BUCHER automation

Motion-API-Dokumentation



60888786_00

MCX API 2.0

Motion Control eXtended

Dieses Dokument wurde von der Bucher Automation AG mit der gebotenen Sorgfalt und basierend auf dem ihr bekannten Stand der Technik erstellt. Änderungen und technische Weiterentwicklungen an unseren Produkten werden nicht automatisch in einem überarbeiteten Dokument zur Verfügung gestellt.
Die Bucher Automation AG übernimmt keine Haftung und Verantwortung für inhaltliche oder formale Fehler, fehlende Aktualisierungen sowie daraus eventuell entstehende Schäden oder Nachteile.

Bucher Automation AG
Thomas-Alva-Edison-Ring 10
71672 Marbach am Neckar, Deutschland
T +49 7141 2550-0
info@bucherautomation.com

Technischer Support
T +49 7141 2550-444
support@bucherautomation.com

Vertrieb
T +49 7141 2550-663
sales@bucherautomation.com

www.bucherautomation.com

Originaldokument

Dokumentenversion	1.00
Ausgabedatum	12.08.2025

Inhaltsverzeichnis

1 Einleitung ... 26
1.1 Informationen zum Dokument ... 26
1.2 Darstellungskonventionen ... 26
2 MCAxis.AxisConfigData ... 27
2.1 Syntax für MCAxis.AxisConfigData ... 27
2.2 Eigenschaften von MCAxis.AxisConfigData ... 27
2.2.1 DeviceName ... 27
2.2.2 AxisMode ... 28
2.2.3 AxisDriveType ... 28
2.2.4 ObjectID ... 29
2.2.5 DeviceID ... 29
2.2.6 IsSimulation ... 29
3 MCAxis.Buffer ... 31
3.1 Syntax für MCAxis.Buffer ... 31
3.2 Funktionen von MCAxis.Buffer ... 31
3.2.1 Create() ... 32
3.2.2 Clear() ... 32
3.2.3 Delete() ... 33
3.2.4 Suspend() ... 34
3.2.5 Resume() ... 34
3.2.6 Wait() ... 35
3.3 Eigenschaften von MCAxis.Buffer ... 35
3.3.1 BufferState ... 36
3.3.2 IsCyclic ... 36
3.3.3 IsNormal ... 37
3.4 Beispieltask für MCAxis.Buffer ... 37
4 MCAxis.Diagnostics ... 39
4.1 Syntax für MCAxis.Diagnostics ... 39
4.2 Funktionen von MCAxis.Diagnostics ... 39
4.2.1 ClearErrors() ... 39
4.2.2 GetErrorCode() ... 40
4.2.3 GetMessageCode() ... 40
4.3 Eigenschaften von MCAxis.Diagnostics ... 41
4.3.1 ErrorCount ... 41
4.3.2 IsErrorPending ... 41
4.3.3 IsMessagePending ... 42

4.3.4 MessageCount ... 42
4.4 Beispieltask für MCAxis.Diagnostics ... 42
5 MCAxis.Drive.Amplifier ... 44
5.1 Syntax für MCAxis.Drive.Amplifier ... 44
5.2 Eigenschaften von MCAxis.Drive.Amplifier ... 44
5.2.1 ActualDCLinkVoltage ... 44
5.2.2 ActualHeatSinkTemperature ... 45
5.2.3 ActualInternalTemperature ... 45
5.2.4 DCLinkVoltageMax ... 45
5.2.5 DCLinkVoltageMin ... 46
5.2.6 PwmFrequency ... 46
5.2.7 PwmFrequencyReduction ... 47
5.2.8 RatedCurrent ... 47
5.2.9 RatedDCLinkVoltage ... 47
5.2.10 WireTestMode ... 48
5.2.11 I2tValue ... 48
6 MCAxis.Drive.Brake ... 49
6.1 Syntax für MCAxis.Drive.Brake ... 49
6.2 Funktionen von MCAxis.Drive.Brake ... 49
6.2.1 Apply() ... 49
6.2.2 Release() ... 50
6.3 Eigenschaften von MCAxis.Drive.Brake ... 50
6.3.1 ApplyDelay ... 50
6.3.2 ReleaseDelay ... 51
6.3.3 Mode ... 51
6.3.4 State ... 52
7 MCAxis.Drive.Control.Current ... 53
7.1 Syntax für MCAxis.Drive.Control.Current ... 53
7.2 Eigenschaften von MCAxis.Drive.Control.Current ... 53
7.2.1 Kp ... 53
7.2.2 Tn ... 54
7.2.3 ActualMotorCurrent ... 54
7.2.4 ActualMotorVoltage ... 55
7.2.5 MaxMotorCurrent ... 55
8 MCAxis.Drive.Control.Position ... 56
8.1 Syntax für MCAxis.Drive.Control.Position ... 56
8.2 Funktionen von MCAxis.Drive.Control.Position ... 56
8.2.1 ResetTrailingIndicators() ... 56

8.3 Eigenschaften von MCAxis.Drive.Control.Position ... 56
8.3.1 Kv ... 57
8.3.2 FeedbackSource ... 57
8.3.3 SpeedFeedForwardScale ... 58
9 MCAxis.Drive.Control.Speed ... 60
9.1 Syntax für MCAxis.Drive.Control.Speed ... 60
9.2 Eigenschaften von MCAxis.Drive.Control.Speed ... 60
9.2.1 Kp ... 60
9.2.2 Tn ... 61
9.2.3 Tf ... 62
9.2.4 ActualValue ... 62
9.2.5 Setpoint ... 63
9.2.6 Isum ... 63
9.2.7 IsumPreset ... 63
9.2.8 OverloadFactor ... 64
9.2.9 MaxMotorSpeed ... 64
9.2.10 SetpointLimit ... 65
9.2.11 DifferenceWindow ... 65
9.2.12 TotalInertia ... 65
9.2.13 TorqueFeedForwardScale ... 66
10 MCAxis.Drive.Control.Torque ... 67
10.1 Syntax für MCAxis.Drive.Control.Current ... 67
10.2 Eigenschaften von MCAxis.Drive.Control.Torque ... 67
10.2.1 ActualValue ... 67
10.2.2 Setpoint ... 68
10.2.3 OverloadFactor ... 68
10.2.4 MaxMotorTorque ... 68
10.2.5 SetpointLimit ... 69
11 MCAxis.Drive.Diagnostics ... 70
11.1 Syntax für MCAxis.Drive.Diagnostics ... 70
11.2 Funktionen von MCAxis.Drive.Diagnostics ... 70
11.2.1 IsWarningPending() ... 70
11.3 Eigenschaften von MCAxis.Drive.Diagnostics ... 71
11.3.1 ActualError ... 71
11.3.2 IsErrorPending ... 72
12 MCAxis.Drive.DigitalInput ... 73
12.1 Syntax für MCAxis.Drive.DigitalInput ... 73
12.2 Eigenschaften von MCAxis.Drive.DigitalInput ... 73

12.2.1 PolarityReferenceSwitch ... 73
12.2.2 LimitSwitchMode ... 74
12.2.3 PolarityLimitSwitchMin ... 74
12.2.4 PolarityLimitSwitchMax ... 75
12.2.5 IsPulseEnabled ... 75

13 MCAxis.Drive.EncoderChannel ... 76
13.1 Syntax für MCAxis.Drive.EncoderChannel ... 76
13.2 Eigenschaften von MCAxis.Drive.EncoderChannel ... 76
13.2.1 EncType ... 77
13.2.2 EncLines ... 77
13.2.3 GearMotor ... 78
13.2.4 GearEncoder ... 79
13.2.5 MultiTurnPosition ... 79
13.2.6 SingleTurnPosition ... 79
13.2.7 MaxNoOfAbsRevolutions ... 80

14 MCAxis.Drive.GenericAccess ... 81
14.1 Syntax für MCAxis.Drive.GenericAccess ... 81
14.2 Funktionen von MCAxis.Drive.GenericAccess ... 81
14.2.1 CANopen.ReadUInt8() ... 82
14.2.2 CANopen.WriteUInt8() ... 83
14.3 Eigenschaften von MCAxis.Drive.GenericAccess ... 83
14.3.1 CANopen.OSVersion ... 83
14.3.2 CANopen.IsAvailable ... 84

15 MCAxis.Drive.InvertDirection ... 85
15.1 Syntax für MCAxis.Drive.InvertDirection ... 85
15.2 Eigenschaften von MCAxis.Drive.InvertDirection ... 85
15.2.1 InvertDirection ... 85

16 MCAxis.Drive.IsReadyForOperation ... 87
16.1 Syntax für MCAxis.Drive.IsReadyForOperation ... 87
16.2 Eigenschaften von MCAxis.Drive.IsReadyForOperation ... 87
16.2.1 IsReadyForOperation ... 87

17 MCAxis.Drive.Motor ... 88
17.1 Syntax für MCAxis.Drive.Motor ... 88
17.2 Funktionen von MCAxis.Drive.Motor ... 88
17.2.1 MeasureCommutationOffset() ... 88
17.3 Eigenschaften von MCAxis.Drive.Motor ... 89
17.3.1 MotorDesign ... 90
17.3.2 RatedCurrent ... 91

17.3.3 MaxCurrent ... 92
17.3.4 RatedSpeed ... 92
17.3.5 RatedTorque ... 93
17.3.6 Inductance ... 93
17.3.7 Resistance ... 94
17.3.8 VoltageConstant ... 95
17.3.9 TorqueConstant ... 95
17.3.10 PolePairs ... 96
17.3.11 CommutationOffset ... 97
17.3.12 ThermalDelayTime ... 98
17.3.13 ShaftPosition ... 98
17.3.14 Inertia ... 99
17.3.15 TemperatureSensorType ... 99
17.3.16 TemperatureSensorConnector ... 100
17.3.17 Temperature ... 101
17.3.18 I2tOperatingMode ... 102
17.3.19 I2tValue ... 102

18 MCAxis.Drive.QuickStopDeceleration ... 104
18.1 Syntax für MCAxis.Drive.QuickStopDeceleration ... 104
18.2 Eigenschaften von MCAxis.Drive.QuickStopDeceleration ... 104
18.2.1 QuickStopDeceleration ... 104

19 MCAxis.Drive.TorqueShutOff ... 105
19.1 Syntax für MCAxis.Drive.TorqueShutOff ... 107
19.2 Funktionen von MCAxis.Drive.TorqueShutOff ... 108
19.2.1 Configure() ... 108
19.2.2 ReadConfiguration() ... 109
19.2.3 Activate() ... 110
19.2.4 Deactivate() ... 110
19.3 Eigenschaften von MCAxis.Drive.TorqueShutOff ... 111
19.3.1 State ... 111
19.4 Beispieltask MCAxis.Drive.TorqueShutOff ... 111

20 MCAxis.Info ... 113
20.1 Syntax für MCAxis.Info ... 113
20.2 Eigenschaften der Funktionalität ... 113
20.2.1 Name ... 113
20.2.2 ObjectID ... 113
20.2.3 ClassID ... 114
20.3 Beispieltask MCAxis.Info ... 115

21 MCAxis.IsInTargetWindow 116
21.1 Syntax für MCAxis.IsInTargetWindow 116
21.2 Eigenschaften von MCAxis.IsInTartegWindow 116
21.2.1 IsInTargetWindow 116
21.3 Beispieltask MCAxis.IsInTargetWindow 116
22 MCAxis.Mechanism.AxisMode 118
22.1 Syntax für MCAxis.Mechanism.AxisMode 119
22.2 Eigenschaften von MCAxis.Mechanism.AxisMode 119
22.2.1 AxisMode 119
23 MCAxis.Mechanism.AxisType 120
23.1 Syntax für MCAxis.Mechanism.AxisType 120
23.2 Eigenschaften von MCAxis.Mechanism.AxisType 120
23.2.1 AxisType 120
23.3 Beschreibung der Achstypen 121
23.3.1 Rotatorische Achse 121
23.3.2 Lineare Achse 122
23.3.3 Nichtlineare Exzenterachse 123
23.3.4 Nichtlineare Kolbenachse 124
23.4 Beispieltask für MCAxis.Mechanism.AxisType 125
24 MCAxis.Mechanism.GearRatioLoad, MCAxis.Mechanism.GearRatioMotor 126
24.1 Syntax für MCAxis.Mechanism.GearRatioLoad, MCAxis.Mechanism.GearRatioMotor 126
24.2 Eigenschaften von MCAxis.Mechanism.GearRatioLoad, MCAxis.Mechanism.GearRatioMotor 126
24.2.1 GearRatioMotor 126
24.2.2 GearRatioLoad 127
25 MCAxis.Mechanism.Kinematics 128
25.1 Syntax für MCAxis.Mechanism.Kinematics 128
25.2 Funktionen von MCAxis.Mechanism.Kinematics 128
25.2.1 Crank.Configure(), Piston.Configure() 128
25.3 Eigenschaften von MCAxis.Mechanism.Kinematics 129
25.3.1 .Offset / .InvertDirection / .Radius / .RodLength 129
25.4 Beispieltask MCAxis.Mechanism.Kinematics 130
26 MCAxis.Mechanism.Limitations 131
26.1 Syntax für MCAxis.Mechanism.Limitations 131
26.2 Funktionen von MCAxis.Mechanism.Limitations 131
26.2.1 SetTravelLimits() 131
26.3 Eigenschaften von MCAxis.Mechanism.Limitations 132
26.3.1 TravelLimitMin 132
26.3.2 TravelLimitMax 133

26.3.3 MaxSpeed 134
26.3.4 MaxAcceleration 134
26.3.5 MaxJerk 135

27 MCAxis.Mechanism.LinearRotationRatio 136
27.1 Syntax für MCAxis.Mechanism.LinearRotationRatio 136
27.2 Eigenschaften von MCAxis.Mechanism.LinearRotationRatio 136
27.2.1 LinearRotationRatio 136

28 MCAxis.Mechanism.Monitoring 138
28.1 Syntax für MCAxis.Mechanism.Monitoring 138
28.2 Eigenschaften von MCAxis.Mechanism.Monitoring 138
28.2.1 IsReferenceSwitchEnabled 138
28.2.2 IsHWLimitSwitchMinActive 139
28.2.3 IsHWLimitSwitchMaxActive 139

29 MCAxis.Mechanism.Slope 140
29.1 Syntax für MCAxis.Mechanism.Slope 140
29.2 Eigenschaften von MCAxis.Mechanism.Slope 140
29.2.1 SlopeState 141
29.2.2 IsAccerlerating 141
29.2.3 IsAtConstantSpeed 141
29.2.4 IsDecelerating 142
29.2.5 IsStopped 142
29.2.6 SlopeType 142
29.2.7 ActualMaxSpeed 143
29.2.8 ActualMaxAcceleration 143
29.2.9 ActualMaxDeceleration 144
29.2.10 AccelerationSetpoint 144
29.2.11 SpeedSetpoint 144
29.2.12 Speed 145
29.2.13 Acceleration 145
29.2.14 Deceleration 146
29.2.15 Config.RampBreakMode 146

30 MCAxis.Mechanism.Units 148
30.1 Syntax für MCAxis.Mechanism.Units 148
30.2 Funktionen von MCAxis.Mechanism.Units 148
30.2.1 SetLinearUnits() 148
30.2.2 SetRotaryUnits() 149
30.3 Eigenschaften von MCAxis.Mechanism.Units 150
30.3.1 LinearPositionUnit 150

30.3.2 LinearSpeedUnit 151
30.3.3 LinearAccelerationUnit 151
30.3.4 LinearJerkUnit 152
30.3.5 RotaryPositionUnit 152
30.3.6 RotatorySpeedUnit 153
30.3.7 RotatoryAccelerationUnit 153
30.3.8 RotatoryJerkUnit 154

31 MCAxis.MoveHalt 155

31.1 Syntax für MCAxis.MoveHalt 155
31.2 Funktionen von MCAxis.MoveHalt 155
31.2.1 Start() 155
31.3 Beispieltask MCAxis.MoveHalt 157

32 MCAxis.MoveHome 158

32.1 Syntax für MCAxis.MoveHome 158
32.2 Funktionen von MCAxis.MoveHome 158
32.2.1 Start() 158
32.2.2 SetReference() 160
32.2.3 SetAbsEncoderReference() 161
32.2.4 ClearAbsEncoderReferenceStatus() 162
32.3 Eigenschaften von MCAxis.MoveHome 162
32.3.1 Direction 163
32.3.2 SwitchConfig 164
32.3.3 Mode 165
32.3.4 Acceleration 166
32.3.5 SpeedSwitchSearch 166
32.3.6 MaxDistanceSwitchSearch 167
32.3.7 SpeedReferenceSearch 168
32.3.8 MaxDistanceReferenceSearch 168
32.3.9 ReferenceOffset 169
32.3.10 NormalPosition 170
32.3.11 ReferenceState 170
32.3.12 IsReferenceRunStarted, IsSwitchFound, IsReferenceFound, IsReferenceSet 171
32.3.13 IsAbsEncoderReferenceValid 172
32.3.14 ReferenceShift 172

33 MCAxis.MovePtp 174

33.1 Syntax für MCAxis.MovePtp 174
33.2 Funktionen von MCAxis.MovePtp 174
33.2.1 Start() 174

33.2.2 StartNewTargetPosition() 177
33.2.3 NewSpeed() 180
33.2.4 NewAcceleration() 181
33.2.5 NewDeceleration() 182
33.3 Eigenschaften von MCAxis.MovePtp 183
33.3.1 PositioningMode 183
33.3.2 TargetPosition 184
33.3.3 TargetWindow 185
33.4 Beispieltask MCAxis.MovePtp 186
34 MCAxis.MoveVelocity 188
34.1 Syntax für MCAxis.MoveVelocity 188
34.2 Funktionen von MCAxis.MoveVelocity 188
34.2.1 Start() 188
34.2.2 Reverse() 191
34.2.3 NewSpeed() 191
34.2.4 NewAcceleration() 192
34.2.5 NewDeceleration() 193
34.3 Eigenschaften von MCAxis.MoveVelocity 194
34.3.1 Direction 194
34.4 Beispieltask MCAxis.MoveVelocity 195
35 MCAxis.Position 196
35.1 Syntax für MCAxis.Position 196
35.2 Funktionen von MCAxis.Position 196
35.2.1 AbsEncOverflowHandler.Activate() 196
35.2.2 AbsEncOverflowHandler.Deactivate() 197
35.3 Eigenschaften von MCAxis.Position 197
35.3.1 Target 198
35.3.2 Actual 198
35.3.3 Setpoint 198
35.3.4 EncoderPosition 199
35.3.5 AcutalTrackingError 199
35.3.6 TrailingIndicatorMax 199
35.3.7 TrailingIndicatorMin 200
35.3.8 AbsEncOverflowHandler.IsActive 200
36 MCAxis.Power 201
36.1 Syntax für MCAxis.Power 201
36.2 Funktionen von MCAxis.Power 201
36.2.1 Enable() 201

36.2.2 Disable() 202
36.2.3 QuickStop() 203
36.3 Beispieltask MCAxis.Power 204
37 MCAxis.Settings 205
37.1 Syntax für MCAxis.Settings 205
37.2 Funktionen von MCAxis.Settings 205
37.2.1 SaveToFile() 206
37.2.2 LoadFromFile() 206
38 MCAxis.State 208
38.1 Syntax für MCAxis.State 210
38.2 Eigenschaften von MCAxis.State 211
38.2.1 Value 211
38.2.2 IsDisabled 211
38.2.3 IsEnabled 212
38.2.4 IsRunning 212
38.2.5 IsSuspended 212
38.2.6 IsDisabledError 213
38.2.7 IsEnabledError 213
38.2.8 IsInactive 213
38.2.9 OperatingMode.Value 213
38.2.10 TypeOfMotion.Value 216
38.3 Beispieltask MCAxis.State 220
39 MCAxis.State.Techno 221
39.1 Syntax für MCAxis.State.Techno 221
39.2 Eigenschaften von MCAxis.State.Techno 221
39.2.1 TechnoID 221
39.2.2 SyncState 222
39.2.3 IsDecoupled 222
39.2.4 IsDecoupling 222
39.2.5 IsCoupling 223
39.2.6 IsCoupled 223
39.2.7 SyncPosition 224
39.2.8 Cam.CamID 224
39.2.9 Cam.CamSegmentID 225
39.2.10 Cam.MasterOffset 225
39.2.11 Cam.FollowerOffset 226
39.2.12 Cam.MasterFactor 226
39.2.13 Cam.FollowerFactor 227

40 MCAxis.State.Transitions 228
40.1 Syntax für MCAxis.State.Transitions 230
40.2 Funktionen von MCAxis.State.Transitions 231
40.2.1 Enable() 231
40.2.2 Disable() 231
40.2.3 ErrorDisable() 231
40.2.4 ErrorStop() 232
40.2.5 ErrorLockUserInput() 233
40.2.6 Suspend() 233
40.2.7 Resume() 234
40.2.8 UserPositionReset() 235
40.3 Eigenschaften von MCAxis.State.Transitions 235
40.3.1 ErrorDisableMode 236
40.3.2 ErrorStopMode 236
41 MCAxis.Stop 237
41.1 Funktionen von MCAxis.Stop 237
41.1.1 Stop() 237
41.2 Beispieltask MCAxis.Stop 238
42 MCAxis.TouchProbe 240
42.1 Syntax für MCAxis.TouchProbe 243
42.2 Funktionen von MCAxis.TouchProbe 243
42.2.1 Configure() 243
42.2.2 Enable() 244
42.2.3 Disable() 245
42.2.4 ReadConfiguration() 245
42.2.5 ReadState() 246
42.3 Beispieltask MCAxis.TouchProbe 248
43 MCCouplingGroup 250
43.1 Syntax für MCCouplingGroup 250
43.2 Funktionen von MCCouplingGroup 251
43.2.1 AddFollowerAxis() 251
43.2.2 Redefine() 251
43.2.3 Undefine() 252
43.3 Eigenschaften von MCCouplingGroup 252
43.3.1 FollowerCount 253
43.3.2 MasterAxis 253
43.3.3 FollowerAxis[] 253
43.3.4 Name 254

44 MCGantryGroup 255
44.1 Syntax für MCGantryGroup 256
44.2 Funktionen von MCGantryGroup 256
44.2.1 AddFollowerAxis() 256
44.2.2 Redefine() 256
44.2.3 Undefine() 257
44.3 Eigenschaften von MCGantryGroup 257
44.3.1 GantryWindow[i] 258
44.3.2 ConfigGantryWindow[i] 258
44.3.3 FollowerCount 258
44.3.4 MasterAxis 259
44.3.5 FollowerAxis[i] 259
44.3.6 Name 259
44.4 Beispieltask MCGantryGroup 260
45 MCGeo.Activate, MCGeo.Deactivate 262
45.1 Syntax für MCGeo.Activate, MCGeo.Deactivate 262
45.2 Funktionen von MCGeo.Activate, MCGeo.Deactivate 262
45.2.1 Activate() 262
45.2.2 Deactivate() 263
46 MCGeo.Buffer 265
46.1 Syntax für MCGeo.Buffer 265
46.2 Funktionen von MCGeo.Buffer 265
46.2.1 Clear() 265
46.2.2 Suspend() 266
46.2.3 Resume() 267
46.2.4 UserIndex.Enable() 267
46.2.5 UserIndex.Disable() 268
46.2.6 UserIndex.Reset() 269
46.3 Eigenschaften von MCGeo.Buffer 269
46.3.1 UserIndex.Actual 269
47 MCGeo.Coordinates 270
47.1 Syntax für MCGeo.Coordinates 270
47.2 Eigenschaften von MCGeo.Coordinates 270
47.2.1 Actual 270
47.2.2 Setpoint 271
47.2.3 Target 271
48 MCGeo.Coupling 273
48.1 Syntax für MCGeo.Coupling 274

48.2 Funktionen von MCGeo.Coupling 274
48.2.1 Couple() 274
48.2.2 Decouple() 276
48.2.3 GetPositioningMode() 277
48.2.4 GetCoupleType() 277
48.2.5 GetCoupleMode() 277
48.2.6 GetDecoupleMode() 278
48.3 Eigenschaften von MCGeo.Coupling 278
48.3.1 CoupleState 278
49 MCGeo.Diagnostics 279
49.1 Syntax für MCGeo.Diagnostics 279
49.2 Funktionen von MCGeo.Diagnostics 279
49.2.1 ClearErrors(), Software.ClearErrors() – Bahnverbund 279
49.2.2 Software.GetErrorCode() 280
49.2.3 Software.GetMessageCode() 280
49.3 Eigenschaften von MCGeo.Diagnostics 281
49.3.1 Software.ErrorCount 281
49.3.2 IsErrorPending, Software.IsErrorPending 281
49.3.3 Software.MessageCount 282
49.3.4 IsMessagePending, Software.IsMessagePending 282
49.4 Beispieltask für MCGeo.Diagnostics 282
50 MCGeo.Events 284
50.1 Syntax für MCGeo.Events 284
50.2 Funktionen von MCGeo.Events 284
50.2.1 SetFlag() 284
50.2.2 ReadFlag() 285
50.2.3 ReadFlags() 286
50.2.4 ClearFlags() 286
50.3 Beispieltask MCGeo.Events 286
51 MCGeo.Info 288
51.1 Syntax für MCGeo.Info 288
51.2 Eigenschaften der Funktionalität 288
51.2.1 Name 288
51.2.2 ObjectID 288
51.2.3 ClassID 289
51.3 Beispieltask MCGeo.Info 289
52 MCGeo.Mechanism.Kinematics 290
52.1 Syntax für MCGeo.Mechanism.Kinematics 290

52.2 Eigenschaften von MCGeo.Mechanism.Kinematics 290
52.2.1 Cartestian2D 290
52.2.2 Cartessian3D 291
52.2.3 Scara 292
52.3 Konfigurationsfunktionen von MCGeo.Mechanism.Kinematics 293
52.3.1 Cartesian2D.X.Configure(),Cartesian2D.Y.Configure() 293
52.3.2 Cartesian3D.X.Configure(), Cartesian3D.Y.Configure() 294
52.3.3 Scara.Shoulder.Configure(), Scara.Ellbow.Configure() 295
52.3.4 Scara.Vertical.Configure() 296
52.3.5 Scara.Wrist.Configure() 296
53 MCGeo.Mechanism.Limitations 298
53.1 Syntax für MCGeo.Mechanism.Limitations 299
53.2 Eigenschaften von MCGeo.Mechanism.Limitations 299
53.2.1 WorkspaceLimitMin 300
53.2.2 WorkspaceLimitMax 301
53.2.3 OrientationLimitMin 301
53.2.4 OrientationLimitMax 302
53.2.5 OrientationFilter 303
53.2.6 MaxPathVelocity 304
53.2.7 MaxPathAcceleration 305
53.2.8 MaxPathJerk 305
53.2.9 SingularityDistance1 306
54 MCGeo.Mechanism.Slope 308
54.1 Syntax für MCGeo.Mechanism.Slope 308
54.2 Eigenschaften von MCGeo.Mechanism.Slope 308
54.2.1 SlopeState 309
54.2.2 IsAccerlerating 309
54.2.3 IsAtConstantSpeed 309
54.2.4 IsDecelerating 310
54.2.5 IsStopped 310
54.2.6 ActualMaxSpeed 310
54.2.7 ActualMaxAcceleration 311
54.2.8 SpeedSetpoint 311
54.2.9 AccelerationSetpoint 311
54.2.10 Speed 312
54.2.11 MaxVeloPoly 312
54.2.12 SpeedOverride 312
54.2.13 Acceleration 313

55 MCGeo.Mechanism.Units 314
55.1 Syntax für MCGeo.Mechanism.Units 314
55.2 Funktionen von MCGeo.Mechanism.Units 314
55.2.1 SetUnits() 314
55.3 Eigenschaften von MCGeo.Mechanism.Units 316
55.3.1 PositionUnit 316
55.3.2 OrientationUnit 316
55.3.3 SpeedUnit 317
55.3.4 AccelerationUnit 317
55.3.5 JerkUnit 318
56 MCGeo.Members 319
56.1 Syntax für MCGeo.Members 319
56.2 Funktionen von MCGeo.Members 319
56.2.1 AddSyncAxis() 319
56.2.2 RemoveSyncAxis() 320
57 MCGeo.MoveCircle2D 321
57.1 Syntax für MCGeo.MoveCircle2D 321
57.2 Funktionen von MCGeo.MoveCircle2D 321
57.2.1 Start() 322
57.3 Eigenschaften von MCGeo.MoveCircle2D 324
57.3.1 PositioningMode 324
57.3.2 BlockEnd 325
57.3.3 CoordinateSystemPlane 326
57.3.4 TravelAngle 327
57.3.5 TargetPositionHelixAxis 327
57.3.6 SmoothingDistance 327
57.3.7 Delay 328
57.3.8 TargetWindow 328
57.3.9 CenterPoint 329
57.3.10 TargetOrientation 329
58 MCGeo.MoveCircle3D 330
58.1 Syntax für MCGeo.MoveCircle3D 331
58.2 Funktionen von MCGeo.MoveCircle3D 331
58.2.1 Start() 331
58.3 Eigenschaften von MCGeo.MoveCircle3D 333
58.3.1 PositioningMode 333
58.3.2 BlockEnd 334
58.3.3 SmoothingDistance 335

58.3.4 Delay 336
58.3.5 TargetWindow 336
58.3.6 IntermediatePoint 336
58.3.7 TargetPosition 337
58.3.8 TargetOrientation 337

59 MCGeo.MoveLinear 339
59.1 Syntax für MCGeo.MoveLinear 339
59.2 Funktionen von MCGeo.MoveLinear 339
59.2.1 Start() 339
59.3 Eigenschaften von MCGeo.MoveLinear 341
59.3.1 PositioningMode 341
59.3.2 BlockEnd 342
59.3.3 SmoothingDistance 343
59.3.4 Delay 344
59.3.5 TargetWindow 344
59.3.6 TargetCoordinates 344
59.3.7 TargetPosition 345
59.3.8 TargetOrientation 345

60 MCGeo.MoveOrientation 347
60.1 Syntax für MCGeo.MoveOrientation 347
60.2 Funktionen von MCGeo.MoveOrientation 347
60.2.1 Start() 347
60.3 Eigenschaften von MCGeo.MoveOrientation 349
60.3.1 PositioningMode 349
60.3.2 BlockEnd 350
60.3.3 Speed 351
60.3.4 Acceleration 351
60.3.5 Jerk 351
60.3.6 Delay 352
60.3.7 TargetWindow 352
60.3.8 TargetOrientation 352

61 MCGeo.MovePtp 354
61.1 Syntax für MCGeo.MovePtp 354
61.2 Funktionen von MCGeo.MovePtp 354
61.2.1 Start() 354
61.3 Eigenschaften von MCGeo.MovePtp 356
61.3.1 PositioningMode 356
61.3.2 BlockEnd 357

61.3.3 Delay 358
61.3.4 TargetWindow 358
61.3.5 TargetPosition 358
61.3.6 TargetOrientation 359
61.3.7 TargetCoordinates 359
62 MCGeo.MoveSpline 361
62.1 Syntax für MCGeo.MoveSpline 361
62.2 Funktionen von MCGeo.MoveCircle2D 361
62.2.1 Start() 361
62.3 Eigenschaften von MCGeo.MoveSpline 364
62.3.1 NodeCount 364
62.3.2 PositioningMode 365
62.3.3 BlockEnd 366
62.3.4 StartCondition 366
62.3.5 TargetCondition 367
62.3.6 SmoothingDistance 368
62.3.7 Delay 369
62.3.8 TargetWindow 369
62.3.9 VirtualStartPosition[x] 369
62.3.10 VirtualTargetPosition[x] 370
62.3.11 NodePosition[n] 370
62.3.12 NodeOrientation[n] 371
62.3.13 NodeCoordinates[n] 372
63 MCGeo.MoveSyncAxis 373
63.1 Syntax für MCGeo.MoveSyncAxis 373
63.2 Funktionen von MCGeo.MoveSyncAxis 373
63.2.1 Start() 373
63.2.2 GetTargetPosition() 375
63.2.3 SetTargetPosition() 375
63.2.4 GetPositioningMode() 375
63.2.5 SetPositioningMode() 376
64 MCGeo.Orientation 377
64.1 Syntax für MCGeo.Orientation 377
64.2 Eigenschaften von MCGeo.Orientation 377
64.2.1 Actual 378
64.2.2 Setpoint 378
64.2.3 Target 379
65 MCGeo.Position 380

65.1 Syntax für MCGeo.Position 380
65.2 Eigenschaften von MCGeo.Position 380
65.2.1 Actual 380
65.2.2 Setpoint 381
65.2.3 Target 381
66 MCGeo.Power 382
66.1 Syntax für MCGeo.Power 382
66.2 Funktionen von MCGeo.Power 382
66.2.1 Enable() 382
66.2.2 Disable() 383
66.2.3 QuickStop() 384
66.3 Beispieltask MCGeo.Power 385
67 MCGeo.Settings 386
67.1 Syntax für MCGeo.Settings 386
67.2 Funktionen von MCGeo.Settings 386
67.2.1 SaveToFile() 387
67.2.2 LoadFromFile() 387
68 MCGeo.State 389
68.1 Eigenschaften von MCGeo.State 391
68.1.1 Value 392
68.1.2 IsDisabled 392
68.1.3 IsDisabledError 392
68.1.4 IsEnabled 393
68.1.5 IsEnabledError 393
68.1.6 IsInactive 393
68.1.7 IsRunning 394
68.1.8 IsSuspended 394
68.1.9 OperatingMode.Value 394
68.1.10 TypeOfMotion.Value 396
68.2 Beispieltask MCGeo.State 399
69 MCGeo.State.Transitions 400
69.1 Syntax für MCGeo.State.Transitions 402
69.2 Funktionen von MCGeo.State.Transitions 402
69.2.1 Enable() 403
69.2.2 Disable() 403
69.2.3 Stop() 403
69.2.4 ErrorDisable() 405
69.2.5 ErrorStop() 405

69.2.6 ErrorLockUserInput() 406
69.2.7 Suspend() 406
69.2.8 ErrorSuspend() 407
69.2.9 Resume() 408
69.3 Eigenschaften von MCGeo.State.Transitions 409
69.3.1 ErrorDisableMode 409
69.3.2 ErrorStopMode 409
69.3.3 ErrorLevel 410
70 MCGeo.Stop 411
70.1 Syntax für MCGeo.Stop 411
70.2 Funktionen von MCGeo.Stop 411
70.2.1 Stop() 411
71 MCManager (MCMgr) 413
71.1 Syntax für MCMgr 413
71.2 Funktionen von MCMgr 413
71.2.1 CheckVersions() 413
71.2.2 GetDataSource() 414
71.2.3 WriteLogToFile() 414
71.2.4 GetAxisByID() 415
71.2.5 GetAxisByName() 415
71.2.6 GetTechnoByID() 415
71.2.7 GetTechnoByName() 416
71.2.8 GetGeoByID() 416
71.2.9 GetTechnoByName() 416
71.3 Eigenschaften von MCMgr 417
71.3.1 AxisCount 417
71.3.2 AxisList[] 417
71.3.3 TechnoCount 418
71.3.4 TechnoList[] 418
71.3.5 GeoCount 418
71.3.6 GeoList[] 419
71.3.7 FaultyObjectCount 419
71.3.8 FaultyObjectList[] 419
71.3.9 Name 421
71.3.10 Version 421
71.3.11 IsInitialized 421
72 MCTechno.Activate, MCTechno.Deactivate 422
72.1 Syntax für MCTechno.Activate, MCTechno.Deactivate 422

72.2 Funktionen von MCTechno.Activate, MCTechno.Deactivate 422
72.2.1 Activate() 422
72.2.2 Deactivate() 423
73 MCTechno.Coupling 425
73.1 Syntax für MCTechno.Coupling 425
73.2 Funktionen von MCTechno.Coupling 425
73.2.1 Couple() 426
73.2.2 Decouple() 428
73.2.3 GetCoupleMasterPosition() 429
73.2.4 GetCoupleFollowerPosition() 429
73.2.5 GetCoupleMasterDirection() 429
73.2.6 GetDecoupleDeceleration() 430
73.2.7 GetDecoupleMasterPosition() 430
73.2.8 GetDecoupleMode() 431
73.2.9 IsCoupling() 431
73.2.10 IsCoupled() 431
73.2.11 IsDecoupling() 432
73.2.12 IsDecoupled() 432
73.2.13 Filter.GetCoefficient(), Filter.SetCoefficient() 432
73.2.14 Gearing.Configure() 433
73.2.15 Gearing.GetFactor() 435
73.2.16 Gearing.GetDivisor() 435
73.2.17 Gearing.GetActivationMode() 435
73.3 Eigenschaften von MCTechno.Coupling 436
73.3.1 ExternalMaster.Offset 436
73.3.2 ExternalMaster.Smoothing 437
74 MCTechno.Coupling.Cam 438
74.1 Syntax für MCTechno.Coupling.Cam 438
74.2 Funktionen von MCTechno.Coupling.Cam 438
74.2.1 Create() 438
74.2.2 DefineSegment() 439
74.2.3 DefineSegmentAuto() 441
74.2.4 Activate() 443
74.2.5 ReadSegment() 444
74.2.6 Evaluate() 445
74.2.7 AnalyseSegment() 446
74.2.8 AnalyseSegmentDerivates() 448
75 MCTechno.Diagnostics 451

75.1 Syntax für MCTechno.Diagnostics 451
75.2 Funktionen von MCTechno.Diagnostics 451
75.2.1 ClearErrors(), Software.ClearErrors() – Technologieverbund 451
75.2.2 Software.GetErrorCode() 452
75.2.3 Software.GetMessageCode() 452
75.3 Eigenschaften von MCTechno.Diagnostics 453
75.3.1 Software.ErrorCount 453
75.3.2 IsErrorPending, Software.IsErrorPending 453
75.3.3 Software.MessageCount 454
75.3.4 IsMessagePending, Software.IsMessagePending 454
75.4 Beispieltask für MCTechno.Diagnostics 455
76 MCTechno.Info 456
76.1 Syntax für MCTechno.Info 456
76.2 Eigenschaften der Funktionalität 456
76.2.1 Name 456
76.2.2 ObjectID 456
76.2.3 ClassID 457
76.3 Beispieltask MCTechno.Info 457
77 MCTechno.Mechanism.Slope 458
77.1 Syntax für MCTechno.Mechanism.Slope 458
77.2 Funktionen von MCTechno.Mechanism.Slope 458
77.2.1 GetSlopeType(), SetSlopeType() 458
77.2.2 GetSpeed(), SetSpeed() 459
77.2.3 GetAcceleration(), SetAcceleration() 460
77.2.4 GetDeceleration(), SetDeceleration() 460
77.3 Eigenschaften von MCTechno.Mechanism.Slope 461
77.3.1 SlopeState 461
77.3.2 IsAccerlerating 462
77.3.3 IsAtConstantSpeed 462
77.3.4 IsDecelerating 462
77.3.5 IsStopped 462
78 MCTechno.Members 464
78.1 Syntax für MCTechno.Members 464
78.2 Eigenschaften von MCTechno.Members 464
78.2.1 FollowerCount 464
78.2.2 Follower[n] 465
78.2.3 FollowerCamCount 465
78.2.4 FollowerCam[n] 465

78.2.5 FollowerGearCount 466
78.2.6 FollowerGear[n] 466
78.2.7 Master 466
79 MCTechno.MoveHalt 467
79.1 Syntax für MCTechno.MoveHalt 467
79.2 Funktionen von MCTechno.MoveHalt 467
79.2.1 Start() 467
80 MCTechno.MovePtp 470
80.1 Syntax für MCTechno.MovePtp 470
80.2 Funktionen von MCTechno.MovePtp 470
80.2.1 Start() 470
80.2.2 StartNewTargetPosition() 473
80.2.3 NewSpeed() 474
80.2.4 NewAcceleration() 475
80.2.5 NewDeceleration() 476
80.2.6 GetPositioningMode(), SetPositioningMode() 477
80.2.7 GetTargetPosition(),SetTargetPosition() 479
80.2.8 GetTargetWindow(), SetTargetWindow() 480
80.2.9 StartSuperPose() 481
80.2.10 StartNewTargetPositionSuperPose() 482
81 MCTechno.MoveVelocity 484
81.1 Syntax für MCTechno.MoveVelocity 484
81.2 Funktionen von MCTechno.MoveVelocity 484
81.2.1 Start() 484
81.2.2 Reverse() 487
81.2.3 NewSpeed() 488
81.2.4 NewAcceleration() 489
81.2.5 NewDeceleration() 490
81.2.6 GetDirection(), SetDirection() 491
81.2.7 StartSuperPose() 492
81.2.8 ReverseSuperPose() 494
82 MCTechno.Power 496
82.1 Syntax für MCTechno.Power 496
82.2 Funktionen von MCTechno.Power 496
82.2.1 Enable() 496
82.2.2 Disable() 497
82.2.3 QuickStop() 498
82.3 Beispieltask MCTechno.Power 499

83 MCTechno.Settings 500
83.1 Syntax für MCTechno.Settings 500
83.2 Funktionen von MCTechno.Settings 500
83.2.1 SaveToFile() 501
83.2.2 LoadFromFile() 501
84 MCTechno.State 503
84.1 Syntax für MCTechno.State 505
84.2 Eigenschaften von MCTechno.State 505
84.2.1 Value 506
84.2.2 IsDisabled 506
84.2.3 IsDisabledError 506
84.2.4 IsEnabled 507
84.2.5 IsEnabledError 507
84.2.6 IsInactive 507
84.2.7 IsRunning 508
84.2.8 IsSuspended 508
84.2.9 OperatingMode.Value 508
84.2.10 TypeOfMotion.Value 510
85 MCTechno.State.Transitions 512
85.1 Syntax für MCTechno.State.Transitions 514
85.2 Funktionen von MCTechno.State.Transitions 514
85.2.1 Enable() 515
85.2.2 Disable() 515
85.2.3 Stop() 515
85.2.4 ErrorDisable() 517
85.2.5 ErrorStop() 517
85.2.6 ErrorLockUserInput() 518
85.2.7 Suspend() 518
85.2.8 Resume() 519
85.3 Eigenschaften von MCTechno.State.Transitions 520
85.3.1 ErrorDisableMode 520
85.3.2 ErrorStopMode 521
85.3.3 ErrorLevel 521
86 MCTechno.Stop 523
86.1 Syntax für MCTechno.Stop 523
86.2 Funktionen von MCTechno.Stop 523
86.2.1 Stop() 523
Glossar 525

1 Einleitung

1.1 Informationen zum Dokument

Dieses Dokument ist Teil des Produkts und muss vor dem Einsatz des Geräts gelesen und verstanden werden. Es enthält wichtige und sicherheitsrelevante Informationen, um das Produkt sachgerecht und bestimmungsgemäß zu betreiben.

Start | www.bucherautomation.com

Folgende Informationsprodukte ergänzen dieses Dokument:

– Application Notes
  Fachberichte und Anwendungsbeispiele
– Themenhandbücher
  Produktübergreifende Dokumentation

1.2 Darstellungskonventionen

Unterschiedliche Formatierungen erleichtern es, Informationen zu finden und einzuordnen. Im Folgenden das Beispiel einer Schritt-für-Schritt-Anweisung:

✓ Dieses Zeichen weist auf eine Voraussetzung hin, die vor dem Ausführen der nachfolgenden Handlung erfüllt sein muss.

► Dieses Zeichen oder eine Nummerierung zu Beginn eines Absatzes markiert eine Handlungsanweisung, die vom Benutzer ausgeführt werden muss. Arbeiten Sie Handlungsanweisungen der Reihe nach ab.

⇨ Der Pfeil nach Handlungsanweisungen zeigt Reaktionen oder Ergebnisse dieser Handlungen auf.

ⓘ Info	**Weiterführende Informationen und praktische Tipps** In der Info-Box finden Sie weiterführende Informationen und praktische Tipps zu Ihrem Produkt.

2 MCAxis.AxisConfigData

Mit der Funktionalität **AxisConfigData** fragen Sie verschiedene Informationen einer MCX-Achse ab, die sich aus der Konfiguration im Programmiertool JetSym ergeben.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

2.1 Syntax für MCAxis.AxisConfigData

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.AxisConfigData.<Zugriff auf Wert>;
```

2.2 Eigenschaften von MCAxis.AxisConfigData

Die Funktionalität bietet folgende Eigenschaften:

- DeviceName
- AxisMode
- AxisDriveType
- ObjectID
- DeviceID
- IsSimulation

2.2.1 DeviceName

Beschreibung

Datentyp	Bedeutung
String	Liest den in JetSym für diese der MCX-Achse vergebenen Namen. Dieser Name bezeichnet auch das Achsobjekt, das von JetSym für die Programmierung angelegt wird.

Syntax

```
AchsName := AX1.AxisConfigData.DeviceName;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

2.2.2 AxisMode

Beschreibung

Datentyp	Bedeutung
Typ MCAxisModes	Definiert den Achstyp.
MCAxisModes.InternalAxis	Voll funktionsfähige, reale Achse mit Servoverstärker und Motor.
MCAxisModes.ExternalAxis	Achse, die nur über eine Ist-Position verfügt. Sie kann als Leitachse in einem Techno-Verbund verwendet werden. Definiert wird sie über den zweiten Drehgeber einer realen Achse.
MCAxisModes.SimulatedAxis	Achse, die sich wie eine interne Achse verhält, bei der jedoch Servoverstärker und Motor simuliert werden. Diese Konfiguration ermöglicht es, ein Programm zu erstellen und zu testen, wenn die realen Hardwarekomponenten noch nicht zur Verfügung stehen.
MCAxisModes.VirtualAxis	Achse, die im MCX-Kernel berechnet wird. Sie dient als ideale Leitachse für Techno-Verbünde.
MCAxisModes.ShadowAxis	Mitlaufachse, die in erster Linie dazu dient, kaskadierte Techno-Verbünde zu generieren. Sie übernimmt immer die Soll- und Ist-Position einer anderen (virtuellen oder realen) Achse. Eine Mitlaufachse muss immer Leitachse in einem Technologieverbund sein.

Syntax

```
ArtDerAchse := AX1.AxisConfigData.AxisMode;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

2.2.3 AxisDriveType

Beschreibung

Mit der Eigenschaft lesen Sie die grundsätzlichen Merkmale der MCX-Achse. Diese Information dient dazu, bei Schleifenbefehlen über alle Achsen zu unterscheiden, ob ein Befehl für diese Achse sinnvoll ist oder nicht. Beispielsweise ergibt es bei Achsen mit `MCAxisDriveType.None` keinen Sinn, Parameter, die sich auf den Servoverstärker beziehen, zu lesen oder zu beschreiben.

Datentyp	Bedeutung
MCAxisDriveTypes	Definiert den Achstyp.
MCAxisDriveTypes.Managed	Reale Achse (`InternalAxis`), externe Achse (`ExternalAxis`), simulierte (reale) Achse (`SimulatedAxis`).
MCAxisDriveTypes.None	Virtuelle Achse, Mitlaufachse (`ShadowAxis`)

Syntax

```
AchsTyp := AX1.AxisConfigData.AxisDriveType;
```

Zustandsverwaltung

Die Eigenschaft dient der Diagnose. Auf sie kann nur lesend zugegriffen werden. Der Zustand der Achse bleibt unverändert.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

2.2.4 ObjectID

Beschreibung

Datentyp	Bedeutung
Integer	Liest die interne Objektnummer der MCX-Achse in Bezug auf alle Achsobjekte der im JetSym-Hardware-Manager angelegten MCX-Achsen aus. Jedem Achsobjekt ist eine eindeutige Nummer zugeordnet.

Syntax

```
AchsObjektID := AX1.AxisConfigData.ObjectID;
```

Zustandsverwaltung

Die Eigenschaft dient der Diagnose. Auf sie kann nur lesend zugegriffen werden. Der Zustand der Achse bleibt unverändert.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

2.2.5 DeviceID

Beschreibung

Datentyp	Bedeutung
Int	Liest die interne Gerätenummer der MCX-Achse in Bezug auf das Bussystem. Diese Nummer wird vom JetSym-Hardware-Manager für die interne Kommunikation vergeben. Jedem Busteilnehmer ist eine eindeutige DeviceID zugeordnet. Achsen, die kein Busteilnehmer sind (z. B. virtuelle Achsen), liefern den Wert `-1` zurück.

Syntax

```
BusObjektID := AX1.AxisConfigData.DeviceID;
```

Zustandsverwaltung

Die Eigenschaft dient der Diagnose. Auf sie kann nur lesend zugegriffen werden. Der Zustand der Achse bleibt unverändert.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

2.2.6 IsSimulation

Beschreibung

Datentyp	Bedeutung
Bool	Liest, ob die MCX-Achse simuliert konfiguriert wurde. In Projekten, in denen eine MCX-Achse nicht in jeder Maschine verwendet wird, ermöglich diese Eigenschaft, im Programm zu erkennen, ob die betreffende Achse in der für diese Maschine verwendeten Konfiguration Verwendung findet. `True`: Die MCX-Achse wird simuliert betrieben.

Syntax

```
If Not AX1.State.IsSimulation then
...
End_If;
```

Zustandsverwaltung

Die Eigenschaft dient der Diagnose. Auf sie kann nur lesend zugegriffen werden. Der Zustand der Achse bleibt unverändert.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

3 MCAxis.Buffer

Mit der Funktionalität **MCAxis.Buffer** schalten Sie die Achse zwischen gepuffertem und normalem Betrieb hin und her. Darüber hinaus können Sie mit den Funktionen und Eigenschaften der Funktionalität den Puffer verwalten und den Pufferstatus abfragen.

ⓘ Info

Achsverhalten im gepufferten Betrieb

Die Konfiguration und Parametrierung einer Achse sind im gepufferten Betrieb gesperrt.

Nach dem Einschalten befindet sich die Achse in der Betriebsart *normal*, in der Bewegungsbefehle „verdrängend“ sind. Das bedeutet, dass ein neuer Bewegungsbefehl mit sofortiger Wirkung die bisher aktive Verfahrbewegung ersetzt.

In der Betriebsart *gepuffert* werden Bewegungsbefehle für die Achse in einen MCX-internen Puffer geschrieben und nacheinander ausgeführt. Die Bedingung für den Übergang von einer Verfahrbewegung zur nächsten (Satzwechselbedingung) hängt von der Art des Bewegungsbefehls ab. Im Einzelnen gelten folgende Satzwechselbedingungen:

Bewegungsbefehl	Bedingung
MCAxis.MovePtp()	Erreichen der Zielposition des Sollwerts (Interpolatorstopp).
MCAxis.MoveVelocity()	Erreichen der Zielgeschwindigkeit des Sollwerts.
MCAxis.MoveHome()	Erreichen der Referenzposition des Sollwerts (Interpolatorstopp).
MCAxis.MoveHalt()	Erreichen der Sollgeschwindigkeit 0 (Interpolatorstopp).

Tab. 1: Übersicht über Satzwechselbedigungen

MCX-Achsen können den Puffer auf zwei Weisen abarbeiten:

– *Normal*

  Der Puffer wird einmal durchlaufen. Danach wartet das System auf neue Bewegungsbefehle.

– *Cyclic*

– Die Abarbeitung des Puffers startet nach jedem Durchlauf erneut mit dem ersten Bewegungsbefehl und endet erst durch einen expliziten Abbruch (Delete().

3.1 Syntax für MCAxis.Buffer

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Buffer.<Funktion>(optional <Parameter>);
```

3.2 Funktionen von MCAxis.Buffer

Die Funktionalität bietet folgende Funktionen:

– Create()
– Clear()
– Delete()
– Suspend()
– Resume()
– Wait()

3.2.1 Create()

Die Funktion aktiviert den Puffer für diese Achse. Dadurch wechselt die Betriebsart der Achse von *normal* zu *gepuffert*.

Übergabeparameter

Parameter	Datentyp	Bedeutung
MCBufferTypes	MCBufferTypes (optional)	Definiert den Puffertyp.
	MCBufferTypes.Normal (Nach Reset aktiv)	Definiert einen Puffer, dessen Inhalt einmalig durchlaufen wird.
	MCBufferTypes.Cyclic	Definiert einen Puffer, dessen Inhalt von vorne beginnend solange durchlaufen wird, bis die Abarbeitung explizit abgebrochen wird.

Syntax

```
AX1.Buffer.Create();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Der Zustandsübergang kann nicht in einem MCX-Zyklus abgearbeitet werden. Bevor Sie weitere Befehle für die Achse aufrufen, fragen Sie das Erreichen des neuen Betriebszustands im Programm wie folgt ab:

```
When AX1.Buffer.BufferState == MCBufferStates.Normal Continue;
```

oder

```
When AX1.Buffer.BufferState == MCBufferStates.Cyclic Continue;
```

3.2.2 Clear()

Die Funktion deaktiviert die Abarbeitung des Puffers mit folgenden Auswirkungen:

– Die Betriebsart der Achse wechselt von *gepuffert* zu *normal*.

– Der Puffer wird gelöscht.

– Eine Bewegung, die schon begonnen wurde, wird zu Ende gefahren.

Die Befehle `Power.Disable()` und `Power.Quickstop()` führen ebenfalls zu einer Deaktivierung des Puffers.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.Buffer.Clear();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

3.2.3 Delete()

Die Funktion ist nur für den zyklischen Puffertyp von Bedeutung. Sie teilt dem System mit, wo der zyklische Pufferinhalt endet. D. h. kommt die Abarbeitung an diesen Befehl, beginnt sie den Puffer wieder bei dem Befehl abzuarbeiten, der nach dem `Buffer.Create()` geschrieben wurde.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.Buffer.Delete();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

3.2.4 Suspend()

Die Funktion hält die Pufferabarbeitung am nächsten Satzübergang an mit folgenden Auswirkungen:

– Die laufende Bewegung wird zu Ende geführt.

– Die nächste im Puffer beschriebene Bewegung wird nicht mehr gestartet.

– Die Achse bleibt in der Betriebsart *Gepuffert*.

Durch Aufrufen der Funktion Resume() [▶ 34] wird die Abarbeitung des Puffers wieder aufgenommen.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.Buffer.Suspend();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

3.2.5 Resume()

Die Funktion nimmt die mit Suspend() [▶ 34] angehaltene Abarbeitung des Puffers wieder auf.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.Buffer.Resume();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

3.2.6 Wait()

Die Funktion setzt im Puffer eine Wartezeit, die die Ausführung eines Befehls gegenüber dem unmittelbar davor in den Puffer geschriebenen Befehl verzögern.

Übergabeparameter

Parameter	Datentyp	Bedeutung
WaitingTime	Integer (optional)	Definiert die Wartezeit in [ms].

Syntax

```
AX1.Buffer.Wait(100);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

3.3 Eigenschaften von MCAxis.Buffer

Die Funktionalität bietet folgende Eigenschaften:

– BufferState
– IsCyclic
– IsNormal

3.3.1 BufferState

Beschreibung

Mit der Eigenschaft lesen Sie den aktuellen Pufferstatus der MCX-Achse.

Nach dem Einschalten befindet sich die Achse in der Betriebsart *normal*, d.h. im nichtgepufferten Betrieb. Der Pufferstatus ist `0`.

Datentyp	Bedeutung
MCBufferStates	Pufferstatus der MCX-Achse:
0	Die Achse befindet sich in der Betriebsart *normal*: kein gepufferter Betrieb.
Normal (1)	Die Achse befindet sich in Pufferstatus *normal*: Der Pufferinhalt wird einmalig durchlaufen.
Cyclic (2)	Die Achse befindet sich im Pufferstatus *cyclic*: Der Pufferinhalt wird endlos durchlaufen.

Syntax

```
If Not AX1.State.IsSimulation then
...
End_If;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

3.3.2 IsCyclic

Beschreibung

Die Eigenschaft fragt ab, ob der aktuelle Pufferstatus der MCX-Achse *cyclic* ist.

Datentyp	Bedeutung
Bool	Pufferstatus der MCX-Achse:
True	Die Achse befindet sich in Pufferstatus *cyclic*: Der Pufferinhalt wird endlos durchlaufen.
False	– Die Achse befindet sich nicht im gepufferten Betrieb. – Die Achse befindet sich im Pufferstatus *normal*.

Syntax

```
If AX1.Buffer.IsCyclic then
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

3.3.3 IsNormal

Beschreibung

Die Eigenschaft fragt ab, ob der aktuelle Pufferstatus der MCX-Achse *normal* ist.

Datentyp	Bedeutung
Bool	Pufferstatus der MCX-Achse:
True	Die Achse befindet sich in Pufferstatus *normal*: Der Pufferinhalt wird einmalig durchlaufen.
False	– Die Achse befindet sich nicht im gepufferten Betrieb. – Die Achse befindet sich im Pufferstatus *cyclic*.

Syntax

```
If AX1.Buffer.IsNormal then
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

3.4 Beispieltask für MCAxis.Buffer

Hier finden Sie einen Beispieltask einer Einzelachse, die im Puffertyp *cyclic* betrieben wird.

Es wird ein zyklischer Puffer erstellt, der das Reversieren einer Achse zwischen Position 1000 und 0 mit der Geschwindigkeit gemäß der Variable `Speed` startet. Am Ziel hält die Achse jeweils 200 ms lang an.

```
Task Sample3 Autorun

    // Einschalten des Servo-Verstärkers
    If Not AX1.State.IsEnabled AND AX1.Drive.IsReadyForOperation Then
        AX1.Power.Enable();
    End_IF;
    When AX1.State.IsEnabled Continue;

    // Pufferbetrieb cyclic aktivieren
    AX1.Buffer.Create(MCBufferTypes.Cyclic);
    When AX1.Buffer.IsCyclic Continue;

    // Den Puffer füllen mit Befehlen für Reversierbetrieb
    AX1.MovePtp.Start(AxisPositioningModes.AbsNor-
mal,1000,Speed,1000,5000,0.1);
    AX1.Buffer.Wait(200);
    AX1.MovePtp.Start(,0,,,,);
    AX1.Buffer.Wait();
    AX1.Buffer.Delete(); // Zeigt der MCX das Ende des zyklischen Puffers

    // Nach 10 Sekunden das Reversieren unterbrechen
    delay(t#10s);
    AX1.Buffer.Suspend();

    // Nach weiteren 5 Sekunden wieder fortfahren
    delay(t#5s);
AX1.Buffer.Resume();
```

```
    // Nach weiteren 20 Sekunden den Reversierbetrieb beenden und
    // Achse sperren, wenn die letzte Bewegung beendet ist
    delay(t#20s);
AX1.Buffer.Clear();
    When (AX1.Buffer.BufferState == 0) AND (Not AX1.State.IsRunning) Conti-
nue;

    AX1.Power.Disable();
    When AX1.State.IsDisabled Continue;

End_Task;
```

4 MCAxis.Diagnostics

Mit der Funktionalität **MCAxis.Diagnostics** können Sie die Fehlerbehandlung einer MCX-Achse verwalten.

ⓘ Info

Geltungsbereich der Funktionalität

Mit Ausnahme der Funktion `ClearErrors()` beziehen sich die hier aufgeführten Funktionen und Eigenschaften auf MCX-Fehler. Fehler und Warnungen, die von den Servoverstärkern (z. B. JetMove 3xxx) erkannt werden, müssen mit der Funktionalität **MCAxis.Drive.Diagnostics** behandelt werden.

Im Motion-Control-eXtended-System werden Fehler (*errors*) und Meldungen (*messages*) unterschieden. Fehler rufen eine Systemreaktion hervor, Meldungen hingegen nicht. Fehler und Meldungen werden in getrennten Puffern protokolliert, von wo aus sie ausgelesen werden können. Der jeweilige Puffer sammelt die Ereignisse, die seit dem letzten Aufruf der Funktion `ClearErrors`() aufgetreten sind (siehe ClearErrors() [▶ 39]).

Die Bedeutung der Fehler- und Meldungsnummern ist in der JetSym -Online-Hilfe unter „MotionControl-Fehler“ aufgeführt.

4.1 Syntax für MCAxis.Diagnostics

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Diagnostics.<Funktion oder Zugriff auf Wert>(optional <Parameter>);
```

4.2 Funktionen von MCAxis.Diagnostics

Die Funktionalität bietet folgende Funktionen:

– ClearErrors()
– GetErrorCode(BufferIndex)
– GetMessageCode(BufferIndex)

4.2.1 ClearErrors()

Bei Fehlern führt die Funktion das Quittieren eines Fehlerzustandes aus. Die Einträge im Fehlerpuffer werden gelöscht.

Meldungen haben keine Fehlerreaktion zur Folge, und müssen deshalb nicht quittiert werden. Bei Meldungen bewirkt die Funktion, dass die Einträge im Meldungspuffer gelöscht werden.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.Diagnostics.ClearErrors();
```

ⓘ Info

Geltungsbereich des Befehls

Mit diesem Befehl werden alle Fehler gelöscht, die der Achse zugeordnet sind. Das können Fehler sein, die vom Servoverstärker (Drive) gemeldet wurden, ebenso wie Fehler des Motion-Control-Teils in der Steuerung (MCX).

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand	
Gesperrt	–	Gesperrt	
Freigegeben	–	Freigegeben	
Läuft	–	Läuft	
Unterbrochen	–	Unterbrochen	
Fehler: freigegeben	√	Freigegeben	In diesen Zustand wechselt das System nur, wenn die Fehlerursache behoben ist. Andernfalls bleibt u. U. auch der Fehlerzustand erhalten oder wird sofort wieder erreicht.
Fehler: gesperrt	√	Gesperrt	

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

4.2.2 GetErrorCode()

Die Funktion liefert die Nummer des Fehlers an die mit `BufferIndex` angegebene Fehlerpufferstelle zurück. Wird kein `BufferIndex` übergeben, wird die Nummer des zuletzt aufgezeichneten Fehlers zurückgegeben (`BufferIndex` = `0`).

Übergabeparameter

Parameter	Datentyp	Bedeutung
BufferIndex	Integer (optional)	Der `BufferIndex` übergibt die Stelle im Fehlerpuffer, deren Fehlernummer ausgelesen werden soll. Der Maximalwert des Parameters `BufferIndex` kann über die Eigenschaft `ErrorCount` ausgelesen werden.

Syntax

```
AX1.Diagnostics.GetErrorCode();
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

4.2.3 GetMessageCode()

Die Funktion liefert die Nummer der Meldung an die mit `BufferIndex` angegebene Meldungspufferstelle zurück. Wird kein `BufferIndex` übergeben, wird die Nummer der zuletzt aufgezeichneten Meldung zurückgegeben (`BufferIndex` = 0).

Übergabeparameter

Parameter	Datentyp	Bedeutung
BufferIndex	Integer (optional)	Der `BufferIndex` übergibt die Stelle im Meldungspuffer, deren Meldungsnummer ausgelesen werden soll. Der Maximalwert des Parameters `BufferIndex` kann über die Eigenschaft `Message-Count` ausgelesen werden.

Syntax

```
AX1.Diagnostics.GetMessageCode();
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

4.3 Eigenschaften von MCAxis.Diagnostics

Die Funktionalität bietet folgende Eigenschaften:

- ErrorCount
- IsErrorPending
- MessageCount
- IsMessagePending

4.3.1 ErrorCount

Beschreibung

Eigenschaft	Datentyp	Bedeutung
ErrorCount	Integer	Liest die Anzahl der erfassten Fehler seit dem letzten Ausführen der Funktion `ClearErrors()` oder seit dem Neustart der Steuerung aus.

Syntax

```
FehlerAnzahl := AX1.Diagnostics.ErrorCount;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

4.3.2 IsErrorPending

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob ein Fehler im MCX-System vorliegt. Dies ist gleichbedeutend mit einem Vergleich von `ErrorCount` mit 0.

Syntax

```
If AX1.Diagnostics.IsErrorPending then
...
End_if;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

4.3.3 IsMessagePending

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob eine Meldung im MCX-System vorliegt. Dies ist gleichbedeutend mit einem Vergleich von `MessageCount` mit `0`.

Syntax

```
If AX1.Diagnostics.IsMessagePending then
...
End_if;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

4.3.4 MessageCount

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Anzahl der erfassten Meldungen seit dem letzten Ausführen der Funktion `ClearErrors()` oder seit dem Neustart der Steuerung aus.

Syntax

```
MeldungsAnzahl := AX1.Diagnostics.MessageCount;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

4.4 Beispieltask für MCAxis.Diagnostics

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Var
    AchsError: Array[10] of INT; //Variablen zum Anzeigen des Fehlerpuffers
    i: INT; //Laufvariable
    AcknolegeError: Bool; // Hilfs-Flag für eine Quittier-Bedingung

End_Var;

Task Sample_Error Autorun

Loop

    //Fehlervariablen zum Anzeigen löschen
    For i := 0 To 9 Do
        AchsError[i] := 0;
    End_For;
    AcknowledgeError := False;
```

```
    When AX1.Diagnostics.IsErrorPending Continue;

        Taskbreak Automatic;
        // hier weitere Fehlerbehandlung einfügen

        //Fehler in Variablen z. B. zum Anzeigen ablegen
        For i := 0 To AX1.Diagnostics.ErrorCount-1 Do
            AchsError[i] := AX1.Diagnostics.GetErrorCode(i);
        End_For;

    // Fehler sind behoben
    When AcknowledgeError Continue;

        AX1.Diagnostics.ClearErrors();
        Taskrestart Automatic;
End_Loop;

End_Task;
```

5 MCAxis.Drive.Amplifier

Mit der Funktionalität **MCAxis.Drive.Amplifier** können Sie die Eigenschaften des Servoverstärkers der MCX-Achse lesen und anpassen.

Es handelt sich dabei im Wesentlichen um die Parameter, die im JetSym-Programmiertool im Motion-Setup auf der Inbetriebnahme-Seite „Verstärker“ für Bucher-Automation-Achsen zur Verfügung stehen. Durch die Initialisierung der Steuerung werden die Parameter mit den bei der Inbetriebnahme abgespeicherten Werten belegt.

Die Motion-API-Funktionalitäten ermöglichen es, diese Parameter auch über das STX-Programm zu ändern oder auszulesen.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

5.1 Syntax für MCAxis.Drive.Amplifier

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.Amplifier.<Zugriff auf Wert>;
```

5.2 Eigenschaften von MCAxis.Drive.Amplifier

Die Funktionalität bietet folgende Eigenschaften:

- ActualDCLinkVoltage
- ActualHeatSinkTemperature
- ActualInternalTemperature
- DCLinkVoltageMax
- DCLinkVoltageMin
- PwmFrequency
- PwmFrequencyReduction
- RatedCurrent
- RatedDCLinkVoltage
- WireTestMode
- I2tValue

5.2.1 ActualDCLinkVoltage

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest die aktuelle Zwischenkreisspannung des Servoverstärkers.

Syntax

```
Zwischenkreisspannung := AX1.Drive.Amplifier.ActualDCLinkVoltage;
```

Zustandsverwaltung

Auf die Eigenschaften kann nur lesend zugegriffen werden, da die Zwischenkreisspannung nur durch die Hardware und die mechanische Last bestimmt wird.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

5.2.2 ActualHeatSinkTemperature

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest die aktuelle Kühlkörpertemperatur des Servoverstärkers.

Syntax

```
Kuehlkoerpertemperatur := AX1.Drive.Amplifier.ActualHeatSinkTemperature;
```

Zustandsverwaltung

Auf die Eigenschaften kann nur lesend zugegriffen werden, da die Kühlkörpertemperatur nur durch die Hardware und den Laststrom bestimmt wird.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

5.2.3 ActualInternalTemperature

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest die aktuelle Innenraumtemperatur des Servoverstärkers.

Syntax

```
Innenraumtemperatur := AX1.Drive.Amplifier.ActualInternalTemperature;
```

Zustandsverwaltung

Auf die Eigenschaften kann nur lesend zugegriffen werden, da die Innenraumtemperatur nur durch die Hardware und die Umgebungstemperatur bestimmt wird.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

5.2.4 DCLinkVoltageMax

Beschreibung

Die Zwischenkreisspannung wird durch die Eingangsspannung und die Rückspeiseenergie bei Bremsvorgängen bestimmt. Wenn sie den hier definierten Wert überschreitet, wird ein Überspannungsfehler ausgelöst, der das weitere Ansteigen verhindert.

Datentyp	Bedeutung
Double (optional)	Liest die Fehlerauslöseschwelle für die Überspannung im Zwischenkreis des Servoverstärkers in [V_{dc}].

Syntax

```
MaxZwischenkreisspannung := AX1.Drive.Amplifier.DCLinkVoltageMax;
```

Zustandsverwaltung

Auf die Eigenschaft kann nur lesend zugegriffen werden, da diese Fehlerauslöseschwelle zum Geräteschutz von der Hardware bestimmt wird.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

5.2.5 DCLinkVoltageMin

Beschreibung

Wird die Zwischenkreisspannung aufgrund zu geringer Netzspannung oder zu hoher Last zu niedrig, so wird ein Fehler ausgelöst. Die Fehlerschwelle ist mit dieser Eigenschaft auslesbar. Für manche Gerätetypen ist auch die Anpassung der Fehlerschwelle möglich, was aber nur in sehr wenigen Ausnahmefällen zu empfehlen ist.

Datentyp	Bedeutung
Double (optional)	Liest oder schreibt die Fehlerauslöseschwelle für die Unterspannung im Zwischenkreis des Servoverstärkers in [V_{dc}].

Syntax

```
MinZwischenkreisspannung := AX1.Drive.Amplifier.DCLinkVoltageMax;
AX1.Drive.Amplifier.DCLinkVoltageMax := 12.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

5.2.6 PwmFrequency

Beschreibung

In der Regel ist der Default-Wert die beste Wahl, da der Servoverstärker für diese Frequenz optimiert ist. Die Verlustleistung des Servoverstärkers ändert sich mit der Einstellung dieser Eigenschaft.

In wenigen Fällen kann es sinnvoll sein, diesen Wert zu verändern:

– Bei sehr hoher Last und kleinen Umdrehungszahlen kann ein kleinerer Wert sinnvoll sein.

– Bei sehr hohen Drehzahlen und/oder extrem dynamischen Positionierungen kann ein höherer Wert sinnvoll sein.

Diese Eigenschaft kann nicht bei allen Geräten verändert werden.

Datentyp	Bedeutung
DrivePwmFrequencies (optional)	Liest oder schreibt die Schaltgeschwindigkeit des Wechselrichters im Servoverstärker (PWM-Frequenz) in [kHz]. Frequenzoptionen wie im Datentyp angegeben.
DrivePwmFrequencies.kHz4	
DrivePwmFrequencies.kHz8	
DrivePwmFrequencies.kHz16	

Syntax

```
PWMFrequenz := AX1.Drive.Amplifier.PwmFrequency;
AX1.Drive.Amplifier.PwmFrequency := DrivePwmFrequencies.kHz8;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

5.2.7 PwmFrequencyReduction

Beschreibung

In einigen Geräten kann diese Eigenschaft aktiviert werden, um bei sehr niedrigen Motordrehzahlen die PWM-Frequenz automatisch abzusenken. In Ausnahmefällen kann dies sinnvoll sein, um die Verlustleistung des Servoverstärkers bei hoher Last und sehr kleinen Drehzahlen zu reduzieren.

Diese Eigenschaft kann nicht bei allen Geräten verändert werden.

Datentyp	Bedeutung
DrivePwmFrequenciesReductionModes (optional)	Liest oder schreibt die Absenkung der PWM-Frequenz des Wechselrichters im Servoverstärker. Frequenzoptionen wie angegeben.
DrivePwmFrequenciesReductionModes.Disable	Deaktiviert die Eigenschaft.
DrivePwmFrequenciesReductionModes.Enable	Aktiviert die Eigenschaft.

Syntax

```
PWMFrequenzAbsenkung := AX1.Drive.Amplifier.PwmFrequencyReduction;
AX1.Drive.Amplifier.PwmFrequencyReduction := DrivePwmFrequencyReductionMo-
des.Disabled;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

5.2.8 RatedCurrent

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den Nennstrom des Servoverstärkers, der dieser Achse zugeordnet ist in [A_{ac}].

Syntax

```
GeraeteNennstrom := AX1.Drive.Amplifier.RatedCurrent;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

5.2.9 RatedDCLinkVoltage

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest die Zwischenkreis-Nennspannung des Servoverstärkers, der dieser Achse zugeordnet ist in [V_{dc}].

Syntax

```
GeraeteNennZwischenkreisspannung := AX1.Drive.Amplifier.RatedDCLinkVoltage;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

5.2.10 WireTestMode

Beschreibung

Die Ausführung der Eigenschaft kann kurze Zeit in Anspruch nehmen. Sollte sie deshalb in der Applikation nicht verwendbar ist, kann die Funktion konfiguriert werden.

Nicht alle Geräte unterstützen den Motorkabeltest.

Datentyp	Bedeutung
DriveWireTestModes (optional)	Liest oder schreibt, ob der Servoverstärker beim ersten Freigeben der Motorfunktion einen Motorkabeltest durchführt.
DriveWireTestModes.Disable	Deaktiviert die Eigenschaft.
DriveWireTestModes.Enable	Aktiviert die Eigenschaft.

Syntax

```
MotorKabelTestStatus := AX1.Drive.Amplifier.WireTestMode;
AX1.Drive.Amplifier.WireTestMode := DriveWireTestModes.Disabled;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

5.2.11 I2tValue

Beschreibung

Nicht alle Geräte unterstützen die Berechnung des I²t-Wertes für den Servoverstärker.

Datentyp	Bedeutung
Double (optional)	Liest den Integrationswert der I²t-Überwachung des Servoverstärkers, der dieser Achse zugeordnet ist in [%]

Syntax

```
GeraeteI2tWert := AX1.Drive.Amplifier.I2tValue;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

6 MCAxis.Drive.Brake

Mit der Funktionalität **MCAxis.Drive.Brake** können Sie die Eigenschaften der Motorhaltebremse der MCX-Achse lesen und anpassen.

Es handelt sich dabei im Wesentlichen um die Parameter, die im JetSym -Programmiertool im Motion-Setup auf der Inbetriebnahme-Seite „Bremse“ für Bucher-Automation-Achsen zur Verfügung stehen. Durch die Initialisierung der Steuerung werden die Parameter mit den bei der Inbetriebnahme abgespeicherten Werten belegt.

Die Motion-API-Funktionalitäten ermöglichen es, diese Parameter auch über das STX-Programm zu ändern oder auszulesen.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

6.1 Syntax für MCAxis.Drive.Brake

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.Brake.<Funktion oder Zugriff auf Wert>;
```

6.2 Funktionen von MCAxis.Drive.Brake

Die Funktionalität bietet folgende Funktionen:

– Apply()
– Release()

6.2.1 Apply()

Beschreibung

Die Funktion schließt die Motorhaltebremse. Sie wirkt nur, wenn die Einstellung der Eigenschaft `EncDriv.Drive.Brake.Mode` (Motorbremse-Betriebsart) *manual* ist. In der Betriebsart *auto* übernimmt der Servoverstärker die Steuerung der Bremse entsprechend dem Antriebszustand.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.Drive.Brake.Apply();
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

6.2.2 Release()

Beschreibung

Die Funktion öffnet die Motorhaltebremse. Sie wirkt nur, wenn die Einstellung der Eigenschaft `EncDriv.Drive.Brake.Mode` (Motorbremse-Betriebsart) *manual* ist. In der Betriebsart *auto* übernimmt der Servoverstärker die Steuerung der Bremse entsprechend dem Antriebszustand.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.Drive.Brake.Release();
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

6.3 Eigenschaften von MCAxis.Drive.Brake

Die Funktionalität bietet folgende Eigenschaften:

- ApplyDelay
- ReleaseDelay
- Mode
- State

6.3.1 ApplyDelay

Beschreibung

Die Verzögerung bewirkt, dass beim Sperren (*disable*) der Achse zunächst der Bremsenausgang geschaltet wird, und erst nach der hier eingestellten Verzögerungszeit der Antrieb stromlos wird. Dadurch wird z. B. vermieden, dass eine vertikale Achse beim Sperren nicht absackt, obwohl die Bremse aufgrund der mechanischen Bewegung der Bremsbeläge nicht sofort wirksam werden kann.

Datentyp	Bedeutung
Double	Liest oder schreibt die Verzögerungszeit beim Schließen der Motorhaltebremse in [ms]. Die Verzögerung wirkt nur, wenn die Einstellung der Eigenschaft `EncDriv.Drive.Brake.Mode` (Motorbremse-Betriebsart) *auto* ist.

Syntax

```
BremseVerzoegerungSchließen := AX1.Drive.Brake.ApplyDelay;
AX1.Drive.Brake.ApplyDelay := 150.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

6.3.2 ReleaseDelay

Beschreibung

Die Verzögerung bewirkt, dass beim Freigeben (*enable*) der Achse zunächst der Bremsenausgang geschaltet wird, und erst nach Ablauf der hier eingestellten Verzögerungszeit die Meldung `isEnabled` aktiv wird. Der Ausgang für die Motorhaltebremse schaltet sofort mit der Bestromung des Motors. Da die Bremse u. U. nicht sofort geöffnet ist, stellt die Verzögerungszeit sicher, dass der Motor nicht bewegt wird, solange die Bremse womöglich nicht vollständig geöffnet wurde.

Datentyp	Bedeutung
Double	Liest oder schreibt die Verzögerungszeit beim Öffnen der Motorhaltebremse in [ms]. Die Verzögerung wirkt nur, wenn die Einstellung der Eigenschaft `EncDriv.Drive.Brake.Mode` (Motorbremse-Betriebsart) *auto* ist.

Syntax

```
BremseVerzoegerungOeffnen := AX1.Drive.Brake.ReleaseDelay;
AX1.Drive.Brake.ReleaseDelay := 30.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

6.3.3 Mode

Beschreibung

Mit der Eigenschaft können Sie die Betriebsart für die Bedienung der Motorhaltebremse lesen oder anpassen.

Datentyp	Bedeutung
DriveBrakeModes	Betriebsart der Motorhaltebremse:
DriveBrakeModes.Manual	Die Motorhaltebremse wird nicht vom Servoverstärker verwaltet. Der Anwender steuert die Bremse selbst über das Programm oder den Motion-Setup.
DriveBrakeModes.Auto	Diese Betriebsart ist in den folgenden Fällen zu wählen: – Es ist keine Bremse angeschlossen. – Die Bremse soll zu Servicezwecken bei stromlosem Motor betätigt werden.

Syntax

```
BremseBetriebsart := AX1.Drive.Brake.Mode;
AX1.Drive.Brake.Mode := DriveBrakeModes.Manual;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

6.3.4 State

Beschreibung

Mit der Eigenschaft können Sie die Stellung der Motorhaltebremse lesen.

Datentyp	Bedeutung
DriveBrakeStates	Stellung der Motorhaltebremse:
DriveBrakeStates.Applied	Die Motorhaltebremse ist geschlossen.
DriveBrakeStates.Released	Die Motorhaltebremse ist geöffnet.

Syntax

```
BremseStellung := AX1.Drive.Brake.State;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

7 MCAxis.Drive.Control.Current

Mit der Funktionalität **MCAxis.Drive.Control.Current** können Sie die Eigenschaften des Strom-/Drehmomentreglers des Servoverstärkers der MCX-Achse lesen und anpassen.

Es handelt sich dabei im Wesentlichen um die Parameter, die im JetSym-Programmiertool im Motion-Setup auf den Inbetriebnahme-Seiten Regelung > Strom/Drehmoment und Begrenzungswerte für Bucher-Automation-Achsen zur Verfügung stehen. Durch die Initialisierung der Steuerung werden die Parameter mit den bei der Inbetriebnahme abgespeicherten Werten belegt.

Mithilfe der Motion-API-Befehle ist es möglich, diese Parameter auch über das STX-Programm auszulesen und teilweise anzupassen.

In einem Servoverstärker wird die Motorregelung so ausgeführt, dass der Motorstrom proportional zum Motordrehmoment ist. Aus diesem Grund kann ein solcher Regler sowohl als Strom- als auch als Drehmomentregler bezeichnet werden, obwohl tatsächlich nur der Strom geregelt wird. Die Drehmomentwerte werden durch Multiplikation mit der Drehmomentkonstante aus den Stromwerten berechnet.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

7.1 Syntax für MCAxis.Drive.Control.Current

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.Control.Current.<Zugriff auf Wert>;
```

7.2 Eigenschaften von MCAxis.Drive.Control.Current

Die Funktionalität bietet folgende Eigenschaften:

- Kp
- Tn
- ActualMotorCurrent
- ActualMotorVoltage
- MaxMotorCurrent

7.2.1 Kp

Beschreibung

Mit der Eigenschaft können Sie den Verstärkungswert für den Stromregler des Servoverstärkers lesen oder anpassen.

Der Stromregler ist als PI-Regler ausgeführt. Die Einstellung des Stromreglers hängt weitgehend von Motordaten ab und lässt sich deshalb gut auf rechnerischem Weg ermitteln. Dies ist die bevorzugte Vorgehensweise bei der Inbetriebnahme. Die automatische Einstellung des Stromreglers kann im Motion-Setup von JetSym über die Schaltfläche Regleroptimierung ausgeführt werden.

Datentyp	Bedeutung
Double	Definiert den Verstärkungsfaktor des Stromreglers in [V/A].

Syntax

```
Stromreglerverstaerkung := AX1.Drive.Control.Current.Kp;
AX4.Drive.Control.Current.Kp := 1.45;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden. In den meisten Fällen ist dies allerdings nur während der Inbetriebnahme der Achse zum Abgleich der Regelung sinnvoll. **HINWEIS! Eine Änderung während des Maschinenablaufs kann zu unkontrollierten Bewegungen führen, wenn der geschriebene Wert nicht zu der Belastung des Antriebs passt.**

7.2.2 Tn

Beschreibung

Mit der Eigenschaft können Sie die Nachstellzeit für den Stromregler des Servoverstärkers lesen oder anpassen.

Der Stromregler ist als PI-Regler ausgeführt. Die Einstellung des Stromreglers hängt weitgehend von Motordaten ab und lässt sich deshalb gut auf rechnerischem Weg ermitteln. Dies ist die bevorzugte Vorgehensweise bei der Inbetriebnahme ist. Die automatische Einstellung des Stromreglers kann im Motion-Setup von JetSym über die Schaltfläche Regleroptimierung ausgeführt werden.

Datentyp	Bedeutung
Double	Definiert die Nachstellstellzeit des Stromreglers in [ms].

Syntax

```
Stromreglernachstellzeit := AX1.Drive.Control.Current.Tn;
AX4.Drive.Control.Current.Tn := 2.5;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden. In den meisten Fällen ist dies allerdings nur während der Inbetriebnahme der Achse zum Abgleich der Regelung sinnvoll. **HINWEIS! Eine Änderung während des Maschinenablaufs kann zu unkontrollierten Bewegungen führen, wenn der geschriebene Wert nicht zu der Belastung des Antriebs passt.**

7.2.3 ActualMotorCurrent

Beschreibung

Datentyp	Bedeutung
Double	Liest den aktuellen Motorphasenstrom, der vom Servoverstärker ausgegeben wird in [A_{eff}].

Syntax

```
IstStrom := AX1.Drive.Control.Current.ActualMotorCurrent;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

7.2.4 ActualMotorVoltage

Beschreibung

Datentyp	Bedeutung
Double	Liest die aktuelle Motorspannung zwischen zwei Motorphasen, die vom Servoverstärker ausgegeben wird in [V_{eff}].

Syntax

```
IstSpannung := AX1.Drive.Control.Current.ActualMotorVoltage;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

7.2.5 MaxMotorCurrent

Beschreibung

Datentyp	Bedeutung
Double	Liest den maximal zur Verfügung stehenden Motorphasenstrom, der aufgrund der Einstellungen des Motor-Nenndrehmoments und des Überlastfaktor-Drehmoments sowie der Drehmomentkonstante vom Servoverstärker zur Verfügung gestellt wird, in [A_{eff}].

Syntax

```
MaxStrom := AX1.Drive.Control.Current.MaxMotorCurrent;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

HINWEIS



Schreibender Zugriff: Gefahr von unkontrollierten Bewegungen

Eine Änderung während des Maschinenablaufs kann zu unkontrollierten Bewegungen führen, wenn der geschriebene Wert nicht zu der Belastung des Antriebs passt.

- ► Vermeiden Sie Änderungen während des Maschinenbetriebs.
- ► Änderungen sind in der Regel nur während der Inbetriebnahme der Achse zum Abgleich der Regelung sinnvoll.

8 MCAxis.Drive.Control.Position

Mit der Funktionalität **MCAxis.Drive.Control.Position** können Sie die Eigenschaften des Positionsreglers des Servoverstärkers der MCX-Achse lesen und anpassen.

Es handelt sich dabei im Wesentlichen um die Parameter, die im JetSym-Programmiertool im Motion-Setup auf den Inbetriebnahme-Seiten Regelung > Lage und Vorsteuerung für Bucher-Automation-Achsen zur Verfügung stehen. Durch die Initialisierung der Steuerung werden die Parameter mit den bei der Inbetriebnahme abgespeicherten Werten belegt.

Mithilfe der Motion-API-Befehle ist es möglich, diese Parameter auch über das STX-Programm auszulesen und teilweise anzupassen.

Der Positionsregler in einem Servoverstärker ermöglicht eine exakte Regelung der Position des Motors entsprechend der Sollwertgenerierung in der Steuerung. Er ist als P-Regler mit Drehzahlvorsteuerung ausgeführt.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

8.1 Syntax für MCAxis.Drive.Control.Position

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.Control.Position.<Funktion oder Zugriff auf Wert>;
```

8.2 Funktionen von MCAxis.Drive.Control.Position

Die Funktionalität bietet folgende Funktionen:

– ResetTrailingIndicators()

8.2.1 ResetTrailingIndicators()

Beschreibung

Die Schleppzeigererfassung zeichnet den höchsten in positive und negative Richtung aufgetretenen Positionsschleppfehler auf. Die Funktion `ResetTrailingIndicators()` setzt die Schleppzeigererfassung auf Null zurück und startet eine neue Messung.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.Drive.Control.Position.ResetTrailingIndicators();
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

8.3 Eigenschaften von MCAxis.Drive.Control.Position

Die Funktionalität bietet folgende Eigenschaften:

– Kv
– FeedbackSource
– SpeedFeedForwardScale

8.3.1 Kv

Beschreibung

Der Positionsregler ist als P-Regler mit Drehzahlvorsteuerung ausgeführt. Je höher der Wert für die Verstärkung ist, desto dynamischer wird der Antrieb. Bei zu hohen Werten beginnt der Antrieb zu schwingen.

Datentyp	Bedeutung
Double	Liest oder verändert den Verstärkungsfaktor des Positionsreglers, in [1/s].

Syntax

```
Positionsreglerverstaerkung := AX1.Drive.Control.Position.Kv;
AX4.Drive.Control.Position.Kv := 80.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

HINWEIS



Schreibender Zugriff: Gefahr von unkontrollierten Bewegungen

Eine Änderung während des Maschinenablaufs kann zu unkontrollierten Bewegungen führen, wenn der geschriebene Wert nicht zu der Belastung des Antriebs passt.

- ► Vermeiden Sie Änderungen während des Maschinenbetriebs.
- ► Änderungen sind in der Regel nur während der Inbetriebnahme der Achse zum Abgleich der Regelung sinnvoll.

8.3.2 FeedbackSource

Beschreibung

In den meisten Anwendungen wird der erste Positionsgeber, der auch für die Kommutierung und die Drehzahlregelung verwendet wird, eingestellt.

Der zweite Positionsgeber kann in besonderen Fällen für die Positionsregelung ausgewählt werden, um einen Schlupf zwischen Material und Motor auszugleichen. Der zweite Positionsgeber ist dann zusätzlich zum Motor-Positionsgeber direkt an der Last montiert. Wenn dieser zusätzliche Positionsgeber für die Positionsregelung verwendet wird, gleich die Regelung selbständig eine kleine Abweichung zwischen der Position der Last und der Motorwelle (Schlupf) aus.

Datentyp	Bedeutung
DriveFeedbackSources	Liest, welcher Positionsgeber für die Positionsregelung gewählt ist oder ändert die Auswahl:
DriveFeedbackSources.Encoder1	Der Positionsgeber an der ersten Encoder-Schnittstelle wird verwendet.
DriveFeedbackSources.Encoder2	Der Positionsgeber an der zweiten Encoder-Schnittstelle wird verwendet.

Syntax

```
Positionsgeber1 := AX1.Drive.Control.Position.FeedbackSource;
AX1.Drive.Control.Position.FeedbackSource := DriveFeedbackSources.Encoder1;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

HINWEIS



Schreibender Zugriff: Gefahr von unkontrollierten Bewegungen beim Wechsel des Positionsgebers im Betrieb

Diese Einstellung sollte während der Erstinbetriebnahme der Achse über den Motion-Setup festgelegt werden. Muss während des Maschinenbetriebs zwischen den beiden Positionsgebern gewechselt werden, kann es zu Positionssprüngen kommen, die wiederum zu unkontrollierten Bewegungen des Antriebs führen können. Außerdem kann sich beim Wechsel des Positionsgebers die Referenzposition verändern.

► Der Zugriff auf diese Eigenschaft darf im nur im Zustand **gesperrt** erfolgen.

8.3.3 SpeedFeedForwardScale

Beschreibung

Mit der Eigenschaft kann die Wirkung der Drehzahlvorsteuerung bei der Regelung der MCX-Achse angepasst werden. Die Drehzahlvorsteuerung kompensiert die bleibende Abweichung des Positionsreglers (P-Regler).

Die Drehzahlvorsteuerung ermittelt eine Solldrehzahl für den Drehzahlregler durch Multiplikation der berechneten Sollgeschwindigkeit mit der Skalierung (SpeedFeedForwardScale). Dieser Wert wird zur Ausgabe des Positionsreglers addiert. Ein anderer Wert als 100 % ist nur bei besonders schwierigen Regelverhältnissen zu empfehlen.

Datentyp	Bedeutung
Double	Liest oder ändert die Skalierung der Wirkung der Drehzahlvorsteuerung in [%].

Syntax

```
DrehmomentVorsteuerSkalierung := AX1.Drive.Control.Position.SpeedFeedFor-
wardScale;
AX4.Drive.Control.Position.SpeedFeedForwardScale:= 95.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

HINWEIS



Schreibender Zugriff: Gefahr von unkontrollierten Bewegungen

Eine Änderung während des Maschinenablaufs kann zu unkontrollierten Bewegungen führen, wenn der geschriebene Wert nicht zu der Belastung des Antriebs passt.

- ► Vermeiden Sie Änderungen während des Maschinenbetriebs.
- ► Änderungen sind in der Regel nur während der Inbetriebnahme der Achse zum Abgleich der Regelung sinnvoll.

9 MCAxis.Drive.Control.Speed

Mit der Funktionalität **MCAxis.Drive.Control.Speed** können Sie die Eigenschaften des Drehzahlreglers des Servoverstärkers der MCX-Achse lesen und anpassen.

Es handelt sich dabei im Wesentlichen um die Parameter, die im JetSym-Programmiertool im Motion-Setup auf den Inbetriebnahme-Seiten Regelung > Drehzahl, Begrenzungswerte und Vorsteuerung für Bucher-Automation-Achsen zur Verfügung stehen. Durch die Initialisierung der Steuerung werden die Parameter mit den bei der Inbetriebnahme abgespeicherten Werten belegt.

Mithilfe der Motion-API-Befehle ist es möglich, diese Parameter auch über das STX-Programm auszulesen und teilweise anzupassen.

Der Drehzahlregler in einem Servoverstärker ermöglicht eine dynamische Anpassung der Geschwindigkeit des Motors an die Vorgaben des Positionsreglers. Er ist als PI-Regler mit Drehmomentvorsteuerung ausgeführt.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

9.1 Syntax für MCAxis.Drive.Control.Speed

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.Control.Speed.<Zugriff auf Wert>;
```

9.2 Eigenschaften von MCAxis.Drive.Control.Speed

Die Funktionalität bietet folgende Eigenschaften:

- Kp
- Tn
- Tf
- ActualValue
- Setpoint
- Isum
- IsumPreset
- OverloadFactor
- MaxMotorSpeed
- SetpointLimit
- DifferenceWindow
- TotalInertia
- TorqueFeedForwardScale

9.2.1 Kp

Beschreibung

Mit der Eigenschaft können Sie den Verstärkungswert für den Drehzahlregler des Servoverstärkers lesen oder anpassen.

Der Drehzahlregler ist als PI-Regler ausgeführt. Je höher der Wert für die Verstärkung ist, umso dynamischer wird der Antrieb. Bei zu hohen Werten beginnt der Antrieb zu schwingen.

Datentyp	Bedeutung
Double	Definiert den Verstärkungsfaktor des Drehzahlreglers in [Nm/1000 (U/min)].

Syntax

```
Drehzahlreglerverstaerkung := AX1.Drive.Control.Speed.Kp;
AX4.Drive.Control.Speed.Kp := 5.5;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

HINWEIS



Schreibender Zugriff: Gefahr von unkontrollierten Bewegungen

Eine Änderung während des Maschinenablaufs kann zu unkontrollierten Bewegungen führen, wenn der geschriebene Wert nicht zu der Belastung des Antriebs passt.

- ► Vermeiden Sie Änderungen während des Maschinenbetriebs.
- ► Änderungen sind in der Regel nur während der Inbetriebnahme der Achse zum Abgleich der Regelung sinnvoll.

9.2.2 Tn

Beschreibung

Mit der Eigenschaft können Sie die Nachstellzeit für den Drehzahlregler des Servoverstärkers lesen oder anpassen.

Der Drehzahlregler ist als PI-Regler ausgeführt. Je niedriger der Wert für die Nachstellzeit ist, umso dynamischer wird der Antrieb. Bei zu kleinen Werten beginnt der Antrieb zu schwingen. Bei hohen mechanischen Schwungmassen an der Motorwelle muss die Nachstellzeit tendenziell höher gewählt werden.

Datentyp	Bedeutung
Double	Definiert die Nachstellstellzeit des Drehzahlreglers in [ms].

Syntax

```
Drehzahlreglernachstellzeit := AX1.Drive.Control.Speed.Tn;
AX4.Drive.Control.Speed.Tn := 10.5;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

HINWEIS



Schreibender Zugriff: Gefahr von unkontrollierten Bewegungen

Eine Änderung während des Maschinenablaufs kann zu unkontrollierten Bewegungen führen, wenn der geschriebene Wert nicht zu der Belastung des Antriebs passt.

- ► Vermeiden Sie Änderungen während des Maschinenbetriebs.
- ► Änderungen sind in der Regel nur während der Inbetriebnahme der Achse zum Abgleich der Regelung sinnvoll.

9.2.3 Tf

Beschreibung

Mit der Eigenschaft können Sie die Filterzeitkonstante für die Drehzahlerfassung des Servoverstärkers lesen oder anpassen.

Da die Güte der Drehzahlerfassung durch die Quantisierung (Differenzierung der Position) und durch Störungen auf der Geberleitung beeinflusst wird, wird der Istwert der Drehzahl über einen Filter geführt.

Es gelten folgende Abhängigkeiten:

– Je höher die Filterzeitkonstante ist, umso ruhiger wird der Drehzahl-Istwert.
– Je höher die Auflösung des Positionsgebers, umso niedriger kann die Filterzeitkonstante gewählt werden.
– Da der Drehzahlfilter eine Totzeit in der Drehzahlregelung darstellt, kann die Verstärkung umso höher eingestellt werden, je niedriger die Filterzeitkonstante ist.

Datentyp	Bedeutung
Double	Definiert die Filterzeitkonstante für die Drehzahlerfassung in [ms].

Syntax

```
Filterzeitkontante := AX1.Drive.Control.Speed.Tf;
AX4.Drive.Control.Speed.Tf := 1.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

HINWEIS



Schreibender Zugriff: Gefahr von unkontrollierten Bewegungen

Eine Änderung während des Maschinenablaufs kann zu unkontrollierten Bewegungen führen, wenn der geschriebene Wert nicht zu der Belastung des Antriebs passt.

► Vermeiden Sie Änderungen während des Maschinenbetriebs.
► Änderungen sind in der Regel nur während der Inbetriebnahme der Achse zum Abgleich der Regelung sinnvoll.

9.2.4 ActualValue

Beschreibung

Datentyp	Bedeutung
Double	Liest die aktuelle Motordrehzahl, die über den ersten Positionsgeber des Servoverstärkers ermittelt wird in [U/min].

Syntax

```
IstDrehzahl := AX1.Drive.Control.Speed.ActualValue;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

9.2.5 Setpoint

Beschreibung

Datentyp	Bedeutung
Double	Liest die aktuelle Solldrehzahl für den Drehzahlregler in [U/min]. Es handelt sich um den Ausgabewert des Positionsreglers; er wird vom Wert der Drehzahlbegrenzung limitiert.

Syntax

```
SollDrehzahl := AX1.Drive.Control.Speed.Setpoint;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

9.2.6 Isum

Beschreibung

Datentyp	Bedeutung
Double	Liest den berechneten I-Anteil des Drehzahlreglers oder schreibt einen anwendungsspezifischen Wert in [Nm]. Dieser Wert wird vom Drehzahlregler in jedem Reglerzyklus durch Addition der aktuellen Regelabweichung neu berechnet.

Syntax

```
AktuellerIAnteil := AX1.Drive.Control.Speed.Isum;
AX4.Drive.Control.Speed.Isum := 0.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

HINWEIS



Schreibender Zugriff: Gefahr von unkontrollierten Bewegungen

Eine Änderung während des Maschinenablaufs kann zu unkontrollierten Bewegungen führen, wenn der geschriebene Wert nicht zu der Belastung des Antriebs passt.

- ► Vermeiden Sie Änderungen während des Maschinenbetriebs.
- ► Änderungen sind in der Regel nur während der Inbetriebnahme der Achse zum Abgleich der Regelung sinnvoll.

Ein denkbarer Anwendungsfall ist folgender: Wenn eine Achse bewusst in eine Blockliersituation gefahren wird (z. B. zum Pressen), kann es hilfreich sein den I-Anteil einmalig zu löschen, bevor die Positionierung aus der Blockierung heraus gestartet wird.

9.2.7 IsumPreset

Beschreibung

Mit der Eigenschaft können Sie den Vorgabewert des I-Anteils des PI-Drehzahlreglers bei der Pulsfreigabe des Servoverstärkers lesen oder anpassen.

Dieser Wert wirkt wie ein „Start-Drehmoment“ bei der Pulsfreigabe. Ein Wert ungleich Null kann bei Vertikal-Achsen sinnvoll sein, um ein „Absacken“ beim Einschalten zu verhindern. Ein sinnvoller Wert für `IsumPreset` kann z. B. ermittelt werden, indem der Wert für `Isum` im Stillstand einer Vertikal-Achse vor dem Sperren ausgelesen wird.

Datentyp	Bedeutung
Double	Liest oder schreibt den Vorgabewert für das Drehmoment bei der Pulsfreigabe [Nm].

Syntax

```
DrehzahlreglerPreset := AX1.Drive.Control.Speed.IsumPreset;
AX4.Drive.Control.Speed.IsumPreset := 1.25;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

9.2.8 OverloadFactor

Beschreibung

Mit der Eigenschaft können Sie den Überlastfaktor für die Drehzahleinstellung des Servoverstärkers lesen oder anpassen.

Ausgehend von der Nenndrehzahl des Motors kann über diese Eigenschaft die maximal zulässige Drehzahl festgelegt werden. Diese kann je nach Art der Anwendung über oder unter dem Nennwert liegen.

Datentyp	Bedeutung
Double	Liest oder schreibt den Überlastfaktor der Drehzahl [%].

Syntax

```
DrehzahlUeberlastfaktor := AX1.Drive.Control.Speed.OverloadFactor;
AX4.Drive.Control.Speed.OverloadFactor := 100.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Ein geänderter Wert wirkt sich jedoch u. U. erst nach einem (erneuten) Wechsel in den Zustand **Freigegeben** aus.

9.2.9 MaxMotorSpeed

Beschreibung

Datentyp	Bedeutung
Double	Liest die maximal zur Verfügung stehende Motordrehzahl in [U/min]. Der Wert errechnet sich aus der Nenndrehzahl des Motors und dem Überlastfaktor.

Syntax

```
MaxDrehzahl := AX1.Drive.Control.Speed.MaxMotorSpeed;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

9.2.10 SetpointLimit

Beschreibung

Mit der Eigenschaft können Sie eine dynamische Drehzahlbegrenzung während des Maschinenprozesses einstellen. Der Wert ist bezogen auf die maximal zur Verfügung stehende Motordrehzahl (MaxMotorSpeed [▶ 64]).

Datentyp	Bedeutung
Double	Liest oder schreibt die Begrenzung für die Solldrehzahl in [%].

Syntax

```
DrehzahlLimit := AX1.Drive.Control.Speed.SetpointLimit;
AX4.Drive.Control.Speed.SetpointLimit := 100.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

9.2.11 DifferenceWindow

Beschreibung

Mit der Eigenschaft können Sie die Grenze für die Überwachung der Drehzahlabweichung lesen oder anpassen. Kann der Antrieb der Vorgabe des Positionsreglers nicht folgen und ergibt sich eine Abweichung von Soll- und Ist-Drehzahl, die den in dieser Eigenschaft definierten Wert übersteigt, so wird ein Fehler ausgelöst. Auf diese Weise wird beispielsweise das ungewollte Blockieren eines Antriebs überwacht.

Datentyp	Bedeutung
Double	Liest oder schreibt die Fehlerschwelle für die Abweichung der Ist- von der Soll-Drehzahl in [%].

Syntax

```
DrehzahlFehlerLimit := AX1.Drive.Control.Speed.DifferenceWindow;
AX4.Drive.Control.Speed.DifferenceWindow := 10.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

9.2.12 TotalInertia

Beschreibung

Mit der Eigenschaft können Sie die Einstellung für das Massenträgheitsmoment der Mechanik bezogen auf die Motorwelle lesen oder anpassen. Der Wert dieser Eigenschaft fließt als Faktor in die Berechnung der Drehmomentvorsteuerung ein. Besonders in sehr dynamischen Anwendungen hat die Drehmomentvorsteuerung einen wichtigen Einfluss auf das Regelergebnis einer MCX-Achse.

Die Drehmomentvorsteuerung ermittelt aus der berechneten Sollbeschleunigung multipliziert mit der Skalierung (TorqueFeedForwardScale [▶ 66]) und dem Trägheitsmoment (`TotalInertia`) ein Solldrehmoment für den Drehmomentregler. Dieser Wert wird zur Ausgabe des Drehzahlreglers addiert.

Datentyp	Bedeutung
Double	Liest oder schreibt das Gesamt-Trägheitsmoment bezogen auf die Motorwelle in [kg cm²].

Syntax

```
GesamtTraegheitsmoment := AX1.Drive.Control.Speed.TotalInertia;
AX4.Drive.Control.Speed.TotalInertia := 8.53;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

9.2.13 TorqueFeedForwardScale

Beschreibung

Mit der Eigenschaft können Sie die Skalierung der Wirkung der Drehmomentvorsteuerung lesen oder anpassen. Der Wert dieser Eigenschaft fließt als Faktor in die Berechnung der Drehmomentvorsteuerung ein. Besonders in sehr dynamischen Anwendungen hat die Drehmomentvorsteuerung einen wichtigen Einfluss auf das Regelergebnis einer MCX-Achse.

Die Drehmomentvorsteuerung ermittelt aus der berechneten Sollbeschleunigung multipliziert mit der Skalierung (`TorqueFeedForwardScale`) und dem Trägheitsmoment (TotalInertia [▶ 65]) ein Solldrehmoment für den Drehmomentregler. Dieser Wert wird zur Ausgabe des Drehzahlreglers addiert.

Datentyp	Bedeutung
Double	Liest oder schreibt die Skalierung der Wirkung der Drehmomentvorsteuerung in [%].

Syntax

```
DrehmomentVorsteuerSkalierung := AX1.Drive.Control.Speed.TorqueFeedForwardS-
cale;
AX4.Drive.Control.Speed.TorqueFeedForwardScale := 90.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

10 MCAxis.Drive.Control.Torque

Mit der Funktionalität **MCAxis.Drive.Control.Torque** können Sie die Eigenschaften des Strom-/Drehmomentreglers des Servoverstärkers der MCX-Achse lesen und anpassen.

Es handelt sich dabei im Wesentlichen um die Parameter, die im JetSym-Programmiertool im Motion-Setup auf den Inbetriebnahme-Seiten Regelung > Strom/Drehmoment, und Begrenzungswerte für Bucher-Automation-Achsen zur Verfügung stehen. Durch die Initialisierung der Steuerung werden die Parameter mit den bei der Inbetriebnahme abgespeicherten Werten belegt.

Mithilfe der Motion-API-Befehle ist es möglich, diese Parameter auch über das STX-Programm auszulesen und teilweise anzupassen.

In einem Servoverstärker wird die Motorregelung so ausgeführt, dass der Motorstrom proportional zum Motordrehmoment ist. Aus diesem Grund kann ein solcher Regler sowohl als Strom- als auch als Drehmomentregler bezeichnet werden, obwohl tatsächlich nur der Strom geregelt wird. Die Drehmomentwerte werden durch Multiplikation mit der Drehmomentkonstante aus den Stromwerten berechnet.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

10.1 Syntax für MCAxis.Drive.Control.Current

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.Control.Torque.<Zugriff auf Wert>;
```

10.2 Eigenschaften von MCAxis.Drive.Control.Torque

Die Funktionalität bietet folgende Eigenschaften:

- ActualValue
- Setpoint
- OverloadFactor
- MaxMotorTorque
- SetpointLimit

10.2.1 ActualValue

Beschreibung

Datentyp	Bedeutung
Double	Liest das aktuelle Motordrehmoment, das aus dem gemessenen Motorstrom des Servoverstärkers ermittelt wird in [Nm]

Syntax

```
IstDrehmoment := AX1.Drive.Control.Torque.ActualValue;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

10.2.2 Setpoint

Beschreibung

Datentyp	Bedeutung
Double	Liest das aktuelle Solldrehmoment für den Drehmoment-/Stromregler in [Nm]. Es handelt sich um den Ausgabewert des Drehzahlreglers; er wird vom Wert der Drehzahlbegrenzung limitiert.

Syntax

```
SollDrehmoment := AX1.Drive.Control.Torque.Setpoint;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

10.2.3 OverloadFactor

Beschreibung

Ausgehend vom Nenndrehmoment des Motors kann über diese Eigenschaft das maximal zulässige Drehmoment festgelegt werden. Diese kann je nach Art der Anwendung über oder unter dem Nennwert liegen.

Datentyp	Bedeutung
Double	Liest oder schreibt den Überlastfaktor für die Drehmomenteinstellung des Servoverstärkers in [%].

Syntax

```
Ueberlastfaktor := AX1.Drive.Control.Torque.OverloadFactor;
AX4.Drive.Control.Torque.OverloadFactor := 200.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Ein geänderter Wert wirkt sich jedoch u. U. erst nach einem (erneuten) Wechsel in den Zustand **Freigegeben** aus.

10.2.4 MaxMotorTorque

Beschreibung

Datentyp	Bedeutung
Double	Liest das maximal zur Verfügung stehende Motordrehmoment in [Nm]. Der Wert errechnet sich aus dem Nenndrehmoment des Motors und dem Überlastfaktor.

Syntax

```
MaxDrehmoment := AX1.Drive.Control.Torque.Torque;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

10.2.5 SetpointLimit

Beschreibung

Mit der Eigenschaft können Sie eine dynamische Drehmomentbegrenzung während des Maschinenprozesses einstellen. Der Wert ist bezogen auf das maximal zur Verfügung stehende Motordrehmoment (MaxMotorTorque [▶ 68]).

Datentyp	Bedeutung
Double	Liest oder schreibt die Begrenzung für die Solldrehzahl in [%].

Syntax

```
DrehmomentLimit := AX1.Drive.Control.Torque.SetpointLimit;
AX4.Drive.Control.Torque.SetpointLimit := 25.3;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

11 MCAxis.Drive.Diagnostics

Mit der Funktionalität **MCAxis.Drive.Diagnostics** können Sie die Fehlerbehandlung des Servoverstärkers einer MCX-Achse verwalten.

Geltungsbereich der Funktionalität

Die hier aufgeführten Funktionen und Eigenschaften beziehen sich auf Fehler und Warnungen , die von den Servoverstärkern (z. B. JetMove 3xxx) erkannt werden.

Fehler rufen eine Systemreaktion hervor, Warnungen hingegen nicht. Fehler müssen vom Anwender quittiert werden, um den Betrieb des Antriebs fortzusetzen, während Warnungen selbstständig gelöscht werden, sobald die Ursache nicht mehr vorliegt.

Die Bedeutung der Fehlernummern ist in der JetSym–Online-Hilfe oder in der Bedienungsanleitung des Servoverstärkers aufgeführt.

Mit dem Befehl `MCAxis.Diagnostics.ClearErrors()` quittieren Sie alle Fehler, auch die der Servoverstärker.

Fehler und Meldungen, die sich auf Positionsgenerierungsfunktionen (MCX) beziehen, werden über den die Funktionalität **MCAxis.Diagnostics** verwaltet.

11.1 Syntax für MCAxis.Drive.Diagnostics

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.Diagnostics.<Funktion oder Zugriff auf Wert>;
```

11.2 Funktionen von MCAxis.Drive.Diagnostics

Die Funktionalität bietet folgende Funktionen:

– Is WarningPending()

11.2.1 IsWarningPending()

Die Funktion gibt zurück, ob eine bestimmte bzw. ob überhaupt eine Warnung des Servoverstärkers vorliegt. Nicht jeder Servoverstärkertyp stellt alle Warnungen zur Verfügung.

Übergabeparameter

Parameter	Datentyp	Bedeutung
WarningNr	DriveWarningTypes (optional)	
	DriveWarningTypes.AnyWarning (Default)	Der Servoverstärker zeigt eine Warnung an.
	DriveWarningTypes.UndervoltageD-CLink	– Die Zwischenkreisspannung ist zu niedrig. – Es liegt eine Zwischenkreisspannung an, während die Regelung gesperrt ist.
	DriveWarningTypes.Overtemperature-Drive	Die Temperatur des Kühlkörpers oder des Geräteinnenraums ist nahe an der Übertemperatur-Abschaltschwelle.
	DriveWarningTypes.Overtemperature-Motor	Die Motortemperatur ist nahe an der Übertemperatur-Abschaltschwelle.
	DriveWarningTypes.STOactive	Die STO-Funktion ist aktiv, während die Regelung gesperrt ist.
	DriveWarningTypes.PowerSupply	Eine Warnung in der Spannungsversorgung liegt vor.
	DriveWarningTypes.I2TDrive	Der I²t-Wert für den Gerätestrom liegt nahe an der Abschaltschwelle.
	DriveWarningTypes.I2TMotor	Der I²t-Wert für den Motorstrom liegt nahe an der Abschaltschwelle.

Syntax

```
If AX1.Drive.Diagnostics.IsWarningPending(DriveWarningTypes.AnyWarning) then
...
End_if
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

11.3 Eigenschaften von MCAxis.Drive.Diagnostics

Die Funktionalität bietet folgende Eigenschaften:

– AcutalError
– IsErrorPending

11.3.1 ActualError

Beschreibung

Eigenschaft	Datentyp	Bedeutung
ActualError	dword	Liest die Fehlernummer eines am Servoverstärker anliegenden Fehlers. Ist der Rückgabewert `0`,ist der Servoverstärker fehlerfrei.

Syntax

```
AxisError := AX1.Drive.Diagnostics.ActualError;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

11.3.2 IsErrorPending

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob ein Fehler am Servoverstärker der MCX-Achse vorliegt. `0`: Der Servoverstärker ist fehlerfrei. `1`: Der Servoverstärker zeigt einen Fehler an. Dies ist gleichbedeutend mit einem Vergleich von `ErrorCount` mit `0`.

Syntax

```
If AX1.Drive.Diagnostics.IsErrorPending then
...
End_if;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

12 MCAxis.Drive.DigitalInput

Mit der Funktionalität **MCAxis.Drive.DigitalInput** können Sie die Eigenschaften der digitalen Eingänge des Servoverstärkers der MCX-Achse lesen und anpassen.

Es handelt sich dabei im Wesentlichen um die Parameter, die im JetSym-Programmiertool im Motion-Setup auf der Inbetriebnahme-Seite EA für Bucher-Automation-Achsen zur Verfügung stehen. Durch die Initialisierung der Steuerung werden die Parameter mit den bei der Inbetriebnahme abgespeicherten Werten belegt.

Mithilfe der Motion-API-Befehle ist es möglich, diese Parameter auch über das STX-Programm auszulesen und teilweise anzupassen.

Die Signale an den digitalen Eingängen werden in der MCX für Funktionen von Referenzschaltern, Endschaltern und STO (safe torque-off) verwendet. Den Zustand der End- und Referenzschalter fragen Sie über die Eigenschaften der Funktionalität MCAxis.Mechanism.Monitoring [▶ 138] ab.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

12.1 Syntax für MCAxis.Drive.DigitalInput

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.DigitalInput.<Zugriff auf Wert>;
```

12.2 Eigenschaften von MCAxis.Drive.DigitalInput

Die Funktionalität bietet folgende Eigenschaften:

- PolarityReferenceSwitch
- LimitSwitchMode
- PolarityLimitSwitchMin
- PolarityLimitSwitchMax
- IsPulseEnabled

12.2.1 PolarityReferenceSwitch

Beschreibung

Mit der Eigenschaft können Sie die Polaritätseinstellung für den Referenzschalter, der am entsprechenden digitalen Eingang des Servoverstärkers angeschlossen ist, lesen oder anpassen. So definieren Sie, bei welcher Spannung am digitalen Eingang der Referenzschalter als aktiv ausgewertet werden soll.

Datentyp	Bedeutung
DrivePolarityModes	Liest oder schreibt den Typ der Polaritätseinstellung.
DrivePolarityModes.HighActive	Der Referenzschalter ist aktiv, wenn ein 24-V-Signal am digitalen Eingang anliegt.
DrivePolarityModes.LowActive	Der Referenzschalter ist aktiv, wenn ein 0-V-Signal am digitalen Eingang anliegt.

Syntax

```
ReferenzschalterPolaritaet := AX1.Drive.DigitalInput.PolarityReferenceSwitch;
AX1.Drive.DigitalInput.PolarityReferenceSwitch := DrivePolarityModes.HighActive;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

12.2.2 LimitSwitchMode

Beschreibung

Mit der Eigenschaft können Sie die Funktion der Hardware-Endschalter, die am entsprechenden digitalen Eingang des Servoverstärkers angeschlossen sind, lesen oder anpassen. So definieren Sie, ob ein Eingangssignal als Endschalter verwendet werden soll.

Datentyp	Bedeutung
DriveLimitSwitchModes	Liest oder schreibt den Typ der Endschalterfunktion.
DriveLimitSwitchModes.Disabled	Hardware-Endschalter werden nicht verwendet (ausgewertet).
DriveLimitSwitchModes.Both	Beide Hardware-Endschalter werden verwendet (ausgewertet).
DriveLimitSwitchModes.Min	Nur der Hardware-Endschalter in negative Verfahrrichtung wird verwendet (ausgewertet).
DriveLimitSwitchModes.Max	Nur der Hardware-Endschalter in positive Verfahrrichtung wird verwendet (ausgewertet).

Syntax

```
AktiveEndschalter := AX1.Drive.DigitalInput.LimitSwitchMode;
AX1.Drive.DigitalInput.LimitSwitchMode := DriveLimitSwitchModes.Both;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

12.2.3 PolarityLimitSwitchMin

Beschreibung

Mit der Eigenschaft können Sie die Polaritätseinstellung für den Hardware-Endschalter in negative Verfahrrichtung, der am entsprechenden digitalen Eingang des Servoverstärkers angeschlossen sind, lesen oder anpassen. So definieren Sie, bei welcher Spannung am digitalen Eingang der Endschalter als aktiv ausgewertet werden soll.

Datentyp	Bedeutung
DrivePolarityModes	Liest oder schreibt den Typ der Polaritätseinstellung.
DrivePolarityModes.HighActive	Der Hardware-Endschalter in negative Verfahrrichtung ist aktiv, wenn ein 24-V-Signal am digitalen Eingang anliegt.
DrivePolarityModes.LowActive	Der Hardware-Endschalter in negative Verfahrrichtung ist aktiv, wenn ein 0-V-Signal am digitalen Eingang anliegt.

Syntax

```
PolaritaetEndschalterNeg := AX1.Drive.DigitalInput.PolarityLimitSwitchMin;
AX1.Drive.DigitalInput.PolarityLimitSwitchMin := DrivePolarityModes.LowActive;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

12.2.4 PolarityLimitSwitchMax

Beschreibung

Mit der Eigenschaft können Sie die Polaritätseinstellung für den Hardware-Endschalter in positive Verfahrrichtung, der am entsprechenden digitalen Eingang des Servoverstärkers angeschlossen sind, lesen oder anpassen. So definieren Sie, bei welcher Spannung am digitalen Eingang der Endschalter als aktiv ausgewertet werden soll.

Datentyp	Bedeutung
DrivePolarityModes	Liest oder schreibt den Typ der Polaritätseinstellung.
DrivePolarityModes.HighActive	Der Hardware-Endschalter in positive Verfahrrichtung ist aktiv, wenn ein 24-V-Signal am digitalen Eingang anliegt.
DrivePolarityModes.LowActive	Der Hardware-Endschalter in positive Verfahrrichtung ist aktiv, wenn ein 0-V-Signal am digitalen Eingang anliegt.

Syntax

```
PolaritaetEndschalterPos := AX1.Drive.DigitalInput.PolarityLimitSwitchMax;
AX1.Drive.DigitalInput.PolarityLimitSwitchMax := DrivePolarityModes.LowActive;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden.

12.2.5 IsPulseEnabled

Beschreibung

Mit der Eigenschaft können Sie Sie den Status der Eingänge für sicheren Halt (STO) des Servoverstärkers abfragen.

Die Abfrage gibt `TRUE` zurück, wenn die entsprechenden digitalen Eingänge so beschaltet sind, dass die Verstärkerendstufe freigegeben ist, d.h. es können Spannungsimpulse auf den Motor ausgegeben werden.

Datentyp	Bedeutung
Bool	Liest den Status der STO-Funktion des Servoverstärkers: `True`: Der Motor kann betrieben werden. `False`: Der Motor ist sicher gesperrt gegen die Erzeugung von Drehmoment.

Syntax

```
STOStatus := AX1.Drive.DigitalInput.IsPulseEnabled;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

13 MCAxis.Drive.EncoderChannel

Mit der Funktionalität **MCAxis.Drive.EncoderChannel** können Sie die Eigenschaften der angeschlossenen Positionsgeber (Encoder) der MCX-Achse lesen und anpassen.

Es handelt sich dabei im Wesentlichen um die Parameter, die im JetSym-Programmiertool im Motion-Setup auf der Inbetriebnahme-Seite Geber für Bucher-Automation-Achsen zur Verfügung stehen. Durch die Initialisierung der Steuerung werden die Parameter mit den bei der Inbetriebnahme abgespeicherten Werten belegt.

Mithilfe der Motion-API-Befehle ist es möglich, diese Parameter auch über das STX-Programm auszulesen und teilweise zu ändern.

Die Servoverstärker besitzen in der Regel 2 Schnittstellen für einen Positionsgeber. Darum ist diese Funktionalität zwei Mal vorhanden.

Eine Positionsrückführung wird für die Kommutierung sowie für Geschwindigkeits- und Lageregelung benötigt. Es sind verschiedene Typen von Positionsgebern anschließbar. Welche Typen im Anwendungsfall unterstützt werden, können Sie der Bedienungsanleitung des Servoverstärkers entnehmen.

Der zweite Positionsgeber kann für 2 unterschiedliche Funktionen verwendet werden:

1. Als zusätzlicher Positionsgeber für die Achse, um etwaigen Schlupf zwischen Material und Motor auszugleichen.

   In manchen Anwendungen ist es sinnvoll, zusätzlich zum Motor-Positionsgeber einen zweiten Positionsgeber direkt an der Last zu montieren. In diesem Fall dient der Motor-Positionsgeber für die Kommutierung und Geschwindigkeitsregelung, und der zusätzliche Positionsgeber für die Positionsregelung. Dies führt dazu, dass die Positionsregelung direkt an der Last und nicht auf die Rotorstellung des Motors erfolgt. Ein Schlupf zwischen Motorwelle und Last wird so automatisch ausgeglichen.

   Vorgehen:

   Wählen Sie für den Parameter `AX1.Drive.Control.Position.FeedbackSource` den Wert `DriveFeedbackSources.Encoder2` oder stellen Sie im Motion-Setup von JetSym auf der Seite Regelung > Lage den Parameter Geber für Lageregelung auf Geber 2.

2. Als Position einer externen Achse in einem Technologieverbund.

   Vorgehen:

   Wählen Sie im Motion-Setup von JetSym auf der Seite Achsparameter die Option Geber 2 als externe Achse nutzen.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

13.1 Syntax für MCAxis.Drive.EncoderChannel

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.EncoderChannel[1..2].<Zugriff auf Wert>;
```

13.2 Eigenschaften von MCAxis.Drive.EncoderChannel

Die Funktionalität bietet folgende Eigenschaften:

– EncType
– EncLines

– GearMotor
– GearEncoder
– MultiTurnPosition
– SingleTurnPosition
– MaxNoOfAbsRevolutions

13.2.1 EncType

Beschreibung

Mit der Eigenschaft können Sie den Typ des Positionsgebers, der am entsprechenden digitalen Eingang des Servoverstärkers angeschlossen sind, lesen oder anpassen.

Datentyp	Bedeutung
DriveEncoderTypes	Liest oder schreibt den Typ des angeschlossenen Positionsgebers.
DriveEncoderTypes.Disabled	Es ist kein Positionsgeber angeschlossen.
DriveEncoderTypes.Resolver	Es ist ein Resolver angeschlossen.
DriveEncoderTypes.Hiperface	Es ist ein Positionsgeber mit Hiperface-Schnittstelle angeschlossen.
DriveEncoderTypes.EnDat	Es ist ein Positionsgeber mit EnDat-2.2-Schnittstelle angeschlossen.
DriveEncoderTypes. RectangleIncrementalEncoder	Es ist ein Inkrementalgeber mit Rechtecksignalen angeschlossen.
DriveEncoderTypes.SinCosIncrementalEncoder	Es ist ein Inkrementalgeber mit Sinussignalen angeschlossen.
DriveEncoderTypes.HiperfaceDSL	Es ist ein Positionsgeber mit Hiperface-DSL-Schnittstelle angeschlossen.
DriveEncoderTypes.LinMot	Es ist ein LinMot-Antrieb angeschlossen.

Syntax

```
Positionsgeber1 := AX1.Drive.EncoderChannel[1].EncType;
AX1.Drive.EncoderChannel[1].EncType := DriveEncoderTypes.Resolver;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Ein geänderter Wert wirkt sich jedoch erst beim Zustandsübergang von **Gesperrt** nach **Freigegeben** aus.

13.2.2 EncLines

Beschreibung

Mit der Eigenschaft können Sie den Parameter für die Auflösung des Positionsgebers, der an der entsprechenden Positionsgeberschnittstelle des Servoverstärkers angeschlossen ist, lesen oder anpassen. Je nach Typ des Positionsgebers hat der Auflösungsparameter eine andere Bedeutung (siehe nachfolgende Tabelle).

Datentyp	Positionsgebertyp	Bedeutung
Uint	-/-	Liest oder schreibt den Auflösungsparameter, dessen Bedeutung vom Positionsgebertyp abhängt:
	Resolver	Polpaarzahl (fast immer 1).
	Hiperface	Digitale Positionsauflösung wird beim Start übermittelt. Nur lesender Zugriff.
	EnDat	Digitale Positionsauflösung wird beim Start übermittelt. Nur lesender Zugriff.
	Rechteck-Inkrementalgeber	Strichzahl
	Sinus-Inkrementalgeber	Anzahl der Sinusperioden pro Umdrehung.
	Hiperface-DSL	Digitale Auflösung (nur lesender Zugriff).
	LinMot	Kein Eintrag erforderlich.

Syntax

```
Geberstrichzahl := AX1.Drive.EncoderChannel[1].EncLines;
AX1.Drive.EncoderChannel[1].EncLines := 2500;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Ein geänderter Wert wirkt sich jedoch erst beim Zustandsübergang von **Gesperrt** nach **Freigegeben** aus.

13.2.3 GearMotor

Beschreibung

Mit der Eigenschaft können Sie die Übersetzung zwischen der Motorwelle und dem Positionsgeber, der an der entsprechenden Positionsgeberschnittstelle des Servoverstärkers angeschlossen ist, lesen oder anpassen.

Über die Eigenschaft teilen Sie der MCX-Achse mit, um wie viele Umdrehungen sich die Motorwelle dreht, wenn sich der Positionsgeber um die Anzahl der Umdrehungen gemäß der Eigenschaft GearEncoder [▶ 79] dreht.

Datentyp	Bedeutung
Int	Liest oder schreibt die Anzahl der Motorumdrehungen in [Umdr].

Syntax

```
MotorUebersetzungMotor := AX1.Drive.EncoderChannel[1].GearMotor;
AX1.Drive.EncoderChannel[1].GearMotor := 1;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden.

13.2.4 GearEncoder

Beschreibung

Mit der Eigenschaft können Sie die Übersetzung zwischen der Motorwelle und dem Positionsgeber, der an der entsprechenden Positionsgeberschnittstelle des Servoverstärkers angeschlossen ist, lesen oder anpassen.

Über die Eigenschaft teilen Sie der MCX-Achse mit, um wie viele Umdrehungen sich der Positionsgeber dreht, wenn sich die Motorwelle um die Anzahl der Umdrehungen gemäß der Eigenschaft GearMotor [▶ 78] dreht.

Datentyp	Bedeutung
Int	Liest oder schreibt die Anzahl der Positionsgeberumdrehungen in [Umdr].

Syntax

```
EncoderUebersetzungEncoder := AX1.Drive.EncoderChannel[1].GearEncoder;
AX1.Drive.EncoderChannel[1].GearEncoder := 3;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden.

13.2.5 MultiTurnPosition

Beschreibung

Mit der Eigenschaft können Sie den Ist-Wert des Umdrehungszählers des Positionsgebers, der an der entsprechenden Positionsgeberschnittstelle des Servoverstärkers angeschlossen ist, lesen.

Datentyp	Bedeutung
Integer	Liest den Wert des Umdrehungszählers des Positionsgebers in [Umdr]. Der Wert ist nur bei Absolutwertgebern ungleich null.

Syntax

```
EncoderUmdrehungen := AX1.Drive.EncoderChannel[1].MultiTurnPosition;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

13.2.6 SingleTurnPosition

Beschreibung

Mit der Eigenschaft können Sie die Ist-Winkelstellung der Welle des Positionsgebers, der an der entsprechenden Positionsgeberschnittstelle des Servoverstärkers angeschlossen ist, lesen.

Datentyp	Bedeutung
Int	Liest die Winkelstellung der Welle des Positionsgebers in [°].

Syntax

```
EncoderWinkel := AX1.Drive.EncoderChannel[1].SingleTurnPosition;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

13.2.7 MaxNoOfAbsRevolutions

Beschreibung

Mit der Eigenschaft können Sie die Anzahl der möglichen Absolut-Umdrehungen des Positionsgebers, der an der entsprechenden Positionsgeberschnittstelle des Servoverstärkers angeschlossen ist, lesen.

Der Wert wird von der MCX-Achse aus dem Absolutwertgeber ausgelesen und kann zur Identifikation des Positionsgebers verwendet werden.

Datentyp	Bedeutung
Int	Liest die Anzahl der Absolut-Umdrehungen, die der Positionsgeber zur Verfügung stellt, in [Umdr]. Der Wert ist nur bei Absolutwertgebern ungleich null.

Syntax

```
EncoderNoOfMultiturn := AX1.Drive.EncoderChannel[1].MaxNoOfAbsRevolutions;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

14 MCAxis.Drive.GenericAccess

Mit der Funktionalität **MCAxis.Drive.GenericAccess** können Sie auf spezielle Parameter des Servoverstärkers einer MCX-Achse zugreifen.

Für den Betrieb einer Achse über den MCX-Kernel aus JetSym heraus sind im Normallfall alle erforderlichen Parameter des Servoverstärkers normiert und werden in SI-Einheiten über den Motion-Setup und die Motion-API-Funktionen zur Verfügung gestellt.

In speziellen Fällen kann es hilfreich sein, auf einen Parameter auf einem Servoverstärker zuzugreifen, der nicht über die Motion-API-Befehle erreichbar ist. Üblicherweise stellt Ihnen der Technische Support der Bucher Automation hierfür Beispiel-Befehlssätze zur Verfügung.

HINWEIS



Gefahr von unerwünschtem Verhalten des Servoverstärkers!

Der generische Zugriff auf einen Parameter über den Befehl **MCAxis.Drive.GenericAccess** erfüllt die gewünschte Funktion nur für die Servoverstärker einer Baureihe! Der Befehl ist nicht automatisch auf andere Baureihen übertragbar.

► Verwenden Sie den Befehl nicht bei Servoverstärkern unterschiedlicher Baureihen.

Der generische Zugriff auf einen Servoverstärker wird in einer Bucher-Automation-Steuerung über das CANopen-Protokoll ausgeführt. Dies gilt für Kommunikationsbusse sowohl auf der Basis von CAN wie auch von EtherCat.

CANopen definiert für die Adressierung eines Parameters eine Objekt-Adresse (hexadezimal notierte16-Bit-Zahl) und einem Subindex (8-Bit-Zahl).

Für eine korrekte Funktionsweise des Befehls ist außerdem erforderlich, dass für den über *Objekt* und *Subindex* angesprochenen Parameter der richtige Datentyp verwendet wird. Aus diesem Grund gibt es in der Funktionalität **MXAxis.Drive.GenericAccess** für jeden Datentyp jeweils eine eigene Funktion für das Lesen und das Schreiben von Daten.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

14.1 Syntax für MCAxis.Drive.GenericAccess

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.GenericAccess.CANopen.<Funktion oder Zugriff auf
Wert>(optional <Paramter 1, Paramter 2, Parameter n>);
```

14.2 Funktionen von MCAxis.Drive.GenericAccess

Die Funktionalität bietet folgende Funktionen:

Lesezugriff:

- CANopen.ReadUInt8()
- CANopen.ReadInt8()
- CANopen.ReadUInt16()
- CANopen.ReadInt32()

– CANopen.ReadUInt32()
– CANopen.ReadInt32()
– CANopen.ReadFloat32()
– CANopen.ReadFloat64()
– CANopen.ReadVisibleString()

Schreibzugriff:

– CANopen.WriteUInt8()
– CANopen. WriteInt8()
– CANopen. WriteUInt16()
– CANopen. WriteInt32()
– CANopen. WriteUInt32()
– CANopen. WriteInt32()
– CANopen. WriteFloat32()
– CANopen. WriteFloat64()

14.2.1 CANopen.ReadUInt8()

Beschreibung

Mit der Funktion können Sie einen Parameter mit dem Datentyp `UInt8` aus dem Servoverstärker lesen.

Übergabeparameter

Parameter	Datentyp	Bedeutung
ObjectIndex	Int	Objektadresse des Parameters im Servoverstärker.
SubIndex	Word	Subindexnummer des CANopen-Objekts.

Analoges Vorgehen für alle Datentypen

In gleicher Weise greifen Sie auf Objekte mit einem anderen Datentyp zu. Dabei muss der Datentyp der Variable, der der Objektinhalt zugewiesen wird, dem Befehl angepasst sein.

Die zur Verfügung stehenden Befehle sind:

– CANopen.ReadInt8()
– CANopen.ReadUInt16()
– CANopen.ReadInt32()
– CANopen.ReadUInt32()
– CANopen.ReadInt32()
– CANopen.ReadFloat32()
– CANopen.ReadFloat64()
– CANopen.ReadVisibleString()

Syntax am Beispiel von .ReadUInt8()

```
test1 := AX1.Drive.GenericAccess.CANopen.ReadUInt8(0x2102,0x16);
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

14.2.2 CANopen.WriteUInt8()

Beschreibung

Mit der Funktion können Sie einen Wert in einen Parameter mit dem Datentyp `UInt8` auf dem Servoverstärker schreiben.

Übergabeparameter

Parameter	Datentyp	Bedeutung
ObjectIndex	Int	Objektadresse des Parameters im Servoverstärker.
SubIndex	Word	Subindexnummer des CANopen-Objekts.
Value	Byte	Wert, der in das Objekt geschrieben werden soll.

Analoges Vorgehen für alle Datentypen

In gleicher Weise schreiben Sie Daten in Objekte mit einem anderen Datentyp. Dabei muss der Datentyp des Parameters `Value` dem Befehl angepasst sein.

Die zur Verfügung stehenden Befehle sind:

– CANopen.WriteUInt8()
– CANopen.WriteInt8()
– CANopen.WriteUInt16()
– CANopen.WriteInt32()
– CANopen.WriteUInt32()
– CANopen.WriteInt32()
– CANopen.WriteFloat32()
– CANopen.WriteFloat64()

Syntax für am Beispiel von .WriteUInt8()

```
AX1.Drive.GenericAccess.CANopen.WriteUInt8(0x2101,0x19,1)
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

14.3 Eigenschaften von MCAxis.Drive.GenericAccess

Die Funktionalität bietet folgende Eigenschaften:

– CANopen.OSVersion
– CANopen.IsAvailable

14.3.1 CANopen.OSVersion

Beschreibung

Datentyp	Bedeutung
String	Liest die Firmware-Version des Servoverstärkers der MCX-Achse aus. Die FW-Version wird bei EtherCAT-/CANopen-Objekten als String gelesen.

Syntax

```
FW_Version := AX1.Drive.GenericAccess.CANopen.OSVersion;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

14.3.2 CANopen.IsAvailable

Beschreibung

Datentyp	Bedeutung
String	Liest, ob die MCX-Achse ein CANopen-Interface zur Verfügung stellt.

Syntax

```
If AX1.Drive.GenericAccess.CANopen.IsAvailable Then
…
End_if;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

15 MCAxis.Drive.InvertDirection

Mit der Funktionalität **MCAxis.Drive.InvertDirection** können Sie die Drehrichtung der Motorwelle einer MCX-Achse im Verhältnis zur Positionierrichtung lesen und anpassen.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

15.1 Syntax für MCAxis.Drive.InvertDirection

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.InvertDirection;
```

15.2 Eigenschaften von MCAxis.Drive.InvertDirection

Die Funktionalität bietet folgende Eigenschaften:

– InvertDirection

15.2.1 InvertDirection

Beschreibung

Mit der Eigenschaft können Sie die Drehrichtung der Motorwelle in Bezug auf die Positionierrichtung lesen oder anpassen.

Bei einer Standardverkabelung löst ein Positionierbefehl in positive Richtung üblicherweise eine Rechtsdrehung der Motorwelle aus (aus Sicht des Betrachters). Ist die Eigenschaft `MCAxis.Drive.InvertDirection` aktiviert, so bewirkt ein Positionierbefehl in positive Richtung statt dessen eine Linksdrehung der Motorwelle.

Datentyp	Bedeutung
DriveInvertDirectionModes	Liest oder legt fest, ob die Umkehr der Drehrichtung aktiv oder inaktiv ist.
DriveInvertDirectionModes.Disabled	Drehrichtungsumkehr ist deaktiviert.
DriveInvertDirectionModes.Enabled	Drehrichtungsumkehr ist aktiviert.

Syntax

```
StateDirectionInverting := AX1.Drive.Drive.InvertDirection;
AX1.Drive.InvertDirection := DriveInvertDirectionModes.Enabled;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Zwar ist das Schreiben der Eigenschaft ebenfalls in jedem Zustand möglich, jedoch **nicht sinnvoll** in den Zuständen **freigegeben** und **läuft**. Schreibender Zugriff in diesen Zustanden kann je nach Typ des Servoverstärkers dazu führen, dass die Eingabe bis zum nächsten Zustandswechsel auf **freigeben** ignoriert wird, oder dass der Servoverstärker einen Fehler ausgibt.

Da es sich um die Eigenschaft der Maschine handelt, wird diese nicht während des Betriebs geändert. Vielmehr wird diese Eigenschaft mit den anderen Achsdaten in der Steuerung gespeichert und beim Booten der Steuerung für die Achse wieder konfiguriert.

16 MCAxis.Drive.IsReadyForOperation

Mit der Funktionalität **MCAxis.Drive.IsReadyForOperation** können Sie lesen, ob der Servoverstärker bereit ist, in den Zustand **freigegeben** zu geschaltet zu werden.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

16.1 Syntax für MCAxis.Drive.IsReadyForOperation

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.IsReadyForOperation;
```

16.2 Eigenschaften von MCAxis.Drive.IsReadyForOperation

Die Funktionalität bietet folgende Eigenschaften:

– IsReadyForOperation

16.2.1 IsReadyForOperation

Beschreibung

Es wird empfohlen, diese Eigenschaft abzufragen, bevor der `Power.Enable`-Befehl gegeben wird.

Diese Eigenschaft ist in der Regel `true`, wenn folgende Bedingungen erfüllt sind:

– Es besteht eine Kommunikationsverbindung zum Servoverstärker.
– Der Servoverstärker ist fehlerfrei.
– Die Zwischenkreisspannung ist ausreichend hoch (Netzspannung ist zugeschaltet).
– Das STO-Signal (sicherer Halt) ist nicht aktiv.
– Ein Positionsgeber ist konfiguriert.

Datentyp	Bedeutung
Bool	Liest, ob der Servoverstärker einschaltbereit ist (Zustand **freigegeben**).

Syntax

```
If AX1.Drive.IsReadyForOperation then
...
end_if;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

17 MCAxis.Drive.Motor

Mit der Funktionalität **MCAxis.Drive.Motor** können Sie die Eigenschaften des Motors der MCX-Achse lesen und anpassen.

Es handelt sich dabei im Wesentlichen um die Parameter, die im JetSym-Programmiertool im Motion-Setup auf der Inbetriebnahme-Seiten Motor für Bucher-Automation-Achsen zur Verfügung stehen. Durch die Initialisierung der Steuerung werden die Parameter mit den bei der Inbetriebnahme abgespeicherten Werten belegt.

Mithilfe der Motion-API-Befehle ist es möglich, diese Parameter auch über das STX-Programm auszulesen und anzupassen.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

17.1 Syntax für MCAxis.Drive.Motor

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.Motor.<Funktion oder Zugriff auf Wert>(optional <Parameter>);
```

17.2 Funktionen von MCAxis.Drive.Motor

Die Funktionalität bietet folgende Funktionen:

– MeasureCommutationOffset()

17.2.1 MeasureCommutationOffset()

Die Funktion führt eine Messung des Kommutierungsoffsets für die MCX-Achse durch. Der übergebene Parameter definiert die Messmethode.

Die Messung des Kommutierungsoffsets wird benötigt, wenn dem System nach dem Einschalten nicht bekannt ist, welcher Wert für den Kommutierungsoffset gewählt werden muss. Das ist vor allem bei Achsen mit Inkremental- oder Sinus-Cosinus-Gebern der Fall, da diese Geber keine absolute Winkelposition für die Stellung des Rotors messen können.

HINWEIS

Gefahr von unbeabsichtigtem Anlauf

Das Aufrufen der Funktion bewirkt, dass der Motor Drehmoment aufbringt und in Bewegung gesetzt wird. Treffen Sie folgende Maßnahmen vor Verwendung der Funktion:

- ► Stellen Sie sicher, dass die Einstellungen der Reglerparameter, mindestens jedoch des Strom-/Drehmomentreglers, korrekt sind.
- ► Stellen Sie sicher, dass die Motorwelle frei drehen kann.

Übergabeparameter

Datentyp	Bedeutung
Enum DriveCommOffsetMeasurement-Modes (optional)	Legt die Messmethode für den Kommutierungsoffset fest.
DriveCommOffsetMeasurementMo-des.StaticMode	Die statische Methode bewirkt, dass der Motor in zwei Winkelstel-lungen einrastet, indem er auf bestimmte Weise bestromt wird. Dabei kann sich die Motorwelle im Bereich eines Polpaars des Motors bewegen.
DriveCommOffsetMeasurementMo-des.DynamicMode	Die dynamische Methode regelt das Drehfeld während der Mes-sung der sich ergebenden Bewegung nach. Diese Methode führt zu einer deutlich reduzierten Bewegung der Motorwelle, erfordert jedoch eine präzisere Einstellung der Regler.

Syntax

```
AX1.Drive.Motor.MeasureCommutationOffset(DriveCommOffsetMeasurementMo-
des.StaticMode);
```

Zustandsverwaltung

ⓘ Info

Exklusive Achsfunktion

Während die Funktion ausgeführt wird, sind keine anderen Achsfunktionen für diese Achse möglich. Der nächste STX-Befehl wird erst ausgeführt, wenn die Messfunktion beendet ist.

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während der Ausführung	Endzustand
Gesperrt	√	Inaktiv	Gesperrt
Freigegeben	X		Freigegeben
Läuft	X		Fehler: gesperrt
Unterbrochen	X		Freigegeben
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3 Eigenschaften von MCAxis.Drive.Motor

Die Funktionalität bietet folgende Eigenschaften:

– MotorDesign
– RatedCurrent
– MaxCurrent
– RatedSpeed
– RatedTorque

– Inductance
– Resistance
– VoltageConstant
– TorqueConstant
– PolePairs
– CommutationOffset
– ThermalDelayTime
– ShaftPosition
– Inertia
– TemperatureSensorType
– TemperatureSensorConnector
– Temperature
– I2tOperatingMode
– I2tValue

17.3.1 MotorDesign

Beschreibung

Mit der Eigenschaft können Sie die eingestellte Motorbauart lesen.

Bei manchen Servoverstärkern kann hier zusätzlich zum permanenterregten Synchronmotor (Servo) auch eine andere Bauart ausgewählt werden, beispielsweise ein Schrittmotor.

Datentyp	Bedeutung
DriveMotorDesignTypes (optional)	Bauart des Motors.
DriveMotorDesignTypes.Unknown	Der Antrieb kann nicht betrieben werden. Nur lesbar bei fehlerhafter Initialisierung.
DriveMotorDesignTypes.PMSM_Servo (Standard)	Permanenterregter Synchronmotor (Servomotor). Dies ist die vorherrschende Motorbauart in Servo-Anwendungen.
DriveMotorDesignTypes.Stepper_2Ph_CL	Geregelter Zweiphasen Schrittmotor. Diese Bauart erfordert einen Positionsgeber für die Regelung.
DriveMotorDesignTypes.Stepper_2pH_OL	Ungeregelter Zweiphasen-Schrittmotor.
DriveMotorDesignTypes.Stepper_3pH_OL	Ungeregelter Dreiphasen-Schrittmotor.
DriveMotorDesignTypes.BLDC_3Ph	Elektrisch kommutierter Dreiphasen-Gleichstrommotor.
DriveMotorDesignTypes.DC_HBridge	Permanenterregter Gleichstrommotor.
DriveMotorDesignTypes.ASM	Asynchronmotor.
DriveMotorDesignTypes.Reluctance	Reluktanzmotor.

Syntax

```
MotorBauart := AX1.Drive.Amplifier.MotorDesign;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.2 RatedCurrent

Beschreibung

Datentyp	Bedeutung
Double	Liest den Nennstrom des Motors, der dieser Achse zugeordnet ist in [A_{rms}].

Syntax

```
MotorNennstrom := AX1.Drive.Motor.RatedCurrent;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.3 MaxCurrent

Beschreibung

Datentyp	Bedeutung
Double	Liest den Maximalstrom des Motors, der dieser Achse zugeordnet ist in [A_{rms}].

Syntax

```
MotorMaxStrom := AX1.Drive.Motor.MaxCurrent;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.4 RatedSpeed

Beschreibung

Datentyp	Bedeutung
Double	Liest die Nenndrehzahl des Motors, der dieser Achse zugeordnet ist in [U/min].

Syntax

```
MotorNenndrehzahl := AX1.Drive.Motor.RatedSpeed;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben

Ausgangszustand	Erlaubt	Endzustand
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.5 RatedTorque

Beschreibung

Datentyp	Bedeutung
Double	Liest das Nenndrehmoment des Motors, der dieser Achse zugeordnet ist in [Nm].

Syntax

```
MotorNenndrehmoment := AX1.Drive.Motor.RatedTorque;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.6 Inductance

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert der Induktivität zwischen zwei Phasen des Motors, der dieser Achse zugeordnet ist in [mH].

Syntax

```
MotorInduktivität := AX1.Drive.Motor.Inductance;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.7 Resistance

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert des elektrischen Widerstands zwischen zwei Phasen des Motors, der dieser Achse zugeordnet ist in [Ω].

Syntax

```
MotorWiderstand := AX1.Drive.Motor.Resistance;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.8 VoltageConstant

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert der Spannungskonstante des Motors, der dieser Achse zugeordnet ist in [V_{eff}/1000 (U/min)].

Syntax

```
MotorSpannungskonstante := AX1.Drive.Motor.VoltageConstant;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.9 TorqueConstant

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert der Drehmomentkonstante des Motors, der dieser Achse zugeordnet ist in [Nm/A].

Syntax

```
MotorDrehmomentkonstante := AX1.Drive.Motor.TorqueConstant;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.10 PolePairs

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert der Polpaarzahl des Motors, der dieser Achse zugeordnet ist.

Syntax

```
MotorPolpaarzahl := AX1.Drive.Motor.PolePairs;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.11 CommutationOffset

Beschreibung

Der Kommutierungsoffset teilt dem Servoverstärker mit, wie sich die Magnetausrichtung des Rotors im Verhältnis zum Nullpunkt der Geberposition verhält.

Datentyp	Bedeutung
Double	Liest den Wert des Kommutierungsoffset des Motors, der dieser Achse zugeordnet ist in elektrischen Winkeleinheiten [°].

Syntax

```
MotorKommutierungsoffset := AX1.Drive.Motor.CommutationOffset;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.12 ThermalDelayTime

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert der thermischen Zeitkonstante des Motors, der dieser Achse zugeordnet ist in [s].

Syntax

```
MotorThermZeitkonstante := AX1.Drive.Motor.ThermalDelayTime;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.13 ShaftPosition

Beschreibung

Datentyp	Bedeutung
Double	Liest die Position der Motorwelle des Motors, der dieser Achse zugeordnet ist in [0 … 360°].

Syntax

```
MotorRotorposition := AX1.Drive.Motor.ShaftPosition;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

17.3.14 Inertia

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert des Trägheitsmoments des Motors, der dieser Achse zugeordnet ist in [kgcm²].

Syntax

```
MotorTrägheitsmoment := AX1.Drive.Motor.Inertia;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.15 TemperatureSensorType

Beschreibung

Mit der Eigenschaft können Sie den Typ des Motortemperatursensors lesen und schreiben. Der Sensortyp wird nicht automatisch erkannt, sondern muss gemäß dem Datenblatt des Motors angegeben werden.

Datentyp	Bedeutung
Typ DriveTemperatureSensorTypes (optional)	Typ des Motortemperatursensors.
DriveTemperatureSensorTypes.NoSensor (Default)	Kein Sensor angeschlossen. Der Sensor wird nicht ausgewertet, d.h. die Motortemperaturüberwachung ist nicht aktiv.
DriveTemperatureSensorTypes.NCC	Temperaturschalter; Die Motortemperatur ist nicht auslesbar, lediglich eine Übertemperaturüberwachung ist aktiv.

Datentyp	Bedeutung
DriveTemperatureSensorTypes.PTC	Temperaturabhängiger Widerstand mit nicht-linearer positiv ansteigender Kennlinie (PTC). Die Motortemperatur ist nicht auslesbar, lediglich eine Übertemperaturüberwachung ist aktiv.
DriveTemperatureSensorTypes.KTY83_110	Halbleiter-Sensor vom Typ KTY83-110.
DriveTemperatureSensorTypes.KTY84_130	Halbleiter-Sensor vom Typ KTY84-130.
DriveTemperatureSensorTypes.PT1000	Platin-Messwiderstand.
DriveTemperatureSensorTypes.MotorSpecific	Für spezielle Motor-Sensor-Kombinationen, z. B. Lin-Mot-Motor.

Syntax

```
MotorTempSensorTyp := AX1.Drive.Amplifier.TemperatureSensorType;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.16 TemperatureSensorConnector

Beschreibung

Mit der Eigenschaft können Sie lesen und schreiben, an welchen Anschluss der Motortemperatursensor angeschlossen ist.

Diese Eigenschaft ist nur bei Servoverstärkern der JM-3000-Familie verfügbar, da an diesen Geräten drei Anschlüsse für den Motortemperatursensor zur Verfügung stehen: der Motor-Stecker sowie beide Positionsgeberstecker einer Achse.

Datentyp	Bedeutung
Typ DriveTemperatureSensorConnectors (optional)	Auswahl des Anschlusssteckers für den Motortemperatursensor.

Datentyp	Bedeutung
DriveTemperatureSensorConnectors.Motor (Standard)	Der Motortemperatursensor ist am Motorstecker angeschlossen. **HINWEIS! Bei einem Motor mit Hiperface-DSL-Positionsgeber ist dieser Datentyp erforderlich.**
DriveTemperatureSensorConnectors.Enc1	Der Motortemperatursensor ist am Positionsgeberstecker mit der Bezeichnung „Enc1“ angeschlossen.
DriveTemperatureSensorConnectors.Enc2	Der Motortemperatursensor ist am Positionsgeberstecker mit der Bezeichnung „Enc2“ angeschlossen.

Syntax

```
MootrTempSensorAnschluss := AX1.Drive.Motor.TemperatureSensorConnector;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.17 Temperature

Beschreibung

Datentyp	Bedeutung
Double	Liest die Ist-Temperatur des Motors, der dieser Achse zugeordnet ist in [°C].

Syntax

```
MotorIstTemperatur := AX1.Drive.Motor.Temperature;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

17.3.18 I2tOperatingMode

Beschreibung

Datentyp	Bedeutung
DriveI2tOperatingModes (optional)	Aktivierungszustand der Motor-I²t-Überwachung
DriveI2tOperatingModes.Disable (Standard)	Die I²t-Überwachung ist deaktiviert.
DriveI2tOperatingModes.Enable	Die I²t-Überwachung ist aktiviert. Für eine sinnvolle Überwachung müssen die Werte für die thermische Zeitkonstante [▶ 98] und den Nennstrom [▶ 91] zum verwendeten Motor passen.

Syntax

```
MotorI2tAktivierung := AX1.Drive.Amplifier.I2tOperatingMode;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Auf diese Eigenschaft kann auch schreibend zugegriffen werden, was jedoch unter normalen Umständen nicht sinnvoll ist, da der Motor nicht während des Betriebs der Maschine gewechselt werden kann.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

17.3.19 I2tValue

Beschreibung

Dieser Wert wird nur berechnet, wenn die I²t-Berechnung aktiviert ist. Für einen sinnvollen Wert müssen die Werte für die thermische Zeitkonstante [▶ 98] den Nennstrom [▶ 91] zum verwendeten Motor passen.

Datentyp	Bedeutung
Double	Liest den I²t-Integrationswert des Motors, der dieser Achse zugeordnet ist in [%].

Syntax

```
MotorI2tWert := AX1.Drive.Amplifier.I2tValue;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

18 MCAxis.Drive.QuickStopDeceleration

Mit der Funktionalität **MCAxis.Drive.QuickStopDeceleration** können Sie die Schnellhaltrampe einer MCX-Achse lesen oder anpassen.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

18.1 Syntax für MCAxis.Drive.QuickStopDeceleration

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.QuickStopDeceleration;
```

18.2 Eigenschaften von MCAxis.Drive.QuickStopDeceleration

Die Funktionalität bietet folgende Eigenschaften:

– QuickStopDeceleration

18.2.1 QuickStopDeceleration

Beschreibung

Mit der Eigenschaft können Sie die Verzögerung lesen oder anpassen, mit der die Achse durch Auslösen des Befehls `Power.Quickstop` angehalten wird. Diese Rampe wird ebenfalls als Reaktion auf einzelne die im Servoverstärker ausgelöste Fehler angewendet.

Die Schnellhaltrampe wird vom Servoverstärker ausgeführt. Die Motion-Control-Funktion der Steuerung hat während der Ausführung der Schnellhaltrampe keinen Einfluss mehr auf die Bewegung.

Der Servoverstärker hält die Motorwelle mit einer linearen Verzögerungsrampe drehzahlgeregelt an. Die Einstellung sollte sichererstellen, dass die Mechanik, die von der MCX-Achse bewegt wird, gerade noch ohne Schaden gestoppt werden kann.

Datentyp	Bedeutung
Double	Liest oder schreibt den Wert für die Verzögerung der Schnellhaltrampe in [rpm/s].

Syntax

```
EmergencyDeceleration := AX1.Drive.QuickStopDeceleration;
AX1.Drive.QuickStopDeceleration := 6000.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

19 MCAxis.Drive.TorqueShutOff

Mit der Funktionalität **MCAxis.Drive.TorqueShutOff** können Sie eine MCX-Achse aufgrund des Erreichens einer Drehmomentschwelle sehr schnell anhalten und danach auf eine Position oder ein Haltemoment regeln. Die Funktionalität stellt einen ganzen Ablauf an Aktionen abhängig von Drehmoment- und Drehzahlwerten zur Verfügung.

ⓘ Info

Erforderliche Anpassung von Parametern, während Drehmomentabschaltung aktiv ist

Die Funktionalität **TorqueShutOff** beeinflusst die Regelung im Servoverstärker bewusst in ungewöhnlicher Weise. Außerdem wird die MCX-Achse durch das Zudrehen fast zum Stillstand gebracht, ohne dass eine Zielposition erreicht wird. Die MCX-Sollwertgenerierung bekommt davon nichts mit. Sie gibt weiterhin Positions-Sollwerte aus, die dem `MovePtp.Start`-Befehl entsprechen. Das hat zur Folge, dass der Positionsschleppfehler wie auch die Drehzahl-Soll-Ist-Differenz höher werden können, als im üblichen Positionierbetrieb.

Aus diesem Grund kann es erforderlich werden, die Parameter für Schleppfehler-Grenze und Drehzahl-Abweichung Fenster, zumindest während die Funktion Drehmomentabschaltung aktiv ist, deutlich anzuheben.

ⓘ Info

Anpassung der Vorsteuerung, während Drehmomentabschaltung aktiv ist

Die Drehmomentvorsteuerung kann ebenfalls zu einem ungünstigen Verhalten der Achse während des Ablaufs führen. Es wird empfohlen diese Vorsteuerung, zumindest während die Funktion Drehmomentabschaltung aktiv ist, zu deaktivieren. Beschreiben Sie dafür den Parameter Skalierung Drehmomentvorsteuerung mit `0`.

ⓘ Info

Parameterverhalten bei Drehrichtungsumkehr

Wird eine MCX-Achse mit Drehrichtungsumkehr betrieben, so hat das keinen Einfluss auf das Vorzeichen der Konfigurationsparameter der Funktionalität **Drehmomentabschaltung**.

Eine typische Anwendung ist das Verschrauben von Flaschen, vor allem aus Kunststoff: Bei Erreichen eines bestimmten Drehmoments (Deckel ist weit genug aufgeschraubt) muss die Bewegung möglichst schnell angehalten und dabei möglichst viel der kinetischen Energie des Verschraubkopfs in elektrische Energie umgewandelt werden. Dies verhindert, dass die kinetische Energie in die Verschraubung übergeht und den Deckel weiter zudreht als gewünscht. Auf diese Art und Weise wird ein gleichmäßigeres Verschraubergebnis erreicht.

Im Weiteren wird dieser Ablauf anhand des Verschraubens eines Deckels auf einer Flasche erklärt. Der Ablauf einer Verschraubung ist im nachfolgenden Diagramm dargestellt, wobei die eigentliche Drehmomentabschaltung (*torque shut-off*) innerhalb des grauen Rahmens ausgeführt wird.

Es gibt 2 Modi, die sich auf das Verhalten am Ende des Ablaufs beziehen.

– Beim Modus `HoldTorque` wird am Ende ein definiertes Drehmoment aufgebracht, mit dem der Deckel im Stillstand gegen das Gewinde gehalten wird.
– Beim Modus `HoldPosition` wird der Deckel auf die Position, die bei der Drehmomentschwelle erreicht wurde, geregelt.

Alle Werte, die in den Befehlen verwendet werden, beziehen sich auf die Motorwelle. Ein in den Antriebsdaten aufgeführtes Getriebe hat auf die Werte in den Parametern `TorqueTrigger`, `BrakeTorque`, `SpeedTrigger` und `HoldingTorque` keinen Einfluss. Die Einheiten sind für die Drehmomente Nm und für die Drehzahlen U/min an der Motorwelle.

Dargestellt sind der qualitative Verlauf des Drehmoments (orange) und der Drehzahl (blau) der Motorwelle über der Zeit:

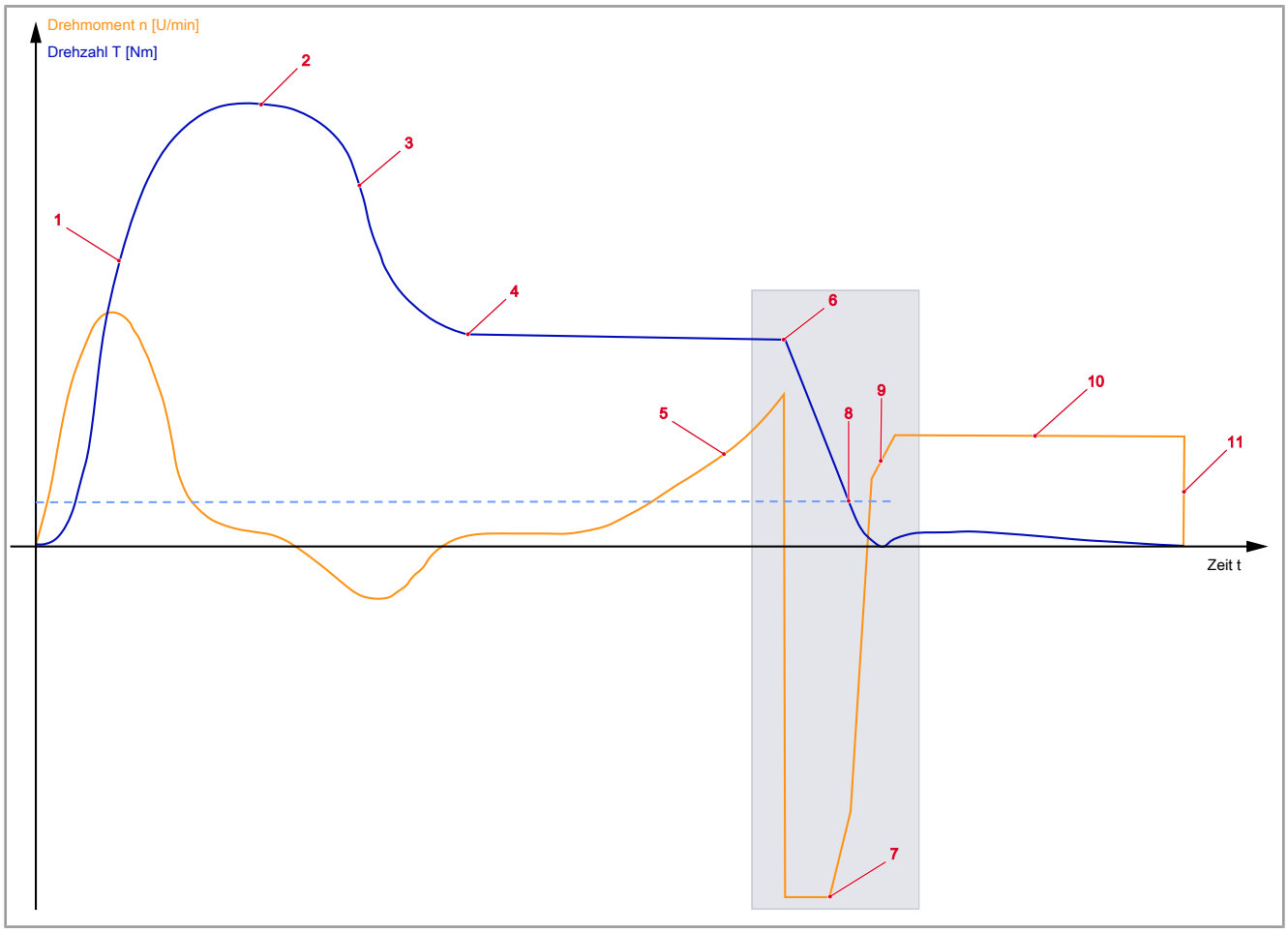


Abb. 1: MCX - TorqueShutOff

Nr.	Beschreibung
1	Beschleunigungsphase auf hohe Drehzahl. Da für diese Beschleunigung ein höheres (dynamisches) Drehmoment entstehen kann, als für die Abschaltung, wird die Drehmomentabschaltung an dieser Stelle noch nicht aktiviert.
2	Mit hoher Drehzahl wird der größte Teil des Gewindes verschraubt.
3	Abbremsen auf niedrige Drehzahl. Auf diese Weise wird die Energie, die in der trägen Masse des Verschraubkopfs und des Motors gespeichert ist, möglichst klein. Diese Energie ist direkt von der Drehzahl abhängig.
4	Niedrige Drehzahl erreicht: Die Drehmomentabschaltung wird aktiviert. Hier beginnt der eigentliche Ablauf innerhalb der MCX-Funktion.
5	Der Deckel nähert sich der Verschraubposition: Das Drehmoment steigt an.
6	Drehmomentschwelle (`TorqueTrigger`) erreicht: Die `TorqueShutOff`-Funktion begrenzt die Soll-Drehzahl auf 0. Dies veranlasst den Drehzahlregler, die Drehzahl an der eingestellten Drehmomentgrenze möglichst zu verringern. Unterstützend wird der I-Anteil des Drehzahlreglers einmalig mit einem „Bremsdrehmoment“ (`BrakeTorque`) beschrieben, um die Regelung möglichst schnell in den Abbremsvorgang zu bringen. Diese beiden Aktionen bewirken ein schnellstmögliches Abbremsen gegen die Trägheit des Verschraubkopfs.

Nr.	Beschreibung	
7	Die Soll-Ist-Differenz der Drehzahl nimmt ab: Das Drehmoment verlässt die Begrenzung.	
8	Bei Erreichen der vorgegebenen Drehzahlschwelle (`SpeedTrigger`) wird die Drehzahlbegrenzung aufgehoben, um ein Zurückdrehen zu vermeiden. Die Schritte 9 und 10 sind abhängig vom Modus:	
	Modus `HoldPosition`	**Modus** `HoldTorque`
9	Einmalig wird ein positiver Drehmomentwert (`HoldingTorque`) in den I-Anteil des Drehzahlreglers geschrieben. Dies beeinflusst die Regelung dergestalt, dass ein möglichst ruckfreier Übergang in die Positionsregelung gewährleistet ist.	Die Drehmomentbegrenzung wird auf den Wert von `HoldingTorque` gesetzt.
10	Die Sollposition wird auf die Position gesetzt, die die Welle an der Drehzahlschwelle hatte: Der Deckel wird an dieser Position in Lageregelung gehalten.	Die Drehmomentbegrenzung bleibt bis zur Deaktivierung der Drehmomentabschaltungsfunktion aktiv. Damit bewegt sich der Deckel u. U. mit einem definierten Drehmoment sehr langsam weiter, bis die finale Position erreicht ist.
11	Am Ende des Verschraubvorgangs wird der Lageregler auf die aktuelle Position gesetzt. Dies ist erforderlich, da die Funktion der Drehmomentabschaltung „unterhalb“ der MCX-Sollwertgenerierung Manipulationen an den Reglern vorgenommen hat, die bewirken, dass der Positions-Istwert nicht wie gewohnt dem Sollwert gefolgt ist. Mit dem Befehl `MCAxis.State.Transitions.UserPositionReset()` können Sie die Sollposition auf den Wert der Istposition und damit die Lageregelung in eine definierte Ausgangsposition setzen. Alternativ dazu können Sie die Achse mit dem Befehl `MCAxis.Power.Disable()` sperren. Beim erneuten Freigeben wird ebenfalls die Sollposition mit der Istposition vorbesetzt, so dass kein Ruck entsteht. Außerdem beginnen in diesem Fall alle internen Regler ohne aufgelaufene I-Summe.	
	Zuletzt wird die Funktion der Drehmomentabschaltung wieder deaktiviert.	

19.1 Syntax für MCAxis.Drive.TorqueShutOff

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Drive.TorqueShutOff.<Funktion>(optional <Parameter>);
```

19.2 Funktionen von MCAxis.Drive.TorqueShutOff

Die Funktionalität bietet folgende Funktionen:

– Configure()
– ReadConfiguration()
– Activate()
– Deactivate()

19.2.1 Configure()

Mit der Funktion übergeben Sie der Funktionalität **Drive.TorqueShutOff** die Informationen, die für Ihre Anwendung passend sind.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Mode	DriveTorqueShutOffModes	Legt den Regelmodus am Ende des Ablaufs fest.
	DriveTorqueShutOffModes.HoldPosition	Regelt am Ende des Ablaufs auf die Position, die die Achse bei der Drehzahlschwelle hatte.
	DriveTorqueShutOffModes.HoldTorque	Wendet am Ende des Ablaufs den Wert von `HoldingTorque` an. Am Beispiel eines Verschraubvorgangs: Drehmoment, mit dem die Achse gegen das Gewinde „drückt“.
TorqueTrigger	Double	Drehmomentwert, ab dem die Drehmomentabschaltung aktiv werden soll in [Nm an der Motorwelle]. Dieser Wert ist ein Betragswert und wird immer positiv verwendet.
BrakeTorque	Double	Drehmoment, mit dem beim schnellen Abbremsen der I-Anteil des Drehzahlreglers vorbelegt werden soll in [Nm an der Motorwelle]. Dieser Wert muss negativ sein, wenn das Bremsen sehr stark unterstützt werden soll.
SpeedTrigger	Double	Drehzahlschwelle, unterhalb derer der Bremsvorgang abgebrochen wird, um den Ablauf je nach Modus weiterzuführen in [U/min an der Motorwelle]. Dieser Wert ist ein Betragswert und wird immer positiv verwendet.

Parameter	Datentyp	Bedeutung
HoldingTorque	Double	– Im Modus `HoldPosition`: Drehmoment, mit dem der I-Anteil des Drehzahlreglers vorbelegt wird, um nach dem Bremsvorgang einen möglichst ruckfreien Übergang zur Positionsregelung zu erhalten. – Im Modus `HoldTorque`: Drehmoment, das die Achse nach dem Bremsvorgang weiter aufbringt. Am Beispiel eines Verschraubvorgangs: Drehmoment, mit dem Achse weiter gegen das Gewinde „drückt“ in [Nm an der Motorwelle]. Dieser Wert ist ein Betragswert und wird immer positiv verwendet.

Syntax

```
AX1.Drive.TorqueShutOff.Configure(DriveTorqueShutOffModes.HoldPosition, 1.5,
BrakeTorque, SpeedTrigger, HoldTorque);
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

19.2.2 ReadConfiguration()

Die Funktion liest die aktuell wirksame Konfiguration der Funktionalität **Drive.TorqueShutOff**.

Übergabeparameter

Die Funktion hat keine Übergabeparameter. Der Rückgabewert wird in einer Struktur vom Typ `DriveTorqueShutOffConfiguration` ausgegeben. Diese hat folgende Elemente:

Parameter	Datentyp	Bedeutung
Mode	DriveTorqueShutOffModes	Legt den Regelmodus am Ende des Ablaufs fest.
	DriveTorqueShutOffModes.HoldPosition	Regelt am Ende des Ablaufs auf die Position, die die Achse bei der Drehzahlschwelle hatte.
	DriveTorqueShutOffModes.HoldTorque	Wendet am Ende des Ablaufs den Wert von `HoldingTorque` an. Am Beispiel eines Verschraubvorgangs: Drehmoment, mit dem die Achse gegen das Gewinde „drückt“.
TorqueTrigger	Double	Drehmomentwert, ab dem die Drehmomentabschaltung aktiv werden soll in [Nm an der Motorwelle]. Dieser Wert ist ein Betragswert und wird immer positiv verwendet.
BrakeTorque	Double	Drehmoment, mit dem beim schnellen Abbremsen der I-Anteil des Drehzahlreglers vorbelegt werden soll in [Nm an der Motorwelle]. Dieser Wert muss negativ sein, wenn das Bremsen sehr stark unterstützt werden soll.

Parameter	Datentyp	Bedeutung
SpeedTrigger	Double	Drehzahlschwelle, unterhalb derer der Bremsvorgang abgebrochen wird, um den Ablauf je nach Modus weiterzuführen in [U/min an der Motorwelle]. Dieser Wert ist ein Betragswert und wird immer positiv verwendet.
HoldingTorque	Double	– Im Modus `HoldPosition`: Drehmoment, mit dem der I-Anteil des Drehzahlreglers vorbelegt wird, um nach dem Bremsvorgang einen möglichst ruckfreien Übergang zur Positionsregelung zu erhalten. – Im Modus `HoldTorque`: Wendet am Ende des Ablaufs den Wert von `HoldingTorque` an. Am Beispiel eines Verschraubvorgangs: Drehmoment, mit dem die Achse gegen das Gewinde „drückt“. Dieser Wert ist ein Betragswert und wird immer positiv verwendet.

Syntax

```
AktuelleKonfiguration := Resol_3000.Drive.TorqueShutOff.ReadConfiguration();
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

19.2.3 Activate()

Mit der Funktion aktivieren Sie die Funktionalität **Drive.TorqueShutOff**. Danach vergleicht die MCX-Achse das Drehmoment an der Motorwelle mit dem Wert, der in `TorqueTrigger` konfiguriert wurde. Sobald das Drehmoment höher ist, wird die schnelle Bremsfunktion eingeleitet.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.Drive.TorqueShutOff.Activate();
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

19.2.4 Deactivate()

Mit der Funktion deaktivieren Sie die Funktionalität **Drive.TorqueShutOff**. Es werden keine Funktionen der Funktion mehr ausgeführt. Alle Drehmoment- und Drehzahlgrenzen werden auf den ursprünglich parametrierten Wert zurückgesetzt.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.Drive.TorqueShutOff.Deactivate();
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

19.3 Eigenschaften von MCAxis.Drive.TorqueShutOff

Die Funktionalität bietet folgende Eigenschaften:

– State

19.3.1 State

Beschreibung

Datentyp	Bedeutung
DriveTorqueShutOffStates (optional)	Liest den Status, den die Funktionalität **Drive.TorqueShutOff** im Ablauf aktuell hat.
DriveTorqueShutOffStates.Disabled	Die Funktion ist nicht aktiv.
DriveTorqueShutOffStates.WaitForTorque	Die Funktion wurde aktiviert. Das Drehmoment an der Motorwelle ist noch kleiner als der Wert in `TorqueTrigger`.
DriveTorqueShutOffStates.WaitForSpeed	`TorqueTrigger` wurde überschritten. Die Bremsung wurde eingeleitet. Die Drehzahl hat den Wert in `SpeedTrigger` noch nicht unterschritten.
DriveTorqueShutOffStates.Finished	Der Ablauf ist beendet. Es wird je nach konfiguriertem Modus auf die Position oder das Drehmoment geregelt.

Syntax

```
When AX1.Drive.TorqueShutOff.State == DriveTorqueShutOffStates.Finished Continue;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

19.4 Beispieltask MCAxis.Drive.TorqueShutOff

Hier finden Sie einen Beispieltask einer Einzelachse, der den Ablauf im Fluddigaramm oben darstellt:

```
Var
    bStartDA: Bool;
    BrakeTorque, SpeedTrigger, HoldTorque: Double;
    MaxThreadLength, TightenSpeed: Double;
End_Var;

Task tTorqueShutOff Autorun

    Resol_3000.Drive.Control.Speed.TorqueFeedForwardScale := 0.0;

If Resol_3000.State.IsDisabled then
        Resol_3000.Power.Enable();
        When Resol_3000.State.IsEnabled Continue;
    End_if;

Loop
    When bStartDA Continue;
```

```
    Resol_3000.Drive.TorqueShutOff.Configure(DriveTorqueShutOffModes.HoldPo-
sition, 1.5, BrakeTorque, SpeedTrigger, HoldTorque);

    Resol_3000.MovePtp.Start(AxisPositioningModes.AbsNormal,MaxThread-
Length,TightenSpeed,10000,10000,0.1);
    When Resol_3000.Mechanism.Slope.IsAtConstantSpeed Continue;

    Resol_3000.Drive.TorqueShutOff.Activate();

    When Resol_3000.Drive.TorqueShutOff.State == DriveTorqueShutOffS-
tates.Finished Continue;

    Resol_3000.State.Transitions.UserPositionReset();
    Delay(4); //[optional] let controllers become stable
    Resol_3000.Drive.TorqueShutOff.Deactivate();

    bStartDA := False;

End_Loop;
End_Task;
```

20 MCAxis.Info

Mit der Funktionalität **MCAxis.Info** können Sie die Achs-Objekte der MotionControl identifizieren.

Es handelt sich um Informationen, die vom Anwender selbst im Hardware-Manager und Motion-Setup im Projekt festgelegt wurden.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

20.1 Syntax für MCAxis.Info

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Info.<Eigenschaft>;
```

20.2 Eigenschaften der Funktionalität

Die Funktionalität bietet folgende Eigenschaften:

– Name
– ObjectID
– ClassID

20.2.1 Name

Beschreibung

Datentyp	Bedeutung
String	Fragt den in der Hardware-Konfiguration definierten Namen der Einzelachse ab. Dies ist hilfreich, wenn Sie den Namen im Trace oder auf einer externen Anzeige verwenden wollen.

Syntax

```
AxisName := AX1.Info.Name;
```

Dabei enthält `AxisName` den String `AX1`.

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

20.2.2 ObjectID

Beschreibung

Datentyp	Bedeutung
Integer	Fragt die steuerungsinterne Nummer des Bahnverbundes ab. Die Eigenschaft wird normalerweise nur im Zusammenhang mit Anwendungen benötigt, die die Funktionen des MC-Managers nutzen. Eine Anwendungsmöglichkeit ist im zugehörigen Abschnitt MCManager (MCMgr) [▶ 413] beschrieben.

Syntax

```
Objektkennung := AX1.Info.ObjectID;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

20.2.3 ClassID

Beschreibung

Datentyp	Bedeutung
MCClassIDs	Fragt die Klasse eines MC-Objekts ab. Ein MC-Objekt kann eine Achse, ein Technologieverbund oder ein Bahnverbund sein. Die Eigenschaft wird normalerweise nur im Zusammenhang mit Anwendungen benötigt, die die Funktionen des MC-Managers nutzen. Eine Anwendungsmöglichkeit ist im zugehörigen Abschnitt MCManager (MCMgr) [▶ 413] beschreiben.

Syntax

```
KlassenID := AX1.Info.ClassID;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

20.3 Beispieltask MCAxis.Info

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

Nach Ausführung des Beispieltasks wird im Trace für alle Achsen angezeigt, ob sie im Zustand **läuft** sind oder nicht. Die Achsen werden mit ihrem Achsnamen aufgeführt.

```
function TraceRunningState(mcAxis: pointer to MCAxis)
    if mcAxis.State.IsRunning then
        trace(mcAxis.Info.Name + 'is running$n');
    else
        trace(mcAxis.Info.Name + 'is not running$n');
    end_if;
end_function;

task Sample1 // Beispieltask
var
    nAxisIndex:    int;
end_var;
    for nAxisIndex := 0 to mcMgr.AxisCount - 1 by 1 do
        TraceRunningState(mcMgr.AxisList[nAxisIndex]);
    end_for;
end_task;
```

21 MCAxis.IsInTargetWindow

Mit der Funktionalität **MCAxis.IsInTargetWindow** können Sie abfragen, ob sich die Ist-Position des Achsobjekts im definierten Zielfenster befindet.

Das Zielfenster wird dem Achsobjekt entweder im MotionSetup oder über die Funktion `MCAxis.MovePtp.Start()` übergeben. Sobald das Achsobjekt sein Zielfester erreicht hat, ist die Eigenschaft `IsInTargetWindow = true`. Startet eine neue Positionierfahrt, ist die Eigenschaft `IsInTargetWindow = false`.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

21.1 Syntax für MCAxis.IsInTargetWindow

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.IsInTargetWindow
```

21.2 Eigenschaften von MCAxis.IsInTartegWindow

`IsInTargetWindow` ist eine Eigenschaft des Achsobjekts.

21.2.1 IsInTargetWindow

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Ist-Position des Achsobjekts im Zielfenster befindet.

Syntax

```
When AX1.IsInTargetWindow Continue;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

21.3 Beispieltask MCAxis.IsInTargetWindow

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Var
    run: bool; //Hilfsflag zum Weiterschalten im Programm
End_Var;

Task Sample2 Autorun

loop
    when run Continue;

    //Einschalten des Servoverstärkers
    If AX1.State.IsDisabled AND AX1.Drive.IsReadyForOperation Then
```

```
        AX1.Power.Enable();
    End_if;
    When AX1.State.IsEnabled Continue;

    AX1.MoveHome.SetReference(0.0); //Referenzposition auf Einschaltposition
setzen.
    When AX1.MoveHome.IsReferenceSet Continue;

    //Achse absolut positionieren auf die Position 10000 mm - Maximalge-
schwindigkeit 500 mm/s -
    //Max. Beschleunigung 1000 mm/s² - Max. Verzögerung 5000 mm/s²
    //Zielfenster 0.1 mm

    AX1.MovePtp.Start(AxisPositioningModes.AbsNor-
mal,10000,500,1000,5000,0.1);
    When AX1.IsInTargetWindow Continue;

    //Wenn die Achse angekommen ist, 1 Sekunde warten.
    Delay(t#1s);

    //Achse um den gleichen Weg zurückpositionieren; hier relativ zur letz-
ten Zielposition.
    AX1.MovePtp.StartNewTargetPosition(AxisPositioningModes.RelTar-
get,-10000,);
    When AX1.IsInTargetWindow Continue;

    //Wenn Die Achse angekommen ist, 1 Sekunde warten.
    Delay(t#1s);

    //Wenn das Hilfsflag vom Anwender im Setup inzwischen gelöscht wurde,
Achse sperren und auf erneutes Setzten warten.
    //Ansonsten das Reversieren der Achse wiederholen.
    If Not run Then

        AX1.Power.Disable();
        When AX1.State.IsDisabled Continue;
    End_if;

End_loop;

End_Task;
```

22 MCAxis.Mechanism.AxisMode

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Mit der Funktionalität **MCAxis.Mechanism.AxisMode** können Sie den Modus einer MCX-Achse auslesen, der im Hardware-Manager von JetSym konfiguriert wurde.

Der Modus entscheidet darüber, welche Funktionalität die Achse bezüglich der verschiedenen MCX-Funktionen hat bzw. wie die Achse eingesetzt werden kann. Verfügbare Achsmodi:

– **InternalAxis**

  Eine interne Achse ist eine reale Achse, die an den Systembus der Steuerung angeschlossen ist und dank Motor und Servoverstärker potenziell Masse bewegen kann.

  Für eine Achse mit diesem Modus können alle Einstellungen vorgenommen werden, die für den realen Betrieb erforderlich sind.

– **VirtualAxis**

  Eine virtuelle Achse fungiert als Hilfsachse, die sich aus Sicht der Bewegungssteuerung wie eine reale Achse verhält, jedoch keinerlei reale Bewegung ausführen kann. Die Ist-Position dieser Achse ist immer gleich der Sollposition. Die Achsfunktion wird lediglich in der Steuerung gerechnet.

  Aufgrund der eingeschränkten Funktion können nur mechanische Parameter eingestellt werden.

  Die virtuelle Achse wird vorrangig als (ideale) Leitachse für Technologieverbunde verwendet.

– **ShadowAxis**

  Eine Mitlaufachse wird verwendet, um Technologieverbunde zu kaskadieren. Die Mitlaufachse wird an eine andere (interne oder virtuelle) Achse „angehängt“; d. h. sie übernimmt alle Parameter und das Verhalten ihrer sogenannten Datenachse. Vor allem die Position der Datenachse dient als Leitachsposition für weitere Technologieverbunde. Auf diese Weise kann beispielsweise die Position einer Folgeachse in einem Technologieverbund als Leitachse eines anderen Technologieverbundes verwendet werden (Kaskadierung).

  Eine Mitlaufachse muss immer als Leitachse in einem Technoverbund verwendet werden.

  Aufgrund der eingeschränkten Funktion können nur die Einheiten der mechanischen Parameter geändert werden.

– **ExternalAxis**

  Eine externe Achse ist eine MCX-Achse, die lediglich die Position eines Positionsgebers aus der Maschine einliest. Die Bewegung dieser Achse wird nicht von der MCX-Steuerung beeinflusst. Sie kann als aktuelle Position dieses Maschinenteils im STX-Programm ausgewertet oder als Leitachsposition für einen Technologieverbund verwendet werden.

  Der in der externen Achse ausgewertete Geber muss am zweiten Geber-Interface einer internen Achse angeschlossen werden. Diese interne Achse muss so parametriert werden, dass die Position dieses Encoders richtig eingelesen und der MCX-Steuerung für externe Achsen bekannt gemacht wird.

  Bei den Achsparametern der externen Achse muss ausgewählt werden, welcher freigegebene zweite Encoder für die externe Achse verwendet werden soll.

  Aufgrund der eingeschränkten Funktion können nur mechanische Parameter eingestellt werden.

– **SimulatedAxis**

  Eine simulierte Achse ist eine Achse, die sich wie eine interne Achse verhält, bei der jedoch Servoverstärker und Motor simuliert werden. Bei einer Simulationsachse findet kein Hardwarezugriff über den Systembus auf den JetMove statt.

– **SynchronousAxis**

  Eine Synchronachse ist eine Achse, die synchron zur Bahnbewegung verfahren werden kann, ohne dass sie mechanisch an der Bahnbewegung teilnimmt.

Den konfigurierten Modus können Sie nur im Hardware Manager von JetSym ändern. Damit die Änderung aktiv wird, müssen Sie die Konfiguration des Projekts neu auf die Steuerung übertragen.

22.1 Syntax für MCAxis.Mechanism.AxisMode

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Mechanism.AxisMode
```

22.2 Eigenschaften von MCAxis.Mechanism.AxisMode

`AxisMode` ist eine Eigenschaft der Funktion `Mechanism`.

22.2.1 AxisMode

Eine detaillierte Beschreibung der Achsmodi finden Sie in der *Application Note 057 MCX-Achsen*.

Beschreibung

Datentyp	Bedeutung
MCAxisModes (optional)	Konfigurierter Modus der Achse
InternalAxis VirtualAxis SchadowAxis ExternalAxis SimulatedAxis	Siehe Beschreibungen im Abschnitt MCAxis.Mechanism.AxisMode [▶ 118].

Syntax

```
AchsModus := AX1.Mechanism.AxisMode;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

23 MCAxis.Mechanism.AxisType

Mit der Funktionalität **MCAxis.Mechanism.AxisType** können Sie die mechanische Ausführung einer MCX-Achse auslesen und anpassen.

Die mechanische Ausführung einer Achse muss der Steuerung bekannt sein, damit die Einheiten und die Kinematik in der MCX richtig berechnet werden. Über diese Eigenschaft wird definiert, ob es sich um eine endlos laufende oder begrenzte Achse handelt, und ob eine lineare oder nichtlineare Mechanik durch die Achse bewegt wird.

23.1 Syntax für MCAxis.Mechanism.AxisType

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Mechanism.AxisType
```

23.2 Eigenschaften von MCAxis.Mechanism.AxisType

`AxisType` ist eine Eigenschaft der Funktion `Mechanism`.

23.2.1 AxisType

Mit der Eigenschaft lesen oder setzen Sie den mechanischen Typ der Achse.

Eine detaillierte Beschreibung der mechanischen Achstypen finden Sie im Kapitel Beschreibung der Achstypen [▶ 121].

Datentyp	Bedeutung
AxisTypes (optional)	Legt den mechanischen Typ der Achse.
Rotatory	Die Achskonfiguration ist rotatorisch/normal.
RotatoryModulo	Die Achskonfiguration ist rotatorisch/modulo.
RotatoryModuloUniPos	Die Achskonfiguration ist rotatorisch/unidirektional/modulo positiv.
RotatoryModuloUniNeg	Die Achskonfiguration ist rotatorisch/unidirektional/modulo negativ.
Linear	Die Achskonfiguration ist linear/normal.
LinearModulo	Die Achskonfiguration ist linear/modulo.
LinearModuloUniPos	Die Achskonfiguration ist linear/unidirektional/modulo positiv.
LinearModuloUniNeg	Die Achskonfiguration ist linear/unidirektional/modulo negativ.
Crank	Die Achse ist als nichtlineare Hebelachse konfiguriert.
Piston	Die Achse ist als nichtlineare Kolbenachse konfiguriert.

Syntax

```
AchsTyp := AX1.Mechanism.AxisType;
AX1.Mechanism.AxisType := AxisTypes.Crank;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

23.3 Beschreibung der Achstypen

Die mechanischen Achstypen lassen sich in 4 grundsätzliche Typen mit Untertypen einteilen:

– Rotatorisch
– Linear
– Crank
– Piston

Die Festlegung ob rotatorische oder lineare Achse erfolgt bereits im MotionSetup von JetSym (Seite „Achsparameter“).

Bei den Typen Crank und Piston handelt es sich um spezielle Mechaniken, die am Ende des Abschnitts beschrieben sind.

23.3.1 Rotatorische Achse

Bei einer rotatorischen Achse dreht sich die angeschlossene Last um eine Achse. Die Bewegungsparameter werden deshalb in rotativen Maßeinheiten (z. B. Grad, °/s usw.) angegeben.

Es kann eine Getriebeübersetzung vorhanden sein, aufgrund der die Anzahl der Motorumdrehungen nicht mit der Anzahl der mechanischen Umdrehungen übereinstimmt. Die Bewegungsparameter (Position, Geschwindigkeit usw.) sind dabei immer auf die Bewegung der Last bezogen.

Rotatorisch/normal

Dieser Achstyp wird verwendet, wenn die Bewegung der Last zwischen zwei Grenzen stattfindet. Daraus ergibt sich, dass die Last in beide Richtungen bewegt wird. Es kann ein maximaler Verfahrbereich für die Last festgelegt werden.

Rotatorisch/modulo

Bei diesem Achstyp gibt es keinen begrenzenden Verfahrbereich. Der Verfahrbereich definiert lediglich den sogenannten Modulo-Verfahrbereich.

Bewegt sich die Achse in positive Drehrichtung und erreicht die positive Verfahrgrenze, wird die Position auf den Wert der negativen Verfahrgrenze zurückgesetzt und die Achse kann weiterfahren mit neuen Positionswerten beginnend bei der negativen Verfahrgrenze.

Bewegt sich die Achse in negative Drehrichtung und erreicht die negative Verfahrgrenze, wird die Position auf den Wert der positiven Verfahrgrenze zurückgesetzt und die Achse kann weiterfahren mit neuen Positionswerten beginnend bei der positiven Verfahrgrenze.

Häufig werden diese Achsen dazu verwendet, immer in die gleiche Richtung zu fahren. Innerhalb des Modulo-Verfahrbereichs wird die Maschinenfunktion einmal ausgeführt (z. B. bei einem Rundschalttisch).

Rotatorisch/modulo/unidirektional

Dieser Achstyp ist zu verwenden, wenn eine Last nur in eine Drehrichtung (positiv oder negativ) bewegt werden darf.

Beispiel:

Bei einer unidirektional-positiven Achse wird die Last auch dann in positive Drehrichtung fahren, wenn bei einem Aufruf des Befehls `MCAxis.MovePtP` eine kleinere Zielposition als die Startposition angegeben wird. Die MCX steuert die Achse in diesem Fall über die positive Modulo-Verfahrgrenze hinaus auf die angegebene Zielposition.

23.3.2 Lineare Achse

Bei einer linearen Achse wird die angeschlossene Last geradlinig bewegt. Bei einem rotativen Motor bedarf es einer Mechanik, die die rotierende Motorachsbewegung (auch einschließlich einer Getriebeübersetzung) in eine lineare Bewegung der Last umsetzt (z. B. eine Spindel). Die Bewegungsparameter werden deshalb in linearen Maßeinheiten (z. B. mm, mm/s usw.) angegeben.

Die Bewegungsparameter (Position, Geschwindigkeit usw.) sind dabei immer auf die Bewegung der Mechanik bezogen.

Linear/normal

Dieser Achstyp wird verwendet, wenn die Bewegung der Last zwischen zwei Grenzen stattfindet. Daraus ergibt sich, dass die Last in beide Richtungen bewegt wird. Es kann ein maximaler Verfahrbereich für die Last festgelegt werden.

Linear/modulo

Bei diesem Achstyp gibt es keinen begrenzenden Verfahrbereich. Der Verfahrbereich definiert lediglich den sogenannten Modulo-Verfahrbereich.

Bewegt sich die Achse in positive Richtung und erreicht die positive Verfahrgrenze, wird die Position auf den Wert der negativen Verfahrgrenze zurückgesetzt und die Achse kann weiterfahren mit neuen Positionswerten beginnend bei der negativen Verfahrgrenze.

Bewegt sich die Achse in negative Richtung und erreicht die negative Verfahrgrenze, wird die Position auf den Wert der positiven Verfahrgrenze zurückgesetzt und die Achse kann weiterfahren mit neuen Positionswerten beginnend bei der positiven Verfahrgrenze.

Häufig werden diese Achsen dazu verwendet, immer in die gleiche Richtung zu fahren. Innerhalb des Modulo-Verfahrbereichs wird die Maschinenfunktion einmal ausgeführt (z. B. bei einem Förderband).

Linear/modulo/unidirektional

Dieser Achstyp ist zu verwenden, wenn eine Last nur in eine Richtung (positiv oder negativ) bewegt werden darf.

Beispiel:

Bei einer unidirektional-positiven Achse wird die Last auch dann in positive Richtung fahren, wenn bei einem Aufruf des Befehls `MCAxis.MovePtP` eine kleinere Zielposition als die Startposition angegeben wird. Die MCX steuert die Achse in diesem Fall über die positive Modulo-Verfahrgrenze hinaus auf die angegebene Zielposition.

23.3.3 Nichtlineare Exzenterachse

Dieser Achstyp wird verwendet, wenn eine lineare Bewegung der Last über einen (drehenden) Exzenter hervorgerufen wird.

Der Verfahrbereich entspricht dem doppelten Abstand zwischen Exzenter und Achsmittelpunkt der rotierenden Mechanik. Über die kinematische Umrechnung in der MCX kann lastbezogen in linearen Maßeinheiten programmiert werden.

Um diesen Achstyp zu verwenden, muss der MCX folgendes mitgeteilt werden:

– Definition des Achstyps
– Mechanische Konfiguration mithilfe des Befehls:

  ```
  AX1.Mechanism.Kinematics.Crank.Configure(Offset,InvertDirection,Radius);
  ```

  Die kinematische Umrechnung in der MCX beruht auf der Formel:

$$lineare\ Position = \pm(Radius * \cos(\propto) - Offset)$$

  Dabei ist:

  `Radius` der Abstand zwischen Exzenter und Achsmittelpunkt der rotierenden Mechanik.

  α der Drehwinkel der Exzenterscheibe.

  `Offset` die Verschiebung der linearen Position.

  Über den Parameter `InvertDirection` wird das Vorzeichen vor der Klammer in der Formel bestimmt.

23.3.4 Nichtlineare Kolbenachse

Dieser Achstyp wird verwendet, wenn eine lineare Bewegung der Last über einen (drehenden) Exzenter (Kurbelwelle) verbunden mit einer Kolbenstange hervorgerufen wird.

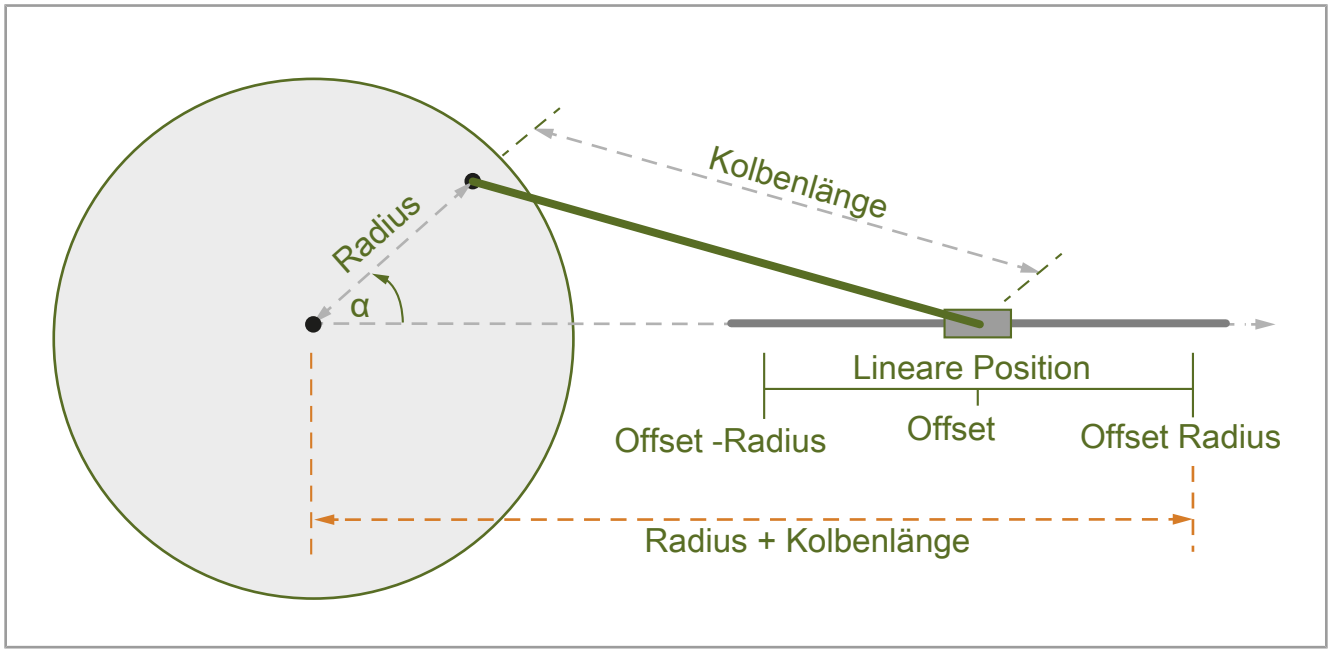


Abb. 2: Aufbau einer nichtlinearen Kolbenachse (piston)

Der Verfahrbereich entspricht dem doppelten Abstand zwischen Exzenter und Achsmittelpunkt der rotierenden Mechanik (Radius). Über die kinematische Umrechnung in der MCX kann Lastbezogen in linearen Maßeinheiten programmiert werden.

Um diesen Achstyp zu verwenden, muss der MCX folgendes mitgeteilt werden:

– Definition des Achstyps
– Mechanische Konfiguration mithilfe des Befehls:

  `AX1.Mechanism.Kinematics.Piston.Configure(Offset,InvertDirection,Radius,RodLength);`

  Die kinematische Umrechnung in der MCX beruht auf der Formel:

$$lineare\ Position = \pm\left(Radius * \cos(\propto) + \sqrt{RodLength^2 - (Radius * \sin(\propto))^2} - RodLength - Offset\right)$$

Dabei ist:

Radius der Abstand zwischen Exzenter und Achsmittelpunkt der rotierenden Mechanik.

RodLength die Länge der Kolbenstange.

α der Drehwinkel der Exzenterscheibe.

Offset die Verschiebung der linearen Position.

Über den Parameter `InvertDirection` wird das Vorzeichen vor der Klammer in der Formel bestimmt.

23.4 Beispieltask für MCAxis.Mechanism.AxisType

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Task Sample_AxisType Autorun
    …
If AX2.State.IsDisabled then;
    AX2.Mechanism.Kinematics.Piston.Configure(Offset,InvertDir,Radius,Rod-
Length);
    AX2.Mechanism.AxisType := AxisTypes.Piston;
    End_If;
    …
If AX1.Mechanism.AxisType == AxisTypes.RotatoryModulo then
        AX1.MoveVelocity.Start(Directions.Positive,Speed,Accel,Decel);
    End_If;
    …
End_Task;
```

24 MCAxis.Mechanism.GearRatioLoad, MCAxis.Mechanism.GearRatioMotor

Mit den Funktionalitäten **MCAxis.Mechanism.GearRatioLoad** und **MCAxis.Mechanism.GearRatioMotor** können Sie die Übersetzung des verwendeten Getriebes auslesen und anpassen.

ⓘ Info

Wird eine der beiden Eigenschaften für eine bereits referenzierte Achse geändert, so geht die Referenzierung verloren. Die Referenzfahrt muss wiederholt werden.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

24.1 Syntax für MCAxis.Mechanism.GearRatioLoad, MCAxis.Mechanism.GearRatioMotor

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Mechanism.GearRatioLoad
```

und

```
<Achsbezeichnung>.Mechanism.GearRatioMotor
```

24.2 Eigenschaften von MCAxis.Mechanism.GearRatioLoad, MCAxis.Mechanism.GearRatioMotor

`GearRatioLoad` und `GearRatioMotor` sind Eigenschaften der Funktion `Mechanism`.

24.2.1 GearRatioMotor

Beschreibung

Datentyp	Bedeutung
Double	Gibt die motorseitige Wellendrehzahl des Getriebes im Verhältnis zur Ausgangsdrehzahl vor. Es sind nur ganzzahlige Werte zulässig.

Beispiel

Bei einer Getriebeübersetzung von i = 3,75 ist der Wert für `GearRatioMotor` = 375 und der Wert für `GearRatioLoad` = 100 zu wählen.

Syntax

```
AX1.Mechanism.GearRatioMotor := 375;
iMotor := AX1.Mechanism.GearRatioMotor;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Getriebeübersetzungswerte gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

24.2.2 GearRatioLoad

Beschreibung

Datentyp	Bedeutung
Double	Gibt die lastseitige Wellendrehzahl des Getriebes im Verhältnis zur Eingangsdrehzahl vor. Es sind nur ganzzahlige Werte zulässig.

Beispiel

Bei einer Getriebeübersetzung von i = 3,75 ist der Wert für `GearRatioMotor` = 375 und der Wert für `GearRatioLoad` = 100 zu wählen.

Syntax

```
AX1.Mechanism.GearRatioLoad := 375;
iLast := AX1.Mechanism.GearRatioLoad;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Getriebeübersetzungswerte gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

25 MCAxis.Mechanism.Kinematics

Mit der Funktionalität **MCAxis.Mechanism.Kinematics** können Sie die mechanische Ausführung einer nichtlinearen Exzenterachse (crank [▶ 123]) oder nichtlinearen Kolbenachse (piston [▶ 124]) auslesen und anpassen.

Die mechanische Ausführung einer Achse muss der Steuerung bekannt sein, damit die Einheiten und die Kinematik in der MCX richtig berechnet werden. Die kinematischen Daten müssen vor der Definition des Achstyps der MCX bekannt gemacht werden. Nach der Kinematikdefinition müssen sie den Achstyp der Achse über den Motion-API-Befehl MCAxis.Mechanism.AxisType umschalten.

Eine genaue Beschreibung der kinematischen Funktion dieser beiden Achstypen unter dem Befehl MCAxis.Mechanism.AxisType [▶ 120].

25.1 Syntax für MCAxis.Mechanism.Kinematics

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Mechanism.Kinematics.<Funktion oder Zugriff auf Wert>(optional <Parameter 1, Parameter 2, Parameter n>);
```

25.2 Funktionen von MCAxis.Mechanism.Kinematics

Die Funktionalität bietet folgende Funktionen:

– Crank.Configure()
– Piston.Configure()

25.2.1 Crank.Configure(), Piston.Configure()

Beschreibung

Mit der Funktion können Sie alle Eigenschaften einer Exzenter- oder Kolbenachse in einem Befehl der MCX übergeben. Mithilfe dieser Eigenschaften kann die MCX die kinematische Transformation berechnen. Das ermöglicht Ihnen, die gewünschte lineare Bewegung der Last in Lineareinheiten zu programmieren, obwohl der antreibende Motor rotatorisch ist.

Auf die Eigenschaften der Funktion kann nur lesend zugegriffen werden.

Übergabeparameter

Parameter / Eigenschaften	Datentyp	Bedeutung
Offset	Double (optional)	Verschiebt die lineare Position.
InvertDirection	Modes (optional)	Definiert die Zählrichtung der linearen Position.
Radius	Double (optional)	Definiert den Abstand zwischen Achsmittelpunkt und Exzenter (bzw. Anbau der Kolbenstange).
RodLength	Double (optional)	Nur bei Piston: Definiert die Länge der Kolbenstange.

Syntax

```
AX1.Mechanism.Kinematics.Crank.Configure(15.5,Modes.On,25.0);
AX1.Mechanism.Kinematics.Piston.Configure(300.5,Modes.Off,200.0,115.3);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

25.3 Eigenschaften von MCAxis.Mechanism.Kinematics

Die Funktionalität bietet folgende Eigenschaften:

- Crank.Radius
- Crank.Offset
- Crank.InvertDirection
- Piston.Radius
- Piston.RodLength
- Piston.Offset
- Piston.InvertDirection

25.3.1 .Offset / .InvertDirection / .Radius / .RodLength

Beschreibung

Mit den Eigenschaften `Offset`, `InvertDirection`, `Radius` und `RodLength` lesen Sie die momentan gültigen Kinematikparameter der Exzenter- oder Kolbenachse.

Eigenschaft	Datentyp	Bedeutung
Offset	Double (optional)	Verschiebt die lineare Position.
InvertDirection	Modes (optional) Modes.On Modes.Off	Definiert die Zählrichtung der linearen Position.
Radius	Double (optional)	Definiert den Abstand zwischen Achsmittelpunkt und Exzenter (bzw. Anbau der Kolbenstange).
RodLength	Double (optional)	Nur bei Piston: Definiert die Länge der Kolbenstange.

Syntax

```
ExcenterVerschiebung := AX1.Mechanism.Kinematics.Crank.Offset;
KolbenAchseRadius := AX1.Mechanism.Kinematics.Piston.Radius;
KolbenStangenLaenge := AX1.Mechanism.Kinematics.Piston.Rodlength;
ExctenerRichtung := AX1.Mechanism.Kinematics.Crank.InvertDirection
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Zum Beschreiben der Eigenschaften verwenden Sie die Funktion `Configure()`.

25.4 Beispieltask MCAxis.Mechanism.Kinematics

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Task Sample_AxisType Autorun

    …

    If AX2.State.IsDisabled then
        AX2.Mechanism.Kinematics.Piston.Configure(Offset,InvertDir,Radi-
us,RodLength);
        AX2.Mechanism.AxisType := AxisTypes.Piston;
    End_If;

    …

    If AX1.Mechanism.AxisType == AxisTypes.RotatoryModulo then
        AX1.MoveVelocity.Start(Directions.Positive,Speed,Accel,Decel);
    End_If;

    …

End_Task;
```

26 MCAxis.Mechanism.Limitations

Mit der Funktionalität **MCAxis.Mechanism.Limitations** können Sie der MCX-Steuerung die mechanischen Gegebenheiten der MCX-Achse in der Maschine übergeben. Diese werden als Grenzwerte für die Bewegungsprogrammierung verwendet.

Werden die Werte für die Verfahrbereichsgrenzen durch die Programmierung einer Bewegung überschritten, so wird ein Fehler ausgegeben und die Bewegung nicht gestartet.

Bei einer Moduloachse definieren die Positionsgrenzen den Modulobereich. Überschreitet die Achse den Wert von `TravelLimitMax`, so wird die Position beim Wert von `TravelLimitMin` ohne anzuhalten weitergeführt. Analog gilt: Unterschreitet die Achse den Wert von `TravelLimitMin` so wird die Position beim Wert von `TravelLimitMax` ohne anzuhalten weitergeführt.

Beim Berechnen des Bewegungsprofils werden die Maximalwerte für die Geschwindigkeit, Beschleunigung und, im Fall von ruckbegrenzten Bewegungsarten, auch für den Ruck berücksichtigt. Ergeben sich aus dem Bewegungsbefehl höhere Werte für Geschwindigkeit, Beschleunigung und Ruck, so werden diese automatisch und ohne Fehlermeldung auf die Maximalwerte begrenzt.

26.1 Syntax für MCAxis.Mechanism.Limitations

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Mechanism.Limitations.<Funktion oder Zugriff auf Wert>(optional <Parameter 1, Parameter 2, Parameter n>);
```

26.2 Funktionen von MCAxis.Mechanism.Limitations

Die Funktionalität bietet folgende Funktionen:

– SetTravelLimits()

26.2.1 SetTravelLimits()

Mit der Funktion geben Sie den Verfahrbereich in Lastpositionseinheiten vor. Dies geschieht durch den kleinsten und größten Positionswert im Verfahrbereich bezogen auf den Referenzpunkt.

Bei einer Moduloachse definieren diese beiden Werte den Modulobereich. Bei einer Moduloachse definieren die Positionsgrenzen den Modulobereich. Überschreitet die Achse den Wert von `TravelLimitMax`, so wird die Position beim Wert von `TravelLimitMin` ohne anzuhalten. Analog gilt: Unterschreitet die Achse den Wert von `TravelLimitMin` so wird die Position beim Wert von `TravelLimitMax` ohne anzuhalten weitergeführt.

ⓘ Info

Verlust der Referenzierung durch Ausführen von SetTravelLimits

Nach dem Ausführen dieser Funktion ist die Achse nicht mehr referenziert. Eine vorher durchgeführte Referenzfahrt, muss wiederholt werden. Dies gilt nicht bei einer Achse mit Absolutwertgeber, deren Referenzierung über Referenzverschiebung (siehe SetAbsEncoderReference() [▶ 161]) durchgeführt wurde.

Zum Lesen der Werte für den Verfahrbereich/Modulobereich verwenden Sie die Eigenschaften `TravelLimitMin` und `TravelLimitMax`.

Übergabeparameter

Parameter	Datentyp	Bedeutung
TravelLimitMin	Double (optional)	Kleinster Positionswert im Verfahrbereich/Modulobereich; absoluter Wert bezogen auf den Referenzpunkt.

Parameter	Datentyp	Bedeutung
TravelLimitMax	Double (optional)	Größter Positionswert im Verfahrbereich/Modulobereich; absoluter Wert bezogen auf den Referenzpunkt.

Syntax

```
AX1.Mechanism.Limitations.SetTravelLimit(0.0, 360.0);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

26.3 Eigenschaften von MCAxis.Mechanism.Limitations

Die Funktionalität bietet folgende Eigenschaften der Funktion `Mechanism`:

- TravelLimitMin
- TravelLimitMax
- MaxSpeed
- MaxAcceleration
- MaxJerk

26.3.1 TravelLimitMin

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest oder schreibt den kleineren Wert für den Verfahr- bzw. Modulobereich (siehe Funktion SetTravelLimits() [▶ 131]).

Syntax

```
VerfahrbereichsGrenzeMin := AX1.Mechanism.Limitations.TravelLimitMin;
AX1.Mechanism.Limitations.TravelLimitMin := -100.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

26.3.2 TravelLimitMax

Beschreibung

Dentyp	Bedeutung
Double (optional)	Liest oder schreibt den größeren Wert für den Verfahr- bzw. Modulobereich (siehe Funktion SetTravelLimits() [▶ 131]).

Syntax

```
VerfahrbereichsGrenzeMax := AX1.Mechanism.Limitations.TravelLimitMax;
AX1.Mechanism.Limitations.TravelLimitMax := 1250.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

26.3.3 MaxSpeed

Beschreibung

Dentyp	Bedeutung
Double (optional)	Liest oder schreibt den Wert für die maximale Geschwindigkeit, mit der die Last bewegt werden kann oder soll. Dieser Wert kann z. B. durch die Funktion der Maschine oder auch durch die Maximaldaten einer Gewindespindel oder andere am Antriebssystem beteiligter Komponenten bestimmt sein.

Syntax

```
Maximalgeschwindigkeit := AX1.Mechanism.Limitations.MaxSpeed;
AX1.Mechanism.Limitations.MaxSpeed := 1.5; // m/s
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

26.3.4 MaxAcceleration

Beschreibung

Dentyp	Bedeutung
Double (optional)	Liest oder schreibt den Wert für die maximale Beschleunigung, mit der die Last bewegt werden kann oder soll. Dieser Wert kann z. B. durch die Funktion der Maschine oder auch durch die Maximaldaten einer Gewindespindel oder andere am Antriebssystem beteiligte Komponenten bestimmt sein. Die Überwachung wendet diesen Wert sowohl auf die Beschleunigungs- wie auch auf die Verzögerungsphase einer Bewegung an.

Syntax

```
Maximalbeschleunigung := AX1.Mechanism.Limitations.MaxAcceleration;
AX1.Mechanism.Limitations.MaxAcceleration := 15000; // °/s²
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

26.3.5 MaxJerk

Beschreibung

Dentyp	Bedeutung
Double (optional)	Liest oder schreibt den Wert für den maximalen Ruck, mit der die Last bewegt werden kann oder soll. Dieser Wert kann z. B. durch die Funktion der Maschine oder auch durch die Maximaldaten einer Gewindespindel oder andere am Antriebssystem beteiligte Komponenten bestimmt sein.

Syntax

```
Maximalruck := AX1.Mechanism.Limitations.MaxJerk;
AX1.Mechanism.Limitations.MaxJerk := 20000.0; // m/s³
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

27 MCAxis.Mechanism.LinearRotationRatio

Mit der Funktionalität **MCAxis.Mechanism.LinearRotationRatio** können Sie die Übersetzung der mechanischen Komponenten, die eine drehende Motorbewegung auf eine lineare Bewegung umsetzen (z. B. Zahnriemen, Spindel, Walzenvorschub), lesen und anpassen. Diese Funktionalität wirkt sich nur auf linear konfigurierte Achsen aus.

Die Übersetzung wird in der Weise berücksichtigt, dass diese Komponenten am Getriebeausgang wirksam werden. D. h. wenn kein Getriebe vorhanden ist, muss dieses mit der Übersetzung 1:1 parametriert werden.

27.1 Syntax für MCAxis.Mechanism.LinearRotationRatio

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Mechanism.LinearRotationRatio
```

27.2 Eigenschaften von MCAxis.Mechanism.LinearRotationRatio

Die Funktionalität bietet folgende Eigenschaften:

Die Funktionalität **LinearRotationRatio** ist eine Eigenschaft der Funktion **Mechanism**.

27.2.1 LinearRotationRatio

Beschreibung

Dentyp	Bedeutung
Double	Gibt die getriebeausgangsseitige Umsetzung der Drehbewegung in eine lineare Bewegung vor. D. h., um welche Strecke soll sich die Last bewegen, wenn die Getriebeausgangswelle eine Umdrehung vollzieht. Beispiel: Für eine Kugelrollspindel mit einer Steigung von 10 mm lautet der anzugebende Wert 10 mm/U.

Syntax

```
AX1.Mechanism.LinearRotationRatio:= 10.0;
Lin-Rot-Umsetzung := AX1.Mechanism.LinearRotationRatio;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

28 MCAxis.Mechanism.Monitoring

Mit der Funktionalität **MCAxis.Mechanism.Monitoring** können Sie den Zustand der Endschaltereingänge sowie des Referenzeingangs des Servoverstärkers der MCX-Achse abfragen.

Die Signale werden unter Berücksichtigung der eingestellten Polarität angezeigt. So führt beispielsweise ein aktiver Hardware-Endschalter bei einer Achse, die in Bewegung ist, zu einem Fehler.

Die Polarität der Signalauswertung und den Modus der Eingänge stellen Sie über die Eigenschaften der Funktionalität MCAxis.Drive.DigitalInput [▶ 73] ein.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

28.1 Syntax für MCAxis.Mechanism.Monitoring

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Mechanism.Monitoring.<Zugriff auf Wert>;
```

28.2 Eigenschaften von MCAxis.Mechanism.Monitoring

Die Funktionalität bietet folgende Eigenschaften:

– IsReferenceSwitchActive
– IsHWLimitSwitchMinActive
– IsHWLimitSwitchMaxActive

28.2.1 IsReferenceSwitchEnabled

Beschreibung

Datentyp	Bedeutung
Bool	Liest den Status des Referenzschalters, der am entsprechenden digitalen Eingang des Servoverstärkers angeschlossen ist. Ist der Status „aktiv“, so wird bei der Referenzfahrt der MCX-Achse der Referenzschalter erkannt und ggf. eine entsprechende Reaktion ausgeführt.

Syntax

```
ReferenzSchalterStatus := AX1.Mechanism.Monitoring.IsReferenceSwitchActive;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Das Beschreiben der Eigenschaft ist nicht möglich und somit keine Aktivierung der Signalfunktion per Software.

28.2.2 IsHWLimitSwitchMinActive

Beschreibung

Mit der Eigenschaft können Sie den Status des am entsprechenden digitalen Eingang des Servoverstärkers angeschlossenen Hardware-Endschalters in negative Verfahrrichtung lesen.

Ist der Status *aktiv*, so wird ein Fehler ausgegeben und die Achse angehalten und gesperrt. Die Positionierung der Achse ist nur in entgegengesetzter Richtung möglich.

Datentyp	Bedeutung
Bool	Wertet den Hardware-Endschalter in negative Verfahrrichtung als *inaktiv* oder *aktiv* aus.

Syntax

```
NegEndschalterStatus := AX1.Mechanism.Monitoring.IsHWLimitSwitchMinActive;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

28.2.3 IsHWLimitSwitchMaxActive

Beschreibung

Mit der Eigenschaft können Sie den Status des am entsprechenden digitalen Eingang des Servoverstärkers angeschlossenen Hardware-Endschalters in positive Verfahrrichtung lesen.

Ist der Status *aktiv*, so wird ein Fehler ausgegeben und die Achse angehalten und gesperrt. Die Positionierung der Achse ist nur in entgegengesetzter Richtung möglich.

Datentyp	Bedeutung
Bool	Wertet den Hardware-Endschalter in positive Verfahrrichtung als *inaktiv* oder *aktiv* aus.

Syntax

```
PosEndschalterStatus := AX1.Mechanism.Monitoring.IsHWLimitSwitchMaxActive;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

29 MCAxis.Mechanism.Slope

Mit der Funktionalität **MCAxis.Mechanism.Slope** können Sie die berechneten Eigenschaften der laufenden oder nächsten Bewegung lesen oder anpassen (Config.RampBreakMode [▶ 146]).

Eine Positionierbewegung besteht aus folgenden Phasen:

1. Beschleunigungsrampe
2. Fahrt mit konstanter Geschwindigkeit

   Diese Phase tritt nur ein, wenn die Geschwindigkeit `Speed` mit den vorgegebenen MovePtp-Parametern erreicht werden kann. Andernfalls geht die Beschleunigungsrampe direkt in die Verzögerungsrampe über (Dreiecksbetrieb).
3. Verzögerungsrampe

Daneben kann auch keine Bewegung stattfinden, wenn die Achse im Ziel steht und noch keine weitere Positionierung gestartet wurde.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

29.1 Syntax für MCAxis.Mechanism.Slope

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Mechanism.Slope.<Zugriff auf Wert>
```

29.2 Eigenschaften von MCAxis.Mechanism.Slope

Die Funktionalität bietet folgende Eigenschaften:

- SlopeState
- IsAccelerating
- IsAtConstantSpeed
- IsDecelerating
- IsStopped
- SlopeType
- ActualMaxSpeed
- ActualMaxAcceleration
- ActualMaxDeceleration
- SpeedSetpoint
- AccelerationSetpoint
- Speed
- Acceleration
- Deceleration
- Config.RampBreakMode

29.2.1 SlopeState

Beschreibung

Datentyp	Bedeutung
AxisSlopeStates	Liest, in welcher Phase der Bewegung sich der berechnete Sollwert der MCX-Achse befindet. Damit können Sie beispielsweise eine Aktion in der Maschine erst dann aktivieren, wenn die MCX-Achse die Beschleunigungsphase verlassen hat.
AxisSlopeStates.Accelerating	Achse befindet sich in der Beschleunigungsrampe.
AxisSlopeStates.AtConstandSpeed	Achse fährt mit konstanter Geschwindigkeit.
AxisSlopeStates.Decelerating	Achse befindet sich in der Verzögerungsrampe.
AxisSlopeStates.Stopped	Achse steht still.

Syntax

```
Rampenstatus := AX1.Mechanism.Slope.SlopeState;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

29.2.2 IsAccerlerating

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der berechnete Sollwert der MCX-Achse in der Beschleunigungsrampe befindet.

Syntax

```
If AX1.Mechanism.Slope.IsAccelerating then …
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

29.2.3 IsAtConstantSpeed

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der berechnete Sollwert der MCX-Achse in der Phase Fahrt mit konstanter Geschwindigkeit befindet.

Syntax

```
If AX1.Mechanism.Slope.IsAtConstantSpeed then …
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

29.2.4 IsDecelerating

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der berechnete Sollwert der MCX-Achse in der Verzögerungsrampe befindet.

Syntax

```
If AX1.Mechanism.Slope.IsDecelerating then …
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

29.2.5 IsStopped

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der berechnete Sollwert der MCX-Achse in der Phase des Stillstands befindet.

Syntax

```
If AX1.Mechanism.Slope.IsStopped then …
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

29.2.6 SlopeType

Beschreibung

Mit der Eigenschaft `SlopeType` können Sie die Rampenform in der Beschleunigungs- und der Verzögerungsphase lesen oder anpassen. **HINWEIS! Diese Einstellung gilt für alle Positionierbewegungen der MCX-Achse so lange, bis der Wert geändert wird.**

Datentyp	Bedeutung
AxisSlopeTypes	Form der Beschleunigungs- und Verzögerungsrampe.
AxisSlopeTypes.Sine	Diese Einstellung berechnet eine Bewegung, die verhältnismäßig wenig Ruck erzeugt, d. h. die Bewegung ist weicher. Da der dem Bewegungsbefehl übergebene Beschleunigungswert nur in der Mitte der Rampe erreicht wird, sind Rampendauer und -länge höher als bei der Einstellung `linear`. Die Bewegung verläuft in der Beschleunigungsphase sinus-förmig und in der Verzögerungsphase sin²-förmig. Die Bewegung wird ruckbegrenzt.

Datentyp	Bedeutung
AxisSlopeTypes.Linear	Diese Einstellung ermöglicht eine Bewegung mit möglichst geringer maximaler Beschleunigung bei gleicher Rampenzeit (und -länge) im Verhältnis zur Einstellung `sine`. Dafür ist der Ruck beim Übergang vom Stillstand in die Beschleunigungsphase und von der Beschleunigungsphase zur Phase mit max. Geschwindigkeit unendlich. Die Bewegung verläuft in der Beschleunigungsphase konstant und in der Verzögerungsphase sinus-förmig. Die Bewegung verläuft ohne Ruckbegrenzung.

Syntax

```
Rampentyp := AX1.Mechanism.Slope.SlopeType;
AX1.Mechanism.Slope.SlopeType := AxisSlopeTypes.Linear;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

29.2.7 ActualMaxSpeed

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert der maximalen Geschwindigkeit, der im Sollwertgenerator der MCX-Achse momentan für die laufende oder kommende Bewegung programmiert ist. Dieser Wert kann durch Vorgabe eines neuen Geschwindigkeitswertes in den Befehlen `MCAxis.MoveVelocity.Start()`, `MCAxis.MoveVelocity.NewSpeed()`, `MCAxis.MovePtp.Start()`, `MCAxis.MovePtp.NewSpeed()` geändert werden. Wird der Parameter in diesen Befehlen nicht angegeben, so wird in der Bewegung der mit `ActualMaxSpeed` lesbare Wert verwendet.

Syntax

```
MaxGeschwindigkeit := AX1.Mechanism.Slope.ActualMaxSpeed;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

29.2.8 ActualMaxAcceleration

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert der maximalen Beschleunigung, der dem Sollwertgenerator der MCX-Achse zuletzt übergeben wurde. Dieser Wert kann durch Vorgabe eines neuen Geschwindigkeitswertes in den Befehlen `MCAxis.MoveVelocity.Start()`, `MCAxis.MoveVelocity.NewAcceleration()`, `MCAxis.MovePtp.Start()`, `MCAxis.MovePtp.NewAcceleration()` geändert werden. Wird der Parameter in diesen Befehlen nicht angegeben, so wird in der Bewegung der mit `ActualMaxAcceleration` lesbare Wert verwendet.

Syntax

```
MaxBeschleunigung := AX1.Mechanism.Slope.ActualMaxAcceleration;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

29.2.9 ActualMaxDeceleration

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert der maximalen Beschleunigung, der dem Sollwertgenerator der MCX-Achse zuletzt übergeben wurde. Dieser Wert kann durch Vorgabe eines neuen Geschwindigkeitswertes in den Befehlen `MCAxis.MoveVelocity.Start()`, `MCAxis.MoveVelocity.NewDeceleration()`, `MCAxis.MovePtp.Start()`, `MCAxis.MovePtp.NewDeceleration()` geändert werden. Wird der Parameter in diesen Befehlen nicht angegeben, so wird in der Bewegung der mit `ActualMaxDeceleration` lesbare Wert verwendet.

Syntax

```
MaxVerzögerung := AX1.Mechanism.Slope.ActualMaxDeceleration;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

29.2.10 AccelerationSetpoint

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert der Sollbeschleunigung, der im Sollwertgenerator der MCX-Achse momentan wirksam ist. Das Vorzeichen des Werts zeigt eine Beschleunigung oder Verzögerung an.

Syntax

```
ActBeschlSollwert := AX1.Mechanism.Slope.AccelerationSetpoint;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

29.2.11 SpeedSetpoint

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert der Soll-Geschwindigkeit, der im Sollwertgenerator der MCX-Achse momentan wirksam ist. Das Vorzeichen des Werts ist abhängig von der Fahrtrichtung.

Syntax

```
ActGeschwSollwert := AX1.Mechanism.Slope.SpeedSetpoint;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

29.2.12 Speed

Beschreibung

Datentyp	Bedeutung
Double	Liest oder schreibt den Wert für die maximale Geschwindigkeit, der für die Positionssollwertgenerierung eines Positionierprofils verwendet wird. Dieser Wert wird normalerweise in den Funktionen `MovePtp.Start()` und `MoveVelocity.Start()` oder auch `MovePtp.NewSpeed()` und `MoveVelocity.NewSpeed()` als Parameter übergeben, wo eine Änderung auf eine laufende Bewegung angewendet wird. Anders hier: Gleich in welchem Zustand die Änderung des Wertes für `Speed` übergeben wird, der neue Wert wirkt sich erst in der nächsten Bewegung aus.

Syntax

```
ProfilGeschw := AX1.Mechanism.Slope.Speed;
AX1.Mechanism.Slope.Speed := 3000.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

29.2.13 Acceleration

Beschreibung

Datentyp	Bedeutung
Double	Liest oder schreibt den Wert für die maximale Beschleunigung, der für die Positionssollwertgenerierung eines Positionierprofils verwendet wird. Dieser Wert wird normalerweise in den Funktionen `MovePtp.Start()` und `MoveVelocity.Start()` oder auch `MovePtp.NewAcceleration()` und `MoveVelocity.NewAcceleration()` als Parameter übergeben. Gleich in welchem Zustand die Änderung des Wertes für `Acceleration` übergeben wird, der neue Wert wirkt sich erst in der nächsten Bewegung aus.

Syntax

```
ProfilBeschleunigung := AX1.Mechanism.Slope.Acceleration;
AX1.Mechanism.Slope.Acceleration := 5500.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

29.2.14 Deceleration

Beschreibung

Datentyp	Bedeutung
Double	Liest oder schreibt den Wert für die maximale Verzögerung, der für die Positionssollwertgenerierung eines Positionierprofils verwendet wird. Dieser Wert wird normalerweise in den Funktionen `MovePtp.Start()` und `Move.Velocity.Start()` oder auch `MovePtp.NewDeceleration()` und `MoveVelocity.NewDeceleration()` als Parameter übergeben. Gleich in welchem Zustand die Änderung des Wertes für `Deceleration` übergeben wird, der neue Wert wirkt sich erst in der nächsten Bewegung aus.

Syntax

```
ProfilVerzoegerung := AX1.Mechanism.Slope.Deceleration;
AX1.Mechanism.Slope.Acceleration := 12000.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

29.2.15 Config.RampBreakMode

Beschreibung

Verzögerungsrampe mit MovePtp.NewDeceleration()

Wird die Verzögerung nicht mit dem `AX1.MovePtp.Start()`-Befehl übergeben, sondern mit `AX1.MovePtp.NewDeceleration()`, so wird Positionierung immer entsprechend der Rampenkonfiguration `immediate` ausgeführt.

Datentyp	Bedeutung
MCAxisRampBreakModes	Liest oder schreibt das Verhalten der Sollwertgenerierung wenn ein neuer `MovePtp.Start()`-Befehl gegeben wird, während sich die Bewegung in der Verzögerungsrampe befindet. Der Wert nach dem Einschalten ist `None`. Es handelt sich um eine statische Konfiguration, die solange aktiv bleibt, bis sie geändert wird.
MCAxisRampBreakModes.Immediate	1. Der Verfahrweg bis zum Stillstand ist ausreichend: Die Verzögerungsrampe wird mit der neu übergebenen maximalen Verzögerung abgebrochen und die zweite Zielposition wird direkt angefahren. 2. Der Verfahrweg bis zum Stillstand ist nicht ausreichend: Die erste Zielposition wird angefahren. Die Positionierung auf die zweite Zielposition wird direkt angeschlossen.
MCAxisRampBreakModes.None	Die erste Zielposition wird angefahren. Die Positionierung auf die zweite Zielposition wird direkt angeschlossen.

Syntax

```
RampenKonfiguration := AX1.Mechanism.Slope.Config.RampBreakMode;
AX1.Mechanism.Slope.Config.RampBreakMode := MCAxisRampBreakModes.Immediate;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

30 MCAxis.Mechanism.Units

Mit der Funktionalität **MCAxis.Mechanism.Units** können Sie die verwendeten Einheiten für die Werte von Positionen, Geschwindigkeiten, Beschleunigungen und Ruck lesen oder anpassen.

Diese Einheiten werden üblicherweise bei der Achsdefinition festgelegt und können mithilfe dieser Befehle im STX-Programm abgefragt werden.

HINWEIS



Veränderte Positionen durch Wechsel der Einheit im Maschinenbetrieb

Es ist möglich, die Einheiten während des Betriebs der Maschine zu ändern. Da jedoch der Positionierbefehl nur die Werte ohne die Angabe einer Einheit enthält, verändert dadurch sich die Bedeutung der Werte.

Beispiel: Die Umstellung von Millimeter auf Zoll bewirkt bei einem Wert von `25.0` im Positionierbefehl eine andere Position in der Maschine.

- ► Vermeiden Sie die Umstellung der Einheiten während des Betriebs.
- ► Nehmen Sie eine Umstellung der Einheiten mit Bedacht vor.

30.1 Syntax für MCAxis.Mechanism.Units

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Mechanism.Units.<Funktion oder Zugriff auf Wert>(optional
<Parameter 1, Parameter 2, Parameter n>);
```

30.2 Funktionen von MCAxis.Mechanism.Units

Die Funktionalität bietet folgende Funktionen:

– SetLinearUnits()
– SetRotatoryUnits()

30.2.1 SetLinearUnits()

Mit der Funktion verändern sie alle Einheiten für Positionen, Geschwindigkeiten, Beschleunigungen und Rucke in einer Funktion für eine lineare Bewegung. Über die Parameter können Sie für jede Einheit unterschiedliche Bedeutungen vorgeben.

Übergabeparameter

Parameter	Datentyp	Bedeutung
PositionUnit	**MCLinearPositionUnits** (optional) – MCLinearPositionUnits.MM – MCLinearPositionUnits.INCH – MCLinearPositionUnits.MM_10TH – MCLinearPositionUnits.MM_100TH	Einheit für die linearen Positionswerte. Mögliche Einheiten: – Millimeter – Zoll – 1/10 Millimeter – 1/100 Millimeter

Parameter	Datentyp	Bedeutung
SpeedUnit	**MCLinearSpeedUnits** (optional) – MCLinearSpeedUnits.MM_PER_SEC – MCLinearSpeedUnits.MM_PER_MIN – MCLinearSpeedUnits.M_PER_SEC – MCLinearSpeedUnits.M_PER_MIN – MCLinearSpeedUnits.INCH_PER_SEC – MCLinearSpeedUnits.INCH_PER_MIN	Einheit für die linearen Geschwindigkeitswerte. Mögliche Einheiten: – Millimeter pro Sekunde – Millimeter pro Minute – Meter pro Sekunde – Meter pro Minute – Zoll pro Sekunde – Zoll pro Minute
AccelerationUnit	**MCLinearAccelerationUnits** (optional) – MCLinearAccelerationUnits.MM_PER_S2 – MCLinearAccelerationUnits.M_PER_S2 – MCLinearAccelerationUnits.INCH_PER_S2	Einheit für die linearen Beschleunigungswerte. Mögliche Einheiten: – Millimeter pro Sekunde² – Meter pro Sekunde² – Zoll pro Sekunde²
JerkUnit	**MCLinearJerkUnits** (optional) – MCLinearJerkUnits.MM_PER_S3 – MCLinearJerkUnits.M_PER_S3 – MCLinearJerkUnits.INCH_PER_S3	Einheit für die linearen Ruckwerte. Mögliche Einheiten: – Millimeter pro Sekunde³ – Meter pro Sekunde³ – Zoll pro Sekunde³

Syntax

```
AX1.Mechanism.Units.SetLinearUnits(
    MCLinearPositionUnits.MM,
    MCLinearSpeedUnits.MM_PER_SEC,
    MCLinearAccelerationUnits.MM_PER_S2,
    MCLinearJerkUnits.MM_PER_S3);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

Das Schreiben der Funktion ist jedem Zustand möglich. Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

30.2.2 SetRotaryUnits()

Mit der Funktion verändern sie alle Einheiten für Positionen, Geschwindigkeiten, Beschleunigungen und Rucke in einer Funktion für eine rotierende Bewegung. Über die Parameter können Sie für jede Einheit unterschiedliche Bedeutungen vorgeben.

Übergabeparameter

Parameter	Datentyp	Bedeutung
PositionUnit	**MCRotatoryPositionUnits** (optional) – MCRotatoryPositionUnits.DEG – MCRotatoryPositionUnits.RAD – MCRotatoryPositionUnits.REV	Einheit für die rotativen Positionswerte. Mögliche Einheiten: – Winkelgrad – Radiant (2 Pi) – Umdrehungen

Parameter	Datentyp	Bedeutung
SpeedUnit	**MCRotatorySpeedUnits** (optional) – MCRotatorySpeedUnits.DEG_PER_SEC – MCRotatorySpeedUnits.RAD_PER_SEC – MCRotatorySpeedUnits.REV_PER_MIN	Einheit für die rotativen Geschwindigkeitswerte. Mögliche Einheiten: – Winkelgrad pro Sekunde – Radiant (2 Pi) pro Sekunde – Umdrehungen pro Minute
AccelerationUnit	**MCRotatoryAccelerationUnits** (optional) – MCRotatoryAccelerationUnits.DEG_PER_S2 – MCRotatoryAccelerationUnits.RAD_PER_S2	Einheit für die rotativen Beschleunigungswerte. Mögliche Einheiten: – Winkelgrad pro Sekunde² – Radiant (2 Pi) pro Sekunde²
JerkUnit	**MCRotatoryJerkUnits** (optional) – MCRotatoryJerkUnits.DEG_PER_S3 – MCRotatoryJerkUnits.RAD_PER_S3	Einheit für die rotativen Ruckwerte. Mögliche Einheiten: – Winkelgrad pro Sekunde³ – Radiant (2 Pi) pro Sekunde³

Syntax

```
AX1.Mechanism.Units.SetRotatoryUnits(
    MCRotatoryPositionUnits.DEG,
    MCRotatorySpeedUnits.REV_PER_MIN,
    MCRotatoryAccelerationUnits.DEG_PER_S2,
    MCRotatoryJerkUnits.DEG_PER_S3);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

Das Schreiben der Funktion ist jedem Zustand möglich. Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

30.3 Eigenschaften von MCAxis.Mechanism.Units

Die Funktionalität bietet folgende Eigenschaften:

– LinearPositionUnit
– LinearSpeedUnit
– LinearAccelerationUnit
– LinearJerkUnit
– RotatoryPositionUnit
– RotatorySpeedUnit
– RotatoryAccelerationUnit
– RotatoryJerkUnit

30.3.1 LinearPositionUnit

Beschreibung

Liest oder schreibt die Einheit, in der die Positionswerte für eine linear definierte Achse verarbeitet und angezeigt werden.

Datentyp	Bedeutung
MCLinearPositionUnits (optional) – MCLinearPositionUnits.MM – MCLinearPositionUnits.INCH – MCLinearPositionUnits.MM_10TH – MCLinearPositionUnits.MM_100TH	Einheit für die linearen Positionswerte. Mögliche Einheiten: – Millimeter – Zoll – 1/10 Millimeter – 1/100 Millimeter

Syntax

```
PositionsEinheiten := AX1.Mechanism.Units.LinearPositionUnit;
AX1.Mechanism.Units.LinearPositionUnit := MCLinearPositionUnits.MM_10TH;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

30.3.2 LinearSpeedUnit

Beschreibung

Liest oder schreibt die Einheit, in der die Geschwindigkeitswerte für eine linear definierte Achse verarbeitet und angezeigt werden.

Datentyp	Bedeutung
MCLinearSpeedUnits (optional) – MCLinearSpeedUnits.MM_PER_SEC – MCLinearSpeedUnits.MM_PER_MIN – MCLinearSpeedUnits.M_PER_SEC – MCLinearSpeedUnits.M_PER_MIN – MCLinearSpeedUnits.INCH_PER_SEC – MCLinearSpeedUnits.INCH_PER_MIN	Einheit für die linearen Geschwindigkeitswerte. Mögliche Einheiten: – Millimeter pro Sekunde – Millimeter pro Minute – Meter pro Sekunde – Meter pro Minute – Zoll pro Sekunde – Zoll pro Minute

Syntax

```
GewchwEinheiten := AX1.Mechanism.Units.LinearSpeedUnit;
AX1.Mechanism.Units.LinearSpeedUnit := MCLinearSpeedUnits.M_SEC;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

30.3.3 LinearAccelerationUnit

Beschreibung

Liest oder schreibt die Einheit, in der die Beschleunigungswerte für eine linear definierte Achse verarbeitet und angezeigt werden.

Datentyp	Bedeutung
MCLinearAccelerationUnits (optional) – MCLinearAccelerationUnits.MM_PER_S2 – MCLinearAccelerationUnits.M_PER_S2 – MCLinearAccelerationUnits.INCH_PER_S2	Einheit für die linearen Geschwindigkeitswerte. Mögliche Einheiten: – Millimeter pro Sekunde² – Meter pro Sekunde² – Zoll pro Sekunde²

Syntax

```
BeschleunigungsEinheiten := AX1.Mechanism.Units.LinearAccelerationUnit;
AX1.Mechanism.Units.LinearAccelerationUnit := MCLinearAccelerationU-
nits.M_PER_S2;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

30.3.4 LinearJerkUnit

Beschreibung

Liest oder schreibt die Einheit, in der die Ruckwerte für eine linear definierte Achse verarbeitet und angezeigt werden.

Datentyp	Bedeutung
MCLinearJerkUnits (optional) – MCLinearJerkUnits.MM_PER_S3 – MCLinearJerkUnits.M_PER_S3 – MCLinearJerkUnits.INCH_PER_S3	Einheit für die linearen Geschwindigkeitswerte. Mögliche Einheiten: – Millimeter pro Sekunde³ – Meter pro Sekunde³ – Zoll pro Sekunde³

Syntax

```
RuckEinheiten := AX1.Mechanism.Units.LinearJerkUnit;
AX1.Mechanism.Units.LinearJerkUnit := MCLinearJerkUnits.MM_PER_S3;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

30.3.5 RotaryPositionUnit

Beschreibung

Liest oder schreibt die Einheit, in der die Positionswerte für eine rotativ definierte Achse verarbeitet und angezeigt werden.

Datentyp	Bedeutung
MCRotatoryPositionUnits (optional) – MCRotatoryPositionUnits.MM – MCRotatoryPositionUnits.INCH – MCRotatoryPositionUnits.MM_10TH – MCRotatoryPositionUnits.MM_100TH	Einheit für die rotativen Positionswerte. Mögliche Einheiten: – Millimeter – Zoll – 1/10 Millimeter – 1/100 Millimeter

Syntax

```
PositionsEinheiten := AX1.Mechanism.Units.RotatoryPositionUnit;
AX1.Mechanism.Units.RotatoryPositionUnit := MCRotatoryPositionUnits.RAD;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

30.3.6 RotatorySpeedUnit

Beschreibung

Liest oder schreibt die Einheit, in der die Drehzahlwerte für eine rotativ definierte Achse verarbeitet und angezeigt werden.

Datentyp	Bedeutung
MCRotatorySpeedUnits (optional) – MCRotatorySpeedUnits.DEG_PER_SEC – MCRotatorySpeedUnits.RAD_PER_SEC – MCLinearSpeedUnits.REV_PER_MIN	Einheit für die linearen Geschwindigkeitswerte. Mögliche Einheiten: – Winkelgrad pro Sekunde – Radiant (2 Pi) pro Sekunde – Umdrehungen pro Minute

Syntax

```
DrehzahlEinheiten := AX1.Mechanism.Units.RotatorySpeedUnit;
AX1.Mechanism.Units.RotatorySpeedUnit := MCRotatorySpeedUnits.REV_PER_MIN;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

30.3.7 RotatoryAccelerationUnit

Beschreibung

Liest oder schreibt die Einheit, in der die Drehbeschleunigungswerte für eine rotativ definierte Achse verarbeitet und angezeigt werden.

Datentyp	Bedeutung
MCRotatoryAccelerationUnits (optional) – MCRotatoryAccelerationUnits.DEG_PER_S2 – MCRotatoryAccelerationUnits.RAD_PER_S2	Einheit für die rotativen Beschleunigungswerte. Mögliche Einheiten: – Winkelgrad pro Sekunde² – Radiant (2 Pi) pro Sekunde²

Syntax

```
BeschleunigungsEinheiten := AX1.Mechanism.Units.LinearAccelerationUnit;
AX1.Mechanism.Units.LinearAccelerationUnit := MCLinearAccelerationU-
nits.M_PER_S2;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

30.3.8 RotatoryJerkUnit

Beschreibung

Liest oder schreibt die Einheit, in der die Ruckwerte für eine rotativ definierte Achse verarbeitet und angezeigt werden.

Datentyp	Bedeutung
MCRotatoryJerkUnits (optional) – MCRotatoryAccelerationU-nits.DEG_PER_S3 – MCRotatoryAccelerationU-nits.RAD_PER_S3	Einheit für die rotativen Beschleunigungswerte. Mögliche Einheiten: – Winkelgrad pro Sekunde³ – Radiant (2 Pi) pro Sekunde³

Syntax

```
RuckEinheiten := AX1.Mechanism.Units.RotatoryJerkUnit;
AX1.Mechanism.Units.RotatoryJerkUnit := MCRotatoryJerkUnits.MM_PER_S3;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

31 MCAxis.MoveHalt

Mit der Funktionalität **MCAxis.MoveHalt** können Sie eine MCX-Achse, die sich in Bewegung befindet, mit einer Rampe anhalten. Im Unterschied zur `Stop()`-Funktion werden die Vorlaufpuffer nicht gelöscht. Im gepufferten Betrieb wird diese Funktion mit in den Puffer geschrieben und dort verarbeitet.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

31.1 Syntax für MCAxis.MoveHalt

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.MoveHalt.Start.(optional <Parameter 1, Parameter 2>);
```

31.2 Funktionen von MCAxis.MoveHalt

Die Funktionalität bietet folgende Funktionen:

– Start()

31.2.1 Start()

Die Funktion führt den Anhaltebefehl unmittelbar aus. Die übergebenen Parameter definieren, auf welche Weise die Achse anhalten soll.

Beachten Sie folgende Regeln:

– Der Parameter `Deceleration` wird in den Anwender-Einheiten eingegeben, die für die Achse vordefiniert sind.

– Eine Anpassung der maximalen Verzögerung über diesen Parameter ändert den Wert im Achsobjekt. Wird beispielsweise beim nächsten `MovePtp.Start`-Befehl kein Wert für den Parameter `Deceleration` übergeben, so wird der hier vorgegebene Wert verwendet.

– Die Parameter beziehen sich auf das Achsobjekt; d. h. sie bilden eine Eigenschaft der Achse. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Übergabeparameter

Parameter	Datentyp	Bedeutung
MCAxisHaltMode	Enum MCAxisHaltModes (optional)	Verhalten der Achse, nachdem sie mit einer Verzögerungsrampe abgebremst wurde:
	MCAxisHaltModes.Normal	Nach Abschluss der Anhalterampe kommt die Achse in Lageregelung zum Stillstand.
	MCAxisHaltModes.AtActualPosition	Nach Abschluss der Anhalterampe fährt die Achse auf die Position zurück, an der sie sich befand, als die Funktion `MoveHalt.Start()` aufgerufen wurde. Die Rückfahrbewegung wird mit den für dieses Achsobjekt eingestellten Parametern ausgeführt.

Parameter	Datentyp	Bedeutung
Deceleration	Double (optional)	Definiert die maximale Verzögerung, die in der Anhalterampe erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.

Syntax

```
AX1.MoveHalt.Start(MCAxisHaltModes.Normal,50000);
AX1.MoveHalt.Start(MCAxisHaltModes.AtActualPosition,50000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	–		Gesperrt
Freigegeben	–		Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	X		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Positionierung stehen Ihnen verschiedene Informationen zur Verfügung:

– Der **Zustand der Achse** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder die Achse angehalten wurde (MCAxis.State [▶ 208]).

```
When Ax1.State.IsEnabled Continue; //die Achse befindet sich in Lageregelung
und steht.
When Ax1.State.IsRunning Continue; //der Sollwertgenerator erzeugt eine Be-
wegung.
```

– Der **Rampenstatus** informiert darüber, in welchem Bewegungsabschnitt sich der Sollwertgenerator aktuell befindet (MCAxis.Mechanism.Slope [▶ 140]).

```
When Ax1.Mechanism.Slope.IsAccelerating Continue;
When Ax1.Mechanism.Slope.IsAtConstantSpeed Continue;
When Ax1.Mechanism.Slope.IsDecelerating Continue;
When Ax1.Mechanism.Slope.IsStopped Continue;
```

31.3 Beispieltask MCAxis.MoveHalt

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Var
    run: bool; //Hilfsflag zum Weiterschalten im Programm
End_Var;

Task Sample2 Autorun

loop
    when run Continue;

    //Einschalten des Servoverstärkers
    If AX1.State.IsDisabled AND AX1.Drive.IsReadyForOperation Then
        AX1.Power.Enable();
    End_if;
    When AX1.State.IsEnabled Continue;

    AX1.MoveHome.SetReference(0.0); //Referenzposition auf Einschaltposition
setzen.
    When AX1.MoveHome.IsReferenceSet Continue;

    //Achse in positive Richtung im Endlosbetrieb starten.
    AX1.MoveVelocity.Start(Directions.Positive,Speed1,5000,);
    //Wenn das Hilfsflag vom Anwender im Setup inzwischen gelöscht wurde,
Achse mit einer Rampe von 500 mm/s² anhalten und sperren.
    //Dieses Ereignis könnte z. B. ein Tipp-Taster sein.
    When Not run Continue;
    AX1.MoveHalt.Start(MCAxisHaltModes.Normal,500);
    When AX1.State.IsDisabled Continue;
    AX1.Power.Disable();
    When AX1.State.IsDisabled Continue;

End_loop;

End_Task;
```

32 MCAxis.MoveHome

Mit der Funktionalität **MCAxis.MoveHome** können Sie eine eine Referenzfahrt einer MCX-Achse starten und/oder parametrieren.

Die JetSym-Onlinehilfe beschreibt auf der Seite *MotionHome-Ablaufdarstellung* detailliert die unterschiedlichen Arten von Referenzfahrten sowie deren Funktion und Ablauf.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

32.1 Syntax für MCAxis.MoveHome

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.MoveHalt.Home.<Funktion oder Zugriff auf Wert> (optional
<Parameter 1, Parameter 2, Parameter n>);
```

32.2 Funktionen von MCAxis.MoveHome

Die Funktionalität bietet folgende Funktionen:

– Start()
– SetReference()

Funktionen für die Referenzverschiebung bei der Verwendung von Absolutwertgebern

– SetAbsEncoderReference()
– ClearAbsEncoderReferenceStatus()

32.2.1 Start()

Die Funktion startet eine Referenzfahrt einer MCX-Achse. Die Achse beginnt unmittelbar ihre Bewegung in die mit `Direction` vorgegebene Richtung. Ihr Verhalten beim Erreichen der Hardware-Endschalter hängt von der gewählten Art der Referenzfahrt ab. So kann sie beispielsweise selbstständig reversieren und die Bewegung in die andere Richtung fortsetzt. Die Bewegung wird beendet, wenn eine der folgenden Möglichkeiten eintritt:

– Der Referenzpunkt wurde gefunden.
– Ein Grenze wurde erreicht
– Ein Fehler wurde erkannt.

Übergabeparameter

Beachten Sie folgende Regeln:

– Alle Parameter werden in den Einheiten angegeben, die vom Anwender für die Achse vordefiniert wurden.
– Die Parameter beziehen sich auf das Achsobjekt; d. h. sie bilden eine Eigenschaft der Achse. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Parameter	Datentyp	Bedeutung
Direction	Directions (optional)	Definiert die Richtung, in die die Referenzfahrt gestartet wird (siehe Direction [▶ 163]).
SwitchConfig	ReferencingSwitchConfigs (optional)	Definiert die Schalterkonfiguration, die für die Referenzfahrt angewandt wird (siehe SwitchConfig [▶ 164]).
Mode	ReferencingModes (optional)	Definiert die Art der Referenzpunktsuche (siehe Mode [▶ 165]).
Acceleration	Double (optional)	Definiert die maximale Beschleunigung, die in der Startrampe während der Positionierung erreicht werden soll (siehe Acceleration [▶ 166]). Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.
SpeedSwitchSearch	Double (optional)	Definiert die Geschwindigkeit, mit der die Achse ihre Positionierung zur Suche der End- und/oder Referenzschalter ausführt. (siehe SpeedSwitchSearch [▶ 166]).
MaxDistanceSwitchSearch	Double (optional)	Definiert den Weg, den die Achse bis zum Erreichen der relevanten Schalter für die Referenzfahrt maximal zurücklegen darf. Wird diese Weglänge überschritten, wird die Referenzfahrt mit einem Fehler abgebrochen (siehe MaxDistanceSwitchSearch [▶ 167]).
SpeedReferenceSearch	Double (optional)	Definiert die Geschwindigkeit, mit der die Achse den Referenzpunkt sucht, nachdem der Referenz-/Endschalter gefunden wurde (siehe SpeedReferenceSearch [▶ 168]).
MaxDistanceReferenceSearch	Double (optional)	Definiert den Weg, den die Achse bis zum Erreichen des Referenzpunkts maximal zurücklegen darf, nachdem der Referenz-/Endschalter gefunden wurde. Wird diese Distanz überschritten, wird die Referenzfahrt mit einem Fehler abgebrochen (siehe MaxDistanceReferenceSearch [▶ 168]).
ReferenceOffset	Double (optional)	Grundstellung: Abstand: Definiert die Anzahl an Positionseinheiten, um die die Achse nach dem Auffinden der Referenz bewegt wird, bevor die Position auf Grundstellung gesetzt wird (siehe ReferenceOffset [▶ 169]).
NormalPosition	Double (optional)	Grundstellung; Definiert die Position, auf die die Achse gesetzt wird, nachdem die Referenzfahrt erfolgreich beendet wurde (siehe NormalPosition [▶ 170]).

Syntax

```
AX1.MoveHome.Start(Directions.Positive,ReferencingSwitchConfigs.RefAndLi-
mitSwitch,ReferencingModes.ZeroPulse_2Phase,3000,500,200,20,15,125,0);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√	Läuft	Freigegeben
Läuft	X		Läuft
Unterbrochen	X		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.2.2 SetReference()

Die Funktion setzt die aktuelle Position der MCX-Achse auf den Wert, der mit der Funktion übergeben wird.

Nach dem Neustart der Steuerung ist dieser Wert `0.0`. Wird für den Parameter kein Wert übergeben, so greift die Motion Control auf den zuletzt übergebenen Wert zurück.

Parameter	Datentyp	Bedeutung
Position	Double (optional)	Definiert die Referenzposition.

Syntax

```
AX1.MoveHome.SetReference(10.0);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.2.3 SetAbsEncoderReference()

ⓘ Info

Referenzverschiebung beim Einsatz von Absolutwertgebern

Die Verwendung dieses Befehls ist sinnvoll, wo für die Referenzverschiebung Absolutwertgeber zum Einsatz kommen.

Beachten Sie hierzu die Informationen in der JetSym-Online-Hilfe, oder im Informationsblatt *Abspeichern und Wiederermitteln des Referenzpunkts bei (Multiturn-) Absolutwertgebern (Gespeicherte Referenzverschiebung)* .

Zur Anwendung des Befehls muss die Speicherfunktion im NVRAM aktiviert sein.

Übergabeparameter

Der übergebene Wert wird auf den Parameter `Grundstellung: Position` (`NormalPosition`) übertragen. Wird kein Wert übergeben, so wird der zuletzt in den Parameter `Grundstellung: Position` (`NormalPosition`) übergebene Wert verwendet.

Im Gegensatz zur Funktion „SetReference“ wird der intern berechnete Referenzoffset im internen batteriegepufferten RAM (und einem eventuell vorhandenen USB-Stick) abgespeichert. Beim erneuten Booten der Steuerung wird dieser berechnete Wert wieder für die Referenzverschiebung verwendet so dass die Istposition der MCX-Achse direkt mit der einmal festgelegten Referenzverschiebung betrieben werden kann. Der Referenzstatus ist dann sofort „Grundstellung gesetzt“.

Parameter	Datentyp	Bedeutung
Position	Double (optional)	Schreibt den Parameter `Grundstellung: Position` (`NormalPosition`).

Syntax

```
AX1.MoveHome.SetAbsEncoderReference(0.0);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.2.4 ClearAbsEncoderReferenceStatus()

ⓘ Info

Referenzverschiebung beim Einsatz von Absolutwertgebern

Die Verwendung dieses Befehls ist sinnvoll, wo für die Referenzverschiebung Absolutwertgeber zum Einsatz kommen.

Beachten Sie hierzu die Informationen in der JetSym-Online-Hilfe, oder im Informationsblatt *Abspeichern und Wiederermitteln des Referenzpunkts bei (Multiturn-) Absolutwertgebern (Gespeicherte Referenzverschiebung)* .

Zur Anwendung des Befehls muss die Speicherfunktion im NVRAM aktiviert sein.

Mit der Funktion löschen Sie den Referenzstatus einer Achse. Das hat folgende Auswirkungen:

– Der Referenzzustand wird auf `NoReference` gesetzt.
– Die Abfrage `IsAbsEncoderReferenceValid` gibt `false` zurück.
– Der anliegende Referenzoffset bleibt bis zum Neustart der Steuerung wirksam.

Diese Funktion kann auch in Anwendungen ohne Absolutwertgeber verwendet werden, um den Referenzzustand auf `NoReference` zu setzen.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.MoveHome.ClearAbsEncoderReferenceStatus();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.
– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.
X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.3 Eigenschaften von MCAxis.MoveHome

Die Funktionalität bietet folgende Eigenschaften:

– Direction
– SwitchConfig
– Mode
– Acceleration

– SpeedSwitchSearch
– MaxDistanceSwitchSearch
– SpeedReferenceSearch
– MaxDistanceReferenceSearch
– ReferenceOffset
– NormalPosition
– ReferenceState
– IsReferenceRunStarted
– IsSwitchFound
– IsReferenceFound
– IsReferenceSet

Funktionen für die Referenzverschiebung bei der Verwendung von Absolutwertgebern

– IsAbsEncoderReferenceValid
– ReferenceShift

32.3.1 Direction

Beschreibung

Datentyp	Bedeutung
Typ Directions (optional)	Liest oder schreibt die Richtung, in die die Referenzfahrt gestartet wird.
Directions.positive	Die Achse startet die Referenzfahrt in positive Richtung (Zählrichtung der Ist-Position).
Directions.negative	Die Achse startet die Referenzfahrt in negative Richtung (Zählrichtung der Ist-Position).

Syntax

```
RefDirection := AX1.MoveHome.Direction;
AX1.MoveHome.Direction := Directions.Negative;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.3.2 SwitchConfig

Beschreibung

Datentyp	Bedeutung
ReferencingSwitchConfigs (optional)	Liest oder schreibt die Schalterkonfiguration, die für die Referenzfahrt angewandt wird.
ReferencingSwitchConfigs.NoSwitch	Es wird kein Schalter berücksichtigt. Das Setzen der Referenz kann nur auf den Nullimpuls des Encoders erfolgen. Diese Einstellung kann beispielsweise bei Modulo-Achsen ohne Getriebe zum Einsatz kommen.
ReferencingSwitchConfigs.RefAndLimitSwitch	Referenz- und Endschalter werden berücksichtigt. Mögliche Szenarien: – Wird der Referenzschalter vor dem Endschalter erreicht, so wird die Referenz am Referenzschalter ermittelt. – Wird der erste Endschalter vor dem Referenzschalter erreicht, so reversiert die Achse. Wird daraufhin der Referenzschalter erreicht, so wird die Referenz am Referenzschalter ermittelt. – Wird der zweite Endschalter vor dem Referenzschalter erreicht, so wird die Referenzfahrt mit Fehler abgebrochen.
ReferencingSwitchConfigs.RefSwitch	Nur der Referenzschalter wird beachtet. Wird ein Endschalter vor dem Referenzschalter erreicht, so wird die Referenzfahrt mit Fehler abgebrochen.
ReferencingSwitchConfigs.LimitSwitch	Nur die Endschalter werden beachtet. Die Endschalter dienen als Referenzschalter.

Syntax

```
RefDirection := AX1.MoveHome.Direction;
AX1.MoveHome.Direction := Directions.Negative;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.3.3 Mode

Beschreibung

Datentyp	Bedeutung
ReferencingModes (optional)	Liest oder schreibt den Modus, der für die Referenzfahrt angewandt wird.
ReferencingModes.ZeroPulse_2Phase	Der Referenzpunkt wird am Nullimpuls des Encoders ermittelt. Die Referenzfahrt wird 2-phasig ausgeführt.
ReferencingModes.SwitchingEdge_2Phase	Der Referenzpunkt wird an einer Schalterflanke (Referenz- oder Endschalter) ermittelt. Die Referenzfahrt wird 2-phasig ausgeführt.
ReferencingModes.ZeroPulse_1Phase	Der Referenzpunkt wird am Nullimpuls des Encoders ermittelt. Die Referenzfahrt wird 1-phasig ausgeführt.
ReferencingModes.SwitchingEdge_1Phase	Der Referenzpunkt wird an einer Schalterflanke (Referenz- oder Endschalter) ermittelt. Die Referenzfahrt wird 1-phasig ausgeführt.

Syntax

```
RefMode := AX1.MoveHome.Mode;
AX1.MoveHome.Mode := ReferencingModes.SwitchingEdge_2Phase;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.3.4 Acceleration

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest oder schreibt die maximale Beschleunigung, die in der Startrampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.

Syntax

```
RefBeschleunigung := AX1.MoveHome.Acceleration;
AX1.MoveHome.Acceleration := 10000;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.3.5 SpeedSwitchSearch

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest oder schreibt die Geschwindigkeit, mit der die Achse ihre Positionierung zur Suche der End- und/oder Referenzschalter ausführt.

Syntax

```
RefSpeed1 := AX1.MoveHome.SpeedSwitchSearch;
AX1.MoveHome.SpeedSwitchSearch := 250;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.3.6 MaxDistanceSwitchSearch

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest oder schreibt den Weg, den die Achse bis zum Erreichen der relevanten Schalter für die Referenzfahrt maximal zurücklegen darf. Wird diese Weglänge überschritten, wird die Referenzfahrt mit einem Fehler abgebrochen.

Syntax

```
MaxReferenzPosition := AX1.MoveHome.MaxDistanceSwitchSearch;
AX1.MoveHome.MaxDistanceSwitchSearch := 5000.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.3.7 SpeedReferenceSearch

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest oder schreibt die Geschwindigkeit, mit der die Achse den Referenzpunkt sucht, nachdem der Referenz-/Endschalter gefunden wurde. Eine Referenzfahrt ohne Schaltersuche sowie eine 1-phasige Referenzfahrt wird nur mit dieser Geschwindigkeit ausgeführt.

Syntax

```
RefSpeed2 := AX1.MoveHome.SpeedReferenceSearch;
AX1.MoveHome.SpeedReferenceSearch := 50;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.3.8 MaxDistanceReferenceSearch

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest oder schreibt den Weg, den die Achse bis zur Erreichung des Referenzpunkts maximal zurücklegen darf, nachdem der Referenz-/Endschalter gefunden wurde. Wird diese Distanz überschritten, wird die Referenzfahrt mit einem Fehler abgebrochen.

Syntax

```
MaxReferenceSearchPos := AX1.MoveHome.MaxDistanceReferenceSearch;
AX1.MoveHome.MaxDistanceReferenceSearch := 100.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.3.9 ReferenceOffset

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest oder schreibt die Anzahl an Positionseinheiten, um die die Achse nach dem Auffinden der Referenz bewegt wird, bevor die Position auf Grundstellung gesetzt wird.

Syntax

```
GrundstellungAbstand := AX1.MoveHome.ReferenceOffset;
AX1.MoveHome.ReferenceOffset := 1265.5;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.3.10 NormalPosition

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest oder schreibt die Position, auf die die Achse gesetzt wird, nachdem die Referenz gefunden wurde. Auf diesen Wert setzt das MCX-System die Soll- und Ist-Position der Achse am Ende einer erfolgreichen Referenzfahrt.

Syntax

```
GrundstellungPosition := AX1.MoveHome.NormalPosition;
AX1.MoveHome.NormalPosition := 0.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.3.11 ReferenceState

Beschreibung

Datentyp	Bedeutung
Typ MCReferenceStates (optional)	Liest den Status der Referenzfahrt. Anhand dieser Information können Sie erkennen ob die Achse bereits referenziert ist, bzw. in welchem Teil der Referenzfahrt sich die Achse befindet. Es gibt 5 Referenzfahrtstatus, die abgefragt werden können. Je nach Art der Referenzfahrt können einzelne dieser Status übersprungen bzw. nicht angenommen werden.
MCReferenceStates.NoReference	– Die Achse wurde noch nicht referenziert, d. h. es ist noch keine erfolgreiche Referenzfahrt erfolgt oder die Funktion `SetReference()` wurde noch nicht ausgeführt. – An einer Achse mit dem Referenzstatus `ReferenceSet` trat beim erneuten Versuch zur Ermittlung der Referenz ein Fehler auf. – Die Funktion ClearAbsEncoderReferenceStatus() [▶ 162] wurde ausgeführt.

Datentyp	Bedeutung
MCReferenceStates.Refe-renceRunStarted	Eine Referenzfahrt wurde gestartet. Der relevante End- oder Referenz-schalter für den Referenzpunkt wurde noch nicht gefunden.
MCReferenceStates.Switch-Found	Der relevante End- oder Referenzschalter wurde gefunden (Übergang von Phase 1 auf Phase 2).
MCReferenceStates.Reference-Found	Die Referenzmarke wurde gefunden.
MCReferenceStates.Reference-Set	Die Grundstellung wurde gesetzt. Damit ist die Achse referenziert und die Referenzfahrt beendet.

Syntax

```
When AX1.MoveHome.ReferenceState == MCReferenceStates.ReferenceRunStarted
Continue;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

32.3.12 IsReferenceRunStarted, IsSwitchFound, IsReferenceFound, IsReferenceSet

Beschreibung

Datentyp	Bedeutung
Bool	Die Eigenschaften lesen, ob sich die Achse in dem abgefragten Referenzstatus befindet (siehe ReferenceState [▶ 170]). Ist keine dieser Eigenschaften erfüllt, ist der Referenzstatus der Achse `NoReference`.

Syntax

```
ReferenzfahrtStart := AX1.MCAxis.MoveHome.IsReferenceRunStarted
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

32.3.13 IsAbsEncoderReferenceValid

Beschreibung

ⓘ Info

Referenzverschiebung beim Einsatz von Absolutwertgebern

Die Verwendung dieses Befehls ist sinnvoll, wo für die Referenzverschiebung Absolutwertgeber zum Einsatz kommen.

Beachten Sie hierzu die Informationen in der JetSym-Online-Hilfe, oder im Informationsblatt *Abspeichern und Wiedererermitteln des Referenzpunkts bei (Multiturn-) Absolutwertgebern (Gespeicherte Referenzverschiebung)* .

Zur Anwendung des Befehls muss die Speicherfunktion im NVRAM aktiviert sein.

Datentyp	Bedeutung
Bool	Frag ab, ob die Referenzverschiebungsfunktion für diese MCX-Achse aktiv ist. Wenn `True`, liegt eine Referenzverschiebung vor (kann auch 0 sein) und wird beim Booten der Steuerung übernommen.

Syntax

```
If AX1.MotionHome.IsAbsEncoderReferenceValid then …
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

32.3.14 ReferenceShift

Beschreibung

ⓘ Info

Referenzverschiebung beim Einsatz von Absolutwertgebern

Die Verwendung dieses Befehls ist sinnvoll, wo für die Referenzverschiebung Absolutwertgeber zum Einsatz kommen.

Beachten Sie hierzu die Informationen in der JetSym-Online-Hilfe, oder im Informationsblatt *Abspeichern und Wiedererermitteln des Referenzpunkts bei (Multiturn-) Absolutwertgebern (Gespeicherte Referenzverschiebung)* .

Zur Anwendung des Befehls muss die Speicherfunktion im NVRAM aktiviert sein.

Datentyp	Bedeutung
Double (optional)	Liest oder schreibt die Anzahl der Positionseinheiten, um die die Nullposition der MCX-Achse mit Absolutwertgeber gegenüber dem Nullpunkt des Gebers verschoben ist. Dieser Wert kann auch beschrieben werden. **HINWEIS! Dies ist allerdings nur für sehr geringe Korrekturen sinnvoll**. Verwenden Sie bevorzugt die Funktion `SetAbsEncoderReference()`, um die Position festzulegen, an der die MCX-Achse stehen soll.

Syntax

```
Referenzverschiebung := AX1.MoveHome.ReferenceShift;
AX1.MoveHome.ReferenceShift := 257.5;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

33 MCAxis.MovePtp

Mit der Funktionalität **MCAxis.MovePtp** können Sie die Punkt-zu-Punkt-Positionierung einer MCX-Achse starten und parametrieren.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

33.1 Syntax für MCAxis.MovePtp

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.MovePtp.<Funktion oder Zugriff auf Wert>(optional <Parame-
ter 1, Parameter 2, Parameter n>);
```

33.2 Funktionen von MCAxis.MovePtp

Die Funktionalität bietet folgende Funktionen:

- Start()
- StartNewTargetPosition()
- NewSpeed()
- NewAcceleration()
- NewDeceleration()

33.2.1 Start()

Die Funktion startet die Punkt-zu-Punkt-Positionierung einer MCX-Achse. Die Achse beginnt ihre Bewegung unmittelbar oder wendet die Daten auf eine laufende Positionierung an.

Da alle für die Positionierung notwendigen Parameter in nur einem Befehl übergeben werden können, ist dies die kompakteste Form, um eine Positionierbewegung vollständig zu parametrieren und zu starten.

Übergabeparameter

Beachten Sie folgende Regeln:

- Alle Parameter werden in den Einheiten angegeben, die vom Anwender für die Achse vordefiniert wurden.
- Die Parameter beziehen sich auf das Achsobjekt; d. h. sie bilden eine Eigenschaft der Achse. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Positionieren im gepufferten Betrieb

Im gepufferten Betrieb können Sie mehrere Positionierungen aufrufen, die nacheinander abgearbeitet werden. Das heißt, die Motion-Control-Einheit startet eine neue Positionierung, sobald der Sollwert einer laufenden Positionierung das Ziel erreicht hat. Weiterführende Information hierzu finden Sie im Abschnitt MCAxis.Buffer [▶ 31].

Positionieren im nichtgepufferten Betrieb (Standard)

Je nachdem, ob die Achse in Bewegung ist oder nicht, ergibt sich folgendes Verhalten:

– Aufruf der Funktion im Zustand **Freigegeben**: Die Bewegung wird gestartet.

– Aufruf der Funktion im Zustand **Läuft**: Die Bewegungsparameter werden gemäß nachfolgender Tabelle für die laufende ODER nachfolgende Bewegung übernommen.

Parameter	Datentyp	Bedeutung	Verhalten beim Funktionsaufruf im Zustand Läuft
PositioningMode	AxisPositioningModes (optional)	Definiert den Positioniermodus [▶ 183].	Der Modus wird sofort wirksam.
TargetPosition	Double (optional)	Definiert die Zielposition, an der die Achse stehen soll, nachdem die Positionierung ausgeführt wurde.	– Kann die neue Zielposition unter Berücksichtigung der Maximalwerte für Geschwindigkeit, Beschleunigung, Verzögerung und Ruck erreicht werden, so wird sie unter Verwendung der neuen Bewegungsparameter direkt angefahren. – Andernfalls wird die laufende Positionierung zu Ende geführt und die neu programmierte direkt angeschlossen.
Speed	Double (optional)	Definiert die Geschwindigkeit, mit der die Achse positioniert werden soll, solange sie sich nicht in den Rampen (Beschleunigung, Verzögerung) befindet. Der Wert muss größer `0` sein.	Wird eine neue Positionierung gestartet, während die vorherige noch läuft (Zustand Läuft), so wird der neue Geschwindigkeitswert unter Berücksichtigung der Maximalwerte für Beschleunigung, Verzögerung und Ruck sofort übernommen. Die Änderung der Geschwindigkeit erfolgt mit den Parametern der Beschleunigungsrampe.

Parameter	Datentyp	Bedeutung	Verhalten beim Funktionsaufruf im Zustand Läuft
Acceleration	Double (optional)	Definiert die maximale Beschleunigung, die in der Startrampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.	Die Geschwindigkeitsrampen werden entsprechend der neuen Werte für Beschleunigung und Verzögerung ausgeführt. – Befindet sich die Bewegung gerade in einer solchen Rampe, kommt es zu Übergangsformen, die nicht dem eingestellten Rampentyp entsprechen. – Die Übergänge werden immer unter der Überwachung der Ruckbegrenzung ausgeführt.
Deceleration	Double (optional)	Definiert die maximale Verzögerung, die in der Stopprampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.	
TargetWindow	Double (optional)	Definiert die Breite des Zielfensters für die Positionierung.	Die Änderung des Zielfensters wird sofort in die laufende Positionierung übernommen.

Positionierungsphasen

Eine Positionierbewegung besteht aus folgenden Phasen:

1. Beschleunigungsrampe
2. Konstante Geschwindigkeit

   Diese Phase tritt nur ein, wenn die Geschwindigkeit `Speed` mit den vorgegebenen MovePtp-Parametern erreicht werden kann. Andernfalls geht die Beschleunigungsrampe direkt in die Verzögerungsrampe über (Dreiecksbetrieb).
3. Verzögerungsrampe

Rampenformen

Die Rampenformen der Beschleunigungs- und Verzögerungsphase werden durch den Achsparameter `MCAxis.Mechanism.Slope.SlopeType` (Typ `AxisSlopeTypes`) vorgegeben (MCAxis.Mechanism.Slope [▶ 140]). Hier gibt es zwei mögliche Einstellungen:

– `AxisSlopeTypes.Linear`

  Beschleunigungsphase = konstante Beschleunigung,
  Verzögerungsphase = sin-förmige Beschleunigung

– `AxisSlopeTypes.Sine`

  Beschleunigungsphase = sin-förmige Beschleunigung,
  Verzögerungsphase = sin²-förmige Beschleunigung

ⓘ Info

Ruckbegrenzung

Die sin- und sin²-Rampen werden auf Einhaltung des maximal zulässigen Rucks (Parameter `MCAxis.Mechanism.Limitations.MaxJerk`) geprüft und bei Bedarf entsprechend angepasst.

Syntax

```
AX1.MovePtp.Start(AxisPositioningModes.AbsNormal,20000,500,1000,1000,0.5);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√	Läuft	Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	X		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Positionierung stehen Ihnen verschiedene Informationen zur Verfügung:

– Der **Zustand der Achse** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder die Achse angehalten wurde (MCAxis.State [▶ 208]).

```
When Ax1.State.IsEnabled Continue; //die Achse befindet sich in Lageregelung
und steht.
When Ax1.State.IsRunning Continue; //der Sollwertgenerator erzeugt eine Be-
wegung.
```

– Der **Rampenstatus** informiert darüber, in welchem Bewegungsabschnitt sich der Sollwertgenerator aktuell befindet (MCAxis.Mechanism.Slope [▶ 140]).

```
When Ax1.Mechanism.Slope.IsAccelerating Continue;
When Ax1.Mechanism.Slope.IsAtConstantSpeed Continue;
When Ax1.Mechanism.Slope.IsDecelerating Continue;
When Ax1.Mechanism.Slope.IsStopped Continue;
```

– Wenn Sie das Erreichen einer definierten Position kurz vor dem Ziel abfragen möchten, kommt das **Zielfenster** zum Einsatz (MCAxis.IsInTargetWindow [▶ 116]).

```
When Ax1.IsInTargetWindow Continue;
```

33.2.2 StartNewTargetPosition()

Die Funktion startet die Punkt-zu-Punkt-Positionierung einer MCX-Achse. Die Achse beginnt unmittelbar ihre Bewegung oder übernimmt die Daten für die Änderung einer laufenden Positionierung.

Die Funktion ist eine vereinfachte Variante der Funktion `Start()`, mit der nur die relevanten Positionsparameter übergeben werden. Geschwindigkeitswerte und Rampeneinstellungen für die Achse werden übernommen.

Übergabeparameter

Beachten Sie folgende Regeln:

– Alle Parameter werden in den Einheiten angegeben, die vom Anwender für die Achse vordefiniert wurden.

– Die Parameter beziehen sich auf das Achsobjekt; d. h. sie bilden eine Eigenschaft der Achse. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Positionieren im gepufferten Betrieb

Im gepufferten Betrieb können Sie mehrere Positionierungen aufrufen, die nacheinander abgearbeitet werden. Das heißt, die Motion-Control-Einheit startet eine neue Positionierung, sobald der Sollwert einer laufenden Positionierung das Ziel erreicht hat. Weiterführende Informationen hierzu finden Sie im Abschnitt MCAxis.Buffer [▶ 31].

Positionieren im nichtgepufferten Betrieb (Standard)

Je nachdem, ob die Achse in Bewegung ist oder nicht, ergibt sich folgendes Verhalten:

– Aufruf der Funktion im Zustand **Freigegeben**: Die Bewegung wird gestartet.

– Aufruf der Funktion im Zustand **Läuft**: Die Bewegungsparameter werden gemäß nachfolgender Tabelle für die laufende ODER nachfolgende Bewegung übernommen.

Parameter	Datentyp	Bedeutung	Verhalten beim Funktionsaufruf im Status Läuft
PositioningMode	AxisPositioningModes (optional)	Definiert den Positioniermodus [▶ 183].	Der Modus wird sofort wirksam.
TargetPosition	Double (optional)	Definiert die Zielposition, an der die Achse stehen soll, nachdem die Positionierung ausgeführt wurde.	– Kann die neue Zielposition unter Berücksichtigung der Maximalwerte für Geschwindigkeit, Beschleunigung, Verzögerung und Ruck erreicht werden, so wird sie unter Verwendung der neuen Bewegungsparameter direkt angefahren. – Andernfalls wird die laufende Positionierung zu Ende geführt und die neu programmierte direkt angeschlossen.
TargetWindow	Double (optional)	Definiert die Breite des Zielfensters für die Positionierung.	Die Änderung des Zielfensters wird sofort in die laufende Positionierung übernommen.

Positionierungsphasen

Eine Positionierbewegung besteht aus folgenden Phasen:

1. Beschleunigung
2. Konstante Geschwindigkeit;

   Diese Phase tritt nur ein, wenn die Geschwindigkeit `Speed` mit den vorgegebenen MovePtp-Parametern erreicht werden kann.
3. Verzögerung

Rampenformen

Die Rampenformen der Beschleunigungs- und Verzögerungsphase werden durch den Achsparameter `MCAxis.Mechanism.Slope.SlopeType (Typ AxisSlopeTypes)` vorgegeben. Hier gibt es zwei mögliche Einstellungen:

– `AxisSlopeTypes.Linear`

  Beschleunigungsphase = konstante Beschleunigung,
  Verzögerungsphase = sin-förmige Beschleunigung

– `AxisSlopeTypes.Sine`

  Beschleunigungsphase = sin-förmige Beschleunigung,
  Verzögerungsphase = sin²-förmige Beschleunigung

ⓘ Info

Ruckbegrenzung

Die sin- und sin²-Rampen werden auf Einhaltung des maximal zulässigen Rucks (Parameter `MCAxis.Mechanism.Limitations.MaxJerk`) geprüft und bei Bedarf entsprechend angepasst.

Syntax

```
AX1.MovePtp.StartNewTargetPosition(AxisPositioningModes.AbsNor-
mal,20000,0.5);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√	Läuft	Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	X		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Positionierung stehen Ihnen verschiedene Informationen zur Verfügung:

– Der **Zustand der Achse** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder die Achse angehalten wurde (MCAxis.State [▶ 208]).

```
When Ax1.State.IsEnabled Continue; //die Achse befindet sich in Lageregelung
und steht.
When Ax1.State.IsRunning Continue; //der Sollwertgenerator erzeugt eine Be-
wegung.
```

– Der **Rampenstatus** informiert darüber, in welchem Bewegungsabschnitt sich der Sollwertgenerator aktuell befindet (MCAxis.Mechanism.Slope [▶ 140]).

```
When Ax1.Mechanism.Slope.IsAccelerating Continue;
```

```
When Ax1.Mechanism.Slope.IsAtConstantSpeed Continue;
When Ax1.Mechanism.Slope.IsDecelerating Continue;
When Ax1.Mechanism.Slope.IsStopped Continue;
```

– Wenn Sie das Erreichen einer definierten Position kurz vor dem Ziel abfragen möchten, kommt das **Zielfenster** zum Einsatz (MCAxis.IsInTargetWindow [▶ 116]).

```
When Ax1.IsInTargetWindow Continue;
```

33.2.3 NewSpeed()

Die Funktion übergibt einen neuen Wert für die Geschwindigkeit einer Punkt-zu-Punkt-Positionierung einer MCX-Achse. Dabei gilt:

– Der Wert muss > `0` sein.

– Ist die Achse bereits in Bewegung (Zustand **läuft**), so wird der neue Geschwindigkeitswert sofort übernommen.

  Der Übergang von aktueller zu neuer Geschwindigkeit bei einer laufenden Achse erfolgt mit der für die Beschleunigungsrampe eingestellten Beschleunigung. Dies ist auch dann der Fall, wenn sich die Geschwindigkeit verringert.

– Ist die Achse im Stillstand (Zustand **freigegeben**), wird der Wert erst bei der nächsten PtP-Positionierung angewendet.

– Der Rampentyp kann während einer Positionierung nicht geändert werden.

– Bei allen Übergängen ist die Ruckbegrenzung aktiv, falls der Rampentyp *sin²* eingestellt ist.

ⓘ Info

Achse zum Stillstand bringen

Um eine Achse zum Stillstand zu bringen sind ausschließlich die Befehle `MoveHalt` oder `Stop` zu verwenden. Es ist **nicht** möglich eine Achse anzuhalten, indem ein Geschwindigkeitswert von `0` übergeben wird.

Übergabeparameter

Parameter	Datentyp	Bedeutung	Verhalten beim Funktionsaufruf im Zustand **läuft**
Speed	Double	Definiert die Geschwindigkeit, mit der die Achse positioniert werden soll, solange sie sich nicht in den Rampen (Beschleunigung, Verzögerung) befindet.	Der neue Geschwindigkeitswert wird unter Berücksichtigung der Maximalwerte für Beschleunigung, Verzögerung und Ruck sofort übernommen.

Syntax

```
AX1.MovePtp.NewSpeed(2000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

33.2.4 NewAcceleration()

Die Funktion übergibt einen neuen Wert für die maximale Beschleunigung einer Punkt-zu-Punkt-Positionierung einer MCX-Achse.

Dabei gilt:

– Eine aktive Beschleunigungsrampe wird nicht verändert.

– Der neue Wert wird bei der nächsten PtP-Positionierung angewendet.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Acceleration	Double (optional)	Definiert die maximale Beschleunigung, die in der Startrampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.

Syntax

```
AX1.MovePtp.NewAcceleration(2000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

33.2.5 NewDeceleration()

Die Funktion übergibt einen neuen Wert für die maximale Verzögerung einer Punkt-zu-Punkt-Positionierung einer MCX-Achse.

Dabei gilt:

– Der Wert wird auch während einer laufenden PtP-Positionierung verwendet, falls das Ziel mit dem neuen Parameter noch erreicht werden kann.

– Andernfalls wird der Wert bei der nächsten PtP-Positionierung angewendet.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Deceleration	Double (optional)	Definiert die maximale Verzögerung, die in der Stopprampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.

Syntax

```
AX1.MovePtp.NewDeceleration(2000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

33.3 Eigenschaften von MCAxis.MovePtp

Die Funktionalität bietet folgende Eigenschaften:

– PositioningMode
– TargetPosition
– TargetWindow

33.3.1 PositioningMode

Beschreibung

Mit der Eigenschaft können Sie den momentan gültigen Positioniermodus einer MCX-Achse lesen oder einen neuen Wert für diese Achse setzen.

Dabei gilt:

– Ein neuer Wert wird bei der nächsten Punkt-zu-Punkt-Positionierung wirksam, auch wenn die Änderung während einer laufenden Positionierung übergeben wird.
– Nach dem Einschalten wird die letzte Zielposition aus der Ist-Position zum Einschaltzeitpunkt berechnet.

Datentyp	Bedeutung
AxisPositioningMode	Modus für die Art der PtP-Bewegung.
AxisPositioningMode.AbsNormal	Bewegt die Achse auf die im Parameter `TargetPosition` übergebene Position. Diese wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Positionierung ergibt sich aus der letzten und der neuen Zielposition.
AxisPositioningMode.RelTarget	Bewegt die Achse um den Weg weiter, der im Parameter `TargetPosition` übergeben wurde. Die Zielposition berechnet sich aus der letzten Zielposition addiert mit dem Wert in `TargetPosition`. Die Fahrtrichtung ergibt sich aus dem Vorzeichen des Parameters `TargetPosition`.
AxisPositioningMode.RelActual	Bewegt die Achse um den Weg weiter, der im Parameter `TargetPosition` übergeben wurde. Die Zielposition berechnet sich aus Ist-Position beim Start der Positionierung addiert mit dem Wert in `TargetPosition`. Die Fahrtrichtung ergibt sich aus dem Vorzeichen des Parameters `TargetPosition`.
AxisPositioningMode.AbsModuloPos	Nur für Modulo-Achsen gültig. Bewegt die Achse auf die im Parameter `TargetPosition` übergebene Position. Diese wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Positionierung ist immer **positiv**. Dabei kann der Modulo-Umbruch überfahren werden, wenn die neue Zielposition kleiner als die letzte Zielposition ist.
AxisPositioningMode.AbsModuloNeg	Nur für Modulo-Achsen gültig. Bewegt die Achse auf die im Parameter `TargetPosition` übergebene Position. Diese wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Positionierung ist immer **negativ**. Dabei kann der Modulo-Umbruch überfahren werden, wenn die neue Zielposition größer als die letzte Zielposition ist.

Datentyp	Bedeutung
AxisPositioningMode.AbsModulo-Auto	Nur für Modulo-Achsen gültig. – Positionierung wird bei angehaltener Achse gestartet: Bewegt die Achse auf die im Parameter `TargetPosition` übergebene Position. Diese wird in Relation zur Referenzposition **auf dem kürzesten Weg** angefahren. Die Fahrtrichtung der Positionierung ergibt sich aus dem kürzesten Weg zum Ziel. – Positionierung wird bei laufender Achse gestartet: Bewegt die Achse auf die im Parameter `TargetPosition` übergebene Position. Diese wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Positionierung ergibt sich aus der Fahrrichtung der laufenden Positionierung.
AxisPositioningMode.AbsModulo-KeepDirection	Nur für Modulo-Achsen gültig. Bewegt die Achse auf die im Parameter `TargetPosition` übergebene Position. Diese wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Positionierung ergibt sich aus der Fahrrichtung der letzten Positionierung.

Syntax

```
AX_Positioniermodus := AX1.MovePtp.PositioningMode;
AX1.MovePtp.PositioningMode := AxisPositioningModes.RelTarget;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	√	Fehler: freigegeben
Fehler: gesperrt	√	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

33.3.2 TargetPosition

Beschreibung

Mit der Eigenschaft können Sie den momentan gültigen Wert von `TargetPosition` einer MCX-Achse lesen oder einen neuen Wert für diese Achse schreiben. Dieser wird beim Start der nächsten Punkt-zu-Punkt-Positionierung wirksam.

Datentyp	Bedeutung
Double	Eigenschaft zur Ermittlung der Zielposition abhängig von der Eigenschaft `PositioningMode`.

Syntax

```
Target := AX1.MovePtp.TargetPosition;
AX1.MovePtp.TargetPosition := 12000;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	√	Fehler: freigegeben
Fehler: gesperrt	√	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

33.3.3 TargetWindow

Beschreibung

Mit der Eigenschaft können Sie den momentan gültigen Wert der Zielfenstereinstellung einer MCX-Achse lesen oder einen neuen Wert für diese Achse schreiben. Dieser wird beim Start der nächsten Punkt-zu-Punkt-Positionierung wirksam.

Datentyp	Bedeutung
Double	Definiert, wann im Bereich der Zielposition einer Positionierung der Status `IsInTargetWindow` aktiv wird.

Syntax

```
Targetwindow := AX1.MovePtp.TargetWindow;
AX1.MovePtp.TargetWindow := 0.7;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben

Ausgangszustand	Erlaubt	Endzustand
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	√	Fehler: freigegeben
Fehler: gesperrt	√	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

33.4 Beispieltask MCAxis.MovePtp

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Var
    run: bool; //Hilfsflag zum Weiterschalten im Programm
End_Var;

Task Sample2 Autorun

loop
    when run Continue;

    //Einschalten des Servoverstärkers
    If AX1.State.IsDisabled AND AX1.Drive.IsReadyForOperation Then
        AX1.Power.Enable();
    End_if;
    When AX1.State.IsEnabled Continue;

    AX1.MoveHome.SetReference(0.0); //Referenzposition auf Einschaltposition
setzten
    When AX1.MoveHome.IsReferenceSet Continue;

    //Achse absolut positionieren auf die Position 10000 mm - Maximalge-
schwindigkeit 500 mm/s -
    //Max. Beschleunigung 1000 mm/s² - Max. Verzögerung 1000 mm/s²
    //Zielfenster 0.1 mm
    AX1.MovePtp.Start(AxisPositioningModes.AbsNor-
mal,10000,500,1000,5000,0.1);
    When AX1.IsInTargetWindow Continue;

    //Wenn die Achse angekommen ist, 1 Sekunde warten.
    Delay(t#1s);

    //Achse wieder um den gleichen Weg zurückpositionieren; hier relativ zur
letzten Zielposition.
    AX1.MovePtp.StartNewTargetPosition(AxisPositioningModes.RelTar-
get,-10000);
    when AX1.IsInTargetWindow Continue;

    //Wenn die Achse angekommen ist, 1 Sekunde warten.
    Delay(t#1s);
```

```
 //Wenn das Hilfsflag inzwischen vom Anwender im Setup gelöscht wurde, Achse
sperren und auf erneutes Setzten warten.
    //Ansonsten das Reversieren der Achse wiederholen.
    If Not run Then
        AX1.Power.Disable();
        When AX1.State.IsDisabled Continue;
    End_if;

end_loop;

End_Task;
```

34 MCAxis.MoveVelocity

Mit der Funktionalität **MCAxis.MoveVelocity** können Sie die Endlospositionierung einer MCX-Achse starten und parametrieren.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

34.1 Syntax für MCAxis.MoveVelocity

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.MoveVelocity.<Funktion oder Zugriff auf Wert>(optional
<Parameter 1, Parameter 2, Parameter n>);
```

34.2 Funktionen von MCAxis.MoveVelocity

Die Funktionalität bietet folgende Funktionen:

- Start()
- Reverse()
- NewSpeed()
- NewAcceleration()
- NewDeceleration()

34.2.1 Start()

Die Funktion startet die Endlospositionierung einer MCX-Achse. Die Achse beginnt ihre Bewegung unmittelbar oder wendet die Daten auf eine laufende Positionierung an.

Die Achse wird mit der aktuellen oder programmierten Rampe beschleunigt und fährt so lange mit dem aktuellen oder programmierten Wert für `Speed` bis sie durch einen Halt- oder Stopp-Befehl angehalten wird. Die Endlospositionierung ist lagegeregelt, so dass während der Bewegung auch ein Positionsschleppfehler überwacht wird.

Da alle für die Positionierung notwendigen Parameter in nur einem Befehl übergeben werden können, ist dies die kompakteste Form, um eine Positionierbewegung vollständig zu parametrieren und zu starten.

Übergabeparameter

Beachten Sie folgende Regeln:

- Alle Parameter werden in den Einheiten angegeben, die vom Anwender für die Achse vordefiniert wurden.
- Die Parameter beziehen sich auf das Achsobjekt; d. h. sie bilden eine Eigenschaft der Achse. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Positionieren im gepufferten Betrieb

Im gepufferten Betrieb können Sie mehrere Positionierungen aufrufen, die nacheinander abgearbeitet werden. Das heißt, die Motion-Control-Einheit startet eine neue Positionierung, sobald der Sollwert einer laufenden Positionierung das Ziel erreicht hat. Weiterführende Information hierzu finden Sie im Abschnitt MCAxis.Buffer [▶ 31].

Positionieren im nichtgepufferten Betrieb (Standard)

Je nachdem, ob die Achse in Bewegung ist oder nicht, ergibt sich folgendes Verhalten:

– Aufruf der Funktion im Zustand **Freigegeben**: Die Bewegung wird gestartet.

– Aufruf der Funktion im Zustand **Läuft**: Die Bewegungsparameter werden gemäß nachfolgender Tabelle für die laufende ODER nachfolgende Bewegung übernommen.

Parameter	Datentyp	Bedeutung	Verhalten beim Funktionsaufruf im Zustand Läuft
Direction	Directions (optional)	Definiert die Bewegungsrichtung (siehe Direction [▶ 194]).	Die neuen Richtungsparameter werden direkt übernommen. Wenn die neue Richtung entgegengesetzt zur Richtung der augenblicklichen Bewegung ist, wird ein Reversiervorgang mit den Beschleunigungsrampen ausgeführt.
Speed	Double (optional)	Definiert die Geschwindigkeit, mit der die Achse positioniert werden soll, solange sie sich nicht in den Rampen (Beschleunigung, Verzögerung) befindet. Der Wert muss größer `0` sein.	Wird eine neue Positionierung gestartet, während die vorherige noch läuft (Zustand **Läuft**), so wird der neue Geschwindigkeitswert unter Berücksichtigung der Maximalwerte für Beschleunigung, Verzögerung und Ruck sofort übernommen. Die Änderung der Geschwindigkeit erfolgt mit den Parametern der Beschleunigungsrampe.
Acceleration	Double (optional)	Definiert die maximale Beschleunigung, die in der Startrampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.	Die Geschwindigkeitsrampen werden entsprechend der neuen Werte für Beschleunigung und Verzögerung ausgeführt. – Befindet sich die Bewegung gerade in einer solchen Rampe, kommt es zu Übergangsformen, die nicht dem eingestellten Rampentyp entsprechen. – Die Übergänge werden immer unter der Überwachung der Ruckbegrenzung ausgeführt.
Deceleration	Double (optional)	Definiert die maximale Verzögerung, die beim nächsten `MoveHalt.Stop()`-Befehl in der Stopprampe während der Positionierung erreicht werden soll, wenn dort kein anderer Wert übergeben wird. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.	

Rampenformen

Die Rampenformen der Beschleunigungs- und Verzögerungsphase werden durch den Achsparameter `MCAxis.Mechanism.Slope.SlopeType` (Typ `AxisSlopeTypes`) vorgegeben (siehe MCAxis.Mechanism.Slope [▶ 140]). Hier gibt es zwei mögliche Einstellungen:

– `AxisSlopeTypes.Linear`

  Beschleunigungsphase = konstante Beschleunigung,
  Verzögerungsphase = sin-förmige Beschleunigung

– `AxisSlopeTypes.Sine`

  Beschleunigungsphase = sin-förmige Beschleunigung,
  Verzögerungsphase = sin²-förmige Beschleunigung

ⓘ Info

Ruckbegrenzung

Die sin- und sin²-Rampen werden auf Einhaltung des maximal zulässigen Rucks (Parameter `MCAxis.Mechanism.Limitations.MaxJerk`) geprüft und bei Bedarf entsprechend angepasst.

Syntax

```
AX1.MoveVelocity.Start(Directions.Positive,5000,2000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√	Läuft	Läuft
Läuft	√	Läuft	Läuft
Unterbrochen	X		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Positionierung stehen Ihnen verschiedene Informationen zur Verfügung:

– Der **Zustand der Achse** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder die Achse angehalten wurde (MCAxis.State [▶ 208]).

```
When Ax1.State.IsEnabled Continue; //die Achse befindet sich in Lageregelung
und steht.
When Ax1.State.IsRunning Continue; //der Sollwertgenerator erzeugt eine Be-
wegung.
```

– Der **Rampenstatus** informiert darüber, in welchem Bewegungsabschnitt sich der Sollwertgenerator aktuell befindet (MCAxis.Mechanism.Slope [▶ 140]).

```
When Ax1.Mechanism.Slope.IsAccelerating Continue;
When Ax1.Mechanism.Slope.IsAtConstantSpeed Continue;
```

```
When Ax1.Mechanism.Slope.IsDecelerating Continue;
When Ax1.Mechanism.Slope.IsStopped Continue;
```

34.2.2 Reverse()

Die Funktion kehrt die Richtung einer laufenden Endlosbewegung um bewegt die Achse mit der zuletzt eingestellten Geschwindigkeit in die neue Richtung. Für die Umkehr der Bewegungsrichtung wird die zuletzt eingestellte Beschleunigungsrampe angewandt.

ⓘ Info

Anwendbarkeit der `Reverse()`-Funktion

– Der Befehl kann nicht auf eine laufende Punkt-zu-Punkt-Bewegung (`MovePtp`) angewandt werden.

– Wird der Befehl erteilt, wenn sich die Achse im Zustand **freigegeben** befindet, erfolgt keine Aktion.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.MoveVelocity.Reverse();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während der Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	–		Freigegeben
Läuft	√	Läuft	Läuft
Unterbrochen	X		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

34.2.3 NewSpeed()

Die Funktion übergibt einen neuen Wert für die Geschwindigkeit einer Endlospositionierung einer MCX-Achse. Dabei gilt:

– Der Wert muss > `0` sein.

– Ist die Achse bereits in Bewegung (Zustand **läuft**), so wird der neue Geschwindigkeitswert sofort übernommen.

  Der Übergang von aktueller zu neuer Geschwindigkeit bei einer laufenden Achse, erfolgt mit den für die Beschleunigungsrampen eingestellten Werten. Dies ist auch dann der Fall, wenn sich die Geschwindigkeit verringert.

– Ist die Achse im Stillstand (Zustand **freigegeben**), wird der Wert erst bei der nächsten Positionierung angewendet.

– Der Rampentyp kann während einer Positionierung nicht geändert werden.
– Bei allen Übergängen ist die Ruckbegrenzung aktiv, falls der Rampentyp *sin²* eingestellt ist.

ⓘ Info

Achse zum Stillstand bringen

Um eine Achse zum Stillstand zu bringen sind ausschließlich die Befehle `MoveHalt` oder `Stop` zu verwenden. Es ist **nicht** möglich eine Achse anzuhalten, indem ein Geschwindigkeitswert von `0` übergeben wird.

Übergabeparameter

Parameter	Datentyp	Bedeutung	Verhalten beim Funktionsaufruf im Zustand Läuft
Speed	Double	Definiert die Geschwindigkeit, mit der die Achse positioniert werden soll, solange sie sich nicht in den Rampen (Beschleunigung, Verzögerung) befindet.	Wird eine neue Positionierung gestartet, während die vorherige noch läuft (Zustand Läuft), so wird der neue Geschwindigkeitswert unter Berücksichtigung der Maximalwerte für Beschleunigung, Verzögerung und Ruck sofort übernommen. Die Änderung der Geschwindigkeit erfolgt mit den Parametern der Beschleunigungsrampe.

Syntax

```
AX1.MoveVelocity.NewSpeed(2000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

34.2.4 NewAcceleration()

Die Funktion übergibt einen neuen Wert für die maximale Beschleunigung einer Endlospositionierung einer MCX-Achse.

Dabei gilt:

– Eine aktive Beschleunigungsrampe wird nicht verändert.
– Der neue Wert wird bei der nächsten Endlospositionierung angewendet.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Acceleration	Double (optional)	Definiert die maximale Beschleunigung, die in der Startrampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.

Syntax

```
AX1.MoveVelocity.NewAcceleration(2000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

34.2.5 NewDeceleration()

Die Funktion übergibt einen neuen Wert für die maximale Verzögerung einer Endlospositionierung einer MCX-Achse.

Der Wert wird beim nächsten `MoveHalt.Start()`-Befehl oder der nächsten Positionierung verwendet.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Deceleration	Double (optional)	Definiert die maximale Verzögerung, die in der Stopprampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.

Syntax

```
AX1.MoveVelocity.NewDeceleration(2000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

34.3 Eigenschaften von MCAxis.MoveVelocity

Die Funktionalität bietet folgende Eigenschaften:

– Direction

34.3.1 Direction

Beschreibung

Mithilfe der Eigenschaft `Direction` lesen Sie die momentan gültige Richtung einer MCX-Achse oder setzen einen neuen Wert für diese Achse. Dieser wird bei der nächsten Endlos-Positionierung wirksam. Dies ist auch dann der Fall, wenn die Änderung während einer laufenden Positionierung übergeben wird.

Datentyp	Bedeutung
Directions (optional)	Richtung der Endlosbewegung.
Directions.Positive	Die Achse wird in positive Richtung bewegt.
Directions.Negative	Die Achse wird in negative Richtung bewegt.

Syntax

```
AX_Richtung := AX1.MoveVelocity.Direction;
AX1.MoveVelocity.Direction := Directions.Negative;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen

Ausgangszustand	Erlaubt	Endzustand
Fehler: freigegeben	√	Fehler: freigegeben
Fehler: gesperrt	√	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

34.4 Beispieltask MCAxis.MoveVelocity

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Var
    run: bool; //Hilfsflag zum Weiterschalten im Programm
    Direction_positive: bool;
End_Var;

Task Sample4 Autorun

Direction_positive := true;

loop
    //Das Flag "run" dient als Tipp-Taster. Solange es true ist, fährt die
Achse
When run Continue;

    // Einschalten des Servo-Verstärkers
    If AX1.State.IsDisabled AND AX1.Drive.IsReadyForOperation Then
        AX1.Power.Enable();
    End_if;
    When AX1.State.IsEnabled Continue;

    //Das Flag "Direction_positiv" dient zur Vorgabe der Bewegungsrichtung
    If Direction_positive Then
        AX1.MoveVelocity.Start(Directions.Positive,100,1000,5000);
    Else
        AX1.MoveVelocity.Start(Directions.Negative,100,1000,5000);
    End_If;

    //Wenn angehalten werden soll (run = false), MoveHalt aufrufen
    When not run Continue;
        AX1.MoveHalt.Start(MCAxisHaltModes.Normal,);
    //Wenn Achse im Stillstand, wieder abschalten;
    When AX1.State.IsEnabled Continue;
    AX1.Power.Disable();
    When AX1.State.IsDisabled Continue;
end_loop;

End_Task;
```

35 MCAxis.Position

Mit der Funktionalität **MCAxis.Position** können Sie auf Positionsinformationen einer MCX-Achse zugreifen.

35.1 Syntax für MCAxis.Position

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Position.<Zugriff auf Wert>;
```

35.2 Funktionen von MCAxis.Position

Die Funktionalität bietet folgende Funktionen:

Funktionen für MCX-Achsen mit Absolutwertgeber

- AbsEncOverflowHandler.Activate()
- AbsEncOverflowHandler.Deactivate()

35.2.1 AbsEncOverflowHandler.Activate()

ⓘ Info

MCX-Achsen mit Absolutwertgebern

Für MCX-Achsen mit Absolutwertgebern gibt es die Funktionalität der Überlaufbehandlung, für deren Verwendung die hier beschriebenen Funktionen und Eigenschaften der Motion-API 2.x vorgesehen sind.

Beachten Sie in diesem Zusammenhang die Information in der JetSym-Hilfe oder die gesonderten Informationsblätter *Konsistenter Betrieb einer Achse mit Multiturn-Absolutwertgeber (mit nicht binärem Getriebe) bei Überlauf der Geberposition*.

Beachten Sie, dass für die Anwendung dieser Befehle die Speicherfunktion im NVRAM aktiviert sein muss.

Mit der Funktion `AbsEncOverflowHandler.Activate()` aktivieren Sie die Überlaufbehandlung der MCX-Achse mit Absolutwertgeber.

Im Normalfall wird diese Funktion nur einmal bei der Inbetriebnahme der Achse aktiviert. Die Einstellung erfolgt im Motion-Setup der Achse über die Seite Referenzierung. Deshalb wird dieser Befehl nur benötigt, wenn die Funktion nachträglich über das STX-Programm aktiviert werden muss.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.Position.AbsEncOverflowHandler.Activate();
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

35.2.2 AbsEncOverflowHandler.Deactivate()

Info

MCX-Achsen mit Absolutwertgebern

Für MCX-Achsen mit Absolutwertgebern gibt es die Funktionalität der Überlaufbehandlung, für deren Verwendung die hier beschriebenen Funktionen und Eigenschaften der Motion-API 2.x vorgesehen sind.

Beachten Sie in diesem Zusammenhang die Information in der JetSym-Hilfe oder die gesonderten Informationsblätter *Konsistenter Betrieb einer Achse mit Multiturn-Absolutwertgeber (mit nicht binärem Getriebe) bei Überlauf der Geberposition*.

Beachten Sie, dass für die Anwendung dieser Befehle die Speicherfunktion im NVRAM aktiviert sein muss.

Mit der Funktion `AbsEncOverflowHandler.Deactivate()` deaktivieren Sie die Überlaufbehandlung der MCX-Achse mit Absolutwertgeber.

Im Normalfall muss diese Funktion nicht mehr deaktiviert werden, außer sie wurde versehentlich aktiviert. Die Einstellung erfolgt im Motion-Setup der Achse über die Seite Referenzierung.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.Position.AbsEncOverflowHandler.Deactivate();
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

35.3 Eigenschaften von MCAxis.Position

Die Funktionalität bietet folgende Eigenschaften:

Allgemeine Eigenschaften für MCX-Achsen

- Target
- Actual
- Setpoint
- EncoderPosition
- ActualTrackingError
- TrackingErrorLimit
- TrailingIndicatorMax
- TrailingIndicatorMin

Eigenschaften speziell für MCX-Achsen mit Absolutwertgeber

- AbsEncOverflowHandler.Deactivate() [▶ 197]
- AbsEncOverflowHandler.Activate() [▶ 196]
- AbsEncOverflowHandler.IsActive

35.3.1 Target

Beschreibung

Datentyp	Bedeutung
Double	Liest die Zielposition der MCX-Achse [in der Anwender-Position-Einheit]. Dabei gilt: – Befindet sich die Achse in einer Bewegung, so ist dies die Position, bei der die Positionierung beendet wird. – Ist die Achse im Stillstand oder Fehler, so ist dies die Zielposition des zuletzt ausgeführten Positionierbefehls.

Syntax

```
Zielposition := AX1.Position.Target;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

35.3.2 Actual

Beschreibung

Datentyp	Bedeutung
Double	Liest die Position, an der sich die MCX-Achse momentan befindet (Ist-Position) [in der Anwender-Position-Einheit].

Syntax

```
Istposition := AX1.Position.Actual;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

35.3.3 Setpoint

Beschreibung

Datentyp	Bedeutung
Double	Liest die aktuell berechnete Sollposition der MCX-Achse [in der Anwender-Position-Einheit]. Die Sollposition wird von der MCX in jedem MCX-Zyklus auf der Grundlage der Positionierdaten neu berechnet.

Syntax

```
Sollposition := AX1.Position.Setpoint;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

35.3.4 EncoderPosition

Beschreibung

Datentyp	Bedeutung
Double	Liest die Position, die der Positionssensor der MCX-Achse ausgibt [in der Anwender-Position-Einheit]. Dieser Wert unterscheidet sich von der Ist-Position durch Offsets, die beim Referenzieren oder gegebenenfalls bei Moduloüberläufen berechnet werden. Ausgabe in Anwender-Positions-Einheiten unter Berücksichtigung der mechanischen Übersetzung.

Syntax

```
EncoderRohWert := AX1.Position.EncoderPosition;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

35.3.5 AcutalTrackingError

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert für den momentanen Schleppfehler [in der Anwender-Position-Einheit]. Der Schleppfehler wird im Antrieb durch die Subtraktion von Soll- und Ist-Position in jedem Zyklus des Positionsreglers berechnet. Er ist ein Maß dafür, wie gut der Antrieb den Sollwertvorgaben der MCX folgen kann.

Syntax

```
Schleppfehler := AX1.Position.ActualTrackingError;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

35.3.6 TrailingIndicatorMax

Beschreibung

Datentyp	Bedeutung
Double	Liest den höchsten von der Regelung erkannten positiven Schleppfehler seit dem letzten Zurücksetzen der Messung [in der Anwender-Position-Einheit]. Die Schleppzeigermessung kann dazu verwendet werden, langsame Veränderungen des Schleppfehlers bei immer gleichen Bewegungen zu erkennen (siehe auch `AX1.Drive.Control.Position.ResetTrailingIndicators()`).

Syntax

```
MaxPosSchleppfehler := AX1.Position.TrailingIndicatorMax;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

35.3.7 TrailingIndicatorMin

Beschreibung

Datentyp	Bedeutung
Double	Liest den höchsten von der Regelung erkannten negativen Schleppfehler seit dem letzten Zurücksetzen der Messung [in der Anwender-Position-Einheit]. Die Schleppzeigermessung kann dazu verwendet werden, langsame Veränderungen des Schleppfehlers bei immer gleichen Bewegungen zu erkennen (siehe auch `AX1.Drive.Control.Position.ResetTrailingIndicators()`).

Syntax

```
MinNegSchleppfehler := AX1.Position.TrailingIndicatorMin;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

35.3.8 AbsEncOverflowHandler.IsActive

ⓘ Info

MCX-Achsen mit Absolutwertgebern

Für MCX-Achsen mit Absolutwertgebern gibt es die Funktionalität der Überlaufbehandlung, für deren Verwendung die hier beschriebenen Funktionen und Eigenschaften der Motion-API 2.x vorgesehen sind.

Beachten Sie in diesem Zusammenhang die Information in der JetSym-Hilfe oder die gesonderten Informationsblätter *Konsistenter Betrieb einer Achse mit Multiturn-Absolutwertgeber (mit nicht binärem Getriebe) bei Überlauf der Geberposition*.

Beachten Sie, dass für die Anwendung dieser Befehle die Speicherfunktion im NVRAM aktiviert sein muss.

Übergabeparameter

Datentyp	Bedeutung
Bool	Fragt ab, ob die Funktion zur Überlaufbehandlung für diese MCX-Achse aktiviert wurde.

Syntax

```
If AX1.Position.AbsEncOverflowHandler.IsActive then …
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

36 MCAxis.Power

Mit der Funktionalität **MCAxis.Power** können Sie die Regelung und damit die Drehmomenterzeugung einer MCX-Achse freigeben oder sperren.

36.1 Syntax für MCAxis.Power

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Power.<Funktion>(optional <Parameter>;
```

36.2 Funktionen von MCAxis.Power

Die Funktionalität bietet folgende Funktionen:

– Enable()
– Disable()
– Quickstop()

36.2.1 Enable()

Die Funktion aktiviert die Regler des Servoverstärkers und gibt die Spannung auf den Motor frei. Durch die Spannungsfreigabe wird der Antrieb in die Lage versetzt, ein Drehmoment an der Motorwelle zu erzeugen. Wenn die Regelung richtig eingestellt ist und das Drehmoment ausreicht, dann wird die Achse an der augenblicklichen Position gehalten, da der Positionssollwert dem Positionsistwert vorbesetzt wird.

Freigabevoraussetzungen

Eine reale Achse kann nur freigegeben werden, wenn der Servoverstärker betriebsbreit ist. Fragen Sie den Zustand des Servoverstärkers wie folgt ab:

```
If AX1.Drive.IsReadyForOperation Then
```

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.Power.Enable();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Freigegeben
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Der Zustandsübergang kann nicht in einem MCX-Zyklus abgearbeitet werden. Fragen Sie deshalb das Erreichen des neuen Zustands im Programm ab, bevor Sie weitere Befehle für die Achse aufrufen.

```
When Ax1.State.IsEnabled Continue;
```

36.2.2 Disable()

Die Funktion sperrt den Servoverstärker für die Drehmomenterzeugung (Impulssperre). Der Motor wird sofort stromlos und trudelt aus, falls keine Bremse aktiv wird.

Freigabevoraussetzungen

Eine reale Achse kann nur freigegeben werden, wenn der Servoverstärker betriebsbreit ist. Fragen Sie den Zustand des Servoverstärkers wie folgt ab:

```
If AX1.Drive.IsReadyForOperation Then
```

Übergabeparameter

Parameter	Datentyp	Bedeutung
DisableModes	MCPowerDisableMode (optional)	Modus für die Ausführung der `Disable()`-Funktion
	MCPowerDisableMode.Forced (Default)	Die Impulssperre wirkt in jedem Fall sofort. Dieser Modus ist auch aktiv, wenn kein Parameter angegeben wird.
	MCPowerDisableMode.StateCheck	Die Impulssperre wird nur dann wirksam, wenn sich die Achse im Betriebszustand *Freigegeben* befindet. In allen anderen Zuständen wird ein Bedienfehler ausgegeben.

Syntax

```
Ax1.Power.Disable();
Ax1.Power.Disable(MCPowerDisableModes.Forced);
Ax1.Power.Disable(MCPowerDisableModes.StateCheck);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Für den Aufruf ohne Parameter oder mit dem Parameter `MCPowerDisableModes.Forced` gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	–	Gesperrt
Freigegeben	√	Gesperrt
Läuft	√	Gesperrt
Unterbrochen	√	Gesperrt
Fehler: freigegeben	√	Fehler: gesperrt
Fehler: gesperrt	–	Fehler: gesperrt

Für den Aufruf mit dem Parameter `MCPowerDisableModes.StateCheck` gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	–	Gesperrt
Freigegeben	√	Gesperrt
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Der Zustandsübergang kann nicht in einem MCX-Zyklus abgearbeitet werden. Bevor Sie weitere Befehle für die Achse aufrufen, fragen Sie deshalb das Erreichen des neuen Zustands im Programm wie folgt ab:

```
When Ax1.State.IsDisabled Continue;
```

Hat sich die Achse vorher im Zustand *Fehler: freigegeben* befunden, verwenden Sie folgenden Befehl. Es werden die Positionspuffer geleert.

```
When AX1.State.IsDisabledError Continue;
```

36.2.3 QuickStop()

Die Funktion hält die Achse mit der Schnellhaltverzögerung an und sperrt den Servoverstärker nach Erreichen des Stillstands.

Freigabevoraussetzungen

Eine reale Achse kann nur freigegeben werden, wenn der Servoverstärker betriebsbreit ist. Fragen Sie den Zustand des Servoverstärkers wie folgt ab:

```
If AX1.Drive.IsReadyForOperation Then
```

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
Ax1.Power.QuickStop();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	–	Gesperrt
Freigegeben	√	Gesperrt
Läuft	√	Gesperrt
Unterbrochen	√	Gesperrt
Fehler: freigegeben	√	Fehler: gesperrt
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Der Zustandsübergang kann nicht in einem MCX-Zyklus abgearbeitet werden. Bevor Sie weitere Befehle für die Achse aufrufen, fragen Sie deshalb das Erreichen des neuen Zustands im Programm wie folgt ab:

```
When Ax1.State.IsDisabled Continue;
```

36.3 Beispieltask MCAxis.Power

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Task Sample1 Autorun
    //Einschalten des Servoverstärkers
    If AX1.State.IsDisabled AND AX1.Drive.IsReadyForOperation Then
        AX1.Power.Enable();
    End_if;
    When AX1.State.IsEnabled Continue;

    //hier kann ein Prozess mit der Achse stehen oder andere Tasks bewegen
die Achse
    //…

    // Achse sperrren
    If AX1.State.IsRunning Then    // falls die Achse noch in Bewegung ist
        AX1.Power.QuickStop();    // mit der Schnellhaltrampe anhalten
    Else
        AX1.Power.Disable();    // sonst gleich sperren
    End_if;
    When AX1.State.IsDisabled Continue;

End_Task;
```

37 MCAxis.Settings

Mit der Funktionalität **MCAxis.Settings** können Sie alle Parameter einer MCX-Achse gemeinsam anpassen oder abspeichern.

Parameter bei aktivem Speichervorgang nicht anpassen

Das Speichern der Parameter in eine Datei ist ein Vorgang, der im Hintergrund abgearbeitet wird. D. h. er erstreckt sich über mehrere Zyklen der MCX. Werden während des Abspeicherns Parameterwerte geändert, ist nicht vorhersehbar, ob der alte oder neue Wert in die Datei geschrieben wird.

Funktionsweise der Parameterverwaltung

Parameter, die im Motion-Setup von JetSym gespeichert werden, werden in einer Textdatei auf dem PC und, auf Nachfrage, auch im Dateisystem der Steuerung (Unterordner Motion) abgelegt. Der Dateiname der Parameterdatei auf der Steuerung ist *MCAxis_X.ini*, wobei das *X* für die Nummer der Achs-ID steht. Die Achs-ID wird auf der Seite Achsparameter im Motion-Setup der MCX-Achse angezeigt. Außerdem zeigt der Tooltipp im Hardware-Baum die Achs-ID an, wenn der Mauszeiger auf der Achse liegt.)

Beim Booten der Steuerung liest der MCX-Kernel die Daten der *MCAxis_X.ini*-Datei und verwendet diese für die betreffende Achse. Ist eine solche Datei auf der Steuerung vorhanden, muss also keine Parametrierung mehr vorgenommen werden. Ist keine entsprechende Datei für die Achse vorhanden, werden die Parameter mit Default-Werten geladen.

Im Normalfall ist keine Anpassung an der Datei aus dem STX-Programm heraus erforderlich.

In ganz wenigen Ausnahmefällen kann es jedoch sinnvoll sein, die Parameterdatei auf der Steuerung neu einzulesen oder die Parameter abzuspeichern. Beim Abspeichern muss jedoch bedacht werden, dass die im JetSym-Projekt gespeicherte Parameter-Datei dann unter Umständen nicht mehr mit den auf der Maschine wirksamen Parametern übereinstimmt.

37.1 Syntax für MCAxis.Settings

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.Settings.<Funktion>(optional <Parameter>;
```

37.2 Funktionen von MCAxis.Settings

Die Funktionalität bietet folgende Funktionen:

– SaveToFile()
– LoadFromFile()

37.2.1 SaveToFile()

Übergabeparameter

Parameter	Datentyp	Bedeutung
Suffix	Integer (optional)	Aktiviert das Abspeichern der Parameter der MCX-Achse in eine Datei im Unterordner Motion im Dateisystem der Steuerung. Abhängig vom Übergabeparameter werden unterschiedliche Dateien erzeugt bzw. überschrieben – Parameter `0` oder kein Parameter: Die Parameterdatei *MCaxis_X.ini* für diese Achse wird mit den aktuellen Werten erstellt oder überschrieben. – Parameter ungleich `0`: Die Parameterdatei *MCAxis_X_TmpY.ini* wird mit den aktuellen Werten erstellt oder überschrieben. Dabei ist *Y* gleich dem Parameter `Suffix`.

Syntax

```
AX1.Settings.SaveToFile();    // überschreibt (oder erzeugt) die Datei MCAxis_1.ini (wenn AX1 die Achs-ID 1 hat)
AX1.Settings.SaveToFile(3);    // erzeugt die Datei MCAxis_1_Tmp3.ini (wenn AX1 die Achs-ID 1 hat)
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

37.2.2 LoadFromFile()

Übergabeparameter

Parameter	Datentyp	Beschreibung
Suffix	Integer (optional)	Aktiviert das Übernehmen der Parameter der MCX-Achse aus einer Datei im Unterordner Motion im Dateisystem der Steuerung. Abhängig vom Übergabeparameter werden unterschiedliche Dateien gelesen. – Parameter `0` oder kein Parameter: Die Daten aus der Datei *MCaxis_X.ini* werden für die betreffende Achse gelesen und verwendet. – Parameter ungleich 0: Die Daten aus der Datei *MCaxis_TmpY.ini* werden für die betreffende Achse gelesen und verwendet. Dabei ist *Y* gleich dem Parameter `Suffix`.

Syntax

```
AX1.Settings.LoadFromFile();    //Liest die Datei MCAxis_1.ini (wenn AX1 die Achs-ID 1 hat)
AX1.Settings.LoadFromFile(3);    //Liest die Datei MCAxis_1_Tmp3.ini (wenn AX1 die Achs-ID 1 hat)
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben

Ausgangszustand	Erlaubt	Endzustand
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

38 MCAxis.State

Mit der Funktionalität **MCAxis.State** können Sie verschiedene Informationen einer MCX-Achse abfragen, wie z. B. Betriebsart, Bewegungsart etc.

Grundlegende und weiterführende Informationen

Beachten Sie in diesem Zusammenhang ebenso folgende Informationen:

– Für Technologieverbunde den Abschnitt MCAxis.State.Techno [▶ 221];
– Für Zustandsübergänge den Abschnitt MCAxis.State.Transitions [▶ 228];
– Für eine detaillierte Beschreibung des Aufbaus des MCX-Kerns die JetSym–Onlinehilfe (JetSym Motion und MotionControl > JetSym MotionControl > MotionControl-Grundlagen > Bedienkonzept > Überblick: Bedienkonzept)
– Für grundlegende Informationen die Motion Control Application Note Nr. 54 (MCX-Motion-Control-Grundlagen).

Betriebsart

Die Betriebsart (`OperatingMode`) gibt an, auf welche Weise die MCX-Achse grundsätzlich arbeitet. Es gibt fünf Betriebsarten:

Betriebsart	Bedeutung
`Inactive` (inaktiv)	Die Achse kann keine Befehle entgegen nehmen, z. B. weil sie von einem Verbund verwendet wird.
`Normal` (normal)	Die Achse ist verwendbar. Verfahrbefehle werden verdrängend ausgeführt (siehe MCAxis.Buffer [▶ 31]).
`Buffered` (gepuffert)	Die Achse ist verwendbar. Verfahrbefehle werden gepuffert ausgeführt (siehe MCAxis.Buffer [▶ 31]).
`External` (extern)	Die Achse ist eine externe Achse. Es ist nur die aktuelle Position auslesbar. Sie kann nur als Leitachse in einem Technologieverbund verwendet werden.
`Error` (Fehler)	Es liegt ein Bedienfehler vor. Der Betriebszustand bleibt erhalten, Es können jedoch keine neuen Funktionen und Parameter übergeben werden.

Tab. 2: Betriebsarten von MCX-Achsen

Betriebszustand

Eine MCX-Achse befindet sich zu jedem Zeitpunkt in einem, und nur einem, klar definierten Betriebszustand (`OperatingState`). Ist die Achse beispielsweise im Betriebszustand **Läuft**, dann ist der Betriebszustand **Freigegeben** nicht aktiv, obwohl sie natürlich eine Impulsfreigabe hat.

In einem bestimmten Zustand ist immer nur eine Untermenge aller Funktionsaufrufe zulässig. Das Zustandsmodell für die Achse sieht folgendermaßen aus:

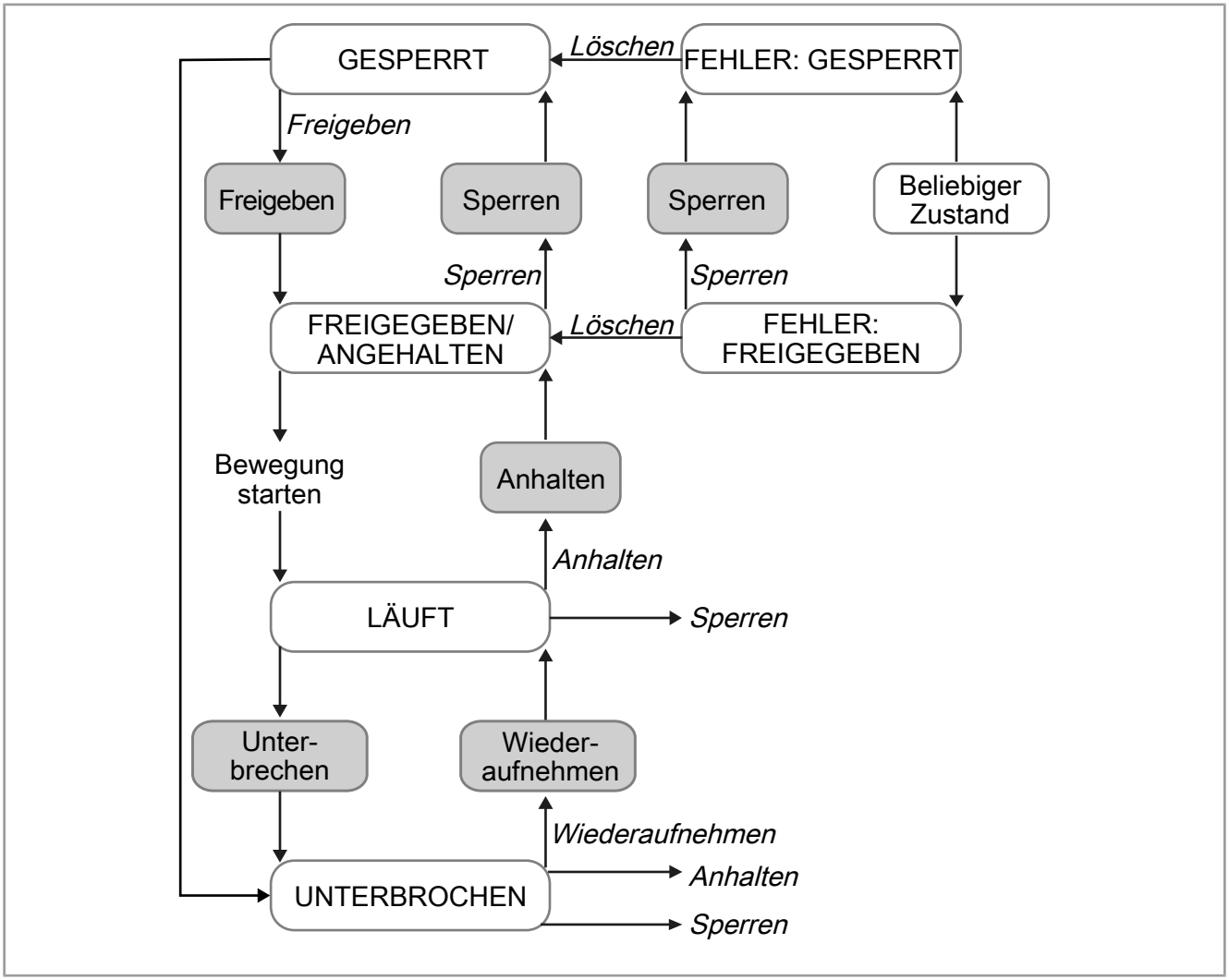


Abb. 3: Zustandsamodell für MCX-Achsen

Betriebszustand	Bedeutung
Inactive (Inaktiv)	Die Achse kann keine Befehle entgegen nehmen, etwa weil sie von einem Verbund verwendet wird.
Disabled (Gesperrt)	– Die Regler des Servoverstärkers sind geöffnet (keine Impulsfreigabe). – Es wird kein Drehmoment erzeugt.
Enabled (Freigegeben)	– Die Regler sind geschlossen (Impulsfreigabe). – Der Sollwertinterpolator befindet sich im Stillstand (der Vorlaufpuffer ist leer).
Running (Läuft)	Die Achse ist in Bewegung; d. h. der Sollwertinterpolator gibt Bewegungssollwerte an den Servoverstärker weiter.
Suspended (Unterbrochen)	– Entspricht dem Betriebszustand **Freigegeben** mit der Ausnahme, dass sämtliche Vorlaufpuffer gefüllt bleiben, aber gesperrt sind. Eine mit `MCAxis.State.Transitions.Suspend()` unterbrochene Bewegung kann über den Befehl `MCAxis.State.Transitions.Resume()` (Zustandsübergang wiederaufnehmen) fortgesetzt werden. – Im gepufferten Betrieb ist es auch möglich, die Abarbeitung des Puffers zu unterbrechen (`MCAxis.Buffer.Suspend()`) und wiederaufzunehmen (`MCAxis.Buffer.Resume()`).
EnabledError (Fehler: freigegeben)	Es ist ein Fehler aufgetreten. Nach der Fehlerreaktion bleibt die Achse weiterhin in Regelung (Fehlerstufe 4).

Betriebszustand	Bedeutung
DisabledError (Fehler: gesperrt)	Es ist ein Fehler aufgetreten. Nach der Fehlerreaktion sperrt die Achse die Regelung (Fehlerstufen 5 und 6).

Tab. 3: Betriebszustände von MCX-Achsen

Bewegungsart

Die Bewegungsart (`TypeOfMotion`) gibt an, welche Art der Bewegung der Sollwertgenerator für die MCX-Achse aktuell erzeugt.

Die Abfrage der Bewegungsart ist beispielsweise in einem Überwachungstask interessant, in dem der Prozess nicht bekannt ist. Ausgehend von der aktiven Bewegungsart vor dem Abschalten ist es möglich, das Wiederanlaufen korrekt zu steuern.

Betriebszustand	Bedeutung
Keiner (`MoveNone`)	Der Sollwertgenerator gibt Stillstand aus, weil keine Bewegung angefordert ist.
Positionierung (`MovePtp`)	Es wird eine Positionierung (mit programmiertem Ziel) ausgeführt.
Endlosbewegung (`MoveVelocity`)	Es wird eine Endlospositionierung ausgeführt: Start mit Beschleunigungsrampe, Fahrt mit programmierter Geschwindigkeit.
Referenzfahrt (`MoveHome`)	Es wird eine Referenzfahrt ausgeführt.
Einkoppeln aktiv (`MoveCoupling`) [1)]	Die Achse ist Folgeachse in einem Technologieverbund. Der Einkoppelvorgang wird ausgeführt.
Eingekoppelt (`MoveCoupled`) [1)]	Die Achse ist Folgeachse in einem Technologieverbund. Sie folgt der Leitachse entsprechend der programmierten Koppelbewegung (elektrisches Getriebe, Kurvenscheibe)
Auskoppeln aktiv (`MoveDecoupling`) [1)]	Die Achse ist Folgeachse in einem Technologieverbund. Der Auskoppelvorgang wird ausgeführt.
Ausgekoppelt (`MoveDecoupled`) [1)]	Die Achse ist Folgeachse in einem Technologieverbund. Sie ist nicht mit der Leitachse verbunden.
Positionierung, überlagert (`MovePtpSuperPose`) [1)]	Die Achse ist Folgeachse in einem Technologieverbund. Sie befindet sich in einer überlagerten Positionierung (siehe MCTechno.MovePtp.StartSuperPose() [▶ 492]).
Endlos-Bewegung, überlagert (`MoveVelocitySuperPose`) [1)]	Die Achse ist Folgeachse in einem Technologieverbund. Sie befindet sich in einer überlagerten Endlosbewegung (siehe MCTechno.MovePtp.StartSuperPose() [▶ 492]).

[1)] Es wird empfohlen, diese Abfragen über den Technologieverbund auszuführen, in dem die MCX-Achse als Folgeachse aktiv ist.

38.1 Syntax für MCAxis.State

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.State.<Zugriff auf Wert>
```

38.2 Eigenschaften von MCAxis.State

Die Funktionalität bietet folgende Eigenschaften:

- Value
- IsDisabled
- IsEnabled
- IsRunning
- IsSupended
- IsDisabledError
- IsEnabledError
- IsInactive
- OperatingMode
- TypeOfMotion

38.2.1 Value

Beschreibung

Datentyp	Bedeutung
OperatingStates	Liest den Betriebszustand, in dem sich die Achse befindet (für eine detaillierte Beschreibung siehe Abschnitt *Betriebszustand* in MCAxis.State [▶ 208]).
OperatingStates.Inactive	Inaktiv
OperatingStates.Disabled	Gesperrt
OperatingStates.Enabled	Freigegeben
OperatingStates.Running	Läuft
OperatingStates.Suspended	Unterbrochen
OperatingStates.EnabledError	Fehler: freigegeben
OperatingStates.DisabledError	Fehler: gesperrt

Syntax

```
OpState := AX1.State.Value;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

38.2.2 IsDisabled

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse im Betriebszustand **Gesperrt** befindet.

Syntax

```
When AX1.State.IsDisabled Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

38.2.3 IsEnabled

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse im Betriebszustand **Freigegeben** befindet.

Syntax

```
When AX1.State.IsEnabled Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

38.2.4 IsRunning

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse im Betriebszustand **Läuft** befindet.

Syntax

```
When AX1.State.IsRunning Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

38.2.5 IsSuspended

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse im Betriebszustand **Unterbrochen** befindet.

Syntax

```
When AX1.State.IsSuspended Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

38.2.6 IsDisabledError

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse im Betriebszustand **Fehler: gesperrt** befindet.

Syntax

```
When AX1.State.IsDisabledError Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

38.2.7 IsEnabledError

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse im Betriebszustand **Fehler: freigegeben** befindet.

Syntax

```
When AX1.State.IsEnabledError Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

38.2.8 IsInactive

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse im Betriebszustand **Inaktiv** befindet.

Syntax

```
When AX1.State.IsInactive Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

38.2.9 OperatingMode.Value

Beschreibung

Datentyp	Bedeutung
Typ MCOperatingModes	Liest die Betriebsart, in der sich die Achse befindet (für eine detaillierte Beschreibung siehe Abschnitt *Betriebsart* in MCAxis.State [▶ 208]).
MCOperatingModes.Inactive	Die Achse ist inaktiv, etwa weil sie von einem Verbund verwaltet wird.

Datentyp	Bedeutung
MCOperatingModes.Normal	Die Achse ist im nichtgepufferten Betrieb. Dies ist der meistverwendete Modus.
MCOperatingModes.Buffered	Die Achse ist im gepufferten Betrieb. Diese Betriebsart wird normalerweise nur von Achsen eines Bahnverbundes verwendet.
MCOperatingModes.External	Die Achse ist eine externe Achse. Sie wertet lediglich die Ist-Position des zweiten Encoders einer realen (internen) Achse aus.
MCOperatingModes.Error	Die Achse befindet sich in einem Bedienfehler.

Syntax

```
OpModeAchse := AX1.State.OperatingMode.Value;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

OperatingMode.IsInInactiveMode

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Betriebsart *Inaktiv* befindet.

Syntax

```
When AX1.State.OperatingMode.IsInInactiveMode Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

OperatingMode.IsInNormalMode

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Betriebsart *Normal* befindet.

Syntax

```
When AX1.State.OperatingMode.IsInNormalMode Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

OperatingMode.IsInBufferedMode

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Betriebsart *Gepuffert* befindet.

Syntax

```
When AX1.State.OperatingMode.IsInBufferedMode Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

OperatingMode.IsInExternalMode

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Betriebsart *Extern* befindet.

Syntax

```
When AX1.State.OperatingMode.IsInExternalMode Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

OperatingMode.IsInErrorMode

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Betriebsart *Fehler* befindet.

Syntax

```
When AX1.State.OperatingMode.IsInErrorMode Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

38.2.10 TypeOfMotion.Value

Beschreibung

Datentyp	Bedeutung
Typ MCAxisTypesOfMotion	Liest die Bewegungsart, in der sich die Achse befindet (für eine detaillierte Beschreibung siehe Abschnitt *Bewegungsart* in MCAxis.State [▶ 208]).
MCAxisTypesOfMotion.MoveNone	Keine
MCAxisTypesOfMotion.MovePtp	Positionierung
MCAxisTypesOfMotion.MoveVelocity	Endlosbewegung
MCAxisTypesOfMotion.MoveHome	Referenzfahrt
MCAxisTypesOfMotion.MoveCoupling	Einkoppeln aktiv
MCAxisTypesOfMotion.MoveCoupled	Eingekoppelt
MCAxisTypesOfMotion.MoveDecoupling	Auskoppeln aktiv
MCAxisTypesOfMotion.MoveDecoupled	Ausgekoppelt
MCAxisTypesOfMotion.MovePtpSuperPose	Positionierung (überlagert)
MCAxisTypesOfMotion.MoveVelocitySuperPose	Endlosbewegung (überlagert)

Syntax

```
TypeOfMotion := AX1.State. TypeOfMotion.Value;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveNone

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Bewegungsart `Keine` befindet.

Syntax

```
When AX1.State.TypeOfMotion.IsMoveNone Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMovePtp

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Bewegungsart `Positionierung` befindet.

Syntax

```
When AX1.State.TypeOfMotion.IsMovePtp Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveVelocity

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Bewegungsart `Endlosbewegung` befindet.

Syntax

```
When AX1.State.TypeOfMotion.IsMoveVelocity Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveHome

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Bewegungsart `Referenzfahrt` befindet.

Syntax

```
When AX1.State.TypeOfMotion.IsMoveHome Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveCoupling

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Bewegungsart `Einkoppeln` befindet.

Syntax

```
When AX1.State.TypeOfMotion.IsMoveCoupling Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveCoupled

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Bewegungsart `Eingekoppelt` befindet.

Syntax

```
When AX1.State.TypeOfMotion.IsMoveCoupled Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveDecoupling

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Bewegungsart `Auskoppeln` befindet.

Syntax

```
When AX1.State.TypeOfMotion.IsMoveDecoupling Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveDeoupled

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Bewegungsart `Ausgekoppelt` befindet.

Syntax

```
When AX1.State.TypeOfMotion.IsMoveDecoupled Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMovePtpSuperPose

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Bewegungsart `Positionierung (überlagert)` befindet.

Syntax

```
When AX1.State.TypeOfMotion.IsMovePtpSuperPose Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveVelocitySuperPose

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Bewegungsart `Endlosbewegung (überlagert)` befindet.

Syntax

```
When AX1.State.TypeOfMotion.IsMoveVelocitySuperPose Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

38.3 Beispieltask MCAxis.State

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Task Sample2 Autorun

    //Einschalten des Servoverstärkers
    If AX1.State.IsDisabled AND AX1.Drive.IsReadyForOperation Then
        AX1.Power.Enable();
    End_if;
    When AX1.State.IsEnabled Continue;

    //Hier kann ein Prozess mit der Achse stehen, oder andere Tasks bewegen
die Achse
    //…

    // Achse sperren
    If AX1.State.IsRunning Then //falls die Achse noch in Bewegung ist
        AX1.Power.QuickStop(); //mit der Schnellhaltrampe anhalten
    Else
        AX1.Power.Disable(); //sonst gleich sperren
    End if;
    When AX1.State.IsDisabled Continue
End_Task;
```

39 MCAxis.State.Techno

Mit der Funktionalität **MCAxis.State.Techno** können Sie den Status für die Kopplung einer MCX-Achse, die Folgeachse in einem Technologieverbund (`techno`) ist, lesen.

Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 49 - 56**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

39.1 Syntax für MCAxis.State.Techno

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.State.Techno.<Zugriff auf Wert>;
```

39.2 Eigenschaften von MCAxis.State.Techno

Die Funktionalität bietet folgende Eigenschaften:

- TechnoID
- SyncState
- IsDecoupled
- IsDecoupling
- IsCoupling
- IsCoupled
- SyncPosition
- Cam.CamID
- Cam.CamSegmentID
- Cam.MasterOffset
- Cam.FollowerOffset
- Cam.MasterFactor
- Cam.FollowerFactor

39.2.1 TechnoID

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Nummer des aktiven Techno-Verbunds, an den die MCX-Achse eingebunden ist. Ist der Techno-Verbund nicht aktiv wird der Wert `0` zurückgegeben.

Syntax

```
AktiverTechno := AX1.State.Techno.TechnoID;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

39.2.2 SyncState

Beschreibung

Datentyp	Bedeutung
MCSyncStates	Liest den Status der Kopplung der MCX-Achse im aktiven Technologieverbund. Ist die MCX-Achse an keinem aktiven Technologieverbund beteiligt ist, so wird `Decoupled` zurückgegeben. Ist die MCX-Achse Leitachse in einem aktiven Technologieverbund, so wird immer `Coupled` zurückgegeben.
MCSyncStates.Decoupled	Die MCX-Achse ist ausgekoppelt.
MCSyncStates.Decoupling	Die MCX-Achse befindet sich im Auskoppelvorgang.
MCSyncStates.Coupling	Die MCX-Achse befindet sich im Einkoppelvorgang.
MCSyncStates.Coupled	Die MCX-Achse ist eingekoppelt.

Syntax

```
If AX1.State.Techno.SyncState == MCSyncStates.Coupled then …
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

39.2.3 IsDecoupled

Beschreibung

Datentyp	Bedeutung
Bool	Liest, ob die MCX-Achse in einem aktiven Technologieverbund ausgekoppelt ist. Ist die MCX-Achse an keinem aktiven Technologieverbund beteiligt, ist `IsDecoupled` ebenfalls `true`.

Syntax

```
If AX1.State.Techno.IsDecoupled then …
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

39.2.4 IsDecoupling

Beschreibung

Datentyp	Bedeutung
Bool	Liest, ob sich die MCX-Achse in einem aktiven Technologieverbund im Auskoppelvorgang befindet.

Syntax

```
If AX1.State.Techno.IsDecoupling then …
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

39.2.5 IsCoupling

Beschreibung

Datentyp	Bedeutung
Bool	Liest, ob sich die MCX-Achse in einem aktiven Technologieverbund im Einkoppelvorgang befindet.

Syntax

```
If AX1.State.Techno.IsCoupling then …
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

39.2.6 IsCoupled

Beschreibung

Datentyp	Bedeutung
Bool	Liest, ob die MCX-Achse in einem aktiven Technologieverbund eingekoppelt ist. Ist die MCX-Achse Leitachse in einem aktiven Technologieverbund, ist `IsCoupled` ebenfalls `true`.

Syntax

```
If AX1.State.Techno.IsCoupled then …
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

39.2.7 SyncPosition

Beschreibung

Datentyp	Bedeutung
Double	Liest die Position, an der eine ausgekoppelte Folgeachse stehen müsste, um ohne Versatz einkoppelbereit zu sein. Die Abfrage dieser Eigenschaft ist nur im ausgekoppelten Zustand von Interesse. Sie können damit erkennen, welche Folgeachsposition zu der augenblicklichen Leitachsposition gehört, um dann beispielsweise die Folgeachse an die Synchronposition zu bewegen, wo sie ohne Versatz eingekoppelt werden kann. Ist die MCX-Achse eingekoppelt oder fungiert als Leitachse ist, entspricht die `SyncPosition` der Sollposition der Folgeachse.

Syntax

```
Sync_Pos := AX1.State.Techno.SyncPosition;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

39.2.8 Cam.CamID

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie Informationen zur Funktionsweise einer Kurvenscheibe in einem Technologieverbund den **Motion Control Application Notes (APN) Nr. 52** und **56**.

Die Eigenschaft `Cam` beinhaltet einige weitere Eigenschaften, über die Sie Informationen zu einer aktiven Kurvenscheibe auslesen können, an der die MCX-Achse beteiligt ist.

Die Abfragen sind nur für Achsen sinnvoll, die Folgeachsen in einem Technologieverbund sind:

– Ist die MCX-Achse an keiner aktiven Kurvenscheibe beteiligt, so wird bei allen lesenden Zugriffen auf diese Eigenschaften der Wert `0` zurückgegeben.

– Auch wenn die MCX-Achse die Leitachse im Technologieverbund ist, wird bei allen lesenden Zugriffen auf diese Eigenschaften der Wert `0` zurückgegeben.

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Nummer der aktiven Kurvenscheibe, an der die MCX-Achse beteiligt ist. Ist der betreffende Technologieverbund nicht aktiv wird der Wert `0` zurückgegeben.

Syntax

```
AktiveCAM := AX1.State.Techno.Cam.CamID;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

39.2.9 Cam.CamSegmentID

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie Informationen zur Funktionsweise einer Kurvenscheibe in einem Technologieverbund den **Motion Control Application Notes (APN) Nr. 52** und **56**.

Die Eigenschaft `Cam` beinhaltet einige weitere Eigenschaften, über die Sie Informationen zu einer aktiven Kurvenscheibe auslesen können, an der die MCX-Achse beteiligt ist.

Die Abfragen sind nur für Achsen sinnvoll, die Folgeachsen in einem Technologieverbund sind:

– Ist die MCX-Achse an keiner aktiven Kurvenscheibe beteiligt, so wird bei allen lesenden Zugriffen auf diese Eigenschaften der Wert `0` zurückgegeben.

– Auch wenn die MCX-Achse die Leitachse im Technologieverbund ist, wird bei allen lesenden Zugriffen auf diese Eigenschaften der Wert `0` zurückgegeben.

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Nummer des aktiven Segments der aktiven Kurvenscheibe, an dem die MCX-Achse beteiligt ist. Ist der Technologieverbund nicht aktiv, wird der Wert `0` zurückgegeben.

Syntax

```
AktiveCamSegment := AX1.State.Techno.Cam.CamSegementID;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

39.2.10 Cam.MasterOffset

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie Informationen zur Funktionsweise einer Kurvenscheibe in einem Technologieverbund den **Motion Control Application Notes (APN) Nr. 52** und **56**.

Die Eigenschaft `Cam` beinhaltet einige weitere Eigenschaften, über die Sie Informationen zu einer aktiven Kurvenscheibe auslesen können, an der die MCX-Achse beteiligt ist.

Die Abfragen sind nur für Achsen sinnvoll, die Folgeachsen in einem Technologieverbund sind:

– Ist die MCX-Achse an keiner aktiven Kurvenscheibe beteiligt, so wird bei allen lesenden Zugriffen auf diese Eigenschaften der Wert `0` zurückgegeben.

– Auch wenn die MCX-Achse die Leitachse im Technologieverbund ist, wird bei allen lesenden Zugriffen auf diese Eigenschaften der Wert `0` zurückgegeben.

Beschreibung

Datentyp	Bedeutung
Double	Liest den Offset der Leitachse, der beim Start der Kurvenscheibenkopplung für die aktive Kurvenscheibe, an der die MCX-Achse beteiligt ist, übergeben wurde. Ist der Technologieverbund oder die Kurvenscheibe nicht aktiv, oder ist die MCX-Achse eine Leitachse, wird der Wert `0` zurückgegeben.

Syntax

```
AktiverLeitOffset := AX1.State.Techno.Cam.MasterOffset;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

39.2.11 Cam.FollowerOffset

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie Informationen zur Funktionsweise einer Kurvenscheibe in einem Technologieverbund den **Motion Control Application Notes (APN) Nr. 52** und **56**.

Die Eigenschaft `Cam` beinhaltet einige weitere Eigenschaften, über die Sie Informationen zu einer aktiven Kurvenscheibe auslesen können, an der die MCX-Achse beteiligt ist.

Die Abfragen sind nur für Achsen sinnvoll, die Folgeachsen in einem Technologieverbund sind:

– Ist die MCX-Achse an keiner aktiven Kurvenscheibe beteiligt, so wird bei allen lesenden Zugriffen auf diese Eigenschaften der Wert `0` zurückgegeben.

– Auch wenn die MCX-Achse die Leitachse im Technologieverbund ist, wird bei allen lesenden Zugriffen auf diese Eigenschaften der Wert `0` zurückgegeben.

Beschreibung

Datentyp	Bedeutung
Double	Liest den Offset der Folgeachse, der beim Start der Kurvenscheibenkopplung für die aktive Kurvenscheibe, an der die MCX-Achse beteiligt ist, übergeben wurde. Ist der Technologieverbund oder die Kurvenscheibe nicht aktiv, oder ist die MCX-Achse eine Leitachse, wird der Wert `0` zurückgegeben.

Syntax

```
AktiverFolgeOffset := AX1.State.Techno.Cam.FollowerOffset;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

39.2.12 Cam.MasterFactor

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie Informationen zur Funktionsweise einer Kurvenscheibe in einem Technologieverbund den **Motion Control Application Notes (APN) Nr. 52** und **56**.

Die Eigenschaft `Cam` beinhaltet einige weitere Eigenschaften, über die Sie Informationen zu einer aktiven Kurvenscheibe auslesen können, an der die MCX-Achse beteiligt ist.

Die Abfragen sind nur für Achsen sinnvoll, die Folgeachsen in einem Technologieverbund sind:

– Ist die MCX-Achse an keiner aktiven Kurvenscheibe beteiligt, so wird bei allen lesenden Zugriffen auf diese Eigenschaften der Wert `0` zurückgegeben.

– Auch wenn die MCX-Achse die Leitachse im Technologieverbund ist, wird bei allen lesenden Zugriffen auf diese Eigenschaften der Wert `0` zurückgegeben.

Beschreibung

Datentyp	Bedeutung
Double	Liest den Faktor der Leitachse, der beim Start der Kurvenscheibenkopplung für die aktive Kurvenscheibe, an der die MCX-Achse beteiligt ist, übergeben wurde. Ist der Technologieverbund oder die Kurvenscheibe nicht aktiv, oder ist die MCX-Achse eine Leitachse, wird der Wert `0` zurückgegeben.

Syntax

```
AktiverLeitFaktor := AX1.State.Techno.Cam.MasterFactor;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

39.2.13 Cam.FollowerFactor

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie Informationen zur Funktionsweise einer Kurvenscheibe in einem Technologieverbund den **Motion Control Application Notes (APN) Nr. 52** und **56**.

Die Eigenschaft `Cam` beinhaltet einige weitere Eigenschaften, über die Sie Informationen zu einer aktiven Kurvenscheibe auslesen können, an der die MCX-Achse beteiligt ist.

Die Abfragen sind nur für Achsen sinnvoll, die Folgeachsen in einem Technologieverbund sind:

– Ist die MCX-Achse an keiner aktiven Kurvenscheibe beteiligt, so wird bei allen lesenden Zugriffen auf diese Eigenschaften der Wert `0` zurückgegeben.

– Auch wenn die MCX-Achse die Leitachse im Technologieverbund ist, wird bei allen lesenden Zugriffen auf diese Eigenschaften der Wert `0` zurückgegeben.

Beschreibung

Datentyp	Bedeutung
Double	Liest den Faktor der Folgeachse, der beim Start der Kurvenscheibenkopplung für die aktive Kurvenscheibe, an der die MCX-Achse beteiligt ist, übergeben wurde. Ist der Technologieverbund oder die Kurvenscheibe nicht aktiv, oder ist die MCX-Achse eine Leitachse, wird der Wert `0` zurückgegeben.

Syntax

```
AktiverFolgeFaktor := AX1.State.Techno.Cam.FollowerFactor;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

40 MCAxis.State.Transitions

Mit der Funktionalität **MCAxis.State.Transitions** können Sie für spezielle Anwendungen Zustandsübergänge einer MCX-Achse steuern.

In der Regel werden die Zustandsübergänge einer MCX-Achse durch die Bewegungsbefehle und das Verhalten der Achsbewegung automatisch ausgelöst. Nur in Ausnahmefällen – z. B. zur Reaktion im Fehlerfall – ist die direkte Steuerung der Zustandsverwaltung erforderlich. Durch die Funktionalität **MCAxis.State.Transitions**, wie auch `Diagnostics.ClearErrors()`, können Betriebszustandsübergänge explizit ausgelöst werden.

ⓘ Info

Wirksamwerden der Befehle

Auch im gepufferten Betrieb werden die Funktionen und Eigenschaften von **MCAxis.State.Transitions** sofort wirksam. Sämtliche Vorlaufpuffer werden geleert.

Mit der Funktionalität **MCAxis.State** können Sie verschiedene Informationen einer MCX-Achse abfragen, wie z. B. Betriebsart, Bewegungsart etc.

Grundlegende und weiterführende Informationen

Beachten Sie in diesem Zusammenhang ebenso folgende Informationen:

– Für Technologieverbunde den Abschnitt MCAxis.State [▶ 208];
– Für Zustandsübergänge den Abschnitt MCAxis.State.Techno [▶ 221];
– Für eine detaillierte Beschreibung des Aufbaus des MCX-Kerns die JetSym–Onlinehilfe (JetSym Motion und MotionControl > JetSym MotionControl > MotionControl-Grundlagen > Bedienkonzept > Überblick: Bedienkonzept)
– Für grundlegende Informationen die Motion Control Application Note Nr. 54 (MCX-Motion-Control-Grundlagen).

Betriebsart

Die Betriebsart (`OperatingMode`) gibt an, auf welche Weise die MCX-Achse grundsätzlich arbeitet. Es gibt fünf Betriebsarten:

Betriebsart	Bedeutung
`Inactive` (inaktiv)	Die Achse kann keine Befehle entgegen nehmen, z. B. weil sie von einem Verbund verwendet wird.
`Normal` (normal)	Die Achse ist verwendbar. Verfahrbefehle werden verdrängend ausgeführt (siehe MCAxis.Buffer [▶ 31]).
`Buffered` (gepuffert)	Die Achse ist verwendbar. Verfahrbefehle werden gepuffert ausgeführt (siehe MCAxis.Buffer [▶ 31]).
`External` (extern)	Die Achse ist eine externe Achse. Es ist nur die aktuelle Position auslesbar. Sie kann nur als Leitachse in einem Technologieverbund verwendet werden.
`Error` (Fehler)	Es liegt ein Bedienfehler vor. Der Betriebszustand bleibt erhalten, Es können jedoch keine neuen Funktionen und Parameter übergeben werden.

Tab. 4: Betriebsarten von MCX-Achsen

Betriebszustand

Eine MCX-Achse befindet sich zu jedem Zeitpunkt in einem, und nur einem, klar definierten Betriebszustand (`OperatingState`). Ist die Achse beispielsweise im Betriebszustand **Läuft**, dann ist der Betriebszustand **Freigegeben** nicht aktiv, obwohl sie natürlich eine Impulsfreigabe hat.

In einem bestimmten Zustand ist immer nur eine Untermenge aller Funktionsaufrufe zulässig. Das Zustandsmodell für die Achse sieht folgendermaßen aus:

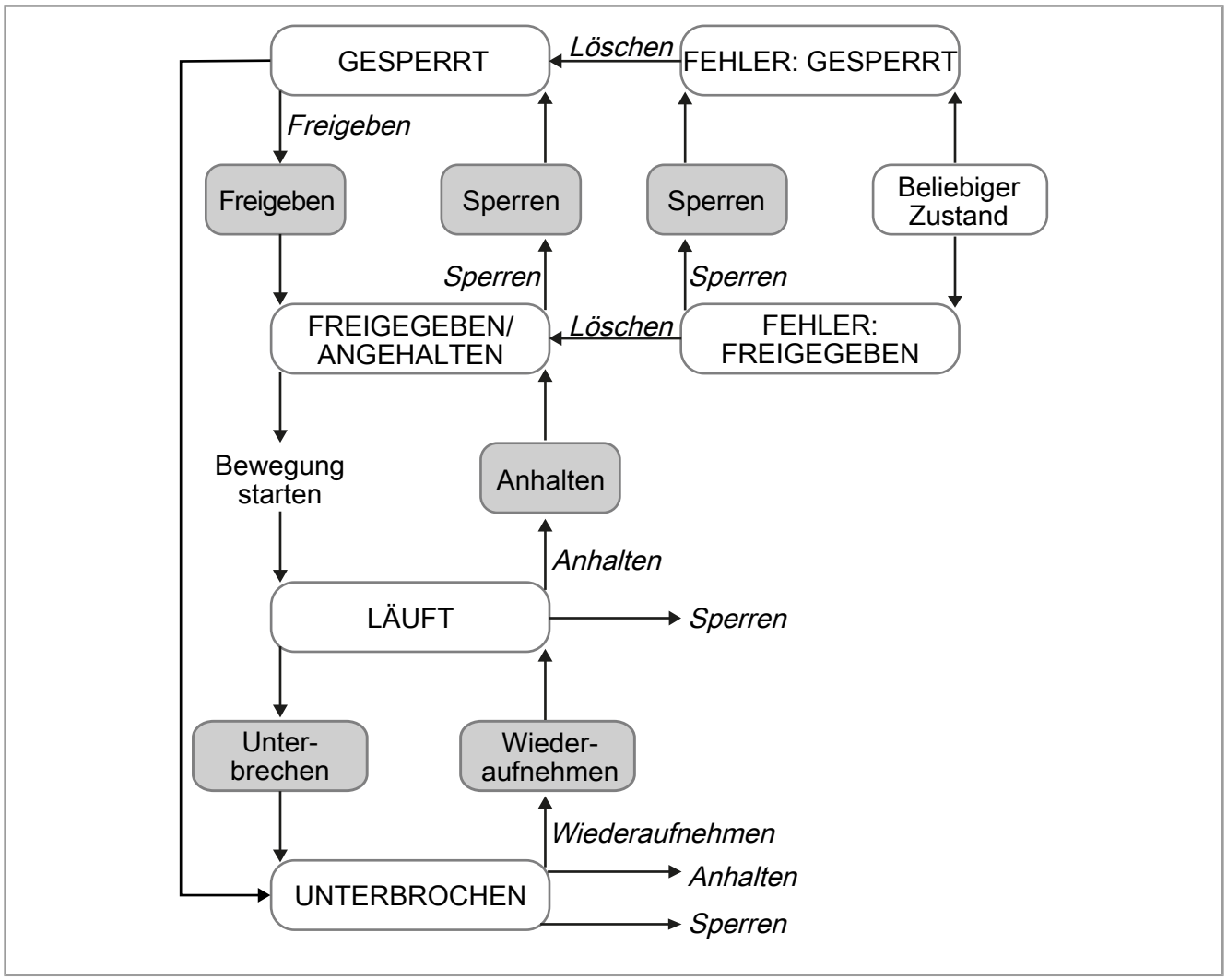


Abb. 4: Zustandsamodell für MCX-Achsen

Betriebszustand	Bedeutung
Inactive (Inaktiv)	Die Achse kann keine Befehle entgegen nehmen, etwa weil sie von einem Verbund verwendet wird.
Disabled (Gesperrt)	– Die Regler des Servoverstärkers sind geöffnet (keine Impulsfreigabe). – Es wird kein Drehmoment erzeugt.
Enabled (Freigegeben)	– Die Regler sind geschlossen (Impulsfreigabe). – Der Sollwertinterpolator befindet sich im Stillstand (der Vorlaufpuffer ist leer).
Running (Läuft)	Die Achse ist in Bewegung; d. h. der Sollwertinterpolator gibt Bewegungssollwerte an den Servoverstärker weiter.
Suspended (Unterbrochen)	– Entspricht dem Betriebszustand **Freigegeben** mit der Ausnahme, dass sämtliche Vorlaufpuffer gefüllt bleiben, aber gesperrt sind. Eine mit `MCAxis.State.Transitions.Suspend()` unterbrochene Bewegung kann über den Befehl `MCAxis.State.Transitions.Resume()` (Zustandsübergang wiederaufnehmen) fortgesetzt werden. – Im gepufferten Betrieb ist es auch möglich, die Abarbeitung des Puffers zu unterbrechen (`MCAxis.Buffer.Suspend()`) und wiederaufzunehmen (`MCAxis.Buffer.Resume()`).

Betriebszustand	Bedeutung
EnabledError (Fehler: freigegeben)	Es ist ein Fehler aufgetreten. Nach der Fehlerreaktion bleibt die Achse weiterhin in Regelung (Fehlerstufe 4).
DisabledError (Fehler: gesperrt)	Es ist ein Fehler aufgetreten. Nach der Fehlerreaktion sperrt die Achse die Regelung (Fehlerstufen 5 und 6).

Tab. 5: Betriebszustände von MCX-Achsen

Bewegungsart

Die Bewegungsart (`TypeOfMotion`) gibt an, welche Art der Bewegung der Sollwertgenerator für die MCX-Achse aktuell erzeugt.

Die Abfrage der Bewegungsart ist beispielsweise in einem Überwachungstask interessant, in dem der Prozess nicht bekannt ist. Ausgehend von der aktiven Bewegungsart vor dem Abschalten ist es möglich, das Wiederanlaufen korrekt zu steuern.

Betriebszustand	Bedeutung
Keiner (`MoveNone`)	Der Sollwertgenerator gibt Stillstand aus, weil keine Bewegung angefordert ist.
Positionierung (`MovePtp`)	Es wird eine Positionierung (mit programmiertem Ziel) ausgeführt.
Endlosbewegung (`MoveVelocity`)	Es wird eine Endlospositionierung ausgeführt: Start mit Beschleunigungsrampe, Fahrt mit programmierter Geschwindigkeit.
Referenzfahrt (`MoveHome`)	Es wird eine Referenzfahrt ausgeführt.
Einkoppeln aktiv (`MoveCoupling`) [1)]	Die Achse ist Folgeachse in einem Technologieverbund. Der Einkoppelvorgang wird ausgeführt.
Eingekoppelt (`MoveCoupled`) [1)]	Die Achse ist Folgeachse in einem Technologieverbund. Sie folgt der Leitachse entsprechend der programmierten Koppelbewegung (elektrisches Getriebe, Kurvenscheibe)
Auskoppeln aktiv (`MoveDecoupling`) [1)]	Die Achse ist Folgeachse in einem Technologieverbund. Der Auskoppelvorgang wird ausgeführt.
Ausgekoppelt (`MoveDecoupled`) [1)]	Die Achse ist Folgeachse in einem Technologieverbund. Sie ist nicht mit der Leitachse verbunden.
Positionierung, überlagert (`MovePtpSuperPose`) [1)]	Die Achse ist Folgeachse in einem Technologieverbund. Sie befindet sich in einer überlagerten Positionierung (siehe MCTechno.MovePtp.StartSuperPose() [▶ 492]).
Endlos-Bewegung, überlagert (`MoveVelocitySuperPose`) [1)]	Die Achse ist Folgeachse in einem Technologieverbund. Sie befindet sich in einer überlagerten Endlosbewegung (siehe MCTechno.MovePtp.StartSuperPose() [▶ 492]).

[1)] Es wird empfohlen, diese Abfragen über den Technologieverbund auszuführen, in dem die MCX-Achse als Folgeachse aktiv ist.

40.1 Syntax für MCAxis.State.Transitions

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.State.Transitions.<Funktion>(optional <Parameter>);
```

40.2 Funktionen von MCAxis.State.Transitions

Die Funktionalität bietet folgende Funktionen:

- Enable()
- Disable()
- ErrorDisable()
- ErrorStop()
- ErrorLockUserInput()
- Suspend()
- Resume()
- UserPositionReset()

40.2.1 Enable()

Die Funktion `State.Transistions.Enable()` ist identisch zur Funktion `MCAxis.Power.Enable()`. Eine detaillierte Beschreibung der Funktion finden Sie im Abschnitt MCAxis.Power.Enable() [▶ 201].

Syntax

```
AX1.State.Transitions.Enable();
```

40.2.2 Disable()

Die Funktion führt je nach übergebenem Parameter (siehe nachfolgende Tabelle) dieselbe Aktion aus, wie die Befehle `MCAxis.Power.Disable()` und `MCAxis.Power.Qickstop()`. Entnehmen Sie die Details den Beschreibungen zu MCAxis.Power [▶ 201].

Übergabeparameter

Parameter	Datentyp	Bedeutung
Mode	MCStateTransitionDisableModes	Modus für die Ausführung der `Disable()`-Funktion
	MCStateTransitionDisableModes.ImmediatePulseBlocking	Führt die gleiche Funktion aus wie der Befehl `MCAxis.Power.Disable()` bzw. `MCAxis MCAxis.Power.Disable(MCPowerDisableModes.Forced)`
	MCStateTransitionDisableModes.VelocityControlSlope	Führt die gleiche Funktion aus wie der Befehl `AX1.Power.QuickStop()`.

Syntax

```
AX1.State.Transitions.Disable(MCStateTransitionDisableModes.ImmediatePulse-
Blocking);
AX1.State.Transitions.Disable(MCStateTransitionDisableModes.VelocityControl-
Slope);
```

40.2.3 ErrorDisable()

Mit der Funktion lösen Sie einen Fehler in der MCX-Achse aus, der zum Zustand **Fehler: gesperrt** führt. Die Fehlerreaktion kann eingestellt werden (siehe Eigenschaft ErrorDisableMode [▶ 236]). Im gepufferten Betrieb, wird der Vorlaufpuffer gelöscht.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.State.Transitions.ErrorDisable();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Fehler: gesperrt
Freigegeben	√	Fehler: gesperrt
Läuft	√	Fehler: gesperrt
Unterbrochen	√	Fehler: gesperrt
Fehler: freigegeben	√	Fehler: gesperrt
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

40.2.4 ErrorStop()

Mit der Funktion lösen Sie einen Fehler in der MCX-Achse aus, der zum Zustand **Fehler: freigegeben** führt. Die Fehlerreaktion kann eingestellt werden (siehe Eigenschaft ErrorStopMode [▶ 236]). Im gepufferten Betrieb, wird der Vorlaufpuffer gelöscht.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.State.Transitions.ErrorStop();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Fehler: freigegeben
Freigegeben	√	Fehler: freigegeben
Läuft	√	Fehler: freigegeben
Unterbrochen	√	Fehler: freigegeben
Fehler: freigegeben	-	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

40.2.5 ErrorLockUserInput()

Mit der Funktion lösen Sie einen Bedienfehler in der MCX-Achse aus Im gepufferten Betrieb, wird der Vorlaufpuffer nicht gelöscht.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.State.Transitions.ErrorLockUserInput();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

40.2.6 Suspend()

Mit der Funktion unterbrechen Sie eine laufende Positionierung. Das Anhalten geschieht entweder mit der zuletzt programmierten Verzögerungsrampe oder mit der konfigurierten Maximalverzögerung. Der Zustand nach dem Anhalten ist nicht *Freigegeben* sondern *Unterbrochen*. Eine Achse in diesem Zustand kann mit der Funktion `Resume()` an die ursprüngliche Zielposition gefahren werden.

Kann die MCX-Achse mit der vorgegebenen Verzögerung nicht mehr vor der programmierten Zielposition angehalten werden, so wird der Befehl ignoriert und die Positionierung zu Ende geführt. Der Zustand ist danach *Freigegeben* und nicht *Unterbrochen*. Im gepufferten Betrieb wird der Vorlaufpuffer nicht gelöscht.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Mode	MCStateTransitionStopModes	Modus für die Verzögerungsrampe der `Suspend()`-Funktion
	MCStateTransitionStopModes.ProgDeceleration	Verwendet die für die MCX-Achse zuletzt programmierte Verzögerung.
	MCStateTransitionStopModes.MaxDeceleration	Verwendet die für die MCX-Achse konfigurierte maximale Verzögerung.

Syntax

```
AX1.State.Transitions.Suspend(MCStateTransitionStopModes.ProgDeceleration);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	X	Freigegeben
Läuft	√	Unterbrochen
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

40.2.7 Resume()

Die Funktion führt eine mit der Funktion Suspend() [▶ 233] unterbrochene Positionierung fort. Es wird die ursprünglich vorgegebene Zielposition angefahren. Auch die Geschwindigkeits- und Beschleunigungswerte werden aus der ursprünglichen Positionierung übernommen.

Die Funktion `Resume()` wirkt sich nur aus, wenn sich die MCX-Achse im Zustand *Unterbrochen* befindet.

Im gepufferten Betrieb wird der Vorlaufpuffer nicht gelöscht.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.State.Transitions.Resume();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	X	Freigegeben
Läuft	–	Läuft
Unterbrochen	√	Läuft
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

40.2.8 UserPositionReset()

Die Funktion führt eine Spezialfunktion zum Zurücksetzen des Sollwertgenerators aus. **Der Aufruf dieser Funktion ist nur sinnvoll, wenn sich die Achse nicht mehr bewegt.**

Durch das Ausführen der Funktion wird jegliche laufende Positionierung abgebrochen und der Sollwert der MCX-Achspostion mit dem Istwert überschrieben. Auf diese Weise werden der Positionsschleppfehler und der Geschwindigkeitssollwert unmittelbar null, wodurch eine sich bewegende Achse sofort angehalten wird.

Ein typischer Anwendungsfall ist das Referenzieren auf einen Anschlag. In diesem Fall wird, mit verminderter Drehmomentbegrenzung, eine Positionierung auf eine Position ausgeführt, die sich sicher hinter dem Anschlag befindet. Wenn sich die Achse nicht mehr bewegt, wird die Funktion `UserPositionReset()` ausgeführt, um den Sollwertgenerator zum Stillstand auf die Anschlagsposition zu bringen. Danach wird mit `SetReference()` die Referenzposition festgelegt.

Auch nach dem Ablauf der Funktionalität MCAxis.Drive.TorqueShutOff [▶ 105] kommt die Funktion `UserPositionReset()` zum Einsatz.

Übergabeparemeter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.State.Transitions.Resume();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Freigegeben
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

40.3 Eigenschaften von MCAxis.State.Transitions

Die Funktionalität bietet folgende Eigenschaften:

– ErrorDisableMode

– ErrorStopMode

40.3.1 ErrorDisableMode

Beschreibung

Datentyp	Bedeutung
MCStateTransitionDisableModes	Fehlerreaktion, die beim Auslösen der Funktion `ErrorDisable()` verwendet wird.
MCStateTransitionDisableModes.VelocityControlSlope	Hält die Achse mit der Schnellhaltverzögerung an. Ist der Stillstand erreicht, wird der Servoverstärker gesperrt.
MCStateTransitionDisableModes.ImmediatePulseBlocking	Sperrt den Servoverstärker sofort für die Drehmomenterzeugung (Impulssperre).

Syntax

```
AX1.State.Transitions.ErrorDisableMode := MCStateTransitionDisableModes.ImmediatePulseBlocking;
Fehlerreaktion := AX1.State.Transitions.ErrorDisableMode;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Bei Lesezugriff der Eigenschaft wird die aktuell gültige Fehlerreaktion für den Befehl zurückgegeben.

40.3.2 ErrorStopMode

Beschreibung

Datentyp	Bedeutung
MCStateTransitionErrorStopModes	Fehlerreaktion, die bei der Funktion `ErrorStop()` verwendet wird
MCStateTransitionErrorStopModes.ProgDeceleration	Hält die Achse mit der programmierten Verzögerung an.
MCStateTransitionErrorStopModes.MaxDeceleration	Hält die Achse mit der maximalen Verzögerung an.
MCStateTransitionErrorStopModes.AtBlockEnd	Im gepufferten Betrieb: Hält die Achse am Satzende an.

Syntax

```
AX1.State.Transitions.ErrorStopMode := MCStateTransitionStopModes.MaxDeceleration;
Fehlerreaktion := AX1.State.Transitions.ErrorStopMode;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

41 MCAxis.Stop

Mit der Funktionalität **MCAxis.Stop** können Sie eine MCX-Achse, die sich in Bewegung befindet, mit einer definierten Rampe anhalten.

Im Unterschied zur Funktion `MoveHalt.Start()` werden bei `Stop` alle Vorlaufpuffer gelöscht.

Die Funktionalität besitzt keine Funktionen und Eigenschaften.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

41.1 Funktionen von MCAxis.Stop

`MCAxis.Stop` ist eine Funktion des Achsobjekts.

41.1.1 Stop()

Übergabeparameter

Beachten Sie folgendes Verhalten:

– Die Funktion führt den Anhaltebefehl unmittelbar aus. Der übergebene Parameter (siehe Tabelle) gibt die maximale Verzögerung dafür an.

– Wird kein Parameter übergeben, so wird der zuletzt übergebene `StopMode` verwendet.

– Dieser Parameter kann auch von den Befehlen `MCAxis.State.Transitions.Suspend(..)` oder `MCAxis.State.Transitions.Stop(..)` verändert werden.

Parameter	Datentyp	Bedeutung
StopModes	Enum StopModes (optional)	Es stehen zwei Verzögerungswerte zur Verfügung.
	StopModes.ProgDeceleration	Zum Anhalten wird die zuletzt für die Achse vorgegebene Verzögerung verwendet (Standard).
	StopModes.MaxDeceleration	Zum Anhalten wird die maximale Beschleunigung der Achse als Verzögerung verwendet. Diese wurde entweder bei der Inbetriebnahme im Motion Setup unter „Maximale Verzögerung“ vorgegeben oder mit dem Befehl AX1.Mechanism.Limitations.MaxAcceleration programmiert.

Syntax

```
AX1.Stop(StopModes.ProgDeceleration);
AX1.Stop(StopModes.MaxDeceleration);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während der Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√		Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	√		Freigegeben
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Positionierung stehen Ihnen verschiedene Informationen zur Verfügung.

– Der **Zustand der Achse** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder die Achse angehalten ist.

```
When Ax1.State.IsEnabled Continue; //Die Achse befindet sich in Lageregelung
und steht.
When Ax1.State.IsRunning Continue; //Der Sollwertgenerator erzeugt eine Be-
wegung.
```

– Der **Rampenstatus** informiert, in welchem Bewegungsabschnitt sich der Sollwertgenerator gerade befindet.

```
When Ax1.Mechanism.Slope.IsAccelerating Continue;
When Ax1.Mechanism.Slope.IsAtConstantSpeed Continue;
When Ax1.Mechanism.Slope.IsDecelerating Continue;
When Ax1.Mechanism.Slope.IsStopped Continue;
```

– Der **Zielfensterstatus** wird von dieser Funktion nicht beeinflusst.

41.2 Beispieltask MCAxis.Stop

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Var
    run: bool; //Hilfsflag zum Weiterschalten im Programm
End_Var;

Task Sample2 Autorun

loop
    When run Continue;

    // Einschalten des Servo-Verstärkers
    If AX1.State.IsDisabled AND AX1.Drive.IsReadyForOperation Then
        AX1.Power.Enable();
    End_if;
    When AX1.State.IsEnabled Continue;
```

```
    AX1.MoveHome.SetReference(0.0); //Referenzposition auf Einschaltposition
setzten
    When AX1.MoveHome.IsReferenceSet Continue;

    //Achse positionieren („Automatik-Ablauf")
    AX1.MovePtp.Start(AxisPositioningModes.AbsNor-
mal,1000,Speed1,1000,5000,0.1);
    //Wenn das Hilfsflag vom Anwender im Setup inzwischen gelöscht wurde,
Achse mit der programmierten Maximal-Verzögerung
    //anhalten und sperren
    //Dieses Ereignis könnte z. B. das Betätigen des Nothalt-Schalters sein
    When not run Continue;
    AX1.Stop(StopModes.MaxDeceleration);
    when AX1.State.IsEnabled Continue;
    AX1.Power.Disable();
    When AX1.State.IsDisabled Continue;

end_loop;

End_Task;
```

42 MCAxis.TouchProbe

Mit der Funktionaltät **MCAxis.TouchProbe** können Sie aufgrund des Signals eines Sensors sehr schnell die Position der MCX-Achse zu speichern.

Typische Anwendungsfälle:

– Positionskorrekturen aufgrund von Sensorsignalen (z. B. Druckmarkenkorrektur)
– Positionserkennung (z. B. unregelmäßig verteilte Produkte auf einem Förderband)
– Längenmessung von Produkten

Für diese Funktion stellt der Servoverstärker, je nach Gerätefamilie, einen oder zwei digitale Eingänge zur Verfügung, die eine besonders schnelle Reaktion erlauben. Da bis zu 2 Touch-Probe-Kanäle zur Verfügung stehen, ist diese Funktion als Array ausgeführt. In diesem Array sind nur die Werte 1 (erster Touch-Probe-Kanal) und 2 (zweiter Touch-Probe-Kanal) gültig.

Anwendungen mit 2 Touch-Probe-Kanälen

Wird bei einem Servoverstärker mit zwei Touch-Probe-Kanälen nur ein Kanal benötigt, sollten Sie immer den ersten Kanal verwenden, da die interne Verarbeitung beim ersten Kanal schneller ist als beim zweiten.

Die Position kann aufgrund der steigenden oder der fallenden Flanke des Sensorsignals gespeichert werden. Es ist auch möglich beide Flanken auszuwerten. In diesem Fall wird die Position zweimal gespeichert, einmal bei steigender und weiteres Mal bei fallender Flanke.

Single Mode und Continuous Mode

Im **Single Mode** wird die Funktion für eine Positionserfassung vom STX-Programm aus gestartet und die gespeicherte Position verarbeitet, bevor eine neue Erfassung gestartet wird. Dies ist in den meisten Fällen der sinnvolle Betrieb.

Im Single Mode kann ein Positionsfenster vorgegeben werden, das eingrenzt, in welchem Positionsbereich das Sensorsignal akzeptiert werden soll. Wird das Sensorsignal außerhalb des Positionsfensters erkannt, wird es ignoriert. Wird das ganze Positionsfenster durchfahren (in positive oder negative Richtung) und innerhalb des Positionsfensters kein Signal erkannt, so wird der Status „noEventInWindow“ eingenommen.

Im **Continuous Mode** ist die Funktion hingegen dauerhaft aktiv, wobei die gespeicherte Position bei jedem Sensorsignal überschrieben wird.

Einschränkungen

– Bei einer normalen Achse muss der Wert für die obere Positionsfenstergrenze immer größer sein als der Wert für die untere Positionsfenstergrenze.
– Ist bei einer Modulo-Achse der Wert für die untere Positionsfenstergrenze größer als der Wert für die obere Positionsfenstergrenze, so wird das Positionsfenster über den Nullpunkt (Modulo-Umbruch) gelegt.
– Die Positionsfenstergrenzen müssen innerhalb von oder identisch mit den Grenzen des Verfahrbereichs der MCX-Achse sein. Soll die Fenster-Funktion nicht verwendet werden, ist es sinnvoll, die Positionsfenstergrenzen gleich den Verfahrbereichsgrenzen zu setzen.
– Nur im *Single Mode*: Bei Verwendung der Konfiguration *Both edges* wechselt der Status von *wait* erst dann zu *both found*, wenn beide Flanken erkannt wurden. Es wird kein Zwischenstatus *rising edge found* oder *falling edge found* eingenommen.
– Im *Continuous Mode* ist die Positionsfenster-Überwachung nicht anwendbar.
– Im *Continuous Mode* ist die Konfiguration auf beide Flanken (*both edges*) nicht anwendbar.

Beispielhafte Flussdiagramme für verschiedene Konfigurationen

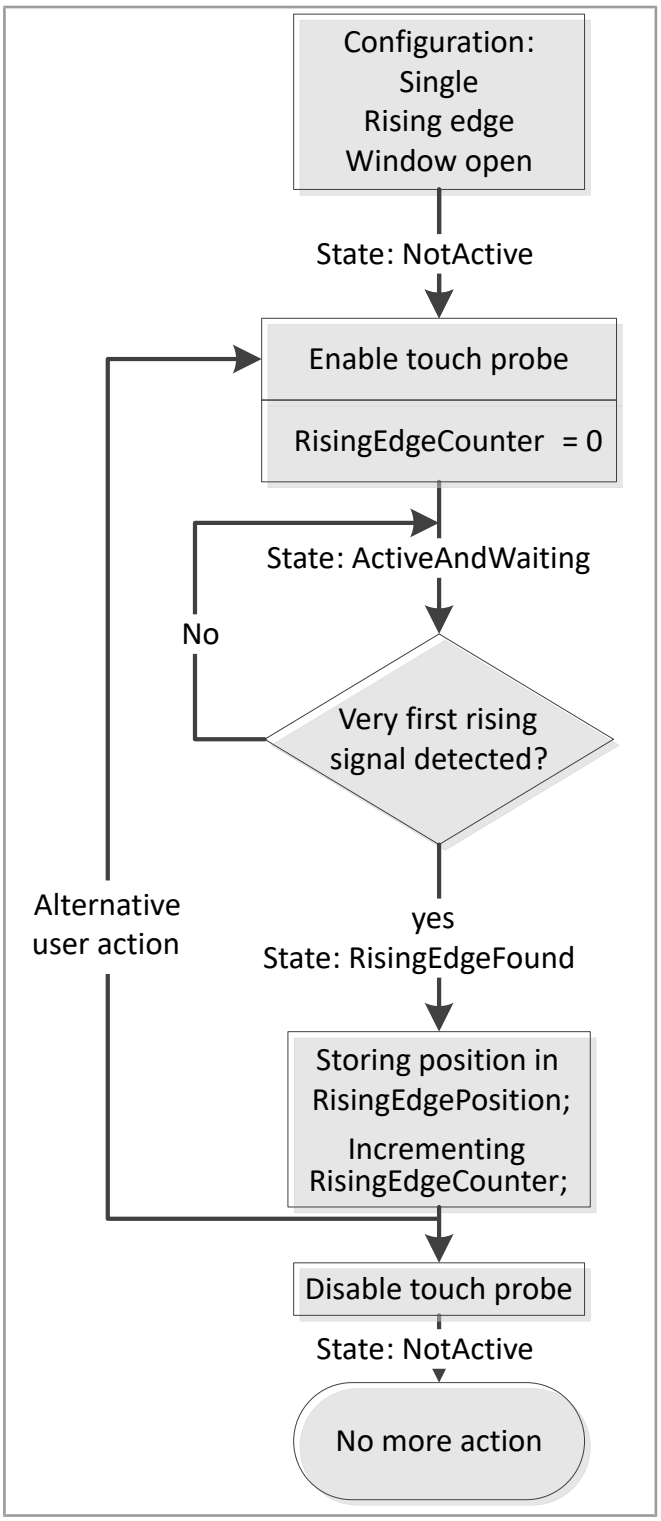


Abb. 5: *Flussdiagramm für Touch-Probe-Konfiguration Single Mode, steigende Flanke, offenes Positionsfenster*

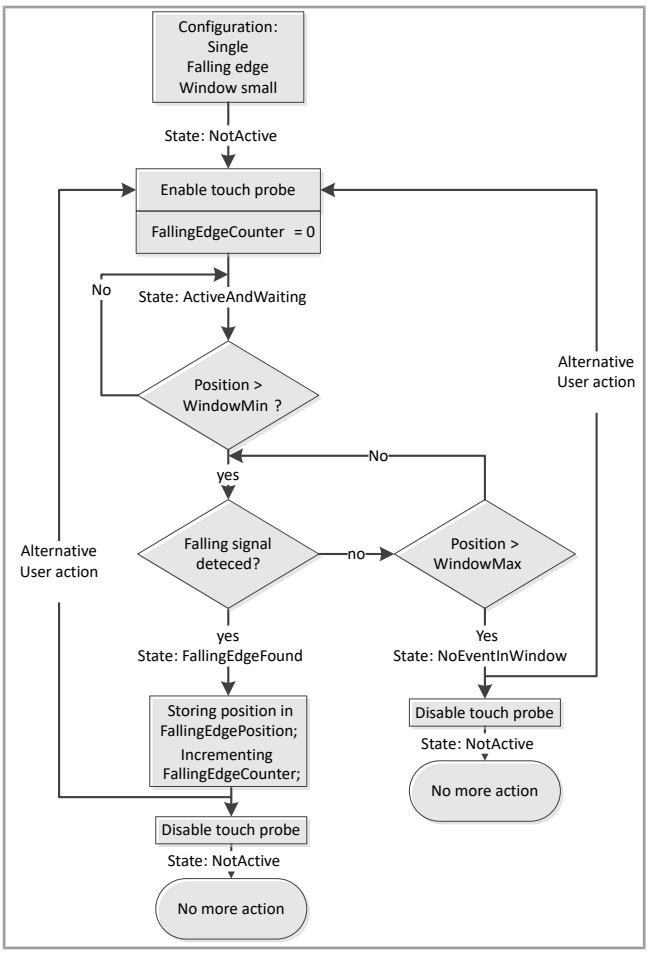


Abb. 6: *Flussdiagramm für Touch-Probe-Konfiguration Single Mode, fallende Flanke, kleines Positionsfenster*

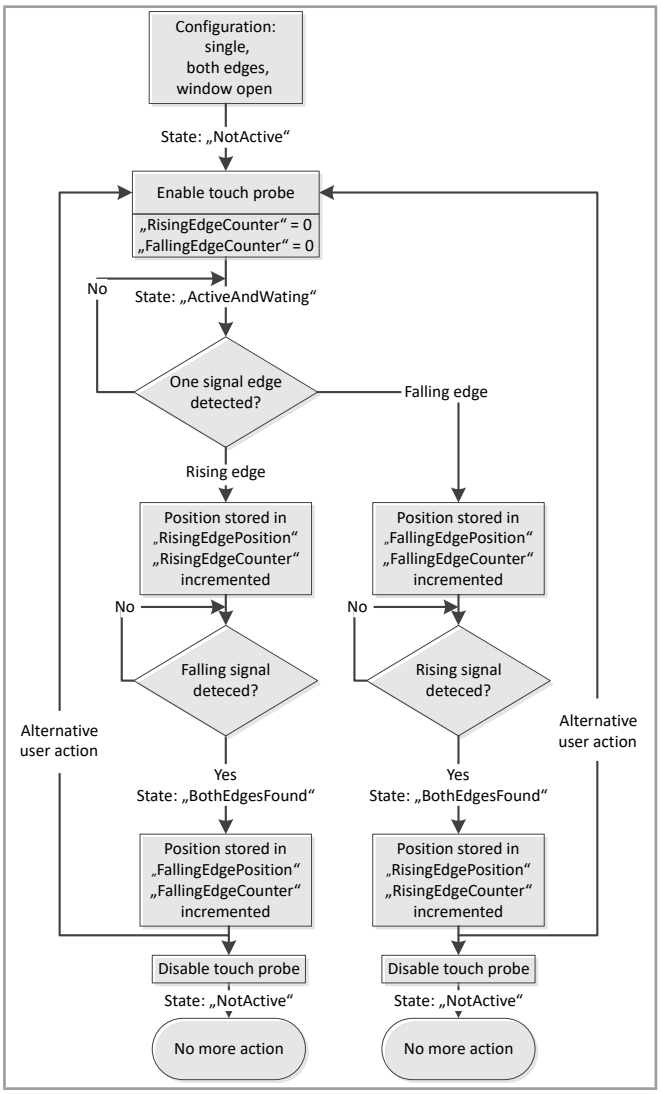


Abb. 7: *Flussdiagramm für Touch-Probe-Konfiguration Single Mode, beide Flanken, offenes Positionsfenster*

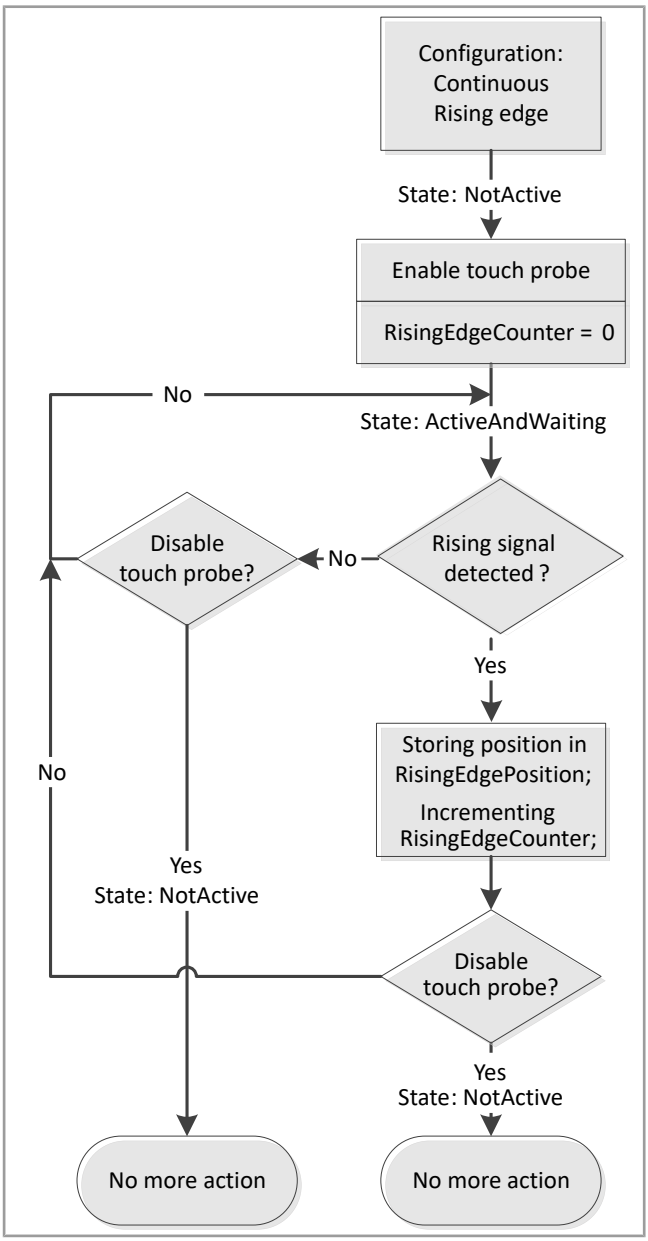


Abb. 8: *Flussdiagramm für Touch-Probe-Konfiguration Continuous Mode, steigende Flanke*

Touch-Probe-Prozess

Der Ablauf bei der Verwendung dieser Funktion ist folgender:

1. Konfigurieren Sie die Touch-Probe-Anwendung, indem Sie folgende Parameter festlegen:
   – Modus (`single`, `continuous`)
   – Flankenerkennung (`rising`, `falling`, `both`)
   – Positionsfenstergrenzen (`WindowMin`, `WindowMax`)
2. Aktivieren Sie die Touch-Probe-Funktion mit dem `Enable`-Befehl.

   ⇨ Der Flankenzähler (*edge counter*) wird automatisch auf Null zurückgesetzt.
3. Fragen Sie den Touch-Probe-Status im **Single Mode** ab.

   ⇨ Zunächst liegt der Status *active and wait* an.

   ⇨ Der Status wechselt zu *…edge found*, wenn:
   – Die momentane Istposition der MCX-Achse im Parameter *EdgePosition* gespeichert wurde.
   – Der Zähler für die konfigurierte Flanke (*EdgeCounter*) inkrementiert wurde.

4. Starten Sie die Touch-Probe-Funktion mit einem `Enable`-Kommando neu. Warten Sie mindestens zwei MCX-Zyklen (typischerweise 4 ms) vor einem erneuten Aufruf des Touch-Probe-`Enable`-Befehls.

   ODER

5. Beenden Sie die Touch-Probe-Funktion mit einem `Disable`-Kommando endgültig.
6. Fragen Sie den Touch-Probe-Status im **Continuous Mode** ab.
   ⇨ Zunächst liegt der Status *active and wait* an.
   ⇨ Eine Änderung im Flankenzähler für die konfigurierte Flanke (…*EdgeCounter*) liegt vor, wenn:
   – Die momentane Istposition der MCX-Achse im Parameter „..EdgePosition“ gespeichert wurde.
7. Beenden Sie die Touch-Probe-Funktion mit dem `Disable`-Befehl.

⇨ Der Touch-Probe-Ablauf ist abgeschlossen.

42.1 Syntax für MCAxis.TouchProbe

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Achsbezeichnung>.TouchProbe[n].<Funktion>(optional <Parameter>);
```

42.2 Funktionen von MCAxis.TouchProbe

Die Funktionalität bietet folgende Funktionen:

– Configure()
– Enable()
– Disable()
– ReadConfiguration()
– ReadState()

42.2.1 Configure()

Mit der Funktion übergeben Sie der Funktionalität **TouchProbe** die Informationen, die für Ihre Anwendung passend sind.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Edge	MCAxisTouchProbeEdges	Signalflanke, die erkannt werden soll.
	MCAxisTouchProbeEdges.Rising	Steigende Flanke
	MCAxisTouchProbeEdges.Falling	Fallende Flanke
	MCAxisTouchProbeEdges.Both	Beide Flanken
Mode	Typ MCAxisTouchProbeModes	*Touch-Probe-Mode*, der zur Anwendung kommen soll.
	MCAxisTouchProbeModes.TriggerFirstEvent	*Single Mode*: nur einmaliges Erfassen.
	MCAxisTouchProbeModes.Continuous	*Continuous Mode*: dauerhafte Auswertung der Flanke
WindowMin	Double	Untere Positionsfenstergrenze
WindowMax	Double	Obere Positionsfenstergrenze

Syntax

```
AX1.TouchProbe[1].Configure(MCAxisTouchProbeEdges.Rising,MCAxisTouchProbeMo-
des.TriggerFirstEvent,150.3,256.8);
AX1.TouchProbe[1].Configure(tpNewConfig.Edge,
                            tpNewConfig.Mode,
                            tpNewConfig.WindowMin,
                            tpNewConfig.WindowMax);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

42.2.2 Enable()

Mit der Funktion aktivieren Sie die Touch-Probe-Funktion.

Damit setzt der MCX-Kern die Touch-Probe-Zähler (`RisingEdgeCounter`, `FallingEdgeCounter`) auf Null zurück und der setzt den Touch-Probe-Status auf *AcitveAndWaiting*.

Im *Single Mode* können Sie mit der Funktion `Enable()` auch ohne vorheriges `Disable()`-Kommando die Touch-Probe-Funktion neu aktivieren.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.TouchProbe[1].Enable();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

42.2.3 Disable()

Mit der Funktion beenden Sie die Touch-Probe-Funktion.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
AX1.TouchProbe[1].Disable();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

42.2.4 ReadConfiguration()

Mit der Funktion lesen Sie die momentan aktive Touch-Probe-Konfiguration aus. Die Funktion übergibt Daten der Struktur `MCAxisTouchProbeConfiguration`. Es können auch die einzelnen Elemente der Struktur ausgelesen werden.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Edge	MCAxisTouchProbeEdges	Signalflanke, die erkannt werden soll.
	MCAxisTouchProbeEdges.Rising	Steigende Flanke
	MCAxisTouchProbeEdges.Falling	Fallende Flanke
	MCAxisTouchProbeEdges.Both	Beide Flanken

Parameter	Datentyp	Bedeutung
Mode	Typ MCAxisTouchProbeModes	Touch-Probe-Modus, der zur Anwendung kommen soll.
	MCAxisTouchProbeModes.TrigerFirstEvent	Single Mode: nur einmaliges Erfassen.
	MCAxisTouchProbeModes.Continuous	Continuous Mode: dauerhafte Auswertung der Flanke
WindowMin	Double	Untere Positionsfenstergrenze
WindowMax	Double	Obere Positionsfenstergrenze

Syntax

```
TPAktKonfiguration := AX1.TouchProbe[1].ReadConfiguration();

TPKonfigurierteFlanke := AX1.TouchProbe[1].ReadConfiguration().Edge;
```

Zustandsverwaltung

Für das Lesen der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

42.2.5 ReadState()

Mit der Funktion lesen Sie dem momentan aktiven Touch-Probe-Status aus. Die Funktion gibt Daten einer Struktur vom Typ `MCAxisTouchProbeState` zurück. Es können auch die einzelnen Elemente der Struktur ausgelesen werden.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Triggerstate	MCAxisTouchProbeTrigger-States	Status im Ablauf der Touch-Probe-Funktion.
	MCAxisTouchProbeTrigger-States.NotActive	Die Touch-Probe-Funktion ist nicht aktiv.
	MCAxisTouchProbeTrigger-States.ActiveAndWaiting	Die Touch-Probe-Funktion ist aktiv. Es wurde noch keine Signalflanke im Positionsfenster erkannt.
	MCAxisTouchProbeTrigger-States.RisingEdgeFound	Die Touch-Probe-Funktion ist auf steigende Flanke konfiguriert und es wurde eine steigende Flanke erkannt. Daraufhin wurde die aktuelle Position der MCX-Achse in `RisingeEdgePosition` kopiert und `RisingEdgeCounter` inkrementiert
	MCAxisTouchProbeTrigger-States.FallingEdgeFound	Die Touch-Probe-Funktion ist auf fallende Flanke konfiguriert und es wurde eine fallende Flanke erkannt. Daraufhin wurde die aktuelle Position der MCX-Achse in `FallingEdgePosition` kopiert und `FallingEdgeCounter` inkrementiert.
	MCAxisTouchProbeTrigger-States.BothEdgesFound	Die Touch-Probe-Funktion ist auf beide Flanken konfiguriert und beide Flanken wurden nacheinander erkannt. Beim Signal der steigenden Flanke wurde die aktuelle Position der MCX-Achse in `RisingeEdgePosition` kopiert und `RisingEdgeCounter` inkrementiert. Beim Signal der fallenden Flanke wurde die aktuelle Position der MCX-Achse in `FallingEdgePosition` kopiert und `FallingEdgeCounter` inkrementiert.
	MCAxisTouchProbeTrigger-States.NoEventInWindow	Die Ist-Position ist einmal ins Positionsfenster eingetreten und hat es wieder verlassen. Die Touch-Probe-Funktion hat jedoch die konfigurierte Signalflanke nicht erkannt.
RisingEdge-Counter	Int	Anzahl der erkannten steigenden Flanken des Sensorsignals. Im Single Mode kommen nur die Werte 0 und 1 vor.
RisingEdge-Position	Double	Kopierte Ist-Position bei der steigenden Flanke des Sensorsignals. Der Wert ist nach dem Aktivieren der Funktion so lange unbestimmt, bis eine steigende Flanke erkannt wurde (Angabe in der Anwender-Positions-Einheit).
FallingEdge-Counter	Int	Anzahl der erkannten fallenden Flanken des Sensorsignals. Im Single Mode kommen nur die Werte 0 und 1 vor.
FallingEdge-Position	Double	Kopierte Ist-Position bei der fallenden Flanke des Sensorsignals. Der Wert ist nach dem Aktivieren der Funktion so lange unbestimmt, bis eine fallende Flanke erkannt wurde (Angabe in der Anwender-Positions-Einheit).

Syntax

```
TPAktStatus := AX1.TouchProbe[1].ReadState();

If AX1.TouchProbe[1].ReadState().TriggerState == MCAxisTouchProbeTrigger-
States.RisingEdgeFound then
```

Zustandsverwaltung

Für das Lesen der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

42.3 Beispieltask MCAxis.TouchProbe

Hier finden Sie einen Beispieltask einer Einzelachse, der den Ablauf im zweiten Flussdiagramm oben darstellt:

```
Var
    Run             : bool; //starts and stops the entire process (set by
user)
    TriggerdPosition: Double;
    SignalState     : Enum TypeSignalState (undefined= 0, ok, notFound);
End_Var;

Task TouchProbe Autorun

AX1.Diagnostics.ClearErrors();
AX1.MoveHome.SetReference(5.0);

//AX1: modulo axis: modulo range = 0°.. 360°
//Touchprobe Configuration: Position Window 100° .. 180°
AX1.TouchProbe[1].Configure(MCAxisTouchProbeEdges.Falling,
                            MCAxisTouchProbeModes.TriggerFirstEvent,
                            100.0,
                            180.0);
Loop

When Run Continue;
    AX1.Power.Enable();
    When AX1.State.IsEnabled Continue;
    // start axis in endless mode
    AX1.MoveVelocity.Start(Directions.Positive,50.0,10000.0,10000.0);
    When AX1.State.IsRunning Continue;
```

```
    Repeat
        //enable touch probe function
        SignalState := TypeSignalState.undefined;
        TriggerdPosition := 0.0;
        AX1.TouchProbe[1].Enable();
        When AX1.TouchProbe[1].ReadState().TriggerState == MCAxisTouchPro-
beTriggerStates.ActiveAndWaiting Continue;

        When AX1.TouchProbe[1].ReadState().TriggerState == MCAxisTouchPro-
beTriggerStates.FallingEdgeFound Then
            TriggerdPosition := AX1.TouchProbe[1].ReadState().FallingEdgePo-
sition;
            SignalState := TypeSignalState.ok
        Else_When AX1.TouchProbe[1].ReadState().TriggerState == MCAxisTouch-
ProbeTriggerStates.NoEventInWindow Then
            SignalState := TypeSignalState.notFound;
        End_When;
        //Wait for modulo overflow
        When Ax1.Position.Actual > 350.0 Continue;
        When Ax1.Position.Actual < 5.0 Continue;

    Until Not Run;

    AX1.TouchProbe[1].Disable();
    AX1.MoveHalt.Start(MCAxisHaltModes.Normal,10000.0);
    When AX1.State.IsEnabled Continue;
    AX1.Power.Disable();
    When AX1.State.IsDisabled Continue;

End_Loop;
End_Task;
```

43 MCCouplingGroup

Mit der Funktionalität **MCCouplingGroup** (Koppelachsengruppe) können Sie die Position einer Achse als unterlagerte Positionsverarbeitung einer anderen Achse definieren und nutzen. Mögliche Anwendungsgebiete dafür sind:

1. Mechanisch gekoppelte Achsen unabhängig verfahren

   Es gibt mechanische Anordnungen, bei denen zwei Achsen über die Mechanik so gekoppelt sind, dass sich die zweite Achse mitdreht, wenn die erste bewegt wird. Um zu verhindern, dass sich die Position für die Anwendung der zweiten Achse ändert, kann die Position der ersten Achse der zweiten unterlagert werden. Auf diese Weise werden die beiden Achsen aus Sicht des Anwenders „entkoppelt“ und können unabhängig voneinander verfahren werden.

2. (Druckmarken-)Korrektur der Position einer Achse während der Bewegung

   Durch Kopplung einer virtuellen mit einer realen Achse können Sie die Referenzposition der realen Achse während der Bewegung aufgrund von externen Informationen (z. B. einer Druckmarke) korrigieren. Da die Bewegung der unterlagerten virtuellen Achse geschwindigkeits-, beschleunigungs- und ruckbegrenzt ausgeführt wird (es handelt sich dabei um eine MovePtP-Bewegung), können Sie das Verhalten der Ausgleichsbewegung vorgeben.

Für die Kopplung können Sie außerdem eine Übersetzung in Form eines Faktors und eines Divisors mitgeben. Durch einen negativen Faktor kann die Wirkung der Kopplung in der Drehrichtung angepasst werden.

Die Kopplung berücksichtigt die Getriebeeinstellungen der Achsen.

ⓘ Info

Bedingung für Erstellung einer Koppelgruppe

Die Kopplung an die Masterachse ist nur mit einer einzigen Folgeachse möglich. In den Funktionen sind Parameter vorgesehen, die es möglich machen in zukünftigen Erweiterungen auch mehrere Folgeachsen an eine Masterachse zu koppeln.

Koppelgruppe erstellen

Um eine Koppelgruppe zu verwenden, müssen Sie diese im STX-Programm zunächst als Variable definieren:

```
Var
DMKorr: MCCouplingGroup('DruckMKorrektur', AX10, mcMgr);
End_Var;
```

Dieser Aufruf erstellt eine neue Koppelgruppe (`MCCouplingGroup`) unter Verwendung der folgenden Parameter:

Parameter	Datentyp	Bedeutung
Groupname	String	Frei wählbarer Name.
MasterAxis	Pointer to MCAxis	Achse, die von „Folgeachsen“ als unterlagerte Achse verwendet wird.
mcMgr	IMCAxisManager	Interface-Parameter, der in dieser Weise mit übergeben werden muss.

43.1 Syntax für MCCouplingGroup

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Variable vom Typ MCCouplingGroup>.<Funktion oder Zugriff auf Wert>(optional
<Parameter1, Parameter 2, Parameter n>);
```

43.2 Funktionen von MCCouplingGroup

Die Funktionalität bietet folgende Funktionen:

– AddFollowerAxis()
– Redefine()
– Undefine()

43.2.1 AddFollowerAxis()

Mit der Funktion fügen sie der Koppelgruppe eine Folgeachse hinzu. Dieser Achse wird die Leitachse, die in der Variablendefinition übergeben wurde, unterlagert. Außerdem legen Sie die Übersetzung für die Unterlagerung fest.

Die Kopplung wird erst wirksam, nachdem die Funktion Redefine() [▶ 251] aufgerufen wurde.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Idx	Integer	Index der Folgeachse. Muss immer `0` sein.
AxFollower	Pointer to MCAxis	Objekt der Folgeachse. Dieser Achse wird die Leitachse unterlagert.
Factor	Integer	Faktor für die Übersetzung zwischen Leit- und Folgeachse.
Divisor	Integer	Divisor für die Übersetzung zwischen Leit- und Folgeachse.

Syntax

```
DMKorr.AddFollowerAxis(0,AX1,1,5);
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand der Leit- und Folgeachse möglich.

43.2.2 Redefine()

Mit der Funktion machen Sie die über AddFollowerAxis() [▶ 251] erstellte Kopplung wirksam.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
DMKorr.Redefine();
```

Zustandsverwaltung

Für das Aufrufen der Funktion für die Masterachse gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Das Aufrufen der Funktion ist in jedem Zustand der Folgeachse möglich.

43.2.3 Undefine()

Mit der Funktion lösen Sie die Koppelgruppe auf. Die Achsen werden voneinander entkoppelt. Dabei werden alle internen Positionsoffsets gelöscht.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
DMKorr.Undefine();
```

Zustandsverwaltung

Für das Aufrufen der Funktion gilt für die Leit- und Folgeachse folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

43.3 Eigenschaften von MCCouplingGroup

Die Funktionalität bietet folgende Eigenschaften:

– FollowerCount
– MasterAxis
– FollowerAxis
– Name

43.3.1 FollowerCount

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Anzahl der definierten Folgeachsobjekte der Koppelgruppe aus. Diese Eigenschaft greift zukünftige Erweiterungen vor, da bisher einer Koppelgruppe nur eine einzige Folgeachse hinzugefügt werden kann. Die Abfrage der Eigenschaft vor dem Aufrufen der Funktion `AddFollower()` gibt den Wert 0 zurück. Die Abfrage der Eigenschaft nach dem Aufrufen der Funktion `AddFollower()` gibt den Wert 1 zurück.

Syntax

```
AnzahlFolgeachsen := DMKorr.FollowerCount;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand der Leit- und Folgeachse möglich.

43.3.2 MasterAxis

Beschreibung

Datentyp	Bedeutung
Pointer to MCAxis	Liest das Achsobjekt aus, das als Leitachse in der Koppelgruppe definiert wurde.

Syntax

```
LeitAchsobjekt := DMKorr.MasterAxis;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand der Leit- und Folgeachse möglich.

43.3.3 FollowerAxis[]

Beschreibung

HINWEIS



STX-Ausnahmefehler bei fehlender Kopplung

Wenn Sie diese Eigenschaft abfragen, ohne dass ein Folgeachsobjekt vorhanden ist, wird im STX-Programm ein Ausnahmefehler erzeugt.

► Prüfen Sie vor Abfrage der Eigenschaft, ob eine Kopplung vorhanden ist.

Datentyp	Bedeutung
Pointer to MCAxis	Liest das Achsobjekt aus, das als Folgeachse in der Koppelgruppe definiert wurde. Der Array-Index ist für zukünftige Erweiterungen vorgesehen und muss bisher immer 0 sein.

Syntax

```
if DMKorr.FollwerAxis[0] != null then
Name_KoppelFolgeAchse := DMKorr.FollowerAxis[0].Info.Name;
end_if
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand der Leit- und Folgeachse möglich.

43.3.4 Name

Beschreibung

Datentyp	Bedeutung
String	Liest den Namen, den die Koppelgruppe bei der Variablendefinition erhalten hat.

Syntax

```
Koppelname := DMKorr.Name;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand der Leit- und Folgeachse möglich.

44 MCGantryGroup

Mit der Funktionalität **MCGantryGroup** (1:1 mechanisch gekoppelte Achsen) können Sie 2 oder mehr mechanisch gekoppelte Antriebe, die gemeinsam eine Last bewegen, so koppeln, dass sie wie eine Achse funktionieren. Dabei müssen alle Achsen die gleiche mechanische Konfiguration haben. Dabei gilt folgendes Verhalten:

– Ist eine Folgeachse einer Gantrygruppe zugeordnet, so erhält sie immer den gleichen Positionssollwert wie die Leitachse und legt auf diese Weise den gleichen Weg zurück.

– Die `Enable()`- und `Disable()`-Funktionen werden immer in allen Achsen der Gantrygruppe ausgeführt.

  Bei jedem `Enable()` der Gantrygruppe oder der Leitachse der Gantrygruppe werden die Ist- und die Sollposition der Folgeachsen der Gantrygurppe auf den Wert der Istposition der Leitachse gesetzt.

– Die Abweichung von der Istposition wird überwacht.

– Eine gemeinsame Referenzierung aller beteiligten Achsen ist möglich.

  Dies ist dann von Vorteil ist, wenn nicht eine Achse alleine in der Lage ist, die Last zu bewegen.

  Die Referenzfahrt wird über die Leitachse durchgeführt. Die Folgeachsen werden am Referenzpunkt mit dem gleichen Referenzwert geladen, wie ihn die Leitachse ermittelt hat.

– Ist die Masterachse Teil eines Technologieverbundes, werden die Folgeachsen wie die Masterachse behandelt. Sie müssen nicht eigens in den Technologieverbund aufgenommen werden.

Sie können die Gantry-Gruppe sowohl über den Objektnamen der Gantrygruppe gefolgt von der Benennung `MasterAxis` als auch über das Achsobjekt der Leitachse bedienen. Die Bedienung der Folgeachse ist nach der Kopplung nicht mehr direkt möglich. Aktionen auf die Folgeachse führen zu einer Fehlermeldung, solange die Kopplung aktiv ist.

ⓘ Info

Bedingung für Erstellung einer Koppelgruppe

Die Kopplung an die Masterachse ist nur mit einer einzigen Folgeachse möglich. In den Funktionen sind Parameter vorgesehen, die es möglich machen in zukünftigen Erweiterungen auch mehrere Folgeachsen an eine Masterachse zu koppeln.

Portalgruppe erstellen

Um eine Portalgruppe zu verwenden, müssen Sie diese im STX-Programm zunächst als Variable definieren:

```
Var
Portal1: MCGantryGroup('Portal1', AX1, mcMgr);
End_Var;
```

Dieser Aufruf erstellt eine neue Portalgruppe (`MCGantryGroup`) unter Verwendung der folgenden Parameter:

Parameter	Datentyp	Bedeutung
Groupname	String	Frei wählbarer Name.
MasterAxis	Pointer to MCAxis	Leitachse in der Portalgruppe, über die alle Befehle ausgeführt werden.
mcMgr	IMCAxisManager	Interface-Parameter, der in dieser Weise mit übergeben werden muss.

44.1 Syntax für MCGantryGroup

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Variable vom Typ MCGantryGroup>.<Funktion oder Zugriff auf Wert>(optional
<Parameter 1, Parameter 2, Parameter n>);
```

44.2 Funktionen von MCGantryGroup

Die Funktionalität bietet folgende Funktionen:

– AddFollowerAxis()
– Redefine()
– Undefine()

44.2.1 AddFollowerAxis()

Mit der Funktion fügen sie der Portalgruppe eine Folgeachse hinzu. Damit erhält die Achse die gleichen Sollwerte wie die Leitachse und ist über die Leitachse bedienbar.

Die Kopplung wird erst wirksam, nachdem die Funktion Redefine() [▶ 256] aufgerufen wurde.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Idx	Integer	Index der Folgeachse.
AxFollower	Pointer to MCAxis	Objekt der Folgeachse.

Syntax

```
DMKorr.AddFollowerAxis(0, AX2);
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand der Leit- und Folgeachse möglich.

44.2.2 Redefine()

Mit der Funktion machen Sie die über AddFollowerAxis() [▶ 256] erstellte Kopplung wirksam.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
Portal1.Redefine();
```

Zustandsverwaltung

Für das Aufrufen der Funktion gilt für die **Leit- und Folgeachse** folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben

Ausgangszustand	Erlaubt	Endzustand
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

44.2.3 Undefine()

Mit der Funktion lösen Sie die Portalgruppe auf. Die Achsen werden voneinander entkoppelt.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
Portal1.Undefine();
```

Zustandsverwaltung

Für das Aufrufen der Funktion gilt für die Leit- und Folgeachse folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

44.3 Eigenschaften von MCGantryGroup

Die Funktionalität bietet folgende Eigenschaften:

– GantryWindow[]
– ConfigGantryWindow[]
– FollowerCount
– MasterAxis
– FollowerAxis[]
– Name

44.3.1 GantryWindow[i]

Beschreibung

Datentyp	Bedeutung
Double	Liest oder schreibt die maximal erlaubte Positionsabweichung der Folgeachse mit dem Index `i` von der Leitachse in der Portalgruppe; Angabe in der Anwender-Position-Einheit. Standardwert: `10,0` Bei Überschreiten der zulässigen Positionsabweichung geht die Portalgruppe in den Fehlerzustand. Die Fehlerreaktion wird mit der Eigenschaft `ConfigGantryWindow` festgelegt.

Syntax

```
Portal1.GantryWindow[0] := 20.0;
GantryFenster:= Portal1.GantryWindow[0];
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand der Folgeachse möglich.

44.3.2 ConfigGantryWindow[i]

Beschreibung

Datentyp	Bedeutung
Integer	Liest oder schreibt die Fehlerreaktion der Portalgruppe, wenn die Positionsabweichung der Folgeachse mit dem Index `i` den Grenzwert `GantryWindow[i]` überschreitet; Standardwert: `5`
1	Löst nur eine Meldung aus. Bewirkt keine Reaktion der Achsen.
4	Die Achsen werden mit der programmierten Rampe angehalten und bleiben in Regelung nach dem Stillstand.
5	Die Achsen werden mit der Not-HALT-Rampe angehalten und nach dem Stillstand gesperrt (Default)
6	Die Achsen werden direkt gesperrt.

Syntax

```
Portal1.ConfigGantryWindow[0] := 5;
GantryKonfig:= Portal1.ConfigGantryWindow[0];
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand der Folgeachse möglich.

44.3.3 FollowerCount

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Anzahl der definierten Folgeachsobjekte der Portalgruppe aus.

Syntax

```
AnzahlFolgeachsen := Portal1.FollowerCount;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand der Leit- und Folgeachse möglich.

44.3.4 MasterAxis

Beschreibung

Datentyp	Bedeutung
Pointer to MCA-xis	Liest das Achsobjekt aus, das als Leitachse in der Portalgruppe definiert wurde.

Syntax

```
LeitAchsobjekt := Portal1.MasterAxis;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand der Leit- und Folgeachse möglich.

44.3.5 FollowerAxis[i]

Beschreibung

HINWEIS



STX-Ausnahmefehler bei fehlender Kopplung

Wenn Sie diese Eigenschaft abfragen, ohne dass ein Folgeachsobjekt vorhanden ist, wird im STX-Programm ein Ausnahmefehler erzeugt.

► Prüfen Sie vor Abfrage der Eigenschaft, ob eine Kopplung vorhanden ist.

Datentyp	Bedeutung
Pointer to MCA-xis	Liest das Achsobjekt aus, das in der Portalgruppe als Folgeachse mit dem Index `i` definiert wurde.

Syntax

```
if Portal.FollwerAxis[0] != null then
Name_GantryFolgeAchse := Portal1.FollowerAxis[0].Info.Name;
end_if
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand der Leit- und Folgeachse möglich.

44.3.6 Name

Beschreibung

Datentyp	Bedeutung
String	Liest den Namen, den die Portalgruppe bei der Variablendefinition erhalten hat.

Syntax

```
Gantryname := Portal1.Name;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand der Leit- und Folgeachse möglich.

44.4 Beispieltask MCGantryGroup

Hier finden Sie einen Beispieltask einer Portalgruppe, der verschiedene Motion-API-Funktionen enthält:

```
Var
//Definition eines Gantry-Verbunds mit Name(für den Verbund), Master-Achse
(typischer Weise eine der beiden Achsen)und MC-Managerfunktion
    Portal1: MCGantryGroup('Portal1', AX1, mcMgr);
//Zusätzliche Variablen für das Beispielprogramm
    Run: bool;
    Gantry_Zahl: int;
    GantryFenster: double;
    GantryKonfig: int;
End_Var;

Task tGantry Autorun

//Auflösen einer Portalgruppe - das ist normalerweise nicht nötig, kann aber
interessant sein, z. B. wenn Sie die Achsen auch alleine verfahren möchten.
    if Portal1.FollowerCount > 0 then
        Portal1.Undefine();
    end_if;

    //Zuerst wird dem Verbund eine Folgeachse hinzugefügt. Es können auch
mehrere Folgeachsen hinzugefügt werden
    if Portal1.FollowerCount = 0 then
        Portal1.AddFollowerAxis(0, AX2);
        Portal1.Redefine();
    end_if;

    Gantry_Zahl := Portal1.FollowerCount; //FollowerCont zeigt an, wie viele
Folgeachsen zur Portalgruppe gehören.

    Portal1.GantryWindow[0] := 20; //max. Schleppfehler zwischen Leit- und
Folgeachse in Anwendereinheiten
    GantryFenster:= Portal1.GantryWindow[0]; //Der Wert kann auch zurückge-
lesen werden.
    //Der erlaubte Schleppfehler zwischen den zwei Achsen wird in "Gantry-
Window[x] definiert.
    //Es muss ein Array-Index angegeben werden, weil auch mehr als zwei Ach-
sen in den Gantry-Verbund aufgenommen werden können.
    //Wird die programmierte Abweichung überschritten, geht die GantryGruppe
in einen Fehler
    //Die Fehlerreaktion kann konfiguriert werden:
    Portal1.ConfigGantryWindow[0] := 5; // 5 ist der Standardwert
    // auch die Konfiguration kann zurückgelesen werden
    GantryKonfig := Portal1.ConfigGantryWindow[0];

    //Referenzierung
```

```
    //Die Referenzfahrt wird mit der Leitachse durchgeführt. Wenn die Refe-
renz gefunden wurde, wird auch die Position der Folgeachse auf diesen Wert
gesetzt.
    //Bei jedem Enable werden die Ist- und Sollposition der Folgeachse auf
die der Leitachse gesetzt.

    if not Portal1.MasterAxis.State.IsEnabled then
        Portal1.MasterAxis.Power.Enable();
        When Portal1.MasterAxis.State.IsEnabled Continue;
    end_if;
    Portal1.MasterAxis.MoveHome.Start(Directions.Positive,ReferencingSwitch-
Configs.RefSwitch,
                        ReferencingModes.SwitchingEdge_1Phase,
                        100000.0, 300.0, 10000.0, 100.0, 36000.0, 0.0, 0.0);
    When Portal1.MasterAxis.MoveHome.IsReferenceSet Continue;
    // Statt Portal1.MasterAxis kann auch AX1 verwendet werden

    Portal1.MasterAxis.Power.Disable();
    When Portal1.MasterAxis.State.IsDisabled Continue

Loop
    //Die Bedienung erfolgt nur über die Leitachse.
    //Die Sollwerte der Leitachse werden auf die Folgeachse(n) angewandt
(inklusive eines internen Offsets, der beim Freigeben ermittelt wird.)
When Run Continue;
    // so wird der Gantry als Achse verwendet

        // Bei jedem Enable werden die Ist- und Sollposition der Folgeachse
auf die der Masterachse gesetzt.
        if not Portal1.MasterAxis.State.IsEnabled then
            Portal1.MasterAxis.Power.Enable();
            When Portal1.MasterAxis.State.IsEnabled Continue;
        end_if;

        //mit diesen Befehlen werden beide Achsen miteinander zwischen
1800.0 und 0.0 hin und hergefahren
        Portal1.MasterAxis.MovePtp.Start(AxisPositioningModes.AbsNor-
mal,1800.0,1000.0,10000.0,10000.0,0.1);
        when Portal1.MasterAxis.IsInTargetWindow Continue;
            Delay(t#500ms);

        AX1.MovePtp.Start(AxisPositioningModes.AbsNor-
mal,0.0,1000.0,10000.0,10000.0,0.1);
        when AX1.IsInTargetWindow Continue;
        Delay(t#500ms);
        if not run then
            Portal1.MasterAxis.Power.Disable();
            When Portal1.MasterAxis.State.IsDisabled Continue;
        end_if;
End_Loop;
End_Task;
```

45 MCGeo.Activate, MCGeo.Deactivate

Mit den Funktionalitäten **MCGeo.Activate** und **MCGeo.Deactivate** können Sie einen Bahnverbund aktivieren und deaktivieren. Änderungen an einem Bahnverbund sind in allen Betriebszuständen des Verbundes mit Ausnahme von **inaktiv** zulässig.

Sobald ein Bahnverbund aktiviert wurde, werden alle Mitgliedsachsen in den Betriebszustand **inaktiv** gesetzt. Auf diese Achsen können Sie nur noch über den Bahnverbund zugreifen, in den die Achsen eingebunden sind.

Ein Bahnverbund kann nur aktiviert und deaktiviert werden, wenn sich alle Mitgliedsachsen dieses Verbundes entweder im Betriebszustand **freigegeben** oder **gesperrt** befinden und alle Mitgliedsachsen den gleichen Betriebszustand haben. Andernfalls wird eine Fehlermeldung ausgegeben.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

45.1 Syntax für MCGeo.Activate, MCGeo.Deactivate

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Activate();
<Verbundbezeichnung>.Deactivate();
```

45.2 Funktionen von MCGeo.Activate, MCGeo.Deactivate

Die Funktionalität bietet folgende Funktionen:

– Activate()
– Deactivate()

45.2.1 Activate()

Mit der Funktion aktiveren Sie einen Bahnverbund sofern sich alle Mitgliedsachsen entweder im Zustand **Gesperrt** oder **Freigegeben** befinden.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.Activate();
```

Zustandsverwaltung

Für das Aufrufen der Funktion gilt für die Leit- und Folgeachse folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Inaktiv	√	Gesperrt/Freigegeben
Gesperrt	–	Gesperrt

Ausgangszustand	Erlaubt	Endzustand
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Fragen Sie das Erreichen des neuen Zustands im Programm ab, bevor Sie weitere Befehle für die Achsen aufrufen.

```
When Not geoGroup.State.IsInactive Continue;
```

Alternativ können Sie auch auf den Betriebszustand **IsEnabled** oder **IsDisabled** der Mitgliedsachsen abfragen – je nachdem, welchen Betriebszustand der Verbund vor dem Aufruf des Befehls hatte.

45.2.2 Deactivate()

Mit der Funktion deaktivieren Sie einen Bahnverbund. Nach diesem Befehl ist der Verbund im Zustand **inaktiv**. Die Mitgliedsachsen des Verbundes sind wieder unabhängig steuerbar. Sie übernehmen den Betriebszustand des Bahnverbundes.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.Deactivate();
```

Zustandsverwaltung

Für das Aufrufen der Funktion gilt für die Leit- und Folgeachse folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Inaktiv	–	Inaktiv
Gesperrt	√	Inaktiv
Freigegeben	√	Inaktiv
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Fragen Sie das Erreichen des neuen Zustands im Programm ab, bevor Sie weitere Befehle für die Achsen aufrufen.

```
When geoGroup.State.IsInactive Continue;
```

Alternativ können Sie auch auf den Betriebszustand **IsEnabled** oder **IsDisabled** der Mitgliedsachsen abfragen – je nachdem, welchen Betriebszustand der Verbund vor dem Aufruf des Befehls hatte.

46 MCGeo.Buffer

Mit der Funktionalität **MCGeo.Buffer** können Sie den Vorlaufpuffer des Bahnverbundes verwalten.

Bewegungsbefehle des Bahnverbundes werden immer gepuffert ausgeführt, d. h. sie werden in einen Vorlaufpuffer aufgenommen und nacheinander abgearbeitet. Dies ermöglicht auch ein Verschleifen der Bewegungen und das Programmieren ganzer Konturen, die z. B. aus mehreren Linear- und Kreisbewegungen bestehen.

Die Bedingung für den Übergang von einer Bahnbewegung zur nächsten (Satzwechselbedingung) wird im Bewegungsbefehl über den Parameter `MCGeoPathBlockEnd` mit in den Vorlaufpuffer übergeben.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

46.1 Syntax für MCGeo.Buffer

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Buffer.<Funktion>(optional <Parameter>);
```

46.2 Funktionen von MCGeo.Buffer

Die Funktionalität bietet folgende Funktionen:

- Clear()
- Suspend()
- Resume()
- UserIndex.Enable()
- UserIndex.Disable()
- UserIndex.Reset()

46.2.1 Clear()

Die Funktion löscht den Vorlaufpuffer mit folgenden Auswirkungen:

- Eine mit Verschleifen programmierte Bewegung wird entlang des Verschleifpolynoms abgebremst.
- Eine Bewegung, die schon begonnen wurde, wird zu Ende gefahren.

Die Befehle `Power.Disable()`, `Power.Quickstop()` und `Stop()` führen ebenfalls zu einer Löschung des Vorlaufpuffers.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.Buffer.Clear();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	–	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

46.2.2 Suspend()

Die Funktion hält die Vorlaufpufferabarbeitung am nächsten Satzübergang an mit folgenden Auswirkungen:

– Die laufende Bewegung wird zu Ende geführt.
– Die nächste im Vorlaupuffer beschriebene Bewegung wird nicht mehr gestartet.
– Der Verbund bleibt in der Betriebsart *Gepuffert*.
– Eine mit Verschleifen programmierte Bewegung wird entlang des Verschleifpolynoms abgebremst.

Sobald der Satz, während dessen Abarbeitung die `Buffer.Suspend()`-Funktion aufgerufen wurde, zu Ende gefahren wurde, befindet sich der Bahnverbund im Zustand **Freigegeben (angehalten)**.

Durch Aufrufen der Funktion Resume() [▶ 267] wird die Abarbeitung des Puffers wieder aufgenommen.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.Buffer.Suspend();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	–	Gesperrt
Freigegeben	–	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

46.2.3 Resume()

Die Funktion nimmt die mit Suspend() [▶ 266] angehaltene Abarbeitung des Vorlaufpuffers wieder auf.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.Buffer.Resume();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

46.2.4 UserIndex.Enable()

Die Funktion aktiviert die Funktion des Bahnstückzählers. Der Bahnstückzähler wird inkrementiert, wenn eine programmierte Bahnbewegung des Bahnverbundes gestartet wurde. Wird zwischen zwei Bahnbewegungen verschliffen, so wird der Bahnstückzähler nach dem Abschluss des Überschleifens inkrementiert.

Mit dem Befehl UserIndex.Reset() [▶ 269] kann er auf null zurückgesetzt werden.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.Buffer.UserIndex.Enable();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

46.2.5 UserIndex.Disable()

Die Funktion deaktiviert die Funktion des Bahnstückzählers.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.Buffer.UserIndex.Disable();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

46.2.6 UserIndex.Reset()

Die Funktion setzt den Bahnstückzähler auf Null zurück. Mit dem Reset wird auch ein Enable des Bahnstückzählers ausgeführt.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.Buffer.UserIndex.Reset();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

46.3 Eigenschaften von MCGeo.Buffer

Die Funktionalität bietet folgende Eigenschaften:

– UserIndex.Actual

46.3.1 UserIndex.Actual

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Anzahl der gefahrenen Bahnstücke seit dem letzten Zurücksetzen des Bahnstückzählers aus.

Syntax

```
AnzahlBahnstuecke := geoGroup.Buffer.UserIndex.Actual;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

47 MCGeo.Coordinates

Mit der Funktionalität **MCGeo.Coordinates** können Sie die aktuellen Werte für die Bahnbewegung nach Koordinaten sortiert abfragen.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

47.1 Syntax für MCGeo.Coordinates

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Coordinates.<Zugriff auf Wert>;
```

47.2 Eigenschaften von MCGeo.Coordinates

Die Funktionalität bietet folgende Eigenschaften:

– Actual
– Setpoint
– Target

47.2.1 Actual

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 6D-Koordinatenpunkte des Bahnverbunds.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) und 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
Actual	Coordinates6D	Istwert der Position des TCP in 6D-Raumkoordinaten.
Actual.X	Coordinates6D.X (Double)	Aktueller Wert der Position des TCP in X-Richtung.
Actual.Y	Coordinates6D.Y (Double)	Aktueller Wert der Position des TCP in Y-Richtung.
Actual.Z	Coordinates6D.Z (Double)	Aktueller Wert der Position des TCP in Z-Richtung.
Actual.U	Coordinates6D.U (Double)	Aktueller Wert der Orientierung des TCP in U-Richtung.
Actual.V	Coordinates6D.V (Double)	Aktueller Wert der Orientierung des TCP in V-Richtung.
Actual.W	Coordinates6D.W (Double)	Aktueller Wert der Orientierung des TCP in W-Richtung.

Syntax

```
Ist6D := geoGroup.Coordinates.Actual;
IstX := geoGroup.Coordinates.Actual.X;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

47.2.2 Setpoint

Beschreibung

Mithilfe der Eigenschaft lesen Sie die momentanen 6D-Koordinatenpunkte des Sollwertgenerators des Bahnverbunds.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) und 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
Setpoint	Coordinates6D	Sollwert der Position des TCP in 6D-Raumkoordinaten.
Setpoint.X	Coordinates6D.X (Double)	Sollwert der Position des TCP in X-Richtung.
Setpoint.Y	Coordinates6D.Y (Double)	Sollwert der Position des TCP in Y-Richtung.
Setpoint.Z	Coordinates6D.Z (Double)	Sollwert der Position des TCP in Z-Richtung.
Setpoint.U	Coordinates6D.U (Double)	Sollwert der Orientierung des TCP in U-Richtung.
Setpoint.V	Coordinates6D.V (Double)	Sollwert der Orientierung des TCP in V-Richtung.
Setpoint.W	Coordinates6D.W (Double)	Sollwert der Orientierung des TCP in W-Richtung.

Syntax

```
Sollwert6D := geoGroup.Coordinates.Setpoint;
SollwertX := geoGroup.Coordinates.Setpoint.X;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

47.2.3 Target

Beschreibung

Mithilfe der Eigenschaft lesen Sie die momentanen 6D-Koordinatenpunkte des Ziels der aktuellen Bahnbewegung des Bahnverbunds.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) und 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
Target	Coordinates6D	Ziel der Position des TCP in 6D-Raumkoordinaten.
Target.X	Coordinates6D.X (Double)	Ziel der Position des TCP in X-Richtung.
Target.Y	Coordinates6D.Y (Double)	Ziel der Position des TCP in Y-Richtung.
Target.Z	Coordinates6D.Z (Double)	Ziel der Position des TCP in Z-Richtung.
Target.U	Coordinates6D.U (Double)	Ziel der Orientierung des TCP in U-Richtung.
Target.V	Coordinates6D.V (Double)	Ziel der Orientierung des TCP in V-Richtung.

Eigenschaft	Datentyp	Bedeutung
Target.W	Coordinates6D.W (Double)	Ziel der Orientierung des TCP in W-Richtung.

Syntax

```
Zielwert6D := geoGroup.Coordinates.Target;
ZielwertX := geoGroup.Coordinates.Target.X;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

48 MCGeo.Coupling

Mit der Funktionalität **MCGeo.Coupling** steuern Sie die Tool–Center-Point-Bewegung eines Bahnverbundes mithilfe der Position von bis zu sechs anderen unabhängigen MCX-Achsen. Dabei ist der Bahnverbund an die andere(n) Achse(n) gekoppelt und folgt deren Bewegung. So kann beispielsweise der Greifarm eines Mehrachsroboters an die Bewegung eines Förderbandes gekoppelt werden, von dem er etwas aufnehmen soll.

Kopplungstypen

Es gibt 2 Möglichkeiten der Bewegungssteuerung:

1. Die Bahnbewegung folgt den MCX-Achsen in einer linearen Bewegung.
2. Die Bahnbewegung folgt den MCX-Achsen in Form einer Kreisbewegung.

Sie können den Bahnverbund an bis zu sechs MCX-Achsen koppeln. Je nach Wahl des Kopplungstyps steuern diese Achsen in unterschiedlicher Art die Bewegung des Bahnverbundes.

Bewegungssteuerung für Kopplungstyp 1: lineare Bahnbewegung:

– Die Bewegung des Bahnverbundes in X-Richtung folgt linear der ersten im Koppelbefehl angegebenen Achse.
– Die Bewegung des Bahnverbundes in Y-Richtung folgt linear der zweiten im Koppelbefehl angegebenen Achse.
– Die Bewegung des Bahnverbundes in Z-Richtung folgt linear der dritten im Koppelbefehl angegebenen Achse.
– Die U-Orientierung des Bahnverbundes folgt linear der vierten im Koppelbefehl angegebenen Achse.
– Die V-Orientierung des Bahnverbundes folgt linear der fünften im Koppelbefehl angegebenen Achse.
– Die W-Orientierung des Bahnverbundes folgt linear der sechsten im Koppelbefehl angegebenen Achse.

Bewegungssteuerung für Kopplungstyp 2: kreisförmige Bahnbewegung:

Dieser Kopplungstyp erfordert zusätzlich die Mittelpunktkoordinaten des Kreises und die Koordinatenebene, in der der Kreis liegen soll.

Die ersten 3 gekoppelten Achsen geben die Kreisbewegung in Polarkoordinaten vor.

– Der Radius der Kreisbewegung des Bahnverbundes folgt der ersten im Koppelbefehl angegebenen Achse.
– Der Winkel der Kreisbewegung des Bahnverbundes folgt der zweiten im Koppelbefehl angegebenen Achse.
– Die Helix der Kreisbewegung des Bahnverbundes folgt linear der dritten im Koppelbefehl angegebenen Achse.
– Die U-Orientierung des Bahnverbundes folgt linear der vierten im Koppelbefehl angegebenen Achse.
– Die V-Orientierung des Bahnverbundes folgt linear der fünften im Koppelbefehl angegebenen Achse.
– Die W-Orientierung des Bahnverbundes folgt linear der sechsten im Koppelbefehl angegebenen Achse.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Synchronachsen in Bahnverbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund* und *Bahnverbund – Synchrone Achse* . Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

48.1 Syntax für MCGeo.Coupling

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Coupling.<Funktion oder Zugriff auf Wert>(optional <Parameter 1, Parameter 2, Parameter n>);
```

48.2 Funktionen von MCGeo.Coupling

Die Funktionalität bietet folgende Funktionen:

– Couple()
– Decouple()
– GetPositioningMode()
– GetCoupleType()
– GetCoupleMode()
– GetDecoupleMode()

48.2.1 Couple()

Mit der Funktion konfigurieren Sie die Koppelfunktion zwischen den MCX-Achsen und dem Bahnverbund und stellen die Kopplung her. Der Prozess setzt voraus, dass der Bahnverbund im Betriebszustand **Freigegeben** ist. Im gekoppelten Zustand ist der Betriebszustand des Bahnverbundes immer **Läuft** und er lässt sich nicht mehr über die `Move`-Befehle bewegen.

Übergabeparameter

Parameter	Datentyp	Bedeutung
PositioningMode	MCGeoPositioningModes (optional)	Definiert die Art, wie der Verbund den Positionen der MCX-Achsen folgt.
	MCGeoPositioningModes.Absolute	Der Verbund folgt der Position der MCX-Achsen bezogen auf die gleichen Werte der Sollposition und der Offsets: *Sollposition der TCP-Koordinate = Sollposition der MCXAchse + Offset*. Beim Start der Kopplung müssen die Positionswerte des TCP und der beteiligten MCX-Achsen (um die Offsets verschoben) übereinstimmen, um eine ruckartige Bewegung des TCP zu vermeiden.
	MCGeoPositioningModes.Relative	Der Verbund folgt der Position der MCX-Achsen zuzüglich seiner Startpositionen und der Offsets der MCX-Achsen: *Sollposition der TCP-Koordinate = Sollposition der MCX-Achse + Startposition der TCP-Koordinate + Offset*

Parameter	Datentyp	Bedeutung
CoupleType	MCGeoCoupleTypes (optional)	Art der Bewegung des TCP in Abhängigkeit von den MCX-Achsen.
	MCGeoCoupleTypes.Linear	Die Sollpositionen der MCX-Achsen dienen direkt als Vorgabe für die Sollpositionen der Bahnkoordinaten.
	MCGeoCoupleTypes.CircularXY	Die Sollpositionen der MCX-Achsen dienen als Vorgabe für die Polarkoordinaten (s. o.) einer Kreisbewegung in der XY-Ebene.
	MCGeoCoupleTypes.CircularYZ	Die Sollpositionen der MCX-Achsen dienen als Vorgabe für die Polarkoordinaten (s. o.) einer Kreisbewegung in der YZ-Ebene.
	MCGeoCoupleTypesCircularZX	Die Sollpositionen der MCX-Achsen dienen als Vorgabe für die Polarkoordinaten (s. o.) einer Kreisbewegung in der ZX-Ebene.
CoupleMode	MCGeoCoupleModes.Immediate (optional)	Die Kopplung wird sofort ausgeführt.
CoupleID_1	Integer (optional)	Objekt-ID der ersten MCX-Achse, auf die der Bahnverbund gekoppelt werden soll (siehe MCAxis.Info [▶ 113]).
CoupleID_2	Integer (optional)	Objekt-ID der zweiten MCX-Achse, auf die der Bahnverbund gekoppelt werden soll (siehe MCAxis.Info [▶ 113]).
CoupleID_3	Integer (optional)	Objekt-ID der dritten MCX-Achse, auf die der Bahnverbund gekoppelt werden soll (siehe MCAxis.Info [▶ 113]).
CoupleID_4	Integer (optional)	Objekt-ID der vierten MCX-Achse, auf die der Bahnverbund gekoppelt werden soll (siehe MCAxis.Info [▶ 113]).
CoupleID_5	Integer (optional)	Objekt-ID der fünften MCX-Achse, auf die der Bahnverbund gekoppelt werden soll (siehe MCAxis.Info [▶ 113]).
CoupleID_6	Integer (optional)	Objekt-ID der sechsten MCX-Achse, auf die der Bahnverbund gekoppelt werden soll (siehe MCAxis.Info [▶ 113]).
Offset_1	Double (optional)	Offset zwischen der Sollposition der ersten MCX-Achse und dem Sollwert für die entsprechende Bahnkoordinate.
Offset_2	Double (optional)	Offset zwischen der Sollposition der zweiten MCX-Achse und dem Sollwert für die entsprechende Bahnkoordinate.
Offset_3	Double (optional)	Offset zwischen der Sollposition der dritten MCX-Achse und dem Sollwert für die entsprechende Bahnkoordinate.
Offset_4	Double (optional)	Offset zwischen der Sollposition der vierten MCX-Achse und dem Sollwert für die entsprechende Bahnkoordinate.
Offset_5	Double (optional)	Offset zwischen der Sollposition der fünften MCX-Achse und dem Sollwert für die entsprechende Bahnkoordinate.
Offset_6	Double (optional)	Offset zwischen der Sollposition der sechsten MCX-Achse und dem Sollwert für die entsprechende Bahnkoordinate.
CenterPointX	Double (optional)	X-Koordinate des Kreismittelpunkts, wenn als `CoupleType` = `CircularXY` oder `CircularZX` gewählt wurde.
CenterPointY	Double (optional)	Y-Koordinate des Kreismittelpunkts wenn als `CoupleType` `CircularXY` oder `CircularYZ` gewählt wurde.
CenterPointZ	Double (optional)	Z-Koordinate des Kreismittelpunkts im Fall, dass der CoupleType `CircularYZ` oder `CircularZX` gewählt wurde.

Syntax

```
geoGroup.Coupling.Couple(MCGeoPositioningModes.Absolute,MCGeoCoupleTypes.Linear,MCGeoCoupleModes.Immediate,AX1.Info.ObjectID,,,,,,10.0,,,,,,);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	√	Läuft
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

48.2.2 Decouple()

Übergabeparameter

Parameter	Datentyp	Bedeutung
Mode	MCGeoDecoupleModes.Immeditate (optional)	Entkoppelt den Bahnvorbund von den unabhängigen MCX-Achsen, an die der Bahnverbund gekoppelt war.

Syntax

```
geoGroup.Coupling.Decouple(MCGeoDecoupleModes.Immediate);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	–	Freigegeben
Läuft	√	Freigegeben
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

48.2.3 GetPositioningMode()

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest den aktuellen Positioniermodus des gekoppelten Bahnverbundes aus. Der Rückgabewert ist vom Typ `MCGeoPositioningModes`.

Syntax

```
CouplePM := geoGroup.Coupling.GetPositioningMode(AX1);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

48.2.4 GetCoupleType()

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest den aktuellen Kopplungstyp des gekoppelten Bahnverbundes aus. Der Rückgabewert ist vom Typ `MCGeoCoupleTypes`.

Syntax

```
CoupleT := geoGroup.Coupling.GetCoupleType(AX1);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

48.2.5 GetCoupleMode()

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest den aktuellen Kopplungs-Modus des gekoppelten Bahnverbundes aus. Der Rückgabewert ist vom Typ `MCGeoCoupleModes`.

Syntax

```
CoupleM := geoGroup.Coupling.GetCoupleMode(AX1);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

48.2.6 GetDecoupleMode()

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest den aktuellen Auskopplungmodus des gekoppelten Bahnverbundes aus. Der Rückgabewert ist vom Typ `MCGeoDecoupleModes`**. Dieser ist bisher immer** `Immediate` **(**`1`**).**

Syntax

```
AuskopDecoupleM := geoGroup.Coupling.GetDecoupleMode(AX1);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

48.3 Eigenschaften von MCGeo.Coupling

Die Funktionalität bietet folgende Eigenschaften:

– CoupleState

48.3.1 CoupleState

Beschreibung

Eigenschaft	Datentyp	Bedeutung
CoupleState	MCGeoCoupleStates	Liest den Koppelstatus des Bahnverbundes.
	MCGeoCoupleStates.Decoupled	Der Bahnverbund ist nicht an andere Achsen gekoppelt.
	MCGeoCoupleStates.Coupled	Der Bahnverbund ist im gekoppelten Zustand.

Syntax

```
CoupleStatus := geoGroup.Coupling.CoupleState;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

49 MCGeo.Diagnostics

Mit der Funktionalität **MCGeo.Diagnostics** können Sie die Fehlerbehandlung eines Bahnverbundes verwalten.

Im Motion-Control-eXtended-System werden Fehler (*errors*) und Meldungen (*messages*) unterschieden. Fehler rufen eine Systemreaktion hervor, Meldungen hingegen nicht. Fehler und Meldungen werden in getrennten Puffern protokolliert, von wo aus sie ausgelesen werden können. Der jeweilige Puffer sammelt die Ereignisse, die seit dem letzten Aufruf der Funktion `ClearErrors`() aufgetreten sind (siehe ClearErrors() [▶ 39]).

Die Bedeutung der Fehler- und Meldungsnummern ist in der JetSym -Online-Hilfe unter „MotionControl-Fehler“ aufgeführt.

49.1 Syntax für MCGeo.Diagnostics

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundsbezeichnung>.Diagnostics.<Funktion oder Zugriff auf Wert>(optional
<Parameter>);
```

49.2 Funktionen von MCGeo.Diagnostics

Die Funktionalität bietet folgende Funktionen:

– ClearErrors()
– Software.ClearErrors()
– Software.GetErrorCode()
– Software.GetMessageCode()

49.2.1 ClearErrors(), Software.ClearErrors() – Bahnverbund

Bei Fehlern quittieren die Funktionen einen Fehlerzustand im Bahnverbund. Die Einträge im Fehlerpuffer werden gelöscht.

Meldungen haben keine Fehlerreaktion zur Folge, und müssen deshalb nicht quittiert werden. Bei Meldungen bewirken die beiden Funktionen, dass die Einträge im Meldungspuffer gelöscht werden.

Übergabeparameter

Diese Funktionen besitzen keine Übergabeparameter.

Syntax

```
geoGroup.Diagnostics.ClearErrors();
geoGroup.Diagnostics.Software.ClearErrors();
```

ⓘ Info

Geltungsbereich des Befehls

Mit diesen Befehlen werden alle Fehler gelöscht, die vom Verbund zugeordnet sind. Das können Fehler sein, die von einem Verbundobjekt ausgelöst wurden, ebenso wie Fehler, die von den Verbundachsen gemeldet wurden.

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand	
Gesperrt	–	Gesperrt	
Freigegeben	–	Freigegeben	
Läuft	–	Läuft	
Unterbrochen	–	Unterbrochen	
Fehler: freigegeben	√	Freigegeben	In diesen Zustand wechselt das System nur, wenn die Fehlerursache behoben ist. Andernfalls bleibt u. U. auch der Fehlerzustand erhalten oder wird sofort wieder erreicht.
Fehler: gesperrt	√	Gesperrt	

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

49.2.2 Software.GetErrorCode()

Die Funktion liefert die Nummer des Fehlers an der mit `BufferIndex` angegebenen Fehlerpufferstelle zurück. Wird kein `BufferIndex` übergeben, wird die Nummer des zuletzt aufgezeichneten Fehlers zurückgegeben (`BufferIndex` = `0`).

Übergabeparameter

Parameter	Datentyp	Bedeutung
BufferIndex	Integer (optional)	Der `BufferIndex` übergibt die Stelle im Fehlerpuffer, deren Fehlernummer ausgelesen werden soll. Der Maximalwert des Parameters `BufferIndex` kann über die Eigenschaft `ErrorCount` ausgelesen werden.

Syntax

```
geoGroup.Diagnostics.Software.GetErrorCode();
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

49.2.3 Software.GetMessageCode()

Die Funktion liefert die Nummer der Meldung an der mit `BufferIndex` angegebene Meldungspufferstelle zurück. Wird kein `BufferIndex` übergeben, wird die Nummer der zuletzt aufgezeichneten Meldung zurückgegeben (`BufferIndex` = `0`).

Übergabeparameter

Parameter	Datentyp	Bedeutung
BufferIndex	Integer (optional)	Der `BufferIndex` übergibt die Stelle im Meldungspuffer, deren Meldungsnummer ausgelesen werden soll. Der Maximalwert des Parameters `BufferIndex` kann über die Eigenschaft `MessageCount` ausgelesen werden.

Syntax

```
geoGroup.Diagnostics.Software.GetMessageCode();
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

49.3 Eigenschaften von MCGeo.Diagnostics

Die Funktionalität bietet folgende Eigenschaften:

- Software.ErrorCount
- IsErrorPending
- Software.IsErrorPending
- Software.MessageCount
- IsMessagePending
- Software.IsMessagePending

49.3.1 Software.ErrorCount

Beschreibung

Eigenschaft	Datentyp	Bedeutung
Software.ErrorCount	Int	Liest die Anzahl der erfassten Fehler seit dem letzten Ausführen der Funktion `ClearErrors()` bzw. `Software.ClearErrors()`, oder seit dem Neustart der Steuerung aus.

Syntax

```
FehlerAnzahl := geoGroup.Diagnostics.Software.ErrorCount;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

49.3.2 IsErrorPending, Software.IsErrorPending

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob ein Fehler im MCX-System vorliegt. Dies ist gleichbedeutend mit einem Vergleich von `ErrorCount` mit `0`.

Syntax

```
If geoGroup.Diagnostics.IsErrorPending then
...
End_if;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

49.3.3 Software.MessageCount

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Anzahl der erfassten Meldungen seit dem letzten Ausführen der Funktion `ClearErrors()` bzw. `Software.ClearErrors()`, oder seit dem Neustart der Steuerung aus.

Syntax

```
MeldungsAnzahl := geoGroup.Diagnostics.Software.MessageCount;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

49.3.4 IsMessagePending, Software.IsMessagePending

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob eine Meldung im MCX-System vorliegt. Dies ist gleichbedeutend mit einem Vergleich von `MessageCount` mit `0`.

Syntax

```
If geoGroup.Diagnostics.IsMessagePending then
...
End_if;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

49.4 Beispieltask für MCGeo.Diagnostics

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Var
    GeoError: Array[10] of INT; //Variablen zum Anzeigen des Fehlerpuffers
    i: INT; //Laufvariable
    AcknolegeError: Bool; // Hilfs-Flag für eine Quittier-Bedingung

End_Var;

Task Sample_Error Autorun

Loop

    //Fehlervariablen zum Anzeigen löschen
    For i := 0 To 9 Do
        GeoError[i] := 0;
    End_For;
    AcknowledgeError := False;

    When geoGroup.Diagnostics.IsErrorPending Continue;
```

```
        Taskbreak Automatic;
        // hier weitere Fehlerbehandlung einfügen

        //Fehler in Variablen z. B. zum Anzeigen ablegen
        For i := 0 To geoGroup.Diagnostics.Software.ErrorCount-1 Do
            GeoError[i] := geoGroup.Diagnostics.Software.GetErrorCode(i);
        End_For;

    // Fehler sind behoben
    When AcknowledgeError Continue;

        geoGroup.Diagnostics.ClearErrors();
        Taskrestart Automatic;
End_Loop;

End_Task;
```

50 MCGeo.Events

Mit der Funktionalität **MCGeo.Events** können Sie das Verhalten von Flags verwalten, die abhängig vom Ablauf der Bewegung im Vorlaufpuffer des Bahnverbundes gesetzt werden.

ⓘ Info

Funktion `SetFlag()` in Verbindung mit dem Vorlaufpuffer

Die `Event`-Befehle werden in den Vorlaufpuffer mitübernommen. Eine Bedingung für das Setzen eines Flags (Funktion `SetFlag()`) muss vor dem Bahnstück, auf das es sich bezieht, in den Vorlaufpuffer geschrieben werden.

Das Arbeiten mit dem Vorlaufpuffer ermöglicht es, Bahnkonturen abzufahren, die aus mehreren Bahnstücken bestehen und/oder durch Überschleifen ineinander übergehen. Die Funktionen der Funktionalität **Events** ermöglichen es Ihnen, vom MCX-Kernel Flags setzen zu lassen. Die Flags sind abhängig davon, welches Bahnstück im Vorlaufpuffer gerade abgefahren wird, und wo sich die Bewegung im Bahnstück gerade befindet. Mithilfe dieser Flags wiederum können Sie im STX-Programm Aktionen steuern, z. B. das Öffnen eines Dosierventils.

Mit dem Aktivieren des Bahnverbundes werden alle Flags gelöscht.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

50.1 Syntax für MCGeo.Events

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Events.<Funktion oder Zugriff auf Wert>(optional <Parameter);
```

50.2 Funktionen von MCGeo.Events

Die Funktionalität bietet folgende Funktionen:

- SetFlag()
- ReadFlags()
- ReadFlag()
- ClearFlags()

50.2.1 SetFlag()

Mit der Funktion veranlassen Sie den MCX-Kernel, beim Ausführen des folgenden Bahnstücks im Vorlaufpuffer ein Flag abhängig vom Übergabeparameter zu setzten.

Wird die Bedingung am Bewegungsende für eine Bewegung mit Überschleifen gewählt, so wird das Flag am Beginn des Überschleifpolynoms gesetzt.

Übergabeparameter

Parameter	Datentyp	Bedeutung
BlockCondition	MCGeoEventConditions	Bedingung, die definiert, an welcher Stelle der Bewegung das Flag mit der `EventNo` gesetzt wird.
	MCGeoEventConditions.BlockStart	Das Flag wird beim Start der Bewegung gesetzt.
	MCGeoEventConditions.BlockEnd_InPosState	Das Flag wird am Ende der Bewegung, sobald der Sollwertgenerator das Ziel erreicht hat, gesetzt.
	MCGeoEventConditions.BlockEnd_InPosState_DestWnd	Das Flag wird vor Ende der Bewegung, sobald das Zielfenster erreicht wurde, gesetzt.
	MCGeoEventConditions.BlockEnd_InPosState_DestWnd_Delay	Das Flag wird am Ende der Bewegung und nach der programmierten Wartezeit gesetzt.
EventNo	Integer (optional)	Nummer des Flags (1 ... 16), das unter obiger Bedingung gesetzt werden soll.

Syntax

```
geoGroup.Events.SetFlag(MCGeoEventConditions.BlockStart,2);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	√	Gesperrt
Läuft	√	Gesperrt
Unterbrochen	X	Gesperrt
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

50.2.2 ReadFlag()

Mit der Funktion lesen Sie den Status eines bestimmten Event-Flags.

Übergabeparameter

Parameter	Datentyp	Bedeutung
EventNo	Integer	Nummer des Flags (1 ... 16), dessen Status gelesen werden soll.

Syntax

```
If geoGroup.Events.ReadFlag(2) then;
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

50.2.3 ReadFlags()

Mit der Funktion lesen Sie den Status aller 16 Event-Flags. Die Funktion gibt einen Wert vom Datentyp Integer zurück. Die letzten 16 Bit dieses Wertes entsprechen den Flags mit den Nummern 1 … 16. (Bit 0 = EventFlag 1, ...).

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
FlagWord := geoGroup.Events.ReadFlags();
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

50.2.4 ClearFlags()

Mit der Funktion löschen Sie die in der Eventmaske übergebenen Flags des Bahnverbunds.

Das Löschen der Flags wird nicht im Vorlaufpuffer gespeichert, sondern sofort ausgeführt.

Übergabeparameter

Parameter	Datentyp	Bedeutung
EventMask	Integer	Bit 0 ... 15 sind den Flags 1 ... 16 zugeordnet. Ein gesetztes Bit löscht das entsprechende Event-Flag.

Syntax

```
geoGroup.Events.ClearFlags(0x0001);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Zustandsüberwachung

Folgt dem `ClearFlags()`-Befehl unmittelbar ein `Deactivate()`-Befehl, so kann es sein, dass die entsprechenden Flags nicht gelöscht werden, da der `Deactivate()`-Befehl höhere Priorität hat, wenn er im selben MCX-Zyklus bearbeitet wird. Da die `ClearFlags()`-Funktion jedoch im Zustand **inaktiv** nicht ausgeführt wird, ist es sinnvoll in diesem Fall vor dem `Deactivate()`-Befehl abzufragen, ob die entsprechenden Flags gelöscht wurden.

50.3 Beispieltask MCGeo.Events

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Task Bahnbewegung Autorun
...
    //Starten von drei Linearbewegungen - am Anfang der zweiten Bewegung
soll das EventFlag 2 gesetzt werden
    geoGroup.Power.Enable();
    When geoGroup.State.IsEnabled Continue;
    geoGroup.Mechanism.Limitations.MaxPathAcceleration := 12000;
```

```
    geoGroup.MoveLinear.Start(MCGeoPositioningModes.Absolute,MCGeoPathBlo-
ckEnd.InPosState,300,20,0,0.1,100,50);
geoGroup.Events.SetFlag(MCGeoEventConditions.BlockStart,2);
    geoGroup.MoveLinear.Start(MCGeoPositioningModes.Absolute,MCGeoPathBlo-
ckEnd.InPosState,300,10,500,0.1,150,150);
    geoGroup.MoveLinear.Start(MCGeoPositioningModes.Absolute,MCGeoPathBlo-
ckEnd.InPosState,300,10,0,0.1,0,0);
...
End_Task;


Task Dosierung Autorun

    If geoGroup.Events.ReadFlag(2) Then
        oDosieung := true; //Dosierung am Beginn des zweiten Bahnstücks zu-
schalten
    End_if;
...
End_Task;
```

51 MCGeo.Info

Mit der Funktionalität **MCGeo.Info** können Sie die Verbundobjekte der MotionControl identifizieren.

Es handelt sich um Informationen, die vom Anwender selbst im Hardware-Manager und Motion-Setup im Projekt festgelegt wurden.

51.1 Syntax für MCGeo.Info

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Info.<Eigenschaft>;
```

51.2 Eigenschaften der Funktionalität

Die Funktionalität bietet folgende Eigenschaften:

– Name
– ObjectID
– ClassID

51.2.1 Name

Beschreibung

Datentyp	Bedeutung
String	Fragt den in der Hardware-Konfiguration definierten Namen des Bahnverbundes ab. Dies ist hilfreich, wenn Sie den Namen im Trace oder auf einer externen Anzeige verwenden wollen.

Syntax

```
GeoName := Geo.Info.Name;
```

Dabei enthält `GeoName` den String `Geo`.

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

51.2.2 ObjectID

Beschreibung

Datentyp	Bedeutung
Integer	Fragt die steuerungsinterne Nummer des Bahnverbundes ab. Die Eigenschaft wird normalerweise nur im Zusammenhang mit Anwendungen benötigt, die die Funktionen des MC-Managers nutzen. Eine Anwendungsmöglichkeit ist im zugehörigen Abschnitt MCManager (MCMgr) [▶ 413] beschrieben.

Syntax

```
Verbundkennung := geoGroup.Info.ObjectID;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

51.2.3 ClassID

Beschreibung

Datentyp	Bedeutung
MCClassIDs	Fragt die Klasse eines MC-Objekts ab. Ein MC-Objekt kann eine Achse, ein Technologieverbund oder ein Bahnverbund sein. Die Eigenschaft wird normalerweise nur im Zusammenhang mit Anwendungen benötigt, die die Funktionen des MC-Managers nutzen. Eine Anwendungsmöglichkeit ist im zugehörigen Abschnitt MCManager (MCMgr) [▶ 413] beschreiben.

Syntax

```
KlassenID := geoGroup.Info.ClassID;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

51.3 Beispieltask MCGeo.Info

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

Nach Ausführung des Beispieltasks wird im Trace für alle Bahnverbunde angezeigt, ob sie im Zustand **läuft** sind oder nicht. Die Verbunde werden mit ihrem Verbundnamen aufgeführt.

```
function TraceRunningState(mcGeoGroup: pointer to MCGeo)
    if mcGeoGroup.State.IsRunning then
        trace(mcGeoGroup.Info.Name + 'is running$n');
    else
        trace(mcGeoGroup.Info.Name + 'is not running$n');
    end_if;
end_function;

task Sample1 // Beispieltask
var
    nGeoIndex:    int;
end_var;
    for nGeoIndex := 0 to mcMgr.GeoCount - 1 by 1 do
        TraceRunningState(mcMgr.GeoList[nGeoIndex]);
    end_for;
end_task;
```

52 MCGeo.Mechanism.Kinematics

Mit der Funktionalität **MCAxis.Mechanism.Kinematics** können Sie die mechanische Ausführung eines Bahnverbundes auslesen und in Ausnahmefällen konfigurieren. Üblicherweise erfolgt die Konfiguration im MotionSetup während der Inbetriebnahme.

Die mechanische Ausführung eines Bahnverbundes muss der Steuerung bekannt sein, damit die Einheiten und die Kinematik im MCX-Kernel richtig berechnet werden. Dies geschieht in JetSym mithilfe der Definition im Hardware-Baum und dem Motion-Setup. Diese Daten werden in der Datei *MCGeo.ini* für den Verbund abgespeichert und in die Steuerung geladen. Beim Booten werden diese Informationen aus der ini-Datei eingelesen und sind beim Start des Anwendungsprogramms bekannt.

Sonderfall Änderung der Kinematik-Konfiguration mit `Configure`

Für seltene Ausnahmefälle ist eine Änderung der Kinematik-Konfiguration über die Unterfunktionen `Configure` möglich. Beachten Sie, dass in diesem Fall die im JetSym-Projekt bekannten Mechanikparameter in der Maschine nicht zum Einsatz kommen. Dies kann z. B. beim Service-Einsatz zu Verwirrung führen. Da sich diese mechanischen Parameter in einer Maschine während des Betriebs normalerweise nicht ändern, ist es auch nicht sinnvoll sie im Lauf des Anwendungsprogramms anzupassen.

52.1 Syntax für MCGeo.Mechanism.Kinematics

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Mechanism.Kinematics.<Funktion oder Zugriff auf Wert>(optional <Parameter 1, Parameter 2, Parameter n>);
```

52.2 Eigenschaften von MCGeo.Mechanism.Kinematics

Die Funktionalität bietet folgende Eigenschaften der Funktion `Mechanism`:

- Cartesian2D
- Cartesian3D
- Scara

52.2.1 Cartestian2D

Beschreibung

Mit den Eigenschaften `Offset` und `InvertDirection` lesen Sie die momentan gültigen Kinematikparameter der Koordinaten des Bahnverbundes.

Eigenschaft	Datentyp	Bedeutung
X.Offset	Double	Verschiebung der linearen Position in X-Richtung.
X.InvertDirection	Bool	Zählrichtungsumkehr der linearen Position in X-Richtung.
Y.Offset	Double	Verschiebung der linearen Position in Y-Richtung.
Y.InvertDirection	Bool	Zählrichtungsumkehr der linearen Position in Y-Richtung.

Syntax

```
XVerschiebung := geoGroup.Mechanism.Kinematics.Cartesian2D.X.Offset;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

52.2.2 Cartessian3D

Beschreibung

Mit den Eigenschaften `Offset` und `InvertDirection` lesen Sie die momentan gültigen Kinematikparameter der Koordinaten des Bahnverbundes.

Eigenschaft	Datentyp	Bedeutung
X.Offset	Double	Verschiebung der linearen Position in X-Richtung.
X.InvertDirection	Bool	Zählrichtungsumkehr der linearen Position in X-Richtung.
Y.Offset	Double	Verschiebung der linearen Position in Y-Richtung.
Y.InvertDirection	Bool	Zählrichtungsumkehr der linearen Position in Y-Richtung.
Z.Offset	Double	Verschiebung der linearen Position in Z-Richtung.
Z.InvertDirection	Bool	Zählrichtungsumkehr der linearen Position in Z-Richtung.

Syntax

```
XVerschiebung := geoGroup.Mechanism.Kinematics.Cartesian3D.X.Offset;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

52.2.3 Scara

Die folgende Abbildung zeigt schematisch die grundsätzlichen Begriffe für eine Bahnkinematik am Beispiel eines Scara-Roboters.

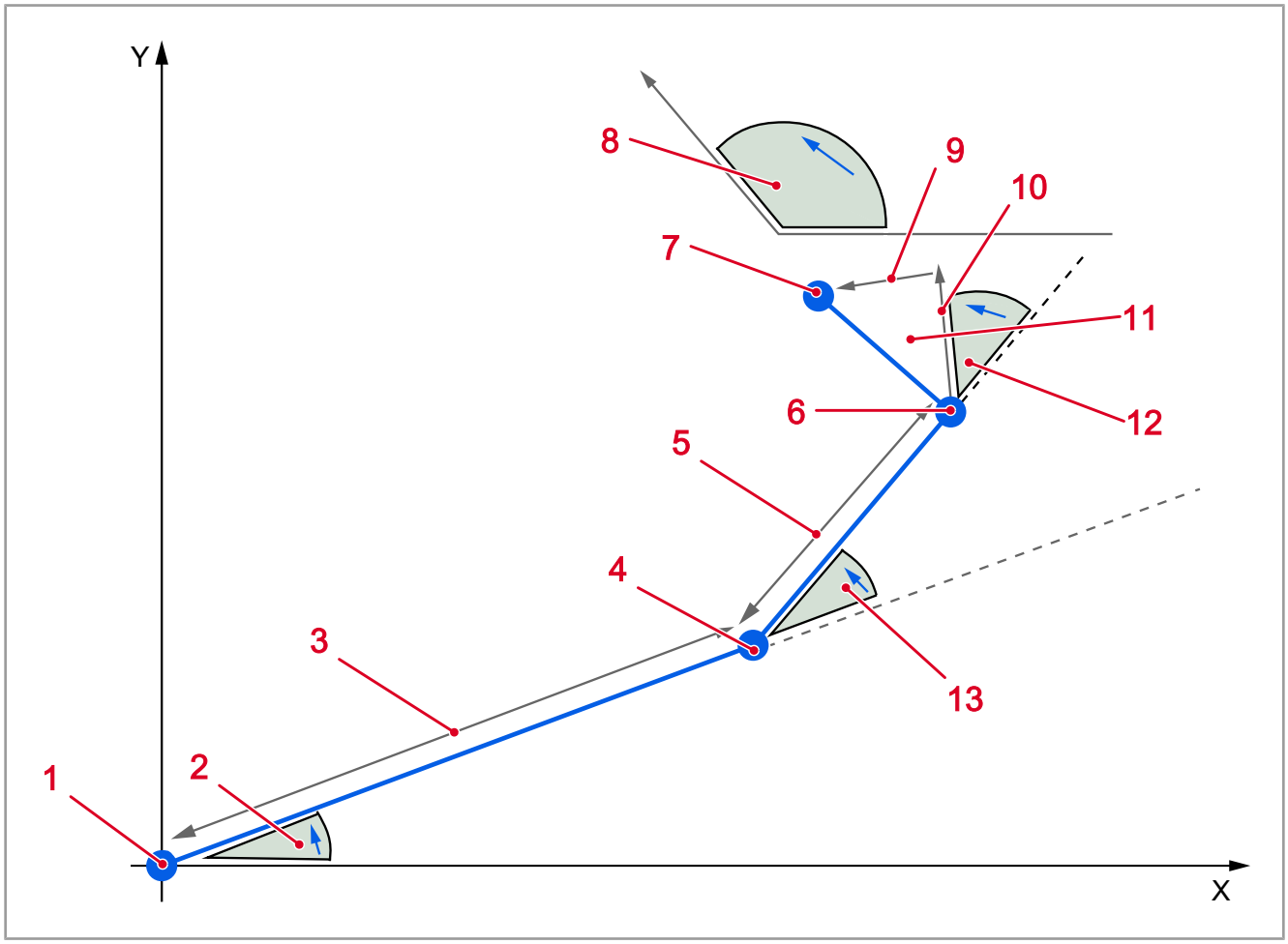


Abb. 9: Scara-Kinematik

1	Schulter
2	Schulterwinkel (Motor 1)
3	Länge Oberarm
4	Ellbogen
5	Länge Unterarm
6	Handgelenk
7	Tool Center Point, TCP = (X, Y, Z)
8	Orientierug (U)
9	TCP_Offset.y
10	TCP_Offset.x
11	TCP-Winkel = Schulterinkel + Ellbogenwinkel + Handgelenkwinkel + TCP-Winkel
12	Handgelenkwinkel (Motor 4)
13	Ellbogenwinkel (Motor 2)

Beschreibung

Mit den Eigenschaften `Offset`, `InvertDirection` lesen Sie die momentan gültigen Kinematikparameter der Koordinaten des Bahnverbundes.

Eigenschaft	Datentyp	Bedeutung
Shoulder.Offset	Double	Verschiebung der Position der Schulter-Achse.
Shoulder.InvertDirection	Bool	Zählrichtungsumkehr der Position der Schulter-Achse.
Shoulder.ArmLength	Double	Armlänge der Schulter-Achse (Oberarm).
Ellbow.Offset	Double	Verschiebung der Position der Ellenbogen-Achse.
Ellbow.InvertDirection	Bool	Zählrichtungsumkehr der Position der Ellenbogen –Achse.
Ellbow.ArmLength	Double	Armlänge der Ellenbogen-Achse (Unterarm).
Vertical.Offset	Double	Verschiebung der Position der Vertikal-Achse.
Vertical.InvertDirection	Bool	Zählrichtungsumkehr der Position der Vertical-Achse.
Wrist.Offset	Double	Verschiebung der Position der Hand-Achse.
Wrist.InvertDirection	Bool	Zählrichtungsumkehr der Position der Hand-Achse.
Wrist.TcpOffsetX	Double	Verschiebung des Tool-Center-Points vom Zentrum der Hand-Achse in X-Richtung.
Wrist.TcpOffsetY	Double	Verschiebung des Tool-Center-Points vom Zentrum der Hand-Achse in Y-Richtung.

Syntax

```
HandVerschiebung := geoGroup.Mechanism.Kinematics.Scara.Wrist.Offset;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

52.3 Konfigurationsfunktionen von MCGeo.Mechanism.Kinematics

Für die erwähnten seltenen Ausnahmefälle können Sie die Konfiguration der Kinematikparameter vom STX-Programm aus vornehmen. Dafür steht ihnen jeweils die Unterfunktion „Configure“ zur Verfügung:

– Cartesian2D.X.Configure(), Cartesian2D.Y.Configure()
– Cartesian3D.X.Configure(),Cartesian3D.Y.Configure()
– Scara.Shoulder.Configure(), Scara.Ellbow.Configure()
– Scara.Vertical.Configure()
– Scara.Wrist.Configure()

52.3.1 Cartesian2D.X.Configure(),Cartesian2D.Y.Configure()

Die Unterfunktion `Configure()` übergibt die Kinematikparameter für die Kinematikdefinition einer Koordinatenachse des Bahnverbundes.

Wird kein Parameter übergeben, so wird der zuletzt übergebene Wert verwendet.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Offset	Double (optional)	Verschiebung der linearen Position.
InvertDirection	Modes (optional)	Zählrichtungsumkehr der linearen Position.
	Modes.On	Aktiviert die Zählrichtungsumkehr.
	Modes.Off	Deaktiviert die Zählrichtungsumkehr.

Syntax

```
geoGroup.Mechanism.Kinematics.Cartesian2D.X.Configure(1200.0,Modes.Off);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

52.3.2 Cartesian3D.X.Configure(), Cartesian3D.Y.Configure()

Die Unterfunktion `Configure()` übergibt die Kinematikparameter für die Kinematikdefinition einer Koordinatenachse des Bahnverbundes.

Wird kein Parameter übergeben, so wird der zuletzt übergebene Wert verwendet.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Offset	Double (optional)	Verschiebung der linearen Position.
InvertDirection	Modes (optional)	Zählrichtungsumkehr der linearen Position.
	Modes.On	Aktiviert die Zählrichtungsumkehr.
	Modes.Off	Deaktiviert die Zählrichtungsumkehr.

Syntax

```
geoGroup.Mechanism.Kinematics.Cartesian3D.X.Configure(1200.0,Modes.Off);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

52.3.3 Scara.Shoulder.Configure(), Scara.Ellbow.Configure()

Die Unterfunktion `Configure()` übergibt die Kinematikparameter für die Kinematikdefinition einer Koordinatenachse des Bahnverbundes.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Offset	Double (optional)	Verschiebung der linearen Position.
InvertDirection	Modes (optional)	Zählrichtungsumkehr der linearen Position.
	Modes.On	Aktiviert die Zählrichtungsumkehr.
	Modes.Off	Deaktiviert die Zählrichtungsumkehr.
ArmLength	Double (optional)	Armlänge der Schulter-/Ellbogenachse.

Syntax

```
geoGroup.Mechanism.Kinematics.Scare.Shoulder.Configure(10.0,Mo-
des.Off,205.5);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

52.3.4 Scara.Vertical.Configure()

Die Unterfunktion `Configure()` übergibt die Kinematikparameter für die Kinematikdefinition einer Koordinatenachse des Bahnverbundes.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Offset	Double (optional)	Verschiebung der linearen Position.
InvertDirection	Modes (optional)	Zählrichtungsumkehr der linearen Position.
	Modes.On	Aktiviert die Zählrichtungsumkehr.
	Modes.Off	Deaktiviert die Zählrichtungsumkehr.

Syntax

```
geoGroup.Mechanism.Kinematics.Scare.Vertical.Configure(25.0,Modes.Off);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

52.3.5 Scara.Wrist.Configure()

Die Unterfunktion `Configure()` übergibt die Kinematikparameter für die Kinematikdefinition einer Koordinatenachse des Bahnverbundes.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Offset	Double (optional)	Verschiebung der linearen Position.
InvertDirection	Modes (optional)	Zählrichtungsumkehr der linearen Position.
	Modes.On	Aktiviert die Zählrichtungsumkehr.
	Modes.Off	Deaktiviert die Zählrichtungsumkehr.
TcpOffsetX	Double (optional)	Verschiebung des Tool-Center-Points vom Zentrum der Hand-Achse in X-Richtung.
TcpOffsetY	Double (optional)	Verschiebung des Tool-Center-Points vom Zentrum der Hand-Achse in Y-Richtung.

Syntax

```
geoGroup.Mechanism.Kinematics.Scare.Wrist.Configure(10.0,Mo-
des.Off,50.0,50.0);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

53 MCGeo.Mechanism.Limitations

Mit der Funktionalität **MCGeo.Mechanism.Limitations** können Sie der MCX-Steuerung die mechanischen Gegebenheiten des Bahnverbunds in der Maschine übergeben. Diese werden als Grenzwerte für die Bewegungsprogrammierung verwendet.

Werden die Werte für die Arbeitsraumbegrenzung durch die Programmierung einer Bewegung überschritten, so wird ein Fehler ausgegeben und die Bewegung nicht gestartet.

Beim Berechnen des Bewegungsprofils werden die Maximalwerte für die Geschwindigkeit, Beschleunigung und, im Fall von ruckbegrenzten Bewegungsarten, auch für den Ruck berücksichtigt. Ergeben sich aus dem Bewegungsbefehl höhere Werte für Geschwindigkeit, Beschleunigung und Ruck, so werden diese automatisch und ohne Fehlermeldung auf die Maximalwerte begrenzt.

Zur Veranschaulichung der kinematischen Größen zeigt die folgende Abbildung einen schematischen mechanischen Aufbau am Beispiel eines Scara-Roboters.

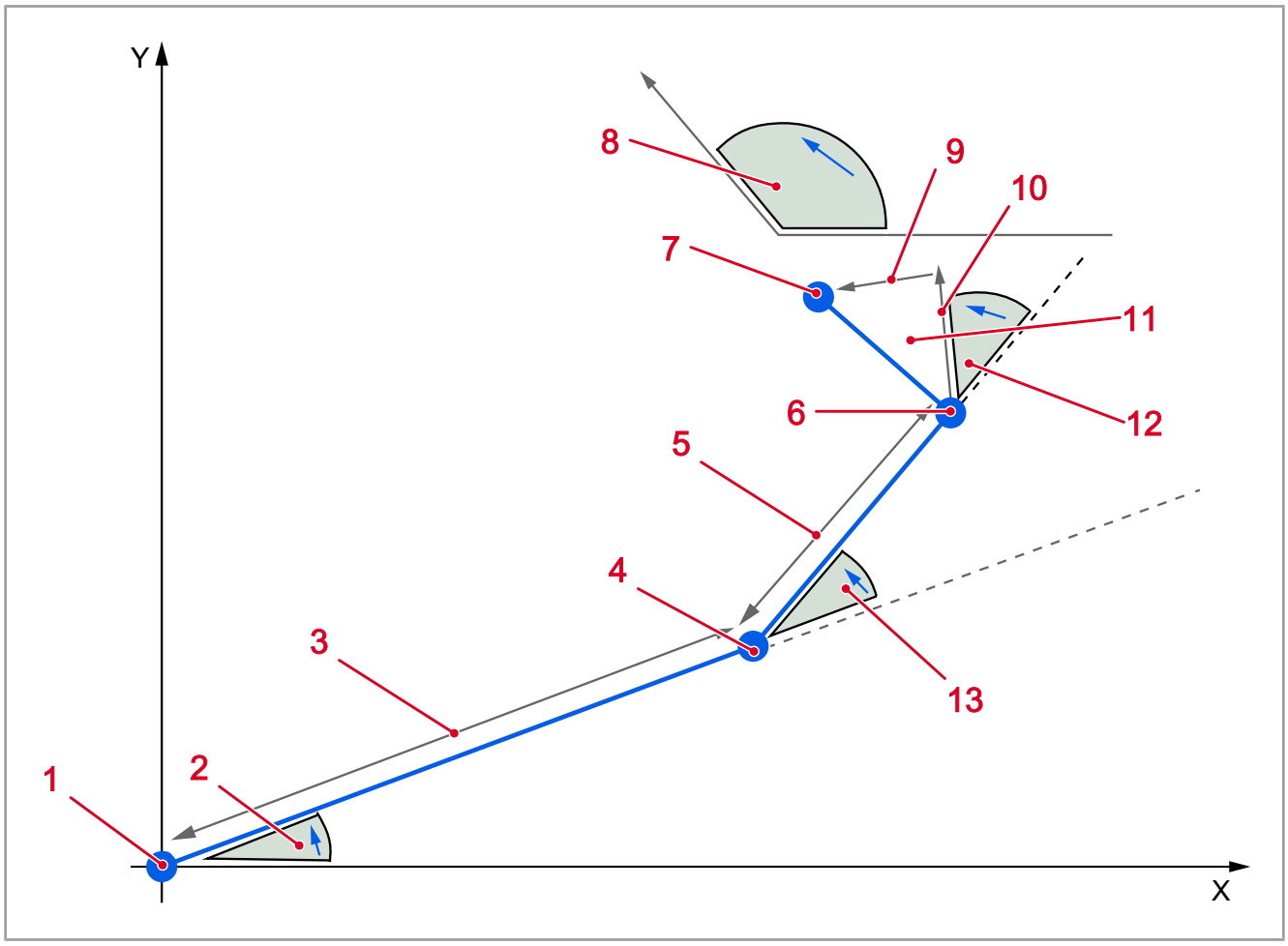


Abb. 10: Scara-Kinematik

1	Schulter
2	Schulterwinkel (Motor 1)
3	Länge Oberarm
4	Ellbogen
5	Länge Unterarm
6	Handgelenk
7	Tool Center Point, TCP = (X, Y, Z)
8	Orientierug (U)

9	TCP_Offset.y
10	TCP_Offset.x
11	TCP-Winkel = Schulterinkel + Ellbogenwinkel + Handgelenkwinkel + TCP-Winkel
12	Handgelenkwinkel (Motor 4)
13	Ellbogenwinkel (Motor 2)

Die Orientierung des Tool Center Points (TCP) bezeichnet die sogenannten **Kardanwinkel**, in der Drehachsenreihenfolge ***z – y‘ – x‘‘***, in Bezug auf das Basis-Koordinatensystem.

Ermittlung der Kardanwinkel

Um aus dem Basis-Koordinatensystem (x, y, z) das Koordinatensystem des TCP ***(x‘‘‘, y‘‘‘, z‘‘‘)*** abzuleiten, wird folgendes Schema angewandt:

1. Zuerst wird das ortsfeste Koordinatensystem um den Winkel U um die ***z***-Achse gedreht. Es entsteht das Koordinatensystem ***(x‘, y‘, z‘)***.
2. Dieses wird nun um den Winkel V um die ***y‘***-Achse gedreht, wobei das Koordinatensystem ***(x‘‘, y‘‘, z‘‘)*** entsteht.
3. Zuletzt wird dieses Koordinatensystem um den Winkel W um die ***x‘‘***-Achse gedreht. Dadurch entsteht das Koordinatensystem des TCP ***(x‘‘‘, y‘‘‘, z‘‘‘)***.

Bedeutung der Kardanwinkel

– Der Winkel U (auch Gierwinkel) beschreibt die Ausrichtung des TCP in Bezug auf die ***x-y***-Ebene des Basiskoordinatensystems.
– Der Winkel V (auch Nickwinkel) beschreibt die Ausrichtung des TCP in Bezug auf die ***y-z***-Ebene des Basiskoordinatensystems.
– Der Winkel W (auch Rollwinkel) beschreibt die Drehung des TCP um die ***x***-Achse des TCP-Koordinatensystems.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

53.1 Syntax für MCGeo.Mechanism.Limitations

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Mechanism.Limitations.<Funktion oder Zugriff auf Wer-
t>(optional <Parameter 1, Parameter 2, Parameter n>);
```

53.2 Eigenschaften von MCGeo.Mechanism.Limitations

Die Funktionalität bietet folgende Eigenschaften der Funktion `Mechanism`:

– WorkspaceLimitMin
– WorkspaceLimitMax
– OrientationLimitMin
– OrientationLimitMax
– OrientationFilter

– MaxPathVelocity
– MaxPathAcceleration
– MaxPathJerk
– SingularityDistance1

53.2.1 WorkspaceLimitMin

Beschreibung

Die Eigenschaft liest oder schreibt den kleinsten Wert für die Arbeitsbereichskoordinaten des Bahnverbunds. Die Werte werden entweder einzeln für die Verbundkoordinaten X, Y und Z oder über eine Variable vom Typ `Position3D` übergeben und gelesen.

Eigenschaft	Datentyp	Bedeutung
WorkspaceLimitMin	Position3D	Kleinste Positionswerte für die Raumkoordinaten im Arbeitsbereich; absoluter Wert bezogen auf den Referenzpunkt.
WorkspaceLimitMin.X	Double	Kleinster Positionswert für die Raumkoordinate X im Arbeitsbereich.
WorkspaceLimitMin.Y	Double	Kleinster Positionswert für die Raumkoordinate Y im Arbeitsbereich.
WorkspaceLimitMin.Z	Double	Kleinster Positionswert für die Raumkoordinate Z im Arbeitsbereich.

Syntax

```
ArbeitsbereichMin := geoGroup.Mechanism.Limitations.WorkspaceLimitMin;
geoGroup.Mechanism.Limitations.WorkspaceLimitMin := -100.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

53.2.2 WorkspaceLimitMax

Beschreibung

Die Eigenschaft liest oder schreibt den größten Wert für die Arbeitsbereichskoordinaten des Bahnverbunds. Die Werte werden entweder einzeln für die Verbundkoordinaten X, Y und Z oder über eine Variable vom Typ `Position3D` übergeben und gelesen.

Eigenschaft	Datentyp	Bedeutung
WorkspaceLimitMax	Position3D	Größte Positionswerte für die Raumkoordinaten im Arbeitsbereich; absoluter Wert bezogen auf den Referenzpunkt.
WorkspaceLimitMax.X	Double	Größter Positionswert für die Raumkoordinate X im Arbeitsbereich.
WorkspaceLimitMax.Y	Double	Größter Positionswert für die Raumkoordinate Y im Arbeitsbereich.
WorkspaceLimitMax.Z	Double	Größter Positionswert für die Raumkoordinate Z im Arbeitsbereich.

Syntax

```
ArbeitsbereichMax := geoGroup.Mechanism.Limitations.WorkspaceLimitMax;
geoGroup.Mechanism.Limitations.WorkspaceLimitMax := 1000.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

53.2.3 OrientationLimitMin

Beschreibung

Die Eigenschaft liest oder schreibt den kleinsten Wert für die Orientierungswinkel im Arbeitsbereich des Bahnverbunds. Die Werte werden entweder einzeln für die Verbundorientierungen U, V und W oder über eine Variable vom Typ `Position3D` übergeben und gelesen.

Eigenschaft	Datentyp	Bedeutung
OrientationLimitMin	Position3D	Kleinster Orientierungswinkelsatz (Kardanwinkel U, V, W).

Eigenschaft	Datentyp	Bedeutung
OrientationLimitMin.U	Double	Kleinster Orientierungswinkel U.
OrientationLimitMin.V	Double	Kleinster Orientierungswinkel V.
OrientationLimitMin.W	Double	Kleinster Orientierungswinkel W.

Syntax

```
OrientierungsbereichMin := geoGroup.Mechanism.Limitations.OrientationLimit-
Min;
geoGroup.Mechanism.Limitations.OrientationLimitMin.U := 0.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

53.2.4 OrientationLimitMax

Beschreibung

Die Eigenschaft liest oder schreibt den größten Wert für die Orientierungswinkel im Arbeitsbereich des Bahnverbunds. Die Werte werden entweder einzeln für die Verbundorientierungen U, V und W oder über eine Variable vom Typ `Position3D` übergeben und gelesen.

Eigenschaft	Datentyp	Bedeutung
OrientationLimitMax	Position3D	Größter Orientierungswinkelsatz (Kardanwinkel U, V, W).
OrientationLimitMax.U	Double	Größter Orientierungswinkel U.
OrientationLimitMax.V	Double	Größter Orientierungswinkel V.
OrientationLimitMax.W	Double	Größter Orientierungswinkel W.

Syntax

```
OrientierungsbereichMax := geoGroup.Mechanism.Limitations.OrientationLimit-
Max;
geoGroup.Mechanism.Limitations.OrientationLimitMax.U := 360.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

53.2.5 OrientationFilter

Beschreibung

Die Eigenschaft liest oder schreibt den Wert für die Filtereinstellung der Orientierungsnachführung des Bahnverbundes. Da die Orientierung nicht in die Berechnung der Bahn einfließt, wird sie der Bahnbewegung nachgeführt. Speziell bei Überschleifbewegungen würde ein Verfahren ohne Filter dazu führen, dass die Orientierung im Überschleifbereich einen Sprung macht. Durch den Filter wird die Orientierung jedoch auch in diesem Fall weich nachgeführt.

Eigenschaft	Datentyp	Bedeutung
OrientationFilter	Double	Filterwert für die Nachführung der Orientierung. Filterwirkung in %. 0% = keine Filterwirkung; 100% höchste Filterwirkung; Standard: 70 %; in der Regel sollte diese Einstellung ausreichend sein.

Syntax

```
OrientierungsFilter := geoGroup.Mechanism.Limitations.OrientationFilter;
geoGroup.Mechanism.Limitations.OrientationFilter := 50.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

53.2.6 MaxPathVelocity

Beschreibung

Datentyp	Bedeutung
Double	Liest oder schreibt den Wert für die maximale Bahngeschwindigkeit, mit der der TCP bewegt werden kann oder soll [in der Anwendereinheit der Geschwindigkeit]. Dieser Wert kann z. B. durch die Funktion der Maschine oder auch durch die Maximaldaten der am Antriebssystem beteiligten Komponenten bestimmt sein. Die Sollgeschwindigkeit der Bahn wird auf diesen Wert begrenzt, auch wenn sich durch die Parametrierung der Bewegung eine höhere Geschwindigkeit ergeben würde.

Syntax

```
Maximalgeschwindigkeit := geoGroup.Mechanism.Limitations.MaxPathVelocity;
geoGroup.Mechanism.Limitations.MaxPathVelocity := 200.5; // mm/s
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

53.2.7 MaxPathAcceleration

Beschreibung

Datentyp	Bedeutung
Double	Liest oder schreibt den Wert für die maximale Bahnbeschleunigung, mit der der TCP bewegt werden kann oder soll in der Anwendereinheit der Geschwindigkeit. Dieser Wert kann z. B. durch die Funktion der Maschine oder auch durch die Maximaldaten der am Antriebssystem beteiligten Komponenten bestimmt sein. Die Sollbeschleunigung und -verzögerung der Bahn wird auf diesen Wert begrenzt, auch wenn sich durch die Parametrierung der Bewegung eine höhere Geschwindigkeit ergeben würde.

Syntax

```
Maximalbeschl := geoGroup.Mechanism.Limitations.MaxPathAcceleration;
geoGroup.Mechanism.Limitations.MaxPathAcceleration := 1300.0; // mm/s
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

53.2.8 MaxPathJerk

Beschreibung

Datentyp	Bedeutung
Double	Liest oder schreibt den Wert für den maximalen Bahnruck, mit dem der TCP bewegt werden kann oder soll in der Anwendereinheit der Beschleunigung. Dieser Wert kann z. B. durch die Funktion der Maschine oder auch durch die Maximaldaten der am Antriebssystem beteiligten Komponenten bestimmt sein. Die Sollruck der Bahn wird auf diesen Wert begrenzt, auch wenn sich durch die Parametrierung der Bewegung eine höhere Geschwindigkeit ergeben würde.

Syntax

```
Maximalruck := geoGroup.Mechanism.Limitations.MaxPathJerk;
geoGroup.Mechanism.Limitations.MaxPathJerk := 100000.0; // mm/s³
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

53.2.9 SingularityDistance1

Eine Singularität tritt bei seriellen Kinematiken dann auf, wenn die Bewegungsfreiheit aufgrund der Stellung der einzelnen Gelenke eingeschränkt ist. Dies ist z. B. der Fall, wenn ein Roboter vollständig gestreckt ist. Es kann vor Erreichen einer Singularität vorkommen, dass die Geschwindigkeit eines Gelenks sehr hoch werden muss, um der Sollgeschwindigkeit des TCP zu folgen. Deshalb ist es sinnvoll diese singulären Stellungen nicht nur zu vermeiden, sondern auch einen gewissen Abstand zu diesen Positionen zu sicherzustellen. Der MCX-Kernel gibt deshalb einen Fehler aus, wenn die Position des TCP einer singulären Stellung so nahe kommt, dass der Abstand kleiner wird, als der über die Eigenschaft `SingularityDistance1` definierte Grenzabstand.

Beschreibung

Datentyp	Bedeutung
Double	Liest oder schreibt den Wert für den Singularitätsabstand, der eingehalten werden muss, um Fehler wegen des Erreichens einer singulären Stellung zu verhindern [in der Anwendereinheit der Position].

Syntax

```
Singularitaetsgrenze := geoGroup.Mechanism.Limitations.SingularityDistance1;
geoGroup.Mechanism.Limitations.SingularityDistance1:= 10.0;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben

Ausgangszustand	Erlaubt	Endzustand
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

54 MCGeo.Mechanism.Slope

Mit der Funktionalität **MCGeo.Mechanism.Slope** können Sie die berechneten Eigenschaften der Bewegung des nächsten Verfahrsatzes eines Bahnverbundes lesen oder anpassen.

Eine Positionierbewegung besteht aus folgenden Phasen:

1. Beschleunigungsrampe
2. Gegenbenenfalls: Fahrt mit konstanter Geschwindigkeit;

   Die Beschleunigungsrampe kann auch direkt in die Verzögerungsrampe übergehen (Dreiecksbetrieb).
3. Verzögerungsrampe

Daneben kann ein Verschleifpolynom gefahren werden, wenn zwei Bahnstücke ohne Anhalten ineinander übergehen sollen.

Bei Bahnverbunden ergibt sich der Verlauf der Bahngeschwindigkeit durch eine trapezförmige Ausführung der Beschleunigung. Die Steilheit der Rampe im Trapez wird durch die Vorgabe des maximalen Bahnrucks bestimmt. Außerdem sind die Beschleunigung und Verzögerung im Bahnstück gleich. Beide werden über die Eigenschaft `Acceleration` gemeinsam geändert.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

54.1 Syntax für MCGeo.Mechanism.Slope

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Mechanism.Slope.<Zugriff auf Wert>
```

54.2 Eigenschaften von MCGeo.Mechanism.Slope

Die Funktionalität bietet folgende Eigenschaften:

- SlopeState
- IsAccelerating
- IsAtConstantSpeed
- IsDecelerating
- IsStopped
- ActualMaxSpeed
- ActualMaxAcceleration
- SpeedSetpoint
- AccelerationSetpoint
- Speed
- SpeedOverride
- MaxVeloPoly
- Acceleration

54.2.1 SlopeState

Beschreibung

Datentyp	Bedeutung
MCGeoSlopeStates	Liest, in welcher Phase der Bewegung (Beschleunigung, konstante Geschwindigkeit, Verzögerung oder Stillstand) sich der Bahnsollwert des Bahnverbunds aktuell befindet. Damit können Sie beispielsweise eine Aktion in der Maschine erst dann aktivieren, wenn der TCP die Beschleunigungsphase verlassen hat.
GeoSlopeStates.Accelerating	Verbund befindet sich in der Beschleunigungsrampe.
GeoSlopeStates.AtConstandSpeed	Verbund fährt mit konstanter Geschwindigkeit.
GeoSlopeStates.Decelerating	Verbund befindet sich in der Verzögerungsrampe.
GeoSlopeStates.Stopped	Verbund steht still.

Syntax

```
Rampenstatus := geoGroup.Mechanism.Slope.SlopeState;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

54.2.2 IsAccerlerating

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Bahnsollwerte in der Beschleunigungsrampe befinden.

Syntax

```
If geoGroup.Mechanism.Slope.IsAccelerating then …
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

54.2.3 IsAtConstantSpeed

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich Bahnsollwerte in der Phase Fahrt mit konstanter Geschwindigkeit befinden.

Syntax

```
If geoGroup.Mechanism.Slope.IsAtConstantSpeed then …
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

54.2.4 IsDecelerating

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Bahnsollwerte in der Verzögerungsrampe befinden.

Syntax

```
If geoGroup.Mechanism.Slope.IsDecelerating then …
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

54.2.5 IsStopped

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Bahnsollwerte in der Phase des Stillstands befinden.

Syntax

```
If geoGroup.Mechanism.Slope.IsStopped then …
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

54.2.6 ActualMaxSpeed

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert der maximalen Bahngeschwindigkeit, der im Sollwertgenerator des Verbunds aktuell aktiv ist. Dieser Wert unterscheidet sich vom Wert der Eigenschaft Speed [▶ 312] beispielsweise dann, wenn Bewegungen im Vorlaufpuffer abgearbeitet werden, jedoch der Wert für die maximale Geschwindigkeit inzwischen geändert wurde.

Syntax

```
MaxGeschwindigkeit := geoGroup.Mechanism.Slope.ActualMaxSpeed;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

54.2.7 ActualMaxAcceleration

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert der maximalen Bahnbeschleunigung, der dem Sollwertgenerator des Verbunds aktiv ist. Dieser Wert unterscheidet sich vom Wert der Eigenschaft Acceleration [▶ 313] beispielsweise dann, wenn Bewegungen im Vorlaufpuffer abgearbeitet werden, jedoch der Wert für die maximale Beschleunigung inzwischen geändert wurde.

Syntax

```
MaxBeschleunigung := geoGroup.Mechanism.Slope.ActualMaxAcceleration;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

54.2.8 SpeedSetpoint

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert der Soll-Bahngeschwindigkeit, der im Sollwertgenerator der MCX-Achse momentan wirksam ist. Das Vorzeichen des Werts ist immer positiv.

Syntax

```
ActGeschwSollwert := geoGroup.Mechanism.Slope.SpeedSetpoint;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

54.2.9 AccelerationSetpoint

Beschreibung

Datentyp	Bedeutung
Double	Liest den Wert der Sollbeschleunigung, der im Sollwertgenerator des Verbunds momentan wirksam ist. Das Vorzeichen des Werts zeigt eine Beschleunigung oder Verzögerung an.

Syntax

```
ActBeschlSollwert := geo.Mechanism.Slope.AccelerationSetpoint;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

54.2.10 Speed

Beschreibung

Datentyp	Bedeutung
Double	Liest oder schreibt den Wert für die maximale Geschwindigkeit, der für die Bahnsollwertgenerierung einer Bahnbewegung verwendet wird. Dieser Wert wird normalerweise in den Bewegungsbefehlen als Parameter übergeben. Die Änderung des Werts wirkt sich erst nach der Abarbeitung des Vorlaufpuffers aus.

Syntax

```
ProfilGeschw := geoGroup.Mechanism.Slope.Speed;
geoGroup.Mechanism.Slope.Speed := 300.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

54.2.11 MaxVeloPoly

Beschreibung

Datentyp	Bedeutung
Double	Liest oder schreibt den Wert für die maximale Geschwindigkeit, der für die Bahnsollwertgenerierung für die Fahrt auf Überschleifpolynomen verwendet wird. Dadurch wird auf dem vorhergehenden Verfahrsatz auf diese Geschwindigkeit abgebremst, wenn die Geschwindigkeit auf dem Verfahrsatz höher war. Falls ein höherer Wert als der des Versfahrsatzes programmiert wird, wird auf die Verfahsatzgeschwindigkeit begrenzt. Die Änderung des Werts wirkt sich erst nach der Abarbeitung des Vorlaufpuffers aus.

Syntax

```
UeberschleifGeschw := geoGroup.Mechanism.Slope.MaxVeloPoly;
geoGroup.Mechanism.Slope.MaxVeloPoly := 300.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

54.2.12 SpeedOverride

Beschreibung

Die Geschwindigkeitsübersteuerung verändert die Sollgeschwindigkeit im Bahnsollwertgenerator gegenüber der im Bewegungsbefehl programmierten Geschwindigkeit um einen bestimmten Prozentsatz.

Die maximal mögliche Geschwindigkeit je Bahnbewegung wird durch den *Look Ahead* bestimmt. Der Look Ahead berechnet die Bahn unter Einbeziehung der programmierten Bewegung und der konfigurierten Begrenzungswerte und ist somit auch an die programmierte Geschwindigkeit geknüpft. Es kann sich dadurch ein Unterschied ergeben, ob der Verbund z.B. mit `Speed = 100mm/s` und `SpeedOveride = 150%` oder mit `Speed = 150mm/s` und `SpeedOverride = 100%` fahren! Um einen möglichst gleichmäßigen Geschwindigkeitsverlauf zu ermöglichen, ist es deshalb ratsam, mit der höchsten zu fahrenden Bahngeschwindigkeit zu programmieren und für `SpeedOverride` einen Wert ≤ `100 %` zu verwenden.

Als Basis für `SpeedOverride` wird die im Interpolator tatsächlich berechnete Sollgeschwindigkeit verwendet. Wird z. B. durch eine wirksame Ruckbegrenzung eine kleinere Sollgeschwindigkeit als die programmierte berechnet, so bezieht sich der Prozentsatz von `SpeedOverride` auf diese kleinere Sollgeschwindigkeit und nicht auf die programmierte Geschwindigkeit.

Datentyp	Bedeutung
Double	Liest oder schreibt den Wert für die Geschwindigkeitsübersteuerung, der für die Bahnsollwertgenerierung einer Bahnbewegung verwendet wird. Dieser Wert wird normalerweise in den Bewegungsbefehlen als Parameter übergeben. Die Änderung des Werts wirkt sich sofort und auf alle Bahnbewegungsarten aus, solange die Bahnbewegung nicht in einer Beschleunigungs- oder Verzögerungsrampe befindet. `SpeedOverride` ist generell auf 200 % beschränkt.

Syntax

```
GeschwUebersteuerung := geoGroup.Mechanism.Slope.SpeedOverride;
geoGroup.Mechanism.Slope.SpeedOverride := 50.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

54.2.13 Acceleration

Beschreibung

Datentyp	Bedeutung
Double	Liest oder schreibt den Wert für die maximale Beschleunigung, der für die Bahnsollwertgenerierung einer Bahnbewegung verwendet wird. Dieser Wert wird normalerweise in den Bewegungsbefehlen als Parameter übergeben. Die Änderung des Werts wirkt sich erst nach der Abarbeitung des Vorlaufpuffers aus.

Syntax

```
ProfilBeschleunigung := geoGroup.Mechanism.Slope.Acceleration;
geoGroup.Mechanism.Slope.Acceleration := 5500.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

55 MCGeo.Mechanism.Units

Mit der Funktionalität **MCGeo.Mechanism.Units** können Sie die verwendeten Einheiten für die Werte von Positionen, Geschwindigkeiten, Beschleunigungen und Ruck lesen oder anpassen.

Diese Einheiten werden üblicherweise bei der Verbunddefinition festgelegt und können mithilfe dieser Befehle im STX-Programm abgefragt werden.

HINWEIS



Veränderte Positionen durch Wechsel der Einheit im Maschinenbetrieb

Es ist möglich, die Einheiten während des Betriebs der Maschine zu ändern. Da jedoch der Positionierbefehl nur die Werte ohne die Angabe einer Einheit enthält, verändert dadurch sich die Bedeutung der Werte.

Beispiel: Die Umstellung von Millimeter auf Zoll bewirkt bei einem Wert von `25.0` im Positionierbefehl eine andere Position in der Maschine.

- ► Vermeiden Sie die Umstellung der Einheiten während des Betriebs.
- ► Nehmen Sie eine Umstellung der Einheiten mit Bedacht vor.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

55.1 Syntax für MCGeo.Mechanism.Units

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Mechanism.Units.<Funktion oder Zugriff auf Wert>(optional <Parameter 1, Parameter 2, Parameter n>);
```

55.2 Funktionen von MCGeo.Mechanism.Units

Die Funktionalität bietet folgende Funktionen:

– SetUnits()

55.2.1 SetUnits()

Mit der Funktion verändern sie alle Einheiten für Positionen, Geschwindigkeiten, Beschleunigungen und Ruck in einer Funktion für eine Bahnbewegung. Über die Parameter können Sie für jede Einheit unterschiedliche Bedeutungen vorgeben.

Übergabeparameter

Parameter	Datentyp	Bedeutung
PositionUnit	**MCLinearPositionUnits** (optional) – MCLinearPositionUnits.MM – MCLinearPositionUnits.INCH – MCLinearPositionUnits.MM_10TH – MCLinearPositionUnits.MM_100TH	Einheit für die linearen Positionswerte. Mögliche Einheiten: – Millimeter – Zoll – 1/10 Millimeter – 1/100 Millimeter
SpeedUnit	**MCLinearSpeedUnits** (optional) – MCLinearSpeedUnits.MM_PER_SEC – MCLinearSpeedUnits.MM_PER_MIN – MCLinearSpeedUnits.M_PER_SEC – MCLinearSpeedUnits.M_PER_MIN – MCLinearSpeedUnits.INCH_PER_SEC – MCLinearSpeedUnits.INCH_PER_MIN	Einheit für die linearen Geschwindigkeitswerte. Mögliche Einheiten: – Millimeter pro Sekunde – Millimeter pro Minute – Meter pro Sekunde – Meter pro Minute – Zoll pro Sekunde – Zoll pro Minute
AccelerationUnit	**MCLinearAccelerationUnits** (optional) – MCLinearAccelerationUnits.MM_PER_S2 – MCLinearAccelerationUnits.M_PER_S2 – MCLinearAccelerationUnits.INCH_PER_S2	Einheit für die linearen Beschleunigungswerte. Mögliche Einheiten: – Millimeter pro Sekunde² – Meter pro Sekunde² – Zoll pro Sekunde²
JerkUnit	**MCLinearJerkUnits** (optional) – MCLinearJerkUnits.MM_PER_S3 – MCLinearJerkUnits.M_PER_S3 – MCLinearJerkUnits.INCH_PER_S3	Einheit für die linearen Ruckwerte. Mögliche Einheiten: – Millimeter pro Sekunde³ – Meter pro Sekunde³ – Zoll pro Sekunde³
OrientationUnit	**MCGeoOrientationUnits** (optional) – MCGeoOrientationUnits.RPY_Angles	Einheit für die Orientierung. Mögliche Einheiten: – Kardanwinkel (s. MCGeo.Orientation)

Syntax

```
geoGroup.Mechanism.Units.SetUnits(
    MCLinearPositionUnits.MM,
    MCLinearSpeedUnits.MM_PER_SEC,
    MCLinearAccelerationUnits.MM_PER_S2,
    MCLinearJerkUnits.MM_PER_S3
    MCGeoOrientationUnits.RPY.Angels);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

Das Schreiben der Funktion ist jedem Zustand möglich. Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

55.3 Eigenschaften von MCGeo.Mechanism.Units

Die Funktionalität bietet folgende Eigenschaften der Funktion `Mechanism`:

– PositionUnit
– OrientationUnit
– SpeedUnit
– AccelerationUnit
– JerkUnit

55.3.1 PositionUnit

Beschreibung

Liest oder schreibt die Einheit, in der die Positionswerte für eine linear definierte Achse verarbeitet und angezeigt werden.

Datentyp	Bedeutung
MCLinearPositionUnits (optional) – MCLinearPositionUnits.MM – MCLinearPositionUnits.INCH – MCLinearPositionUnits.MM_10TH – MCLinearPositionUnits.MM_100TH	Einheit für die linearen Positionswerte. Mögliche Einheiten: – Millimeter – Zoll – 1/10 Millimeter – 1/100 Millimeter

Syntax

```
PositionsEinheiten := geoGroup.Mechanism.Units.PositionUnit;
geoGroup.Mechanism.Units.PositionUnit := MCLinearPositionUnits.MM_10TH;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

55.3.2 OrientationUnit

Beschreibung

Liest oder schreibt die Einheit, in der die Orientierung der Bahnbewegung verarbeitet und angezeigt wird.

Datentyp	Bedeutung
MCGeoOrientationUnits (optional) – MCGeoOrientationUnits.RPY_Angles – MCGeoOrientationUnits.TCP_Vector	Einheit für die Orientierung. Mögliche Einheiten: – Kardanwinkel (siehe MCGeo.Orientation [▶ 377])

Syntax

```
Ori_Einheiten := geoGrouß.Mechanism.Units.OrientationUnit;
geoGroup.Mechanism.Units.OrientationUnit := MCLinearOrientationU-
nits.RPY_Angles;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

55.3.3 SpeedUnit

Beschreibung

Liest oder schreibt die Einheit, in der die Geschwindigkeitswerte für Bahnbewegung verarbeitet und angezeigt werden.

Datentyp	Bedeutung
MCLinearSpeedUnits (optional) – MCLinearSpeedUnits.MM_PER_SEC – MCLinearSpeedUnits.MM_PER_MIN – MCLinearSpeedUnits.M_PER_SEC – MCLinearSpeedUnits.M_PER_MIN – MCLinearSpeedUnits.INCH_PER_SEC – MCLinearSpeedUnits.INCH_PER_MIN	Einheit für die linearen Geschwindigkeitswerte. Mögliche Einheiten: – Millimeter pro Sekunde – Millimeter pro Minute – Meter pro Sekunde – Meter pro Minute – Zoll pro Sekunde – Zoll pro Minute

Syntax

```
GewchwEinheiten := geoGroup.Mechanism.Units.SpeedUnit;
geoGroup.Mechanism.Units SpeedUnit := MCLinearSpeedUnits.M_SEC;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

55.3.4 AccelerationUnit

Beschreibung

Liest oder schreibt die Einheit, in der die Beschleunigungswerte für die Bahnbewegung verarbeitet und angezeigt werden.

Datentyp	Bedeutung
MCLinearAccelerationUnits (optional) – MCLinearAccelerationUnits.MM_PER_S2 – MCLinearAccelerationUnits.M_PER_S2 – MCLinearAccelerationUnits.INCH_PER_S2	Einheit für die linearen Geschwindigkeitswerte. Mögliche Einheiten: – Millimeter pro Sekunde² – Meter pro Sekunde² – Zoll pro Sekunde²

Syntax

```
BeschleunigungsEinheiten := geoGroup.Mechanism.Units.AccelerationUnit;
geoGroup.Mechanism.Units.AccelerationUnit := MCLinearAccelerationU-
nits.M_PER_S2;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

55.3.5 JerkUnit

Beschreibung

Liest oder schreibt die Einheit, in der die Ruckwerte für die Bahnbewegung verarbeitet und angezeigt werden.

Datentyp	Bedeutung
MCLinearJerkUnits (optional) – MCLinearJerkUnits.MM_PER_S3 – MCLinearJerkUnits.M_PER_S3 – MCLinearJerkUnits.INCH_PER_S3	Einheit für die linearen Geschwindigkeitswerte. Mögliche Einheiten: – Millimeter pro Sekunde³ – Meter pro Sekunde³ – Zoll pro Sekunde³

Syntax

```
RuckEinheiten := geoGroup.Mechanism.Units.JerkUnit;
geoGroup.Mechanism.Units.JerkUnit := MCLinearJerkUnits.MM_PER_S3;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Es wird jedoch empfohlen, Einheiten nur im Zustand **Gesperrt** zu ändern.

56 MCGeo.Members

Die Funktionalität **MCGeo.Members** nutzen Sie in erster Linie dazu, einem Bahnverbund Synchronachsen hinzuzufügen oder diese aus dem Bahnverbund entfernen. Synchronachsen sind Achsen, die synchron zur Bahnbewegung verfahren werden, ohne dass sie mechanisch an der Bahnbewegung teilnehmen. Eine Beschreibung der Achstypen finden Sie unter MCAxis.Mechanism.AxisMode [▶ 118].

Für sehr spezielle Fälle können Sie auch Verbundachsen aus dem STX-Programm heraus hinzufügen oder entfernen. Da dies nur in Ausnahmen empfohlen wird, werden die entsprechenden Befehle hier nicht beschrieben. Bitte setzen Sie sich mit Bucher Automation in Verbindung, wenn Sie diese Funktionen verwenden möchten.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

56.1 Syntax für MCGeo.Members

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Members.<Funktion oder Zugriff auf Wert>(optional <Parameter>);
```

56.2 Funktionen von MCGeo.Members

Die Funktionalität bietet folgende Funktionen:

– AddSyncAxis()
– RemoveSyncAxis()

56.2.1 AddSyncAxis()

Mit der Funktion fügen Sie dem Bahnverbund eine Synchronachse hinzu. Diese erhält danach den Betriebszustand **Inaktiv** und wird über den Bahnverbund bewegt.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Name der MCX-Achse	Pointer to MCAxis	Achsobjekt, das synchron mit dem Verbund bewegt werden soll.

Syntax

```
geoGroup.Members.AddSyncAxis(axSyncAxis);
```

Zustandsverwaltung

Damit die Funktionen fehlerfrei ausgeführt wird, müssen Bahnverbund und Synchronachse den gleichen Betriebszustand haben. Dies kann entweder der Zustand **Gesperrt** oder **Freigegeben** sein.

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Fehler: freigegeben
Freigegeben	√	Fehler: freigegeben
Läuft	X	Fehler: freigegeben
Unterbrochen	X	Fehler: freigegeben
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Das Aufrufen der Funktion ist jedem Zustand der Leit- und Folgeachse möglich.

56.2.2 RemoveSyncAxis()

Mit der Funktion entfernen Sie eine Synchronachse aus dem Bahnverbund. Die entfernte Achse erhält den Betriebszustand, den der Bahnverbund vor dem Ausführen des Befehls hatte. Dies ist entweder **Freigegeben** oder **Gesperrt**.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Name der MCX-Achse	Pointer to MCAxis	Achsobjekt, das aus dem Verbund bewegt werden soll.

Syntax

```
geoGroup.Members.RemoveSyncAxis(axSyncAxis);
```

Zustandsverwaltung

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Fehler: freigegeben
Freigegeben	√	Fehler: freigegeben
Läuft	X	Fehler: freigegeben
Unterbrochen	X	Fehler: freigegeben
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Das Aufrufen der Funktion ist jedem Zustand der Leit- und Folgeachse möglich.

57 MCGeo.MoveCircle2D

Mit der Funktionalität **MCGeo.MoveCircle2D** können Sie eine Kreisbewegung des Tool Center Points (TCP) in einer Raum-Ebene eines MCX-Bahnverbundes starten und parametrieren. Orthogonal zu dieser Ebene kann eine Linearbewegung stattfinden, so dass der TCP eine Helix-Bewegung beschreibt.

Definiert wird die Kreisbewegung über die Mittelpunktkoordinaten, die Ausgangskoordinaten und den Winkel des Kreissegments. Des Weiteren werden Länge und Richtung der Bewegung in orthogonaler Richtung vorgegeben.

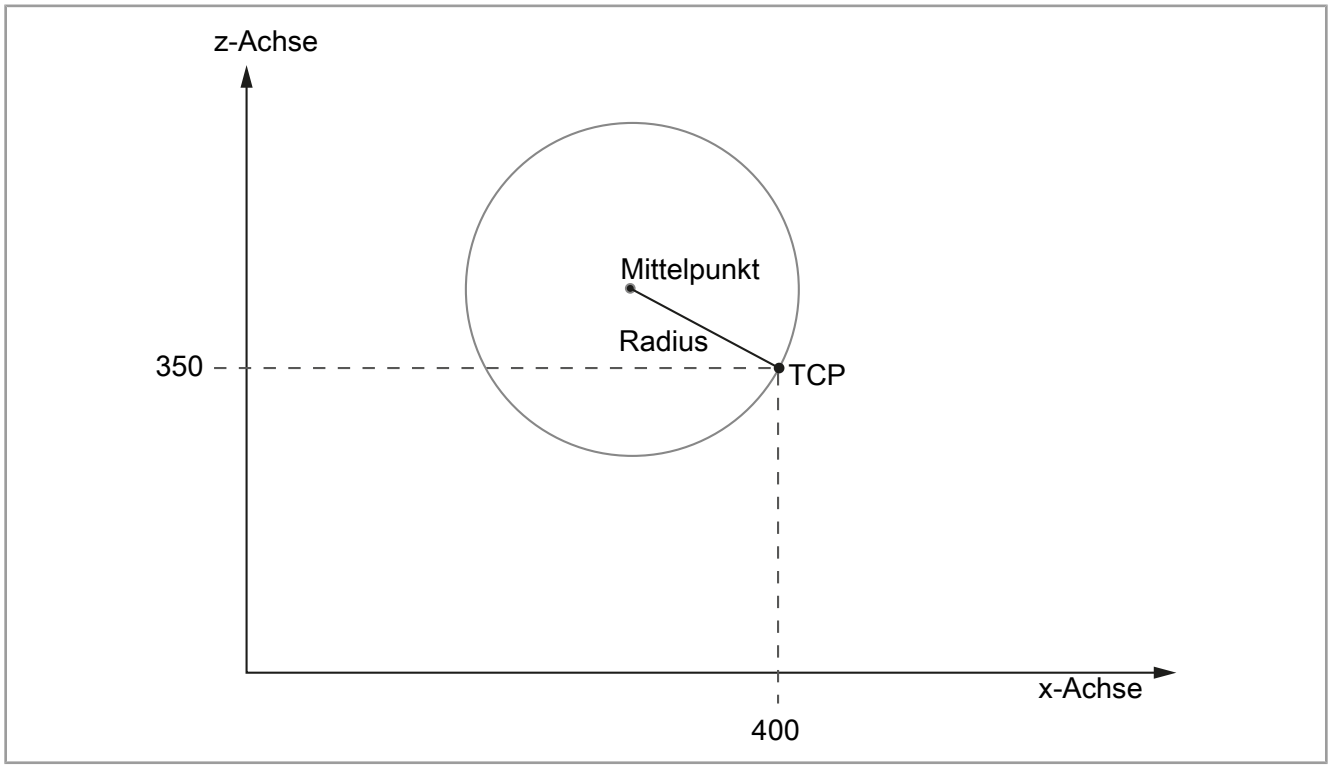


Abb. 11: Prinzip der Funktionalität MCGeo.MoveCircle2D

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden, insbesondere zur Arbeit mit Überschleifen, den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

57.1 Syntax für MCGeo.MoveCircle2D

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.MoveCircle2D.<Funktion oder Zugriff auf Wert>(optional
<Parameter 1, Parameter 2, Parameter n>);
```

57.2 Funktionen von MCGeo.MoveCircle2D

Die Funktionalität bietet folgende Funktionen:

– Start()

57.2.1 Start()

Die Funktion startet eine Kreisbewegung in einer Raumebene des TCP eines Bahnverbunds. Die Kreisbahn ist definiert durch:

– die Koordinaten des Zielpunkts der letzten Bewegung;
– die Mittelpunktkoordinaten;
– den zu überstreichenden Winkel.

Die Lage der Raumebene in orthogonaler Richtung (Startwert der Helixbewegung) wird durch den Startwert (Zielkoordinate der letzten Bewegung) bestimmt. Falls die Mittelpunktkoordinate in diese Richtung vom Koordinatenwert des Startwerts abweicht, wird sie ignoriert.

Da ein Bahnverbund im gepufferten Betrieb mit Vorlaufpuffer arbeitet, wird die erste Bewegung als Bahnstück sofort begonnen. Werden während der Bewegung weitere Fahrbefehle geschrieben, so werden diese Bahnstücke nacheinander ausgeführt, sobald das vorherige Bahnstück beendet wurde. Es ist möglich, Bahnstücke ohne Anhalten durch Überschleifen ineinander übergehen zu lassen.

Übergabeparameter

Beachten Sie folgende Regeln:

– Alle Parameter werden in den Einheiten angegeben, die vom Anwender für die Achse vordefiniert wurden.
– Die Parameter beziehen sich auf das Verbundobjekt; d. h. sie bilden eine Eigenschaft des Verbunds. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Parameter	Datentyp	Bedeutung
PositioningMode	MCGeoPositioningModes (optional)	Legt den Positioniermodus fest (siehe PositioningMode [▶ 324]).
BlockEnd	MCGeoPathBlockEnd (optional)	Abschlussbedingung eines Bahnstücks im Vorlaufpuffer. Sie definiert, wann das folgende Bahnstück im Vorlaufpuffer begonnen wird. (Satzwechselbedingung) (siehe BlockEnd [▶ 325]).
CoordinateSystemPlane	MCGeoCoordinateSystemPlanes (optional)	Koordinatenebene, in der das Kreissegment gefahren werden soll.
	MCGeoCoordinateSystemPlanes.XY	Das Kreissegment wird in der XY-Ebene gefahren.
	MCGeoCoordinateSystemPlanes.YZ	Das Kreissegment wird in der YZ-Ebene gefahren.
	MCGeoCoordinateSystemPlanes.ZX	Das Kreissegment wird in der ZX-Ebene gefahren.
TravelAngle	Double (optional)	Winkel, den das Kreissegment überstreichen soll in [Grad]. Der Winkel wird in der vorgegebenen Ebene in mathematisch positive Richtung überstrichen.
TargetPositionHelixAxis	Double (optional)	Zielposition des TCP auf der Koordinate, die orthogonal zur Kreisebene steht.
Speed	Double (optional)	Definiert die Bahngeschwindigkeit, mit der der TCP bewegt werden soll, solange sich die Bewegung nicht in den Rampen (Beschleunigung, Verzögerung) befindet. Der Wert muss größer Null sein.

Parameter	Datentyp	Bedeutung
SmoothingDistance	Double (optional)	Überschleifabstand: Der Wert muss immer kleiner sein als die Hälfte des kürzeren der beiden Bahnstücke, die mithilfe des Überschleifens ineinander übergehen sollen. Der Wert muss größer Null sein, selbst wenn die `BlockEnd`-Bedingung so gewählt ist, dass kein Überschleifen stattfindet.
Delay	Integer (optional)	Definiert die Wartezeit bis zum Start der nächsten im Puffer gespeicherten Bewegung in [ms]. Die Wartezeit wird nur wirksam, wenn der Positioniermodus `InPosState_DestWnd_Delay` oder `InPosState_DestWnd_Delay_SyncOri` ausgewählt wurde.
TargetWindow	Double (optional)	Definiert die Breite des Zielfensters bezogen auf den Bahnweg für die Bewegung.
CenterPointX CenterPointY CenterPointZ	Double (optional)	Drei Koordinatenparameter für den Kreismittelpunt des Kreissegments, das abgefahren werden soll.
TargetOrientationU TargetOrientationV TargetOrientationW	Double (optional)	Drei Koordinatenparameter für die Zielorientierung des Werkzeugs, die nach Ende des Bahnstücks erreicht werden soll.

Syntax

```
geoGroup.MoveCircle2D.Start(MCGeoPositioningModes.Absolute,MCGeoPathBlo-
ckEnd.InPosState, MCGeoCoordinateSytemPla-
nes.XY,45.0,10.0,300.0,1.0,0,0.1,100.0,50.0,150.0,0.0,0.0,0.0);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√	Läuft	Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	√		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Bahnbewegung stehen Ihnen verschiedene Informationen zur Verfügung:

– Der **Zustand des Verbunds** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder der Verbund angehalten wurde (MCGeo.State [▶ 389]).

```
When geoGroup.State.IsEnabled Continue; //der Verbund befindet sich in
Lageregelung und steht.
When geoGroup.State.IsRunning Continue; //der Sollwertgenerator erzeugt eine
Bewegung.
```

– Der **Rampenstatus** informiert darüber, in welchem Bewegungsabschnitt sich der Sollwertgenerator aktuell befindet (MCGeo.Mechanism.Slope [▶ 308]).

```
When geoGroup.Mechanism.Slope.IsAccelerating Continue;
When geoGroup.Mechanism.Slope.IsAtConstantSpeed Continue;
When geoGroup.Mechanism.Slope.IsDecelerating Continue;
When geoGroup.Mechanism.Slope.IsStopped Continue;
```

57.3 Eigenschaften von MCGeo.MoveCircle2D

Die Funktionalität bietet folgende Eigenschaften:

– PositioningMode
– BlockEnd
– CoordinateSystemPlane
– TravelAngle
– TargetPositionHelixAxis
– SmoothingDistance
– Delay
– TargetWindow
– CenterPoint
– TargetOrientation

57.3.1 PositioningMode

Beschreibung

Mit der Eigenschaft können Sie den momentan gültigen Positioniermodus eines Bahnverbundes lesen oder einen neuen Wert für diesen Verbund setzen.

ⓘ Info

Ermittlung der Zielposition nach Enable()-Befehl

Nach dem Freischalten (`Enable()`) wird die letzte Zielposition aus der Istposition zum Freischaltzeitpunkt berechnet.

Datentyp	Bedeutung
MCGeoPositioningModes (optional)	Modus für die Art der Bahnbewegung.
MCGeoPositioningModes.Absolut	Bewegt den TCP auf den Positionspunkt, der aus den Parameter für Verfahrwinkel, Mittelpunktkoordinaten und letzten Zielpositionskoordiaten berechneten wurde. Dieser wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Bahnbewegung ergibt sich aus dem Vorzeichen des Parameters für den Verfahrwinkel und der Helixzielposition in Relation zur letzten Zielpositionskoordinate in diese Richtung. Die Orientierungsachsen werden so gesteuert, dass die neuen Orientierungswinkel auf kürzestem Weg angefahren werden. **Beispiel**: Bei einer letzten Zielorientierungsposition von 40° und einer neuen Zielorientierungsposition von 350° bewegt sich die Orientierungsachse in negative Richtung um 50°.
MCGeoPositioningModes.Relative	Gibt sowohl die Mittelpunktkoordinaten relativ zum Startpunkt wie auch den Helixweg relativ zur Ebene im Startpunkt an. Die Fahrtrichtung der Bahnbewegung ergibt sich aus dem Vorzeichen des Parameters für den Verfahrwinkel und der Helixzielposition.
MCGeoPositioningModes.AbsOriNormal	Bewegt die Raumkoordinaten des TCP gleich wie im Modus `Absolut`. Steuert die Orientierungsachsen so, dass die neuen Orientierungswinkel entsprechend der Differenz zwischen letztem Sollwinkel und dem Zielwinkel angefahren werden. **Beispiel**: Bei einer letzten Zielorientierungsposition von 40° und einer neuen Zielorientierungsposition von 350° bewegt sich die Orientierungsachse in positive Richtung um 310°.

Syntax

```
Geo_Positioniermodus := geoGroup.MoveCircle2D.PositioningMode;
geoGroup.MoveCircle2D.PositioningMode := MCGeoPositioningModes.Relative;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

57.3.2 BlockEnd

Beschreibung

Mit der Eigenschaft können Sie die momentan gültige Abschlussbedingung für ein Bahnstück im Vorlaufpuffer des Bahnverbundes lesen, oder einen neuen Wert für diesen Verbund setzen.

Datentyp	Bedeutung
MCGeoPathBlockEnd (optional)	Abschlussbedingung eines Bahnstücks im Vorlaufpuffer. Sie definiert, wann das folgende Bahnstück im Vorlaufpuffer begonnen wird (Satzwechselbedingung).
MCGeoPathBlockEnd.InPosState	Das nächste Bahnstück wird begonnen, wenn das hier definierte Bahnstück die Sollgeschwindigkeit `0` erreicht hat.

Datentyp	Bedeutung
MCGeoPathBlockEnd.InPos-State_DestWnd	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Bahn die Zielposition bis auf den Zielfensterabstand erreicht hat.
MCGeoPathBlockEnd.InPos-State_DestWnd_Delay	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Bahn die Zielposition erreicht hat und die Verweilzeit abgelaufen ist.
MCGeoPathBlockEnd.InPos-State_SyncOri	Das nächste Bahnstück wird begonnen, wenn das hier definierte Bahnstück die Sollgeschwindigkeit `0` erreicht hat und die Sollwerte der Orientierungsachsen die Zielwerte erreicht haben.
MCGeoPathBlockEnd.InPos-State_DestWnd_SyncOri	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Bahn die Zielposition bis auf den Zielfensterabstand erreicht hat und die Sollwerte der Orientierungsachsen die Zielwerte erreicht haben.
MCGeoPathBlockEnd.InPos-State_DestWnd_Delay_SyncOri	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Bahn die Zielposition erreicht hat und die Verweilzeit abgelaufen ist. Außerdem müssen die Sollwerte der Orientierungsachsen die Zielwerte erreicht haben.
MCGeoPathBlockEnd.Smoothing	Der nächste Satzwechsel wird ohne Anhalten durchgeführt. Abhängig vom Parameter `SmoothingDistance` wird ein Polynom 5. Grades berechnet, das als Übergangssatz in den Verfahrpuffer eingefügt wird. Diese Satzwechselbedingung darf nicht beim letzten Bahnstück im Vorlaufpuffer verwendet werden.

Syntax

```
Geo_Satzuebergang := geoGroup.MoveCircle2D.BlockEnd;
geoGroup.MoveCircle2D.BlockEnd := MCGeoPathBlockEnd.InPosState_DestWnd;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

57.3.3 CoordinateSystemPlane

Beschreibung

Mit der Eigenschaft können Sie den momentan gültigen Wert für die Koordinatensystemebene des 2D-Kreis-Bahnstücks im Vorlaufpuffer oder des zuletzt gefahrenen 2D-Kreis-Bahnstücks des Bahnverbundes lesen. Mit dem Schreiben der Eigenschaft setzten Sie einen neuen Wert für diesen Verbund. Dieser wird beim nächsten `MoveCircle2D`-Befehl wirksam, falls dort kein anderer Wert geschrieben wird.

Datentyp	Bedeutung
Typ MCGeoCoordinateSystemPlanes (optional)	Koordinatenebene, in der das Kreissegment gefahren werden soll.
MCGeoCoordinateSystemPlanes.XY	Das Kreissegment wird in der XY-Ebene gefahren.
MCGeoCoordinateSystemPlanes.YZ	Das Kreissegment wird in der YZ-Ebene gefahren.
MCGeoCoordinateSystemPlanes.ZX	Das Kreissegment wird in der ZX-Ebene gefahren.

Syntax

```
Segmentebene := geoGroup.MoveCircle2D.CoordinateSystemPlane;
geoGroup.MoveCircle2D.CoordinateSystemPlane := MCGeoCoordinateSystemPlanes.ZX;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

57.3.4 TravelAngle

Beschreibung

Mit der Eigenschaft können Sie den momentan gültigen Wert für den Kreissegmentwinkel des Bahnverbundes lesen oder einen neuen Wert für den Bahnverbund vorgeben.

Datentyp	Bedeutung
Double (optional)	Wert für den Winkel des Kreissegments, das mit `MoveCircle2D` überstrichen wird.

Syntax

```
Segmentwinkel := geoGroup.MoveCircle2D.TravelAngle;
geoGroup.MoveCircle2D.TraveöAmgöe:= 45.2;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

57.3.5 TargetPositionHelixAxis

Beschreibung

Mit der Eigenschaft können Sie den momentan gültigen Wert für die Zielposition der Helixkoordinate des Bahnverbundes lesen oder einen neuen Wert für den Bahnverbund vorgeben.

Datentyp	Bedeutung
Double (optional)	Wert für die Zielkoordinate des TCP in Richtung der Orthogonalen auf die Koordinatensystemebene des Kreissegments.

Syntax

```
Helixposition := geoGroup.MoveCircle2D.TargetPositionHelixAxis;
geoGroup.MoveCircle2D.TargetPositionHelixAxis:= -250.0;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

57.3.6 SmoothingDistance

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für den Überschleifabstand des Bahnverbundes beim Übergang von einem Bahnstück zum nächsten, oder gibt einen neuen Wert für den Bahnverbund vor. Der Wert darf nicht `0` sein, auch wenn die `BlockEnd`-Bedingung nicht `smoothing` ist.

Syntax

```
Ueberschleifabstand := geoGroup.MoveCircle2D.SmoothingDistance;
geoGroup.MoveCircle2D.SmoothingDistance := 25.6;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

57.3.7 Delay

Beschreibung

Datentyp	Bedeutung
Integer (optional)	Liest den momentan gültigen Wert für die Wartezeit zwischen zwei Bahnstücken im Vorlaufpuffer des Bahnverbundes oder gibt einen neuen Wert für den Bahnverbund vor in [ms]. Der Wert wird nur wirksam, wenn als `BlockEnd`-Bedingung `InPosState_DestWnd_Delay` oder `InPosState_DestWnd_Delay_SyncOri` verwendet wird.

Syntax

```
Wartezeit := geoGroup.MoveCirle2D.Delay;
geoGroup.MoveCircle2D.Delay:= 150;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

57.3.8 TargetWindow

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für die Zielfenstereinstellung des Bahnverbundes, oder schreibt einen neuen Wert für diesen Verbund. Dies ist der Bahnabstand vom Zielpunkt, ab dem das neue Bahnstück begonnen wird. Der Wert wird nur wirksam, wenn als `BlockEnd`-Bedingung `InPosState_DestWnd`, `InPosState_DestWnd_Delay`, `InPosState_DestWnd_SyncOri` oder `InPosState_DestWnd_Delay_SyncOri` verwendet wird.

Syntax

```
Zielfenster := geoGroup. MoveCircle2D.TargetWindow;
geoGroup.MoveCircle2D.TargetWindow := 0.7;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

57.3.9 CenterPoint

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte des Kreismittelpunkts der aktuellen Bahnbewegung des Bahnverbunds oder geben neuen Mittelpunkt für den Verbund vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) übergeben. Es werden immer nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
CenterPoint	Position3D (optional)	Kreismittelpunt in 3D-Richtungskoordinaten.
CenterPoint.X	Position3D.X (Double)	X-Koordinate des Kreismittelpunkts.
CenterPoint.Y	Position3D.Y (Double)	Y-Koordinate des Kreismittelpunkts.
CenterPoint.Z	Position3D.Z (Double)	Z-Koordinate des Kreismittelpunkts.

Syntax

```
Mittelpunkt := geoGroup.MoveCircle2D.CenterPoint;
MittelpunktX := geoGroup.MoveCircle2D.CenterPoint.X;
geoGroup.MoveCircle2D.CenterPoint := Mittelpunkt;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

57.3.10 TargetOrientation

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte des Ziels der aktuellen Bahnbewegung des Bahnverbunds oder geben diese vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

TargetOrientation	Datentyp	Bedeutung
TargetOrientation	Orientation3D (optional)	Ziel der Orientierung des TCP in 3D-Orientierungskoordinaten.
TargetOrientation.U	Orientation3D.U (Double)	Zielwert der Orientierung des TCP in U-Richtung.
TargetOrientation.V	Orientation3D.V (Double)	Zielwert der Orientierung des TCP in V-Richtung.
TargetOrientation.W	Orientation3D.W (Double)	Zielwert der Orientierung des TCP in W-Richtung.

Syntax

```
ZielwertOri3D := geoGroup.MoveCircle2D.TargetOrientation;
ZielwertU := geoGroup.MoveCircle2D.TargetOrientation.U;
geoGroup.MoveCircle2D.TargetOrientation := NeueOrientierung;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

58 MCGeo.MoveCircle3D

Mit der Funktionalität **MCGeo.MoveCircle3D** können Sie eine beliebig im Raum liegende Kreisbewegung des Tool Center Points (TCP) eines MCX-Bahnverbundes starten und parametrieren.

Definiert wird die Kreisbewegung über drei Raumpunkte, die das Kreissegment eindeutig definieren:

– Ausgangspunkt (letzter Zielpunkt),
– Zielpunkt (Endpunkt der Kreissegmentbewegung),
– Ein Zwischenpunkt auf dem Kreissegment.

Im Fall einer 2D-Kinematik werden die Z-Koordinaten der drei Punkte ignoriert und das Kreissegment wird in der XY-Ebene gefahren.

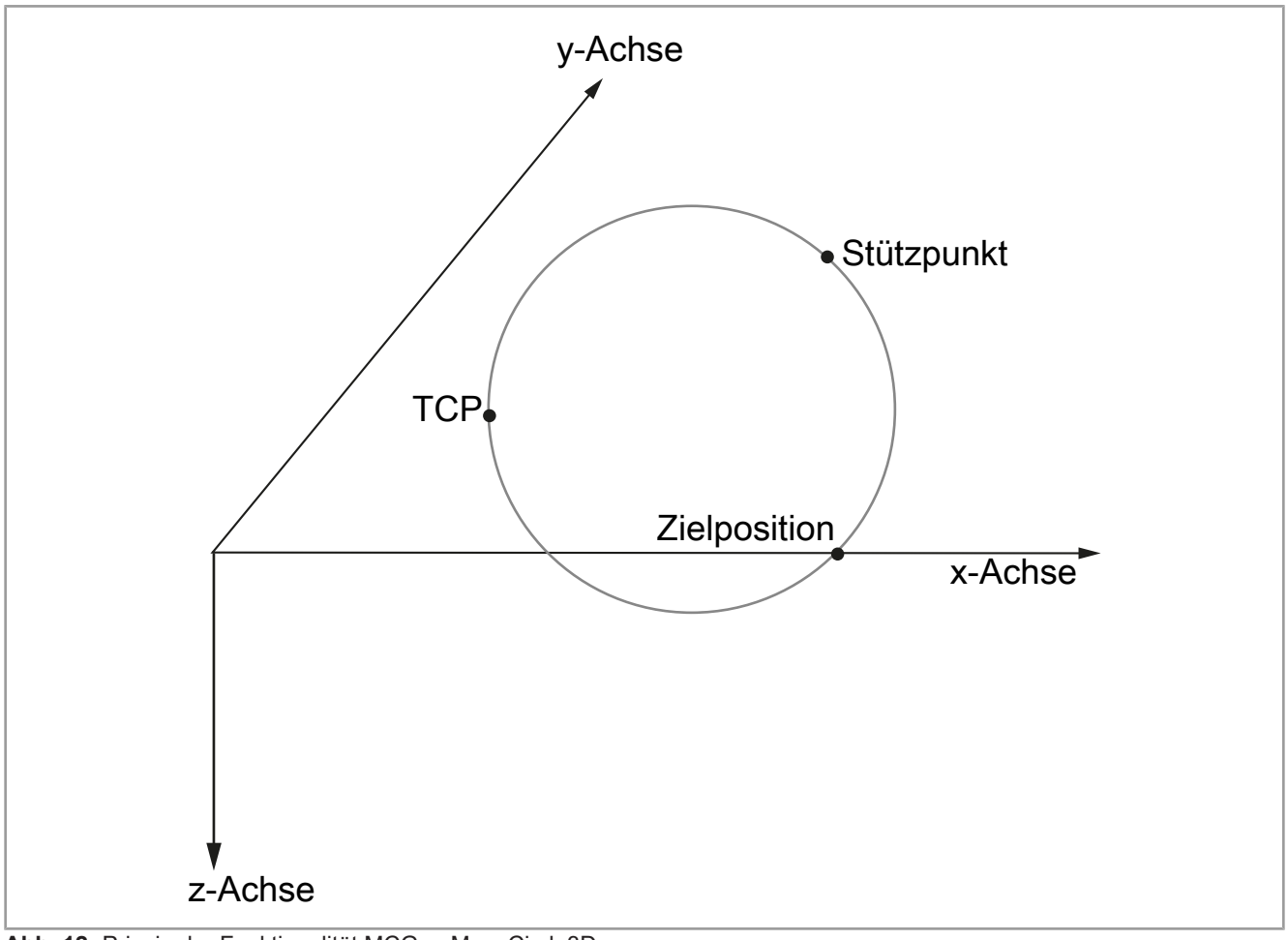


Abb. 12: Prinzip der Funktionalität MCGeo.MoveCircle3D

Eine Positionierbewegung besteht aus folgenden Phasen:

1. Beschleunigungsrampe
2. Gegenbenenfalls: Fahrt mit konstanter Geschwindigkeit;

   Die Beschleunigungsrampe kann auch direkt in die Verzögerungsrampe übergehen (Dreiecksbetrieb).
3. Verzögerungsrampe

Daneben kann ein Verschleifpolynom gefahren werden, wenn zwei Bahnstücke ohne Anhalten ineinander übergehen sollen.

Bei Bahnverbunden ergibt sich der Verlauf der Bahngeschwindigkeit durch eine trapezförmige Ausführung der Beschleunigung. Die Steilheit der Rampe im Trapez wird durch die Vorgabe des maximalen Bahnrucks bestimmt. Außerdem sind die Beschleunigung und Verzögerung im Bahnstück gleich. Beide werden über die Eigenschaft `Acceleration` gemeinsam geändert.

Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden, insbesondere zur Arbeit mit Überschleifen, den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

58.1 Syntax für MCGeo.MoveCircle3D

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.MoveCircle3D.<Funktion oder Zugriff auf Wert>(optional
<Parameter 1, Parameter 2, Parameter n>);
```

58.2 Funktionen von MCGeo.MoveCircle3D

Die Funktionalität bietet folgende Funktionen:

– Start()

58.2.1 Start()

Die Funktion startet eine Kreisbewegung des TCP eines Bahnverbunds im Raum. Die Kreisbahn ist definiert durch:

– die Koordinaten des Zielpunkts der letzten Bewegung;

– die Koordinaten eines Zwischenpunktes auf dem Kreissegment;

– die Koordinaten eines Zielpunkts.

Die Lage der Raumebene in orthogonaler Richtung (Startwert der Helixbewegung) wird durch den Startwert (Zielkoordinate der letzten Bewegung) bestimmt. Falls die Mittelpunktkoordinate in diese Richtung vom Koordinatenwert des Startwerts abweicht, wird sie ignoriert.

Da ein Bahnverbund im gepufferten Betrieb mit Vorlaufpuffer arbeitet, wird die erste Bewegung als Bahnstück sofort begonnen. Werden während der Bewegung weitere Fahrbefehle geschrieben, so werden diese Bahnstücke nacheinander ausgeführt, sobald das vorherige Bahnstück beendet wurde. Es ist möglich, Bahnstücke ohne Anhalten durch Überschleifen ineinander übergehen zu lassen.

Übergabeparameter

Beachten Sie folgende Regeln:

– Alle Parameter werden in den Einheiten angegeben, die vom Anwender für die Achse vordefiniert wurden.

– Die Parameter beziehen sich auf das Verbundobjekt; d. h. sie bilden eine Eigenschaft des Verbunds. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Parameter	Datentyp	Bedeutung
PositioningMode	MCGeoPositioningModes (optional)	Legt den Positioniermodus fest (siehe PositioningMode [▶ 333]).
BlockEnd	MCGeoPathBlockEnd (optional)	Abschlussbedingung eines Bahnstücks im Vorlaufpuffer. Sie definiert, wann das folgende Bahnstück im Vorlaufpuffer begonnen wird. (Satzwechselbedingung) (siehe BlockEnd [▶ 334]).

Parameter	Datentyp	Bedeutung
Speed	Double (optional)	Definiert die Bahngeschwindigkeit, mit der der TCP bewegt werden soll, solange sich die Bewegung nicht in den Rampen (Beschleunigung, Verzögerung) befindet. Der Wert muss größer `0` sein.
SmoothingDistance	Double (optional)	Überschleifabstand: Der Wert muss immer kleiner sein als die Hälfte des kürzeren der beiden Bahnstücke, die mithilfe des Überschleifens ineinander übergehen sollen. Der Wert muss größer `0` sein, selbst wenn die `BlockEnd`-Bedingung so gewählt ist, dass kein Überschleifen stattfindet.
Delay	Integer (optional)	Definiert die Wartezeit bis zum Start der nächsten im Puffer gespeicherten Bewegung in [ms]. Die Wartezeit wird nur wirksam, wenn der Positioniermodus `InPosState_DestWnd_Delay` oder `InPosState_DestWnd_Delay_SyncOri` ausgewählt wurde.
TargetWindow	Double (optional)	Definiert die Breite des Zielfensters bezogen auf den Bahnweg für die Bewegung.
IntermediatePointX IntermediatePointY IntermediatePointZ	Double (optional)	Drei Koordinatenparameter für den Zwischenpunkt auf dem Kreissegment, das abgefahren werden soll.
TargetPosiitionX TargetPosiitionY TargetPosiitionZ	Typ Double [optional]	Drei Koordinatenparameter für den Bahnzielpunkt des Kreissegments, das abgefahren werden soll.
TargetOrientationU TargetOrientationV TargetOrientationW	Double (optional)	Drei Koordinatenparameter für die Zielorientierung des Werkzeugs, die nach Ende des Bahnstücks erreicht werden soll.

Syntax

```
geoGroup.MoveCircle3D.Start(MCGeoPositioningModes.Absolute,MCGeoPathBlo-
ckEnd.InPosState,300.0,1.0,0,0.1,10.0,40.0,50.0,50.0,30.0,10.0,0.0,0.0,0.0);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√	Läuft	Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	√		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Bahnbewegung stehen Ihnen verschiedene Informationen zur Verfügung:

– Der **Zustand des Verbunds** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder der Verbund angehalten wurde (MCGeo.State [▶ 389]).

```
When geoGroup.State.IsEnabled Continue; //der Verbund befindet sich in
Lageregelung und steht.
When geoGroup.State.IsRunning Continue; //der Sollwertgenerator erzeugt eine
Bewegung.
```

– Der **Rampenstatus** informiert darüber, in welchem Bewegungsabschnitt sich der Sollwertgenerator aktuell befindet (MCGeo.Mechanism.Slope [▶ 308]).

```
When geoGroup.Mechanism.Slope.IsAccelerating Continue;
When geoGroup.Mechanism.Slope.IsAtConstantSpeed Continue;
When geoGroup.Mechanism.Slope.IsDecelerating Continue;
When geoGroup.Mechanism.Slope.IsStopped Continue;
```

58.3 Eigenschaften von MCGeo.MoveCircle3D

Die Funktionalität bietet folgende Eigenschaften:

– PositioningMode
– BlockEnd
– SmoothingDistance
– Delay
– TargetWindow
– IntermediatePoint
– TargetPosition
– TargetOrientation

58.3.1 PositioningMode

Beschreibung

Mit der Eigenschaft können Sie den momentan gültigen Positioniermodus eines Bahnverbundes lesen oder einen neuen Wert für diesen Verbund setzen.

ⓘ Info

Ermittlung der Zielposition nach Enable()-Befehl

Nach dem Freischalten (`Enable()`) wird die letzte Zielposition aus der Istposition zum Freischaltzeitpunkt berechnet.

Datentyp	Bedeutung
MCGeoPositioningModes (optional)	Modus für die Art der Bahnbewegung.
MCGeoPositioningModes.Absolut	Bewegt den TCP auf den Positionspunkt, der aus den Parametern für Verfahrwinkel, Mittelpunktkoordinaten und letzten Zielpositionskoordiaten berechneten wurde. Dieser wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Bahnbewegung ergibt sich aus dem Vorzeichen des Parameters für den Verfahrwinkel und der Helixzielposition in Relation zur letzten Zielpositionskoordinate in diese Richtung. Die Orientierungsachsen werden so gesteuert, dass die neuen Orientierungswinkel auf kürzestem Weg angefahren werden. **Beispiel**: Bei einer letzten Zielorientierungsposition von 40° und einer neuen Zielorientierungsposition von 350° bewegt sich die Orientierungsachse in negative Richtung um 50°.
MCGeoPositioningModes.Relative	Gibt sowohl die Mittelpunktkoordinaten relativ zum Startpunkt wie auch den Helixweg relativ zur Ebene im Startpunkt an. Die Fahrtrichtung der Bahnbewegung ergibt sich aus dem Vorzeichen des Parameters für den Verfahrwinkel und der Helixzielposition.
MCGeoPositioningModes.AbsOriNormal	Bewegt die Raumkoordinaten des TCP gleich wie im Modus `Absolut`. Steuert die Orientierungsachsen so, dass die neuen Orientierungswinkel entsprechend der Differenz zwischen letztem Sollwinkel und dem Zielwinkel angefahren werden. **Beispiel**: Bei einer letzten Zielorientierungsposition von 40° und einer neuen Zielorientierungsposition von 350° bewegt sich die Orientierungsachse in positive Richtung um 310°.

Syntax

```
Geo_Positioniermodus := geoGroup.MoveCircle3D.PositioningMode;
geoGroup.MoveCircle3D.PositioningMode := MCGeoPositioningModes.Relative;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

58.3.2 BlockEnd

Beschreibung

Mit der Eigenschaft können Sie die momentan gültige Abschlussbedingung für ein Bahnstück im Vorlaufpuffer des Bahnverbundes lesen, oder einen neuen Wert für diesen Verbund setzen.

Datentyp	Bedeutung
MCGeoPathBlockEnd (optional)	Abschlussbedingung eines Bahnstücks im Vorlaufpuffer. Sie definiert, wann das folgende Bahnstück im Vorlaufpuffer begonnen wird (Satzwechselbedingung).
MCGeoPathBlockEnd.InPosState	Das nächste Bahnstück wird begonnen, wenn das hier definierte Bahnstück die Sollgeschwindigkeit `0` erreicht hat.

Datentyp	Bedeutung
MCGeoPathBlockEnd.InPos-State_DestWnd	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Bahn die Zielposition bis auf den Zielfensterabstand erreicht hat.
MCGeoPathBlockEnd.InPos-State_DestWnd_Delay	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Bahn die Zielposition erreicht hat und die Verweilzeit abgelaufen ist.
MCGeoPathBlockEnd.InPos-State_SyncOri	Das nächste Bahnstück wird begonnen, wenn das hier definierte Bahnstück die Sollgeschwindigkeit `0` erreicht hat und die Sollwerte der Orientierungsachsen die Zielwerte erreicht haben.
MCGeoPathBlockEnd.InPos-State_DestWnd_SyncOri	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Bahn die Zielposition bis auf den Zielfensterabstand erreicht hat und die Sollwerte der Orientierungsachsen die Zielwerte erreicht haben.
MCGeoPathBlockEnd.InPos-State_DestWnd_Delay_SyncOri	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Bahn die Zielposition erreicht hat und die Verweilzeit abgelaufen ist. Außerdem müssen die Sollwerte der Orientierungsachsen die Zielwerte erreicht haben.
MCGeoPathBlockEnd.Smoothing	Der nächste Satzwechsel wird ohne Anhalten durchgeführt. Abhängig vom Parameter `SmoothingDistance` wird ein Polynom 5. Grades berechnet, das als Übergangssatz in den Verfahrpuffer eingefügt wird. Diese Satzwechselbedingung darf nicht beim letzten Bahnstück im Vorlaufpuffer verwendet werden.

Syntax

```
Geo_Satzuebergang := geoGroup.MoveCircle3D.BlockEnd;
geoGroup.MoveCircle3D.BlockEnd := MCGeoPathBlockEnd.InPosState_DestWnd;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

58.3.3 SmoothingDistance

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für den Überschleifabstand des Bahnverbundes beim Übergang von einem Bahnstück zum nächsten, oder gibt einen neuen Wert für den Bahnverbund vor. Der Wert darf nicht `0` sein, auch wenn die `BlockEnd`-Bedingung nicht `smoothing` ist.

Syntax

```
Ueberschleifabstand := geoGroup.MoveCircle3D.SmoothingDistance;
geoGroup.MoveCircle3D.SmoothingDistance := 25.6;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

58.3.4 Delay

Beschreibung

Datentyp	Bedeutung
Integer (optional)	Liest den momentan gültigen Wert für die Wartezeit zwischen zwei Bahnstücken im Vorlaufpuffer des Bahnverbundes oder gibt einen neuen Wert für den Bahnverbund vor in [ms]. Der Wert wird nur wirksam, wenn als `BlockEnd`-Bedingung `InPosState_DestWnd_Delay` oder `InPosState_DestWnd_Delay_SyncOri` verwendet wird.

Syntax

```
Wartezeit := geoGroup.MoveCirle3D.Delay;
geoGroup.MoveCircle3D.Delay:= 150;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

58.3.5 TargetWindow

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für die Zielfenstereinstellung des Bahnverbundes, oder schreibt einen neuen Wert für diesen Verbund. Dies ist der Bahnabstand vom Zielpunkt, ab dem das neue Bahnstück begonnen wird. Der Wert wird nur wirksam, wenn als `BlockEnd`-Bedingung `InPosState_DestWnd`, `InPosState_DestWnd_Delay`, `InPosState_DestWnd_SyncOri` oder `InPosState_DestWnd_Delay_SyncOri` verwendet wird.

Syntax

```
Zielfenster := geoGroup. MoveCircle3D.TargetWindow;
geoGroup.MoveCircle3D.TargetWindow := 0.7;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

58.3.6 IntermediatePoint

Beschreibung

Mit der Eigenschaft können Sie die momentanen 3D-Koordinatenpunkte des Zwischenpunkts der aktuellen Bahnbewegung des Bahnverbunds lesen oder einen neuen Zwischenpunkt für den Verbund vorgeben.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) übergeben. Es werden immer nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
IntermediatePoint	Position3D (optional)	Kreiszwischenpunt auf dem Kreissegment in 3D-Richtungskoordinaten.
IntermediatePoint.X	Position3D.X (Double)	X-Koordinate des Zwischenpunkts.

Eigenschaft	Datentyp	Bedeutung
IntermediatePoint.Y	Position3D.Y (Double)	Y-Koordinate des Zwischenpunkts.
IntermediatePoint.Z	Position3D.Z (Double)	Z-Koordinate des Zwischenpunkts.

Syntax

```
Zwischenpunkt := geoGroup.MoveCircle2D.IntermediatePoint;
ZwischenpunktX := geoGroup.MoveCircle2D.IntermediatePoint.X;
geoGroup.MoveCircle2D.IntermediatePoint := Zwischenpunkt;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

58.3.7 TargetPosition

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte des Ziels der aktuellen Bahnbewegung des Bahnverbunds oder geben einen neuen Zielpunkt für das Kreissegment vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) übergeben. Es werden immer nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
TargetPosition	Position3D (optional)	Ziel der Position des TCP in 3D-Richtungskoordinaten.
TargetPosition.X	Position3D.X (Double)	Zielwert der Position des TCP in X-Richtung.
TargetPosition.Y	Position3D.Y (Double)	Zielwert der Position des TCP in Y-Richtung.
TargetPosition.Z	Position3D.Z (Double)	Zielwert der Position des TCP in Z-Richtung.

Syntax

```
Zielwert3D := geoGroup.MoveCircle3D.TargetPosition;
ZielwertX := geoGroup.MoveCircle3D.TargetPosition.X;
geoGroup.MoveCircle3D.TargetPosition := NeuerZielpunkt;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

58.3.8 TargetOrientation

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenwerte des Ziels der aktuellen Bahnbewegung des Bahnverbunds oder geben diese vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

TargetOrientation	Datentyp	Bedeutung
TargetOrientation	Orientation3D (optional)	Ziel der Orientierung des TCP in 3D-Orientierungskoordinaten.
TargetOrientation.U	Orientation3D.U (Double)	Zielwert der Orientierung des TCP in U-Richtung.

TargetOrientation	Datentyp	Bedeutung
TargetOrientation.V	Orientation3D.V (Double)	Zielwert der Orientierung des TCP in V-Richtung.
TargetOrientation.W	Orientation3D.W (Double)	Zielwert der Orientierung des TCP in W-Richtung.

Syntax

```
ZielwertOri3D := geoGroup.MoveCircle3D.TargetOrientation;
ZielwertU := geoGroup.MoveCircle3D.TargetOrientation.U;
geoGroup.MoveCircle3D.TargetOrientation := NeueOrientierung;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

59 MCGeo.MoveLinear

Mit der Funktionalität **MCGeo.MoveCircle3D** können Sie eine lineare Bahnbewegung des Tool Center Points (TCP) eines MCX-Bahnverbundes starten und parametrieren.

Eine Positionierbewegung besteht aus folgenden Phasen:

1. Beschleunigungsrampe
2. Gegebenenfalls: Fahrt mit konstanter Geschwindigkeit;

   Die Beschleunigungsrampe kann auch direkt in die Verzögerungsrampe übergehen (Dreiecksbetrieb).
3. Verzögerungsrampe

Daneben kann ein Verschleifpolynom gefahren werden, wenn zwei Bahnstücke ohne Anhalten ineinander übergehen sollen.

Bei Bahnverbunden ergibt sich der Verlauf der Bahngeschwindigkeit durch eine trapezförmige Ausführung der Beschleunigung. Die Steilheit der Rampe im Trapez wird durch die Vorgabe des maximalen Bahnrucks bestimmt. Außerdem sind die Beschleunigung und Verzögerung im Bahnstück gleich. Beide werden über die Eigenschaft `Acceleration` gemeinsam geändert.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden, insbesondere zur Arbeit mit Überschleifen, den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

59.1 Syntax für MCGeo.MoveLinear

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.MoveLinear.<Funktion oder Zugriff auf Wert>(optional
<Parameter 1, Parameter 2, Parameter n>);
```

59.2 Funktionen von MCGeo.MoveLinear

Die Funktionalität bietet folgende Funktionen:

– Start()

59.2.1 Start()

Die Funktion startet eine Linearbewegung des TCP eines Bahnverbundes.

Da ein Bahnverbund im gepufferten Betrieb mit Vorlaufpuffer arbeitet, wird die erste Bewegung als Bahnstück sofort begonnen. Werden während der Bewegung weitere Fahrbefehle geschrieben, so werden diese Bahnstücke nacheinander ausgeführt, sobald das vorherige Bahnstück beendet wurde. Es ist möglich, Bahnstücke ohne Anhalten durch Überschleifen ineinander übergehen zu lassen.

Übergabeparameter

Beachten Sie folgende Regeln:

– Alle Parameter werden in den Einheiten angegeben, die vom Anwender für die Achse vordefiniert wurden.

– Die Parameter beziehen sich auf das Verbundobjekt; d. h. sie bilden eine Eigenschaft des Verbunds. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Parameter	Datentyp	Bedeutung
PositioningMode	MCGeoPositioningModes (optional)	Legt den Positioniermodus fest (siehe Positioning-Mode [▶ 341]).
BlockEnd	MCGeoPathBlockEnd (optional)	Abschlussbedingung eines Bahnstücks im Vorlaufpuffer. Sie definiert, wann das folgende Bahnstück im Vorlaufpuffer begonnen wird. (Satzwechselbedingung) (siehe BlockEnd [▶ 342]).
Speed	Double (optional)	Definiert die Bahngeschwindigkeit, mit der der TCP bewegt werden soll, solange sich die Bewegung nicht in den Rampen (Beschleunigung, Verzögerung) befindet. Der Wert muss größer `0` sein.
SmoothingDistance	Double (optional)	Überschleifabstand: Der Wert muss immer kleiner sein als die Hälfte des kürzeren der beiden Bahnstücke, die mithilfe des Überschleifens ineinander übergehen sollen. Der Wert muss größer `0` sein, selbst wenn die `BlockEnd`-Bedingung so gewählt ist, dass kein Überschleifen stattfindet.
Delay	Integer (optional)	Definiert die Wartezeit bis zum Start der nächsten im Puffer gespeicherten Bewegung in [ms]. Die Wartezeit wird nur wirksam, wenn der Positioniermodus `InPosState_DestWnd_Delay` oder `InPosState_DestWnd_Delay_SyncOri` ausgewählt wurde.
TargetWindow	Double (optional)	Definiert die Breite des Zielfensters bezogen auf den Bahnweg für die Bewegung.
TargetPosiitionX TargetPosiitionY TargetPosiitionZ	Typ Double [optional]	Drei Koordinatenparameter für die Zielposition im Raum, auf der TCP am Ende der Bahnbewegung stehen soll.
TargetOrientationU TargetOrientationV TargetOrientationW	Double (optional)	Drei Koordinatenparameter für die Zielorientierung des Werkzeugs, die nach Ende des Bahnstücks erreicht werden soll.

Syntax

```
geoGroup.MoveLinear.Start(MCGeoPositioningModes.Absolute,MCGeoPathBlo-
ckEnd.InPosState, 300.0,0.0,0,0.1,100.0,50.0,150.0,0.0,0.0,0.0);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√	Läuft	Freigegeben
Läuft	√	Läuft	Freigegeben

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Unterbrochen	√		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Bahnbewegung stehen Ihnen verschiedene Informationen zur Verfügung:

– Der **Zustand des Verbunds** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder der Verbund angehalten wurde (MCGeo.State [▶ 389]).

```
When geoGroup.State.IsEnabled Continue; //der Verbund befindet sich in
Lageregelung und steht.
When geoGroup.State.IsRunning Continue; //der Sollwertgenerator erzeugt eine
Bewegung.
```

– Der **Rampenstatus** informiert darüber, in welchem Bewegungsabschnitt sich der Sollwertgenerator aktuell befindet (MCGeo.Mechanism.Slope [▶ 308]).

```
When geoGroup.Mechanism.Slope.IsAccelerating Continue;
When geoGroup.Mechanism.Slope.IsAtConstantSpeed Continue;
When geoGroup.Mechanism.Slope.IsDecelerating Continue;
When geoGroup.Mechanism.Slope.IsStopped Continue;
```

59.3 Eigenschaften von MCGeo.MoveLinear

Die Funktionalität bietet folgende Eigenschaften:

– PositioningMode
– BlockEnd
– SmoothingDistance
– Delay
– TargetWindow
– TargetCoordinates
– TargetPosition
– TargetOrientation

59.3.1 PositioningMode

Beschreibung

Mit der Eigenschaft können Sie den momentan gültigen Positioniermodus eines Bahnverbundes lesen oder einen neuen Wert für diesen Verbund setzen.

ⓘ Info

Ermittlung der Zielposition nach Enable()-Befehl

Nach dem Freischalten (`Enable()`) wird die letzte Zielposition aus der Istposition zum Freischaltzeitpunkt berechnet.

Datentyp	Bedeutung
MCGeoPositioningModes (optional)	Modus für die Art der Bahnbewegung.
MCGeoPositioningModes.Absolut	Bewegt den TCP auf den Positionspunkt, der aus den Parametern für die `TargetPosition` berechneten wurde. Dieser wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Bahnbewegung ergibt sich aus der letzten und neuen Zielposition. Die Orientierungsachsen werden so gesteuert, dass die neuen Orientierungswinkel auf kürzestem Weg angefahren werden. **Beispiel**: Bei einer letzten Zielorientierungsposition von 40° und einer neuen Zielorientierungsposition von 350° bewegt sich die Orientierungsachse in negative Richtung um 50°.
MCGeoPositioningModes.Relative	Bewegt den TCP um den Weg weiter, der im Parameter `TargetPosition` übergeben wurde. Der Zielpunkt berechnet sich aus den Koordinaten des letzten Zielpunkts addiert mit dem Wert der Strecken in `TargetPosition`. Die Streckenkoordinaten können mit positivem und negativem Vorzeichen vorgegeben werden.
MCGeoPositioningModes.AbsOriNormal	Bewegt die Raumkoordinaten des TCP gleich wie im Modus `Absolut`. Steuert die Orientierungsachsen so, dass die neuen Orientierungswinkel entsprechend der Differenz zwischen letztem Sollwinkel und dem Zielwinkel angefahren werden. **Beispiel**: Bei einer letzten Zielorientierungsposition von 40° und einer neuen Zielorientierungsposition von 350° bewegt sich die Orientierungsachse in positive Richtung um 310°.

Syntax

```
Geo_Positioniermodus := geoGroup.MoveLinear.PositioningMode;
geoGroup.MoveLinear.PositioningMode := MCGeoPositioningModes.Relative;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

59.3.2 BlockEnd

Beschreibung

Mit der Eigenschaft können Sie die momentan gültige Abschlussbedingung für ein Bahnstück im Vorlaufpuffer des Bahnverbundes lesen, oder einen neuen Wert für diesen Verbund setzen.

Datentyp	Bedeutung
MCGeoPathBlockEnd (optional)	Abschlussbedingung eines Bahnstücks im Vorlaufpuffer. Sie definiert, wann das folgende Bahnstück im Vorlaufpuffer begonnen wird (Satzwechselbedingung).
MCGeoPathBlockEnd.InPosState	Das nächste Bahnstück wird begonnen, wenn das hier definierte Bahnstück die Sollgeschwindigkeit `0` erreicht hat.

Datentyp	Bedeutung
MCGeoPathBlockEnd.InPos-State_DestWnd	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Bahn die Zielposition bis auf den Zielfensterabstand erreicht hat.
MCGeoPathBlockEnd.InPos-State_DestWnd_Delay	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Bahn die Zielposition erreicht hat und die Verweilzeit abgelaufen ist.
MCGeoPathBlockEnd.InPos-State_SyncOri	Das nächste Bahnstück wird begonnen, wenn das hier definierte Bahnstück die Sollgeschwindigkeit `0` erreicht hat und die Sollwerte der Orientierungsachsen die Zielwerte erreicht haben.
MCGeoPathBlockEnd.InPos-State_DestWnd_SyncOri	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Bahn die Zielposition bis auf den Zielfensterabstand erreicht hat und die Sollwerte der Orientierungsachsen die Zielwerte erreicht haben.
MCGeoPathBlockEnd.InPos-State_DestWnd_Delay_SyncOri	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Bahn die Zielposition erreicht hat und die Verweilzeit abgelaufen ist. Außerdem müssen die Sollwerte der Orientierungsachsen die Zielwerte erreicht haben.
MCGeoPathBlockEnd.Smoothing	Der nächste Satzwechsel wird ohne Anhalten durchgeführt. Abhängig vom Parameter `SmoothingDistance` wird ein Polynom 5. Grades berechnet, das als Übergangssatz in den Verfahrpuffer eingefügt wird. Diese Satzwechselbedingung darf nicht beim letzten Bahnstück im Vorlaufpuffer verwendet werden.

Syntax

```
Geo_Satzuebergang := geoGroup.MoveLinear.BlockEnd;
geoGroup.MoveLinear.BlockEnd := MCGeoPathBlockEnd.InPosState_DestWnd;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

59.3.3 SmoothingDistance

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für den Überschleifabstand des Bahnverbundes beim Übergang von einem Bahnstück zum nächsten, oder gibt einen neuen Wert für den Bahnverbund vor. Der Wert darf nicht `0` sein, auch wenn die `BlockEnd`-Bedingung nicht `smoothing` ist.

Syntax

```
Ueberschleifabstand := geoGroup.MoveLinear.SmoothingDistance;
geoGroup.MoveLinear.SmoothingDistance := 25.6;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

59.3.4 Delay

Beschreibung

Datentyp	Bedeutung
Integer (optional)	Liest den momentan gültigen Wert für die Wartezeit zwischen zwei Bahnstücken im Vorlaufpuffer des Bahnverbundes oder gibt einen neuen Wert für den Bahnverbund vor in [ms]. Der Wert wird nur wirksam, wenn als `BlockEnd`-Bedingung `InPosState_DestWnd_Delay` oder `InPosState_DestWnd_Delay_SyncOri` verwendet wird.

Syntax

```
Wartezeit := geoGroup.MoveLinear.Delay;
geoGroup.MoveLinear.Delay:= 150;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

59.3.5 TargetWindow

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für die Zielfenstereinstellung des Bahnverbundes, oder schreibt einen neuen Wert für diesen Verbund. Dies ist der Bahnabstand vom Zielpunkt, ab dem das neue Bahnstück begonnen wird. Der Wert wird nur wirksam, wenn als `BlockEnd`-Bedingung `InPosState_DestWnd`, `InPosState_DestWnd_Delay`, `InPosState_DestWnd_SyncOri` oder `InPosState_DestWnd_Delay_SyncOri` verwendet wird.

Syntax

```
Zielfenster := geoGroup. MoveLinear.TargetWindow;
geoGroup.MoveLinear.TargetWindow := 0.7;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

59.3.6 TargetCoordinates

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 6D-Koordinatenpunkte des Ziels der aktuellen Bahnbewegung des Bahnverbunds oder geben einen neuen Zielpunkt für den Verbund vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) und 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

TargetOrientation	Datentyp	Bedeutung
TargetCoordinates	Coordinate6D (optional)	Ziel der Position des TCP in 6D-Raumkoordinaten.
TargetCoordinates.X	Coordinate6D.X (Double)	Zielwert der Position des TCP in X-Richtung.
TargetCoordinates.Y	Coordinate6D.Y (Double)	Zielwert der Position des TCP in Y-Richtung.

TargetOrientation	Datentyp	Bedeutung
TargetCoordinates.Z	Coordinate6D.Z (Double)	Zielwert der Position des TCP in Z-Richtung.
TargetCoordinates.U	Coordinate6D.U (Double)	Zielwert der Orientierung des TCP in U-Richtung.
TargetCoordinates.V	Coordinate6D.V (Double)	Zielwert der Orientierung des TCP in V-Richtung.
TargetCoordinates.W	Coordinate6D.W (Double)	Zielwert der Orientierung des TCP in W-Richtung.

Syntax

```
Zielwert6D := geoGroup.MoveLinear.TargetCoordinates;
ZielwertY := geoGroup.MoveLinear.TargetCoordinates.Y;
geoGroup.MoveLinear.TargetCoordinates := NeuerZielpunkt;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

59.3.7 TargetPosition

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte des Ziels der aktuellen Bahnbewegung des Bahnverbunds oder geben einen neuen Zielpunkt für den Bahnverbund vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) übergeben. Es werden immer nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
TargetPosition	Position3D (optional)	Ziel der Position des TCP in 3D-Richtungskoordinaten.
TargetPosition.X	Position3D.X (Double)	Zielwert der Position des TCP in X-Richtung.
TargetPosition.Y	Position3D.Y (Double)	Zielwert der Position des TCP in Y-Richtung.
TargetPosition.Z	Position3D.Z (Double)	Zielwert der Position des TCP in Z-Richtung.

Syntax

```
Zielwert3D := geoGroup.MoveLinear.TargetPosition;
ZielwertX := geoGroup.MoveLinear.TargetPosition.X;
geoGroup.MoveLinear.TargetPosition := NeuerZielpunkt;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

59.3.8 TargetOrientation

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenwerte des Ziels der aktuellen Bahnbewegung des Bahnverbunds oder geben diese vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

TargetOrientation	Datentyp	Bedeutung
TargetOrientation	Orientation3D (optional)	Ziel der Orientierung des TCP in 3D-Orientierungskoordinaten.

TargetOrientation	Datentyp	Bedeutung
TargetOrientation.U	Orientation3D.U (Double)	Zielwert der Orientierung des TCP in U-Richtung.
TargetOrientation.V	Orientation3D.V (Double)	Zielwert der Orientierung des TCP in V-Richtung.
TargetOrientation.W	Orientation3D.W (Double)	Zielwert der Orientierung des TCP in W-Richtung.

Syntax

```
ZielwertOri3D := geoGroup.MoveLinear.TargetOrientation;
ZielwertU := geoGroup.MoveLinear.TargetOrientation.U;
geoGroup.MoveLinear.TargetOrientation := NeueOrientierung;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

60 MCGeo.MoveOrientation

Mit der Funktionalität **MCGeo.MoveOrientation** können Sie eine Bewegung der Orientierungsachsen des Tool Center Points (TCP) eines MCX-Bahnverbundes starten und/oder parametrieren.

Im Gegensatz zu den anderen `MCGeo.Move`-Befehlen werden mit `MoveOrientation` nur die Orientierungsachsen bewegt, während die Raumkoordinatenachsen unverändert bleiben. Ein Überschleifen dieser Bewegung ist nicht möglich.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

60.1 Syntax für MCGeo.MoveOrientation

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.MoveOrientation.<Funktion oder Zugriff auf Wert>(optional <Parameter 1, Parameter 2, Parameter n>);
```

60.2 Funktionen von MCGeo.MoveOrientation

Die Funktionalität bietet folgende Funktionen:

– Start()

60.2.1 Start()

Die Funktion startet eine Orientierungsbewegung des TCP eines Bahnverbundes. Da ein Bahnverbund im gepufferten Betrieb mit Vorlaufpuffer arbeitet, wird die erste Bewegung als Bahnstück sofort begonnen. Werden während der Bewegung weitere Fahrbefehle geschrieben, so werden diese Bahnstücke nacheinander ausgeführt, sobald das vorherige Bahnstück beendet wurde. Es ist möglich, Bahnstücke ohne Anhalten durch Überschleifen ineinander übergehen zu lassen.

Übergabeparameter

Parameter	Datentyp	Bedeutung
PositioningMode	MCGeoPositioningModes (optional)	Legt den Positioniermodus fest (siehe PositioningMode [▶ 349]).
BlockEnd	MCGeoPathBlockEnd (optional)	Abschlussbedingung eines Bahnstücks im Vorlaufpuffer. Sie definiert, wann das folgende Bahnstück im Vorlaufpuffer begonnen wird (Satzwechselbedingung).
Speed	Double (optional)	Definiert die Geschwindigkeit der Bewegung in [°/s].
Acceleration	Double (optional)	Definiert die Beschleunigung der Bewegung in [°/s²].
Jerk	Double (optional)	Definiert den Ruck der Bewegung in [°/s³].

Parameter	Datentyp	Bedeutung
Delay	Integer (optional)	Definiert die Wartezeit bis zum Start der nächsten im Puffer gespeicherten Bewegung in [ms]. Die Wartezeit wird nur wirksam, wenn der Positioniermodus `InPosState_DestWnd_Delay` ausgewählt wurde.
TargetWindow	Double (optional)	Definiert die Breite des Zielfensters bezogen auf den Bahnweg in [°].
TargetOrientationU TargetOrientationV TargetOrientationW	Double (optional)	Drei Koordinatenparameter für die Zielorientierung des Werkzeugs, die nach Ende des Bahnstücks erreicht werden soll.

Syntax

```
geoGroup.MoveOrientation.Start(MCGeoPositioningModes.Absolute,MCGeoPathBlo-
ckEnd.InPosState,85.0,0.0,0.1,100.0,50.0,150.0,0.0,0.0,0.0);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√	Läuft	Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	√		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Bahnbewegung stehen Ihnen verschiedene Informationen zur Verfügung:

– Der **Zustand des Verbunds** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder der Verbund angehalten wurde (MCGeo.State [▶ 389]).

```
When geoGroup.State.IsEnabled Continue; //die Achse befindet sich in Lagere-
gelung und steht.
When geoGroup.State.IsRunning Continue; //der Sollwertgenerator erzeugt eine
Bewegung.
```

– Der **Rampenstatus** informiert darüber, in welchem Bewegungsabschnitt sich der Sollwertgenerator aktuell befindet (MCGeo.Mechanism.Slope [▶ 308]).

```
When geoGroup.Mechanism.Slope.IsAccelerating Continue;
When geoGroup.Mechanism.Slope.IsAtConstantSpeed Continue;
When geoGroup.Mechanism.Slope.IsDecelerating Continue;
When geoGroup.Mechanism.Slope.IsStopped Continue;
```

60.3 Eigenschaften von MCGeo.MoveOrientation

Die Funktionalität bietet folgende Eigenschaften:

- PositioningMode
- BlockEnd
- Speed
- Acceleration
- Jerk
- Delay
- TargetWindow
- TargetOrientation

60.3.1 PositioningMode

Beschreibung

Mit der Eigenschaft können Sie den momentan gültigen Positioniermodus eines Bahnverbundes lesen oder einen neuen Wert für diesen Verbund setzen.

ⓘ Info

Ermittlung der Zielposition nach Enable()-Befehl

Nach dem Freischalten (`Enable()`) wird die letzte Zielposition aus der Istposition zum Freischaltzeitpunkt berechnet.

Datentyp	Bedeutung
MCGeoPositioningModes (optional)	Modus für die Art der Bahnbewegung.
MCGeoPositioningModes.Absolut	Bewegt die Orientierung des TCP auf die im Parameter TargetOrientation übergebenen Werte. Diese wird in Relation zur Referenzposition angefahren. Die Orientierungsachsen werden so gesteuert, dass die neuen Orientierungswinkel auf kürzestem Weg angefahren werden. **Beispiel**: Bei einer letzten Zielorientierungsposition von 40° und einer neuen Zielorientierungsposition von 350° bewegt sich die Orientierungsachse in negative Richtung um 50°.
MCGeoPositioningModes.Relative	Bewegt die Orientierung des TCP um die Werte weiter, die im Parameter `TargetOrientation` übergeben wurde. Der Zielpunkt berechnet sich aus den Koordinaten des letzten Zielpunkts addiert mit dem Wert der Strecken in `TargetOrientation`. Die Werte können mit positivem und negativem Vorzeichen vorgegeben werden.

Datentyp	Bedeutung
MCGeoPositioningModes.AbsOriNormal	Bewegt die Orientierungsachsen so, dass die neuen Orientierungswinkel entsprechend der Differenz zwischen letztem Sollwinkel und dem Zielwinkel angefahren werden. **Beispiel**: Bei einer letzten Zielorientierungsposition von 40° und einer neuen Zielorientierungsposition von 350° bewegt sich die Orientierungsachse in positive Richtung um 310°.

Syntax

```
Geo_Positioniermodus := geoGroup.MoveOrientation.PositioningMode;
geoGroup.MoveOrientation.PositioningMode := MCGeoPositioningModes.Relative;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

60.3.2 BlockEnd

Beschreibung

Mit der Eigenschaft können Sie die momentan gültige Abschlussbedingung für ein Bahnstück im Vorlaufpuffer des Bahnverbundes lesen, oder einen neuen Wert für diesen Verbund setzen.

Datentyp	Bedeutung
MCGeoPathBlockEnd (optional)	Abschlussbedingung eines Bahnstücks im Vorlaufpuffer. Sie definiert, wann das folgende Bahnstück im Vorlaufpuffer begonnen wird (Satzwechselbedingung).
MCGeoPathBlockEnd.InPosState	Das nächste Bahnstück wird begonnen, wenn die Achse die Sollgeschwindigkeit null erreicht hat.
MCGeoPathBlockEnd.InPosState_DestWnd	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Achse mit dem längsten Verfahrweg die Zielposition bis auf den Zielfensterabstand erreicht hat.
MCGeoPathBlockEnd.InPosState_DestWnd_Delay	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Achse mit dem längsten Verfahrweg die Zielposition erreicht hat und die Verweilzeit abgelaufen ist.

Syntax

```
Geo_Satzuebergang := geoGroup.MoveOrientation.BlockEnd;
geoGroup.MoveOrientation.BlockEnd := MCGeoPathBlockEnd.InPosState_DestWnd;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

60.3.3 Speed

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für die Orientierungsgeschwindigkeit oder gibt einen neuen Wert vor in [°/s]. Dieser Parameter definiert die Orientierungsgeschwindigkeit, mit der der TCP bewegt werden soll, solange sich die Bewegung nicht in den Rampen (Beschleunigung, Verzögerung) befindet. Der Wert muss größer Null sein.

Syntax

```
OriGeschw := geoGroup.MoveOrientation.Speed;
geoGroup.MoveOrientation.Speed:= 15.4;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

60.3.4 Acceleration

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für die Orientierungsbeschleunigung oder gibt einen neuen Wert vor in [°/s²]. Dieser Parameter definiert die Orientierungsbeschleunigung, mit der der TCP bewegt werden soll, solange sich die Bewegung in den Rampen (Beschleunigung, Verzögerung) befindet. Der Wert muss größer Null sein.

Syntax

```
OriRampe := geoGroup.MoveOrientation.Acceleration;
geoGroup.MoveOrientation.Acceleration:= 120.1;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

60.3.5 Jerk

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für den Orientierungsruck oder gibt einen neuen Wert vor in [°/s³]. Dieser Parameter definiert den Orientierungsruck, mit dem der TCP bewegt werden soll, solange sich die Bewegung in den Rampen (Beschleunigung, Verzögerung) befindet. Der Ruck definiert die Steilheit der Rampen. Der Wert muss größer Null sein.

Syntax

```
OriRuck := geoGroup.MoveOrientation.Jerk;
geoGroup.MoveOrientation.Jerk:= 1514.4;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

60.3.6 Delay

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für die Wartezeit zwischen zwei Bahnstücken im Vorlaufpuffer des Bahnverbundes, oder gibt einen neuen Wert für den Bahnverbund vor in [ms]. Der Wert wird nur wirksam, wenn als `BlockEnd`-Bedingung `InPosState_DestWnd_Delay` verwendet wird.

Syntax

```
Wartezeit := geoGroup.MoveOrientation.Delay;
geoGroup.MoveOrientation.Delay:= 150;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

60.3.7 TargetWindow

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für den Sollwinkelabstand vom Zielpunkt der Orientierung, ab dem das neue Bahnstück begonnen wird, oder schreibt einen neuen Wert für diesen Verbund. Der Wert wird nur wirksam, wenn als `BlockEnd`-Bedingung `InPosState_DestWnd` oder `InPosState_DestWnd_Delay` verwendet wird.

Syntax

```
Zielfenster := geoGroup.MoveOrientation.TargetWindow;
geoGroup.MoveOrientation.TargetWindow := 0.7;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

60.3.8 TargetOrientation

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte des Ziels der aktuellen Bahnbewegung des Bahnverbunds oder geben diese vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

TargetOrientation	Datentyp	Bedeutung
TargetOrientation	Orientation3D (optional)	Ziel der Orientierung des TCP in 3D-Orientierungskoordinaten.
TargetOrientation.U	Orientation3D.U (Double)	Zielwert der Orientierung des TCP in U-Richtung.
TargetOrientation.V	Orientation3D.V (Double)	Zielwert der Orientierung des TCP in V-Richtung.
TargetOrientation.W	Orientation3D.W (Double)	Zielwert der Orientierung des TCP in W-Richtung.

Syntax

```
ZielwertOri3D := geoGroup.MoveOrientation.TargetOrientation;
ZielwertU := geoGroup.MovePto.TargetOrientation.U;
geoGroup.MoveOrientation.TargetOrientation := NeueOrientierung;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

61 MCGeo.MovePtp

Mit der Funktionalität **MCGeo.MovePtp** können Sie eine Punkt-zu-Punkt-Bewegung des Tool Center Points (TCP) im Raum eines MCX-Bahnverbundes starten und/oder parametrieren.

Im Gegensatz zu den anderen `MCGeo.Move`-Befehlen wird bei `MovePtp` keine vorgeschriebene Bahn beschrieben. Die Achsen des Bahnverbundes werden so in ihr Ziel gesteuert, dass alle Achsen gleichzeitig ankommen. Die Achse mit der längsten Verfahrzeit bestimmt die Verfahrgeschwindigkeit. Die anderen Achsen werden entsprechend ihres Verfahrweges langsamer bewegt. Je nach Aufbau der Kinematik und der Start- und der Zielposition im Raum, ergibt sich daraus die Bahnkurve. Das Überschleifen in eine Ptp-Bewegung und aus einer Ptp-Bewegung ist nicht möglich.

Die Bewegung mit MovePtp ist häufig die schnellste Bewegung des TCP im Raum und wird verwendet, wenn der Bahnverlauf für die Funktion keine Rolle spielt, und wenn sicher ist, dass der Bewegungsraum ohne Hindernisse ist.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden, insbesondere zur Arbeit mit Überschleifen, den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

61.1 Syntax für MCGeo.MovePtp

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.MovePtp.<Funktion oder Zugriff auf Wert>(optional <Parameter 1, Parameter 2, Parameter n>);
```

61.2 Funktionen von MCGeo.MovePtp

Die Funktionalität bietet folgende Funktionen:

– Start()

61.2.1 Start()

Die Funktion startet eine Orientierungsbewegung des TCP eines Bahnverbundes. Da ein Bahnverbund im gepufferten Betrieb mit Vorlaufpuffer arbeitet, wird die erste Bewegung als Bahnstück sofort begonnen. Werden während der Bewegung weitere Fahrbefehle geschrieben, so werden diese Bahnstücke nacheinander ausgeführt, sobald das vorherige Bahnstück beendet wurde. Es ist möglich, Bahnstücke ohne Anhalten durch Überschleifen ineinander übergehen zu lassen.

Übergabeparameter

Beachten Sie folgende Regeln:

– Alle Parameter werden in den Einheiten angegeben, die vom Anwender für den Verbund vordefiniert wurden.

– Die Parameter beziehen sich auf das Verbundobjekt; d. h. sie bilden eine Eigenschaft des Verbunds. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Parameter	Datentyp	Bedeutung
PositioningMode	MCGeoPositioningModes (optional)	Legt den Positioniermodus fest (siehe PositioningMode [▶ 356]).
BlockEnd	MCGeoMovePtpBlockEnd (optional)	Abschlussbedingung eines Bahnstücks im Vorlaufpuffer. Sie definiert, wann das folgende Bahnstück im Vorlaufpuffer begonnen wird (Satzwechselbedingung).
Speed	Double (optional)	Definiert die Geschwindigkeit der Bewegung. Sie wird angegeben in Prozent der Maximalgeschwindigkeit der Achse mit dem längsten Verfahrweg. Die anderen Achsen werden entsprechend der Länge ihres Verfahrweges langsamer bewegt.
Delay	Integer (optional)	Definiert die Wartezeit bis zum Start der nächsten im Puffer gespeicherten Bewegung in [ms]. Die Wartezeit wird nur wirksam, wenn der Positioniermodus `InPosState_DestWnd_Delay` ausgewählt wurde.
TargetWindow	Double (optional)	Definiert die Breite des Zielfensters für die Bewegung bezogen auf den Achse mit dem längsten Verfahrweg.
TargetPositionX TargetPositionY TargetPositionZ	Double (optional)	Drei Koordinatenparameter für die Zielposition im Raum, auf der der TCP stehen soll, nachdem die Bahnbewegung ausgeführt wurde.
TargetOrientationU TargetOrientationV TargetOrientationW	Double (optional)	Drei Koordinatenparameter für die Zielorientierung des Werkzeugs, die nach Ende des Bahnstücks erreicht werden soll.

Syntax

```
geoGroup.MovePtp.Start(MCGeoPositioningModes.Absolute,MCGeoPathBlockEnd.In-
PosState,85.0,0.0,0.1,100.0,50.0,150.0,0.0,0.0,0.0);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√	Läuft	Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	√		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Bahnbewegung stehen Ihnen verschiedene Informationen zur Verfügung:

– Der **Zustand des Verbunds** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder der Verbund angehalten wurde (MCGeo.State [▶ 389]).

```
When geoGroup.State.IsEnabled Continue; //der Verbund befindet sich in
Lageregelung und steht.
When geoGroup.State.IsRunning Continue; //der Sollwertgenerator erzeugt eine
Bewegung.
```

– Der **Rampenstatus** informiert darüber, in welchem Bewegungsabschnitt sich der Sollwertgenerator aktuell befindet (MCGeo.Mechanism.Slope [▶ 308]).

```
When geoGroup.Mechanism.Slope.IsAccelerating Continue;
When geoGroup.Mechanism.Slope.IsAtConstantSpeed Continue;
When geoGroup.Mechanism.Slope.IsDecelerating Continue;
When geoGroup.Mechanism.Slope.IsStopped Continue;
```

61.3 Eigenschaften von MCGeo.MovePtp

Die Funktionalität bietet folgende Eigenschaften:

– PositioningMode
– BlockEnd
– Delay
– TargetWindow
– TargetPosition
– TargetOrientation
– TargetCoordinates

61.3.1 PositioningMode

Beschreibung

Mit der Eigenschaft können Sie den momentan gültigen Positioniermodus eines Bahnverbundes lesen oder einen neuen Wert für diesen Verbund setzen.

ⓘ Info

Ermittlung der Zielposition nach Enable()-Befehl

Nach dem Freischalten (`Enable()`) wird die letzte Zielposition aus der Istposition zum Freischaltzeitpunkt berechnet.

Datentyp	Bedeutung
MCGeoPositioningModes (optional)	Modus für die Art der Bahnbewegung.
MCGeoPositioningModes.Absolut	Bewegt die Orientierung des TCP auf die im Parameter `TargetOrientation` übergebenen Werte. Diese wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Positionierung ergibt sich aus der letzten und der neuen Zielposition. Die Orientierungsachsen werden so gesteuert, dass die neuen Orientierungswinkel auf kürzestem Weg angefahren werden. **Beispiel**: Bei einer letzten Zielorientierungsposition von 40° und einer neuen Zielorientierungsposition von 350° bewegt sich die Orientierungsachse in negative Richtung um 50°.

Datentyp	Bedeutung
MCGeoPositioningModes.Relative	Bewegt die Orientierung des TCP um die Werte weiter, die im Parameter `TargetOrientation` übergeben wurde. Der Zielpunkt berechnet sich aus den Koordinaten des letzten Zielpunkts addiert mit dem Wert der Strecken in `TargetOrientation`. Die Werte können mit positivem und negativem Vorzeichen vorgegeben werden.
MCGeoPositioningModes.AbsOriNormal	Bewegt die Orientierungsachsen so, dass die neuen Orientierungswinkel entsprechend der Differenz zwischen letztem Sollwinkel und dem Zielwinkel angefahren werden. **Beispiel**: Bei einer letzten Zielorientierungsposition von 40° und einer neuen Zielorientierungsposition von 350° bewegt sich die Orientierungsachse in positive Richtung um 310°.

Syntax

```
Geo_Positioniermodus := geoGroup.MovePtp.PositioningMode;
geoGroup.MovePtp.PositioningMode := MCGeoPositioningModes.Relative;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

61.3.2 BlockEnd

Beschreibung

Mit der Eigenschaft können Sie die momentan gültige Abschlussbedingung für ein Bahnstück im Vorlaufpuffer des Bahnverbundes lesen, oder einen neuen Wert für diesen Verbund setzen.

Datentyp	Bedeutung
MCGeoMovePtpBlockEnd (optional)	Abschlussbedingung eines Bahnstücks im Vorlaufpuffer. Sie definiert, wann das folgende Bahnstück im Vorlaufpuffer begonnen wird (Satzwechselbedingung).
MCGeoMovePtpBlockEnd.InPosState	Das nächste Bahnstück wird begonnen, wenn die Achse die Sollgeschwindigkeit null erreicht hat.
MCGeoMovePtpBlockEnd.InPosState_DestWnd	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Achse mit dem längsten Verfahrweg die Zielposition bis auf den Zielfensterabstand erreicht hat.
MCGeoMovePtpBlockEnd.InPosState_DestWnd_Delay	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Achse mit dem längsten Verfahrweg die Zielposition erreicht hat und die Verweilzeit abgelaufen ist.

Syntax

```
Geo_Satzuebergang := geoGroup.MovePtp.BlockEnd;
geoGroup.MovePtp.BlockEnd := MCGeoMovePtpBlockEnd.InPosState_DestWnd;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

61.3.3 Delay

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für die Wartezeit zwischen zwei Bahnstücken im Vorlaufpuffer des Bahnverbundes, oder gibt einen neuen Wert für den Bahnverbund vor in [ms]. Der Wert wird nur wirksam, wenn als `BlockEnd`-Bedingung `InPosState_DestWnd_Delay` verwendet wird.

Syntax

```
Wartezeit := geoGroup.MovePtp.Delay;
geoGroup.MovePtp.Delay:= 150;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

61.3.4 TargetWindow

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für den Sollwinkelabstand vom Zielpunkt der Orientierung, ab dem das neue Bahnstück begonnen wird, oder schreibt einen neuen Wert für diesen Verbund. Der Wert wird nur wirksam, wenn als `BlockEnd`-Bedingung `InPosState_DestWnd` oder `InPosState_DestWnd_Delay` verwendet wird.

Syntax

```
Zielfenster := geoGroup.MovePtp.TargetWindow;
geoGroup.MovePtp.TargetWindow := 0.7;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

61.3.5 TargetPosition

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte des Ziels der aktuellen Bahnbewegung des Bahnverbunds oder geben einen neuen Zielpunkt für den Verbund vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) übergeben. Es werden immer nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
TargetPosition	Position3D (optional)	Ziel der Position des TCP mit 3D-Richtungskoordinaten.
TargetPosition.X	Position3D.X (Double)	Zielwert der Position des TCP in X-Richtung.
TargetPosition.Y	Position3D.Y (Double)	Zielwert der Position des TCP in Y-Richtung.
TargetPosition.Z	Position3D.Z (Double)	Zielwert der Position des TCP in Z-Richtung.

Syntax

```
Zielwert3D := geoGroup.MovePtp.TargetPosition;
ZielwertX := geoGroup.MovePtp.TargetPosition.X;
geoGroup.MovePtp.TargetPosition := NeuerZielpunkt;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

61.3.6 TargetOrientation

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte des Ziels der aktuellen Bahnbewegung des Bahnverbunds oder geben diese vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

TargetOrientation	Datentyp	Bedeutung
TargetOrientation	Orientation3D (optional)	Ziel der Orientierung des TCP in 3D-Orientierungskoordinaten.
TargetOrientation.U	Orientation3D.U (Double)	Zielwert der Orientierung des TCP in U-Richtung.
TargetOrientation.V	Orientation3D.V (Double)	Zielwert der Orientierung des TCP in V-Richtung.
TargetOrientation.W	Orientation3D.W (Double)	Zielwert der Orientierung des TCP in W-Richtung.

Syntax

```
ZielwertOri3D := geoGroup.MovePtp.TargetOrientation;
ZielwertU := geoGroup.MovePto.TargetOrientation.U;
geoGroup.MovePtp.TargetOrientation := NeueOrientierung;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

61.3.7 TargetCoordinates

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 6D-Koordinatenpunkte des Ziels der aktuellen Bahnbewegung des Bahnverbunds oder geben einen neuen Zielpunkt für den Verbund vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) und 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

TargetOrientation	Datentyp	Bedeutung
TargetCoordinates	Coordinate6D (optional)	Ziel der Position des TCP in 6D-Raumkoordinaten.
TargetCoordinates.X	Coordinate6D.X (Double)	Zielwert der Position des TCP in X-Richtung.
TargetCoordinates.Y	Coordinate6D.Y (Double)	Zielwert der Position des TCP in Y-Richtung.
TargetCoordinates.Z	Coordinate6D.Z (Double)	Zielwert der Position des TCP in Z-Richtung.

TargetOrientation	Datentyp	Bedeutung
TargetCoordinates.U	Coordinate6D.U (Double)	Zielwert der Orientierung des TCP in U-Richtung.
TargetCoordinates.V	Coordinate6D.V (Double)	Zielwert der Orientierung des TCP in V-Richtung.
TargetCoordinates.W	Coordinate6D.W (Double)	Zielwert der Orientierung des TCP in W-Richtung.

Syntax

```
Zielwert6D := geoGroup.MovePtp.TargetCoordinates;
ZielwertY := geoGroup.MovePtp.TargetCoordinates.Y;
geoGroup.MovePtp.TargetCoordinates := NeuerZielpunkt;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

62 MCGeo.MoveSpline

Mit der Funktionalität **MCGeo.MoveSpline** können eine spline-förmige Bahnbewegung des Tool Center Points (TCP) eines MCX-Bahnverbundes im Raum starten bzw. parametrieren. Bei dieser Art von Interpolation wird die Bahnkurve durch kubische Splines an bis zu 6 Stützpunkten approximiert. Kubische Splines sind Polynome dritten Grades, die bis zur zweiten Ableitung stetig sind. Der Startpunkt (letzter Zielpunkt des vorherigen Bahnbefehls) wird als erster Punkt mit in die Polynomberechnung einbezogen.

Weitere Erläuterungen finden Sie in der JetSym-Hilfe zum Thema Spline.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 50 – 52, 54 – 57** sowie die Dokumente *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

62.1 Syntax für MCGeo.MoveSpline

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.MoveSpline.<Funktion oder Zugriff auf Wert>(optional
<Parameter 1, Parameter 2, Parameter n>);
```

62.2 Funktionen von MCGeo.MoveCircle2D

Die Funktionalität bietet folgende Funktionen:

– Start()

62.2.1 Start()

Die Funktion startet eine Spline-Bewegung des TCP eines Bahnverbundes. Da ein Bahnverbund im gepufferten Betrieb mit Vorlaufpuffer arbeitet, wird die erste Bewegung als Bahnstück sofort begonnen. Werden während der Bewegung weitere Fahrbefehle geschrieben, so werden diese Bahnstücke nacheinander ausgeführt, sobald das vorherige Bahnstück beendet wurde. Es ist möglich, Bahnstücke ohne Anhalten durch Überschleifen ineinander übergehen zu lassen.

Übergabeparameter

Beachten Sie folgende Regeln:

– Alle Parameter werden in den Einheiten angegeben, die vom Anwender für den Verbund vordefiniert wurden.

– Die Parameter beziehen sich auf das Verbundobjekt; d. h. sie bilden eine Eigenschaft des Verbunds. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Parameter	Datentyp	Bedeutung
Nodes	Integer (optional)	Anzahl der Stützpunkte, die für die Spline-Bewegung verwendet werden sollen (mindestens **2,** höchstens 6).
PositioningMode	MCGeoPositioningModes (optional)	Definiert den Positioniermodus.

Parameter	Datentyp	Bedeutung
BlockEnd	MCGeoPathBlockEnd (optional)	Abschlussbedingung eines Bahnstücks im Vorlaufpuffer. Sie definiert, wann das folgende Bahnstück im Vorlaufpuffer begonnen wird (Satzwechselbedingung). Siehe BlockEnd [▶ 366]
StartCondition	Tyo MCGeoMoveSplineBlockTransitions (optional)	Definiert das Startverhalten der Spline-Bewegung (siehe StartCondition [▶ 366]).
TargetCondition	Tyo MCGeoMoveSplineBlockTransitions (optional)	Definiert das Zielverhalten der Spline-Bewegung (siehe TargetCondition [▶ 367]).
Speed	Double (optional)	Definiert die Bahngeschwindigkeit, mit der der TCP positioniert werden soll, solange die Bewegung sich nicht in den Rampen (Beschleunigung, Verzögerung) befindet. Der Wert muss größer Null sein.
SmoothingDistance	Double (optional)	Definiert den Überschleifabstand: Der Wert muss immer kleiner sein, als die Hälfte des kürzeren der beiden Bahnstücke, die mithilfe des Überschleifens ineinander übergehen sollen. Der Wert muss größer Null sein, selbst wenn die `BlockEnd`-Bedingung so gewählt ist, dass kein Überschleifen stattfindet.
Delay	**Integer** (optional)	Definiert die Wartezeit bis die folgende Bewegung im Puffer gestartet wird in [ms]. Die Wartezeit wird nur ausgeführt, wenn der Positioniermodus `InPosState_DestWnd_Delay` oder `InPosState_DestWnd_Delay_SyncOri` ausgewählt wurde.
TargetWindow	Double (optional)	Definiert die Breite des Zielfensters bezogen auf den Bahnweg für die Bewegung.
VirtualStartPositionX VirtualStartPositionY VirtualStartPositionZ	Double (optional)	Gibt eine gedachte Startposition vor, die für die Tangentenberechnung zum Startverhalten `ExplicitTangent` verwendet wird. **HINWEIS! Beachten Sie, dass in der Regel die virtuelle Startposition von der Bahn nicht erreicht wird. Es wird eine Tangente zur Bahn im ersten Stützpunkt gefahren.**
VirtualTargetPositionX VirtualTargetPositionY VirtualTargetPositionZ	Double (optional)	Gibt eine gedachte Zielposition vor, die für die Tangentenberechnung zum Zielverhalten `ExplicitTangent` verwendet wird. **HINWEIS! Beachten Sie, dass in der Regel die virtuelle Zielposition von der Bahn nicht erreicht wird. Es wird eine Tangente zur Bahn im letzten Stützpunkt gefahren.**
Node0PositionX Node0PositionY Node0PositionZ	Double (optional)	Definiert drei Koordinatenparameter für den ersten Stützpunkt im Raum, der zur Berechnung der Bahnbewegung herangezogen werden.

Parameter	Datentyp	Bedeutung
Node0OrientationU Node0OrientationV Node0OrientationW	Double (optional)	Definiert drei Koordinatenparameter für die Orientierung des Werkzeugs im ersten Stützpunkt, die zur Berechnung der Bahn herangezogen werden.
….		Koordinaten und Orientierungen für die Stützpunkte 1 … 4 analog zu `Node0` und `Node5`.
Node5PositionX Node5PositionY Node5PositionZ	Double (optional)	Definiert drei Koordinatenparameter für den sechsten Stützpunkt im Raum, der zur Berechnung der Bahnbewegung herangezogen werden.
Node5OrientationU Node5OrientationV Node5OrientationW	Double (optional)	Definiert drei Koordinatenparameter für die Orientierung des Werkzeugs im sechsten Stützpunkt, die zur Berechnung der Bahn herangezogen werden.

Syntax

```
geoGroup.MoveSpline.Start(5, //Anzahl der Knoten
MCGeoPositioningModes.Absolute, //PositioningMode
MCGeoPathBlockEnd.InPosState, //BlockEnd
MCGeoMoveSplineBlockTransitions.AutomaticMode, //StartCondition
MCGeoMoveSplineBlockTransitions.AutomaticMode, //TargetCondition
PathSpeed, //Speed
1.0, //Smoothing Distance
, // Verzögerung – nicht verwenden mit "inPosState"
0.5, //TargetWindow
,,, //VirtualStartPosition – nicht verwenden mit "AutomaticMode"
,,, //VirtualTargetPosition – nicht verwenden mit "AutomaticMode"
Node0X,Node0Y,Node0Z,Node0U,,, //Node 0 Position und Orientierung
Node1X,Node1Y,Node1Z,Node1U,,, //Node 1 Position und Orientierung
Node2X,Node2Y,Node2Z,Node2U,,, //Node 2 Position und Orientierung
Node3X,Node3Y,Node3Z,Node3U,,, //Node 3 Position und Orientierung
Node4X,Node4Y,Node4Z,Node4U); //Node 4 Position und Orientierung
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√	Läuft	Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	√		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Bahnbewegung stehen Ihnen verschiedene Informationen zur Verfügung:

– Der **Zustand des Verbunds** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder der Verbund angehalten wurde (MCGeo.State [▶ 389]).

```
When geoGroup.State.IsEnabled Continue; //der Verbund befindet sich in
Lageregelung und steht.
When geoGroup.State.IsRunning Continue; //der Sollwertgenerator erzeugt eine
Bewegung.
```

– Der **Rampenstatus** informiert darüber, in welchem Bewegungsabschnitt sich der Sollwertgenerator aktuell befindet (MCGeo.Mechanism.Slope [▶ 308]).

```
When geoGroup.Mechanism.Slope.IsAccelerating Continue;
When geoGroup.Mechanism.Slope.IsAtConstantSpeed Continue;
When geoGroup.Mechanism.Slope.IsDecelerating Continue;
When geoGroup.Mechanism.Slope.IsStopped Continue;
```

62.3 Eigenschaften von MCGeo.MoveSpline

Die Funktionalität bietet folgende Eigenschaften:

– NodeCount
– PositioningMode
– BlockEnd
– StartCondition
– TargetCondition
– SmoothingDistance
– Delay
– TargetWindow
– VirtualStartPosition
– VirtualTargetPosition
– NodePosition
– NodeOrientation
– NodeCoordinates

62.3.1 NodeCount

Beschreibung

Datentyp	Bedeutung
Integer	Liest den momentan gültigen Wert für die Anzahl Stützpunkte einer Spline-Bewegung des Bahnverbundes oder gibt einen neuen Wert für den Bahnverbund vor. Kleinster Wert = 2, größter Wert = 6.

Syntax

```
StuetzpunnteZahl := geoGroup.MoveSpline.NodeCount;
geoGroup.MoveSpline.NodeCount:= 6;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

62.3.2 PositioningMode

Beschreibung

Mit der Eigenschaft können Sie den momentan gültigen Positioniermodus eines Bahnverbundes lesen oder einen neuen Wert für diesen Verbund setzen.

> **ⓘ Info**
>
> **Ermittlung der Zielposition nach Enable()-Befehl**
>
> Nach dem Freischalten (`Enable()`) wird die letzte Zielposition aus der Istposition zum Freischaltzeitpunkt berechnet.

Datentyp	Bedeutung
MCGeoPositioningModes (optional)	Modus für die Art der Bahnbewegung.
MCGeoPositioningModes.Absolut	Bewegt den TCP auf den in den Parametern für die letzte `NodeTargetPosition` übergebenen Positionspunkt. Dieser wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Positionierung ergibt sich aus der letzten und der neuen Zielposition. Die Orientierungsachsen werden so gesteuert, dass die neuen Orientierungswinkel auf kürzestem Weg angefahren werden. Die Raumkoordinaten des TCP bewegen sich gleich wie im Modus `AbsOriNormal`. **Beispiel**: Bei einer letzten Zielorientierungsposition von 40° und einer neuen Zielorientierungsposition von 350° bewegt sich die Orientierungsachse in negative Richtung um 50°.
MCGeoPositioningModes.Relative	Bewegt den TCP um den Weg weiter, der in den Parametern für die `NodeTargetPosition` für die einzelnen Stützpunkte übergeben wurde. Der Zielpunkt berechnet sich aus den Koordinaten des letzten Zielpunkts addiert mit dem Wert der Strecken in `NodeTargetPosition`. Die Streckenkoordinaten können mit positivem und negativem Vorzeichen vorgegeben werden.
MCGeoPositioningModes.AbsOriNormal	Die Raumkoordinaten des TCP bewegen sich gleich wie im Modus `Absolut`. Bewegt die Orientierungsachsen so, dass die neuen Orientierungswinkel entsprechend der Differenz zwischen letztem Sollwinkel und dem Zielwinkel angefahren werden. **Beispiel**: Bei einer letzten Zielorientierungsposition von 40° und einer neuen Zielorientierungsposition von 350° bewegt sich die Orientierungsachse in positive Richtung um 310°.

Syntax

```
Geo_Positioniermodus := geoGroup.MoveSpline.PositioningMode;
geoGroup.MoveSpline.PositioningMode := MCGeoPositioningModes.Relative;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

62.3.3 BlockEnd

Beschreibung

Mit der Eigenschaft können Sie die momentan gültige Abschlussbedingung für ein Bahnstück im Vorlaufpuffer des Bahnverbundes lesen, oder einen neuen Wert für diesen Verbund setzen.

Datentyp	Bedeutung
MCGeoPathBlockEnd (optional)	Abschlussbedingung eines Bahnstücks im Vorlaufpuffer. Sie definiert, wann das folgende Bahnstück im Vorlaufpuffer begonnen wird (Satzwechselbedingung).
MCGeoPathBlockEnd.InPosState	Das nächste Bahnstück wird begonnen, wenn die Achse die Sollgeschwindigkeit null erreicht hat.
MCGeoPathBlockEnd.InPosState_DestWnd	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Achse mit dem längsten Verfahrweg die Zielposition bis auf den Zielfensterabstand erreicht hat.
MCGeoPathBlockEnd.InPosState_DestWnd_Delay	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Achse mit dem längsten Verfahrweg die Zielposition erreicht hat und die Verweilzeit abgelaufen ist.
MCGeoPathBlockEnd.InPosState_SyncOri	Das nächste Bahnstück wird begonnen, wenn das hier definierte Bahnstück die Sollgeschwindigkeit `0` erreicht hat und die Sollwerte der Orientierungsachsen die Zielwerte erreicht haben.
MCGeoPathBlockEnd.InPosState_DestWnd_SyncOri	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Bahn die Zielposition bis auf den Zielfensterabstand erreicht hat und die Sollwerte der Orientierungsachsen die Zielwerte erreicht haben.
MCGeoPathBlockEnd.InPosState_DestWnd_Delay_SyncOri	Das nächste Bahnstück wird begonnen, wenn die Sollposition der Bahn die Zielposition erreicht hat und die Verweilzeit abgelaufen ist. Außerdem müssen die Sollwerte der Orientierungsachsen die Zielwerte erreicht haben.
MCGeoPathBlockEnd.Smoothing	Der nächste Satzwechsel wird ohne Anhalten durchgeführt. Abhängig vom Parameter `SmoothingDistance` wird ein Polynom 5. Grades berechnet, das als Übergangssatz in den Verfahrpuffer eingefügt wird.

Syntax

```
Geo_Satzuebergang := geoGroup.MoveSpline.BlockEnd;
geoGroup.MoveSpline.BlockEnd := MCGeoPathBlockEnd.InPosState_DestWnd;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

62.3.4 StartCondition

Beschreibung

Mit der Eigenschaft können Sie das momentan gültige Startverhalten für die Bewegungssteuerung der Spline-Bewegung lesen oder einen neuen Wert vorgeben.

Datentyp	Bedeutung
MCGeoMoveSplineBlockTransitions (optional)	Startverhalten für die Spline-Bewegung.
MCGeoMoveSplineBlockTransitions.ExplicitTangent	Die Option *explizite Tangentenvorgabe* erfordert eine virtuelle Startposition. Diese Position wird zusammen mit dem ersten Stützpunkt verwendet, um die Tangentenrichtung zu berechnen, mit welcher der erste Stützpunkt angefahren wird. Beachten Sie, dass in der Regel die virtuelle Startposition von der Bahn nicht erreicht wird. Es wird eine Tangente zur Bahn im ersten Stützpunkt gefahren.
MCGeoMoveSplineBlockTransitions.AutomaticMode	Im *Automatik*-Modus wird der Übergang vom vorherigen Satz automatisch berechnet. Dabei wird normalerweise eine parabelförmige Bahn über die ersten drei Punkte verwendet. Wenn kein Vorsatz existiert wird der TCP mit dem im Parameter für die Bahnbeschleunigung gesetzten Wert beschleunigt.
MCGeoMoveSplineBlockTransitions.Tangential	Die Option *Tangential zu Vorsatz* bewirkt dasselbe Verhalten, wie `ExplicitTangent`; mit der Ausnahme, dass die virtuelle Startposition durch die letzte Zielposition des Vorsatzes definiert wird und im Vorsatz angefahren wird.
MCGeoMoveSplineBlockTransitions.NaturalMode	Im *natürlichen Modus* wird der Übergang vom vorherigen Satz in der Weise berechnet, dass während des Übergangs der TCP keine Beschleunigung erfährt. D. h., dass der Übergang mit konstanter Bahngeschwindigkeit stattfindet. Wenn kein Vorsatz existiert, wird der TCP mit dem im Parameter für die Bahnbeschleunigung gesetzten Wert beschleunigt.

Syntax

```
Geo_Startverahlten := geoGroup.MoveSpline.StartCondition;
geoGroup.MoveSpline.StartCondition := MCGeoMoveSplineBlockTransitions.Auto-
maticMode;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

62.3.5 TargetCondition

Beschreibung

Mit der Eigenschaft können Sie das momentan gültige Zielverhalten für die Bewegungssteuerung der Spline-Bewegung lesen oder einen neuen Wert vorgeben.

Datentyp	Bedeutung
MCGeoMoveSplineBlockTransitions (optional)	Zielverhalten für die Spline-Bewegung.
MCGeoMoveSplineBlockTransitions.ExplicitTangent	Die Option *explizite Tangentenvorgabe* erfordert eine virtuelle Startposition. Diese Position wird zusammen mit dem ersten Stützpunkt verwendet, um die Tangentenrichtung zu berechnen, mit welcher der erste Stützpunkt angefahren wird. Beachten Sie, dass in der Regel die virtuelle Startposition von der Bahn nicht erreicht wird. Es wird eine Tangente zur Bahn im ersten Stützpunkt gefahren.

Datentyp	Bedeutung
MCGeoMoveSplineBlockTransitions.AutomaticMode	Im *Automatik*-Modus wird der Übergang vom vorherigen Satz automatisch berechnet. Dabei wird normalerweise eine parabelförmige Bahn über die ersten drei Punkte verwendet. Wenn kein Vorsatz existiert wird der TCP mit dem im Parameter für die Bahnbeschleunigung gesetzten Wert beschleunigt.
MCGeoMoveSplineBlockTransitions.Tangential	Die Option *Tangential* zu Vorsatz bewirkt dasselbe Verhalten, wie `ExplicitTangent`; mit der Ausnahme, dass die virtuelle Startposition durch die letzte Zielposition des Vorsatzes definiert wird und im Vorsatz angefahren wird.
MCGeoMoveSplineBlockTransitions.NaturalMode	Im *natürlichen Modus* wird der Übergang vom vorherigen Satz in der Weise berechnet, dass während des Übergangs der TCP keine Beschleunigung erfährt. D. h., dass der Übergang mit konstanter Bahngeschwindigkeit stattfindet. Wenn kein Vorsatz existiert, wird der TCP mit dem im Parameter für die Bahnbeschleunigung gesetzten Wert beschleunigt.

Syntax

```
Geo_Zielverhalten := geoGroup.MoveSpline.TargetCondition;
geoGroup.MoveSpline.TargetCondition := MCGeoMoveSplineBlockTransitions.Tangential;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

62.3.6 SmoothingDistance

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für den Überschleifabstand des Bahnverbundes beim Übergang von einem Bahnstück zum nächsten, oder gibt einen neuen Wert für den Bahnverbund vor. Der Wert darf nicht `0` sein, auch wenn die `BlockEnd`-Bedingung nicht `smoothing` ist.

Syntax

```
Ueberschleifabstand := geoGroup.MoveSpline.SmoothingDistance;
geoGroup.MoveLinear.SmoothingDistance := 25.6;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

62.3.7 Delay

Beschreibung

Datentyp	Bedeutung
Integer (optional)	Liest den momentan gültigen Wert für die Wartezeit zwischen zwei Bahnstücken im Vorlaufpuffer des Bahnverbundes oder gibt einen neuen Wert für den Bahnverbund vor in [ms]. Der Wert wird nur wirksam, wenn als `BlockEnd`-Bedingung `InPosState_DestWnd_Delay` oder `InPosState_DestWnd_Delay_SyncOri` verwendet wird.

Syntax

```
Wartezeit := geoGroup.MoveSpline?.Delay;
geoGroup.MoveLinear.Delay:= 150;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

62.3.8 TargetWindow

Beschreibung

Datentyp	Bedeutung
Double (optional)	Liest den momentan gültigen Wert für die Zielfenstereinstellung des Bahnverbundes, oder schreibt einen neuen Wert für diesen Verbund. Dies ist der Bahnabstand vom Zielpunkt, ab dem das neue Bahnstück begonnen wird. Der Wert wird nur wirksam, wenn als `BlockEnd`-Bedingung `InPosState_DestWnd`, `InPosState_DestWnd_Delay`, `InPosState_DestWnd_SyncOri` oder `InPosState_DestWnd_Delay_SyncOri` verwendet wird.

Syntax

```
Zielfenster := geoGroup. MoveSpline.TargetWindow;
geoGroup.MoveLinear.TargetWindow := 0.7;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

62.3.9 VirtualStartPosition[x]

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte der virtuellen Startposition der aktuellen Spline-Bewegung des Bahnverbundes oder geben eine neue virtuelle Startposition für die Spline-Bewegung des Verbundes vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) übergeben. Es werden immer nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
VirtualStartPosition	Position3D (optional)	Position für die virtuelle Startposition des TCP in 3D-Richtungskoordinaten.

Eigenschaft	Datentyp	Bedeutung
VirtualStartPosition.X	Position3D.X (Double)	Position für die virtuelle Startposition des TCP in X-Richtung.
VirtualStartPosition.Y	Position3D.Y (Double)	Position für die virtuelle Startposition des TCP in Y-Richtung.
VirtualStartPosition.Z	Position3D.Z (Double)	Position für die virtuelle Startposition des TCP in Z-Richtung.

Syntax

```
VirtStartPosition_3D := geoGroup.MoveSpline.VirtualStartPosition;
VirtStartPositionX := geoGroup.MoveSpline.VirtualStartPosition.X;
Geo.MoveSpline. VirtualStartPosition:= VirtStartPos_3D;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

62.3.10 VirtualTargetPosition[x]

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte der virtuellen Zielposition der aktuellen Spline-Bewegung des Bahnverbundes oder geben eine neue virtuelle Zielposition für die Spline-Bewegung des Verbundes vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) übergeben. Es werden immer nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
VirtualTargetPosition	Position3D (optional)	Position für die virtuelle Zielposition des TCP in 3D-Richtungskoordinaten.
VirtualTargetPosition.X	Position3D.X (Double)	Position für die virtuelle Zielposition des TCP in X-Richtung.
VirtualTargetPosition.Y	Position3D.Y (Double)	Position für die virtuelle Zielposition des TCP in Y-Richtung.
VirtualTargetPosition.Z	Position3D.Z (Double)	Position für die virtuelle Zielposition des TCP in Z-Richtung.

Syntax

```
VirtTargetPosition_3D := geoGroup.MoveSpline.VirtualTargetPosition;
VirtTargetPositionX := geoGroup.MoveSpline.VirtualTargetPosition.X;
Geo.MoveSpline. VirtualTargetPosition:= VirtTargetPos_3D;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

62.3.11 NodePosition[n]

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte der Stützpunkte der aktuellen Spline-Bewegung des Bahnverbunds oder geben einen neuen Zielpunkt für den Verbund vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) übergeben. Es werden immer nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
NodePosition[n]	Position3D (optional)	Position für die Stützpunkte des TCP in 3D-Richtungskoordinaten.
NodePosition[n].X	Position3D.X (Double)	Position für den Stützpunkt des TCP in X-Richtung.
NodePosition[n].Y	Position3D.Y (Double)	Position für den Stützpunkt des TCP in Y-Richtung.
NodePosition[n].Z	Position3D.Z (Double)	Position für den Stützpunkt des TCP in Z-Richtung.

Dabei ist *[n]* die Nummer des Stützpunktes.

Syntax

```
Stuetzppunkt3_3D := geoGroup.MoveSpline.NodePosition[3];
Stuetzpunkt3X := geoGroup.MoveSpline.NodePosition[3].X;
Geo.MoveSpline.NodePosition[2]:= Pos2_3D;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

62.3.12 NodeOrientation[n]

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte der Stützpunkte der aktuellen Spline-Bewegung des Bahnverbunds oder geben einen neuen Zielpunkt für den Verbund vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
NodePosition[n]	Position3D (optional)	Ziel der Orientierung für die Stützpunkte des TCP in 3D-Orientierungskoordinaten.
NodePosition[n].U	Position3D.U (Double)	Orientierung für den Stützpunkt des TCP in U-Richtung.
NodePosition[n].V	Position3D.V (Double)	Orientierung für den Stützpunkt des TCP in V-Richtung.
NodePosition[n].W	Position3D.W (Double)	Orientierung für den Stützpunkt des TCP in W-Richtung.

Syntax

```
StuetzpunktOri2_3D := geoGroup.MoveSpline.NodeOrientation[2];
Stuetzpunkt0U := geoGroup.MoveSpline.NodeOrientation[0].U;
geoGroup.MoveSpline.NodeOrientation[3] := Orientierung3_3D;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

62.3.13 NodeCoordinates[n]

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 6D-Koordinatenpunkte der Stützpunkte der aktuellen Spline-Bewegung des Bahnverbunds oder geben einen neuen Stützpunkt für den Verbund vor.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) und 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
NodeCoordinates[n]	Coordinate6D (optional)	Stützpunktkoordinaten des TCP in 6D-Raumkoordinaten.
NodeCoordinates[n].X	Coordinate6D.X (Double)	Position für den Stützpunkt des TCP in X-Richtung.
NodeCoordinates[n].Y	Coordinate6D.Y (Double)	Position für den Stützpunkt des TCP in Y-Richtung.
NodeCoordinates[n].Z	Coordinate6D.Z (Double)	Position für den Stützpunkt des TCP in Z-Richtung.
NodeCoordinates[n].U	Coordinate6D.U (Double)	Orientierung der Stützpunkte des TCP in U-Richtung.
NodeCoordinates[n].V	Coordinate6D.V (Double)	Orientierung der Stützpunkte des TCP in V-Richtung.
NodeCoordinates[n].W	Coordinate6D.W (Double)	Orientierung der Stützpunkte des TCP in W-Richtung.

Dabei ist *[n]* die Nummer des Stützpunktes.

Syntax

```
Stuetzpunkt2_6D := geoGroup.MoveSpline.NodeCoordinates[2];
Stuetzpunkt0Y := geoGroup.MoveSpline.NodeCoordinates[0].Y;
geoGroup.MoveSpline.NodeCoordinates[4]:= Stuetzpunkt4_6D;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

63 MCGeo.MoveSyncAxis

Mit der Funktionalität **MCGeo.MoveSyncAxis** können Sie können Sie Informationen von Synchronachsen eines Bahnverbundes abfragen und den Synchronachsen Bewegungsparameter zuweisen, damit sich diese in gewünschter Weise mit dem Bahnverbund bewegen.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Synchronachsen in Bahnverbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund* und *Bahnverbund – Synchrone Achse* . Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

63.1 Syntax für MCGeo.MoveSyncAxis

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.MoveSyncAxis.<Funktion oder Zugriff auf Wert>(optional
<Parameter>);
```

63.2 Funktionen von MCGeo.MoveSyncAxis

Die Funktionalität bietet folgende Funktionen:

- Start()
- GetTargetPosition()
- SetTargetPosition()
- GetPositioningMode()
- SetPositioningMode()

63.2.1 Start()

Die Funktion fügt dem Bahnpuffer eine neue Bewegung für die Synchronache hinzu. Die Bewegung wird synchron mit den nächsten folgenden Bahnbewegungen des Bahnverbundes gestartet. Die Synchronbewegung wird so auf die Bahnbewegungen aufgeteilt, dass die Zielposition (Parameter `TargetPosition`) erreicht wird, wenn der Bahnverbund mit dem `BlockEnd`-Parameter `InPosState` anhält. Werden nach dem `Start()`-Befehl für die Synchronachse mehrere Bahnstücke in den Bahnpuffer geschrieben, die ohne Anhalten aneinandergehängt werden (`BlockEnd`-Parameter `Smoothing`), so wird die Bewegung der Synchronachse über diese Bahnstücke hinweg ausgeführt und erreicht erst dann die Zielposition, wenn das nächste Bahnstück mit dem `Blockend`-Parameter `InPosState` anhält.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Name der MCAchse	Pointer to MCAxis	Achsobjekt, das synchron mit dem Verbund bewegt werden soll. Diese Achse muss vorher mit AddSyncAxis() [▶ 319] zum Bahnverbund hinzugefügt worden sein.
PositioningMode	MCGeoPositioningModes (optional)	Legt den Positioniermodus für die Synchronbewegung fest.
	MCGeoPositioningModes.Absolute	Die Synchronachse wird mit der nächsten Bahnbewegung auf die im Parameter `TargetPosition` übergebene Position gefahren. Diese wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Positionierung ergibt sich aus der letzten und der neuen Zielposition.
	MCGeoPositioningModes.Relative	Die Synchronachse wird mit der nächsten Bahnbewegung um den Weg weiter gefahren, der im Parameter `TargetPosition` übergeben wurde. Die Zielposition berechnet sich aus der letzten Zielposition addiert mit dem Wert in `TargetPosition`. Die Fahrtrichtung richtet sich nach dem Vorzeichen des Parameters `TargetPosition`.
TargetPosition	Double (optional)	Legt die Zielposition für die Synchronbewegung fest.

Syntax

```
geoGroup.MoveSyncAxis.Start(AxSync, MCGeoPositioningModes.Absolute,300.0);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Freigegeben
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

63.2.2 GetTargetPosition()

Übergabeparameter

Parameter	Datentyp	Bedeutung
Name der MCAchse	Pointer to MCAxis	Liest den zuletzt übergebenen Wert für die Zielposition der Synchronachse. Diese Achse muss vorher mit AddSyncAxis() [▶ 319] zum Bahnverbund hinzugefügt worden sein.

Syntax

```
Zielpos := geoGroup.MoveSyncAxis.GetTargetPosition(AxSync);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

63.2.3 SetTargetPosition()

Übergabeparameter

Parameter	Datentyp	Bedeutung
Name der MCAchse	Pointer to MCAxis	Übergibt einen neuen Wert für die Zielposition der Synchronachse. Diese Achse muss vorher mit AddSyncAxis() [▶ 319] zum Bahnverbund hinzugefügt worden sein. Dieser Wert wird nur im Achsobjekt gespeichert. Wirksam wird er nur mit einem Start() [▶ 373]-Befehl, in dem keine Zielposition übergeben wird.

Syntax

```
geoGroup.MoveSyncAxis.SetTargetPosition(AxSync,400.0);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

63.2.4 GetPositioningMode()

Übergabeparameter

Parameter	Datentyp	Bedeutung
Name der MCAchse	Pointer to MCAxis	Liest den zuletzt übergebenen Wert für den Positioniermodus der Synchronachse. Die Achse muss vorher mit AddSyncAxis() [▶ 319] zum Bahnverbund hinzugefügt worden sein.

Syntax

```
PosMod := geoGroup.MoveSyncAxis.GetPositioningMode(AxSync);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

63.2.5 SetPositioningMode()

Übergabeparameter

Parameter	Datentyp	Bedeutung
Name der MCAchse	Pointer to MCAxis	Achsobjekt, das synchron mit dem Verbund bewegt werden soll. Diese Achse muss vorher mit AddSyncAxis() [▶ 319] zum Bahnverbund hinzugefügt worden sein.
PositioningMode	MCGeoPositioningModes (optional)	Legt den Positioniermodus für die Synchronbewegung fest.
	MCGeoPositioningModes.Absolute	Die Synchronachse wird mit der nächsten Bahnbewegung auf die im Parameter `TargetPosition` übergebene Position gefahren. Diese wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Positionierung ergibt sich aus der letzten und der neuen Zielposition.
	MCGeoPositioningModes.Relative	Die Synchronachse wird mit der nächsten Bahnbewegung um den Weg weitergefahren, der im Parameter `TargetPosition` übergeben wurde. Die Zielposition berechnet sich aus der letzten Zielposition addiert mit dem Wert in `TargetPosition`. Die Fahrtrichtung richtet sich nach dem Vorzeichen des Parameters `TargetPosition`.

Syntax

```
geoGroup.MoveSyncAxis.SetPositioningMode(AxSync,MCGeoPositioningModes.Absolute);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

64 MCGeo.Orientation

Mit der Funktionalität **MCGeo.Orientation** können Sie bei Kinematiken, die eine oder mehrere Orientierungsachsen besitzen, die aktuellen Werte für die 3D-Orientierung nach Koordinaten sortiert abfragen.

Die Orientierung des Tool Center Points (TCP) bezeichnet die sogenannten **Kardanwinkel**, in der Drehachsenreihenfolge ***z – y‘ – x‘‘***, in Bezug auf das Basis-Koordinatensystem.

Ermittlung der Kardanwinkel

Um aus dem Basis-Koordinatensystem (x, y, z) das Koordinatensystem des TCP ***(x‘‘‘, y‘‘‘, z‘‘‘)*** abzuleiten, wird folgendes Schema angewandt:

1. Zuerst wird das ortsfeste Koordinatensystem um den Winkel U um die ***z***-Achse gedreht. Es entsteht das Koordinatensystem ***(x‘, y‘, z‘)***.
2. Dieses wird nun um den Winkel V um die ***y‘***-Achse gedreht, wobei das Koordinatensystem ***(x‘‘, y‘‘, z‘‘)*** entsteht.
3. Zuletzt wird dieses Koordinatensystem um den Winkel W um die ***x‘‘***-Achse gedreht. Dadurch entsteht das Koordinatensystem des TCP ***(x‘‘‘, y‘‘‘, z‘‘‘)***.

Bedeutung der Kardanwinkel

– Der Winkel U (auch Gierwinkel) beschreibt die Ausrichtung des TCP in Bezug auf die ***x-y***-Ebene des Basiskoordinatensystems.

– Der Winkel V (auch Nickwinkel) beschreibt die Ausrichtung des TCP in Bezug auf die ***y-z***-Ebene des Basiskoordinatensystems.

– Der Winkel W (auch Rollwinkel) beschreibt die Drehung des TCP um die ***x***-Achse des TCP-Koordinatensystems.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

64.1 Syntax für MCGeo.Orientation

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Orientation.<Zugriff auf Wert>;
```

64.2 Eigenschaften von MCGeo.Orientation

Die Funktionalität bietet folgende Eigenschaften:

– Actual

– Setpoint

– Target

64.2.1 Actual

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte des Bahnverbunds.

Die Koordinatenwerte werden in einer Struktur mit 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
Actual	Orientation3D	Istwert der Orientierung des TCP in 3D-Orientierungskoordinaten.
Actual.U	Orientation3D.U (Double)	Aktueller Wert der Orientierung des TCP in U-Richtung.
Actual.V	Orientation3D.V (Double)	Aktueller Wert der Orientierung des TCP in V-Richtung.
Actual.W	Orientation3D.W (Double)	Aktueller Wert der Orientierung des TCP in W-Richtung.

Syntax

```
IstOri3D := geoGroup.Orientation.Actual;
IstOriU := geoGroup.Orientation.Actual.U;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

64.2.2 Setpoint

Beschreibung

Mithilfe der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte des Sollwertgenerators des Bahnverbunds.

Die Koordinatenwerte werden in einer Struktur mit 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
Setpoint	Orientation3D	Sollwert der Orientierung des TCP in 3D-Orientierungskoordinaten.
Setpoint.U	Orientation3D.U (Double)	Sollwert der Orientierung des TCP in U-Richtung.
Setpoint.V	Orientation3D.V (Double)	Sollwert der Orientierung des TCP in V-Richtung.
Setpoint.W	Orientation3D.W (Double)	Sollwert der Orientierung des TCP in W-Richtung.

Syntax

```
SollwertOri3D := geoGroup.Orientation.Setpoint;
SollwertU := geoGroup.Orientation.Setpoint.U;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

64.2.3 Target

Beschreibung

Mithilfe der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte des Ziels der aktuellen Bahnbewegung des Bahnverbunds.

Die Koordinatenwerte werden in einer Struktur mit 3 Orientierungskoordinaten (U, V, W) übergeben. Es werden nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
Target	Orientation3D	Ziel der Orientierung des TCP in 3D-Orientierungskoordinaten.
Target.U	Orientation3D.U (Double)	Ziel der Orientierung des TCP in U-Richtung.
Target.V	Orientation3D.V (Double)	Ziel der Orientierung des TCP in V-Richtung.
Target.W	Orientation3D.W (Double)	Ziel der Orientierung des TCP in W-Richtung.

Syntax

```
ZielwertOri3D := geoGroup.Orientation.Target;
ZielwertU := geoGroup.Orientation.Target.U;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

65 MCGeo.Position

Mit der Funktionalität **MCGeo.Orientation** können Sie die aktuellen Werte für die 3D-Bahnbewegung nach Koordinaten sortiert abfragen.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

65.1 Syntax für MCGeo.Position

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Position.<Zugriff auf Wert>;
```

65.2 Eigenschaften von MCGeo.Position

Die Funktionalität bietet folgende Eigenschaften:

- Actual
- Setpoint
- Target

65.2.1 Actual

Beschreibung

Mit der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte des Bahnverbunds.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) übergeben. Es werden immer nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
Actual	Position3D	Istwert der Position des TCP in 3D-Richtungskoordinaten.
Actual.X	Position3D.X (Double)	Aktueller Wert der Position des TCP in X-Richtung
Actual.Y	Position3D.Y (Double)	Aktueller Wert der Position des TCP in Y-Richtung
Actual.Z	Position3D.Z (Double)	Aktueller Wert der Position des TCP in Z-Richtung

Syntax

```
Ist3D := geoGroup.Position.Actual;
IstX := geoGroup.Position.Actual.X;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

65.2.2 Setpoint

Beschreibung

Mithilfe der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte des Sollwertgenerators des Bahnverbunds.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) übergeben. Es werden immer nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
Setpoint	Position3D	Sollwert der Position des TCP in 3D-Raumkoordinaten.
Setpoint.X	Position3D.X (Double)	Sollwert der Position des TCP in X-Richtung.
Setpoint.Y	Position3D.Y (Double)	Sollwert der Position des TCP in Y-Richtung.
Setpoint.Z	Position3D.Z (Double)	Sollwert der Position des TCP in Z-Richtung.

Syntax

```
Sollwert3D := geoGroup.Position.Setpoint;
SollwertX := geoGroup.Position.Setpoint.X;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

65.2.3 Target

Beschreibung

Mithilfe der Eigenschaft lesen Sie die momentanen 3D-Koordinatenpunkte des Ziels der aktuellen Bahnbewegung des Bahnverbunds.

Die Koordinatenwerte werden in einer Struktur mit 3 Richtungskoordinaten (X, Y, Z) übergeben. Es werden immer nur die Werte übergeben, die für die Kinematik des Bahnverbunds relevant sind. Die anderen Werte bleiben Null.

Eigenschaft	Datentyp	Bedeutung
Target	Position3D	Ziel der Position des TCP in 3D-Raumkoordinaten.
Target.X	Position3D.X (Double)	Ziel der Position des TCP in X-Richtung.
Target.Y	Position3D.Y (Double)	Ziel der Position des TCP in Y-Richtung.
Target.Z	Position3D.Z (Double)	Ziel der Position des TCP in Z-Richtung.

Syntax

```
Zielwert3D := geoGroup.Position.Target;
ZielwertX := geoGroup.Position.Target.X;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

66 MCGeo.Power

Mit der Funktionalität **MCGeo.Power** können Sie die Regelung und damit die Drehmomenterzeugung eines Bahnverbundes freigeben oder sperren. Alle am Verbund beteiligten Achsen werden gleichzeitig in den neuen Betriebszustand versetzt.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

66.1 Syntax für MCGeo.Power

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Power.<Funktion>(optional <Parameter>;
```

66.2 Funktionen von MCGeo.Power

Die Funktionalität bietet folgende Funktionen:

– Enable()
– Disable()
– Quickstop()

66.2.1 Enable()

Die Funktion aktiviert die Regler der Servoverstärker aller Verbundachsen und gibt die Spannung auf den Motor frei. Ein Bahnverbund kann nur freigegeben werden, wenn alle Verbundachsen betriebsbreit sind.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.Power.Enable();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Freigegeben
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Der Zustandsübergang kann nicht in einem MCX-Zyklus abgearbeitet werden. Fragen Sie deshalb das Erreichen des neuen Zustands im Programm ab, bevor Sie weitere Befehle für den Verbund aufrufen.

```
When geoGroup.State.IsEnabled Continue;
```

66.2.2 Disable()

Die Funktion sperrt den Servoverstärker aller Verbundachsen für die Drehmomenterzeugung (Impulssperre). Der Motoren werden sofort stromlos und trudelt aus, falls keine Bremse aktiv wird.

Übergabeparameter

Parameter	Datentyp	Bedeutung
DisableModes	MCPowerDisableMode (optional)	Modus für die Ausführung der `Disable()`-Funktion.
	MCPowerDisableMode.Forced (Default)	Die Impulssperre wirkt in jedem Fall sofort. Dieser Modus ist auch aktiv, wenn kein Parameter angegeben wird.
	MCPowerDisableMode.StateCheck	Die Impulssperre wird nur dann wirksam, wenn sich der Verbund im Betriebszustand **Freigegeben** befindet. In allen anderen Zuständen wird ein Bedienfehler ausgegeben.

Syntax

```
geoGroup.Power.Disable();
geoGroup.Power.Disable(MCPowerDisableModes.Forced);
geoGroup.Power.Disable(MCPowerDisableModes.StateCheck);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Für den Aufruf ohne Parameter oder mit dem Parameter `MCPowerDisableModes.Forced` gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	–	Gesperrt
Freigegeben	√	Gesperrt
Läuft	√	Gesperrt
Unterbrochen	√	Gesperrt
Fehler: freigegeben	√	Fehler: gesperrt
Fehler: gesperrt	–	Fehler: gesperrt

Für den Aufruf mit dem Parameter `MCPowerDisableModes.StateCheck` gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	–	Gesperrt
Freigegeben	√	Gesperrt
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Der Zustandsübergang kann nicht in einem MCX-Zyklus abgearbeitet werden. Bevor Sie weitere Befehle für den Verbund aufrufen, fragen Sie deshalb das Erreichen des neuen Zustands im Programm wie folgt ab:

```
When geoGroup.State.IsDisabled Continue;
```

Hat sich die Achse vorher im Zustand **Fehler: freigegeben** befunden, verwenden Sie folgenden Befehl.

```
When geoGroup.State.IsDisabledError Continue;
```

66.2.3 QuickStop()

Die Funktion hält alle Verbundachsen mit der Schnellhaltverzögerung an und sperrt die Servoverstärker nach Erreichen des Stillstands.

HINWEIS

Keine Lageregelung der Verbundachsen bei Schnellhalt

Während der Schnellhaltverzögerung findet keine Lageregelung statt, und damit auch keine koordinierte Bewegung der Verbundachsen entsprechend der Bahnbewegung. Jede Verbundachse kommt abhängig von ihrer aktuellen Geschwindigkeit und der Einstellung für die Schnellhaltverzögerung zu einem anderen Zeitpunkt zum Stehen.

► Verwenden Sie den Befehl nur in Notfällen!

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.Power.QuickStop();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	–	Gesperrt
Freigegeben	√	Gesperrt
Läuft	√	Gesperrt
Unterbrochen	√	Gesperrt
Fehler: freigegeben	√	Fehler: gesperrt
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Der Zustandsübergang kann nicht in einem MCX-Zyklus abgearbeitet werden. Bevor Sie weitere Befehle für den Verbund aufrufen, fragen Sie deshalb das Erreichen des neuen Zustands im Programm wie folgt ab:

```
When geoGroup.State.IsDisabled Continue;
```

66.3 Beispieltask MCGeo.Power

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Task Sample1 Autorun
    //Freigeben der Verbundachsen
    If geoGroup.State.IsDisabled Then
        geoGroup.Power.Enable();
    End_if;
    When geoGroup.State.IsEnabled Continue;

    //hier kann ein Prozess mit dem Verbund stehen oder andere Tasks bewegen
den Verbund
    //…

    // Verbund sperren
    If geoGroup.State.IsRunning Then    //falls der Verbund noch in Bewegung
ist, mit sinnvoller Verzögerung anhalten

        geoGroup.MoveHalt.Start();
    When geoGroup.Mechanism.Slope.IsStopped Continue;

    End_if;
    geoGroup.Power.Disable(); // Verbund sperren
    When geoGroup.State.IsDisabled Continue;

End_Task;
```

67 MCGeo.Settings

Mit der Funktionalität **MCGeo.Settings** können Sie alle Parameter einer MCX-Bahnverbunds gemeinsam anpassen oder abspeichern.

Parameter bei aktivem Speichervorgang nicht anpassen

Das Speichern der Parameter in eine Datei ist ein Vorgang, der im Hintergrund abgearbeitet wird. D. h. er erstreckt sich über mehrere Zyklen der MCX. Werden während des Abspeicherns Parameterwerte geändert, ist nicht vorhersehbar, ob der alte oder neue Wert in die Datei geschrieben wird.

Funktionsweise der Parameterverwaltung

Parameter, die im Motion-Setup von JetSym gespeichert werden, werden in einer Textdatei auf dem PC und, auf Nachfrage, auch im Dateisystem der Steuerung (Unterordner Motion) abgelegt. Der Dateiname der Parameterdatei auf der Steuerung ist *MCGeo_X.ini*, wobei das *X* für die Nummer der Bahnverbund-ID steht. Die Bahnverbund-ID wird auf der Seite Eigenschaften im Motion-Setup des MCX-Verbunds angezeigt. Außerdem zeigt der Tooltipp im Hardware-Baum die Bahnverbund-ID an, wenn der Mauszeiger auf dem Verbund liegt.)

Beim Booten der Steuerung liest der MCX-Kernel die Daten der *MCGeo_X.ini*-Datei und verwendet diese für den betreffenden Verbund. Ist eine solche Datei auf der Steuerung vorhanden, muss also keine Parametrierung mehr vorgenommen werden. Ist keine entsprechende Datei für den Verbund vorhanden, werden die Parameter mit Default-Werten geladen.

Im Normalfall ist keine Anpassung an der Datei aus dem STX-Programm heraus erforderlich.

In ganz wenigen Ausnahmefällen kann es jedoch sinnvoll sein, die Parameterdatei auf der Steuerung neu einzulesen oder die Parameter abzuspeichern. Beim Abspeichern muss jedoch bedacht werden, dass die im JetSym-Projekt gespeicherte Parameter-Datei dann unter Umständen nicht mehr mit den auf der Maschine wirksamen Parametern übereinstimmt.

67.1 Syntax für MCGeo.Settings

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Settings.<Funktion>(optional <Parameter>;
```

67.2 Funktionen von MCGeo.Settings

Die Funktionalität bietet folgende Funktionen:

– SaveToFile()
– LoadFromFile()

67.2.1 SaveToFile()

Übergabeparameter

Parameter	Datentyp	Bedeutung
Suffix	Integer (optional)	Aktiviert das Abspeichern der Parameter des MCX-Verbunds in eine Datei im Unterordner Motion im Dateisystem der Steuerung. Abhängig vom Übergabeparameter werden unterschiedliche Dateien erzeugt bzw. überschrieben: – Parameter `0` oder kein Parameter: Die Parameterdatei *MCGeo_X.ini* für diesen Verbund wird mit den aktuellen Werten erstellt oder überschrieben. – Parameter ungleich `0`: Die Parameterdatei *MCGeo_X_TmpY.ini* wird mit den aktuellen Werten erstellt oder überschrieben. Dabei ist *Y* gleich dem Parameter `Suffix`.

Syntax

```
geoGroup.Settings.SaveToFile();    // überschreibt (oder erzeugt) die Datei
MCGeo_1.ini (wenn geoGroup die Geo-ID 1 hat)
geoGroup.Settings.SaveToFile(3);    // erzeugt die Datei MCGeo_1_Tmp3.ini
(wenn geoGroup die Geo-ID 1 hat)
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

67.2.2 LoadFromFile()

Übergabeparameter

Parameter	Datentyp	Beschreibung
Suffix	Integer (optional)	Aktiviert das Übernehmen der Parameter des MCX-Verbunds aus einer Datei im Unterordner Motion im Dateisystem der Steuerung. Abhängig vom Übergabeparameter werden unterschiedliche Dateien gelesen. – Parameter `0` oder kein Parameter: Die Daten aus der Datei *MCGeo_X.ini* werden für den betreffenden Verbund gelesen und verwendet. – Parameter ungleich `0`: Die Daten aus der Datei *MCGeo_TmpY.ini* werden für den betreffenden Verbund gelesen und verwendet. Dabei ist *Y* gleich dem Parameter `Suffix`.

Syntax

```
geoGroup.Settings.LoadFromFile();    //Liest die Datei MCGeo_1.ini (wenn
geoGroup die Geo-ID 1 hat)
geoGroup.Settings.LoadFromFile(3);    //Liest die Datei MCGeo_1_Tmp3.ini
(wenn geoGroup die Geo-ID 1 hat)
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

68 MCGeo.State

Mit der Funktionalität **MCGeo.State** können Sie verschiedene Informationen eines Bahnverbundes abfragen, wie z. B. Betriebsart, Bewegungsart etc.

Grundlegende und weiterführende Informationen

Beachten Sie in diesem Zusammenhang ebenso folgende Informationen:

– Für Zustandsübergänge den Abschnitt MCGeo.State.Transitions [▶ 400];

– Für eine detaillierte Beschreibung des Aufbaus des MCX-Kerns die JetSym–Onlinehilfe (JetSym Motion und MotionControl > JetSym MotionControl > MotionControl-Grundlagen > Bedienkonzept > Überblick: Bedienkonzept)

– Für grundlegende Informationen die Motion Control Application Note Nr. 54 (MCX-Motion-Control-Grundlagen).

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

Betriebsart

Die Betriebsart (`OperatingMode`) gibt an, auf welche Weise der Bahnverbund grundsätzlich arbeitet. Es gibt 3 Betriebsarten:

Betriebsart	Bedeutung
`Inactive` (inaktiv)	Der Verbund kann keine Befehle entgegen nehmen, z. B. weil er nicht aktiviert wurde.
`Normal` (normal)	Der Verbund ist verwendbar. Verfahrbefehle werden gepuffert ausgeführt (siehe MCGeo.Buffer [▶ 265]). D. h. alle Bewegungsbefehle werden in einen Vorlaufpuffer aufgenommen und nacheinander abgearbeitet.
`Error` (Fehler)	Es liegt ein Bedienfehler vor. Der Betriebszustand bleibt erhalten, Es können jedoch keine neuen Funktionen und Parameter übergeben werden.

Tab. 6: Betriebsarten von MCX-Bahnverbunden

Betriebszustand

Ein Verbund befindet sich zu jedem Zeitpunkt in einem, und nur einem, klar definierten Betriebszustand (`OperatingState`). Ist der Verbund beispielsweise im Betriebszustand **Läuft**, dann ist der Betriebszustand **Freigegeben** nicht aktiv, obwohl natürlich alle Verbundachsen eine Impulsfreigabe haben.

In einem bestimmten Zustand ist immer nur eine Untermenge aller Funktionsaufrufe zulässig. Das Zustandsmodell für die Achse sieht folgendermaßen aus:

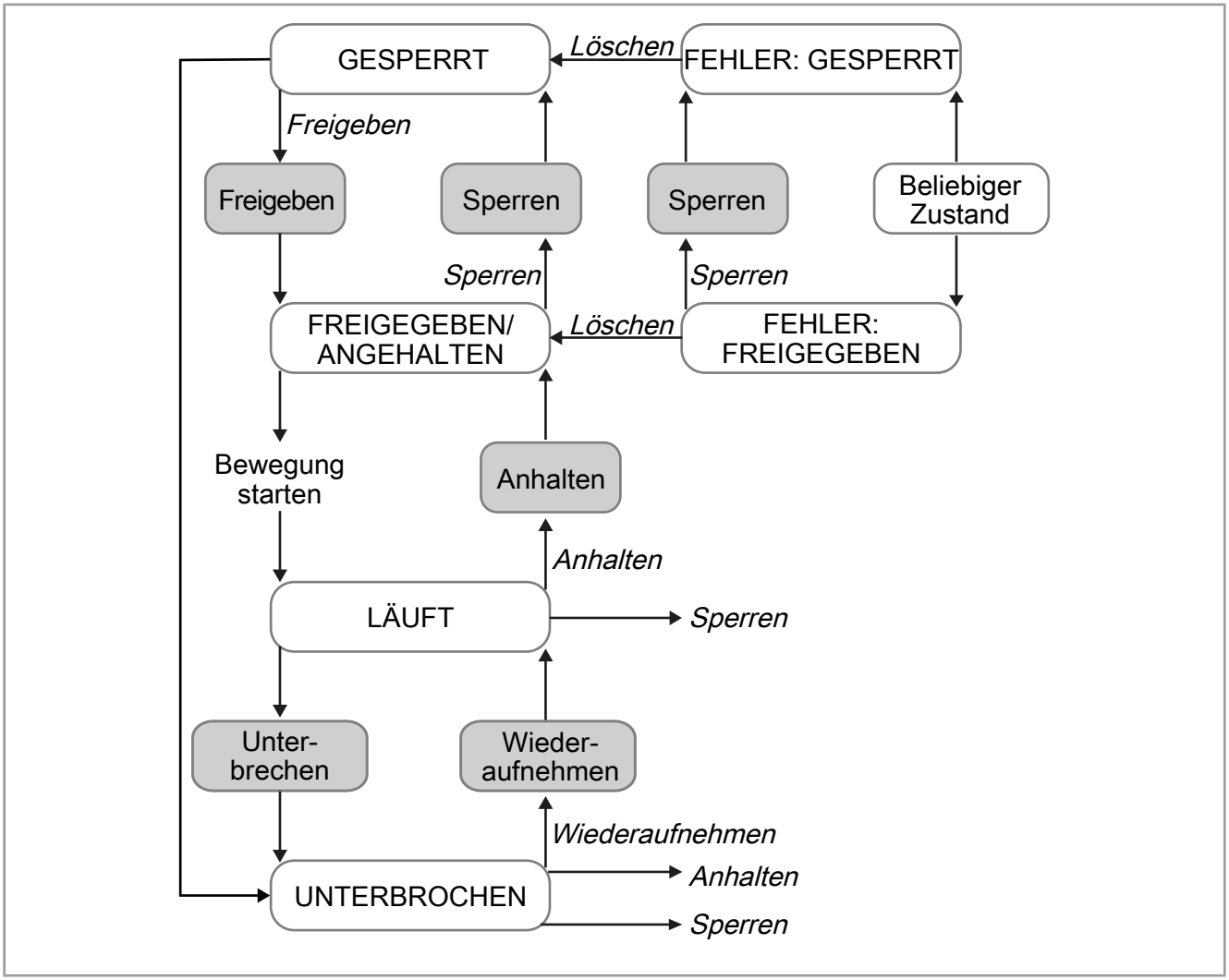


Abb. 13: Zustandsamodell für MCX-Verbunde

Betriebszustand	Bedeutung
Inactive (Inaktiv)	Der Verbund kann keine Befehle entgegen nehmen, z. B. weil er nicht aktiviert wurde.
Disabled (Gesperrt)	– Die Regler des Servoverstärkers aller Verbundachsen sind geöffnet (keine Impulsfreigabe). – Es wird kein Drehmoment erzeugt.
Enabled (Freigegeben)	– Die Regler aller Verbundachsen sind geschlossen (Impulsfreigabe). – Der Sollwertinterpolator befindet sich im Stillstand.
Running (Läuft)	Mindestens eine Achse ist in Bewegung; d. h. der Sollwertinterpolator gibt Bewegungssollwerte an den Servoverstärker weiter.
Suspended (Unterbrochen)	– Entspricht dem Betriebszustand **Freigegeben** mit der Ausnahme, dass die Zielposition erhalten bleibt. Eine mit `MCGeo.State.Transitions.Suspend()` unterbrochene Bewegung kann über den Befehl `MCGeo.State.Transitions.Resume()` (Zustandsübergang wiederaufnehmen) fortgesetzt werden.
EnabledError (Fehler: freigegeben)	Es ist ein Fehler aufgetreten. Nach der Fehlerreaktion bleiben alle Achsen weiterhin in Regelung (Fehlerstufe 4).
DisabledError (Fehler: gesperrt)	Es ist ein Fehler aufgetreten. Nach der Fehlerreaktion ist die Regelung aller Achsen gesperrt (Fehlerstufen 5 und 6).

Tab. 7: Betriebszustände von Bahnverbunden

Bewegungsart

Die Bewegungsart (`TypeOfMotion`) gibt an, welche Art der Bewegung der Sollwertgenerator für den Bahnverbund aktuell erzeugt.

Die Abfrage der Bewegungsart ist beispielsweise in einem Überwachungstask interessant, in dem der Prozess nicht bekannt ist. Ausgehend von der aktiven Bewegungsart vor dem Abschalten ist es möglich, das Wiederanlaufen korrekt zu steuern.

Betriebszustand	Bedeutung
Keine Bewegung	Diese Bewegungsart ist bis zur ersten Bewegung des Verbundes aktiv.
Gerade (`MoveLinear`)	Der Bahnverbund führt aktuell, oder führte zuletzt, eine Linearbewegung aus.
Kreisbewegung 2D (`MoveCircular2D`)	Der Bahnverbund führt aktuell, oder führte zuletzt, eine 2D-Kreisbewegung aus.
Kreisbewegung 3D (`MoveCircular3D`)	Der Bahnverbund führt aktuell, oder führte zuletzt, eine 3D-Kreisbewegung aus.
Spline (`MoveSpline`)	Der Bahnverbund führt aktuell, oder führte zuletzt, eine Spline-Bewegung aus.
Polynom (`MovePolynominal`)	Der Bahnverbund führt eine Überschleifbewegung aus.
Punkt zu Punkt (`MovePtp`)	Der Bahnverbund führt aktuell, oder führte zuletzt, eine Punkt-zu-Punkt-Bewegung aus.
Orientierung (`MoveOrientation`)	Der Bahnverbund führt aktuell, oder führte zuletzt, eine Orientierungsbewegung aus.
Gekoppelte Bewegung (`MoveExternal`)	Der Bahnverbund wird aktuell, oder wurde zuletzt, über eine gekoppelte Bewegung verfahren.

68.1 Eigenschaften von MCGeo.State

Die Funktionalität bietet folgende Eigenschaften:

- Value
- IsDisabled
- IsEnabled
- IsRunning
- IsSupended
- IsDisabledError
- IsEnabledError
- IsInactive
- OperatingMode
- TypeOfMotion
- Transitions (Beschreibung siehe MCGeo.State.Transitions [▶ 400])

68.1.1 Value

Beschreibung

Datentyp	Bedeutung
OperatingStates	Liest den Betriebszustand, in dem sich der Bahnverbund befindet (für eine detaillierte Beschreibung siehe Abschnitt *Betriebszustand* in MC-Geo.State [▶ 389]).
OperatingStates.Inactive	Inaktiv
OperatingStates.Disabled	Gesperrt
OperatingStates.Enabled	Freigegeben
OperatingStates.Running	Läuft
OperatingStates.Suspended	Unterbrochen
OperatingStates.EnabledError	Fehler: freigegeben
OperatingStates.DisabledError	Fehler: gesperrt

Syntax

```
OpState := geoGroup.State.Value;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

68.1.2 IsDisabled

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund im Betriebszustand **Gesperrt** befindet.

Syntax

```
When geoGroup.State.IsDisabled Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

68.1.3 IsDisabledError

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund im Betriebszustand **Fehler: gesperrt** befindet.

Syntax

```
When geoGroup.State.IsDisabledError Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

68.1.4 IsEnabled

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich Verbund im Betriebszustand **Freigegeben** befindet.

Syntax

```
When geoGroup.State.IsEnabled Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

68.1.5 IsEnabledError

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund im Betriebszustand **Fehler: freigegeben** befindet.

Syntax

```
When geoGroup.State.IsEnabledError Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

68.1.6 IsInactive

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund im Betriebszustand **Inaktiv** befindet.

Syntax

```
When geoGroup.State.IsInactive Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

68.1.7 IsRunning

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund im Betriebszustand **Läuft** befindet.

Syntax

```
When geoGroup.State.IsRunning Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

68.1.8 IsSuspended

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund im Betriebszustand **Unterbrochen** befindet.

Syntax

```
When geoGroup.State.IsSuspended Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

68.1.9 OperatingMode.Value

Beschreibung

Datentyp	Bedeutung
Typ MCOperatingModes	Liest die Betriebsart, in der sich der Verbund befindet (für eine detaillierte Beschreibung siehe Abschnitt Betriebsart [▶ 389]).
MCOperatingModes.Inactive	Der Verbund ist inaktiv, z. B. weil er nicht aktiviert wurde.
MCOperatingModes.Normal	Der Verbund wurde aktiviert und befindet sich nicht im Fehlerzustand.
MCOperatingModes.Error	Der Verbund befindet sich nach einem Bedienfehler im Fehlerzustand.

Syntax

```
OpModeAchse := geoGroup.State.OperatingMode.Value;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

OperatingMode.IsInInactiveMode

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Betriebsart *Inaktiv* befindet.

Syntax

```
When geoGroup.State.OperatingMode.IsInInactiveMode Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

OperatingMode.IsInErrorMode

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Vebund in der Betriebsart *Fehler* befindet.

Syntax

```
When geoGroup.State.OperatingMode.IsInErrorMode Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

OperatingMode.IsInNormalMode

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund in der Betriebsart *Normal* befindet.

Syntax

```
When geoGroup.State.OperatingMode.IsInNormalMode Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

68.1.10 TypeOfMotion.Value

Beschreibung

Betriebszustand	Bedeutung
Typ MCGeoTypesOfMotion	Bewegungsart in der sich der Bahnverbund gerade befindet (für eine detaillierte Beschreibung siehe Abschnitt MCGeo.State [▶ 391]).
MCGeoTypesOfMotion.MoveLinear	Der Bahnverbund führt aktuell, oder führte zuletzt, eine Linearbewegung aus.
MCGeoTypesOfMotion.MoveCircular2D	Der Bahnverbund führt aktuell, oder führte zuletzt, eine 2D-Kreisbewegung aus.
MCGeoTypesOfMotion.MoveCircular3D	Der Bahnverbund führt aktuell, oder führte zuletzt, eine 3D-Kreisbewegung aus.
MCGeoTypesOfMotion.MoveSpline	Der Bahnverbund führt aktuell, oder führte zuletzt, eine Spline-Bewegung aus.
MCGeoTypesOfMotion.MovePolynominal	Der Bahnverbund führt eine Überschleifbewegung aus.
MCGeoTypesOfMotion.MovePtp	Der Bahnverbund führt aktuell, oder führte zuletzt, eine Punkt-zu-Punkt-Bewegung aus.
MCGeoTypesOfMotion.MoveOrientation	Der Bahnverbund führt aktuell, oder führte zuletzt, eine Orientierungsbewegung aus.
MCGeoTypesOfMotion.MoveExternal	Der Bahnverbund wird aktuell, oder wurde zuletzt, über eine gekoppelte Bewegung verfahren.

Syntax

```
TypeOfMotion := geoGroup.State. TypeOfMotion.Value;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveLinear

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund in der Bewegungsart `Linear` befindet.

Syntax

```
When geoGroup.State.TypeOfMotion.IsMoveLinear Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveCircular2D

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob der Verbund aktuell eine `2D-Kreisbewegung` ausführt oder zuletzt ausgeführt hat.

Syntax

```
When geoGroup.State.TypeOfMotion.IsMoveCircular2D Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveCircular3D

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob der Verbund aktuell eine `3D-Kreisbewegung` ausführt oder zuletzt ausgeführt hat.

Syntax

```
When geoGroup.State.TypeOfMotion.IsMoveCircular3D Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveSpline

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob der Verbund aktuell eine `Spline-Bewegung` ausführt oder zuletzt ausgeführt hat.

Syntax

```
When geoGroup.State.TypeOfMotion.IsMoveSpline Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMovePolynominal

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob der Verbund aktuell eine `Überschleifbewegung` ausführt oder zuletzt ausgeführt hat.

Syntax

```
When geoGroup.State.TypeOfMotion.IsMovePolynominal Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMovePtp

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob der Verbund aktuell eine `Punkt-zu-Punkt-Bewegung` ausführt oder zuletzt ausgeführt hat.

Syntax

```
When geoGroup.State.TypeOfMotion.IsMovePtp Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveOrientation

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob der Verbund aktuell eine `Orientierungsbewegung` ausführt oder zuletzt ausgeführt hat.

Syntax

```
When geoGroup.State.TypeOfMotion.IsMoveOrientation Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveExternal

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob der Verbund aktuell eine `gekoppelte Bewegung` ausführt oder zuletzt ausgeführt hat.

Syntax

```
When geoGroup.State.TypeOfMotion.IsMoveExternal Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

68.2 Beispieltask MCGeo.State

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Task Sample1 Autorun

    //Freischalten des Bahnvebundes
    If geoGroup.State.IsDisabled Then
        geoGroup.Power.Enable();
    End_if;
    When geoGroup.State.IsEnabled Continue;

    //Hier kann ein Prozess mit dem Verbund stehen, oder andere Tasks bewe-
gen den Verbund
    //…

    // Verbund sperren
    If geoGroup.State.IsRunning Then //falls der Verbund noch in Bewegung
ist
        geoGroup.MoveHalt.Start(); //Verbund anhalten
    When geoGroup.State.IsEnabled Continue
    End if;

    geoGroup.Power.Disable() //Verbund sperren
    When geoGroup.State.IsDisabled Continue

End_Task;
```

69 MCGeo.State.Transitions

Mit der Funktionalität **MCGeo.State.Transitions** können Sie für spezielle Anwendungen Zustandsübergänge eines Bahnverbundes steuern.

Mit Ausnahme der Funktionen Suspend() und Resume() werden in der Regel die Zustandsübergänge eines Verbundes durch die Bewegungsbefehle und das Verhalten der Achsbewegung automatisch ausgelöst. Nur in Ausnahmefällen – z. B. zur Reaktion im Fehlerfall – ist die direkte Steuerung der Zustandsverwaltung erforderlich. Durch die Funktionalität **MCGeo.State.Transitions**, wie auch `Diagnostics.ClearErrors()`, können Betriebszustandsübergänge explizit ausgelöst werden.

Mit der Funktionalität **MCGeo.State** können Sie verschiedene Informationen eines Bahnverbundes abfragen, wie z. B. Betriebsart, Bewegungsart etc.

Grundlegende und weiterführende Informationen

Beachten Sie in diesem Zusammenhang ebenso folgende Informationen:

– Für eine Beschreibung der übrigen Funktionalitäten von MCGeo.State den Abschnitt MCGeo.State [▶ 389];

– Für eine detaillierte Beschreibung des Aufbaus des MCX-Kerns die JetSym–Onlinehilfe (JetSym Motion und MotionControl > JetSym MotionControl > MotionControl-Grundlagen > Bedienkonzept > Überblick: Bedienkonzept)

– Für grundlegende Informationen die Motion Control Application Note Nr. 54 (MCX-Motion-Control-Grundlagen).

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. ??** *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Darüber hinaus bietet die JetSym–Onlinehilfe nützliche Informationen zu Bahnverbunden.

Betriebsart

Die Betriebsart (`OperatingMode`) gibt an, auf welche Weise der Bahnverbund grundsätzlich arbeitet. Es gibt 3 Betriebsarten:

Betriebsart	Bedeutung
`Inactive` (inaktiv)	Der Verbund kann keine Befehle entgegen nehmen, z. B. weil er nicht aktiviert wurde.
`Normal` (normal)	Der Verbund ist verwendbar. Verfahrbefehle werden gepuffert ausgeführt (siehe MCGeo.Buffer [▶ 265]). D. h. alle Bewegungsbefehle werden in einen Vorlaufpuffer aufgenommen und nacheinander abgearbeitet.
`Error` (Fehler)	Es liegt ein Bedienfehler vor. Der Betriebszustand bleibt erhalten, Es können jedoch keine neuen Funktionen und Parameter übergeben werden.

Tab. 8: Betriebsarten von MCX-Bahnverbunden

Betriebszustand

Ein Verbund befindet sich zu jedem Zeitpunkt in einem, und nur einem, klar definierten Betriebszustand (`OperatingState`). Ist der Verbund beispielsweise im Betriebszustand **Läuft**, dann ist der Betriebszustand **Freigegeben** nicht aktiv, obwohl natürlich alle Verbundachsen eine Impulsfreigabe haben.

In einem bestimmten Zustand ist immer nur eine Untermenge aller Funktionsaufrufe zulässig. Das Zustandsmodell für die Achse sieht folgendermaßen aus:

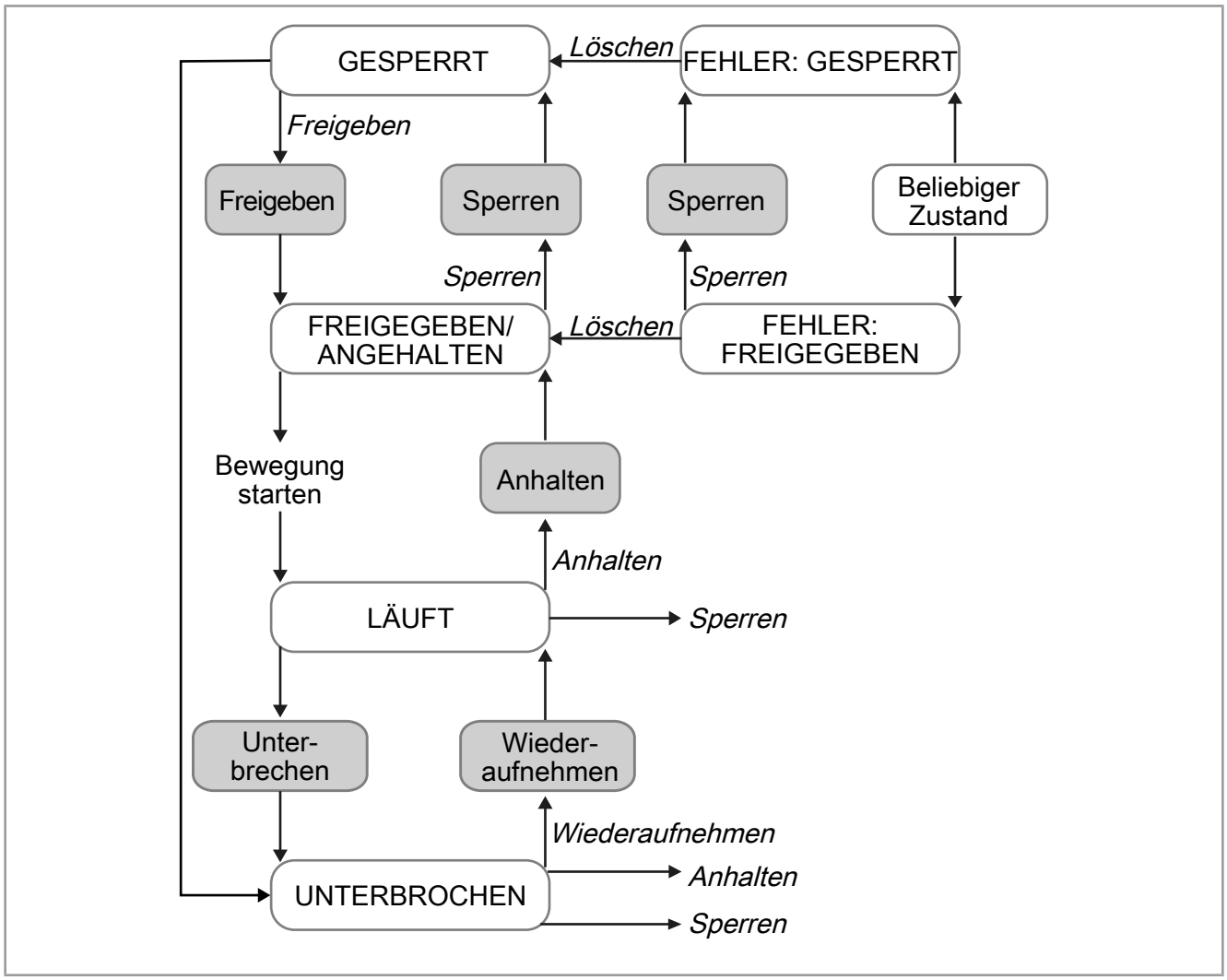


Abb. 14: Zustandsamodell für MCX-Verbunde

Betriebszustand	Bedeutung
Inactive (Inaktiv)	Der Verbund kann keine Befehle entgegen nehmen, z. B. weil er nicht aktiviert wurde.
Disabled (Gesperrt)	– Die Regler des Servoverstärkers aller Verbundachsen sind geöffnet (keine Impulsfreigabe). – Es wird kein Drehmoment erzeugt.
Enabled (Freigegeben)	– Die Regler aller Verbundachsen sind geschlossen (Impulsfreigabe). – Der Sollwertinterpolator befindet sich im Stillstand.
Running (Läuft)	Mindestens eine Achse ist in Bewegung; d. h. der Sollwertinterpolator gibt Bewegungssollwerte an den Servoverstärker weiter.
Suspended (Unterbrochen)	– Entspricht dem Betriebszustand **Freigegeben** mit der Ausnahme, dass die Zielposition erhalten bleibt. Eine mit `MCGeo.State.Transitions.Suspend()` unterbrochene Bewegung kann über den Befehl `MCGeo.State.Transitions.Resume()` (Zustandsübergang wiederaufnehmen) fortgesetzt werden.
EnabledError (Fehler: freigegeben)	Es ist ein Fehler aufgetreten. Nach der Fehlerreaktion bleiben alle Achsen weiterhin in Regelung (Fehlerstufe 4).

Betriebszustand	Bedeutung
DisabledError (Fehler: gesperrt)	Es ist ein Fehler aufgetreten. Nach der Fehlerreaktion ist die Regelung aller Achsen gesperrt (Fehlerstufen 5 und 6).

Tab. 9: Betriebszustände von Bahnverbunden

Bewegungsart

Die Bewegungsart (`TypeOfMotion`) gibt an, welche Art der Bewegung der Sollwertgenerator für den Bahnverbund aktuell erzeugt.

Die Abfrage der Bewegungsart ist beispielsweise in einem Überwachungstask interessant, in dem der Prozess nicht bekannt ist. Ausgehend von der aktiven Bewegungsart vor dem Abschalten ist es möglich, das Wiederanlaufen korrekt zu steuern.

Betriebszustand	Bedeutung
Keine Bewegung	Diese Bewegungsart ist bis zur ersten Bewegung des Verbundes aktiv.
Gerade (`MoveLinear`)	Der Bahnverbund führt aktuell, oder führte zuletzt, eine Linearbewegung aus.
Kreisbewegung 2D (`MoveCircular2D`)	Der Bahnverbund führt aktuell, oder führte zuletzt, eine 2D-Kreisbewegung aus.
Kreisbewegung 3D (`MoveCircular3D`)	Der Bahnverbund führt aktuell, oder führte zuletzt, eine 3D-Kreisbewegung aus.
Spline (`MoveSpline`)	Der Bahnverbund führt aktuell, oder führte zuletzt, eine Spline-Bewegung aus.
Polynom (`MovePolynomial`)	Der Bahnverbund führt eine Überschleifbewegung aus.
Punkt zu Punkt (`MovePtp`)	Der Bahnverbund führt aktuell, oder führte zuletzt, eine Punkt-zu-Punkt-Bewegung aus.
Orientierung (`MoveOrientation`)	Der Bahnverbund führt aktuell, oder führte zuletzt, eine Orientierungsbewegung aus.
Gekoppelte Bewegung (`MoveExternal`)	Der Bahnverbund wird aktuell, oder wurde zuletzt, über eine gekoppelte Bewegung verfahren.

69.1 Syntax für MCGeo.State.Transitions

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.State.Transitions.<Funktion>(optional <Parameter>);
```

69.2 Funktionen von MCGeo.State.Transitions

Die Funktionalität bietet folgende Funktionen:

- Enable()
- Disable()
- Stop()
- ErrorDisable()
- ErrorStop()
- ErrorLockUserInput()

– Suspend()
– ErrorSuspend()
– Resume()

69.2.1 Enable()

Die Funktion `State.Transistions.Enable()` ist identisch zur Funktion `MCGeo.Power.Enable()`.Eine detaillierte Beschreibung der Funktion finden Sie im Abschnitt MCGeo.Power.Enable() [▶ 382].

Syntax

```
geoGroup.State.Transitions.Enable();
```

69.2.2 Disable()

Die Funktion führt je nach übergebenem Parameter (siehe nachfolgende Tabelle) dieselbe Aktion aus, wie die Befehle `MCGeo.Power.Disable()` und `MCGeo.Power.Qickstop()`. Entnehmen Sie die Details den Beschreibungen zu MCGeo.Power [▶ 382].

Übergabeparameter

Parameter	Datentyp	Bedeutung
Mode	MCStateTransitionDisableModes	Modus für die Ausführung der `Disable()`-Funktion
	MCStateTransitionDisableM-odes.ImmediatePulseBlocking	Führt die gleiche Funktion aus wie der Befehl `MC-Geo.Power.Disable()` bzw. `MCGeo MC-Geo.Power.Disable(MCPowerDisableM-odes.Forced)`
	MCStateTransitionDisableM-odes.VelocityControlSlope	Führt die gleiche Funktion aus wie der Befehl `MC-Geo.Power.QuickStop()`.

Syntax

```
geoGroup.State.Transitions.Disable(MCStateTransitionDisableModes.Immediate-
PulseBlocking);
geoGroup.State.Transitions.Disable(MCStateTransitionDisableModes.Velocity-
ControlSlope);
```

69.2.3 Stop()

Beschreibung

Die Funktion führt je nach dem übergebenen Parameter (siehe Tabelle unten) dieselbe Aktion aus wie der Befehl MCGeo.Stop() [▶ 411].

Der Unterschied besteht darin, dass die Übergabe eines Funktionsparameters mit `MCGeo.State.Transi-tion.Stop()` zwingend erforderlich ist, während dies mit `MCGeo.Stop()` optional ist (Details siehe dort).

Übergabeparameter

Parameter	Datentyp	Bedeutung
Mode	MCStateTransitionStopModes	Es stehen zwei Verzögerungswerte zur Verfügung.
	StopModes.ProgDeceleration	Zum Anhalten wird die zuletzt für den Verbund vorgegebene Verzögerung verwendet.
	StopModes.MaxDeceleration	Zum Anhalten wird die maximale Beschleunigung des Verbunds als Verzögerung verwendet. Diese wurde entweder bei der Inbetriebnahme im Motion Setup unter „Maximale Verzögerung“ vorgegeben oder mit dem Befehl `geoGroup.Mechanism.Limitations.MaxAcceleration` programmiert.

Syntax

```
geoGroup.State.Transitions.Stop(MCStateTransitionStopModes.ProgDecelerati-
on);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während der Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√		Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	√		Freigegeben
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Zustandsüberwachung

Für die Überwachung der Positionierung stehen Ihnen verschiedene Informationen zur Verfügung.

– Der **Zustand der Achse** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder die Achse angehalten ist.

```
When geoGroup.State.IsEnabled Continue; //Der Verbund befindet sich in
Lageregelung und steht.
When geoGroup.State.IsRunning Continue; //Der Sollwertgenerator mindestens
einer Verbundachse erzeugt eine Bewegung.
```

– Der **Rampenstatus** informiert, in welchem Bewegungsabschnitt sich der Sollwertgenerator gerade befindet.

```
When geoGroup.Mechanism.Slope.IsAccelerating Continue;
```

```
When geoGroup.Mechanism.Slope.IsAtConstantSpeed Continue;
When geoGroup.Mechanism.Slope.IsDecelerating Continue;
When geoGroup.Mechanism.Slope.IsStopped Continue;
```

– Der **Zielfensterstatus** wird von dieser Funktion nicht beeinflusst.

69.2.4 ErrorDisable()

Mit der Funktion lösen Sie einen Fehler im Bahnverbund aus, der zum Zustand **Fehler: gesperrt** führt. Die Fehlerreaktion kann eingestellt werden (siehe Eigenschaft `ErrorDisableMode`).

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.State.Transitions.ErrorDisable();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Fehler: gesperrt
Freigegeben	√	Fehler: gesperrt
Läuft	√	Fehler: gesperrt
Unterbrochen	√	Fehler: gesperrt
Fehler: freigegeben	√	Fehler: gesperrt
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

69.2.5 ErrorStop()

Die Funktion lösen Sie einen Fehler im Verbund aus, der zum Zustand **Fehler: freigegeben** führt. Die Fehlerreaktion kann eingestellt werden (siehe Eigenschaft ErrorStopMode [▶ 409]).

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.State.Transitions.ErrorStop();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Fehler: freigegeben
Freigegeben	√	Fehler: freigegeben

Ausgangszustand	Erlaubt	Endzustand
Läuft	√	Fehler: freigegeben
Unterbrochen	√	Fehler: freigegeben
Fehler: freigegeben	-	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

69.2.6 ErrorLockUserInput()

Mit der Funktion lösen Sie einen Bedienfehler im Verbund aus.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.State.Transitions.ErrorLockUserInput();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

69.2.7 Suspend()

Mit der Funktion unterbrechen Sie eine laufende Verbundbewegung. Das Anhalten geschieht entweder mit der zuletzt programmierten Verzögerungsrampe oder mit der konfigurierten Maximalverzögerung. Der Zustand nach dem Anhalten ist nicht **Freigegeben** sondern **Unterbrochen**. Mit der Funktion `Resume()` kann die unterbrochene Bahnbewegung eines Verbunds wiederaufgenommen und an die ursprüngliche Zielposition gefahren werden. Falls der Vorlaufpuffer noch nicht leer ist, werden die weiteren Bewegungen angehängt.

Die Rampeneinstellung (Beschleunigung und Verzögerung) kann während des Abfahrens der Bewegungen im Vorlaufpuffer nicht verändert werden, da die vorberechneten Bewegungen im Vorlaufpuffer abgelegt werden. Deshalb wird für die programmierte Verzögerung beim Befehl `Suspend()` der Wert verwendet, der auch für die Bahnbewegungen im Vorlaufpuffer Verwendung findet. Der Wert kann erst nach Ablauf der Bewegungen im Vorlaufpuffer geändert werden.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Mode	MCStateTransitionStopModes	Modus für die Verzögerungsrampe der `Suspend()`-Funktion
	MCStateTransitionStopModes.ProgDeceleration	Verwendet die für den Verbund zuletzt programmierte Verzögerung.
	MCStateTransitionStopModes.MaxDeceleration	Verwendet die für den Verbund konfigurierte maximale Verzögerung.

Syntax

```
geoGroup.State.Transitions.Suspend(MCStateTransitionStopModes.ProgDeceleration);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	X	Freigegeben
Läuft	√	Unterbrochen
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

69.2.8 ErrorSuspend()

Mit der Funktion lösen Sie einen Fehler im Bahnverbund aus, der zum Zustand **Fehler: freigegeben** führt. Die Bahnbewegung wird mit der programmierten Verzögerung angehalten. Der Vorlaufpuffer wird nicht gelöscht, so dass der Betriebszustand nach dem Löschen des Fehlers **Unterbrochen** ist und die Bewegung mit Resume() [▶ 408] fortgesetzt werden kann.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.State.Transitions.ErrorSuspend();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Fehler: freigegeben
Freigegeben	√	Fehler: freigegeben

Ausgangszustand	Erlaubt	Endzustand
Läuft	√	Fehler: freigegeben
Unterbrochen	√	Fehler: freigegeben
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

69.2.9 Resume()

Die Funktion führt eine mit der Funktion Suspend() [▶ 406] unterbrochene Positionierung fort. Es wird die ursprünglich vorgegebene Zielposition angefahren und weitere im Vorlaufpuffer befindliche Bewegungen werden angehängt. Auch die Geschwindigkeits- und Beschleunigungswerte werden aus der ursprünglichen Bahnbewegung übernommen.

Die Funktion `Resume()` wirkt sich nur aus, wenn sich der Verbund im Zustand **Unterbrochen** befindet.

Im gepufferten Betrieb wird der Vorlaufpuffer nicht gelöscht.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
geoGroup.State.Transitions.Resume();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	X	Freigegeben
Läuft	–	Läuft
Unterbrochen	√	Läuft
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

69.3 Eigenschaften von MCGeo.State.Transitions

Die Funktionalität bietet folgende Eigenschaften:

– ErrorDisableMode
– ErrorStopMode
– ErrorLevel

69.3.1 ErrorDisableMode

Beschreibung

Datentyp	Bedeutung
MCStateTransitionDisableModes	Fehlerreaktion, die beim Auslösen der Funktion `ErrorDisable()` verwendet wird.
MCStateTransitionDisableModes.VelocityControlSlope	Hält den Verbund mit der Schnellhaltverzögerung an. Ist der Stillstand erreicht, wird der Servoverstärker gesperrt.
MCStateTransitionDisableModes.ImmediatePulseBlocking	Sperrt den Servoverstärker sofort für die Drehmomenterzeugung (Impulssperre).

Syntax

```
geoGroup.State.Transitions.ErrorDisableMode := MCStateTransitionDisableModes.ImmediatePulseBlocking;
Fehlerreaktion := geoGroup.State.Transitions.ErrorDisableMode;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Bei Lesezugriff der Eigenschaft wird die aktuell gültige Fehlerreaktion für den Befehl zurückgegeben.

69.3.2 ErrorStopMode

Beschreibung

Datentyp	Bedeutung
MCStateTransitionErrorStopModes	Fehlerreaktion, die beim Auslösen der Funktion `ErrorStop()` verwendet wird.
MCStateTransitionErrorStopModes.ProgDeceleration	Hält den Verbund mit der programmierten Verzögerung an.
MCStateTransitionErrorStopModes.MaxDeceleration	Hält den Verbund mit der maximalen Verzögerung an.
MCStateTransitionErrorStopModes.AtBlockEnd	Hält den Verbund am Ende der momentanen Bewegung an. Die Abarbeitung des Vorlaufpuffers wird gestoppt.

Syntax

```
geoGroup.State.Transitions.ErrorStopMode := MCStateTransitionStopModes.MaxDeceleration;
Fehlerreaktion := geoGroup.State.Transitions.ErrorStopMode;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

69.3.3 ErrorLevel

Beschreibung

Mit der Eigenschaft stellen Sie die Fehlerstufe ein, die der Bahnverbund im Falle eines Drive-Fehlers einnimmt. Dies ist standardmäßig Fehlerstufe 5 (Anhalten mit Not-HALT-Rampe und Betriebszustand **Fehler: gesperrt**).

Datentyp	Bedeutung
MCStateTransitionErrorLevels	Fehlerstufe des Bahnverbundes bei einem Drive-Fehler. Ausführliche Informationen zu den Unterschieden der Fehlerreaktionsstufen finden Sie in der JetSym-Onlinehilfe unter der Rubrik *Fehlerverwaltung*.
MCStateTransitionErrorLevels.Message	Der Verbund reagiert mit der Fehlerstufe 1: Meldung/Warnung wird erstellt.
MCStateTransitionErrorLevels.Channel-Stop	Der Verbund reagiert mit der Fehlerstufe 2: Befehlskanal wird gesperrt.
MCStateTransitionErrorLevels.Cannel-StopClearBuffer	Der Verbund reagiert mit der Fehlerstufe 3: Befehlskanal wird gesperrt und Vorlaufpuffer geleert.
MCStateTransitionErrorLevels.Stop	Der Verbund reagiert mit der Fehlerstufe 4: Betriebs-HALT wird ausgeführt.
MCStateTransitionErrorLevels.Quickstop (Default)	Der Verbund reagiert mit der Fehlerstufe 5: Not-HALT wird ausgeführt.
MCStateTransitionErrorLevels.Puls-Blocking	Der Verbund reagiert mit der Fehlerstufe 6: Regeler wird gesperrt (Impulssperre).

Syntax

```
geoGroup.State.Transitions.ErrorLevel := MCStateTransitionErrorLevels.Puls-
Blocking;
ReaktionBeiDriveFehler := geoGroup.State.Transitions.ErrorLevel;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

70 MCGeo.Stop

Mit der Funktionalität **MCGeo.Stop** können Sie einen Bahnverbund, der sich in Bewegung befindet, mit einer definierten Rampe anhalten.

Im Unterschied zur Funktion `MoveHalt.Start()` wird bei `Stop` der Vorlaufpuffer geleert, d.h. alle noch im Puffer befindlichen Verfahrsätze werden gelöscht.

Die Funktionalität besitzt keine weiteren Funktionen oder Eigenschaften.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 50 – 52, 54 – 57** sowie die Dokumente *Bahnverbund*, *Bahnverbund – Synchrone Achse* und *Bahnverbund-Bahnzähler*. Diese stehen auf der Website von Bucher Automation zum Download bereit.

70.1 Syntax für MCGeo.Stop

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Stop(optional <Parameter 1>)
```

70.2 Funktionen von MCGeo.Stop

`MCTGeo.Stop` ist eine Funktion des Verbundobjekts.

70.2.1 Stop()

Übergabeparameter

Beachten Sie folgendes Verhalten:

– Die Funktion führt den Anhaltebefehl unmittelbar aus. Der übergebene Parameter (siehe Tabelle) gibt die maximale Verzögerung dafür an.

– Wird kein Parameter übergeben, so wird der zuletzt übergebene `StopMode` verwendet.

– Das Anhalten geschieht bahntreu.

Parameter	Datentyp	Bedeutung
StopModes	Enum StopModes (optional)	Es stehen zwei Verzögerungswerte zur Verfügung.
	StopModes.ProgDeceleration	Zum Anhalten wird die zuletzt für den Verbund vorgegebene Verzögerung verwendet (Standard).
	StopModes.MaxDeceleration	Zum Anhalten wird die maximale Beschleunigung der Achse als Verzögerung verwendet. Diese wurde entweder bei der Inbetriebnahme im Motion Setup unter Maximale Verzögerung vorgegeben oder mit dem Befehl `geoGroup.Mechanism.Limitations.MaxAcceleration` programmiert.

Syntax

```
geoGroup.Stop(StopModes.ProgDeceleration);
geoGroup.Stop(StopModes.MaxDeceleration);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während der Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√		Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	√		Freigegeben
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Positionierung stehen Ihnen verschiedene Informationen zur Verfügung.

– Der **Zustand der Achse** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder die Achse angehalten ist.

```
When geoGroup.State.IsEnabled Continue; //Der Verbund befindet sich in
Lageregelung und steht.
When geoGroup.State.IsRunning Continue; //Der Sollwertgenerator mindestens
einer Verbundachse erzeugt eine Bewegung.
```

– Der **Rampenstatus** informiert, in welchem Bewegungsabschnitt sich der Sollwertgenerator gerade befindet.

```
When geoGroup.Mechanism.Slope.IsAccelerating Continue;
When geoGroup.Mechanism.Slope.IsAtConstantSpeed Continue;
When geoGroup.Mechanism.Slope.IsDecelerating Continue;
When geoGroup.Mechanism.Slope.IsStopped Continue;
```

– Der **Zielfensterstatus** wird von dieser Funktion nicht beeinflusst.

71 MCManager (MCMgr)

Mit der Funktionalität **MCMgr** können Sie Informationen zu den im Hardwarebaum des Projektes definierten Achs- und Verbundobjekten lesen. Diese Informationen können z. B. im STX-Programm zu Programmierschleifen-Befehlen für alle oder einen Teil der Achsobjekte verwendet werden.

71.1 Syntax für MCMgr

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
MCMgr.<Funktion oder Zugriff auf Wert>(optional <Parameter 1, Parameter 2,
Parameter n>);
```

71.2 Funktionen von MCMgr

Die Funktionalität bietet folgende Funktionen:

- CheckVersions()
- GetDataSource()
- WriteLogToFile()
- GetAxisByID()
- GetAxisByName()
- GetTechnoByID()
- GetTechnoByName()
- GetGeoByID()
- GetGeoByName()

71.2.1 CheckVersions()

Mit der Funktion vergleichen Sie die Version des verwendeten MCX-Kernels mit der für die verwendete Motion-API-Version mindestens notwendigen MCX-Kernel-Version. Auf diese Weise können Sie prüfen, ob die Betriebssystemsoftware der Steuerung (mit dem enthaltenen MCX-Kernel) passend für die Funktionalitäten der Motion-API-Version ist.

Ist die MCX-Kernel-Version kleiner als die mindestens notwendige, wird `false` zurückgegeben. In diesem Fall wird im aktivierten Trace-Fenster von JetSym die mindestens notwendige OS-Version für die Steuerung ausgegeben.

Übergabeparameter

Diese Funktion besitzt keine Übergabeparameter.

Syntax

```
if Not MCMgr.CheckVersions() then
    Taskbreak tInit;
end_if;
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

71.2.2 GetDataSource()

Übergabeparameter

Parameter	Datentyp	Bedeutung
ShadowAxis	Pointer to MCAxis	Liest das Achsobjekt aus, das die Datenachse für die der Funktion übergebenen Mitlaufachse ist.

Syntax

```
DataAxis := MCMgr.GetDataSource(axShadow);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

71.2.3 WriteLogToFile()

Mit der Funktion schreiben Sie den Inhalt des Puffers des MCX-Loggers in eine Textdatei, die unter dem Namen *MCLogger.dat* im Dateisystem der JetControl abgelegt wird. Dies kann dazu dienen, die MCX-Aktionen während einer bestimmten Programmsequenz (z. B. im Fehlerfall) abzuspeichern, um sie später analysieren zu können. Die Datei enthält die Logger-Nachrichten in Zahlen codiert. Für die Daten analysieren zu können, müssen Sie die Datei von der Steuerung von den PC übertragen und in Text decodieren.

Datendatei auf den PC übertragen und decodieren

1. Übertragen Sie die Datei *MCLogger.dat* mithilfe eines Tool mit ftp-Kommunikation oder JetSym von der Steuerung auf den Rechner.
2. Öffnen Sie in JetSym den Reiter Hardware > MC-Global.
3. Wählen Sie Motion-Setup > Status.
4. Stellen Sie sicher, dass der Setup deaktiviert ist (Offline Betrieb).
5. Klicken Sie auf Protokolldatei öffnen und wählen Sie den Speicherort der Datei *MCLogger.dat* aus.
   ⇨ Es öffnet sich ein weiterer Dialog, der es Ihnen ermöglicht, die dekodierte Text-Datei auf dem PC abzuspeichern.
6. Folgen Sie den Anweisungen.

⇨ Die Logger-Nachrichten werden als Text im Status angezeigt.

Übergabeparameter

Parameter	Datentyp	Bedeutung
MaxLength	Integer (optional)	Definiert die Länge der zuletzt im Puffer aufgezeichneten Daten [in Byte], die in die Datei übertragen werden sollen. Wird kein Parameter oder `0` übergeben, wird der gesamte bisher aufgezeichnete Pufferspeicher in die Datei übernommen. Die Länge ist vom Steuerungstyp abhängig und kann bis zu 40 kByte groß sein. Die Übertragung wird im Hintergrund ausgeführt und kann einige Zeit in Anspruch nehmen.

Syntax

```
MCMgr.WriteLogToFile(255);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

71.2.4 GetAxisByID()

Die ObjectID der Achse wird von JetSym vergeben und muss in einem Projekt einmalig sein. In JetSym wird sie als Achs-ID im Tooltipp des Hardware-Baums und auf der Seite Achsparameter des Motion-Setups angezeigt.

Übergabeparameter

Parameter	Datentyp	Bedeutung
ObjectID	Integer	Liest das Achsobjekt (Rückgabewert vom Typ `Pointer to MCAxis`) aus, das mit der übergebenen ObjectID verbunden ist.

Syntax

```
MCAxis := MCMgr.GetAxisByID(8);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

71.2.5 GetAxisByName()

Der Achsname muss in einem Projekt einmalig sein. Er wird von JetSym beim Hinzufügen der Achse zum Hardware-Baum vergeben, kann jedoch angepasst werden.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Name	String	Liest das Achsobjekt (Rückgabewert vom Typ `Pointer to MCAxis`) aus, das mit dem übergebenen Namens-String verbunden ist.

Syntax

```
MCAxis := MCMgr.GetAxisByName(`AX1´);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

71.2.6 GetTechnoByID()

Die ObjectID des Technologieverbundes wird von JetSym vergeben und muss in einem Projekt einmalig sein. In JetSym wird sie als TechnoID im Tooltipp des Hardware-Baums und auf der Seite Eigenschaften des Motion-Setups angezeigt.

Übergabeparameter

Parameter	Datentyp	Bedeutung
ObjectID	Integer	Liest das Verbundobjekt (Rückgabewert vom Typ `Pointer to MCTechno`) aus, das mit der übergebenen ObjectID verbunden ist.

Syntax

```
MCTechVerbund := MCMgr.GetTechnoByID(5);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

71.2.7 GetTechnoByName()

Der Verbundname muss in einem Projekt einmalig sein. Er wird von JetSym beim Hinzufügen des Verbunds zum Hardware-Baum vergeben, kann jedoch angepasst werden.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Name	String	Liest das Verbundobjekt (Rückgabewert vom Typ `Pointer to MCTechno`) aus, das mit dem übergebenen Namens-String verbunden ist.

Syntax

```
MCTechno := MCMgr.GetTechnoByName(`AV1´);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

71.2.8 GetGeoByID()

Die ObjectID des Bahnverbundes wird von JetSym vergeben und muss in einem Projekt einmalig sein. In JetSym wird sie als GeoID im Tooltipp des Hardware-Baums und auf der Seite Eigenschaften des Motion-Setups angezeigt.

Übergabeparameter

Parameter	Datentyp	Bedeutung
ObjectID	Integer	Liest das Verbundobjekt eines Bahnverbunds (Rückgabewert vom Typ `Pointer to MCGeo`) aus, das mit der übergebenen ObjectID verbunden ist.

Syntax

```
MCGeoVerbund := MCMgr.GetGeoByID(2);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

71.2.9 GetTechnoByName()

Der Verbundname muss in einem Projekt einmalig sein. Er wird von JetSym beim Hinzufügen des Verbunds zum Hardware-Baum vergeben, kann jedoch angepasst werden.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Name	String	Liest das Verbundobjekt eines Bahnverbunds (Rückgabewert vom Typ `Pointer to MCTechno`) aus, das mit dem übergebenen Namens-String verbunden ist.

Syntax

```
MCGeo := MCMgr.GetGeoByName(`Bahn1´);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

71.3 Eigenschaften von MCMgr

Die Funktionalität bietet folgende Eigenschaften:

- AxisCount
- AxisList
- TechnoCount
- TechnoList
- GeoCount
- GeoList
- FaultyObjectCount
- FaultyObject
- Name
- Version
- IsInitialized

71.3.1 AxisCount

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Anzahl der Achsobjekte des Projekts aus. Dies kann vor allem bei Schleifen-Befehlen hilfreich sein, bei denen alle Achsen eines Projekts nacheinander die gleiche Funktion ausführen sollen.

Syntax

```
//diese Sequenz sperrt alle im Projekt definierten MCX-Achsen
for nAxisIndex := 0 to mcMgr.AxisCount - 1 by 1 do
    mcMgr.AxisList[nAxisIndex].Power.Disable();
end_for;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

71.3.2 AxisList[]

Beschreibung

Datentyp	Bedeutung
Array of pointer to MCAxis	Greift mit einem Index auf die MCX-Achsen des Projekts zu. Dies kann vor allem bei Schleifen-Befehlen hilfreich sein, bei denen alle Achsen eines Projekts nacheinander die gleiche Funktion ausführen sollen.

Syntax

```
//diese Sequenz sperrt alle im Projekt definierten MCX-Achsen
for nAxisIndex := 0 to mcMgr.AxisCount - 1 by 1 do
    mcMgr.AxisList[nAxisIndex].Power.Disable();
end_for;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

71.3.3 TechnoCount

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Anzahl der Technologieverbundobjekte des Projekts aus. Dies kann vor allem bei Schleifen-Befehlen hilfreich sein, bei denen alle Technologieverbunde eines Projekts nacheinander die gleiche Funktion ausführen sollen.

Syntax

```
//diese Sequenz aktiviert alle im Projekt definierten Technologieverbunde
for nTechnoIndex := 0 to mcMgr.TechnoCount - 1 by 1 do
    mcMgr.TechnoList[nTechnoIndex].Activate();
end_for;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

71.3.4 TechnoList[]

Beschreibung

Datentyp	Bedeutung
Array of pointer to MCTechno	Greift mit einem Index auf die Technologieverbunde des Projekts zu. Dies kann vor allem bei Schleifen-Befehlen hilfreich sein, bei denen alle Technologieverbunde eines Projekts nacheinander die gleiche Funktion ausführen sollen.

Syntax

```
//diese Sequenz aktiviert alle im Projekt definierten Technologieverbunde
for nTechnoIndex := 0 to mcMgr.TechnoCount - 1 by 1 do
    mcMgr.TechnoList[nTechnoIndex].Activate();
end_for;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

71.3.5 GeoCount

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Anzahl der Bahnverbundobjekte des Projekts aus. Dies kann vor allem bei Schleifen-Befehlen hilfreich sein, bei denen alle Bahnverbunde eines Projekts nacheinander die gleiche Funktion ausführen sollen.

Syntax

```
//diese Sequenz aktiviert alle im Projekt definierten Bahnverbunde
for nGeoIndex := 0 to mcMgr.GeoCount - 1 by 1 do
    mcMgr.GeoList[nGeoIndex].Activate();
end_for;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

71.3.6 GeoList[]

Beschreibung

Datentyp	Bedeutung
Array of pointer to MCTechno	Greift mit einem Index auf die Bahnverbunde des Projekts zu. Dies kann vor allem bei Schleifen-Befehlen hilfreich sein, bei denen alle Bahnverbunde eines Projekts nacheinander die gleiche Funktion ausführen sollen.

Syntax

```
//diese Sequenz aktiviert alle im Projekt definierten Bahnverbunde
for nGeoIndex := 0 to mcMgr.GeoCount - 1 by 1 do
    mcMgr.GeoList[nGeoIndex].Activate();
end_for;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

71.3.7 FaultyObjectCount

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Anzahl der MCX-Objekte des Projekts, die sich im Fehlerzustand befinden aus. Meldungen werden dabei nicht berücksichtigt. Anwendungsbeispiel siehe Eigenschaft FaultyObjectList[] [▶ 419].

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

71.3.8 FaultyObjectList[]

Beschreibung

Datentyp	Bedeutung
Array of pointer to MCTechno	List, welche MCX-Objekte des Projekts Fehler melden. Meldungen werden dabei nicht berücksichtigt. Mithilfe dieser Liste können Sie am ermittelten MCX-Objekt beispielsweise die Fehlernummer auslesen.

Syntax

```
//diese Sequenz prüft, ob Fehler in den Achsen, den Techno- oder Bahnverbunden vorliegen und ermittelt die Achsen- und Verbundobjekte, die Fehler anzeigen. Die Nummer des letzten Fehlers wird in einer Variablen abgespeichert.

var
    AxErrorNo : int;
    TecErrorNo : int;
```

```
    GeoErrorNo : int;
end_var;

Task tErrorRead

Var
    n, i : int;
    mcObj : IMCObject; //gemeinsame Schnittstelle für MCX-Objekte
    mcAx : pointer to MCAxis;
    mcTec : pointer to MCTechno;
    mcGe : pointer to MCGeo;
End_Var;


    AxErrorNO := 0;
    TecErrorNo := 0;
    GeoErrorNo := 0;

    n := mcMgr.FaultyObjectCount;

    if n > 0 then
        for i := 1 to n do
            mcObj := mcMgr.FaultyObject[i-1];

         Trace('Fehlerhaftes Objekt: ' + mcObj.Info.Name + '$n');

            // Prüfen, ob fehlerhaftes Objekt eine Achse ist
            if mcObj.Info.ClassID == MCClassIDs.Axis then
                // Achse mit betreffender Objekt-ID holen
                mcAx := mcMgr.GetAxisByID(mcObj.Info.ObjectID);
                // Fehlernummer auslesen
                AxErrorNo := mcAx.Diagnostics.GetErrorCode();
            end_if

            // Prüfen, ob fehlerhaftes Objekt ein Techno-Verbund ist
            if mcObj.Info.ClassID == MCClassIDs.Techno then
                // Techno-Objekt mit betreffender Objekt-ID holen
                mcTec := mcMgr.GetTechnoByID(mcObj.Info.ObjectID);
                // Fehlernummer auslesen
                TecErrorNo := mcTec.Diagnostics.Software.GetErrorCode();
            end_if;

            // Prüfen, ob fehlerhaftes Objekt ein Bahnverbund ist
            if mcObj.Info.ClassID == MCClassIDs.Geo then
                // Geo-Objekt mit betreffender Objekt-ID holen
                mcGe := mcMgr.GetGeoByID(mcObj.Info.ObjectID);
                // Fehlernummer auslesen
                GeoErrorNo := mcGe.Diagnostics.Software.GetErrorCode()
            end_if;

        end_for;
    end_if;

End_Task
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

71.3.9 Name

Beschreibung

Datentyp	Bedeutung
String	Liest den Typ der verwendeten MCX-Steuerung aus. Der Rückgabewert ist ein String (z. B. *JC-965EXT*).

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

71.3.10 Version

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Version des MCX-Kernels auf der Steuerung. Die Anzeige erfolgt im IP-Format.

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

71.3.11 IsInitialized

Beschreibung

Datentyp	Bedeutung	
Integer	Prüft, ob der MCX-Kernel beim Booten vollständig mit allen Achsen initialisiert wurde und betriebsbereit ist. `True`: der MCX-Kernel wurde beim Booten vollständig initialisiert und ist betriebsbereit `False`: Es wurden nicht alle MCX-Achsen am EtherCAT gefunden oder deren Initialisierung war fehlerhaft. Im MCX-Logger wird das durch folgende Zeilen angezeigt:	
	11:01:42:112	`Setup: Antriebe synchron`
	11:01:42:112	`Setup: MC V. 1.2.114.2 erfolgreich initialisiert und gestartet`

Syntax

```
When mcMgr.IsInitialized continue;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

72 MCTechno.Activate, MCTechno.Deactivate

Mit den Funktionalitäten **MCTechno.Activate** und **MCTechno.Deactivate** können Sie einen Technologieverbund aktivieren und deaktivieren. Änderungen an einem Technologieverbund sind in allen Betriebszuständen des Verbundes mit Ausnahme von **inaktiv** zulässig.

Sobald ein Technologieverbund aktiviert wurde, werden alle Mitgliedsachsen in den Betriebszustand **inaktiv** gesetzt. Auf diese Achsen können Sie nur noch über den Bahnverbund zugreifen, in den die Achsen eingebunden sind.

Ein Technologieverbund kann nur aktiviert und deaktiviert werden, wenn sich alle Mitgliedsachsen dieses Verbundes entweder im Betriebszustand **freigegeben** oder **gesperrt** befinden und alle Mitgliedsachsen den gleichen Betriebszustand haben. Andernfalls wird eine Fehlermeldung ausgegeben.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 49 - 56**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

72.1 Syntax für MCTechno.Activate, MCTechno.Deactivate

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Activate();
<Verbundbezeichnung>.Deactivate();
```

72.2 Funktionen von MCTechno.Activate, MCTechno.Deactivate

Die Funktionalität bietet folgende Funktionen:

– Activate()
– Deactivate()

72.2.1 Activate()

Mit der Funktion aktiveren Sie einen Technologieverbund sofern sich alle Mitgliedsachsen entweder im Zustand **Gesperrt** oder **Freigegeben** befinden.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
techGroup.Activate();
```

Zustandsverwaltung

Für das Aufrufen der Funktion gilt für die Leit- und Folgeachse folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Inaktiv	√	Gesperrt/Freigegeben
Gesperrt	–	Gesperrt
Freigegeben	–	Freigegeben

Ausgangszustand	Erlaubt	Endzustand
Läuft	–	Läuft
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Fragen Sie das Erreichen des neuen Zustands im Programm ab, bevor Sie weitere Befehle für die Achsen aufrufen.

```
When Not techGroup.State.IsInactive Continue;
```

Alternativ können Sie auch auf den Betriebszustand **IsEnabled** oder **IsDisabled** der Mitgliedsachsen abfragen – je nachdem, welchen Betriebszustand der Verbund vor dem Aufruf des Befehls hatte.

72.2.2 Deactivate()

Mit der Funktion deaktivieren Sie einen Technologieverbund. Nach diesem Befehl ist der Verbund im Zustand **inaktiv**. Die Mitgliedsachsen des Verbundes sind wieder unabhängig steuerbar. Sie übernehmen den Betriebszustand des Technologieverbundes.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
techGroup.Deactivate();
```

Zustandsverwaltung

Für das Aufrufen der Funktion gilt für die Leit- und Folgeachse folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Inaktiv	–	Inaktiv
Gesperrt	√	Inaktiv
Freigegeben	√	Inaktiv
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Fragen Sie das Erreichen des neuen Zustands im Programm ab, bevor Sie weitere Befehle für die Achsen aufrufen.

```
When techGroup.State.IsInactive Continue;
```

Alternativ können Sie auch auf den Betriebszustand **IsEnabled** oder **IsDisabled** der Mitgliedsachsen abfragen – je nachdem, welchen Betriebszustand der Verbund vor dem Aufruf des Befehls hatte.

73 MCTechno.Coupling

Mit der Funktionalität **MCTechno.Coupling** können Sie die Koppelfunktion des Technologieverbunds verwalten.

Eine detaillierte Beschreibung der der Funktionen, die der Eigenschaft `Cam` zugeordnet werden, in einem gesonderten Abschnitt (MCTechno.Coupling.Cam [▶ 438]) beschrieben.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 49 - 56**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

73.1 Syntax für MCTechno.Coupling

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Coupling.<Funktion oder Zugriff auf Wert>(optional <Parameter 1, Parameter 2, Parameter n>);
```

73.2 Funktionen von MCTechno.Coupling

Die Funktionalität bietet folgende Funktionen:

- Couple()
- Decouple
- GetCoupleMasterPosition()
- GetCoupleFollowerPosition()
- GetCoupleMasterDirection()
- GetCoupleMode()
- GetDecoupleDeceleration()
- GetDecoupleMasterPosition()
- GetDecoupleMode()
- IsCoupled()
- IsCoupling()
- IsDecoupled()
- IsDecoupling()
- Filter.GetCoefficient()
- Filter.SetCoefficient()
- Gearing.Configure()
- Gearing.GetFactor()
- Gearing.GetDivisor()
- Gearing.GetActivationMode()

73.2.1 Couple()

Mit der Funktion veranlassen Sie das Einkoppeln einer Folgeachse in einem Technologieverbund. Der Prozess setzt voraus, dass der Verbund aktiviert wurde. Je nach Einkoppelmodus wird das Einkoppeln entweder direkt ausgeführt, oder ein Prozess gestartet, der das Einkoppeln abhängig von verschieden Parametern der Leit- und Folgeachse vornimmt. Während dieses Prozesses ist der Koppelzustand *IsCoupling*. Am Ende des Prozesses ist der Koppelzustand *IsCoupled*.

Beim Ausführen der Funktion auf die Leitachse wird eine Meldung ausgegeben.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Folgeachsobjekt, das Teilnehmer des Technologieverbundes sein muss und das eingekoppelt werden soll.
CoupleMode	MCTechnoCoupleModes (optional)	Modus mit dem das Einkoppeln ausgeführt werden soll. Machen Sie sich mit den genauen Abläufe vertraut, die bei den vier Einkoppelmodi zum Einsatz kommen: – Für ein elektrisches Getriebes: Application Note 050 – Für eine Kurvenscheibe: Application Note 056
	MCTechnoCoupleModes Immediate	Die Kopplung wird direkt ausgeführt und ist sofort aktiv.
	MCTEechnoCouple-Modes ImmediateSnap	Die Kopplung wird direkt ausgeführt und ist sofort aktiv. – Bei einem **elektrischen Getriebe** ist das Verhalten identisch zum Einkoppelmodus *Immediate*. – Bei einer **Kurvenscheibe** wird die Kurvenscheibe so verschoben, dass der Sollwert zum Sollwert vor dem Einkoppeln passt.
	MCTechnoCoupleModes Fast	Die Kopplung erfolgt: – mit Beschleunigung über ein Einkoppelpolynom zur Einkoppelposition (siehe MasterPosition und FollwowerPosition); – gegebenenfalls mit Überhöhung der Einkoppelgeschwindigkeit;
	MCTechnoCoupleModes Wait	Die Kopplung erfolgt – mit Beschleunigung zur Einkoppelposition (siehe MasterPosition und FollwowerPosition); – gegebenenfalls mit Abwarten eines Moduloumbruchs, um die Überhöhung der Einkoppelgeschwindigkeit zu vermeiden;
MasterPosition	Double (optional)	Für die Einkoppelmodi *Fast* und *Wait*: Leitachsposition, an der das Einkoppeln beendet sein soll. Der Wert muss größer `0` sein.

Parameter	Datentyp	Bedeutung
MasterDirection	Directions (optional)	Für die Einkoppelmodi *Fast* und *Wait*: Bewegungsrichtung der Leitachse. Kein Einkoppeln bei entgegengesetzter Bewegungsrichtung.
	Directions.Positive	Die Leitachse bewegt sich in positive Richtung.
	Directions.Negativ	Die Leitachse bewegt sich in negative Richtung.
FollowerPosition	Double (optional)	Für die Einkoppelmodi *Fast* und *Wait*: Folgeachsposition, an der das Einkoppeln beendet sein soll.

Syntax

```
tecGroup.Coupling.Couple(Follower1,MCTechnoCoupleModes.Fast,1000.0,Directi-
ons.Positive,500.0);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Fehler: gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	√	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

73.2.2 Decouple()

Mit der Funktion veranlassen Sie das Auskoppeln einer Folgeachse in einem Technologieverbund. Der Prozess setzt voraus, dass der Verbund aktiviert wurde. Je nach Auskoppelmodus wird das Auskoppeln entweder direkt ausgeführt, oder ein Prozess gestartet, der das Auskoppeln abhängig von verschieden Parametern der Leit- und Folgeachse vornimmt. Während dieses Prozesses ist der Koppelzustand *IsDecoupling*. Am Ende des Prozesses ist der Koppelzustand *IsDecoupled*.

Beim Ausführen der Funktion auf die Leitachse wird eine Meldung ausgegeben.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Folgeachsobjekt, das Teilnehmer des Technologieverbundes sein muss und das ausgekoppelt werden soll.
DecoupleMode	MCTechnoDecouple-Modes (optional)	Modus mit dem das Auskoppeln ausgeführt werden soll. Machen Sie sich mit den genauen Abläufe vertraut, die bei den drei Auskoppelmodi zum Einsatz kommen: – Für ein elektrisches Getriebes: Application Note 050; – Für eine Kurvenscheibe: Application Note 056;
	MCTechnoDecouple-Modes Immediate_Stop	Das Auskoppeln wird direkt ausgeführt. Die Folgeachse wird mit der im Parameter `Deceleration` festgelegten Verzögerung angehalten.
	MCTEechnoDecouple-Modes_KeepVelocity	Das Auskoppeln wird direkt ausgeführt. Die Folgeachse wird mit der Geschwindigkeit, die sie beim Auskoppeln hatte, weiterbewegt analog zu zum `MoveVelocity()`-Befehl). Befindet sich die Leitachse in der Stopprampe, so behält die Folgeachse im Modus *Immediate_KeepVelocity* ihre Geschwindigkeit nicht bei, sondern wird wie im Modus *Immediate_Stop* mit spezifizierter Verzögerung angehalten.
	MCTechnoDecouple-Modes.AtMasterPosition_Stop	Die Folgeachse wird ausgekoppelt, sobald die Leitachse den Positionswert im Parameter `MasterPosition` erreicht hat. Danach wird sie mit der im Parameter `Deceleration` definierten Verzögerung angehalten.
MasterPosition	Double (optional)	Auskoppelposition beim Auskoppelmodus *AtMasterPosition_Stop*.
Deceleration	Double (optional)	Verzögerung zum Anhalten der Folgeachse nach dem Auskoppeln.

Syntax

```
tecGroup.Coupling.Decouple(Follower1, MCTechnoDecoupleModes.Immediate_Stop);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Fehler: gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen

Ausgangszustand	Erlaubt	Endzustand
Fehler: freigegeben	√	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

73.2.3 GetCoupleMasterPosition()

Mit der Funktion lesen Sie die zuletzt über die Funktion Couple vorgegebene Leitachsposition, an der das Einkoppeln beendet sein soll. (Nur aktiv bei den Modi *Fast* und *Wait*)

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest die Position des Leitachsobjekts des Technologieverbundes, an der das Einkoppeln beendet sein soll. (Nur aktiv bei den Modi *Fast* und *Wait*)

Syntax

```
LeitAchs_KoppelPos := tecGroup.Coupling.GetCoupleMasterPosition(axLeitach-
se);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

73.2.4 GetCoupleFollowerPosition()

Mit der Funktion lesen Sie die zuletzt über die Funktion Couple vorgegebene Folgeachsposition, an der das Einkoppeln beendet sein soll. (Nur aktiv bei den Modi *Fast* und *Wait*)

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest die Position des Folgeachsobjekts des Technologieverbundes, an der das Einkoppeln beendet sein soll. (Nur aktiv bei den Modi *Fast* und *Wait*)

Syntax

```
FolgeAchs_KoppelPos := tecGroup.Coupling.GetCoupleFollowerPosition(Fol-
lower1);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

73.2.5 GetCoupleMasterDirection()

Mit der Funktion lesen Sie die zuletzt über die Funktion Couple vorgegebene Leitachsbewegungsrichtung. (Nur aktiv bei den Modi *Fast* und *Wait*)

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest die Leitachsbewegungsrichtung des Leitachsobjekts des Technologieverbundes, dessen Fahrtrichtung ausgelesen werden soll. (Nur aktiv bei den Modi *Fast* und *Wait*)

Syntax

```
LeitAchs_Richtung := tecGroup.Coupling.GetCoupleMasterDirection(axLeitach-
se);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

73.2.6 GetDecoupleDeceleration()

Mit der Funktion lesen Sie den Wert der Verzögerung, der für das Auskoppeln bei den Auskoppelmodi *Immediate_Stop* und *AtMasterPosition_Stop* zur Anwendung kommt.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest den Wert für die Verzögerung des Folgeachsobjekts des Technologieverbunds, der für das Auskoppeln ausgelesen werden soll

Syntax

```
AuskoppelVerz := tecGroup.Coupling.GetDecoupleDeceleration(Follower1);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

73.2.7 GetDecoupleMasterPosition()

Mit der Funktion lesen Sie die zuletzt in der Funktion Decouple übergebene Auskoppelposition. (Nur beim Auskoppelmodus *AtMasterPosition_Stop*.)

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest die Position des Folgeachsobjekts des Technologieverbundes, dessen Leitachs-Auskoppelposition ausgelesen werden soll. (Nur bei Auskoppelmoduls *AtMasterPosition_Stop*)

Syntax

```
LeitAchs_AusKoppelPos := tecGroup.Coupling.GetDecoupleMasterPosition(Foll-
wer1);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

73.2.8 GetDecoupleMode()

Mit der Funktion lesen Sie den Modus, der für das Auskoppeln zur Anwendung kommt.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest den Auskoppelmodus, der für das Folgeachsobjekt des Technologieverbundes zur Anwendung kommt.

Syntax

```
AuskoppelModus := tecGroup.Coupling.GetDecoupleMode(Follower1);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

73.2.9 IsCoupling()

Diese Funktion entspricht der Eigenschaft `IsCoupling` der Folgeachse (MCAxis.State.Techno.IsCoupling [▶ 223]).

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest den Koppelzustand einer Folgeachse aus. `IsCoupling` ist `true`, wenn die Achse in den Koppelmodi *fast* und *wait* das Koppelkommando erhalten, den Einkoppelpunkt aber noch nicht erreicht hat.

Syntax

```
When tecGroup.Coupling.IsCoupling(Follower1) Continue;
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

73.2.10 IsCoupled()

Diese Funktion entspricht der Eigenschaft `IsCoupled` der Folgeachse (MCAxis.State.Techno.IsCoupled [▶ 223]).

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest den Koppelzustand einer Folgeachse aus. `IsCoupled` ist `true`, wenn die Achse eingekoppelt ist.

Syntax

```
When tecGroup.Coupling.IsCoupled(Follower1) Continue;
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

73.2.11 IsDecoupling()

Diese Funktion entspricht der Eigenschaft `IsDecoupling` der Folgeachse (MCAxis.State.Techno.IsDecoupling [▶ 222]).

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest den Koppelzustand einer Folgeachse aus. `IsDecoupling` ist `true`, wenn die Achse in dem Auskoppelmodus *AtMasterPosition_Stop* das Auskoppelkommando erhalten, den Einkoppelpunkt aber noch nicht erreicht hat.

Syntax

```
When tecGroup.Coupling.IsDecoupling(Follower1) Continue;
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

73.2.12 IsDecoupled()

Diese Funktion entspricht der Eigenschaft `IsDecoupled` der Folgeachse (MCAxis.State.Techno.IsDecoupled [▶ 222]).

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest den Koppelzustand einer Folgeachse aus. `IsDecoupled` ist `true`, wenn die Achse ausgekoppelt ist.

Syntax

```
When tecGroup.Coupling.IsDecoupled(Follower1) Continue;
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

73.2.13 Filter.GetCoefficient(), Filter.SetCoefficient()

`Get/SetCoefficient()` sind Unterfunktionen der Eigenschaft `Filter`. Mit den Funktionen können Sie den Wert für die Positionsfilterung in einem Technologieverbund lesen oder verändern. Als Parameter können eine Achse des Technologieverbundes und ein Filterkoeffizient mitgegeben werden.

Beachten Sie folgende Regeln:

– Bei Auswahl einer **Leitachse** wird eine Istwert-Filterung vorgenommen. Dies ist nur dann sinnvoll, wenn mindestens eine Folgeachse über Istwert-Kopplung angebunden ist.
– Bei Auswahl einer **Folgeachse** wird ein Filter im Sollwertpfad aktiviert.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Verbundsachse, deren Filtereinstellung verändert oder gelesen werden soll.

Parameter	Datentyp	Bedeutung
Coefficient	Double	– Filterkoeffizient; der Wert wird zwischen `0` … `100` vorgegeben. Dabei ist: `0`: keine Filterwirkung (Standard), `100`: höchste Filterwirkung;

Syntax

```
tecGroup.Coupling.Filter.SetCoefficient(Technoaxis,22.4);
FilterKoeff := tecGroup.Coupling.Filter.GetCoefficient(Technoaxis);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Fehler: gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

73.2.14 Gearing.Configure()

`Configure()` ist eine Unterfunktion der Eigenschaft `Gearing`. Mit der Funktion können Sie die Koppelparameter für eine Folgeachse, die als elektrisches Getriebe konfiguriert ist, parametrieren.

Über die Parameter `Factor` und `Divisor` stellen sie das Übersetzungsverhältnis zwischen den Positionen der Folgeachse und der Leitachse ein.

Beachten Sie folgende Regeln:

– Eine Folgeachse, die als **elektrisches Getriebe** konfiguriert ist, wird mit dem Aktivieren des Technologieverbundes direkt eingekoppelt.

  Ist dies für die Anwendung nicht sinnvoll, so kann mit dem Parameter `MCTechnoGearingActivationMode` das automatische Einkoppeln bei Aktivierung des Verbundes abgeschaltet werden. Dies setzt voraus, dass der Verbund aktiviert wurde. Danach ist das automatische Einkoppeln solange inaktiv, bis es wieder aktiviert wird.

– Wird der Befehl auf eine Folgeachse angewendet, die als **Kurvenscheibe** konfiguriert ist, wird keine Funktion ausgeführt und keine Fehlermeldung ausgegeben.

HINWEIS

Gefahr der Umparametrierung der Leitachsfunktion

Die Anwendung des Befehls auf eine Leitachse kann zur einer Umparametrierung der Leitachsfunktion führen.

► Achten Sie darauf, den Befehl nur auf Folgeachsen anzuwenden.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Folgeachsobjekt, das dem Technologieverbund angehören muss, und dessen Koppelparameter gelesen oder verändert werden sollen.
Factor	Double (optional)	Faktor der Getriebeübersetzung.
Divisor	Double (optional)	Divisor der Getriebeübersetzung: Muss > 0 sein.
ActivationMode	MCTechnoGearingActivationModes (optional)	Status der Kopplung bei Aktivierung des Technologieverbundes.
	MCTechnoGearingActivationModes.Coupled [Default]	Bei Aktivierung gekoppelt.
	MCTechnoGearingActivationModes.Decoupled	Bei Aktivierung ausgekoppelt.

Syntax

```
tecGroup.Coupling.Gearing.Configure(Follower1, 1.0, 2.0, MCTechnoGearingAc-
tivationModes.Decoupled);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Fehler: gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

73.2.15 Gearing.GetFactor()

`GetFactor()` ist eine Unterfunktion der Eigenschaft `Gearing`.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest den Faktor des Übersetzungsverhältnisses für eine Folgeachse in einem Technologieverbund aus, die als elektrisches Getriebe konfiguriert ist. Wird der Befehl auf eine Leitachse oder Folgeachse, die als Kurvenscheibe konfiguriert ist, angewendet, wird ein Wert ohne Bedeutung zurückgegeben und keine Fehlermeldung ausgegeben

Syntax

```
GetriebeFaktor := tecGroup.Coupling.Gearing.GetFactor(Follower1);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

73.2.16 Gearing.GetDivisor()

`GetDivisor()` ist eine Unterfunktion der Eigenschaft `Gearing`.Die Anwendung des Befehls auf eine Leitachse, oder auf Folgeachse, die als Kurvenscheibe konfiguriert ist, gibt einen Wert ohne Bedeutung zurück und keine Fehlermeldung aus.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest den Divisor des Übersetzungsverhältnisses für eine Folgeachse in einem Technologieverbund aus, die als elektrisches Getriebe konfiguriert ist.

Syntax

```
GetriebeDivisor := tecGroup.Coupling.Gearing.GetDivisor(Follower1);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

73.2.17 Gearing.GetActivationMode()

`GetActivationMode()` ist eine Unterfunktion der Eigenschaft `Gearing`. Die Anwendung des Befehls auf eine Leitachse, oder auf Folgeachse, die als Kurvenscheibe konfiguriert ist, gibt einen Wert ohne Bedeutung zurück und keine Fehlermeldung aus.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Liest den Koppelstatus beim Aktivieren des Technologieverbundes für eine Folgeachse aus, die als elektrisches Getriebe konfiguriert ist.

Syntax

```
GetriebeAktivierung := tecGroup.Coupling.Gearing.GetActiviationMode(Fol-
lower1);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

73.3 Eigenschaften von MCTechno.Coupling

Die Funktionalität bietet folgende Eigenschaften:

– Cam (siehe MCTechno.Coupling.Cam [▶ 438])
– ExternalMaster.Offset
– ExternalMaster.Smoothing
– Filter (siehe Beschreibung der Unterfunktionen unter Funktionen von MCTechno.Coupling [▶ 425])
– Gearing (siehe Beschreibung der Unterfunktionen unter Funktionen von MCTechno.Coupling [▶ 425])

73.3.1 ExternalMaster.Offset

Beschreibung

Diese Eigenschaft ist nur sinnvoll, wenn die Leitachse eines Technologieverbundes eine externe Achse ist.

Eine externe Achse zeichnet sich dadurch aus, dass sie nicht vom Sollwertgenerator des MCX-Kernels gesteuert werden kann, da es sich lediglich um die Position eines Positionsgebers handelt. Dieser Position kann über diese Eigenschaft ein Offset mitgegeben werden, bevor sie als Leitachsposition im Technologieverbund verwendet wird.

Dieser Befehl wird ignoriert, wenn die Leitachse des Technologieverbunds keine externe Achse ist.

Datentyp	Bedeutung
Double	Liest oder schreibt einen Positionsoffset für die Leitachsposition aus einer externen Achse.

Syntax

```
ExternalPosOffset := tecGroup.Coupling.ExternalMaster.Offset;
tecGroup.Coupling.ExternalMaster.Offset := 536,2;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

73.3.2 ExternalMaster.Smoothing

Beschreibung

Mit der Eigenschaft ExternalMaster.Smoothing lesen sie den Status oder setzen den Zustand einer Glättung der Leitachs-Positionswerte für die Externe Achse. Wenn die Eigenschaft aktiviert ist, wird die Position der Externen Achse über die letzten vier Eingangswerte geglättet um mögliche Schwankungen auszugleichen. Diese Funktion kann zusätzlich zur Istwertglättung verwendet werden.

Diese Eigenschaft ist nur sinnvoll, wenn die Leitachse eines Technologieverbundes eine externe Achse ist.

Dieser Befehl wird ignoriert, wenn die Leitachse des Technologieverbunds keine externe Achse ist.

Datentyp	Bedeutung
MCTechnoSmoothingModes	Liest den Status oder setzt den Zustand einer Glättung der Leitachs-Positionswerte für die externe Achse.
MCTechnoSmoothingModes.Disable	Es wird keine Glättung durchgeführt
MCTechnoSmoothingModes.Standard	Es wird über die letzten 4 Eingangswerte geglättet, um mögliche Schwankungen auszugleichen. Diese Funktion kann zusätzlich zur Istwert-Glättung verwendet werden.

Syntax

```
External_Glaettung := tecGroup.Coupling.ExternalMasterSmoothing;
tecGroup.Coupling.ExternalMaster.Smoothing := MCTechnoSmoothingModes.Standard;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

Für das Schreiben der Eigenschaft gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

74 MCTechno.Coupling.Cam

Mit der Funktionalität **MCTechno.Coupling.Cam** können Sie das Erstellen und Konfigurieren von Kurvenscheiben einer Folgeachse in einem Technologieverbund verwalte.

Eine detaillierte Beschreibung der übrigen Funktionen und Eigenschaften der Funktionalität finden Sie im zugehörigen Abschnitt MCTechno.Coupling [▶ 425].

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 49 - 56**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

APN 52 und APN 56 – Kurvenscheiben

Beachten Sie insbesondere die ausführlichen Informationen speziell zur Handhabung von Kurvenscheiben in den **Application Notes 052 und 056**.

74.1 Syntax für MCTechno.Coupling.Cam

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Coupling.Cam.<Funktion oder Zugriff auf Wert>(optional
<Parameter 1, Parameter 2, Parameter n>);
```

74.2 Funktionen von MCTechno.Coupling.Cam

Die Funktionalität bietet folgende Funktionen:

- Create()
- DefineSegment()
- DefineSegmentAuto()
- Activate()
- ReadSegment()
- Evaluate()
- AnalyseSegment()
- AnalyseSegmentDerivates()

74.2.1 Create()

Mit der Funktion stellen Sie eine Kurvenscheibe bereit. Dabei erhält die Kurvenscheibe eine Identifikation und in der Steuerung wird Speicherplatz für die Kurvenscheibe bereitgestellt. Diese Schritte sind erforderlich, um die Segmente einer Kurvenscheibe zu definiert, die Kurvenscheibe zu aktivieren und zu koppeln.

Wird die Funktion auf die Leitachse oder auf eine Folgeachse angewandt, die als elektrisches Getriebe konfiguriert ist, wird ein Fehler ausgegeben.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Folgeachsobjekt, das Teilnehmer des Technologieverbundes sein muss, und für das eine Kurvenscheibe angelegt werden soll.
CamId	Integer	Legt eine eindeutige Nummer zur Identifikation dieser Kurvenscheibe fest.

Parameter	Datentyp	Bedeutung
SegmentCount	Integer	Definiert die Anzahl der Segmente dieser Kurvenscheibe.

Syntax

```
tecGroup.Coupling.Cam.Create(Z_Axis,1,3);
```

Zustandsverwaltung

Sofern der Technologieverbund aktiv ist, ist das Aufrufen der Funktion in jedem Zustand möglich.

74.2.2 DefineSegment()

HINWEIS



Gefahr von unkontrollierten Achsbewegungen

Änderungen der Segmentdefinitionen werden sofort wirksam.

- ► Vermeiden Sie Änderungen der Segmentdefinition bei einer gerade aktiven Kurvenscheibe.
- ► Gehen Sie im unvermeidbaren Fall mit äußerster Vorsicht vor.

Mit der Funktion definieren Sie in Relation zur Leitachse die Bewegungsform, die für einen Teilbereich der Kurvenscheibe gelten soll. Die Segmentdefinition ist nur möglich, wenn die Kurvenscheibe zuvor erstellt wurde.

Wird die Funktion auf die Leitachse oder auf eine Folgeachse angewandt, die als elektrisches Getriebe konfiguriert ist, wird ein Fehler ausgegeben.

Unterschied zwischen DefineSegment() und DefineSegmentAuto()

`DefineSegment()`: die Definition der Polynomabschnitte wird über die Polynomkoeffizienten vorgenommen.

`DefineSegmentAuto()`: die Definition der Polynomabschnitte wird über Anfangs- und Endbedingungen vorgenommen.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Folgeachsobjekt, das Teilnehmer des Technologieverbundes sein muss, und für das ein Kurvenscheibensegment definiert werden soll.
CamId	Integer	Definiert die Kurvenscheiben-ID (`CamID`), der das Segment zugeordnet werden soll.
SegmentId	Integer	Eindeutige Nummer zur Indentifikation des Segments in der Kurvenscheibe, die mit der `CamId` ausgewählt wurde.

Parameter	Datentyp	Bedeutung
SegmentType	MCTechnoCamSegment-Types (optional)	Typ der Kurvenform dieses Segments in Bezug auf die Leitachsposition.
	MCTechnoCamSegment-Types.StraightLine	Kurvenform: linear (Polynom ersten Grades).
	MCTechnoCamSegment-Types.Poly3rd	Kurvenform: Polynom dritten Grades.
	MCTechnoCamSegment-Types.Poly5th	Kurvenform: Polynom fünften Grades.
	MCTechnoCamSegment-Types.Poly7th	Kurvenform: Polynom siebten Grades.
	MCTechnoCamSegment-Types.Sine	Kurvenform: Sinusfunktion.
SegmentStart	Double (optional)	Leitachsposition, ab der dieses Segment gültig ist.
Option	MCTechnoCamOptions (optional)	Optionen für die Ausführung des Segments.
	MCTechnoCamOpti-ons.NoOption	Die Definition des Segments wird so genommen wie angegeben.
	MCTechnoCamOpti-ons.AutoAppend	Diese Option ist erst mit ab dem zweiten Segment verfügbar. Sie soll die Definition dahingehend vereinfachen, dass der Folgeachs-Offset der Segmente nicht angegeben werden muss.
	MCTechnoCamOptions. UserDefinedMasterCam-Range	Diese Option ist nur im ersten Segment verfügbar. Hier wird für die komplette Kurvenscheibe definiert, dass der Leitachsbereich der Kurvenscheibe ungleich des Modulobereichs der Leitachse ist. Soll im Gebrauch die Kurvenscheibe in Leitachsrichtung gestreckt/gestaucht werden, so muss auch hier die Option `UserDefinedMasterRange` gewählt werden, auch wenn der angegebene `MasterCamRange` dem Modulo-Verfahrbereich der Leitachse entspricht.
Masteroffset	Double (optional)	Offset des Leitachswertes, der zur aktuellen Leitachsposition hinzugezählt wird.
FollowerOffset	Double (optional)	Offset des Folgeachswertes, der zum Funktionsergebnis hinzugezählt wird.
Coefficient0	Double (optional)	Koeffizienten der ausgewählten Funktion entsprechend des `SegmentType`. Es werden nicht alle Koeffizienten bei allen `SegmentTypes` verwendet. **HINWEIS! Beachten Sie hierzu die ausführlichen Informationen zu den Polynomformeln in der Application Note 052.**
Coefficient1	Double (optional)	
Coefficient2	Double (optional)	
Coefficient3	Double (optional)	
Coefficient4	Double (optional)	
Coefficient5	Double (optional)	
Coefficient6	Double (optional)	
Coefficient7	Double (optional)	
MasterCam-Range	Double (optional)	Im Fall der Kurvenscheibenoption `UserDefinedMaster-CamRange` muss hier der abweichende Leitachsbereich definiert werden.

Syntax

```
tecGroup.Coupling.Cam.DefineSegment(xFollower,1,1,MCTechnoCamSegmentTy-
pes.StraightLine, X1Offset, MCTechnoCamOptions.NoOption,-1.0 * X1Off-
set,Y1Offset,0.0,1.0)
```

Zustandsverwaltung

Sofern der Technologieverbund aktiv ist, ist das Aufrufen der Funktion in jedem Zustand möglich.

74.2.3 DefineSegmentAuto()

HINWEIS



Gefahr von unkontrollierten Achsbewegungen

Änderungen der Segmentdefinitionen werden sofort wirksam.

- ► Vermeiden Sie Änderungen der Segmentdefinition bei einer gerade aktiven Kurvenscheibe.
- ► Gehen Sie im unvermeidbaren Fall mit äußerster Vorsicht vor.

Mit der Funktion definieren Sie in Relation zur Leitachse die Bewegungsform, die für einen Teilbereich der Kurvenscheibe gelten soll. Die Segmentdefinition ist nur möglich, wenn die Kurvenscheibe zuvor erstellt wurde.

Wird die Funktion auf die Leitachse oder auf eine Folgeachse angewandt, die als elektrisches Getriebe konfiguriert ist, wird ein Fehler ausgegeben.

Unterschied zwischen DefineSegment() und DefineSegmentAuto()

`DefineSegment()`: die Definition der Polynomabschnitte wird über die Polynomkoeffizienten vorgenommen.

`DefineSegmentAuto()`: die Definition der Polynomabschnitte wird über Anfangs- und Endbedingungen vorgenommen.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Folgeachsobjekt, das Teilnehmer des Technologieverbundes sein muss, und für das ein Kurvenscheibensegment definiert werden soll.
CamId	Integer	Definiert die Kurvenscheiben-ID (`CamID`), der das Segment zugeordnet werden soll.
SegmentId	Integer	Eindeutige Nummer zur Indentifikation des Segments in der Kurvenscheibe, die mit der `CamId` ausgewählt wurde.

Parameter	Datentyp	Bedeutung
SegmentType	MCTechnoCamSegment-AutoTypes (optional)	Typ der Kurvenform dieses Segments in Bezug auf die Leitachsposition.
	MCTechnoCamSegment-AutoTypes.AutoPoly1st	Kurvenform: linear (Polynom ersten Grades).
	MCTechnoCamSegment-AutoTypes.AutoPoly3rd	Kurvenform: Polynom dritten Grades.
	MCTechnoCamSegment-AutoTypes.AutoPoly5th	Kurvenform: Polynom fünften Grades.
	MCTechnoCamSegment-AutoTypes.AutoPoly7th	Kurvenform: Polynom siebten Grades.
SegmentStart	Double (optional)	Leitachsposition, ab der dieses Segment gültig ist.
Option	MCTechnoCamOptions (optional)	Optionen für die Ausführung des Segments.
	MCTechnoCamOptions.NoOption	Die Definition des Segments wird so genommen wie angegeben.
	MCTechnoCamOptions.AutoAppend	Diese Option ist erst mit ab dem zweiten Segment verfügbar. Sie soll die Definition dahingehend vereinfachen, dass der Folgeachs-Offset der Segmente nicht angegeben werden muss.
	MCTechnoCamOptions.UserDefinedMasterCamRange	Diese Option ist nur im ersten Segment verfügbar. Hier wird für die komplette Kurvenscheibe definiert, dass der Leitachsbereich der Kurvenscheibe ungleich des Modulobereichs der Leitachse ist. Soll im Gebrauch die Kurvenscheibe in Leitachsrichtung gestreckt/gestaucht werden, so muss auch hier die Option `UserDefinedMasterRange` gewählt werden, auch wenn der angegebene `MasterCamRange` dem Modulo-Verfahrbereich der Leitachse entspricht.
FollowerOffset	Double (optional)	Offset des Folgeachswertes, der zum Funktionsergebnis hinzugezählt wird.
MasterRange	Double (optional)	Leitachsweg, den das Segment überstreicht.
FollowerRange	Double (optional)	Folgeachsweg, den das Segment überstreicht.
vStart	Double (optional)	Anfangs- und Endbedingungen für die erste, zweite und dritte Ableitung der Folgeachsposition bezüglich der Leitachsposition. Es werden nicht alle Parameter bei allen `SegmentTypes` verwendet. **HINWEIS! Beachten Sie hierzu die ausführlichen Informationen zu den Ableitungen in der Application Note 052.**
vEnd	Double (optional)	
aStart	Double (optional)	
aEnd	Double (optional)	
jStart	Double (optional)	
jEnd	Double (optional)	
MasterCamRange	Double (optional)	Im Fall der Kurvenscheibenoption `UserDefinedMasterCamRange` muss hier der abweichende Leitachsbereich definiert werden.

Syntax

```
tecGroup.Coupling.Cam.DefineSegmentAuto(xFollower,1,1,MCTechnoCamSegmentAutoTypes.AutoPoli1st,X1Offset,MCTechnoCamOptions.NoOption,Y1Offset,0.0,1.0)
```

Zustandsverwaltung

Sofern der Technologieverbund aktiv ist, ist das Aufrufen der Funktion in jedem Zustand möglich.

74.2.4 Activate()

Mithilfe der Funktion aktivieren Sie eine Kurvenscheibe in einem aktiven Technologieverbund. Das Aktivieren ist erst möglich, nachdem die Kurvenscheibe erstellt wurde. Das Aktivieren ist wiederum Voraussetzung für das Einkoppeln der Kurvenscheiben-Folgeachse.

Um von einer Kurvenscheibe zu einer anderen zu wechseln, aktivieren Sie die neue (bereits definierte) Kurvenscheibe mit einer Wechselbedingung. Die Folgeachse bleibt dabei eingekoppelt.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Folgeachsobjekt, das Teilnehmer des Technologieverbundes sein muss, und für das eine Kurvenscheibe aktiviert werden soll.
CamId	Integer	Definiert die Kurvenscheiben-ID (`CamID`) der zu aktivierenden Kurvenscheibe.
MasterOffset	Double (optional)	Definiert die Verschiebung der Kurvenscheibe in Leitachsrichtung.
FollowerOffset	Double (optional)	Definiert die Verschiebung der Kurvenscheibe in Folgeachsrichtung.
MasterFaktor	Double (optional)	Definiert die Skalierung der Kurvenscheibe in Leitachsrichtung.
FollowerFaktor	Double (optional)	Definiert die Skalierung der Kurvenscheibe in Folgeachsrichtung.
StartMode	MCTechnoCamStartModes (optional)	Definiert die Aktivierungsmethode.
	MCTechnoCamStartModes.Immediate	Die Kurvenscheibe wird sofort aktiv.
	MCTechnoCamStartModes.AtCamEnd	Die Kurvenscheibe wird aktiv, sobald das Ende der laufenden Kurvenscheibe erreicht ist.
	MCTechnoCamStartModes.AtMasterPosition	Die Kurvenscheibe wird aktiv, wenn die im Parameter `Position` definierte Leitachsposition erreicht wird.
	MCTechnoCamStartModes.AtFollowerPosition	Die Kurvenscheibe wird aktiv, wenn die im Parameter `Position` definierte Folgeachsposition erreicht wird.
Position	Double (optional)	Definiert die Leit- bzw Folgeachsposition, je nach `StartMode`.

Syntax

```
tecGroup.Coupling.Cam.Activate(xFollower,1,0.0,0.0,1.0,1.0,MCTechnoCamStartModes.Immediate,0.0);
```

Zustandsverwaltung

Sofern der Technologieverbund aktiv ist, ist das Aufrufen der Funktion in jedem Zustand möglich.

74.2.5 ReadSegment()

Mithilfe der Funktion lesen Sie die Daten eines Kurvenscheibensegments einer definierten Kurvenscheibe aus. Die Daten des Segments werden in einer Struktur vom Typ `CamSegmentData` zurückgegeben.

Intern arbeitet der MCX-Kernel stets mit Polynomkoeffizienten, auch wenn Auto-Polynome für die Definition des Segments verwendet wurden. Die Funktion `ReadSegment()` gibt im Fall von Auto-Polynomen deshalb die in Polynomkoeffizienten umgerechneten Werte zurück.

Wird die Funktion auf die Leitachse oder auf eine Folgeachse angewandt, die als elektrisches Getriebe konfiguriert ist, wird ein Fehler ausgegeben

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Folgeachsobjekt, das Teilnehmer des Technologieverbundes sein muss, und für das die Kurvenscheibendaten gelesen werden sollen.
CamId	Integer	Definiert die ID (`CamID`) der Kurvenscheibe, deren Segmentdaten gelesen werden sollen.
SegmentId	Integer	Definiert die ID des Segments, dessen Daten gelesen werden sollen.

Die Struktur vom Typ `CamSegmentData`, in der die Daten gespeichert werden, hat folgende Parameter:

Parameter	Datentyp	Bedeutung
SegmentType	MCTechnoCamSegmentTypes	Typ der Kurvenform dieses Segments in Bezug auf die Leitachsposition.
	MCTechnoCamSegmentTypes.StraightLine	Kurvenform: linear (Polynom ersten Grades).
	MCTechnoCamSegmentTypes.Poly3rd	Kurvenform: Polynom dritten Grades.
	MCTechnoCamSegmentTypes.Poly5th	Kurvenform: Polynom fünften Grades.
	MCTechnoCamSegmentTypes.Poly7th	Kurvenform: Polynom siebten Grades.
	MCTechnoCamSegmentTypes.Sine	Kurvenform: Sinusfunktion
	MCTechnoCamSegmentTypes.AutoPoly1th	Kurvenform: Auto-Polynom ersten Grades (linear)
	MCTechnoCamSegmentTypes.AutoPoly3rd	Kurvenform: Auto-Polynom dritten Grades.
	MCTechnoCamSegmentTypes.AutoPoly5th	Kurvenform: Auto-Polynom fünften Grades.
	MCTechnoCamSegmentTypes.AutoPoly7th	Kurvenform: Auto-Polynom siebten Grades.
SegmentStart	Double	Leitachsposition, ab der dieses Segment gültig ist.
Option	Int	Optionen für die Ausführung des Segments `0`: noOption `1`: AutoAppend `2`: UserDefinedMasterCamRange
Masteroffset	Double	Offset des Leitachswertes, der zur aktuellen Leitachsposition hinzugezählt wird.

Parameter	Datentyp	Bedeutung
FollowerOffset	Double	Offset des Folgeachswertes, der zum Funktionsergebnis hinzugezählt wird.
Coefficient	Array[0 .. 7] of Double	Koeffizienten der Funktion dieses Segments (Angabe immer in Polynomkoeffizienten).
MasterCam-Range	Double	Beim ersten Segment einer Kurvenscheibe wird hier der Leitachsbereich der Kurvenscheibe zurückgegeben. Bei jedem weiteren Segment einer Kurvenscheibe wird hier der Modulobereich der Leitachse zurückgegeben.

Syntax

```
SegmentData := tecGroup.Coupling.Cam.ReadSegment(xFollower,1,2);
```

Zustandsverwaltung

Sofern der Technologieverbund aktiv ist, ist das Aufrufen der Funktion in jedem Zustand möglich.

74.2.6 Evaluate()

Mithilfe der Funktion lesen Sie die Positions- und Segmentdaten einer Kurvenscheibe aus. Diese Funktion gibt Ihnen die Möglichkeit, die Folgeachsposition und das zugehörige Segment zu ermitteln, die bzw. das zu einer von Ihnen übergebenen Leitachsposition gehört.

Die Daten werden in einer Struktur vom Typ `CamEvaluateData` zurückgegeben.

Bei Verwendung der Option `AutoAppend` im Befehl `DefineSegment` oder `DefineSegmentAuto` lässt sich erst nach dem Starten der Kurvenscheibe ein richtiger Wert für die Folgeachsposition auslesen, da die Berechnung des Folgeachsoffsets erst erfolgt, nachdem die Kurvenscheibe aktiviert wurde.

Wird die Funktion auf die Leitachse oder auf eine Folgeachse angewandt, die als elektrisches Getriebe konfiguriert ist, wird ein Fehler ausgegeben

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Folgeachsobjekt, das Teilnehmer des Technologieverbundes sein muss, und für das die Kurvenscheibendaten gelesen werden sollen.
CamId	Integer	ID (`CamID`) der Kurvenscheibe, deren Positions- und Segmentdaten gelesen werden sollen.
MasterPosition	Double	Leitachsposition, zu der die korrespondierenden Daten ermittelt werden sollen.

Die Struktur vom Typ `CamEvaluateData`, in der die Daten gespeichert werden, hat folgende Parameter:

Parameter	Datentyp	Bedeutung
FollowerPosition	Double	Sollposition der Folgeachse, die von der Kurvenscheibe an der Stelle der übergebenen Leitachsposition berechnet wird.
SegmentNr	Integer	ID des Segments, in dem die Position ermittelt wurde.
MasterCamRange	Double	Leitachsbereich der Kurvenscheibe.

Syntax

```
CamPosData := tecGroup.Coupling.Cam.Evaluate(xFollower,1,1721.2);
```

Zustandsverwaltung

Sofern der Technologieverbund aktiv ist, ist das Aufrufen der Funktion in jedem Zustand möglich.

74.2.7 AnalyseSegment()

Mithilfe der Funktion lesen Sie die Daten eines Kurvenscheibensegments einer definierten Kurvenscheibe aus, deren Segmente mit dem Typ `AutoPoly` erstellt wurden. Der Funktionsaufruf gibt einen Fehler zurück, wenn die Kurvenscheibe mit Segmenten anderen Typs erstellt wurde.

Die Daten des Segments werden in einer Struktur vom Typ `CamAnalyseData` zurückgegeben.

Diese Funktion kann vor allem dazu verwendet werden, die Minimal- und Maximalwerte der Position (also Geschwindigkeit, Beschleunigung und Ruck) in einem Segment auszulesen. **HINWEIS! Beachten Sie, dass diese Positionen bezogen auf das Segment sind. Für die tatsächliche Minimal- oder Maximalposition muss der Folgeachsoffset hinzuaddiert werden.**

Wird die Funktion auf die Leitachse oder auf eine Folgeachse angewandt, die als elektrisches Getriebe konfiguriert ist, wird ein Fehler ausgegeben

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Folgeachsobjekt, das Teilnehmer des Technologieverbundes sein muss, und für das die Kurvenscheibendaten gelesen werden sollen.
CamId	Integer (optional)	ID (`CamID`) der Kurvenscheibe, deren Segmentdaten gelesen werden sollen.
SegmentId	Integer (optional)	ID des Segments, dessen Daten gelesen werden sollten.
Option	MCTechnoCamAnalysisOptions (optional)	
	MCTechnoCamAnalysisOptions.AsCreated	Gibt die Offsets in `CamAnalyseData.MasterOffest` und `CamAnalyseData.FollowerOffset` so aus, wie sie vom Anwender definiert wurden.
	MCTechnoCamAnalysisOptions.AsActivated	Gibt die Offsets in `CamAnalyseData.MasterOffest` und `CamAnalyseData.FollowerOffset` mit den zusätzlichen Aktivierungsoffsets aus, die die MCX beim Aktivieren der Kurvenscheibe berechnet.

Die Struktur vom Typ `CamAnalyseData`, in der die Daten gespeichert werden, hat folgende Parameter:

Parameter	Datentyp	Bedeutung
SegmentType	MCTechnoCamSegmentTypes	Typ der Kurvenform dieses Segments in Bezug auf die Leitachsposition.
	MCTechnoCamSegmentTypes.StraightLine	Kurvenform: linear (Polynom ersten Grades).
	MCTechnoCamSegmentTypes.Poly3rd	Kurvenform: Polynom dritten Grades.
	MCTechnoCamSegmentTypes.Poly5th	Kurvenform: Polynom fünften Grades.
	MCTechnoCamSegmentTypes.Poly7th	Kurvenform: Polynom siebten Grades.
	MCTechnoCamSegmentTypes.Sine	Kurvenform: Sinusfunktion
	MCTechnoCamSegmentTypes.AutoPoly1st	Kurvenform: Auto-Polynom ersten Grade (linear).
	MCTechnoCamSegmentTypes.AutoPoly3rd	Kurvenform: Auto-Polynom dritten Grades.
	MCTechnoCamSegmentTypes.AutoPoly5th	Kurvenform: Auto-Polynom fünften Grades.
	MCTechnoCamSegmentTypes.AutoPoly7th	Kurvenform: Auto-Polynom siebten Grades.
	-1	Es ist ein Fehler aufgetreten, z. B. weil das Segment nicht vom Typ `AutoPoly` definiert wurde. Alle anderen Rückgabewerte haben keine Bedeutung mehr.
SegmentStart	Double	Leitachsposition, ab der dieses Segment gültig ist.
Option	Integer	Optionen für die Ausführung des Segments: `0`: noOption `1`: AutoAppend `2`: UserDefinedMasterCamRange
MasterOffset	Double	Offset des Leitachswertes, der zur aktuellen Leitachsposition hinzugezählt wird.
FollowerOffset	Double	Offset des Folgeachswertes, der zum Funktionsergebnis hinzugezählt wird.
MinValue	Double	`[0]`: Minimalwert der 1. Ableitung der Position (Geschwindigkeit). `[1]`: Minimalwert der 2. Ableitung der Position (Beschleunigung). `[2]`: Minimalwert der 3. Ableitung der Position (Ruck).

Parameter	Datentyp	Bedeutung
MaxValue	Double	`[0]`: Maximalwert der 1. Ableitung der Position (Geschwindigkeit). `[1]`: Maximalwert der 2. Ableitung der Position (Beschleunigung). `[2]`: Maximalwert der 3. Ableitung der Position (Ruck).

Syntax

```
SegmentData := tecGroup.Coupling.Cam.AnalyseSegment(xFollower,1,2);
```

Zustandsverwaltung

Sofern der Technologieverbund aktiv ist, ist das Aufrufen der Funktion in jedem Zustand möglich.

74.2.8 AnalyseSegmentDerivates()

Mithilfe der Funktion lesen Sie lesen Sie die Daten eines Kurvenscheibensegments einer definierten Kurvenscheibe aus, deren Segmente mit dem Typ `AutoPoly` erstellt worden sind. Der Funktionsaufruf gibt einen Fehler zurück, wenn die Kurvenscheibe mit Segmenten anderen Typs erstellt wurde.

Die Daten des Segments werden in einer Struktur vom Typ `CamAnalyseDerivatesData` zurückgegeben.

Diese Funktion kann vor allem dazu verwendet werden, um die Minimal- und Maximalwerte der Ableitungen der Position (also Geschwindigkeit, Beschleunigung und Ruck in Bezug auf die Leitachsgeschwindigkeit) in einem Segment auszulesen.

Wird die Funktion auf die Leitachse oder auf eine Folgeachse angewandt, die als elektrisches Getriebe konfiguriert ist, wird ein Fehler ausgegeben

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Folgeachsobjekt, das Teilnehmer des Technologieverbundes sein muss, und für das die Kurvenscheibendaten gelesen werden sollen.
CamId	Integer (optional)	ID der Kurvenscheibe, deren Segmentdaten gelesen werden sollen.
SegmentID	Inteter (optional)	ID des Segments, dessen Daten gelesen werden sollten.

Die Struktur vom Typ `CamAnalyseDerivatesData`, in der die Daten gespeichert werden, hat folgende Parameter:

Parameter	Datentyp	Bedeutung
SegmentType	MCTechnoCamSegmentTypes	Typ der Kurvenform dieses Segments in Bezug auf die Leitachsposition.
	MCTechnoCamSegmentTypes.StraightLine	Kurvenform: linear (Polynom ersten Grades).
	MCTechnoCamSegmentTypes.Poly3rd	Kurvenform: Polynom dritten Grades.
	MCTechnoCamSegmentTypes.Poly5th	Kurvenform: Polynom fünften Grades.
	MCTechnoCamSegmentTypes.Poly7th	Kurvenform: Polynom siebten Grades.
	MCTechnoCamSegmentTypes.Sine	Kurvenform: Sinusfunktion
	MCTechnoCamSegmentTypes.AutoPoly1th	Kurvenform: Auto-Polynom ersten Grade (linear).
	MCTechnoCamSegmentTypes.AutoPoly3rd	Kurvenform: Auto-Polynom dritten Grades.
	MCTechnoCamSegmentTypes.AutoPoly5th	Kurvenform: Auto-Polynom fünften Grades.
	MCTechnoCamSegmentTypes.AutoPoly7th	Kurvenform: Auto-Polynom siebten Grades.
	-1	Es ist ein Fehler aufgetreten, z. B. weil das Segment nicht vom Typ `AutoPoly` definiert wurde. Alle anderen Rückgabewerte haben keine Bedeutung mehr.
SegmentStart	Double	Leitachsposition, ab der dieses Segment gültig ist.
Option	Integer	Optionen für die Ausführung des Segments: `0`: noOption `1`: AutoAppend `2`: UserDefinedMasterCamRange
MasterOffset	Double	Offset des Leitachswertes, der zur aktuellen Leitachsposition hinzugezählt wird.
FollowerOffset	Double	Offset des Folgeachswertes, der zum Funktionsergebnis hinzugezählt wird.
MinValue	Array[0 .. 2] of Double	`[0]`: Minimalwert der 1. Ableitung der Position (Geschwindigkeit). `[1]`: Minimalwert der 2. Ableitung der Position (Beschleunigung). `[2]`: Minimalwert der 3. Ableitung der Position (Ruck).

Parameter	Datentyp	Bedeutung
MaxValue	Array[0 .. 2] of Double	`[0]`: Maximalwert der 1. Ableitung der Position (Geschwindigkeit). `[1]`: Maximalwert der 2. Ableitung der Position (Beschleunigung). `[2]`: Maximalwert der 3. Ableitung der Position (Ruck).
MasterCam-Range	Double	Leitachsbereich der Kurvenscheibe.

Syntax

```
SegmentData := tecGroup.Coupling.Cam.AnalyseSegmentDerivates(xFollower,1,2);
```

Zustandsverwaltung

Sofern der Technologieverbund aktiv ist, ist das Aufrufen der Funktion in jedem Zustand möglich.

75 MCTechno.Diagnostics

Mit der Funktionalität **MCTechno.Diagnostics** können Sie die Fehlerbehandlung eines Technologieverbundes verwalten.

Im Motion-Control-eXtended-System werden Fehler (*errors*) und Meldungen (*messages*) unterschieden. Fehler rufen eine Systemreaktion hervor, Meldungen hingegen nicht. Fehler und Meldungen werden in getrennten Puffern protokolliert, von wo aus sie ausgelesen werden können. Der jeweilige Puffer sammelt die Ereignisse, die seit dem letzten Aufruf der Funktion `ClearErrors`() aufgetreten sind (siehe ClearErrors() [▶ 39]).

Die Bedeutung der Fehler- und Meldungsnummern ist in der JetSym -Online-Hilfe unter „MotionControl-Fehler“ aufgeführt.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 49 - 56**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

75.1 Syntax für MCTechno.Diagnostics

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundsbezeichnung>.Diagnostics.<Funktion oder Zugriff auf Wert>(optional
<Parameter>);
```

75.2 Funktionen von MCTechno.Diagnostics

Die Funktionalität bietet folgende Funktionen:

- ClearErrors()
- Software.ClearErrors()
- Software.GetErrorCode()
- Software.GetMessageCode()

75.2.1 ClearErrors(), Software.ClearErrors() – Technologieverbund

Bei Fehlern quittieren die Funktionen einen Fehlerzustand im Technologieverbund. Die Einträge im Fehlerpuffer werden gelöscht.

Meldungen haben keine Fehlerreaktion zur Folge, und müssen deshalb nicht quittiert werden. Bei Meldungen bewirken die beiden Funktionen, dass die Einträge im Meldungspuffer gelöscht werden.

Übergabeparameter

Diese Funktionen besitzen keine Übergabeparameter.

Syntax

```
tecGroup.Diagnostics.ClearErrors();
tecGroup.Diagnostics.Software.ClearErrors();
```

ⓘ Info

Geltungsbereich des Befehls

Mit diesen Befehlen werden alle Fehler gelöscht, die vom Verbund zugeordnet sind. Das können Fehler sein, die von einem Verbundobjekt ausgelöst wurden, ebenso wie Fehler, die von den Verbundachsen gemeldet wurden.

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand	
Gesperrt	–	Gesperrt	
Freigegeben	–	Freigegeben	
Läuft	–	Läuft	
Unterbrochen	–	Unterbrochen	
Fehler: freigegeben	√	Freigegeben	In diesen Zustand wechselt das System nur, wenn die Fehlerursache behoben ist. Andernfalls bleibt u. U. auch der Fehlerzustand erhalten oder wird sofort wieder erreicht.
Fehler: gesperrt	√	Gesperrt	

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

75.2.2 Software.GetErrorCode()

Die Funktion liefert die Nummer des Fehlers an der mit `BufferIndex` angegebenen Fehlerpufferstelle zurück. Wird kein `BufferIndex` übergeben, wird die Nummer des zuletzt aufgezeichneten Fehlers zurückgegeben (`BufferIndex` = `0`).

Übergabeparameter

Parameter	Datentyp	Bedeutung
BufferIndex	Integer (optional)	Der `BufferIndex` übergibt die Stelle im Fehlerpuffer, deren Fehlernummer ausgelesen werden soll. Der Maximalwert des Parameters `BufferIndex` kann über die Eigenschaft `ErrorCount` ausgelesen werden.

Syntax

```
tecGroup.Diagnostics.Software.GetErrorCode();
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

75.2.3 Software.GetMessageCode()

Die Funktion liefert die Nummer der Meldung an der mit `BufferIndex` angegebene Meldungspufferstelle zurück. Wird kein `BufferIndex` übergeben, wird die Nummer der zuletzt aufgezeichneten Meldung zurückgegeben (`BufferIndex` = `0`).

Übergabeparameter

Parameter	Datentyp	Bedeutung
BufferIndex	Integer (optional)	Der `BufferIndex` übergibt die Stelle im Meldungspuffer, deren Meldungsnummer ausgelesen werden soll. Der Maximalwert des Parameters `BufferIndex` kann über die Eigenschaft `MessageCount` ausgelesen werden.

Syntax

```
tecGroup.Diagnostics.Software.GetMessageCode();
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

75.3 Eigenschaften von MCTechno.Diagnostics

Die Funktionalität bietet folgende Eigenschaften:

- Software.ErrorCount
- IsErrorPending
- Software.IsErrorPending
- Software.MessageCount
- IsMessagePending
- Software.IsMessagePending

75.3.1 Software.ErrorCount

Beschreibung

Eigenschaft	Datentyp	Bedeutung
Software.ErrorCount	Int	Liest die Anzahl der erfassten Fehler seit dem letzten Ausführen der Funktion `ClearErrors()` bzw. `Software.Clear.Errors()` oder seit dem Neustart der Steuerung aus.

Syntax

```
FehlerAnzahl := tecGroup.Diagnostics.Software.ErrorCount;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

75.3.2 IsErrorPending, Software.IsErrorPending

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob ein Fehler im MCX-System vorliegt. Dies ist gleichbedeutend mit einem Vergleich von `ErrorCount` mit 0.

Syntax

```
If tecGroup.Diagnostics.IsErrorPending then
...
End_if;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

75.3.3 Software.MessageCount

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Anzahl der erfassten Meldungen seit dem letzten Ausführen der Funktion `ClearErrors()` bzw. `Software.ClearErrors()`, oder seit dem Neustart der Steuerung aus.

Syntax

```
MeldungsAnzahl := tecGroup.Diagnostics.Software.MessageCount;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

75.3.4 IsMessagePending, Software.IsMessagePending

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob eine Meldung im MCX-System vorliegt. Dies ist gleichbedeutend mit einem Vergleich von `MessageCount` mit `0`.

Syntax

```
If tecGroup.Diagnostics.IsMessagePending then
...
End_if;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

75.4 Beispieltask für MCTechno.Diagnostics

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Var
    TecError: Array[10] of INT; //Variablen zum Anzeigen des Fehlerpuffers
    i: INT; //Laufvariable
    AcknolegeError: Bool; // Hilfs-Flag für eine Quittier-Bedingung

End_Var;

Task Sample_Error Autorun

Loop

    //Fehlervariablen zum Anzeigen löschen
    For i := 0 To 9 Do
        TecError[i] := 0;
    End_For;
    AcknowledgeError := False;

    When tecGroup.Diagnostics.IsErrorPending Continue;

        Taskbreak Automatic;
        // hier weitere Fehlerbehandlung einfügen

        //Fehler in Variablen z. B. zum Anzeigen ablegen
        For i := 0 To tecGroup.Diagnostics.Software.ErrorCount-1 Do
            TecError[i] := tecGroup.Diagnostics.Software.GetErrorCode(i);
        End_For;

    // Fehler sind behoben
    When AcknowledgeError Continue;

        tecGroup.Diagnostics.ClearErrors();
        Taskrestart Automatic;
End_Loop;

End_Task;
```

76 MCTechno.Info

Mit der Funktionalität **MCTechno.Info** können Sie die Verbundobjekte der MotionControl identifizieren.

Es handelt sich um Informationen, die vom Anwender selbst im Hardware-Manager und Motion-Setup im Projekt festgelegt wurden.

76.1 Syntax für MCTechno.Info

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Technologieverbundbezeichnung>.Info.<Eigenschaft>;
```

76.2 Eigenschaften der Funktionalität

Die Funktionalität bietet folgende Eigenschaften:

- Name
- ObjectID
- ClassID

76.2.1 Name

Beschreibung

Datentyp	Bedeutung
String	Fragt den in der Hardware-Konfiguration definierten Namen des Technologieverbundes ab. Dies ist hilfreich, wenn Sie den Namen im Trace oder auf einer externen Anzeige verwenden wollen.

Syntax

```
TechnoName := Techno.Info.Name;
```

Dabei enthält `TechnoName` den String `Techno`.

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

76.2.2 ObjectID

Beschreibung

Datentyp	Bedeutung
Integer	Fragt die steuerungsinterne Nummer des Technologieverbundes ab. Die Eigenschaft wird normalerweise nur im Zusammenhang mit Anwendungen benötigt, die die Funktionen des MC-Managers nutzen. Eine Anwendungsmöglichkeit ist im zugehörigen Abschnitt MCManager (MCMgr) [▶ 413] beschrieben.

Syntax

```
Verbundkennung := TechnoGroup.Info.ObjectID;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

76.2.3 ClassID

Beschreibung

Datentyp	Bedeutung
MCClassIDs	Fragt die Klasse eines MC-Objekts ab. Ein MC-Objekt kann eine Achse, ein Technologieverbund oder ein Bahnverbund sein. Die Eigenschaft wird normalerweise nur im Zusammenhang mit Anwendungen benötigt, die die Funktionen des MC-Managers nutzen. Eine Anwendungsmöglichkeit ist im zugehörigen Abschnitt MCManager (MCMgr) [▶ 413] beschreiben.

Syntax

```
KlassenID := tecGroup.Info.ClassID;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

76.3 Beispieltask MCTechno.Info

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

Nach Ausführung des Beispieltasks wird im Trace für alle Technologieverbunde angezeigt, ob sie im Zustand **läuft** sind oder nicht. Die Verbunde werden mit ihrem Verbundnamen aufgeführt.

```
function TraceRunningState(mcTechnoGroup: pointer to MCTechno)
    if mcTechnoGroup.State.IsRunning then
        trace(mcTechnoGroup.Info.Name + 'is running$n');
    else
        trace(mcTechnoGroup.Info.Name + 'is not running$n');
    end_if;
end_function;

task Sample1 // Beispieltask
var
    nTechnoIndex:    int;
end_var;
    for nTechnoIndex := 0 to mcMgr.TechnoCount - 1 by 1 do
        TraceRunningState(mcMgr.TechnoList[nTechnoIndex]);
    end_for;
end_task;
```

77 MCTechno.Mechanism.Slope

Mit der Funktionalität **MCTechno.Mechanism.Slope** können Sie die berechneten Eigenschaften der laufenden oder nächsten Bewegung lesen oder anpassen.

Eine Positionierbewegung besteht aus folgenden Phasen:

1. Beschleunigungsrampe
2. Fahrt mit konstanter Geschwindigkeit

   Diese Phase tritt nur ein, wenn die Geschwindigkeit `Speed` mit den vorgegebenen MovePtp-Parametern erreicht werden kann. Andernfalls geht die Beschleunigungsrampe direkt in die Verzögerungsrampe über (Dreiecksbetrieb).
3. Verzögerungsrampe

Daneben kann auch keine Bewegung stattfinden, wenn die Achse im Ziel steht und noch keine weitere Positionierung gestartet wurde.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 49 - 56**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

77.1 Syntax für MCTechno.Mechanism.Slope

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Mechanism.Slope.<Zugriff auf Wert>
```

77.2 Funktionen von MCTechno.Mechanism.Slope

Die Funktionalität bietet folgende Funktionen:

– GetSlopeType()
– SetSlopeType()
– GetSpeed()
– SetSpeed()
– GetAcceleration()
– SetAcceleration()
– GetDeceleration()
– SetDeceleration()

77.2.1 GetSlopeType(), SetSlopeType()

Mit den Funktionen lesen oder verändern Sie die Rampenform in der Beschleunigungs- und der Verzögerungsphase der im ersten Parameter aufgeführten Verbundachse. **HINWEIS! Diese Einstellung gilt für alle Positionierbewegungen der MCX-Achse so lange, bis der Wert geändert wird.**

Die Funktion `GetSlopeType()` liefert einen Wert des Typs `MCTechnoSlopeTypes` zurück. Ihr muss nur der Pointer auf ein Achsobjekt, das Teilnehmer im Technologieverbund ist, übergeben werden.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	pointer to MCAxis	Achsobjekt (das Teilnehmer des Technologieverbundes sein muss) dessen Rampenform gelesen oder verändert werden soll
SlopeType	MCTechnoSlopeTypes	Form der Beschleunigungs- und Verzögerungsrampe.
	AxisSlopeTypes.Sine (Standard)	Diese Einstellung berechnet eine Bewegung, die verhältnismäßig wenig Ruck erzeugt, d. h. die Bewegung ist weicher. Da der dem Bewegungsbefehl übergebene Beschleunigungswert nur in der Mitte der Rampe erreicht wird, sind Rampendauer und -länge höher als bei der Einstellung linear. Die Bewegung verläuft in der Beschleunigungsphase sinus-förmig und in der Verzögerungsphase sin²-förmig. Die Bewegung wird ruckbegrenzt.
	AxisSlopeTypes.Linear	Diese Einstellung ermöglicht eine Bewegung mit möglichst geringer maximaler Beschleunigung bei gleicher Rampenzeit (und -länge) im Verhältnis zur Einstellung `sine`. Dafür ist der Ruck beim Übergang vom Stillstand in die Beschleunigungsphase und von der Beschleunigungsphase zur Phase mit maximaler Geschwindigkeit unendlich. Die Bewegung verläuft in der Beschleunigungsphase konstant und in der Verzögerungsphase sinus-förmig. Die Bewegung verläuft ohne Ruckbegrenzung.

Syntax

```
Rampentyp := tecGroup.Mechanism.Slope.GetSlopeType(axLeitachse);
tecGroup.Mechanism.Slope.SetSlopeType(axLeitachse,MCTechnoSlopeTypes.Linear);
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

Wird die Rampenform mit `SetSlopeType()` während einer Positionierung (Zustand **Läuft**) geändert, wirkt sich diese Änderung erst bei der nächsten Positionierung aus.

77.2.2 GetSpeed(), SetSpeed()

Beschreibung

Liest oder schreibt den Wert für die maximale Geschwindigkeit, der für die Positionssollwertgenerierung eines Positionierprofils verwendet wird. Dieser Wert gilt für die im ersten Parameter aufgeführten Verbundachse.

Diese Einstellung gilt für alle Positionierbewegungen der Verbundachse so lange, bis der Wert geändert wird.

Dieser Wert wird normalerweise in den Funktionen `MovePtp.Start()` und `Move.Velocity.Start()` oder auch `MovePtp.NewSpeed()` und `MoveVelocity.NewSpeed()` als Parameter übergeben, wo eine Änderung auf eine laufende Bewegung angewendet wird.

Anders hier: Gleich in welchem Zustand die Änderung des Wertes für `Speed` übergeben wird, der neue Wert wirkt sich erst in der nächsten Bewegung aus.

Parameter	Datentyp	Bedeutung
Axis	pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss und dessen Profilgeschwindigkeit gelesen oder verändert werden soll.
Speed	Double	Maximale Verzögerung im Positionierprofil.

Syntax

```
ProfilGeschw := tecGroup.Mechanism.Slope.GetSpeed(axLeitachse);
tecGroup.Mechanism.Slope.SetSpeed(axLeitachse, 3000.0);
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

Wird die Profilgeschwindigkeit mit `SetSpeed()` während einer Positionierung (Zustand **Läuft**) geändert, wirkt sich diese Änderung erst bei der nächsten Positionierung aus.

77.2.3 GetAcceleration(), SetAcceleration()

Beschreibung

Liest oder schreibt den Wert für die maximale Beschleunigung, der für die Positionssollwertgenerierung eines Positionierprofils verwendet wird. Dieser Wert gilt für die im ersten Parameter aufgeführte Verbundachse.

Diese Einstellung gilt für alle Positionierbewegungen der Verbundachse so lange, bis der Wert geändert wird.

Dieser Wert wird normalerweise in den Funktionen `MovePtp.Start()` und `Move.Velocity.Start()` oder auch `MovePtp.NewAcceleration()` und `MoveVelocity.NewAcceleration()` als Parameter übergeben

Parameter	Datentyp	Bedeutung
Axis	pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss und dessen Profilbeschleunigung gelesen oder verändert werden soll.
Acceleration	Double	Maximale Beschleunigung im Positionierprofil.

Syntax

```
ProfilBeschleunigung := tecGroup.Mechanism.Slope.GetAcceleration(axLeitach-
se);
tecGroup.Mechanism.Slope.SetAcceleration(axLeitachse, 5500.0);
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

Wird die Profilbeschleundigung mit `SetAcceleration()` während einer Positionierung (Zustand **Läuft**) geändert, wirkt sich diese Änderung erst bei der nächsten Positionierung aus.

77.2.4 GetDeceleration(), SetDeceleration()

Beschreibung

Liest oder schreibt den Wert für die maximale Verzögerung, der für die Positionssollwertgenerierung eines Positionierprofils verwendet wird. Dieser Wert gilt für die im ersten Parameter aufgeführte Verbundachse.

Diese Einstellung gilt für alle Positionierbewegungen der Verbundachse so lange, bis der Wert geändert wird.

Dieser Wert wird normalerweise in den Funktionen `MovePtp.Start()` und `Move.Velocity.Start()` oder auch `MovePtp.NewDeceleration()` und `MoveVelocity.NewDeceleration()` als Parameter übergeben.

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss und dessen Profilverzögerung gelesen oder verändert werden soll.
Deceleration	Double	Maximale Verzögerung im Positionierprofil.

Syntax

```
ProfilVerzögerung := tecGroup.Mechanism.Slope.GetDeceleration(axLeitachse);
tecGroup.Mechanism.Slope.SetDeceleration(axLeitachse,12000.0);
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

Wird die Profilverzögerung mit `SetDeceleration()` während einer Positionierung (Zustand **Läuft**) geändert, wirkt sich diese Änderung erst bei der nächsten Positionierung aus.

77.3 Eigenschaften von MCTechno.Mechanism.Slope

Die Funktionalität bietet folgende Eigenschaften:

- SlopeState
- IsAccelerating
- IsAtConstantSpeed
- IsDecelerating
- IsStopped

77.3.1 SlopeState

Beschreibung

Datentyp	Bedeutung
MCTechnoSlopeStates	Liest, in welcher Phase der Bewegung (Beschleunigung, konstante Geschwindigkeit, Verzögerung oder Stillstand) sich der Bahnsollwert des Bahnverbunds aktuell befindet. Es wird der Rampenstatus der Leitachse zurückgegeben. Ist der Technologieverbund inaktiv, so wird **angehalten** zurückgegeben. Mit dieser Eigenschaft können Sie beispielsweise eine Aktion in der Maschine erst dann aktivieren, wenn der Verbund die Beschleunigungsphase verlassen hat.
MCTechnoSlopeStates.Accelerating	Verbund befindet sich in der Beschleunigungsrampe.
MCTechnoSlopeStates.AtConstandSpeed	Verbund fährt mit konstanter Geschwindigkeit.
MCTechnoSlopeStates.Decelerating	Verbund befindet sich in der Verzögerungsrampe.
MCTechnoSlopeStates.Stopped	Verbund steht still.

Syntax

```
Rampenstatus := tecGroup.Mechanism.Slope.SlopeState;
```

Zustandsverwaltung

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

77.3.2 IsAccerlerating

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der berechnete Sollwert der Leitachse des Technologieverbundes in der Beschleunigungsrampe befindet.

Syntax

```
If tecGroup.Mechanism.Slope.IsAccelerating then …
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

77.3.3 IsAtConstantSpeed

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der berechnete Sollwert der Leitachse des Technologieverbunds in der Phase Fahrt mit konstanter Geschwindigkeit befinden.

Syntax

```
If tecGroup.Mechanism.Slope.IsAtConstantSpeed then …
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

77.3.4 IsDecelerating

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der berechnete Sollwert der Leitachse des Technologieverbunds in der Verzögerungsrampe befindet.

Syntax

```
If tecGroup.Mechanism.Slope.IsDecelerating then …
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

77.3.5 IsStopped

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die der berechnete Sollwert der Leitachse des Technologieverbunds in der Phase des Stillstands befindet.

Syntax

```
If tecGroup.Mechanism.Slope.IsStopped then …
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand möglich.

78 MCTechno.Members

Mit der Funktionalität **MCTechno.Members** können Sie die Zusammenstellung eines Technologieverbundes auslesen.

Für sehr spezielle Fälle können Sie auch Verbundachsen aus dem STX-Programm heraus hinzufügen oder entfernen. Da diese Fälle nur in Ausnahmen empfohlen werden, werden diese Befehle hier nicht beschrieben. Bitte setzen Sie sich mit Bucher Automation AG in Verbindung, falls Sie diese Funktionen verwenden möchten.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 49 - 56**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

78.1 Syntax für MCTechno.Members

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Members.<Zugriff auf Wert>(optional <Parameter 1>)
```

78.2 Eigenschaften von MCTechno.Members

Die Funktionalität bietet folgende Eigenschaften:

- FollowerCount
- Follower[]
- FollowerCamCount
- FollowerCam[]
- FollwoerGearCount
- FollowerGear[]
- Master

78.2.1 FollowerCount

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Anzahl der beteiligten Folgeachsobjekte des Technologieverbundes aus.

Syntax

```
FolgeAchsenAnzahl := tecGroup.Techno.Members.FollowerCount;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand des Technologieverbunds möglich, auch im Zustand **inaktiv**.

78.2.2 Follower[n]

Beschreibung

Datentyp	Bedeutung
Pointer to MCAxis	Liest das Achsobjekt zurück, das in der Liste der Folgeachsen des Technologieverbundes die Nummer *n* hat. Die Liste beginnt bei `0` und geht bis `FollwerCount-1`.

Syntax

```
FolgeAchsName := tecGroup.Members.Follower[1].Info.Name;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand des Technologieverbunds möglich, auch im Zustand **inaktiv**.

78.2.3 FollowerCamCount

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Anzahl der beteiligten Folgeachsobjekte des Technologieverbundes aus, die als Kurvenscheiben definiert sind.

Syntax

```
KurvenAchsAnzahl := tecGroup.Techno.Members.FollowerCamCount;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand des Technologieverbunds möglich, auch im Zustand **inaktiv**.

78.2.4 FollowerCam[n]

Beschreibung

Datentyp	Bedeutung
Pointer to MCAxis	Liest das Achsobjekt, das in der Liste der Kurvenscheiben-Folgeachsen des Technologieverbundes die Nummer *n* hat. Die Liste beginnt bei `0` und geht bis `FollwerCamCount-1`.

Syntax

```
KurvenAchsObjekt := tecGroup.Members.FollowerCam[1].Info.ObjektID;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand des Technologieverbunds möglich, auch im Zustand **inaktiv**.

78.2.5 FollowerGearCount

Beschreibung

Datentyp	Bedeutung
Integer	Liest die Anzahl der beteiligten Folgeachsobjekte des Technologieverbundes aus, die als elektrisches Getriebe definiert sind.

Syntax

```
elGetrAchsAnzahl := tecGroup.Techno.Members.FollowerGearCount;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand des Technologieverbunds möglich, auch im Zustand **inaktiv**.

78.2.6 FollowerGear[n]

Beschreibung

Datentyp	Bedeutung
Pointer to MCAxis	Liest das Achsobjekt, das in der Liste der als elektrisches Getriebe definierten Folgeachsen des Technologieverbundes die Nummer *n* hat. Die Liste beginnt bei `0` und geht bis `FollwerGearCount-1`.

Syntax

```
elGetrAchsObjekt := tecGroup.Members.FollowerGear[0].Info.ObjektID;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand des Technologieverbunds möglich, auch im Zustand **inaktiv**.

78.2.7 Master

Beschreibung

Datentyp	Bedeutung
Pointer to MCAxis	Liest das Achsobjekt zurück, das die Leitachse des Technologieverbundes ist.

Syntax

```
LeitAchsName := tecGroup.Members.Master.Info.Name;
```

Zustandsverwaltung

Die Abfrage der Eigenschaft ist in jedem Zustand des Technologieverbunds möglich, auch im Zustand **inaktiv**.

79 MCTechno.MoveHalt

Mit der Funktionalität **MCTechno.MoveHalt** können Sie eine Verbundachse eines Technologieverbundes, die sich in Bewegung befindet, mit einer Rampe anhalten.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 49 - 56**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

79.1 Syntax für MCTechno.MoveHalt

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.MoveHalt.Start.(optional <Parameter 1, Parameter 2>);
```

79.2 Funktionen von MCTechno.MoveHalt

Die Funktionalität bietet folgende Funktionen:

– Start()

79.2.1 Start()

Die Funktion führt den Anhaltebefehl auf die im ersten Parameter genannte Achse unmittelbar aus. Die übergebenen Parameter definieren, auf welche Weise die Achse anhalten soll.

Beachten Sie folgende Regeln:

– Wird die Funktion auf die Leitachse ausgeführt, so halten alle Achsen entsprechend ihrer Kopplungsart an. Alle Achsen bleiben eingekoppelt.

– Wird die Funktion auf eine ausgekoppelte Folgeachse ausgeführt, so hält diese entsprechend der übergebenen Parameter an.

– Für eine eingekoppelte Folgeachse wirkt die Funktion auf die mögliche überlagerte Bewegung. Die Bewegung der Folgeachse aufgrund der Kopplung mit der Masterachse wird nicht beeinflusst.

  Um eine eingekoppelte Folgeachse auszukoppeln und anzuhalten, verwenden Sie die Funktion `MCTechno.Coupling.Decouple(axFolgeachse,MCTechnoDecoupleModes.Immediate_Stop,,DecelerationRamp).`

– Der Parameter `Deceleration` wird in den Anwender-Einheiten eingegeben, die für die Achse vordefiniert sind.

  Eine Anpassung der maximalen Verzögerung über diesen Parameter ändert den Wert im Achsobjekt. Wird beispielsweise beim nächsten `MovePtp.Start`-Befehl kein Wert für den Parameter `Deceleration` übergeben, so wird der hier vorgegebene Wert verwendet.

– Die Parameter beziehen sich auf das Achsobjekt; d. h. sie bilden eine Eigenschaft der Achse. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss und angehalten werden soll.

Parameter	Datentyp	Bedeutung
MCAxisHaltMode	Enum MCAxisHaltModes (optional)	Verhalten der Achse, nachdem sie mit einer Verzögerungsrampe abgebremst wurde:
	MCAxisHaltModes.Normal	Nach Abschluss der Anhalterampe kommt die Achse in Lageregelung zum Stillstand.
	MCAxisHaltModes.AtActualPosition	Nach Abschluss der Anhalterampe fährt die Achse auf die Position zurück, an der sie sich befand, als die Funktion `MoveHalt.Start()` aufgerufen wurde. Die Rückfahrbewegung wird mit den für dieses Achsobjekt eingestellten Parametern ausgeführt.
Deceleration	Double (optional)	Definiert die maximale Verzögerung, die in der Anhalterampe erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.

Syntax

```
tecGroup.MoveHalt.Start(axLeitachse,MCTechnoHaltModes.Normal,50000);
tecGroup.MoveHalt.Start(axLeitachse,MCTechnoHaltModes.AtActualPositi-
on,50000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	–		Gesperrt
Freigegeben	–		Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	X		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Bei der Abfrage, ob das Anhalten beendet ist, sollte nur der **Rampenstatus** berücksichtigt werden.

Der **Betriebszustand** des Verbunds ist eine Summe der Betriebszustände der Verbundachsen. Sobald sich ein Mitglied bewegt, wechselt der Betriebszustand auf **Running** und nur wenn alle Mitglieder stehen, wird der Zustand **Enabled** erreicht. Der Zustand kann deshalb je nach Anwendung mehrdeutig sein.

Rampenstatus der Leitachse:

Der Verbund selbst besitzt auch einen Rampenstatus. Dies ist der Rampenstatus der Leitachse! Deshalb muss hier bei der Rampenstatusabfrage auch dieser berücksichtigt werden, wenn die Leitachse positioniert wird. Der Rampenstatus der Leitachse als Einzelachse bleibt während des aktiven Verbunds immer auf **Stopped** und kann somit nicht genutzt werden.

```
// Leitachse
tecGroup.MoveHalt.Start(axLeitachse, MCTechnoHaltModes.Normal, 1000.0);
when tecGroup.Mechanism.Slope.IsStopped continue;
// Folgeachse
tecGroup.MoveHalt.Start(axFolgeachse, MCTechnoHaltModes.Normal, 1000.0);
when axFolgeachse.Mechanism.Slope.IsStopped continue;
```

80 MCTechno.MovePtp

Mit der Funktionalität **MCTechno.MovePtp** können Sie die Punkt-zu-Punkt-Positionierung einer Achse eines Technologieverbundes starten und parametrieren.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 49 - 56**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

80.1 Syntax für MCTechno.MovePtp

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.MovePtp.<Funktion oder Zugriff auf Wert>(optional <Parameter 1, Parameter 2, Parameter n>);
```

80.2 Funktionen von MCTechno.MovePtp

Die Funktionalität bietet folgende Funktionen:

- Start()
- StartNewTargetPosition()
- NewSpeed()
- NewAcceleration()
- NewDeceleration()
- GetPositioningMode()
- SetPositioningMode()
- GetTargetPosition()
- SetTargetPosition()
- GetTargetWindow()
- SetTargetWindow()
- StartSuperPose()
- StartNewTargetPositionSuperPose()

80.2.1 Start()

Die Funktion startet die Punkt-zu-Punkt-Positionierung einer Achse eines Technologieverbundes. Die Achse beginnt ihre Bewegung unmittelbar oder wendet die Daten auf eine laufende Positionierung an.

Da alle für die Positionierung notwendigen Parameter in nur einem Befehl übergeben werden können, ist dies die kompakteste Form, um eine Positionierbewegung vollständig zu parametrieren und zu starten.

Wird die Funktion auf die Leitachse ausgeführt, so bewegen sich alle eingekoppelten Folgeachsen entsprechend ihrer Koppelfunktion mit.

Wird die Funktion auf eine gekoppelte Folgeachse ausgeführt, so wird die betreffende Achse erst entkoppelt und dann entsprechend den Parametern der Funktion bewegt. Eine ursprünglich gekoppelte Folgeachse wird nach Abschluss der Bewegung nicht automatisch wieder gekoppelt.

Übergabeparameter

Beachten Sie folgende Regeln:

– Alle Parameter werden in den Einheiten angegeben, die vom Anwender für die Achse vordefiniert wurden.

– Die Parameter beziehen sich auf das Achsobjekt; d. h. sie bilden eine Eigenschaft der Achse. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Positionieren im nichtgepufferten Betrieb (Standard)

Je nachdem, ob die Achse in Bewegung ist oder nicht, ergibt sich folgendes Verhalten:

– Aufruf der Funktion im Zustand **freigegeben**: Die Bewegung wird gestartet.

– Aufruf der Funktion im Zustand **läuft**: Die Bewegungsparameter werden gemäß nachfolgender Tabelle für die laufende ODER nachfolgende Bewegung übernommen.

Parameter	Datentyp	Bedeutung	Verhalten beim Funktionsaufruf im Zustand läuft
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.	
PositioningMode	MCTechnoPositioningModes (optional)	Definiert den Positioniermodus (siehe Get/SetPositioningMode [▶ 477]).	Der Modus wird sofort wirksam.
TargetPosition	Double (optional)	Definiert die Zielposition, an der die Achse stehen soll, nachdem die Positionierung ausgeführt wurde.	– Kann die neue Zielposition unter Berücksichtigung der Maximalwerte für Geschwindigkeit, Beschleunigung, Verzögerung und Ruck erreicht werden, so wird sie unter Verwendung der neuen Bewegungsparameter direkt angefahren. – Andernfalls wird die laufende Positionierung zu Ende geführt und die neu programmierte direkt angeschlossen.
Speed	Double (optional)	Definiert die Geschwindigkeit, mit der die Achse positioniert werden soll, solange sie sich nicht in den Rampen (Beschleunigung, Verzögerung) befindet. Der Wert muss größer 0 sein.	Wird eine neue Positionierung gestartet, während die vorherige noch läuft (Zustand läuft), so wird der neue Geschwindigkeitswert unter Berücksichtigung der Maximalwerte für Beschleunigung, Verzögerung und Ruck sofort übernommen. Die Änderung der Geschwindigkeit erfolgt mit den Parametern der Beschleunigungsrampe.

Parameter	Datentyp	Bedeutung	Verhalten beim Funktionsaufruf im Zustand läuft
Acceleration	Double (optional)	Definiert die maximale Beschleunigung, die in der Startrampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.	Die Geschwindigkeitsrampen werden entsprechend der neuen Werte für Beschleunigung und Verzögerung ausgeführt. – Befindet sich die Bewegung gerade in einer solchen Rampe, kommt es zu Übergangsformen, die nicht dem eingestellten Rampentyp entsprechen. – Die Übergänge werden immer unter der Überwachung der Ruckbegrenzung ausgeführt.
Deceleration	Double (optional)	Definiert die maximale Verzögerung, die in der Stopprampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.	
TargetWindow	Double (optional)	Definiert die Breite des Zielfensters für die Positionierung.	Die Änderung des Zielfensters wird sofort in die laufende Positionierung übernommen.

Positionierungsphasen

Eine Positionierbewegung besteht aus folgenden Phasen:

1. Beschleunigungsrampe
2. Konstante Geschwindigkeit

   Diese Phase tritt nur ein, wenn die Geschwindigkeit `Speed` mit den vorgegebenen MovePtp-Parametern erreicht werden kann. Andernfalls geht die Beschleunigungsrampe direkt in die Verzögerungsrampe über (Dreiecksbetrieb).
3. Verzögerungsrampe

Rampenformen

Die Rampenformen der Beschleunigungs- und Verzögerungsphase werden durch den Achsparameter MCTechno.Mechanism.Slope.SetSlopeType (Typ MCTechnoSlopeTypes) vorgegeben. Hier gibt es zwei mögliche Einstellungen:

– `AxisSlopeTypes.Linear`

  Beschleunigungsphase = konstante Beschleunigung,
  Verzögerungsphase = sin-förmige Beschleunigung

– `AxisSlopeTypes.Sine`

  Beschleunigungsphase = sin-förmige Beschleunigung,
  Verzögerungsphase = sin²-förmige Beschleunigung

ⓘ Info

Ruckbegrenzung

Die sin- und sin²-Rampen werden auf Einhaltung des maximal zulässigen Rucks (Parameter `MCAxis.Mechanism.Limitations.MaxJerk`) geprüft und bei Bedarf entsprechend angepasst.

Syntax

```
tecGroup.MovePtp.Start(axLeitachse,AxisPositioningModes.AbsNor-
mal,20000,500,1000,1000,0.5);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√	Läuft	Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	X		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Positionierung stehen Ihnen verschiedene Informationen zur Verfügung:

– Der **Zustand des Verbundes** informiert darüber, ob der Sollwertgenerator aller Verbundachsen eine Bewegung generiert oder angehalten wurde.

```
When tecGroup.State.IsEnabled Continue; //die Verbundachsen befinden sich in
Lageregelung und stehen.
When tecGroup.State.IsRunning Continue; //der Sollwertgenerator erzeugt eine
Bewegung.
```

– Wenn Sie das Erreichen einer definierten Position kurz vor dem Ziel abfragen möchten, kommt das **Zielfenster** zum Einsatz. Die Abfrage ist nur für das betreffende Achsobjekt der Verbundachse möglich.

```
When axLeitachse.IsInTargetWindow Continue;
```

Sehen Sie dazu auch

MCAxis.Buffer [▶ 31]

MCAxis.Mechanism.Slope [▶ 140]

80.2.2 StartNewTargetPosition()

Die Funktion startet die Punkt-zu-Punkt-Positionierung einer Verbundachse des Technologieverbundes. Die Achse beginnt unmittelbar ihre Bewegung oder übernimmt die Daten für die Änderung einer laufenden Positionierung.

Die Funktion ist eine vereinfachte Variante der Funktion `Start()`, mit der nur die relevanten Positionsparameter übergeben werden. Geschwindigkeitswerte und Rampeneinstellungen für die Achse werden übernommen.

In diesem Abschnitt sind die Spezifika für die Funktion `StartNewTargetPosition()` beschrieben. Alle weitere Informationen entnehmen Sie bitte der Beschreibung zur Funktion Start() [▶ 470] des Technologieverbundes.

Übergabeparameter

Beachten Sie folgende Regeln:

– Alle Parameter werden in den Einheiten angegeben, die vom Anwender für die Achse vordefiniert wurden.
– Die Parameter beziehen sich auf das Achsobjekt; d. h. sie bilden eine Eigenschaft der Achse. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Parameter	Datentyp	Bedeutung	Verhalten beim Funktionsaufruf im Status Läuft
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.	
PositioningMode	MCTechnoPositioningModes (optional)	Der PositioningMode (siehe Get/SetPositioningMode [▶ 477]).	Der Modus wird sofort wirksam.
TargetWindow	Double (optional)	Definiert die Breite des Zielfensters für die Positionierung.	Die Änderung des Zielfensters wird sofort in die laufende Positionierung übernommen.

Syntax

```
tecGroup.MovePtp.StartNewTargetPosition(axLeitachse,AxisPositioningModes.AbsNormal,20000,0.5);
```

80.2.3 NewSpeed()

Die Funktion übergibt einen neuen Wert für die Geschwindigkeit einer Punkt-zu-Punkt-Positionierung einer Verbundachse eines Technologieverbundes. Dabei gilt:

– Der Wert muss > `0` sein.
– Ist die Achse bereits in Bewegung (Zustand **Läuft**), so wird der neue Geschwindigkeitswert sofort übernommen.

  Der Übergang von aktueller zu neuer Geschwindigkeit bei einer laufenden Achse erfolgt mit der für die Beschleunigungsrampe eingestellten Beschleunigung. Dies ist auch dann der Fall, wenn sich die Geschwindigkeit verringert.
– Der Rampentyp kann während einer Positionierung nicht geändert werden.
– Bei allen Übergängen ist die Ruckbegrenzung aktiv, falls der Rampentyp *sin²* eingestellt ist.

ⓘ Info

Achse zum Stillstand bringen

Um eine Achse zum Stillstand zu bringen sind ausschließlich die Befehle `MoveHalt` oder `Stop` zu verwenden. Es ist **nicht** möglich eine Achse anzuhalten, indem ein Geschwindigkeitswert von `0` übergeben wird.

Übergabeparameter

Parameter	Datentyp	Bedeutung	Verhalten beim Funktionsaufruf im Zustand Läuft
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.	
Speed	Double	Definiert die Geschwindigkeit, mit der die Achse positioniert werden soll, solange sie sich nicht in den Rampen (Beschleunigung, Verzögerung) befindet.	Der neue Geschwindigkeitswert wird unter Berücksichtigung der Maximalwerte für Beschleunigung, Verzögerung und Ruck sofort übernommen. Die Änderung der Geschwindigkeit erfolgt mit den Parametern der Beschleunigungsrampe.

Syntax

```
tecgroup.MovePtp.NewSpeed(axLeitachse,2000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

80.2.4 NewAcceleration()

Die Funktion übergibt einen neuen Wert für die maximale Beschleunigung einer Punkt-zu-Punkt-Positionierung einer Verbundachse eines Technologieverbundes.

Dabei gilt:

– Eine aktive Beschleunigungsrampe wird nicht verändert.

– Der neue Wert wird bei der nächsten PtP-Positionierung angewendet.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.

Parameter	Datentyp	Bedeutung
Acceleration	Double (optional)	Definiert die maximale Beschleunigung, die in der Startrampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.

Syntax

```
tecGroup.MovePtp.NewAcceleration(axLeitachse,2000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

80.2.5 NewDeceleration()

Die Funktion übergibt einen neuen Wert für die maximale Verzögerung einer Punkt-zu-Punkt-Positionierung einer MCX-Achse.

Dabei gilt:

– Der Wert wird auch während einer laufenden PtP-Positionierung verwendet, falls das Ziel mit dem neuen Parameter noch erreicht werden kann.

– Andernfalls wird der Wert bei der nächsten PtP-Positionierung angewendet.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.
Deceleration	Double (optional)	Definiert die maximale Verzögerung, die in der Stopprampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.

Syntax

```
tecGroup.MovePtp.NewDeceleration(axLeitachse,2000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

80.2.6 GetPositioningMode(), SetPositioningMode()

Beschreibung

Mit den Funktionen können Sie den momentan gültigen Positioniermodus einer Verbundachse eines Technologieverbundes lesen oder einen neuen Wert für diese Achse setzen.

Dabei gilt:

– Ein neuer Wert wird bei der nächsten Punkt-zu-Punkt-Positionierung wirksam, auch wenn die Änderung während einer laufenden Positionierung übergeben wird.

– Nach dem Einschalten wird die letzte Zielposition aus der Ist-Position zum Einschaltzeitpunkt berechnet.

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAcis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.

Parameter	Datentyp	Bedeutung
PositioningMode	MCTechnoPositioningMode	Art der PtP-Bewegung.
	MCTechnoPositioningMode.AbsNormal	Bewegt die Achse auf die im Parameter `TargetPosition` übergebene Position. Diese wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Positionierung ergibt sich aus der letzten und der neuen Zielposition.
	MCTechnoPositioningMode.RelTarget	Bewegt die Achse um den Weg weiter, der im Parameter `TargetPosition` übergeben wurde. Die Zielposition berechnet sich aus der letzten Zielposition addiert mit dem Wert in `TargetPosition`. Die Fahrtrichtung ergibt sich aus dem Vorzeichen des Parameters `TargetPosition`.
	MCTechnoPositioningMode.RelActual	Bewegt die Achse um den Weg weiter, der im Parameter `TargetPosition` übergeben wurde. Die Zielposition berechnet sich aus Ist-Position beim Start der Positionierung addiert mit dem Wert in `TargetPosition`. Die Fahrtrichtung ergibt sich aus dem Vorzeichen des Parameters `TargetPosition`.
	MCTechnoPositioningMode.AbsModuloPos	Nur für Modulo-Achsen gültig. Bewegt die Achse auf die im Parameter `TargetPosition` übergebene Position. Diese wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Positionierung ist immer **positiv**. Dabei kann der Modulo-Umbruch überfahren werden, wenn die neue Zielposition kleiner als die letzte Zielposition ist.
	MCTechnoPositioningMode.AbsModuloNeg	Nur für Modulo-Achsen gültig. Bewegt die Achse auf die im Parameter `TargetPosition` übergebene Position. Diese wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Positionierung ist immer **negativ**. Dabei kann der Modulo-Umbruch überfahren werden, wenn die neue Zielposition größer als die letzte Zielposition ist.
	MCTechnoPositioningMode.AbsModuloAuto	Nur für Modulo-Achsen gültig. – Positionierung wird bei angehaltener Achse gestartet: Bewegt die Achse auf die im Parameter `TargetPosition` übergebene Position. Diese wird in Relation zur Referenzposition **auf dem kürzesten Weg** angefahren. Die Fahrtrichtung der Positionierung ergibt sich aus dem kürzesten Weg zum Ziel. – Positionierung wird bei laufender Achse gestartet: Bewegt die Achse auf die im Parameter `TargetPosition` übergebene Position. Diese wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Positionierung ergibt sich aus der Fahrrichtung der laufenden Positionierung.
	MCTechnoPositioningMode.AbsModuloKeepDirection	Nur für Modulo-Achsen gültig. Bewegt die Achse auf die im Parameter `TargetPosition` übergebene Position. Diese wird in Relation zur Referenzposition angefahren. Die Fahrtrichtung der Positionierung ergibt sich aus der Fahrrichtung der letzten Positionierung.

Syntax

```
LeitachsPositioniermodus := tecGroup.MovePtp.GetPositioningMode(axLeitach-
se);
tecGroup.MovePtp.SetPositioningMode(axFolgeachse,MCTechnoPositioningMo-
des.RelTarget);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	√	Fehler: freigegeben
Fehler: gesperrt	√	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

80.2.7 GetTargetPosition(),SetTargetPosition()

Beschreibung

Mit den Funktionen können Sie den momentan gültigen Wert von `TargetPosition` einer Verbundachse eines Technologieverbundes lesen oder einen neuen Wert für diese Achse schreiben. Der Aufruf der Funktion im Zustand **Läuft** werden die Bewegungsparameter beim Start der nächsten Punkt-zu-Punkt-Positionierung wirksam.

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAcis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss und bewegt werden soll.
TargetPosition	Double	Eigenschaft zur Ermittlung der Zielposition abhängig von der Eigenschaft `PositioningMode`.

Syntax

```
Target := tecGroup.MovePtp.GetTargetPosition(axFolgeachse);
tecGroup.MovePtp.SetTargetPosition(axLeitachse,12000);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben

Ausgangszustand	Erlaubt	Endzustand
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	√	Fehler: freigegeben
Fehler: gesperrt	√	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

80.2.8 GetTargetWindow(), SetTargetWindow()

Beschreibung

Mit den Funktionen können Sie den momentan gültigen Wert der Zielfenstereinstellung einer Verbundachse eines Technologieverbundes lesen oder einen neuen Wert für diese Achse schreiben. Dieser wird beim Start der nächsten Punkt-zu-Punkt-Positionierung wirksam.

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.
TargetWindow	Double	Definiert, wann im Bereich der Zielposition einer Positionierung der Status `IsInTargetWindow` aktiv wird.

Syntax

```
Targetwindow := tecGroup.MovePtp.GetTargetWindow(axLeitachse);
tecGroup.MovePtp.SetTargetWindow(axFolgeachse,0.7);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	√	Fehler: freigegeben
Fehler: gesperrt	√	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

80.2.9 StartSuperPose()

ⓘ Info

APN 51 – Überlagerte Positionierung

Beachten Sie die ausführlichen Informationen in der **Application Note 051** zur Funktion der überlagerten Bewegung einer Folgeachse in einem elektrischen Getriebe.

Die Funktion startet eine überlagerte Punkt-zu-Punkt-Positionierung einer Folgeachse eines Technologieverbundes, die über ein elektrisches Getriebe gekoppelt ist.

Dabei gilt folgendes Verhalten:

– Wird die Funktion auf eine gekoppelte Folgeachse ausgeführt, die als elektrisches Getriebe konfiguriert ist, ausgeführt, so wird die überlagerte Bewegung gestartet.

– Wird die Funktion auf eine ausgekoppelte Folgeachse, die als elektrisches Getriebe konfiguriert ist, oder die Leitachse ausgeführt, so wird für diese Achse eine Bewegung gestartet, die so verhält, als wäre sie mit der Funktion Start() [▶ 470] ausgeführt worden. Weitere Informationen entnehmen Sie bitte dem verlinkten Abschnitt.

– Wird die Funktion auf eine Folgeachse ausgeführt, die als Kurvenscheibe konfiguriert ist, wird eine Meldung ausgegeben.

Übergabeparameter

Beachten Sie folgende Regeln:

– Alle Parameter werden in den Einheiten angegeben, die vom Anwender für die Achse vordefiniert wurden.

– Die Parameter beziehen sich auf das Achsobjekt; d. h. sie bilden eine Eigenschaft der Achse. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.
Positioning-Mode	MCTechnoPositioningModes (optional)	Definiert den Positioniermodus (siehe Get/SetPositioningMode [▶ 477]).
TargetPosition	Double (optional)	Definiert die Zielposition, an der die Achse stehen soll, nachdem die Positionierung ausgeführt wurde.
Speed	Double (optional)	Definiert die Geschwindigkeit, mit der die Achse positioniert werden soll, solange sie sich nicht in den Rampen (Beschleunigung, Verzögerung) befindet. Der Wert muss größer `0` sein.
Acceleration	Double (optional)	Definiert die maximale Beschleunigung, die in der Startrampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.

Parameter	Datentyp	Bedeutung
Deceleration	Double (optional)	Definiert die maximale Verzögerung, die in der Stopprampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.
TargetWindow	Double (optional)	Definiert die Breite des Zielfensters für die Positionierung.

Syntax

```
tecGroup.MovePtp.StartSuperPose(axElGetriebe.AxisPositioningModes.AbsNor-
mal,200,500,1000,1000,0.5);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√	Läuft	Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	X		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

80.2.10 StartNewTargetPositionSuperPose()

ⓘ Info

APN 51 – Überlagerte Positionierung

Beachten Sie die ausführlichen Informationen in der **Application Note 051** zur Funktion der überlagerten Bewegung einer Folgeachse in einem elektrischen Getriebe.

Die Funktion ist eine abgekürzte Variante der Funktion `StartSuperPose()`. Beide Funktionen starten eine überlagerte Punkt-zu-Punkt-Positionierung einer Folgeachse eines Technologieverbundes, die über ein elektrisches Getriebe gekoppelt ist. Mit der Variante werden nur die für die Position relevanten Parameter übergeben werden. Die Werte für Geschwindigkeit und Rampeneinstellungen werden so, wie zuletzt für die Achse eingestellt, übernommen.

Dabei gilt folgendes Verhalten:

– Wird die Funktion auf eine gekoppelte Folgeachse ausgeführt, die als elektrisches Getriebe konfiguriert ist, ausgeführt, so wird die überlagerte Bewegung gestartet.

– Wird die Funktion auf eine ausgekoppelte Folgeachse, die als elektrisches Getriebe konfiguriert ist, oder die Leitachse ausgeführt, so wird für diese Achse eine Bewegung gestartet, die sich so verhält, als wäre sie mit der Funktion StartNewTargetPosition() [▶ 473] ausgeführt worden. Weitere Informationen entnehmen Sie bitte dem verlinkten Abschnitt.

– Wird die Funktion auf eine Folgeachse ausgeführt, die als Kurvenscheibe konfiguriert ist, wird eine Meldung ausgegeben.

Übergabeparameter

– Alle Parameter werden in den Einheiten angegeben, die vom Anwender für die Achse vordefiniert wurden.
– Die Parameter beziehen sich auf das Achsobjekt; d. h. sie bilden eine Eigenschaft der Achse. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.
PositioningMode	MCTechnoPositioningModes (optional)	Definiert den Positioniermodus, siehe GetPositioningMode(), SetPositioningMode() [▶ 477].
TargetWindow	Double (optional)	Definiert die Breite des Zielfensters für die Positionierung.

Syntax

```
tecGroup.MovePtp.StartTargetPositionSuperPose(axElGetriebe,AxisPositioning-
Modes.AbsNormal,200,0.5);
```

Details zu Zustandsveraltung und Zustandsüberwachgung sind identisch mit der Funktion Start() [▶ 470].

81 MCTechno.MoveVelocity

Mit der Funktionalität **MCTechno.MoveVelocity** können Sie die Endlospositionierung einer MCX-Achse in einem Technologieverbund starten und parametrieren.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Achsen den **Motion Control Application Notes (APN) Nr. 54, 55 und 57**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

81.1 Syntax für MCTechno.MoveVelocity

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.MoveVelocity.<Funktion oder Zugriff auf Wert>(optional
<Parameter 1, Parameter 2, Parameter n>);
```

81.2 Funktionen von MCTechno.MoveVelocity

Die Funktionalität bietet folgende Funktionen:

- Start()
- Reverse()
- NewSpeed()
- NewAcceleration()
- NewDeceleration()
- GetDirection()
- SetDirection()
- StartSuperPose()
- ReverseSuperPose()

81.2.1 Start()

Die Funktion startet die Endlospositionierung einer Achse in einem Technologieverbund. Die Achse beginnt ihre Bewegung unmittelbar oder wendet die Daten auf eine laufende Positionierung an.

Dabei gilt:

- Wird die Funktion auf die **Leitachse** ausgeführt, so bewegen sich alle eingekoppelten Folgeachsen entsprechend ihrer Koppelfunktion mit.
- Wird die Funktion auf eine gekoppelte **Folgeachse** ausgeführt, so wird die betreffende Achse erst entkoppelt und dann entsprechend den Parametern der Funktion bewegt. Eine ursprünglich gekoppelte Folgeachse wird nach Abschluss der Bewegung nicht automatisch wieder gekoppelt.
- Die Achse wird mit der aktuellen oder programmierten Rampe beschleunigt und fährt so lange mit dem aktuellen oder programmierten Wert für `Speed` bis sie durch einen Halt- oder Stopp-Befehl angehalten wird. Die Endlospositionierung ist lagegeregelt, so dass während der Bewegung auch ein Positionsschleppfehler überwacht wird.

Übergabeparameter

Beachten Sie folgende Regeln:

– Alle Parameter werden in den Einheiten angegeben, die vom Anwender für die Achse vordefiniert wurden.
– Die Parameter beziehen sich auf das Achsobjekt; d. h. sie bilden eine Eigenschaft der Achse. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Positionieren im nichtgepufferten Betrieb

Je nachdem, ob die Achse in Bewegung ist oder nicht, ergibt sich folgendes Verhalten:

– Aufruf der Funktion im Zustand **Freigegeben**: Die Bewegung wird gestartet.
– Aufruf der Funktion im Zustand **Läuft**: Die Bewegungsparameter werden gemäß nachfolgender Tabelle für die laufende ODER nachfolgende Bewegung übernommen.

Parameter	Datentyp	Bedeutung	Verhalten beim Funktionsaufruf im Zustand Läuft
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.	
Direction	Directions (optional) – Directions.Positive – Directions.Negative	Definiert die Bewegungsrichtung (siehe GetDirection(), SetDirection() [▶ 491]).	Die neuen Richtungsparameter werden direkt übernommen. Wenn die neue Richtung entgegengesetzt zur Richtung der augenblicklichen Bewegung ist, wird ein Reversiervorgang mit den Beschleunigungsrampen ausgeführt.
Speed	Double (optional)	Definiert die Geschwindigkeit, mit der die Achse positioniert werden soll, solange sie sich nicht in den Rampen (Beschleunigung, Verzögerung) befindet. Der Wert muss größer `0` sein.	Wird eine neue Positionierung gestartet, während die vorherige noch läuft (Zustand **Läuft**), so wird der neue Geschwindigkeitswert unter Berücksichtigung der Maximalwerte für Beschleunigung, Verzögerung und Ruck sofort übernommen. Die Änderung der Geschwindigkeit erfolgt mit den Parametern der Beschleunigungsrampe.

Parameter	Datentyp	Bedeutung	Verhalten beim Funktionsaufruf im Zustand Läuft
Acceleration	Double (optional)	Definiert die maximale Beschleunigung, die in der Startrampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.	Die Geschwindigkeitsrampen werden entsprechend der neuen Werte für Beschleunigung und Verzögerung ausgeführt. – Befindet sich die Bewegung gerade in einer solchen Rampe, kommt es zu Übergangsformen, die nicht dem eingestellten Rampentyp entsprechen. – Die Übergänge werden immer unter der Überwachung der Ruckbegrenzung ausgeführt.
Deceleration	Double (optional)	Definiert die maximale Verzögerung, die beim nächsten `MoveHalt.Stop()`-Befehl in der Stopprampe während der Positionierung erreicht werden soll, wenn dort kein anderer Wert übergeben wird. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.	

Rampenformen

Die Rampenformen der Beschleunigungs- und Verzögerungsphase werden durch den Achsparameter `MCTechno.Mechanism.Slope.SlopeType` (Typ `AxisSlopeTypes`) vorgegeben (siehe MCAxis.Mechanism.Slope [▶ 140]. Hier gibt es zwei mögliche Einstellungen:

– `AxisSlopeTypes.Linear`

  Beschleunigungsphase = konstante Beschleunigung,
  Verzögerungsphase = sin-förmige Beschleunigung

– `AxisSlopeTypes.Sine`

  Beschleunigungsphase = sin-förmige Beschleunigung,
  Verzögerungsphase = sin²-förmige Beschleunigung

ⓘ Info

Ruckbegrenzung

Die sin- und sin²-Rampen werden auf Einhaltung des maximal zulässigen Rucks (Parameter `MCAxis.Mechanism.Limitations.MaxJerk`) geprüft und bei Bedarf entsprechend angepasst.

Syntax

```
tecGroup.MoveVelocity.Start(axLeitachse,Directions.Positive,
Speed1,5000,2000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√	Läuft	Läuft
Läuft	√	Läuft	Läuft
Unterbrochen	X		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Positionierung stehen Ihnen verschiedene Informationen zur Verfügung:

Der **Zustand des Verbundes** informiert darüber, ob der Sollwertgenerator aller Verbundachsen eine Bewegung generiert oder der Verbund angehalten wurde.

```
When tecGroup.State.IsEnabled Continue; //der Verbund befindet sich in
Lageregelung und steht.
When tecGroup.State.IsRunning Continue; //der Sollwertgenerator erzeugt eine
Bewegung.
```

81.2.2 Reverse()

Die Funktion kehrt die Richtung einer laufenden, überlagerten Endlospositionierung einer Folgeachse in einem Technologieverbund um, die über ein elektrisches Getriebe gekoppelt ist. D.h. die Bewegungsrichtung wird umgekehrt und die Verbundachse mit der zuletzt eingestellten Geschwindigkeit in die entgegengesetzte Richtung verfahren.

Dabei gilt:

– Die Änderung der Bewegungsrichtung wird mit der zuletzt eingestellten Beschleunigungsrampe ausgeführt.

– Befindet sich die Achse im Zustand **Freigegeben**, hat der Befehl keine Auswirkung.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.

ⓘ Info

Ruckbegrenzung

Die sin- und sin²-Rampen werden auf Einhaltung des maximal zulässigen Rucks (Parameter `MCAxis.Mechanism.Limitations.MaxJerk`) geprüft und bei Bedarf entsprechend angepasst.

Syntax

```
tecGroup.MoveVelocity.Reverse(axLeitachse);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	–		Freigegeben
Läuft	√	Läuft	Läuft
Unterbrochen	X		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

81.2.3 NewSpeed()

Die Funktion übergibt einen neuen Wert für die Geschwindigkeit einer Endlospositionierung einer Verbundachse. Dabei gilt:

– Der Wert muss > `0` sein.

– Ist die Achse bereits in Bewegung (Zustand **Läuft**), so wird der neue Geschwindigkeitswert sofort übernommen.

  Der Übergang von aktueller zu neuer Geschwindigkeit bei einer laufenden Achse, erfolgt mit den für die Beschleunigungsrampen eingestellten Werten. Dies ist auch dann der Fall, wenn sich die Geschwindigkeit verringert.

– Ist die Achse im Stillstand (Zustand **Freigegeben**), wird der Wert erst bei der nächsten Positionierung angewendet.

– Der Rampentyp kann während einer Positionierung nicht geändert werden.

– Bei allen Übergängen ist die Ruckbegrenzung aktiv, falls der Rampentyp *sin²* eingestellt ist.

ⓘ Info

Achse zum Stillstand bringen

Um eine Achse zum Stillstand zu bringen sind ausschließlich die Befehle `MoveHalt` oder `Stop` zu verwenden. Es ist **nicht** möglich eine Achse anzuhalten, indem ein Geschwindigkeitswert von `0` übergeben wird.

Übergabeparameter

Parameter	Datentyp	Bedeutung	Verhalten beim Funktionsaufruf im Zustand Läuft
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.	
Speed	Double	Definiert die Geschwindigkeit, mit der die Achse positioniert werden soll, solange sie sich nicht in den Rampen (Beschleunigung, Verzögerung) befindet.	Wird eine neue Positionierung gestartet, während die vorherige noch läuft (Zustand Läuft), so wird der neue Geschwindigkeitswert unter Berücksichtigung der Maximalwerte für Beschleunigung, Verzögerung und Ruck sofort übernommen. Die Änderung der Geschwindigkeit erfolgt mit den Parametern der Beschleunigungsrampe.

Syntax

```
tecGroup.MoveVelocity.NewSpeed(axLeitachse,2000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

81.2.4 NewAcceleration()

Die Funktion übergibt einen neuen Wert für die maximale Beschleunigung einer Endlospositionierung einer Verbundachse.

Dabei gilt:

– Eine aktive Beschleunigungsrampe wird nicht verändert.

– Der neue Wert wird bei der nächsten Endlospositionierung angewendet.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.

Parameter	Datentyp	Bedeutung
Acceleration	Double (optional)	Definiert die maximale Beschleunigung, die in der Startrampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.

Syntax

```
tecGroup.MoveVelocity.NewAcceleration(axLeitachse,2000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

81.2.5 NewDeceleration()

Die Funktion übergibt einen neuen Wert für die maximale Verzögerung einer Endlospositionierung einer Verbundachse.

Dabei gilt:

– Der Wert wird auch während einer laufenden Positionierung verwendet, falls das Ziel mit dem neuen Parameter noch erreicht werden kann.

– Andernfalls wird der Wert bei der nächsten Positionierung angewendet.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.
Deceleration	Double (optional)	Definiert die maximale Verzögerung, die in der Stopprampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.

Syntax

```
tecGroup.MoveVelocity.NewDeceleration(axLeitachse,2000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

81.2.6 GetDirection(), SetDirection()

Beschreibung

Mit den Funktionen lesen Sie die momentan gültige Richtung einer Verbundachse oder setzen einen neuen Wert für diese Achse. Dabei gilt:

– Der Wert wird bei der nächsten Endlos-Positionierung wirksam.

– Dies ist auch dann der Fall, wenn die Änderung während einer laufenden Positionierung übergeben wird.

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.
Direction	Directions (optional)	Richtung der Endlosbewegung.
	Directions.Positive	Die Achse wird in positive Richtung bewegt.
	Directions.Negative	Die Achse wird in negative Richtung bewegt.

Syntax

```
AX_Richtung := tecGroup.MoveVelocity.GetDirection(axLeitachse);
tecGroup.MoveVelocity.SetDirection(axLeitachse,Directions.Negative);
```

Zustandsverwaltung

Das Lesen der Funktion ist in jedem Zustand möglich.

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft

Ausgangszustand	Erlaubt	Endzustand
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	√	Fehler: freigegeben
Fehler: gesperrt	√	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

81.2.7 StartSuperPose()

ⓘ Info

APN 51 – Überlagerte Positionierung

Beachten Sie die ausführlichen Informationen in der **Application Note 051** zur Funktion der überlagerten Bewegung einer Folgeachse in einem elektrischen Getriebe.

Die Funktion startet eine überlagerte Endlospositionierung einer Folgeachse in einem Technologieverbund, die über ein elektrisches Getriebe gekoppelt. Die Achse beginnt ihre Bewegung unmittelbar oder wendet die Daten auf eine laufende Positionierung an.

Dabei gilt:

– Wird die Funktion auf eine **gekoppelte Folgeachse** ausgeführt, die als elektrisches Getriebe konfiguriert ist, ausgeführt, so wird die überlagerte Bewegung gestartet.

– Wird die Funktion auf eine **entkoppelte Folgeachse**, die als elektrisches Getriebe konfiguriert ist, oder die **Leitachse** ausgeführt, startet die betreffende Achse eine Bewegung, die sich identisch zur Ausführung mit der Funktion Start() [▶ 484] verhält.

– Wird die Funktion auf eine **Folgeachse** ausgeführt, die als Kurvenscheibe konfiguriert ist, wird eine Meldung ausgegeben.

Übergabeparameter

Beachten Sie folgende Regeln:

– Alle Parameter werden in den Einheiten angegeben, die vom Anwender für die Achse vordefiniert wurden.

– Die Parameter beziehen sich auf das Achsobjekt; d. h. sie bilden eine Eigenschaft der Achse. Wird für einen Parameter kein Wert übergeben, so greift Motion Control auf den zuletzt übergebenen Wert des Parameters zurück.

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.
Direction	Directions (optional) – Directions.Positive – Directions.Negative	Definiert die Bewegungsrichtung (siehe GetDirection(), SetDirection() [▶ 491]).
Speed	Double (optional)	Definiert die Geschwindigkeit, mit der die Achse positioniert werden soll, solange sie sich nicht in den Rampen (Beschleunigung, Verzögerung) befindet. Der Wert muss größer 0 sein.

Parameter	Datentyp	Bedeutung
Acceleration	Double (optional)	Definiert die maximale Beschleunigung, die in der Startrampe während der Positionierung erreicht werden soll. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.
Deceleration	Double (optional)	Definiert die maximale Verzögerung, die beim nächsten `MoveHalt.Stop()`-Befehl in der Stopprampe während der Positionierung erreicht werden soll, wenn dort kein anderer Wert übergeben wird. Mögliche Rampentypen: – *sin²*: Die maximale Verzögerung wird in der Mitte der Rampe erreicht. – *linear*: Die Rampe wird durchgängig mit der maximalen Verzögerung gefahren.

Rampenformen

Die Rampenformen der Beschleunigungs- und Verzögerungsphase werden durch den Achsparameter `MCTechno.Mechanism.Slope.SlopeType` (Typ `AxisSlopeTypes`) vorgegeben (siehe SlopeType [▶ 142]). Hier gibt es zwei mögliche Einstellungen:

– `AxisSlopeTypes.Linear`

  Beschleunigungsphase = konstante Beschleunigung,
  Verzögerungsphase = sin-förmige Beschleunigung

– `AxisSlopeTypes.Sine`

  Beschleunigungsphase = sin-förmige Beschleunigung,
  Verzögerungsphase = sin²-förmige Beschleunigung

ⓘ Info

Ruckbegrenzung

Die sin- und sin²-Rampen werden auf Einhaltung des maximal zulässigen Rucks (Parameter `MCAxis.Mechanism.Limitations.MaxJerk`) geprüft und bei Bedarf entsprechend angepasst.

Syntax

```
tecGroup.MoveVelocity.StartSuperPose(axElGetriebe,Directions.Negati-
ve,200,1000,1000);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√	Läuft	Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	X		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der Positionierung stehen Ihnen verschiedene Informationen zur Verfügung:

Der **Zustand des Verbundes** informiert darüber, ob der Sollwertgenerator aller Verbundachsen eine Bewegung generiert oder der Verbund angehalten wurde.

```
When tecGroup.State.IsEnabled Continue; //der Verbund befindet sich in
Lageregelung und steht.
When tecGroup.State.IsRunning Continue; //der Sollwertgenerator erzeugt eine
Bewegung.
```

81.2.8 ReverseSuperPose()

Die Funktion kehrt die Richtung einer laufenden, überlagerten Endlospositionierung einer Folgeachse in einem Technologieverbund um, die über ein elektrisches Getriebe gekoppelt ist. D.h. die Bewegungsrichtung wird umgekehrt und die Verbundachse mit der zuletzt eingestellten Geschwindigkeit in die entgegengesetzte Richtung verfahren.

Dabei gilt:

– Die Änderung der Bewegungsrichtung wird mit der zuletzt eingestellten Beschleunigungsrampe ausgeführt.

– Befindet sich die Achse nicht in einer Endlospositionierung, hat der Befehl keine Auswirkung.

– Wird die Funktion auf eine **entkoppelte Folgeachse**, die als elektrisches Getriebe konfiguriert ist, oder die **Leitachse** ausgeführt, startet die betreffende Achse eine Bewegung, die sich identisch zur Ausführung mit der Funktion Reverse() [▶ 191] verhält.

– Beim Versuch, die Funktion auf eine **Folgeachse** auszuführen, die als Kurvenscheibe konfiguriert ist, wird eine Meldung ausgegeben.

ⓘ Info

APN 51 – Überlagerte Positionierung

Beachten Sie die ausführlichen Informationen in der **Application Note 051** zur Funktion der überlagerten Bewegung einer Folgeachse in einem elektrischen Getriebe.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Axis	Pointer to MCAxis	Achsobjekt, das Teilnehmer des Technologieverbundes sein muss, und bewegt werden soll.

ⓘ Info

Ruckbegrenzung

Die sin- und sin²-Rampen werden auf Einhaltung des maximal zulässigen Rucks (Parameter `MCAxis.Mechanism.Limitations.MaxJerk`) geprüft und bei Bedarf entsprechend angepasst.

Syntax

```
tecGroup.MoveVelocity.ReverseSuperPose(axElGetriebe);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	–		Freigegeben
Läuft	√	Läuft	Läuft
Unterbrochen	X		Unterbrochen
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

82 MCTechno.Power

Mit der Funktionalität **MCTechno.Power** können Sie die Regelung und damit die Drehmomenterzeugung eines Technologieverbundes freigeben oder sperren. Alle am Verbund beteiligten Achsen werden gleichzeitig in den neuen Betriebszustand versetzt.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 49 - 56**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

82.1 Syntax für MCTechno.Power

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Power.<Funktion>(optional <Parameter>;
```

82.2 Funktionen von MCTechno.Power

Die Funktionalität bietet folgende Funktionen:

– Enable()
– Disable()
– Quickstop()

82.2.1 Enable()

Die Funktion aktiviert die Regler der Servoverstärker aller Verbundachsen und gibt die Spannung auf den Motor frei. Ein Technologieverbund kann nur freigegeben werden, wenn alle Verbundachsen betriebsbreit sind.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
tecGroup.Power.Enable();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Freigegeben
Freigegeben	–	Freigegeben
Läuft	–	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Der Zustandsübergang kann nicht in einem MCX-Zyklus abgearbeitet werden. Fragen Sie deshalb das Erreichen des neuen Zustands im Programm ab, bevor Sie weitere Befehle für den Verbund aufrufen.

```
When tecGroup.State.IsEnabled Continue;
```

82.2.2 Disable()

Die Funktion sperrt den Servoverstärker aller Verbundachsen für die Drehmomenterzeugung (Impulssperre). Der Motoren werden sofort stromlos und trudelt aus, falls keine Bremse aktiv wird.

Übergabeparameter

Parameter	Datentyp	Bedeutung
DisableModes	MCPowerDisableMode (optional)	Modus für die Ausführung der `Disable()`-Funktion.
	MCPowerDisableMode.Forced (Default)	Die Impulssperre wirkt in jedem Fall sofort. Dieser Modus ist auch aktiv, wenn kein Parameter angegeben wird.
	MCPowerDisableMode.StateCheck	Die Impulssperre wird nur dann wirksam, wenn sich der Verbund im Betriebszustand **Freigegeben** befindet. In allen anderen Zuständen wird ein Bedienfehler ausgegeben.

Syntax

```
tecGroup.Power.Disable();
tecGroup.Power.Disable(MCPowerDisableModes.Forced);
tecGroup.Power.Disable(MCPowerDisableModes.StateCheck);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Für den Aufruf ohne Parameter oder mit dem Parameter `MCPowerDisableModes.Forced` gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	–	Gesperrt
Freigegeben	√	Gesperrt
Läuft	√	Gesperrt
Unterbrochen	√	Gesperrt
Fehler: freigegeben	√	Fehler: gesperrt
Fehler: gesperrt	–	Fehler: gesperrt

Für den Aufruf mit dem Parameter `MCPowerDisableModes.StateCheck` gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	–	Gesperrt
Freigegeben	√	Gesperrt
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Der Zustandsübergang kann nicht in einem MCX-Zyklus abgearbeitet werden. Bevor Sie weitere Befehle für den Verbund aufrufen, fragen Sie deshalb das Erreichen des neuen Zustands im Programm wie folgt ab:

```
When tecGroup.State.IsDisabled Continue;
```

Hat sich die Achse vorher im Zustand **Fehler: freigegeben** befunden, verwenden Sie folgenden Befehl.

```
When geoGroup.State.IsDisabledError Continue;
```

82.2.3 QuickStop()

Die Funktion hält alle Verbundachsen mit der Schnellhaltverzögerung an und sperrt die Servoverstärker nach Erreichen des Stillstands.

HINWEIS



Keine Lageregelung der Verbundachsen bei Schnellhalt

Während der Schnellhaltverzögerung findet keine Lageregelung statt, und damit auch keine koordinierte Bewegung der Verbundachsen entsprechend der Definition als Kurvenscheibe oder elektrisches Getriebe. Jede Verbundachse kommt abhängig von ihrer aktuellen Geschwindigkeit und der Einstellung für die Schnellhaltverzögerung zu einem anderen Zeitpunkt zum Stehen.

► Verwenden Sie den Befehl nur in Notfällen!

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
tecGroup.Power.QuickStop();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	–	Gesperrt
Freigegeben	√	Gesperrt
Läuft	√	Gesperrt
Unterbrochen	√	Gesperrt
Fehler: freigegeben	√	Fehler: gesperrt
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Der Zustandsübergang kann nicht in einem MCX-Zyklus abgearbeitet werden. Bevor Sie weitere Befehle für den Verbund aufrufen, fragen Sie deshalb das Erreichen des neuen Zustands im Programm wie folgt ab:

```
When tecGroup.State.IsDisabled Continue;
```

82.3 Beispieltask MCTechno.Power

Hier finden Sie einen Beispieltask einer Einzelachse, der verschiedene Motion-API-Funktionen enthält:

```
Task Sample1 Autorun
    //Freigeben der Verbundachsen
    If tecGroup.State.IsDisabled Then
        tecGroup.Power.Enable();
    End_if;
    When tecGroup.State.IsEnabled Continue;

    //hier kann ein Prozess mit dem Verbund stehen oder andere Tasks bewegen
den Verbund
    //…

    // Verbund sperren
    If tecGroup.State.IsRunning Then    //falls der Verbund noch in Bewegung
ist, mit sinnvoller Verzögerung anhalten

        tecGroup.MoveHalt.Start(Techno.Members.Master,MCTechnoHaltModes.Nor-
mal,1000.0);
    When tecGroup.Mechanism.Slope.IsStopped Continue;

    End_if;
    tecGroup.Power.Disable(); // Verbund sperren
    When tecGroup.State.IsDisabled Continue;

End_Task;
```

83 MCTechno.Settings

Mit der Funktionalität **MCTechno.Settings** können Sie alle Parameter einer MCX-Technologieverbunds gemeinsam anpassen oder abspeichern.

Parameter bei aktivem Speichervorgang nicht anpassen

Das Speichern der Parameter in eine Datei ist ein Vorgang, der im Hintergrund abgearbeitet wird. D. h. er erstreckt sich über mehrere Zyklen der MCX. Werden während des Abspeicherns Parameterwerte geändert, ist nicht vorhersehbar, ob der alte oder neue Wert in die Datei geschrieben wird.

Funktionsweise der Parameterverwaltung

Parameter, die im Motion-Setup von JetSym gespeichert werden, werden in einer Textdatei auf dem PC und, auf Nachfrage, auch im Dateisystem der Steuerung (Unterordner Motion) abgelegt. Der Dateiname der Parameterdatei auf der Steuerung ist *MCTechno_X.ini*, wobei das *X* für die Nummer der Technologieverbund-ID steht. Die Technologieverbund-ID wird auf der Seite Eigenschaften im Motion-Setup des MCX-Verbunds angezeigt. Außerdem zeigt der Tooltipp im Hardware-Baum die Technologieverbund-ID an, wenn der Mauszeiger auf dem Verbund liegt.)

Beim Booten der Steuerung liest der MCX-Kernel die Daten der *MCTechno_X.ini*-Datei und verwendet diese für den betreffenden Verbund. Ist eine solche Datei auf der Steuerung vorhanden, muss also keine Parametrierung mehr vorgenommen werden. Ist keine entsprechende Datei für den Verbund vorhanden, werden die Parameter mit Default-Werten geladen.

Im Normalfall ist keine Anpassung an der Datei aus dem STX-Programm heraus erforderlich.

In ganz wenigen Ausnahmefällen kann es jedoch sinnvoll sein, die Parameterdatei auf der Steuerung neu einzulesen oder die Parameter abzuspeichern. Beim Abspeichern muss jedoch bedacht werden, dass die im JetSym-Projekt gespeicherte Parameter-Datei dann unter Umständen nicht mehr mit den auf der Maschine wirksamen Parametern übereinstimmt.

83.1 Syntax für MCTechno.Settings

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Settings.<Funktion>(optional <Parameter>;
```

83.2 Funktionen von MCTechno.Settings

Die Funktionalität bietet folgende Funktionen:

– SaveToFile()
– LoadFromFile()

83.2.1 SaveToFile()

Übergabeparameter

Parameter	Datentyp	Bedeutung
Suffix	Integer (optional)	Aktiviert das Abspeichern der Parameter des MCX-Verbunds in eine Datei im Unterordner Motion im Dateisystem der Steuerung. Abhängig vom Übergabeparameter werden unterschiedliche Dateien erzeugt bzw. überschrieben: – Parameter gleich `0` oder kein Parameter: Die Parameterdatei *MCTechno_X.ini* für diesen Verbund wird mit den aktuellen Werten erstellt oder überschrieben. – Parameter ungleich `0`: Die Parameterdatei *MCTechno_X_TmpY.ini* wird mit den aktuellen Werten erstellt oder überschrieben. Dabei ist *Y* gleich dem Parameter `Suffix`.

Syntax

```
tecGroup.Settings.SaveToFile();    // überschreibt (oder erzeugt) die Datei
MCTechno_1.ini (wenn tecGroup die Techno-ID 1 hat)
tecGroup.Settings.SaveToFile(3);    // erzeugt die Datei MCTecho_1_Tmp3.ini
(wenn tecGroup die Techno-ID 1 hat)
```

Zustandsverwaltung

Das Aufrufen der Funktion ist jedem Zustand möglich.

83.2.2 LoadFromFile()

Übergabeparameter

Parameter	Datentyp	Beschreibung
Suffix	Integer (optional)	Aktiviert das Übernehmen der Parameter des MCX-Verbunds aus einer Datei im Unterordner Motion im Dateisystem der Steuerung. Abhängig vom Übergabeparameter werden unterschiedliche Dateien gelesen. – Parameter gleich `0` oder kein Parameter: Die Daten aus der Datei *MCTechno_X.ini* werden für den betreffenden Verbund gelesen und verwendet. – Parameter ungleich `0`: Die Daten aus der Datei *MCTechno_TmpY.ini* werden für den betreffenden Verbund gelesen und verwendet. Dabei ist *Y* gleich dem Parameter `Suffix`.

Syntax

```
tecGroup.Settings.LoadFromFile();    //-> Liest die Datei MCTechno_1.ini
(wenn tecGroup die Geo-ID 1 hat)
tecGroup.Settings.LoadFromFile(3);    //-> Liest die Datei MCTech-
no_1_Tmp3.ini (wenn tecGroup die Geo-ID 1 hat)
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	X	Freigegeben
Läuft	X	Läuft
Unterbrochen	X	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

84 MCTechno.State

Mit der Funktionalität **MCTechno.State** können Sie verschiedene Informationen eines Technologieverbundes abfragen, wie z. B. Betriebsart, Bewegungsart etc.

Grundlegende und weiterführende Informationen

Beachten Sie in diesem Zusammenhang ebenso folgende Informationen:

– Für Zustandsübergänge den Abschnitt MCTechno.State.Transitions [▶ 512];

– Für eine detaillierte Beschreibung des Aufbaus des MCX-Kerns die JetSym–Onlinehilfe (JetSym Motion und MotionControl > JetSym MotionControl > MotionControl-Grundlagen > Bedienkonzept > Überblick: Bedienkonzept)

– Für grundlegende Informationen die Motion Control Application Note Nr. 54 (MCX-Motion-Control-Grundlagen).

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 49 - 56**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Betriebszustand

Ein Verbund befindet sich zu jedem Zeitpunkt in einem, und nur einem, klar definierten Betriebszustand (`OperatingState`). Ist der Verbund beispielsweise im Betriebszustand **Läuft**, dann ist der Betriebszustand **Freigegeben** nicht aktiv, obwohl natürlich alle Verbundachsen eine Impulsfreigabe haben.

In einem bestimmten Zustand ist immer nur eine Untermenge aller Funktionsaufrufe zulässig. Das Zustandsmodell für die Achse sieht folgendermaßen aus:

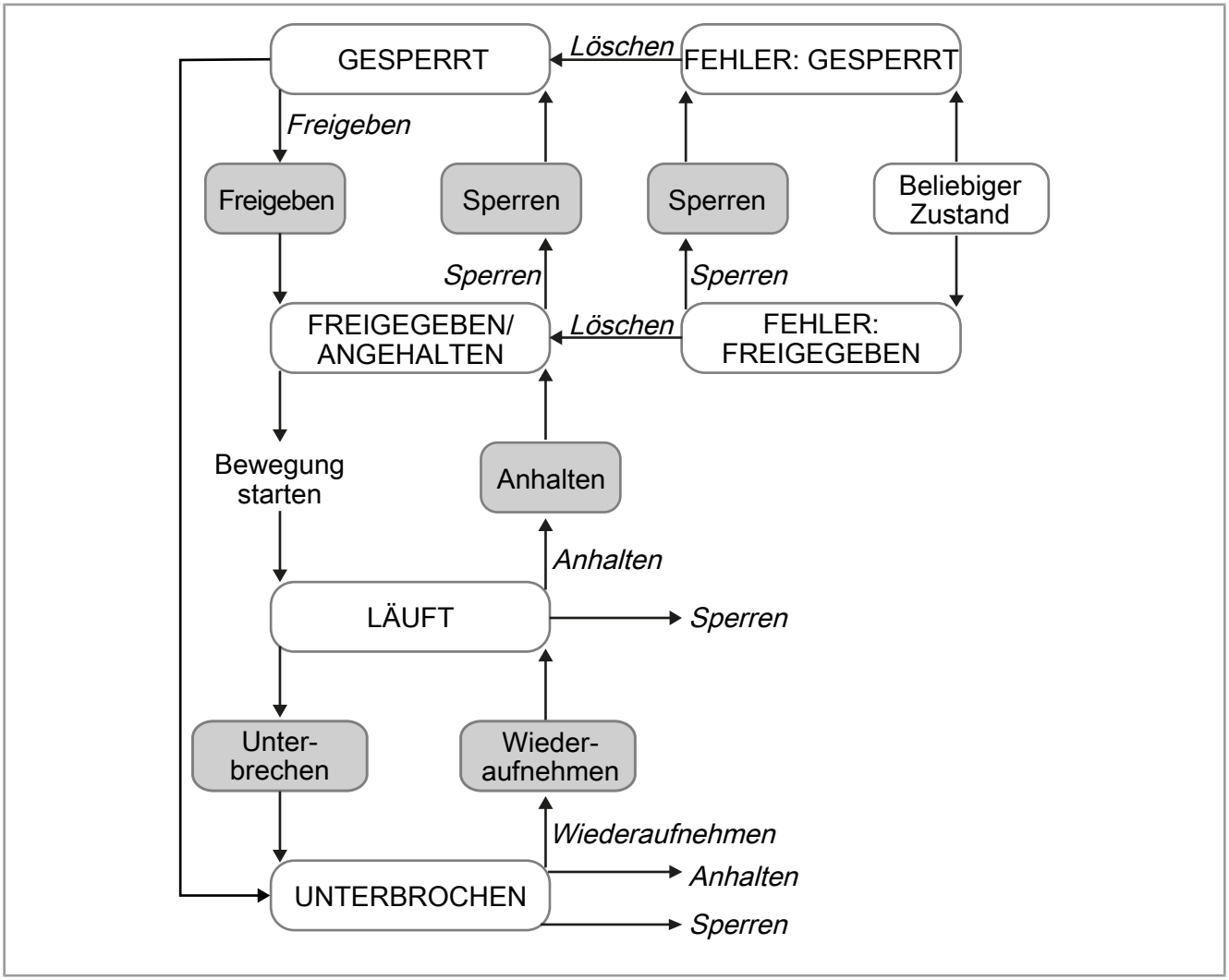


Abb. 15: Zustandsamodell für MCX-Verbunde

Betriebszustand	Bedeutung
Inactive (Inaktiv)	Der Verbund kann keine Befehle entgegen nehmen, z. B. weil er nicht aktiviert wurde.
Disabled (Gesperrt)	– Die Regler des Servoverstärkers aller Verbundachsen sind geöffnet (keine Impulsfreigabe). – Es wird kein Drehmoment erzeugt.
Enabled (Freigegeben)	– Die Regler aller Verbundachsen sind geschlossen (Impulsfreigabe). – Der Sollwertinterpolator befindet sich im Stillstand.
Running (Läuft)	Mindestens eine Achse ist in Bewegung; d. h. der Sollwertinterpolator gibt Bewegungssollwerte an den Servoverstärker weiter.
Suspended (Unterbrochen)	– Entspricht dem Betriebszustand **Freigegeben** mit der Ausnahme, dass die Zielposition erhalten bleibt. Eine mit `MCTechno.State.Transitions.Suspend()` unterbrochene Bewegung kann über den Befehl `MCTechno.State.Transitions.Resume()` (Zustandsübergang wiederaufnehmen) fortgesetzt werden.
EnabledError (Fehler: freigegeben)	Es ist ein Fehler aufgetreten. Nach der Fehlerreaktion bleiben alle Achsen weiterhin in Regelung (Fehlerstufe 4).
DisabledError (Fehler: gesperrt)	Es ist ein Fehler aufgetreten. Nach der Fehlerreaktion ist die Regelung aller Achsen gesperrt (Fehlerstufen 5 und 6).

Tab. 10: Betriebszustände von Technologieverbunden

Betriebsart

Die Betriebsart (`OperatingMode`) gibt an, auf welche Weise der Technologieverbund grundsätzlich arbeitet. Es gibt 3 Betriebsarten:

Betriebsart	Bedeutung
`Inactive` (inaktiv)	Der Verbund kann keine Befehle entgegen nehmen, z. B. weil er nicht aktiviert wurde.
`Normal` (normal)	Der Verbund ist verwendbar. Verfahrbefehle werden stets verdrängend ausgeführt, da ein gepufferter Betrieb bei Technologieverbunden nicht möglich ist.
`Error` (Fehler)	Es liegt ein Bedienfehler vor. Der Betriebszustand bleibt erhalten, Es können jedoch keine neuen Funktionen und Parameter übergeben werden.

Tab. 11: Betriebsarten von MCX-Technologieverbunden

Bewegungsart

Die Bewegungsart (`TypeOfMotion`) gibt an, welche Art der Bewegung der Sollwertgenerator für den Technologieverbund aktuell erzeugt.

Die Abfrage der Bewegungsart ist beispielsweise in einem Überwachungstask interessant, in dem der Prozess nicht bekannt ist. Ausgehend von der aktiven Bewegungsart vor dem Abschalten ist es möglich, das Wiederanlaufen korrekt zu steuern.

Betriebszustand	Bedeutung
Keine Bewegung (`MoveNone`)	Der Sollwertgenerator gibt Stillstand aus, weil keine Bewegung angefordert ist.
Positionierung (`MovePtp`)	Die Leitachse des Technologieverbundes führt aktuell, oder führte zuletzt, eine Punkt-zu-Punkt-Bewegung aus. Auch nach dem ersten Aktivieren des Verbundes ist diese Bewegungsart aktiv. Die Bewegungsart einer Folgeachse, die unabhängig von der Leitachse gestartet wird, wird nicht berücksichtigt.
Endlosbewegung (`MoveVelocity`)	Die Leitachse des Technologieverbundes führt aktuell, oder führte zuletzt eine Endlosbewegung aus. Die Bewegungsart einer Folgeachse, die unabhängig von der Leitachse gestartet wird, wird nicht berücksichtigt.

84.1 Syntax für MCTechno.State

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.State.<Zugriff auf Wert>
```

84.2 Eigenschaften von MCTechno.State

Die Funktionalität bietet folgende Eigenschaften:

- Value
- IsDisabled
- IsEnabled
- IsRunning
- IsSupended
- IsDisabledError
- IsEnabledError

– IsInactive
– OperatingMode
– TypeOfMotion
– Transitions (Beschreibung siehe MCTechno.State.Transitions [▶ 512])

84.2.1 Value

Beschreibung

Datentyp	Bedeutung
OperatingStates	Liest den Betriebszustand, in dem sich der Technologieverbund befindet (für eine detaillierte Beschreibung siehe Abschnitt *Betriebszustand* in MCGeo.State.Transitions [▶ 400]).
OperatingStates.Inactive	Inaktiv
OperatingStates.Disabled	Gesperrt
OperatingStates.Enabled	Freigegeben
OperatingStates.Running	Läuft
OperatingStates.Suspended	Unterbrochen
OperatingStates.EnabledError	Fehler: freigegeben
OperatingStates.DisabledError	Fehler: gesperrt

Syntax

```
OpState := tecGroup.State.Value;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

84.2.2 IsDisabled

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund im Betriebszustand **Gesperrt** befindet.

Syntax

```
When techgroup.State.IsDisabled Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

84.2.3 IsDisabledError

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund im Betriebszustand **Fehler: gesperrt** befindet.

Syntax

```
When tecGroup.State.IsDisabledError Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

84.2.4 IsEnabled

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich Verbund im Betriebszustand **Freigegeben** befindet.

Syntax

```
When tecGroup.State.IsEnabled Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

84.2.5 IsEnabledError

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund im Betriebszustand **Fehler: freigegeben** befindet.

Syntax

```
When tecGroup.State.IsEnabledError Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

84.2.6 IsInactive

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund im Betriebszustand **Inaktiv** befindet.

Syntax

```
When tecGroup.State.IsInactive Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

84.2.7 IsRunning

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund im Betriebszustand **Läuft** befindet.

Syntax

```
When tecGroup.State.IsRunning Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

84.2.8 IsSuspended

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund im Betriebszustand **Unterbrochen** befindet.

Syntax

```
When tecGroup.State.IsSuspended Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

84.2.9 OperatingMode.Value

Beschreibung

Datentyp	Bedeutung
Typ MCOperatingModes	Liest die Betriebsart, in der sich der Verbund befindet (für eine detaillierte Beschreibung siehe Abschnitt Betriebsart [▶ 400]).
MCOperatingModes.Inactive	Der Verbund ist inaktiv, z. B. weil er nicht aktiviert wurde.
MCOperatingModes.Normal	Der Verbund wurde aktiviert und befindet sich nicht im Fehlerzustand.
MCOperatingModes.Error	Der Verbund befindet sich nach einem Bedienfehler im Fehlerzustand.

Syntax

```
OpModeAchse := tecGroup.State.OperatingMode.Value;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

OperatingMode.IsInInactiveMode

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich die Achse in der Betriebsart *Inaktiv* befindet.

Syntax

```
When tecGroup.State.OperatingMode.IsInInactiveMode Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

OperatingMode.IsInErrorMode

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund in der Betriebsart *Fehler* befindet.

Syntax

```
When tecGroup.State.OperatingMode.IsInErrorMode Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

OperatingMode.IsInNormalMode

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund in der Betriebsart *Normal* befindet.

Syntax

```
When tecGroup.State.OperatingMode.IsInNormalMode Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

84.2.10 TypeOfMotion.Value

Beschreibung

Betriebszustand	Bedeutung
Typ MCTechnoTypesOfMotion	Bewegungsart in der sich der Technologieverbund gerade befindet (für eine detaillierte Beschreibung siehe Abschnitt MC-Techno.State [▶ 505].
MCTechnoTypesOfMotion.MoveNone	Bewegungsart `Keine`: Der Verbund ist nicht aktiv.
MCGeoTypesOfMotion.MovePtp	Die Bewegungsart `Positionierung` wird nach dem Aktivieren des Technologieverbundes solange angezeigt, bis eine andere Bewegungsart ausgeführt wird.
MCGeoTypesOfMotion.MoveVelocity	Die Bewegungsart `Endlos` wird solange angezeigt, bis vom Technologieverbund eine andere Bewegungsart ausgeführt wird.

Syntax

```
TypeOfMotion := tecGroup.State. TypeOfMotion.Value;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveNone

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob sich der Verbund in der Bewegungsart `Keine` befindet.

Syntax

```
When tecGroup.State.TypeOfMotion.IsMoveLinear Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMovePtp

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob der Verbund aktuell eine `Positionierbewegung` ausführt.

Syntax

```
When tecGroup.State.TypeOfMotion.IsMovePtp Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

TypeOfMotion.IsMoveVelocity

Beschreibung

Datentyp	Bedeutung
Bool	Fragt ab, ob der Verbund aktuell eine `Endlosbewegung` ausführt.

Syntax

```
When tecGroup.State.TypeOfMotion.IsMoveVelocity Continue;
```

Zustandsverwaltung

Die Eigenschaft dient zu Diagnosezwecken. Auf sie kann nur lesend zugegriffen werden.

Das Lesen der Eigenschaft ist in jedem Zustand möglich.

85 MCTechno.State.Transitions

Mit der Funktionalität **MCTechno.State.Transitions** können Sie für spezielle Anwendungen Zustandsübergänge eines Bahnverbundes steuern.

In der Regel werden die Zustandsübergänge eines Verbundes durch die Bewegungsbefehle und das Verhalten der Achsbewegung automatisch ausgelöst. Nur in Ausnahmefällen – z. B. zur Reaktion im Fehlerfall – ist die direkte Steuerung der Zustandsverwaltung erforderlich. Durch die Funktionalität **MCTechno.State.Transitions**, wie auch `Diagnostics.ClearErrors()`, können Betriebszustandsübergänge explizit ausgelöst werden.

Mit der Funktionalität **MCTechno.State** können Sie verschiedene Informationen eines Technologieverbundes abfragen, wie z. B. Betriebsart, Bewegungsart etc.

Grundlegende und weiterführende Informationen

Beachten Sie in diesem Zusammenhang ebenso folgende Informationen:

– Für eine Beschreibung der übrigen Funktionalitäten von MCTechno.State den Abschnitt MCTechno.State [▶ 503];

– Für eine detaillierte Beschreibung des Aufbaus des MCX-Kerns die JetSym–Onlinehilfe (JetSym Motion und MotionControl > JetSym MotionControl > MotionControl-Grundlagen > Bedienkonzept > Überblick: Bedienkonzept)

– Für grundlegende Informationen die Motion Control Application Note Nr. 54 (MCX-Motion-Control-Grundlagen).

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 49 - 56**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

Betriebszustand

Ein Verbund befindet sich zu jedem Zeitpunkt in einem, und nur einem, klar definierten Betriebszustand (`OperatingState`). Ist der Verbund beispielsweise im Betriebszustand **Läuft**, dann ist der Betriebszustand **Freigegeben** nicht aktiv, obwohl natürlich alle Verbundachsen eine Impulsfreigabe haben.

In einem bestimmten Zustand ist immer nur eine Untermenge aller Funktionsaufrufe zulässig. Das Zustandsmodell für die Achse sieht folgendermaßen aus:

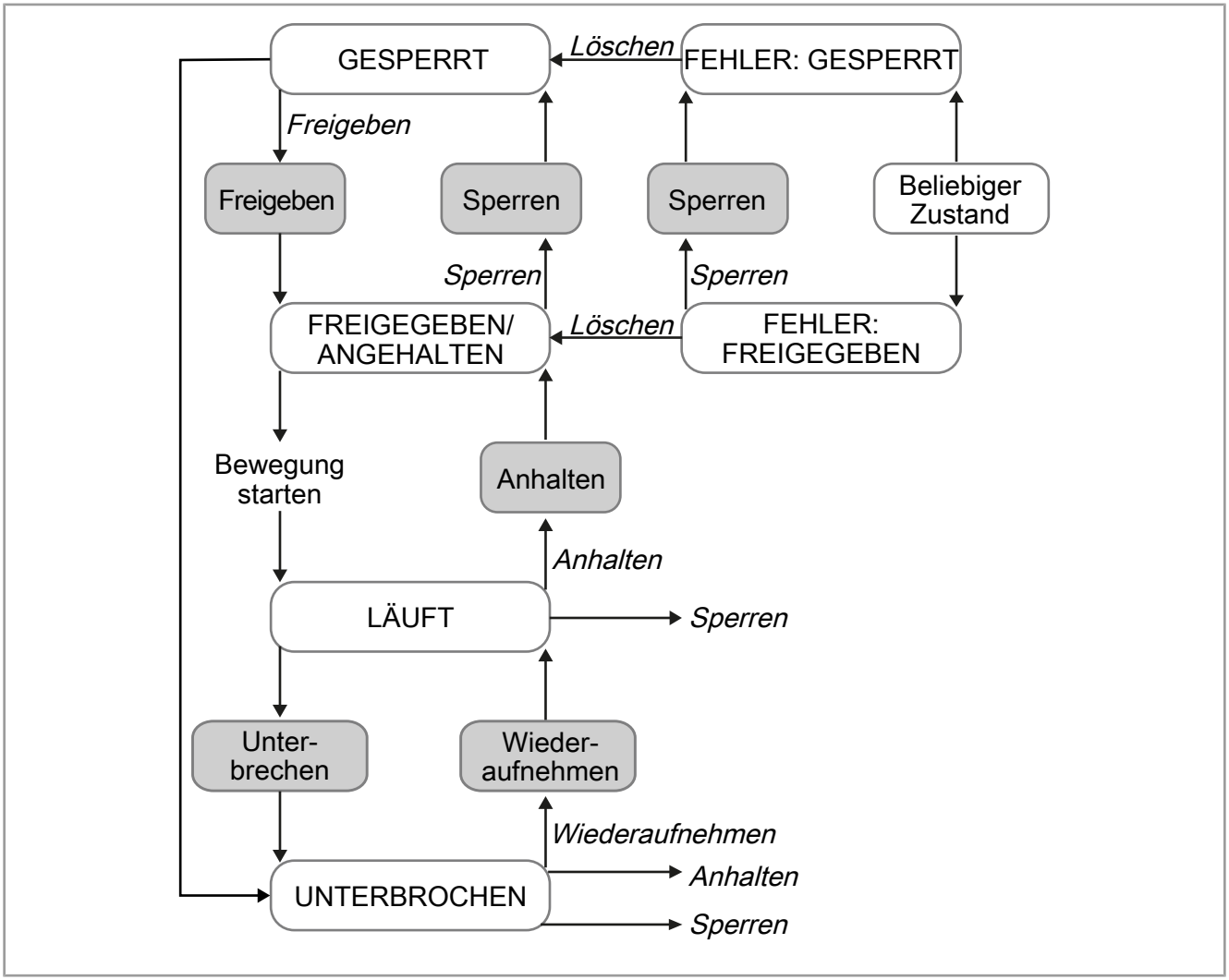


Abb. 16: Zustandsamodell für MCX-Verbunde

Betriebszustand	Bedeutung
Inactive (Inaktiv)	Der Verbund kann keine Befehle entgegen nehmen, z. B. weil er nicht aktiviert wurde.
Disabled (Gesperrt)	– Die Regler des Servoverstärkers aller Verbundachsen sind geöffnet (keine Impulsfreigabe). – Es wird kein Drehmoment erzeugt.
Enabled (Freigegeben)	– Die Regler aller Verbundachsen sind geschlossen (Impulsfreigabe). – Der Sollwertinterpolator befindet sich im Stillstand.
Running (Läuft)	Mindestens eine Achse ist in Bewegung; d. h. der Sollwertinterpolator gibt Bewegungssollwerte an den Servoverstärker weiter.
Suspended (Unterbrochen)	– Entspricht dem Betriebszustand **Freigegeben** mit der Ausnahme, dass die Zielposition erhalten bleibt. Eine mit `MCTechno.State.Transitions.Suspend()` unterbrochene Bewegung kann über den Befehl `MCTechno.State.Transitions.Resume()` (Zustandsübergang wiederaufnehmen) fortgesetzt werden.
EnabledError (Fehler: freigegeben)	Es ist ein Fehler aufgetreten. Nach der Fehlerreaktion bleiben alle Achsen weiterhin in Regelung (Fehlerstufe 4).
DisabledError (Fehler: gesperrt)	Es ist ein Fehler aufgetreten. Nach der Fehlerreaktion ist die Regelung aller Achsen gesperrt (Fehlerstufen 5 und 6).

Tab. 12: Betriebszustände von Technologieverbunden

Betriebsart

Die Betriebsart (`OperatingMode`) gibt an, auf welche Weise der Technologieverbund grundsätzlich arbeitet. Es gibt 3 Betriebsarten:

Betriebsart	Bedeutung
`Inactive` (inaktiv)	Der Verbund kann keine Befehle entgegen nehmen, z. B. weil er nicht aktiviert wurde.
`Normal` (normal)	Der Verbund ist verwendbar. Verfahrbefehle werden stets verdrängend ausgeführt, da ein gepufferter Betrieb bei Technologieverbunden nicht möglich ist.
`Error` (Fehler)	Es liegt ein Bedienfehler vor. Der Betriebszustand bleibt erhalten, Es können jedoch keine neuen Funktionen und Parameter übergeben werden.

Tab. 13: Betriebsarten von MCX-Technologieverbunden

Bewegungsart

Die Bewegungsart (`TypeOfMotion`) gibt an, welche Art der Bewegung der Sollwertgenerator für den Technologieverbund aktuell erzeugt.

Die Abfrage der Bewegungsart ist beispielsweise in einem Überwachungstask interessant, in dem der Prozess nicht bekannt ist. Ausgehend von der aktiven Bewegungsart vor dem Abschalten ist es möglich, das Wiederanlaufen korrekt zu steuern.

Betriebszustand	Bedeutung
Keine Bewegung (`MoveNone`)	Der Sollwertgenerator gibt Stillstand aus, weil keine Bewegung angefordert ist.
Positionierung (`MovePtp`)	Die Leitachse des Technologieverbundes führt aktuell, oder führte zuletzt, eine Punkt-zu-Punkt-Bewegung aus. Auch nach dem ersten Aktivieren des Verbundes ist diese Bewegungsart aktiv. Die Bewegungsart einer Folgeachse, die unabhängig von der Leitachse gestartet wird, wird nicht berücksichtigt.
Endlosbewegung (`MoveVelocity`)	Die Leitachse des Technologieverbundes führt aktuell, oder führte zuletzt eine Endlosbewegung aus. Die Bewegungsart einer Folgeachse, die unabhängig von der Leitachse gestartet wird, wird nicht berücksichtigt.

85.1 Syntax für MCTechno.State.Transitions

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.State.Transitions.<Funktion>(optional <Parameter>);
```

85.2 Funktionen von MCTechno.State.Transitions

Die Funktionalität bietet folgende Funktionen:

- Enable()
- Disable()
- Stop()
- ErrorDisable()
- ErrorStop()
- ErrorLockUserInput()
- Suspend()
- Resume()

85.2.1 Enable()

Die Funktion `State.Transistions.Enable()` ist identisch zur Funktion `MCTechno.Power.Enable()`.Eine detaillierte Beschreibung der Funktion finden Sie im Abschnitt MCTechno.Power.Enable() [▶ 496].

Syntax

```
tecGroup.State.Transitions.Enable();
```

85.2.2 Disable()

Die Funktion führt je nach übergebenem Parameter (siehe nachfolgende Tabelle) dieselbe Aktion aus, wie die Befehle `MCTechno.Power.Disable()` und `MCTechno.Power.Qickstop()`. Entnehmen Sie die Details den Beschreibungen zu MCTechno.Power [▶ 496].

Übergabeparameter

Parameter	Datentyp	Bedeutung
Mode	MCStateTransitionDisableModes	Modus für die Ausführung der `Disable()`-Funktion
	MCStateTransitionDisableModes.ImmediatePulseBlocking	Führt die gleiche Funktion aus wie der Befehl `MCTechno.Power.Disable()` bzw. `MCTechno MCTechno.Power.Disable(MCPowerDisableModes.Forced)`
	MCStateTransitionDisableModes.VelocityControlSlope	Führt die gleiche Funktion aus wie der Befehl `MCTechno.Power.QuickStop()`.

Syntax

```
tecGroup.State.Transitions.Disable(MCStateTransitionDisableModes.Immediate-
PulseBlocking);
tecGroup.State.Transitions.Disable(MCStateTransitionDisableModes.Velocity-
ControlSlope);
```

85.2.3 Stop()

Beschreibung

Die Funktion führt je nach dem übergebenen Parameter (siehe Tabelle unten) dieselbe Aktion aus wie der Befehl MCGeo.Stop() [▶ 411].

Der Unterschied besteht darin, dass die Übergabe eines Funktionsparameters mit `MCGeo.State.Transition.Stop()` zwingend erforderlich ist, während dies mit `MCGeo.Stop()` optional ist (Details siehe dort).

Übergabeparameter

Parameter	Datentyp	Bedeutung
Mode	MCStateTransitionStopModes	Es stehen zwei Verzögerungswerte zur Verfügung.
	StopModes.ProgDeceleration	Zum Anhalten wird die zuletzt für den Verbund vorgegebene Verzögerung verwendet.
	StopModes.MaxDeceleration	Zum Anhalten wird die maximale Beschleunigung des Verbunds als Verzögerung verwendet. Diese wurde entweder bei der Inbetriebnahme im Motion Setup unter „Maximale Verzögerung“ vorgegeben oder mit dem Befehl `geoGroup.Mechanism.Limitations.MaxAcceleration` programmiert.

Syntax

```
geoGroup.State.Transitions.Stop(MCStateTransitionStopModes.ProgDecelerati-
on);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während der Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√		Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	√		Freigegeben
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Zustandsüberwachung

Für die Überwachung der Positionierung stehen Ihnen verschiedene Informationen zur Verfügung.

– Der **Zustand der Achse** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder die Achse angehalten ist.

```
When geoGroup.State.IsEnabled Continue; //Der Verbund befindet sich in
Lageregelung und steht.
When geoGroup.State.IsRunning Continue; //Der Sollwertgenerator mindestens
einer Verbundachse erzeugt eine Bewegung.
```

– Der **Rampenstatus** informiert, in welchem Bewegungsabschnitt sich der Sollwertgenerator gerade befindet.

```
When geoGroup.Mechanism.Slope.IsAccelerating Continue;
```

```
When geoGroup.Mechanism.Slope.IsAtConstantSpeed Continue;
When geoGroup.Mechanism.Slope.IsDecelerating Continue;
When geoGroup.Mechanism.Slope.IsStopped Continue;
```

– Der **Zielfensterstatus** wird von dieser Funktion nicht beeinflusst.

85.2.4 ErrorDisable()

Mit der Funktion lösen Sie einen Fehler im Bahnverbund aus, der zum Zustand **Fehler: gesperrt** führt. Die Fehlerreaktion kann eingestellt werden (siehe Eigenschaft `ErrorDisableMode`).

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
tecGroup.State.Transitions.ErrorDisable();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Fehler: gesperrt
Freigegeben	√	Fehler: gesperrt
Läuft	√	Fehler: gesperrt
Unterbrochen	√	Fehler: gesperrt
Fehler: freigegeben	√	Fehler: gesperrt
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

85.2.5 ErrorStop()

Die Funktion lösen Sie einen Fehler im Verbund aus, der zum Zustand **Fehler: freigegeben** führt. Die Fehlerreaktion kann eingestellt werden (siehe Eigenschaft ErrorStopMode [▶ 521]).

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
tecGroup.State.Transitions.ErrorStop();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Fehler: freigegeben
Freigegeben	√	Fehler: freigegeben

Ausgangszustand	Erlaubt	Endzustand
Läuft	√	Fehler: freigegeben
Unterbrochen	√	Fehler: freigegeben
Fehler: freigegeben	-	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

85.2.6 ErrorLockUserInput()

Mit der Funktion lösen Sie einen Bedienfehler im Verbund aus.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
tecGroup.State.Transitions.ErrorLockUserInput();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	√	Gesperrt
Freigegeben	√	Freigegeben
Läuft	√	Läuft
Unterbrochen	√	Unterbrochen
Fehler: freigegeben	–	Fehler: freigegeben
Fehler: gesperrt	–	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

85.2.7 Suspend()

Mit der Funktion unterbrechen Sie eine laufende Verbundbewegung. Das Anhalten geschieht entweder mit der zuletzt programmierten Verzögerungsrampe oder mit der konfigurierten Maximalverzögerung. Der Zustand nach dem Anhalten ist nicht **Freigegeben** sondern **Unterbrochen**. Mit der Funktion `Resume()` kann die unterbrochene Bahnbewegung eines Verbunds wiederaufgenommen und an die ursprüngliche Zielposition (der Leitachse) gefahren werden.

Kann die Leitachse mit der vorgegebenen Verzögerung nicht mehr vor der eigentlichen Zielposition angehalten werden, so wird der Befehl ignoriert und die Positionierung zu Ende geführt. Der Zustand ist danach **Freigegeben** und nicht **Unterbrochen**.

Übergabeparameter

Parameter	Datentyp	Bedeutung
Mode	MCStateTransitionStopModes	Modus für die Verzögerungsrampe der `Suspend()`-Funktion
	MCStateTransitionStopModes.ProgDeceleration	Verwendet die für die Leitachse zuletzt programmierte Verzögerung.
	MCStateTransitionStopModes.MaxDeceleration	Verwendet die für die Leitachse konfigurierte maximale Verzögerung.

Syntax

```
tecGroup.State.Transitions.Suspend(MCStateTransitionStopModes.ProgDecelera-
tion);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	X	Freigegeben
Läuft	√	Unterbrochen
Unterbrochen	–	Unterbrochen
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

85.2.8 Resume()

Die Funktion führt eine mit der Funktion Suspend() [▶ 518] unterbrochene Positionierung fort. Es wird die ursprünglich vorgegebene Zielposition der Leitachse angefahren. Auch die Geschwindigkeits- und Beschleunigungswerte werden aus der ursprünglichen Positionierung übernommen.

Die Funktion `Resume()` wirkt sich nur aus, wenn sich der Verbund im Zustand **Unterbrochen** befindet.

Übergabeparameter

Die Funktion besitzt keine Übergabeparameter.

Syntax

```
tecGroup.State.Transitions.Resume();
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Endzustand
Gesperrt	X	Gesperrt
Freigegeben	X	Freigegeben
Läuft	–	Läuft
Unterbrochen	√	Läuft
Fehler: freigegeben	X	Fehler: freigegeben
Fehler: gesperrt	X	Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

85.3 Eigenschaften von MCTechno.State.Transitions

Die Funktionalität bietet folgende Eigenschaften:

– ErrorDisableMode
– ErrorStopMode
– ErrorLevel

85.3.1 ErrorDisableMode

Beschreibung

Datentyp	Bedeutung
MCStateTransitionDisableModes	Fehlerreaktion, die beim Auslösen der Funktion `ErrorDisable()` verwendet wird.
MCStateTransitionDisableModes.VelocityControlSlope	Hält den Verbund mit der Schnellhaltverzögerung an. Ist der Stillstand erreicht, wird der Servoverstärker gesperrt.
MCStateTransitionDisableModes.ImmediatePulseBlocking	Sperrt den Servoverstärker sofort für die Drehmomenterzeugung (Impulssperre).

Syntax

```
tecGroup.State.Transitions.ErrorDisableMode := MCStateTransitionDisableModes.ImmediatePulseBlocking;
Fehlerreaktion := tecGroup.State.Transitions.ErrorDisableMode;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

Bei Lesezugriff der Eigenschaft wird die aktuell gültige Fehlerreaktion für den Befehl zurückgegeben.

85.3.2 ErrorStopMode

Beschreibung

Datentyp	Bedeutung
MCStateTransitionErrorStopModes	Fehlerreaktion, die beim Auslösen der Funktion `ErrorStop()` verwendet wird.
MCStateTransitionErrorStopModes.ProgDeceleration	Hält den Verbund mit der programmierten Verzögerung an.
MCStateTransitionErrorStopModes.MaxDeceleration	Hält den Verbund mit der maximalen Verzögerung an.
MCStateTransitionErrorStopModes.AtBlockEnd	Dieser Modus hat für den Technologieverbund keine Bedeutung, da der gepufferte Betrieb mit dem Technologieverbund nicht möglich ist.

Syntax

```
tecGroup.State.Transitions.ErrorStopMode := MCStateTransitionStopModes.Max-
Deceleration;
Fehlerreaktion := tecGroup.State.Transitions.ErrorStopMode;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

85.3.3 ErrorLevel

Beschreibung

Mit der Eigenschaft stellen Sie die Fehlerstufe ein, die der Bahnverbund im Falle eines Drive-Fehlers einnimmt. Dies ist standardmäßig Fehlerstufe 4 (Betriebs-HALT und Betriebszustand **Fehler: freigegeben**).

Datentyp	Bedeutung
MCStateTransitionErrorLevels	Fehlerstufe des Bahnverbundes bei einem Drive-Fehler. Ausführliche Informationen zu den Unterschieden der Fehlerreaktionsstufen finden Sie in der JetSym-Onlinehilfe unter der Rubrik *Fehlerverwaltung*.
MCStateTransitionErrorLevels.Message	Der Verbund reagiert mit der Fehlerstufe 1: Meldung/Warnung wird erstellt.
MCStateTransitionErrorLevels.ChannelStop	Der Verbund reagiert mit der Fehlerstufe 2: Befehlskanal wird gesperrt.
MCStateTransitionErrorLevels.CannelStopClearBuffer	Ausnahme für Technologieverbunde: Identische Funktion wie Fehlerreaktionsstufe 2, da ein gepufferter Betrieb bei Technologieverbunden nicht möglich ist.
MCStateTransitionErrorLevels.Stop	Der Verbund reagiert mit der Fehlerstufe 4: Betriebs-HALT wird ausgeführt (Standard).
MCStateTransitionErrorLevels.Quickstop (Default)	Der Verbund reagiert mit der Fehlerstufe 5: Not-HALT wird ausgeführt.
MCStateTransitionErrorLevels.PulsBlocking	Der Verbund reagiert mit der Fehlerstufe 6: Regeler wird gesperrt (Impulssperre).

Syntax

```
tecGroup.State.Transitions.ErrorLevel := MCStateTransitionErrorLevels.Puls-
Blocking;
ReaktionBeiDriveFehler := tecGroup.State.Transitions.ErrorLevel;
```

Zustandsverwaltung

Das Lesen und Schreiben der Eigenschaft ist in jedem Zustand möglich.

86 MCTechno.Stop

Mit der Funktionalität **MCTechno.Stop** können Sie einen Technologieverbund, der sich in Bewegung befindet, mit einer definierten Rampe anhalten.

Im Unterschied zur Funktion `MoveHalt.Start()` wirkt sich die `Stop`-Funktion immer auf den gesamten Verbund aus.

Die Funktionalität besitzt keine weiteren Funktionen oder Eigenschaften.

ⓘ Info

Weiterführende Informationen

Bitte entnehmen Sie die Abläufe sowie Beispielprojekte für den Umgang mit Verbunden den **Motion Control Application Notes (APN) Nr. 49 - 56**. Diese stehen auf der Website von Bucher Automation zum Download bereit.

86.1 Syntax für MCTechno.Stop

Die Funktionalität geben Sie im JetSym-Editor am Einfachsten wie folgt über IntelliSense ein:

```
<Verbundbezeichnung>.Stop(optional <Parameter 1>)
```

86.2 Funktionen von MCTechno.Stop

`MCTechno.Stop` ist eine Funktion des Verbundobjekts.

86.2.1 Stop()

Übergabeparameter

Beachten Sie folgendes Verhalten:

- Die Funktion führt den Anhaltebefehl unmittelbar aus. Der übergebene Parameter (siehe Tabelle) gibt die maximale Verzögerung dafür an.
- Wird kein Parameter übergeben, so wird der zuletzt übergebene `StopMode` verwendet.
- Die Folgeachsen werden beim Anhalten entsprechend ihrer Koppelfunktion synchron zur Leitachse zum Stillstand gebracht.

Parameter	Datentyp	Bedeutung
StopModes	Enum StopModes (optional)	Es stehen zwei Verzögerungswerte zur Verfügung.
	StopModes.ProgDeceleration	Zum Anhalten wird die zuletzt für den Verbund vorgegebene Verzögerung verwendet (Standard).
	StopModes.MaxDeceleration	Zum Anhalten wird die maximale Beschleunigung der Leitachse als Verzögerung verwendet. Diese wurde entweder bei der Inbetriebnahme im Motion Setup unter Maximale Verzögerung vorgegeben oder mit dem Befehl `axLeitachse.Mechanism.Limitations.MaxAcceleration` programmiert.

Syntax

```
tecGroup.Stop(StopModes.ProgDeceleration);
```

```
tecGroup.Stop(StopModes.MaxDeceleration);
```

Zustandsverwaltung

Für das Schreiben der Funktion gilt folgende Tabelle:

Ausgangszustand	Erlaubt	Während der Ausführung	Endzustand
Gesperrt	X		Gesperrt
Freigegeben	√		Freigegeben
Läuft	√	Läuft	Freigegeben
Unterbrochen	√		Freigegeben
Fehler: freigegeben	X		Fehler: freigegeben
Fehler: gesperrt	X		Fehler: gesperrt

Dabei ist:

√ der Zustand, in dem die Funktion aufgerufen werden kann.

– der Zustand, in dem der Aufruf der Funktion nicht empfohlen wird, jedoch keinen Fehler auslöst.

X der Zustand, in dem der Aufruf der Funktion nicht erlaubt ist und einen Fehler auslöst.

Zustandsüberwachung

Für die Überwachung der `Stop()`-Funktion stehen Ihnen verschiedene Informationen zur Verfügung.

– Der **Zustand der Achse** informiert darüber, ob der Sollwertgenerator eine Bewegung generiert oder die Achse angehalten ist.

```
When tecGroup.State.IsEnabled Continue; //Der Verbund befindet sich in
Lageregelung und alle Achsen stehen.
When tecGroup.State.IsRunning Continue; //Der Sollwertgenerator mindestens
einer Verbundachse erzeugt eine Bewegung.
```

– Der **Rampenstatus** informiert, in welchem Bewegungsabschnitt sich der Sollwertgenerator gerade befindet.

```
When tecGroup.Mechanism.Slope.IsAccelerating Continue;
When tecGroup.Mechanism.Slope.IsAtConstantSpeed Continue;
When tecGroup.Mechanism.Slope.IsDecelerating Continue;
When tecGroup.Mechanism.Slope.IsStopped Continue;
```

– Der **Zielfensterstatus** wird von dieser Funktion nicht beeinflusst.

Glossar

Fehler
Benachrichtigungstyp, der das Erreichen einer Fehlerschwelle anzeigt. Fehler lösen eeine Systemreaktion aus, und müssen quittiert werden, um den Betrieb fortzusetzen. Vergleiche dazu Warnungen und Meldung.

Masterachse
Im Zusammenhang mit der Funktionalität MCCoupling Group ist die Masterachse eine überlagerte Achse, die nich tmit der Leitachse zu verwechseln ist, da sich die Funktionen grundlegend unterscheiden.

Meldung
Benachrichtigungstyp des Motion-Control-Systems, der wichtige Informationen enthält oder vor dem Erreichen einer Fehlerschwelle warnt. Meldungen lösen keine Systemreaktion aus, müssen jedoch quittiert werden. Vergleiche dazu Warnungen.

Synchronachse
Achse, die synchron zur Bahnbewegung verfahren werden, ohne dass sie mechanisch an der Bahnbewegung teilnehmen.

Technologieverbund
Ein Technologieverbund ist eine Kopplung von mehreren MCX-Achsen, bei denen eine oder mehrere Folgeachsen in Abhängigkeit der Position einer Leitachse bewegt werden.

Warnung
Benachrichtigungstyp von EtherCAT-Geräten, der vor dem Erreichen einer Fehlerschwelle warnt. Warnungen lösen keine Systemreaktion aus und werden automatisch gelöscht, sobald die Ursache der Warnung nicht mehr vorliegt. Vergleiche dazu Meldungen.

Bucher Automation AG
Thomas-Alva-Edison-Ring 10
71672 Marbach am Neckar, Deutschland
T +49 7141 2550-0
info@bucherautomation.com

www.bucherautomation.com