umdrehung.
- Warum führt die abgesetzte Zielposition nicht zu einer Positionsfahrt?
  - Der Antrieb muss in der Betriebsart "... & IPOS" in Betrieb genommen sein.
  - Über SHELL P960 muss eine Verfahrstrategie angewählt worden sein.
  - Vor dem Beschreiben der IPOS$^{plus®}$-Variable H454 ModTagPos muss der Antriebsumrichter im Zustand A "Technologieoption" sein.
- Darf die Zielposition zyklisch beschrieben werden?
  - Bei Zielen die im Bereich 0° bis 359,999° bzw 0 Inkremente ... 65535 Inkremente: ja!
  - Zyklisches Beschreiben von Zielpositionen ≥ 360° bewirkt eine "Endlospositionierung".
- Kann auch bei aktivierter Modulo-Funktion inkrementell positioniert werden?
  - Ja, aber um ungewollte "Quereffekte" zu vermeiden, sollte die Modulo-Positionierung durch SHELL-Parameter 960 abgeschaltet werden.
  - Nach Durchführung einer Referenzfahrt wird nur die IPOS$^{plus®}$ Variable H455 MOD.ACTPOS auf 0 gesetzt, NICHT die inkrementelle Ist-Position ACTPOS.MOT in Variable H511.
- Warum wird der Fehler IPOS-ILOOP F10 ausgelöst?
  - Der Interpreter des IPOS-Programms stößt auf einen Befehl, dessen Operanden nicht zulässig sind.

- Wie kann erreicht werden, die Positionierung nach Toggeln der Freigabe fortzusetzen?
  - Durch Setzen des Bits H453.0 (Variable ModuloCtrl und Bit TargetReset_Off).
- Wie verhält sich die Achse, wenn während einer Positionierfahrt der _AxisStop(AS_PSTOP) Befehl abgesetzt wird?
  - Es wird an der Positionierrampe verzögert, die Ist-Position wird nach Stopp der Achse angezeigt. Um die Positionierfahrt fortzusetzen muss die Zielposition mit einem neuen Wert beschrieben werden (z. B. Inkrementierung um 1 Bit). Zweite Lösungsvariante wäre anstelle der _AxisStop(AS_PSTOP) Befehle den _AxisStop(AS_RSTOP) zu verwenden. In diesem Fall würde bei aktiviertem Bit H453.0 (Variable ModuloCtrl und Bit TargetReset_Off) die Zielposition erhalten bleiben.

6.8 Nockenschaltwerke

Mit Nockenschaltwerken werden abhängig von der Position eines Antriebs Ausgänge gesetzt oder rückgesetzt. Damit können Sie wegabhängig weitere Aktoren wie z. B. Pneumatikzylinder ansteuern, eine zweite Achse starten (z. B. zum Verschleifen der Bahnkontur bei einem xy-Portal) oder zwei Achsen im selben Arbeitsbereich auf Kollision überwachen.

Jedes MOVIDRIVE® hat ein Standardnockenschaltwerk mit 1 Ausgang. Der Ausgang wird jedes Mal dann neu gebildet, wenn der Befehl im IPOS$^{plus®}$-Programm bearbeitet wird. Theoretisch sind beliebig viele Ausgänge möglich, die Anzahl der Ausgänge ist praktisch jedoch durch die IPOS$^{plus®}$-Programmlänge und die hinnehmbare Laufzeit begrenzt.

Neue MOVIDRIVE®-Geräte (MDx_A / MCV / MCS / MCF ab Version .14, MCH ab Version .13 und MDx_B) und die Technologieoption haben zusätzlich ein erweitertes Nockenschaltwerk mit 8 Ausgängen, das zyklisch von der Firmware im Hintergrund gerechnet wird.

Um ein Nockenschaltwerk im Antrieb zu initialisieren und den Status der Nocken auszuwerten wird der GETSYS-Befehl benutzt.

Der GETSYS-Befehl greift auf eine Datenstruktur zu. Mit dem höchstwertigen Bit der ersten Variable dieser Datenstruktur wird entschieden, auf welches Nockenschaltwerk sich der GETSYS-Befehl bezieht (Bit 31 = 0: Standardnockenschaltwerk und Bit 31 = 1: Erweitertes Nockenschaltwerk).

Sind im vorliegenden Gerät beide Nockenschaltwerke möglich, empfiehlt SEW zunächst das erweiterte Nockenschaltwerk zu benutzen. Es ist auch möglich beide Nockenschaltwerke parallel zu verwenden. Ausgänge beider Nockenschaltwerke können auf demselben Binärausgangswort ausgegeben werden, indem z. B. auf den niederwertigen 2 Bits Ausgänge des Standardnockenschaltwerks und auf den höherwertigen 4 Bits Ausgänge des erweiterten Nockenschaltwerks um 2 nach links geschoben ausgegeben werden.

i	**HINWEIS** Wird das Ausgangsbit einer Nocke auf einen binären Klemmenausgang kopiert, so wird wie bei allen Bits, die in IPOS$^{plus®}$ auf Ausgänge kopiert werden, der Ausgang 1 ms später gesetzt..

6.8.1 Standardnockenschaltwerk

Eigenschaften des Standardnocken-schaltwerks

- Es ist bei allen Betriebsarten mit Geber verfügbar.
- Pro Deklaration und Aufruf einer Datenstruktur wird 1 Ausgang mit Totzeitkompensation abhängig von 4 Positionsfenstern (definiert durch einen linken und rechten Grenzwert) gesetzt bzw. rückgesetzt.
- Die Grenzen eines Positionsfensters können während der Laufzeit verändert werden und beim nächsten GETSYS-Befehl berücksichtigt werden. Damit ist es möglich, bei einer reversierenden Achse auf der Rückfahrt andere Nockenbereiche zu verwenden als bei der Hinfahrt.
- Der Ausgang des Nockens kann einem beliebigen Bit einer Variable zugeordnet werden.
- Theoretisch sind beliebig viele Ausgänge möglich, die Anzahl der Ausgänge ist praktisch jedoch durch die IPOS$^{plus®}$-Programmlänge und die hinnehmbare Laufzeit begrenzt.
- Der Ausgang wird beim Aufruf des GETSYS-Befehls neu gebildet unabhängig davon, ob der Antrieb referenziert ist oder nicht.
- Der GETSYS-Befehl initialisiert die Funktion und bildet, wenn der Befehl aufgerufen wird, einmalig den neuen Status des Ausgangs. Der Befehl muss also jedesmal dann aufgerufen werden, wenn ein neuer Status im IPOS$^{plus®}$-Programm benötigt wird - die Neubildung des Nockenausgangs ist abhängig von der Programm-Zykluszeit.
- Die Bezugsgröße ist einstellbar, typische Bezugsgrößen sind:
  - H511 - aktuelle Ist-Position des Motorgebers
  - H510 - aktuelle Ist-Position des externen Gebers
  - H509 - aktuelle Ist-Position eines SSI-Absolutgebers (Option DIP11A)
  - H455 - aktuelle Ist-Position des Motorgebers im Modulo-Format
  - H376 - aktuelle Ist-Position des Leitwertes (nur bei Technologiefunktion Kurvenscheibe oder interner Synchronlauf)
- Die Ausgänge der Nocken behalten ihre Werte zwischen den GETSYS-Befehlen und werden lediglich nach dem Reset gelöscht.
- Wird die Nockenfunktion n Mal pro 1 ms aufgerufen können n Nockenausgänge gebildet werden (z. B. in einer schnellen Task, wie Task 3 bei MOVIDRIVE® B, die mehrere IPOS$^{plus®}$-Befehle pro 1 ms abarbeiten kann). Da das MOVIDRIVE® jede 1 ms einen neuen Lagewert bildet, arbeiten jedoch alle innerhalb der 1 ms bearbeitenden Befehle mit demselben Lagewert.

Aufruf des Standardnockenschaltwerks

Compiler: _GetSys(Cam1 ,GS_CAM);

initialisiert das Nockenschaltwerk und bildet den Status eines Ausgangs mit der Datenstruktur ab der Variable Cam1

Assembler: GETSYS Hxx = CAM

initialisiert das Nockenschaltwerk und bildet den Status eines Ausgangs mit der Datenstruktur ab der Variable Hxx

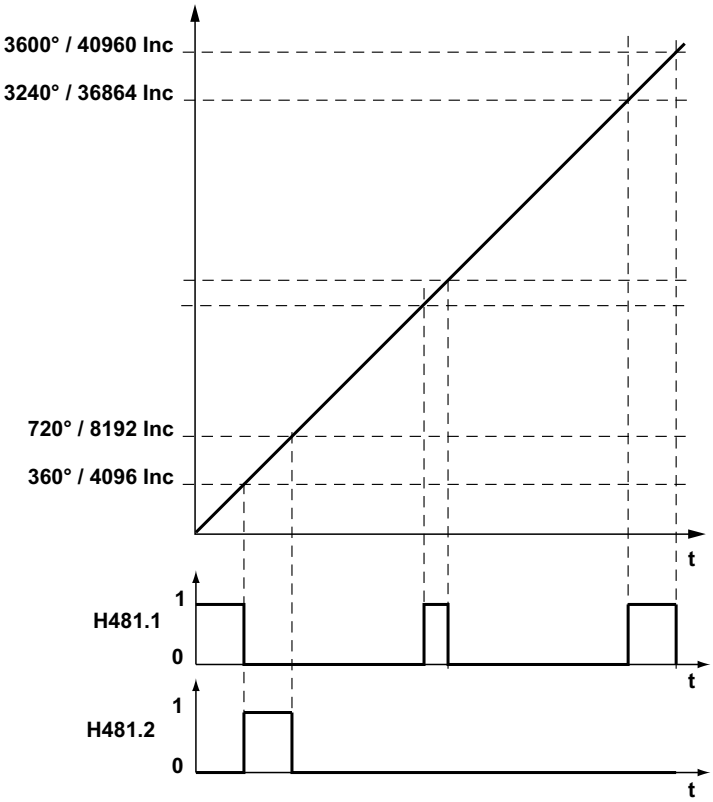
Datenstruktur des Standardnockenschaltwerks

Variable	Symbolischer Name des Elements in der SEW-Standard-Struktur	Kurzbeschreibung
H+0	GSCAM.SourceVar	Nummer der Bezugsvariable für die Nockenberechnung, typische Bezugsvariablen sind: • H511 (Ist-Position Motorgeber) • H510 (Ist-Position SSI-Geber) • H509 (Ist-Position ext. Geber) • H455 (Ist-Position des Motorgebers im Modulo-Format) z. B. H+0 = 511 für Bezuggröße H511 Bit 31 dieser Variable muss 0 sein!
H+1	GSCAM.DbPreCtrl	Totzeitkompensation in 0,1 ms zur Kompensation der Totzeit eines am Umrichter angeschlossenen Aktors. Abhängig von der Änderungsgeschwindigkeit des Wertes der Bezugsvariable wird der Ausgang so vorgesteuert, dass der Ausgang um diese Zeit vorher geschaltet wird.
H+2	GSCAM.DestVar	Nummer der Variablen in der der Ausgang gesetzt oder rückgesetzt wird.
H+3	GSCAM.BitPosition	Position des Bits in der Variablen H+2, wird der Ausgang des Nockens auf einen Geräteausgang (z.B. H481) gelegt, ist dieser Binärausgang mit P620 - P639 als IPOS$^{plus®}$-Ausgang zu reservieren.
H+4	GSCAM.BitValue	Polarität des Ausgangs, 0 = Bit gesetzt, wenn Bezugsvariable H+0 innerhalb der Positionsfenster H+6 bis H+13 1 = Bit gesetzt, wenn Bezugsvariable H+0 außerhalb der Positionsfenster H+6 bis H+13
H+5	GSCAM.NumOfCam	Anzahl der nachfolgend in H+6 bis H+13 definierten Positionsfenster, der linke Grenzwert muss immer kleiner sein wie der rechte Grenzwert, wird bei einer Modulo-Achse ein Positionsfenster über die 360° - 0°-Grenze benötigt, muss dieser Bereich in 2 Positionsfenster unterteilt werden. Damit können für diesen Ausgang max. 3 zusammenhängende Bereiche eingestellt werden.
H+6	GSCAM.PosL1	Linker Grenzwert des ersten Positionsfensters
H+7	GSCAM.PosR1	Rechter Grenzwert des ersten Positionsfensters
H+8	GSCAM.PosL2	Linker Grenzwert des zweiten Positionsfensters
H+9	GSCAM.PosR2	Rechter Grenzwert des zweiten Positionsfensters
H+10	GSCAM.PosL3	Linker Grenzwert des dritten Positionsfensters
H+11	GSCAM.PosR3	Rechter Grenzwert des dritten Positionsfensters
H+12	GSCAM.PosL4	Linker Grenzwert des vierten Positionsfensters
H+13	GSCAM.PosR4	Rechter Grenzwert des vierten Positionsfensters

Beispiel

Ein Fahrantrieb hat einen Verfahrbereich von 10 Motorumdrehungen.

Ein Ausgang soll gesetzt werden, wenn der Antrieb sich in der ersten, in der letzten Motorumdrehung oder in einem Bereich von +/- 10° um die Mitte des Verfahrbereichs befindet. Ein zweiter Ausgang soll gesetzt werden, wenn der Antrieb sich in der zweiten Umdrehung befindet.



477561867

Notwendige Parametereinstellungen

P620 = IPOS$^{plus®}$-Ausgang

P621 = IPOS$^{plus®}$-Ausgang

Beispiel gelöst im Compiler

```
/*=============================================
IPOS-Quelldatei
===============================================*/
#include <const.h>
#include <io.h>
//Deklaration
GSCAM Cam1, Cam2; //Deklaration von Nockenausgang 1 und 2
/*=============================================
Hauptfunktion (IPOS-Eintrittsfunktion)
===============================================*/
main()
{
   /*-------------------------------------
   Initialisierung
   -------------------------------------*/
   Cam1.SourceVar = numof(ActPos_Mot); // Bezugsgröße Motorgeber
   Cam1.DbPreCtrl = 0; // keine Totzeitkompensation
   Cam1.DestVar = numof(StdOutpIPOS); // Ausgabe auf Do01 (H481.1)
   Cam1.BitPosition = 1;
   Cam1.BitValue = 0; // Ausgang = 1, wenn Wert im Fenster
   Cam1.NumOfCam = 3; // Anzahl der Fenster
   Cam1.PosL1 = 0; // 1. Fenster, linker Grenzwert
   Cam1.PosR1 = 4096; // 1. Fenster, rechter Grenzwert
   Cam1.PosL2 = 20366; // 2. Fenster, linker Grenzwert
   Cam1.PosR2 = 20594; // 2. Fenster, rechter Grenzwert
   Cam1.PosL3 = 36864; // 3. Fenster, linker Grenzwert
   Cam1.PosR3 = 40960; // 3. Fenster, rechter Grenzwert
   Cam2.SourceVar = numof(ActPos_Mot); // Bezugsgröße Motorgeber
   Cam2.DbPreCtrl = 0; // keine Totzeitkompensation
   Cam2.DestVar = numof(StdOutpIPOS); // Ausgabe auf Do02 (H481.2)
   Cam2.BitPosition = 2;
   Cam2.BitValue = 0; // Ausgang = 1, wenn Wert im Fenster
   Cam2.NumOfCam = 1; // Anzahl der Fenster
   Cam2.PosL1 = 4096; // 1. Fenster, linker Grenzwert
   Cam2.PosR1 = 8192; // 1. Fenster, rechter Grenzwert
   /*-------------------------------------
   Hauptprogramm-Schleife
   -------------------------------------*/
   while(1)
   {
      _GetSys(Cam1 ,GS_CAM ); // Ausgang der 1. Nocke bilden
      _GetSys(Cam2 ,GS_CAM ); // Ausgang der 2. Nocke bilden
   }

}
```

Beispiel gelöst im Assembler

(Der SET-Befehl ist nicht notwendig und dient nur zur Veranschaulichung)

```
SET     H13   = H511
GETSYS  H0    = CAM
GETSYS  H14   = CAM
END
```

Identifier		Value
H0	Source-Va	13
H1	DbPreCont	0
H2	Dest.-Var	481
H3	Bit-Posit	1
H4	Bit-Polar	0
H5		3
H6		0
H7		4096
H8		20366
H9		20594
H10		36864
H11		40960
H12		0
H13		4016
H14	Source-Va	511
H15	DbPreCont	0
H16	Dest.-Var	481
H17	Bit-Posit	2
H18	Bit-Polar	0
H19		1
H20		4096
H21		8192

477809419

6.8.2 Erweitertes Nockenschaltwerk

Eigenschaften des erweiterten Nockenschaltwerks

- Es ist ab MDx_A / MCV / MCS / MCF Software Version .14, MCH ab Software Version .13 und MDx_B verfügbar.
- Es kann nur in den Betriebsarten CFC oder Servo mit Technologieoption verwendet werden.
- Es stehen 8 Ausgänge (Nockenbits) zur Verfügung.
- Für jeden Ausgang können bis zu 4 Positionsfenster und eine Totzeitkompensation definiert werden (das entspricht vier Nocken auf einer mechanischen Nockenscheibe).
- Die Ausgänge 1-4 werden alle 1 ms, die Ausgänge 5-8 alle 4 ms abgearbeitet.
- Der GETSYS-Befehl initialisiert und startet die Funktion. Die Nocken werden mit festem Zeitraster im Hintergrund gebildet und sind unabhängig von der Zykluszeit des IPOS$^{plus®}$-Programms. Die Datenstrukturen sind vor dem ersten Aufruf mit sinnvollen Werten vorzubelegen.

- Änderungen in der Datenstruktur werden jede 1 ms übernommen. Damit können die Grenzen eines Positionsfensters während der Laufzeit verändert werden und werden im nächsten Bearbeitungszyklus der Nocke berücksichtigt. Damit ist es möglich bei einer reversierenden Achse auf der Rückfahrt andere Nockenbereiche zu verwenden als bei der Hinfahrt.
- Alle Ausgänge können zusammenhängend ab einem beliebigen Bit einer Variable abgelegt werden.
- Es ist möglich Ausgänge zu setzen, d. h. im Programm fest auf 1 oder 0 zu setzen.
- Die Bezugsgröße ist einstellbar, typische Bezugsgrößen sind:
  - H511 - aktuelle Ist-Position des Motorgebers
  - H510 - aktuelle Ist-Position des externen Gebers
  - H509 - aktuelle Ist-Position eines SSI-Absolutgebers (Option DIP11A)
  - H455 - aktuelle Ist-Position des Motorgebers im Modulo-Format
  - H376 - aktuelle Ist-Position des Leitwertes (nur bei Technologiefunktion Kurvenscheibe oder interner Synchronlauf)
- Die erweiterte Nockenfunktion kann gestoppt werden, indem GETSYS mit Bit 31 = 0 aufgerufen wird. Dadurch wird die Abarbeitung in der Firmware gestoppt. D. h. die Funktion nimmt keine Prozessorleistung mehr in Anspruch. Wird im Gegensatz dazu CamState mit 0x8000 0000 beschrieben, wird die Nockenfunktion ebenfalls gestoppt, läuft aber noch im Hintergrund weiter und setzt keine Ausgänge.

Aufruf des erweiterten Nockenschaltwerks

Compiler: _GetSys(CamArray ,GS_CAM);

initialisiert das Nockenschaltwerk und bildet den Status aller Ausgänge mit der Datenstruktur ab der Variable CamArray

Assembler: GETSYS Hxx = CAM

initialisiert das Nockenschaltwerk und bildet den Status aller Ausgänge mit der Datenstruktur ab der Variable Hxx

Datenstruktur des erweiterten Nockenschaltwerks

Die erweiterte Nockenfunktion wird mit Hilfe zweier Datenstrukturen parametriert, die CamControl-Struktur und die CamOutput-Struktur.

- Durch die CamControl-Struktur wird das globale Verhalten der erweiterten Nockenfunktion kontrolliert.
- Die CamOutput-Struktur dient der Definition eines Ausgangs (Ausgang einer Nockenscheibe) und wird folglich bis zu acht mal benötigt.

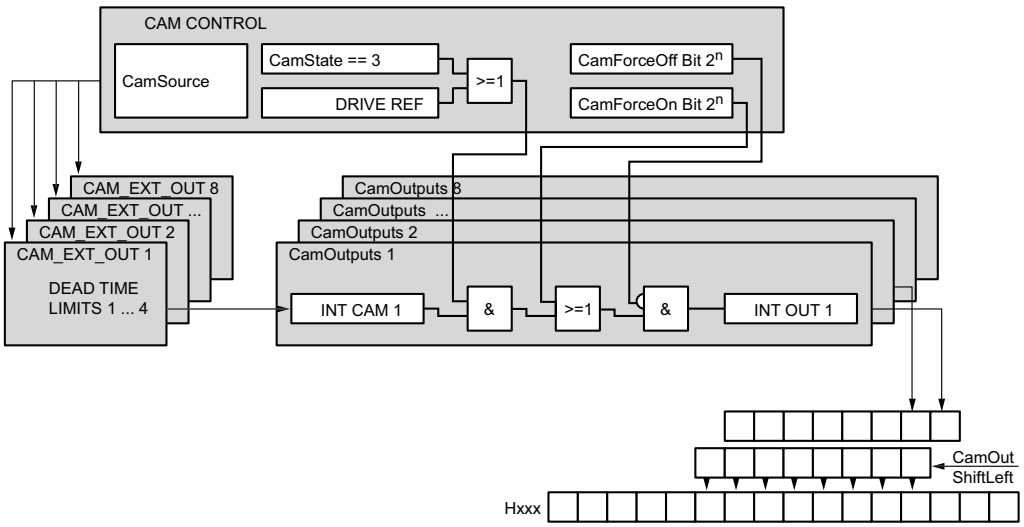
SEW-Standard-Struktur GSCAM_EXT

Variable	Name	Beschreibung
H+0	CamState	Bit 31 muss immer gesetzt sein ansonsten ist die Abarbeitung in der Firmware gestoppt. • 0x8000 000**0** = Funktion inaktiv, die Ausgänge der Nocken werden nicht mehr neu gebildet, gesetzte Ausgänge bleiben erhalten und werden erst nach einem Reset oder Spannung Aus/Ein gelöscht. • 0x8000 000**1** = Funktion aktiv, aber alle Nockenausgänge werden ausgeschaltet • 0x8000 000**2** = Funktion aktiv, wenn Antrieb referenziert ist (H473, Bit20 =1) • 0x8000 000**3** = Funktion auch ohne referenzierten Antrieb aktiv
H+1	CamReserved1	Reserviert
H+2	CamOutShiftLeft	Schiebt den internen Datenpuffer der Ausgänge vor dem Schreiben auf die Zielvariable H+6 um n Stellen nach links. Achtung: Beim Schieben geht die Information der oberen Ausgänge verloren. D. h. wenn Sie um 3 schieben, sind die oberen 3 Ausgänge mit 4 ms Zykluszeit nicht mehr nutzbar und die 4 Ausgänge mit 1 ms Zykluszeit sind dann den Bits 3 ... 6 und der eine Ausgang mit 4 ms Zykluszeit dem Bit 7 zugeordnet.
H+3	CamForceOn	Maske um Ausgänge zwingend zu setzen. Die Maske wirkt auf den internen Datenpuffer vor dem Shiften mit H+2 (NICHT auf die mit H+6 definierte Zielvariable)
H+4	CamForceOff	Maske um Ausgänge zwingend zu löschen. Die Maske wirkt auf den internen Datenpuffer vor dem Shiften mit H+2 (NICHT auf die mit H+6 definierte Zielvariable) CamForceOff ist dominant gegenüber CamForceOn
H+5	CamSource	Bit 31 schaltet zwischen voreingestellten Bezugsvariablen und einem Zeiger auf eine beliebige Bezugsvariable um. Bit 31= 0: • 0 = Geber X15 (Motorgeber, H511) • 1 = Geber X14 (externer Geber, H510) • 2 = Geber H509 (Absolutgeber DIP11A) • 3 = virtueller Geber • alle folgenden Werte sind reserviert! Bit 31= 1: CamSource enthält einen Zeiger auf eine IPOS$^{plus®}$-Variable + 2^{31}
H+6	CamDestination	Zeiger auf Zielvariable. Die in dem Wort der Zielvariablen nicht benutzten Bits können für andere Funktionen verwendet werden (Schiebt man z.B. mit Shift Left die Ausgänge um 4 nach links, kann man Bits 0-3 frei verwenden, Bit 4-7 sind für die Nockenfunktion und Bit 8-31 kann man frei verwenden. Werden die Ausgänge der Nocken auf Geräteausgänge (z.B. H481) gelegt, sind diese Binärausgänge mit P620 - P639 als IPOS-Ausgänge zu reservieren. Die in diesem Wort nicht benutzten Bits können für andere Ausgänge verwendet werden.
H+7	CamOutputs	Anzahl der Ausgänge (max. 8)
H+8	CamData 1	Zeiger auf erste CamOutput-Struktur (1. Ausgang)
...	...	
H+15	CamData 8	Zeiger auf letzte CamOutput-Struktur (8. Ausgang)

Struktur CAM_EXT_OUT

Variable	Name	Beschreibung
H+0	DeadTime	Totzeitkompensation für diesen Kanal (-500 ms ... 0 ... +500 ms), zur Kompensation der Totzeit eines am Umrichter angeschlossenen Aktors. Abhängig von der Änderungsgeschwindigkeit des Wertes der Bezugsvariable wird der Ausgang so vorgesteuert, dass der Ausgang um diese Zeit vorher geschaltet wird.
H+1	CamAreas	Anzahl der Positionsfenster für diesen Kanal (1 ... 4), der linke Grenzwert muss immer kleiner sein wie der rechte Grenzwert. Wird bei einer Modulo-Achse ein Positionsfenster über die 360° - 0°-Grenze benötigt, muss dieser Bereich in 2 Positionsfenster unterteilt werden. Damit können für diesen Ausgang max. 3 zusammenhängende Bereiche eingestellt werden.
H+2	LeftLimit1	Linke Grenze, Fenster 1
H+3	RightLimit1	Rechte Grenze, Fenster 1
...	...	...
H+8	LeftLimit4	Linke Grenze, Fenster 4
H+9	RightLimit4	Rechte Grenze, Fenster 4

Funktionsplan des erweiterten Nockenschaltwerks



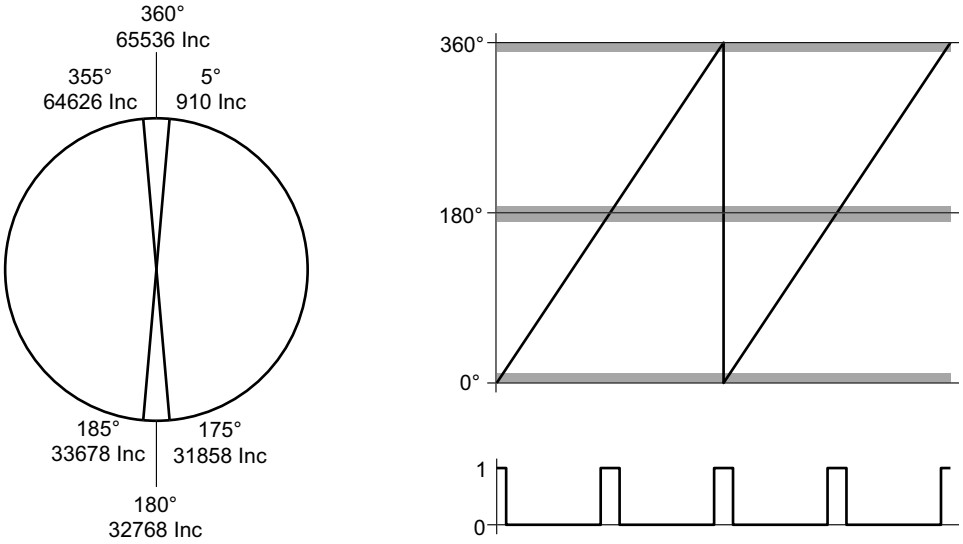
477814283

INT CAM 1: internes Nockensignal 1
INT OUT 1: internes Ausgangssignal 1
Hxxx: Ausgangsvariable definiert mit CamDestination

Beispiel

Ein Modulo-Rundtisch hat 2 Bearbeitungsstationen, die um 180° versetzt angeordnet sind. Er wird über ein Getriebe mit der Übersetzung 5:1 angetrieben. Ein Ausgang soll gesetzt werden, wenn der Antrieb sich im Bereich der Stationen +/-5° befindet.

(Anmerkung: bei aktiver Modulo-Funktion wird eine Lastumdrehung von 360° mit 65536 Inkrementen aufgelöst - siehe Modulo-Funktion)



477818635

Notwendige Parametereinstellungen

P620 = IPOS$^{plus®}$-Ausgang

P960 = z. B. KURZ

P961 = 5

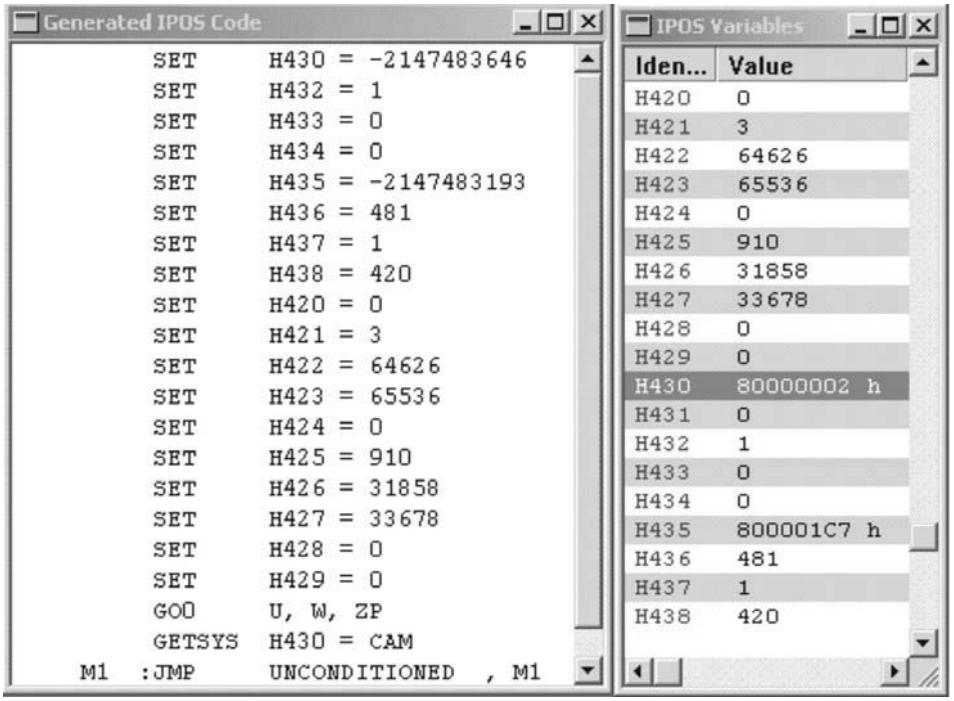
P962 = 1

P963 = 4096

Beispiel gelöst im Compiler

```
/*
=========================================================================
Programm-Gerüst für Applikationen mit der erw. Nockenfunktion
=========================================================================
*/
#include <const.h>
#include <io.h>
// Variablenstruktur für Nockenschaltwerk wird extern in *.h definiert
#include <camdef.h>
CamStructure CamData0;
ControlStructure MyCamControl;
/*==============================================
Hauptfunktion (IPOS-Eintrittsfunktion)
================================================*/
main()
{
   /*--------------------------------------
   Initialisierung
   --------------------------------------*/
   MyCamControl.CamControl=0x80000002; // erw. Funktion aktiv
   MyCamControl.CamOutShiftLeft=1;
   MyCamControl.CamForceOn=//Maske zwingend einzuschalten
   MyCamControl.CamForceOff=0; // Maske: Ausgänge zwingend auszuschalten
   MyCamControl.CamSource=numof(ModActPos) ¦ (1<<31);
   // Positions-Istwert Modulo-Format
   MyCamControl.CamDestination=481;  // Ausgänge Grundgerät
   MyCamControl.CamOutputs=1;  // Anzahl der Nockenscheiben (max. 8)
   MyCamControl.CamDataStr0=numof(CamData0);
   // Start Nockenstruktur 1 (Ausgangsbit 0)
   CamData0.DeadTime=0;
   CamData0.CamAreas=3;  // 3 Nockenbereiche wg. Modulo-Überl. im Fenster
   CamData0.LeftLimit1=  64626;  // 355° an der Last = 360° x 64626/65536
   CamData0.RightLimit1= 65536;  // 360° an der Last
   CamData0.LeftLimit2=  0;  // 0° an der Last
   CamData0.RightLimit2= 910;  // 5° an der Last
   CamData0.LeftLimit3=  31858;  // 175° an der Last
   CamData0.RightLimit3= 33678;  // 185° an der Last
   CamData0.LeftLimit4=  0;  // nicht verwendet
   CamData0.RightLimit4= 0;  // nicht verwendet
   _Go0( GO0_U_W_ZP );
   _GetSys( MyCamControl.CamControl ,GS_CAM );
   /*--------------------------------------
   Hauptprogramm-Schleife
   --------------------------------------*/
   while(1)
   {
   }
}
```

Beispiel gelöst im Assembler



477835787

7 Wegerfassung über binäre Eingänge

7.1 Arten von Einbaugebern

Für Asynchrone Drehstrommotoren bietet SEW-EURODRIVE eine Wegerfassung mit einem einfachen Inkrementalgeber (ohne Lageregelung) an. Diese Geber sind als Einbaugeber ausgeführt.

Die folgende Tabelle zeigt Ihnen, welcher Geber zu welchem Gerät von SEW-EURODRIVE empfohlen wird:

Geber	Gerät		
	MOVIDRIVE® B	MOVITRAC® B	MQx-Modul
EI7C		X	
EI76	X		X
EI72			
EI71			
ES16			X
NV26 (Näherungsgeber)			X

- Sie können auch einen, von der SEW-Empfehlung abweichenden Geber einsetzen, beachten Sie dazu den folgenden Hinweis:

HINWEIS

Wenn Sie einen Geber mit zu hoher Auflösung verwenden, kann das dazu führen, dass dieser bei Nenndrehzahl des Motors nicht mehr korrekt ausgewertet wird.

- Wählen Sie einen Geber dessen Auflösung zu der max. Impulsfrequenz des Umrichters passt.
- Beachten Sie dazu die technischen Daten der Einbaugeber im Handbuch "Praxis der Antriebstechnik - Gebersysteme" und die technischen Daten der binären Eingänge (Zählereingänge).

7.2 Prinzip der Wegerfassung

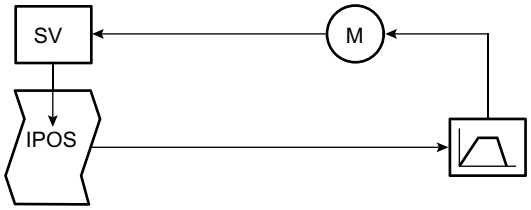
Die Spursignale werden auf 2 binäre Eingänge Ihres Umrichters übertragen und ohne Geberoption ausgewertet. Um die "Wegerfassung" zu aktivieren, wurde der Index 8845 eingeführt.

Der Index 8845 hat die folgende Belegung:

- Wegerfassung deaktiviert: 8845 = "0"
- Wegerfassung aktiviert: 8845 = "1"

Sie können den Index 8845 in einem IPOS$^{plus®}$-Programm mit dem MOVILINK-Befehl setzen.

Die folgende Tabelle zeigt die Verarbeitung der Ist-Position im IPOS$^{plus®}$-Programm:

Verarbeitung der Ist-Position im IPOS$^{plus®}$-Programm	
 2211727115	• Die Ist-Position des Einbaugebers wird automatisch auf der IPOS$^{plus®}$-Variablen H511abgebildet und kann programmgesteuert verarbeitet werden. • Die Verwendung des Einbaugebers kann insbesondere zum Einsatz von Applikationen verwendet werden, bei denen sonst über mehrere Initiatoren mit Eilgang-Schleichgang positioniert wird. • Der Geber kann nicht zur Motorführung (Betriebsart "…&n-Regelung") eingesetzt werden. • Der Geber kann nicht zur direkten Lageregelung (Betriebsart "…& IPOS") eingesetzt werden. • Zur Realisierung von einfachen Positionieraufgaben mit MOVITRAC® B steht Ihnen in MOVITOOLS® MotionStudio das Applikationsmodul "Einfach-Positionierung" zur Verfügung. SV = Systemvariable IPOS = IPOS$^{plus®}$-Programm

7.3 Wegerfassung mit MOVIDRIVE® B

SEW-EURODRIVE empfiehlt für das MOVIDRIVE® B den Einbaugeber EI76.

Die binären Eingänge (Zählereingänge) des MOVIDRIVE® B haben die folgenden technischen Daten:

Binäre Eingänge	
Gebersignale (zwei Spuren)	Spur A und Spur B
Phasenlage	90° ± 20°
Tastverhältnis	1:1 ±20 %
Max. Impulsfrequenz	350 Hz
Anschluss Spur A	MOVIDRIVE® B: Klemme X13:3 (DI02)
Anschluss Spur B	MOVIDRIVE® B: Klemme X13:4 (DI03)
Bezugspotenzial	DCOM

Die Einfach-Positionierung mit MOVIDRIVE® B setzt voraus, dass sich der Umrichter in einer der folgenden Betriebsarten befindet:

- VFC (ohne Rückführung)
- U/f-Kennlinie

	HINWEIS Wenn sich im MOVIDRIVE® B eine Geberoption befindet, können die Spursignale von den binären Eingängen (Zählereingänge) nicht ausgewertet werden. • Betreiben Sie das MOVIDRIVE® B **ohne** Geberoption.

Um einen Einbaugeber zu verwenden, gehen sie folgendermaßen vor:

1. Verbinden Sie den Geber mit den digitalen Eingängen des MOVIDRIVE® B. Verwenden Sie dazu die Klemmen X13:3 (DI02) und X13:4 (DI03).
2. Stellen Sie die folgenden Parameter (Indizes) ein:
   - *P601 Binäreingang DI02* auf "IPOS-Eingang" (Index 8336 auf "16")
   - *P602 Binäreingang DI03* auf "IPOS-Eingang" (Index 8337 auf "16")
3. Setzen Sie den Index 8845 auf "1", um die Einfach-Positionierung zu aktivieren.

Die Position steht in der IPOS-Variablen H511 (ActPos_Mot) und hat nach dem Einschalten der Netzspannung immer den Wert "0".

Eine Referenzfahrt kann nur durch ein IPOS$^{plus®}$-Programm ausgeführt werden.

7.4 Wegerfassung mit MOVITRAC® B

Die binären Eingänge (Zählereingänge) des MOVITRAC® B haben die folgenden technischen Daten:

Binäre Eingänge	
Gebersignale (zwei Spuren)	Spur A und Spur B
Phasenlage	90° ± 20°
Tastverhältnis	1:1 ±20 %
Max. Impulsfrequenz	120 kHz
Anschluss Spur A	MOVITRAC® B: Klemme X12:5 (DI04)
Anschluss Spur B	MOVITRAC® B: Klemme X12:4 (DI03)
Bezugspotenzial	GND auf PE-Potenzial

SEW-EURODRIVE empfiehlt für das MOVITRAC® B den Einbaugeber EI7C.

Um den Einbaugeber zu verwenden, gehen sie folgendermaßen vor:

1. Verbinden Sie den Geber mit den digitalen Eingängen des MOVITRAC® B. Verwenden Sie dazu die Klemmen X12:5 (DI04) und X12:4 (DI03).
2. Stellen Sie die folgenden Parameter (Indizes) ein:
   - *P602 Binäreingang DI03* auf "IPOS-Eingang" (Index 8338 auf "16"),
   - *P603 Binäreingang DI04* auf "IPOS-Eingang" (Index 8339 auf "16"),
3. Setzen Sie den Index 8845 auf "1", um die Einfach-Positionierung zu aktivieren.

Die Position steht in der IPOS-Variablen H511 (ActPos_Mot) und hat nach dem Einschalten der Netzspannung immer den Wert "0".

Eine Referenzfahrt kann nur durch ein IPOS$^{plus®}$-Programm ausgeführt werden.

	HINWEIS Wenn Sie für die Positionierung ein Applikationsmodul einsetzen möchten, beziehen Sie sich auf das Handbuch "MOVITRAC® B Applikatonsmodul Einfach-Positionierung".

7.5 Wegerfassung mit MQx

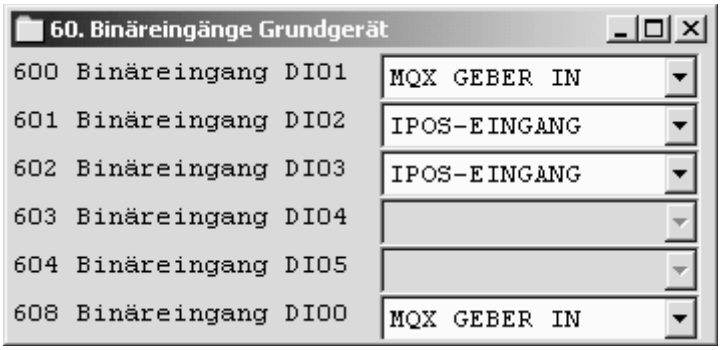
7.5.1 Näherungsgeberauswertung

Mit den MQx-Modulen lässt sich eine einfache Positionierung realisieren, die auf dem Näherungsgebersystem NV26 basiert. Die beiden Initiatoren liefern 24 Winkelinformationen pro Umdrehung, die von der MQx gezählt werden und auf der IPOS$^{plus®}$-Variablen H511 (ActPos_Mot) als Positionswert zur Verfügung gestellt werden. Die Position kann somit in IPOS$^{plus®}$ verarbeitet werden. Das ermöglicht z. B. das Stillsetzen des Antriebs auf einer definierten Position. Für das genaue Erreichen der Position ist eine Schleichfahrt erforderlich. Eine Lageregelung ist nicht möglich.

	HINWEIS Bei Näherungsgeberauswertung kann nur ein MOVIMOT® an die MQx-Feldbusschnittstelle angeschlossen werden. Für die Steuerung des MOVIMOT® ist nur der Movcom-Befehl zulässig.

7.5.2 Klemmenbelegung DI0 und DI1

Die Näherungsgeber-Auswertung wird durch das Umparametrieren der beiden digitalen Eingänge DI0 und DI1 (P600 und P608) auf „MQX GEBER IN“ erreicht. Das Umparametrieren kann mit Hilfe der Parametrieroberfläche MOVITOOLS® MotionsStudio oder im Initialisierungsteil von IPOS$^{plus®}$ erfolgen. IPOS$^{plus®}$ kann beliebige MQx-Parameter mit dem Movilink-Befehl auf der Adresse 253 ansprechen. Die Indizes der Eingänge sind 8844_{dez} für DI0 (P608) und 8335_{dez} für DI1 (P600). Mit zwei Schreibzugriffen wird der Wert 32 auf die beiden Indizes übertragen. Die Eingänge werden nach Werkseinstellung mit 4 ms gefiltert. Die Klemmenbelegung „MQX GEBER IN“ schaltet diese Filterung für die Näherungsgeberauswertung ab.



480483467

7.5.3 Wegerfassung mit Einbaugeber

Die binären Eingänge (Zählereingänge) des MQx haben die folgenden technischen Daten:

Binäre Eingänge	
Gebersignale (zwei Spuren)	Spur A und Spur B
Phasenlage	90° ± 20°
Tastverhältnis	1:1 ±20 %
Max. Impulsfrequenz	4 kHz
Anschluss Spur A	DI0
Anschluss Spur B	DI1

SEW-EURODRIVE empfiehlt für das MQx-Modul den Einbaugeber EI76.

Um den Einbaugeber zu verwenden, gehen sie folgendermaßen vor:

1. Verbinden Sie den Geber mit den digitalen Eingängen des MQx-Modul. Verwenden Sie dazu die Eingänge DI0 und DI1.
2. Stellen Sie die folgenden Parameter (Indizes) ein:
   - *P608 Binäreingang DI0* auf "MQX GEBER IN" (Index 8844 auf "32"),
   - *P600 Binäreingang DI1* auf "MQX GEBER IN" (Index 8335 auf "32"),

Die Position steht in der IPOS-Variablen H511 (ActPos_Mot) und hat nach dem Einschalten der Netzspannung immer den Wert "0".

Eine Referenzfahrt kann nur durch ein IPOS$^{plus®}$-Programm ausgeführt werden.

7.5.4 Geberüberwachung

Die beiden Geberleitungen werden auf Drahtbruch überwacht. Die Überwachung erfolgt nur bei freigegebenem MOVIMOT®. Falls auf mindestens einer der beiden Geberleitungen für 1 s kein Signalwechsel erfolgt, wird ein Drahtbruch detektiert und durch den Fehler 14 angezeigt. Das MOVIMOT® wird stillgesetzt und kann erst nach Reset der MQx wieder gestartet werden. Die aktuelle Position geht hierbei verloren und es ist eine erneute Referenzierung notwendig. Die Geberüberwachung kann über Parameter P504 (Geberüberwachung Motor) ein- bzw. ausgeschaltet werden.

7.5.5 Abspeichern der Ist-Position

Mit einem MOVLNK-Befehl auf die Adresse 253 lässt sich zu jedem beliebigen Zeitpunkt die Ist-Position auf eine Variable im Bereich H0 ... H127 spannungsausfallsicher speichern. Empfehlenswert ist das Speichern bei Erreichen einer neuen Ist-Position. Durch diese Maßnahme ist nach aus- und einschalten des Geräts keine erneute Referenzierung mehr notwendig. Eine Referenzierung ist nach wie vor notwendig, wenn die Spannung der MQx während eines laufenden Positioniervorgangs abgeschaltet wird. Das Modul ist für 10 Milliarden Schreibzyklen ausgelegt.

7.5.6 Counter

Die MQx-Module besitzen einen Counter, der wahlweise auf DI0 oder DI1 gelegt werden kann. Gezählt werden die positiven Flanken bis zu einer maximalen Eingangsfrequenz von 4 kHz. Für die Aktivierung der Counterfunktion wird der entsprechende Eingang auf „MQX GEBER IN“ umprogrammiert. Die Umprogrammierung bewirkt gleichzeitig das Ausschalten der Eingangsfilterung. Das Eingangssignal für den Counter muss deswegen störungsfrei und prellfrei sein. Der Wert des Counters wird in Variable H511 geschrieben.

	HINWEIS Wenn beide Eingänge DI0 und DI1 auf „MQX GEBER IN“ programmiert werden, wird automatisch die Näherungsgeberauswertung aktiviert, die Counterfunktion ist dann ausgeschaltet.

7.5.7 Anschluss der Einbaugeber

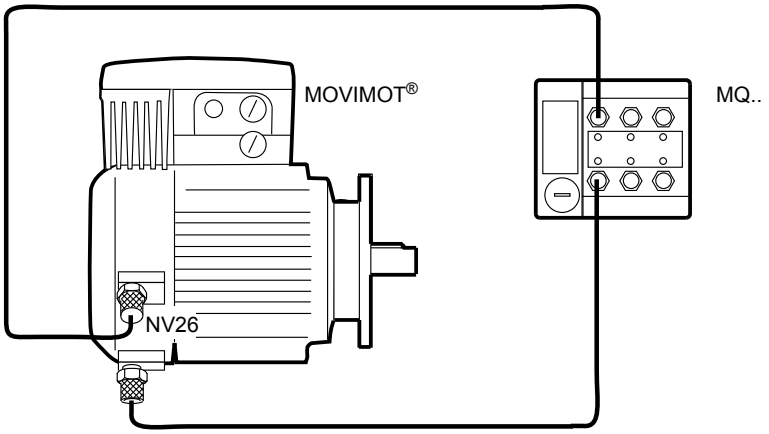
Die Verkabelung der Einbaugeber kann mit den LEDs der MQx geprüft werden. Bei langsam drehendem Motor blinken die LEDs der Eingänge DI0 und DI1.

Anschluss Näherungsgeber NV26

Um die Einbaugeber NV26 (Näherungsgeber) anzuschließen, beziehen Sie sich auf das Handbuch "Antriebssystem für dezentrale Installation PROFIBUS-Schnittstellen, -Feldverteiler".

Für die Verdrahtung sind zwei einfache Sensorkabel, 4polig, geschirmt mit Stecker und Buchse in M12 notwendig. Die Kabel verbinden die Initiatoren des NV26 mit DI0 und DI1 des Busmoduls.

Es wird empfohlen M12-Stecker und Buchsen in Metallausführung zu verwenden und den Schirm beidseitig aufzulegen.



480466315

Falls die MQx-Schnittstelle die Motorposition in H511 in die falsche Richtung zählt, müssen die M12-Stecker an den Eingängen DI0 und DI1 getauscht werden.

Anschluss Inkrementalgeber EI76, ES16

Um die Einbaugeber EI76 und ES16 anzuschließen, beziehen Sie sich auf das Handbuch "Antriebssystem für dezentrale Installation PROFIBUS-Schnittstellen, -Feldverteiler".

8 IPOS$^{plus®}$ und Feldbus

8.1 Einführung

Über Feldbus, SBus oder RS-485 können zwischen einer SPS und den Teilnehmern Ein-/Ausgangssignale, zyklische Prozessdaten und azyklisch Parameter ausgetauscht werden.

Die von den Geräten unterstützten Kommunikations-Schnittstellen entnehmen Sie dem entsprechenden Systemhandbuch.

Eine genauere Beschreibung der Kommunikations-Schnittstellen befindet sich in dem Handbuch "Kommunikation und Feldbus-Geräteprofil“ und in den Handbüchern zu den einzelnen Feldbussen. Um das MOVIDRIVE®-Gerät über Feldbus zu steuern sind im Normalfall folgende Parameter zu ändern:

- P100 Sollwertquelle = zum Bsipiel Feldbus, falls Sollwerte über Feldbus gesendet werden
- P101 Steuerquelle = zum Beispiel Feldbus, falls Steuerwort über Feldbus gesendet wird
- P870-875 Prozessdaten-Konfiguration: Spezifikation, welche Daten über den Bus ausgetauscht werden

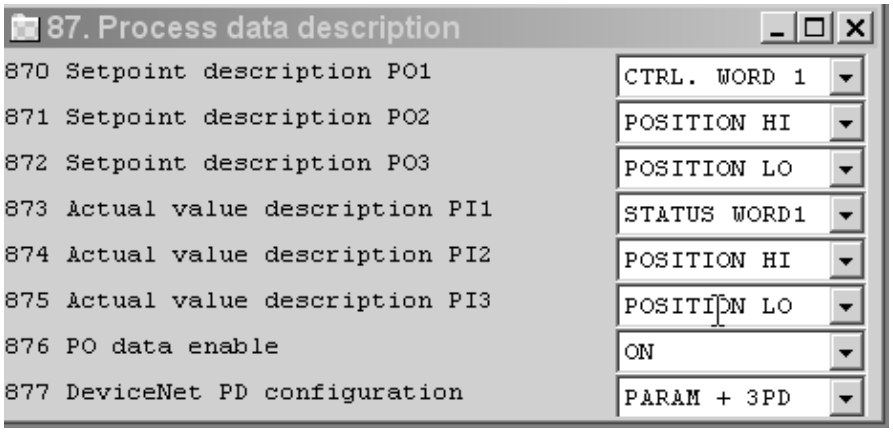
Generell ist der IPOS$^{plus®}$-Programmcode feldbusneutral, d. h. identisch für INTERBUS und PROFIBUS.

Einige Programmbeispiele, die die Verbindung von IPOS$^{plus®}$ und Feldbus aufzeigen, sind bereits in dem Handbuch "Kommunikation und Feldbus-Geräteprofil" beschrieben.

Mit den Parametern P870 ... P877 können bis zu 3 Prozessdatenworte in beide Richtungen eingestellt werden, ohne dass dazu ein IPOS$^{plus®}$-Programm notwendig ist. Feldbusspezifisch können mehr Prozessdatenworte ausgetauscht werden.

Alle Worte ab dem Wort 4 sind fest mit der Bedeutung IPOS$^{plus®}$ PE-DATA bzw. IPOS$^{plus®}$ PA-DATA belegt. Die Prozessdaten, die so belegt sind, werden vom Umrichter nicht direkt interpretiert, über die Datenstrukturen des GETSYS- bzw. SETSYS-Befehls kann jedoch auf alle Prozessdaten zugegriffen werden.

Prozessdatenbelegung mit 3 Worten:



477958795

8.2 Binäre Eingänge und Ausgänge

Ist in einem MOVIDRIVE® keine DIO und keine DIP gesteckt, können die Bits im Steuerwort 2 / Statuswort 2 im IPOS$^{plus®}$-Programm folgendermaßen angesprochen werden:

- direkt mit den symbolischen Namen DI10 ... DI17 bzw. DO10 ... DO17
- indirekt mit GETSYS lesen und mit SETSYS schreiben

Man spricht dann davon, dass über virtuelle Klemmen binäre Ein- und Ausgänge emuliert werden. Die Bedeutung der Klemmen kann über die Parameter P610 ... P617 bzw. P630...P637 eingestellt werden.

8.2.1 Feldbus-Schnittstelle, DIO und DIP

Sind in einem MOVIDRIVE® zusätzlich eine DIO oder DIP gesteckt, verschiebt sich die Einteilung der Klemmenbelegung wie bei den betreffenden Systemvariablen (s.u.) beschrieben und der Zugriff auf die Prozessdaten ist nur noch über GETSYS und SETSYS möglich:

Gerät	Ausgänge	Eingänge
MOVIDRIVE® A	H481 StdOutpIPOS H480 OptOutpIPOS H482 OutpLevel (nur Lesen)	H483 InpLevel
MOVIDRIVE® B	H481 StdOutpIPOS H480 OptOutpIPOS H521 OutpLevel (nur Lesen)	H520 InpLevel

8.3 Zyklische Prozessdaten

Das Lesen und Schreiben der zyklischen Prozessdaten erfolgt in einer Zeitscheibe von 5 ms.

8.3.1 Zyklische voreingestellte Prozessdaten

Wird mit den Parametern P870 ... P875 eine Größe wie z. B. DREHZAHL eingestellt, wird das Prozessdatum direkt mit einer internen Größe verknüpft.

Der Antrieb empfängt ein Doppelwort als Positions-Sollwert. Dieser Wert wird vom MOVIDRIVE® auf die Variable SetpPosBus H499 kopiert und wenn P916 Rampenform = BUSRAMPE gesetzt ist, automatisch als Lagesollwert verwendet. Bei P916 Rampenform = LINEAR, SINUS, QUADRATISCH oder RUCKBEGRENZT kann im Anwenderprogramm der Sollwert weiterverarbeitet werden oder mit einem der folgenden Befehle direkt auf eine Zielposition H492 TargetPos oder H454 ModTagPos kopiert werden.

- TargetPos = SetpPosBus; (Compiler)
- SET H492 = H499 (Assembler)

HINWEIS

Das im Beispiel vom Antrieb an die SPS gesendete Doppelwort mit dem Positions-Istwert ist immer diejenige der Positionen H509 ... H511, des in P941 Quelle Ist-Position ausgewählten Gebers.

8.3.2 Zyklische anwenderspezifische Prozessdaten

Der Anwender hat die Möglichkeit die Bedeutung der zyklischen Prozessdaten selbst zu wählen. Hierzu ist die Prozessdaten-Konfiguration mit der Parametereinstellung PO-Data für die Ausgangsdaten bzw. PI-Data für die Eingangsdaten zu belegen. Die Prozessausgangsdaten werden dann nicht mehr unmittelbar vom MOVIDRIVE® ausgewertet, sondern müssen mit dem Befehl GETSYS (PO-Data) und SETSYS (PI-Data) auf IPOS$^{plus®}$-Variablen gelegt werden. Die Dekodierung der Variablen erfolgt im IPOS$^{plus®}$-Programm. Auf diese Weise kann der Anwender z. B. Positions-Sollwerte in Anwendereinheiten (z. B. Motorumdrehungen) übergeben, indem der vom Feldbus übertragene Wert durch Multiplikation bzw. Division skaliert wird, bevor dieser zur Positionierung herangezogen wird.

Beispiel

Es sollen 6 Prozessdaten mit anwenderspezifischer Bedeutung übertragen werden (P870-877 = IPOS PE-DATA bzw. IPOS PA_DATA). Auf Ausgangswort 2, 3 überträgt die SPS den Positions-Sollwert an den Antrieb, auf Eingangswort 3 sendet der Antrieb die Ist-Position im Modulo-Format 1/10° (0,0° ... 360,0).

Compiler

```
#include <const.h>
// Datenstrukturen Prozessdaten
GSPODATA10 tPA;           //Ausgangsdaten (SPS ->  Antrieb)
SSPIDATA10 tPE;           //Eingangsdaten (Antrieb ->  SPS)
/*=============================================
Hauptfunktion (IPOS-Eintrittsfunktion)
=============================================*/
main()
{
   /*-------------------------------------
   Initialisierung
   -------------------------------------*/
   // Feldbusvariablen für Getsys und Setsys-Befehle initialisieren
   tPA.BusType = GS_BT_FBUS;
   //Prozessdatenbetrieb über Feldbus-Schnittstelle s.o.
   tPA.Len = tPE.Len = 6; //PD-Länge 6 Worte
   /*-------------------------------------
   Hauptprogramm-Schleife
   -------------------------------------*/
   while(1)
   {
      // PA Daten einlesen
      _GetSys( tPA, GS_PODATA );
      // Doppelwort 2,3 auf Modulo-Zielposition kopieren
      ModTagPos = (tPA.PO3 & 0xFFFF) + (tPA.PO2 & 0xFFFF)<<16; //PO2,PO3
      // .....
      //Prozesseingangsdaten neu bilden und an SPS senden
      tPE.PI3 = 3600* ModActPos/ 65536; //Istpos in 1/10 Grad auf Wort 3
      _SetSys( SS_PIDATA, tPE );  //PD abschicken
   } // Ende while(1)
} // Ende main==========================================================
```

Assembler

```
    SET     H320  = 3
    SET     H332  = 6
    SET     H321  = 6
M1 :GETSYS  H320  = PO-DATA
    SET     H300  = H324
    AND     H300  & FFFF     hex
    SET     H301  = H323
    AND     H301  & FFFF     hex
    ADD     H300  + H301
    SHL     H300  << 16
    SET     H354  = H300
    SET     H300  = 3600
    MUL     H300  * H355
    DIV     H300  / 65536
    SET     H335  = H300
    SETSYS  PI-DATA         = H332
    JMP     UNCONDITIONED  , M1
```

8.4 Azyklische Kommunikation

Das MOVIDRIVE® unterstützt bei jedem Feldbus die Möglichkeit, alle Parameter, Variablen, Kurvenscheiben und das IPOS$^{plus®}$-Programm über eine azyklische Kommunikation (auch Parameterkanal oder Parameter-Service genannt) zu lesen und zu schreiben. Dazu sind kein IPOS$^{plus®}$-Programm und keine Parametereinstellungen notwendig.

Der Zugriff auf die Daten im Umrichter erfolgt über Index-Adressierung. Den Index eines Parameters können Sie Parameter-Verzeichnis entnehmen oder durch Drücken von STRG-F1 im Eingabefeld des Parameters.

Der Index einer H-Variable ist die Nummer der Variable plus 11000 (Beispiel: H34 hat den Index 11034).

Die Bearbeitung eines Parameter-Service erfolgt zum Beispiel beim MOVIDRIVE® B in einer Zeitscheibe von 5 ms.

8.5 Besonderheiten bei der Kommunikation über SBus

Wenn an Stelle von Feldbus der SBus als Datenquelle für Steuerung und Sollwert eingestellt ist, so ist die gleiche Funktionalität wie über Feldbus gegeben, nur erfolgt das Lesen und Schreiben von zyklischen Prozessdaten in einer Zeitscheibe von 1 ms.

Weiterhin ist es möglich, mit einem IPOS$^{plus®}$-Programm über den SBus weitere zyklische und azyklische Telegramme zu empfangen oder auch zu senden. Siehe dazu MOVILINK und SCOM in den Kapiteln "Compiler - Funktionen" und "Assembler - Befehle".

8.6 Besonderheiten bei der Kommunikation über RS-485

Wegen der geringen Übertragungsgeschwindigkeit sollte bei MOVIDRIVE® und MOVITRAC® B die Nutzung der RS-485-Schnittstelle auf azyklische (Parameter-) Kommunikation beschränkt werden.

8.7 Feldbus-Steuerworte und Feldbus-Statusworte

Die Belegung der Steuer- und Statusworte ist ausführlich im Kapitel "SEW-Geräteprofil" im Handbuch "Kommunikation und Feldbus-Geräteprofil" beschrieben.

9 IPOS^plus®^ und synchronisierte Bewegungen

9.1 Einführung

Für MOVIDRIVE® und MOVITRAC B ist ein Master-Slave-Betrieb möglich.

Darüberhinaus ist es mit MOVIDRIVE® möglich mehrere Geräte noch genauer zu synchronisieren, z. B. für Anwendungen mit folgenden Funktionen:

- Mechanische Achse durch mehrere Antriebe angetrieben (z. B: Gantry-Portale, Mehrsäulenhubwerke). Hinweis: Beachten Sie bei der Projektierung in diesem Fall auch das "Handbuch MOVDRIVE® Mehrmotorenantriebe").
- Drehzahl oder Position einer Slave-Achse aus der Position einer Master-Achse abgeleitet (Elektronische Welle, elektronische Kurvenscheibe).

Für beide Anwendungsfälle bietet das MOVIDRIVE® vorbereitete Hardware- oder Software-Funktionen an, die vom IPOS^plus®^-Anwenderprogramm aktiviert werden können. Diese werden in den nachfolgenden Abschnitten vorgestellt.

9.2 Drehzahlgleichlauf über Master-Slave-Funktion

Über die Parameter P750 und P751 kann für MOVIDRIVE® und MOVITRAC® B ein einfacher Drehzahlgleichlauf aktiviert werden, ohne dass dazu ein IPOS^plus®^-Programm oder ein Technologiegerät nötig ist.

Ein typisches Anwendungsbeispiel ist der Gleichlauf von 2 Förderbändern. Weitere Informationen finden Sie im Systemhandbuch.

9.3 Synchronlauf mit DRS-Optionskarte

	HINWEIS Für den Einsatz der Synchronlaufkarte DRS11 ist prinzipiell kein IPOS^plus®^-Programm notwendig.

Mit der Synchronlaufkarte DRS11 können mehrere Achsen winkelsynchron zueinander oder in einem einstellbaren Proportionalverhältnis betrieben werden (elektronisches Getriebe). Dabei wird zwischen Master-Antrieb und Slave-Antrieb unterschieden. Der Master-Antrieb, der für die Positionsvorgabe eines oder mehrerer Slave-Antriebe zuständig ist, kann ebenso ein Inkrementalgeber sein. Der oder die Slave-Antriebe folgen den vorgegebenen Master-Positionen.

Die Grundlage für den Synchronlauf ist der ständige Vergleich zwischen Master- und Slave-Positionen. Das System ermittelt die Differenz der Weginformationen von Master und Slave und speichert diesen Wert in Form von Inkrementalgebersignalen in einem für den Anwender nicht zugänglichen, internen Differenzzähler. In Abhängigkeit dieser Differenz werden binäre Meldungen, wie "DRS SLAVE IN POS", "DRS SCHLEPPFEHLER", "DRS VORWARNUNG", und "MASTER STILLSTAND" gesetzt. Dieser Zähler wird in Abhängigkeit des eingestellten Betriebs-Modes unterschiedlich ausgewertet. Im Synchronlauf dient der interne Differenzzähler zur Ausregelung auf Winkelversatz zwischen Slave und Master zu 0.

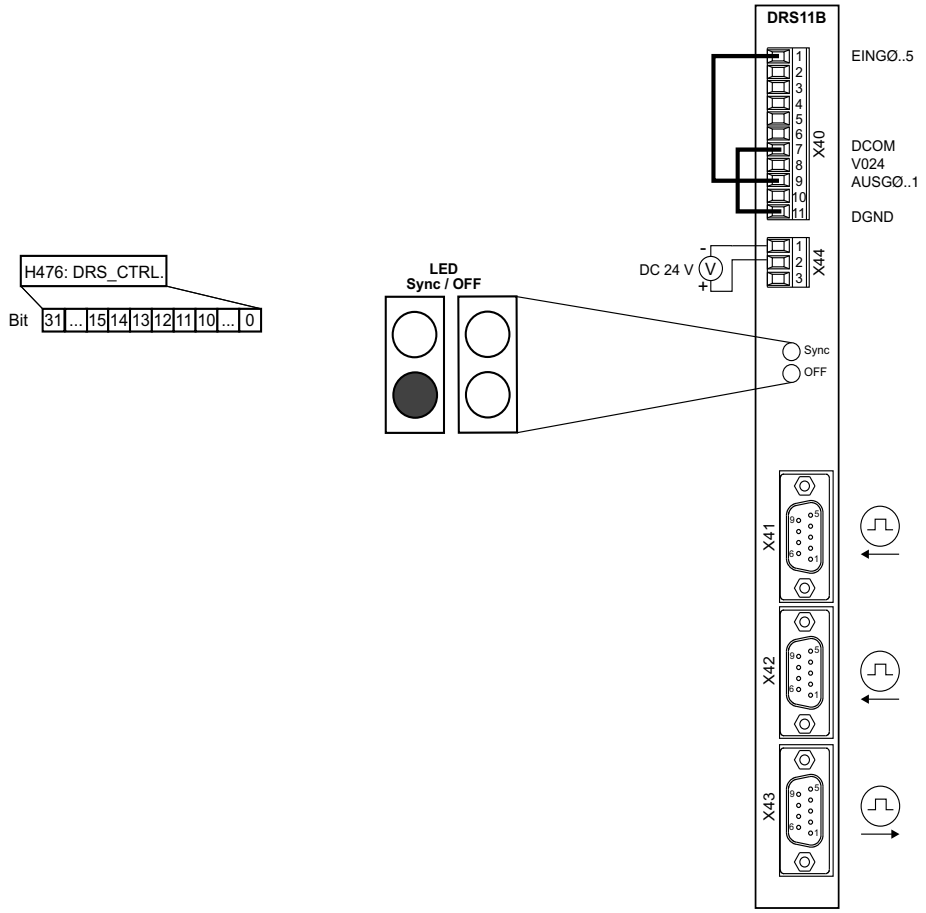
Die Steuerung der DRS erfolgt über die Variablen H473, H476, H477 und H484 (siehe auch Kap. "Übersicht über die Systemvariablen"). Nachfolgend wird beschrieben, wie sich die DRS aus dem IPOS^plus®^-Programm heraus ansprechen lässt.

Weiterführende Information können Sie dem "Handbuch Synchronlaufkarte Typ DRS11" entnehmen.

9.3.1 Aktivieren und Deaktivieren der Funktion Freilauf

Über die Systemvariable H476 DRS CTRL. können die zwei programmierbaren Ausgänge der DRS11 gesetzt und zurückgesetzt werden.

Funktion Freilauf:



478829067

Voraussetzung

Über eine Kabelverbindung der Klemme X40:9 (AUSG0) auf X40:1 (Freilauf) kann die DRS11 über IPOS$^{plus®}$ in den Freilauf geschaltet werden.

Befehlsfolge Assembler

Setzen von AUSG0 und damit dem DRS-Eingang "Freilauf": im Freilauf leuchtet rote LED an DRS-Karte.

```
BSET H476.0 = 1
```

Zurückschalten in die Funktion Synchronlauf: im Synchronlauf ist rote LED aus.

```
BCLR H476.0 = 0
```

Zusammen über BCLR H476.0 schreiben.

Befehlsfolge Compiler

```
_BitSet( 476, 0 );
_BitClear( 476, 0 );
```

9.3.2 DRS11B Nullpunkt setzen

Mittels IPOS$^{plus®}$ kann der Winkelversatz der DRS11 zurückgesetzt werden, ohne dass es externer Signale bedarf.

Dazu wird die Systemvariable H484 CTRL. WORD verwendet.

Das Zurücksetzen der Winkeldifferenz zwischen Master und Slave kann an den Status-LEDs der DRS-Karte beobachtet werden.

rot = Freilauf

grün = DRS nicht in Position (Winkelversatz ist größer als P514) Zähler LED-Anzeige

Befehlsfolge Assembler

```
BSET H484.22 = 1 Setzen der DRS11 in die Fkt. 'Nullpunkt Setzen'
WAIT 15 ms       DRS-spezifische Wartezeit von 15 ms
BCLR H484.22 = 0 Zurücksetzen der Funktion 'Nullpunkt setzen'
```

Befehlsfolge Compiler

```
_BitSet( 484,22 );
_Wait( 15 );
_BitClear( 484,22 );
```

Beispiel zur Steuerung

- Mittels Eingang **DI10** soll der Antrieb in den **Freilauf** geschaltet werden. Die Eingänge DI10 ... DI17 können physikalische Klemmen auf der DIO11A oder DIP11A sein oder virtuelle Klemmen im Feldbussteuerwort 2.
  - DI10 = 1 Freilauf aktiviert
  - DI10 = 0 Freilauf deaktiviert, Antrieb fährt im Synchronlauf
- Über den Eingang **DI11** wird der aktuelle Winkelversatz gelöscht (DRS **Nullpunkt setzen**).

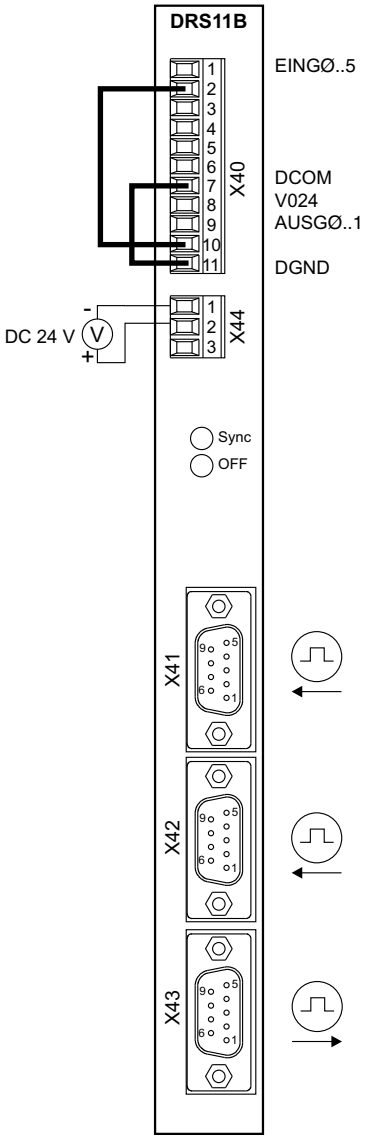
Programmbeispiel mit IPOS^plus®-Compiler

```
/*=================================================================
IPOS-Quelldatei
=================================================================*/
#include <const.h>
#include <io.h>
/*----- Definition Eingänge -------------------------------------*/
#define E_Freilauf         DI10 // Eingang DI10
#define E_NullpunktSetzen  DI11 // Eingang DI11
/*----- Definition Ausgänge -------------------------------------*/
#define A_DRS_AUSG0 0 // Ausgang DRS X40:9
/*----- Definition Steuerbits im IPOS Control Wort ---------------*/
#define _DRS_NullpunktSetzen  22 // Bit 22
*=================================================================
Unterprogramme
=================================================================*/
Freilauf_Ein()
{
   /* Durch die externe Brücke zwischen X40:9 und X40:0 wird durch Setzen
      des Ausgangs X40:9 der Freilauf aktiviert. */
   _BitSet( DRS_Ctrl, A_DRS_AUSG0 );
}
/*=================================================================*/
Freilauf_Aus ()
{
   /* Durch die externe Brücke zwischen X40:9 und X40:0 wird durch Löschen
      des Ausgangs X40:9 der Freilauf deaktiviert. */
   _BitSet( DRS_Ctrl, A_DRS_AUSG0 );
}
/*=================================================================*/
DRS_Nullpunkt()
{
   _BitSet( ControlWord, _DRS_NullpunktSetzen );
   // Nullpunkt setzen über Control Word
   _Wait( 15 ); // Reaktionszeit in ms
   _BitClear( ControlWord, _DRS_NullpunktSetzen ); // Bit wieder löschen
}
/*=================================================================
Hauptfunktion (IPOS-Eintrittsfunktion)
=================================================================*/
main()
{
   if( E_Freilauf )  // Der Eingang E_Freilauf (hier DI10)
      Freilauf_Ein(); // bewirkt die Umschaltung zwischen
   else  // Freilauf und Synchronlauf
      Freilauf_Aus();

   if( E_NullpunktSetzen ) // Die Funktion "Nullpunkt Setzen"
      DRS_Nullpunkt();  // wird aufgerufen
}
```

9.3.3 Aktivieren und Deaktivieren der Funktion Offset

Über die Systemvariable DRS CTRL. H476 können die zwei programmierbaren Ausgänge der DRS11 gesetzt und zurückgesetzt werden.

Verdrahtungsbeispiel Offset:



478833419

Durch den Befehl BSET H476.1 wird der Ausgang X40:10 gesetzt. Durch die Brücke von X40:10 zu X40:2 wird somit ein Signal auf X40:2 geschaltet und die Funktion "Offset" wird aktiviert.

Voraussetzung

Über eine Kabelverbindung der Klemme X40:10 auf X40:2 (Offset1) kann über IPOS$^{plus®}$ ein Positionsoffset aufgeschaltet werden.

Befehlsfolge Assembler

Setzten des Antriebs in die Funktion "Offset1": Slave-Antrieb versetzt seine Position zum Master um den Wert des Offset1.

```
BSET H476.1 = 1
```

Zurückschalten der Ausgangsklemme: Slave-Antrieb fährt wieder auf die vorherige Position zum Master.

```
BCLR H476.1 = 0
```

Zusammen über BCLR H476.0 schreiben.

Befehlsfolge Compiler

```
_BitSet( 476, 1 );
_BitClear( 476, 1 );
```

9.3.4 Umschaltung zwischen Positionierung und Synchronlauf

Die Funktion "Freilauf" ermöglicht ein drehzahlgeregeltes Abkoppeln vom Synchronlauf. Allerdings kann der Antrieb in dieser Funktion nicht positioniergeregelt eine feste Position anfahren.

Dazu ist die Umschaltung in die Betriebsart Positionierung (IPOS$^{plus®}$) notwendig.

Im folgenden Programmbeispiel kann über die Eingangsklemme DI12 zwischen den Betriebsarten Synchronlauf und Positionierung umgeschaltet werden.

Dabei ist folgende wichtige Randbedingung zu beachten:

i	**HINWEIS** **Eine Umschaltung von Synchronlauf auf Positionierung kann während der Fahrt des Antriebs nur in den Betriebsarten CFC oder SERVO erfolgen.** Bei den VFC-Betriebsarten muss sich das MOVIDRIVE® im Zustand Reglersperre befinden.

Das folgende Beispielprogramm behandelt die Umschaltung zwischen den Betriebsarten CFC & IPOS und CFC & SYNC mit SETSYS.

Dabei gilt (Klemmenfunktion durch IPOS$^{plus®}$-Programm bestimmt Parametrierung aller benutzten Eingänge auf IPOS-Eingang):

- DI10 = 0 kein Freilauf
- DI10 = 1 Freilauf, falls Betriebsart CFC & SYNC eingestellt
- DI11 = 0 keine Funktion
- DI11 = 1 DRS Nullpunkt setzen (Impuls)
- DI12 = 1 Positionierung, Betriebsart CFC & IPOS
- DI12 = 0 Synchronlauf, Betriebsart CFC & SYNC

Die Umschaltung der Betriebsarten erfolgt über den Befehl

```
_SetSys(SS_OPMODE, H)
```

wobei der Wert der Variablen H folgende Bedeutung hat:

SS_OPMODE: Setzen der Betriebsart
H = 11: Betriebsart CFC (Drehzahlregelung),
H = 12: Betriebsart CFC & Momentenregelung,
H = 13: Betriebsart CFC & IPOS (Positionierung),
H = 14: Betriebsart CFC & Synchronlauf (DRS11A),
H = 16: Betriebsart SERVO (Drehzahlregelung),
H = 17: Betriebsart SERVO & Momentenregelung,
H = 18: Betriebsart SERVO & IPOS (Positionierung),
H = 19: Betriebsart SERVO & Synchronlauf (DRS11A)

```
/*=================================================================
IPOS-Quelldatei
=================================================================*/
#include <const.h>
#include <io.h>
/*----- Definition Eingänge -----------------------------------*/
#define E_Freilauf           DI10// Eingang DI10
#define E_NullpunktSetzen    DI11 // Eingang DI11
#define E_Umschalt_Pos_Sync  DI12 // Eingang zur Umschaltung zwischen
// Positionierung und Synchronlauf
// DI 12 = 1 Positionierung /
// DI 12 = 0 Synchronlauf
/*----- Definition Ausgänge -----------------------------------*/
#define A_DRS_AUSG0 0 // Ausgang DRS X40:9
/*----- Definition Steuerbits im IPOS Control Wort --------------*/
#define _Freilauf              1     // Bit  1
#define _DRS_NullpunktSetzen  22     // Bit 22
/*----- Definition Variablen zur Umschaltung zwischen
/*----- Positionierung und Synchronlauf ---*/
#define Betriebsart           H300
#define Zielposition          H0
#define CFC_und_IPOS          13     // Betriebsart CFC & IPOS
#define CFC_und_SYNC          14     // Betriebsart CFC & Synchronlauf
/*=================================================================
Unterprogramme
=================================================================*/
Freilauf_Ein()
{
   /* Durch die externe Brücke zwischen X40:9 und X40:0 wird durch Setzen
      des Ausgangs X40:9 der Freilauf aktiviert. */
   _BitSet( DRS_Ctrl, A_DRS_AUSG0 );
}
/*=================================================================*/
Freilauf_Aus ()
{
   /* Durch die externe Brücke zwischen X40:9 und X40:0 wird durch Löschen
      des Ausgangs X40:9 der Freilauf deaktiviert. */
   _BitSet( DRS_Ctrl, A_DRS_AUSG0 );
}
/*=================================================================*/
DRS_Nullpunkt()
{
   _BitSet( ControlWord, _DRS_NullpunktSetzen );
   // Nullpunkt setzen über Control Word
   _Wait( 15 );  // Reaktionszeit in ms
   _BitClear( ControlWord, _DRS_NullpunktSetzen ); // Bit wieder löschen
}
/*=================================================================*/
Aktiviere_Synchronlauf()
{
   Betriebsart = CFC_und_SYNC;
   _SetSys( SS_OPMODE,Betriebsart );  // Umschaltung Betriebsart
   DRS_Nullpunkt(); // Winkelfehler löschen
}
/*=================================================================*/
Aktiviere_IPOS()
{
   Betriebsart = CFC_und_IPOS;
   _SetSys( SS_OPMODE,Betriebsart );
}
```

```
/*=====================================================================
Hauptfunktion (IPOS-Eintrittsfunktion)
=====================================================================*/
main()
{
   if( E_Freilauf )
      Freilauf_Ein();
   else
      Freilauf_Aus();
   if( E_NullpunktSetzen )
      DRS_Nullpunkt();
   if( E_Umschalt_Pos_Sync )
   {
      Aktiviere_IPOS();
      _GoAbs( GO_NOWAIT,Zielposition );
   }
   else if ( !E_Freilauf )
      Aktiviere_Synchronlauf();
}
```

9.4 Synchronlauf mit Technologie-Option "Interner Synchronlauf"

Beim internen Synchronlauf handelt es sich um eine reine Firmware-Lösung, mit der mehrere Achsen winkelsynchron betrieben werden können. Diese Software-Lösung erwartet lediglich Impulse von einer Master-Einheit. Diese Masterquelle kann entweder der X14-Eingang (physikalischer Leitantrieb) oder eine beliebige IPOS$^{plus®}$-Variable (virtueller Leitantrieb) sein (z. B. in Verbindung mit dem SBus oder virtuellem Geber). Ab der MOVIDRIVE® B-Reihe kann die Quelle der Ist-Position der Achse beliebig gewählt werden (Absolutwertgeber, Streckengeber oder jede beliebige IPOS$^{plus®}$-Variable).

Damit können auch schlupfbehaftete Achsen mit dem internen Synchronlauf synchronisiert werden. Beim MOVIDRIVE® A kann als Quelle Ist-Position nur der Motorgeber verwendet werden.

Im Synchronlauf sind verschiedene Funktionalitäten enthalten. Zum einen kann ein vorgegebener Offset und Einkuppelung wegbezogen gefahren werden. Nach einer vorgegebenen Anzahl Masterinkremente besteht ein Offset zwischen Master- und Slave-Antrieb.

Implementiert ist ebenfalls der von der Synchronlaufkarte DRS11 bekannte Aufholmechanismus (zeitgesteuerter Aufholvorgang). Eine durch den Freilauf entstandene Winkeldifferenz des Slave-Antriebs wird zu null reduziert.

Weiterhin kann auch wegbezogen aufsynchronisiert werden. Der Slave-Antrieb fährt genau nach einer vorgegebenen Anzahl Masterinkremente winkelsynchron mit dem Master-Antrieb (Einkuppelvorgang). Bei dieser Art des Einkuppelns verfährt der Slave-Antrieb mit einer quadratischen Rampe.

9.4.1 Voraussetzungen

Der Synchronlauf ist für das MOVIDRIVE® konzipiert und stellt folgende Anforderungen an das Antriebssystem:

- Geberrückführung
- keine Unterstützung der Optionskarten DRS
- Betriebsart: nicht U/f

- IPOS$^{plus®}$-Variablen ab H360 bis H450 sind für den Synchronlauf reserviert und dürfen vom Anwendungsprogramm nicht benutzt werden, siehe auch Kapitel "Übersicht über die Systemvariablen" (Seite 29).
- Die Steuerung des Synchronlaufs erfolgt über IPOS$^{plus®}$-Variablen innerhalb eines IPOS$^{plus®}$-Programmes. Mit den Variablen des Synchronlaufs (H360 - H450) sind alle Zustände des Synchronlaufs sichtbar und einstellbar.
- Die Inbetriebnahme des ISYNC wird von einer graphischen Benutzeroberfläche unterstützt.
- Slave nicht schlupfbehaftet (nur bei MOVIDRIVE® A).

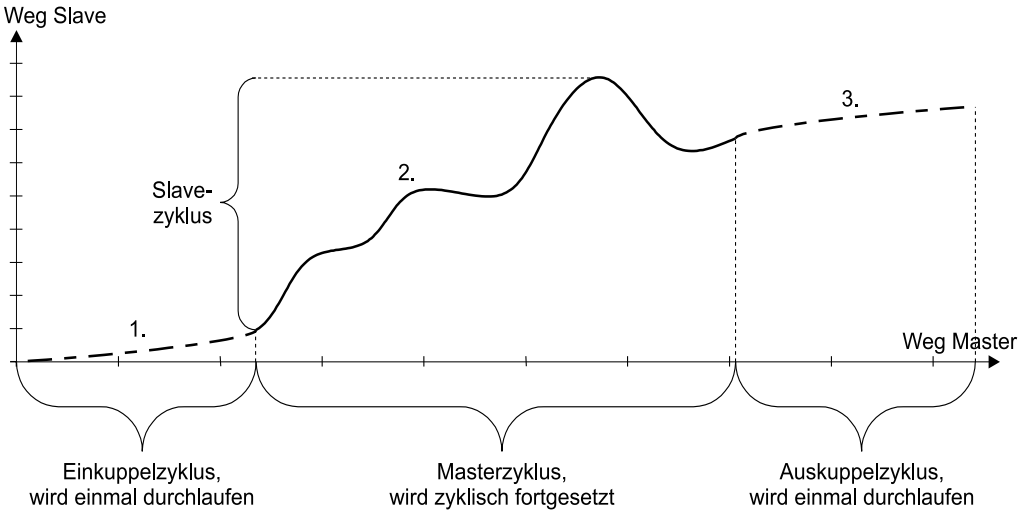
Weiterführende Information können Sie dem "Handbuch Interner Synchronlauf (ISYNC)" entnehmen.

9.5 Synchronlauf mit Technologie-Option "Kurvenscheibe"

Eine Master-Bewegung wird normalerweise als Maschinenwinkel von 0 bis 360 Grad abgebildet. Bezogen auf diesen Maschinenwinkel wird eine Anzahl von Stützstellen (der "Bewegungsplan") definiert. Diese Stützstellen definieren die jeweilige Lage des Slave-Antriebs zum Master.

Der Leitantrieb kann entweder ein physikalisch vorhandener Antrieb sein oder ein virtueller Leitwertgeber. Es ist auch möglich, den Leitgeber über den synchronisierten Systembus (SBus) umzuschalten. Der Bezug der Positionen zwischen Leitantrieb und Folgeantrieb wird häufig in einem 2-dimensionalen Graphen angegeben. In der Horizontalen wird die Position des Master-Antriebs aufgetragen, in der Vertikalen die Position des Slave-Antriebs. Der Positionsumfang in der Horizontalen wird mit Master-Zyklus, der Positionsumfang in der Vertikalen mit Slave-Zyklus bezeichnet.

Kurvenscheibe:



478837771

[1] Einkuppelzyklus, wird einmal durchlaufen
[2] Master-Zyklus, wird zyklisch fortgesetzt
[3] Auskuppelzyklus, wird einmal durchlaufen
[4] Slave-Zyklus
s1 Weg Master
s2 Weg Slave

9.5.1 Voraussetzungen

Die Kurvenscheibe stellt folgende Anforderungen an das Antriebssystem:

- Geberrückführung
- Betriebsart: "CFC" oder "Servo" ... & IPOS
- IPOS$^{plus®}$-Variablen ab H370 bis H450 sind für den Synchronlauf reserviert und sollten vom Anwendungsprogramm nicht benutzt werden (siehe auch Kap.3.2 "Übersicht über die Systemvariablen").
- Die Steuerung erfolgt über IPOS$^{plus®}$-Variablen innerhalb eines IPOS$^{plus®}$-Programms. Mit den Variablen des Synchronlaufs (H370 - H450) sind alle Zustände der Kurvenscheibe sichtbar und einstellbar.

Weiterführende Information können Sie dem "Zusatz zum Systemhandbuch Kurvenscheibe" entnehmen.

	HINWEIS Hat der Bewegungsplan eine konstante Steigung, so ergibt sich ein Winkelsynchronlauf als Sonderfall der Kurvenscheibe

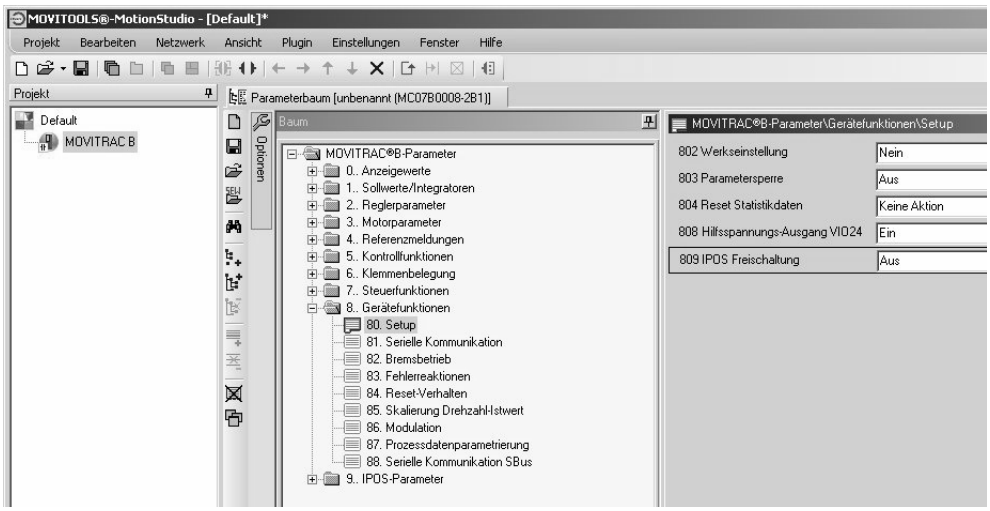
10 Besonderheiten bei IPOS^plus®^ für MOVITRAC® B

10.1 Voraussetzungen

Die Gerätereihe MOVITRAC® B ist sowohl als Standardgerät als auch als Technologiegerät bestellbar. Im Unterschied zu MOVIDRIVE® B muss beim Standardgerät IPOS^plus®^ vor der Programmierung freigeschaltet werden.

Freischaltung IPOS^plus®^ im Standardgerät:

1. Parameterbaum öffnen
2. In der Parametergruppe Gerätefunktionen / Setup muss "IPOS Freischaltung" auf EIN gestellt werden.



769774603

Anschließend kann IPOS^plus®^ wie gewohnt über Compiler oder Assembler programmiert werden.

i	Beim Technologiegerät ist IPOS^plus®^ standardmäßig freigeschaltet.

10.2 Funktionalität

Prinzipiell hat das MOVITRAC® B IPOS$^{plus®}$ die selbe Funktionalität wie MOVIDRIVE® B in der Betriebsart VFC.

Hinzu kommen die folgenden Einschränkungen:

- Einschränkungen gegenüber dem MOVIDRIVE® B in Bezug auf den Befehlsumfang
- Die Größe des IPOS-Programmspeichers beträgt 8kB und damit nur 50 % des MOVIDRIVE® B IPOS$^{plus®}$ Programmspeichers von 16 kB.
- Task 3 darf nicht verwendet werden. Programmcode, der in Task 3 programmiert ist, wird nicht ausgeführt. Hierzu gibt es keine Rückmeldung.
- Nicht unterstützte Funktionen verursachen Fehler 10 IPOS ILOOP.

	Der IPOS Compiler kennt diese Unterschiede nicht, d. h. es wird immer komplette MOVIDRIVE® B Funktionalität angeboten.

Für eine detailierte Beschreibung der Unterschiede beziehen Sie sich auf die folgenden Abschnitte:

- Befehlsübersicht Standardfunktionen (Seite 205)
- Technischen Daten (Seite 25) für MOVITRAC® B
- Wegerfassung (Seite 98) mit MOVITRAC® B

11 Besonderheiten bei IPOS$^{plus®}$ für MQx

11.1 Einleitung

Die Module der MQx-Baureihe ermöglichen, wie die MFx-Module, eine kostengünstige Feldbus-Schnittstelle der MOVIMOT®-Antriebe. Zusätzlich sind die MQx-Module mit Steuerungsfunktionalität ausgestattet, die es Ihnen erlaubt, das Verhalten des Antriebs auf externe Vorgaben über den Feldbus und die integrierten I/Os weitgehend selbst zu bestimmen.

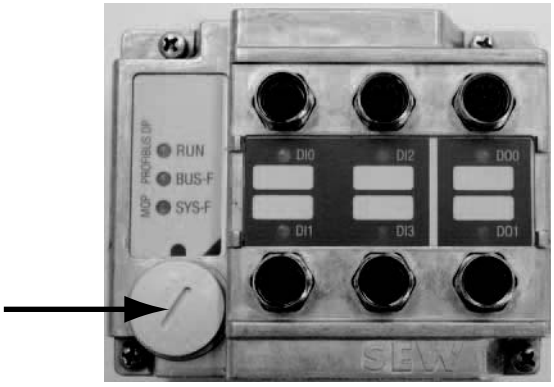
Sie bekommen dadurch die Möglichkeit, beispielweise Sensorsignale direkt in der Feldbus-Schnittstelle zu verarbeiten oder ihr eigenes Kommunikationsprofil über die Feldbus-Schnittstelle zu definieren. In Verbindung mit dem Näherungsgeber NV26 erhalten Sie ein einfaches Positioniersystem, das in Verbindung mit einem MQx-Steuerungsprogramm als Standardkomponente in Anlagen integriert werden kann. In der MQx-Schnittstelle läuft prinzipiell das gleiche IPOS$^{plus®}$ wie im MOVIDRIVE® A. An einigen Stellen ist jedoch nicht der volle Funktionsumfang nutzbar. Diese Unterschiede werden im Folgenden beschrieben.

- Die Parametergruppe P900 IPOS$^{plus®}$-Parameter ist bei der MQx-Schnittstelle nicht implementiert.
- Die MQx-Feldbusschnittstelle stellt genauso wie das MOVIDRIVE® A zwei Tasks mit den gleichen Befehlsabarbeitungszeiten von IPOS$^{plus®}$-Befehlen zur Verfügung.
- Die IPOS$^{plus®}$-Variable H511 (ActPos_Mot) zählt bei Verwendung des NV26-Gebers nicht 4096 Ink/Umd sondern 24 Ink/Umd.
- Es sind nicht alle IPOS$^{plus®}$-Befehle nutzbar.

11.2 Start der Programmier-Tools

Über die Diagnose- und Programmierschnittstelle unter der Verschraubung auf der Vorderseite der MQx-Module haben Sie Zugriff auf die Feldbus-Schnittstelle.

- Verbinden Sie die serielle Schnittstelle Ihres PCs mit der Programmierschnittstelle der MQx. Verwenden Sie dazu die Option UWS21A..



480461579

- Starten Sie die gewünschte Programmieroberfläche aus dem MOVITOOLS® MotionStudio heraus.

  Beziehen Sie sich dazu auf den Abschnitt: "Schritt 1: IPOSplus-Compiler mit MOVITOOLS MotionStudio starten (Seite 143)"

11.3 Ablaufsteuerung

Im Folgenden sind alle IPOS$^{plus®}$-Funktionen die in der MQx benutzt werden können aufgezählt. Eine genaue Beschreibung der Befehle ist in den Compiler-Funktionen und Assembler-Befehle zu finden. Weiterhin erhalten Sie Informationen in der Online-Hilfe des verwendeten Programmier-Tools. Bei Schleifen oder Operatoren gibt es keine Einschränkungen zum MOVIDRIVE® A.

Der Zugriff auf die Feldbus-Prozessdatenpuffer erfolgt über die Befehle _GetSys und _SetSys. Die RS-485-Schnittstelle zum MOVIMOT® kann über die _MovComm Befehle beeinflusst werden.

11.4 Digitale Ein- und Ausgänge

Die digitalen Ausgänge der MQx (nicht bei MQx32. vorhanden) DO0 und DO1 können über die Variable H481 StdOutpIPOS ein- und ausgeschaltet werden. Hierzu müssen die Parameter P620 (DO1) und P628 (DO0) auf IPOS-Ausgang gesetzt werden.

H481

...	3	2	1	0
			DO1	DO0

Die digitalen Eingänge der MQx können über die Variable H483 InputLevel gelesen werden. DI4 und DI5 sind nur bei MQx32. vorhanden. Um die Eingänge als IPOS$^{plus®}$-Eingänge zu nutzen, müssen diese in der Parametergruppe 62. auf IPOS$^{plus®}$-Eingang gesetzt werden.

H483

...	5	4	3	2	1	0
	(DI5)	(DI4)	DI3	DI2	DI1	DI0

11.5 Werte der Variablen DIAG11 beim Fehler IPOS ILLOP

Als allgemeine Fehlermeldung wird bei einem Fehler im IPOS$^{plus®}$-Programm ein IPOS-ILLOP gemeldet.

Zur genauen Fehleranalyse wird eine interne Fehlernummer an die Diagnose-Variable H469 (DIAG11) übergeben. Jeder mögliche Fehler erhält eine eigene Fehlernummer. 999 bedeutet, dass die verwendete Funktion oder ein Argument der Funktion vom MQx-Modul nicht unterstützt wird.

Folgende Fehlernummern können bei der Anwendung des MOVLNK und des MOV-COM Befehls auftreten.

DIAG11	Befehl	Fehlerursache
500	MOVLNK	MOVLNK-Befehl wurde aufgerufen, nachdem die zyklische Kommunikation gestartet wurde. MOVLNK wird durch _MovCommOn-Befehl gesperrt.
501	MOVLNK	Nummer der Variablen H, ab der die gelesenen Daten abgelegt bzw. die zu schreibenden Daten geholt werden, liegt nicht in einem gültigen Bereich (H0 ... H450).
502	MOVLNK	Der Bustyp ist ungültig. Nur 2 = RS-485#2 erlaubt.
503	MOVLNK	Ein nicht erlaubter PDU-Typ wurde im Format eingetragen. Nur azyklische Frames (128 ... 134) erlaubt.
504	MOVCOM	_MovCommDef Befehl wurde aufgerufen, nachdem die zyklische Kommunikation gestartet wurde. _MovCommDef wird durch MovComOn gesperrt.
505	MOVCOM	_MovCommDef Befehl wurde zum 9. Mal aufgerufen. Maximal 8 Kommunikationsbeziehungen möglich.
506	MOVCOM	Der Bustyp ist ungültig. Nur 2 = RS-485#2 erlaubt.
507	MOVCOM	Eine nicht gültige Teilnehmer-Adresse wurde eingetragen. Adresse 253 und 254 nicht erlaubt (0 -252, 255).
508	MOVCOM	Ein nicht erlaubter PDU-Typ wurde im Format eingetragen. Nur zyklische Frames erlaubt (0-6).
509	MOVCOM	Nummer der Variablen H, ab der die gelesenen Daten abgelegt bzw. die zu schreibenden Daten geholt werden, liegt nicht in einem gültigen Bereich. (H0 ... H450).
511	MOVCOM	Teilnehmer unterstützt im Format eingetragenen PDU-Typ nicht.

12 P9xx IPOS-Parameter

GEFAHR!

Quetschgefahr durch unbeabsichtigtes Anlaufen des Motors.
Tod oder schwerste Verletzungen.

- Verhindern Sie unbeabsichtigtes Anlaufen des Motors.
- Beachten Sie, dass eine Veränderung dieser Parameter ohne Kenntnis des eventuell aktiven IPOS$^{plus®}$-Programms unerwartete Verfahrbewegungen und ungewollte Belastungen des mechanischen Antriebsstranges zur Folge haben kann. Die Kenntnis des Handbuches IPOS$^{plus®}$ ist unbedingte Voraussetzung zur Einstellung dieser Parameter.

12.1 P90x IPOS Referenzfahrt

Die Referenzfahrt dient dazu, einen **Maschinennullpunkt** festzulegen, auf den sich alle absoluten Positionierbefehle beziehen. Hierzu sind verschiedene so genannte Referenzfahrtstrategien P903 Referenzfahrttyp (Seite 124) wählbar. Diese definieren entsprechende Verfahrmodi, um z. B. einen Referenznocken zu suchen. Ausgehend von dem durch die Referenzfahrt gefundenen **Referenzpunkt** kann mit P900 Referenzoffset (Seite 123) der Maschinennullpunkt gemäß der Gleichung

Maschinennullpunkt = Referenzpunkt + Referenz-Offset

verschoben werden.

Die Drehzahlen der nach **Referenzfahrttyp** notwendigen Verfahrbewegungen werden durch P901 Referenzdrehzahl 1 (Seite 124) / P902 Referenzdrehzahl 2 (Seite 124) eingestellt.

Anzeigebereich: Anzahl der Inkremente vom Verlassen des Referenznockens bis zum Erreichen des Nullimpulses des in P941 eingestellten Gebers.

Der Wert wird erst nach erfolgter Referenzfahrt angezeigt, idealerweise sollte er die Hälfte der Geberauflösung (nach vierfacher Auswertung) betragen. Gegebenenfalls Nocken verschieben.

12.1.1 P900 Referenzoffset

Einstellbereich: $-(2^{31}-1) - \underline{0} - 2^{31}-1$

Der Referenz-Offset (Nullpunktkorrektur) wird verwendet, um den Maschinennullpunkt festzulegen. Es gilt:

Maschinennullpunkt = Referenzpunkt + Referenz-Offset

Der Referenz-Offset bezieht sich immer auf den über P941 Quelle Istposition (Seite 134) eingestellten Geber.

Dies kann der Motorgeber, ein externer Geber oder der DIP-Geber sein. Die entsprechenden Istpositionen werden in IPOS$^{plus®}$-Variablen angezeigt.

- H509 Istposition DIP-Geber
- H510 Istposition externer Geber
- H511 Istposition Motorgeber

Der Referenz-Offset wird nach erfolgreich beendeter Referenzfahrt aktiv.

	HINWEIS
	Bei einer Referenzfahrt eines Antriebssystems mit Absolutwertgeber (HIPERFACE® oder DIP) werden, je nach eingestellter Quelle Istposition durch die Referenzfahrt P905 Hiperface Offset X15 (Seite 126) / P947 Hiperface Offset X14 (Seite 136) oder DIP Offset P953 Positionsoffset (Seite 139) neu berechnet und überschrieben.

12.1.2 P901 Referenzdrehzahl 1

Einstellbereich: 0 – 200 – 6000 1/min

Mit der Referenzdrehzahl 1 wird die Verfahrdrehzahl für den ersten Teil der Referenzfahrt festgelegt. Zur Drehzahländerung wird immer die Stopprampe t13 verwendet. Die Suchrichtungen während der Referenzfahrt sind durch den entsprechenden Referenzfahrttyp festgelegt. Die Drehzahl wird benutzt, bis der Referenznocken erreicht wird.

12.1.3 P902 Referenzdrehzahl 2

Einstellbereich: 0 – 50 – 6000 1/min

Mit der Referenzdrehzahl 2 wird die Verfahrdrehzahl für den zweiten Teil der Referenzfahrt festgelegt. Zur Drehzahländerung wird immer die Stopprampe t13 verwendet. Die Suchrichtungen während der Referenzfahrt sind durch den entsprechenden Referenzfahrttyp festgelegt. Die Drehzahl wird benutzt, wenn der Referenznocken verlassen wird, bis der 1. Nullimpuls erreicht wurde.

12.1.4 P903 Referenzfahrttyp

Einstellbereich: 0 – 8

Der Referenzfahrttyp legt fest, mit welcher Referenzfahrtstrategie der Maschinennullpunkt einer Anlage festgelegt werden soll.

Mit dieser Einstellung werden auch die Suchrichtung für den Referenznocken in den einzelnen Phasen der Referenzierung festgelegt.

Über den Parameter P904 Referenzierung auf Nullimpuls (Seite 126) wird eingestellt, ob die Referenzfahrt auf den Flankenwechsel des Referenznockens oder den darauffolgenden Nullimpuls des Gebers erfolgt.

Mit Ausnahme des Referenzfahrttyps 8 ist bei allen übrigen Referenzfahrttypen ein **betriebsbereiter** und **freigegebener** Antrieb die Voraussetzung für die Durchführung der Referenzfahrt.

Es stehen auch Typen zur Verfügung, die ohne einen Referenznocken arbeiten können.

- **Typ 0: Linker Nullimpuls**
  - Erste Suchrichtung ist links
  - Referenzpunkt = linker Nullimpuls von aktueller Position
  - Maschinennullpunkt = Referenzpunkt + Referenz-Offset
- **Typ 1: Linkes Ende des Referenznockens**
  - Erste Suchrichtung ist links
  - Referenzpunkt = Erster Nullimpuls oder fallende Flanke links vom Referenznocken
  - Maschinennullpunkt = Referenzpunkt + Referenz-Offset

- **Typ 2: Rechtes Ende des Referenznockens**
  - Erste Suchrichtung ist rechts
  - Referenzpunkt = Erster Nullimpuls oder fallende Flanke rechts vom Referenznocken
  - Maschinennullpunkt = Referenzpunkt + Referenz-Offset
- **Typ 3: Endschalter rechts**
  - Erste Suchrichtung ist rechts
  - Referenzpunkt = Erster Nullimpuls oder fallende Flanke links vom rechten Endschalter
  - Maschinennullpunkt = Referenzpunkt + Referenz-Offset
  - Die Referenzfahrt sollte auf Nullimpuls erfolgen.
- **Typ 4: Endschalter links**
  - Erste Suchrichtung ist links
  - Referenzpunkt = Erster Nullimpuls oder fallende Flanke rechts vom linken Endschalter
  - Maschinennullpunkt = Referenzpunkt + Referenz-Offset
  - Die Referenzfahrt sollte auf Nullimpuls erfolgen
- **Typ 5: Keine Referenzfahrt**
  - Referenzpunkt = Aktuelle Position
  - Maschinennullpunkt = Referenz-Offset
- **Typ 6: Referenznocken bündig zum rechten Endschalter**
  - Erste Suchrichtung ist rechts
  - Referenzpunkt = Erster Nullimpuls oder fallende Flanke links vom Referenznocken
  - Maschinennullpunkt = Referenzpunkt + Referenz-Offset
  - Hinweis: Referenznocken und Endschalter müssen bündig sein!
- **Typ 7: Referenznocken bündig zum linken Endschalter**
  - Erste Suchrichtung ist links
  - Referenzpunkt = Erster Nullimpuls oder fallende Flanke rechts vom Referenznocken
  - Maschinennullpunkt = Referenzpunkt + Referenz-Offset
  - Hinweis: Referenznocken und Endschalter müssen bündig sein!
- **Typ 8: Ohne Freigabe**

  Die Referenzierung kann bei nicht freigegebenem Antrieb erfolgen.
  - Referenzpunkt = Aktuelle Position
  - Maschinennullpunkt = Referenz-Offset

12.1.5 P904 Referenzierung auf Nullimpuls

Einstellbereich: JA / NEIN

- JA: Die Referenzfahrt erfolgt auf den Nullimpuls des eingestellten IPOS$^{plus®}$-Gebers.
- NEIN: Die Referenzfahrt erfolgt auf der fallenden Flanke des Referenznockens.

12.1.6 P905 Hiperface Offset X15

Einstellbereich: $-(2^{31} - 1) - 0 - 2^{31} - 1$

Mit dem Parameter wird der Nullpunkt der Geberanzeige festgelegt.

Dieser Parameter wird verwendet, um den Maschinennullpunkt ohne Referenzfahrt zu definieren. Dazu wird der Geberwert mit dem Offset verrechnet.

- P905 Hiperface-Offset X15 (Seite 126) wirkt auf die Istposition des Motorgebers H511.

  H511 = Geberwert – P905
- P947 Hiperface-Offset X14 (Seite 136) wirkt auf die Istposition des externen Gebers H510.

  H510 = Geberwert – P947

Die Umrechnung der Istposition erfolgt direkt nach der Eingabe der Werte. Ein Hiperface®-Multiturn-Geber muss einmal, ein Hiperface®-Singleturn-Geber muss immer referenziert werden.

Hinweis:

Bei einer Referenzfahrt eines Antriebssystems mit Hiperface®-Geber werden die Hiperface-Offsets (P905 oder P947) je nach eingestellter Quelle Istposition durch die Referenzfahrt neu berechnet und überschrieben.

Es gilt:

- P905 = Geberwert – P900
- P947 = Geberwert – P900

12.1.7 P906 Nockenabstand

Der Parameter *P906* ist ein Anzeigeparameter.

Er zeigt die Anzahl der Inkremente vom Verlassen des Referenznockens bis zum Erreichen des Nullimpulses des in *P941* eingestellten Gebers. Der Wert sollte idealerweise die Hälfte der Geberauflösung (nach vierfacher Auswertung) betragen (gegebenenfalls Nocken verschieben).

Der Wert wird erst nach erfolgter Referenzfahrt angezeigt,

12.2 P91x IPOS$^{plus®}$ Verfahrparameter

12.2.1 P910 Verstärkung X-Regler

Einstellbereich: 0,1 – 0,5 – 32

Einstellwert für den P-Regler des Lageregelkreises von IPOS$^{plus®}$. Hier wird in der Grundeinstellung der Wert von P210 P-Verstärkung Halteregler übernommen.

12.2.2 P911 / 912 Positionier-Rampe 1 / 2

Einstellbereich: 0,01 – 1 – 20 s

Einstellwert für die Rampe, die während des Positioniervorgangs verwendet wird. Für Beschleunigung und Verzögerung wird bei P916 Rampenform (Seite 128) SINUS und QUADRATISCH immer die gleiche Rampe (Positionier-Rampe 1) verwendet. Bei eingestellter Rampenform LINEAR wird die Verzögerung in Abhängigkeit von P917 Rampenmode (Seite 130) eingestellt:

- P917 Rampenmode = Mode 1: Die Verzögerung zur Anfahrt der Zielposition (Zielbremsung) wird nur mit der Positionier-Rampe 2 (P912) durchgeführt. Bei allen anderen Positioniervorgängen wird die Positionier-Rampe 1 (P911) verwendet.
- P917 Rampenmode = Mode 2: Bei Änderung der Fahrgeschwindigkeit während der Fahrt wird zur Verzögerung immer die Positionier-Rampe 2 (P912) verwendet. Zur Beschleunigung wird die Positionier-Rampe 1 (P911) verwendet.

12.2.3 P913 / P914 Verfahrdrehzahl RECHTS / LINKS

Einstellbereich: 0 – 1500 – 6000 1/min

Gibt an, mit welcher Drehzahl positioniert werden soll. Die Einstellung muss an die Maximaldrehzahl des Motors angepasst werden.

i	**HINWEIS**
	P302 Maximaldrehzahl 1 / P312 Maximaldrehzahl 2 begrenzt P913 / P914, deshalb P302 Maximaldrehzahl 1 / P312 Maximaldrehzahl 2 immer größer (ca. 10 %) als P913 / P914 einstellen, sonst könnte ein Schleppfehler auftreten!

12.2.4 P915 Geschwindigkeitsvorsteuerung

Einstellbereich: –199,99 – 0 – 100 – 199,99 %

Mit der Einstellung auf 100 % fährt der Antrieb zeitoptimal mit linearem Geschwindigkeitsprofil. Wird der Wert kleiner als 100 % eingestellt, entsteht beim Positioniervorgang ein größerer Abstand zwischen Sollposition und Istposition (Schleppabstand). Dadurch ergibt sich für den Beschleunigungsvorgang ein “sanftes” Einlaufen in die Zielposition.

i	**HINWEIS**
	Der Parameter P915 wirkt nur bei der Rampenform LINEAR und RUCKBEGRENZT. Für die Rampenformen "SINUS" und "QUADRATISCH" ist die Funktion unwirksam!

12.2.5 P916 Rampenform

Mit diesem Parameter wird die Art der Positionier-Rampe festgelegt. Dies hat Einfluss auf den Drehzahl- bzw. Beschleunigungsverlauf während der Positionierung.

HINWEIS

Ist P702 Motorkategorie auf "Linear" eingestellt, werden die folgenden Rampenformen nicht unterstützt:

- GESCHWINDIGKEITSINTERPOLATION
- POSITIONSINTERPOLATION 12 BIT
- POSITIONSINTERPOLATION 16 BIT

Rampenform	Positionierverhalten
LINEAR	Zeitoptimal, jedoch blockförmiger Beschleunigungsverlauf.
QUADRATISCH	Sanfterer Beschleunigungsverlauf und höherer Momentenbedarf als LINEAR.
SINUS	Sehr sanfter Beschleunigungsverlauf, höherer Momentenbedarf als bei QUADRATISCH.
BUSRAMPE	Einstellung zum Betrieb des Antriebsumrichters mit übergeordneter Steuerung. Die Steuerung generiert einen zyklischen Positions-Sollwert, der direkt auf den Lageregler geschrieben wird. Der Rampengenerator ist deaktiviert. Die von der externen Steuerung zyklisch gesendeten Positionsvorgaben werden linear interpoliert. Zur Konfiguration muss ein Prozess-Ausgangsdatenwort auf "Position HIGH" und ein weiteres auf "Position LOW" parametriert werden.
RUCKBEGRENZT	Die Ruckbegrenzung basiert auf dem Prinzip der Linear-Rampe. Bei der Ruckbegrenzung wird, um den Ruck zu begrenzen, das Drehmoment und somit auch die Beschleunigung trapezförmig aufgebaut. Die Ruckbegrenzung baut das Drehmoment beim Beschleunigen linear mit der Zeit auf, bis der Maximalwert erreicht ist. Ebenso wird das Drehmoment linear mit der Zeit wieder auf Null abgebaut. Dadurch wird erreicht, dass das System kaum zum Schwingen angeregt wird. Der Einstellbereich kann von 0.005 s bis 2 s (P933) gewählt werden. Die Positionierzeit verlängert sich gegenüber der linearen Rampe um die eingestellte Ruckzeit. Die Beschleunigung und das Drehmoment erhöhen sich gegenüber der linearen Rampe nicht.
KURVENSCHEIBE	Aktivierung der Technologiefunktion "Elektronische Kurvenscheibe".
I-SYNCHRONLAUF	Aktivierung der Technologiefunktion "Elektronischer Synchronlauf".
QUERSCHNEIDER	Aktivierung der Technologiefunktion "Querschneider".
GESCHWINDIGKEITSINTERPOLATION	Die von der externen Steuerung zyklisch gesendeten Drehzahlvorgaben werden linear interpoliert. • Drehzahlvorgabe über Prozessdaten: – Stellen Sie die *P888 Synchronisationszeit SBus* auf 5 ms oder 10 ms ein – Stellen Sie die *P100 Sollwertquelle* auf "SBus" oder "Feldbus" ein – Sie müssen ein Prozess-Ausgangsdatenwort auf "Drehzahl" parametrieren • Drehzahlvorgabe über SBus/SCOM-Objekt: – Stellen Sie *P888 Synchronisationszeit SBus* auf 1 ... 10 ms ein – Stellen Sie die *P100 Sollwertquelle* auf "BIPOL./FESTSOLL." ein – Sie dürfen kein Prozess-Ausgangsdatenwort auf "Drehzahl" parametrieren – Richten Sie ein SCOM-Receive-Objekt (mit SCOM-Receive Befehl → Handbuch IPOS$^{plus®}$) mit der Zielvariable *SetpPosBus* (H499) ein

Rampenform	Positionierverhalten
POSITIONSINTERPOLATION 12 BIT	Die von der externen Steuerung zyklisch gesendeten Positionsvorgaben werden interpoliert. Positionsauflösung: 1 Umdrehung entspricht 4096 Inkremente (12 Bit). • Positionsvorgabe über Prozessdaten: – Stellen Sie die *P888 Synchronisationszeit SBus* auf 5 ms oder 10 ms ein – Stellen Sie *P100 Sollwertquelle* auf "SBus" oder "Feldbus" ein – Stellen Sie ein Prozess-Ausgangsdatenwort auf "Position HIGH" und ein weiteres auf "Position LOW" ein. • Positionsvorgabe über SBus/SCOM-Objekt: – Stellen Sie die *P888 Synchronisationszeit SBus* auf 1 ... 10 ms ein – Stellen Sie *P100 Sollwertquelle* auf "BIPOL./FESTSOLL." ein – Sie dürfen kein Prozess-Ausgangsdatenwort auf "Position HIGH" oder auf "Position LOW" parametrieren. – Richten Sie ein SCOM-Receive-Objekt (mit SCOM-Receive Befehl → Handbuch IPOS$^{plus®}$) mit der Zielvariable *SetpPosBus* (H499) ein
POSITIONSINTERPOLATION 16 BIT	Die von der externen Steuerung zyklisch gesendeten Positionsvorgaben werden interpoliert. Positionsauflösung: 1 Umdrehung entspricht 65536 Inkremente (16 Bit). • Positionsvorgabe über Prozessdaten: – Stellen Sie die *P888 Synchronisationszeit SBus* auf 5 ms oder 10 ms ein – Stellen Sie *P100 Sollwertquelle* auf "SBus" oder "Feldbus" ein – Parametrieren Sie ein Prozess-Ausgangsdatenwort auf "Position HIGH" und ein weiteres auf "Position LOW". **Beachten Sie:** Die Positionsauflösung über PE-Datenbelegung ist 4096 Inkremente pro Umdrehung (= 12 Bit). Auf der IPOS$^{plus®}$-Variable H508 steht die auf 16 Bit erweiterte Motorposition zur Verfügung. Die IPOS$^{plus®}$-Variable *ActPos_Mot* (H511) hat die Positionsauflösung von 4096 Inkremente pro Umdrehung (= 12 Bit) • Positionsvorgabe über SBus/SCOM-Objekt: – Stellen Sie die *P888 Synchronisationszeit SBus* auf 1 ... 10 ms ein – Stellen Sie *P100 Sollwertquelle* auf "BIPOL./FESTSOLL." ein – Sie dürfen kein Prozess-Ausgangsdatenwort auf "Position HIGH" oder auf "Position LOW" parametrieren. – Richten Sie ein SCOM-Receive-Objekt (mit SCOM-Receive-Befehl → Handbuch IPOS$^{plus®}$) mit der Zielvariable *SetPosBus* (H499) ein **Beachten Sie:** Die Positionsauflösung über PE-Datenbelegung ist 4096 Inkremente pro Umdrehung (= 12 Bit) Auf der IPOS$^{plus®}$-Variable H508 steht die auf 16 Bit erweiterte Positionsauflösung von 4096 Inkremente pro Umdrehung (= 12 Bit) Die IPOS$^{plus®}$-Variable *ActPos_Mot* (H511) hat die Positionsauflösung von 4096 Inkremente pro Umdrehung (= 12 Bit).

HINWEIS

Beachten Sie bei der Rampenform "POSITIONSINTERPOLATION 16 BIT":

- Die IPOS-Variable H508 wird auch verwendet, wenn S14 = ON eingestellt ist. Die IPOS-Variable H508 liefert nur sinnvolle Werte, wenn
  - DIP-Schalter S14 = "ON" **oder**
  - P916 Rampenform = "Positionsinterpolation 16 Bit"

12.2.6 P917 Rampenmode

Einstellbereich: MODE 1 / MODE 2

Mit diesem Parameter wird die Verwendung der Positionierrampe 2 P912 bei eingestellter Rampenform LINEAR bestimmt.

- P917 = MODE 1: Die Verzögerung zur Anfahrt der Zielposition (Zielbremsung) wird nur mit der Positionierrampe 2 P912 durchgeführt. Bei allen anderen Positioniervorgängen wird die Positionierrampe 1 P911 verwendet. Wurde die Positionsinterpolation 12 oder 16 Bit aktiviert, arbeitet diese in Mode 1 ohne Totzeitkompensation.
- P917 = MODE 2: Bei Änderung der Fahrgeschwindigkeit während der Fahrt wird zur Verzögerung immer die Positionierrampe 2 P912 verwendet. Zur Beschleunigung wird die Positionierrampe 1 P911 verwendet. Wurde die Positionsinterpolation 12 oder 16 Bit aktiviert, schaltet der Mode 2 eine Totzeitkompensation ein.

12.2.7 P918 Quelle Bussollwert

Bei Betrieb mit EtherCAT kann über diesen Parameter die Quelle für den Sollwert in IPOS$^{plus®}$ eingestellt werden. Der voreingestellte Wert ist H499.

12.3 P92x IPOS-Überwachungen

12.3.1 P920 / P921 SW-Endschalter RECHTS / LINKS

Einstellbereich: $-(2^{31}-1)$ – 0 – $2^{31}-1$

Mit den Software-Endschaltern kann der Anwender softwaremäßig die Zielvorgabe eingrenzen, in dem Verfahrbefehle akzeptiert werden. Mit diesen beiden Parametern (Software-Endschalter) werden die Zielbereichsgrenzen festgelegt. Wird *P941 Quelle Istposition* auf Motorgeber oder Externer Geber eingestellt, sind diese erst nach einer durchgeführten Referenzfahrt wirksam. Wird (Seite 134) auf Absolutwertgeber DIP eingestellt, sind diese sofort ohne Referenzfahrt wirksam. Sind die Software-Endschalter wirksam, wird überprüft, ob die Zielposition H492 des aktuellen Verfahrbefehls außerhalb der Software-Endschalter liegt. Liegt die Zielposition außerhalb, wird der Verfahrbefehl nicht ausgeführt. Der Antrieb reagiert entsprechend der eingestellten Fehlerreaktion in P838. Ist *P838 Fehlerreaktion SW-Endschalter* auf "... / Warnung" oder "... / Störung" eingestellt, wird die Fehlermeldung F78 (IPOS SW-Endschalter) erzeugt. Die Überwachung der Software-Endschalter ist nur in den Betriebsarten "...& IPOS" (P700) wirksam.

Ist *P838 Fehlerreaktion SW-Endschalter* auf ".../Störung" programmiert, so ist nach einem Fehler-Reset der Antrieb mit Inkrementalgeber nicht mehr referenziert, ein Antrieb mit Absolutwertgeber bleibt referenziert.

Ist der Antrieb nicht referenziert, sind die Software-Endschalter unwirksam und werden erst nach neuer Referenzierung wieder aktiv.

Ist *P838 Fehlerreaktion SW-Endschalter* auf ".../Warnung" programmiert, bleibt der Antrieb nach dem Reset referenziert. Durch die Massenträgheit der Maschine oder durch Fehlparametrierung des Reglers kann das Ziel überfahren werden. Die Software-Endschalter fangen diesen Fall nicht ab.

Deaktivierung: Beim Endlosverfahren sind beide Parameterwerte auf 0 zu stellen, damit ist die Software-Endschalterfunktion deaktiviert.

12.3.2 P922 Positionsfenster

Einstellbereich: 0 – 50 – 32 767 Inc

Der Parameter definiert einen Entfernungsbereich (Positionsfenster) um die Zielposition eines Verfahr- oder STOPP-Befehls. Befindet sich ein Antrieb im Positionsfenster um die aktuelle Zielposition (H492), gilt der Zustand “Achse in Position” = JA. Die Information “Achse in Position” wird als abschließende Bedingung für wartende Positionierbefehle verwendet und kann als Ausgangsklemmenfunktion weiterverwendet werden.

12.3.3 P923 Schleppfehlerfenster

Einstellbereich: 0 – 5000 – $2^{31}-1$ Inc

Das Schleppfehlerfenster definiert eine zulässige Betragsdifferenz zwischen Soll- und Istposition. Wird diese überschritten, so wird eine Schleppfehlermeldung bzw. Schleppfehlerreaktion ausgelöst. Die Reaktion können Sie mit P834 Reaktion Schleppfehler einstellen.

Deaktivierung: Einstellwert = 0 deaktiviert die Schleppfehlerüberwachung

12.3.4 P924 Erkennung Positionierunterbrechung

Einstellbereich: EIN / AUS

Mit diesem Parameter wird eingestellt, ob die Unterbrechung eines Positioniervorgangs (z. B. Wegnahme der Freigabe) überwacht werden soll. Die Reaktion wird mit *P839 Reaktion Positionierunterbrechung* festgelegt

12.4 P93x IPOS$^{plus®}$-Sonderfunktionen

12.4.1 P930 Override

Einstellbereich: EIN / AUS

Die Overridefunktion ermöglicht die Veränderung der im IPOS$^{plus®}$-Programm programmierten Verfahrgeschwindigkeit der Positioniervorgänge im Bereich von 0 bis 150 % der jeweils programmierten Geschwindigkeit. Hierzu wird der Analogeingang benutzt, wobei 0 bis 150 % dann 0 – 10 V am Analogeingang entsprechen. Der Maximalwert der Geschwindigkeit wird auf jeden Fall durch P302 Maximaldrehzahl 1 / P312 Maximaldrehzahl 2 begrenzt.

12.4.2 P931 IPOS-STW. Task 1

Einstellbereich: STOPP / START / HALT

IPOS-STW. Task 1 nur im Bediengerät DBG60B, nicht in SHELL.

STOPP: Die Task 1 des IPOS$^{plus®}$-Programms wird gestoppt.

START: Die Task 1 des IPOS$^{plus®}$-Programms wird gestartet.

HALT: Task 1, 2 und 3 des IPOS$^{plus®}$-Programms werden gestoppt.

12.4.3 P932 IPOS-STW. Task 2

Anzeigebereich: START / STOPP

IPOS-STW. Task 2 nur im Bediengerät DBG60B, nicht in SHELL.

Anzeigeparameter, kann mit dem DBG60B nicht eingestellt werden.

START= Task 2 des IPOS$^{plus®}$-Programmes wird momentan abgearbeitet.

STOPP= Task 2 des IPOS$^{plus®}$-Programmes ist gestoppt.

12.4.4 P933 Ruckzeit

Einstellbereich: 0,005 – 2 s

Die Ruckzeit gibt die Dauer des Momentenaufbaus an. Die Positionierzeit verlängert sich gegenüber der linearen Rampe um die eingestellte Ruckzeit.

Für die Funktion Ruckbegrenzung einzustellende Ruckzeit (0,005 – 2 s). Dabei ist zu beachten, dass P911 Positionier-Rampe 1 (Seite 127) / P912 Positionier-Rampe 2 (Seite 127) einen größeren oder gleichen Wert aufweisen.

P933 ≤ P911 & P912

Ist P933 > P911 & P912, so ist weiterhin ein trapezförmiger Momentenaufbau vorhanden, wobei dann die eingestellte Ruckzeit nicht die Zeit für den Momentenaufbau ist.

12.4.5 P938 Geschwindigkeit Task 1

Einstellbereich: 0 – 9 zusätzliche Assemblerbefehle/ms

Die Standardeinstellung für Task 1 ist 1 Assemblerbefehl/ms. Mit P938 kann die Geschwindigkeit um bis zu 9 zusätzliche Assemblerbefehle/ms erhöht werden. Die Ressourcen für die Geschwindigkeitserhöhung teilt sich P938 mit P939, d. h. Task 1 und Task 2 dürfen **zusammen** 9 zusätzliche Assemblerbefehle/ms zugeordnet werden. Beispiel:

Task 1 **+ 2 zusätzliche Assemblerbefehle/ms** = 3 Assemblerbefehle/ms

Task 2 **+ 7 zusätzliche Assemblerbefehle/ms** = 9 Assemblerbefehle/ms

12.4.6 P939 Geschwindigkeit Task 2

Einstellbereich: 0 – 9 zusätzliche Assemblerbefehle/ms

Die Standardeinstellung für Task 2 ist 2 Assemblerbefehle/ms. Mit P939 kann die Geschwindigkeit um bis zu 9 zusätzliche Assemblerbefehle/ms erhöht werden. Die Ressourcen für die Geschwindigkeitserhöhung teilt sich P939 mit P938, d. h. Task 1 und Task 2 dürfen **zusammen** 9 zusätzliche Assemblerbefehle/ms zugeordnet werden. Beispiel:

Task 1 **+ 2 zusätzliche Assemblerbefehle/ms** = 3 Assemblerbefehle/ms

Task 2 **+ 7 zusätzliche Assemblerbefehle/ms** = 9 Assemblerbefehle/ms

12.5 P94x IPOS$^{plus®}$-Geber

12.5.1 P940 IPOS-Variablen Edit

Einstellbereich: EIN / AUS

IPOS$^{plus®}$-Variablen Edit nur im Bediengerät DBG60B, nicht in SHELL.

Wird P940 = “EIN” gestellt, können die IPOS$^{plus®}$-Variablen verändert werden.

12.5.2 P941 Quelle Ist-Position

Einstellbereich: Motorgeber (X15) / Ext. Geber (X14) Absolutwertgeber (X62)

Legt fest, auf welchen Geber IPOS$^{plus®}$ positioniert.

12.5.3 P942 / P943 Geberfaktor Zähler / Nenner

Einstellbereich: 1 – 32767

Stellen Sie **zuerst** P944 Geberskalierung Ext. Geber (Seite 135) oder P955 Geberskalierung (Seite 139) (bei Einsatz der Option DIP11B oder DEH21B) ein. Danach fahren Sie mit der Einstellung von P942/P943 fort.

Wird auf einen externen Geber (X14) oder einen Absolutwertgeber (X62) positioniert, wird mit diesen beiden Parametern eine Anpassung der Auflösung zum Motorgeber (X15) vorgenommen.

Vorgehensweise zur Ermittlung:

- Notieren Sie sich die Werte der Variablen H509 Absolutposition (H510 bei externem Geber) und H511 Aktuelle Motorposition.
- Verfahren Sie den Antrieb um ca. 30 000 Inkremente (H511).
- Bilden Sie die Differenz zwischen den notierten und den neuen Werten der Variablen:
  - H509 neu – H509 alt = H509 Differenz
  - H511 neu – H511 alt = H511 Differenz
- Die Differenzen dürfen nicht größer sein als 32 767 (2^{15} –1). Bei größeren Werten dividieren Sie beiden Differenzen durch die gleiche Zahl, so dass Sie entsprechend kleinere Werte erhalten oder wiederholen Sie den Vorgang mit einer geringeren Verfahrstrecke.
- Tragen Sie das Ergebnis H511 Differenz in P942 Geberfaktor Zähler ein und H509 in P943 Geberfaktor Nenner.

12.5.4 P944 Geberskalierung Ext. Geber

Einstellbereich: x1 / x2 / x4 / x8 / x16 / x32 / x64

Achten Sie **vor** der Einstellung von P944 darauf, dass P942 und P943 auf "1" eingestellt sind.

Die Wertigkeit der Wegauflösung von Motorgeber und externem Geber wird angepasst. Der Parameter wird so eingestellt, dass das Verhältnis der Weginformation zwischen dem Motorgeber und dem ext. Geber möglichst nahe "1" ist. Stellen Sie den Parameter zuerst auf "x1". Notieren Sie sich hierzu die Werte in den Variablen H510 und H511.

- Verfahren Sie den Antrieb um ca. 1000 Inkremente (H511).
- Ermitteln Sie die Differenz zwischen den notierten und den aktuellen Werten:
  - H510 neu – H510 alt = H510 Differenz
  - H511 neu – H511 alt = H511 Differenz
- Bilden Sie den Quotienten H511 Differenz dividiert durch H510 Differenz. Den Parameter P944 Geberskalierung Ext. Geber stellen Sie auf den Wert ein, der dem ermittelten Quotienten am nächsten kommt.

.

Achtung: Die Geberskalierung hat direkten Einfluss auf die Parameter P900 Referenzoffset (Seite 123), P942 Geberfaktor Zähler (Seite 134) und P943 Geberfaktor Nenner (Seite 134) und auf die Parametergruppe P92x IPOS-Überwachungen. Weiterhin sind alle Positionen des IPOS$^{plus®}$-Programms bei Nutzung des ext. Gebers anzupassen. Bei jeder Änderung der Geberskalierung muss die Einstellung aller aufgeführten Parameter nachgeführt werden!

Die Anzahl der an X14 erfassten Impulse wird mit P944 multipliziert und dann auf H510 abgebildet. Der externe Geber muss immer weniger Impulse liefern als der Motorgeber. Sollte dies nicht möglich sein, halten Sie bitte Rücksprache mit SEW-EURODRIVE.

12.5.5 P945 Streckengeber Typ (X14)

Einstellbereich: TTL / SIN/COS / HIPERFACE

Geben Sie hier den verwendeten Gebertyp ein. Mögliche Gebertypen sind:

- TTL: Geber mit digitalem, rechteckförmigen Ausgangssignal (TTL-Pegel 0 V, 5 V, mit negierten Spuren, Geber mit Signalpegel gemäß RS422
- SIN/COS: Geber mit analogem, sinusförmigen Ausgangssignal (1 V_{SS})
- HIPERFACE®: Geber mit der Bezeichnung AV1H, AS1H, ES1H, EV1H

SEW-Gebertyp	Inbetriebnahmeparameter Gebertyp / Geberstrichzahl
ES1S / ES2S / EV1S / EH1S	SINUS-GEBER / 1024
AV1Y	SINUS-GEBER / 512
ES1R / ES2R / EV1R / EH1R	INKREM. GEBER TTL / 1024
ES1T[1] / ES2T[1] / EV1T[1] / EH1T[1]	INKREM. GEBER TTL / 1024
AV1H / AS1H / ES1H / EV1H	HIPERFACE®

1) Nur über DWI11A

12.5.6 P946 Streckengeber Zählrichtung (X14)

Einstellbereich: NORMAL / INVERTIERT

Legt die Zählrichtung des Streckengebers fest. Die Einstellung muss so erfolgen, dass die Zählrichtung von Motorgeber (X15) und Streckengeber (X14) übereinstimmen.

12.5.7 P947 Hiperface Offset X14

Einstellbereich: $-(2^{31} - 1) - 0 - 2^{31} - 1$

Mit dem Parameter wird der Nullpunkt der Geberanzeige festgelegt.

Dieser Parameter wird verwendet, um den Maschinennullpunkt ohne Referenzfahrt zu definieren. Dazu wird der Geberwert mit dem Offset verrechnet.

- P905 Hiperface Offset X15 (Seite 126) wirkt auf die Istposition des Motorgebers H511.

  H511 = Geberwert – P905

- P947 Hiperface Offset X14 (Seite 136) wirkt auf die Istposition des externen Gebers H510.

  H510 = Geberwert – P947

Die Umrechnung der Istposition erfolgt direkt nach der Eingabe der Werte. Eine vorherige Referenzfahrt ist nicht notwendig.

Hinweis:

Bei einer Referenzfahrt eines Antriebssystems mit Hiperface®-Geber wird je nach eingestellter Quelle Istposition durch die Referenzfahrt die Hiperface Offsets (P905 oder P947) neu berechnet und überschrieben.

Es gilt:

- P905 = Geberwert – P900
- P947 = Geberwert – P900

12.5.8 P948 Automatische Gebertauscherkennung

Einstellbereich: EIN / AUS

Dieser Parameter ist nur bei Hiperface®-Gebern wirksam.

- EIN: Ein Tausch eines Hiperface®-Gebers wird erkannt. Bevor das Bit "IPOS referenziert" gesetzt wird, ist eine Referenzfahrt nötig.

  Beachten Sie beim Einsatz eines Linearmotors mit einem Motorgeber AL1H:

  Wurde der Linearmotor bei der Erstinbetriebnahme im Dialog "Geberjustage" kommutiert, wird nach einem Gebertausch das Bit "LSM kommutiert" im IPOS-Statuswort gelöscht. Zur Freigabe des Umrichters muss eine neue Kommutierungsfahrt angestoßen werden. Dadurch wird das Bit "LSM kommutiert" wieder gesetzt. Die Kommutierungsfahrt wird automatisch angestoßen, wenn die Klemme "/REGLERSPERRE" auf "1" gesetzt wird, sofern eine auf "Freigabe" programmierte Klemme kein "1-Signal" erhält. Erhält eine auf "Freigabe" programmierte Klemme ein "1-Signal", oder ist keine Klemme auf "Freigabe" programmiert, wird die Fehlermeldung F81 ausgegeben.

  SEW-EURODRIVE empfiehlt, bei Linearmotoren mit Motorgeber AL1H den Parameter P948 auf "AUS" einzustellen. Testen Sie nach einem Gebertausch das Antriebssystem mit reduzierter Geschwindigkeit und Kraft im Tippbetrieb.

- AUS: Der Hiperface®-Geber ist immer referenziert. Das Bit "IPOS referenziert" ist gesetzt.

  Beachten Sie beim Einsatz eines Linearmotors mit einem Motorgeber AL1H:

  Wurde der Linearmotor bei der Erstinbetriebnahme im Dialog "Geberjustage" kommutiert, bleibt die Kommutierung erhalten und das Bit "LSM kommutiert" im IPOS-Statuswort bleibt gesetzt. Die Freigabe des Antriebssystems kann sofort erfolgen.

i	**HINWEIS**
	Wird P948 aus- und wieder eingeschaltet, ist nach einem Neustart des MOVIDRIVE® B das Bit "IPOS referenziert" auf "0". Um das Bit "IPOS referenziert" wieder auf "1" zu setzen, ist eine Referenzfahrt nötig.

12.6 P95x Absolutwertgeber (SSI)

Die DIP-Parameter werden ausführlich im Handbuch "MOVIDRIVE® MDX61B Absolutwert-Geberkarten DIP11B/DEH21B" beschrieben. Die Option DIP11B kann nicht mit MOVIDRIVE® MDX61B der Baugröße 0 verwendet werden.

12.6.1 P950 Gebertyp

Der an der Option DEH21B/DIP11B (X62) angeschlossene Absolutwertgeber wird ausgewählt. Es sind zurzeit die Geber aus der folgenden Auswahlliste zulässig:

- VISOLUX EDM
- T&R CE65, CE58, CE100 MSSI
- T&R LE100, LE200
- T&R LA66K
- AV1Y / ROQ424
- STEGMANN AG100 MSSI
- SICK DME-3000-111
- STAHL WCS2-LS311
- STEGMANN AG626
- IVO GM401, GXMMW A202PA2
- STAHL WCS3
- LEUZE OMS1, OMS2
- T&R ZE 65M
- LEUZE BPS37
- SICK DME 5000-111
- POMUX KH53
- KÜBLER 9081
- LEUZE AMS200
- MTS TEMPOSONICS RP
- P+F AVM58X-1212
- Hübner HMG161 S24 H2048
- Balluf BTL5-S112B-M1500
- T&R LA41K
- Elgo LIMAX2

12.6.2 P951 Zählrichtung

Einstellbereich: NORMAL / INVERTIERT

Legt die Zählrichtung des Absolutwertgebers fest. Die Einstellung muss so erfolgen, dass die Zählrichtungen von Motorgeber (X15) und Absolutwertgeber (X62) übereinstimmen.

12.6.3 P952 Taktfrequenz

Einstellbereich: 1 – 200 %

Legt die Taktfrequenz fest, mit der die Absolutwertinformationen vom Geber an den Umrichter übertragen werden. Taktfrequenz = 100 % entspricht der Nominalfrequenz des Gebers, bezogen auf 100 m Leitungslänge.

12.6.4 P953 Positionsoffset

Einstellbereich: $-(2^{31}-1) - 0 - 2^{31}-1$

Der Positionsoffset P953 muss nur bei Drehgebern eingestellt werden, bei anderen Gebern sollte er auf 0 eingestellt werden.

Hinweis: Durch eine Referenzfahrt wird der Positionswert nach erfolgreicher Beendigung der Referenzfahrt automatisch neu berechnet und überschrieben.

12.6.5 P954 Nullpunktoffset

Einstellbereich: $-(2^{31}-1) - 0 - 2^{31}-1$

Mit dem Nullpunktoffset wird einer bestimmten Position ein gewünschter Wert zugeordnet. Der Wertebereich kann positive und negative Positionswerte annehmen. Zu beachten ist die maximal gültige Größe. Die Begrenzung wird durch den Wertebereich des Zählers (2^{31}) und den Wertebereich des Absolutwertgebers bestimmt. Verfahren Sie den Antrieb auf eine bekannte Position. Lesen Sie den Wert der Variablen H509 ACT.POS.ABS aus und tragen Sie in Parameter P954 Nullpunktoffset folgenden Wert ein: P954 = Variable H509 – gewünschter Wert.

Der gewünschte Wert ist der Anzeigewert an der aktuellen Position.

12.6.6 P955 Geberskalierung

Einstellbereich: x1 / x2 / x4 / x8 / x16 / x32 / x64

Achten Sie **vor** der Einstellung von P955 darauf, dass P942 und P943 auf "1" eingestellt sind.

Die Wertigkeit der Wegauflösung von Motorgeber und Absolutwertgeber wird angepasst. Der Parameter wird so eingestellt, dass das Verhältnis der Weginformation zwischen dem Motorgeber und dem Absolutwertgeber möglichst nahe "1" ist. Stellen Sie den Parameter zuerst auf "x1". Notieren Sie sich hierzu die Werte in den Variablen H509 und H511.

- Verfahren Sie den Antrieb um ca. 1000 Inkremente (H511).
- Ermitteln Sie die Differenz zwischen den notierten und den aktuellen Werten:
  - H509 neu – H509 alt = H509 Differenz
  - H511 neu – H511 alt = H511 Differenz
- Bilden Sie den Quotienten H511 Differenz dividiert durch H509 Differenz. Den Parameter P955 Geberskalierung stellen Sie auf den Wert ein, der dem ermittelten Quotienten am nächsten kommt.

Achtung: Die Geberskalierung hat direkten Einfluss auf die Parameter P900 Referenzoffset (Seite 123), P942 Geberfaktor Zähler (Seite 134), P943 Geberfaktor Nenner (Seite 134), P954 Nullpunktoffset (Seite 139) und auf die Parametergruppe P92x IPOS-Überwachungen (Seite 131). Weiterhin sind alle Positionen des IPOS$^{plus®}$-Programms bei Nutzung des ext. Gebers anzupassen. Bei jeder Änderung der Geberskalierung muss die Einstellung aller aufgeführten Parameter nachgeführt werden!

12.6.7 P956 Baudrate CAN-Geber

Einstellbereich: 125 kBaud / 250 kBaud / 500 kBaud / 1 MBaud

Stellt die Baudrate des angeschlossenen CAN-Gebers ein.

12.7 P96x IPOS$^{plus®}$ Modulofunktion

Die IPOS$^{plus®}$-Modulofunktion dient zur Endlospositionierung, beispielsweise bei Rundtakttischen oder Kettenförderern. Ausführliche Informationen hierzu finden Sie im IPOS$^{plus®}$-Handbuch.

12.7.1 P960 Modulofunktion

Einstellbereich: AUS / KURZ / RECHTS / LINKS

- AUS: Die Modulofunktion ist ausgeschaltet.
- KURZ: Die Modulofunktion "kurzer Weg" ist aktiv. Der Antrieb bewegt sich von der Istposition auf kürzestem Weg zur Zielposition. Beide Drehrichtungen sind möglich.
- RECHTS: Die Modulofunktion "Rechts" ist aktiv. Der Antrieb bewegt sich von der Istposition mit Drehrichtung "Rechts" zur Zielposition, auch wenn damit der längere Weg gefahren wird. Die Drehrichtung "Links" ist nicht möglich.
- LINKS: Die Modulofunktion "Links" ist aktiv. Der Antrieb bewegt sich von der Istposition mit Drehrichtung "Links" zur Zielposition, auch wenn damit der längere Weg gefahren wird. Die Drehrichtung "Rechts" ist nicht möglich.

12.7.2 P961 Modulo - Zähler

Einstellbereich: 0 – 1 – 2^{31} – 1

Nachbildung des Getriebes durch Eingabe der Zahnzahlen von Getriebe und Vorgelege.

Modulo-Zähler = Zähler Getriebe i x Zähler Vorgelege i

12.7.3 P962 Modulo - Nenner

Einstellbereich: 0 – 1 – 2^{31} – 1

Nachbildung des Getriebes durch Eingabe der Zahnzahlen von Getriebe und Vorgelege.

Modulo-Nenner = Nenner Getriebe i x Nenner Vorgelege i

12.7.4 P963 Modulo - Geberauflösung

Einstellbereich: 0 – 4096 – 65535

Auflösung des gewählten IPOS$^{plus®}$-Gebersystems in Inkrementen.

Bei Positionierung auf den Motorgeber wird die IPOS$^{plus®}$-Geberauflösung von 4096 Inkrementen eingetragen (Voraussetzung ist die Geberauflösung von 512 bis 2048).

12.8 P97x IPOS-Synchronisation

12.8.1 P970 DPRAM-Synchronisierung

Einstellbereich: EIN / AUS

MOVIDRIVE® B kann synchronisiert mit Optionskarten (z. B. DHP11B, DFE24B) betrieben werden.

EIN: Die Synchronisation zur Optionskarte ist aktiviert.

Beachten Sie: Die Umrichter dürfen nur entweder von SBus1, SBus2 oder vom DPRAM synchronisiert werden. Die Umrichter dürfen **nicht von mehreren Schnittstellen gleichzeitig** synchronisiert werden. SEW-EURODRIVE empfiehlt, P885 / P895 auf einen im gesamten CAN-Netzwerk nicht genutzten Identifier einzustellen. Um eine Synchronisation mit interpolierender Sollwertverarbeitung zu realisieren, beachten Sie Parameter P888 und P916.

AUS: Die Synchronisation zur Optionskarte ist nicht aktiviert.

12.8.2 P971 Synchronisierungsphase

Einstellbereich: −2 – 0 – 2 ms

Zeitlicher Abstand zwischen Taktsignal und Übernahme der Daten.

13 Compiler – Editor

13.1 Technische Merkmale

- Integrierte IPOS$^{plus®}$ Positionierung und Ablaufsteuerung. IPOS$^{plus®}$-Geräte benötigen keine zusätzliche Hardware.
- Programmerstellung in einer Hochsprache
- Symbolische Variablennamen
- Möglichkeit zur Erstellung von Programmmodulen, die in anderen Projekten erneut verwendet werden können
- Übersichtliche modulare und strukturierte Programmierung wird möglich
- Verschiedene Schleifentechniken stehen zur Verfügung
- Compilersteuerung durch Präprozessorbefehle
- Standardstrukturen
- Anwenderdefinierte Strukturen
- Zugriff auf alle Systemvariablen
- Standardfunktionen stehen zur Verfügung
- Debugger zur Fehlersuche
- Umfangreiche Möglichkeiten zur Kommentierung
- Integriert in die Windows-Oberfläche
- Im Software-Paket MOVITOOLS® MotionStudio integriert

13.2 Erste Schritte

Diese Beschreibung soll Ihnen einen kurzen und möglichst schnellen Einstieg in die Programmierung mit dem IPOS$^{plus®}$-Compiler ermöglichen. Anhand eines Beispiels, das im Laufe der Kapitel schrittweise erstellt und erweitert wird, werden Sie in die grundsätzliche Funktionalität des Compilers eingeführt.

In dieser Einführung wird vorausgesetzt, dass Sie mit einem gängigen Betriebssystem von Windows vertraut sind und Erfahrung im Umgang mit Standard-Office-Anwendungen haben.

Die Einführung umfasst die folgenden 4 Schritte:

Schritt 1: IPOS$^{plus®}$-Compiler mit MOVITOOLS® MotionStudio starten

In diesem Kapitel lernen Sie die grundlegenden Schritte, um aus MOVITOOLS® MotionStudio heraus den IPOS$^{plus®}$-Compiler zu starten.

Schritt 2: Erstellung eines neuen Projekts

In diesem Kapitel lernen Sie die Oberfläche des IPOS$^{plus®}$-Compiler kennen und wie sie ein neues IPOS$^{plus®}$-Projekt erstellen.

Schritt 3: Das erste IPOS$^{plus®}$-Programm

In diesem Kapitels werden Sie Ihr erstes IPOS$^{plus®}$-Programm erstellen.

Schritt 4: Compilieren und Starten des IPOS$^{plus®}$-Programms

In diesem Kapitel werden Sie das in Schritt 3 erstellte Programm compilieren, in das MOVIDRIVE® laden und ausführen.

13.2.1 Schritt 1: IPOS$^{plus®}$-Compiler mit MOVITOOLS® MotionStudio starten

Voraussetzungen

Der IPOS$^{plus®}$-Compiler ist im Software-Paket MOVITOOLS® MotionStudio integriert.

Bevor Sie den IPOS$^{plus®}$-Compiler aus dem MOVITOOLS® MotionStudio heraus aufrufen, wird das Folgende vorausgesetzt:

- Auf Ihrem PC ist MOVITOOLS® MotionStudio 5.40 oder höher installiert.
- Das Gerät, für das Sie ein IPOS$^{plus®}$-Programm erstellen möchten, ist mit Ihrem PC verbunden. Verwenden Sie dazu einen geeigneten Schnittstellenumsetzer.

MOVITOOLS® MotionStudio starten und ein Projekt anlegen

Um MOVITOOLS® MotionStudio zu starten und ein Projekt anzulegen, gehen Sie folgendermaßen vor:

1. Starten Sie MOVITOOLS® MotionStudio aus dem Startmenü von Windows unter dem folgenden Menüpunkt:

   [Start] / [Programme] / [SEW] / [MOVITOOLS-MotionStudio] / [MOVITOOLS-MotionStudio]

2. Legen Sie ein Projekt mit Namen und Speicherort an.

Kommunikation aufbauen und Netzwerk scannen

Um mit MOVITOOLS® MotionStudio eine Kommunikation aufzubauen und Ihr Netzwerk zu scannen, gehen Sie folgendermaßen vor:

1. Richten Sie einen Kommunikationskanal (zum Beispiel "seriell") ein, um mit Ihren Geräten zu kommunizieren.

   Detailierte Angaben, um einen Kommunikationskanal zu konfigurieren, finden Sie in der Dokumentation (Handbuch oder Online-Hilfe) zum Software-Paket MOVITOOLS® MotionStudio.

2. Scannen Sie Ihr Netzwerk (Geräte-Scan). Betätigen Sie dazu die Schaltfläche [Netzwerk-Scan starten] [1] in der Symbolleiste.

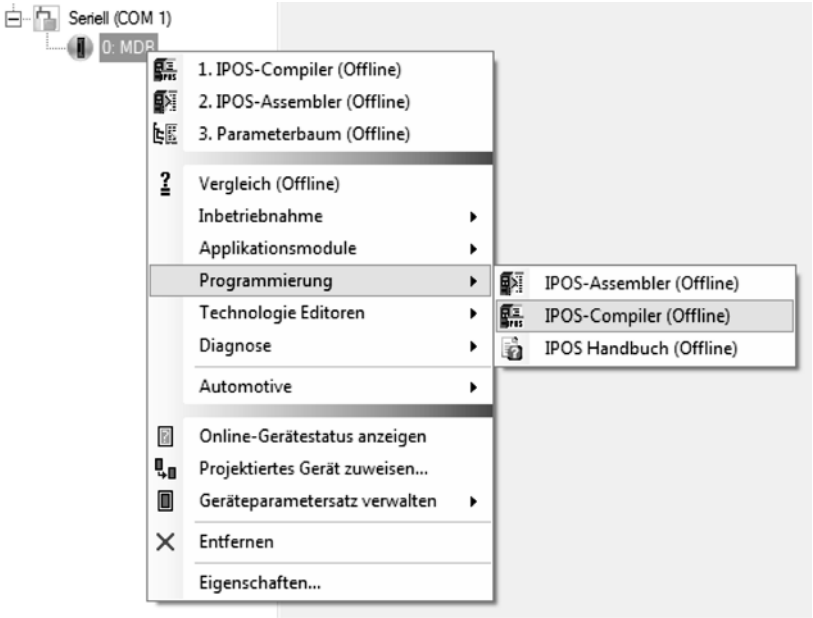
1132720523

IPOSplus-Compiler starten

Um mit MOVITOOLS® MotionStudio den IPOS^plus®-Compiler zu starten, gehen Sie folgendermaßen vor:

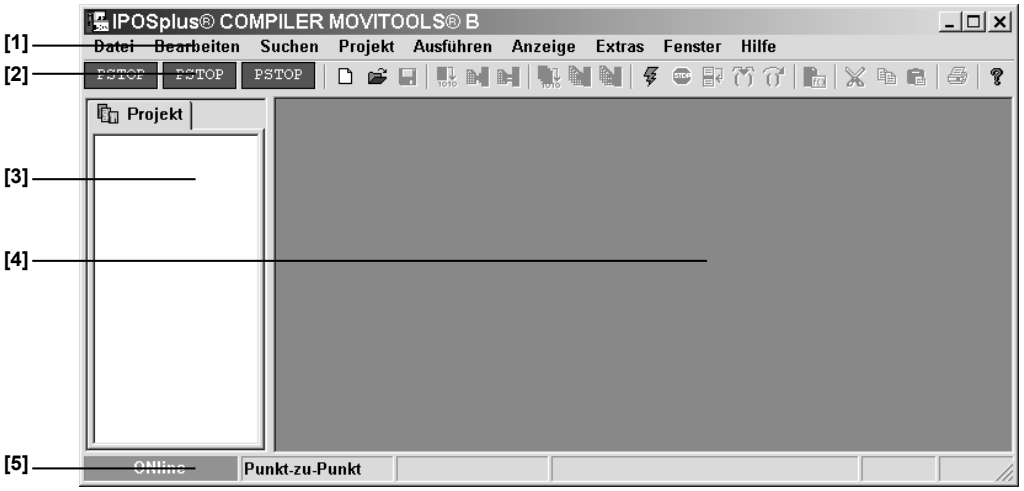
1. Markieren Sie das Gerät, für das Sie den IPOS^plus®-Compiler starten möchten.
2. Öffnen Sie mit der rechten Maustaste das Kontextmenü.
3. Starten Sie den IPOS^plus®-Compiler unter dem folgenden Menüpunkt:

   [Programmierung] / [IPOS-Compiler] /



2123686283

Als Ergebnis öffnet sich die IPOS$^{plus®}$-Programmieroberfläche:.



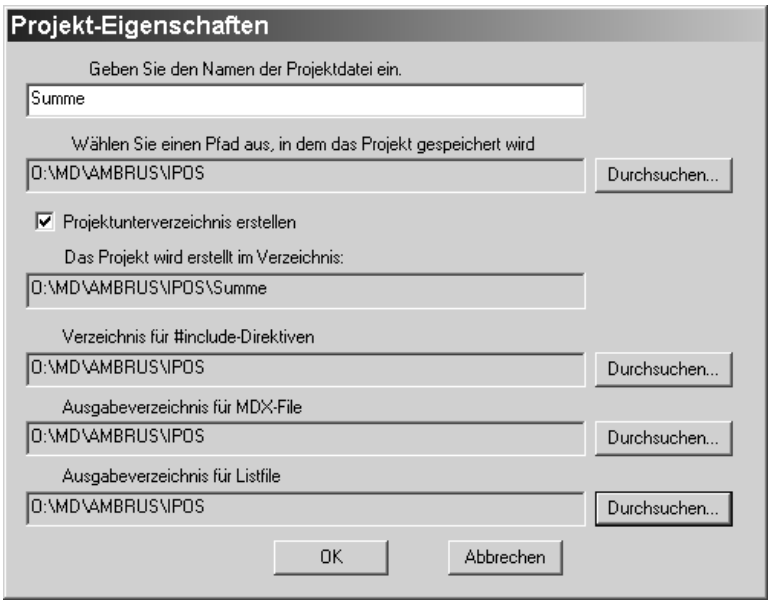
482065675

[1] Menüleiste
[2] Symbolleiste
[3] Projektfenster
[4] Hauptfenster
[5] Statuszeile

13.2.2 Schritt 2: Erstellung eines neuen Projekts

Um ein neues Projekt zu erstellen, klicken Sie den Menüpunkt [Projekt] / [Neu anlegen...]. Ein Dialogfenster erscheint, in dem zunächst die allgemeinen Projekteigenschaften festgelegt werden müssen:

Projekteigenschaften festlegen



482580363

Die erste Zeile enthält den Namen des Projekts. Hier geben Sie dem Projekt einen eindeutigen, signifikanten Namen.

Die zweite Zeile gibt das Verzeichnis an, in dem das Projekt abgelegt wird. Wählen Sie das Verzeichnis mit der Schaltfläche [Durchsuchen...] aus. Das Verzeichnis muss bereits existieren.

Soll im angegebenen Pfad ein Unterverzeichnis mit dem Projektnamen erstellt werden, dann muss ein Haken bei "Projektunterverzeichnis erstellen" gemacht werden. Die Projektdatei wird dann in dem Unterverzeichnis abgelegt.

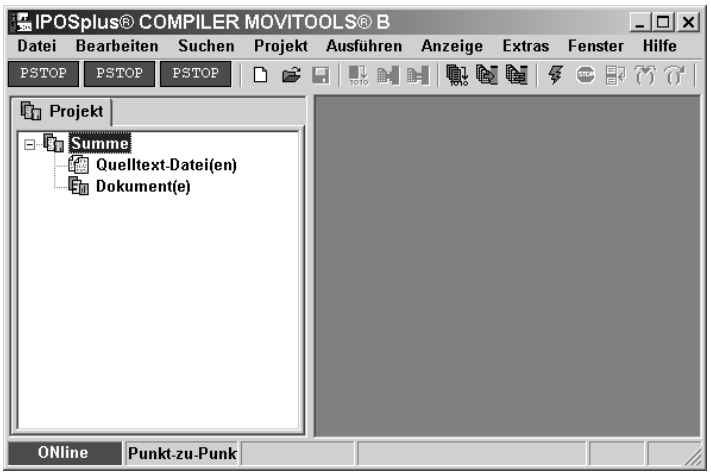
Die vierte Zeile gibt dem Compiler an, in welchem Verzeichnis die Dateien gesucht werden sollen, die mittels #include-Anweisung zu Quelltextdateien hinzugefügt werden. Hier steht als Default das bei der Installation angelegte Verzeichnis, z. B.: c:\programme\sew\moviools\projects\include.

Die Zeilen 5 und 6 geben an, in welchen Verzeichnissen die MDX-Datei (Datei mit IPOS$^{plus®}$-Programm) und die List-Datei (Datei mit zusätzlichen Programm-Informationen) angelegt werden sollen. Diese Dateien werden nur erzeugt, wenn im Menü [Extras] / [Einstellungen] / [Compiler] die entsprechenden Häkchen gesetzt sind.

Nach Betätigen der Schaltfläche [OK] führt der Compiler die folgenden Schritte aus:

- Anlegen des Ordners Summe im angegebenen Verzeichnis (nur wenn ein Haken bei "Projektunterverzeichnis erstellen" gesetzt ist).
- Anlegen einer Projektdatei mit dem Namen Summe.icp im Ordner Summe.
- Schließen des Dialogfensters.

Jetzt erscheint das Projekt als hierarchischer Baum im Projektfenster des Hauptfensters:



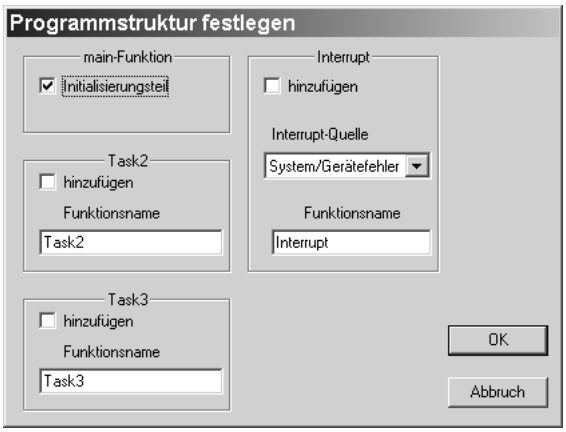
482803339

Als nächstes muss eine neue Quelltextdatei erzeugt und dem Projekt hinzugefügt werden. Dazu klicken Sie auf [Datei] / [Neu] / [Quelltext-Datei...].

Bestätigen Sie dieses Dialogfenster mit [Ja] und eine neue Quelltextdatei wird dem Projekt hinzugefügt. Jetzt werden Sie noch nach einem Namen für die neue Quelltextdatei gefragt. Geben Sie den Namen `summe.ipc` ein.

Nach Betätigen der Schaltfläche [Speichern] erscheint ein weiteres Fenster. Hier legen Sie die Programmstruktur fest.

Programmstruktur festlegen



482809355

Markieren Sie in der Gruppe "main-Funktion" das Kontrollfeld "Initialisierungsteil". Nach Verlassen des Dialogfensters mit der Schaltfläche [OK] wird die Funktion "Main" mit Initialisierungsteil automatisch generiert. Die Quelltextdatei hat jetzt folgenden Inhalt:

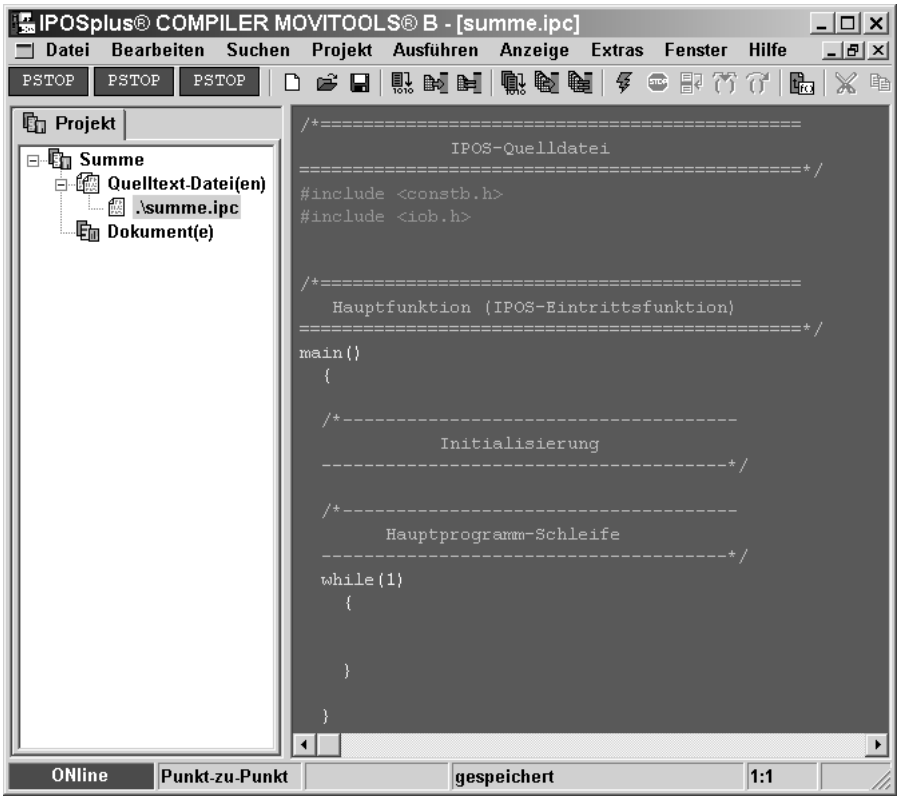
```
/*=============================================
IPOS-Quelldatei
===============================================*/
#include <constb.h>
#include <iob.h>

/*=============================================
Hauptfunktion (IPOS-Eintrittsfunktion)
===============================================*/
main()
{

   /*-------------------------------------
   Initialisierung
   -------------------------------------*/

   /*-------------------------------------
   Hauptprogramm-Schleife
   -------------------------------------*/
   while(1)
   {

   }

}
```

Der Befehl `#include <constb.h>` für MOVIDRIVE® B fügt die Header-Datei ein, in der die Argumente aller Systemfunktionen definiert sind.

Der Befehl `#include <iob.h>` für MOVIDRIVE® B fügt eine Datei ein, die Definitionen der digitalen Eingänge und Ausgänge enthält. Auf diese Konstanten und Definitionen kann bei der Programmierung direkt zugegriffen werden.

Die Funktion "Main" enthält einen Initialisierungsteil und die Hauptprogramm-Schleife. Dies ist ein korrektes, lauffähiges Programm, welches jedoch noch keine Funktionalität enthält.

Das Hauptfenster hat jetzt folgendes Aussehen:



482850699

Wenn Sie anschließend Änderungen am Quelltext vornehmen, speichern Sie das Projekt mit [Datei] / [Alles speichern] ab. Bitte schließen Sie das Programm mit [Datei] / [Beenden].

13.2.3 Schritt 3: Das erste IPOS$^{plus®}$-Programm

Im Laufe dieses Kapitels werden Sie ihr erstes IPOS$^{plus®}$-Programm erstellen.

Editieren des IPOS$^{plus®}$-Programms

Starten Sie den Compiler erneut. Diesmal wird das Projekt einschließlich der Datei `summe.ipc` automatisch geladen, da es beim Beenden des Programms geöffnet war.

Um die weiteren Funktionen des IPOS$^{plus®}$-Compilers kennen zu lernen, werden Sie jetzt ein Programm schreiben, das alle Zahlen von 1 bis 100 addiert.

Hierzu soll jedoch nicht die Formel (n + 1) × (n / 2) benutzt werden, sondern eine Schleife programmiert werden, die die Summe iterativ addiert.

Das Programm sollte also folgendermaßen aussehen:

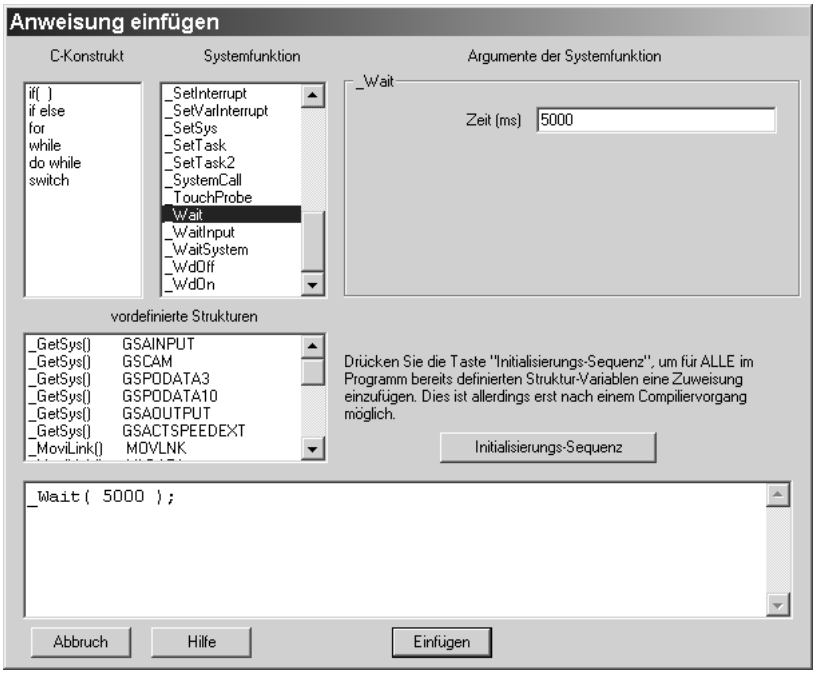
483418763

In der Statuszeile steht das Wort "geändert". Dies bedeutet, dass sich das Programm gegenüber dem letzten Speichervorgang geändert hat. Speichern Sie das Programm. Nun steht in der Statuszeile "gespeichert".

Durch das so genannte Syntax-Highlighting werden Wörter mit unterschiedlicher Bedeutung zur besseren Übersicht in unterschiedlichen Farben dargestellt. So werden z. B. alle dem Compiler bekannten Namen (Schlüsselwörter) gelb dargestellt. Die Systemfunktionen, die das Gerät zur Verfügung stellt, werden in der Farbe blau hervorgehoben.

Während des Editierens des Programms kann die Eingabeunterstützung verwendet werden. Durch Klicken mit der rechten Maustaste wird das Kontextmenü geöffnet, das den Menüpunkt [Anweisungen einfügen] enthält. Mit diesem Menüpunkt können Sie die Eingabeunterstützung aufrufen.

Eingabeunterstützung aufrufen



483423627

Mit der Eingabeunterstützung können Sie verschiedene C-Konstrukte, Systemfunktionen und vordefinierte Strukturen auswählen. Bei der Auswahl einer Systemfunktion müssen noch die Argumente dieser Funktion auf der rechten Seite des Fensters angegeben werden. Mit [Einfügen] wird der entsprechende Befehl an der Stelle, an der der Cursor steht, in den Quelltext eingefügt.

Um die _Wait-Funktion einzufügen, müssen Sie zuerst die _Wait-Funktion in der Liste der Systemfunktionen auswählen. In der rechten Fensterhälfte sind die für die jeweilige Funktion relevanten Argumente angegeben. Tragen Sie für unser Beispiel den Wert 5000 (für 5000 ms) ein.

Wenn Sie weiter gehende Informationen zu einem C-Konstrukt oder zu einer Systemfunktion wünschen, wählen Sie in einer der beiden Listen den gewünschten Begriff an und drücken die Taste <F1> oder Schaltfläche [Hilfe].

Aus dem Quelltext heraus können Sie die Hilfe aktivieren, indem Sie den Cursor auf dem Schlüsselwort _Wait platzieren und die Taste <F1> drücken.

	HINWEIS Der Compiler unterscheidet Groß-/Kleinschreibung, d. h. es können 2 unterschiedliche Variablen MYVAR und myvar existieren.

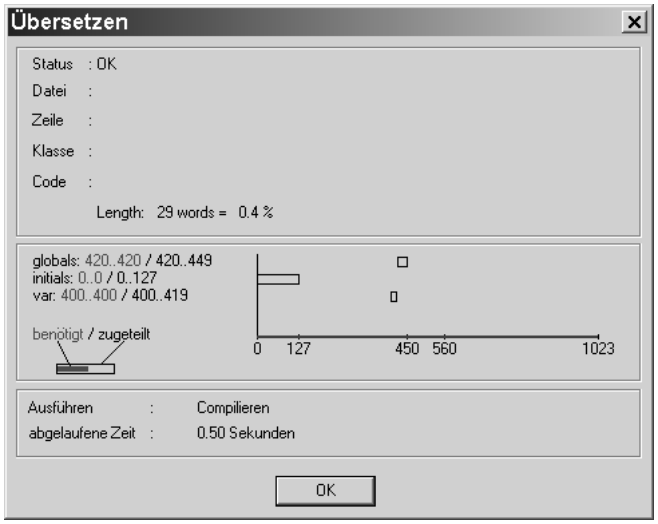
13.2.4 Schritt 4: Compilieren und Starten des IPOS$^{plus®}$-Programms

In diesem Kapitel werden Sie das in Schritt 3 erstellte IPOS$^{plus®}$-Programm compilieren, in das laden und ausführen.

Compilieren des Programms

Um das gewünschte Programm in eine für den Umrichter verständliche Form zu bringen, muss das Projekt compiliert werden. Dazu drücken Sie das Symbol oder Sie wählen [Projekt] / [Compilieren].

Meldungsfenster nach dem Compilieren



483686539

Nach dem Compilieren des Projekts wird das oben abgebildete Meldungsfenster angezeigt. Wenn das Programm keine Fehler beinhaltet, wird der Status OK ausgegeben. Ebenfalls wichtig ist die Größe des Programms. Sie wird als Länge der genutzten Code-Worte des Assembler-Codes angegeben. Diese absolute Zahl wird auch in eine Prozentzahl umgerechnet, die angibt, wieviel Speicherplatz im IPOS$^{plus®}$ verbraucht wird.

In unserem Beispiel war die Übersetzung des Programms erfolgreich und das Programm ist 29 IPOS-Worte groß und belegt damit 0,4 Prozent des gesamten IPOS$^{plus®}$-Speicherplatzes.

Mit der Schaltfläche [OK] wird das Fenster geschlossen.

Fehlermeldungen beim Compilieren

Da bei der Programmierung Syntaxfehler auftreten können, wurde in den IPOS$^{plus®}$-Compiler ein Fehlermeldesystem integriert. Wird ein Fehler erkannt, so wird dem Programmierer die Zeile, in der der Fehler aufgetreten ist angezeigt und eine entsprechende Fehlermeldung ausgegeben, die den Fehler klassifiziert.

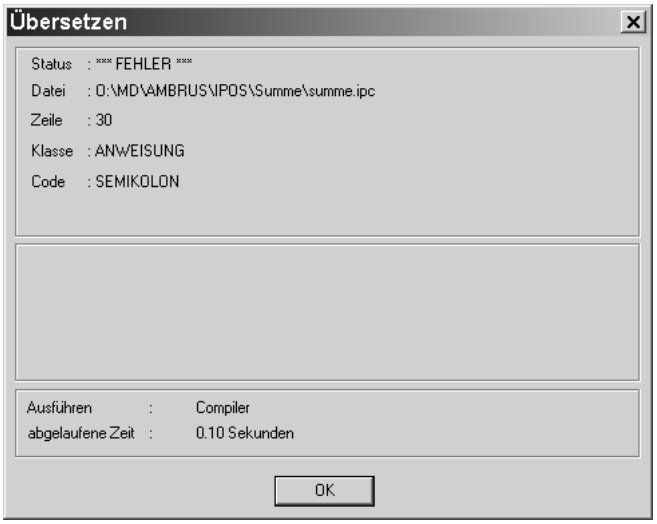
Dies wollen wir anhand eines Beispiels untersuchen.

Ändern Sie die FOR-Schleife folgendermaßen ab:

```
while(1)
{
   for (H1 = 0; H1 <= 100 )
   {
      // Summe bilden
      H2 = H2 + H1;
      // Warte 5 s lang
      _Wait( 5000 );
   }
}
```

In dem Beispiel fehlt in der FOR-Schleife das dritte Argument. Wird das Programm mit dem Symbol compiliert, so erscheint die folgende Meldung:

Fehlermeldung beim Compilieren



483692427

Da ein Fehler aufgetreten ist, wird der Status FEHLER ausgegeben. Außerdem wird die Zeile, die Fehlerklassifikation und der Fehlercode angezeigt. Zusätzlich wird die fehlerhafte Programmzeile in der Quelldatei mit einem roten Balken markiert.

Nach Drücken der Schaltfläche [OK] kann der Fehler beseitigt werden. Anschließend muss das Projekt neu compiliert werden.

Beseitigen Sie den Fehler, indem Sie die FOR-Schleife folgendermaßen korrigieren:

```
while(1)
{
   for (H1 = 0; H1 <= 100;H1++)
   {
      // Summe bilden
      H2 = H2 + H1;
      // Warte 5 s lang
      _Wait( 5000 );
   }
}
```

Laden des Programms in das Gerät

Im nächsten Schritt muss das Programm in den Umrichter geladen werden. Drücken Sie dazu das Symbol oder wählen Sie [Compilieren + Download]. Jetzt wird das Programm nochmals compiliert und nach erfolgreichem Abschluss in den Umrichter geladen.

Nach erfolgreichem Download des Programms wird die erste Programmzeile in der Funktion "Main" mit einem hellblauen Balken markiert. Das Programm kann nun gestartet werden.

Starten und Stoppen des Programms

Mit dem Symbol wird das Programm gestartet. Nun läuft das Programm im MOVIDRIVE® und in der Symbolleiste wird der Status START angezeigt.

Status START

483713291

Gleichzeitig wird der hellbaue Balken im Programm gelöscht.

Sie können sehen, dass das Programm abgearbeitet wird. In diesem kleinen Testprogramm wird die Variable H2 in 5-s-Schritten hochgezählt.

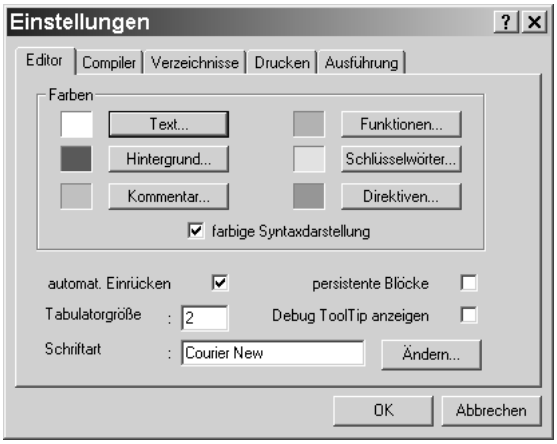
Öffnen Sie hierzu das Variablenfenster mit [Anzeigen] / [Alle Variablen]. Sie können nun die Variable H2 beobachten.

Nun wollen wir das Programm stoppen. Dies erfolgt durch Drücken des Symbols . Anschließend wird die erste Programmzeile in der Funktion "Main" mit einem hellblauen Balken markiert.

13.3 Einstellungen für den IPOS$^{plus-®}$Compiler

Im Menüpunkt [Einstellungen] des Menüs [Extras] besteht die Möglichkeit umfangreiche Einstellungen für den gesamten Compiler zu treffen. Wird dieser Menüpunkt ausgewählt, dann öffnet sich das folgende Fenster.

Einstellung Editor



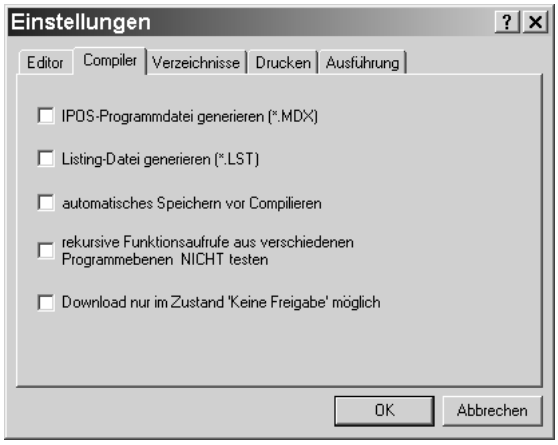
483719691

In den Einstellungen für den Editor können die Farben für den Hintergrund und für den Text gewählt werden. Ebenso können die Farben für das Syntax-Highlighting, für die farbige Syntaxdarstellung von Funktionen und Schlüsselwörtern eingestellt werden.

Zusätzlich können folgende Einstellungen gemacht werden:

- Farbige Syntaxdarstellung: Aktivieren und Deaktivieren des Syntax-Highlighting von Funktionen und Schlüsselwörtern.
- Automatisches Einrücken: Beim Wechsel in eine neue Zeile durch die Enter-Taste wird der Cursor automatisch unter das erste Zeichen der vorherigen Zeile eingerückt.
- Persistente Blöcke: Markierte Blöcke bleiben solange markiert, bis eine neue Markierung vorgenommen wird. Ist diese Option ausgeschaltet, bewirkt eine Bewegung des Cursors, dass die Markierung des Textblocks verschwindet. Eine Tastatureingabe ersetzt den markierten Block.
- Tabulatorgröße: Anzahl der Zeichen, die beim Drücken der Tabulatortaste eingerückt werden soll.
- Schriftart: Auswahl der Schriftart durch Drücken der Schaltfläche [Ändern...].
- Debug ToolTip anzeigen: Ist die Option [Debug ToolTip anzeigen] aktiviert, so werden Inhalte von Variablen direkt im Editorfenster angezeigt, wenn der Cursor auf die gewünschte Variable bewegt wird.

Einstellungen Compiler:



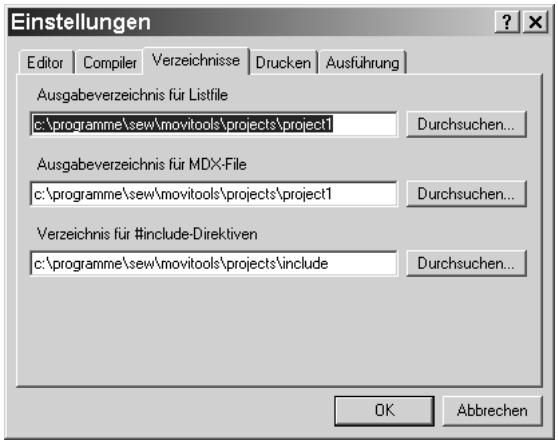
483721227

Innerhalb der Registerkarte für den Compiler können Einstellungen für den Compilier-Prozess vorgenommen werden.

- IPOS-Programmdatei generieren (*.MDX): Beim Compilieren wird eine *.MDX-Datei generiert. Die MDX-Datei enthält den Assembler-Code des Programms in textueller Form und kann über SHELL (Gerätedaten kopieren) in den Umrichter geladen oder im Assembler geöffnet werden.
- Listing-Datei generieren (*.LST): Beim Compilieren wird eine Listing-Datei generiert und gespeichert, die Angaben über die Ressourcen-Nutzung und den Programmablauf enthält.
- Automatisches Speichern vor Compilieren: Ist das Kontrollkästchen aktiviert, dann wird der Quellcode vor jeder Kompilierung automatisch gespeichert.
- Rekursive Funktionsaufrufe nicht testen: Beim Erkennen eines rekursiven Funktionsaufrufs meldet der Compiler einen Fehler, sofern das Kontrollkästchen nicht aktiviert ist. Sollen rekursive Funktionsaufrufe zugelassen werden, dann kann das Kontrollkästchen deaktiviert werden. Der Compiler gibt dann nur eine Warnung aus.
- Download nur im Zustand "keine Freigabe" möglich: Ist das Kontrollkästchen aktiviert, dann kann ein IPOS$^{plus®}$-Programm nur im Zustand "keine Freigabe" in den Umrichter geladen werden.

i	**HINWEIS** Vom Umrichter hochgeladene Programme, die ursprünglich vom Compiler erstellt wurden, lassen sich mit dem Assembler öffnen jedoch nicht bearbeiten.

Einstellungen Verzeichnisse:



483837963

Im IPOS$^{plus®}$-Compiler kann ein Programm entweder als Projekt oder als einzelne Quelltextdatei angelegt werden.

Wird das Programm als einzelne Quelltextdatei angelegt, dann müssen in dem obigen Fenster folgende Einstellungen gemacht werden:

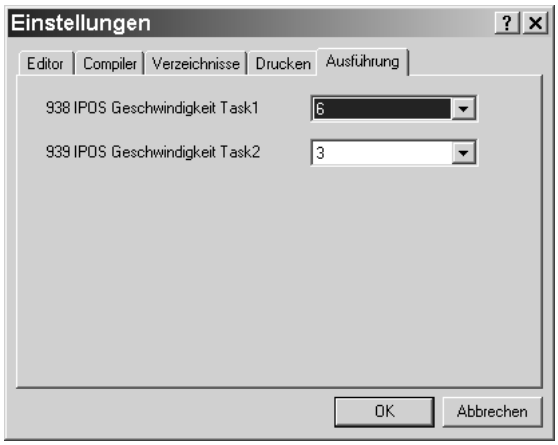
In dem Feld [Ausgabeverzeichnis für Listfile] muss das Verzeichnis angegeben werden, in dem das List-File gespeichert werden soll, sofern diese Funktion in den Einstellungen für den Compiler aktiviert worden ist. Dieses Verzeichnis kann durch Betätigen des Buttons [Durchsuchen] gesucht und ausgewählt werden.

In dem Feld [Ausgabeverzeichnis für MDX-File] muss das Verzeichnis angegeben werden, in dem die MDX-Datei gespeichert werden soll, sofern diese Funktion in den Einstellungen für den Compiler aktiviert worden ist. Dieses Verzeichnis kann durch Betätigen des Buttons [Durchsuchen] gesucht und ausgewählt werden.

Die Einstellung des [Verzeichnis für die #include-Direktiven] erfolgt in der letzten Zeile dieser Karteikarte. Dieses Feld enthält Angaben über das Verzeichnis in dem die mit dem #include-Befehl eingebundenen Header-Dateien abgelegt sind.

Wird ein IPOS$^{plus®}$-Programm als Projekt angelegt, so sind die Einstellungen in der Karteikarte [Verzeichnisse] nicht relevant.

Einstellungen der Task-Interpreter-Steps:



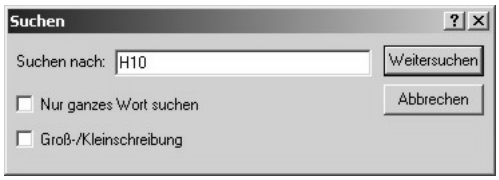
483839499

Auf der Karteikarte [Ausführung] können die Parameter für die Geschwindigkeit der Task 1 und Task 2 eingestellt werden. Eine ausführliche Beschreibung ist in Kapitel "Task-Verwaltung und Interrupts / Tasks bei MOVIDRIVE® B" zu finden.

13.4 Suchfunktion

Wählen Sie in der Menüleiste den Menüpunkt [Suchen], und dort [Suchen nach...] aus. Haben Sie zuvor eine Textstelle markiert, so wird der Text als Such-String übernommen (im folgenden Bild "H10"). Nun öffnet sich das folgende Fenster:

Fenster [Suchen]:

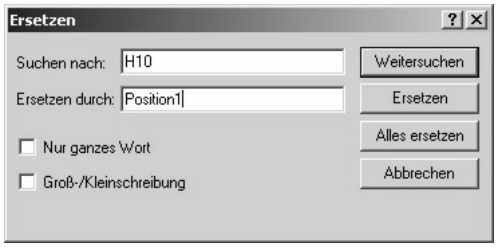


484333835

Durch Drücken des Buttons [Suchen] wird nach dem nächsten Vorkommen des entsprechenden Worts im gerade aktiven Fenster gesucht. Durch [Abbrechen] kann das Fenster wieder geschlossen werden.

Die gleiche Funktionalität gilt auch für die Funktion [Ersetzen...] aus dem Menüpunkt [Suchen] der Menüleiste.

Fenster [Ersetzen]:



484363787

Durch Betätigen des Buttons [Suchen] kann das entsprechende Wort gesucht werden und mit [Ersetzen] kann es mit dem zu ersetzenden Wort überschrieben werden. Beim Betätigen des Buttons [Alles ersetzen] werden alle dem Suchbegriff entsprechenden Zeichenfolgen ersetzt. Mit [Abbrechen] kann dieses Fenster geschlossen werden.

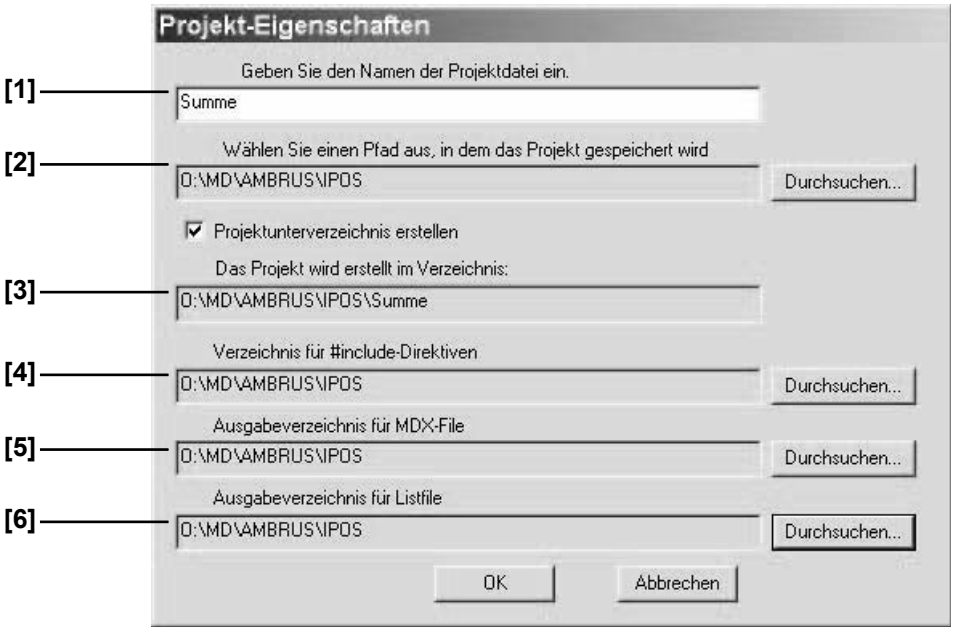
13.5 Erstellen eines neuen Projekts

Ein IPOS$^{plus®}$-Projekt besteht aus einem oder mehreren Quelltextmodulen. Jedes Modul ist in einer separaten Datei mit der Erweiterung *.IPC abgelegt. Die Informationen über das Projekt werden in einer Projektdatei mit der Erweiterung *.ICP gespeichert. Diese binäre Datei wird vom Compiler angelegt und verwaltet.

13.5.1 Projekteigenschaften festlegen

Durch Aufrufen des Menüpunkts [Projekt] / [Neu anlegen] erscheint der Eingabedialog, über den Sie die allgemeinen Projekteigenschaften festlegen.

Projekteigenschaften



484369163

[1] Name des Projekts.
[2] Verzeichnis des Projekts.
[3] Verzeichnis, in dem der Projektordner angelegt wird.
[4] Verzeichnis, in dem die Dateien, die mittels #include-Anweisung eingebunden werden, abgelegt sind.
[5] Ausgabeverzeichnis für MDX-Datei (falls aktiviert).
[6] Ausgabeverzeichnis für Listing-Datei (falls aktiviert).

Nach dem Betätigen der Schaltfläche [OK] werden die Einstellungen übernommen und das neu angelegte Projekt erscheint in der Baumstruktur im Projektfenster.

Der Wurzelknoten ist der Projektname, darunter angeordnet sind die Knoten `Quell-text-Datei(en)` und `Dokumente`. Unter dem Knoten `Quelltext-Datei(en)` werden die Quelltext-Module (*.IPC-Dateien) angeordnet.

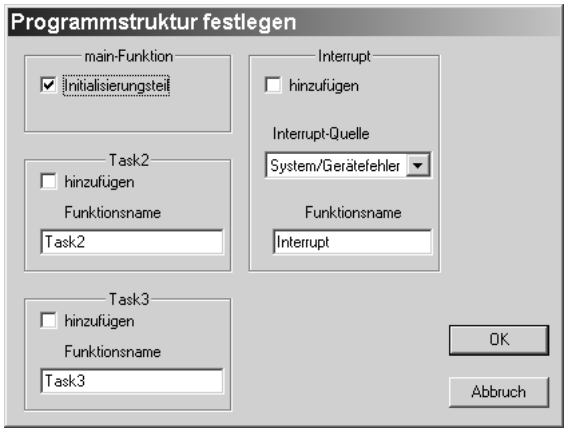
Alle darin enthaltenen Quelltext-Module werden zusammen zu einem IPOS$^{plus®}$-Programm übersetzt.

Unter dem Knoten `Dokumente` können beliebige, zur Dokumentation benötigte Dateien (z. B. auch Word-Dokumente) angehängt werden (rechter Mausklick: "Dokument dem Projekt hinzufügen"). Alle Dateien dieses Knotens sind vom Übersetzungsprozess ausgeschlossen.

Im folgenden Dialogfenster muss der neuen Quelltext-Datei noch ein Name gegeben werden. Wird das Fenster mit der Schaltfläche [Speichern] verlassen, öffnet sich ein weiteres Dialogfenster.

In diesem Fenster kann eine grundlegende Programmstruktur festgelegt werden, die als leere Programmschablone im Editorfenster angezeigt wird.

Programmstruktur festlegen



484387339

13.5.2 Programmstruktur festlegen

Soll das Hauptprogramm einen Initialisierungsteil haben, in dem beispielsweise Variablen initialisiert werden, so ist der Kontrollkasten [Initialisierungsteil] zu aktivieren.

Ebenso ist der Kontrollkasten [hinzufügen] für die Task 2 / Task 3 zu aktivieren, wenn eine Grundstruktur für die Task 2 / Task 3 angelegt werden soll. In diesem Fall kann auch gleich der Funktionsname für die Task 2 / Task 3 eingetragen werden, der dann direkt in die Grundstruktur übernommen wird. Falls eine Task 2 / Task 3 hinzugefügt wird, dann wird automatisch auch ein Initialisierungsteil in die Grundstruktur eingefügt, in dem der Befehl zum Start der Task 2 eingetragen ist.

Falls eine Interrupt-Routine programmiert werden soll, kann auch deren Grundstruktur bereits an dieser Stelle angelegt werden. Diese Struktur wird angelegt, wenn der zugehörige Kontrollkasten [hinzufügen] aktiviert ist. Über das Auswahlfeld [Interrupt-Quelle] kann gewählt werden, ob es sich um einen Interrupt auf einen Fehler, Timer oder Touch-Probe handeln soll. Der eingetragene Funktionsname wird als Name für die Interrupt-Funktion in die Grundstruktur übernommen. In den Initialisierungsteil wird eine Anweisungszeile zur Aktivierung der Interrupt-Routine eingefügt.

Mit dem Betätigen des Buttons [OK] wird die Auswahl der Programmstruktur abgeschlossen. Wird die Auswahl durch Betätigen des Buttons [Abbruch] beendet, so wird ein leeres Editorfenster ohne eine Programmstruktur geöffnet, wie es z. B. zur Erstellung einer eigenen Header-Datei benötigt wird.

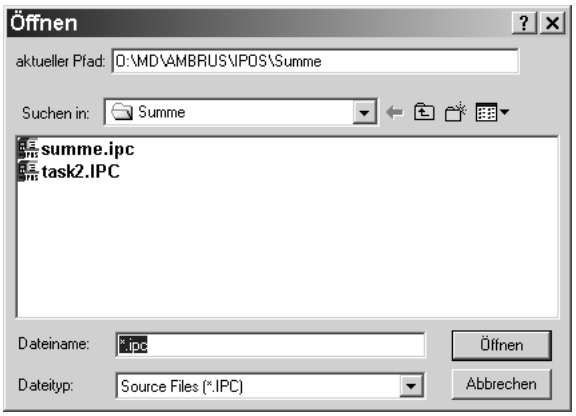
Wird beispielsweise eine Struktur mit Initialisierungsteil und Task 2 ausgewählt, wobei die Task 2 den Funktionsnamen Ueberwachung bekommen soll, so ergibt sich ein Editorfenster mit der folgenden Programmstruktur.

Compiler-Programm:

484392715

	HINWEIS Der IPOS$^{plus®}$-Compiler unterscheidet Groß-/Kleinschreibung, d. h. es können verschiedene Variablen MYVAR und myvar existieren. "ß" ist nicht zulässig.

Es besteht auch die Möglichkeit, einem Projekt eine bereits bestehende Quelltext-Datei hinzuzufügen. Dies geschieht über einen rechten Mausklick auf den Wurzelknoten `Quelltext-Datei` und dem folgenden Pop-Up-Menüpunkt [Quelltext-Datei dem Projekt hinzufügen]. Dann erscheint folgendes Dialogfenster.



484398091

Der Dateityp ist auf *.ipc voreingestellt. Dateien mit der Endung *.ipc kennzeichnen Quelltextdateien. Zusätzlich können Header-Dateien mit der Endung *.h ausgewählt und dem Projekt zugewiesen werden.

Wird eine Datei ausgewählt, so erscheint sie unter dem Wurzelknoten Quelltext-Datei(en) und ist damit dem Projekt zugeordnet.

13.6 Projekt speichern

Um ein Projekt und die im Projekt enthaltenen Quelltext-Dateien zu speichern, gibt es mehrere Möglichkeiten.

- Über den Menüpunkt [Datei] / [Alles speichern] wird das komplette Projekt und alle im Projekt eingebundenen Quelltext-Dateien gespeichert.
- Sollen nur Änderungen in der gerade bearbeiteten Quelltext-Datei gespeichert werden, so kann dies über den Menüpunkt [Datei] / [Speichern] oder über das Symbol der Symbolleiste gemacht werden.
- Über den Menüpunkt [Datei] / [Speichern unter] kann die im Projektfenster aktive Quelltext-Datei unter einem anderen Namen gespeichert werden.

13.7 Aufbau einer Projektverwaltung

Eine Projektverwaltung ermöglicht den übersichtlichen Zugriff auf alle ein Projekt betreffenden Daten. So ist beispielsweise folgende Verzeichnisstruktur sinnvoll:

- Hauptprojekt (z. B. Maschine oder Kunde)
  - Gesamtdokumentation (Unterlagen zum Gesamtprojekt)
  - Projekt (einzelner Umrichter)
    - Dokumentation (Dokumentation zum einzelnen Antrieb, falls nicht in Gesamtdokumentation abgelegt)
    - Source (alle *.IPC-Files, alle *.h-Files, auch const.h)
    - Parameter (*.mdx-File zum Geräteaustausch)
    - Messungen (Scope-Files)
    -
  - Projekt (einzelner Umrichter)
    - Dokumentation (Dokumentation zum einzelnen Antrieb, falls nicht in Gesamtdokumentation abgelegt)
    - Source (alle *.IPC-Files, alle *.h-Files, auch const.h)
    - Parameter (*.mdx-File zum Geräteaustausch)
    - Messungen (Scope-Files)

Beispiel:

Kunde: Müller Maschine: Hubstation MOVIDRIVE®: Hubachse, Gabelantrieb
Gesamtdokumentation
Hubachse
Dokumentation
Source
Parameter
Messungen
Gabelantrieb
Dokumentation
Source
Parameter
Messungen

Eine so aufgebaute Projektverwaltung ermöglicht jedem, der sich in die Anlage oder in das Programm einarbeiten muss, sich einen schnellen Überblick zu verschaffen. Er findet Dokumentationen und Quelltexte in kurzer Zeit und behält dabei den Überblick. Somit erleichtert sich die Pflege der Software bzw. des Gesamtsystems erheblich.

Die Verzeichnisse und Unterverzeichnisse können innerhalb der Projektverwaltung angelegt werden.

13.8 Projekt öffnen

Wird der Compiler geöffnet, so wird der Editor mit dem zuletzt bearbeiteten Quelltext des zuletzt geöffneten Projekts geöffnet, sofern der IPOS$^{plus®}$-Compiler bei geöffnetem Editor-Fenster beendet worden ist.

Ein vorhandenes Projekt kann außerdem über den Menüpunkt [Projekt] / [Öffnen] geöffnet werden.

Über das Dialogfenster kann die Projektdatei gesucht und mit der Schaltfläche [Öffnen] geöffnet werden.

13.9 Projekte handhaben mit MOVIDRIVE® B

	HINWEIS Bei Einsatz eines MOVIDRIVE® B besteht die Möglichkeit, ein fehlerfreies, compilierbares Projekt mit allen dazugehörigen Dateien im MOVIDRIVE® zu speichern. Der komplette Quellcode kann so im MOVIDRIVE® gespeichert und jederzeit abgerufen werden.

13.9.1 Projekt in Umrichter speichern

Über den Menüpunkt [Projekt] / [Download] wird das aktuell geöffnete Projekt und alle dazugehörigen Dateien, einschließlich der Projektdatei selbst, im MOVIDRIVE® gespeichert.

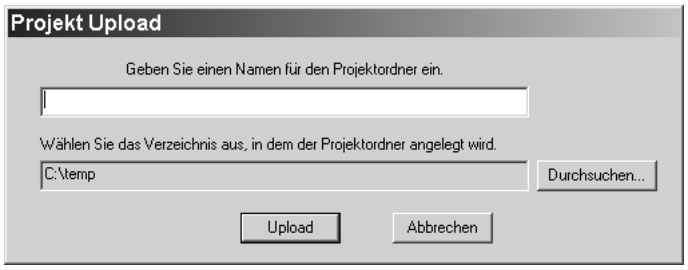
Das Projekt wird vor dem Download compiliert. Schlägt die Übersetzung fehl, wird kein Download durchgeführt.

Ist der im Umrichter vorhandenen Speicher zu klein, wird eine Fehlermeldung generiert und der Vorgang abgebrochen.

	HINWEIS Der Download der Projektdaten führt nicht automatisch zu einem Download des compilierten IPOS$^{plus®}$-Programms.

13.9.2 Projekt aus Umrichter laden

Über den Menüpunkt [Projekt] / [Upload...] kann ein im Umrichter gespeichertes Projekt auf den PC / Laptop geladen werden.



484532491

Der Name des Projektordners kann über ein Dialogfenster eingegeben werden. Über den Button [Durchsuchen] kann das Verzeichnis, in welches das Projekt abgelegt werden soll, ausgewählt werden.

Ist eine gleichnamige Projektdatei bereits im Verzeichnis enthalten, erfolgt eine Abfrage, ob diese überschrieben werden soll.

Sind keine Projektdaten im Umrichter vorhanden wird der Vorgang abgebrochen.

13.9.3 Projekt aus Umrichter abrufen

Diese Funktion dient dazu, das auf dem PC / Laptop vorhandene Projekt mit den Dateien aus dem Umrichter zu aktualisieren.

Der Menüpunkt lädt die im Umrichter gespeicherten Dateien und legt diese, im Unterschied zum Menüpunkt [Projekt] / [Upload], in den Verzeichnissen ab, aus denen sie ursprünglich stammen.

Ist eine gleichnamige Datei bereits im Verzeichnis enthalten, erfolgt eine Abfrage, ob diese überschrieben werden soll. Als Hilfestellung wird dazu das Erstellungsdatum der beiden Dateien angegeben.

Sind keine Projektdaten vorhanden wird der Vorgang abgebrochen.

13.10 Kompilieren eines Projekts

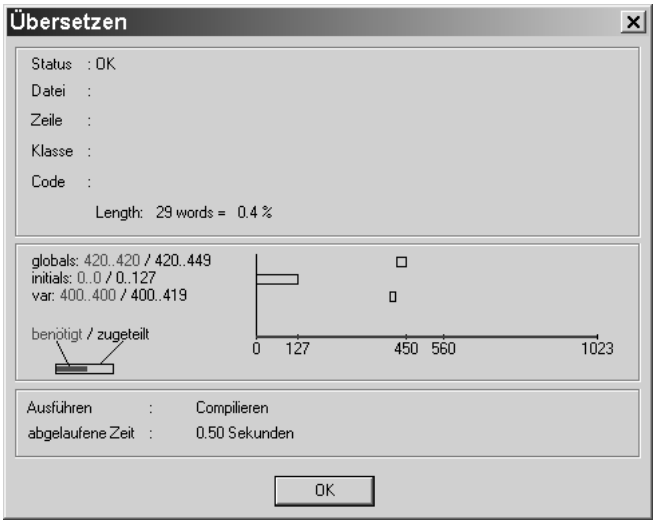
Um das geschriebene Programm in eine für den Umrichter verständliche Form zu übersetzen, muss das Programm compiliert werden. Besteht ein Projekt aus mehreren Quelltext-Dateien, dann werden alle Quelltext-Dateien beim Compilieren zu einem IPOS$^{plus®}$-Programm übersetzt.

Die Kompilierung eines Projekts kann über den Menüpunkt [Compilieren] des Menüs [Projekt] oder durch Betätigen des Symbols gestartet werden.

Falls in den Einstellungen zum Compiler die Funktion [automatisches Speichern vor Compilieren] aktiviert ist, so wird das Projekt abgespeichert. Ebenso wird nun eine IPOS$^{plus®}$-Programmdatei und eine Listing-Datei generiert, wenn dies in den Einstellungen zum Compiler aktiviert ist.

Nach dem Abschluss des Compilier-Vorgangs erscheint das folgende Fenster:

Statusfenster Compilieren



484538891

- Status: Ergebnis des Compilier-Vorgangs: OK oder ***FEHLER***.
- Datei: Quelltext-Datei des Projekts, in der ein Fehler entdeckt wurde.
- Zeile: Programmzeile, in der ein Fehler entdeckt wurde.
- Klasse: Fehlerklasse des aufgetretenen Fehlers.
- Code: Fehlercode des aufgetretenen Fehlers.

Zusätzlich enthält das Statusfenster Informationen über die Länge des erzeugten Programm-Codes und über die Speicherauslastung im Umrichter. Die Länge des Programm-Codes wird als Anzahl der genutzten Code-Worte des Assembler-Codes angegeben. Daraus wird die prozentuale Speicherauslastung errechnet und angezeigt.

Mit der Schaltfläche [OK] kann das Statusfenster verlassen werden. Im Fehlerfall steht der Cursor dann in Form eines roten Balkens in der Zeile, in der der beschriebene Fehler aufgetaucht ist.

Treten in einem Programm mehrere Fehler auf, so wird im Statusfenster immer nur der erste aufgetretene Fehler angezeigt. Ist dieser Fehler behoben und wird das Projekt erneut kompiliert, so wird im folgenden Statusfenster der nächste Fehler angezeigt.

13.11 Kompilieren und Download

Um das Programm in den Umrichter zu übertragen sind prinzipiell zwei Schritte nötig. Erstens muss der Quelltext compiliert werden, und zweitens muss dieses Programm zum Umrichter übertragen werden. Diese beiden Schritte werden ausgelöst, wenn [Compilieren + Download] aus dem Menü [Projekt] der Menüleiste ausgewählt wird. Prinzipiell gelten für das Compilieren die gleichen Rahmenbedingungen wie sie im vorherigen Abschnitt beschrieben worden sind. Eine explizite Meldung gibt es allerdings nur dann, wenn ein Fehler aufgetreten ist. Der Erfolg des Compilieren + Downloads wird dadurch sichtbar, dass im Anschluss an diesen Prozess die erste Programmzeile der MAIN-Funktion mit einem hellblauen Balken markiert ist.

13.12 Starten eines Programms

Nachdem das Programm in den Umrichter übertragen worden ist, kann es gestartet werden. Dazu wird aus dem Menü [Ausführen] der Menüleiste der Menüpunkt [Start] ausgewählt. Alternativ kann auch der Button in der Symbolleiste betätigt werden. Nach dem Starten des Programms wird der hellblaue Balken im Editor gelöscht und die Anzeige für den Task-Status springt von PSTOP auf START um.

Statusanzeigen für Task 1 und Task 2:

484635147

13.13 Stoppen eines Programms

Um die Programme in Task 1, Task 2 und Task 3 zu stoppen wird aus dem Menü [Ausführen] der Menüleiste der Menüpunkt [Stopp] ausgewählt. Alternativ kann auch der Button in der Symbolleiste betätigt werden. Die Anzeige für den Task-Status in der Symbolleiste springt von START auf PSTOP um.

13.14 Vergleich mit Gerät

Um den Inhalt des Editorfensters mit dem Programm im Umrichter zu vergleichen gibt es eine Vergleichsfunktion. Diese Funktion kann über den Menüpunkt [Vergleiche Datei mit Gerät] im Menü [Projekt] der Menüleiste aufgerufen werden. Ein Aufruf ist ebenfalls über das Symbol der Symbolleiste möglich.

13.15 Debugger

Der integrierte Debugger ist ein sinnvolles Hilfsmittel um ein Programm zu Testzwecken oder zur Fehlersuche in einzelnen Schritten abzuarbeiten. Um mit dem Debugger zu arbeiten, muss das Programm in den Umrichter übertragen werden. Drei unterschiedliche Funktionen stehen zu diesem Zweck zur Verfügung.

Funktion	Symbol	Taste	Beschreibung
Ausführen bis Cursor		<F4>	Das Programm wird bis zur aktuellen Cursor-Position ausgeführt.
Einzelschritt		<F7>	Die Programmzeile, auf der der Cursor steht, wird ausgeführt. Beim Aufruf einer Funktion wird in die Funktion verzweigt.
Überspringen		<F8>	Enthält die Programmzeile, auf der der Cursor steht, einen Funktionsaufruf, so wird nicht in die Funktion verzweigt und die Programmzeile wird übersprungen.

Mit dem Button aus der Symbolleiste, der Funktionstaste F5 oder dem Menüpunkt [Stopp] aus dem Menü [Ausführen] der Menüleiste kann das Programm jederzeit während des Debuggens gestoppt und zurückgesetzt werden.

Mit dem Button aus der Symbolleiste, der Funktionstaste F9 oder dem Menüpunkt [Start] aus dem Menü [Ausführen/Run] der Menüleiste kann das Programm jederzeit während des Debuggens ab der aktuellen Cursor-Position gestartet werden.

Während das Programm läuft, kann es mit der Tastenkombination Alt+F5 oder über den Menüpunkt [Ausführen] / [Pause] unterbrochen werden. Der Ausführungsbalken wird nun auf dem nächsten auszuführenden Befehl positioniert.

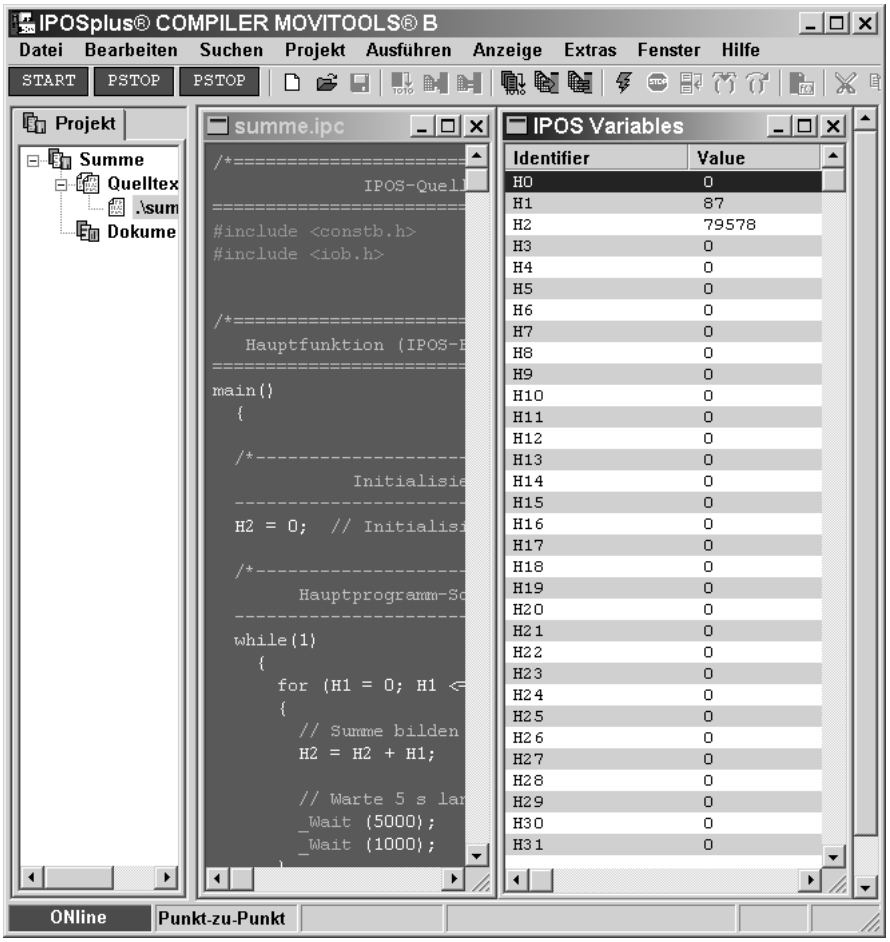
Mit der Funktionstaste F4 oder dem Menüpunkt [Ausführen bis Cursor], kann das laufende Programm unterbrochen werden. Das Programm wird in der Befehlszeile angehalten, auf der der Cursor steht.

13.16 Variablenfenster

Um den Inhalt von Variablen während des Debuggens oder der normalen Programmabarbeitung beobachten zu können, ist es hilfreich ein Variablenfenster zu öffnen.

Das Variablenfenster wird mit dem Menüpunkt [Alle Variablen] aus dem Menü [Anzeige] der Menüleiste aufgerufen.

Anzeige der Variablen



485381387

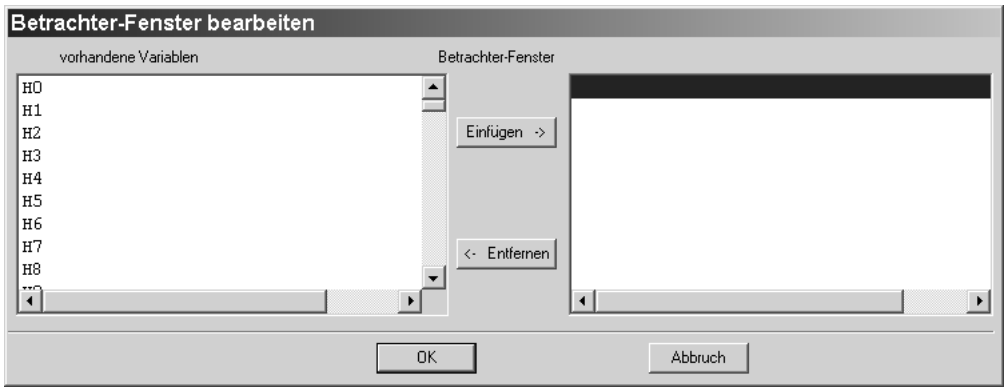
Im Variablenfenster können durch Verschieben der Bildlaufleiste alle Variablen beobachtet werden.

Eine weitere Möglichkeit, Variablen zu beobachten, ist das Einrichten eines Betrachterfensters. Im Betrachterfenster werden nur ausgewählte Variablen angezeigt. Je weniger Variablen gleichzeitig sichtbar sind, desto schneller wird der einzelne Wert aktualisiert.

	HINWEIS Die IPOS-Variablen, wie sie in der Datensicherung des Geräts enthalten sind, können auch im Offline-Modus angezeigt werden. Es werden dabei die nichtflüchtigen Variablen mit ihren permanent gespeicherten Werten angezeigt. Die durch Programmcode erfolgten flüchtigen Änderungen sind **nicht** in der Datensicherung gespeichert.

Das Einrichten eines Betrachterfensters geschieht über den Menüpunkt [Anzeige] / [Ausgewählte Variablen] / [Bearbeiten]. Folgendes Fenster erscheint:

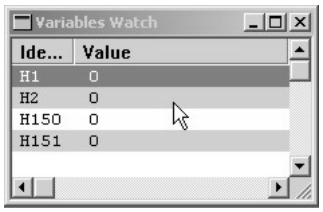
Variablen auswählen



485591563

Durch einen Doppelklick auf die gewünschte Variable oder durch Markieren und anschließendes Drücken der Schaltfläche [Einfügen] können die gewünschten Variablen dem Betrachterfenster zugewiesen werden. Die ausgewählten Variablen erscheinen in der Liste auf der rechten Seite des Fensters. Um eine Variable aus dem Betrachterfenster zu entfernen, muss diese in der Liste markiert und der Button [Entfernen] betätigt werden.

Ausgewählte Variable anzeigen:



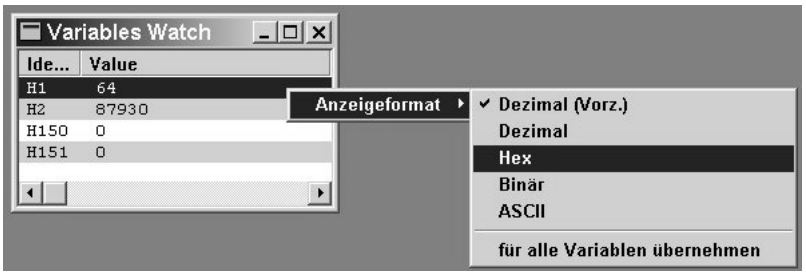
485597579

Das Fenster hat das gleiche Aussehen wie das komplette Variablenfenster, enthält jedoch nur die ausgewählten Variablen.

Da die Verwendung der Standard-Variablennamen (z.B. H1, H2 usw.) bei großen Programmen recht unübersichtlich wird, können den Variablen symbolische Bezeichner zugeordnet werden, die dann an dieser Stelle auch angezeigt werden.

Die Werte der Variablen können in unterschiedlichen Formaten angezeigt werden. Die Formate dezimal mit Vorzeichen, hexadezimal, binär oder ASCII stehen zur Auswahl. Zum Umschalten des Formates wird die gewünschte Variable zunächst durch Anklicken markiert. Anschließend wird ein Kontextmenü durch Betätigen der rechten Maustaste geöffnet und dort das gewünschte Format ausgewählt.

Variablenformat ändern:



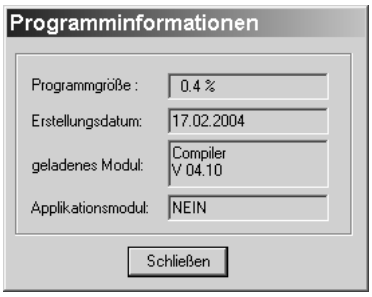
485601931

Im Betrachterfenster können einzelne Variablen auch mehrfach vorhanden sein. Auf diese Weise kann eine Variable in unterschiedlichen Formaten gleichzeitig angezeigt werden.

13.17 Programminformationen

In der Menüleiste steht im Menü [Anzeige] der Menüpunkt [Programminformationen] zur Verfügung. Wird dieser Menüpunkt angewählt, so öffnet sich das folgende Fenster.

Programminformationen



485608587

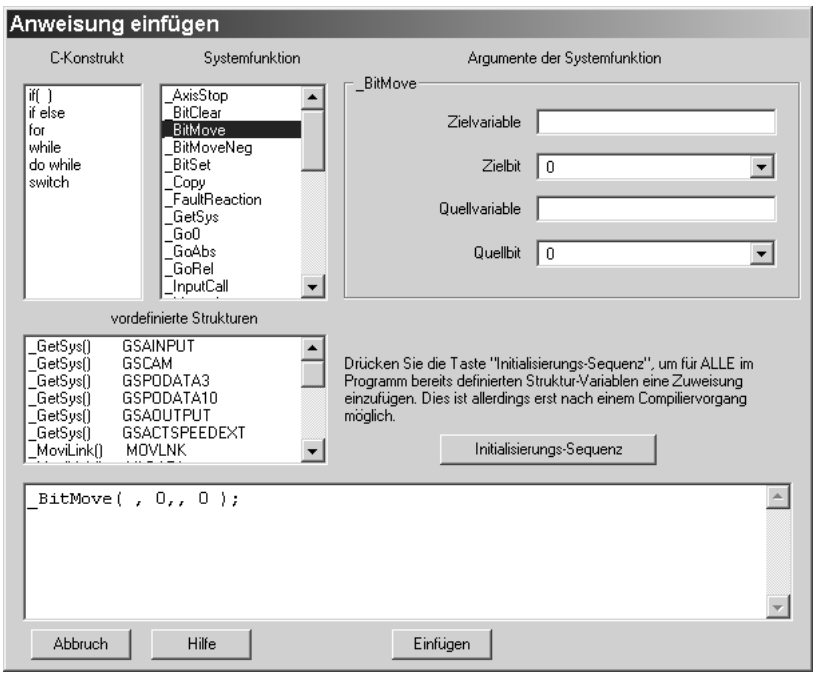
Diese Programminformationen beziehen sich auf das im Umrichter gespeicherte Programm. Als Informationen steht die Programmgröße, das Erstellungsdatum und der Name der Quelldatei zur Verfügung. Mit dem Button [Quelldatei öffnen] kann der zum Programm im Umrichter gehörende Quelltext in einem Editorfenster geöffnet werden. Dies setzt voraus, dass die Quelldatei ihren Namen nicht geändert hat und unter dem gleichen Pfad zu finden ist, wie zu dem Zeitpunkt als das Programm in den Umrichter übertragen wurde.

13.18 Anweisungseingabe

Prinzipiell kann der gesamte Quelltext eines IPOS$^{plus®}$-Programms als Text über die Tastatur eingegeben werden. Dabei ist die Syntax, die an die Programmiersprache C angelehnt ist, zu beachten. Über das Tastenkürzel [Strg + Z] können die letzten fünf Eingaben rückgängig gemacht werden.

Während des Editierens des Programms kann die Eingabeunterstützung genutzt werden. Sie wird aufgerufen, indem durch Klicken der rechten Maustaste ein Kontextmenü geöffnet und dort der Menüpunkt [Anweisung einfügen/Insert Instruction] ausgewählt wird. Ein Aufruf der Eingabeunterstützung ist ebenfalls durch Betätigen des Buttons in der Symbolleiste möglich.

Anweisung einfügen:



485614475

Über die Eingabeunterstützung können C-Konstrukte, Systemfunktionen oder vordefinierte Strukturen in den Quelltext eingefügt werden.

Wird ein C-Konstrukt oder eine Standardstruktur markiert, dann erscheint im unteren Teil des Fensters der Text, der später in den Quelltext eingefügt wird. Soll eine Systemfunktion eingefügt werden, müssen im rechten Teil des Fensters zusätzlich noch verschiedene Argumente der Funktion eingegeben werden. Nach Betätigen des Buttons [Einfügen] wird die ausgewählte Funktion an der Stelle im Quelltext eingefügt, an der der Cursor beim Aufruf der Eingabeunterstützung gestanden hat.

Über die Schaltfläche [Initialisierungs-Sequenz] wird für jede bereits definierte Struktur-Variable ein Initialisierungsteil eingefügt. Dazu muss das Programm mindestens einmal compiliert worden sein.

13.19 Kommentare

Eine gute Kommentierung des Quelltextes erleichtert die Lesbarkeit des Programms und ermöglicht auch einer fremden Person eine zügige Einarbeitung in das Programm.

Ein umfangreicher Kommentar, der zum Beispiel über mehrere Zeilen geht, beginnt mit der Zeichenkombination '/*' und endet mit der Zeichenkombination '*/'. Ein einzeiliger Kommentar beginnt mit der Zeichenkombination '//' und benötigt kein Endzeichen. Ein einzeiliger Kommentar kann auch im Anschluss an eine Befehlszeile im Quelltext stehen.

Kommentare

486824075

13.20 Symbolübersicht

Symbol	Menüpunkt	Beschreibung
	Datei → Neu	Neue Quelltextdatei erstellen
	Datei → Öffnen	Quelltextdatei öffnen
	Datei → Speichern	Quelltextdatei speichern
	Datei → Kompilieren	Quelltextdatei kompilieren
	Datei → Kompilieren + Download	Quelltextdatei kompilieren und in Umrichter laden
	Datei → Vergleiche mit Gerät	Quelltextdatei mit Programm im Umrichter vergleichen
	Projekt → Compilieren	Projekt compilieren
	Projekt → Compilieren + Download	Projekt compilieren und in Umrichter laden
	Projekt → Vergleiche mit Gerät	Projekt mit Programm im Umrichter vergleichen
	Ausführen → Start	IPOS$^{plus®}$-Programm starten
	Ausführen → Stopp	IPOS$^{plus®}$-Programm stoppen
	Ausführen → Ausführen bis Cursor	Programm ausführen bis Cursor
	Ausführen → Einzelschritt	Einzelschritt ausführen
	Ausführen → Überspringen	Anweisung überspringen
	Bearbeiten → Anweisung einfügen	Eingabeunterstützung aufrufen
	Datei → Drucken	Quelltextdatei drucken
	Hilfe → Benutzerhandbuch	Hilfe aufrufen

14 Compiler - Programmierung

Der Quelltext eines mit dem IPOS[plus®]-Compiler geschriebenen Programms setzt sich aus verschiedenen Teilen zusammen. Diese sollen zunächst einzeln betrachtet werden

```
/*********************************************
File name:          Program_structure.IPC
Date:               04.02.2002
Author:             Thomas Ambrus
SEW-EURODRIVE Bruchsal
Technical Documentation
Brief description:  Source code program struc-
ture
*********************************************/
*/===========================================
IPOS Source File
===========================================*/
#include <const.h>
#include <io.h>
/*===========================================
Main Function (IPOS Entry Function)
===========================================*/
main()
{
   /*-------------------------------------
   Initialisation
   -------------------------------------*/
   // activate task 2
   _SetTask2(T2_START, Monitor); // testing

   /*-------------------------------------
   Main Loop
   -------------------------------------*/
   while(1)
   {
   }
}
/*===========================================
Task2
===========================================*/
Monitor()
{
}
/*===========================================
User function
===========================================*/
Reference_travel()
{
}
Automatic_mode()
{
}
Manual_mode()
{
}
```

Kommentar mit Hinweisen zum Programm

Programmkopf mit Präprozessor-anweisungen und ggf. Variablendeklaration

Hauptfunktion beinhaltet den Initialisierungsteil und die Endlosschleife der Task 1

Initialisierungsteil

Endlosschleife für Task 1

Task 2, keine Endlosschleife erforderlich

Vom Anwender erstellte Funktionen (Unterprogramme), aus Task 1 oder Task 2 aufgerufen

14.1 Präprozessor

Der IPOS$^{plus®}$-Compiler ist ein Multi-Pass-Compiler, der den Quelltext in mehreren Durchläufen verarbeitet. Im ersten Durchlauf verarbeitet der Präprozessor für ihn bestimmte Anweisungen, - im folgenden Direktiven genannt - testet Anweisungen für die bedingte Kompilierung, entfernt Kommentare und erzeugt schließlich eine Zwischendatei für den Compiler. Der Präprozessor erhöht die Flexibilität und Produktivität beim Programmieren in folgenden Bereichen:

- Einbinden von Text aus anderen Dateien (Header-Dateien), die vorgefertigte und / oder benutzerdefinierte Konstanten oder Quelltext-Funktionen enthalten.
- Definieren von symbolischen Bezeichnern, um die Lesbarkeit des Quelltexts zu verbessern.
- Festlegen von Direktiven für die bedingte Kompilierung, um die Portierbarkeit zu verbessern und Testphasen zu vereinfachen.

Jede Zeile, die mit einem # beginnt, wird als Präprozessor-Direktive behandelt, außer # ist Teil eines Kommentars. Leerzeichen, die dem Zeichen # vorausgehen oder folgen, werden ignoriert.

Präprozessor-Direktiven werden normalerweise an den Anfang des Quelltexts geschrieben, können aber an jeder beliebigen Stelle im Programm stehen. Je nach Bedeutung gelten die Direktiven ab der Quelltextzeile, in der sie stehen oder sie gelten unabhängig davon für das ganze Programm.

14.2 Präprozessor-Anweisungen

Nach den Kommentarzeilen im Programmkopf folgen die Präprozessor-Anweisungen. Eine solche Anweisung wird bereits standardmäßig eingefügt, wenn man ein neues Editorfenster öffnet.

Die Anweisung '#include <const.h> ' bindet eine Header-Datei mit dem Namen const.h beim späteren Compilieren mit ein. Diese Datei ist fest vorgegeben und darf nicht verändert werden. Dennoch soll an dieser Datei die Funktion einer Header-Datei erläutert werden. Die Datei ist hier in gekürzter Form abgedruckt, da sich bereits so wesentliches zeigen lässt.

```
/*==============================================
File name: Const.h
File version: 2.20

SEW Include-File for IPOSplus Compiler

Please do not modify this file!

(C) 1999 SEW-EURODRIVE
==============================================*/

#ifndef _CONST_H
#define _CONST_H
.
.
.
#define Scope474    H474
#define Scope475    H475
#define DRS_Ctrl    H476
#define DRS_Status  H477
#define AnaOutIPOS2 H478
#define AnaOutpIPOS H479
#define OptOutpIPOS H480
```

```
#define StdOutpIPOS H481
#define OutputLevel H482
#define InputLevel  H483
#define ControlWord H484
#define T0_Reload   H485
#define Reserve4    H486
#define Timer_2     H487
#define Timer_1     H488
#define Timer_0     H489
#define WdogTimer   H490
#define SetpointPos H491
#define TargetPos   H492
#define PosWindow   H493
#define LagWindow   H494
#define LagDistance H495
#define SLS_right   H496
#define SLS_left    H497
#define RefOffset   H498
#define SetpPosBus  H499
#define Reserve6    H500
#define Reserve7    H501
#define TpPos2_Abs  H502
#define TpPos1_Abs  H503
#define TpPos2_Ext  H504
#define TpPos2_Mot  H505
#define TpPos1_Ext  H506
#define TpPos1_Mot  H507
#define Reserve8    H508
#define ActPos_Abs  H509
#define ActPos_Ext  H510
#define ActPos_Mot  H511

#endif
```

Nach einem allgemeinen Kommentarteil beginnt die eigentliche Struktur der Header-Datei mit einem '#ifndef _CONST_H'. Zu dieser '#ifndef'-Anweisung gehört mindestens immer ein '#endif'. Diese '#endif'-Anweisung findet man hier in der letzten Programmzeile der Header-Datei wieder. Diese Konstruktion aus '#ifndef' und '#endif' hat die Aufgabe eine Mehrfacheinbindung der Datei zu verhindern.

Die Anweisungen innerhalb dieses Konstrukts werden nur dann ausgeführt, wenn ein Makro-Bezeichner, hier '_CONST_H', noch nicht definiert ist (**ifn**ot **def**ined, wenn nicht definiert). In der nächsten Programmzeile steht ein '#define _CONST_H' um genau diesen Makro-Bezeichner zu definieren. Wird die Header-Datei also beim Compilieren durch das Abarbeiten des Befehls '#include <const.h> ' verarbeitet, so wird die Abfrage '#ifndef _CONST_H' zunächst bejaht, da der Makro-Bezeichner '_CONST_H' noch nicht bekannt ist. Anschließend wird er mit der '#define _CONST_H' bekannt gemacht. Wird an einer anderen Stelle des Programms nochmals die Header-Datei 'const.h' eingebunden, so ist der Makro-Bezeichner '_CONST_H' bekannt und die Abfrage '#ifndef _CONST_H' führt dazu, dass diese Frage verneint wird und zur '#endif'-Anweisung gesprungen wird. Somit wird verhindert, dass diese Datei unnötig mehrfach eingebunden wird, was zu einer Fehlermeldung führen würde.

Neben der Anweisung '#ifndef' gibt es auch die Anweisung '#ifdef' (**ifdef**ined, wenn definiert). Diese Anweisung kommt ohne eine Verneinung aus. Auch eine if-else-Konstruktion ist möglich. Dies bedeutet, dass in dem Fall, in dem die Bedingung der '#ifdef' bzw. '#ifndef'-Abfrage nicht erfüllt ist, der Anweisungsteil nach dem #else abgearbeitet wird. Somit ergeben sich die folgenden Möglichkeiten:

#ifdef Bezeichner_1
  Programmtext_1
#else
  Programmtext_2
#endif
	#ifndef Bezeichner_2
	  Programmtext_3
	#else
	  Programmtext_4
	#endif
		#ifdef Bezeichner_3
		  Programmtext_5
		#endif

Es sei darauf hingewiesen, dass diese Präprozessor-Anweisungen nicht nur in Header-Dateien verwendet werden können, sondern auch sinnvoll im Hauptprogramm eingesetzt werden können. So lassen sich beispielsweise Programmteile anlagenspezifisch übersetzen, ohne größere Änderungen im Quelltext vornehmen zu müssen.

14.3 #include

Diese Direktive erlaubt das Einbinden von Quelltexten aus anderen Dateien (Header-Dateien) in die Quelltext-Datei. Header-Dateien werden gewöhnlich dazu benutzt, um mehrfach benutzte Konstanten oder Makros zu definieren und diese in verschiedenen Projekten zu verwenden. Die Syntax lautet:

```
#include <DateiName>
```

`DateiName` ist der vollständige Name der einzubindenden Datei. Er wird in spitzen Klammern eingeschlossen. Liegt die einzubindende Datei im aktuellen Verzeichnis, genügt die Angabe des Dateinamens ohne Pfadangabe

Die Datei `BEISPIEL.IPC` enthält das Hauptprogramm.

```
#include <CONST.H>
H10 = MAXIMUM_SPEED;
```

Die Datei `CONST.H` ist eine Header-Datei.

```
#define MAXIMUM_SPEED 3000
```

Der Präprozessor ersetzt die `#include`-Direktive durch den Inhalt der Datei `CONST.H`:

```
#define MAXIMUM_SPEED 3000
H10 = MAXIMUM_SPEED
```

Nach der Makro-Erweiterung bleibt als Ergebnis:

```
H10 = 3000;
```

Die #include-Direktiven können auch verschachtelt angewendet werden, d. h. eine eingebundene Datei kann wiederum eine #include-Direktive enthalten, um eine weitere Datei einzubinden. Es ist darauf zu achten, dass sich Dateien nicht gegenseitig einbinden. Das führt zu einem Präprozessorfehler. Aus Gründen der Übersichtlichkeit sollte man auf die Verschachtelung von #include-Direktiven verzichten.

14.4 Verzeichnisse von include

Abhängig vom Verzeichnis in dem die einzubindende Datei abgelegt ist ergeben sich verschiedene Vorgehensweisen.

1. Falls der Pfad für die einzubindende Datei in der Registerkarte Verzeichnisse bei den Einstellungen zum Compiler eingestellt ist, dann lautet die Anweisung #include <DateiName> , wobei der DateiName der Name der Header-Datei ist.
2. Steht die einzubindende Datei im aktuellen Arbeitsverzeichnis, so lautet der Befehl #include "DateiName". DateiName steht wieder für den Namen der einzubindenden Datei.
3. Steht die einzubindende Datei in einem anderen als den hier bereits vorgestellten Verzeichnissen, so muss der Verzeichnispfad angegeben werden. Steht beispielsweise eine Header-Datei mit dem Namen Test.h im Hauptverzeichnis, so muss die Anweisung zum Einbinden dieser Datei '#include "c:\Test.h" ' lauten.

Header-Dateien die selbst erstellt worden sind werden sinnvollerweise im aktuellen Arbeitsverzeichnis abgelegt. Damit kann das Programm unabhängig vom Verzeichnispfad erstellt werden. Soll das Programm in einem anderen Verzeichnis übersetzt werden und wird die Verzeichnisstruktur des Programms beibehalten, so müssen keine Veränderungen vorgenommen werden. Das Programm kann sofort wieder übersetzt werden.

Bei der ersten Methode muss die Einstellung für den Compiler verändert werden und bei der dritten Methode muss im Programm der #include-Pfad editiert werden, um das Programm wieder übersetzen zu können.

HINWEIS

Werden alle drei Methoden gemischt verwendet, so werden die Pfade in folgender Reihenfolge durchsucht:

1. Direkte Pfadzuweisung in der #include-Anweisung (Methode 3)
2. Relative Pfadzuweisung zur Quelldatei (Methode 2)
3. Pfadzuweisung in der #include-Direktive im Einstellungsfenster.

14.5 #define

Die Direktive '#define' wurde bisher benutzt um einen Makro-Bezeichner bekannt zu machen. Grundsätzliche Aufgabe der Direktive '#define' ist es jedoch, ein Makro zu definieren. Makros dienen dazu, Symbole im Quelltext durch Zeichenketten zu ersetzen. Mit diesem Mechanismus lassen sich beispielsweise Konstanten, Variablen usw. symbolisch formulieren. Der Compiler unterstützt nur Makros ohne Parameter. Die Syntax lautet:

```
#define MakroBezeichner   <SymbolSequenz>
```

Jedes Auftreten von 'MakroBezeichner' im Quelltext nach dieser Direktive wird durch die möglicherweise leere 'SymbolSequenz' ersetzt. Bei einer leeren Symbolsequenz wird der MakroBezeichner bekannt gemacht, hat aber keine weitere Funktion. Die Symbolsequenz darf maximal 75 Zeichen beinhalten.

In der Header-Datei const.h wird auf diese Art und Weise den Systemvariablen eine symbolische Schreibweise zugeordnet. So kann nach dem Einbinden der Header-Datei const.h zum Beispiel die Variable H474 über den symbolischen Namen 'Scope474' oder die Variable H484 über 'ControlWord' angesprochen werden.

Ebenso können mit der Direktive '#define' konstanten Werten symbolische Namen zugeordnet werden. Die Zeile '#define MAXDREHZAHL 1500' ermöglicht es nun statt 1500 explizit 'MAXDREHZAHL' im Quelltext zu schreiben. Der Quelltext wird dadurch lesbarer.

Das folgende Beispiel soll dies verdeutlichen.

```
#define setpoint  H123
#define maximum   2000
setpoint = maximum;// in this line the macro def. "setpoint" and "maximum"
                   // are replaced, meaning: H123 = 2000;
```

Nach jeder erzeugten Makroerweiterung wird der erzeugte Text nochmals durchsucht. Dies erlaubt den Einsatz von geschachtelten Makros.

```
#define setpoint  H10
#define variable1 setpoint
#define minimum   20+H11
variable1 = minimum;// in this line the macro definition "variable1" is
                   // replaced by "setpoint", then "setpoint" is replaced by
                   // "H10", meaning: H10=20+H11;
```

HINWEIS

Stellen Sie sicher, dass eine mit #define bezeichnete Variable **nicht** unbeabsichtigt mit einer Systemvariable doppelt belegt ist.

HINWEIS

Wird eine Variable mit dem selben Namen wie eine Struktur angelegt, so wird das beim Compilieren nicht erkannt. Der Umrichter generiert Fehler 10 IPOS-ILLOP.

Beispiel:

```
#define Position H2VARINT Position;
```

14.6 #undef

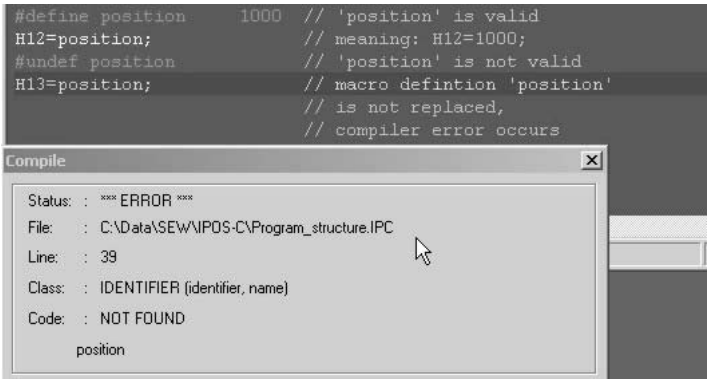
Mit dieser Direktive kann ein zuvor mit '#define ...' erzeugtes Makro außer Kraft gesetzt werden:

```
Syntax: #undef MakroBezeichner
```

Ein kleines Beispiel soll die Anwendung verdeutlichen:

```
#define position 1000// "position" is valid
H12=position;        // meaning: H12=1000;
#undef position      // "position" is not valid
H13=position;        // macro definition "position" is not replaced,
                     // compiler error occurs
```

Compiler-Fehler wegen fehlender Definition:



493925131

14.7 #declare

Mit dieser Direktive können IPOS$^{plus®}$-Variablen symbolisch und relativ zu einer Basis-Variablen deklariert werden. Dies erleichtert die Portierbarkeit von Quelltextmodulen bezüglich der Variablennummernvergabe, da zur Änderung aller im Quelltext verwendeten Variablennummern lediglich die Nummer der Basisvariablen geändert werden muss.

Auf diese Weise lassen sich vorgefertigte Module leicht in den eigenen Quelltext integrieren, wenn diese Module mit relativen Variablennummern versehen sind.

```
Syntax: #declare BezeichnerNeu  BezeichnerAlt : Offset
```

Ein Beispiel soll diese Funktion verdeutlichen:

```
#define basevariable  H100

#declare setpoint     basevariable:0
#declare actvalue     basevariable:1
#declare i            basevariable:5
```

Nun stehen die Variablen Sollwert, Istwert und i als symbolische Variablen zur Verfügung. Außerdem ist festgelegt worden, dass damit die IPOS$^{plus®}$-Variablen H100, H101 und H105 belegt werden.

HINWEIS

Insgesamt sind maximal 600 #define- und #declare-Direktiven zulässig.

HINWEIS

Die Integration von Modulen wird durch die Bildung von Variablenblöcken mit der declare-Direktiven zwar erleichtert, ist aber noch recht schwierig zu handhaben, da der Anwender einen Überblick über die belegten und noch zur Verfügung stehenden Variablen haben muss. Deshalb ist der Einsatz von Strukturen, SEW-Standardstrukturen oder anwenderdefinierten Strukturen besonders dann sinnvoll, wenn eine feste Variablenfolge bereitgestellt werden muss (z. B. SETSYS, GETSYS, MOVLNK ...). Alle anderen Variablen sollten mit den ebenfalls weiter unten beschriebenen Schlüsselwörtern long oder initial long deklariert werden, um dem Compiler eine Zuordnung der Variablennummer zu überlassen.

14.8 SEW-Standardstrukturen

Mit den SEW-Standardstrukturen stehen für die Befehle, die auf Strukturen angewiesen sind, bereits vorgefertigte Strukturen zur Verfügung.

Die folgende Tabelle zeigt eine Aufstellung der für die jeweilige Anweisung zur Verfügung stehenden Standardstrukturen. Die dazugehörigen Elemente finden Sie in der Beschreibung zu dem jeweiligen Befehl, siehe Standardfunktionen (Seite 208).

Anweisung	Standardstruktur
_GetSys	GSAINPUT
	GSAOUTPUT
	GSCAM
	GSCAM_EXT
	CAM_EXT_OUT
	GSPODATA3
	GSACTSPEEDEXT
	GSPODATA10
_MovCommDef (nur bei MQx)	MOVCOM
	MCPDATA
	MCPARDATA
_MoviLink	MOVLNK
	MLDATA
_SBusCommDef	SCREC
	SCTRACYCL
	SCTRCYCL
_SetSys	SSPOSRAMP
	SSPOSSPEED
	SSPIDATA3
	SSPIDATA10

Diese Standardstrukturen werden wie folgt verwendet. Zunächst wird im Deklarationsteil eine Variable als Strukturvariable deklariert. Anschließend werden die Elemente der Struktur angesprochen wie es in dem folgenden Beispiel erläutert wird. Innerhalb des Befehls wird die Struktur angesprochen, indem der Name der Strukturvariable ohne Zusätze verwendet wird.

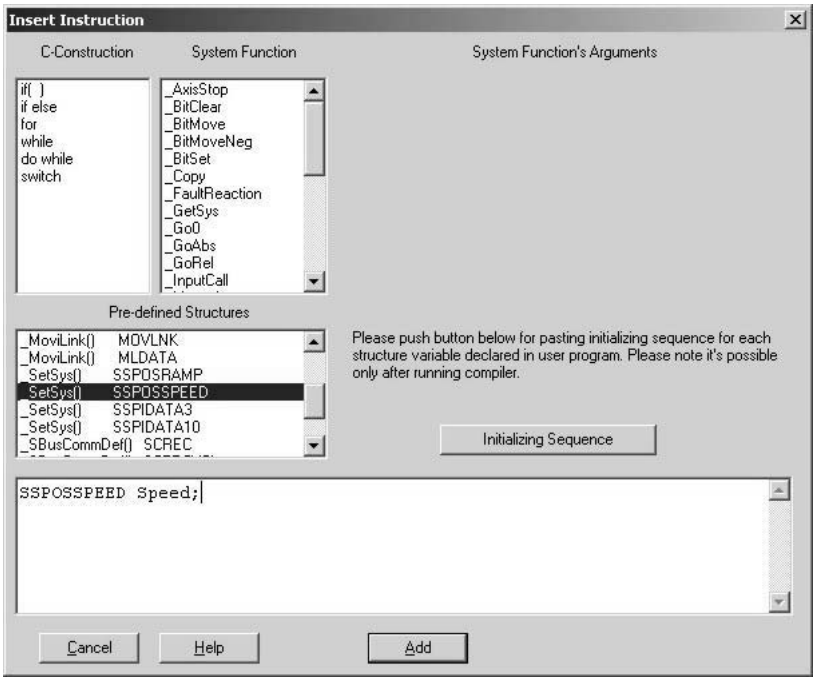
```
// Declare
SSPOSSPEED rapid speed, slow speed;

// Initiate
rapid_speed.cw = 14000;     // rapid speed cw 1400 rpm
rapid_speed.ccw = 12500;    // rapid speed ccw 1250 rpm
slow_speed.cw = 3000;       // slow speed cw 300 rpm
slow_speed.ccw = 4500;      // slow speed ccw 450 rpm

// set rapid speed
_SetSys( SS_POSSPEED,rapid speed );

// set slow speed
_SetSys( SS_POSSPEED,slow speed );
```

Zur Unterstützung bei der Eingabe von Deklaration und Initialisierung von Standardstrukturen kann die Eingabehilfe genutzt werden. Im ersten Schritt muss der Cursor im Editorfenster an die Stelle gesetzt werden, an der die Deklaration der Strukturvariablen eingefügt werden soll. Anschließend wird die Eingabehilfe aufgerufen, indem durch Klicken der rechten Maustaste ein Kontextmenü geöffnet und dort der Menüpunkt [Anweisung einfügen (Insert instruction)] ausgewählt wird.

Ein Aufruf der Eingabeunterstützung ist ebenfalls durch Betätigen des Buttons in der Symbolleiste oder durch den Aufruf des Menüpunktes [Anweisung einfügen (Insert instruction)] aus dem Menü [Bearbeiten (Edit)] der Menüleiste möglich. Nun wird die vordefinierte Struktur ausgewählt und dann kann der Name der Variablen in dem Editierfenster der Eingabehilfe geändert und ergänzt werden. Sollen mehrere Strukturvariablen vom gleichen Strukturtyp deklariert werden, so werden diese durch Komma getrennt.

Nachdem alle Strukturvariablen deklariert sind müssen diese je nach Anwendungsfall initialisiert werden. Auch dafür kann die Eingabehilfe zur Unterstützung verwendet werden. Setzen Sie den Cursor in dem Editorfenster an die Stelle, an der die Initialisierungs-Sequenz eingefügt werden soll. Compilieren Sie das Programm und rufen Sie anschließend die Eingabehilfe auf. Drücken Sie dann den Button [Initialisierungs-Sequenz (Initializing Sequence)]. Für jede deklarierte Strukturvariable wird nun ein Initialisierungsblock abgelegt.

Anweisung einfügen:



493929483

14.9 Anwenderdefinierte Strukturen

Ergänzend zu den SEW-Standardstrukturen kann sich der Anwender eigene Strukturen erstellen. Zunächst muss die Struktur angelegt werden. Dies geschieht im Deklarationsteil des Programms. Dazu wird das Schlüsselwort 'typedef struct' verwendet. An einem Beispiel, das eine Positionstabelle anlegt, soll dies erläutert werden.

```
// Define user structure
typedef struct
{
   long pos1;
   long pos2;
   long pos3;
   long pos4;
   long pos5;
} table;
```

Nun ist also eine Struktur mit dem Namen `table` angelegt worden. Jetzt kann diese Struktur verwendet werden, wie es bereits bei den Standardstrukturen erläutert worden ist. Im nächsten Schritt wird eine Variable als Strukturvariable deklariert.

```
// Declare structuretable postable;
```

Nun ist die Variable PosTabelle als eine Strukturvariable vom Strukturtyp `table` deklariert worden. Im nächsten Schritt soll auf die Elemente zugegriffen werden. Dazu soll die Tabelle initialisiert werden.

```
// Initiate
postable.pos1 = 100000;
postable.pos2 = 120000;
postable.pos3 = 50000;
postable.pos4 = 200000;
postable.pos5 = 10000;
```

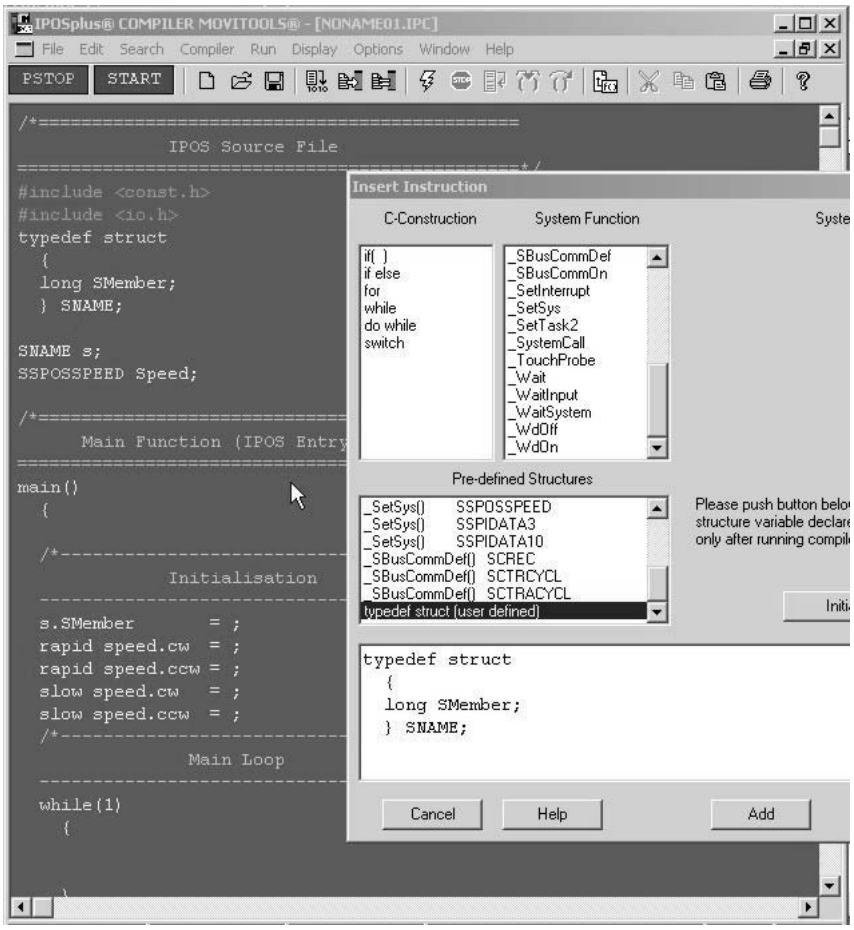
Ganz allgemein kann man den Aufbau einer Anwenderstruktur wie folgt beschreiben:

```
typedef struct
{
   Typ Bezeichner1;
   Typ Bezeichner2;
   ...
   Typ BezeichnerN;
} Strukturname;

Strukturname Variablenname;

Variablenname.Bezeichner1 = ...;
Variablenname.Bezeichner2 = ...;

Variablenname.BezeichnerN = ...;
```

Auch bei den anwenderdefinierten Strukturen kann die Eingabehilfe verwendet werden. Wählen Sie dazu aus dem Fenster vordefinierte Strukturen der Eingabehilfe die Zeile 'typedef struct (user defined)' aus. Nun können Sie die Namen der Elemente und den Namen der Struktur verändern.

Nach dem Einfügen ist dann auch eine Deklarationszeile im Editorfenster eingefügt die Strukturvariablen dieses Strukturtyps deklariert. Diese Zeile muss dann noch im Editor editiert werden. Ebenso können innerhalb des Strukturtyps noch weitere Elemente hinzugefügt werden. Eine Initialisierungs-Sequenz kann nach dem Compilieren auch wieder über die Eingabehilfe ausgelöst werden. Der Cursor muss zuvor an die Stelle im Editor gesetzt werden, an der die Initialisierungs-Sequenz eingefügt werden soll.

Strukturen:



493933835

14.10 long

Alternativ zur Variablenzuordnung mit #define kann das Schlüsselwort long zur Deklaration einer einzelnen Variable eingesetzt werden. Die Variablennummer wird in dem Fall beim Compilieren durch den Compiler vergeben.

Das Schlüsselwort long leitet eine Vereinbarung einer oder mehrerer globaler Variablen ein. Das folgende Beispiel zeigt die Verwendung des Schlüsselwortes.

Die Syntax einer Vereinbarung einer oder mehrerer globaler Variablen lautet:

```
long Bezeichner1 [, Bezeichner n] ;
```

Beispiel:

```
long setpoint,actual_value;
```

Beim Compilieren wird den symbolischen Variablen Sollwert und Istwert eine IPOS$^{plus®}$-Variable zugewiesen, wobei der Anwender immer durch die Verwendung des symbolischen Namens auf die Variable zugreift.

14.11 initial long

Als weiteres Schlüsselwort steht 'initial long' zur Verfügung. Mit 'initial long' wird eine Variable deklariert, die dann beim Compilieren in den Variablenbereich von H0 bis H127 abgelegt wird. Damit wird die Variable dann im netzausfallsicher speicherbaren Variablenbereich abgelegt.

Beispiel:

```
initial long start position, end position;
```

	HINWEIS Die Lesbarkeit des Programmtextes wird deutlich erhöht, wenn man einheitlich alle Konstanten in Großbuchstaben (z.B. SEKUNDE, MAXIMUM, ...) und Variablen in Groß-/Kleinschreibung (z.B. DrehzahlSollwert, PositionRechts, ...) schreibt.

14.12 #pragma

Mit der Direktive #pragma kann der Variablenbereich der mit den Schlüsselworten 'long' und 'initial long' belegt wird beeinflusst werden.

```
Syntax: #pragma  Direktive  Parameter1  Parameter2 ...
```

Der Compiler unterstützt folgende #pragma-Direktiven:

#pragma list	Bewirkt, dass die Quelltextzeilen als Kommentare in das erzeugte IPOS$^{plus®}$-Programm aufgenommen werden.
#pragma var Hmin Hmax	Weist den Compiler an, als Hilfsvariablen für die Berechnung von Ausdrücken die IPOS$^{plus®}$-Variablen Hmin bis einschließlich Hmax zu benutzen. Hmax muss größer sein als Hmin. Benutzt der Programmierer dieselben Variablen innerhalb des Programms, wird eine Fehlermeldung erzeugt. Erfahrungsgemäß braucht der Compiler etwa 10 Hilfsvariablen. Wird diese Direktive nicht explizit angegeben, benutzt der Compiler die Variablen H400 bis einschließlich H419.
#pragma globals Hmin Hmax	Weist den Compiler an, den mit dem Schlüsselwort long deklarierten Variablen eine Variablennummer aus dem Variablenbereich Hmin bis Hmax zuzuweisen. Der Anwender ist selbst dafür verantwortlich, Überschneidungen bei der symbolischen Vereinbarung von Variablennamen mit #define zu vermeiden. Wird diese Direktive nicht explizit angegeben, benutzt der Compiler die Variablen H420 bis einschließlich H449.
#pragma initials Hmin Hmax	Weist den Compiler an, den mit dem Schlüsselwort initial deklarierten globalen Variablen die Nummern Hmin bis einschließlich Hmax zuzuweisen. Initial-Variablen sind die netzaus-gespeicherten Variablen H0 bis H127. Ausnahme: H0 bis H15 bei MDS, MDV, MCS, MCV mit Kurvenscheibe. Wird diese Direktive nicht explizit angegeben, benutzt der Compiler die Variablen H0 bis einschließlich H127.

HINWEIS

Da bei Technologieoption "Synchronlauf" und "Kurvenscheibe" der Variablenbereich H360 bis H450 mit weiteren Systemvariablen belegt ist, wird empfohlen, die Hilfsvariablen und globalen Variablen immer mit der #pragma-Direktive in einen anderen Bereich zu legen.

Beispiel:

```
#pragma var 350 365
#pragma globals 130 160
#pragma initials 10 30

long pos speed cw, pos speed ccw;
initial long start position, end position;
```

Diese Zeilen führen dazu, dass der Compiler die IPOS$^{plus®}$-Variablen ab H350 als Hilfsvariablen verwendet. Die mit dem Schlüsselwort long deklarierten Variablen PosDrezahlRechts und PosDrehzahlLinks werden nun wegen der Befehlszeile #pragma globals 130 160 auf IPOS$^{plus®}$-Variablen zwischen H130 und H160 abgelegt. Die mit dem Schlüsselwort 'initial long' deklarierten Variablen StartPosition und EndPosition werden wegen der Befehlszeile #pragma initials 10 30 auf IPOS$^{plus®}$-Variablen zwischen H10 und H30 abgelegt. Da diese damit also im Variablenbereich von H0 bis H127 liegen, können diese Variablen netzausfallsicher gespeichert werden.

HINWEIS

Die **Variablennummern** werden bei Verwendung innerhalb der Direktiven #pragma **ohne** den voran gestellten Buchstaben **H** verwendet.

14.13 Erläuterung zur const.h und io.h / constb.h und iob.h

Die Header-Datei const.h definiert sehr viele nützliche Bezeichner. Für den Einsteiger sind zunächst nur die symbolischen Namen der Systemvariablen von Interesse. Die weiteren Bezeichner sind wichtig für den geübten Anwender der die Eingabeunterstützung nicht mehr verwendet. Er findet dort die Definitionen der Argumente für die Aufrufe der Standardfunktionen.

Die Header-Datei io.h ist ebenso wie die Datei const.h eine fest vorgegebene Datei die nicht verändert werden darf. Die Datei ist nachfolgend abgedruckt.

```
/*==============================================
File name: Io.h
File version: 2.01

Definition of bitmasks for digital in- and outputs

Please do not modify this file!

(C) 1999 SEW-EURODRIVE
==============================================*/

#ifndef  _IO_H
#define  _IO_H

#define DI00 (H483 & 0b1)
#define DI01 (H483 & 0b10)
#define DI02 (H483 & 0b100)
#define DI03 (H483 & 0b1000)
#define DI04 (H483 & 0b10000)
#define DI05 (H483 & 0b100000)
#define DI10 (H483 & 0b1000000)
#define DI11 (H483 & 0b10000000)
#define DI12 (H483 & 0b100000000)
#define DI13 (H483 & 0b1000000000)
#define DI14 (H483 & 0b10000000000)
#define DI15 (H483 & 0b100000000000)
#define DI16 (H483 & 0b1000000000000)
#define DI17 (H483 & 0b10000000000000)

#define DB00 (H482 & 0b1)
#define DO01 (H482 & 0b10)
#define DO02 (H482 & 0b100)
#define DO10 (H482 & 0b1000)
#define DO11 (H482 & 0b10000)
#define DO12 (H482 & 0b100000)
#define DO13 (H482 & 0b1000000)
#define DO14 (H482 & 0b10000000)
#define DO15 (H482 & 0b100000000)
#define DO16 (H482 & 0b1000000000)
#define DO17 (H482 & 0b10000000000)

#endif
```

In der Header-Datei io.h sind Makros definiert, die die Abfrage von Klemmenpegeln erleichtern. Dies soll am folgenden Beispiel erläutert werden.

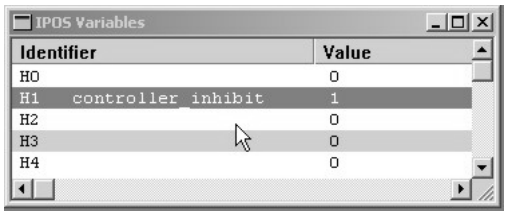
```
if( DI00 )
{
   H1 = 1;  // execute command block, if terminal DI00 has NOT level 0
}
else
{
   H1 = 0;  // execute command block, if terminal DI00 has level 0
}
```

Mit der if-Anweisung wird die Klemme DI00, die Klemme /REGLERSPERRE, abgefragt. Ist das Argument der if-Anweisung null, so werden die Anweisungen im else-Teil abgearbeitet, falls ein solcher else-Teil vorhanden ist. Hier wird in Abhängigkeit des Eingangspegels der Klemme DI00 die IPOS$^{plus®}$-Variable H1 auf null oder eins gesetzt. Es ist zu beachten, dass die Abfrage nicht auf eins (DI00 == 1) erfolgen kann, da das Makro eine Binärauswertung liefert. Praktisch ist eine Abfrage auf null (DI00 == 0) oder ungleich null (DI00 != 0) möglich.

Dieses Programmstück kann mit den inzwischen bekannten Befehlen übersichtlicher gestaltet werden, indem man weitere symbolische Bezeichner einführt.

```
#define controller_inhibit    H1
#define HI                    1
#define LO                    0

if( DI00 )
{
   controller_inhibit = 1;  // execute command block, if terminal DI00 has NOT
                            //level 0
}
else
{
   controller_inhibit = 0;  // execute command block, if terminal DI00 has
                            //level 0
}
```

Im Variablenfenster ergibt sich nun folgende Darstellung:



499400587

	HINWEIS Es ist zu beachten, dass die Header-Datei io.h mit der Befehlszeile #include io.h vor ihrer Benutzung eingebunden werden muss.

14.14 Bezeichner

Obwohl Bezeichner nun schon mehrfach verwendet worden sind, soll dieser Abschnitt dazu nochmals einige Hinweise geben. Unter einem Bezeichner versteht man den Namen, den ein MakroBezeichner (Kapitel Define), ein symbolischer Variablenname oder ein Funktionsname annehmen kann. Innerhalb eines Bezeichners dürfen nur Buchstaben, Ziffern und der Unterstrich verwendet werden, wobei der Bezeichner mit einem Buchstaben oder Unterstrich beginnen muss.

Ein Bezeichner darf maximal 32 Zeichen lang sein.

Folgende Bezeichner sind gültig:	Folgende Namen sind keine Bezeichner:
KlemmeX13_4 Sollwert1 _Steuer_Wort	KlemmeX13.4 Sollwert 1 1.Sollwert 1_Eingang Meine Funktion DieIstEinVielZuLangerBezeichnerName

HINWEIS

Der IPOS$^{plus®}$-Compiler unterscheidet Groß-/Kleinschreibung.

14.15 Konstanten

Der IPOS$^{plus®}$-Compiler unterstützt verschiedene Typen von Konstanten, die im Quelltext durch ihre spezifische Schreibweise unterschieden werden. Je nach Verwendungszweck kann die Darstellung in unterschiedlichen Formaten die Lesbarkeit des Quelltextes verbessern.

Als Darstellungsformen sind die Formate dezimal, hexadezimal oder binär möglich.

Hexadezimal-Konstanten beginnen mit der Zeichenfolge '0x', Binärkonstanten mit der Zeichenfolge '0b'. Hier einige Beispiele:

Dezimalkonstanten	Hexadezimal-Konstanten		Binärkonstanten	
123	0x23	=35 dez	0b000100	= 4 dez
-50	0xabc	= 2748 dez	0b10	= 2 dez
030	0xFFFFFFFF	= -1 dez	0b11111111	= 255 dez

14.16 IPOS$^{plus®}$-Variablen im Compiler

Die IPOS$^{plus®}$-Variablen sind praktisch Bestandteil der Sprache und dürfen nicht explizit deklariert werden. Sie besitzen alle denselben Datentyp (32 Bit, vorzeichenbehaftet) und sind global im gesamten Quelltext gültig. Die folgende Zeile ist quasi implizit in jedem Modul vorhanden:

```
long H0, H1, H2, H3, ... ,H1023.
```

Um Variablen symbolisch zu bezeichnen, besteht die Möglichkeit, mittels der #define- oder #declare-Direktive einen symbolischen Namen zu definieren.

14.16.1 Beispiel

```
#define TESTVAR1 H73 //H73 erhält den symbolischen Namen "TESTVAR1"
```

H73 erhält dann im Programm den Wert 134 durch eine der 3 Zuweisungen:

```
TESTVAR1 = 134;
TESTVAR1 = 0x86;
TESTVAR1 = 0b10000110;
```

14.17 Die Vereinbarung globaler Variablen

Eine andere Möglichkeit besteht darin globale Variablen mit dem Schlüsselwort long zu deklarieren, wie dies oben bereits erläutert worden ist. Der Compiler legt dann eigenständig die Nummern der Variablen (siehe #pragma) fest. Die Variablennummern werden aufsteigend, dem Auftreten der Variablendeklarationen im Quelltext entsprechend, vergeben. Eine Deklaration beginnt mit dem Schlüsselwort long, gefolgt von der Aufzählung der mit Komma getrennten, symbolischen Bezeichner. Die Deklaration endet mit einem Semikolon. Die Deklaration kann über mehrere Quelltextzeilen verteilt sein.

Eine globale Variable darf an beliebiger Stelle im Programm, aber nur außerhalb von Blöcken (i.a. Funktionen) deklariert werden. Aus Gründen der besseren Übersicht sollten Variablen am Beginn des Quelltextmoduls deklariert werden. Es gilt weiterhin, dass eine globale Variable vor ihrer Verwendung deklariert sein muss.

Beispiele:

```
long a, b;                // Variablen liegen in dem für
long this_is_a_variable;  // globale Var. definierten Bereich
long c, d;
```

Der Compiler ordnet den Bezeichnern a, b, this_is_a_variable, c und d die Variablen H420 bis H424 zu. Im folgenden Beispiel weist die Direktive '#pragma globals' den Compiler an, den Bezeichnern die Variablen H150 bis H160 zu benutzen.

```
#pragma globals 150 160
long a, b;
long this_is_a_variable;
long c, d;
```

	HINWEIS
	Nicht erlaubt sind mehrfache Vereinbarungen globaler Variablen:
	`long a, b, c;`
	`long d, a;`

Die Möglichkeit der globalen Variablendeklaration ist dann vorteilhaft, wenn die Nummer der Variablen für die Anwendung nicht von Bedeutung ist. Im Regelfall ist dies so, da auf die Variablen immer wieder symbolisch zugegriffen wird. Außerdem wird durch die globale Variablendeklaration die Modularität unterstrichen und eine Weiterverwendung von Modulen erleichtert. Sind Variablengruppen z.B. für Systemfunktionen erforderlich, so ist auf Standard- oder Anwenderstrukturen zurückzugreifen.

Die Beschreibung der Systemvariablen H473 bis H511 ist dem Kapitel "Übersicht über die Systemvariablen" zu entnehmen. Eine Aufstellung der Systemvariablen und ihrer symbolischen Bezeichner ist im Anhang abgedruckt.

14.18 Indirekte Adressierung – Pointer

Um die Möglichkeit von IPOS$^{plus®}$ auszunutzen und auf Variablen indirekt zuzugreifen (SET [H] = H), sind als Variablennamen auch die Bezeichnungen *H0 bis *H511 zulässig. Diese Namen können genauso auf der rechten und linken Seite von Zuweisungen oder in Ausdrücken verwendet werden, wie H0 bis H511. Der Compiler setzt in diesen Fällen aber die indirekten Befehle ein.

Ein Beispiel zur Verwendung indirekter Variablen:

```
H2 = 5;
H3 = 6;

H5 = 7;
H6 = 3;

H1 = *H2 + *H3;
```

Die Variable H1 erhält den Wert 10, da über *H2 indirekt auf den Wert 7 (siehe H5) und über *H3 indirekt auf den Wert 3 (siehe H6) zugegriffen wird.

14.19 numof()

Das Schlüsselwort numof() liefert die Nummer einer Variablen. Als Argument wird der Bezeichner einer direkten oder symbolischen Variablen angegeben. Das Argument darf kein zusammengesetzter Ausdruck sein.

```
#define setpoint H200
#declare setpoint2 setpoint:1

H1 = numof (H7);
H2 = numof (setpoint);
H3 = numof (setpoint2);
```

Diese Programmzeilen liefern auf den IPOS$^{plus®}$-Variablen folgende Werte:

```
H1 = 7
H2 = 200
H3 = 201
```

HINWEIS

Nicht zulässig sind die folgenden Programmzeilen:

```
#define Sollwert H10+H30
#define Var1     H200

H1 = numof(*H1);
H1 = numof(H1 + H4);
H1 = numof(H3 + 6);
H1 = numof(Sollwert);
H1 = numof(Var1:1);
```

15 Compiler – Operatoren

Mit Operatoren werden Bezeichner miteinander zu Anweisungen verknüpft, um so bestimmte Operationen auszuführen. Der IPOS$^{plus®}$-Compiler stellt Operatoren zur Verfügung um arithmetische, Bit-, Zuweisungs- oder Vergleichsoperationen durchzuführen.

Die Operatoren sind in verschiedene Kategorien eingeteilt und besitzen untereinander eine bestimmte Rangfolge. Diese Rangfolge bestimmt die Reihenfolge, in der die Operatoren innerhalb einer Anweisung ausgeführt werden. Die folgende Tabelle listet alle vom IPOS$^{plus®}$-Compiler unterstützten Operatoren in der Rangfolge auf.

15.1 Rangfolge der Operatoren

Kategorie	Operator	Beschreibung
1.	()	Klammerung
2. Unär	! ~ + - ++ --	Logische Negation (NOT)Bitweises Komplement Unäres Plus Unäres Minus Prä- oder Postinkrementierung Prä- oder Postdekrementierung
3. Multiplikativ	* / %	Multiplikation Ganzzahldivision Modulo Rest
4. Additiv	+ -	Binäres Plus Binäres Minus
5. Shift	<< > >	Linksschieben Rechtsschieben
6. Relational	< <= > > =	Kleiner Kleiner gleich Größer Größer gleich
7. Gleichheit	== !=	Gleich Ungleich
8.	&	Bitweises UND
9.	^	Bitweises XOR
10.	¦	Bitweises ODER
11.	&&	Logisches UND
12.	¦¦	Logisches ODER
13. Bedingt	? :	Ternärer Operator siehe Kapitel "Ternäre Operatoren"
14. Zuweisung	= *= /= %= += -= &= ^= ¦= <<= > > =	Einfache Zuweisung Produkt zuweisen Quotient zuweisen Rest zuweisen Summe zuweisen Differenz zuweisen Bitweises UND zuweisen Bitweises XOR zuweisen Bitweises ODER zuweisen Linksschieben zuweisen Rechtsschieben zuweisen
15. Komma	,	Auswerten

Die Kategorie 1 hat die höchste Priorität; die Kategorie 2 (Unäre Operatoren) hat die zweithöchste Priorität usw. bis zum Komma-Operator, der die niedrigste Priorität besitzt.

Die Operatoren innerhalb einer Kategorie haben den gleichen Rang.

Die unären (Kategorie 2), bedingten (Kategorie 13) und Zuweisungsoperatoren (Kategorie 14) ordnen von rechts nach links zu, alle anderen von links nach rechts.

Da der Operator für die Multiplikation (*) einen höheren Rang besitzt als der Additions-Operator (+), wird in der nachfolgenden Anweisung die Multiplikation vor der Addition ausgeführt.

```
H1 = 3 * 7 + 2 * 4;
```

H1 erhält den Wert 29.

Soll die Addition vor der Multiplikation ausgeführt werden, so erreicht man das durch Setzen von Klammern:

```
H1 = 3 * ( 7 + 2 ) * 4;
```

H1 erhält den Wert 108 (aus 3 * 9 *4).

	HINWEIS Um eine gewünschte Reihenfolge der Operationen zu gewährleisten, kann man die Rangfolge über eine Klammerung erzwingen. Ineinander verschachtelte Klammern sind zulässig. Überflüssige Klammerungen haben keinen Einfluss auf die Programmfunktion.

15.2 Unäre Operatoren

Unäre Operatoren sind Operatoren, die vor oder hinter einem Operand stehen und nur diesen beeinflussen.

```
H1 = -H2;// der unäre Minusoperator(-) bildet den Wert von H2 mit verändertem Vorzeichen
H1 = ~H2;// der unäre Komplement-Operator (~) bildet das bitweise Komplement von H2
++H3;    // der Präinkrementoperator (++) erhöht den Wert von H3 um eins
```

15.3 Binäre Operatoren

Diese Operatoren verbinden zwei Operanden miteinander bzw. stehen zwischen zwei Operanden.

15.3.1 Beispiel

```
H1 = H2;    // der binäre Zuweisungsoperator(=) weist der Variablen H1 den Wert von H2 zu
H1 = H2 - 3;// der binäre Minusoperator(-) bildet die Differenz zwischen H2 und 3
```

Die kombinierten Zuweisungsoperatoren führen zu einer abkürzenden Schreibweise, die die Lesbarkeit eines Programms verringern, aber der Vollständigkeit halber erwähnt werden sollen. Im Beispiel wird mit H1 = 2 (0b10) und H2 =3 (0b11) die Operation durchgeführt.

Operator	Operation	Beispiel	Entsprechung	Beispiel H1 =
=	Einfache Zuweisung	H1 = H2;	H1 = H2;	3
*=	Produkt zuweisen	H1 *= H2;	H1 = H1 * H2;	6
/=	Quotient zuweisen	H1 /= H2;	H1 = H1 / H2;	0
%=	Rest zuweisen	H1 %= H2:	H1 = H1 % H2;	2
+=	Summe zuweisen	H1 += H2;	H1 = H1 + H2;	5
-=	Differenz zuweisen	H1 -= H2;	H1 = H1 - H2;	-1
&=	Bitweises UND zuweisen	H1 &= H2;	H1 = H1 & H2;	0b10
^=	Bitweises XOR zuweisen	H1 ^= H2;	H1 = H1 ^ H2;	0b01
¦=	Bitweises ODER zuweisen	H1 ¦= H2;	H1 = H1 ¦ H2;	0b11
<<=	Linksschieben zuweisen	H1 <<= H2;	H1 = H1 << H2;	0b1000
> > =	arithmetisch Rechtsschieben zuweisen	H1 > > = H2;	H1 = H1 > > H2;	0b0

15.4 Ternäre Operatoren

Der IPOS$^{plus®}$-Compiler kennt nur einen Operator, der drei Operanden miteinander verknüpft: den Bedingt-Operator. Er besitzt die Form

a ? b : c bedeutet: wenn a dann b, sonst c,

wobei a ein logischer Ausdruck ist und b und c Ausdrücke sind.

15.4.1 Beispiel

```
H1 = H2 == 3 ? H3 : H4; // wenn H2 gleich 3 ist, wird H1 der Wert von H3,
                        // sonst der Wert von H4 zugewiesen
```

Das Beispiel ist die verkürzende Schreibweise von:

```
if (H2 == 3)
   H1 = H3;
else
   H1 = H4;
```

Auf den ternären Operator sollte aufgrund der schlechten Lesbarkeit des Quellcodes grundsätzlich verzichtet werden.

16 Compiler – Konstrukte

Der IPOS$^{plus®}$-Compiler stellt Konstrukte zur Verfügung, die auch aus anderen Hochsprachen bekannt sind.

Als Konstrukte stehen zur Verfügung:

- if...else
- for
- while
- do...while
- switch...case...default

Hinzu kommen noch Anweisungen wie 'continue' und 'break' die zur Steuerung innerhalb dieser Konstrukte Anwendung finden.

16.1 if...else

16.1.1 Syntax

```
if ( Ausdruck )// Anweisungelse// Anweisung
```

Die Schlüsselwörter if und else steuern den Programmfluss abhängig davon, ob der nach dem Schlüsselwort if angegebene Ausdruck den Wert WAHR (ungleich Null) oder FALSCH (gleich Null) liefert. Der else-Zweig ist optional. Er wird dann ausgeführt, wenn der Ausdruck den Wert FALSCH liefert. Eine Anweisung kann im speziellen Fall auch ein Block sein, in dem mehrere Anweisungen angegeben werden können. In diesem Fall muss der Anweisungsblock durch geschweifte Klammern eingeschlossen sein ({ Anweisungsblock}).

Ohne else-Zweig	Mit else-Zweig	Mit Block als if-Zweig	Mit Block als else-Zweig

`if ( H1 == 3 )`
`  H2 = 10;`
	`if ( H1 == 3 )`
	`  H2 = 10;`
	`else`
	`  H2 = 8;`
		`if ( H1 >  3 )`
		`{`
		`  H2 = 10;`
		`  H3 = 11;`
		`}`
			`if ( H1 >  3 )`
			`  H2 = 9;`
			`else`
			`{`
			`  H2 = 10;`
			`  H3 = 11;`
			`}`

Der Ausdruck kann sich auch aus mehreren Bedingungen zusammensetzen, die miteinander logisch verknüpft sind. Als Verknüpfung steht somit das logische UND (&&) und das logische ODER (||) zur Verfügung.

HINWEIS

Fehlerquelle: Ein ; (Semikolon) am Ende der if-Anweisung macht die Bedingung immer wahr.

Beispiel

```
if ( ( H1 > = 3 ) && ( H1 <= 12 ) )
   H2 = 10;
```

Die Variable H2 wird auf den Wert 10 gesetzt, wenn H1 größer oder gleich 3 ist und gleichzeitig kleiner oder gleich 12 ist. Mit anderen Worten: H2 wird auf den Wert 10 gesetzt, wenn H1 einen Wert zwischen 3 und 12 annimmt.

Die inneren Klammern sind nicht notwendig, erhöhen aber die Lesbarkeit des Programms.

Beispiel

```
if ( H1 < 2 || H1 >  14 )
   H2 = 10;
```

Die Variable H2 wird auf den Wert 10 gesetzt, wenn H1 kleiner als 2 oder größer als 14 ist. Mit anderen Worten: H2 wird auf den Wert 10 gesetzt, wenn H1 einen Wert außerhalb des Bereichs von 2 bis 14 annimmt.

16.2 for

16.2.1 Syntax

```
for ( Ausdruck1; Ausdruck2; Ausdruck3 )
// Anweisung
```

Mit der for-Anweisung lassen sich Programmschleifen konstruieren, die nach einer definierten Wiederholung verlassen werden. Dabei haben die drei Ausdrücke die folgende Bedeutung:

Der Ausdruck1 wird einmalig zu Beginn der for-Schleife ausgeführt. Dort wird die Initialisierung der Laufvariablen durchgeführt. Der Ausdruck2 bestimmt, wann die Schleife abgebrochen wird. Liefert der Ausdruck den logischen Wert FALSCH (bzw. gleich Null), dann wird die Schleife abgebrochen. Der Ausdruck3 wird nach Ausführung der Anweisung verarbeitet und wird üblicherweise zum Verändern der Laufvariablen verwendet. Die Anweisung bildet den Schleifenkörper, der aus einer Anweisung oder aus einem Anweisungsblock bestehen kann.

Beispiel

```
H1 = 20;
for ( H0 = 0; H0 < 10; ++H0 )
   H1 = H1 + 2;
```

Zu Beginn wird H0 auf Null gesetzt. Anschließend wird getestet, ob H0 den Wert 10 erreicht hat. Ist das nicht der Fall, dann wird die Anweisung abgearbeitet, also in diesem Beispiel wird H1 um 2 erhöht. Danach wird die Laufvariable H0 um eins erhöht. Dann erfolgt wieder der Test, ob H0 den Wert 10 erreicht hat usw.

Nach Ende der Schleife hat H0 den Wert 10, H1 den Wert 40, da die Schleife 10 mal (Schleifenzähler H0 läuft von 0 bis 9, dann erfolgt der Abbruch) durchlaufen wird.

Beispiel mit Anweisungsblock

```
H1 = 20;
H2 = 0;
for ( H0 = 0; H0 < 10; ++H0 )
{
   H1 = H1 + 2;
   ++H2;
}
```

Während die Variable H1 bei jedem Schleifendurchlauf um zwei erhöht wird, wird die Variable H2 nur um eins erhöht (++H2 bedeutet Präinkrement).

Wird innerhalb des Anweisungsblocks eine continue-Anweisung verarbeitet, so bedeutet dies, dass das Programm an das Ende des Anweisungsblocks springt, dann den Ausdruck3 verarbeitet und dadurch eine Überprüfung der Bedingung in der for-Anweisung für einen erneuten Schleifendurchlauf erfolgt.

Beispiel

```
H1 = 20;
H2 = 0;
for ( H0 = 0; H0 < 10; ++H0 )
{
   H1 = H1 + 2;
   if ( H1 >  32 )
      continue;
   ++H2;
}
```

Die if-Abfrage mit der Anweisung continue bewirkt, dass die Inkrementierung der Variablen H2 aussetzt, sobald H1 größer als 32 ist. Nach dem vollständigen Ablauf der Schleife hat nun also die Variable H1 den Wert 40 und die Variable H2 den Wert 6.

Wird innerhalb des Anweisungsblocks eine break-Anweisung verarbeitet, so bedeutet dies, dass das Programm die for-Schleife an dieser Stelle verlässt. Es erfolgt kein weiterer Schleifendurchlauf.

Beispiel

```
H1 = 20;
H2 = 0;
for ( H0 = 0; H0 < 10; ++H0 )
{
   H1 = H1 + 2;
   if ( H1 >  32 )
      break;
   ++H2;
}
```

Die if-Abfrage mit der Anweisung break bewirkt, dass die for-Schleife verlassen wird, sobald H1 größer als 32 ist. Nach dem Verlassen der Schleife hat nun die Variable H1 den Wert 34 und die Variable H2 den Wert 6.

16.3 while

16.3.1 Syntax

```
:  while ( Ausdruck )
// Anweisung
```

Die while-Anweisung ist eine bedingte Schleife, die die Anweisung solange ausführt, wie der Ausdruck den Wert WAHR (ungleich Null) besitzt. Besitzt der Ausdruck niemals den Wert WAHR, wird die Anweisung niemals ausgeführt. Der Ausdruck wird grundsätzlich vor der Abarbeitung der Anweisung verarbeitet.

Anweisung kann auch ein Anweisungsblock sein, in dem mehrere Anweisungen angegeben werden können.

Der Ausdruck kann sich auch aus mehreren logisch miteinander verknüpften Bedingungen zusammensetzen.

Beispiel

```
H2 = 0;
H1 = 10;
while ( H1 >  5 )
{
   H2 = H2 + 1;
   --H1;
}
```

Solange H1 größer ist als 5, werden die Anweisungen innerhalb des Blocks ausgeführt. Nach dem Abbruch der Schleife enthält H2 den Wert 5.

Ebenso wie bei der for-Schleife können auch hier die Anweisungen break und continue eingesetzt werden. Die break-Anweisung führt dabei wieder zum Verlassen der while-Schleife, die continue-Anweisung zu einem Sprung ans Ende des Anweisungsblocks und damit zu einer erneuten Überprüfung des Ausdrucks für einen weiteren Schleifendurchlauf.

Beispiel

```
H1 = 0;
while ( H1 < 20 )
{
   ++H1;
   if ( H1 >  10 )
      continue;
   H2 = H2 + 2;
}
```

Solange H1 kleiner oder gleich 10 ist, wird H2 um 2 erhöht. Hat H1 einen Wert größer 10, dann wird an das Ende der while-Schleife gesprungen und die Bedingung für einen weiteren Schleifendurchlauf getestet. H2 wird bei den weiteren Schleifendurchläufen nicht weiter verändert. Besitzt H1 den Wert 20, dann wird die Schleife verlassen.

Beispiel

```
H1 = 0;
while ( 1 )
{
   ++H1;
   if ( H1 == 20 )
      break;
   if ( H1 >  10 )
      continue;
   H2 = H2 + 2;
}
```

Dieses Beispiel bewirkt dasselbe wie das zuvor dargestellte Beispiel. Die quasi "Endlos-Schleife" wird mittels break-Anweisung verlassen, wenn H1 gleich 20 ist.

Hier ist also auch gleich eine Endlos-Schleife vorgestellt worden. Sie lässt sich also mit dem Konstrukt

```
while ( 1 )
// Anweisung
```

herstellen, da der Ausdruck 1 immer den Wert WAHR liefert.

Typischerweise läuft der Task1 in solch einer Endlos-Schleife.

16.4 do...while

16.4.1 Syntax

```
do
// Anweisungen der Schleife
while ( Ausdruck );
```

Die do-Anweisung ist eine bedingte Schleife, bei der die Abbruchbedingung am Ende der Schleife überprüft wird. Deshalb führt diese Schleife mit der do-Anweisung mindestens eine Iteration aus (mindestens ein Durchlauf).

Zuerst wird Anweisung ausgeführt, die als Anweisungsblock auch mehrere Anweisungen enthalten kann. Anschließend erfolgt der Test, ob der Ausdruck den Wert WAHR (ungleich Null) oder FALSCH (gleich Null) besitzt. Ist der Wert WAHR, wird wieder die Anweisung ausgeführt, sonst wird die Schleife verlassen.

Der Ausdruck kann sich auch aus mehreren logisch miteinander verknüpften Bedingungen zusammensetzen.

Im Gegensatz zur while-Schleife wird in der do-while-Schleife die Anweisung mindestens einmal ausgeführt.

HINWEIS

Fehlerquelle: Die while (...);-Zeile wird immer mit Semikolon abgeschlossen.

Beispiel

```
H2 = 0;
H1 = 10;
do
{
   H2 = H2 +1;
   H1 = H1 -1;
} while ( H1 >  5 );
```

Solange H1 größer als 5 ist, werden die Anweisungen innerhalb des Blocks ausgeführt. Nach Abbruch der Schleife enthält H2 den Wert 4.

Bleibt der Ausdruck immer WAHR, erhält man eine Endlos-Schleife:

```
do
   H2 = H2 +3;
while ( 1 );
```

Der Ausdruck hat in diesem Falle den Wert 1. Somit wird die Schleife nie abgebrochen.

Ein Abbruch dieser Schleife ist wiederum mit der break-Anweisung möglich.

```
H2 = 0;
do
{
   H2 = H2 +3;
   if ( H2 >  20 )
      break;
} while ( 1 );
```

In diesem Beispiel wird die do-Schleife mit Hilfe der break-Anweisung verlassen, wenn der Wert der IPOS$^{plus®}$-Variablen H2 größer als 20 ist.

Auch eine continue-Anweisung ist möglich. Sie bewirkt wieder einen Sprung zum Ende des Anweisungsblocks und dann eine Überprüfung des Ausdrucks.

```
H2 = 0;
do
{
   H2 = H2 +3;
   if ( H2 >  20 )
      break;
   if ( H2 >  10 )
      continue;
   ++H0;
} while ( 1 );
```

In diesem Beispiel wird das Inkrementieren der IPOS$^{plus®}$-Variablen H0 nicht mehr durchgeführt, sobald die IPOS$^{plus®}$-Variable H2 einen Wert größer als 10 hat.

16.5 switch...case...default

16.5.1 Syntax

```
switch ( Ausdruck )
{
   case Wert 1:// Anweisung 1
      break;
   case Wert 2:// Anweisung 2
      break;
    .
    .
   default:// Anweisung n
}
```

Mit der switch-Anweisung lassen sich Mehrfach-Programmverzweigungen in Abhängigkeit vom Wert eines Ausdrucks erzeugen.

Besitzt Ausdruck den Wert 1, wird Anweisung 1 ausgeführt, besitzt Ausdruck den Wert 2, wird Anweisung 2 ausgeführt usw. Stimmt keiner der hinter case stehenden Werte mit Ausdruck überein, wird - falls vorhanden - die default-Anweisung (Anweisung n) ausgeführt.

HINWEIS

Anweisung 1, 2,..., n sind üblicherweise Folgen von Anweisungen, die mit einer break-Anweisung enden. Endet die Anweisungsfolge nicht mit einer break-Anweisung, werden alle folgenden case-Zweige ausgeführt, bis eine break-Anweisung erreicht wird. Dabei wird der Wert mit dem Ausdruck dann nicht mehr verglichen.

Anweisung 1, 2,..., n können auch Funktionsaufrufe sein. So kann beispielsweise ein Sprungverteiler aufgebaut werden.

Wert 1, Wert 2, Wert n müssen Konstanten oder konstante Ausdrücke sein. Hier sind keine Variablen erlaubt.

Der default-Zweig muss - falls vorhanden - als letzter innerhalb einer switch-Anweisung stehen.

```
switch ( H1 )
{
   case 1: ++H2;
      break;
   case 2: ++H3;
      break;
   default: ++H4;
      break;
}
```

Dieses Programmstück inkrementiert die IPOS$^{plus®}$-Variable H2, wenn die IPOS$^{plus®}$-Variable H1 den Wert 1 hat. Hat sie den Wert 2, so wird die IPOS$^{plus®}$-Variable H3 inkrementiert. Bei jedem anderen Wert der IPOS$^{plus®}$-Variablen H1 wird die IPOS$^{plus®}$-Variable H4 inkrementiert.

Auch die folgende Variante ist möglich:

```
switch ( H1 )
{
   case 1:
   case 2: ++H3;
      break;
   default: ++H4;
      break;
}
```

Dieses Programmstück inkrementiert die IPOS$^{plus®}$-Variable H3, wenn die IPOS$^{plus®}$-Variable H1 den Wert 1 oder 2 hat. Bei jedem anderen Wert der IPOS$^{plus®}$-Variablen H1 wird die IPOS$^{plus®}$-Variable H4 inkrementiert.

16.6 return

Das Schlüsselwort `return` beendet die Bearbeitung einer Funktion und kehrt zu dem Befehl zurück, der dem Funktionsaufruf folgt. Die `return`-Anweisung erlaubt ein vorzeitiges Beenden von Funktionen, um z. B die Übersichtlichkeit eines C-Programms zu erhöhen. Eine zu häufige Benutzung dieser Anweisung könnte allerdings auch das Gegenteil bewirken. Allgemein gilt: Eine Funktion sollte sowenig Austrittspunkte wie möglich beinhalten.

Das folgende Beispiel zeigt zwei Möglichkeiten der Kodierung, um dasselbe Ergebnis zu erreichen. Während das linke Beispiel die `return`-Anweisung verwendet, um die Funktion vorzeitig zu verlassen, kommt das rechte Beispiel ohne `return` aus.

```
Funktion ()
{
   // Funktion verlassen, wenn H1
gleich 5
   if (H1 == 5)
      return;
   H2=3;
}
```

```
Funktion ()
{
   // Anweisung überspringen, wenn H1
= 5
   if (H1 != 5)
   {
      H2=3;
   }
}
```

17 Compiler – Funktionen

17.1 Anwenderdefinierte Funktionen

Der Anwender kann Funktionen (Unterprogramme) programmieren. Ein Aufruf mit der Übergabe von Funktionsargumenten ist bei den vom Anwender erstellten Funktionen nicht möglich aber auch nicht nötig, da alle Variablen global sind und keine Kapselung mit lokalen Variablen möglich ist. Eine Funktion hat folgenden Aufbau:

```
FunktionsName()
{
   // Anweisungen
}
```

Das Definieren von Funktionsprototypen im Programmkopf entfällt im Gegensatz zu ANSI-C.

Nachdem die Funktion abgearbeitet worden ist, wird die nächste Zeile nach dem Funktionsaufruf abgearbeitet.

Eine Funktion kann mit Hilfe der return-Anweisung vorzeitig beendet werden. Dann wird die Abarbeitung des Programms wiederum mit der nächsten Zeile nach dem Funktionsaufruf fortgesetzt.

Die Verwendung der return-Anweisung kann die Übersichtlichkeit eines Programmes erhöhen. Eine zu häufige Benutzung dieser Anweisung kann allerdings auch das Gegenteil bewirken. Allgemein gilt: Eine Funktion sollte sowenig Austrittspunkte wie möglich beinhalten.

```
Beispielfunktion()
{
   //Funktion verlassen, wenn H1 gleich 5
   if( H1 == 5 )
      return;
   H2=3;
}
```

HINWEIS

Eine anwenderdefinierte Funktion darf nicht von mehreren Tasks aufgerufen werden.

17.2 Befehlsübersicht Standardfunktionen

Ein großer Teil der IPOS$^{plus®}$-Maschinenbefehle die aus der IPOS$^{plus®}$-Assembler-Sprache bekannt sind werden durch die Hochsprache des IPOS$^{plus®}$-Compilers in Form bestimmter syntaktischer Konstruktionen abgebildet. So werden z.B. arithmetische Befehle (ADD, SUB, ...) durch entsprechende Operatoren (+, -, ...) erzeugt oder die Setzbefehle (SET...) durch den Zuweisungsoperator (=) ersetzt. Es existieren aber auch Befehle, für die kein Äquivalent in der Programmiersprache existiert. Diese Befehle (GOA, BSET,...) werden durch Funktionen nachgebildet, die Bestandteil des Compilers sind und deshalb im Gegensatz zu anwenderdefinierten Funktionen als Standardfunktionen bezeichnet werden.

Die Parameter der IPOS$^{plus®}$-Maschinenbefehle werden zu Funktionsargumenten der Standardfunktionen. Die Namen aller Standardfunktionen beginnen mit einem Unterstrich (_), um sie im Quelltext leichter von Anwenderfunktionen unterscheiden zu können.

Die bei vielen Funktionen als Argument angegebenen Konstanten sind in der Header-Datei CONSTB.H (MOVIDRIVE® B) definiert. Sollen statt dessen eigene Bezeichnungen verwendet werden, können diese mit der #define-Direktive angelegt werden.

In den folgenden Tabellen sind die Standardfunktionen in Gruppen zusammengefasst und die gerätespezifische Verfügbarkeit angegeben.

17.2.1 Bit-Standardfunktionen

Befehl	Funktion	Verfügbarkeit			Verweis
		MOVIDRIVE® B	MOVITRAC® B	MQx	
_BitClear	Löscht ein Bit innerhalb einer Variablen.	X	X	X	(Seite 209)
_BitMove	Kopiert ein Bit einer Variablen auf ein Bit in einer anderen Variablen.	X	X	X	(Seite 209)
_BitMoveNeg	Kopiert ein Bit einer Variablen auf ein Bit einer anderen Variablen und negiert es.	X	X	X	(Seite 209)
_BitSet	Setzt ein Bit innerhalb einer Variablen.	X	X	X	(Seite 210)

17.2.2 Kommunikations-Standardfunktionen

Befehl	Funktion	Verfügbarkeit			Verweis
		MOVIDRIVE® B	MOVITRAC® B	MQx	
_MoviLink	Prozess- und/oder Parameter-Datenaustausch über RS-485 oder Systembus.	gerätespezifische Befehlsstruktur beachten			(Seite 222)
_MovCommDef	Prozessdatenübertragung über RS-485 (Speziell mit MQx - MOVIMOT®).	-	-	X	(Seite 227)
_MovCommOn	Start der Prozessdatenübertragung über RS-485.	-	-	X	(Seite 229)
_SBusCommDef	Definition des Prozessdatenaustausches über Systembus.	X	X	-	(Seite 230)
_SBusCommOn	Start der Prozessdatenübertragung über Systembus.	X	X	-	(Seite 234)
_SBusCommState	Start der Prozessdatenübertragung über Systembus	X	(X)	-	(Seite 234)

17.2.3 Positionier-Standardfunktionen

Befehl	Funktion	Verfügbarkeit			Verweis
		MOVIDRIVE® B	MOVITRAC® B	MQx	
_Go0	Führt eine Referenzfahrt durch.	X	-	-	(Seite 217)
_GoAbs	Positioniert absolut.	X	-	-	(Seite 218)
_GoRel	Positioniert relativ.	X	-	-	(Seite 219)

17.2.4 Programm-Standardfunktionen

Befehl	Funktion	Verfügbarkeit			Verweis
		MOVIDRIVE® B	MOVITRAC® B	MQx	
_InputCall	Ruft eine definierte Funktion auf, wenn an den Eingangsklemmen bestimmte ausmaskierte Bits gesetzt oder gelöscht sind.	X	X	X	(Seite 220)
_Nop	Keine Operation.	X	X	X	(Seite 229)
_SystemCall	Ruft eine definierte Funktion auf, wenn das Systemereignis auftritt.	X	X	-	(Seite 241)
_SetTask	Definiert eine Funktion als Task 2 oder Task 3 und startet oder stoppt diese.	X	nur für Task 2	-	(Seite 238)
_SetTask2	Definiert eine Funktion als Task 2 und startet oder stoppt diese.	X	X	X	(Seite 239)
_Wait	Wartet eine angegebene Zeit.	X	X	X	(Seite 243)
_WaitInput	Wartet so lange, bis an bestimmten Eingangsklemmen ein bestimmter Pegel anliegt.	X	X	X	(Seite 243)
_WaitSystem	Wartet so lange, bis ein Systemereignis auftritt.	X	X	-	(Seite 244)

17.2.5 Setz-Standardfunktionen

Befehl	Funktion	Verfügbarkeit			Verweis
		MOVIDRIVE® B	MOVITRAC® B	MQx	
_Copy	Blockweises, konsistentes Kopieren von Variablen.	X	X	X	(Seite 210)
_GetSys	Liest eine interne Systemgröße.	gerätespezifische Befehlsstruktur beachten			(Seite 211)
_SetInterrupt	Definiert eine Funktion als Interrupt-Routine und aktiviert bzw. deaktiviert sie.	X	X	X	(Seite 235)
_SetVarInterrupt	Definiert eine Funktion als Variablen-Interrupt und aktiviert bzw. deaktiviert sie (nur MOVIDRIVE® B).	X	-	-	(Seite 240)
_SetSys	Setzt eine interne Systemgröße.	gerätespezifische Befehlsstruktur beachten			(Seite 236)

17.2.6 Spezielle Geräte-Standardfunktionen

Befehl	Funktion	Verfügbarkeit			Verweis
		MOVIDRIVE® B	MOVITRAC® B	MQx	
_AxisStop	Der Antrieb wird angehalten.	X	-	-	(Seite 208)
_FaultReaction	Setzt die Fehlerreaktion bei einem ausgewählten Fehler.	X	X	-	(Seite 210)
_Memorize	Speichern oder Laden von Variablen oder IPOS$^{plus®}$-Programm.	X	X	X	(Seite 221)
_TouchProbe	Gibt einen Touch-Probe-Eingang frei oder sperrt diesen.	X	-	X	(Seite 242)
_WdOn	Setzt den Watchdog-Timer auf einen bestimmten Wert.	X	X	X	(Seite 245)
_WdOff	Schaltet den Watchdog aus.	X	X	X	(Seite 244)

17.3 Standardfunktionen

In diesem Kapitel werden die Standardfunktionen in alphabetischer Reihenfolge dargestellt, um so beim Nachschlagen ein Auffinden der Standardfunktion zu erleichtern.

17.3.1 _AxisStop

Syntax

```
_AxisStop( typ )
```

Beschreibung

Die Antriebsachse wird angehalten, indem das IPOS$^{plus®}$-Steuerwort beschrieben wird. Ein erneutes Starten muss durch Freigabe über das IPOS$^{plus®}$-Steuerwort erfolgen. Über das Argument kann die Art des Stoppens der Achse bzw. die Aufhebung der Verriegelung über das IPOS$^{plus®}$-Steuerwort angegeben werden.

Argument

typ

AS_RSTOP	Abbremsen an der Stopp-Rampe P136, anschließend im Zustand "keine Freigabe". Die zuletzt abgesetzte Zielposition H492 bleibt erhalten. Verriegelung über Steuerwort (Befehl ASTOP (IPOS ENABLE) vor anschließendem Verfahrbefehl notwendig). Bei aktivierter Bremsenfunktion fällt die Bremse ein. Die Meldung "In Position" wird gesetzt.
AS_HCTRL	Abbremsen an der Rampe des Grundgeräts P131 / P133, anschließend Lageregelung, die zuletzt abgesetzte Zielposition bleibt erhalten, Verriegelung über Steuerwort (Befehl ASTOP (IPOS ENABLE) vor anschließendem Verfahrbefehl notwendig). Bei aktivierter Bremsenfunktion fällt die Bremse **nicht** ein.
AS_PSTOP	Positionierstopp mit Positionierrampe P911 / P912 und berechneter "STOP"-Zielposition (nur in Betriebsart Positionierung möglich), anschließend Lageregelung. Die zuletzt abgesetzte Zielposition H492 wird mit der Stopp-Position überschrieben, keine Verriegelung über Steuerwort (kein Befehl ASTOP (IPOS ENABLE) vor anschließendem Verfahrbefehl notwendig). Bei aktivierter Bremsenfunktion fällt die Bremse **nicht** ein. Hinweis: Da im Stillstand die Ist-Position als Sollposition übernommen wird, darf der Befehl nicht zyklisch bearbeitet werden. Bei Achsen mit Prozesskräften oder Hubwerken driftet sonst die Achse langsam weg. **Achtung:Steht ein Fehler an, wird "AS_PSTOP" nicht zu Ende geführt. Das Programm bleibt stehen.**
AS_ENABLE	Die Verriegelung wird mittels IPOS$^{plus®}$-Steuerwort aufgehoben.

HINWEIS

Da im Stillstand die Ist-Position als Sollposition übernommen wird, darf der Befehl mit dem Argument AS_PSTOP nicht zyklisch bearbeitet werden. Bei Achsen mit Prozesskräften oder Hubwerken driftet die Achse sonst langsam weg.

Beispiel

```
main()
{
   _GoAbs( GO_NOWAIT,3000 );     // Fahrauftrag starten
   _AxisStop( AS_PSTOP );        // Fahrvorgang abbrechen
                                 // Anweisungen während Stillstand
   _AxisStop( AS_ENABLE );       // Verriegelung aufheben
   _GoAbs( GO_NOWAIT,3000 );     // Fahrauftrag neu absetzen
}
```

17.3.2 _BitClear

Syntax

```
_BitClear( H, bit )
```

Beschreibung

_BitClear setzt innerhalb der Variablen H das Bit bit auf Null.

Argumente

H Variablenname

bit konstanter Ausdruck für Bitposition

Beispiel

```
main()
{
   _BitClear( H100, 3 );      // löscht Bit 3 in H100
}
```

17.3.3 _BitMove

Syntax

```
_BitMove( H2 , bit2, H1, bit1 )
```

Beschreibung

Kopiert das Bit mit der Nummer bit1 von IPOS$^{plus®}$-Variable H1 in das Bit mit der Nummer bit2 von IPOS$^{plus®}$-Variable h2. Alle Bits von H1 und alle anderen Bits von H2 bleiben unverändert. Die Bitstellen einer Variablen besitzen die Nummern 0 bis 31. Das niederwertigste Bit hat die Nummer 0.

Argumente

H2 Name der Ziel-Variable

bit2 Nummer des Ziel-Bits

H1 Name der Quell-Variable

bit1 Nummer des Quell-Bits

Beispiel

```
main()
{
   _BitMove( H1, 3,H2, 4 );     // kopiert H1.3 = H2.4
   _BitMove( H1, 1,H1, 0 );     // kopiert H1.1 = H1.0
}
```

17.3.4 _BitMoveNeg

Syntax

```
_BitMoveNeg( H2 , bit2, H1, bit1 )
```

Beschreibung

Kopiert das Bit mit der Nummer bit1 von IPOS$^{plus®}$-Variable H1 in das Bit mit der Nummer bit2 von IPOS$^{plus®}$-Variable H2, wobei das Bit dabei negiert wird. Alle Bits von H1 und alle anderen Bits von H2 bleiben unverändert. Die Bitstellen einer Variablen besitzen die Nummern 0 bis 31. Das niederwertigste Bit hat die Nummer 0.

Argumente

H2 Name der Ziel-Variable

bit2 Nummer des Ziel-Bits

H1 Name der Quell-Variable

bit1 Nummer des Quell-Bits

Beispiel

```
main()
{
   _BitMoveNeg( H1, 3,H2, 4 );     // kopiert H1.3 = NOT (H2.4)
   _BitMoveNeg( H1, 1,H1, 0 );     // kopiert H1.1 = NOT (H1.0)
}
```

17.3.5 _BitSet

Syntax

```
_BitSet( H, bit )
```

Beschreibung

_BitSet setzt innerhalb der IPOS$^{plus®}$-Variablen h das Bit mit der Nummer bit auf Eins.

Argumente

H Variablenname

bit Konstanter Ausdruck mit der Nummer des zu setzenden Bits

Beispiel

```
main()
{
   _BitSet( H100, 3 );        // setzt Bit 3 in H100
}
```

17.3.6 _Copy

Syntax

```
_Copy( H2 , H1, anz )
```

Beschreibung

Kopiert in der Anzahl (anz) aufeinanderfolgende Variablen als Variablenblock. H1 gibt den Namen der ersten Quell-Variablen, H2 den Namen der ersten Ziel-Variablen an. Es können maximal 10 Variablen kopiert werden.

Argumente

H2 Name der ersten Ziel-Variable

H1 Name der ersten Quell-Variable

anz konstanter Ausdruck für Anzahl der zu kopierenden IPOS$^{plus®}$-Variablen

Beispiel

```
main()
{
   _Copy( H1,H5, 3 );         // kopiere H1 = H5, H2 = H6, H3 = H7
}
```

17.3.7 _FaultReaction

Syntax

```
_FaultReaction( fnr, r )
```

Beschreibung

Dieser Befehl erlaubt die Programmierung der Reaktion auf einen Gerätefehler. Dabei muss der Befehl vor dem Auftreten des Fehlers ausgeführt werden. In den Argumenten werden der Fehler und die entsprechende Reaktion beim Auftreten dieses Fehlers angegeben.

Es sind alle Fehler-Reaktionen programmierbar, die in der Fehlerliste der Betriebsanleitung oder des Systemhandbuchs in der Spalte P einen Punkt haben.

Argumente

fnr konstanter Ausdruck für Nummer des Fehlers (siehe Fehlerliste in der Betriebsanleitung)

r konstanter Ausdruck für Fehlerreaktion, der folgende Werte annehmen kann:

FR_NORESP: Keine Reaktion, auch keine Anzeige des Fehlers.
FR_DISPLAY: Fehler wird lediglich angezeigt; Gerät weiterhin lauffähig.
FR_SWOFF_F: Endstufensperre und Geräteverriegelung. Reset notwendig.
FR_ESTOP_F: Stopp an Notstopp-Rampe mit Geräteverriegelung. Reset notwendig.
FR_RSTOP_F: Stopp an Schnellstopp-Rampe mit Geräteverriegelung. Reset notwendig.
FR_SWOFF_W: Endstufensperre ohne Geräteverriegelung. Reset gibt Gerät wieder frei.
FR_ESTOP_W: Stopp an Notstopp-Rampe ohne Geräteverriegelung. Reset gibt Gerät wieder frei.
FR_RSTOP_W: Stopp an Schnellstopp-Rampe ohne Geräteverriegelung. Reset gibt Gerät wieder frei.

FR_SWOFF_F / FR_ESTOP_F / FR_RSTOP_F: Das Gerät initialisiert neu, d. h. IPOS$^{plus®}$ startet neu.

FR_SWOFF_W / FR_ESTOP_W / FR_STOP_W: Das Gerät initialisiert nicht neu, d. h. IPOS$^{plus®}$ arbeitet weiter.

Beispiel

```
main()
{
   _FaultReaction( 26,FR_SWOFF_F ); // Notstopp / Störung bei ext. Fehler
}
```

17.3.8 _GetSys

Syntax

```
_GetSys( H, sys )
```

Beschreibung

Lädt den Wert einer internen Systemgröße in eine oder mehrere IPOS$^{plus®}$-Variablen.

Argumente

H Name der Ziel-Variable oder Ziel-Struktur

sys Ausdruck, der die Systemgröße bezeichnet. sys kann einen der folgenden Werte annehmen:

GS_ACTCUR: Wirkstrom in 0,1 % Gerätenennstrom
GS_ACTSPEED: Ist-Drehzahl in 0,1 min^{-1}
GS_SPSPEED: Solldrehzahl in 0,1 min^{-1}
GS_ERROR: Fehlercode entsprechend der Tabelle "Fehlermeldungen und Fehlerliste" im Systemhandbuch
GS_SYSSTATE: Wert der 7-Segment-Anzeige entsprechend der Tabelle "Betriebsanzeigen" im Systemhandbuch
GS_ACTPOS: Ist-Position abhängig von dem in P941 ausgewählten Geber H509, H510 oder H511
GS_SPPOS: Sollposition H491
GS_TPOS: Zielposition des Profilgenerators
GS_INPUTS: Binäre Eingänge H483 (MOVIDRIVE® A) / H520 (MOVIDRIVE® B) Grundgerät und Optionen
GS_DEVSTATE: Identisch mit Statuswort 1 des Feldbus-Geräteprofils (Fehlercode und Betriebszustand)
GS_OUTPUTS: Binäre Ausgänge H482 Grundgerät und Optionen
GS_IxT: Geräteauslastung in 0,1 % Gerätenennstrom
GS_ACTPOS / GS_SPPOS / GS_TPOS: Auflösung abhängig von mit P941 ausgewählten Geber:
– Motorgeber: 4096 Ink./Umdrehung
– Externer Geber X14: Geberauflösung × P944
– DIP (SSI-Geber): Geberauflösung × P955

GS_ANINPUTS: Spannungswert / Stromwert der Analogeingänge 1 und 2

Spannungseingang: -10 V ... 0 ... +10 V = -10000 ... 0 ... 10000

Stromeingang: 0 ... 20 mA = 0 ... 5000 / 4 ... 20 mA = 1000 ... 5000

– H + 0 = Analogeingang 1
– H + 1 = Analogeingang 2

Nur für MOVITRAC® B:

GS_ANINPUTS:

– H + 0 = Analogeingang 1
– H + 1 = Potenziometer des FBG11B

GS_ANINPUTS3:

– H + 0 = Analogeingang 1
– H + 1 = Potenziometer des FBG11B
– H + 2 = Analogeingang 2 (Option FIO11B)

Das Attribut GS_ANINPUTS3 des Befehls _GetSys erzeugt bei MOVIDRIVE® B einen "Fehler 10 IPOS ILLOP".

GS_CAM: Dient zur Realisierung eines Nockenschaltwerkes

– Pro Antrieb kann mit dem GETSYS-Befehl ein Standard-Nockenschaltwerk mit 4 Ausgängen und bei MOVIDRIVE®-Geräten ein erweitertes Nockenschaltwerk mit 8 Ausgängen genutzt werden.
– Hxx ist die erste Variable einer Datenstruktur (CamControl oder GS_CAM). Mit dem höchstwertigen Bit 31 in Hxx wird entschieden, auf welches Nockenschaltwerk sich der GETSYS-Befehl bezieht.
– Bit 31 = 0: Standard-Nockenschaltwerk. Der GETSYS-Befehl aktiviert das Nockenschaltwerk, die Nocken werden, wenn der GETSYS-Befehl bearbeitet wird, einmalig gebildet. Soll das Nockenschaltwerk zyklisch arbeiten, muss der Befehl zyklisch aufgerufen werden.
– Bit 31 = 1: Erweitertes Nockenschaltwerk mit Technologieoption. Der GETSYS-Befehl aktiviert das Nockenschaltwerk, die Nocken werden zyklisch im Hintergrund gebildet.
– Weitere Informationen zu den Nockenschaltwerken und der Datenstruktur finden Sie im Abschnitt "Nockenschaltwerke" im Kapitel "Wegerfassung und Positionierung".

GS_ANOUTPUTS: Analogausgänge optional, wobei -10 V ... 0 ... +10 V = -10000 ... 0 ... 10000.

– h = Analogausgang 1
– h + 1 = Analogausgang 2

GS_TIMER0: Zählwert von TIMER 0 H489 in ms

GS_TIMER1: Zählwert von TIMER 1 H488 in ms

GS_PODATA: Lesen des PA-Daten-Buffers, abhängig von der Anzahl der PA-Daten werden 3 PA-Daten oder 10 PA-Daten gelesen (vom Master an das Gerät gesendete Daten).

– H + 0: Bustyp
  0 = reserviert
  1 = S0 (RS485 #1)
  2 = S1 (RS485 #2)
  3 = Feldbus
  4 = reserviert
  5 = SBus
  8 = SBus 2 (nur MOVIDRIVE® B)
– H + 1 = Anzahl PA-Daten
– H + 2 = PA1
– H + 3 = PA2
– h + 4 = PA3
– H + 5 = PA4
– H + 6 = PA5
– H + 7 = PA6
– H + 8 = PA7
– H + 9 = PA8
– H + 10 = PA9
– H + 11 = PA10

GS_DCVOLT: Zwischenkreis-Spannung [V]

GS_RELTORQUE: Relatives Moment. Die Größe steht in den Betriebsarten CFC... und SERVO... zur Verfügung.

GS_RELTORQUEVFC: Das relative Moment ist der auf den Gerätenennstrom bezogene Anzeigewert für das Drehmoment an der Motorabtriebswelle in 0,1 % Gerätenennstrom. Aus dieser Größe lässt sich das absolute Drehmoment nach der folgenden Formel berechnen:

$M_{abs} = M_{rel} \times I_N \times M_N / 1000 / I_{QN}$

M_{abs} = absolutes Moment

I_N = Gerätenennstrom

M_{rel} = relatives Moment bezogen auf 0,1 % I_N

M_N = Bemessungs-Drehmoment des Motors [Nm]

I_{QN} = Nenn-Q-Strom [A] für gewählte Schaltungsart. Die Größe steht in den Betriebsarten CFC und SERVO / VFC1, VFC1 & Hubwerk, VFC1 & DC-Bremsung und VFC1 & Fangen zur Verfügung.

GS_ACTSPEEDEXT: Ist-Drehzahl des externen Gebers (X14)

- H = Time-Base, Mittelwertfilter für die Drehzahlerfassung des externen Gebers. Einstellbereich: 5 ms ... 31 ms
- H + 1 = Encoder Typ
  - 0 = Geber X14,
  - 1 = DIP Geber
- H + 2 = Numerator, Zähler für die Anwenderskalierung Wertebereich: -2^{15} ... 0 ... $+(2^{15} - 1)$
- H + 3 = Denominator, Nenner für die Anwenderskalierung Wertebereich: 1 ... $(2^{15} - 1)$
- H + 4 = DPointer, Zeiger auf die Ergebnisvariable H", wobei H" = Result; Einheit: [nX14] = (Inc/Time-Base)

Beispiel: Die Geschwindigkeit soll in Bögen pro Stunde angegeben werden. Dazu wurde eine Struktur `GS_ACTSPEEDEXT gLAActSpeed;` deklariert.

```
gLAActSpeed.TimeBase = 30;    // Mittelwertfilter 30 ms
gLAActSpeed.EncType = 0;      // Geber ist an X14 angeschlossen
gLAActSpeed.Numerator = -11250; // Umrechnung in Bögen pro Stunde 11250 / 384
gLAActSpeed.Denominator = 384;  // = (1000 ms x 60 s x 60 min) / (Ink. x Time-
Base)
gLAActSpeed.DPointer = numof(hBogenProStunde); // -11250 invertierte Darstel-
lung

_GetSys(gLAActSpeed,GS_ACTSPEEDEXT);
```

SPEEDMONITOR

Zählerwert der Drehzahl-Überwachung

Der GETSYS-Befehl kann als Vorwarnung für Drehzahl-Überwachung verwendet werden. Die Drehzahl-Überwachung spricht an, wenn der Strom P501 Sekunden in der Stromgrenze ist. Ist z. B. P501 = 200 ms, so kann mit GETSYS der Zählerwert abgefragt werden und z. B. nach 50 ms die Verfahrdrehzahl reduziert werden. Damit ist es möglich eine Leerfahrt im Eilgang zu fahren und unter Last die Drehzahl im Umrichter intern automatisch zu reduzieren.

Für die Anweisung _GetSys stehen folgende SEW-Standard-Strukturen zur Verfügung:

Anweisung	Standardstruktur	Elemente	Kurzbeschreibung
_GetSys	GSAINPUT	Input1	Spannungswert des Analogeingangs 1
		Input2	Spannungswert des Analogeingangs 2
	GSAOUTPUT	Output1	Spannungswert für optionalen Analogausgang 1
		Output2	Spannungswert für optionalen Analogausgang 2
	GSCAM	SourceVar	Nummer der Variable, auf die der Befehl angewendet wird
		DbPreCtrl	Totzeitvorsteuerung in 0.1 ms
		DestVar	Nummer der Variable, die Ergebnis aufnehmen soll
		BitPosition	Bitposition in Ergebnisvariable
		BitValue	Polarität in Ergebnisvariable
		NumOfCam	Anzahl der Nockenblöcke (max. 4)
		PosL1	Linker Grenzwert Nockenblock 1
		PosR1	Rechter Grenzwert Nockenblock 1
		PosL2	Linker Grenzwert Nockenblock 2
		PosR2	Rechter Grenzwert Nockenblock 2
		PosL3	Linker Grenzwert Nockenblock 3
		PosR3	Rechter Grenzwert Nockenblock 3
		PosL4	Linker Grenzwert Nockenblock 4
		PosR4	Rechter Grenzwert Nockenblock 4
	GSCAM_EXT	CamControl	Bit 31 muss immer gesetzt sein. 0x8000 0000 = Funktion inaktiv, die Ausgänge der Nocken werden nicht mehr neu gebildet, gesetzte Ausgänge bleiben erhalten und werden erst nach einem Reset oder Spannung Aus/Ein gelöscht. 0x8000 0001 = Funktion intern aktiv, aber alle Nockenausgänge werden ausgeschaltet 0x8000 0002 = Funktion aktiv, wenn Antrieb referenziert ist (H473, Bit20 =1) 0x8000 0003 = Funktion auch ohne referenzierten Antrieb aktiv
		CamReserved1	Reserviert
		CamOutShiftLeft	Schiebt den internen Datenpuffer der Ausgänge vor dem Schreiben auf die Zielvariable CamDestination um n Stellen nach links. Achtung: Beim Schieben geht die Information der oberen Ausgänge verloren. D. h. wenn Sie um 3 schieben, sind die oberen 3 Ausgänge mit 4 ms Zykluszeit nicht mehr nutzbar und die 4 Ausgänge mit 1 ms Zykluszeit sind dann den Bits 3-6, der eine Ausgang mit 4 ms Zykluszeit dem Bit 7 zugeordnet.
		CamForceOn	Maske um Ausgänge zwingend zu setzen, die Maske wirkt auf den internen Datenpuffer vor dem Shiften mit CamOutShiftLeft (NICHT auf die mit CamDestination definierte Zielvariable)
		CamForceOff	Maske um Ausgänge zwingend zu löschen, die Maske wirkt auf den internen Datenpuffer vor dem Shiften mit CamOutShiftLeft (NICHT auf die mit CamDestination definierte Zielvariable) CamForceOff ist dominant gegenüber CamForceOn

Anweisung	Standardstruktur	Elemente	Kurzbeschreibung
		CamSource	Bit 31 schaltet zwischen voreingestellten Bezugsvariablen und einem Zeiger auf eine beliebige Bezugsvariable um. Bit 31 = 0: • 0 = Geber X15 (Motorgeber, H511) • 1 = Geber X14 (externer Geber, H510) • 2 = Geber H509 (Absolutgeber DIP11) • 3 = virtueller Geber • alle folgenden Werte sind reserviert! Bit 31 = 1: CamSource enthält einen Zeiger auf eine IPOS$^{plus®}$-Variable + 2^{31}
		CamDestination	Zeiger auf Zielvariable. Die in dem Wort der Zielvariablen nicht benutzten Bits können für andere Funktionen verwendet werden (Schiebt man z.B. mit Shift Left die Ausgänge um 4 nach links, kann man Bits 0-3 frei verwenden, Bit 4-7 sind für die Nockenfunktion und Bit 8-31 kann man frei verwenden. Werden die Ausgänge der Nocken auf Geräteausgänge (z.B. H481) gelegt, sind diese Binärausgänge mit P620 - P639 als IPOS$^{plus®}$-Ausgänge zu reservieren. Die in diesem Wort nicht benutzten Bits können für andere Ausgänge verwendet werden.
		CamOutputs	Anzahl der Ausgänge (max. 8)
		CamData 1	Zeiger auf erste CamOutput-Struktur (1. Ausgang)
		...	
		CamData 8	Zeiger auf letzte CamOutput-Struktur (8. Ausgang)
	CAM_EXT_OUT	DeadTime	Totzeitkompensation für diesen Kanal (-500 ms ... 0 ... +500 ms), zur Kompensation der Totzeit eines am Umrichter angeschlossenen Aktors. Abhängig von der Änderungsgeschwindigkeit des Wertes der Bezugsvariable wird der Ausgang so vorgesteuert, dass der Ausgang um diese Zeit vorher geschaltet wird.
		CamAreas	Anzahl der Positionsfenster für diesen Kanal (1 ... 4), der linke Grenzwert muss immer kleiner sein wie der rechte Grenzwert, wird bei einer Modulo-Achse ein Positionsfenster über die 360° - 0°-Grenze benötigt, muss dieser Bereich in 2 Positionsfenster unterteilt werden. Damit können für diesen Ausgang max. 3 zusammenhängende Bereiche eingestellt werden.
		LeftLimit1	Linke Grenze, Fenster 1
		RightLimit1	Rechte Grenze, Fenster 1
		...	...
		LeftLimit4	Linke Grenze, Fenster 4
		RightLimit4	Rechte Grenze, Fenster 4
	GSPODATA3	BusType	0 = reserviert 1 = S0 (RS485 #1) 2 = S1 (RS485 #2) 3 = Feldbus 4 = reserviert 5 = SBus 8 = SBus 2
		Len	Anzahl der Prozessausgangsdaten
		PO1	Prozessausgangsdatum 1
		PO2	Prozessausgangsdatum 2
		PO3	Prozessausgangsdatum 3

Anweisung	Standardstruktur	Elemente	Kurzbeschreibung
	GSACTSPEEDEXT	TimeBase	Abtastzeit für die Drehzahlerfassung des externen Gebers, Einstellbereich: 5 ms ... 31 ms
		EncType	0 = Geber X14,1 = DIP Geber
		Numerator	Zähler für die Anwenderskalierung Wertebereich: -2^{15} ... 0 ... $+(2^{15}-1)$
		Denominator	Nenner für die Anwenderskalierung Wertebereich: 1 ... $+(2^{15}-1)$
		DPointer	Zeiger auf die Ergebnisvariable H"
	GSPODATA10	BusType	0 = reserviert 1 = S0 (RS485 #1) 2 = S1 (RS485 #2) 3 = Feldbus 4 = reserviert 5 = SBus 8 = SBus 2
		Len	Anzahl der Prozessausgangsdaten
		PO1	Prozessausgangsdatum 1
		PO2	Prozessausgangsdatum 2
		PO3	Prozessausgangsdatum 3
		PO4	Prozessausgangsdatum 4
		PO5	Prozessausgangsdatum 5
		PO6	Prozessausgangsdatum 6
		PO7	Prozessausgangsdatum 7
		PO8	Prozessausgangsdatum 8
		PO9	Prozessausgangsdatum 9
		PO10	Prozessausgangsdatum 10

Gerätespezifischen Besonderheiten:

Element	Gerätespezifische Besonderheiten	
	MOVIDRIVE® B	**MOVITRAC® B**
BusType (H+0)	GS_BT_S0 (RS485 an XT) GS_BT_S1 (RS485 an X13) GS_BT_FBUS=3 (Feldbus-Option) GS_BT_SBUS1 (SBus an X12) GS_BT_SBUS2 (über DFC11B)	nur GS_BT_S1 (RS485 an FSC/FIO11B) GS_BT_SBUS1 (SBus an FSC/FIO21B)

Beispiel

```
#include <const.h>

GSAINPUT Ain;

main()
{
   _GetSys( Ain,GS_ANINPUTS ); // Analogeingänge in Struktur Ain einlesen
}
```

17.3.9 _Go0

Syntax

```
_Go0( typ )
```

Beschreibung

Dieser Befehl löst eine Referenzfahrt der Achse aus. Das Argument legt das Verhalten bei der Referenzfahrt fest. Der Referenzfahrttyp wird mit P903 fest eingestellt und kann nur dort geändert werden.

Argumente

typ Ausdruck für die Einstellung des Verhaltens während der Referenzfahrt. typ kann einen der folgenden Werte annehmen:

GO0_C_W_ZP
GO0_U_W_ZP
GO0_C_NW_ZP
GO0_U_NW_ZP
GO0_C_W_CAM
GO0_U_W_CAM
GO0_C_NW_CAM
GO0_U_NW_CAM
GO0_RESET

Die einzelnen Buchstaben haben dabei folgende Bedeutung:

C (Conditioned) =	referenziert nur, wenn noch nicht referenziert ist
U (Unconditioned) =	referenziert immer, unabhängig davon ob die Achse bereits referenziert ist oder nicht
W (Wait) =	wartet in dieser Anweisungszeile, bis referenziert ist
NW (NoWait) =	nächste Anweisungszeile abarbeiten, während referenziert wird (Empfehlung)
ZP (Zero Pulse) =	referenziert auf Nullimpuls
CAM =	referenziert auf Referenznocken
RESET =	begonnene Referenzfahrt wird unterbrochen und die Anforderung zurückgesetzt. Eine referenzierte Achse wird dereferenziert.

HINWEIS

ZP und CAM sind wirkungslos, wenn Referenzfahrttyp P903 auf 0, 5 oder 8 eingestellt ist. Bei Referenzfahrttyp P903 Typ 3 und 4 darf CAM nicht eingestellt werden.

Beispiel

```
main()
{
   _Go0( GO0_C_W_ZP ); //     referenziere wartend auf Nullimpuls
}
```

17.3.10 _GoAbs

HINWEIS

Wird mit der Modulo-Funktion positioniert, so dürfen die Befehle GOA und GOR nicht verwendet werden. Die Zielposition wird direkt auf H454 geschrieben.

Syntax

```
_GoAbs( typ, pos )
```

Beschreibung

Positioniert absolut auf die angegebene Position.

Die Meldung "IPOS in Position" wird innerhalb eines GOA-Befehls oder GOR-Befehls aktualisiert, d. h. die Meldung kann direkt in der nächsten Programmzeile abgefragt werden.

Argumente

typ Ausdruck für den Typ des Verfahrbefehls. typ kann einen der folgenden Werte annehmen:

GO_NOWAIT: nicht wartend, setzt die Programmabarbeitung sofort nach dem Absetzen des Verfahrbefehls mit der folgenden Anweisungszeile fort (Empfehlung)
GO_WAIT: wartet in dieser Anweisungszeile bis der Verfahrauftrag abgeschlossen ist

pos Enthält die absolute Zielposition, für pos kann folgendes stehen:

konstanter Ausdruck für Zielposition,
Name einer Variablen, die die Zielposition enthält,
Zeiger auf eine Variable, die das Wegstück enthält (indirekte Adressierung).

Parametereinstellungen für alle Positionierbefehle

Parameter	Erklärung
P913 / P914	Verfahrdrehzahlen (änderbar im Programm über SETSYS).
P911 / P912	Positionier-Rampen (Beschleunigung) (änderbar im Programm über SETSYS).
P915 / P203	Vorsteuerungen, mit denen der Ruck beeinflusst werden kann.
P933	Ruckbegrenzung (nur mit MOVIDRIVE® B).
P916	Rampenform.
P917	Rampenmode.

Beispiel

```
// Standardstrukturen fuer Geschwindigkeit und Rampe
SSPOSSPEED tPosSpeed;
SSPOSRAMP tPosRamp;

main()
{
   // Geschwindigkeit und Rampe setzen
   tPosSpeed.CW = tPosSpeed.CCW = 1000 * 10; // Geschwindigkeit 1000 1/min
   tPosRamp.Up = tPosRamp.Down = 1000;       // Rampe bezogen auf 3000 1/min
   _SetSys (SS_POSRAMP, tPosRamp);

   _SetSys (SS_POSSPEED, tPosSpeed);
   // Werden Geschwindigkeit und Rampe nicht im Programm geaendert,
   // so gelten die Werte in der SHELL / siehe Tabelle

   _GoRel (GO_WAIT, 3000);                   // Faehrt auf Position 3000 Ink.
}
```

17.3.11 _GoRel

HINWEIS

Wird mit der Modulo-Funktion positioniert, so dürfen die Befehle GOA und GOR nicht verwendet werden. Die Zielposition wird direkt auf H454 geschrieben.

Syntax

```
_GoRel( typ, pos )
```

Beschreibung

Positioniert relativ um ein Wegstück bezogen auf die aktuelle Position.

Die Meldung "IPOS in Position" wird innerhalb eines GOA-Befehls oder GOR-Befehls aktualisiert, d. h. die Meldung kann direkt in der nächsten Programmzeile abgefragt werden.

Argumente

typ Ausdruck für den Typ des Verfahrbefehls. typ kann einen der folgenden Werte annehmen:

GO_NOWAIT: nicht wartend, setzt die Programmabarbeitung sofort nach dem Absetzen des Verfahrbefehls mit der folgenden Anweisungszeile fort (Empfehlung)
GO_WAIT: wartet in dieser Anweisungszeile bis der Verfahrauftrag abgeschlossen ist

pos Enthält das relative Wegstück, für pos kann folgendes stehen:

konstanter Ausdruck für Wegstück,
Name einer Variablen, die das Wegstück enthält,
Zeiger auf eine Variable, die das Wegstück enthält (indirekte Adressierung),

Beispiel

```
// Standardstrukturen fuer Geschwindigkeit und Rampe
SSPOSSPEED tPosSpeed;
SSPOSRAMP tPosRamp;

main()
{
   // Geschwindigkeit und Rampe setzen
   tPosSpeed.CW = tPosSpeed.CCW = 1000 * 10; // Geschwindigkeit 1000 1/min
   tPosRamp.Up = tPosRamp.Down = 1000;       // Rampe bezogen auf 3000 1/min
   _SetSys (SS_POSRAMP, tPosRamp);

   _SetSys (SS_POSSPEED, tPosSpeed);
   // Werden Geschwindigkeit und Rampe nicht im Programm geaendert,
   // so gelten die Werte in der SHELL / siehe Tabelle

   _GoRel (GO_WAIT, 3000);                   // Faehrt auf Position 3000 Ink.
}
```

17.3.12 _InputCall

Syntax

```
_InputCall ( pegel, maske, funktionsname )
```

Beschreibung

Die Funktion dient dazu, eine selbstdefinierte Funktion in Abhängigkeit der an den Eingangsklemmen anliegenden Pegel aufzurufen. Der Name der Funktion, die gewünschte Polarität der Eingangspegel und die relevanten Klemmen werden als Argumente angegeben. Die Ereignisfunktion wird aufgerufen, wenn alle in maske mit Eins markierten Eingangsklemmen "1"-Pegel (pegel = IC_HIGH) bzw. "0"-Pegel (pegel = IC_LOW) haben.

Argumente

pegel Konstanter Ausdruck, der angibt, auf welchen Signalpegel die Klemmen zu testen sind. Dieser Ausdruck kann einen der folgenden Werte annehmen:

IC_HIGH: HIGH-Pegel ("1"-Pegel)
IC_LOW: LOW-Pegel ("0"-Pegel)

maske Konstanter binärer Ausdruck, der angibt, welche Klemmen zu testen sind.

Die Bits innerhalb des Ausdrucks haben folgende Bedeutung:

Bit 0: DI00, maske = 0b1
Bit 1: DI01, maske = 0b10
Bit 2: DI02, maske = 0b100
Bit 3: DI03, maske = 0b1000
Bit 4: DI04, maske = 0b10000
Bit 5: DI05, maske = 0b100000
Bit 6: DI10, maske = 0b1000000
Bit 7: DI11, maske = 0b10000000
Bit 8: DI12, maske = 0b100000000
Bit 9: DI13, maske = 0b1000000000
Bit 10: DI14, maske = 0b10000000000
Bit 11: DI15, maske = 0b100000000000
Bit 12: DI16, maske = 0b1000000000000
Bit 13: DI17, maske = 0b10000000000000
Bit 14-31: reserviert

Eine Eingangskombination kann gewählt werden, indem die zugehörigen Bits in der Maske auf 1 gesetzt werden. Um beispielsweise DI00 und DI03 abzufragen muss maske lauten: 0b1001

funktionsname Name der Ereignisfunktion (Achtung: Im Gegensatz zu einem Funktionsaufruf wird hier nur der Name der Funktion ohne () angegeben)

Beispiel

```
#include <constb.h>

#define DI02 0b100 // DI02 = 0b100

KlemmeIstEins ()
{
   // Anweisungen der Ereignis-Funktion
}

main()
{
   while(1)
   {
      // Hauptprogrammschleife Task 1
      _InputCall( IC_HIGH,DI02,KlemmeIstEins );
      // wenn Klemme DI02 == HIGH ("1"), Funktion aufrufen
   }
}
```

17.3.13 _Memorize

	HINWEIS Bei Nutzung des Befehls in einer schnellen Task müssen Sie nach dem _Memorize-Befehl einen _Wait-Befehl von mindestens 1 ms setzen.

Syntax

```
_Memorize( aktion )
```

Beschreibung

Ermöglicht das Speichern oder Laden von IPOS$^{plus®}$-Programmen und/oder Variablen im oder vom nichtflüchtigen Speicher (EEPROM) auf dem Gerät. Die Aktion wird über das Argument angegeben.

Argumente

aktion Konstanter Ausdruck für Aktion. aktion kann einen der folgenden Werte annehmen:

MEM_NOP: es findet kein Speichern statt
MEM_STALL: Speichern von Programm und Variablen
MEM_LDALL: Laden von Programm und Variablen
MEM_STPRG: nur Programm speichern
MEM_LDPRG: nur Programm laden
MEM_STDATA: nur Variablen speichern
MEM_LDDATA: nur Variablen laden

	HINWEIS Beim Benutzen des _Memorize()-Befehls ist bei MOVIDRIVE® A und MC07B darauf zu achten, dass die fest speicherbaren Variablen H0 ... H127 sowie alle Parameter nicht zyklisch beschrieben werden, da die Anzahl der Speichervorgänge beim verwendeten Speichermedium EEPROM auf 10^5 Speichervorgänge begrenzt ist. Bei MOVIDRIVE® B besteht diese Einschränkung nicht.

Beispiel

```
main()
{
   _Memorize( MEM_STDATA ); // Variablen H0 ... H127 auf EEPROM speichern
}
```

17.3.14 _MoviLink

Syntax

```
_MoviLink ( H )
```

Beschreibung

Der Befehl MOVLNK ermöglicht die weitgehende Veränderung von Parametern des Umrichters und auch weiterer eventuell über den Systembus oder RS-485 angeschlossene Geräte. Im Hinblick auf die Sicherheit von Personen und Anlagen ist bei Parameterveränderungen des Umrichters besondere Sorgfalt nötig, auf jeden Fall müssen übergeordnete Schutzmaßnahmen greifen um ggf. einer Fehlprogrammierung zu begegnen.

MOVLNK liest und schreibt einmalig bei Befehlsaufruf Prozessdaten, Variablen oder Parameter von einem Gerät zum anderen oder liest oder schreibt einmalig bei Befehlsaufruf Variablen oder Parameter innerhalb eines Geräts.

Das Lesen / Schreiben von Parametern erfolgt über Index-Adressierung. Die jeweilige Index-Nummer kann dem Systemhandbuch und dem Parameter-Verzeichnis entnommen werden. Sie können sich auch die Index-Nummern in MOVITOOLS® MotionStudio anzeigen lassen, indem Sie die Maus auf dem Eingabe-/Anzeigefeld des Parameters platzieren (Tooltip).

Die Kommunikation zwischen 2 Geräten kann über SBus oder RS-485-Schnittstellen erfolgen.

Mit MOVILINK kann innerhalb eines Geräts z. B. die Variable eines Stückzahlzählers netzausfallsicher gespeichert werden, ohne dass mit dem MEM-Befehl der gesamte netzausfallgesicherte Bereich gespeichert wird. Ein Prozessdatenaustausch mit dem eigenen Gerät ist mit dem MOVLINK-Befehl nicht möglich.

Über den Indexzugriff mittels MOVILINK können auch von IPOS$^{plus®}$ aus eigene Werte des Umrichters geschrieben / gelesen werden, die mit GETSYS / SETSYS nicht erreichbar sind. Damit kann der Umrichter sich z. B. im Initialisierungsteil selbst parametrieren.

Bevor der Befehl aufgerufen wird, sind die Variablen zu initialisieren, mit denen der Befehl arbeitet (Befehlsstruktur). Der Anfang der Befehlsstruktur wird dem Befehl als Argument übergeben. In der Datenstruktur liegen die Daten, die geschrieben oder gelesen werden.

Im Sender (Master) und im Empfänger (Slave) sind die Parameter für die Kommunikation einzustellen. Der MOVILINK-Befehl wird nur beim Sender (Master) aufgerufen.

Argumente

H Anfangsvariable der Befehlsstruktur

Die Befehlsstruktur hat folgenden Aufbau:

H + 0: Bus-Typ (Kommunikations-Schnittstelle)

H + 1: Adress (Zieladresse)

H + 2: Format (Prozess- und/oder Parameterdaten)

H + 3: Service (Dienst Lesen/Schreiben)

H + 4: Index (Nummer des Parameters, der geändert oder gelesen werden soll (siehe Parameter-Verzeichnis)

H + 5: DPointer (Nummer der Variablen H", ab der die gelesenen Daten abgelegt bzw. die zu schreibenden Daten geholt werden, also die erste Varialennummer der Datenstruktur.)

– H" + 0: Beinhaltet die Daten für die Parameter-Write-Dienste
– H" + 1: Beinhaltet die Daten, die bei einem Parameter-Dienst gelesen werden
– H" + 2: PA1-Daten des Prozessdatenaustauschs
– H" + 3: PA2-Daten des Prozessdatenaustauschs
– H" + 4: PA3-Daten des Prozessdatenaustauschs
– H" + 5: PE1-Daten des Prozessdatenaustauschs
– H" + 6: PE2-Daten des Prozessdatenaustauschs
– H" + 7: PE3-Daten des Prozessdatenaustauschs

H + 6: Result (Beinhaltet den Fehlercode nach Ausführung des Dienstes bzw. Null, wenn kein Fehler vorhanden ist.)

Für die Anweisung _MoviLink stehen SEW-Standardstrukturen zur Verfügung:

Anweisung	Standard-struktur	Elemente	Kurzbeschreibung
_MoviLink	MOVLNK	BusType (H+0)	mögliche Bustypen: ML_BT_S0 = 1 (RS485 #1) ML_BT_S1 = 2 (RS485 #2) ML_BT_SBUS1 = ML_BT_SBUS = 5 ML_BT_SBUS2 = 8
		Address (H+1)	0...99: Einzeladresse 100...199: Gruppenadresse 253: Eigene Adresse des Umrichters 254: Punkt-zu-Punkt-Verbindung 255: Broadcast Wird eine SBus-Gruppenadresse (z. B. 43) angesprochen, so muss der Offset 100 hinzuaddiert werden. In diesem Fall ergibt das 143.

		Format (H+2)	Angabe der Prozess (PD)- und Parameterkanäle (PARAM) zur Datenübertragung:
			//MoviLink Cyclic Frame Types / //Acyclic
			ML_FT_PAR1 = 0: PARAM+1PD / ML_CFT_PAR1 = 128
			ML_FT_1 = 1: 1PD / ML_CFT_1 = 129
			ML_FT_PAR2 = 2: PARAM+2PD / ML_CFT_PAR2 = 130
			ML_FT_2 = 3: 2PD / ML_CFT_2 = 131
			ML_FT_PAR3 = 4: PARAM+3PD / ML_CFT_PAR3 = 132
			ML_FT_3 = 5: 3PD / ML_CFT_3 = 133
			ML_FT_PAR = 6: Parameter (ohne PD) / ML_CFT_PAR = 134

		Service (H+3)	Kommunikationsdienst für Parameter ML_S_RD = 1: Lese-Dienst ML_S_WR = 2: Schreiben auf nichtflüchtigen Speicher ML_S_WRV = 3: Schreiben ohne Speichern
		Index (H+4)	Index-Nummer des Parameters, der geändert oder gelesen werden soll (siehe Parameter-Verzeichnis) Der Subindex muss im Index-Element auf den Bits 23-16 eingetragen werden (niederwertigstes Byte des High-Words). Berechnung: H+4 oder `MOVILNK.Index = Index + (SubIndex << 16);`
		DPointer (H+5)	Nummer der Variablen, ab der die gelesenen Daten abgelegt bzw. die zu schreibenden Daten geholt werden (Struktur MLDATA)
		Result (H+6)	Beinhaltet den Fehlercode nach Ausführung des Dienstes bzw. Null, wenn kein Fehler vorhanden (siehe "Rückkehr-Codes der Parametrierung" im Handbuch "Kommunikation und Feldbus-Geräteprofil" mit Parameter-Verzeichnis").

Anweisung	Standard-struktur	Elemente	Kurzbeschreibung
	MLDATA	WritePar (H"+0)	Parameter, der bei Write-Diensten verschickt wird
		ReadPar (H"+1)	Parameter, der bei Read-Diensten empfangen wird
		PO1 (H"+2)	Prozessausgangsdatum 1
		PO2 (H"+3)	Prozessausgangsdatum 2
		PO3 (H"+4)	Prozessausgangsdatum 3
		PI1 (H"+5)	Prozesseingangsdatum 1
		PI2 (H"+6)	Prozesseingangsdatum 2
		PI3 (H"+7)	Prozesseingangsdatum 3

Die folgende Tabelle zeigt Ihnen diejenigen Elemente, die gerätespezifischen Besonderheiten aufweisen.

Element	Gerätespezifische Besonderheiten		
	MOVIDRIVE® B	MOVITRAC® B	MQx
BusType (H+0)	nur ML_BT_S1 (RS485 an X13) ML_BT_SBUS1 (SBUS an X12) ML_BT_SBUS2 (über DFC11B)	nur ML_BT_S1 (RS485 an FSC/FIO11B) ML_BT_SBUS1 (SBUS an FSC/FIO21B)	nur ML_BT_S1 (RS485 zu MOVIMOT®)
Format (H+2)	keine Einschränkung	keine Einschränkung	nur ML_CFT_2 (2PD azyklisch) ML_CFT_PAR2 (Param + 2PD azyklisch) ML_CFT_3 (3PD azyklisch) ML_CFT_PAR3 (Param + 3PD azyklisch) ML_CFT_PAR (Param azykl isch) zyklische Frametyps sind möglich, es wird aber hierfür _MovCommDef empfohlen.

HINWEIS

Das Element DPointer der MOVLNK-Struktur muss mit der Anfangsvariablennummer der Datenstruktur initialisiert werden in der die Daten für die Kommunikation abgelegt sind.

Liegen diese Daten beispielsweise in einer Struktur mit dem Namen Busdaten (Deklarationszeile MLDATA Busdaten;), so muss die Initialisierungszeile innerhalb der MOVLNK-Struktur für das Element DPointer lauten:

```
***.DPointer = numof(Busdaten);
```

MOVLNK ist ein wartender Befehl. Der nächste Befehl wird erst abgearbeitet, wenn der MOVLNK-Befehl komplett ausgeführt wurde.

Sollen zwei oder mehr MOVLNK-Befehle zyklisch aufgerufen werden, so müssen diese in **einer** Task abgearbeitet werden. Vorzugsweise geschieht dies beim MOVIDRIVE® B in Task 2 oder Task 3.

Beispiel 1

```
#include <constb.h>

MOVLNK ml;
MLDATA mld;

main()
{
   while(1)
   {
      // Struktur ml initialisieren
      ml.BusType  = ML_BT_S1; // RS-485 #2
      ml.Address  = 1;
      ml.Format   = ML_FT_PAR2; // 2 PD mit Parameter
      ml.Service  = ML_S_RD; // Lesen
      ml.Index    = 8300+(0<<16); // Index des Umrichter-Status (8300.0)
      ml.DPointer = numof(mld); // Ziel-Struktur

      _MoviLink( ml ); // eigentlicher Befehlsaufruf
   }
}
```

Weitere Beispiele

Weitere Beispiele zum `_MoviLink`-Befehl finden Sie in den folgenden Abschnitten:

- Lesen eines geräteeigenen Parameters (Seite 253)
- Beschreiben einer Variablen über SBus (Seite 254)
- Lesen eines Paramters über SBus (Seite 255)

Parametereinstellungen beim Sender (Master)

Adressierung über RS-485: Keine Einstellungen erforderlich.

Adressierung über SBus:

Parameter	Adresse	Erklärung
P816		Die Baudrate ist abhängig von Busleitungslänge und muss bei Sender und Empfänger gleich sein

HINWEIS

Bei RS485-Netzwerken darf nur **ein** Master vorhanden sein.

In SBus- (CAN-) Netzwerken können auch mehrere Master aktiv sein, siehe dazu auch `_SBusCommDef`-Befehl. (Seite 230)

Parametereinstellungen beim Empfänger

Datenaustausch über Parameterkanal

Adressierung über RS-485 (P810 ... P812)

Parameter	Adresse	Erklärung
P810	0 ... 99	Einzeladressierung (Sender Adresse)
P811	101 ... 199	Gruppenadressierung (Multicast), alle Empfänger mit gleicher Gruppenadresse können vom Sender gleichzeitig beschrieben werden
P812		Timeout-Zeitüberwachung (deaktiviert mit Einstellung 0 ms oder 650 ms)

Adressierung über SBus P88_ und P89_

Parameter	Adresse	Erklärung
P881 / P891	0 ... 63	Einzeladressierung
P882 / P892	0 ... 63[1)]	Gruppenadressierung (Multicast), alle Empfänger mit gleicher Gruppenadresse können vom Sender gleichzeitig beschrieben werden
P883 / P893		Timeout-Zeitüberwachung (deaktiviert mit Einstellung 0 ms oder 650 ms)
P884 / P894		Die Baudrate ist abhängig von Busleitungslänge und muss bei Sender und Empfänger gleich sein

1) Bei Verwendung der Gruppenadresse ist der Eingabewert für Zieladressen um 100 zu erhöhen.

Parameter *P880 / P890 SBus Protokoll* muss auf "MoviLink" gestellt sein.

Bei dem physikalisch ersten und letzten Teilnehmer müssen SBus-Abschlusswiderstände eingeschaltet oder angeschlossen sein.

Datenaustausch über Prozessdatenkanal

Für den Prozessdatenaustausch ist die Einstellung der seriellen Kommunikation entsprechend obigen Tabellen (Adressierung über RS-485 / SBus) vorzunehmen. Zur aktiven Nutzung der Prozessdaten sind diese weiteren Einstellungen erforderlich:

Parameter	Erklärung
P100	Sollwertquelle einstellen auf "RS-485" oder "SBus" (Nur wenn Sollwertvorgabe über Prozessdaten-Kommunikation gewünscht ist)
P101	Steuerquelle einstellen auf "RS-485" oder "SBus"
P870...876	Prozessdatenbeschreibung (siehe detaillierte Beschreibung im Handbuch "Kommunikation und Feldbus-Geräteprofil").

Parameter *P880 / P890 SBus Protokoll* muss auf "MoviLink" gestellt sein.

HINWEIS

Beim Benutzen des MOVLNK-Befehls ist darauf zu achten, dass die fest speicherbaren Variablen (H0...127) sowie alle Parameter nicht zyklisch mit dem Kommunikationsdienst = 2 (nicht flüchtig schreiben) beschrieben werden, da die Anzahl der Speichervorgänge beim verwendeten Speichermedium begrenzt ist.

	HINWEIS Bei MOVIDRIVE® B und MOVITRAC® B wird für die Timeout-Überwachung überprüft, ob in der parametrierten Timeout-Zeit Telegramme empfangen werden. Bei MOVIMOT® wird die Timeout-Überwachung mit den ersten empfangenen zyklischen Frames (ML_FT... beim Sender) aktiviert. Azyklische Kommunikation schaltet beim MOVIMOT® die Timeout-Überwachung wieder aus. Nach Start der zyklischen Kommunikation mit _MovCommOn ist dann nur noch der _MoviLink-Befehl zur Adresse 253 (intern) möglich, mit dem _MoviLink-Befehl kann nicht mehr auf andere Geräte zugegriffen werden.

17.3.15 _MovCommDef

Der Befehl _MovCommDef kann nur mit MQx-Modulen verwendet werden.

Syntax

```
_MovCommDef(H)
```

Beschreibung

Die MovComm-Befehle ermöglichen den zyklischen Datenaustausch zwischen MQx und typischerweise bis zu 4 MOVIMOT® über die RS-485-Schnittstelle mit MOVILINK-Profil. Mit _MovCommDef wird eine Kommunikationsverbindung zum MOVIMOT® eingerichtet, indem Parameter wie z. B. die Geräteadresse eingestellt werden. Mit _MovCommOn wird die zyklische Kommunikation gestartet. Danach läuft die zyklische Kommunikation im Hintergrund, unabhängig von der aktuellen Befehlsabarbeitung des IPOS$^{plus®}$-Programms. Das Abbild der ausgetauschten Prozessdaten liegt auf IPOS$^{plus®}$-Variablen und kann dort gelesen und geschrieben werden. Mit dem Anhalten des IPOS$^{plus®}$-Programms wird auch die zyklische Kommunikation gestoppt.

Es sind bis zu 8 Kommunikationsbeziehungen zulässig. Beachten Sie bitte, dass die Anzahl der Kommunikationsbeziehungen sehr starken Einfluss auf die Buszykluszeit der RS-485 und damit auf die Reaktionszeit des MOVIMOT® hat. Pro Kommunikationsbeziehung oder Teilnehmer sind ca. 20 ms Buszykluszeit einzurechnen. Voraussetzung für das Erreichen von 20 ms Buszykluszeit pro Teilnehmer ist eine einwandfreie Verkabelung der RS-485. Tritt während der zyklischen Kommunikation ein Time-out auf, wird das über den Fehler 91 Gateway Sysfault angezeigt. Mit der Rückmeldung des MOVIMOT® wird die Fehlermeldung automatisch aufgehoben.

Alle für die Befehlsausführung notwendigen Informationen sind durch das Anwenderprogramm in eine Datenstruktur im Variablenbereich einzutragen. Der Anfang dieser Variablenstruktur wird dem Befehl als Argument übergeben. Die Variable wird durch **MOVCOM Variablenname;** definiert und hat folgende Struktur:

BusType (H+0)	**Bustyp (Schnittstelle)** ML_BT_S1 = 2 (RS485 zum MOVIMOT®)
Address (H+1)	**Einzeladresse oder Gruppenadresse des anzusprechenden MOVIMOT®** 0 ... 99 Einzeladressierung 100 ... 199 Gruppenadressierung 255 Broadcast
Format (H+2)	**Angabe der Prozessdaten zur Datenübertragung** 3 = 2 Prozessdatenworte zyklisch (für MOVIMOT®) = ML_FT_2 5 = 3 Prozessdatenworte zyklisch (für MOVIMOT®) = ML_FT_3
Pd Pointer (H+3)	**Nummer der Variablen H", auf der die Prozessdaten abgelegt bzw. die zu schreibenden Daten geholt werden**. (Die Datenstruktur für H" ist im Anschluss ausführlich beschrieben)
Para Pointer (H+4)	**Nummer der Variablen H*, auf der die Parameterdaten abgelegt bzw. die zu schreibenden Daten geholt werden**. Wird vom MOVIMOT® nicht unterstützt.

Eine Variablenstruktur, die die Prozessdaten beinhaltet wird im Compiler mit **MCPDATA Variablenname;** definiert:

Datenstruktur für H":	
H"+0	Beinhaltet den Fehlercode der Verbindung bzw. Null, wenn kein Fehler vorhanden ist. 0x05000002 zeigt einen Time-out der Verbindung an.
H"+1	PA1-Daten des Prozessdatenaustauschs
H"+2	PE1-Daten des Prozessdatenaustauschs
H"+3	PA2-Daten des Prozessdatenaustauschs
H"+4	PE2-Daten des Prozessdatenaustauschs
H"+5	PA3-Daten des Prozessdatenaustauschs
H"+6	PE3-Daten des Prozessdatenaustauschs

Die Prozessdaten sind nach MOVILINK kodiert.

Eine Variablenstruktur, die die Parameterdaten beinhaltet wird im Compiler mit **MCPARADATA Variablenname;** definiert:

H*+0	Beinhaltet den Fehlercode nach Ausführung des Parameterdienstes bzw. Null, wenn kein Fehler vorhanden ist. Die Fehler sind nach MOVILINK kodiert.
H*+1	0: keine Aktion oder Parameter-Datenaustausch abgeschlossen. 1: Start des Parameter-Datenaustausch
H*+2	ML_S_RD = 1: Lese-Dienst ML_S_WR = 2: Schreiben mit Speichern auf nichtflüchtigen Speicher ML_S_WRV = 3: Schreiben ohne Speichern
H*+3	Index-Nummer des Parameters, der geändert oder gelesen werden soll
H*+4	Gelesene Daten nach Read-Dienst. Zu schreibende Daten bei einem Write-Dienst.

Bei der Parametrierung ist folgendes Vorgehen einzuhalten:

1. Eintragen von Service, Index und Daten
2. Starten der Parametrierung mit einer 1 auf StartPar
3. Abwarten der Ausführung, Ende wird durch eine 0 auf StartPar angezeigt
4. Auswertung des ParaResult. Falls ein Fehler vorliegt, ist der Datenwert ungültig. Falls kein Fehler vorliegt, wurde der Dienst erfolgreich ausgeführt.

Argument **H** Erste Variable der Variablenstruktur

Beispiel Siehe Funktion _MovCommOn (Seite 229)

17.3.16 _MovCommOn

Der Befehl _MovCommOn kann nur mit MQx-Modulen verwendet werden.

Syntax

```
_MovCommOn()
```

Beschreibung

Der Befehl startet die zyklische Kommunikation, durch MovCommDef eingerichtete Kommunikationsbeziehungen werden aktiviert. Ab jetzt ist kein MovCommDef-Befehl mehr zulässig, ebenso kann kein MOVILINK-Befehl auf Adresse ≠ 253 (intern) verwendet werden.

Argumente

Keine

Beispiel

```
/*=============================================
IPOS Source File
===============================================*/
#include <const.h>
#include <io.h>

#pragma initials 0 127
#pragma globals 128 300
#pragma var 301 400

MOVCOM mc1;          // control values for communication link to MOVIMOT
MCPDATA mcpd1;       // process data exchange with MOVIMOT
MCPARADATA mcpara;   // parameter data exchange with MOVIMOT (not used)

/*=============================================
Main Function (IPOS Entry Function)
===============================================*/
main()
{
   // Initialization ============================
   // fill control structure for communication link to MOVIMOT
   mc1.BusType     = ML_BT_S1; // communication via RS-485 to MOVIMOT
   mc1.Address     = 1;  // MOVIMOT address 1
   mc1.Format      = ML_FT_3;  // PDU type: 3 process data words cyclic
   mc1.PdPointer   = numof(mcpd1); // pointer to process data block
   mc1.ParaPointer = numof(mcpara1); // pointer to parameter data block
   _MovCommDef( mc1 );
   _MovCommOn(  );

   while( )
   {

   }
}
```

17.3.17 _Nop

Syntax

```
_Nop()
```

Beschreibung

Es wird keine Operation ausgeführt. Mit diesem Befehl können z.B. Wartezeiten auf Basis der Befehlszykluszeit erreicht werden.

Argument

Dieser Befehl hat kein Argument.

Beispiel

```
main()
{
   _Nop(  );
}
```

17.3.18 _SBusCommDef

Syntax

```
_SBusCommDef( objekttyp, H)
```

Beschreibung

Mit dieser Anweisung wird ein Datenobjekt zur zyklischen oder azyklischen Datenübertragung eingerichtet. Damit können bis zu 2 Variablen (8 Byte) über den Systembus übertragen werden. Die Beschreibung des Datenobjekts erfolgt in einer Variablenstruktur, deren Anfangsvariable in h angegeben wird. Die zyklische Datenübertragung muss mit der Funktion SBusCommOn () oder SBusCommState (aktion) gestartet werden (siehe folgende Seiten). Der Typ des Datenobjekts wird in objekttyp angegeben.

Argumente

objekttyp Ausdruck, der einen der folgenden Werte annehmen kann:

SCD_REC: Empfangen
SCD_TRCYCL: zyklisches Senden
SCD_TRACYCL: azyklisches Senden

H Erste Variable der Variablen-Struktur

Für die Objekttypen sind erntsprechende Datenstrukturen definiert.

SCD_REC

Initialisiert ein Datenobjekt, das an sich gesendete Daten empfängt. Es können maximal 32 Empfangsobjekte eingerichtet werden.

Das Objekt hat folgenden Aufbau:

H+0: Objektnummer

H+1: Anzahl der Daten-Bytes und Datenformat

H+2: Nummer der Variablen H", ab der die empfangenen Daten abgelegt werden.

SCD_TRCYCL

Initialisiert ein Datenobjekt, dessen Daten zyklisch gesendet werden. .

Das Objekt hat folgenden Aufbau:

H+0: Objektnummer

H+1: Zykluszeit in ms

H+2: Offsetzeit in ms

H+3: Anzahl der Daten-Bytes und Datenformat

H+4: Nummer der Variablen H", wo die zu sendenden Daten beginnen

H+5: Ergebnis des SCOM-Befehls

SCD_TRACYCL Initialisiert ein Datenobjekt, dessen Daten sofort einmalig gesendet werden.

Das Objekt hat folgenden Aufbau:

H+0: Objektnummer

H+1: Anzahl der Daten-Bytes und Datenformat

H+2: Nummer der Variablen H", ab der die zu sendenden Daten beginnen

H+3: Zustand des Sendebefehls

HINWEIS

Vor dem Senden von azyklischen Telegrammen muss der SBus ebenfalls mit _SBusCommOn oder _SBusCommState aktiviert werden.

MOVIDRIVE® B: Das IPOS$^{plus®}$-Programm wartet an diesem Befehl so lange, bis das Telegramm gesendet wurde, aber maximal 10 ms. Nur durch eine Auswertung des Zustands (H+3) oder der erwarteten Antwort kann überwacht werden, ob das Telegramm richtig gesendet wurde.

Für die Anweisung _SBusCommDef stehen SEW-Standardstrukturen zur Verfügung:

Anweisung	Standardstruktur	Elemente	Kurzbeschreibung
_SBusCommDef	SCREC	ObjectNo (H+0)	Objektnummer CAN Identifier (z. B. 1024)
		Format (H+1)	Datenformat
		DPointer (H+2)	Nummer der Variablen H", ab der die empfangenen Daten abgelegt werden. Es müssen 2 Variablen (H" und H"+1)für den Empfang der bis zu 8 Byte Daten reserviert werden.
	SCTRCYCL	ObjectNo (H+0)	Objektnummer (z. B. ohne CANopen-Profil ab 1024
		CycleTime (H+1)	Zykluszeit [ms]
		Offset (H+2)	Offsetzeit [ms]
		Format (H+3)	Anzahl der Daten-Bytes und Datenformat
		DPointer (H+4)	Nummer der Variablen H", ab der die zu sendenden Daten beginnen
		Result (H+5)	Zustand des Sendebefehls: ≥0 Freie Buskapazität in % (berechneter Wert dieses Geräts) -1 Falsche Zykluszeit -2 Zu viele Objekte eingerichtet -3 Busüberlastung -5 Falsche Objektnummer -6 Falsche Länge
	SCTRACYCL	ObjectNo (H+0)	Objektnummer (z. B. ohne CANopen-Profil ab 1024
		Format (H+1)	Anzahl der Daten-Bytes und Datenformat
		DPointer (H+2)	Nummer der Variablen H", ab der die zu sendenden Daten beginnen
		Result (H+3)	Zustand des Sendebefehls: 0 = Bereit 1 = Beim Senden 2 = Senden erfolgreich 10 = Sendefehler

Wenn über SCREC definierte Objekte über SBus1 oder SBus2 empfangen werden, wird dieses in den entsprechenden Systemvariablen H522 und H523 signalisiert, siehe Übersicht über die Systemvariablen (Seite 29).

Bit 24 der ObjectNo definiert, ob die Daten mit dem CAN-Identifier (Bit 0 ... 10) über SBus1 oder SBus2 gesendet oder empfangen werden sollen:

Bit 24 = 0: SBus1 (.ObjectNo = CAN_ID + SCD_SBUS1)

Bit 24 = 1: SBus2 (.ObjectNo = CAN_ID + SCD_SBUS2); nur MDX B mit DFC11B

Gerätespezifischen Besonderheiten:

Element	Gerätespezifische Besonderheiten	
	MOVIDRIVE® B	**MOVITRAC® B**
CycleTime (H+1)	Gültige Zykluszeiten[1]: - 1, 2 ... 9 ms, Anzahl der Objekte 15 - 10, 20, ... 65530 ms	Gültige Zykluszeiten: 1 ... 255 ms, Granularität 1 ms, Anzahl der Objekte 16
Offset (H+2)	Gültige Offsetzeiten: - 0, 1, 2 ... 8 ms für Zykluszeiten < 10 ms - 0, 10, 20, ... 65530 ms für Zyklus-zeiten ≥ 10 ms	Gültige Offsetzeiten: 0 ... 255 ms, Granularität 1 ms Dabei muss gelten: Offsetzeit + Zykluszeit ≤ 255

1) Die Zykluszeit muss immer größer sein als die größte Offsetzeit.

Anzahl der Daten-Bytes und Datenformate:

Bit	Wert	Funktion
0 ... 3	0 ... 8	Anzahl der Daten-Bytes
4 ... 7	--	reserved
8	0	MOTOROLA-Format
	1	INTEL-Format
9 ... 31	--	reserved

Gegenüberstellung von MOTOROLA- und INTEL-Format:

	MOTOROLA-Format		**INTEL-Format**	
CAN-Byte	3 2 1 0	3 2 1 0	0 1 2 3	0 1 2 3
Variable	H"+1	H"	H"	H"+1
Var.Byte	3 2 1 0	3 2 1 0	0 1 2 3	0 1 2 3

	HINWEIS Bei der Wahl der Objektnummer CAN Identifier sind die folgenden Regeln einzuhalten: 1. Im gesamten SBus-Netzwerk darf eine Objektnummer nur 1 mal zum Senden eingerichtet sein. 2. Innerhalb eines Geräts darf eine Objektnummer nur 1 mal eingerichtet werden, entweder 1 mal zum Senden oder 1 mal zum Empfangen. Insbesondere bei weiterem Datenaustausch zwischen den Slaves ist sicher zu stellen, dass die gesamte rechnerische Busauslastung 70 % nicht überschreitet. Berechnet wird die Busauslastung in Bit pro Sekunde über die Formel: Anzahl Telegramme × Bit/Telegramm × 1/Zykluszeit z. B. 2 Telegramme mit 100 Bit im 1-ms-Zyklus = 200000 Bit/s = 200 kBaud Bezogen auf die gewählte Baudrate ergibt sich die prozentuale Buslast. z. B. 200 kBaud / 500 kBaud = 40 % < 70 %

Zu den Punkten 1. und 2. ist zu beachten, dass die Firmware des Geräts einige Objektnummern automatisch reserviert:

- Die Objektnummer, die in Parameter P885 / P895 für die SBus-Synchronisation eingetragen ist.
- Für die Kommunikation über das MOVILINK-Profil werden in Abhängigkeit der SBus-Adresse in Parameter P881 / P891 und der SBus-Gruppenadresse in Parameter P882 / P892 die folgenden Objektnummern verwendet:
  - 8 × SBus-Adresse + 3 für Prozess-Ausgangsdaten
  - 8 × SBus-Adresse + 4 für Prozess-Eingangsdaten
  - 8 × SBus-Adresse + 5 für synchrone Prozess-Ausgangsdaten
  - 8 × SBus-Adresse + 3 + 512 für Parameter-Request-Service
  - 8 × SBus-Adresse + 4 + 512 für Parameter-Response-Service
  - 8 × SBus-Gruppenadresse + 6 für Gruppen-Prozessdaten
  - 8 × SBus-Gruppenadresse + 6 + 512 für Gruppen-Parameter-Request

Für die Kommunikation über das CANopen-Profil werden die im DS301 von CANopen definierten Objektnummern (Identifier) verwendet.

Beispiel Siehe _SBusCommStat

17.3.19 _SBusCommOn

Syntax

```
_SBusCommOn()
```

Beschreibung Der Befehl wurde bei MOVIDRIVE® B durch _SBusCommState ersetzt, ist aber aus Gründen der Abwärtskompatibilität weiterhin verwendbar.

Diese Anweisung stößt den Empfang von Daten und das zyklische Senden von zuvor festgelegten Datenobjekten an. Die Initialisierung der Datenobjekte erfolgt durch die Funktion SBusCommDef () mit dem Argument SCD_TRCYCL und SCD_REC.

	HINWEIS Dieser Befehl aktiviert nur SBus 1, nicht SBus 2

Argument Dieser Befehl hat kein Argument.

Beispiel Siehe Kapitel "Compiler - Beispiele".

17.3.20 _SBusCommState

Syntax

```
_SBusCommState ( aktion )
```

Beschreibung Diese Anweisung initialisiert die CAN-Schnittstelle, startet oder stoppt den Empfang von Daten und das zyklische Senden von zuvor festgelegten Datenobjekten über SBus 1 oder SBus 2. Die Initialisierung der Datenobjekte erfolgt durch Funktion SBusCommDef.

Beim MOVITRAC® B wirkt der Befehl _SBusCommState (aktion) unabhängig vom Wert für "aktion" immer wie _SBusCommOn (Seite 234).

Argumente aktion kann einen der folgenden Werte annehmen:

Wert	Argument	Bedeutung
0	SCS_STARTALL	Synchroner Start der zyklischen Kommunikation von SBus 1 und SBus 2.
1	SCS_STOPALL	Synchroner Stopp der zyklischen Kommunikation von SBus 1 und SBus 2.
2	SCS_START1	Start der zyklischen Kommunikation von SBus 1.
3	SCS_STOP1	Stopp der zyklischen Kommunikation von SBus 1.
4	SCS_START2	Start der zyklischen Kommunikation von SBus 2.
5	SCS_STOP2	Stopp der zyklischen Kommunikation von SBus 2.

Beispiel

```
#include <constb.h>

#define  DATEN     H20
#declare DATUM DATEN:0

#define  INTEL     0x100
#define  ANZ_BYTES 4

SCTRCYCL Obj1;

main()
{
   Obj1.ObjectNo  = 1090;
   Obj1.CycleTime = 10;
   Obj1.Offset    = 0;
   Obj1.Format    = INTEL ¦ ANZ_BYTES // High und Low Byte setzen
   Obj1.DPointer  = numof(DATEN);
   DATUM = 0;

   _SBusCommDef( SCD_TRCYCL,Obj1 ); // Obj1 zyklisch senden
   _SBusCommState( SCS_START1 ); // Starten der Übertragung für SBus1

   while(1)
   {
      // Hauptprogramm Task 1
   }
}
```

17.3.21 _SetInterrupt

Syntax

```
_SetInterrupt( ereignis , funktionsname )
```

Beschreibung

Die Funktion dient dazu, eine selbstdefinierte Funktion als Interrupt-Routine festzulegen. Der Name der Funktion wird als Argument angegeben. Ein Interrupt kann durch verschiedene Ereignisse ausgelöst werden. Das gewünschte Ereignis wird als Argument angegeben.

Argumente

ereignis konstanter Ausdruck, der einen der folgenden Werte annehmen kann:

SI_DISABLE:	Interrupt ist gesperrt
SI_ERROR:	löst einen Interrupt bei Systemfehler aus
SI_TIMER0:	löst einen Interrupt bei Überlauf von Timer0 aus
SI_TOUCHP1:	löst einen Interrupt bei Flankenwechsel an einer Touchprobe-Klemme aus, wenn Touch-Probe aktiviert wurde

funktionsname Name der Interrupt-Funktion. (Achtung: Im Gegensatz zu einem Funktionsaufruf wird hier nur der Name der Funktion ohne () angegeben)

Beispiel

```
#include <constb.h>

T0Interrupt ()
{
   // Anweisungen der Interrupt-Routine für Timer 0
}

main()
{
   // T0Interrupt dem System bekanntmachen und starten
   _SetInterrupt( SI_TIMER0,T0Interrupt );
   while(1)
   {
      // Hauptprogramm Task 1
   }
}
```

17.3.22 _SetSys

Syntax

```
_SetSys( sys , H)
```

Beschreibung

Setzt den Wert einer internen Systemgröße mit dem Wert einer IPOS$^{plus®}$-Variablen

Argumente

H Name der Quellvariable

sys Konstanter Ausdruck, der die Systemgröße bezeichnet. sys kann einen der folgenden Werte annehmen:

SS_N11:	interner Festsollwert n11
SS_N12:	interner Festsollwert n12
SS_N13:	interner Festsollwert n13
SS_N21:	interner Festsollwert n21
SS_N22:	interner Festsollwert n22
SS_N23:	interner Festsollwert n23

Achtung:

Der neue Festsollwert wird erst nach 5 ms sicher übernommen. Programmabarbeitung nach _SetSys-Befehl evtl. mit _Wait-Befehl 5 ms verzögern.

Wird für den Festsollwert ein Wert eingegeben, der den zulässigen Bereich überschreitet, wechselt das Vorzeichen.

SS_PIDATA: Aktualisieren der PE-Daten[1)]

- H = Anzahl PE-Daten
- H + 1 = PE-Datum 1
- H + 2 = PE-Datum 2
- H + 3 = PE-Datum 3

SS_OPMODE: Setzen der Betriebsart

- H = 11: CFC (Drehzahlregelung)
- H = 12: CFC & Momentenregelung
- H = 13: CFC & IPOS (Positionierung)
- H = 14: CFC & Synchronlauf (DRS11)
- H = 16: SERVO (Drehzahlregelung)
- H = 17: SERVO & Momentenregelung
- H = 18: SERVO & IPOS (Positionierung)
- H = 19: SERVO & Synchronlauf (DRS11)

SS_IMAX: Einstellung der Drehmomentgrenze durch Setzen des Parameters P 304 Drehmomentgrenze (nur CFC oder SERVO); Einheit: 0.1 %

SS_POSRAMP: Positionierrampen; Einheit: 1 ms

- H = Positionierrampe 1
- H+ 1 = Positionierrampe 2

SS_POSSPEED: Positionierdrehzahl; Einheit: 0.1 U/min

- H = Drehzahl rechts
- H + 1 = Drehzahl links

SS_OVERRIDE: Override ein-/ausschalten

- H = 0 -> aus
- H = 1 -> ein

SS_BRAKE: Bremsenfunktion ein-/ausschalten
– H = 0 -> aus
– H = 1 -> ein

SS_RAMPTYPE: Positionierrampenform angeben (ändert P916)
– H = 0 -> linear
– H = 1 -> sinus
– H = 2 -> quadratisch
– H = 3 -> Busrampe
– H = 4 -> ruckbegrenzt
– H = 5 -> Kurvenscheibe
– H = 6 -> I-Synchronlauf

SS_RESET: Rücksetzen des Systemfehlers mit der Fehlernummer in Variable H
– H = Variable mit der Fehlernummer

SS_ACTPOS: Setzen der Ist-Position
– H = Positionswert

SS_SPLINE:

antriebsinterne Berechnung einer analytischen Kurvenscheibe. Die Funktion ist z. Zt. nur im MCH mit SK-Version -0C oder MDxB SK -0C/5C verfügbar.

Nach der Vorgabe von bis zu 20 Stützstellen (x-y-Wertepaaren, x = Masterposition, y = Slave-Position) in einem Leitgeberbereich wird über die Systemfunktion die Spline-Berechnung initialisiert. Danach wird über `H+0 SplineMode` die Berechnung gestartet und eine komplette oder ein Segment einer ausgewählten Kurvenscheibe gefüllt. Bisher sind Spline-0-Verfahren (für optimale Laufruhe) und Spline-1-Verfahren (für Rast-in-Rast-Bewegungen und Geradenstücke) verfügbar. Die Berechnung ist nach ≤ 200 ms abgeschlossen.

– H+0 = SplineMode: (Wertebereich: 0 ... 3)
  - = 0: Interpolation nicht aktiv, bzw. Berechnung beendet
  - = 1: Interpolation starten, interpolierte Werte von Index 0 beginnend in die Kurvenscheibe eintragen (vorwärts, d. h. von Index 0 bis 512)
  - = 2: Interpolation starten, interpolierte Werte von Index 512 beginnend in die Kurvenscheibe eintragen (rückwärts, d. h. von Index 512 bis 0)
  - = 3: vorbereitende Parameterberechnung für die Interpolation abgeschlossen, Beginn des Eintragens der interpolierten Werte in die Kurvenscheibe
– H+1 = SplineModeControl: reserviert
– H+2 = SplineDest: (Wertebereich: 0 ... 5)
  Die Nummer der Kurvenscheibe in die die interpolierten Werte eingetragen werden sollen.
– H+3 = SplineNUser: (Wertebereich: 2 ... 20)
  Die Anzahl der Stützstellen, die für die Interpolation verwendet werden sollen und das Berechnungsverfahren (Bit 0 ... Bit 4 = Anzahl der Stützstellen, Bit 7 = 0: Spline 0, Bit 7 = 1: Spline 1)
– H+4 = SplineX0User: (Hier darf nur ein Wert > = 0 eingetragen werden!)
  Stützstellen-Nr. der X-Achse (Master) eintragen.
– H+5 = SplineY0User: (Wertebereich: long = -2^{31} ... 0 ... (2^{31} -1))
  Y-Wert (= Lagewert) der 1. Stützstelle; wenn ACTPOSSCALE ≠ 0, dann muss in der Struktur der skalierte Wert eingetragen werden
– ...
– H+42 = SplineX19User: (Hier darf nur ein Wert <= 512 eingetragen werden!)
  Stützstellen-Nr. der X-Achse (Master) eintragen.
– H+43 = SplineY19User: (Wertebereich: long = -2^{31} ... 0 ... (2^{31} -1))
  Y-Wert der 20. Stützstelle; wenn ACTPOSSCALE ≠ 0, dann muss in der Struktur der skalierte Wert eingetragen werden

SS_MULTIAXIS: antriebsübergreifende Berechnung einer Bewegungsbahn

Nur auf Anfrage erhältlich. Siehe auch Zusatz zur Betriebsanleitung „Sonderkonstruktion SK-0C für MCH oder MDxB: Berechnete Kurven mit MCH“.

1) Gilt unter der Bedingung, dass für den Parameter P101 "RS485", "Feldbus" oder "SBus" eingestellt ist.

Für die Anweisung _SetSys stehen SEW-Standardstrukturen zur Verfügung:

Anweisung	Standardstruktur	Elemente	Kurzbeschreibung
_SetSys	SSPOSRAMP	Up	Positionierrampe Beschleunigung (ms)
		Down	Positionierrampe Verzögerung (ms)
	SSPOSSPEED	CW	Positionierdrehzahl rechts (0.1 U/min)
		CCW	Positionierdrehzahl links (0.1 U/min)
	SSPIDATA3[1)]	Len	Anzahl der zu sendenden Prozess-Eingangsdaten
		PI1	Prozesseingangsdatum 1
		PI2	Prozesseingangsdatum 2
		PI3	Prozesseingangsdatum 3
	SSPIDATA10[1)]	Len	Anzahl der zu sendenden Prozess-Eingangsdaten
		PI1	Prozesseingangsdatum 1
		PI2	Prozesseingangsdatum 2
		PI3	Prozesseingangsdatum 3
		PI4	Prozesseingangsdatum 4
		PI5	Prozesseingangsdatum 5
		PI6	Prozesseingangsdatum 6
		PI7	Prozesseingangsdatum 7
		PI8	Prozesseingangsdatum 8
		PI9	Prozesseingangsdatum 9
		PI10	Prozesseingangsdatum 10

1) Gilt unter der Bedingung, dass für den Parameter P101 "RS485", "Feldbus" oder "SBus" eingestellt ist.

Beispiel

```
main()
{
   // Betriebsart Drehzahlregelung einstellen
   H0 = 11;
   _SetSys( SS_OPMODE,H0 );
}
```

17.3.23 _SetTask

Syntax

```
_SetTask ( steuerwort, funktionsname )
```

Beschreibung

Die Funktion dient dazu, eine selbstdefinierte Funktion als Task festzulegen und diese zu starten bzw. zu stoppen. Der Name der Funktion und das Steuerwort werden als Argumente angegeben.

Argument

steuerwort Konstanter Ausdruck, der einen der folgenden Werte annehmen kann:

MOVIDRIVE® B	**MOVIDRIVE® A**
ST2_STOP: Task 2 stoppen	T2_START: Task 2 starten
ST2_START: Task 2 starten	T2_STOP: Task 2 stoppen
ST3_STOP: Task 3 stoppen	
ST3_START: Task 3 starten	

funktionsname Name der Task-Funktion.

Beispiel

```
#include <constb.h>

MeineTask3 ()
{
   // Anweisungen der Task 3
}

main()
{
   // Task 3 dem System bekanntmachen und starten
   _SetTask ( ST3_START,MeineTask3 );
   while(1)
   {
      // Hauptprogramm
   }
}
```

17.3.24 _SetTask2

Syntax

```
_SetTask2( steuerwort, funktionsname )
```

Beschreibung

Die Funktion dient dazu, eine selbstdefinierte Funktion als Task 2 festzulegen und diese zu starten bzw. zu stoppen. Der Name der Funktion und das Steuerwort werden als Argumente angegeben. Nach Netz-Ein stehen Steuerwort und Startadresse auf 0, d. h. Task 2 ist deaktiviert.

Der Befehl wurde bei MOVIDRIVE® B durch _SetTask ersetzt, ist aus Abwärtskompatibilität aber weiterhin auch im MOVIDRIVE® B verfügbar.

Argumente

steuerwort Konstanter Ausdruck, der einen der folgenden Werte annehmen kann:

T2_STOP: Task 2 stoppen
T2_START: Task 2 starten

funktionsname Name der Task 2-Funktion (Achtung: Im Gegensatz zu einem Funktionsaufruf wird hier nur der Name der Funktion ohne () angegeben)

Beispiel

```
#include <constb.h>

MeineTask2 ()
{
   // Anweisungen der Task 2
}

main()
{
   // Task 2 dem System bekanntmachen und starten
   _SetTask2( T2_START,MeineTask2 );
   while(1)
   {
      // Hauptprogramm
   }
}
```

17.3.25 _SetVarInterrupt

Syntax

```
_SetVarInterrupt ( h1 , funktionsname )
```

Beschreibung

Der Befehl ist im MOVIDRIVE® A nicht verfügbar, nur ab MOVIDRIVE® B.

Der Befehl aktiviert einen Variablen-Interrupt mit der Datenstruktur ab der Variable H1. Ist die Bedingung für den Interrupt erfüllt, so wird die Funktion funktionsname ausgeführt. Das Ereignis für den Interrupt ist der Vergleich mit einem Variablenwert (siehe H+4). Wenn die Datenstruktur initialisiert ist, kann zur Laufzeit mit einem IPOS^plus®-Befehl das Interrupt-Verhalten eines kompletten VarInterrupts dynamisch angepasst werden.

Hinweis: Die Daten aus der Datenstruktur werden nur übernommen, wenn der Befehl `_SetVarInterrupt ( H1 , funktionsname )` aufgerufen wird (Datenkonsistenz). Ausnahme ist die Variable `pSourceVar`.

Beispiel: wird z. B der Wert aus der Datenstruktur Hx+3 CompareVar geändert, so wird der Wert erst mit dem Befehl `_SetVarInterrupt ( H1 , funktionsname )` berücksichtigt.

Argumente

H1 Erste Variable einer Datenstruktur (siehe Tabelle H+0)

funktionsname Name der Interrupt-Funktion. Im Gegensatz zu einem Funktionsaufruf wird hier nur der Name der Funktion ohne () angegeben.

Datenstruktur des Variablen-Interrupts:

Variable	Elemente Struktur VARINT	Beschreibung
H+0	Control	0: Alle VarInterrupt = AUS / Reset 1: Task 2 unterbrechen 2: Task 3 unterbrechen
H+1	IntNum	0 ... 3: Legt eine fortlaufende Nummer des VarInterrupt fest. Ein bereits aktivierter Interrupt mit der Nummer x kann während der Programmlaufzeit über den Befehlsaufruf `_SetVarInterrupt ( h1 , funktionsname )` mit einer anderen Datenstruktur komplett neu aktiviert werden, wenn in der neuen Datenstruktur an der Stelle H+1 dieselbe Interrupt-Nummer angegeben ist. Diese Eigenschaft ist bei den Task 1-Interrupts nicht möglich.
H+2	SrcVar	Nummer der Bezugsvariablen, deren Wert mit dem Vergleichswert verglichen wird.
H+3	CompVar	Vergleichswert oder Maske, mit der der Wert der Bezugsvariable H+2 verglichen wird.

Variable	Elemente Struktur VARINT	Beschreibung
H+4	Mode	0: Kein Interrupt-Event. Damit kann dieser einzelne Interrupt deaktiviert werden, ohne alle Interrupts abzuschalten. 1: Eines der Bits der Bezugsvariable, die mit der Maske CompVar ausmaskiert werden, hat seinen Zustand geändert: ([*SrcVar(t) ^ *SrcVar(t-T)] & CompVar) != 0 2:Solange Wert der Bezugsvariable gleich Vergleichswert (*SrcVar == CompVar) 3:Solange Wert der Bezugsvariable ungleich Vergleichswert (*SrcVar != CompVar) 4:Solange Wert der Bezugsvariable größer gleich Vergleichswert (*SrcVar > = CompVar) 5:Solange Wert der Bezugsvariable kleiner gleich Vergleichswert (*SrcVar <= CompVar) 6: Wert der Bezugsvariable logisch verundet Vergleichswert ungleich 0 ((*SrcVar & CompVar) != 0) 7: Wert der Bezugsvariable logisch verundet Vergleichswert gleich 0 ((*SrcVar & CompVar) == 0) 8: positive Flanke des über CompVar ausmaskierten Bits 9: negative Flanke des über CompVar ausmaskierten Bits 10: wie 2, jedoch wird Interrupt jedesmal nur ein Mal bearbeitet, wenn die Bedingung erfüllt wird (flankengetriggert) 11: wie 3, jedoch wird Interrupt jedesmal nur ein Mal bearbeitet, wenn die Bedingung erfüllt wird (flankengetriggert) 12: wie 4, jedoch wird Interrupt jedesmal nur ein Mal bearbeitet, wenn die Bedingung erfüllt wird (flankengetriggert) 13: wie 5, jedoch wird Interrupt jedesmal nur ein Mal bearbeitet, wenn die Bedingung erfüllt wird (flankengetriggert)
H+5	Priority	Priorität des Interrupts (1 ... 10), Task 2 und Task 3 haben jeweils Priorität 0.
H+6	IntEvent	Prozessabbild der Bezugsvariable von *SourceVar zum Interrupt-Zeitpunkt.

Beispiel Siehe "Task-Verwaltung und Interrupts / Variablen-Interrupts bei MOVIDRIVE® B".

17.3.26 _SystemCall

Syntax

```
_SystemCall( ereignis, funktionsname )
```

Beschreibung Die Funktion dient dazu, eine selbstdefinierte Funktion bei Eintritt eines systembedingten Ereignisses aufzurufen. Der Name der Funktion und das gewünschte Ereignis werden als Argumente angegeben.

Argumente **ereignis** Konstanter Ausdruck, der angibt, wann funktionsname aufgerufen wird. Dieser Ausdruck kann einen der folgenden Werte annehmen:

SC_UC:	unbedingt
SC_N0:	wenn Drehzahl gleich null
SC_N:	wenn Drehzahl ungleich null
SC_NOTPOS:	wenn nicht in Position
SC_TP1:	wenn Flankenwechsel an Touch-Probe Klemme DI02
SC_NTP1:	wenn kein Flankenwechsel an Touch-Probe Klemme DI02
SC_TP2:	wenn Flankenwechsel an Touch-Probe Klemme DI03
SC_NTP2:	wenn kein Flankenwechsel an Touch-Probe Klemme DI03

funktionsname Name der Ereignisfunktion. (Achtung: Im Gegensatz zu einem Funktionsaufruf wird hier nur der Name der Funktion ohne () angegeben)

Beispiel

```
#include <constb.h>

DrehzahlNull () // Ereignis-Funktion
{
   // Anweisungen der Ereignis-Funktion
}

main()
{
   while(1)
   {
      // Hauptprogramm Task 1
      _SystemCall( SC_N0,DrehzahlNull );
      // wenn Drehzahl == Null, Funktion aufrufen
   }
}
```

17.3.27 _TouchProbe

Syntax

```
_TouchProbe( aktion )
```

Beschreibung

Gibt einen Touch-Probe-Eingang frei bzw. sperrt diesen. Touch-Probe-Eingänge sind die Eingangsklemmen DI02 und DI03.

Der Speichervorgang für die Touch Probe-Positionen erfolgt, unabhängig von der aktuellen Programmabarbeitung, innerhalb von 100 µs. Die Änderung des Klemmenpegels muss für mindestens 200 µs anstehen, um sicher erkannt werden zu können. Mit dem Argument kann der Flankenwechsel, der zum Touch Probe führt, ausgewählt werden.

Tritt an einem freigegebenen Eingang ein Flankenwechsel auf, werden die aktuellen Ist-Positionen in dafür bestimmte IPOS$^{plus®}$-Systemvariablen abgelegt. Für eine erneute Messung muss der Touch Probe erneut freigegeben werden.

Die Touch-Probe-Positionen werden in den nachstehenden Variablen abgelegt:

Geber	Geber-Position	Position Touch-Probe 1 (DI02)	Position Touch-Probe 2 (DI03)
Motorgeber X15	H511 ActPos_Mot	H507 TpPos1_Mot	H505 TpPos2_Mot
Externer Geber X14	H510 ActPos_Ext	H506 TpPos1_Ext	H504 TpPos2_Ext
Absolutwertgeber X62	H509 ActPos_Abs	H503 TpPos1_Abs	H502 TpPos2_Abs
Virtueller Geber (nur bei MOVIDRIVE® B)	H376	H501 TpPos1_VE	H500 TpPos2_VE

Argumente

aktion kann einen der folgenden Werte annehmen:

TP_EN1: Freigabe des Touch-Probe-Eingangs DI02
TP_DIS1: Sperren des Touch-Probe-Eingangs DI02
TP_EN2: Freigabe des Touch-Probe-Eingangs DI03
TP_DIS2: Sperren des Touch-Probe-Eingangs DI03
TP_EN1_HI: Freigabe des Touch-Probe-Eingangs DI02 mit steigender Flanke
TP_EN1_LO: Freigabe des Touch-Probe-Eingangs DI02 mit fallender Flanke
TP_EN2_HI: Freigabe des Touch-Probe-Eingangs DI03 mit steigender Flanke
TP_EN2_LO: Freigabe des Touch-Probe-Eingangs DI03 mit fallender Flanke

Beispiel

```
main()
{
   _TouchProbe( TP_EN1 ); // Freigabe des Touch-Probe-Eingangs DI02
}
```

17.3.28 _Wait

Syntax

```
_Wait( zeit )
```

Beschreibung

Wartet die in einer Konstanten angegebene Zeit in Millisekunden (ms).

Argumente

zeit Konstante, welche die Wartezeit in Millisekunden angibt, keine Variable möglich.

	HINWEIS Soll die Wartezeit variabel sein, so muss statt des WAIT-Befehls ein Timer (H487 ... H489) initialisiert werden und eine Schleife programmiert werden, bis der Timer abgelaufen ist.

Beispiel

```
Timer_0 = 20000;     // start value 20 s
while( Timer_0 ){}   // wait 20 s
```

17.3.29 _WaitInput

Syntax

```
_WaitInput ( pegel, maske )
```

Beschreibung

Die Funktion wartet solange, bis an bestimmten Eingangsklemmen ein bestimmter Pegel anliegt. Die gewünschte Polarität der Eingangspegel und die relevanten Klemmen werden als Argumente angegeben. Die Funktion wartet solange, wie alle in maske mit Eins markierten Eingangsklemmen "1"-Pegel bzw. "0"-Pegel haben.

Argumente

pegel Konstanter Ausdruck, der angibt, auf welchen Signalpegel die Klemmen zu testen sind, und einen der folgenden Werte annehmen kann:

1:	HIGH-Pegel ("1"-Pegel)
0:	LOW-Pegel ("0"-Pegel)

maske Konstanter binärer Ausdruck, der angibt, welche Klemmen zu testen sind. Die Bits innerhalb des Ausdrucks haben folgende Bedeutung:

Bit 0:	DI00,	maske = 0b1
Bit 1:	DI01,	maske = 0b10
Bit 2:	DI02,	maske = 0b100
Bit 3:	DI03,	maske = 0b1000
Bit 4:	DI04,	maske = 0b10000
Bit 5:	DI05,	maske = 0b100000
Bit 6:	DI10,	maske = 0b1000000
Bit 7:	DI11,	maske = 0b10000000
Bit 8:	DI12,	maske = 0b100000000
Bit 9:	DI13,	maske = 0b1000000000
Bit 10:	DI14,	maske = 0b10000000000
Bit 11:	DI15,	maske = 0b100000000000
Bit 12:	DI16,	maske = 0b1000000000000
Bit 13:	DI17,	maske = 0b10000000000000
Bit 14-31:	reserviert	

Eine Eingangskombination kann gewählt werden, indem die zugehörigen Bits in der Maske auf 1 gesetzt werden. Um beispielsweise DI00 und DI03 abzufragen muss maske lauten: 0b1001

Beispiel

```
#include <constb.h>

main()
{
   _WaitInput( 1,0b100 );
   // solange Klemme DI02 == HIGH ("1"), warten
}
```

17.3.30 _WaitSystem

Syntax

```
_WaitSystem( ereignis )
```

Beschreibung

Die Funktion wartet solange, wie ein systembedingtes Ereignis vorliegt. Das gewünschte Ereignis wird als Argument angegeben.

Argumente

ereignis Konstanter Ausdruck, der angibt, wann die Funktion wartet. Dieser Ausdruck kann einen der folgenden Werte annehmen:

SC_UC:	unbedingt
SC_N0:	wenn Drehzahl gleich null
SC_N:	wenn Drehzahl ungleich null
SC_NOTPOS:	wenn nicht in Position
SC_TP1:	wenn Flankenwechsel an Touch-Probe Klemme DI02
SC_NTP1:	wenn kein Flankenwechsel an Touch-Probe Klemme DI02
SC_TP2:	wenn Flankenwechsel an Touch-Probe Klemme DI03
SC_NTP2:	wenn kein Flankenwechsel an Touch-Probe Klemme DI03

Beispiel

```
main()
{
   _WaitSystem( SC_N0 ); // solange Drehzahl == Null, warten
}
```

17.3.31 _WdOff

Syntax

```
_WdOff()
```

Beschreibung

Der Watchdog wird abgeschaltet.

Argument

Dieser Befehl hat kein Argument.

Beispiel

```
main()
{
   WdOFF();
}
```

17.3.32 _WdOn

Syntax

```
_WdOn( zeit )
```

Beschreibung

Setzt den Wert des Watchdog-Zählers auf den in zeit angegebenen Wert. Läuft der Watchdog-Timer ab, so werden Task 1 und Task 2 angehalten und es erfolgt eine Fehlermeldung. Das Ablaufen des Watchdog-Timers muss die Anwendung durch zyklisches Neusetzen des Zählers verhindern. Dabei muss der Zählerwert mindestens so groß sein, wie ein Hauptprogrammdurchlauf dauert.

Argumente

zeit Watchdog-Zählerwert in Millisekunden (ms).

Beispiel

```
#define WD_ZEIT 1000

main()
{
   while(1)
   {
      /*
      Anweisungen in Schleife ausführen
      Die Summe der Ausführungszeiten der Anweisungen innerhalb
      der Schleife darf nicht größer als 1000 ms sein, um ein
      Ablaufen des Watchdog zu verhindern.
      */

      _WdOn( WD_ZEIT ); // alle 1000 ms WD neu triggern
   }
}
```

18 Compiler – Beispiele

18.1 Setzen von Bits und Ausgangsklemmen

Für das Setzen von Einzelbits in Variablen gibt es zwei Wege:

1. Die Funktion _BitSet(Hx, y) setzt das Bit y in der Variable x auf Eins.
2. Die bitweise ODER-Verknüpfung Hx ¦ K setzt in der Variablen x diejenigen Bits auf Eins, die auch in der Konstanten K auf Eins stehen.

In beiden Fällen lässt sich die Lesbarkeit des Programms verbessern, wenn die Bitposition bzw. die Konstante symbolisch definiert ist. Da Bitsetzfunktionen hauptsächlich benutzt werden, um binäre Geräteausgänge zu setzen, wird im folgenden Beispiel die Variable H481 (StdOutpIPOS) als Zielvariable der Operation benutzt. Um die Ausgänge der Option anzusprechen, würde man entsprechend die Variable H480 (OptOutpIPOS) verwenden. Im Beispiel soll die Ausgangsklemme des Grundgeräts DO02 gesetzt werden.

Verwendung von _BitSet()	Verwendung der ODER-Verknüpfung

`#include <const.h>`
`#include <io.h>  // MOVIDRIVE A`
`#include <iob.h>  // MOVIDRIVE B`
`main()`
`{`
`  _BitSet( StdOutpIPOS, 2 );`
`}`
	`#include <const.h>`
	`#include <io.h>  // MOVIDRIVE A`
	`#include <iob.h>  // MOVIDRIVE B`
	`main()`
	`{`
	`  StdOutpIPOS ¦= DO02;`
	`}`

Der Quelltext kann noch weiter vereinfacht werden, wenn die jeweiligen Anweisungen neben den Variablen und Konstanten zusätzlich symbolisch bezeichnet werden:

Verwendung von _BitSet():

```
#include <const.h>
#include <io.h>      // MOVIDRIVE A
#include <iob.h>     // MOVIDRIVE B
#define SetzeDO02    _BitSet( StdOutpIPOS, 2);
main()
{
   SetzeDO02
}
```

Sollen mehrere Ausgänge gleichzeitig gesetzt werden, so kann man entweder die Funktion _BitSet() mehrfach hintereinander aufrufen oder dafür die bitweise ODER-Verknüpfung benutzen. Im zweiten Fall kommt man mit einer Anweisung aus. Dies vermindert den Codeumfang und wirkt sich so auch positiv auf die Programmlaufzeit aus.

Das folgende Beispiel benutzt die ODER-Operation, um DO01 und DO02 gleichzeitig zu setzen.

Verwendung der ODER-Verknüpfung:

```
#include <const.h>
#include <io.h>      // MOVIDRIVE A
#include <iob.h>     // MOVIDRIVE B
main()
{
   StdOutpIPOS ¦= DO01 ¦ DO02;
}
```

18.2 Löschen von Bits und Ausgangsklemmen

Für das Löschen von Einzelbits in Variablen gibt es zwei Wege:

1. Die Funktion _BitClear(Hx, y) löscht das Bit y in der Variable x.
2. Die bitweise UND-Verknüpfung Hx & K setzt in der Variablen x diejenigen Bits auf Null, die auch in der Konstanten K auf Null stehen.

In beiden Fällen lässt sich die Lesbarkeit des Programms verbessern, wenn die Bitposition bzw. die Konstante symbolisch definiert ist. Da Bitlöschfunktionen hauptsächlich benutzt werden, um binäre Geräteausgänge zurückzusetzen, wird im folgenden Beispiel die Variable H481 (StdOutpIPOS) als Zielvariable der Operation benutzt. Um die Ausgänge der Option anzusprechen, würde man entsprechend die Variable H480 (OptOutpIPOS) verwenden. Im Beispiel soll die Ausgangsklemme des Grundgeräts DO02 auf Null gesetzt werden.

Verwendung von _BitClear()	Verwendung der UND-Verknüpfung

`#include <const.h>`
`#include <io.h>  // MOVIDRIVE A`
`#include <iob.h>  // MOVIDRIVE B`
`main()`
`{`
`  _BitClear(StdOutpIPOS , 2 );`
`}`
	`#include <const.h>`
	`#include <io.h>  // MOVIDRIVE A`
	`#include <iob.h>  // MOVIDRIVE B`
	`main()`
	`{`
	`  StdOutpIPOS &= ~DO02;`
	`  /*Der Operator "~" bewirkt die bit-weise`
	`  Negation von DO02. Somit sind alle Bits`
	`  von DO02 Eins, bis auf Bit 2  */`
	`}`

Sollen mehrere Ausgänge gleichzeitig rückgesetzt werden, so kann man entweder die Funktion _BitClear() mehrfach hintereinander aufrufen oder dafür die bitweise UND-Verknüpfung benutzen. Im zweiten Fall kommt man mit einer Anweisung aus. Dies vermindert den Codeumfang und wirkt sich so auch positiv auf die Programmlaufzeit aus.

Das folgende Beispiel benutzt die UND-Operation, um DO01 und DO02 gleichzeitig zu löschen.

Verwendung der UND-Verknüpfung:

```
#include <const.h>
#include <io.h>      // MOVIDRIVE A
#include <iob.h>     // MOVIDRIVE B
main()
{
   StdOutpIPOS &= ~DO01 & ~DO02;
}
```

18.3 Abfrage von Bits und Eingangsklemmen

Um abzufragen, welchen Pegel eine bestimmte Eingangsklemme hat, ist es notwendig, das betreffende Bit mithilfe einer Variablen zu testen. Die Variable ist entweder H483 (InputLevel), die die Pegel der binären Eingänge enthält, oder eine beliebige Variable h, die die Pegel nach Ausführung der Funktion _GetSys() enthält.

18.3.1 Testen von Einzelbits

Um ein Bit einer Variable zu testen, führt man eine UND-Verknüpfung mit einer Konstanten durch, in der das zu testende Bit auf Eins steht:

Ist das Ergebnis gleich Null, dann ist auch das zu testende Bit Null und der Eingangsklemmenpegel somit low. Ist das Ergebnis ungleich Null, steht das Bit auf Eins.

Das folgende Beispiel setzt H10 auf 1, wenn am Binäreingang DI03 eine Eins anliegt.

Einzelbit testen ; H483 verwenden	Einzelbit testen ; _GetSys() verwenden

`#include <const.h>`
`#include <io.h>  // MOVIDRIVE A`
`#include <iob.h>  // MOVIDRIVE B`
`main()`
`{`
`  if(( InputLevel & DI03 ) != 0)`
`  {`
`  H10 = 1;`
`  }`
`}`
	`#include <const.h>`
	`#define EINGAENGE H1`
	`#include <io.h>  // MOVIDRIVE A`
	`#include <iob.h>  // MOVIDRIVE B`
	`main()`
	`{`
	`  _GetSys( EINGAENGE, GS_INPUTS );`
	`  if(( EINGAENGE & DI03 ) != 0)`
	`  {`
	`  H10 = 1;`
	`  }`
	`}`

18.3.2 Testen mehrerer Bits

Um mehrere Bits einer Variable auf einen bestimmten Zustand zu testen, maskiert man die zu testenden Bits mittels UND-Verknüpfung aus und vergleicht das Ergebnis mit einer Konstanten, die dem zu testenden Bitmuster entspricht.

Das folgende Beispiel setzt H10 auf 1, wenn an DI01 eine Eins und an DI03 eine Null anliegt.

```
#include <const.h>
#define DI03   0b1000
#define DI01   0b0010
main()
{
   if(( InputLevel & (DI03 ¦ DI01)) == 0b0010)
   {
      H10 = 1;
   }
}
```

18.4 Flankenabfrage

18.4.1 Beispiel 1

Zusätzlich zum Pegel einer Eingangsklemme kann die steigende und fallende Flanke abgefragt und ausgewertet werden. In den folgenden Beispielprogrammen wird der Ausgang DO02 bei positiver oder negativer Flanke an DI02 getoggelt.

Positive Flankenabfrage

```
#include <const.h>  // MOVIDRIVE A
#include <io.h>   // MOVIDRIVE A

// Variables for edge generation
long  lDI02RisingEdge,
      lDI02LastState,
      lDO02State,
      lInputLevel;

main()
{
   while(1)
   {
      // Read DI02
      lInputLevel = (InputLevel & 0x00000004);

      // Generate edge DI02
      lDI02RisingEdge = lInputLevel && (lDI02LastState);
      lDI02LastState = lInputLevel;

      if(lDI02RisingEdge)
      lDO02State = (!lDO02State)

      // Set output DO02
      if (lDO02State)
         _BitSet( StdOutpIPOS, 2 );

      else
         _BitClear( StdOutpIPOS, 2 );
   }
}
```

Negative Flankenabfrage

```
#include <const.h>
#include <io.h>

// Variables for edge generation
long  lDI02FallingEdge,
      lDI02LastState,
      lDO02State,
      lInputLevel;

main()
{
   while(1)
   {
      // Read DI02
      lInputLevel = (InputLevel & 0x00000004);

      // Generate edge DI02
      lDI02FallingEdge = !lInputLevel && (lDI02LastState);
      lDI02LastState = lInputLevel;

      if(lDI02FallingEdge)
      lDO02State = (!lDO02State)

      // Set output DO02
      if (lDO02State)
         _BitSet( StdOutpIPOS, 2 );

      else
         _BitClear( StdOutpIPOS, 2 );
   }
}
```

HINWEIS

Bei der Realisierung einer Flankenabfrage ist zu beachten, dass zur Bildung der Flanke immer eine Hilfsvariable `lInputLevel`, die den Zustand der Eingangsklemme zuvor speichert, benutzt wird und nicht die Eingangsklemme selbst.

Flankenwechsel an DI02

```
// Read DI02
lInputLevel = (InputLevel & 0x00000004);

// Generate edge DI02
lDI02FallingEdge = !lInputLevel && (!DI02LastState);
lDI02LastState = lInputLevel;
```

Würde anstatt der Hilfsvariablen lInputLevel die Eingangsklemme benutzt werden, könnte es vorkommen, dass der Flankenwechsel an der Eingangsklemme genau dann erfolgt, wenn das IPOS$^{plus®}$-Programm zwischen den beiden, zur Flankenbildung benötigten Programmzeilen steht. Somit würde der Flankenwechsel an der Eingangsklemme nicht erkannt werden.

Des Weiteren ist zu beachten, dass die Flanke einer Eingangsklemme nur in der Task abgefragt werden darf, in der die Flanke auch gebildet wurde. Durch Asynchronität der einzelnen Tasks kann es sonst vorkommen, dass eine Flanke nicht erkannt oder falsch interpretiert wird.

Wird die Flanke einer Eingangsklemme in mehreren Tasks benötigt, muss die Flanke in jeder Task separat gebildet werden.

18.4.2 Beispiel 2

In Beispiel 2 wird abhängig von der steigenden Flanke an DI02 der Programmteil innerhalb der if-Abfrage bearbeitet.

```
/*=============================================
IPOS-Quelldatei
===============================================*/
#include <constb.h>
#include <iob.h>
long BinInputsNew, BinInputsOld;
/*=============================================
Hauptfunktion (IPOS-Eintrittsfunktion)
===============================================*/
main()
   {
   /*-------------------------------------
   Initialisierung
   --------------------------------------*/
   /*-------------------------------------
   Hauptprogramm-Schleife
   --------------------------------------*/
   while(1)
      {
         // Binäreingänge einlesen
         _GetSys( BinInputsNew,GS_INPUTS );
         // Flankenabfrage
         if( (BinInputsNew & 0x4) && !(BinInputsOld & 0x4) ) // steigende Flanke
DI02
            {
               // hier steht die Programmanweisung
            }

         // Eingangszustände sichern
         BinInputsOld = BinInputsNew;
      }
   }
```

18.5 Betrag einer Zahl

Das folgende Beispielprogramm soll zeigen, wie mit dem IPOS$^{plus®}$-Compiler der Betrag einer Zahl gebildet werden kann.

In Task 1 läuft ein Reversierprogramm, das den Antrieb um 400000 Inkremente nach rechts und 400000 Inkremente nach links relativ verfährt.

In Task 2 wird die Ist-Drehzahl gelesen und der Betrag der Ist-Drehzahl gebildet. Der Betrag steht in Variablen `lActSpeedAbsolute`.

```
#include <constb.h>  // MOVIDRIVE A
#include <iob.h> // MOVIDRIVE B

#define SEKUNDE  1000

SSPOSSPEED tSpeed;
long lActSpeed, lActSpeedAbsolute;

/*===================================
Task 2
===================================*/
Task2()
{
   _GetSys( lActSpeed,GS_ACTSPEED );

   // Calculate absolute speed value
   if( lActSpeed < 0 )
   lActSpeedAbsolute = -lActSpeed;
   else
   lActSpeedAbsolute = lActSpeed;
}

main()
{
   // Initialization =================
   // Positioning speed 500 rpm
   tSpeed.CW = tSpeed.CCW = 5000;
   _SetSys( SS_POSSPEED,tSpeed );

   // Activate Task 2
   _SetTask2( T2_START,Task2 );

   // Main program loop ==============
   while(1)
   {
      _GoRel( GO_WAIT,400000 );
      _Wait( SEKUNDE );
      _GoRel( GO_WAIT,-400000 );
      _Wait( SEKUNDE );
   }
}
```

18.6 MoviLink-Befehl

Der Befehl `_MoviLink` führt nach Aufruf einen Daten- oder Parameteraustausch von oder zu anderen Geräten über SBus oder RS-485 aus. Ebenso ist es möglich geräteeigene Parameter zu lesen und zu verändern.

Um die Funktion des Befehls `_MoviLink` zu verdeutlichen, sind hier drei Beispiele aufgeführt:

- Lesen eines geräteeigenen Parameters: Über `_MoviLink` wird der eingestellte Referenzfahrttyp gelesen.
- Beschreiben einer Variable über SBus: Auf einem über SBus gekoppelten MOVIDRIVE® wird die Variable H200 je nach Zustand des Binäreingangs DI17 beschrieben.
- Lesen eines Parameters über SBus: Die Prozessdaten-Konfiguration des über SBus gekoppelten Umrichters mit der Adresse 10 wird gelesen.

18.6.1 Lesen eines geräteeigenen Parameters

```
/*=========================================
Type of reference travel actually entered
in P903 is read in task 1 and written to
variable lRefType.
=========================================*/

/*=========================================
IPOS Source file
=========================================*/

#include <constb.h>
#include <iob.h>

// Definition of MOVLNK structures
MOVLNK tRefType;
MLDATA tData;

// Definition of Variables
long lRefType;

/*=========================================
Main program
=========================================*/

main ()
{
   // Initialization of MoviLink for bus transfer
   tRefType.BusType   = ML_BT_S1;
   tRefType.Address = 253 // own inverter
   tRefType.Format    = ML_FT_PAR;    // only parameters
   tRefType.Service   = ML_S_RD;      // read
   tRefType.Index     = 8626;         // P903 RefType
   tRefType.DPointer  = numof(tData); // data buffer

   // Main program loop
   while(1)
   {
      // Read type of reference travel
      _MoviLink( tRefType );
      lRefType = tData.ReadPar;
   }
}
```

18.6.2 Beschreiben einer Variablen über SBus

```
/*=========================================
Variable H200 of inverter connected via
SBus is written depending on DI17 in task 1:
DI17 = 0 ->  -1000
DI17 = 1 ->   1000
=========================================*/

/*=========================================
IPOS Source file
=========================================*/

#include <constb.h>
#include <iob.h>

// Definition of MOVLNK structures
MOVLNK tBus;
MLDATA tBusData;

/*=========================================
Main program
=========================================*/

main ()
{
   // Initialization of MoviLink for bus transfer
   tBus.BusType   = ML_BT_SBUS1;      // bus type SBus1
   tBus.Address   = 10;              // SBus address 10
   tBus.Service   = ML_S_WRV;  // write volatile
   tBus.Index     = 11200;           // variable H200
   tBus.DPointer  = numof(tBusData); // data buffer

   // Main program loop
   while(1)
   {
      if( DI17 )
      {
         tBusData.WritePar = 1000;
         _MoviLink( tBus );
      }
      else
      {
         tBusData.WritePar = -1000;
         _MoviLink( tBus );
      }
   }
}
```

18.6.3 Lesen eines Parameters über SBus

```
/*=========================================
Process data configuration of inverter
connected via SBus is written to variable
lPDData in task 1.
The received values correspond to the
following process data configuration:
0 = PARAM + 1PD
1 = 1PD
2 = PARAM + 2PD
3 = 2PD
4 = PARAM + 3PD
5 = 3PD
6 = PARAM + 6PD
7 = 6PD
8 = PARAM + 10PD
9 = 10PD
=========================================*/

/*=========================================
IPOS Source file
=========================================*/

#include <constb.h>
#include <iob.h>

// Definition of MOVLNK structures
MOVLNK tPD;
MLDATA tData;

// Definition of variables
long lPDData;

/*=========================================
Main program
=========================================*/

main ()
{
   // Initialization of MoviLink for bus transfer
   tPD.BusType   = ML_BT_SBUS;      // bus type SBus
   tPD.Address   = 10;              // SBus address 10
   tPD.Format    = ML_FT_PAR;       // only parameters
   tPD.Service   = ML_S_RD;         // read
   tPD.Index     = 8451;            // P090 PD data config
   tPD.DPointer  = numof(tData);  // data buffer

   // Main program loop
   while(1)
   {
      // Read PD configuration
      _MoviLink( tPD );
      lPDData = tData.ReadPar;
   }
}
```

18.7 SCOM-Kommunikation

Das folgende Beispiel zeigt ein Programm, das zwei Variable zyklisch alle 10 ms über SBus sendet. Ein weiteres Programm empfängt die gesendeten Daten.

Mit dem Befehl `_SBusCommDef` wird ein Datenobjekt zur zyklischen Datenübertragung eingerichtet. Die Beschreibung des Sendeobjekts erfolgt in der Variablenstruktur `tBusTr`, die Beschreibung des Empfangsobjekts in `TBusRec`.

Zum Starten der zyklischen Datenübertragung wird die Funktion `_SBusCommOn` beim MOVIDRIVE® A und `_SBusCommState` beim MOVIDRIVE® B aufgerufen.

18.7.1 Empfänger

Der Empfänger holt die Daten vom SBus und legt sie in den Variablen H305 und H306 ab.

```
/*=========================================
Get data object 1025 from SBus and store
the data in variable H305 and H306.
SHELL settings:
P813 SBus Address ->  2
P816 SBus Baudrate ->  500 kBaud
=========================================*/

/*=========================================
IPOS Source file
=========================================*/

#include <constb.h>
#include <iob.h>

// Definition of SCOM structures
SCREC tBusRec;

// Definition of variables
#define Data_Var1x H305
#define Data_Var2x H306

/*=========================================
Main program
=========================================*/

main ()
{
   // Initialization of SCOM transfer object
   tBusRec.ObjectNo   = 1060;              // object number
   tBusRec.Format     = 8;                 // 8 byte
   tBusRec.Dpointer  = numof(Data_Var1x);  // data buffer

   // Start SCOM
   _SBusCommDef( SCD_REC,tBusRec );
   _SBusCommState( SCS_START1 );   // Start cyclic communication MOVIDRIVE B
   // _SBusCommOn(  );   // Start cyclic communication MOVIDRIVE A

   // Main program loop
   while(1)
   {
   }
}
```

18.7.2 Sender

Die Variablen H208 und H209 werden zyklisch alle 10 ms an einen anderen Umrichter gesendet. Die Werte von H208 und H209 können mit dem Eingang DI17 verändert werden.

DI17 = 0: H208 = 111111 / H209 = 222222

DI17 = 1: H208 = 222222 / H209 = 444444

Task 2 ist in diesem Beispielprogramm nicht implementiert.

```
/*=========================================
Variables H208 and H209 are sent cyclic
every 10 ms to another inverter via SBus.
The values of H208 and H209 can be altered
with input DI17.
DI17 = 0: H208 = 111111 / H209 = 222222
DI17 = 1: H208 = 222222 / H209 = 444444
SHELL settings:
P813 SBus Address ->  1
P816 SBus Baudrate ->  500 kBaud
=========================================*/
/*=========================================
IPOS Source file
=========================================*/
#include <constb.h>
#include <iob.h>

// Definition of SCOM structures
SCTRCYCL tBusTr;

// Definition of variables
#define Data_Var1 H208
#define Data_Var2 H209
/*=========================================
Main program
=========================================*/
main ()
{
   // Initialization of SCOM transfer object
   tBusTr.ObjectNo   = 1025;              // object number
   tBusTr.CycleTime  = 10;                // cycle time
   tBusTr.Offset     = 0;                 // offset
   tBusTr.Format     = 8;                 // 8 byte
   tBusTr.DPointer   = numof(Data_Var1);  // data buffer
   tBusTr.Result     = 1111;              // default value for control

   // Initialize variables
   Data_Var1 = 111111;
   Data_Var2 = 222222;

   // Start SCOM
   _SBusCommDef( SCD_TRCYCL,tBusTr );
   _SBusCommState( SCS_START1 );   // Start cyclic communication MOVIDRIVE B
   // _SBusCommOn(  );   // Start cyclic communication MOVIDRIVE A
```

```
   // Main program loop
   while(1)
   {
      if( DI17 )
      {
         Data_Var1 = 222222;
         Data_Var2 = 444444;
      }
      else
         Data_Var1 = 111111;
      {
         Data_Var2 = 222222;
      }
   }
}
```

18.8 Touch-Probe Interrupt-Verarbeitung

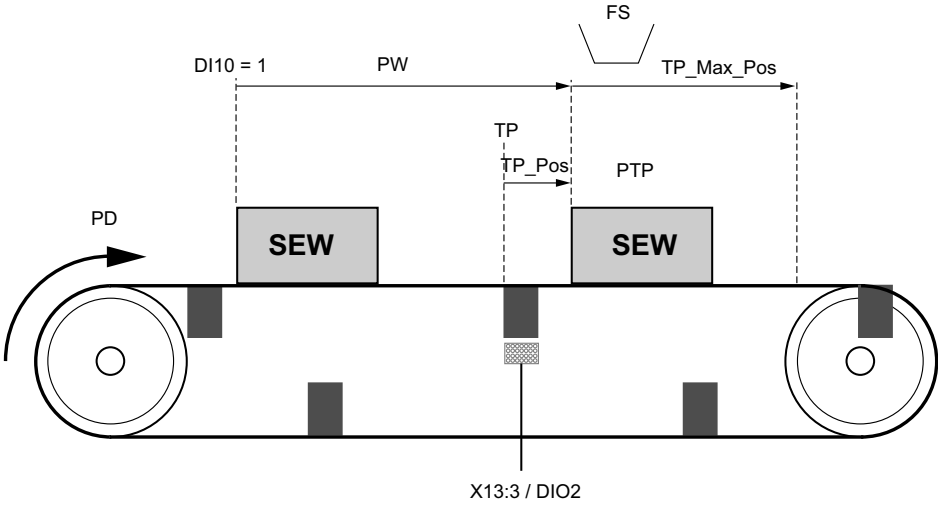
Auf einem Taktband werden Kisten zu einer Füllstation transportiert. Ein Näherungsinitiator (DI02) erkennt die nächste Kiste. Daraufhin wird ein Interrupt ausgelöst und das Band noch einen bestimmten Restweg weitergefahren. Danach steht die Kiste direkt unter der Füllstation. Ist die Kiste befüllt, so wird ein neuer Takt gestartet.

Der Eingang DI10 muss während der gesamten Dauer eines Takts anstehen. Wird DI10 = 0, so bleibt das Taktband lagegeregelt stehen. Bei erneutem Start (DI10 = 1) wird der Takt zu Ende gefahren.

Die Eingabe der Positionswerte sowie der Rampe und Drehzahl erfolgt in den Variablen H11 ... H14.

Variable	Name	Beschreibung
H11	TP_Max_Pos	Maximale Zielposition, wenn der Touch-Probe-Eingang nicht bedämpft wird.
H12	TP_Pos	Restweg, nachdem Touch-Probe-Eingang bedämpft wurde.
H13	Speed	Positionierdrehzahl in min^{-1}.
H14	Ramp	Positionierrampe in ms.

Taktband mit Näherungssensor:



506321163

PD: Taktrichtung	TP: TP-Ereignis
DI10 = 1: Start	TP_Pos: Restweg
PW: Taktweite	PTP: Position nach TP-Ereignis
FS: Füllstation	X13:3 / DI02: Näherungssensor
TP_Max_Pos: Maximale Zielposition	SEW: Kiste auf Transportband

```
/
*==============================================================================
Beschreibung:
Der Start eines Maschinentaktes erfolgt ueber den Eingang DI10. Zielposition ist
die
aktuelle Motorposition (H511) plus TP_Max_Pos (H11). Wird der Touchprobe Eingang
DI02
nicht bedämpft, faehrt der Antrieb auf diese Zielposition. Wird DI02 bedaempft,
wird eine
neue Zielposition berechnet. Die neue Zielposition errechnet sich aus der Motor-
position
zum Zeitpunkt des Touchprobeereignisses TpPos1_Mot (H507) plus dem Restweg
TP_Pos (H12).
Einstellungen in der SHELL :
P601 Binaereingang DI02  IPOS-Eingang
P610 Binaereingang DI10  IPOS-Eingang
P700 Betriebsart         ... & IPOS
==============================================================================*/
#include <const.h>
#include <io.h>
#define CALCTARGET              0
#define BUSSY                   1
#define STOP_AKTIV              2
#define State                   H10
#define TP_Max_Pos              H11
#define TP_Pos                  H12
#define Speed                   H13
#define Ramp                    H14
#define h473_ipos_in_position   (StatusWord & 0x00080000)   //StatusWord & BIT19
long lPosition;
```

```
SSPOSSPEED tPosSpeed;
SSPOSRAMP tPosRamp;
/*=============================================
 Interruptroutine Touchprobe
================================================*/
Touchprobe()
   {
      lPosition = TpPos1_Mot + TP_Pos; //neue Zielposition berechnen
      _TouchProbe( TP_DIS1 ); //Touchprobe deaktivieren
   }
/*=============================================
 Hauptfunktion (IPOS-Eintrittsfunktion)
================================================*/
main()
{
   //Initialisierung
   State = 0;
   // Initialisierung der Interruptroutine fuer den Touch Probe Eingang DI02
   _SetInterrupt( SI_TOUCHP1,Touchprobe );
   //Hauptprogramm-Schleife
   while(1)
   {
      // Geschwindigkeit und Rampe setzen
      tPosSpeed.CW = tPosSpeed.CCW = Speed *10; // Geschwindigkeit
      tPosRamp.Up = tPosRamp.Down = Ramp; // Rampe
      _SetSys(SS_POSRAMP, tPosRamp);
      _SetSys(SS_POSSPEED, tPosSpeed);
      switch(State)
      {
         case CALCTARGET:if(DI10)
                         {
                            lPosition = ActPos_Mot + TP_Max_Pos;
                            _TouchProbe( TP_EN1_HI );
                            // Aktivierung auf steigende Flanke
                            State = BUSSY;
                         }
                         break;
         case BUSSY:     _GoAbs( GO_NOWAIT,lPosition );
                         // _Go-Befehl auf lPosition
                         if(h473_ipos_in_position && !DI10)
                            // Achse in Position und DI10 = 0
                            State = CALCTARGET; // neue Zielposition berechnen
                         if (!h473_ipos_in_position  && !DI10)
                            // Abbruch der Positionierung durch DI10 = 0
                         {
                            _AxisStop(AS_PSTOP);
                            State = STOP_AKTIV;
                         }
                         break;
         case STOP_AKTIV:if(DI10) // DI10 = 1 -->  Positionierung fortsetzen
                         break;
                            State = BUSSY;
         default :       break;
      }//switch(State)
   }// while (1)
}// main
```

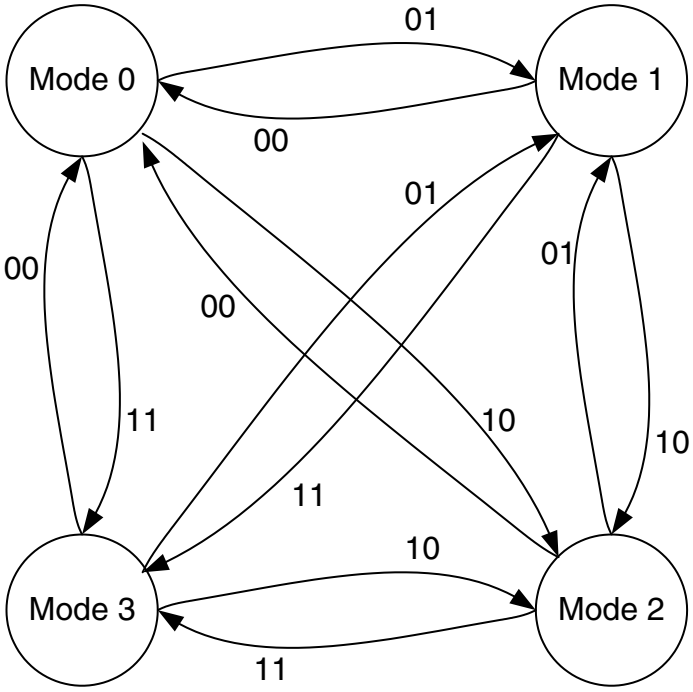
18.9 Zustandsautomat, Feldbussteuerung mit Notbetrieb

Ein Antrieb soll im Normalbetrieb über den Feldbus gesteuert werden. Im Fall einer Busstörung soll aber auch ein manueller Betrieb über Klemme und Analogwert möglich sein. Außerdem ist ein Mischbetrieb (Feldbus-Sollwert + Analog-Sollwert) vorzusehen. Der Betriebs-Mode wird über die Eingangsklemmen DI10 und DI11 eingestellt. Der ausgewählte Betriebs-Mode soll an den Ausgängen DO10 und DO11 angezeigt werden. Folgende Betriebs-Modes sind vorzusehen:

	HINWEIS Verwenden Sie für die PA-Datenbelegung "IPOS" und bilden Sie das Steuerwort über das ControlWord h484 nach, da es sonst ein Problem beim Busausfall und dem Mode 2 gibt.

Für die Übergänge zwischen den Betriebs-Modes ergibt sich der folgende Zustandsgraph:

Mode-Zustandsgraph:



506325259

```
/*==================================================
Operating mode is selected with input terminals
DI10 and DI11 and indicated at the outputs
DO10 and DO11.
The following operating modes are possible:
Mode 0: Control and setpoint via field bus
Mode 1: Control via field bus, setpoint added to
analog value 1
Mode 2: Control via terminals, setpoint analog 1
Mode 3: reserved

SHELL settings:
P100 Setpoint source: BIPOL./FIX.SETPT
P101 Control signal source: TERMINALS
P600 ... P604 Binary input DI01 ... DI05: NO FUNCTION
P610 / P611 Binary input DI10 / DI11: IPOS INPUT
P630 / P631 Binary output DO10 / DO11: IPOS OUTPUT
P700 Operating mode 1: ... & IPOS
P870 ... P872 Setpoint description PO1 ... PO3: IPOS PO-DATA
P873 ... P875 Actual value description PI1 ... PI3: IPOS PI-DATA
P876 PO data enable: ON
==================================================*/
/*==================================================
IPOS Source file
==================================================*/

#include <constb.h>
#include <iob.h>
#pragma globals 350 399

// Definition of structures
GSPODATA3 busdata;                  //structure for fieldbus process data
GSAINPUT analog;                    //structure for analog values

// Definition of variables
#define modeselect              ((InputLevel > >  6) & 0x00000003)
#define setfixedsetpoint        _SetSys( SS_N11,speed )
#define activatefixedsetpoint   _BitSet( ControlWord, 4 )
#define deactivatefixedsetpoint _BitClear( ControlWord, 4 )
#define enable                  _BitClear( ControlWord, 1 )
#define rapidstop               _BitSet( ControlWord, 1 )

// Declaration of variables
long mode, speed, offset;
```

```
/*=====================================================
Main program
=====================================================*/

main ()
{
   // Initialization ==================================
   // Initialize data structure bus data
   busdata.BusType  = 3;          //bus type fieldbus
   busdata.Len      = 3;
   busdata.PO1      = 0;
   busdata.PO2      = 0;
   busdata.PO3      = 0;
   // Activate task 2
   _SetTask2( T2_START,buscontrol );

   // Main program loop ==============================
   while(1)
   {
   }
}
/*=====================================================
Task 2
=====================================================*/

buscontrol()
{
   _GetSys( busdata,GS_PODATA );        //get bus data
   mode = modeselect;                   //read terminals for mode select
   OptOutpIPOS = ((OutputLevel > >  3) & 0xFFFFFFFC) ¦ mode;  //output mode
   switch( mode )
   {
      case  0: mode_0();
         break;
      case  1: mode_1();
         break;
      case  2: mode_2();
         break;
      case  3: mode_3();
         break;
   }
}
```

```
/*=====================================================
Functions
=====================================================*/

mode_0()
{
   if( busdata.PO1 == 6 )
      enable;
   else
      rapidstop;
      speed = busdata.PO2;
      setfixedsetpoint;
      activatefixedsetpoint;
}

mode_1()
{
   if(busdata.PO1 == 6)
      enable;
   else
      rapidstop;
      _GetSys( analog,GS_ANINPUTS );
      offset = (analog.Input1 + 15) / 10;
      speed = busdata.PO2 + offset;
      setfixedsetpoint;
      activatefixedsetpoint;
}

mode_2()
{
   enable;
   deactivatefixedsetpoint;
}

mode_3()
{
   rapidstop;
}
```

18.9.1 Mode 0

Steuerung und Sollwert nur über Feldbus

Die Steuerung erfolgt ausschließlich über den Feldbus. Dabei soll ein reduziertes Steuerwort (0 = Schnellstopp, 6 = Freigabe) verwendet werden. Der Sollwert wird bipolar über den Feldbus vorgegeben (-1500 1/min ... +1500 1/min).

18.9.2 Mode 1

Steuerung über Feldbus, Sollwert = Feldbus-Sollwert + Analog-Sollwert

Die Steuerung erfolgt ausschließlich über den Feldbus. Dabei soll ein reduziertes Steuerwort (0 = Schnellstopp, 6 = Freigabe) verwendet werden. Der Sollwert ergibt sich als Summe aus dem Feldbus-Sollwert (bipolar -1500 1/min ... +1500 1/min) und dem Analog-Sollwert (-10 V ... +10 V = -1500 1/min ... +1500 1/min).

18.9.3 Mode 2

Steuerung und Sollwert über Klemme und Analogwert.

Der Feldbus ist abgeschaltet.

18.9.4 Mode 3

Reserviert

Bis zur sinnvollen Verwendung wird ein Schnellstopp durchgeführt.

18.10 Compiler Programmgerüst

Das folgende Beispielprogramm kann als Grundgerüst bei der Erstellung eines IPOS$^{plus®}$-Programms genutzt werden. Es beinhaltet eine Zustandsmaschine, die vier Betriebsarten kennt:

- DISABLE: keine Betriebsart angewählt
- JOGGING: Tippbetrieb
- HOMING: Referenzfahrt
- POSITIONING: Positionierbetrieb

```
/*=====================================================================
Name:       Basic_program
Version:    03/07/21 (Y/M/D)
Function:   Grundgerüst für ein IPOS-Programm mit Zustandsmaschine
mit Eintritts- und Austrittsfunktion bei Ansteuerung über
Feldbus oder RS485-Monitor mit 3 Prozessdatenworten E/A

notwendige Einstellungen in der SHELL:
-------------------------------------------------------------
P100 = P101 = RS485 für Simulation mit Busmonitor,
= FELDBUS und "#define" bei "//Konstanten" für Betrieb mit Feldbus einkommentieren ???
P6xx        = keine Funktionen, Ausnahme z.B. P602 = REFERENZNOCKEN je nach Referenzfahrtyp
P700        = xxx & IPOS
P870        = STEUERWORT 2
P871        = IPOS PA-DATA
P872        = IPOS PA-DATA
P873        = STATUSWORT 1
P874        = IPOS PE-DATA
P875        = IPOS PE-DATA
P916        = z.B. auf LINEAR
P960        = AUS (bei <> AUS ersetzen: ActPos_Mot =>  ModActPos und TargetPos =>  ModTagPos

Ansteuerung Antrieb durch die SPS/Busmonitor (Prozessausgangsdaten)
-------------------------------------------------------------------------------
/DI00 = REGLERSPERRE

1. Wort = Steuerwort 2
2. Wort = Tipp/Positioniergeschw. [1/10 Umdr/min]
3. Wort = Zielposition
Belegung Steuerwort 2
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  Reglersperre/Freigabe
¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  Freigabe/Schnellstop
¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  Freigabe/Halt
¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  Halteregelung
¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  Integrator-Umschaltung
¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  Parametersatz-Umschaltung
¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  Fehler-Reset
¦  ¦  ¦  ¦  ¦  ¦  ¦  Start Referenzfahrt
¦  ¦  ¦  ¦  ¦  ¦  Tippen +
¦  ¦  ¦  ¦  ¦  Tippen -
¦  ¦  ¦  ¦  Betriebsart Bit 0,  01 = Tippen, 10 = Referenzieren 11 = Automatik
¦  ¦  ¦  Betriebsart Bit 1
```

```
Rückmeldung Antrieb an SPS/Busmonitor (Prozesseingangsdaten)
-------------------------------------------------------------------------------
1. Wort = Statuswort, anwenderspezifisch
2. Wort = Istgeschw. [1/10 Umdr/min]
3. Wort = Istposition in Inkr (nur Low-Wort)
Belegung Statuswort, anwenderspezifisch
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  nicht belegt
¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  Umrichter betriebsbereit
¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  IPOS Referenz (Antrieb referenziert)
¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  Zielposition erreicht
¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  Bremse offen
¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  ¦  Fehler
¦  ¦  ¦  ¦  Quittierung Betriebsart Bit 0,  01 = Tippen, 10 = Referenzieren 11 = Automatik
¦  ¦  ¦  Quittierung Betriebsart Bit 1 (Hinweis: es wird die AKTIVE BA quittiert.)
Kurzanleitung zur Bedienung mit dem Busmonitor:
1.) Busmonitor im MOVITOOLS MotionStudio starten
2.) Button PA-Daten vorgeben drücken
3.) in linkem Feld bei PA1, PA2, PA3 Bits anklicken und Sollwerte vorgeben
4.) den Button mit dem Umrichter "Senden" anklicken
-------------------------------------------------------------------------------
Dies ist eine Beispiel-Software, die Funktionsfähigkeit ist NICHT garantiert,
Benutzung unter Ausschluss jeglicher Garantieleistungen und auf eigene Gefahr.
===============================================================================*/
#include <const.h> //Defaultpfad: c:\programme\sew\movitools\projects\include
#include <io.h> //Einbinden der Namen der Systemvariablen und Konstanten
//Variablenbereiche definieren
#pragma var H128 H149 // (Defaulbereich für Compiler-Hilfsvar. H400 H419)
#pragma globals H380 H449 // (Defaulbereich globale "long"-Variablen H420 H449)
#pragma initials H0 H127 // (Defaulbereich initials H0 H127)
#pragma list // Assembler-Code mit Kommentaren
// Konstanten
#define MY_PD_LENGTH       3          //3 Feldbus mit Prozessdaten
//#define MY_FBUS_TYPE     GS_BT_FBUS //im Betrieb GetSys auf "Feldbus"
#define MY_FBUS_TYPE       GS_BT_S0   //für Busmonitor GetSys auf "RS485"
#define MY_HALT             13        //Position "Halt" im IPOS-Control-Wort H484
// Bit-Masken
#define MY_OP_MODE           0x18 //virtuelle Eingänge DI13/14, InputLevel Bit 9/10
#define MY_READY_TO_RUN      (StatusWord & 0x4)  //Betriebsbereit aus H473
#define MY_NO_ERROR          (StatusWord & 0x2)  //1 = fehlerfrei, 0 = Fehler aus H473
#define MY_IN_POSITION       (StatusWord & 0x80000) //IPOS-Antrieb hat Zielposition erreicht
#define MY_REFERENCED        (StatusWord & 0x100000) //Antrieb referenziert
#define MY_START_HOMING      (lPA_ControlWordHigh & 0x1) //virt. Klemme DI10 Start Ref.-Fahrt
#define MY_START_POSITIONING (lPA_ControlWordHigh & 0x1) //virt. Klemme DI10 Start Positionieren
#define MY_JOG_PLUS          (lPA_ControlWordHigh & 0x2) //virt. Klemme DI11 Tippen +
#define MY_JOG_MINUS         (lPA_ControlWordHigh & 0x4) //virt. Klemme DI12 Tippen -
// Variablen für Soll- / Istwerte, Feldbus-Steuer-/Statuswort
long lPA_ControlWordHigh; // Bit 8-15 des Feldbus-Steuerworts 2
// =DI10 - DI17 der virtuellen Klemmen = Bit 6-13 in InputLevel (!!!)
SSPOSSPEED tPosVelocities; //Datenstruktur Positioniergeschwindigkeiten
long lPE_StatusWord;      // Anwenderstatus-Wort, Bit 8-15 des Feldbus-Statusworts
// =DO10 - DO17 der virtuellen Klemmen = Bit 0-7 in OptOutpIPOS (!!!)
long lActPosition,        // Istposition in Inkr
lScalingNumerator,   // Zaehler fuer Skalierung der Position
lScalingDenominator, // Nenner fuer Skalierung der Position
lActVelocity;        // Istgeschwindigkeit in 1/10 Upm
```

```
// Variablen Betriebsartensteuerung
long lOpMode;              //aktuell vorgewählte Betriebsart
long lGlobalStateMachine;  //Zustand der globalen Zustandsmaschine
#define DISABLE       0    //globale Zustandsmaschine: Zustand DISABLE
#define JOGGING       1    //globale Zustandsmaschine: Zustand JOGGING
#define HOMING        2    //globale Zustandsmaschine: Zustand HOMING
#define POSITIONING   3    //globale Zustandsmaschine: Zustand POSITIONING
long lSubStateHoming;      //Unterzustand im Hauptzustand "Homing"
#define HOMING_STOPPED 0
#define HOMING_STARTED 1
#define HOMING_READY   2
long lSubStatePositioning; //Unterzustand im Hauptzustand "Positioning"
#define POSITIONING_STOPPED 0
#define POSITIONING_STARTED 1
// allgemeine Variablen
long lDriveState;         //Umrichterzustand, entspricht bei MDx der 7-Segmentanzeige
long lErrorCode;          //Fehlercode
// Datenstrukturen Prozessdaten
GSPODATA10 tPA;           //Ausgangsdaten (SPS ->  Antrieb)
SSPIDATA10 tPE;           //Eingangsdaten (Antrieb ->  SPS)
/*=============================================
Hauptfunktion (IPOS-Eintrittsfunktion)
===============================================*/
main()
{
   _WdOn( 5000 );  //Watchdog aktivieren =>  im Fehlerfall Code 41
   while (!MY_READY_TO_RUN)
   { //max. 5000ms warten bis Umrichter-Firmware komplett hochgelaufen
   }
   _WdOff(  ); //Hochlauf-Watchdog abschalten
   /*-------------------------------------
   Initialisierung
   --------------------------------------*/

   // Hauptzustand initialisieren
   lGlobalStateMachine = 0;

   // Initialisierung der Skalierung fuer die Position
   lScalingNumerator = 1;
   lScalingDenominator = 1;

   // Feldbusvariablen für Getsys und Setsys-Befehle initialisieren
   tPA.BusType = MY_FBUS_TYPE; //Prozessdatenbetrieb über Quelle s.o.
   tPA.Len = tPE.Len = MY_PD_LENGTH; //PD-Länge s.o.

   // Task2 aktivieren
   _SetTask2(T2_START, fnTask2); //Task 2 zum Debuggen hier aus- und unten einkommentieren
```

```
   /*-------------------------------------
   Hauptprogramm-Schleife
   ---------------------------------------*/
   while(1)
   {
      //Hauptzustandsmaschine bearbeiten
      switch (lGlobalStateMachine)
      {
         // keine Betriebsart angewaehlt oder moeglich
         case DISABLE:
            break;
         // Betriebsart "Tippen"
         case JOGGING: fnJogging();
            break;
         // Betriebsart "Referenzieren"
         case HOMING:fnHoming();
            break;
         // Betriebsart "Positionieren"
         case POSITIONING: fnPositioning();
            break;
         //Programmierfehler - unzulässiger Zustand
         default: _AxisStop(AS_PSTOP);
            lGlobalStateMachine = lOpMode = -1;
            break;
      } // Ende switch (lGlobalStateMachine)
   } // Ende while(1)
} // Ende main
```

```
/*=============================================
= Funktion: fnJogMode()
= Tippen der Achse. Mit 2 Eingängen kann die Achse nach rechts und links verfahren werden.
= Ist keine Tipptaste gesetzt, steht der Antrieb in Halteregelung. Würde mit Freigabe
= getippt, würde die Hauptzustandsmaschine in den State 99 wegspringen.
==============================================*/
fnJogging()
{
   // Instruktionen beim Eintritt in den Hauptzustand "Jogging"
   // Betriebsart quittieren
   _BitSet(lPE_StatusWord,11);
   _BitClear(lPE_StatusWord,12);

   // zyklische Bearbeitung, solange Hauptzustand "Jogging" aktiv
   do
   {
      // PA Daten einlesen
      _GetSys( tPA.BusType ,GS_PODATA );

      if (MY_JOG_PLUS&&(!MY_JOG_MINUS))
      {
         tPosVelocities.CW = tPosVelocities.CCW = tPA.PO2;
         _SetSys( SS_POSSPEED, tPosVelocities );
         TargetPos = ActPos_Mot + 409600;
      }

      if (MY_JOG_MINUS&&(!MY_JOG_PLUS))
      {
         tPosVelocities.CW = tPosVelocities.CCW = tPA.PO2;
         _SetSys( SS_POSSPEED, tPosVelocities );
         TargetPos = ActPos_Mot - 409600;
      }

      if ((MY_JOG_MINUS && MY_JOG_PLUS)||((!MY_JOG_MINUS)&&(!MY_JOG_PLUS)))
         _AxisStop( AS_PSTOP );

   } while (lGlobalStateMachine==JOGGING);

   // Instruktionen beim Verlassen des Hauptzustands "Jogging"
   // Antrieb stoppen
   _AxisStop( AS_PSTOP );
   // Betriebsart ablöschen
   _BitClear(lPE_StatusWord,11);
   _BitClear(lPE_StatusWord,12);

}// end fnJogging()
```

```
/*=============================================
= Funktion: fnHoming()
= Referenzfahrt der Achse
= Es wirken die Parameter der Gruppe 97x
= Eine positive Flanke an REF-START startet eine neue Referenzfahrt
==============================================*/
fnHoming()
{
   // Instruktionen beim Eintritt in den Hauptzustand "Homing"
   // Unterzustand definieren
   lSubStateHoming = HOMING_STOPPED;
   // Betriebsart quittieren
   _BitClear(lPE_StatusWord,11);
   _BitSet(lPE_StatusWord,12);

   // zyklische Bearbeitung, solange Hauptzustand "Homing" aktiv
   do
   {
      // PA Daten einlesen
      _GetSys( tPA.BusType ,GS_PODATA );

      switch (lSubStateHoming)
      {
         case HOMING_STOPPED: if (MY_START_HOMING)
                              {
                                 _Go0(GO0_U_NW_CAM);
                                 lSubStateHoming = HOMING_STARTED;
                              }
                              break;
         case HOMING_STARTED: if (!MY_START_HOMING)
                              {
                                 _Go0(GO0_RESET);
                                 lSubStateHoming = HOMING_STOPPED;
                              }
                              if (MY_REFERENCED)
                              {
                                 lSubStateHoming = HOMING_READY;
                              }
                              break;
         case HOMING_READY:   if (!MY_START_HOMING)
                              {
                                 lSubStateHoming = HOMING_STOPPED;
                              }
                              break;
      }
   } while (lGlobalStateMachine==HOMING);
   // Instruktionen beim Verlassen des Hauptzustands "Homing"
   // Antrieb stoppen
   if (lSubStateHoming==HOMING_STARTED)
   {
      _Go0(GO0_RESET);
      lSubStateHoming = HOMING_STOPPED;
   }
   // Betriebsart ablöschen
   _BitClear(lPE_StatusWord,11);
   _BitClear(lPE_StatusWord,12);

}// Ende fnHoming
```

```
/*=============================================
= Funktion: fnPositioning()
= Positionierbetrieb
==============================================*/
fnPositioning()
{
   // Instruktionen beim Eintritt in den Hauptzustand "Positioning"
   // Unterzustand definieren
   lSubStatePositioning = POSITIONING_STOPPED;
   // Betriebsart quittieren
   _BitSet(lPE_StatusWord,11);
   _BitSet(lPE_StatusWord,12);

   // zyklische Bearbeitung, solange Hauptzustand "Positioning" aktiv
   do
   {
      // PA Daten einlesen
      _GetSys( tPA.BusType ,GS_PODATA );

      switch (lSubStatePositioning)
      {
         case POSITIONING_STOPPED:if (MY_START_POSITIONING)
                                  {
                                     tPosVelocities.CW = tPosVelocities.CCW = tPA.PO2;
                                     _SetSys( SS_POSSPEED, tPosVelocities );
                                     TargetPos = (lScalingNumerator * tPA.PO3)
                                        / lScalingDenominator;
                                     lSubStatePositioning = POSITIONING_STARTED;
                                  }
                                  break;
         case POSITIONING_STARTED:if (MY_START_POSITIONING)
                                  {
                                     tPosVelocities.CW = tPosVelocities.CCW = tPA.PO2;
                                     _SetSys( SS_POSSPEED, tPosVelocities );
                                     TargetPos = (lScalingNumerator * tPA.PO3)
                                        / lScalingDenominator;
                                  }
                                  else
                                  {
                                     _AxisStop( AS_PSTOP );
                                     lSubStatePositioning = POSITIONING_STOPPED;
                                  }
                                  break;
      }
   }while (lGlobalStateMachine==POSITIONING);

   // Instruktionen beim Verlassen des Hauptzustands "Positioning"
   // Antrieb stoppen
   if (lSubStatePositioning==POSITIONING_STARTED)
      _AxisStop( AS_PSTOP );
      // Betriebsart ablöschen
      _BitClear(lPE_StatusWord,11);
      _BitClear(lPE_StatusWord,12);

}// Ende fnPositioning()
```

```
/*=============================================
= Funktion: fnTask2()
= Für zeitkritische Programmteile, die asynchron
= zu Task 1 laufen können
=
================================================*/
fnTask2()
{

   // Umrichterzustand auslesen
   _GetSys( lDriveState,GS_SYSSTATE );
   // Fehlernummer auslesen
   _GetSys( lErrorCode,GS_ERROR );
   // PA Daten einlesen
   _GetSys( tPA.BusType ,GS_PODATA );

   // Sprungverteiler bilden/ Betriebsart wählen
   // virtuelle Feldbus-Klemmen nur nutzbar, wenn keine DIO oder DIP gesteckt
   // in diesem Fall "lPA_ControlWordHigh   = tPA.PI1 > >  8;" verwenden
   // sonst "lPA_ControlWordHigh   = InputLevel > >  9;" //Bit0 auf Bit 0 schieben
   lPA_ControlWordHigh   = tPA.PO1 > >  8; //Bit8 auf Bit 0 schieben
   lOpMode = (lPA_ControlWordHigh & MY_OP_MODE )> >  3; //Bit 3,4 = Betriebsart

   // Zustandsuebergaenge bilden
   switch ( lGlobalStateMachine )
   {
      // keine Betriebsart angewaehlt oder moeglich
      case DISABLE:  if (lDriveState> =0xA)
                     {
                        if (lOpMode==JOGGING)
                           lGlobalStateMachine = JOGGING;
                        if (lOpMode==HOMING)
                           lGlobalStateMachine = HOMING;
                        if (lOpMode==POSITIONING)
                           lGlobalStateMachine = POSITIONING;
                     }
                     break;
      // Betriebsart "Tippen"
      case JOGGING:  if (lDriveState> =0xA)
                     {
                        if (lOpMode==DISABLE)
                           lGlobalStateMachine = DISABLE;
                        if (lOpMode==HOMING)
                           lGlobalStateMachine = HOMING;
                        if (lOpMode==POSITIONING)
                           lGlobalStateMachine = POSITIONING;
                     }
                     else
                        lGlobalStateMachine = DISABLE;
                     break;
```

```
      // Betriebsart "Referenzieren"
      case HOMING:   if (lDriveState> =0xA)
                     {
                        if (lOpMode==JOGGING)
                           lGlobalStateMachine = JOGGING;
                        if (lOpMode==DISABLE)
                           lGlobalStateMachine = DISABLE;
                        if (lOpMode==POSITIONING)
                           lGlobalStateMachine = POSITIONING;
                     }
                     else
                        lGlobalStateMachine = DISABLE;
                     break;
      // Betriebsart "Positionieren"
      case POSITIONING:if ((lDriveState> =0xA)&&(MY_REFERENCED))
                     {
                        if (lOpMode==JOGGING)
                           lGlobalStateMachine = JOGGING;
                        if (lOpMode==HOMING)
                           lGlobalStateMachine = HOMING;
                        if (lOpMode==DISABLE)
                           lGlobalStateMachine = DISABLE;
                     }
                     else
                        lGlobalStateMachine = DISABLE;
                     break;
      //Programmierfehler - unzulässiger Zustand
      default:       _AxisStop( AS_PSTOP );
                     lGlobalStateMachine = lOpMode = -1;
                     break;
   }//Ende switch (lGlobalStateMachine)

   //Prozesseingangsdaten neu bilden und an SPS senden
   fnBuildStatusWord();  //Statuswort bilden

   _GetSys(lActVelocity,GS_ACTSPEED); //Istdrehzahl lesen
   tPE.PI2 = lActVelocity;            // Istdrehzahl ausgeben

   lActPosition = ActPos_Mot; // Istposition
   tPE.PI3 = (lScalingDenominator * lActPosition) / lScalingNumerator; //Istposition

   _SetSys(SS_PIDATA, tPE.Len); //PD abschicken
}
/*=============================================
= Funktion: fnBuildStatusWord()
= Hier werden Bit 0-7 des Statuswortes gebildet
= und bei Fehler die anderen Ausgänge durch den
= Fehlercode ersetzt.
===============================================*/
fnBuildStatusWord()
{
   _BitMove(lPE_StatusWord,1, StatusWord,2); //Umrichter betriebsbereit
   _BitMove(lPE_StatusWord,2, StatusWord,20); //IPOS referenziert
   _BitMove(lPE_StatusWord,3, StatusWord,19); //Zielposition erreicht
   _BitMoveNeg(lPE_StatusWord,4, StatusWord,1); //Fehler
   if ( !MY_NO_ERROR )
   { // im Fehlerfall die Statusbits der Betriebsarten mit Fehlercode
   // überschreiben
      lPE_StatusWord = lPE_StatusWord & (lErrorCode << 8);
   }
   tPE.PI1 = lPE_StatusWord ;
}//end fnBuildStatusWord()
```

19 Compiler – Fehlermeldungen

Die Quelltextfehler, die vom Präprozessor und Compiler erkannt werden sind in Fehlerklasse und Fehlercode unterteilt.

Error-Class	Error-Code	mögliche Ursache
ANWEISUNG	NICHT GEFUNDEN SEMIKOLON	Im Schleifenkörper fehlen Anweisungen Semikolon hinter Anweisung fehlt
BEDINGT	DOPPELPUNKT	bei Bedingt-Anweisung fehlt Doppelpunkt ":"
BLOCK	ENDE	Block ohne schließende Klammer " }"
BREAK	SEMIKOLON	nach break fehlt Semikolon ";"
CASE	ILLEGALER TYP DOPPELPUNKT DEFAULT	nach case muss Konstante stehen nach case-Konstante muss Doppelpunkt stehen default-Zweig ist fehlerhaft oder an falscher Position
COMPILER	Fehlertext	interner Systemfehler (an SEW wenden)
CONTINUE	SEMIKOLON	nach continue fehlt Semikolon ";"
DECLARE	IDENTIFIER KEINE VARIABLE ZU VIELE #DEFINE	der Bezeichner hinter #declare ist ungültig #declare muss eine Variable beschreiben die Anzahl der #define übersteigt die Ressourcen
DEFINE	IDENTIFIER SYMBOL SEQUENZ ZU VIELE #define	Identifier hinter #define ungültig Symbolsequenz hinter #define ungültig die Anzahl der #define übersteigt die Ressourcen
DO	WHILE KLAMMER AUF KLAMMER ZU SEMIKOLON	fehlendes while nach do-Anweisung nach while fehlt runde Klammer auf "(" nach while fehlt runde Klammer zu ")" nach while fehlt Semikolon ";"
FAKTOR	KLAMMER ZU	nach Ausdruck in Klammern fehlt Klammer zu ")"
FKT.AUFRUF	KLAMMER ZU ANZAHL ARGS	schließende Klammer ")" nach Funktionsname fehlt die Anzahl der Argumente ist falsch
FOR	KLAMMER AUF SEMIKOLON KLAMMER ZU	nach for fehlt runde Klammer auf "(" zwischen den for-Ausdrücken fehlt Semikolon ";" nach for fehlt runde Klammer zu ")"
FUNKTION	KLAMMER AUF KLAMMER ZU	Bei Funktionsdeklaration fehlt (fehlen) runde Klammer(n).
IDENTIFIER	NICHT GEFUNDEN	unbekannter Identifier
IF	KLAMMER AUF KLAMMER ZU	nach if fehlt runde Klammer auf "(" nach if fehlt runde Klammer zu ")"
KONSTANTE	ILLEGALER TYP	Syntax von Dez.-, Binär- oder Hexkonstante falsch
PRAGMA	IDENTIFIER VARIABLEN-BEREICH	hinter #pragma steht ungültiges Schlüsselwort der Variablenbereich ist unzulässig
PRÄPROZESSOR	KEINE VARIABLE ZU VIELE #include QUELLTEXT ZU LANG HEADER FILE NAME DATEI ÖFFNEN DATEI ENDE ZEILE ZU LANG	hinter numof muss Variablenname stehen zu viele #include-Direktiven verschachtelt der Quelltext übersteigt die max. zulässige Länge Name der Header-Datei ungültig kann angegebene Datei nicht öffnen unerwartet Dateiende erreicht Quelltextzeile zu lang
RETURN	SEMIKOLON	nach return fehlt Semikolon ";"
SWITCH	KLAMMER AUF KLAMMER ZU	nach switch fehlt runde Klammer auf "(" oder vom Block fehlt öffnende Klammer "{" nach switch fehlt runde schließende Klammer ")" oder nach Block fehlt schließende Klammer "}"
UNDEF	IDENTIFIER	Identifier hinter #undef ungültig
WHILE	KLAMMER AUF KLAMMER ZU	nach while fehlt runde Klammer auf "(" nach while fehlt runde Klammer zu ")"

20 Assembler – Einführung

20.1 Einstellen der Anwenderverfahreinheiten

Im Programmkopf des Assemblers können die Wegfaktoren "ZÄHLER (Numerator), NENNER (Denominator)" und "EINHEIT (Unit)" zur Bestimmung der Anwenderverfahreinheit (z. B. mm, Umdr.) eingetragen werden.

20.1.1 Wegfaktoren ZÄHLER /NENNER

IPOS$^{plus®}$ rechnet intern immer mit 4096 Inkrementen/Motorumdrehung. Wenn der Anwender seine Verfahrbefehle nicht in Inkremente/Motorumdrehung, sondern in eigene Anwendereinheiten (z. B. mm, Umdr.) programmieren möchte, müssen die Wegfaktoren "ZÄHLER" und "NENNER" wie nachfolgend beschrieben eingestellt werden. Eine Ausnahme bilden Verfahrbefehle mit Variablen als Argument. Diese können dann nur in Inkremente/Motorumdrehung angegeben werden.

Die Umrechnung erfolgt nach:

$$\text{Inkremente} = \frac{\text{ZÄHLER}}{\text{NENNER}} \cdot \text{Anwenderverfahreinheit}$$

	HINWEIS Falls Zähler oder Nenner nicht ganzzahlig sind, kann eine höhere Rechengenauigkeit erzielt werden, wenn Zähler und Nenner um den gleichen Faktor (z. B. 10, 100, 1000, ...) erweitert werden. Durch die Erweiterung ergibt sich keine Einschränkung des Verfahrbereiches.

Wegfaktor ZÄHLER

Anzahl der Inkremente, die der Motor verfährt, um eine definierte Strecke zu verfahren.

Einstellbereich: 0 ..... **1** ..... 2^{31} - 1

Wegfaktor NENNER

Angabe der definierten Strecke in Anwenderverfahreinheiten.

Einstellbereich: 0 ..... **1** ..... 2^{31} - 1

Beispiel

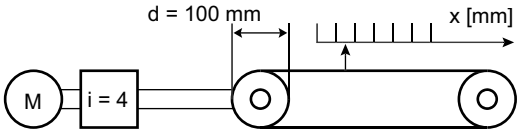
12376 Inkremente entsprechen 120 mm. ZÄHLER (Numerator) = 12376, NENNER (Denominator) = 120, EINHEIT (Unit) = mm.

Beispiel

Anhand von drei Fällen soll die Einstellung der Wegfaktoren Zähler / Nenner einer Lineareinheit zur Positionsvorgabe erläutert werden:

- Fall A: Positionsangabe in mm der Linearachse
- Fall B: Positionsangabe in Inkrementen
- Fall C: Positionsangabe in Abtriebsumdrehungen

Mechanischer Aufbau der Lineareinheit:



511398411

Fall A: mm

Die definierte Strecke zur Berechnung ist eine Umdrehung des Abtriebsrads.

- Wegfaktor ZÄHLER = Inkremente/Motorumdrehung × Getriebeübersetzung i = 4096 × 4 = 16384
- Wegfaktor NENNER = Abtriebsdurchmesser × π = 314,15926

Der Wegfaktor NENNER ist nicht ganzzahlig, die Genauigkeit der Umrechnung lässt sich durch einen Erweiterungsfaktor erhöhen. Der Erweiterungsfaktor sollte möglichst groß sein, allerdings darf das Ergebnis den Einstellbereich nicht überschreiten (Erweiterungsfaktor hier z. B. 100 000).

- Wegfaktor ZÄHLER = 16384 × 100000 = 1638400000
- Wegfaktor NENNER = 314,15926 × 100000 = 31415926

	HINWEIS Da π keine endliche Zahl ist, ist die Angabe der Zielposition immer fehlerhaft

Fall B: Inkremente

- Wegfaktor ZÄHLER = 1
- Wegfaktor NENNER = 1

Fall C: Abtriebs-umdrehungen

- Wegfaktor ZÄHLER = Inkremente/Motorumdrehung × Getriebe i = 4096 × 4 = 16384
- Wegfaktor NENNER = 1

Praxishinweis

Praxishinweis zur Wegfaktorenbestimmung bei der Inbetriebnahme.

z. B. Einstellung der Anwenderverfahreinheiten in mm

1. Wegfaktoren ZÄHLER und NENNER beide auf den Wert 1 stellen (→ Anwenderverfahreinheiten = Inkremente).
2. Im Handbetrieb eine beliebige Anzahl von Anwenderverfahreinheiten (hier Inkremente) verfahren, z.B. 100 000 Inkremente.
3. An der Anlage den in Punkt 2. zurückgelegten Weg messen, z.B.:
   - Startposition = 1000 mm
   - Zielposition = 1453 mm
   - zurückgelegter Weg = 453 mm
4. Eintrag der Wegfaktoren im Programmkopf des Assemblers:
   - Wegfaktor ZÄHLER = 100 000
   - Wegfaktor NENNER = 453

20.1.2 EINHEIT

Im Programmfenster werden alle Wegangaben mit Einheit angezeigt.

Die Einheit kann im Programmkopf bei EINHEIT eingetragen werden und darf maximal fünf Zeichen lang sein.

	HINWEIS Dieser Eintrag ist rein symbolisch und hat keine Auswirkungen auf die Funktion des Antriebs

20.2 Erste Schritte

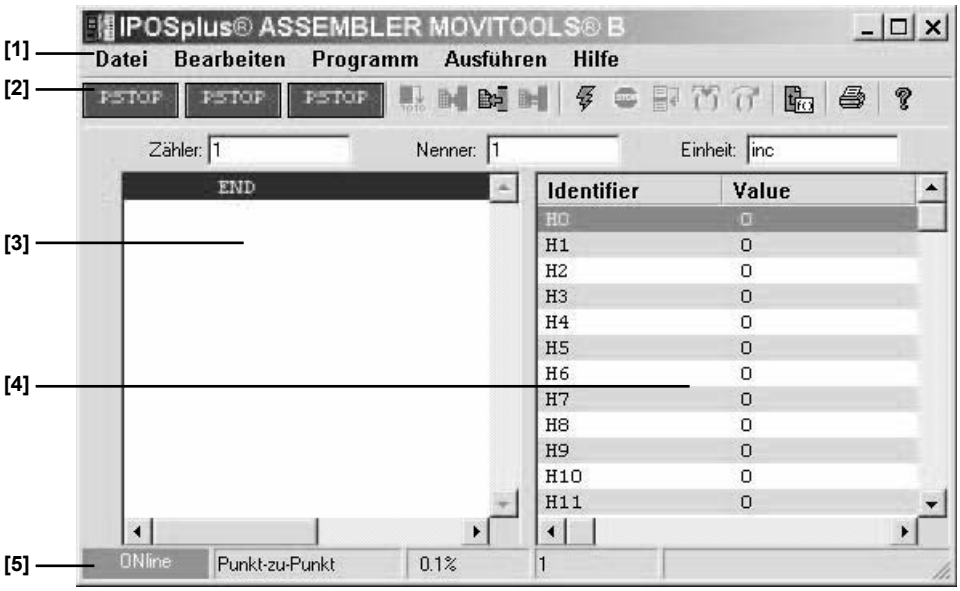
20.2.1 IPOS$^{plus®}$ Assembler starten

Der IPOS$^{plus®}$-Assembler wird aus dem MOVITOOLS® MotionStudio heraus gestartet.

- Starten Sie den IPOS$^{plus®}$-Assembler in analoger Weise wie den IPOS$^{plus®}$-Compiler.

  Beziehen Sie sich dazu auf den Abschnitt: "Schritt 1: IPOSplus-Compiler mit MOVITOOLS MotionStudio starten (Seite 143)"

Nach dem Starten des IPOS$^{plus®}$-Assembler wird folgende Programmoberfläche sichtbar:

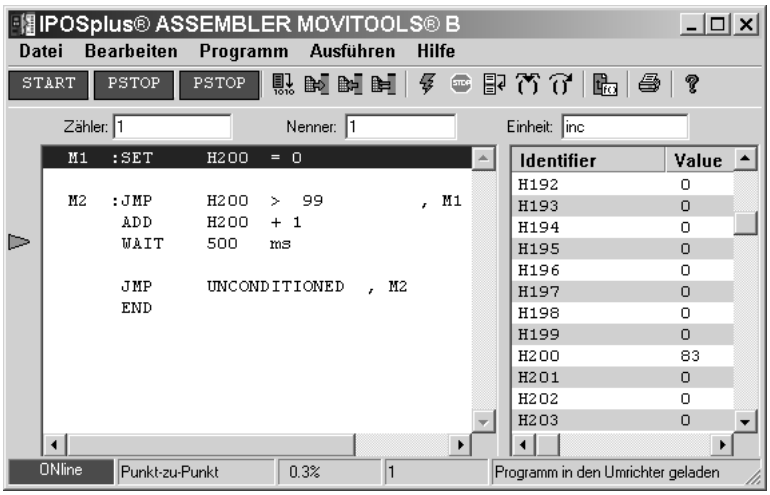


511471115

[1] Menüleiste
[2] Symbolleiste
[3] Programmfenster
[4] Variablenfenster
[5] Statusleiste

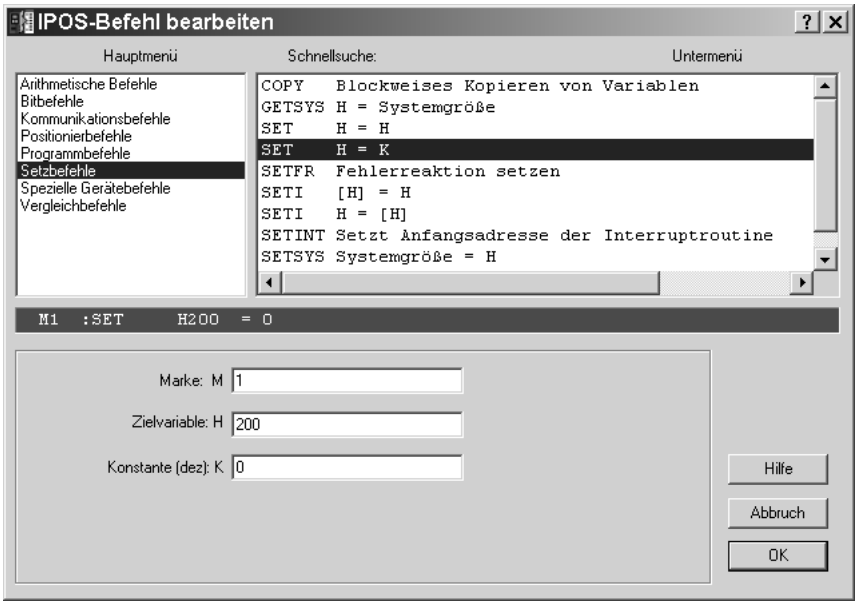
20.2.2 Erstellen eines neuen Programms

Um den IPOS$^{plus®}$-Assembler kennen zu lernen, werden Sie jetzt ein erstes Programm schreiben, welches eine Variable im Takt von 500 ms von 0 bis 99 inkrementiert.



511514635

Die Eingabe der Assembler-Befehle erfolgt über die Eingabeunterstützung. Um das Dialogfenster der Eingabeunterstützung zu öffnen, wählen Sie das Symbol .



511519755

Um den ersten Assembler-Befehl in das Programm einzufügen, klicken Sie in [Hauptmenü] auf [Setzbefehle] und wählen Sie im rechten Fenster "SET H = K" aus.

Im unteren Teil des Dialogfensters müssen Sie jetzt die Sprungmarke der Befehlszeile, die Zielvariable und den Wert (Konstante) auf den die Variable gesetzt werden soll eintragen. Durch Klicken der Schaltfläche [OK] wird die Eingabeunterstützung geschlossen und der Befehl in das Programm eingefügt.

Fügen Sie jetzt die restlichen Befehle mit Hilfe der Eingabeunterstützung in das Programm ein. Die folgende Tabelle gibt die Parameter für die Befehle des gesamten Programms an. Wenn Sie Hilfe zu einem Befehl benötigen, markieren Sie diesen und Drücken Sie die <F1> -Taste.

Command	Label	Target	Condition	Constant	Destination
SET	1	200		0	
JMP	2	200	>	99	1
ADD		200		1	
WAIT				500	
JMP			UNCONDI-TIONED		2

20.2.3 Compilieren und Starten des Programms

Um das geschriebene Assembler-Programm in eine für den Umrichter verständliche Form zu bringen, muss der Quellcode übersetzt werden.

Dies geschieht mit dem Menüpunkt [Programm] / [Übersetzen] oder über das Symbol der Symbolleiste.

War das Übersetzen des Programms erfolgreich, so wird dies in der Statusleiste angezeigt.

511550475

Im nächsten Schritt muss das übersetzte Programm in den Umrichter übertragen werden. Dies geschieht über den Menüpunkt [Programm] / [Übersetzen + Download] oder über das Symbol der Symbolleiste.

In der Statuszeile wird angezeigt, ob der Download des Programms erfolgreich war.

511552011

Jetzt ist das IPOS$^{plus®}$-Programm spannungsausfallsicher im Gerät gespeichert.

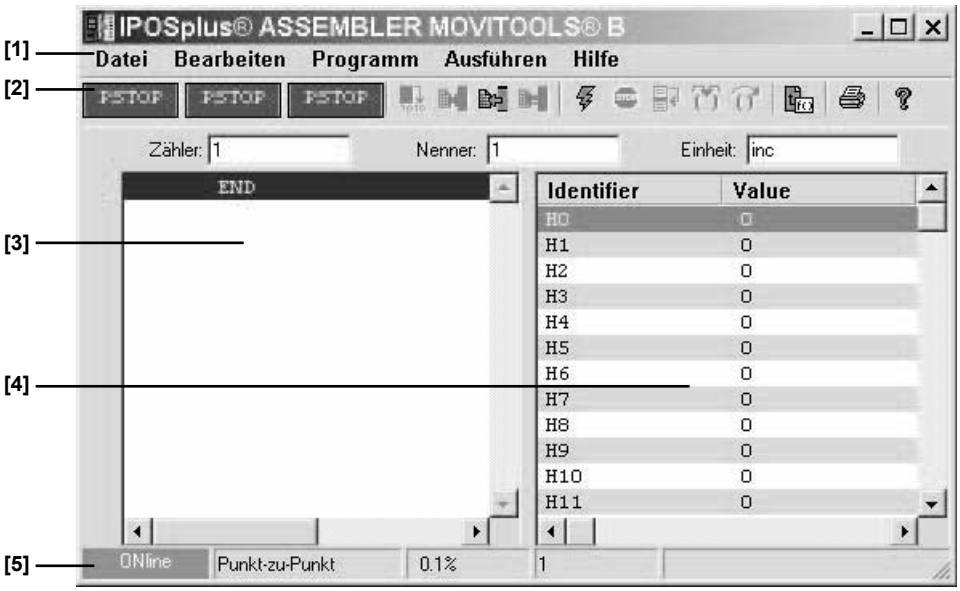
IPOS$^{plus®}$-Programme können auch mit dem Bediengerät DBG60B / DBG11B von einem MOVIDRIVE® auf ein anderes MOVIDRIVE® übertragen werden. Dies geschieht mit den Parametern P807 (Kopie MDX -> DBG) und P806 (Kopie DBG -> MDX).

Nachdem das Programm in den Umrichter übertragen worden ist, kann es gestartet werden. Dazu wird der Menüpunkt [Ausführen] / [Start] ausgewählt. Alternativ kann auch das Symbol in der Symbolleiste betätigt werden. Nach dem Starten des Programms erscheint im Projektfenster ein grüner Pfeil (Programmzeiger), der die aktuell bearbeitete Programmzeile anzeigt. Die Anzeige in der Symbolleiste springt von PSTOP auf START.

Um die Programme in Task 1, Task 2 und Task 3 zu stoppen, wird der Menüpunkt [Ausführen] / [Stopp] ausgewählt. Alternativ kann das Symbol in der Symbolleiste betätigt werden. Nach dem Stoppen des Programms (alle Tasks) wird der Programmzeiger rot und bleibt an der ersten Befehlszeile der Task 1 stehen. Die Statusanzeige für den Programmablauf in der Symbolleiste springt von START auf PSTOP um.

21 Assembler – Editor

Nach dem Starten des IPOS$^{plus®}$-Assembler zeigt sich der Editor:



511872523

[1] Menüleiste
[2] Symbolleiste
[3] Programmfenster
[4] Variablenfenster
[5] Statusleiste

In der Statusleiste wird angezeigt, ob das Gerät online oder offline ist. Außerdem wird die Belegung des Programmspeichers in Prozent und die Nummer der angewählten Programmzeile angezeigt.

Unter der Symbolleiste sind drei Eingabefelder:

- Zähler
- Nenner
- Einheit

Über das Zähler-Nenner-Verhältnis können Positionsvorgaben skaliert und somit in beliebigen, vom Anwender gewählten Einheiten vorgegeben werden. Über Variablen vorgegebene Positionen werden damit nicht skaliert.

21.1 Beispiel

Der Geber eines Motors liefert 4096 Inkremente pro Umdrehung. Am Motor ist eine Spindel mit einer Steigung 10 mm/Umdrehung, die einen Wagen horizontal verfährt.

Der Anwender will die anzufahrenden Positionen in mm angeben.

Zähler und Nenner sind in diesem Fall folgendermaßen zu belegen.

- Zähler: 4096
- Nenner: 10
- Einheit: mm

Beim Einfügen eines Positionierbefehls kann jetzt die gewünschte Position direkt in mm angegeben werden, wenn es sich um eine Konstante handelt.

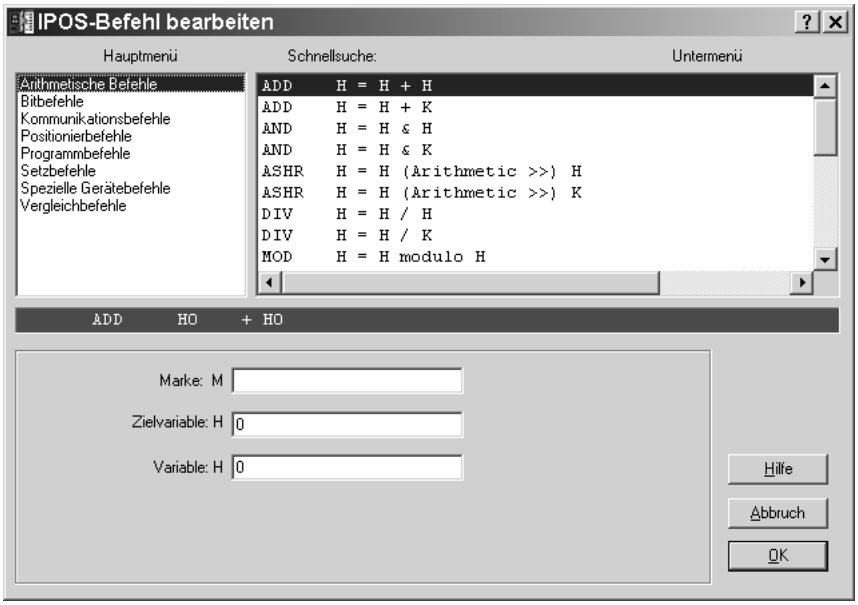
Die Betriebszustände der Anwenderprogramme Task 1, Task 2 und Task 3 sind:

- START (Programm läuft);
- PSTOP (Programm ist gestoppt);
- BREAK (Programm wird nur bis zur markierten Zeile abgearbeitet);
- STEP (Programm wird durch Betätigen der F7-Taste zeilenweise abgearbeitet).

21.2 Programme erstellen

21.2.1 Befehlszeilen einfügen

Durch Betätigen des Symbols , mit der Taste <Einfg> oder über den Menüpunkt [Bearbeiten] / [Befehl einfügen...] gelangt man in die Eingabeunterstützung.



511877899

In der Eingabeunterstützung können alle in IPOS$^{plus®}$ verfügbaren Befehle angewählt werden.

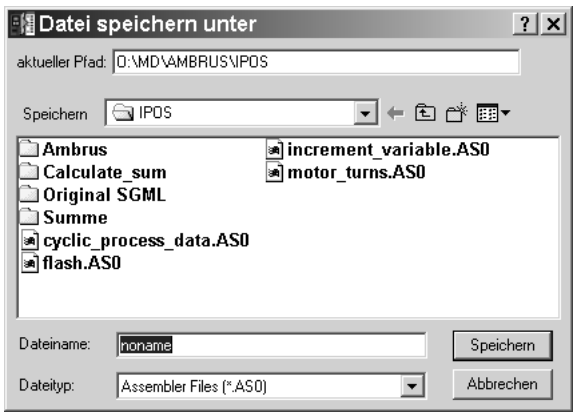
Mit Anwahl eines Befehls erscheint eine Eingabemaske, in der verfügbare Argumente der gewählten Funktion eingegeben werden müssen. Sind alle Argumente eingegeben, so wird der Befehl mit der Schaltfläche [OK] eingefügt.

Mit der Taste <Entf> können markierte Befehlszeilen aus dem Programm gelöscht werden.

Ebenso können ganze Befehlsblöcke mit der Maus markiert und mit [Bearbeiten] / [Kopieren] und [Bearbeiten] / [Einfügen] eingefügt oder mit [Bearbeiten] / [Ausschneiden] gelöscht werden.

Ein eingefügter Befehl kann durch einen Doppelklick auf die Befehlszeile im Projektfenster oder über das Menü [Bearbeiten] / [Befehl ändern...] geändert werden.

Das geschriebene Assembler-Programm kann mit [Datei] / [Speichern...] gespeichert werden.



511934475

Assembler-Programme werden bei MOVIDRIVE® A mit der Endung *.MDX gespeichert, bei MOVIDRIVE® B mit der Endung *.AS0. Im Dialogfenster kann der Name und das Verzeichnis des Assembler-Programms angegeben werden.

	HINWEIS MDX-Dateien, die mit dem Assembler erzeugt wurden, enthalten nur den reinen Programm-Code und keine Parameter. MDX-Dateien, die mit SHELL zur Parametersicherung erzeugt wurden, enthalten Parameter und Programm-Code. Deshalb: Vorsicht beim Überschreiben vorhandener MDX-Dateien.

21.3 Kompilieren und Download

Um das geschriebene Assembler-Programm in eine für den Umrichter verständliche Form zu bringen, muss der Quellcode übersetzt werden.

Dies geschieht über den Menüpunkt [Programm] / [Übersetzen] oder über das Symbol der Symbolleiste.

War das Übersetzen des Programms erfolgreich, so wird dies in der Statusleiste angezeigt.

511940235

Im nächsten Schritt muss das übersetzte Programm in den Umrichter übertragen werden. Dies geschieht über den Menüpunkt [Programm] / [Übersetzen + Download] oder über das Symbol der Symbolleiste.

In der Statuszeile wird angezeigt, ob der Download des Programms erfolgreich war.

512034955

Jetzt ist das IPOS$^{plus®}$-Programm spannungsausfallsicher im Gerät gespeichert.

IPOS$^{plus®}$-Programme können auch mit dem Bediengerät DBG von einem MOVIDRIVE® auf ein anderes MOVIDRIVE® übertragen werden. Dies geschieht mit den Parametern P807 (Kopie MDX -> DBG) und P806 (Kopie DBG -> MDX).

21.4 Programme Starten / Stoppen

Nachdem das Programm in den Umrichter übertragen worden ist, kann es gestartet werden. Dazu wird der Menüpunkt [Ausführen] / [Start] ausgewählt oder das Symbol in der Symbolleiste betätigt. Nach dem Starten des Programms erscheint im Projektfenster ein grüner Pfeil (Programmzeiger), der die aktuell zu bearbeitende Programmzeile anzeigt. Die Anzeige in der Symbolleiste springt von PSTOP auf START.

Um die Programme in Task 1, Task 2 und Task 3 zu stoppen, wird der Menüpunkt [Ausführen] / [Stopp] ausgewählt oder das Symbol in der Symbolleiste betätigt. Nach dem Stoppen des Programms wird der Programmzeiger rot und bleibt an der ersten Befehlszeile stehen. Die Statusanzeige für Task 1, Task 2 und Task 3 in der Symbolleiste springt von START auf PSTOP um.

21.4.1 Variablenfenster

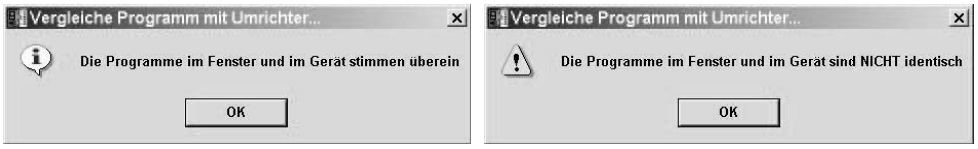
Im Variablenfenster werden alle verfügbaren Variablen und deren Inhalte angezeigt. Durch einen Doppelklick auf eine Variable kann der Inhalt der Variable direkt über die Tastatur verändert werden. Um den neuen Wert zu übernehmen, muss die Eingabetaste gedrückt werden.

21.5 Vergleich Datei mit Gerät

Um ein im Editorfenster geladenes Assembler-Programm mit einem im Umrichter geladenen Programm zu vergleichen, kann die Vergleichsfunktion des IPOS$^{plus®}$-Assemblers genutzt werden.

Der Aufruf der Vergleichsfunktion erfolgt über den Menüpunkt [Programm] / [Vergleiche mit Gerät] oder über das Symbol der Symbolleiste.

Stimmen die Programme überein, wird das linke der folgenden Fenster angezeigt. Wenn die Programme nicht übereinstimmen, wird das rechte Fenster angezeigt.



512040331

21.6 Debugger

Der integrierte Debugger ist ein Hilfsmittel, um ein Programm im Einzelschrittmodus zu durchlaufen. Wenn das Programm in den Umrichter übertragen wurde, stehen drei Funktionen zur Verfügung.

21.6.1 Ausführen bis Cursor

Über den Menüpunkt [Ausführen] / [Ausführen bis Cursor] oder mit dem Symbol aus der Symbolleiste wird das Programm bis zur aktuellen Cursor-Position ausgeführt.

21.6.2 Einzelschritt

Über den Menüpunkt [Ausführen] / [Einzelschritt] oder mit dem Symbol aus der Symbolleiste wird die Programmzeile abgearbeitet, auf der der Cursor gerade steht.

Über den Menüpunkt [Ausführen] / [Überspringen] oder mit dem Symbol aus der Symbolleiste wird die Programmzeile auf der der Cursor gerade steht nicht abgearbeitet. Der Cursor springt auf die nächste Programmzeile. Diese Funktion ist besonders hilfreich, wenn Funktionsaufrufe innerhalb des Programms beim Testen übersprungen werden sollen.

Mit dem Symbol aus der Symbolleiste, der Funktionstaste F5 oder dem Menüpunkt [Stopp] aus dem Menü [Ausführen] der Menüleiste kann das Programm jederzeit während des Debuggens gestoppt und zurücksetzt werden.

Mit dem Button aus der Symbolleiste oder dem Menüpunkt [Ausführen] / [Start] kann das Programm jederzeit während des Debuggens ab der aktuellen Cursor-Position gestartet werden.

Während das Programm läuft, kann es mit der Tastenkombination Alt+F5 unterbrochen werden. Der Ausführungsbalken wird nun auf dem nächsten auszuführenden Befehl positioniert.

Mit der Taste F4 [Ausführen bis Cursor], kann das laufende Programm unterbrochen werden. Das Programm wird in der Befehlszeile angehalten, auf der der Cursor steht.

21.7 Programm aus Umrichter laden

Im IPOS$^{plus®}$-Assembler besteht die Möglichkeit, ein Upload eines im Umrichter gespeicherten Programms durchzuführen. Dies geschieht über den Menüpunkt [Programm] / [Upload] oder über das Symbol der Symbolleiste. Beim Upload wird das gerade geöffnete Assembler-Programm verworfen.

Kommentarzeilen werden nicht im Umrichter gespeichert und sind deshalb nach einem Upload verloren.

21.8 Symbolübersicht

Im Folgenden sind die über die Symbolleiste aufrufbaren Funktionen aufgelistet.

Symbol	Menüpunkt	Beschreibung
	Datei -> Öffnen	Programm öffnen
	Datei -> Speichern	Programm speichern
	Programm -> Übersetzen	Programm übersetzen
	Programm -> Übersetzen + Download	Programm übersetzen und in Umrichter laden
	Programm -> Upload	Programm aus dem Umrichter laden
	Programm -> Vergleiche mit Gerät	Programm im Editor mit Programm im Umrichter vergleichen
	Ausführen -> Start	IPOS$^{plus®}$-Programm starten
	Ausführen -> Stopp	IPOS$^{plus®}$-Programm stoppen
	Ausführen -> Ausführen bis Cursor	Programm ausführen bis Cursor
	Ausführen -> Einzelschritt	Einzelschritt ausführen
	Ausführen -> Überspringen	Anweisung überspringen
	Bearbeiten -> Befehl einfügen	Eingabeunterstützung aufrufen
	Datei -> Drucken	Programm drucken
	Hilfe -> Benutzerhandbuch	Hilfe aufrufen

22 Assembler – Programmierung

22.1 Grundlagen

Der IPOS$^{plus®}$-Assembler ist Bestandteil des Programmpakets MOVITOOLS® MotionStudio. Die Eingabe des Assembler-Programms erfolgt maskengeführt.

22.1.1 Programmkopf

Stellen Sie im Programmkopf bei Anwenderprogrammen, in denen Positionierbefehle verwendet werden, die Anwenderverfahreinheiten ein.

22.1.2 Task 1 / Task 2 / Task 3

Bei der IPOS$^{plus®}$ Positionierung und Ablaufsteuerung kann ein Anwenderprogramm in 3 Teilprogrammen (Task 1 / Task 2 / Task 3) parallel und unabhängig voneinander ausgeführt werden.

22.1.3 Kommentare

Kommentare können wie Befehlszeilen an jeder Stelle im Anwenderprogramm eingefügt werden.

Die Kommentare sind nur auf dem PC speicherbar, beim Herunterladen auf den Umrichter werden sie nicht übertragen.

22.1.4 Programmverzweigungen

Programmverzweigungen sind durch Sprungmarken (M...) in Verbindung mit Sprungbefehlen (JMP... M...) möglich. Sprungmarken können vor jeder Befehlszeile gesetzt werden.

22.1.5 Unterprogrammtechnik

Unterprogramme können mit einem CALL-Befehl (CALL M...) aufgerufen werden. Die zugehörigen Sprungmarken (M...) werden vor dem ersten Befehl des Unterprogramms gesetzt. Ein Unterprogramm endet mit einem Return-Befehl (RET). Mit dem Return-Befehl erfolgt ein Rücksprung in die Zeile unterhalb des CALL-Befehls. Anschließend werden die folgenden Programmzeilen abgearbeitet. Geschachtelte Unterprogramme sind möglich, die Schachtelungstiefe sollte nicht größer als 16 sein.

i	**HINWEIS** Unterprogramme dürfen auf keinen Fall mit einem Sprung in ein Hauptprogramm oder ein weiteres Unterprogramm verlassen werden. Ein bedingtes Verlassen des Unterprogramms muss mit einem Sprung ans Ende (RET) des Unterprogramms erfolgen.

22.1.6 Programmschleifen

Programmschleifen bestehen aus Schleifenbeginn (LOOPB) und Schleifenende (LOOPE). Die Anzahl der Schleifendurchläufe wird im Argument des Befehls LOOPB festgelegt. Geschachtelte Schleifen sind möglich, die Schachtelungstiefe sollte nicht größer als 16 sein.

	HINWEIS Programmschleifen dürfen auf keinen Fall mit einem Sprungbefehl verlassen werden. Sprungbefehle innerhalb einer Programmschleife sind erlaubt.

22.1.7 Positionierbefehle

Die IPOS$^{plus®}$-Positionierung ermöglicht eine Punkt-zu-Punkt-Positionierung des MOVIDRIVE®- und MOVIDRIVE®*compact*-Antriebs.

22.1.8 Binäre / analoge Ein-/Ausgänge

Die Verarbeitung von binären und analogen Ein-/Ausgängen erfolgt über Variable. Zusätzlich können binäre Eingänge direkt über einen Sprungbefehl abgefragt werden.

22.1.9 Zugriff auf Systemgrößen / Parameter

Die in Kapitel "IPOS$^{plus®}$-Parameter" bei den Befehlen GETSYS und SETSYS als Argumente aufgeführten Antriebsparameter werden nachfolgend als Systemgrößen bezeichnet. Diese Systemgrößen können folgendermaßen verwendet werden:

- Lesen mit dem Befehl **GETSYS**, z. B. Wirkstrom und Ist-Drehzahl.
- Lesen über PO-Daten.
- Schreiben mit dem Befehl **SETSYS**, z. B. Festsollwerte.
- Schreiben von Feldbusdaten über PI-Daten.
- Über die Systemvariablen H458 ... H511 bei MOVIDRIVE® A / H458 ... H560 bei MOVIDRIVE® B können ebenfalls Systemgrößen gelesen und geschrieben werden.
- Mit dem Befehl **MOVLNK** können alle Parameter des eigenen Umrichters verändert oder mit anderen Umrichtern über RS-485 ausgetauscht werden.
- Mit dem Befehl **MOVLNK** können alle Parameter des MQX und des MOVIMOT® verändert oder mit anderen Geräten über SBus oder RS-485 ausgetauscht werden.

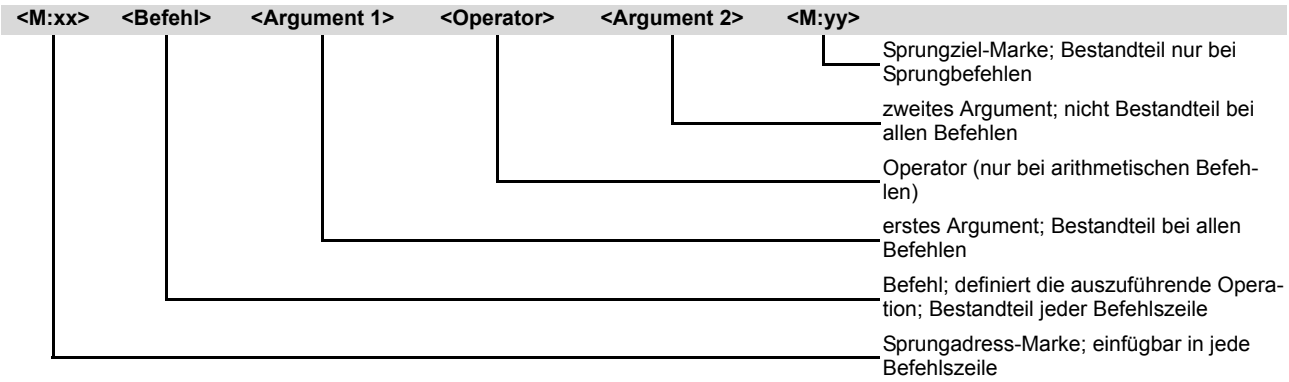
22.1.10 Variablen

Alle Variablen (H0 - H1023) können gelesen und geschrieben werden. Die Variablen haben einen Wertebereich von -2^{31} ... $+2^{31}$ - 1. Die Variablen H0 ... H127 werden sofort nach der Eingabe spannungsausfallsicher gespeichert, wenn sie in der Variablenliste eingetragen werden oder im IPOS$^{plus®}$-Programm mit dem Befehl "MEM" speichernd geschrieben werden. In den Variablen H458 ... H511 werden häufig benötigte Gerätegrößen zyklisch zur Verfügung gestellt (1 ms). Diese Variablen werden nachfolgend als Systemvariablen bezeichnet und sind in Kapitel "Übersicht über die Systemvariablen" genauer erklärt.

	HINWEIS **Vorsicht beim Schreiben von Systemvariablen!** Die Auswirkungen werden in Kap. "IPOS$^{plus®}$ mit Optionen" beschrieben.

22.1.11 Programmzeile

Befehls-Syntax:



Der Schreibbefehl für Variable und Indizes unterscheidet spannungsausfallsicheres und nicht spannungsausfallsicheres Beschreiben. Die Variablen H0 ... H127 können auf beide Weisen beschrieben werden, H128 ... H511 nur nicht spannungsausfallsicher.

Über MOVITOOLS® MotionStudio und Bediengerät werden H0 ... H127 immer spannungsausfallsicher gespeichert. Die SET-Anweisung eines Wertes auf eine Variable in einem IPOS$^{plus®}$-Programm erfolgt immer nicht spannungsausfallsicher. Um den aktuellen Stand spannungsausfallsicher zu speichern, muss im IPOS$^{plus®}$-Programm der Befehl MEM ausgeführt werden.

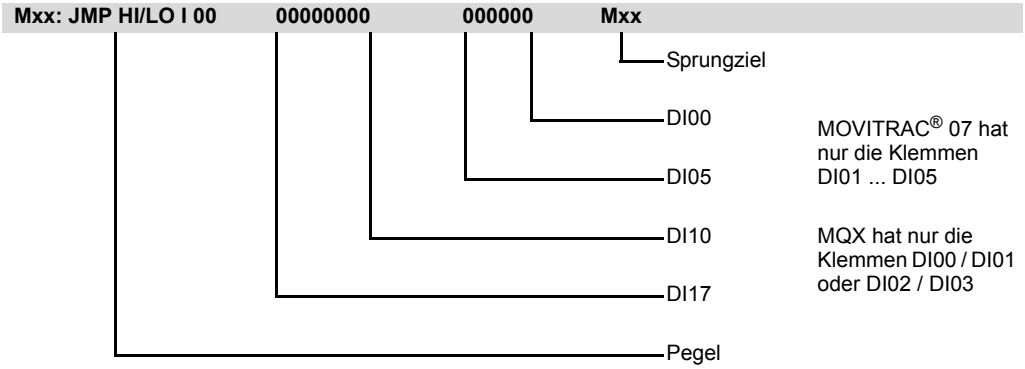
	HINWEIS Beim Benutzen des MEM-Befehls ist darauf zu achten, dass die fest speicherbaren Variablen H0 ... H127 sowie alle Parameter nicht zyklisch beschrieben werden, da die Anzahl der Speichervorgänge beim verwendeten Speichermedium EEPROM auf 10^5 Speichervorgänge begrenzt ist.

22.2 Binäre Ein-/Ausgänge

22.2.1 Binäre Eingänge

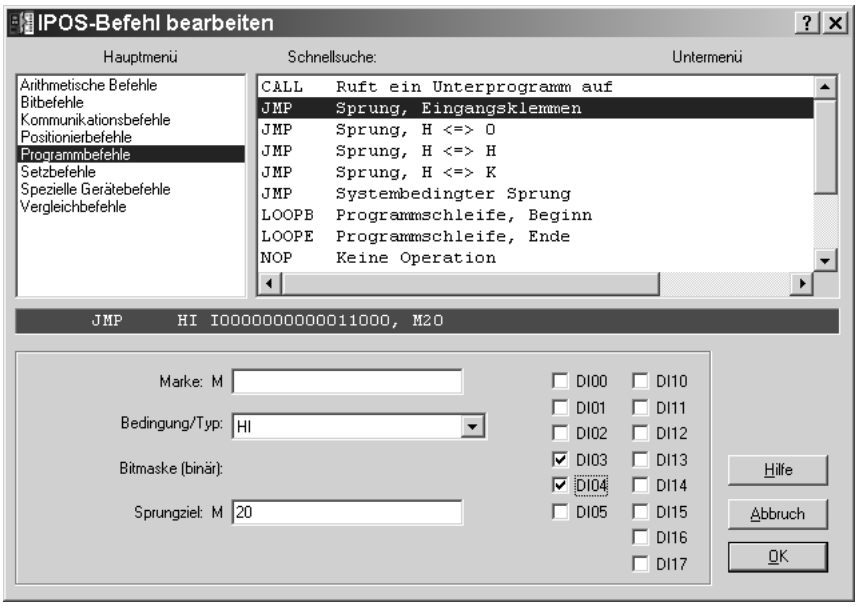
Direkte Abfrage

Im IPOS$^{plus®}$-Programm kann der Klemmenpegel von binären Eingängen mit Sprungbefehlen abgefragt werden. Dabei ist in der Eingabemaske der Klemmenpegel (HI/LO) auszuwählen, der zur Ausführung des Sprungbefehls führen soll. Klemmen, die für diese Funktion benutzt werden sollen, sind mit einer "1" in der Klemmenmaske zu definieren. Damit die Sprungbedingung für den JMP-Befehl erfüllt ist, müssen alle definierten Klemmen den gewählten Klemmenpegel haben.



Beispiel

Sprung zu Marke 20, wenn Eingänge DI03 und DI04 ein High-Signal (1) haben, ansonsten wird die nachfolgende Befehlszeile abgearbeitet:



512410763

Abfrage über Systemvariable

Die Klemmenpegel der binären Eingänge des Grundgeräts und der eventuell vorhandenen Option werden zyklisch auf der Systemvariablen **H483 INPUT LVL** (MOVIDRIVE® A) / **H520 INPUT LVL B** (MOVIDRIVE® B) abgebildet. Dabei sind die Bits der Systemvariablen H483 je einem Hardware-Eingang zugeordnet.

Die Zuordnung der Systemvariablen H483 für MOVIDRIVE® A / H520 für MOVIDRIVE® B zu den binären Eingangsklemmen ist im Kapitel "IPOS$^{plus®}$-Variablen / Übersicht über die Systemvariablen" beschrieben.

Die Abfrage der binären Eingänge im IPOS$^{plus®}$-Programm kann durch den Wert der Variablen H483 / H520 erfolgen. Dieses bietet sich beim Abfragen von Eingängen an, bei denen ein binär-codierter Wert übertragen werden soll, z. B. zur Anwahl einer Tabellenposition.

Tabelle 3: Beispiel für die Übertragung eines binär-codierten Werts über Eingangsklemmen des MOVIDRIVE® A

Beispiel: Lesen Eingänge	**Binäreingänge Gerät**					
Klemmenbezeichnung	DI05	DI04	DI03	DI02	DI01	DI00
Wertigkeit	2^5	2^4	2^3	2^2	2^1	2^0
Klemmenpegel	1	0	0	0	1	1
Bewertung	1×2^5	0×2^4	0×2^3	0×2^2	1×2^1	1×2^0
Variablenwert H483[1)]	32 +	0 +	0 +	0 +	2 +	1 = 35

1) falls alle Eingangsklemmen der DIO11A/DIP11A und das Steuerwort 2 "0"-Pegel haben

Durch eine Kombination der Befehle BMOV und JMP können auch binäre Klemmen abgefragt werden, die mit den höherwertigen Bits der Variablen H483 / H520 abgebildet werden. Das ist der Fall, wenn zwei Optionen mit Klemmenerweiterung gemeinsam gesteckt sind.

22.2.2 Binäre Ausgänge

Lesen der binären Ausgänge

Die Klemmenpegel der binären Ausgänge des Grundgeräts und einer eventuell vorhandenen Option werden zyklisch auf der Systemvariablen **H482 OUTPUT LVL** (MOVIDRIVE® A) / **H521 OUTPUT LVL B** (MOVIDRIVE® B) abgebildet. Dabei sind die Bits der Systemvariablen H482 je einem Hardware-Ausgang zugeordnet.

Im IPOS$^{plus®}$-Programm können einzelne Klemmenpegel von binären Ausgängen mit dem BMOV-Befehl ausgewertet werden. Mit dem Befehl BMOV wird ein Bit von der Systemvariablen H482 (OUTPUT LVL) / H521 (OUTPUT LVL B) auf eine beliebige Bitstelle (Wertigkeit) einer anderen Variablen kopiert. Mit dem folgenden Beispielprogramm wird der Klemmenpegel des Ausgangs DO02 abgefragt. Hierzu wird das Bit 1 der Systemvariablen H482 auf Bit 0 (Wertigkeit 2^0) von H200 kopiert. Dies ermöglicht eine einfache Abfrage (0 oder 1) des Klemmenpegels mit einem JMP-Befehl.

```
SET     H200 = 0
BMOV    H200.0  = H482.1
JMP     H200 == 1          ,Mxx
```

Alternativ können einzelne oder mehrere Klemmenpegel von binären Ausgängen durch logische Verknüpfung mit der Systemvariablen H 482 (OUTPUT LVL) / H521 (OUTPUT LVL B) herausgefiltert werden. Mit dem folgenden Beispielprogramm wird der Klemmenpegel des Ausgangs DO02 abgefragt:

```
M1 : SET     H200 = 2
AND     H200 & H482
JMP     H200 == 2     ,M1
```

Das Ergebnis der AND-Verknüpfung steht in der ersten Variablen, also H200. Das erste Argument muss deshalb eine Variable sein.

UND-Verknüpfung von H200 und H482		
H200 = 2	00000000010	(= DO01)
H482 =	11011100110	(= aktueller Stand der binären Ausgänge)
Ergebnis	00000000010	(= der Sprung wird ausgeführt, da H200 = 2 ist)

Setzen der binären Ausgänge

Voraussetzung für das Setzen der Ausgänge ist die Parametrierung der Binärausgänge (Parameter 620 / 621 (MOVIDRIVE® A), 620 ... 626 (MOVIDRIVE® B), 630 ... 637) auf "IPOS-AUSGANG".

Das Setzen von binären Ausgängen erfolgt mit den Systemvariablen:

- **H 480 (OPT. OUT IP)** für Option DIO11 / DIP11 (DO10 ... DO17)
- **H 481 (STD. OUT IP)** für das Grundgerät (DO01 / DO02 (MOVIDRIVE® A) DO01 ... DO05 (MOVIDRIVE® B); DB00 ist fest auf "/Bremse" eingestellt. Die Bremse wird von der Firmware direkt angesteuert. Der Ausgang darf deshalb nicht beschrieben werden.)

Setzen einzelner Ausgänge

Für das Setzen / Rücksetzen einzelner Ausgänge werden die **Befehle BSET und BCLR** verwendet. Hierzu ist die zur Klemme gehörende Bitnummer als Operand in die Befehlsmaske einzutragen. Im nachfolgenden Beispiel soll der Ausgang DO01 auf "1" gesetzt werden:

512416139

Übersicht über Befehle und Parameter zum Setzen / Rücksetzen von binären Ausgängen:

Tabelle 4: Befehle zum Setzen der binären Ausgänge

Gerät	Ausgang	Setzen (1-Pegel)	Rücksetzen (0-Pegel)	Parameter auf "IPOS-Ausgang"
MOVIDRIVE® A MOVIDRIVE® B	DB00	-	-	Fest auf "/Bremse" eingestellt, d.h. nicht programmierbar. Wird von der Firmware gesteuert.
MOVIDRIVE® A MOVIDRIVE® B	DO01 DO02	BSET H481.1 = 1 BSET H481.2 = 1	BCLR H481.1 = 0 BCLR H481.2 = 0	P620 P621
MOVIDRIVE® B	DO03 DO04 DO05	BSET H481.3 = 1 BSET H481.4 = 1 BSET H481.5 = 1	BCLR H483.1 = 0 BCLR H481.4 = 0 BCLR H481.5 = 0	P622 P623 P624
Option	DO10 ... DO17	BSET H480.0 = 1 ... BSET H480.7 = 1	BCLR H480.0 = 0 ... BCLR H480.7 = 0	P630 ... P637

Setzen mehrerer Ausgänge

Das gleichzeitige Setzen mehrerer binärer Ausgänge, z. B. zur Ausgabe einer binär-codierten Tabellenpositionsnummer, ist möglich, indem die Systemvariable H480 oder H481 mit dem dezimalen Wert der Tabellenpositionsnummer beschrieben wird.

Tabelle 5: Zuordnung Systemvariable H480 / H481 zu binären Ausgangsklemmen

Binäre Ausgänge	**Binärausgänge Option DIO11A/DIP11A H480**								**Binärausgänge Grundgerät**		
Klemmenbez.	DO17	DO16	DO15	DO14	DO13	DO12	DO11	DO10	DO02	DO01	DB00
Bits der System-variablen	7	6	5	4	3	2	1	0	2	1	0
Wertigkeit	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^2	2^1	2^0

Beispiel

Ausgabe der Tabellenpositionsnummer 11 über DIO11 ("11" benötigt 3 Ausgänge, also werden die Ausgänge DO10, DO11 und DO13 benötigt):

SET H480 = 11

Das Rücksetzen aller binären Ausgänge erfolgt durch Beschreiben der Systemvariablen H480 und H481 mit "0".

Das ist nur sinnvoll, wenn alle Ausgänge als IPOS$^{plus®}$-Ausgänge parametriert sind. Anders parametrierte Ausgänge werden zum Beispiel von der Firmware beschrieben und sollen nicht separat manipuliert werden.

SET H480 = 0 Rücksetzen der Ausgänge der Option DIO11 oder DIP11

SET H481 = 0 Rücksetzen der Ausgänge des Grundgeräts

22.3 Analoge Ein-/Ausgänge

Tabelle 6: Übersicht über die analogen Ein-/Ausgänge

Analoge Ein-/Ausgänge	Eingänge						Ausgänge					
	Grundgerät			Option			Option DIO11					
Ein-/Ausgang	AI1			AI2			AO1			AO2		
Klemmenbezeichnung	AI11	AI12	AGND	AI21	AI22	AGND	AOV1	AOC1	AGND	AOV2	AOC2	AGND

Die analogen Eingänge sind Differenzeingänge. Die Ein-/Ausgänge können wahlweise als U- oder I-Ein-/Ausgänge genutzt werden.

Tabelle 7: Zuordnung von Wertebereichen zu Variablenwerten

Ein- / Ausgang	Wertebereich	Variablenwert
Ausgang	- 10 ... 0 ... + 10 V	- 10 000 ... 0 ... + 10 000
	0 ... + 10 V	0 ... + 10 000
	0 ... + 20 mA	0 ... + 10 000
	4 ... + 20 mA	0 ... + 10 000
Eingang	- 10 ... 0 ... + 10 V	- 10 000 ... 0 ... + 10 000
	0 ... + 10 V	0 ... + 10 000
	0 ... + 20 mA	0 ... + 5 000
	4 ... + 20 mA	1000 ... + 5 000

Die Zuordnung Wertebereich / Variablenwert bei den Analogausgängen gilt nur, wenn die Skalierung des Parameters P110 = 1 beträgt.

22.3.1 Lesen analoger Ein-/Ausgänge

Der Zustand der analogen Ein-/Ausgänge des Grundgeräts und der Ein-/Ausgabekarte DIO11 kann mit dem Befehl **GETSYS** in frei wählbare Variablen geschrieben werden. Dabei wird in den GETSYS-Befehl zuerst die Variable und dann die Systemgröße eingetragen (hier: **ANALOG INPUTS** oder **ANALOG OUTPUTS**).

Der erste Eingang / Ausgang wird auf die im GETSYS-Befehl eingetragene Variable geschrieben (Hxxx), der zweite auf die darauf folgende Variable (Hxxx + 1).

Beispiel

Befehl: **GETSYS H310 = ANALOG INPUTS**

H310 enthält den Wert des analogen Eingangs AI1

H311 enthält den Wert des analogen Eingangs AI2

22.3.2 Schreiben analoger Ausgänge

HINWEIS

Voraussetzung für das Schreiben ist die Parametrierung des betreffenden Analogausgangs (P640 und P643) auf "IPOS-AUSGABE".

Das Schreiben der analogen Ausgänge erfolgt mit der Systemvariablen **H479 (ANA.OUT IP)** und **H478 (ANA.OUT IP2)**.

Befehl: **SET H479 = K** (K = beliebige Konstante innerhalb des o.g. Wertebereichs)

SET H479 beschreibt Analogausgang 1

SET H478 beschreibt Analogausgang 2

23 Assembler – Befehle

23.1 Allgemeine Hinweise

- Das Ergebnis der Rechenoperation wird immer dem linken Argument (immer Variable) zugewiesen. Das zweite Argument (Variable oder Konstante) bleibt immer unverändert. Das Ergebnis einer mathematischen Operation ist immer ganzzahlig.
- Die Bitstellen in den Variablen und Konstanten besitzen die Nummern 0 ... 31. Das niederwertigste Bit hat die Nummer 0.

23.2 Befehlsübersicht

23.2.1 Arithmetische Befehle

In dieser Programmgruppe sind alle arithmetischen und logischen Befehle aufgeführt.

Befehl	Argumente	Beschreibung	Siehe
ADD	H + H H + K	Arithmetische Addition.	ADD (Seite 302)
AND	H & H H & K	Logisches UND.	AND (Seite 304)
ASHR ARITHMETIC SHIFT RIGHT	H = H (Arithmetic > >) H H = H (Arithmetic > >) K	Arithmetisches Rechts-Schieben.	ASHR (Seite 306)
DIV DIVISION	H / H H / K	Division.	DIV (Seite 303)
MOD MODULO	H mod H H mod K	Modulo / Divisions-Rest.	MOD (Seite 304)
MUL MULTIPLY	H * H H * K	Multiplikation.	MUL (Seite 303)
NOT	H = NOT (H)	Bitweise Negation.	NOT (Seite 303)
OR	H ¦ H H ¦ K	Logisches ODER.	OR (Seite 304)
SHL SHIFT LEFT	H = H << K H = H << H	Bitweises Links-Schieben.	SHL (Seite 305)
SHR SHIFT RIGHT	H = H > > H H = H > > K	Bitweises Rechts-Schieben.	SHR (Seite 306)
SUB SUBTRACT	H - H H - K	Arithmetisches Subtrahieren.	SUB (Seite 302)
XOR EXCLUSIVE OR	H XOR H H XOR K	Exklusives ODER.	XOR (Seite 305)

23.2.2 Bit-Befehle

Befehle zum Ändern von einzelnen Bits innerhalb einer Variablen. Diese sind:

- Setzen/Löschen/Verschieben von Bits

Befehl	Argumente	Beschreibung	Siehe
BCLR BIT CLEAR	H.Bit = 0	Bit löschen.	BCLR (Seite 307)
BMOV BIT MOVE	H.Bit = H.Bit	Bit kopieren.	BMOV (Seite 307)
BMOVN BIT MOVE NEGATE	H.Bit = NOT (H.Bit)	Bit negiert kopieren.	BMOVN (Seite 308)
BSET BIT SET	H.Bit = 1	Bit setzen.	BSET (Seite 307)

23.2.3 Kommunikationsbefehle

Befehle zum Datenaustausch von/zu anderen Geräten über Schnittstellen.

Befehl	Funktion	Verfügbarkeit[1)]			Verweis
		MDX B	MC07B	MQx	
MOVLNK	Azyklischer Prozess- und/oder Parameter-Datenaustausch über RS-485 und Systembus.	X	X	X	MOVLNK (Seite 309)
MOVCOM	Zyklische Prozessdatenübertragung über RS-485 mit MQx für MOVIMOT®.	-	-	X	MOVCOM (Seite 314)
MOVON	Start der zyklischen Prozessdatenübertragung über RS-485.	-	-	X	MOVON (Seite 315)
SCOM	Zyklischer oder azyklischer Prozessdatenaustausch über Systembus.	X	X	-	SCOM (Seite 316)
SCOMON	Start des zyklischen Prozessdatenaustauschs über Systembus.	X	X	-	SCOMON (Seite 321)
SCOMST	Start der zyklischen Übertragung bei MOVIDRIVE® B.	X	(X)	-	SCOMST (Seite 322)

1) Beachten Sie die gerätespezifische Befehlsstruktur

23.2.4 Positionierbefehle

Befehle zur Positionierung des Antriebs:

- Referenzfahrt
- Positioniere absolut/relativ / auf Touch-Probe

Befehl	Beschreibung	Siehe
GO0 GO POSITION 0	Führt eine Referenzfahrt durch.	GO0 (Seite 323)
GOA GO ABSOLUTE	Positioniere absolut, Variable. Positioniere absolut, Konstante. Positioniere absolut, Variable, indirekt.	GOA (Seite 325)
GOR GO RELATIVE	Positioniere relativ, Variable. Positioniere relativ, Konstante. Positioniere relativ, Variable, indirekt.	GOR (Seite 325)

23.2.5 Programmbefehle

Befehle zur Programmsteuerung. Diese sind:

- Schleifenbefehle
- Unterprogrammaufrufe
- Steuerung von Task 2
- Programmverzweigungsbefehle
- Wartebefehle

Befehl	Beschreibung	Siehe
CALL	Ruft ein Unterprogramm auf.	CALL (Seite 329)
END	Textuelles Ende.	END (Seite 329)
JMP JUMP	Sprung, Eingangsklemmen. Sprung, H <=> 0. Sprung, H <=> H. Sprung, H <=> K. Systembedingter Sprung.	JMP (Seite 330)
LOOPB LOOP BEGIN	Programmschleife, Beginn.	LOOPB (Seite 332)
LOOPE LOOP END	Programmschleife, Ende.	LOOPE (Seite 332)
NOP NO OPERATION	Keine Operation.	NOP (Seite 333)
REM REMARK	Kommentar.	REM (Seite 333)
RET RETURN	Ende eines Unterprogramms.	RET (Seite 333)
TASK2	Setzt die Startadresse von Task 2.	TASK2 (Seite 334)
WAIT	Wartet eine angegebene Zeit.	WAIT (Seite 335)

23.2.6 Setzbefehle

Befehle zum

- Setzen von Variablen
- Fehlerreaktionen
- Laden von Systemgrößen in Variablen
- Schreiben von Systemgrößen in Systemvariablen
- Initialisierung der Interrupt-Routinen

Befehl	Beschreibung / Argumente	Siehe
COPY	Blockweises Kopieren von Variablen.	COPY (Seite 336)
GETSYS GET SYSTEM VALUE	H = Systemgröße	GETSYS (Seite 336)
SET	H = H H = K	SET (Seite 339)
SETFR SET FAULT REACTION	Fehlerreaktion setzen.	SETFR (Seite 339)
SETI SET INDIRECT	[H] = H H = [H]	SETI (Seite 339)
SETINT SET INTERRUPT	Setzt Anfangsadresse der Interrupt-Routine.	SETINT (Seite 339)
SETSYS SET SYSTEM VALUE	Systemgröße = H	SETSYS (Seite 344)
VARINT	Setzt Anfangsadresse und Datenstruktur für Variablen-Interrupt.	VARINT (Seite 347)

23.2.7 Spezielle Gerätebefehle

Befehle zum

- Stoppen der Achse
- Speichern von Variablen und Programmen auf nichtflüchtigen Speicher im Gerät
- Touch-Probe ein-/ausschalten
- Steuern des Watchdog

Befehl	Beschreibung	Siehe
ASTOP AXIS STOP	Achse anhalten.	ASTOP (Seite 349)
MEM MEMORIZE	Speichern und Laden von IPOS$^{plus®}$-Programm und Variablen.	MEM (Seite 349)
TOUCHP TOUCH PROBE	Touchprobe-Befehl.	TOUCHP (Seite 350)
WDOFF WATCHDOG OFF	Watchdog ausschalten.	WDOFF (Seite 353)
WDON WATCHDOG ON	Watchdog in Zeitintervallen aufrufen.	WDON (Seite 353)

23.2.8 Vergleichsbefehle

Befehle zum Vergleich von Variablen und Konstanten.

Befehl	Argumente	Siehe
ANDL LOGICAL AND	H = H && H	ANDL (Seite 358)
CPEQ COMPARE EQUAL	H = H == H H = H == K	CPEQ (Seite 355)
CPGE COMPARE GREA-TER OR EQUAL	H = H > = K H = H > = H	CPGE (Seite 355)
CPGT COMPARE GREA-TER THAN	H = H > H H = H > K	CPGT (Seite 356)
CPLE COMPARE LESS OR EQUAL	H = H <= H H = H <= K	CPLE (Seite 356)
CPLT COMPARE LESS THAN	H = H < H H = H < K	CPLT (Seite 357)
CPNE COMPARE NOT EQUAL	H = H != H H = H != K	CPNE (Seite 357)
NOTL LOGICAL NOT	H = NOT(H)	NOTL (Seite 359)
ORL LOGICAL OR	H = H ¦¦ H	ORL (Seite 358)

23.3 Arithmetische Befehle

23.3.1 Grundrechenarten ADD / SUB / MUL / DIV

Die vier Grundrechenarten werden unter Berücksichtigung der Vorzeichen ausgeführt und können mit Variablen H und Konstanten K ausgeführt werden. Das 1. Argument ist immer eine Variable H, als 2. Argument dient entweder eine zweite Variable H oder eine Konstante K.

ADD

Der Befehl ADD addiert vorzeichenrichtig eine Variable mit einer Konstanten und einer Variablen.

Befehlsaufbau Mxxx ADD X1 + X2	Mxxx: Marke (optional) X1: Variable (Summand und Summe) X2: Variable oder Konstante (Summand)
ADD HXX + HYY	Die Variable HXX ist die Summe von Variable HXX und HYY.
ADD HXX + K	Die Variable HXX ist die Summe von Variable HXX und einer Konstanten K.
Beispiel 1	`SET H01 = 100ADD H01 + H01` Nach dem ADD-Befehl ist H01 = 200.
Beispiel 2	`SET H01 = 100ADD H01 + 1` Nach dem ADD-Befehl ist H01 = 101.

Beispiel 3	`SET H01 = 2000000000SET H02 = 2000000000ADD H01 + H02`
	`[0x77359400][0x77359400][0xEE6B2800]`
	Der Zahlenbereich wird überschritten. Nach der Addition hat H01 den Wert -294967296.

Hinweis	Wird bei der Addition der Zahlenbereich verlassen, so ist das Ergebnis falsch. Es erfolgt keine Fehlermeldung.

SUB / SUBTRACT

Der Befehl SUB subtrahiert vorzeichenrichtig eine Variable oder Konstante von einer Variablen.

Befehlsaufbau Mxxx SUB X1 - X2	Mxxx: Marke (optional) X1: Variable (Minuend und Differenz) X2: Variable oder Konstante (Subtrahend)
SUB HXX - HYY	Die Variable HXX ist die Subtraktion von Variable HXX und HYY.
SUB HXX - K	Die Variable HXX ist die Subtraktion von Variable HXX und einer Konstanten K.
Beispiel 1	`SET H01 = -10SET H02 = +50SUB H01 - H02` Nach dem SUB-Befehl ist H01 = -60.

Beispiel 2	`SET H01 = +50SET H02 = -2147483600SUB H01 - H02`
	`[0x00000032][0x80000030][0x80000002]`
	Der Zahlenbereich wird überschritten. Nach der Subtraktion hat H01 den Wert -2147483646.

Hinweis	Wird bei der Subtraktion der Zahlenbereich verlassen, so ist das Ergebnis falsch. Es erfolgt keine Fehlermeldung.

MUL / MULTIPLY

Der Befehl MUL multipliziert vorzeichenrichtig eine Variable mit einer Variablen oder Konstanten.

Befehlsaufbau Mxxx MUL X1 * X2	Mxxx: Marke (optional) X1: Variable (Faktor und Produkt) X2: Variable oder Konstante (Faktor)
MUL HXX * HYY	Die Variable HXX ist die Multiplikation von Variable HXX und HYY.
MUL HXX * K	Die Variable HXX ist die Multiplikation von Variable HXX und einer Konstanten K.
Beispiel 1	`SET H01 = -3MUL H01 * 50` Nach dem MUL-Befehl ist H01 = -150.
Beispiel 2	`SET H01 = +50000SET H02 = +50000MUL H01 * H02` `[0x0000C350][0x0000C350][0x9502F900]` Der Zahlenbereich wird überschritten. Nach der Multiplikation hat H01 den Wert -1794967296.
Hinweis	Wird bei der Multiplikation der Zahlenbereich verlassen, so ist das Ergebnis falsch. Es erfolgt keine Fehlermeldung.

DIV / DIVISION

Der Befehl DIV dividiert vorzeichenrichtig eine Variable durch eine Variable oder Konstante. Das Ergebnis ist die Vorkommazahl des Quotienten.

Befehlsaufbau Mxxx DIV X1 / X2	Mxxx: Marke (optional) X1: Variable (Dividend und Quotient) X2: Variable oder Konstante (Divisor)
DIV HXX / HYY	Die Variable HXX ist die Division von Variable HXX und HYY.
DIV HXX / K	Die Variable HXX ist die Division von Variable HXX und einer Konstanten K.
Beispiel	`SET H01 = -13SET H02 = +3DIV H01 / H02` Nach dem DIV-Befehl ist H01 = -4.
Hinweis	Eine Division durch Null liefert ein undefiniertes Ergebnis. Es erfolgt keine Fehlermeldung.

23.3.2 Arithmetische Hilfsoperationen NOT / MOD

NOT

Der Befehl negiert bitweise den gesamten Inhalt einer Variablen.

Befehlsaufbau Mxxx NOT X1 = NOT (X2)	Mxxx: Marke (optional) X1: Variable (Ergebnis der Operation) X2: Variable (Ausgangswert)
NOT HXX = NOT (HYY)	Die Variable HXX ist die bitweise Negierung der Variablen HYY. Damit ist die hexadezimale Summe von HXX und HYY = 0xFFFFFFFF.
Beispiel	`SET H02 = +1NOT H01 = NOT (H02)` `[0x00000001][0xFFFFFFFE]` Nach dem NOT-Befehl ist H01 = -2.

MOD / MODULO

Der Befehl liefert den ganzzahligen Rest der Division einer Variablen durch eine Variable oder Konstante. Das Vorzeichen des Ergebnisses ist gleich dem Vorzeichen der ersten Variablen.

Befehlsaufbau Mxxx MOD X1 mod X2	Mxxx: Marke (optional) X1: Variable (Dividend und Rest der Division) X2: Variable oder Konstante (Divisor)
MOD HXX mod HYY	Die Variable HXX ist der ganzzahlige Rest der Division von Variable HXX und HYY.
MOD HXX mod K	Die Variable HXX ist der ganzzahlige Rest der Division von Variable HXX und einer Konstanten K.

Beispiel 1	`SET H01 = -17SET H02 = -5MOD H01 mod H02`
	`SET H01 = -17SET H02 = +5MOD H01 mod H02`
	Nach dem MOD-Befehl ist H01 = -2.

Beispiel 2	`SET H01 = +17SET H02 = +5MOD H01 mod H02`
	`SET H01 = +17SET H02 = -5MOD H01 mod H02`
	Nach dem MOD-Befehl ist H01 = +2.

23.3.3 Logische Verknüpfungen AND / OR / XOR

AND

Der Befehl AND führt eine bitweise UND-Verknüpfung einer Variablen mit einer Variablen oder einer hexadezimalen Konstanten aus.

Befehlsaufbau Mxxx AND X1 & X2	Mxxx: Marke (optional) X1: Variable (Ergebnis und Ausgangswert) X2: Variable oder Konstante (Ausgangswert)
AND HXX & HYY	Die Variable HXX ist die bitweise UND-Verknüpfung von Variable HXX und HYY.
AND HXX & K	Die Variable HXX ist die bitweise UND-Verknüpfung von Variable HXX und einer Konstanten K.

Beispiel 1	`SET H01 = 12SET H02 = 5AND H01 & H02`
	`[0b0000000000001100][0b0000000000000101][0b0000000000000100]`
	Nach dem AND-Befehl ist H01 = 4.

Beispiel 2	Aus der Position des Motorgebers soll die Position innerhalb einer Motorumdrehung ermittelt werden. `SET H01 = H511AND H01 & 0xFFF` Nach dem AND-Befehl hat H01 einen Wert zwischen 0 und 4095.

OR

Der Befehl OR führt eine bitweise ODER-Verknüpfung einer Variablen mit einer Variablen oder einer hexadezimalen Konstanten aus.

Befehlsaufbau Mxxx OR X1 ¦ X2	Mxxx: Marke (optional) X1: Variable (Ergebnis und Ausgangswert) X2: Variable oder Konstante (Ausgangswert)
OR HXX ¦ HYY	Die Variable HXX ist die bitweise ODER-Verknüpfung von Variable HXX und HYY.
OR HXX ¦ K	Die Variable HXX ist die bitweise ODER-Verknüpfung von Variable HXX und einer Konstanten K.

Beispiel	`SET H01 = 12SET H02 = 1OR H01 ¦ H02`
	`[0b0000000000001100][0b0000000000000001][0b0000000000001101]`
	Nach dem OR-Befehl ist H01 = 13.

XOR

Der Befehl XOR führt eine bitweise XOR-Verknüpfung einer Variablen mit einer Variablen oder einer hexadezimalen Konstanten aus.

Befehlsaufbau Mxxx XOR X1 XOR X2	Mxxx: Marke (optional) X1: Variable (Ergebnis und Ausgangswert) X2: Variable oder Konstante (Ausgangswert)
XOR HXX XOR HYY	Die Variable HXX ist die bitweise XOR-Verknüpfung von Variable HXX und HYY.
XOR HXX XOR K	Die Variable HXX ist die bitweise XOR-Verknüpfung von Variable HXX und einer Konstanten K.
Beispiel	`SET H01 = 65535XOR H01 XOR [0x00000FFF][0x0000FF0F] F0F0 hex` Nach dem XOR-Befehl ist H01 = 0xFF0F.

23.3.4 SHIFT-Befehle SHL / SHR / ASHR

SHIFT-Befehle werden verwendet, um den Inhalt einer Variablen bitweise zu verschieben. Alle Bits der Variablen erhalten eine neue Wertigkeit. Die Anzahl der zu verschiebenden Stellen wird im 2. Argument angegeben.

SHL / SHIFT LEFT

Der Befehl SHL schiebt den Inhalt einer Variablen um die Anzahl von Bits nach links, die in einer Variablen oder Konstanten angegeben sind. Von rechts werden Nullen nachgeschoben.

Befehlsaufbau Mxxx SHL X1 << X2	Mxxx: Marke (optional) X1: Variable (Ergebnis und Ausgangswert) X2: Variable oder Konstante (Anzahl der Schiebeoperationen)
SHL HXX << HYY	In der Variablen HXX sind die Bits um HYY Stellen nach links verschoben.
SHL HXX << K	In der Variablen HXX sind die Bits um K Stellen nach links verschoben.
Beispiel 1	`SET H01 = 31SET H02 = 1SHL H01 << H02 [0b0000000000011111][0b0000000000111110]` Nach dem SHL-Befehl ist H01 = 62.
Beispiel 2	Den Ausgangsklemmen von Grundgerät und Option DIO11A ist eine bestimmte binäre Wertigkeit zugeordnet. Um z. B. die Ausgänge DO10 ... DO13 für eine Tabellenpositionierung sinnvoll nutzen zu können (4 Eingänge = 0 ... 15 Positionen), ist die Wertigkeit der Ausgänge so zu verschieben, dass die niederwertigste Klemme DO10 die Wertigkeit 2^0 erhält. `SET H01 = 15SET H02 = 6SHL H01 << H02 [0b0000000000001111][0b0000001111000000]`

SHR / SHIFT RIGHT

Der Befehl SHR schiebt den Inhalt einer Variablen um die Anzahl der Bits nach rechts, die in einer Variablen oder Konstanten angegeben sind. Von links werden Nullen nachgeschoben.

Befehlsaufbau Mxxx SHR X1 > > X2	Mxxx: Marke (optional) X1: Variable (Ergebnis und Ausgangswert) X2: Variable oder Konstante (Anzahl der Schiebeoperationen)
SHR HXX > > HYY	In der Variablen HXX sind die Bits um HYY Stellen nach rechts verschoben.
SHR HXX > > K	In der Variablen HXX sind die Bits um K Stellen nach rechts verschoben.
Beispiel 1	`SET H01 = 62SET H02 = 1SHR H01 > > H02` `[0b0000000000111110][0b0000000000011111]` Nach dem SHR-Befehl ist H01 = 31.
Beispiel 2	Den Eingangsklemmen von Grundgerät und Option DIO11A ist eine bestimmte binäre Wertigkeit zugeordnet. Um z. B. die Eingänge DI10 ... DI13 für eine Tabellenpositionierung sinnvoll nutzen zu können (4 Eingänge = 0 ... 15 Positionen), ist die Wertigkeit der Eingänge so zu verschieben, dass die niederwertigste Klemme DI10 die Wertigkeit 2^0 erhält. `SET H01 = 960SET H02 = 6SHR H01 > > H02` `[0b0000001111000000][0b0000000000001111]`

ASHR / ARITHMETIC SHIFT RIGHT

Der Befehl ASHR schiebt den Inhalt einer Variablen um die Anzahl der Bits nach rechts, die in einer Variablen oder Konstanten angegeben sind. Abhängig vom Vorzeichen des ursprünglichen Wertes werden von links Nullen oder Einsen nachgeschoben. Somit bleibt ein negatives Vorzeichen bei der Schiebeoperation erhalten.

Für positive Zahlen liefert der Befehl damit die Vorkommazahl der Division X1/X2. Für negative Zahlen liefert der Befehl die Vorkommazahl der Division X1/X2 - 1.

Befehlsaufbau Mxxx ASHR X1 > > X2	Mxxx: Marke (optional) X1: Variable (Ergebnis und Ausgangswert) X2: Variable oder Konstante (Anzahl der Schiebeoperationen)
ASHR HXX > > HYY	In der Variablen HXX sind die Bits um HYY Stellen nach rechts verschoben.
ASHR HXX > > K	In der Variablen HXX sind die Bits um K Stellen nach rechts verschoben.
Beispiel 1	`SET H01 = 7ASHR H01 > > 2` `[0b0000000000000111][0b0000000000000001]` Nach dem SHR-Befehl ist H01 = 1.
Beispiel 2	`SET H01 = -7ASHR H01 > > 2` `[0b1111111111111001][0b1111111111111110]` Nach dem ASHR-Befehl ist H01 = -2.

23.4 Bitbefehle

23.4.1 Bitbefehle BSET / BCLR / BMOV / BMOVN

BSET / BIT SET

Der Befehl BSET setzt ein Bit innerhalb einer Variablen auf 1. Die Bitstellen in der Variablen besitzen die Nummern 0 ... 31. Das niederwertigste Bit hat die Nummer 0.

Wird zum Beispiel in der Systemvariablen H481 STD.OUT IP ein Bit gesetzt, so kann damit direkt ein binärer Ausgang gesetzt werden. Der Ausgang ist dazu vorher mit SHELL mit Parameter P62x als IPOS-AUSGANG zu konfigurieren.

Befehlsaufbau Mxxx BSET HX1.X2 = 1	Mxxx: Marke (optional) X1: Zielvariable X2: Bitstelle innerhalb der Zielvariablen
BSET HXX.YY = 1	In der Variablen HXX wird das Bit YY auf 1 gesetzt.
Beispiel	SHELL: P621 = IPOS-AUSGANG `BSET H481.2 = 1` Nach dem BSET-Befehl ist das 3. Bit in Variable H481 und Ausgang DO02 gesetzt.
Hinweis	Ist der Ausgang für eine andere Funktion reserviert (z. B. P621 = MOTORSTILLSTAND), dann wird zwar das Bit in H481 gesetzt, aber nicht der binäre Ausgang.

BCLR / BIT CLEAR

Der Befehl BCLEAR setzt ein Bit innerhalb einer Variablen auf 0. Die Bitstellen in der Variablen besitzen die Nummern 0 ... 31. Das niederwertigste Bit hat die Nummer 0.

Wird zum Beispiel in der Systemvariablen H481 STD.OUT IP ein Bit gelöscht, so kann damit direkt ein binärer Ausgang rückgesetzt werden. Der Ausgang ist dazu vorher mit SHELL mit Parameter P62x als IPOS-AUSGANG zu konfigurieren.

Befehlsaufbau Mxxx BCLEAR HX1.X2 = 1	Mxxx: Marke (optional) X1: Zielvariable X2: Bitstelle innerhalb der Zielvariablen
BCLEAR HXX.YY = 0	In der Variablen HXX wird das Bit YY auf 0 gesetzt.
Beispiel	SHELL: P621 = IPOS-AUSGANG `BCLEAR H481.2 = 0` Nach dem BCLEAR-Befehl ist das 3. Bit in Variable H481 und Ausgang DO02 gelöscht.
Hinweis	Ist der Ausgang für eine andere Funktion reserviert (z. B. P621 = MOTORSTILLSTAND), dann wird zwar das Bit in H481 gelöscht, aber nicht der binäre Ausgang.

BMOV / BIT MOVE

Der Befehl BMOV kopiert ein Bit von einer Variablen in ein Bit von einer anderen Variablen. Die Bitstellen einer Variablen besitzen die Nummern 0 ... 31. Das niederwertigste Bit hat die Nummer 0.

Befehlsaufbau Mxxx BMOV HX1.X2 = HX3.X4	Mxxx: Marke (optional) X1: Zielvariable X2: Bitstelle innerhalb der Zielvariablen X3: Quellvariable X4: Bitstelle innerhalb der Quellvariablen
BMOV HXX.YY = HZZ.AA	In der Variablen HXX wird das Bit YY auf den Wert des Bits AA der Variablen HZZ gesetzt.
Beispiel 1	`BMOV H2.4 = H7.5` Der Befehl kopiert das Bit 5 der Variable H7 in das Bit 4 der Variable H2.
Beispiel 2	`SET H200 = 0BMOV H200.0 = H473.20JMP H200 == 1 M01` Der Sprung zu der Marke M01 wird ausgeführt, wenn der Antrieb referenziert ist (H473 STAT. WORD).

BMOVN / BIT MOVE NEGATE

Der Befehl BMOVN kopiert ein Bit von einer Variablen in ein Bit von einer anderen Variablen und negiert es dabei. Die Bitstellen einer Variablen besitzen die Nummern 0 ... 31. Das niederwertigste Bit hat die Nummer 0.

Befehlsaufbau Mxxx BMOVN HX1.X2 = HX3.X4	Mxxx: Marke (optional) X1: Zielvariable X2: Bitstelle innerhalb der Zielvariablen X3: Quellvariable X4: Bitstelle innerhalb der Quellvariablen
BMOVN HXX.YY = HZZ.AA	In der Variablen HXX wird das Bit YY auf den negierten Wert des Bits AA der Variablen HZZ gesetzt.
Beispiel 1	`BMOVN H2.4 = H7.5` Der Befehl kopiert das negierte Bit 5 der Variable H7 in das Bit 4 der Variable H2.
Beispiel 1	`SET H200 = 0BMOV H200.0 = H473.20JMP H200 == 1 M01` Der Sprung zu der Marke M01 wird ausgeführt, wenn der Antrieb noch keine Referenzfahrt durchgeführt hat (H473 STAT. WORD).

23.5 Kommunikationsbefehle

23.5.1 MOVLNK

Beschreibung siehe _MoviLink (Seite 222)

Befehlsstruktur

Anweisung	Standardstruktur	Elemente	Kurzbeschreibung
_MoviLink	MOVLNK	BusType (H+0)	mögliche Bustypen: ML_BT_S0 = 1 (RS485 #1) ML_BT_S1 = 2 (RS485 #2) ML_BT_SBUS1 = ML_BT_SBUS = 5 ML_BT_SBUS2 = 8
		Address (H+1)	0...99: Einzeladresse 100...199: Gruppenadresse 253: Eigene Adresse des Umrichters 254: Punkt-zu-Punkt-Verbindung 255: Broadcast Wird eine SBus-Gruppenadresse (z. B. 43) angesprochen, so muss der Offset 100 hinzuaddiert werden. In diesem Fall ergibt das 143.

		Format (H+2)	Angabe der Prozess (PD)- und Parameterkanäle (PARAM) zur Datenübertragung:
			//MoviLink Cyclic Frame Types
			ML_FT_PAR1 = 0: PARAM+1PD
			ML_FT_1 = 1: 1PD
			ML_FT_PAR2 = 2: PARAM+2PD
			ML_FT_2 = 3: 2PD
			ML_FT_PAR3 = 4: PARAM+3PD
			ML_FT_3 = 5: 3PD
			ML_FT_PAR = 6: Parameter (ohne PD)
			//Acyclic
			ML_CFT_PAR1 = 128
			ML_CFT_1 = 129
			ML_CFT_PAR2 = 130
			ML_CFT_2 = 131
			ML_CFT_PAR3 = 132
			ML_CFT_3 = 133
			ML_CFT_PAR = 134

		Service (H+3)	Kommunikationsdienst für Parameter ML_S_RD = 1: Lese-Dienst ML_S_WR = 2: Schreiben auf nichtflüchtigen Speicher ML_S_WRV = 3: Schreiben ohne Speichern
		Index (H+4)	Index-Nummer des Parameters, der geändert oder gelesen werden soll (siehe Parameter-Verzeichnis) Der Subindex muss im Index-Element auf den Bits 23-16 eingetragen werden (niederwertigstes Byte des High-Words). Berechnung: H+4 oder `MOVILNK.Index = Index + (SubIndex << 16);`
		DPointer (H+5)	Nummer der Variablen, ab der die gelesenen Daten abgelegt bzw. die zu schreibenden Daten geholt werden (Struktur MLDATA)
		Result (H+6)	Beinhaltet den Fehlercode nach Ausführung des Dienstes bzw. Null, wenn kein Fehler vorhanden (siehe "Rückkehr-Codes der Parametrierung" im Handbuch "Kommunikation und Feldbus-Geräteprofil" mit Parameter-Verzeichnis").
	MLDATA	WritePar (H"+0)	Parameter, der bei Write-Diensten verschickt wird
		ReadPar (H"+1)	Parameter, der bei Read-Diensten empfangen wird
		PO1 (H"+2)	Prozessausgangsdatum 1
		PO2 (H"+3)	Prozessausgangsdatum 2
		PO3 (H"+4)	Prozessausgangsdatum 3
		PI1 (H"+5)	Prozesseingangsdatum 1
		PI2 (H"+6)	Prozesseingangsdatum 2
		PI3 (H"+7)	Prozesseingangsdatum 3

Die folgende Tabelle zeigt Ihnen diejenigen Elemente, die gerätespezifischen Besonderheiten aufweisen.

Element	Gerätespezifische Besonderheiten		
	MOVIDRIVE® B	**MOVITRAC® B**	**MQx**
BusType (H+0)	nur 2: (RS485 an X13) 5: (SBUS an X12) 8: (über DFC11B)	nur 2: (RS485 an FSC/FIO11B) 5: (SBUS an FSC/FIO21B)	nur 2: (RS485 zu MOVIMOT®)
Format (H+2)	keine Einschränkung	keine Einschränkung	nur 130: (Param + 2PD azyklisch) 131: (2PD azyklisch) 132: (Param + 3PD azyklisch) 133: (3PD azyklisch) 134: (Param azyklisch) Zyklische Frametyps sind möglich, es wird aber hierfür _MovCommDef empfohlen.

Befehlsaufbau

Befehlsaufbau Mxxx MOVLNK X1	Mxxx: Marke (optional) X1: Anfangsvariable der Befehlsstruktur
MOVLNK HXX	Der MOVLNK-Befehl wird mit den Befehlsstruktur-Daten ausgeführt, die in Variable HXX beginnen.

Parametereinstellungen beim Sender (Master)

Beschreibung siehe Parametereinstellungen beim Sender (Master) (Seite 225)

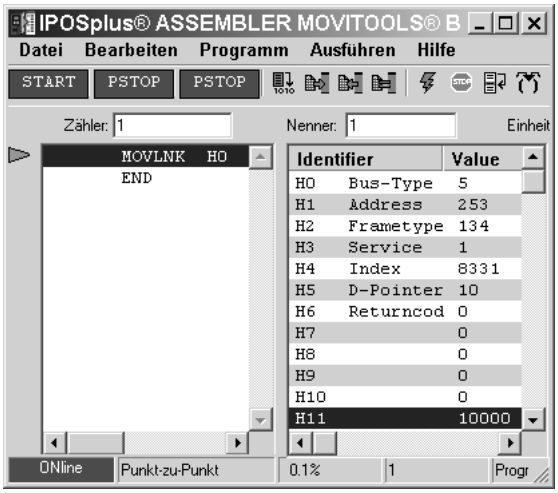
Parametereinstellungen beim Empfänger

Beschreibung siehe Parametereinstellungen beim Empfänger (Seite 226)

Beispiel 1

Lesen eines "geräteinternen" Parameters (Analogeingang AI1)

Der Anzeigeparameter P020 mit der Indexnummer 8331 wird mit nachfolgendem IPOS$^{plus®}$-Programm und Parametereinstellung gelesen und auf die Variable H011 geschrieben. Die Variablenstruktur wurde hier im Variablen-Editierfenster eingegeben. Es ist auch möglich die Variablenstruktur mit SET-Befehlen im Programm zu erstellen.



514449547

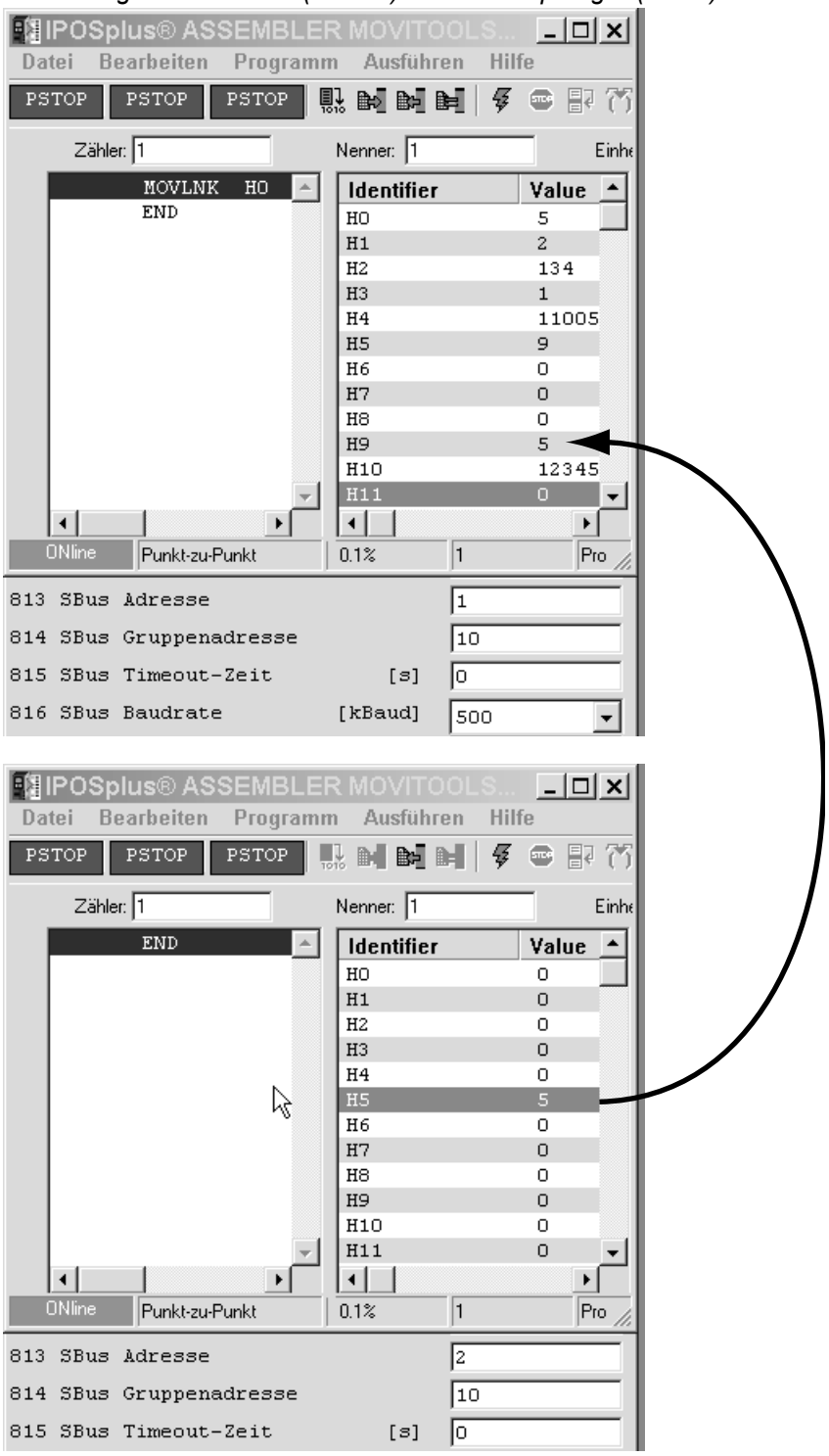
H0	Bus-Type	5	=	SBus (ohne Bedeutung)
H1	Address	253	=	eigene Adresse
H2	Frametype	134	=	nur Para
H3	Service	1	=	Lesen
H4	Index	8331	=	Index von P020
H5	D-Pointer	10	=	Datenpointer Wert liegt bei H10

Beispiel 2

Achs-zu-Achs-Kommunikation: Variablen eines anderen Umrichters über SBus lesen

Auf der Empfänger-Achse wird der Wert der Variablen H005 gelesen und im Sender auf die Variable H010 geschrieben. Hierzu sind 2 Umrichter über den SBus zu verbinden und die Abschlusswiderstände (über DIP-Schalter S12) zu aktivieren.

Einstellung oben: Sender (Master) / unten: Empfänger (Slave)



514453899

H0	Bus-Type	5	=	SBus
H1	Address	2	=	SBus-Adresse des Empfängers (Slave)
H2	Frametype	134	=	nur Para
H3	Service	1	=	Lesen
H4	Index	11005	=	Index von H5
H5	D-Pointer	9	=	Datenpointer Wert liegt bei H9

Beispiel 3

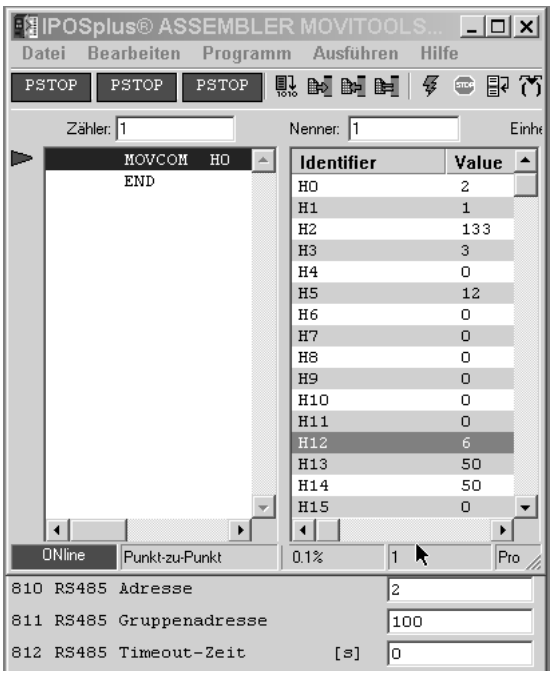
Ansteuern eines MOVIMOT® über RS-485 und 3PD

Das MOVIMOT® ist gemäß MOVIMOT®-Betriebsanleitung in Betrieb zu nehmen.

Eine Kommunikation mit MOVIMOT® ist nur über RS-485 möglich. Eine Ansteuerung kann nur über den Prozessdatenkanal mit 2PD oder 3PD (min. Steuerwort und Drehzahl) erfolgen.

Im nachfolgenden Beispiel wird das MOVIMOT® mit 3 Prozessausgangsdaten (Steuerwort 1, Drehzahl und Rampe) angesteuert.

Die Werte hierfür sind in den Variablen H012 ... H014 einzutragen.



514510219

H0 Bus-Type	2	=	RS-485
H1 Address	1	=	RS-485-Adresse des Empfängers (MOVIMOT®)
H2 Frametype	133	=	3PD
H3 Service	3	=	Schreiben ohne Speichern
H4 Index	0	=	ohne Bedeutung bei PD
H5 D-Pointer	12	=	Datenpointer Wert liegt bei H12
H12	6	=	PO1 Steuerwort
H13	50	=	PO2 Drehzahl in Prozent
H14	50	=	PO3 Rampe in Prozent

	HINWEIS Bei MOVIDRIVE® B und MOVITRAC® B wird für die Timeout-Überwachung überprüft, ob in der parametrierten Timeout-Zeit Telegramme empfangen werden. Bei MOVIMOT® wird die Timeout-Überwachung mit den ersten empfangenen zyklischen Frames (ML_FT... beim Sender) aktiviert. Azyklische Kommunikation schaltet beim MOVIMOT® die Timeout-Überwachung wieder aus. Nach Start der zyklischen Kommunikation mit _MovCommOn ist dann nur noch der _MoviLink-Befehl zur Adresse 253 (intern) möglich, mit dem _MoviLink-Befehl kann nicht mehr auf andere Geräte zugegriffen werden.

23.5.2 MOVCOM

Der Befehl kann nur bei MQx verwendet werden. Die MovComm-Befehle ermöglichen den zyklischen Datenaustausch zwischen MQX und bis zu 8 MOVIMOT® über die RS-485-Schnittstelle mit MOVILINK-Profil.

Detailierte Beschreibung siehe _MovCommDef (Seite 227).

Die Variable wird durch **MOVCOM Variablenname;** im Compiler definiert und hat folgende Struktur:

BusType (H+0)	**Bustyp (Schnittstelle)** ML_BT_S1 = 2 (RS485 zum MOVIMOT®)
Address (H+1)	**Einzeladresse oder Gruppenadresse des anzusprechenden MOVIMOT®** 0 ... 99 Einzeladressierung 100 ... 199 Gruppenadressierung 255 Broadcast
Format (H+2)	**Angabe der Prozessdaten zur Datenübertragung** 3 = 2 Prozessdatenworte zyklisch (für MOVIMOT®) = ML_FT_2 5 = 3 Prozessdatenworte zyklisch (für MOVIMOT®) = ML_FT_3
Pd Pointer (H+3)	**Nummer der Variablen H", auf der die Prozessdaten abgelegt bzw. die zu schreibenden Daten geholt werden**. (Die Datenstruktur für H" ist im Anschluss ausführlich beschrieben)
Para Pointer (H+4)	**Nummer der Variablen H*, auf der die Parameterdaten abgelegt bzw. die zu schreibenden Daten geholt werden**. Wird vom MOVIMOT® nicht unterstützt.

Variablenstruktur der Prozessdaten

Datenstruktur für H":	
H"+0	Beinhaltet den Fehlercode der Verbindung bzw. Null, wenn kein Fehler vorhanden ist. 0x05000002 zeigt einen Time-out der Verbindung an.
H"+1	PA1-Daten des Prozessdatenaustauschs
H"+2	PE1-Daten des Prozessdatenaustauschs
H"+3	PA2-Daten des Prozessdatenaustauschs
H"+4	PE2-Daten des Prozessdatenaustauschs
H"+5	PA3-Daten des Prozessdatenaustauschs
H"+6	PE3-Daten des Prozessdatenaustauschs

Die Prozessdaten sind nach MOVILINK kodiert.

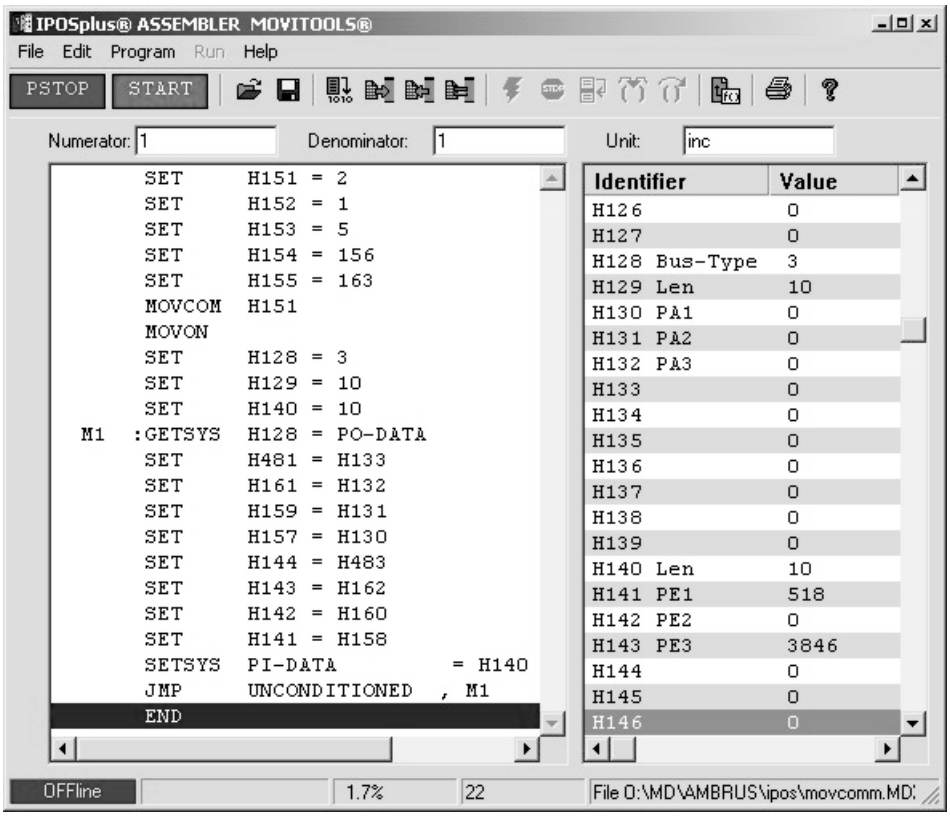
Variablenstruktur der Parameterdaten

H*+0	Beinhaltet den Fehlercode nach Ausführung des Parameterdienstes bzw. Null, wenn kein Fehler vorhanden ist. Die Fehler sind nach MOVILINK kodiert.
H*+1	0: keine Aktion oder Parameter-Datenaustausch abgeschlossen. 1: Start des Parameter-Datenaustausch
H*+2	1: Lese-Dienst 2: Schreiben mit Speichern auf nichtflüchtigen Speicher 3: Schreiben ohne Speichern
H*+3	Index-Nummer des Parameters, der geändert oder gelesen werden soll
H*+4	Gelesene Daten nach Read-Dienst. Zu schreibende Daten bei einem Write-Dienst.

Bei der Parametrierung ist folgendes Vorgehen einzuhalten:

1. Eintragen von Service, Index und Daten
2. Starten der Parametrierung mit einer 1 auf StartPar
3. Abwarten der Ausführung, Ende wird durch eine 0 auf StartPar angezeigt
4. Auswertung des ParaResult. Falls ein Fehler vorliegt, ist der Datenwert ungültig. Falls kein Fehler vorliegt, wurde der Dienst erfolgreich ausgeführt.

Programmbeispiel



514515083

23.5.3 MOVON

Der Befehl kann bei MOVIDRIVE® nicht verwendet werden.

Der Befehl startet die zyklische Kommunikation. Durch MovCommDef eingerichtete Kommunikationsbeziehungen werden aktiviert. Ab jetzt ist kein MovCommDef-Befehl mehr zulässig, ebenso wenig wie ein MOVILINK-Befehl. Nur der MOVILINK-Befehl auf Adresse 253 (intern) kann noch verwendet werden.

23.5.4 SCOM

Mit einem SCOM-Befehl (**SystembusCOM**munication) können bis zu 2 Variablen (8 Byte) über den Systembus übertragen werden. Der SCOM-Befehl initialisiert die Übertragungsobjekte und definiert, ob die Objekte azyklisch oder zyklisch gesendet oder Objekte empfangen werden. In den letzten beiden Fällen muss die Übertragung zusätzlich mit SCOMON oder SCOMSTATE gestartet werden.

Der Datenaustausch ist nur auf dem Systembus möglich und überträgt Inhalte von Variablen. Ein Datenaustausch innerhalb des Umrichters ist nicht möglich. Statt eines SEW-eigenen Protokolls (MOVILINK) wird ein Standard-CAN-Telegramm (11-Bit-Identifier) verwendet, so dass auch mit Fremdprodukten kommuniziert werden kann (siehe Handbuch "MOVIDRIVE® Serielle Kommunikation").

Entsprechend dem Consumer-Producer-Prinzip kann jedes Gerät gleichzeitig Objekte an ein oder mehrere Geräte senden und von einem oder mehreren Geräten empfangen.

Die Buslaufzeit einer Nachricht ist ≤ 2 ms und hängt von der eingestellten Baudrate ab. Eine Kommunikation mit MOVIMOT® oder Feldbus-Schnittstellen MQ ist nicht möglich.

Befehlsaufbau

Befehlsaufbau	Mxxx: Marke (optional)
Mxxx SCOM X1, X2	X1: TRANSMIT CYCLIC: zyklisches Senden RECEIVE: Empfangen TRANSMIT ACYCLIC: azyklisches Senden X2 Hxx = Beginn der Objektstruktur für die Kommunikations- und Nutzdaten

Der Aufbau der Objektstruktur ist abhängig von dem ersten Argument X1.

Beispiel

```
SCOM TRANSMIT CYCLIC, H0
```

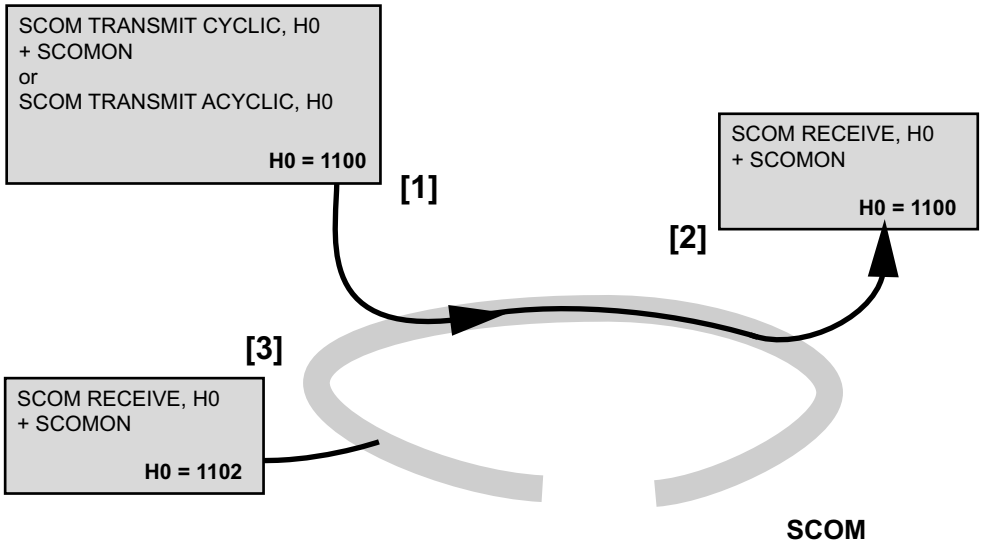
Dieser Befehl initialisiert eine zyklische Übertragung, die Objektstruktur beginnt bei H0.

```
SCOM TRANSMIT ACYCLIC, H10
```

Dieser Befehl initialisiert eine einmalig azyklische Übertragung, die Objektstruktur beginnt bei H10.

```
SCOM RECEIVE, H50
```

Dieser Befehl initialisiert den Empfang von Daten, die Objektstruktur beginnt bei H50.



514545547

Gerät [1] sendet zyklisch oder azyklisch das Objekt mit der Nr. 1100.

Gerät [2] empfängt die Daten. Gerät [3] ignoriert die Daten, wartet aber auf Daten mit der Objektnummer 1102.

Einstellungen bei Sender und Empfänger

Sender	Empfänger
IPOS$^{plus®}$-Programm mit Befehl: – SCOM TRANSMIT CYCLIC H SCOMON *und/ oder* – SCOM TRANSMIT ACYCLIC H • Einstellung der Kommunikationsparameter über Variablen	IPOS$^{plus®}$-Programm mit Befehl: – SCOM RECEIVE H SCOMON • Einstellung der Kommunikationsparameter über Variablen • Timeout-Überwachung P817
SBus-Baudrate (P816 / P884 / P894), bei Sender und Empfänger gleich.	
Physikalische erster und letzter Teilnehmer: Bus-Abschlusswiderstand mit S12 einstellen.	

HINWEIS

Bei der Wahl der Objektnummer sind die folgenden Regeln einzuhalten:

1. Im gesamten SBus-Netzwerk darf eine Objektnummer nur 1 Mal zum Senden eingerichtet sein.
2. Innerhalb eines Geräts darf eine Objektnummer nur 1 Mal eingerichtet werden, entweder 1 Mal zum Senden oder 1 Mal zum Empfangen.

Siehe dazu auch _SbusCommDef (Seite 227)

TRANSMIT CYCLIC

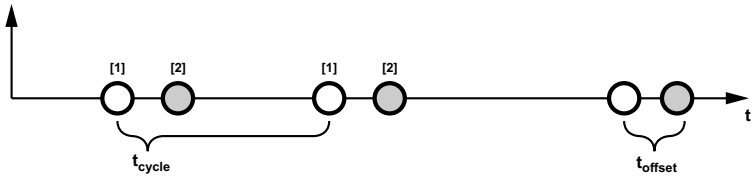
Dieses Argument initialisiert ein Datenobjekt, dessen Nutzdaten nach dem Befehl SCOMON zyklisch gesendet werden. Die Variable H des Befehls SCOM TRANSMIT CYCLIC H definiert den Anfang der Kommunikations- und Nutzdaten.

Nach einmaligem Start läuft die zyklische Datenübertragung im Hintergrund, unabhängig von der aktuellen Befehlsabarbeitung im IPOS$^{plus®}$-Programm. Wird das Programm gestoppt, so wird auch die Datenübertragung gestoppt. Eine Änderung des Datenobjektes wird nur nach einem IPOS$^{plus®}$-Programmneustart übernommen (F5 A/P-STOP / F9 P-Start, oder Netz (24-V-Stützbetrieb) aus- und dann wieder zuschalten)

Pro SCOM TRANSMIT... -Befehl kann maximal ein Datenobjekt eingerichtet werden. Sollen weitere Datenobjekte eingerichtet werden, so müssen weitere SCOM TRANSMIT-Befehle abgesetzt werden. Im Anschluss an mehrere SCOM TRANSMIT-Befehle ist nur ein SCOMON-Befehl erforderlich. Nach dem ersten SCOMON-Befehl werden keine weiteren SCOM TRANSMIT-Befehle mehr akzeptiert.

Die Anzahl der Objekte, die eingerichtet werden können, ist von der Zykluszeit abhängig (max. 5 Objekte mit 1...9 ms, max. 10 Objekte mit 10...65530 ms, somit insgesamt 15 Objekte).

Objektstruktur

H+0	**Objektnummer (CAN-Bus-ID):** die Objektnummer dient zur Adressierung des Datenobjektes. Für den Datenaustausch müssen die Objektnummern bei Sender (TRANSMIT) und Empfänger (RECEIVE) gleich sein.
H+1	**Zykluszeit** in ms, gibt das Zeitintervall an, nach dem die Daten erneut gesendet werden. Gültige Zykluszeiten: Siehe dazu _SbusCommDef (Seite 227) Achtung: Der Wert 0 ms führt zur Fehlermeldung im Return-Code. Die Zykluszeit muss immer größer sein als die größte Offsetzeit.
H+2	**Offset** in ms, dient zur Verteilung der Buslast, wenn mehrere SCOM TRANSMIT... -Befehle benutzt werden.  514549899 Gültige Offsetzeiten: Siehe dazu _SbusCommDef (Seite 227)
H+3	**Anzahl der Daten-Bytes und Datenformat**

Bit	Wert	Funktion
0...3	0...8	Anzahl der Daten-Bytes
4...7	0	reserviert
8	0...1	0 = MOTOROLA-Format 1 = INTEL-Format **Format muss bei Sender und Empfänger gleich sein!**
9...31	0	reserviert

H+4	**Nummer der Variablen H", ab der die zu sendenden Daten beginnen.**
H+5	**Ergebnis (Return Code) des SCOM-Befehls** ≥0 Freie Buskapazität in % (berechneter Wert dieses Geräts) -1 Falsche Zykluszeit -2 Zu viele Objekte eingerichtet -3 Busüberlastung -5 Falsche Objektnummer -6 Falsche Länge
	Insbesondere bei weiterem Datenaustausch zwischen den Slaves ist sicherzustellen, dass die gesamte rechnerische Busauslastung 70 % nicht überschreitet. Berechnet wird die Busauslastung in Bit pro Sekunde über die Formel: Anzahl Telegramme × Bit/Telegramm × 1/Zykluszeit z. B. 2 Telegramme mit 100 Bit im 1-ms-Zyklus = 200000 Bit/s = 200 kBaud Bezogen auf die gewählte Baudrate ergibt sich die prozentuale Buslast. z. B. 200 kBaud / 500 kBaud = 40 % < 70 %

TRANSMIT ACYCLIC

Dieses Argument initialisiert ein Datenobjekt, dessen Nutzdaten sofort einmalig gesendet werden. Die Variable H des Befehls **SCOM TRANSMIT ACYCLIC H** definiert den Anfang der Kommunikations- und Nutzdaten. Es ist kein SCOMON-Befehl notwendig.

Sollen mehrere Variablen gesendet werden, so ist dies mit dem SCOM TRANSMIT ACYCLIC H-Befehl möglich, indem vor jedem Aufruf des Befehls der Variablenzeiger (H+2) im IPOS$^{plus®}$-Programm entsprechend gesetzt wird.

Objektstruktur

H+0	**Objektnummer (CAN-Bus-ID:** die Objektnummer dient zur Adressierung des Datenobjektes. In einem Bussystem darf eine Objektnummer nur einmal vergeben werden. Für den Datenaustausch müssen die Objektnummern bei Sender (TRANSMIT) und Empfänger (RECEIVE) gleich sein. Um eine Datenkollision bei zusätzlicher Verwendung von MOVLNK-Befehlen über SBus zu vermeiden, sollten Objektnummern > 1024 ... 2048 eingestellt werden.
H+1	**Anzahl der Daten-Bytes und Datenformat**

Bit	Wert	Funktion
0...3	0...8	Anzahl der Daten-Bytes
4...7	0	reserviert
8	0...1	0 = MOTOROLA-Format 1 = INTEL-Format **Format muss bei Sender und Empfänger gleich sein!**
9...31	0	reserviert

H+2	Nummer der Variablen H", ab der die zu sendenden Daten beginnen.	
H+3	Zustand des Sendebefehls	
	0	Bereit
	1	Beim Senden
	2	Senden erfolgreich
	10	Sendefehler

HINWEIS

Vor dem Senden von azyklischen Telegrammen, muss ebenfalls der SBus mit SCOMON und SCOMSTATE aktiviert werden.

Das IPOS$^{plus®}$-Programm wartet an diesem Befehl so lange, bis das Telegramm gesendet wurde. Ist kein anderer Teilnehmer angeschlossen, dann kann das Telegramm nicht gesendet werden. Nur durch eine Überwachung z. B. aus der anderen Task, kann der Wartezustand beendet werden.

RECEIVE

Dieses Argument initialisiert ein Datenobjekt, das zyklisch oder azyklisch empfangene Daten enthält. Die Variable im Argument des SCOM RECEIVE-Befehls enthält die Variablennummer, ab der die Empfangsdaten abgelegt werden.

Das Dateneinlesen muss mit dem Befehl SCOMON gestartet werden. Nach einmaligem Start läuft das Dateneinlesen im Hintergrund, unabhängig von der aktuellen Befehlsabarbeitung im IPOS$^{plus®}$-Programm. Nach dem ersten SCOMON-Befehl werden keine weiteren SCOM RECEIVE-Befehle mehr akzeptiert. Eine Änderung des Datenobjektes wird nur nach einem IPOS$^{plus®}$-Programmneustart übernommen (F5 A/P-STOP / F9 P-Start, oder Netz (24-V-Stützbetrieb) aus- und dann wieder zuschalten).

Es können max. 32 Datenobjekte zum Dateneinlesen eingerichtet werden.

Objektstruktur

H+0	**Objektnummer:** die Objektnummer dient zur Adressierung des Datenobjektes. Für den Datenaustausch müssen die Objektnummern bei Sender (TRANSMIT) und Empfänger (RECEIVE) gleich sein.
H+1	**Anzahl der Daten-Bytes und Datenformat**

Bit	Wert	Funktion
0...3	0...8	Anzahl der Daten-Bytes
4...7	0	reserviert
8	0...1	0 = MOTOROLA-Format 1 = INTEL-Format **Format muss bei Sender und Empfänger gleich sein!**
9...31	0	reserviert

H+2	Nummer der Variablen H", ab der die empfangenen Daten abgelegt werden. Unterschiede des Nutzdatenformats zwischen MOTOROLA- und INTEL-Format:

	MOTOROLA-Format								INTEL-Format							
CAN Data Byte	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Variable	H"+1				H"				H"				H"+1			
Variablen Byte	3	2	1	0	3	2	1	0	0	1	2	3	0	1	2	3

HINWEIS

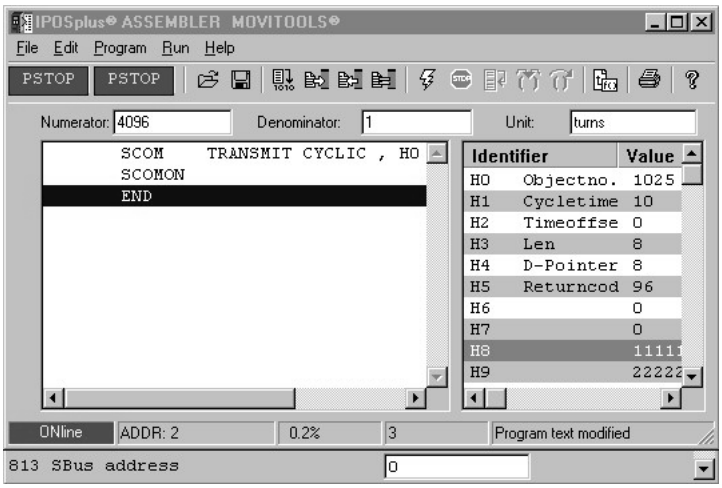
Um die Daten über SBus2 zu senden oder zu empfangen, muss auf die Objektnummer 0x1000000 hinzuaddiert werden (ODER-Verknüpfung).

Beim Benutzen des SCOM-Befehls ist zu beachten, dass selbst die fest speicherbaren Variablen (H0...127) sowie alle Parameter nur flüchtig beschrieben werden.

Beispiel 1

Zyklische Übertragung von zwei Variablenwerten (H008 und H009) mit SCOM-Befehl vom Sender zum Empfänger auf die Variablen H005 und H006.

Einstellung Sender

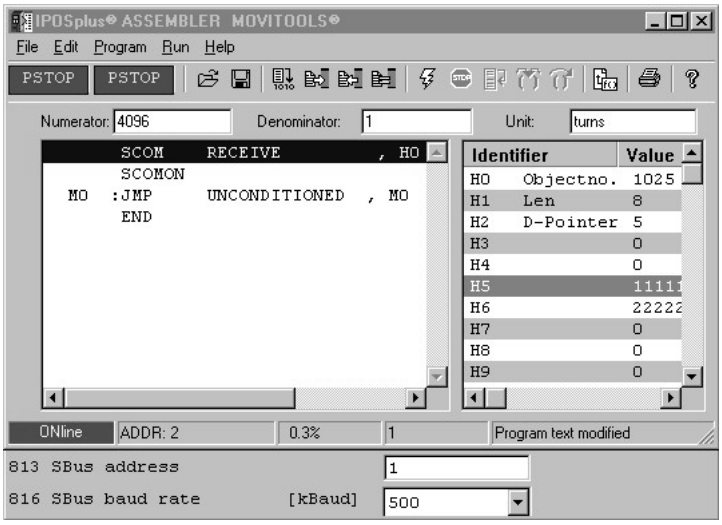


514587147

H0	Objectno.	1025	=	frei wählbar
H1	Cycletime	10	=	10 ms
H2	Timeoffset	0	=	kein Offset
H3	Len	8	=	Variablenlänge 8 Byte
H4	D-Pointer	8	=	Datenpointer Wert liegt bei H8
H5	Returncode	96		
H8		11111	=	gesendeter Wert

Die Länge einer Variablen entspricht 4 Byte. Somit werden bei einer Datenlänge von 8 Byte zwei aufeinanderfolgende Variable übertragen.

Einstellung Empfänger



514591883

H0	Objectno.	1025	=	frei wählbar
H1	Len	8	=	Variablenlänge 8 Byte
H2	D-Pointer	5	=	Datenpointer Wert liegt bei H5
H5		11111	=	gesendeter Wert

HINWEIS

Das Synchronisationsverfahren (Sync-ID) wurde gegenüber dem A-Gerät modifiziert. Anders als bei MOVIDRIVE® A muss bei MOVIDRIVE® B unbedingt darauf geachtet werden, dass im IPOS$^{plus®}$-Programm des Master-Antriebs **zuerst** die Ist-Position und **danach** erst das Sync-Objekt mit SCOM() initialisiert wird.

23.5.5 SCOMON

Systembus **Com**munication **On**

Dieser Befehl stößt den Empfang von Daten oder das zyklische Senden von zuvor festgelegten Datenobjekten an.

Die Initialisierung der Datenobjekte erfolgt durch den SCOM-Befehl mit den Argumenten RECEIVE (Empfang von Daten) oder TRANSMIT CYCLIC (zyklisches Senden von Daten).

Der Befehl wurde bei MOVIDRIVE® B durch SCOMST ersetzt, ist aus Abwärtskompatitbilität aber weiterhin auch im MOVIDRIVE® B verwendbar.

HINWEIS

Dieser Befehl aktiviert nur SBus 1, nicht SBus 2

- Verwenden Sie vorzugsweise den Befehl SCOMSTATE.

Befehlsaufbau

Befehlsaufbau Mxxx SCOMON	Mxxx: Marke (optional)

23.5.6 SCOMST

Diese Anweisung initialisiert die CAN-Schnittstelle und startet oder stoppt den Empfang von Daten und das zyklische Senden von zuvor festgelegten Datenobjekten über SBus 1 oder SBus 2. Die Initialisierung der Datenobjekte erfolgt durch die Funktion SCOM.

Beim MOVITRAC® B wirkt der Befehl SCOMST X1 unabhängig vom Wert für "X1" immer wie SCOMON (Seite 321).

Argument	Bedeutung
START ALL	Synchroner Start der zyklischen Kommunikation von SBus 1 und SBus 2
STOP ALL	Synchroner Stopp der zyklischen Kommunikation von SBus 1 und SBus 2
START1	Start der zyklischen Kommunikation von SBus 1
STOP1	Stopp der zyklischen Kommunikation von SBus 1
START2	Start der zyklischen Kommunikation von SBus 2
STOP2	Stopp der zyklischen Kommunikation von SBus 2

Befehlsaufbau

Befehlsaufbau Mxxx SCOMST X1	Mxxx: Marke (optional) X1: Argument

23.6 Positionierbefehle

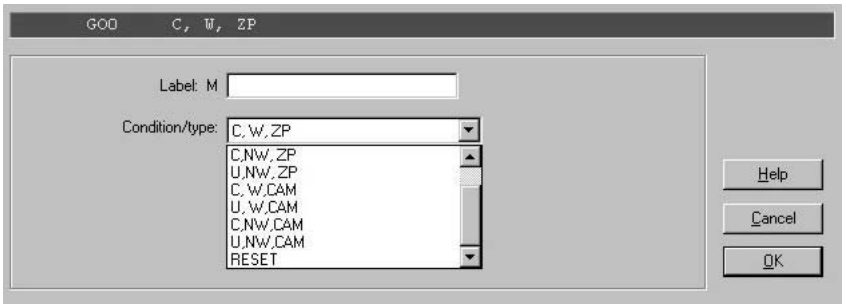
23.6.1 Referenzfahrt GO0

GO0

Der Befehl GO0 löst eine Referenzfahrt aus oder setzt einen Absolutwertgeber. Der Betriebszustand und die 7-Segment-Anzeige wechselt dabei von "A" (Technologie-Option) nach "c" (Referenzbetrieb). Die **Betriebsart P700** bleibt unverändert.

Das Argument des Befehls GO0 bestimmt zusammen mit den Parameter P900 ... P903 das Verhalten der Referenzfahrt.

Das Argument ist eine Kombination aus 3 charakteristischen Eigenschaften (C/U; W/NW; ZP/CAM), hieraus ergeben sich 8 Wahlmöglichkeiten. Mit dem Argument RESET kann eine begonnene Referenzfahrt unterbrochen werden.



514597259

C	(conditional)	Referenziert nur, wenn noch nicht referenziert ist (d. h. H473, Bit 20 = 0).
U	(unconditional)	Referenziert immer.
W	(wait)	Wartet bis Achse referenziert ist. Es wird solange kein weiterer Befehl dieser Task ausgeführt.
NW	(non-wait)	Der nächste Befehl wird abgearbeitet, während referenziert wird (Empfehlung).
ZP	(zero pulse)	Referenziert auf Nullimpuls des Gebersignals (ohne Bedeutung bei 903 = 0 oder P903 = 5).
CAM	(reference cam)	Referenziert auf Referenznocken (ohne Bedeutung bei 903 = 0 oder P903 = 5).
RESET		Begonnene Referenzfahrt wird unterbrochen (Bremsen an der Positionier-Rampe) und die Anforderung zurückgesetzt. Bei einer referenzierten Achse wird die Meldung "Achse referenziert" zurückgesetzt und die Meldung "Achse in Position" gesetzt.

Parametereinstellungen

P60_	Wird ein Referenznocken verwendet, so muss ein Eingang auf die Funktion REFERENCE CAM eingestellt werden.
P900	Referenz-Offset (beschreibt H498).
P901 / P902	Referenzdrehzahlen.
P903	Der Referenzfahrttyp bestimmt zusammen mit dem Argument ZP / CAM die Bedingung für das Ende der Referenzfahrt. Beispiel: P903 = 1, GO0 U,W,ZP Es wird der Nullimpuls nach der Referenznocke ausgewertet.
P904	Ohne Bedeutung bei GO0.

HINWEIS

Um einen Absolutwertgeber mit dem GO0-Befehl zu setzen, ist immer eine Reglerfreigabe notwendig. Alternativ kann ein Geber auch ohne Freigabe gesetzt werden, indem bei Hiperface® der Offset P905 beschrieben wird oder bei SSI-Geber (DIP) die Parameter P953 ... P955 beschrieben werden.

	HINWEIS Bei Typ 3 und 4 und der Einstellung **CAM** muss beachtet werden, dass der Antrieb unmittelbar neben dem Hardware-Endschalter referenziert und positioniert ist. Insbesondere bei Hubwerks-Anwendungen und Referenzpunkt unten kann es dazu kommen, dass der Antrieb beim Positionieren nach unten bereits bei kleinstem Überschwingen den Hardware-Endschalter anspricht. Beim Lösen der Haltebremse besteht die gleiche Gefahr. Abhilfe ist eine zusätzliche Positionierung des Antriebs unmittelbar nach dem Ablauf der Referenzierung, die den Antrieb in ausreichendem Abstand (ca. 0,5 ... 1 Motorumdrehung) vom Hardware-Endschalter weg positioniert.

Sind Software-Endschalter über die Parameter P920 / P921 gesetzt, dann ist die Software-Endschalter-Überwachung erst nach abgeschlossener Referenzfahrt aktiv.

Besitzt der Antrieb keinen Absolutwert- oder Hiperface®-Geber, dann geht die Referenz nach einer Fehlermeldung und anschließendem RESET verloren.

	HINWEIS Wird ein wartender Referenzierbefehl durch Wegnahme von "/Reglersperre" unterbrochen, so wird der Fehlercode 39 (Referenzfahrt) gesetzt. Die Achse läuft nach Wiederkehr des Signals nicht mehr an. Das IPOS$^{plus®}$-Programm bleibt auf dem Befehl stehen. Es muss ein Reset (Binäreingang, Feldbus, SHELL ...) erfolgen. Das IPOS$^{plus®}$-Programm beginnt mit der ersten Anweisung am Anfang.

Befehlsaufbau

Befehlsaufbau	Mxxx: Marke (optional)
Mxxx GO0 X1	X1: Art der Referenzfahrt

23.6.2 Absolutpositionierung GOA / Relativpositionierung GOR

Das Argument des Verfahrbefehls enthält die Zielposition.

	HINWEIS Wird die Zielposition über eine Variable vorgegeben, so kann der Wert nur in Inkrementen (bezogen auf 4096 Inkrementen / Motorumdrehung) eingegeben werden. Konstanten können in Anwendereinheiten eingegeben werden.

	HINWEIS Wird mit der Modulo-Funktion positioniert, so können die Befehle GOA und GOR nicht verwendet werden, die Zielposition wird direkt auf H454 geschrieben.

Man unterscheidet zwischen Absolutpositionierung und Relativpositionierung:

	HINWEIS Endlos in eine Richtung drehende Achsen wie z. B. Rundtische, Förderbänder oder Walzenvorschübe werden typischerweise als Modulo-Achsen deklariert (siehe Modulo-Funktion P960 ... P963). Damit entspricht unabhängig von der Anzahl der verfahrenen Umdrehungen eine mechanische Position der Achse einem Istwert H455 (Voraussetzungen siehe Beschreibung Modulo-Funktion).

Parametereinstellungen für alle Positionierbefehle

Parameter	Erklärung
P913 / P914	Verfahrdrehzahlen (änderbar im Programm über SETSYS).
P911 / P912	Positionier-Rampen (Beschleunigung) (änderbar im Programm über SETSYS).
P915 / P203	Vorsteuerungen, mit denen der Ruck beeinflusst werden kann.
P933	Ruckbegrenzung (nur mit MOVIDRIVE® B).
P916	Rampenform.
P917	Rampenmode.

GOA / GO ABSOLUTE

Dieser Befehl positioniert absolut auf die im zweiten Argument X2 angegebene Position. Argument X2 kann eine Konstante, eine Variable und eine indirekte Variable sein.

Als Verfahrweg wird die Zielposition bezogen auf Position 0 (Maschinen-Nullpunkt) eingegeben. Die daraus resultierende Zielposition wird in der Systemvariablen H492 (TARGET POSITION) abgebildet.

Die Meldung "IPOS In Position" wird innerhalb eines GOA- oder GOR-Befehls aktualisiert, d. h. die Meldung kann direkt in der nächsten Programmzeile abgefragt werden.

Befehlsaufbau

Befehlsaufbau Mxxx GOA X1 X2	Mxxx:	Marke (optional)
	X1:	NoWait: Die Programmabarbeitung wird fortgesetzt während der Antrieb noch verfährt. Dies ermöglicht eine parallele Programmverarbeitung während des Verfahrens. Wait: Die Programmabarbeitung wird erst dann fortgesetzt, wenn der Antrieb mit seiner Ist-Position das Positionsfenster P922 der Zielposition erreicht hat.
	X2:	K = Zielposition in Anwendereinheiten als Konstante. H = Zielposition in Anwendereinheiten als Variable. [H] = Variable, die die Zielposition in Inkrementen enthält, bezogen auf 4096 Inkremente / Motorumdrehung.

GOR / GO RELATIVE

Dieser Befehl positioniert relativ auf die im zweiten Argument X2 angegebene Position. Argument X2 kann eine Konstante, eine Variable und eine indirekte Variable sein.

Die eingetragene Verfahrstrecke wird zur aktuellen Zielposition H492 (TARGET POSITION) des Antriebs hinzuaddiert und in dieser angezeigt.

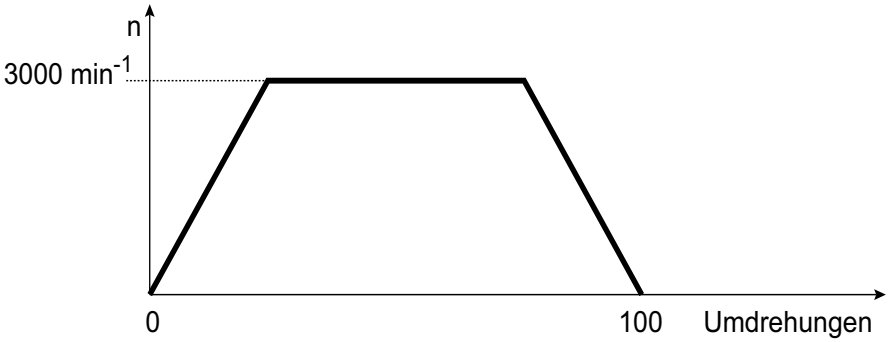
Die Meldung "IPOS In Position" wird innerhalb eines GOA- oder GOR-Befehls aktualisiert, d. h. die Meldung kann direkt in der nächsten Programmzeile abgefragt werden.

Befehlsaufbau

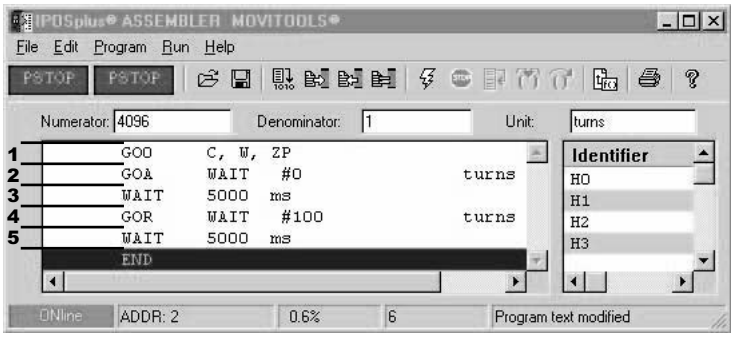
Befehlsaufbau Mxxx GOR X1 X2	Mxxx:	Marke (optional)
	X1:	NoWait: Die Programmabarbeitung wird fortgesetzt während der Antrieb noch verfährt. Dies ermöglicht eine parallele Programmverarbeitung während des Verfahrens (Empfehlung). Wait: Die Programmabarbeitung wird erst dann fortgesetzt, wenn der Antrieb mit seiner Ist-Position das Positionsfenster P922 der Zielposition erreicht hat.
	X2:	K = Zielposition in Anwendereinheiten als Konstante. H = Zielposition in Anwendereinheiten als Variable. [H] = Variable, die die Zielposition in Inkrementen enthält, bezogen auf 4096 Inkremente / Motorumdrehung.

Beispiel 1

Das unten aufgeführte Programm bewirkt, dass zwischen den Positionen 0 Umdrehungen und 100 Umdrehungen verfahren wird (Eingabe im Programmkopf: Zähler, Nenner, Einheit). Nach Erreichen einer Position erfolgt eine Wartezeit von 5 Sekunden.



515070219



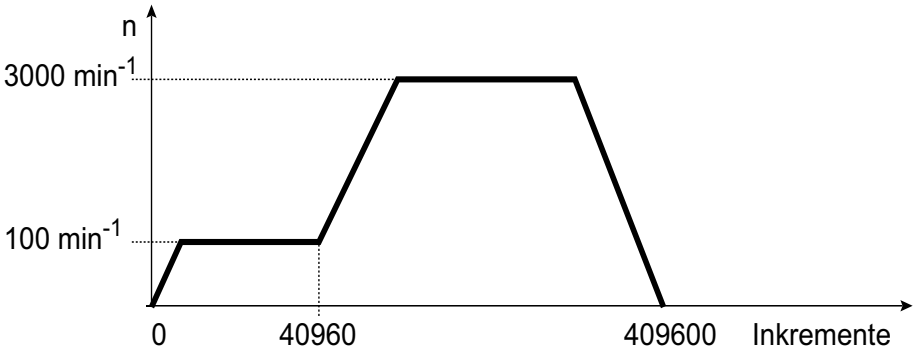
515071755

1. Referenzfahrt
2. Fahre auf Null
3. Warte 5 s
4. Fahre auf 100
5. Warte 5 s

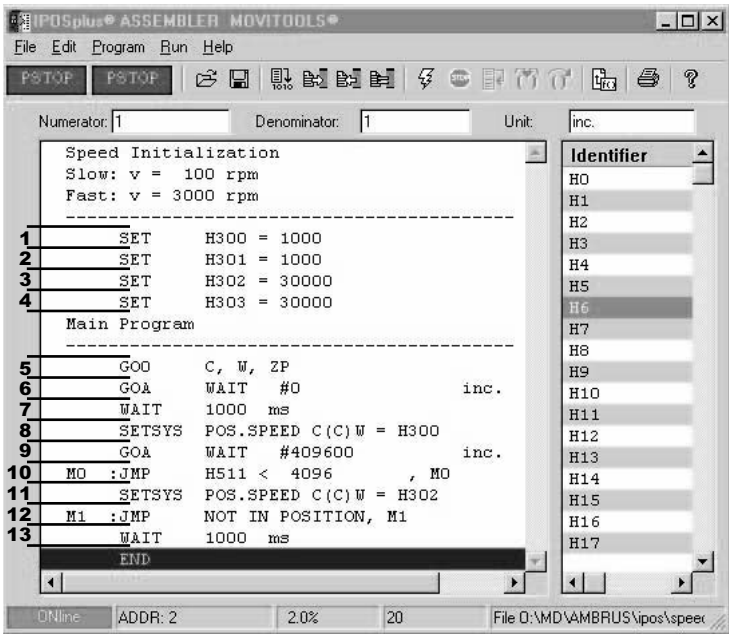
Nach Ausführen des END-Befehls fängt das IPOS[plus®]-Programm automatisch mit der Abarbeitung der ersten Zeile an.

Beispiel 2

Das unten aufgeführte Programm bewirkt, dass zwischen den Positionen 0 und 409600 Inkrementen verfahren wird. Nach Erreichen einer Position erfolgt eine Wartezeit von 1 Sekunde. Bei Überfahren der Position 40960 wird die Geschwindigkeit von 100 min^{-1} auf 3000 min^{-1} erhöht. Die gesamte Rückfahrt erfolgt mit 3000 min^{-1}.



515073291



515074827

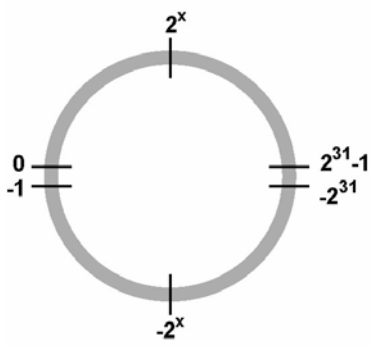
1. Langsame Rechtsfahrt 100 rpm
2. Langsame Linksfahrt 100 rpm
3. Schnelle Rechtsfahrt 3000 rpm
4. Schnelle Linksfahrt 3000 rpm
5. Referenzfahrt
6. Fahre auf Null
7. Warte 1 s
8. Setze langsame Drehzahl
9. Fahre auf Endziel
10. Solange Actpos Mot (H511) kleiner 40960, verbleibe in aktueller Zeile
11. Setze schnelle Drehzahl
12. Verbleibe in aktueller Zeile, solange Antrieb noch fährt
13. Warte 1 s

Endlospositionierung

Der absolute Verfahrbereich von IPOS$^{plus®}$ ist auf den Eingabewert -2^{31} ... 0 ... $2^{31}-1$ begrenzt. Mit dem relativen Verfahrbefehl kann zu jeder beliebigen Zielposition eine maximale Verfahrstrecke von 2^{31} hinzuaddiert werden (siehe Zahlenkreis).

Ein Beispiel für Endlospositionierung wird im Beispielprogramm Tippbetrieb gezeigt.

	HINWEIS Der GOR-Befehl bezieht sich immer auf die Zielposition H492. Setzt man z. B. in einem Programm 100 Mal den Befehl GOR 1000 Inkr. ab, so wird intern die Zielposition auf 100 x 1000 Inkremente gesetzt. Wird der Befehl zyklisch aufgerufen, so kann die Sollposition von der Ist-Position des Motors weglaufen und ab einem kritischen Wert $2^{31}/2$ zu fehlerhaftem Verhalten führen (Antrieb dreht in entgegengesetzte Richtung).



515076363

23.7 Programmbefehle

23.7.1 Programmende END

END

Die END-Anweisung zeigt das textuelle (nicht das logische) Ende eines IPOS$^{plus®}$-Programms an. Die END-Anweisung ist kein IPOS-Befehl, sie kann nicht gelöscht werden.

23.7.2 Unterprogrammaufruf CALL

CALL

Unterprogramme werden mit einem CALL-Befehl (CALL Mxx) aufgerufen. Die zugehörigen Sprungmarken (Mxx) werden vor dem ersten Befehl des Unterprogramms eingefügt. Ein Unterprogramm endet mit einem RETURN-Befehl (RET). Mit diesem RETURN-Befehl erfolgt ein Rücksprung in die Zeile unterhalb des CALL-Befehls. Anschließend werden die folgenden Programmzeilen abgearbeitet. Es sind auch geschachtelte Unterprogrammaufrufe möglich (maximale Verschachtelungstiefe: 32 Ebenen).

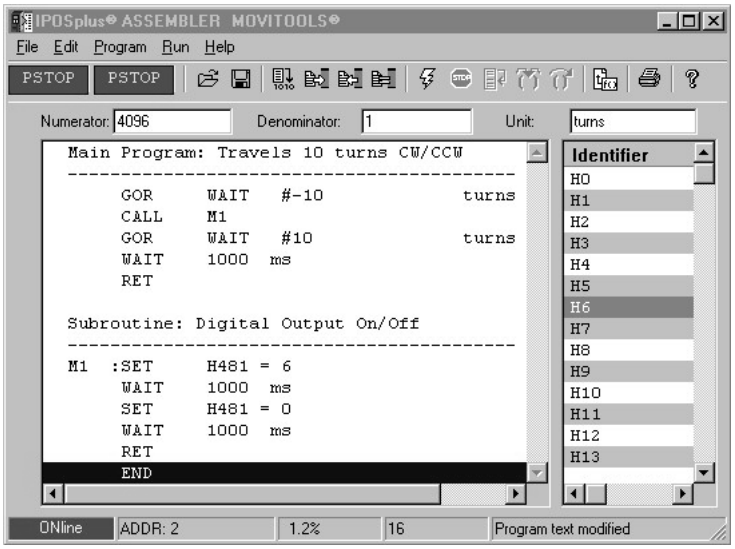
	HINWEIS Unterprogramme dürfen auf keinen Fall mit einem Sprung in ein Hauptprogramm oder ein weiteres Unterprogramm verlassen werden. Ein bedingtes Verlassen des Unterprogramms muss mit einem Sprung ans Ende des Unterprogramms erfolgen.

Befehlsaufbau

Befehlsaufbau	Mxxx: Marke (optional)
Mxxx CALL Myyy	Myyy = Marke, bei der das Unterprogramm beginnt.

Beispiel

Das Hauptprogramm positioniert den Antrieb 10 Umdr. nach links, dann erfolgt der Unterprogrammaufruf (CALL M1). Hier werden 2 Ausgänge (auf "IPOS-AUSGANG" zu parametrieren) des Grundgerätes für 1 s gesetzt. Anschließend erfolgt der Sprung zurück ins Hauptprogramm (RET) und die Abarbeitung des Positionierbefehls GOR WAIT #10.



515186699

23.7.3 Sprungbefehle JMP

JMP / Klemmen

Es wird ein Sprung auf die angegebene Marke ausgeführt, wenn die in der Maske mit markierten Eingangsklemmen alle 1- oder 0-Pegel haben (UND-Verknüpfung).

Die Bits 0 ... 5 zeigen dabei die Klemmen des Grundgeräts, die Bits 6 ... 13 die Klemmen der Optionskarte (DIO11A) an. Die Maske wird durch direkte Eingabe der Klemmenpegel im Eingabefenster erzeugt.

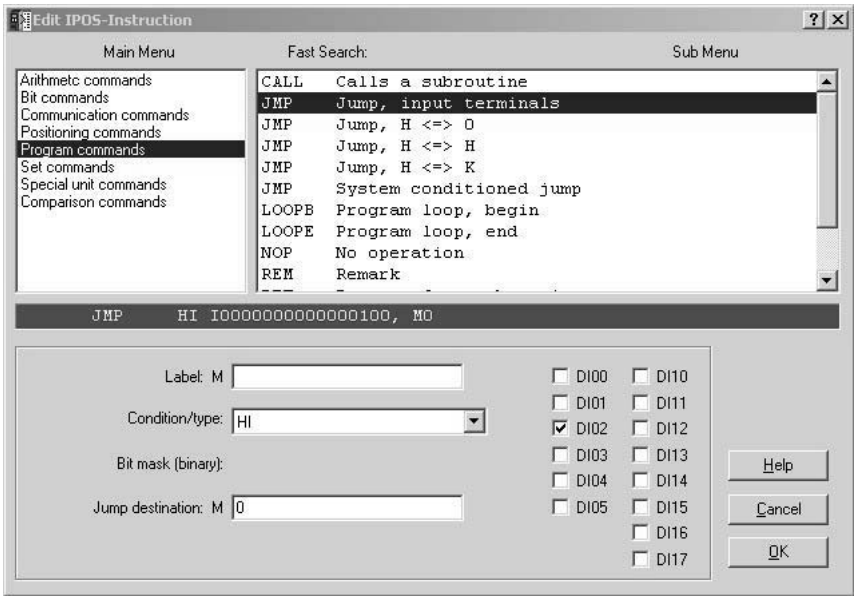
Befehlsaufbau

Befehlsaufbau		
Mxxx JMP X1 X2, Myyy	Mxxx:	Marke (optional)
	X1:	HI = Sprung, falls die in der Maske markierten Eingangsklemmen 1-Pegel haben. LO = Sprung, falls die in der Maske markierten Eingangsklemmen 0-Pegel haben.
	X2:	Ixxx ... = Maske für die Eingangsklemmen.
	Myyy:	Sprungmarke, zu der verzweigt wird.

Beispiel

```
JMP HI I 0000000000000011, M03
```

Nach dem JMP-Befehl wird das Programm bei der Marke M03 fortgesetzt, wenn die Eingangsklemmen DI00 und DI01 1-Pegel haben.



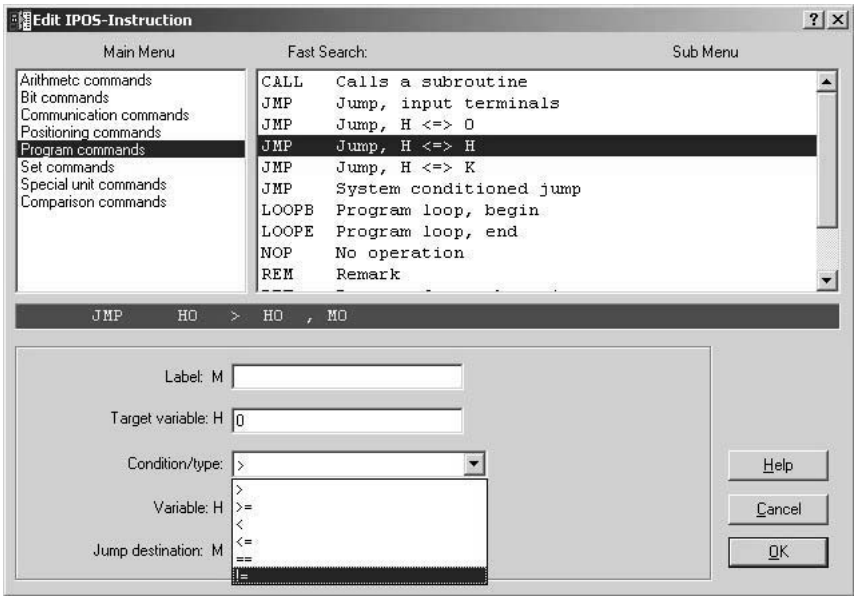
515191051

JMP / Vergleich

Der JMP-Befehl führt einen Sprung zu der angegebenen Marke aus, wenn der im Befehl angegebene Vergleich eine wahre Aussage ergibt.

Befehlsaufbau

Befehlsaufbau Mxxx JMP X1 OP X2, Myyy	Mxxx:	Marke (optional)
	X1:	Variable
	OP:	Operator: > / > = / < / <= / = / !=
	X2:	H = Variable K = Konstante 0 = Null (bei dem Vergleich mit Null sind nur die Operatoren = und != möglich).
	Myyy:	Sprungmarke, zu der bei erfüllter Bedingung verzweigt wird.



515195787

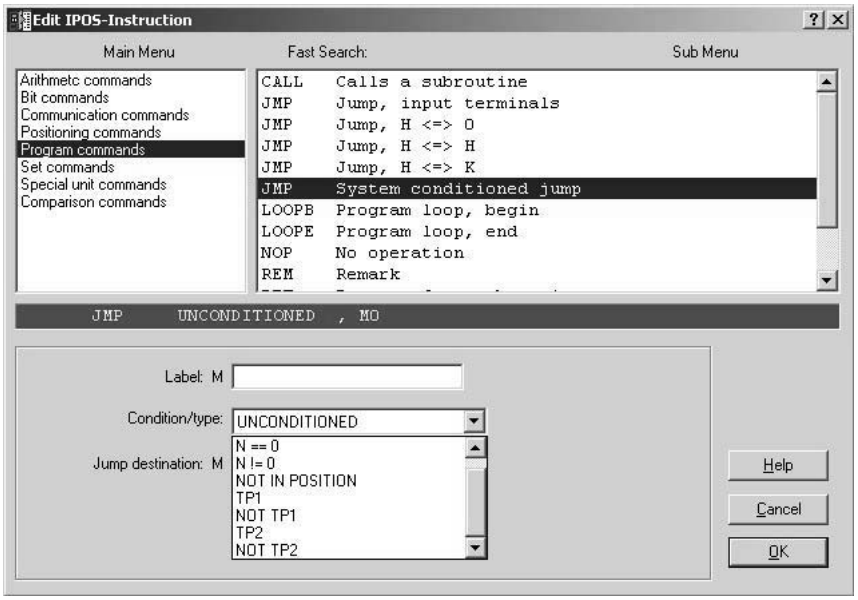
JMP / System

Es wird auf die im Befehl angegebene Marke verzweigt, wenn die Vergleichsoperation erfüllt ist.

Die aufgeführten Systemgrößen können direkt über einen JMP-Befehl abgefragt werden. Andere Systemgrößen stehen als Systemvariablen zur Verfügung oder müssen über den Befehl GETSYS eingelesen und weiterverarbeitet werden.

Befehlsaufbau

Befehlsaufbau Mxxx JMP X1, Myyy	Mxxx:	Marke (optional)
	X1:	UNCONDITIONED: Unbedingter Sprung. N == 0: Sprung, wenn Drehzahl gleich Null. N != 0: Sprung, wenn Drehzahl ungleich Null. NOT IN POSITION: Sprung, wenn nicht in Position TP1: Sprung, wenn Flankenwechsel an Touch-Probe-Klemme DI02. NOT TP1: Sprung, wenn kein Flankenwechsel an Touch-Probe-Klemme DI02. TP2: Sprung, wenn Flankenwechsel an Touch-Probe-Klemme DI03. NOT TP2: Sprung, wenn kein Flankenwechsel an Touch-Probe-Klemme DI03.
	Myyy:	Sprungmarke, zu der bei erfüllter Bedingung verzweigt wird.



515235723

23.7.4 Schleifenbefehle LOOP

	HINWEIS Programmschleifen dürfen auf keinen Fall mit einem Sprungbefehl verlassen werden. Sprungbefehle und Unterprogramme innerhalb einer Programmschleife sind erlaubt.

LOOPB / LOOP BEGIN

Dieser Befehl erlaubt in Verbindung mit einem LOOPE-Befehl die Realisierung einer Programmschleife. Die Anzahl der Schleifendurchläufe (> 0) wird als Konstante angegeben. Die Schleife endet am zugehörigen LOOPE-Befehl. Es ist erlaubt, Programmschleifen zu verschachteln.

Soll die Anzahl der Schleifendurchläufe variabel sein, so muss statt des LOOP-Befehls ein JMP-Befehl verwendet und die variable Bedingung bei jedem Durchlauf geprüft werden.

Befehlsaufbau

Befehlsaufbau		
Mxxx LOOPB X1	Mxxx:	Marke (optional)
	X1:	Anzahl der Schleifendurchläufe (maximal 256)

LOOPE / LOOP END

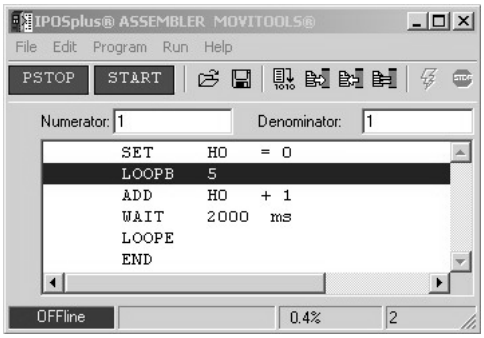
Dieser Befehl kennzeichnet das Ende einer Programmschleife, die mit einem LOOPB-Befehl begonnen wurde.

Befehlsaufbau

Befehlsaufbau Mxxx LOOPE	Mxxx: Marke (optional)

Beispiel

Im nachfolgenden Beispiel wird die Variable H0 in 5 Schleifendurchläufen von 0 auf den Wert 5 hochgezählt. Nach Beendigung der 5 Schleifendurchläufe beginnt die Programmabarbeitung erneut mit dem Befehl SET H0 = 0.



515243147

23.7.5 Keine Operation NOP / Kommentar REM / Rücksprung RET / TASK / TASK2 / Warten WAIT

NOP / NO OPERATION

Es wird keine Operation ausgeführt. Mit diesem Befehl können z. B. Wartezeiten auf Basis der Befehlszykluszeit erreicht werden. Bei dem MOVIDRIVE® A ist dies in Task 1 z. B. 1 Befehl/ms.

Befehlsaufbau

Befehlsaufbau Mxxx NOP	Mxxx: Marke (optional)

REM / REMARK

Der REM-Befehl fügt eine Kommentarzeile in das Programm ein. Kommentarzeilen können nicht im Umrichter gespeichert werden. Nach einem DOWNLOAD und einem erneuten UPLOAD des Programms sind alle Kommentarzeilen verloren.

Kommentarzeilen können nur in Programmdateien auf dem PC gespeichert werden.

Befehlsaufbau

Befehlsaufbau REM X1	X1: Beliebige Zeichenfolge

RET / RETURN

Der RET-Befehl beendet ein Unterprogramm (siehe CALL-Befehl) und kehrt in das aufrufende Programm zurück. Im Hauptprogramm bewirkt der RET-Befehl einen Sprung zum Beginn des Hauptprogramms.

Befehlsaufbau

Befehlsaufbau Mxxx RET	Mxxx:	Marke (optional)

TASK

Dieser Befehl dient dazu die Startadresse von Task 2 und Task 3 festzulegen und diese mit dem Argument X1 (START/ STOP) zu starten oder zu stoppen, das heißt das Steuerwort der Task wird beschrieben. Nach Netz-Ein stehen Steuerwort und Startadresse auf 0, das heißt der Task ist deaktiviert.

Der Befehl ist erst ab MOVIDRIVE® B verfügbar.

Befehlsaufbau

Befehlsaufbau Mxxx TASK X1 Myyy	Mxxx:	Marke (optional)
	X1:	TASK2 STOP: Task 2 stoppen. TASK2 START: Task 2 starten. TASK3 STOP: Task 3 stoppen. TASK3 START: Task 3 starten.
	Myyy:	Marke, bei der der Task beginnt.

Beispiel

```
TASK TASK2 START M03
```

Nach dem Befehl ist Task 2 gestartet und es wird parallel zu Task 1 der erste Befehl nach der Marke M03 bearbeitet.

TASK2

Dieser Befehl dient dazu die Startadresse von TASK2 festzulegen und diese mit dem Argument X1 (START/ STOP) zu starten oder zu stoppen, das heißt das Steuerwort von TASK2 wird beschrieben. Nach Netz-Ein stehen Steuerwort und Startadresse auf 0, das heißt TASK2 ist deaktiviert.

Der Befehl wurde bei MOVIDRIVE® B durch TASK ersetzt, ist aus Abwärtskompatibilität aber weiterhin auch im MOVIDRIVE® B verwendbar.

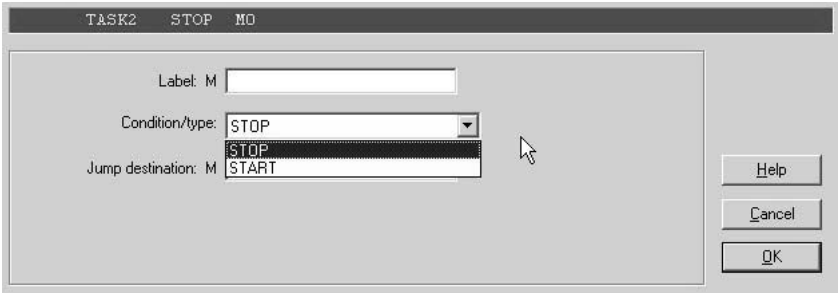
Befehlsaufbau

Befehlsaufbau Mxxx TASK2 X1 Myyy	Mxxx:	Marke (optional)
	X1:	STOP: Task 2 stoppen. START: Task 2 starten.
	Myyy:	Marke, bei der Task 2 beginnt.

Beispiel

```
TASK2 START M03
```

Nach dem Befehl ist Task 2 gestartet und es wird parallel zu Task 1 der erste Befehl nach der Marke M03 bearbeitet.



515247499

WAIT

Der Befehl WAIT wartet die im Argument angegebene Zeit in ms und setzt dann die Programmbearbeitung dieser Task fort.

	HINWEIS Soll die Wartezeit variabel sein, so muss statt des WAIT-Befehls ein Timer (H487 ... H489) initialisiert werden und mit dem JMP-Befehl eine Schleife programmiert werden, bis der Timer abgelaufen ist.

Befehlsaufbau

Befehlsaufbau		
Mxxx WAIT X1	Mxxx:	Marke (optional)
	X1:	Wartezeit in ms, 0 ... 32767.

Beispiel

```
SET H0 = 20000
SET H489 = H0
M01: JMP H489 != 0, M01
```

23.8 Setzbefehle

23.8.1 Variablen kopieren COPY

COPY

Mit dem COPY-Befehl wird die im 3. Argument angegebene Anzahl aufeinanderfolgender Variablen kopiert. Das 2. Argument des COPY-Befehls gibt die Nummer der ersten Quell-Variablen an, das 1. Argument die Nummer der ersten Ziel-Variablen an. Es können maximal 10 Variablen mit einem COPY-Befehl kopiert werden.

Befehlsaufbau

Befehlsaufbau		
Mxxx COPY X1 = X2, X3	Mxxx:	Marke (optional)
	X1:	Hxxx = Nummer der ersten Zielvariablen.
	X2:	Hyyy = Nummer der ersten Quellvariablen.
	X3:	K = Konstante (Anzahl der zu kopierenden Variablen, 1 ... 10).
	Myyy:	Sprungmarke, zu der bei erfüllter Bedingung verzweigt wird.

Beispiel

Der Befehl `COPY H2 = H20, 3` entspricht der Befehlsfolge:

```
SET H2 = H20
SET H3 = H21
SET H4 = H22
```

23.8.2 Systemgrößen lesen GETSYS

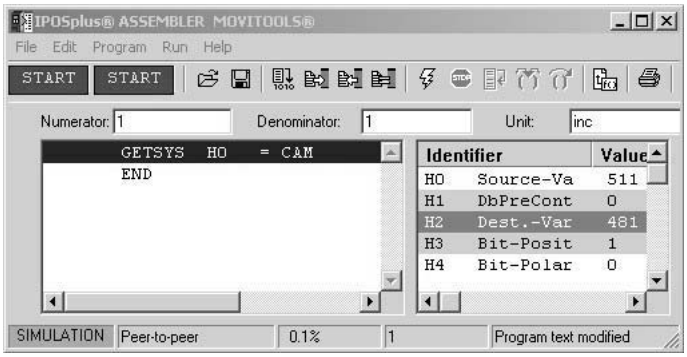
GETSYS / GET SYSTEM VALUE

Der Befehl GETSYS lädt den Wert einer internen Systemgröße Argument X2 in eine oder mehrere Variablen Argument X1.

Befehlsaufbau

Befehlsaufbau		
Mxxx GETSYS X1 X2	Mxxx:	Marke (optional)
	X1:	Hxxx = Anfang der Variablenstruktur, die nach Ausführung des Befehls das Ergebnis enthält.

X2:	
ACTIVE CURRENT	Wirkstrom in 0,1 % Gerätenennstrom
ACT.SPEED	Ist-Drehzahl in 0,1 % min^{-1}
SETP.SPEED	Solldrehzahl in 0,1 % min^{-1}
ERROR	Fehler Code entsprechend der Tabelle "Fehlermeldungen und Fehlerliste" im Systemhandbuch
SYSTEM STATUS	Betriebszustand, Wert der 7-Segment-Anzeige ohne Fehlerstatus entsprechend der Tabelle "Betriebsanzeigen" im Systemhandbuch
ACT.POSITION	Ist-Position, abhängig von dem in P941 angewählten Geber H509, H510 oder H511
SETP.POSITION	Sollposition (aktuelle Sollwertvorgabe des Profilgenerators während der Ausführung eines Verfahrbefehles), identisch mit Systemvariable H491
TARGET POSITION	Zielposition, identisch mit Systemvariable H492
INPUTS	Binäre Eingänge H483 (MOVIDRIVE® A) / H520 (MOVIDRIVE® B) Grundgerät und Option, identisch mit Systemvariable H483
DEVICE STATUS	Identisch mit Statuswort 1 des Feldbus-Geräteprofils (Fehlercode + Betriebszustand)
OUTPUTS	Binäre Ausgänge H482 (MOVIDRIVE® A) / H521 (MOVIDRIVE® B) Grundgerät und Option
IxT	Geräteauslastung in 0,1 % Gerätenennstrom
ACT.POSITION / SETP.POSITION / TARGET POSITION: Auflösung abhängig von mit P941 ausgewählten Geber: – Motorgeber: 4096 Ink./Umdrehung – Externer Geber X14: Geberauflösung P944 – DIP (SSI-Geber): Geberauflösung P955	

ANALOG INPUTS	-10 V ... 0 ... +10 V = -10000 ... 0 ... 10000	
	H+0	Spannungswert Analogeingang 1 [mV]
	H+1	Spannungswert Analogeingang 2 [mV]
CAM		Der Befehl **GETSYS H = CAM** bildet ein Nockenschaltwerk nach. Pro Antrieb kann mit dem GETSYS-Befehl ein Standard-Nockenschaltwerk mit 1 Ausgang pro Nocke genutzt werden. Mit MOVIDRIVE®-Geräten kann ein erweitertes Nockenschaltwerk mit 8 Ausgängen genutzt werden. Hxx ist die erste Variable einer Datenstruktur (CamControl). Mit dem höchstwertigen Bit 31 in Hxx wird entschieden, auf welches Nockenschaltwerk sich der GETSYS-Befehl bezieht. Bit 31 = 0: Standard-Nockenschaltwerk (alle MOVIDRIVE®-Geräte). Der GETSYS-Befehl aktiviert das Nockenschaltwerk, die Nocken werden, wenn der GETSYS-Befehl bearbeitet wird, einmalig gebildet. Soll das Standard-Nockenschaltwerk zyklisch bearbeitet werden, muss der Befehl zyklisch aufgerufen werden. Bit 31 = 1: Erweitertes Nockenschaltwerk mit Technologieoption und Betriebsart CFC oder SERVO). Der GETSYS-Befehl aktiviert das Nockenschaltwerk, die Nocken werden zyklisch im Hintergrund gebildet. Die Struktur der Variablen ist abhängig davon, ob das Standard-Nockenschaltwerk oder das erweiterte Nockenschaltwerk aufgerufen wird. Die Datenstruktur ist im Kapitel "Wegerfassung und Positionierung / Nockenschaltwerke" beschrieben.  515353867
ANALOG OUTPUTS	+/- 10 V entsprechen +/- 10000	
	H	Die Variable im Befehl GETSYS H = ANALOG OUTPUTS definiert den Anfang der nachfolgend beschriebenen Variablenstruktur.
	H+0	Enthält den Spannungswert des analogen Ausgangs 1 (AO1)
	H+1	Enthält den Spannungswert des analogen Ausgangs 1 (AO2)
TIMER 0		Lädt den aktuellen Wert des Timers 0 [ms], identisch mit Systemvariable H489
TIMER 1		Lädt den aktuellen Wert des Timers 1 [ms], identisch mit Systemvariable H488
PO-DATA		Lesen des PA-Datenpuffers (vom Master an das Gerät gesendete Daten). Abhängig von der Anzahl PA-Daten wurden 3 PA- oder 10 PA-Daten gelesen.
	H+0	Bus-Typ 0 = reserviert 1 = TERMINAL 2 = RS-485 3 = Feldbus 4 = reserviert 5 = SBus 8 = SBus 2 (nur MOVIDRIVE® B)
	H+1	Anzahl der PA-Daten
	H+2	PA1
	H+3	PA2
	H+4	PA3
	H+5	PA4
	H+6	PA5
	H+7	PA6
	H+8	PA7
	H+9	PA8
	H+10	PA9
	H+11	PA10

DC-VOLTAGE		Zwischenkreis-Spannung [V]
RELATED TORQUE **REL. TORQUE VFC**		Relatives Drehmoment / Relatives Drehmoment VFC. Das relative Moment ist der auf den Gerätenennstrom bezogene Anzeigewert für das Drehmoment an der Motorabtriebswelle. Aus dieser Größe lässt sich das absolute Drehmoment nach der folgenden Formel berechnen: $M_{abs} = M_{rel} \times I_N \times M_N / (1000 \times I_{QN})$ M_{abs} = absolutes Moment I_N = Gerätenennstrom M_{rel} = relatives Moment bezogen auf 0.1 % I_N M_N = Bemessungs-Drehmoment des Motors [Nm] I_{QN} = Nenn-Q-Strom [A] für gewählte Schaltungsart. Die Größe steht in den Betriebsarten CFC und SERVO / VFC1, VFC1 & Hubwerk, VFC1 & DC-Bremsung und VFC1 & Fangen zur Verfügung.
ACT. SPEED EXT.		Ist-Drehzahl des externen Gebers X14. Es wird die folgende Datenstruktur verwendet:
	H	Time Base 5 ms ... 31 ms: Mittelwertfilter für die Drehzahlerfassung des externen Gebers.
	H+1	Encoder-Typ 0 = Geber X14 1 = DIP-Geber
	H+2	Numerator -2^{15} ... 0 ... + 2^{15} -1: Zähler für die Anwenderskalierung.
	H+3	Denominator 1 ... 2^{15} -1: Nenner für die Anwenderskalierung.
	H+4	D-Pointer 0 ... 458: Zeiger auf die Ergebnisvariable H'.
	H'	Result Einheit: [nX14] = Inc./Time Base.

		Beispiel: Die Geschwindigkeit des Leitgebers soll in Bögen pro Stunde angezeigt werden, mit Mittelwertfilter = 30 ms und Geber X14 berechnen sich die Bögen pro Stunde aus:
		11250 / 384 = (1000 ms × 60 s × 60 min) / (Inkr. pro Lastumdr × TimeBase)
		Durch das Minus bei H32 wird die Drehrichtung invertiert.
		`SET H30 = 30`
		`SET H31 = 0`
		`SET H32 = -11250`
		`SET H33 = 384`
		`SET H34 = 40`
		`GETSYS H30 = ACT.SPEED EXT.`

SPEED MON. TIMER		Zählerwert der Drehzahl-Überwachung. Der GETSYS-Befehl kann als Vorwarnung für die Drehzahl-Überwachung verwendet werden. Die Drehzahl-Überwachung spricht an, wenn der Strom P501 Sekunden in der Stromgrenze ist. Ist z. B. P501 = 200 ms, kann mit GETSYS der Zählerwert abgefragt werden. Damit ist es möglich eine Leerfahrt im Eilgang zu fahren und unter Last die Drehzahl im Umrichter intern automatisch zu reduzieren.

23.8.3 Setzbefehle Variable SET / Fehlerreaktion SETFR / Indirekte Adressierung SETI / Interrupt SETINT / Systemgrößen SETSYS

SET

Mit dem SET-Befehl wird das Argument X1 mit dem Inhalt von Argument X2 (Variable H oder Konstante K) geladen. Das Ergebnis steht in Argument X1, Argument X2 bleibt immer unverändert.

Befehlsaufbau

Befehlsaufbau Mxxx SET X1 = X2	Mxxx:	Marke (optional)
	X1:	Hxxx = Ergebnis der Zuweisung.
	X2:	Hyyy = Quelle.

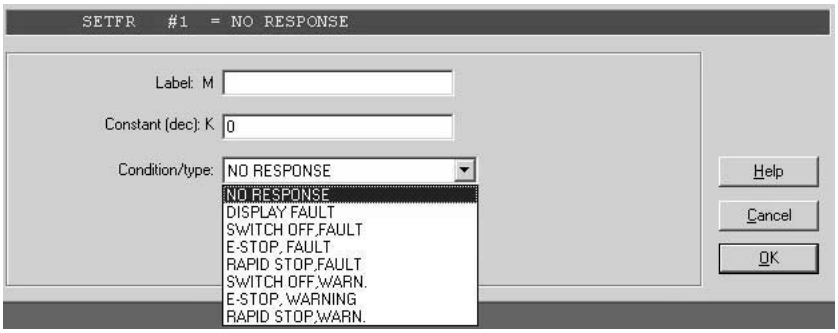
SETFR / SET FAULT REACTION

Der Befehl SETFR bestimmt die Reaktion auf einen Gerätefehler. Im Argument X1 des Befehls wird der Fehlercode eingetragen. Die Reaktion auf den Gerätefehler wird mit Argument 2 gewählt. Die gewählte Fehlerreaktion wird nur durchgeführt, wenn der Befehl SETFR vorher abgearbeitet wurde. Die **zuletzt** gewählte Fehlerreaktion (Aufruf des SETFR-Befehls oder Änderungen in P83_ "Fehlerreaktionen") ist wirksam.

Sie können alle Reaktionen auf einen Fehler programmieren, der in der Fehlerliste der Bedienungsanleitung oder des Systemhandbuchs in der Spalte "P" einen Punkt hat.

Beispiel:

- Fehler 27 "Endschalter fehlen", kein Punkt in Spalte P: nicht programmierbar
- Fehler 28 "Feldbus Time-out", Punkt in Spalte P: programmierbar



515358219

Befehlsaufbau

Befehlsaufbau Mxxx SETFR #X1 = X2	Mxxx:	Marke (optional)
	X1:	Fehlercode des Fehlers, für den die Reaktion gelten soll. Die folgenden Fehlernummern sind zulässig: 08: n-Überwachung 11: Übertemperatur 26: externe Klemme (P830) 28: Feldbus Time-out (P831) 31: TF-Auslöser (P835) 39: Referenzfahrt 42: Schleppfehler (P834) 43: RS-485 Time-out (P833) 47: Timeout SBus (P836) 77: IPOS Steuerwert 78: IPOS Software-Endschalter (P838) 84: Motorschutz (P832) 92: DIP Arbeitsbereich 93: DIP Absolutwertgeber

X2:	
NO RESPONSE	**Keine Reaktion** (auch keine Fehleranzeige).
DISPLAY FAULT	**Keine Reaktion, nur Fehleranzeige** (der Klemmenpegel eines auf "/STÖRUNG" programmierten Ausganges wird von 1 auf 0 gesetzt).
SWITCH OFF, FAULT	**Sperrung der Endstufe, kein Drehmoment, Ansteuerung der Bremse.** Nach Reset: Verhalten wie bei Netz-Aus / Netz-Ein: IPOS$^{plus®}$-Programm, Referenzpunkt, Ausgänge, durch IPOS$^{plus®}$ gesetzte Parameter (SETSYS-Befehl) und Variablen werden zurückgesetzt (Programm beginnt in Zeile 1).
E-STOP, FAULT	**Der Antrieb wird an der Notstopp-Rampe stillgesetzt.** Nach Reset: Verhalten → SWITCH OFF, FAULT.
RAPID STOP, FAULT	**Der Antrieb wird an der Schnellstopp-Rampe stillgesetzt.** Nach Reset: Verhalten → SWITCH OFF, FAULT.
SWITCH OFF WARNING	**Sperrung der Endstufe, kein Drehmoment, Ansteuerung der Bremse.** IPOS$^{plus®}$-Programm läuft weiter, Referenzpunkt, Ausgänge, durch IPOS$^{plus®}$ gesetzte Parameter (SETSYS-Befehl) und Variablen bleiben erhalten.*
E-STOP, WARNING	**Der Antrieb wird an der Notstopp-Rampe stillgesetzt.** IPOS$^{plus®}$-Programm läuft weiter, → SWITCH OFF, WARNING.*
RAPID STOP, WARING	**Der Antrieb wird an der Schnellstopp-Rampe stillgesetzt.** IPOS$^{plus®}$-Programm läuft weiter, → SWITCH OFF, WARNING.*
* auch nach Fehlerquittierung	

SETI / SET INDIRECT H = [H]

Die Variable X1 erhält den Wert der Variablen, deren Nummer in der Variablen X2 steht.

	HINWEIS Ist die Nummer der indirekt adressierten Variablen außerhalb des definierten Bereichs (z. B. MOVIDRIVE® A Bereich 0 ... 512), so wird die Fehlermeldung IPOS INDEX ÜBERL (32) generiert.

Befehlsaufbau

Befehlsaufbau Mxxx SETI X1 = [X2]	Mxxx:	Marke (optional)
	X1:	Hxxx = Zielvariable.
	X2:	Hyyy = Nummer der Quellvariablen.

Beispiel

```
SET H1 = 7
SET H7 = 11
SET H3 [H1]
```

Nach Ausführung des Programms haben die Variablen folgende Werte:

H1 = 7

H7 = 11

H3 = 11

SETI / SET INDIRECT [H] = H

Die Variable mit der Nummer, die in Variable X1 steht, erhält den Wert von Variable X2.

	HINWEIS Ist die Nummer der indirekt adressierten Variablen außerhalb des definierten Bereichs (z. B. MOVIDRIVE® A Bereich 0 ... 512), so wird die Fehlermeldung IPOS INDEX ÜBERL (32) generiert.

Befehlsaufbau

Befehlsaufbau Mxxx SETI [X1] = X2	Mxxx:	Marke (optional)
	X1:	Hxxx = Nummer der Zielvariablen.
	X2:	Hyyy = Quellvariable.

Beispiel

```
 SET H01  = 50
 SET H0 = 10
M01 :SETI  [H0] = H01
 ADD H0 + 1
 ADD H01  + 10
 JMP H0 <= 15 , M01
```

Nach Ausführung des Programms haben die Variablen folgende Werte:

H10 = 50

H11 = 60

...

H15 = 100

SETINT / SET INTERRUPT

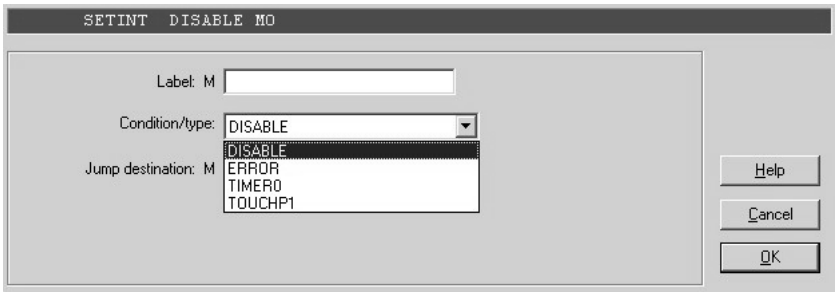
Der Befehl SETINT setzt die Anfangsadresse einer Interrupt-Routine. Die Adresse wird als Marke im Befehl angegeben. Ein Interrupt kann durch verschiedene Ereignisse ausgelöst werden. Die Ereignisse werden im Argument X1 angegeben. Die Interrupt-Routine selbst ist mit einem RET-Befehl abzuschließen.

Ein Sprung in die Interrupt-Routine erfolgt unmittelbar und unabhängig von der aktuell abgearbeiteten Hauptprogrammzeile. Wird die Interrupt-Routine mit dem RET-Befehl abgeschlossen, erfolgt die weitere Programmausführung an der unterbrochenen Stelle (ein unterbrochener "wartender Befehl" wird weiter ausgeführt).

Der Befehl SETINT ist nur in Task 1 wirksam und die Ausführung von Task 1 ist während der Abarbeitung des Interrupts unterbrochen.

Es kann zeitgleich nur ein Interrupt abgearbeitet werden, jedoch kann ein Interrupt mit höherer Priorität die Abarbeitung eines anderen Interrupts unterbrechen. Dabei hat ERROR die höchste Priorität, dann TOUCH PROBE, dann TIMER 0.

Ein Interrupt muss nur einmal durch SETINT initialisiert werden.



515362827

Befehlsaufbau

Befehlsaufbau Mxxx SETINT X1, Myyy	Mxxx: Marke (optional)

X1:	
DISABLE	Deaktivierung des Interrupts, die Sprungmarke (Mxx) ist ohne Bedeutung.
ERROR	Löst einen Interrupt bei Gerätefehler aus. Die Interrupt-Routine wird so lange zyklisch durchlaufen und erst dann mit RET verlassen, wenn der Fehler gelöscht ist. Je nach eingestellter Fehlerreaktion (**Parametergruppe 830** oder Befehl **SETFR**) ergibt sich bei Abarbeitung der Interrupt-Routine abweichend von o.g. Beschreibung folgendes Verhalten: • Sind die Fehler der Parametergruppe 830 auf "keine Funktion" programmiert oder durch den Befehl SETFR = KEINE REAKTION eingestellt, so wird kein Interrupt ausgewertet. • Wird die Fehlerreaktion (**Parametergruppe 830** oder Befehl **SETFR**) auf "..., **FAULT**" gesetzt, so erfolgt ein Neustart des Programms (siehe Befehl SETFR), nachdem der Fehler quittiert wurde. Eine eventuell vorhandene Referenziert-Kennung geht verloren.
TIMER 0	Löst einen Interrupt bei Ablauf der im Timer 0 H489 eingestellten Zeit aus. Nach Ablauf des Timers 0 erfolgt ein "Autoreload" mit der Systemvariablen H485. Dieser Reload-Wert bestimmt die Zeit, mit der die Interrupt-Routine zyklisch ausgeführt wird.
TOUCH PROBE	Löst einen Interrupt bei Flankenwechsel an der Touch Probe-Klemme DI02 aus, wenn der Touch Probe für die Klemme DI02 aktiviert wurde (Parameter P601 = IPOS EIN-GANG) und der Befehl TOUCHP abgesetzt wurde.

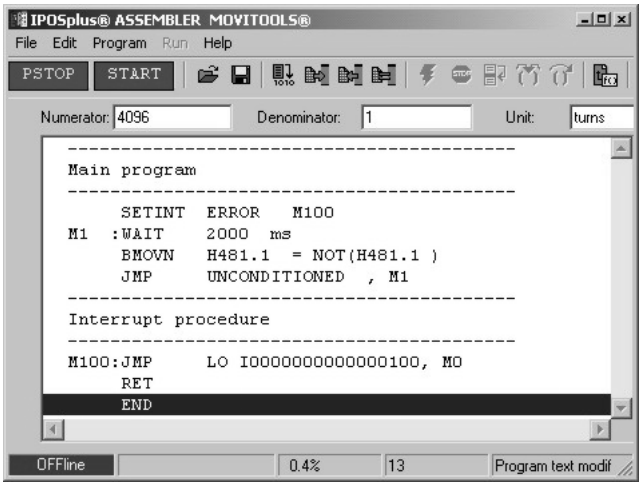
	Myyy: Startmarke der Interrupt-Routine.

Beispiel 1

Interrupt-Verzweigung bei Auftreten eines Gerätefehlers

Im Beispielprogramm wird jeweils nach 2 sec. Pausenzeit der Binäreingang DO01 getoggelt. Beim Auftreten eines Gerätefehlers wird umgehend in die Interrupt-Routine verzweigt. Der Rücksprung (RET) ins Hauptprogramm erfolgt, sobald ein "High"-Signal an Klemme DI02 anliegt. Zum Rücksetzen des Fehlers ist der Eingang DI02 auf "Reset" zu parametrieren.

Gerätefehler des MQX können den Interrupt auf ERROR auslösen. Gerätefehler von angeschlossenen MOVIMOT® können den Interrupt nicht auslösen.



515392779

Beispiel 2

Folgendes Beispiel soll die Wirkungsweise veranschaulichen:

```
M0  :SETINTERROR M01
 JMP UNCONDITIONED , M0
M01 :ADD H0 + 1
 RET
 END
```

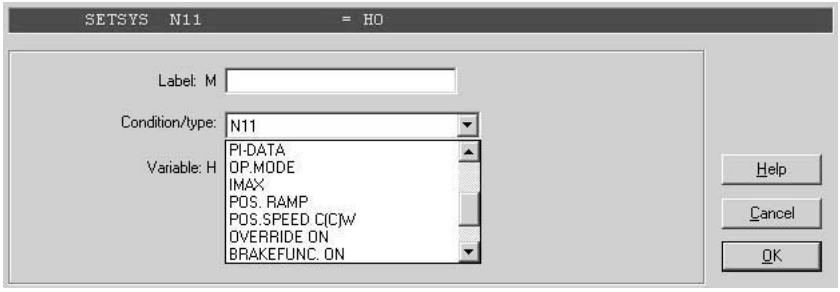
H0 zählt solange hoch, solange ein Gerätefehler ansteht. Nach dem Fehler-Reset hat H0 den Wert aus dem EEPROM, z. B. 0, der während des Fehlers hochgezählte Wert aus dem Arbeitsspeicher geht verloren.

23.8.4 SETSYS

Der Befehl SETSYS schreibt den Wert einer oder mehrerer Variablen in eine interne Systemgröße. Mit dem ersten Argument wird die zu beschreibende Systemgröße ausgewählt, im zweiten Argument steht die Nummer der (ersten) Quellvariablen.

Nach Abschalten der Anlage (Netz und 24 V) und erneutem Zuschalten werden die Systemgrößen wieder auf ihren ursprünglichen Wert zurückgesetzt.

	HINWEIS Mit dem Beschreiben von Systemgrößen ist es möglich, Geräteeinstellungen zu ändern, die bei der Inbetriebnahme auf die Applikation abgestimmt werden. Insbesondere Veränderungen von Positionierrampen und Maximalstrom müssen auf die Anlagengegebenheiten abgestimmt sein, um Beschädigungen und Gefahren (z.B. durch mechanische Überlastung) auszuschließen.



516336139

Befehlsaufbau

Befehlsaufbau Mxxx SETSYS X1, X2	Mxxx:	Marke (optional)

	X2:	Nummer der ersten Quellvariablen.

X1: wählbare Systemgrößen		
	Die internen Festsollwerte (Parametergruppe P160/ P170) können in 0,1 min^{-1}-Schritten über das IPOS$^{plus®}$-Programm verändert werden (ohne Reglersperre auch während des Verfahrens): Achtung: Der neue Festsollwert wird erst nach 5 ms übernommen, Programmabarbeitung nach SetSys evtl. mit Wait-Befehl (5 ms) verzögern. Wird für den Festsollwert ein Wert eingegeben, der den zulässigen Bereich überschreitet, wechselt das Vorzeichen.	
N11 =	interner Festsollwert n11	
N12 =	interner Festsollwert n12	
N13 =	interner Festsollwert n13	
N21 =	interner Festsollwert n21	
N22 =	interner Festsollwert n22	
N23 =	interner Festsollwert n23	
PI-DATA[1)]	Prozess-Eingangsdaten entsprechend Feldbus-Geräteprofil	
	H	Anzahl der PE-Daten
	H+1	PE-Daten 1
	H+2	PE-Daten 2
	H+3	PE Daten 3
OP. MODE	Setzen der Betriebsart. Es kann nur innerhalb eines Regelverfahrens (CFC oder SERVO) die Betriebsart geändert werden (ohne Reglersperre auch während des Verfahrens).	
	11	CFC (Drehzahlregelung)
	12	CFC & Momentenregelung
	13	CFC & IPOS (Positionierung)
	14	CFC & Synchronlauf (DRS11A)
	16	SERVO (Drehzahlregelung)
	17	SERVO & Momentenregelung
	18	SERVO & IPOS (Positionierung)
	19	SERVO & Synchronlauf (DRS11A)
IMAX	Einstellung des Maximalstroms (nur Parametersatz 1) in % bezogen auf den Gerätenennstrom (Einstellbereich: 0,1 ... 150 %, in 0,1 %-Schritten); Einstellung auch während des Verfahrens möglich.	
POS. RAMP	Positionierrampen (auf/ab), Einstellung auch während des Verfahrens möglich (nur bei Rampenform "linear"). Einstellung in ms mit Bezugswert 3000 min^{-1}.	
	H	Positionierrampe 1 (auf)
	H+1	Positionierrampe 2 (ab)
POS. SPEED	Positionierdrehzahl (rechts / links), Einstellung auch während des Verfahrens möglich (nur bei Rampenform "linear"). Einstellung in 0,1 min^{-1}.	
	H	Positionierdrehzahl rechts
	H+1	Positionierdrehzahl links
OVERRIDE ON	Override ein-/ ausschalten, Einstellung auch während des Verfahrens möglich (nur bei Rampenform "linear").	
	H = 0	aus
	H = 1	ein
BRAKE FUNC. ON	Bremsenfunktion ein-/ ausschalten	
	H = 0	aus
	H = 1	ein
RAMP TYPE	Einstellung während des Verfahrens nicht erlaubt (Drehmomentstöße!) (ändert P916)	

	H = 0 linear H = 1 sinus H = 2 quadratisch H = 3 Busrampe H = 4 ruckbegrenzt H = 5 Kurvenscheibe H = 6 I-Synchronlauf
RESET ERROR	Rücksetzen des Systemfehlers in Variable X2
ACT. POSITION	Setzen der Motorgeber-Istposition ACTPOS.MOT (H511)
SPLINE MULTIAXIS	Antriebsinterne Berechnung einer analytischen Kurvenscheibe. Die Funktion ist nur im MDX B mit SK-Version -15C verfügbar.
	Nach der Vorgabe von bis zu 20 Stützstellen (x-y-Wertepaaren, x = Masterposition, y = Slave-Position) in einem Leitgeberbereich wird über die Systemfunktion die Spline-Berechnung initialisiert. Danach wird über `h+0 SplineMode` die Berechnung gestartet und eine komplette oder ein Segment einer ausgewählten Kurvenscheibe gefüllt. Bisher sind Spline-0-Verfahren (für optimale Laufruhe) und Spline-1-Verfahren (für Rast-in-Rast-Bewegungen und Geradenstücke) verfügbar. Die Berechnung ist nach ≤ 200 ms abgeschlossen. – H+0 = SplineMode: (Wertebereich: 0 ... 3) • = 0: Interpolation nicht aktiv, bzw. Berechnung beendet • = 1: Interpolation starten, interpolierte Werte von Index 0 beginnend in die Kurvenscheibe eintragen (vorwärts, d. h. von Index 0 bis 512) • = 2: Interpolation starten, interpolierte Werte von Index 512 beginnend in die Kurvenscheibe eintragen (rückwärts, d. h. von Index 512 bis 0) • = 3: vorbereitende Parameterberechnung für die Interpolation abgeschlossen, Beginn des Eintragens der interpolierten Werte in die Kurvenscheibe – H+1 = SplineModeControl: reserviert – H+2 = SplineDest: (Wertebereich: 0 ... 5) Die Nummer der Kurvenscheibe in die die interpolierten Werte eingetragen werden sollen. – H+3 = SplineNUser: (Wertebereich: 2 ... 10) Die Anzahl der Stützstellen, die für die Interpolation verwendet werden sollen und das Berechnungsverfahren (Bit 7 = 0 Spline 0, Bit 7 = 1 Spline 1) – H+4 = SplineX0User: (Hier darf nur ein Wert > =0 eingetragen werden!) X-Wert der 1. Stützstelle – H+5 = SplineY0User: (Wertebereich: long = -2^{31} ... 0 ... (2^{31} -1)) Y-Wert (= Lagewert) der 1. Stützstelle – ... – H+42 = SplineX19User: (Hier darf nur ein Wert <= 512 eingetragen werden!) X-Wert der 20. Stützstelle – H+43 = SplineY19User: (Wertebereich: long = -2^{31} ... 0 ... (2^{31} -1)) Y-Wert der 20. Stützstelle
	SS_MULTIAXIS: antriebsübergreifende Berechnung einer Bewegungsbahn. Nur auf Anfrage erhältlich. Siehe auch Zusatz zur Betriebsanleitung „Sonderkonstruktion SK-0C für Berechnete Kurven“.

1) Gilt unter der Bedingung, dass für den Parameter P101 "RS485", "Feldbus" oder "SBus" eingestellt ist.

23.8.5 VARINT

Syntax

`VARINT Hxx, Mxx`

Beschreibung

Der Befehl ist im MOVIDRIVE® A nicht verfügbar, nur ab MOVIDRIVE® B.

Der Befehl aktiviert einen Variablen-Interrupt mit der Datenstruktur ab der Variable Hxx.

Ist die Bedingung für den Interrupt erfüllt und die Task 2 oder 3, in der dieser Interrupt bearbeit wird, gestartet, werden die Befehle ab der Marke Mxx ausgeführt. Das Ereignis für den Interrupt ist der Vergleich mit einem Variablenwert (siehe H+4). Wenn die Datenstruktur initialisiert ist, kann zur Laufzeit mit einem IPOS$^{plus®}$-Befehl das Interrupt-Verhalten eines kompletten VarInterrupts dynamisch angepasst werden.

Hinweis: Die Daten aus der Datenstruktur werden nur übernommen, wenn der Befehl `VARINT Hxx, Mxx` aufgerufen wird (Datenkonsistenz).

Beispiel: wird z. B der Wert aus der Datenstruktur Hx+3 CompareVar geändert, so wird der Wert erst mit dem Befehl `VARINT Hxx, Mxx` berücksichtigt.

Argumente

Hxx Erste Variable einer Datenstruktur (siehe Tabelle H+0)

Mxx Label mit dem ersten Befehl der Interrupt-Funktion.

Datenstruktur des Variablen-Interrupts:

Variable	Elemente Struktur VARINT	Beschreibung
H+0	Control	0: Alle VarInterrupt = AUS / Reset 1: Interrupt erhält Rechenzeit von Task 2 und unterbricht diese Task, solange der Interrupt bearbeitet wird. 2: Interrupt erhält Rechenzeit von Task 3 und unterbricht diese Task, solange der Interrupt bearbeitet wird.
H+1	IntNum	0 ... 3: Legt eine fortlaufende Nummer des VarInterrupt fest. Ein bereits aktivierter Interrupt mit der Nummer x kann während der Programmlaufzeit über den Befehlsaufruf `VarInt Hxx, Mxx` mit einer anderen Datenstruktur komplett neu aktiviert werden, wenn in der neuen Datenstruktur an der Stelle H+1 dieselbe Interrupt-Nummer angegeben ist. Diese Eigenschaft ist bei den Task 1-Interrupts nicht möglich.
H+2	SrcVar	Nummer der Bezugsvariablen, deren Wert mit dem Vergleichswert verglichen wird. ScrVar ist der Wert der Bezugsvariable auf die ScrVar zeigt.
H+3	CompVar	Vergleichswert oder Maske, mit der der Wert der Bezugsvariable H+2 verglichen wird.

Variable	Elemente Struktur VARINT	Beschreibung
H+4	Mode	0: Kein Interrupt-Event. Damit kann dieser einzelne Interrupt deaktiviert werden, ohne alle Interrupts abzuschalten. 1: Eines der Bits der Bezugsvariable, die mit der Maske CompVar ausmaskiert werden, hat seinen Zustand geändert: ([*SrcVar(t) ^ *SrcVar(t-T)] & CompVar) != 0 2:Solange Wert der Bezugsvariable gleich Vergleichswert (*SrcVar == CompVar) 3:Solange Wert der Bezugsvariable ungleich Vergleichswert (*SrcVar != CompVar) 4:Solange Wert der Bezugsvariable größer gleich Vergleichswert (*SrcVar > = CompVar) 5:Solange Wert der Bezugsvariable kleiner gleich Vergleichswert (*SrcVar <= CompVar) 6: Wert der Bezugsvariable logisch verundet Vergleichswert ungleich 0 ((*SrcVar & CompVar) != 0) 7: Wert der Bezugsvariable logisch verundet Vergleichswert gleich 0 ((*SrcVar & CompVar) == 0) 8: positive Flanke des über CompVar ausmaskierten Bits 9: negative Flanke des über CompVar ausmaskierten Bits 10: wie 2, jedoch wird Interrupt jedesmal nur ein Mal bearbeitet, wenn die Bedingung erfüllt wird (flankengetriggert) 11: wie 3, jedoch wird Interrupt jedesmal nur ein Mal bearbeitet, wenn die Bedingung erfüllt wird (flankengetriggert) 12: wie 4, jedoch wird Interrupt jedesmal nur ein Mal bearbeitet, wenn die Bedingung erfüllt wird (flankengetriggert) 13: wie 5, jedoch wird Interrupt jedesmal nur ein Mal bearbeitet, wenn die Bedingung erfüllt wird (flankengetriggert)
H+5	Priority	Priorität des Interrupts (1 ... 10), Task 2 und Task 3 haben jeweils Priorität 0.
H+6	IntEvent	Prozessabbild der Bezugsvariable von *SrcVar zum Interrupt-Zeitpunkt.

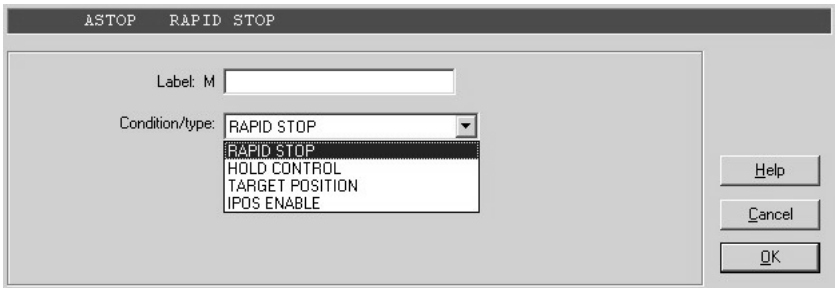
Beispiel Siehe "Task-Verwaltung und Interrupts / Variablen-Interrupts bei MOVIDRIVE® B".

23.9 Spezielle Gerätebefehle

23.9.1 ASTOP / MEM / TOUCHP / WDOFF / WDON

ASTOP / AXIS STOP

Mit dem ASTOP-Befehl wird der Antrieb stillgesetzt bzw. wieder freigegeben (siehe H484 Bit 1). Mit dem Argument des Befehls (RAPID STOP, HOLD CONTROL, TARGET POSITION) wird die Art des Stoppens (Rampe, Regelung im Stillstand usw.) festgelegt bzw. wieder freigegeben (IPOS ENABLE).



516341515

Befehlsaufbau

Befehlsaufbau Mxxx ASTOP X1	Mxxx: Marke (optional)

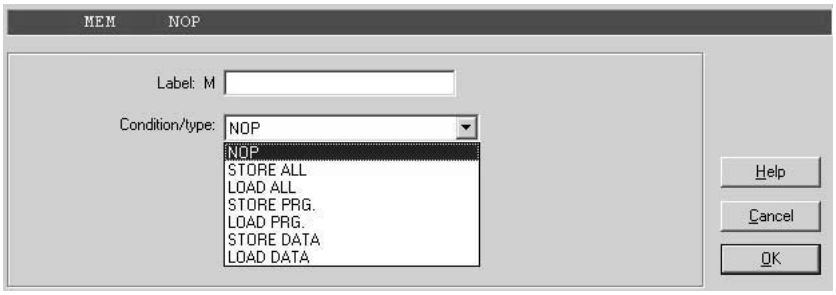
X1:	
RAPID STOP	Abbremsen an der Schnellstopp-Rampe, anschließend Drehzahlregelung. Die zuletzt abgesetzte Zielposition (H492) bleibt erhalten. Verriegelung über Steuerwort (Befehl ASTOP (IPOS ENABLE) vor anschließendem Verfahrbefehl notwendig). Bei aktivierter Bremsenfunktion fällt die Bremse ein.
HOLD CONTROL	Abbremsen an der Rampe des Grundgerätes (P131/P133), anschließend Lageregelung, die zuletzt abgesetzte Zielposition (H492) bleibt erhalten, Verriegelung über Steuerwort (Befehl ASTOP (IPOS ENABLE) bei anschließendem Verfahrbefehl notwendig). Bei aktivierter Bremsenfunktion fällt die Bremse **nicht** ein.
TARGET POSITION	Positionierstopp mit Positionierrampe (P911/P912) und berechneter "STOP"-Zielposition (nur in Betriebsart Positionierung möglich), anschließend Lageregelung. Die zuletzt abgesetzte Zielposition (H492) wird mit der Stopp-Position überschrieben, keine Verriegelung über Steuerwort (kein Befehl ASTOP (IPOS ENABLE) vor anschließendem Verfahrbefehl notwendig). Bei aktivierter Bremsenfunktion fällt die Bremse **nicht** ein. Hinweis: Da im Stillstand die Ist-Position als Sollposition übernommen wird, darf der Befehl nicht zyklisch bearbeitet werden. Bei Achsen mit Prozesskräften oder Hubwerken driftet sonst die Achse langsam weg.
IPOS ENABLE	Die Verriegelung mittels IPOS$^{plus®}$-Steuerwort wird aufgehoben.

MEM / MEMORIZE

Der MEM-Befehl ermöglicht das Speichern (Laden) von IPOS$^{plus®}$-Programmen und/ oder Variablen im (vom) nichtflüchtigen Speicher auf das Gerät. Die Aktion wird über das Argument angegeben.

	HINWEIS Beim Benutzen des MEM-Befehls ist darauf zu achten, dass die fest speicherbaren Variablen (H0...127) sowie alle Parameter nicht zyklisch beschrieben werden, da die Anzahl der Speichervorgänge beim verwendeten Speichermedium (EEPROM) auf 10^5 Speichervorgänge begrenzt ist.

Einzelne Variable können auch mit dem MOVILINK-Befehl abgespeichert werden.



516345867

Befehlsaufbau

Befehlsaufbau Mxxx MEM X1	Mxxx: Marke (optional)

X1:	
NOP	Es werden keine Daten gespeichert.
STORE ALL	Speichern von Programmen und Daten vom Arbeitsspeicher in den permanenten Speicher (EEPROM).
LOAD ALL	Laden von Programmen und Daten vom permanenten Speicher (EEPROM) in den Arbeitsspeicher.
STORE PROG.	Nur Programm vom Arbeitsspeicher in den permanenten Speicher (EEPROM) speichern.
LOAD PROG.	Nur Programm vom permanenten Speicher (EEPROM) in den Arbeitsspeicher laden.
STORE DATA	Nur Variablen vom Arbeitsspeicher in den permanenten Speicher (EEPROM) speichern.
LOAD DATA	Nur Variablen vom permanenten Speicher (EEPROM) in den Arbeitsspeicher laden.

Beispiel

Bei einem Fehler wird in eine Fehler-Interrupt-Routine gesprungen. Dort wird der Befehl MEM STORE DATA aufgerufen um nach dem Fehler-Reset mit den abgespeicherten Zwischenständen der Variablen H0 ... H127 weiter zu arbeiten.

Ohne den Befehl startet das Programm mit den letzten Werten aus dem EEPROM und überschreibt die letzten Arbeitswerte.

TOUCHP / TOUCH PROBE

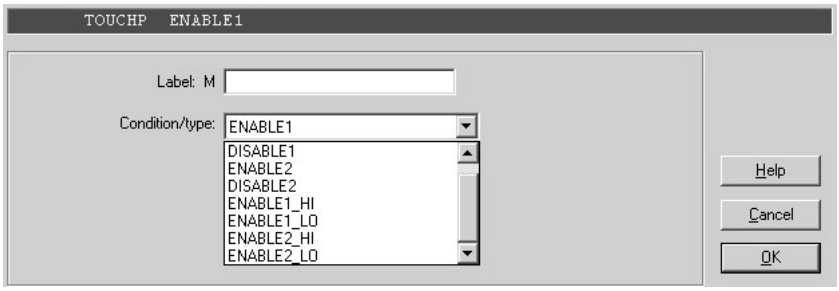
Der Befehl **TOUCHP** gibt einen Touch-Probe-Eingang frei bzw. sperrt diesen. Die Touch-Probe-Funktion ist generell den Eingangsklemmen DI02 und / oder DI03 zugeordnet. Benutzte Eingänge für die Touch-Probe-Funktion sollten auf "IPOS-Eingang" parametriert werden, um eine Doppelbelegung zu vermeiden.

Tritt an einem Touch-Probe-Eingang nach Freigabe durch den TOUCHP-Befehl ein Flankenwechsel auf, so werden einmalig die aktuellen Ist-Positionen (H511, H510, H509) in die dafür bestimmten Variablen (H502 ... H507) abgelegt.

MQX und MOVITRAC® 07 besitzen nur den Zähler mit Variable H511. Für eine erneute Messung muss der Touch-Probe erneut freigegeben werden.

Der Speichervorgang für die Touch-Probe-Positionen erfolgt, unabhängig von der aktuellen Programmabarbeitung, innerhalb von 100 µs. Die Änderung des Klemmenpegels muss für mindestens 200 µs anstehen, um sicher erkannt werden zu können.

Mit dem Argument kann der Flankenwechsel, der zum Touch-Probe führt, ausgewählt werden.



516734219

Die Touch-Probe-Positionen werden in den nachstehenden Variablen abgelegt.			
Geber	Geberposition	Position Touch-Probe 1 (DI02)	Position Touch-Probe 2 (DI03)
Motorgeber (X15)	H511 ACTPOS. MOT	H507 TP.POS1MOT	H505 TP.POS2MOT
Externer Geber (X14)	H510 ACTPOS.EXT	H506 TP.POS1EXT	H504 TP.POS2EXT
Absolutwertgeber (X62)	H509 ACTPOS.ABS	H503 TP.POS1ABS	H502 TP.POS2ABS
Virtueller Geber (nur bei MOVIDRIVE® B)	H376	H501 TpPos1_VE	H500 TpPos2_VE

Befehlsaufbau

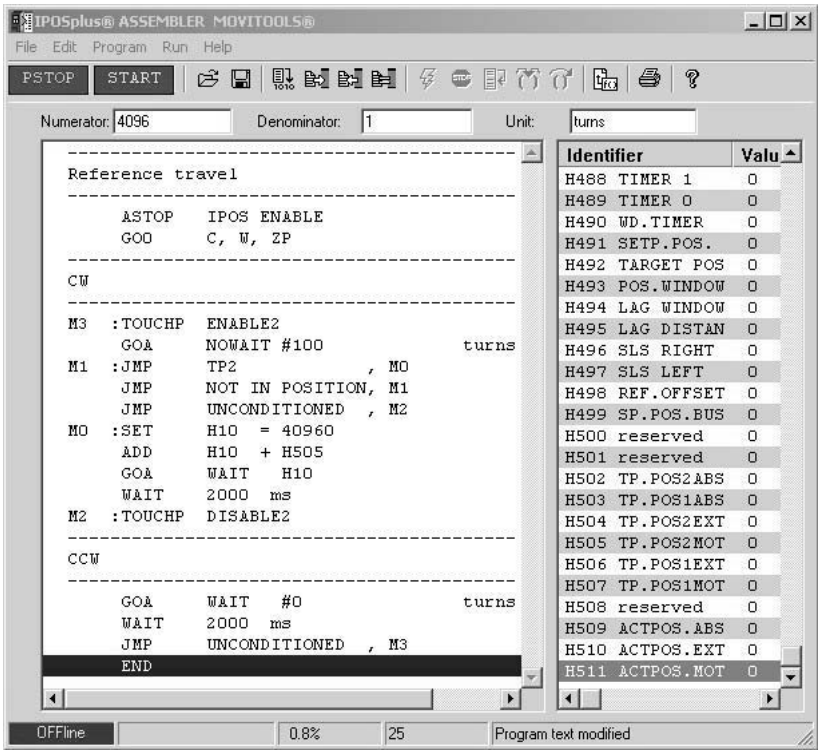
Befehlsaufbau Mxxx TOUCHP X1	Mxxx: Marke (optional)

X1:	
ENABLE 1	Freigabe des Touch-Probe-Eingangs DI02. Bei Wechsel Low/High und High/Low werden die Ist-Positionen gespeichert.
DISABLE 1	Sperren des Touch-Probe-Eingangs DI02
ENABLE 2	Freigabe des Touch-Probe-Eingangs DI03. Bei Wechsel Low/High und High/Low werden die Ist-Positionen gespeichert.
DISABLE 2	Sperren des Touch-Probe-Eingangs DI03
ENABLE 1_HI	Freigabe des Touch-Probe-Eingangs DI02. Bei Wechsel Low/High werden die Ist-Positionen gespeichert.
ENABLE 1_LO	Freigabe des Touch-Probe-Eingangs DI02. Bei Wechsel High/Low werden die Ist-Positionen gespeichert.
ENABLE 2_HI	Freigabe des Touch-Probe-Eingangs DI03. Bei Wechsel Low/High werden die Ist-Positionen gespeichert.
ENABLE 2_LO	Freigabe des Touch-Probe-Eingangs DI03. Bei Wechsel High/Low werden die Ist-Positionen gespeichert.

Die Auswertung, ob ein Touch-Probe-Eingang betätigt wurde, kann entweder im Programm z. B. durch einen JMP TP2, M0 oder durch SETINT TOUCHP1 M0 erfolgen. Die Auswertung, ob ein abgespeicherter Positionswert in einem Positionsbereich liegt (Markenausblendung) kann mit Vergleichen im nachfolgenden Anwenderprogramm erfolgen.

Beispiel 1

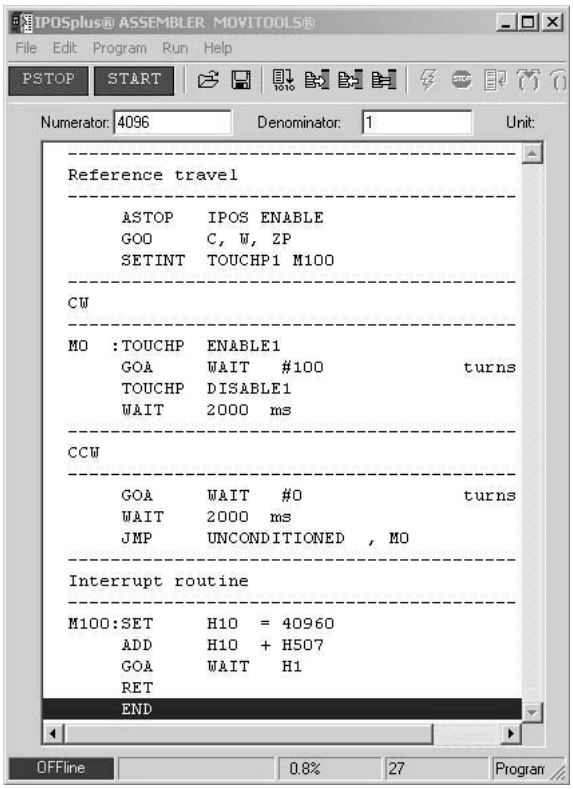
Im Programm wird zwischen den Absolutpositionen 0 Umdr. und 100 Umdr. verfahren. Erfolgt beim Verfahren auf die Zielposition 100 Umdr. ein Flankenwechsel am Touch-Probe-Eingang DI03, so wird exakt ab dieser Touch-Probe-Position noch weitere 10 Umdr. verfahren (40960 Inkr.). Für die Rückfahrt auf Position 0 Umdr. wird die Touch-Probe-Funktion mit dem Befehl DISABLE2 deaktiviert.



516738571

Beispiel 2

Alternativ zum obigen Beispiel kann bei Erreichen der Touch-Probe-Position eine Programmverzweigung (Sprungmarke M100) durch den Befehl "SETINT TOUCHP1 M100" erfolgen.



516742923

WDOFF / WDON / WATCH DOG OFF / ON

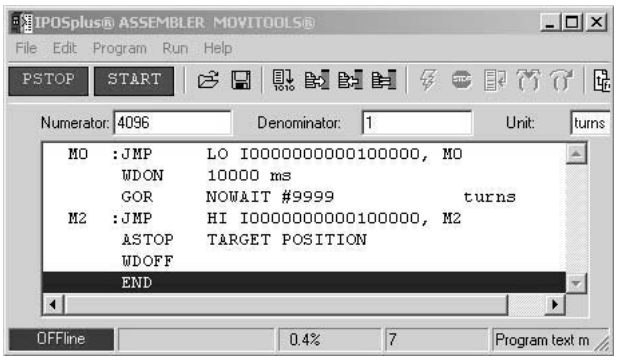
Der Watchdog wird in den im Argument angegebenen Zeitintervallen aufgerufen. Läuft der Watchdog-Timer H490 über, bevor die Überwachungsfunktion mit WDOFF abgeschaltet wird, so werden alle Tasks angehalten und der Antrieb mit Fehler 41 stillgesetzt (Endstufe wird gesperrt und die Bremse fällt ein. Falls keine Bremse vorhanden, läuft der Antrieb aus.).

Befehlsaufbau

Befehlsaufbau		
Mxxx WDON X1 Mxxx WDOFF	Mxxx: X1:	Marke (optional) Intervall in ms, in dem der Watchdog aufgerufen wird.

Beispiel

Der Antrieb wird so lange verfahren wie der Pegel an DI05 = 1 ("high") ist. Mit der "Watchdog"-Funktion wird sichergestellt, dass der Antrieb nicht länger wie 10 sec. verfährt. Bei Überschreiten von 10 sec. wird der Antrieb stillgesetzt.



516804619

23.10 Vergleichsbefehle

23.10.1 Vergleichsoperationen CPEQ / CPGE / CPGT / CPLE / CPLT / CPNE

Eine Variable wird mit einem 2. Argument (Variable oder Konstante) verglichen, wobei folgende Vergleiche möglich sind:

- gleich (CPEQ)
- größer oder gleich (CPGE)
- größer (CPGT)
- kleiner oder gleich (CPLE)
- kleiner (CPLT)
- ungleich (CPNE)

Das Ergebnis kann durch einen nachfolgenden Sprungbefehl verwertet werden.

CPEQ / COMPARE EQUAL

Der Befehl CPEQ vergleicht vorzeichenrichtig, ob Variable X1 gleich Variable oder Konstante X2 ist. Variable X1 enthält das Ergebnis. Es ist ungleich Null, wenn die Bedingung erfüllt ist, sonst ist das Ergebnis Null.

Das Ergebnis kann z. B. durch einen nachfolgenden Sprungbefehl weiter verarbeitet werden. Variable X2 bleibt unverändert.

Befehlsaufbau

Befehlsaufbau		
Mxxx CPEQ X1 == X2	Mxxx:	Marke (optional)
	X1:	Variable (Ergebnis)
	X2:	Variable oder Konstante

Beispiel 1

```
SET  H0  = 13
SET  H1  = 50
CPEQ H0  == H1
```

Nach der Abarbeitung des Programms hat H0 den Wert Null und H1 den Wert 50.

Beispiel 2

```
SET  H0  = 13
CPEQ H0  == 13
```

Nach der Abarbeitung des Programms hat H0 den Wert Eins.

CPGE / COMPARE GREATER OR EQUAL

Der Befehl CPGE vergleicht vorzeichenrichtig, ob Variable X1 größer oder gleich Variable oder Konstante X2 ist. Variable X1 enthält das Ergebnis. Es ist ungleich Null, wenn die Bedingung erfüllt ist, sonst ist das Ergebnis Null.

Das Ergebnis kann z. B. durch einen nachfolgenden Sprungbefehl weiter verarbeitet werden. Variable X2 bleibt unverändert.

Befehlsaufbau

Befehlsaufbau		
Mxxx CPGE X1 > = X2	Mxxx:	Marke (optional)
	X1:	Variable (Ergebnis)
	X2:	Variable oder Konstante

Beispiel 1

```
SET  H0  = 13
SET  H1  = 50
CPGE H0  > = H1
```

Nach der Abarbeitung des Programms hat H0 den Wert Null und H1 den Wert 50.

Beispiel 2

```
SET  H0  = -3
CPGE H0  > = -3
```

Nach der Abarbeitung des Programms hat H0 den Wert Eins.

CPGT / COMPARE GREATER THAN

Der Befehl CPGT vergleicht vorzeichenrichtig, ob Variable X1 größer Variable oder Konstante X2 ist. Variable X1 enthält das Ergebnis. Es ist ungleich Null, wenn die Bedingung erfüllt ist, sonst ist das Ergebnis Null.

Das Ergebnis kann z. B. durch einen nachfolgenden Sprungbefehl weiter verarbeitet werden. Variable X2 bleibt unverändert.

Befehlsaufbau

Befehlsaufbau		
Mxxx CPGT X1 > X2	Mxxx:	Marke (optional)
	X1:	Variable (Ergebnis)
	X2:	Variable oder Konstante

Beispiel 1

```
SET  H0  = -3
CPGT H0  >  -3
```

Nach der Abarbeitung des Programms hat H0 den Wert Null.

Beispiel 2

```
SET  H0  = 3
SET  H2  = 2
CPGT H0  >  H2
```

Nach der Abarbeitung des Programms hat H0 den Wert Eins.

CPLE / COMPARE LESS OR EQUAL

Der Befehl CPLE vergleicht vorzeichenrichtig, ob Variable X1 kleiner oder gleich Variable oder Konstante X2 ist. Variable X1 enthält das Ergebnis. Es ist ungleich Null, wenn die Bedingung erfüllt ist, sonst ist das Ergebnis Null.

Das Ergebnis kann z. B. durch einen nachfolgenden Sprungbefehl weiter verarbeitet werden. Variable X2 bleibt unverändert.

Befehlsaufbau

Befehlsaufbau		
Mxxx CPLE X1 <= X2	Mxxx:	Marke (optional)
	X1:	Variable (Ergebnis)
	X2:	Variable oder Konstante

Beispiel 1

```
SET  H0  = 50
SET  H1  = 13
CPLE H0  <= H1
```

Nach der Abarbeitung des Programms hat H0 den Wert Null und H1 den Wert 13.

Beispiel 2

```
SET  H0  = -3
CPLE H0  <= -3
```

Nach der Abarbeitung des Programms hat H0 den Wert Eins.

CPLT / COMPARE LESS THAN

Der Befehl CPLT vergleicht vorzeichenrichtig, ob Variable X1 kleiner Variable oder Konstante X2 ist. Variable X1 enthält das Ergebnis. Es ist ungleich Null, wenn die Bedingung erfüllt ist, sonst ist das Ergebnis Null.

Das Ergebnis kann z. B. durch einen nachfolgenden Sprungbefehl weiter verarbeitet werden. Variable X2 bleibt unverändert.

Befehlsaufbau

Befehlsaufbau		
Mxxx CPLT X1 < X2	Mxxx:	Marke (optional)
	X1:	Variable (Ergebnis)
	X2:	Variable oder Konstante

Beispiel 1

```
SET  H0  = -3
CPLT H0  < -3
```

Nach der Abarbeitung des Programms hat H0 den Wert Null.

Beispiel 2

```
SET  H0  = 2
SET  H2  = 3
CPLT H0  < H2
```

Nach der Abarbeitung des Programms hat H0 den Wert Eins.

CPNE / COMPARE NOT EQUAL

Der Befehl CPNE vergleicht vorzeichenrichtig, ob Variable X1 ungleich Variable oder Konstante X2 ist. Variable X1 enthält das Ergebnis. Es ist ungleich Null, wenn die Bedingung erfüllt ist, sonst ist das Ergebnis Null.

Das Ergebnis kann z. B. durch einen nachfolgenden Sprungbefehl weiter verarbeitet werden. Variable X2 bleibt unverändert.

Befehlsaufbau

Befehlsaufbau		
Mxxx CPNE X1 != X2	Mxxx:	Marke (optional)
	X1:	Variable (Ergebnis)
	X2:	Variable oder Konstante

Beispiel 1

```
SET  H0  = 13
SET  H1  = 13
CPNE H0  != H1
```

Nach der Abarbeitung des Programms hat H0 den Wert Null und H1 den Wert 13.

Beispiel 2

```
SET  H0  = 50
CPNE H0  != 13
```

Nach der Abarbeitung des Programms hat H0 den Wert Eins.

23.10.2 Logische Operationen ANDL / ORL / NOTL

ANDL / LOGICAL AND

Der Befehl ANDL ist die logische UND-Verknüpfung zweier Variablen. Das Ergebnis steht in Variable X1. Variable X2 bleibt unverändert. Das Ergebnis ist Null, wenn eine der beiden Variablen = 0 ist. Das Ergebnis ist Eins, wenn beide Variablen != 0 sind.

Befehlsaufbau

Befehlsaufbau		
Mxxx ANDL X1 && X2	Mxxx:	Marke (optional)
	X1:	Variable (Ergebnis)
	X2:	Variable

Beispiel 1

```
SET  H01  = 100
SET  H02  = 0
ANDL H01  && H02
```

Nach der Abarbeitung des Programms hat H01 den Wert Null.

Beispiel 2

```
SET  H01  = 100
SET  H02  = 50
ANDL H01  && H02
```

Nach der Abarbeitung des Programms hat H01 den Wert Eins.

ORL / LOGICAL OR

Der Befehl ORL ist die logische ODER-Verknüpfung zweier Variablen. Das Ergebnis steht in Variable X1. Variable X2 bleibt unverändert. Das Ergebnis ist Eins, wenn eine der beiden Variablen != 0 ist. Das Ergebnis ist Null, wenn beide Variablen = 0 sind.

Befehlsaufbau

Befehlsaufbau		
Mxxx ORL X1 ¦¦ X2	Mxxx:	Marke (optional)
	X1:	Variable (Ergebnis)
	X2:	Variable

Beispiel 1

```
SET  H01  = 100
SET  H02  = 0
ORL  H01  ¦¦ H02
```

Nach der Abarbeitung des Programms hat H01 den Wert Eins.

Beispiel 2

```
SET  H01  = 0
SET  H02  = 0
ORL H01  ¦¦ H02
```

Nach der Abarbeitung des Programms hat H01 den Wert Null.

NOTL / LOGICAL NOT

Der Befehl NOTL führt die logische Negation der Variablen durch. Das Ergebnis steht in Variable X1. Variable X2 bleibt unverändert. Das Ergebnis ist Eins, wenn Variable X2 = 0 ist. Das Ergebnis ist Null, wenn Variablen X2 != 0 ist.

Befehlsaufbau

Befehlsaufbau		
Mxxx NOTL X1 = NOT (X2)	Mxxx:	Marke (optional)
	X1:	Variable (Ergebnis)
	X2:	Variable

Beispiel 1

```
SET H02  = 100
NOTL  H01  NOT (H02)
```

Nach der Abarbeitung des Programms hat H01 den Wert Null.

Beispiel 2

```
SET H02  = 0
NOTL  H01  NOT (H02)
```

Nach der Abarbeitung des Programms hat H01 den Wert Eins.

24 Assembler – Beispiele

24.1 Beispielprogramm "Blinklicht"

24.1.1 Beispiel "Steuerung"

Mit dem Beispielprogramm soll der Binärausgang DOØ1 alle 2 s abwechselnd ein- und ausgeschaltet werden.

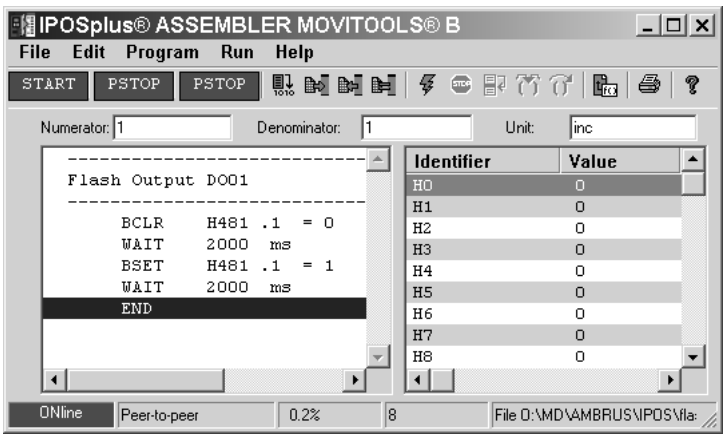
Kurzinbetriebnahme (Beispiel)

Voraussetzungen

Netzanschluss und/oder 24-V-Versorgung (Stützspannung Klemmen X10:9 (+ 24 V / VI24) und X10:10 (0 V / DGND)) angeschlossen; Anschluss von Motor und Geber ist nicht nötig (keine Motorbewegung).

1. Inbetriebnahme der Drehzahlregelung nicht notwendig.
2. Parametrieren des Ausgangs in Shell (P621 Binärausgang DOØ1 → IPOS-AUSGANG).
3. Starten des Assemblers
4. Öffnen / Aktivieren des Fensters "Programm" und Eingabe des Beispielprogramms "Blinken Ausgang DOØ1".
5. Download des Beispielprogramms vom Programmfenster (PC) in den Programmspeicher des Umrichters: "Strg + F9" drücken im aktiven Programmfenster
6. Starten des Beispielprogramms: "F9" drücken im aktiven Programmfenster.
7. Überprüfen des Anwenderprogrammes:
   - Im Programmkopf wechselt die Anzeige Task 1 von PSTOP auf START.
   - Der Programmzeiger im Programmfenster läuft.
   - Im Shell wechselt der Anzeigeparameter P052 der Ausgangsklemme DOØ1 alle zwei Sekunden zwischen 1 und 0.

Programmfenster Assembler:



516819723

Das Programm besteht aus:
3 Kommentarzeilen (zwei gestrichelte Linien zur optischen Hervorhebung des Programmnamens und eine Zeile für den Programmnamen)
Setzen des Ausgangs DO02 (X13:3) auf "0"
Zwei Sekunden warten
Setzen des Ausgangs DO02 (X13:3) auf "1"
Zwei Sekunden warten
Programmende / Sprung zum Programmanfang

24.1.2 Beispiel "Positionierung"

Mit dem Beispielprogramm soll der Antrieb alle 2 s 10 Motorumdrehungen abwechselnd rechts und links positioniert werden.

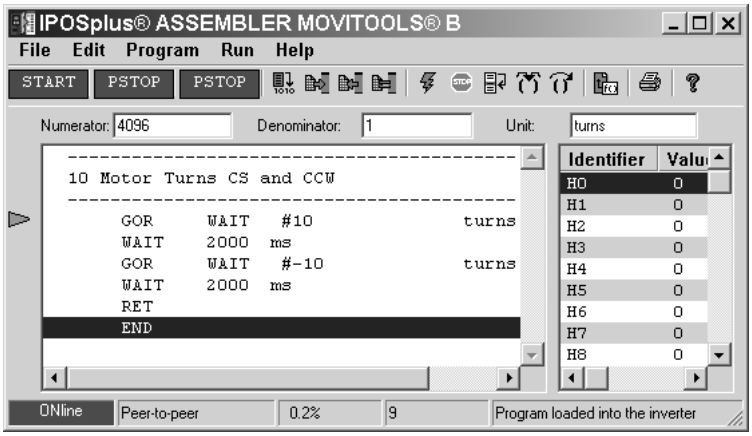
Kurzinbetrieb-nahme (Beispiel)

Voraussetzungen

- Umrichter / Motor / Geber angeschlossen
- Umrichter entsprechend des Systemhandbuchs MOVIDRIVE® in der Betriebsart VFC-n-Reg. & IPOS, CFC & IPOS oder SERVO & IPOS (P700) in Betrieb nehmen. P700 muss nach der Inbetriebnahme eine der genannten Einstellungen aufweisen.
- Überprüfen der Hardware-Endschalter des NOT-AUS-Kreises.

1. Parametereinstellung:
   - P600 Binäreingang DI01 →FREIGABE / STOP
   - P601 Binäreingang DI02 →KEINE FUNKTION
   - P602 Binäreingang DI03 →KEINE FUNKTION
   - P603 Binäreingang DI04 →/ES RECHTS
   - P604 Binäreingang DI05 →/ES LINKS
   - P700 Betriebsart →(VFC-n-Reg. / CFC / SERVO) & IPOS
   - ZÄHLER →4096
   - NENNER →1
   - EINHEIT→Umdr.
2. Beispielprogramm "10 Motorumdrehungen Vor und Zurück" eingeben.
3. Download des Beispielprogramms: "F2" drücken im aktiven Programmfenster
4. Endschalter dürfen nicht angefahren sein. Die Klemmen DIØ4 (X13:5) und DIØ5 (X13:6) müssen "1"-Pegel haben.
5. Starten des Beispielprogramms: "F9" drücken im aktiven Programmfenster
6. Überprüfen des Beispielprogramms:
   - Im Programmkopf wechselt die Anzeige Task 1 von PSTOP auf START.
   - Motor verfährt alle zwei Sekunden 10 Umdrehungen abwechselnd nach rechts und links.
   - Die Positionsänderung kann im Anzeigeparameter P003 verfolgt werden.
   - In den Variablen H492 und H511 werden Positions-Sollwert und Positions-Istwert angezeigt.

Beispielprogramm "10 Motorumdrehungen vor und zurück":



516923787

Das Programm besteht aus:
3 Kommentarzeilen
Verfahre relativ 10 Motorumdrehungen rechts
Zwei Sekunden warten
Verfahre relativ 10 Motorumdrehungen links
Zwei Sekunden warten
Ende Programm
Programmende / Sprung zum Programmanfang

Der RET-Befehl ist in diesem Beispiel nicht zwingend erforderlich, da das Programm nicht als Unterprogramm aufgerufen wurde. Der Rücksprung erfolgt dann auf die erste Programmzeile, was hier zulässig ist.

24.2 Beispielprogramm "Hubwerk"

24.2.1 Eigenschaften

- Referenzfahrt
- Anwahl von drei Hubwerkspositionen über binäre Eingänge
- Meldung bei Erreichen einer angewählten Position
- automatisches Herausfahren aus Hardware-Endschalter

Mit den ersten 3 Eingangsklemmen der Option DIO11B können 3 Positionen angefahren werden.

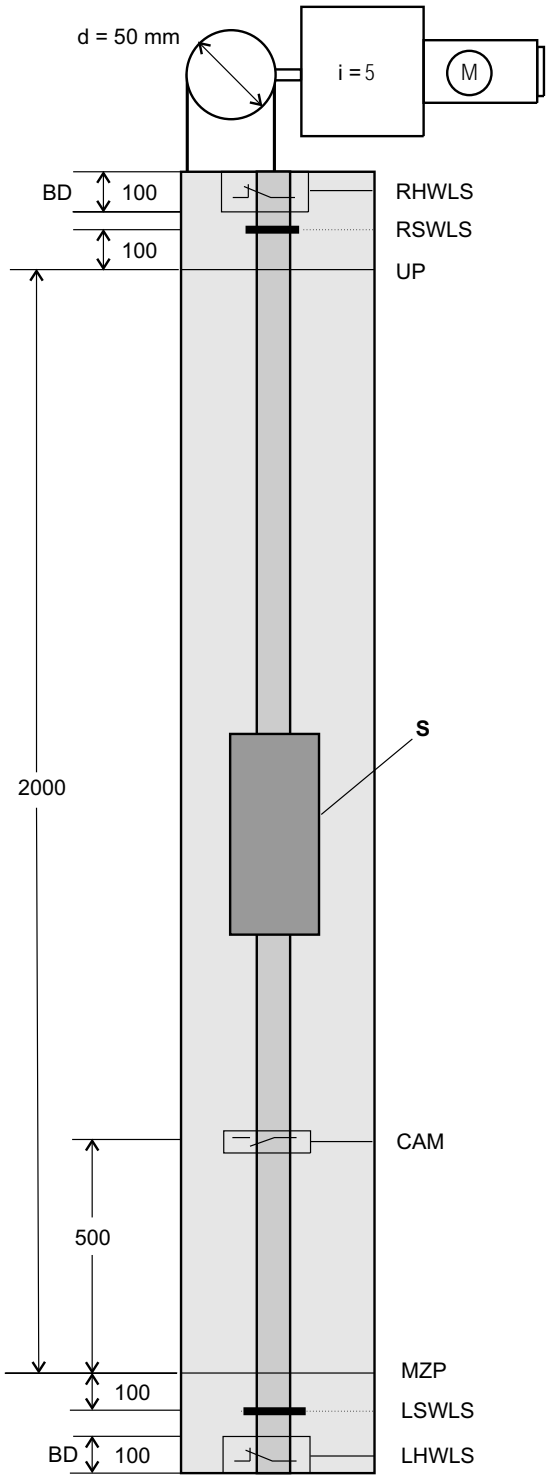
Aus einem angefahrenen Hardware-Endschalter wird durch ein "1"-Signal am "RESET" Eingang (DI02) herausgefahren.

24.2.2 Einstellungen

Die detaillierte Konfiguration der Ein-/Ausgänge ist im Kommentarteil des Programmquellcodes dokumentiert.

24.2.3 Schematischer Aufbau

Schematischer Aufbau Hubwerk mit IPOS$^{plus®}$:

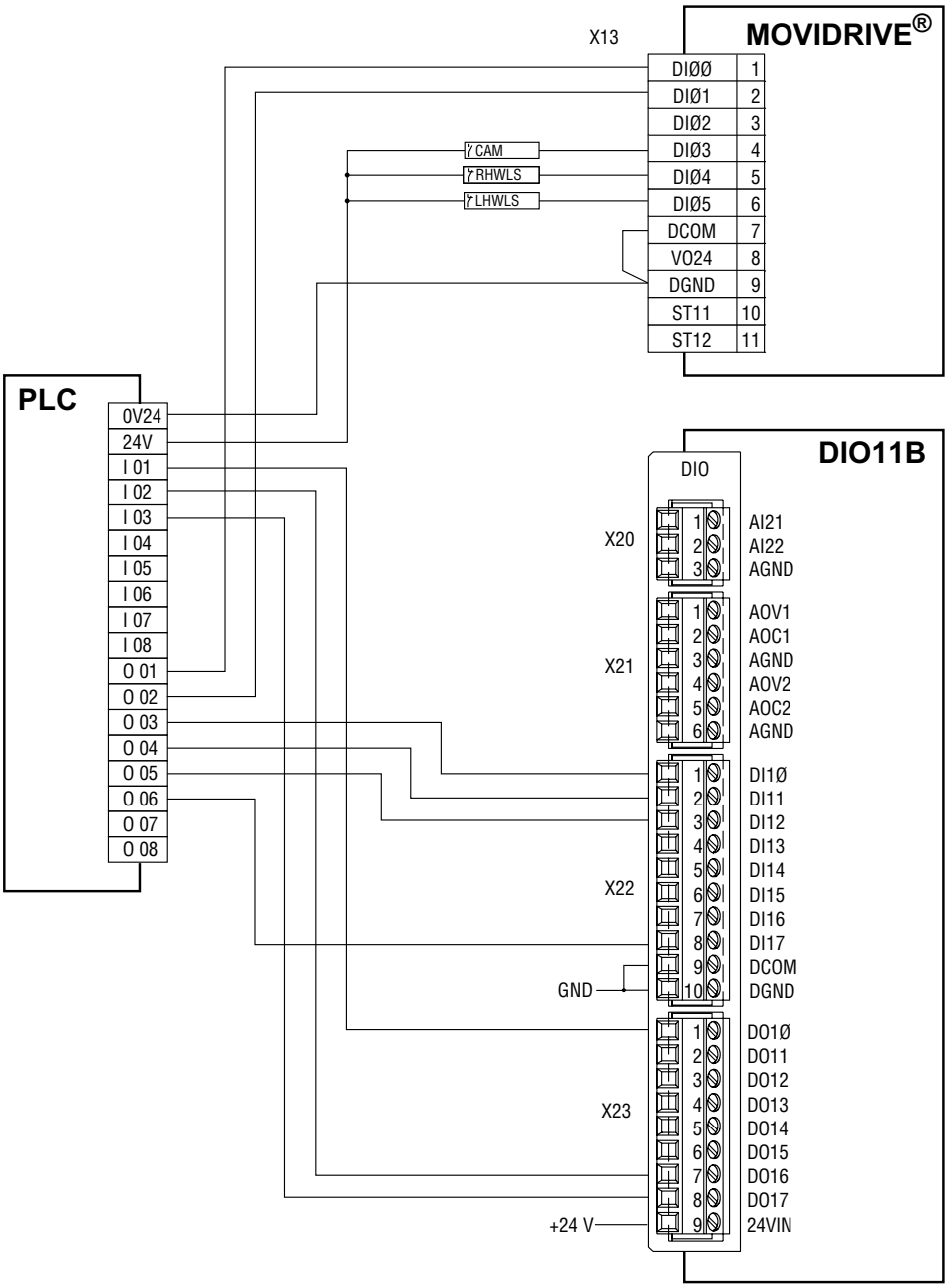


516940939

BD =	Bremsstrecke	CAM =	Referenznocken
RHWLS =	Rechter Hardware-Endschalter	MZP =	Maschinennullpunkt
RSWLS =	Rechter Software-Endschalter	LSWLS =	Linker Software-Endschalter
UP =	Oberer Verfahrbereich	LHWLS =	Linker Hardware-Endschalter
S =	Verfahrschlitten		

24.2.4 Beschaltung der Klemmen

Anschluss-Schaltbild IPOS$^{plus®}$:



516945803

PLC = externe Steuerung
DI00 = /Reglersperre
DI01 = Freigabe
DI02 = Reset
DI03 = Referenznocken
DI04 = /Endschalter Rechts
DI05 = /Endschalter Links
DI10 = Position 1
DI11 = Position 2
DI12 = Position 3
DI16 = Start Referenzfahrt
DI17 = Start Positionierung
DO10 =/Störung
DO16 =IPOS IN POSITION
DO17 =IPOS REFERENZ

24.2.5 Einstellung der für das Beispiel relevanten Parameter

Gruppe	Parameter	Einstellung
30_ Begrenzungen	P302 Maximaldrehzahl 1 [1/min] P350 Drehrichtungsumkehr	1500 AUS
60_ Binäreingänge Grundgerät	P600 Binäreingang DI01 P601 Binäreingang DI02 P602 Binäreingang DI03 P603 Binäreingang DI04 P604 Binäreingang DI05	FREIGABE / STOP RESET REFERENZNOCKEN /ES RECHTS /ES LINKS
61_ Binäreingänge Option DIO11	P610 Binäreingang DI10 ... P617 Binäreingang DI17	IPOS-EINGANG ... IPOS-EINGANG
63_ Binärausgänge Option DIO11	P630 Binärausgang DO10 P636 Binärausgang DO16 P637 Binärausgang DO17	/STÖRUNG IPOS IN POSITION IPOS REFERENZ
7__ Steuerfunktionen	P700 Betriebsart P730 Bremsenfunktion	CFC & IPOS JA
9__ IPOS Parameter	P900 Referenz-Offset [mm] P901 Referenzdrehzahl 1 [1/min] P902 Referenzdrehzahl 2 [1/min] P903 Referenzfahrttyp P910 Verstärkung X-Regler P911 Positionier-Rampe 1 [s] P912 Positionier-Rampe 2 [s] P913 Verfahrdrehzahl RECHTS [1/min] P914 Verfahrdrehzahl LINKS [1/min] P915 Geschwindigkeitsvorsteuerung [%] P916 Rampenform P920 SW-Endschalter RECHTS [mm] P921 SW-Endschalter LINKS [mm] P922 Positionierfenster [inc] P923 Schleppfehlerfenster [inc] P930 Override	500 200 50 1 2.8 1 1 1350 1350 100 SINUS 2100 -100 50 5000 AUS
Wegfaktor ZÄHLER / NENNER	Wegfaktor ZÄHLER Wegfaktor NENNER Einheit	2048000 15708 mm

24.2.6 Berechnung der IPOS$^{plus®}$-Parameter

SW-Endschalter siehe schematischer Aufbau

Wegfaktor Zähler Die Wegeinheit soll in mm festgelegt werden!

Anzahl der Inkremente pro Umdr. des Antriebsrads

Inkr./ Mot.-Umdr. × Getr.-Untersetzung

4096 Inkr. × 5 = 20480

20480 × 100 (Erweiterungsfaktor) =2048000

Wegfaktor Nenner Umfang des Antriebrads in mm

d × π

50 mm × π =157,0796327

157,08 × 100 (Erweiterungsfaktor) =15708

Einheit Die Einheit hinter den verfahrspezifischen Angaben soll in mm erscheinen.

Verfahrdrehzahl 1350 1/min

Positionierfenster Bei Erreichen der Zielposition ± 50 Inkrementen soll Meldung Antrieb in Position kommen.

24.2.7 Eingangsklemmen

Pegel	Klemme	Klemmenfunktion	Bedeutung
0	DI00	/Reglersperre	Zu-/Abschalten des Leistungsteils
0	DI01	Freigabe	geregelter Stillstand
0	DI02	Reset	Reset nach Fehler (Endschalter freifahren)
0	DI03	Referenznocken	Schalter für Nullstellung oder Offsetwert
0	DI04	Endschalter rechts	Begrenzungsschalter zum Stillsetzen (+)
0	DI05	Endschalter links	Begrenzungsschalter zum Stillsetzen (-)
0	DI10	IPOS Eingang	Hubwerksposition 0 mm
0	DI11	IPOS Eingang	Hubwerksposition 1000 mm
0	DI12	IPOS Eingang	Hubwerksposition 2000 mm
0	DI13	IPOS Eingang	-
0	DI14	IPOS Eingang	Tippen positiv
0	DI15	IPOS Eingang	Tippen negativ
0	DI16	IPOS Eingang	Start Referenzfahrt
0	DI17	IPOS Eingang	Start Positionierung

24.2.8 Ausgangsklemmen

Pegel	Klemme	Gerät	Klemmenfunktion	Bedeutung
0	DB00	MDX	/Bremse	Ansteuerung der Bremse über Hilfsrelais
0	DO01	MDX	Betriebsbereit	Controller aktiv, Elektronikversorgung OK
0	DO02	MDX	/Störung	kein Fehler vorhanden
0	DO10	DIO11B	IPOS Ausgang	-
0	DO11	DIO11B	IPOS Ausgang	-
0	DO12	DIO11B	IPOS Ausgang	-
0	DO13	DIO11B	IPOS Ausgang	-
0	DO14	DIO11B	IPOS Ausgang	-
0	DO15	DIO11B	IPOS Ausgang	-
0	DO16	DIO11B	IPOS in Position	Antrieb befindet sich im Positionsfenster
0	DO17	DIO11B	IPOS Referenz	Referenzfahrt erfolgreich durchgeführt

24.2.9 Programmquellcode (mit Kommentaren)

ZÄHLER: 2048000 NENNER: 15708 EINHEIT: mm ********** Programm: Hubwerk Mit den ersten 3 Eingängen der Option DIO11A werden die Position 0;1000;2000 mm angefahren. Datei: Hub 100.mdx Ersteller: SEW/AWT Datum: 01.06.98 Geändert: 01.06.98 Klemmenbelegung Eingänge:---------- DI00 Reglersperre DI01 Freigabe DI02 Reset (ES-Freifahren) DI03 Referenznocken DI04 Endschalter rechts DI05 Endschalter links DI10 Hubwerkposition 0 mm DI11 " 1000 mm DI12 " 2000 mm DI13 - - - DI14 (Tipp-rechts) DI15 (Tipp-links) DI16 Referenzfahrt DI17 Start Positionierung Klemmenbelegung Ausgänge:---------- DB00 Bremse DO01 Betriebsbereit DO16 "IPOS in Position" DO17 "IPOS Referenz"	Kommentar
---------- Programm-Anfang ========== Programm-Sprungverteiler ========== SETINT ERROR M10 M100:CALL M50 JMP LO I0001000000000000, M101 CALL M20 M101:JMP LO I0000010000000000, M102 CALL M30 M102:JMP LO I0000100000000000, M103 CALL M40 M103:JMP UNCONDITIONED , M100 ----------	Programm-Sprungverteiler Interrupt-Routine aktivieren für Hardware-Endschalter-Verarbeitung Reset/Fahren aus Endschalter → Hauptprogramm DI16 = 1 → Referenzfahrt DI15 = 1 → Tipp-rechts DI14 = 1 → Tipp-links
Unterprogramm Reset/Fahren aus Endschalter ========== M10:JMP HI I0000000000110000, M1 M3:JMP HI I0000000000110000, M2 ASTOP IPOS ENABLE JMP UNCONDITIONED , M3 M2:ASTOP TARGET POSITION M1:RET ----------	Reset/Fahren aus Endschalter wenn kein Endschalter (DI04/DI05 Endschalter rechts/links) angefahren, dann zurück Sprungverteiler, wenn ja dann Verfahrentriegelung und warten, bis Antrieb aus Endschalter gefahren ist (DI02 - Eingangsklemmenfunktion "Reset"). Dann Antriebsstopp mit Setzen der Zielposition = Aktuelle Position

`Unterprogramm Referenzieren`
`==========`
`M20:ASTOP  IPOS ENABLE`
`GO0  U,NW, ZP`
`M22:JMP  LO I0000000000000001, M21`
`SET  H319 = 0`
`BMOV  H319.0  = H473.20`
`JMP  H319 == 0  , M22`
`M21 : ASTOP  TARGET POSITION`
`RET`
`----------`
	Referenzfahrt
	
	Verfahrentriegelung
	Referenzfahrt, nicht wartend, auf Nullimpuls starten,
	Referenzfahrt abbrechen
	und das Bit im Statuswort "IPOS Referenz" =0

`(Unterprogramm Tippbetrieb)`
`==========`
`M30:RET`
`M40:RET`
`----------`
	Option: Unterprogramm (z. B. Tippbetrieb)
	Tipp-rechts
	Tipp-links
	Siehe nächstes Beispiel.

`Hauptprogramm Hubwerkspositionierung`
`==========`
`M50:JMP  LO I0000000001000000, M51`
`GOA  WAIT  #0  mm`
`M51:JMP  LO I0000000010000000, M52`
`GOA  WAIT  #1000  mm`
`M52:JMP  LO I0000000100000000, M53`
`GOA  WAIT  #2000  mm`
`M53:RET`
`----------`
`END`
	Hauptprogramm Hubwerkspositionierung
	
	wenn Eingang DI10 gesetzt, dann fahre Position 0 mm an
	wenn Eingang DI11 gesetzt, dann fahre Position 1000 mm an
	wenn Eingang DI12 gesetzt, dann fahre Position 2000 mm an

24.3 Beispielprogramm "Tippbetrieb"

24.3.1 Eigenschaften

- Verfahren in zwei Richtungen mit den Binäreingängen Tipp+ / Tipp-.
- Einstellbare Verfahrdrehzahlen und Rampen.
- Endloses Verfahren möglich.
- Keine Referenzfahrt notwendig.
- Einhalten von Verfahrbereichsgrenzen (Software-Endschalter).
- Automatisches Herausfahren aus Hardware-Endschalter.

Mittels zwei Binäreingängen Tipp+ (DI14) und Tipp- (DI15) kann in zwei Richtungen endlos verfahren werden. Es ist keine Referenzfahrt notwendig. Bei referenziertem Antrieb und eingestelltem Software-Endschalter wird nur innerhalb diesem verfahren. Es wird nur bei einem "1"-Signal an einer der Tippklemmen verfahren. Aus einem angefahrenen Hardware-Endschalter wird durch ein "1"-Signal am RESET-Eingang (DI02) herausgefahren.

24.3.2 Einstellungen

Die detaillierte Konfiguration der Ein-/Ausgänge sowie der im Programm verwendeten Variablen ist im Kommentarteil des Programmquellcodes dokumentiert.

24.3.3 Eingangsklemmen

Pegel	Klemme	Klemmenfunktion	Bedeutung
0	DI00	/Reglersperre	Zu-/Abschalten des Leistungsteils
0	DI01	Freigabe	geregelter Stillstand
0	DI02	Reset	Reset nach Fehler (Endschalter freifahren)
0	DI03	Referenznocken	Schalter für Nullstellung oder Offsetwert
0	DI04	Endschalter rechts	Begrenzungsschalter zum Stillsetzen (+)
0	DI05	Endschalter links	Begrenzungsschalter zum Stillsetzen (-)
0	DI10	IPOS Eingang	-
0	DI11	IPOS Eingang	-
0	DI12	IPOS Eingang	-
0	DI13	IPOS Eingang	-
0	DI14	IPOS Eingang	Tippen positiv
0	DI15	IPOS Eingang	Tippen negativ
0	DI16	IPOS Eingang	Start Referenzfahrt
0	DI17	IPOS Eingang	Start Positionierung

24.3.4 Ausgangsklemmen

Pegel	Klemme	Gerät	Klemmenfunktion	Bedeutung
0	DB00	MDX	/Bremse	Ansteuerung der Bremse über Hilfsrelais
0	DO01	MDX	Betriebsbereit	Controller aktiv, Elektronikversorgung OK
0	DO02	MDX	/Störung	kein Fehler vorhanden
0	DO10	DIO11B	IPOS Ausgang	-
0	DO11	DIO11B	IPOS Ausgang	-
0	DO12	DIO11B	IPOS Ausgang	-
0	DO13	DIO11B	IPOS Ausgang	-
0	DO14	DIO11B	IPOS Ausgang	-
0	DO15	DIO11B	IPOS Ausgang	-
0	DO16	DIO11B	IPOS in Position	Antrieb befindet sich im Positionsfenster
0	DO17	DIO11B	IPOS Referenz	Referenzfahrt erfolgreich durchgeführt

24.3.5 Programmquellcode (mit Kommentaren)

ZÄHLER: 1 NENNER: 1 EINHEIT: inc ********** Beispielprogramm: Tipp-Betrieb Datei: Tipp.mdx Ersteller: SEW/AWT Datum: 01.06.98 Funktion: Tipp-Betrieb - endloses Verfahren möglich - keine Referenzierung der Achse notwendig - Einhalten von Verfahrgrenzen; Software-ES - Verfahrdrehzahlen/-Rampen ab H310 - Eingänge Tipp+ (DI14) / Tipp- (DI15) Parametrierung (P600) der Ein-/Ausgänge: In Hochkommata = angegebene Funktion ohne Hochkommata = IPOS-EINGANG/AUSGANG Klemmenbelegung Eingänge:---------- DI00 "Reglersperre" DI01 "Freigabe" DI02 "Fehler-Reset" (ES-Freifahren) DI04 "Referenznocken" DI03 "Endschalter rechts" DI05 "Endschalter links" DI14 Tippen rechts DI15 Tippen links DI16 Start Referenzfahrt DI17 (Start Positionierung) Klemmenbelegung Ausgänge---------- DB00 Bremse DO01 Betriebsbereit DO16 "IPOS in Position" DO17 "IPOS Referenz" Verwendete Variablen:---------- H310 = V-Tipp rechts (1/10 Umdr./min) H311 = "links" H312 = Beschleunigungs-Rampe (ms) H313 = Verzögerungs-Rampe (ms) H316 - H319 = Tippen-Hilfsvariable ********** ----------	Kommentar
Programm-Anfang ========== Initialisierung ---------- SET H310 = 5000 SET H311 = 5000 SET H312 = 2000 SET H313 = 2000 ----------	Geschwindigkeits- und Beschleunigungswerte für den Tippbetrieb setzen (siehe Kommentar)

`Programm-Sprungverteiler`
`==========`
`SETINT  ERROR  M10`
`M100:JMP  LO I0001000000000000, M101`
`CALL  M20`
`M101:JMP  LO I0000010000000000, M102`
`CALL  M30`
`M102:JMP  LO I0000100000000000, M103`
`CALL  M40`
`M103:JMP  UNCONDITIONED  , M100`
`----------`
	Programm-Sprungverteiler
	
	Interrupt-Routine aktivieren für Hardware-Endschalter-Verarbeitung Reset/Fahren aus Endschalter
	DI16 = 1 → Referenzfahrt
	DI15 = 1 → Tipp-rechts
	DI14 = 1 → Tipp-links

`Unterprogramm Reset/Fahren aus Endschalter`
`==========`
`M10 :JMP  HI I0000000000110000, M1`
`M3  :JMP  HI I0000000000110000, M2`
`ASTOP  IPOS ENABLE`
`JMP  UNCONDITIONED  , M3`
`M2  :ASTOP  TARGET POSITION`
`M1  :RET`
`----------`
	Reset/Fahren aus Endschalter
	wenn kein Endschalter angefahren (DI05/DI06 ES-rechts/links), dann zurück Sprungverteiler. Wenn ja, dann Verfahrentriegelung und warten bis Antrieb aus Endschalter gefahren ist (parametrierte "Reset" Eingangsfunktion DI02). Dann Antriebsstopp mit Setzen der Zielposition = aktuelle Position.

`Unterprogramm Referenzieren`
`==========`
`GO0  U,NW, ZP`
`M20 :ASTOP  IPOS ENABLE`
`M22 :JMP  LO I0000000000000001, M21`
`SET  H309 = 0`
`BMOV  H309.0  = H473.20`
`JMP  H309 == 0  , M22`
`M21 :ASTOP  TARGET POSITION`
`RET`
`----------`
	Referenzfahrt
	Verfahrentriegelung
	Referenzfahrt, nicht wartend, auf Nullimpuls starten, solange wie "Reglersperre" =0
	und das Bit im Statuswort "IPOS Referenz" =0

`Unterprogramm Tippbetrieb`
`==========`
`Tippbetrieb (Abfrage Software-ES aktiv)`
`----------`
`M35 :SETSYS  POS.SPEED C(C)W = H310`
`SETSYS  POS. RAMP  = H312`
`SET  H319 = 0`
`BMOV  H319.0  = H473.20`
`JMP  H319 == 0  , M36`
`SET  H319 = H496`
`OR  H319 ¦ H497`
`JMP  H319 == 0, M36`
`SET  H319 = 1`
`SET  H317 = H496`
`SET  H318 = H497`
`M36 :RET`
	Tippbetrieb
	Abfrage Software-Endschalter aktiv
	Geschwindigkeit setzen
	Rampenzeit setzen
	Abfrage, ob Achse referenziert ist (Software-ES aktiv)
	Abfrage, ob Softwarebereiche beide =0 (Software-ES nicht aktiv)
	wenn Software-ES aktiv, dann Merker H319=1 setzen
	und die Tipp-Verfahrvariablen (H317) mit den Software-Verfahrbereichen (Systemvariablen H496 u. H497) laden

----------	Tipp-rechts
Tipp +	so lange verfahren wie DI14 = 1 und
----------	DI15 = 0,
M30 :JMP LO I0000010000000000, M31	Abfrage Software-ES aktiv
JMP HI I0000100000000000, M31	
CALL M35	1000 Motorumdrehungen zur aktuellen Ist-Position addieren und das Ergebnis als neue Zielposition anfahren
JMP H319 == 1 , M32	
GETSYS H317 = ACT.POSITION	
ADD H317 + 4096000	
M32 :ASTOP IPOS ENABLE	
GOA NOWAIT H317	
JMP UNCONDITIONED , M30	
M31 :ASTOP TARGET POSITION	
RET	

Tipp -	

M40 :JMP LO I0000100000000000, M41	Tipp-links
JMP HI I0000010000000000, M41	so lange verfahren wie DI14 = 0 und
CALL M35	DI15 = 1,
JMP H319 == 1 , M42	Abfrage Software-ES aktiv
GETSYS H318 = ACT.POSITION	
SUB H318 - 4096000	1000 Motorumdrehungen zur aktuellen Ist-Position addieren und das Ergebnis als neue Zielposition anfahren
M42 :ASTOP IPOS ENABLE	
GOA NOWAIT H318	
JMP UNCONDITIONED , M40	
M41 :ASTOP TARGET POSITION	
RET	

Ende Tipp-Betrieb	

END	

24.4 Beispielprogramm "Tabellenpositionierung"

24.4.1 Eigenschaften

- Binärcodierte Anwahl von 16 Tabellenpositionen.
- Binärcodierte Ausgabe der aktuell angewählten Tabellenposition.
- Eindeutige Meldung bei Erreichen der angewählten Tabellenposition.
- Automatisches Herausfahren aus Hardware-Endschalter.

Mit den ersten 4 Binäreingängen der Option DIO11B können 16 Tabellenpositionen (Verfahrvariablen H000 ... H015) binärcodiert angewählt werden. Eine angewählte Verfahrvariablen-Nr. (Tabellenzeiger) wird immer binärcodiert an den ersten 4 Binäreingängen der DIO 11A gespiegelt.

Bevor Tabellenpositionen angefahren werden können, ist mit dem Eingang DI16 "Referenzfahrt" eine Referenzfahrt zu aktivieren. Mit dem Eingang DI17 "Startpositionierung" wird der Verfahrauftrag auf die Tabellenposition freigegeben bzw. unterbrochen (bei "Reglersperre" und "Freigabe" = "1"-Signal). Es ist ratsam, bei der Anwahl einer neuen Tabellenposition den Eingang DI17 solange auf "0"-Signal zu setzen, bis ein sicheres Setzen aller Bits des Tabellenzeigers gewährleistet ist!

Ein "1"-Signal am Ausgang DO15 "Tabellenposition gültig" zeigt das Erreichen der angewählten Tabellenposition an. Bei Anwahl einer neuen Tabellenposition wird dieser Ausgang sofort zurückgesetzt. Durch zusätzliches Auswerten des Ausganges DO16 "IPOS in Position" kann auch bei deaktiviertem Regler ("Reglersperre" = "0") ein Verlassen der angewählten Tabellenposition sicher erkannt werden.

Aus einem angefahrenen Hardware-Endschalter wird durch ein "1"-Signal am RESET-Eingang (DI02) herausgefahren.

24.4.2 Einstellungen

Die detaillierte Konfiguration der Ein-/Ausgänge (s. unten) sowie der im Programm verwendeten Variablen ist im Kommentarteil des Programmquellcodes dokumentiert.

Die Tabellenpositionen können über das Variablenfenster des Assemblers oder mit dem Handbediengerät in die Variablen (H00 ... H15) geschrieben werden. Die Variablen sind somit spannungsausfallsicher gespeichert.

	HINWEIS Die Anwenderverfahreinheiten Zähler und Nenner im Positionierfensterkopf sind hier unwirksam, da die Positionswerte von Verfahrvariablen immer in Inkrementen (4096 Inkr./Motorumdrehung) bewertet werden.

24.4.3 Eingangsklemmen

Pegel	Klemme	Klemmenfunktion	Bedeutung
0	DI00	/Reglersperre	Zu-/Abschalten des Leistungsteils
0	DI01	Freigabe	geregelter Stillstand
0	DI02	Reset	Reset nach Fehler (Endschalter freifahren)
0	DI03	Referenznocken	Schalter für Nullstellung oder Offsetwert
0	DI04	Endschalter rechts	Begrenzungsschalter zum Stillsetzen (+)
0	DI05	Endschalter links	Begrenzungsschalter zum Stillsetzen (-)
0	DI10	IPOS Eingang	Variablenzeiger Bit 2'0
0	DI11	IPOS Eingang	Variablenzeiger Bit 2'1
0	DI12	IPOS Eingang	Variablenzeiger Bit 2'2
0	DI13	IPOS Eingang	Variablenzeiger Bit 2'3
0	DI14	IPOS Eingang	Tippen positiv
0	DI15	IPOS Eingang	Tippen negativ
0	DI16	IPOS Eingang	Start Referenzfahrt
0	DI17	IPOS Eingang	Start Positionierung

24.4.4 Ausgangsklemmen

Pegel	Klemme	Gerät	Klemmenfunktion	Bedeutung
0	DB00	MDX	/Bremse	Ansteuerung der Bremse über Hilfsrelais
0	DO01	MDX	Betriebsbereit	Controller aktiv, Elektronikversorgung OK
0	DO02	MDX	/Störung	kein Fehler vorhanden
0	DO10	DIO11B	IPOS Ausgang	Variablenzeiger Bit 2'0
0	DO11	DIO11B	IPOS Ausgang	Variablenzeiger Bit 2'1
0	DO12	DIO11B	IPOS Ausgang	Variablenzeiger Bit 2'2
0	DO13	DIO11B	IPOS Ausgang	Variablenzeiger Bit 2'3
0	DO14	DIO11B	IPOS Ausgang	-
0	DO15	DIO11B	IPOS Ausgang	Tabellenposition gültig
0	DO16	DIO11B	IPOS in Position	Antrieb befindet sich im Positionsfenster
0	DO17	DIO11B	IPOS Referenz	Referenzfahrt erfolgreich durchgeführt

24.4.5 Programmquellcode (mit Kommentaren)

ZÄHLER: 1 NENNER: 1 EINHEIT: inc ********** Programm: Tabellenpositionierung Datei: Tab.mdx Ersteller: SEW/AWT Datum: 01.06.98 Funktion Tabellenpositionierung: - mit den ersten 4 Eingängen der Option DIO11 werden binärcodiert die Positionen in den entspr. Variablen 0-15 angewählt. - mit dem Eingang DI17 (X22:17) wird der angewählte Verfahrbefehl freigegeben. Parametrierung der Ein-/Ausgänge: In Hochkommata = angegebene Funktion ohne Hochkommata = IPOS-EINGANG/AUSGANG Klemmenbelegung Eingänge:---------- DI00 "Reglersperre" DI01 "Freigabe" DI02 "Fehler-Reset" (ES-Freifahren) DI03 "Referenznocken" DI04 "Endschalter rechts" DI05 "Endschalter links" DI10 Variablenzeigerbit 2'0 DI11 2'1 DI12 2'2 DI13 2'3 DI14 (Tippen rechts) DI15 (Tippen links) DI16 Start Referenzfahrt DI17 Start Positionierung Klemmenbelegung Ausgänge---------- DB00 Bremse DO01 Betriebsbereit DO10 Variablenzeigerbit 2'0 DO11 2'1 DO12 2'2 DO13 2'3 DO14 - DO15 Tabellenposition erreicht DO16 "IPOS in Position" DO17 "IPOS Referenz" Verwendetet Variablen:---------- H300 = Verfahrdrehzahl-rechts (1/10 Umdr./min) H301 = Verfahrdrehzahl-links (1/10 Umdr./min) H302 = Beschleunigungs-Rampe rechts (ms) H303 = Verzögerungs-Rampe links (linear) H320 - H324 = Hilfsvariablen ********** ----------	Kommentar

`Programm-Anfang`
`==========`
`Initialisierung`
`----------`
`SET  H300 = 15000`
`SET  H301 = 15000`
`SET  H302 = 1000`
`SET  H303 = 1000`
`----------`
	Geschwindigkeits- und Beschleunigungswerte für die Tabellenpositionierung setzen (siehe Variablenbeschreibung im Kommentar des Programmquellcodes)

`Programm-Sprungverteiler`
`==========`
`SETINT  ERROR  M10`
`M100:CALL  M50`
`JMP  LO I0001000000000000, M101`
`CALL  M20`
`M101:JMP  LO I0000010000000000, M102`
`CALL  M30`
`M102:JMP  LO I0000100000000000, M103`
`CALL  M40`
`M103:JMP  UNCONDITIONED  , M100`
`----------`
	Programm-Sprungverteiler
	Interrupt-Routine aktivieren für Hardware-Endschalter-Verarbeitung Reset/Fahren aus Endschalter → Hauptprogramm
	DI16 = 1 → Referenzfahrt
	DI14 = 1 → Tipp-rechts
	DI15 = 1 → Tipp-links

`Unterprogramm Reset/Fahren aus Endschalter`
`==========`
`M10 :JMP  HI I0000000000110000, M1`
`M3  :JMP  HI I0000000000110000, M2`
`ASTOP  IPOS ENABLE`
`JMP  UNCONDITIONED  , M3`
`M2  :ASTOP  TARGET POSITION`
`M1  :RET`
`----------`
	Reset/Fahren aus Endschalter
	wenn kein Endschalter angefahren (DI04/DI05 ES-rechts/links), dann zurück Sprungverteiler, wenn ja, dann Verfahrentriegelung und warten bis Antrieb aus Endschalter gefahren ist (parametrierte "Reset" Eingangsfunktion DI02).
	Dann Antriebsstopp mit Setzen der Zielposition = aktuelle Position

`Unterprogramm Referenzieren`
`==========`
`M20 :ASTOP  IPOS ENABLE`
`AND  H480 & FFFFFFF0 hex`
`BCLR  H480.5 = 0`
`GO0  U,NW, ZP`
`M22 :JMP  LO I0000000000000001, M21`
`SET  H319 = 0`
`BMOV  H319.0  = H473.20`
`JMP  H319 == 0  , M22`
`M21 :ASTOP  TARGET POSITION`
`RET`
`----------`
	Referenzfahrt
	Verfahrentriegelung
	Ausgabe binärcodierte Tabellenposition löschen
	Ausgabe "Tabellenposition gültig" löschen
	Referenzfahrt, nicht wartend, auf Nullimpuls starten,
	solange wie "Reglersperre" =0
	und das Bit im Statuswort "IPOS Referenz" =0

`Unterprogramm Tippbetrieb`
`==========`
`M30 :RET`
`M40 :RET`
	Option: Unterprogramm (z.B. Tippbetrieb)
	Tipp-rechts
	Tipp-links

`Hauptprogramm Tabellenpositionierung`	Hauptprogramm Tabellenpositionierung
`==========`	
`Prüfung, ob Achse referenziert ist`	
`----------`	Es werden nur Tabellenpositionen bei referenziertem Antrieb angefahren
`M50 :SET  H321 = 0`	
`BMOV  H321.0  = H473.20`	(DO17 = 10. Bitstelle in Ausgangsklemmen-System-variable H473; parametriert auf "IPOS Referenz")
`JMP  H321 != 0, M51`	
`RET`	
`----------`	
`Verfahrgeschwindigkeit und -Rampe setzen`	Setzen von Verfahrgeschwindigkeit, Beschleunigungs- und Verzögerungsrampe
`----------`	
`M51 :SETSYS  POS.SPEED C(C)W = H300`	
`SETSYS  POS. RAMP  = H302`	
`----------`	
`Variablenzeiger in Variable H320 einlesen`	Tabellenzeiger (Verfahrvariablen-Nr.) binärcodiert mit 4 Eingängen (DI10 - DI13) anwählen
`----------`	
`SET  H320 = H483`	
`ASHR  H320 > >  6`	
`AND  H320 & F  hex`	wenn der Tabellenzeiger verändert wurde, dann Ausgang "Tab.position gültig" zurücksetzen, aktuellen Tabellenzeiger in Vergleichsvariable speichern
`----------`	
`Ausgang "Tabellenposition erreicht" prüfen`	
`----------`	
`JMP  H322 == H320, M54`	
`BCLR  H480.5  = 0`	
`M54 :SET  H322 = H320`	angewählten Tabellenzeiger auf Ausgangsklemmen (DO10 - DO13) schreiben, ohne dass andere Ausgänge der Ausgangsvariablen (H480) verändert werden
`----------`	
`Variablenzeiger binärcodiert ausgeben`	
`----------`	
`SET  H323 = H320`	
`SET  H324 = H480`	
`AND  H324 & FFFFFFF0 hex`	
`OR  H323 ¦ H324`	
`SET  H480 = H323`	
`----------`	wenn DI17 = 1, dann verfahre auf Positionswert der angewählten Verfahrvariable, ansonsten Antriebsstopp
`Tabellenpositionier-Freigabe`	
`----------`	Meldung "Tabellenpos.-Anwahl gültig" zurücksetzen
`M53 :JMP  LO I0010000000000000, M52`	Verfahrverriegelung aufheben
`ASTOP  IPOS ENABLE`	Verfahren auf Tabellenposition, so lange bis Position erreicht oder DI17 = 0 ist
`GOA  NOWAIT [H320]`	
`JMP  NOT IN POSITION, M53`	Meldung "Tabellenpos.-Anwahl gültig" setzen
`BSET  H480.5  = 1`	
`JMP  UNCONDITIONED  , M55`	
`----------`	
`M52 :ASTOP  HOLD CONTROL`	
`M55 :RET`	Antriebsstopp
`----------`	
`Ende Tabellenpositionierung`	
`----------`	
`END`	

Stichwortverzeichnis

A

Abarbeitungszeit Task 43
Absolutwertgeber 53
Absolutwertgeber (SSI) 138
Absolutwert, Istposition 35
ActPos_Abs / ACTPOS ABS 35
ActPos_Extt / ACTPOS EXT 35
ActPos_Mot / ACTPOS MOT 35
AnaOutpIPOS / ANA. OUT IP 31
AnaOutpIPOS2 / ANA. OUT IP2 30
Anwender-Timer 33
Anwender-Watchdog 34
Aufbau der Sicherheitshinweise 14
Ausgabeverzeichnis 156
Automatische Gebertauscherkennung 137

B

Bestimmungsgemäße Verwendung 17
Betriebszustand 283
Binärausgänge 30
Binärausgänge Grundgerät 31, 32, 36, 37
Binäreingänge Grundgerät 32, 36
BREAK 283
Bussysteme 17

C

CAN-Geber 140
CCount_Ext 36
CCount_ExtOn 36
CCount_Mot 36
CCount_MotOn 36
ControlWord / CTRL. WORD 33

D

DIO11A, Analogausgänge 30, 31
DIO11A, Binärausgänge 31
DIP11A, Binärausgänge 31
DPRAM-Synchronisierung 141
DRS11A, Binäreingänge 30
DRS11A, Statusmeldungen 30
DRS_Ctrl / DRS CTRL. 30
DRS_Status / DRS STATUS 30

E

Erkennung Positionierunterbrechung 131
Extener Geber, Istposition 35
Externer Geber, C-Spur 36
Externer Geber, Nullimpulse 36

F

Fachkraft 17
Farbe, Hintergrund 154
Fehler-Interrupt 46
Feldbus, Sollposition 35

G

Geberskalierung 139
Geberskalierung Ext. Geber 135
Gebertyp 53, 138
Geschwindigkeit Ausführung 157
Geschwindigkeit Task 1 132
Geschwindigkeit Task 2 133
Geschwindigkeitsvorsteuerung 127

H

Haftungsausschluss 15
Hintergrundfarbe 154
Hiperface Offset X14 136

I

İinclude, Verzeichnis 156
Inkrementalgeber 53
Inkrementalgebernachbildung 53
InpLevelB / INPUT LVLB 36
INPUT LVL 32, 36
InputLevel / INPUT LVL 32
Interrupts 40, 45
Interrupts, Aktivierung 46
Interrupts, Fehler-Interrupt 46
Interrupt, Timer0-Interrupt 48
Interrupt, Touch-Probe-DI02 Interrupt 47
IPOS Geber 134
IPOS Modulofunktion 140
IPOS Referenzfahrt 123
IPOS Sonderfunktionen 132
IPOS Überwachungen 131
IPOS Variablen Edit 134
IPOS Verfahrparameter 127

IPOS-Parameter 123
IPOS-Sollwert 38
IPOS-STW. Task 1.....132
IPOS-STW. Task 2.....132
IPOS-Synchronisation.....141
IPOS-Zähler 35
IPOS_Setp / IPOS_SETP 38
Istposition Absolutwert 35
Istposition Externer Geber 35
Istposition Motorgeber 35

L

LagDistance / LAG DISTAN.....34
LagWindow / LAG WINDOW 34

M

Mängelhaftung 15
ModActPos / MOD.ACTPOS.....29
ModCount / MOD COUNT 29
ModTagPos / MOD.TAGPOS 29
Modulo Actual Position 29
Modulo Geberauflösung.....140
Modulo Nenner 140
Modulo Target Position 29
Modulo Zähler 29, 140
ModuloCtrl / MODULOCTRL.....29
Modulo-Funktion
  Grundlegendes 73
Modulofunktion.....140
Modulo-Funktion, Steuerwort.....29
Motorgeber, C-Spur 36
Motorgeber, Istposition 35
Motorgeber, Nullimpulse 36

N

Nockenabstand 126
Nullpunktoffset 139

O

OptOutpIPOS / OPT. OUT IP.....31
OutpLevelB / OUTPUT LVLB.....37
OutputLevel / OUTPUT LVL.....32
Override 132

P

Parameterbeschreibung
  P9xx IPOS-Parameter 123
PID-Regler
  Betriebsart 38
  D-Anteil 38
  I-Anteil.....38
  Istwert gefiltert und skaliert 38
  Istwert-Adresse.....38
  Istwert-Maximalwert.....38
  Istwert-Minimalwert.....38
  Istwert-Offset 38
  Istwert-Skalierung 38
  Maximaler Ausgangswert 38
  Maximaler Ausgangswert Stellgröße 39
  Minimaler Ausgangswert 38
  Minimaler Ausgangswert Stellgröße 39
  P-Anteil 38
  Proportionalanteil.....38
  Solllwert-Adresse.....38
  Sollwert 38
  Sollwert-Skalierung.....38
  Statuswort.....39
  Vorsteuerwert 38
PID_ActAdr / PID.ACTADR 38
PID_ActMax / PID.ACTMAX.....38
PID_ActMin / PID.ACTMIN 38
PID_ActNorm / PID.ACTNOR.....38
PID_ActOffset / PID.ACTOFF.....38
PID_ActScale / PID.ACTSCA 38
PID_CmdAdr / PID.CMDADR.....38
PID_CmdScale / PID.CMDSCA.....38
PID_Command / PIDCOMMAND 38
PID_Feedf / PID.FEEDF 38
PID_K_p / PID.KP.....38
PID_LimitMax / PID.LMTMAX 38
PID_LimitMin / PID.LMTMIN.....38
PID_Mode / PID.MODE 38
PID_Outp_D / PID.OUTPD.....38
PID_Outp_I / PID.OUTPI 38
PID_Outp_P / PID.OUTPP 38
PID_SetpMax / PID.SETMAX.....39
PID_SetpMin / PID.SETMIN 39
PID_Status / PID.STATUS 39
Positionier-Rampe 1 127
Positionier-Rampe 2 127
Positionierunterbrechung.....131
Positionsfenster 34, 131
Positionsoffset 139
PosWindow / POS. WINDOW 34
PSTP 283
P9xx IPOS-Parameter 123
P90x IPOS Referenzfahrt 123

P900 Referenzoffset 123
P901 Referenzdrehzahl 1 124
P902 Referenzdrehzahl 2 124
P903 Referenzfahrttyp 124
P904 Referenzierung auf Nullimpuls....126
P905 Hiperface Offset X15 126
P906 Nockenabstand....126
P91x IPOS Verfahrparameter 127
P910 Verstärkung X-Regler 127
P911 Positionier-Rampe 1 127
P912 Positionier-Rampe 2 127
P913 / P914 Verfahrdrehzahl RECHTS/LINKS....127
P915 Geschwindigkeitsvorsteuerung....127
P916 Rampenform....128
P917 Rampenmode 130
P918 Quelle Bussollwert....130
P92x IPOS Überwachungen 131
P920 SW-Endschalter RECHTS....131
P921 SW-Endschalter LINKS 131
P922 Positionsfenster 131
P923 Schleppfehlerfenster....131
P924 Erkennung Positionierunterbrechung 131
P93x IPOS Sonderfunktionen 132
P930 Override....132
P931 IPOS-STW. Task 1 132
P932 IPOS-STW. Task 2 132
P933 Ruckzeit....132
P938 Geschwindigkeit Task 1....132
P939 Geschwindigkeit Task 2....133
P94x IPOS Geber 134
P940 IPOS Variablen Edit....134
P941 Quelle Istposition 134
P942 / P943 Geberfaktor Zähler / Nenner 134
P944 Geberskalierung Ext. Geber 135
P945 Streckengeber Typ (X14)....135
P946 Streckengeber Zählrichtung (X14)....136
P947 Hiperface Offset X14 136
P948 Automatische Gebertauscherkennung 137
P95x Absolutwertgeber (SSI)....138
P950 Gebertyp....138
P951 Zählrichtung....138
P952 Taktfrequenz....139
P953 Positionsoffset 139
P954 Nullpunktoffset....139
P955 Geberskalierung 139
P956 Baudrate CAN-Geber 140
P96x IPOS Modulofunktion....140
P960 Modulofunktion 140
P961 Modulo Zähler....140
P962 Modulo Nenner....140
P963 Modulo Geberauflösung....140
P97x IPOS-Synchronisation 141
P970 DPRAM-Synchronisierung 141
P971 Synchronisierungsphase....141

Q

Quelle Istposition 134

R

Rampenform....128
Rampenmode 130
RecStatS1 / SBUS1REC 37
RecStatS2 / SBUS2REC 37
Referenzdrehzahl 1 124
Referenzdrehzahl 1 / 2 124
Referenzdrehzahl 2 124
Referenzfahrttyp 124
Referenzierung auf Nullimpuls....126
Referenz-Offset 34
Referenzoffset 123
RefOffset / REF. OFFSET 34
Resolver....53
Ruckzeit....132

S

Sachmängelhaftung....15
Schleppabstand 34
Schleppfehlerfenster....34, 131
SCOPE 30
Scope474 / SCOPE 474 30
Scope475 / SCOPE 475 30
SetpointPos / SETP. POS. 34
SetpPosBus / SP. POS. BUS 35
Sicherheitshinweise 14, 16
SLS_left / SLS LEFT....34
SLS_right / SLS RIGHT 34
Software-Endschalter Links 34
Software-Endschalter Rechts 34
Sollposition 34
Sollposition Feldbus....35
START 283
StatusWord / STAT. WORD 30
Statuswort....30
Statuswort SCOM 37
StdOutpIPOS / STD. OUT IP....31
STEP 283
Steuerwort 33

Steuerwort Modulo-Funktion 29
Streckengeber Typ (X14) 135
Streckengeber Zählrichtung (X14) 136
SW-Endschalter LINKS 131
SW-Endschalter RECHTS 131
Synchronisierungsphase 141
Syntaxdarstellung 154
Syntax-Highlighting 154

T

Taktfrequenz 139
TargetPos / TARGET POS 34
Task Abarbeitungszeit 43
Task Geschwindigkeit 157
Task Implementierungshinweise 44
Task 3 44
Tasks MOVIDRIVE® A 43
Tasks MOVIDRIVE® B 43
Task-Verwaltung 40
Timer0-Interrupt 48
Timer_0 / TIMER 0 33
Timer_1 / TIMER 1 33
Timer_2 / TIMER 2 33
Touch Probe 35
Touch-Probe-DI02 Interrupt 47
TpPos1_Abs / TP.POS1ABS 35
TpPos1_Ext / TP.POS1EXT 35
TpPos1_Mot / TP.POS1MOT 35
TpPos1_VE / TP.POS1VE 35
TpPos2_Abs / TP.POS2ABS 35
TpPos2_Ext / TP.POS2EXT 35
TpPos2_VE / TP.POS2VE 35
TP. POS1ABS 35
TP. POS1EXT 35
TP. POS1MOT 35
TP. POS2ABS 35
TP. POS2EXT 35
TP. POS2MOT 35
T0_Reload / T0 RELOAD 33

U

Urheberrechtsvermerk 15

V

Variablen-Interrupt-Request 38
Variablen-Interrupts MOVIDRIVE® B 49
Variablen-Interrupts, Aufruf 49
Variablen-Interrupts, IPOS-Zugriff 50
VarIntReq / VARINTREQ 38
Verfahrdrehzahl RECHTS/LINKS 127
Verstärkung X-Regler 127
Verzeichnis İinclude-Direktiven 156
Verzeichnisse 156
Virtueller Encoder 35

W

Watchdog 34
WdogTimer / WD. TIMER 34
WD. TIMER 34

Z

Zählrichtung 138
Zielgruppe 17
Zielposition 34

Symbole

_GetSys 182, 214
_MoviLink 182, 223, 309
_SBusCommDef 182, 231
_SetSys 182, 238

SEW-EURODRIVE
Driving the world

SEW-EURODRIVE GmbH & Co KG
P.O. Box 3023
D-76642 Bruchsal/Germany
Phone +49 7251 75-0
Fax +49 7251 75-1970
sew@sew-eurodrive.com

→ **www.sew-eurodrive.com**