Electric Drives and Controls | Hydraulics | Linear Motion and Assembly Technologies | Pneumatics | Service

Rexroth
Bosch Group

Rexroth IndraLogic XLC
IndraMotion MLC 13VRS
Generic Application Template

R911336369
Ausgabe 02

Anwendungsbeschreibung



Titel
Rexroth IndraLogic XLC
IndraMotion MLC 13VRS
Generic Application Template

Art der Dokumentation
Anwendungsbeschreibung

Dokumentations-Type
DOK-XLCMLC-TF*GAT**V13-AP02-DE-P

Interner Ablagevermerk
RS-cdff34e18f0001620a6846a000296caa-2-de-DE-17

Zweck der Dokumentation
Dieses Handbuch stellt dem IndraLogic SPS-Programmierer ein strukturiertes Template zum Hinzufügen und Bearbeiten von SPS-Programmcode zur Verfügung. Beschrieben werden das Template selbst, der Wizard und Beispielapplikationen.

Änderungsverlauf

Ausgabe	Stand	Bemerkung
Ausgabe 01	04.2012	Erstausgabe
Ausgabe 02	03.2013	Änderungen eingepflegt

Schutzvermerk
© Bosch Rexroth AG 2013
Alle Rechte bei Bosch Rexroth AG, auch für den Fall von Schutzrechtsanmeldungen. Jede Verfügungsbefugnis, wie Kopier- und Weitergaberecht, bei uns.

Verbindlichkeit
Die angegebenen Daten dienen allein der Produktbeschreibung und sind nicht als zugesicherte Eigenschaften im Rechtssinne zu verstehen. Änderungen im Inhalt der Dokumentation und Liefermöglichkeiten der Produkte sind vorbehalten.

Herausgeber
Bosch Rexroth AG
Bgm.-Dr.-Nebel-Str. 2 ■ D-97816 Lohr a. Main
Telefon +49 9352 18 0 ■ Fax +49 9352 18 8400
http://www.boschrexroth.com/
Systementwicklung Automation Motion Logic Control, AK (FrWe)

Hinweis
Diese Dokumentation ist auf chlorfrei gebleichtem Papier gedruckt.

Inhaltsverzeichnis

Seite

1 Zu dieser Dokumentation ... 11
1.1 Gültigkeit der Dokumentation ... 11
1.2 Gliederung der Dokumentation ... 11
1.3 Erforderliche und ergänzende Dokumentationen ... 12
1.4 Darstellung von Informationen ... 17
1.4.1 Sicherheitshinweise ... 17
1.4.2 Verwendete Symbole ... 18
1.4.3 Bezeichnungen und Abkürzungen ... 18

2 Wichtige Gebrauchshinweise ... 21
2.1 Bestimmungsgemäßer Gebrauch ... 21
2.1.1 Einführung ... 21
2.1.2 Einsatz- und Anwendungsbereiche ... 21
2.2 Nicht bestimmungsgemäßer Gebrauch ... 22

3 Sicherheitshinweise für elektrische Antriebe und Steuerungen ... 25
3.1 Begriffsdefinitionen ... 25
3.2 Grundsätzliche Hinweise ... 26
3.2.1 Benutzung und Weitergabe der Sicherheitshinweise ... 26
3.2.2 Voraussetzungen für den sicheren Gebrauch ... 26
3.2.3 Gefahren durch falschen Gebrauch ... 28
3.3 Gefahrenbezogene Hinweise ... 28
3.3.1 Schutz gegen Berühren elektrischer Teile und von Gehäusen ... 28
3.3.2 Schutzkleinspannung als Schutz gegen elektrischen Schlag ... 30
3.3.3 Schutz vor gefährlichen Bewegungen ... 30
3.3.4 Schutz vor magnetischen und elektromagnetischen Feldern bei Betrieb und Montage ... 32
3.3.5 Schutz gegen Berühren heißer Teile ... 32
3.3.6 Schutz bei Handhabung und Montage ... 33
3.3.7 Schutz beim Umgang mit Batterien ... 33
3.3.8 Schutz vor unter Druck stehenden Leitungen ... 34
3.4 Erläuterung der Signalwörter und der Signalgrafik ... 34

4 Einführung ... 37

5 Automatisierung von Applikationen ... 39
5.1 Allgemeines ... 39
5.2 Initialisierung ... 39
5.3 Betriebsarten ... 40
5.3.1 Allgemeines ... 40
5.3.2 Automatik ... 40
5.3.3 Manual ... 40
5.4 Zyklische Bearbeitung ... 40
5.4.1 Allgemeines ... 40

Inhaltsverzeichnis

Seite

5.4.2 Betriebsartenunabhängige Logik ... 40
5.4.3 Generierung und Anzeige von Diagnosemeldungen ... 40
5.5 Taskhandling ... 41
5.5.1 Allgemeines ... 41
5.5.2 Grundsätze des Taskhandlings ... 41
5.5.3 Empfohlene Taskaufteilung bei Verwendung von IndraWorks/IndraLogic ... 41
Allgemeines ... 41
Typische Aufgaben MotionTask ... 42
Typische Aufgaben PlcTask ... 42
Typische Aufgaben BackgroundTask ... 42
5.5.4 Messung der Zykluszeiten ... 43
Zykluszeitendialog der Steuerung ... 43
Task Execution Viewer der Steuerung ... 44
Zeitmessung im SPS-Projekt ... 44
5.5.5 Applikationsspezifischer Watchdog ... 45
5.6 Fehlerhandling ... 46
5.6.1 Allgemeines ... 46
5.6.2 Grundlagen des Fehlerhandling ... 46
5.6.3 Fehlererkennung ... 46
5.6.4 Fehlerreaktion ... 47
5.6.5 Fehlerdiagnose ... 48
5.6.6 Fehlerquittierung/Wiederanlauf nach Fehler ... 49
5.7 Visualisierung ... 49
5.7.1 Allgemeines ... 49
5.7.2 Typische Inhalte einer Visualisierung ... 49
Anzeige des Maschinen-Status ... 49
Steuern der Maschinenzustände ... 50
5.7.3 Bereitstellung der Daten durch die SPS ... 50
Allgemeines ... 50
Symbolkonfiguration ... 50
Kommunikations-Konfiguration ... 53

6 ML_TechInterface.library ... 55
6.1 Einführung und Übersicht ... 55
6.2 TechInterface Erst-Konfiguration ... 62
6.3 Achs-Interface Funktionsbausteine ... 66
6.3.1 MB_AxisInitType01 ... 66
6.3.2 MB_AxisInterfaceType01 ... 68
6.3.3 MB_AxisInterfaceType02 ... 71
6.4 Achsbetriebsarten ... 73
6.4.1 Dialogbasierte Achskommandierung ... 73
6.4.2 Überblick ... 75
6.4.3 Antrieb Bereit ... 77
6.4.4 Antrieb Halt ... 78
6.4.5 Antrieb Stopp ... 78
6.4.6 Antrieb Referenzieren ... 79

Inhaltsverzeichnis

Seite

6.4.7 Absolutes Positionieren ... 79
6.4.8 Relatives Positionieren ... 80
6.4.9 Additives Positionieren ... 81
6.4.10 Geschwindigkeits-Steuerung ... 81
6.4.11 Betriebsart "Winkelsynchronisation" ... 82
6.4.12 Betriebsart "Geschwindigkeitssynchronisation" ... 83
6.4.13 Betriebsart "Elektronische Kurvenscheibe" ... 83
6.4.14 Betriebsart "Elektronisches Bewegungsprofil" ... 84
6.4.15 Betriebsart "Elektronisches FlexProfile" ... 85
6.4.16 Betriebsart "Robot Control" ... 86
6.4.17 Betriebsart "Antriebsgeführte Unterbrechung" ... 87
6.4.18 Betriebsart "Externer Funktionsbaustein" ... 88
6.5 Globale Variablen ... 88
6.6 Erste Schritte Achs-Interface ... 89
6.6.1 Überblick ... 89
6.6.2 Programm zum Achs-Interface ... 89
6.6.3 Achs-Interface Visualisierungen ... 90
Überblick ... 90
Systemübersicht-Visualisierung ... 91
Achsübersicht-Visualisierung ... 93
Positionierbetrieb-Visualisierung ... 95
Geschwindigkeitsbetrieb-Visualisierung ... 96
Synchronisationsbetrieb-Visualisierung ... 97
Visualisierung – Synchronisationsbetrieb mit Profilen ... 99
Imc-Interface-Steuerung und Status-Visualisierung ... 101
SPS-Auslastungs-Visualisierung ... 103
6.7 Achs-Interface Anwender-Erweiterung ... 103
6.7.1 Überblick ... 103
6.7.2 Erweitern der arAxisCtrl_gb[ ]-Struktur ... 105
6.7.3 Erweitern der arAxisStatus_gb[ ] Struktur ... 106
6.7.4 Erweitern des Funktionsbausteines ... 107
6.8 Achs-Interface-Strukturen ... 109
6.8.1 Überblick ... 109
6.8.2 MB_AXIS_STATUS_TYPE01 Struktur ... 110
MB_AXIS_STATUS_TYPE01 ... 110
MB_AXIS_STATUS_TYPE01 Elemente ... 111
6.8.3 MB_AXIS_CONTROL_TYPE01 Struktur ... 117
MB_AXIS_CONTROL_TYPE01 ... 117
MB_AXIS_CONTROL_TYPE01 Elemente ... 118
6.8.4 MB_AXIS_STATUS_TYPE02 Struktur ... 132
MB_AXIS_STATUS_TYPE02 ... 132
6.8.5 MB_AXIS_CONTROL_TYPE02 Struktur ... 133
MB_AXIS_CONTROL_TYPE02 ... 133
MB_AXIS_CONTROL_TYPE02 Elemente ... 134
6.9 IMC-Interface Strukturen ... 134
6.9.1 Überblick ... 134

Inhaltsverzeichnis

Seite

6.9.2 MB_IMC_Status_Type01 ... 135
6.9.3 MB_IMC_Control_Type01 ... 135
6.10 HowTo: Typische Anwenderaktivitäten ... 136
6.10.1 Zugriff auf Achsdaten ... 136
6.10.2 Achse hinzufügen ... 136
6.10.3 Erhöhung der maximalen Achsanzahl für Achsnummern größer 16 ... 136
6.10.4 Achse entfernen/umbenennen ... 137
6.10.5 Task anlegen ... 137
6.10.6 Achs-Interface Erweiterungen ... 137
6.10.7 Robot Control ... 138

7 Kinematik-Interface ... 141
7.1 Einführung ... 141
7.2 ML_KinTech.library ... 144
7.2.1 Einführung und Übersicht ... 144
7.2.2 Kinematik-Interface-Strukturen ... 146
Überblick ... 146
MB_KINEMATICS_CONTROL_TYPE01 Struktur ... 146
MB_KINEMATICS_STATUS_TYPE01 Struktur ... 164
7.2.3 Kinematik-Interface Funktionsbausteine ... 171
ML_KinematicsInitType01 ... 171
ML_KinematicsInterfaceType01 ... 174
ML_KinJogV1 ... 176
ML_KinJog ... 178
7.2.4 Betriebsarten ... 182
Überblick ... 182
Kinematik Bereit, Antriebe Bereit ... 182
Kinematik Bereit, Antriebe Halt ... 183
Kinematik Referenzieren ... 183
Absolutes PTP Positionieren ... 184
Relatives PTP Positionieren ... 185
Absolutes Lineares Positionieren ... 186
Relatives Lineares Positionieren ... 187
Absolutes Zirkulares Positionieren ... 188
Relatives Zirkulares Positionieren ... 189
Kinematik Stopp ... 190
Betriebsart "Interrupted" ... 191
Betriebsart "Continue" ... 191
Betriebsart "Externer Funktionsbaustein" ... 192
Betriebsart "RCL" ... 192
7.2.5 Globale Variablen ... 193
7.2.6 Erste Schritte Kinematik-Interface ... 194
Überblick ... 194
Programm zum Kinematik-Interface ... 194
Kinematik-Interface Visualisierungen ... 194
7.2.7 Kinematik-Interface Anwender-Erweiterung ... 205

Inhaltsverzeichnis

Seite

Überblick ... 205
Erweitern der Kinematik-Interface-Struktur ... 206
Erweitern des Funktionsbausteins TE_KinematicsInterface ... 206
7.3 Anwendungsbeispiele ... 207
7.3.1 Sequenzielle Programmierung mit dem Kinematik-Interface ... 207
7.3.2 Anmeldung einer benutzerdefinierten Transformation ... 209

8 GAT^compact ... 211
8.1 Überblick GAT^compact ... 211
8.2 Erste Schritte – GAT^compact ... 211
8.2.1 Einfache Motion-Applikation erstellen ... 211
Vorbereitung ... 211
Aufgabenstellung ... 211
Arbeitsablauf ... 212
8.3 Initialisierung ... 218
8.4 Betriebsarten ... 221
8.5 Zyklische Bearbeitung ... 223
8.6 Taskhandling ... 223
8.6.1 Taskkonfiguration GAT^compact ... 223
8.6.2 Hinzufügen einer MotionTask ... 224
8.6.3 Hinzufügen einer BackgroundTask ... 226
8.7 Fehlerhandling ... 227
8.7.1 Fehlererkennung und Diagnose ... 227
8.7.2 Fehlerreaktion ... 229
8.7.3 Fehlerlöschen ... 232
8.8 Visualisierung ... 232
8.8.1 Interne Visualisierung zur Inbetriebnahmeunterstützung ... 232
Allgemeines ... 232
System_Overview ... 232
8.8.2 Externe Visualisierung mit separaten HMI (WinStudio-Visualisierung) ... 234
Allgemeines ... 234
Dialog-Struktur ... 237
Erweitern der Visualisierung ... 238
Kommunikationsanbindung ... 240
Mehrsprachigkeit ... 242
8.9 Schnittstellen zum GAT^compact ... 243
8.9.1 Allgemeines ... 243
8.9.2 Daten zur Zustandsmaschine ... 243
8.9.3 Daten zum Fehlerhandling ... 243
8.9.4 Daten der Achsen ... 244
8.9.5 Daten der Steuerung ... 244
8.10 HowTo: Typische Anwenderaktivitäten ... 244
8.10.1 Zugriff auf Achsdaten ... 244
8.10.2 Achse hinzufügen ... 245
8.10.3 Erhöhung der maximalen Achsanzahl für Achsnummern größer 16 ... 246

Inhaltsverzeichnis

Seite

8.10.4 Achse entfernen/umbenennen ... 246
8.10.5 Betriebsart hinzufügen ... 247
8.10.6 Abarbeitung in MotionTask ... 248
8.10.7 Achs-Interface Erweiterungen ... 248
8.10.8 Robot Control ... 249
8.11 GAT^{compact} Kurzübersicht ... 252
8.11.1 Strukturübersicht ... 252
8.11.2 Übersicht der wichtigsten globalen Variablen der Maschine ... 252
8.11.3 Übersicht der wichtigsten globalen Variablen der Achsen ... 253
8.11.4 Übersicht der Methoden und Aktionen der Maschine ... 253
8.11.5 Zugriff auf Achsen innerhalb der Maschine ... 254

9 GAT^{compact} für IndraLogic XLC ohne sercos Interface ... 255
9.1 Überblick GAT^{compact} ... 255
9.2 Erste Schritte GAT^{compact} ... 255
9.3 Initialisierung ... 256
9.4 Betriebsarten ... 257
9.5 Zyklische Bearbeitung ... 258
9.6 Taskhandling ... 258
9.6.1 Taskkonfiguration GAT^{compact} ... 258
9.6.2 Hinzufügen einer BackgroundTask ... 259
9.7 Fehlerhandling ... 260
9.7.1 Fehlererkennung und Diagnose ... 260
9.7.2 Fehlerreaktion ... 261
9.7.3 Fehlerlöschen ... 262
9.8 Visualisierung ... 262
9.8.1 Interne Visualisierung zur Inbetriebnahmeunterstützung ... 262
Allgemeines ... 262
System_Overview ... 262
9.8.2 Externe Visualisierung mit separaten HMI (WinStudio-Visualisierung) ... 263
Allgemeines ... 263
Dialog-Struktur ... 265
Erweitern der Visualisierung ... 266
Kommunikationsanbindung ... 267
Mehrsprachigkeit ... 269
9.9 Schnittstellen zum GAT^{compact} ... 269
9.9.1 Daten der Machine ... 269
9.9.2 Daten zum Fehlerhandling ... 270
9.9.3 Daten der Steuerung ... 271
9.10 HowTo: Typische Anwenderaktivitäten ... 271
9.10.1 Betriebsart hinzufügen ... 271
9.11 GAT^{compact} Kurzübersicht ... 273
9.11.1 Strukturübersicht ... 273
9.11.2 Übersicht der wichtigsten globalen Variablen der Maschine ... 273
9.11.3 Übersicht der Methoden und Aktionen der Maschine ... 274

Inhaltsverzeichnis

Seite

10 GAT mit zentraler Zustandsmaschine ... 275
10.1 Allgemeines ... 275
10.2 Überblick ... 275
10.2.1 Grundidee ... 275
10.2.2 Voraussetzungen zur Anwendung ... 277
10.3 Erste Schritte – GAT mit zentraler Zustandsmaschine ... 277
10.3.1 Einfache Motion-Applikation erstellen ... 277
Allgemeines ... 277
Aufgabenstellung ... 277
Arbeitsablauf ... 278
10.4 Funktionsprinzip ... 288
10.5 Initialisierung ... 289
10.6 Betriebsarten ... 290
10.6.1 Allgemeines ... 290
10.6.2 Zustände einer Betriebsart ... 290
10.6.3 Einstellen der Zielbetriebsart nach Initialisierung ... 291
10.6.4 Wechsel der Betriebsart ... 292
10.6.5 Anwenderaktivitäten bei Verlassen einer Betriebsart ... 292
10.6.6 Anwenderaktivitäten bei Start einer Betriebsart ... 293
10.6.7 Überbrücken der zentralen Betriebsart ... 293
10.7 Module ... 294
10.7.1 Allgemeines ... 294
10.7.2 Regeln zur Erstellung ... 295
10.7.3 Modulschnittstelle ... 295
Allgemeines ... 295
Control- und Status-Struktur ... 295
Modulachsen ... 295
Modulspezifische I/O-Daten ... 298
Modulinstanzen ... 299
10.7.4 Zustände im Modul ... 299
10.7.5 Modulkoordination ... 300
10.8 Taskverwaltung ... 300
10.8.1 Taskkonfiguration GAT ... 300
10.8.2 GATMotionSyncTask ... 301
10.8.3 GATBackgroundProgTask ... 301
10.9 Fehlerbehandlung ... 301
10.9.1 Fehlererkennung und Diagnose ... 301
10.9.2 Fehlerreaktion ... 303
10.9.3 Fehler löschen ... 306
10.9.4 Fehler protokollieren ... 307
10.10 Visualisierung ... 307
10.10.1 Interne Visualisierung zur Inbetriebnahmeunterstützung ... 307
Allgemeines ... 307
AXIS_GAT ... 307
10.10.2 Externe Visualisierung mit separaten HMI (WinStudio-Visualisierung) ... 308
Allgemeines ... 308

Inhaltsverzeichnis

Seite

Dialog-Struktur ... 311
Erweitern der Visualisierung ... 312
Kommunikationsanbindung ... 314
Mehrsprachigkeit ... 316
10.11 Schnittstellen zum GAT ... 317
10.11.1 Allgemeines ... 317
10.11.2 Daten zur Maschine ... 317
10.11.3 Daten zum Fehlerhandling ... 318
10.11.4 Daten zu den Modulen ... 318
10.11.5 Daten der Achsen ... 320
10.11.6 Daten der Steuerung ... 320
10.12 HowTo: Typische Anwenderaktivitäten ... 321
10.12.1 Zugriff auf Achsdaten ... 321
10.12.2 Achse hinzufügen ... 322
10.12.3 Achse entfernen/umbenennen ... 322
10.12.4 Betriebsart hinzufügen ... 324
10.12.5 Anwahl der gewünschten Betriebsart nach Fehlerlöschen ... 324
10.12.6 Übergeordnete Logik zur Betriebsartenvorgabe ... 325
10.12.7 Zustände und Transitionen hinzufügen ... 325
10.12.8 Neues Modul hinzufügen ... 326
10.12.9 Weitere Instanz eines Moduls hinzufügen ... 327
10.12.10 Modul deaktivieren ... 327
10.12.11 Wiederverwendung von Modulen ... 328
10.12.12 Verwendung von globalen und Adressvariablen in Modulen ... 330
10.12.13 Abarbeitung in MotionTask ... 333
10.12.14 Achs-Interface Erweiterungen ... 334
10.12.15 Robot Control ... 335
10.12.16 Generierung eigener Programmrahmen auf Basis des GAT ... 337
10.12.17 PackML V3.0 Programmrahmen basierend auf GAT ... 338
10.12.18 Zeitkritische applikationsspezifische Fehlerreaktion ... 340
10.12.19 GAT Fehlerreaktion für F0 Fehler abschalten ... 340
10.12.20 Update von Version 10/11 auf 12/13 ... 340
10.13 GAT Kurzübersicht ... 341
10.13.1 Strukturübersicht ... 341
10.13.2 Übersicht der wichtigsten globalen Variablen der Maschine ... 342
10.13.3 Übersicht der wichtigsten globalen Variablen der Module ... 342
10.13.4 Übersicht der wichtigsten globalen Variablen der Achsen ... 342
10.13.5 Übersicht der Methoden und Aktionen der Maschine ... 343
10.13.6 Übersicht der Methoden und Aktionen der Module ... 344
10.13.7 Zugriff auf Achsen innerhalb des Moduls ... 345
10.13.8 Zugriff auf I/O-Daten innerhalb des Moduls ... 345

11 GAT mit dezentraler Zustandsmaschine ... 347
11.1 Allgemeines ... 347
11.2 Überblick ... 347
11.2.1 Grundidee ... 347

Inhaltsverzeichnis

Seite

11.2.2 Voraussetzungen zur Anwendung ... 349
11.3 Erste Schritte – GAT mit dezentraler Zustandsmaschine ... 349
11.3.1 Einfache Motion-Applikation erstellen ... 349
Vorbereitung ... 349
Aufgabenstellung ... 349
Arbeitsablauf ... 350
11.4 Funktionsprinzip ... 360
11.5 Initialisierung ... 360
11.6 Betriebsarten ... 361
11.7 Units ... 363
11.7.1 Allgemeines ... 363
11.7.2 Regeln zur Erstellung ... 364
11.7.3 Unit-Schnittstelle ... 364
Allgemeines ... 364
Control- und Status-Struktur ... 364
Unitachsen ... 364
Unitspezifische I/O-Daten ... 367
Unitinstanzen ... 367
11.7.4 Betriebsarten in der Unit ... 368
11.7.5 Sequentielle Programmierung ... 368
11.7.6 Unitkoordination ... 369
11.8 Taskverwaltung ... 370
11.8.1 Taskkonfiguration GAT ... 370
11.8.2 Hinzufügen einer BackgroundTask ... 370
11.9 Fehlerbehandlung ... 370
11.9.1 Fehlererkennung und Diagnose ... 370
11.9.2 Fehlerreaktion ... 373
11.9.3 Fehlerlöschen ... 375
11.9.4 Fehler protokollieren ... 376
11.10 Produkt- und Maschinendaten-Verwaltung ... 376
11.10.1 Einleitung ... 376
11.10.2 Maschinendaten-Verwaltung ... 376
11.10.3 Produktdaten-Verwaltung ... 376
11.11 Visualisierung ... 377
11.11.1 Umschalten der Visualisierung ... 377
11.11.2 Interne Visualisierung zur Inbetriebnahmeunterstützung ... 378
Allgemeines ... 378
System_Overview ... 378
11.11.3 Externe Visualisierung mit separaten HMI (WinStudio-Visualisierung) ... 380
Allgemeines ... 380
Dialog-Struktur ... 382
Erweitern der Visualisierung ... 383
Erweitern der maximalen Achsanzahl ... 385
Kommunikationsanbindung ... 387
Mehrsprachigkeit ... 389
11.12 Schnittstellen zum GAT mit dezentraler Zustandsmaschine ... 389

Inhaltsverzeichnis

Seite

11.12.1 Allgemeines ... 389
11.12.2 Daten des Maschinenverwalters (TE_MachineControl) ... 389
11.12.3 Daten der zentralen Fehlerverwaltung (TE_ErrorHandling) ... 390
11.12.4 Daten der Units ... 391
11.12.5 Daten der Achsen ... 393
11.12.6 Daten der Steuerung ... 393
11.13 HowTo: Typische Anwenderaktivitäten ... 394
11.13.1 Zugriff auf Achsdaten ... 394
11.13.2 Achse hinzufügen ... 394
11.13.3 Achse entfernen/umbenennen ... 395
11.13.4 Hauptbetriebsart hinzufügen ... 395
11.13.5 Neue Unit hinzufügen ... 396
11.13.6 Weitere Instanz einer Unit hinzufügen ... 397
11.13.7 Units deaktivieren ... 397
11.13.8 Wiederverwendung von Units ... 398
11.13.9 Verwendung von globalen und Adressvariablen in Units ... 399
11.13.10 Achs-Interface Erweiterungen ... 400
11.13.11 Robot Control ... 401
11.13.12 Generierung eines eigenen Programmrahmens auf Basis des GAT ... 403
11.14 GAT – dezentrale Zustandsmaschine Kurzübersicht ... 405
11.14.1 Strukturübersicht ... 405
11.14.2 Übersicht der wichtigsten globalen Variablen der Maschine und der Fehlerverwaltung ... 405
11.14.3 Übersicht der wichtigsten globalen Variablen der Units ... 406
11.14.4 Übersicht der wichtigsten globalen Variablen der Achsen ... 406
11.14.5 Übersicht der Methoden und Aktionen des TE_MachineControl ... 407
11.14.6 Übersicht der Methoden und Aktionen des TE_ErrorHandling ... 407
11.14.7 Übersicht der Methoden und Aktionen der Units ... 408
11.14.8 Zugriff auf Achsen innerhalb der Unit ... 408
11.14.9 Zugriff auf I/O-Daten innerhalb der Unit ... 408

12 GAT Plug-ins ... 409
12.1 Einleitung ... 409
12.2 Plug-in hinzufügen ... 409
12.3 Plug-in entfernen ... 409
12.4 Plug-in installieren ... 409
12.5 Plug-in selbst erstellen ... 410

13 Service und Support ... 415

Index ... 417

1 Zu dieser Dokumentation

1.1 Gültigkeit der Dokumentation

Übersicht über Zielgruppen und Produktphasen

Diese Dokumentation richtet sich an Anwender, die einfach und schnell übersichtliche Logic-Applikationen erstellen wollen.

Zur Verfügung stehen die folgenden Templates und das Achsinterface:

- GAT compact
- GAT (zentrale Zustandsmaschine)
- GAT (dezentrale Zustandsmaschine)
- Benutzerdefiniertes GAT

In der folgenden Grafik beziehen sich die umrandeten Aktivitäten, Produktphasen und Zielgruppen auf die vorliegendende Dokumentation.

Beispiel: In der Produktphase "Engineering" kann die Zielgruppe "Programmierer" mit Hilfe dieser Dokumentation die Aktivität "programmieren" ausführen.

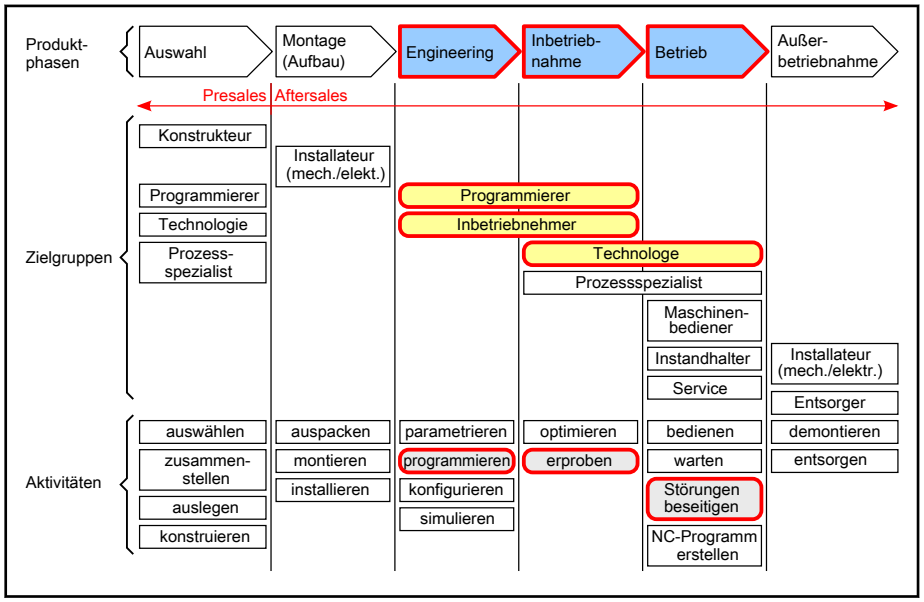


Abb.1-1: Zuordnung der vorliegenden Dokumentation zu den Zielgruppen, Produktphasen und den Aktivitäten der Zielgruppe

1.2 Gliederung der Dokumentation

Der erste Teil des Dokuments enthält wichtige Gebrauchs- und Sicherheitshinweise (Kap. 2 "Wichtige Gebrauchshinweise" auf Seite 21 und Kap. 3 "Sicherheitshinweise für elektrische Antriebe und Steuerungen" auf Seite 25).

Kap. 4 "Einführung" auf Seite 37 gibt eine Übersicht über die für die Systeme IndraMotion MLC/IndraLogic XLC verfügbaren GAT-Varianten.

Kap. 5 "Automatisierung von Applikationen" auf Seite 39 beschreibt die Aufgabenstellungen bei der Automatisierung von Applikationen.

Kap. 6 "ML_TechInterface.library" auf Seite 55 enthält eine Beschreibung der Funktionen und Funktionsbausteine dieser Bibliothek welche vom GAT benutzt werden.

Zu dieser Dokumentation

Kap. 8 "GAT^{compact}" auf Seite 211 beschreibt die Anwendung des Programmierrahmens "GAT^{compact}".

Kap. 9 "GAT^{compact} für IndraLogic XLC ohne sercos Interface" auf Seite 255 enthält eine Beschreibung des Programmierrahmens "GAT^{compact}" für das System IndraLogic XLC ohne sercos Interface.

Kap. 10 "GAT mit zentraler Zustandsmaschine" auf Seite 275 beschreibt die Anwendung des Programmierrahmens "GAT mit zentraler Zustandsmaschine" für das System IndraMotion MLC.

Kap. 11 "GAT mit dezentraler Zustandsmaschine" auf Seite 347 beschreibt die Anwendung des Programmierrahmens "GAT mit dezentraler Zustandsmaschine" für das System IndraMotion MLC.

Kap. 12 "GAT Plug-ins" auf Seite 409 beschreibt die Verwendung von Plug-ins.

Kap. 13 "Service und Support" auf Seite 415 enthält Informationen zu unserem Kundendienst-Helpdesk.

1.3 Erforderliche und ergänzende Dokumentationen

Dokumentationstitel mit Typenschlüssel und Materialnummer

IndraWorks		MLC	XLC
/36/	**Rexroth IndraWorks 13VRS Software-Installation** DOK-IWORKS-SOFTINS*V13-CORS-DE-P, R911336879 Die Dokumentation beschreibt den Installationsprozess von IndraWorks.	X	X
/5/	**Rexroth IndraWorks 13VRS Engineering** DOK-IWORKS-ENGINEE*V13-APRS-DE-P, R911336869 Diese Dokumentation beschreibt die Verwendung von IndraWorks, in das sich die Rexroth Engineering-Tools integrieren. Beschrieben werden das Arbeiten mit IndraWorks und die Bedienung der Oszilloskop-Funktion.	X	X
/20/	**Rexroth IndraMotion MLC 13VRS Funktionsbeschreibung** DOK-MLC***-FUNC****V13-APRS-DE-P, R911336294 Die Dokumentation dient der Beschreibung der Wizards, Kontextmenüs, Dialoge, Steuerungsinbetriebnahme, Gerätekonfiguration, Funktionalitäten der IndraMotion MLC.	X	
/20/	**Rexroth IndraLogic XLC 13VRS Funktionsbeschreibung** DOK-XLC***-FUNC****V13-APRS-DE-P, R911336351 Die Dokumentation dient der Beschreibung der Wizards, Kontextmenüs, Dialoge, Steuerungsinbetriebnahme, Gerätekonfiguration, Funktionalitäten der IndraLogic XLC.		X
/7/	**Rexroth IndraWorks 13VRS CamBuilder** DOK-IWORKS-CAMBUIL*V13-APRS-DE-P, R911336290 Diese Dokumentation dient zur Beschreibung der Grundlagen und der Bedienung des CamBuilder, des Werkzeugs zur Kurvenscheibenbearbeitung.	X	X
/37/	**Rexroth IndraLogic XLC IndraMotion MLC 13VRS Automation Interface** DOK-XLCMLC-AUT*INT*V13-APRS-DE-P, R911336355 Diese Dokumentation beschreibt den skriptbasierten Zugriff auf IndraWorks-Projektdaten über die Schnittstelle des Automation Interfaces.	X	X

Zu dieser Dokumentation

IndraWorks		MLC	XLC
/38/	**Rexroth IndraWorks 12VRS FDT-Container** DOK-IWORKS-FDT*CON*V12-APRS-DE-P, R911334397 Diese Dokumentation beschreibt die FDT-Container-Funktionalität von IndraWorks. Beschrieben wird das Aktivieren der Funktionalität im Projekt und das Arbeiten mit DTMs.	X	X
/29/	**Rexroth IndraLogic XLC IndraMotion MLC 13VRS Projektkonvertierung** DOK-XLCMLC-PROCONV*V13-APRS-DE-P, R911336365 Diese Dokumentation beschreibt die Projektkonvertierung von IndraLogic 04VRS und IndraMotion MLC04VRS auf IndraWorks Version 12 mit IndraLogic 2G. Es wird insbesondere auf die Änderungen im Bereich Motion und SPS eingegangen.	X	X
/28/	**Rexroth IndraMotion MLC 13VRS Inbetriebnahme** DOK-MLC***-STARTUP*V13-CORS-DE-P, R911336307 Das vorliegende Handbuch beschreibt die Inbetriebnahme- und Servicearbeitsschritte für das System IndraMotion MLC. Es beinhaltet Checklisten für häufig durchzuführende Arbeiten und eine ausführliche Beschreibung der Arbeitsschritte.	X	

Abb.1-2: XLC/MLC-Dokumentationsübersicht

Motion		MLC	XLC
/23/	**Rexroth IndraLogic XLC IndraMotion MLC 13VRS PLCopen-Bibliotheken** DOK-XLCMLC-FUNLIB**V13-LIRS-DE-P, R911336359 Diese Dokumentation beschreibt die Funktionsbausteine, Funktionen und Datentypen der Bibliotheken RIL_CommonTypes, ML_Base und ML_PLCopen für die IndraLogic XLC/IndraMotion MLC. Die Dokumentation enthält ferner die Fehlerreaktion der Bausteine.	X	X
/27/	**Rexroth IndraLogic XLC IndraMotion MLC 13VRS Generic Application Template** DOK-XLCMLC-TF*GAT**V13-APRS-DE-P, R911336369 Dieses Handbuch stellt dem IndraLogic SPS-Programmierer ein strukturiertes Template, welches zum Hinzufügen und Bearbeiten des SPS-Programmiercodes verwendet werden kann, zur Verfügung. Beschrieben werden das Template selbst, der Template Wizard und Beispielapplikationen.	X	X
/31/	**Rexroth IndraMotion MLC 13VRS RCL-Programmieranleitung** DOK-MLC***-RCL*PRO*V13-APRS-DE-P, R911336296 Das vorliegende Handbuch informiert über die Robot Control. Die Programmiersprache RCL (Robot Control Language) steht dabei im Vordergrund. Beschrieben werden die Programmstruktur, Variablen, Funktionen, Bewegungsanweisungen sowie die notwendigen Parameter des Systems.	X	
/21/	**Rexroth IndraLogic XLC IndraMotion MLC 13VRS Parameter** DOK-XLCMLC-PARAM***V13-RERS-DE-P, R911336363 Diese Dokumentation beschreibt die Parameter der Systeme XLC/MLC, sowie das Wechselspiel zwischen Parametrieren und Programmieren. Beschrieben werden die Achs-Parameter, die Steuerungs-Parameter, die Kinematik-Parameter, die Messtaster- und die Nockenschaltwerks-Parameter.	X	X
/10/	**Rexroth IndraDrive Firmware für Antriebsregelgeräte MPH-, MPB-, MPD-, MPC-07** DOK-INDRV*-MP*-07VRS**-FKRS-DE-P, R911328669 Diese Dokumentation beschreibt alle funktionellen Eigenschaften der IndraDrive-Firmware in den Varianten MPH-07, MPB-07, MPD-07 und MPC-07.		
/11/	**Rexroth IndraDrive MPx-17 Funktionen** DOK-INDRV*-MP*-17VRS**-APRS-DE-P, R911331235 Diese Dokumentation beschreibt alle funktionellen Eigenschaften der IndraDrive-Firmware in den Varianten MPB-17, MPM-17, MPC-17 und MPE-17.		

Abb.1-3: XLC/MLC-Dokumentationsübersicht

Zu dieser Dokumentation

Feldbusse		MLC	XLC
/4/	**Rexroth IndraWorks 13VRS Feldbusse** DOK-IWORKS-FB******V13-APRS-DE-P, R911336871 Diese Dokumentation dient der Beschreibung der unterstützten Feldbusse und ihrer Diagnose-Funktionsbausteine.	X	X
/4a/	**Rexroth IndraWorks 13VRS Feldbus-Bibliotheken** DOK-IWORKS-FB*LIB**V13-LIRS-DE-P, R911337856 Das vorliegende Handbuch beschreibt die Bibliotheken der Feldbusse: RIL_ProfibusDP_02, RIL_ProfibusDPSlave, RIL_ProfinetIO, RIL_ProfinetIODevice, RIL_EtherNetIPAdapter, RIL_MappingList, RIL_SERCOSIII, RIL_Inline, einschließlich deren Diagnosen, sowie die Fehlerreaktion der Bausteine.	X	X
/63/	**sercos Systemhandbuch für I/O-Geräte** DOK-CONTRL-ILS3*******-APRS-DE-P, R911333511 Diese Dokumentation beschreibt die Projektierung, Parametrierung, Inbetriebnahme und Diagnose von I/O-Geräten mit sercos Schnittstelle.	X	X

Abb.1-4: XLC/MLC-Dokumentationsübersicht

HMI		MLC	XLC
/8/	**Rexroth IndraWorks 13VRS HMI** DOK-IWORKS-HMI*****V13-APRS-DE-P, R911336873 Diese Dokumentation beschreibt die Funktionen, die Konfiguration und die Bedienung der Bedienoberflächen IndraWorks HMI Engineering und IndraWorks HMI Operation.	X	X
/6/	**Rexroth IndraWorks 13VRS WinStudio** DOK-IWORKS-WINSTUD*V13-APRS-DE-P, R911336881 Diese Dokumentation beschreibt die Installation der Software, das Arbeiten mit WinStudio und das Erstellen und den Betrieb von Applikationen.	X	X
/50/	**Rexroth IndraLogic XLC IndraMotion MLC 13VRS HMI-Anbindung** DOK-XLCMLC-HMI*****V13-APRS-DE-P, R911336361 Dieses Dokument beschreibt die von der IndraLogic XLC und IndraMotion MLC unterstützten Visualisierungssysteme sowie deren Anbindung.	X	X

Abb.1-5: XLC/MLC-Dokumentationsübersicht

SPS		MLC	XLC
/3/	**Rexroth IndraWorks 13VRS IndraLogic 2G Programmieranleitung** DOK-IWORKS-IL2GPRO*V13-APRS-DE-P, R911336875 Diese Dokumentation beschreibt das SPS-Programmierwerkzeug IndraLogic 2G und seine Verwendung. Beschrieben werden die Basisverwendung, Erste Schritte, die Visualisierung, Menüpunkte und die Editoren.	X	X
/33/	**Rexroth IndraWorks 13VRS Basis-Bibliotheken IndraLogic 2G** DOK-IL*2G*-BASLIB**V13-LIRS-DE-P, R911336284 Das vorliegende Handbuch beschreibt die systemübergreifenden SPS-Bibliotheken.	X	X

Abb.1-6: XLC/MLC-Dokumentationsübersicht

Technologie		MLC	XLC
/30/	**Rexroth IndraMotion MLC 13VRS Technologie-Bibliotheken** DOK-MLC***-TF*LIB**V13-LIRS-DE-P, R911336323 Diese Dokumentation beschreibt die Funktionsbausteine, Funktionen und Datentypen der Bibliotheken "ML_TechInterface.library", "ML_TechMotion.library", "RMB_TechCam.library" und "ML_TechBase.library" sowie Bibliotheken zur Wicklerfunktionalität, Registerreglerfunktionalität und Querschneiderfunktionalität.	X	
/60/	**Rexroth IndraMotion MLC 13VRS RegisterControl (Bibliothek)** DOK-MLC***-REGI*CO*V13-LIRS-DE-P, R911336305 Diese Dokumentation beschreibt die Ein- und Ausgänge der einzelnen Funktionsbausteine und gibt Hinweise für deren Verwendung.	X	
/62/	**Rexroth IndraMotion MLC 13VRS RegisterControl (Anwendungsbeschreibung)** DOK-MLC***-REGI*CO*V13-APRS-DE-P, R911336303 Dieses Dokument beschreibt die Anwendung der integrierten Registerregelung für eine Tiefdruckmaschine. Beschrieben werden die Komponenten des Markenstromsensors, die HMI-Anwendung und die Fehlerbehebung. Diese Anleitung dient dazu, die Registerregelung zu bedienen und auf Fehler zu reagieren sowie Diagnosen abzufragen. Die Zielgruppe für diese Dokumentation sind Maschineneinrichter und Maschinenbediener.	X	
/49/	**Rexroth IndraMotion MLC 13VRS Anwendung Wicklerfunktionen** DOK-MLC***-TF*WIND*V13-APRS-DE-P, R911336325 Diese anwendungsbezogene Systemdokumentation beschreibt die Anwendung der Wickler-Technologiefunktionen.	X	

Abb.1-7: XLC/MLC-Dokumentationsübersicht

Hardware		MLC	XLC
/1/	**Rexroth IndraControl L45/L65/L85 Steuerung** DOK-CONTRL-ICL45L65L85-PRRS-DE-P, R911332115 Diese Dokumentation dient der Beschreibung der Steuerungen IndraControl L45/L65/L85.	X	X
/2/	**Rexroth IndraControl L25** DOK-CONTRL-IC*L25*****-PRRS-DE-P, R911328473 Diese Dokumentation dient der Beschreibung der Steuerungen IndraControl L25.	X	X
/24/	**Rexroth IndraControl Lxx 13VRS Funktionsmodule** DOK-CONTRL-FM*LXX**V13-APRS-DE-P, R911336282 Diese Dokumentation dient der Beschreibung aller Funktionsmodule der Lxx Steuerungen inkl. Engineering und Diagnose.	X	X
/12/	**Rexroth IndraDrive Antriebsregelgeräte MPx-02 bis MPx-07** DOK-INDRV*-GEN-**VRS**-PARS-DE-P, R911297316 Diese Dokumentation enthält die Beschreibung aller, in der Firmware für Antriebsregelgeräte der IndraDrive-Familie implementierten Parameter. Sie unterstützt die Parametrierung der Antriebsregelgeräte.		

Zu dieser Dokumentation

Hardware		MLC	XLC
/13/	**Rexroth IndraDrive MPx-02 bis MPx-07 und HMV** DOK-INDRV*-GEN-**VRS**-WARS-DE-P, R911297318 Diese Dokumentation enthält die Beschreibung aller in den folgenden Firmwaren implementierten Diagnosen: • Antriebsregelgerät-Firmwaren MPx-02 bis MPx-08 und • Firmwaren der Versorgungsgeräte des Typs "HMV". Sie unterstützt den Maschinenbediener und den Anlagenprogrammierer bei der Fehlerbehebung.		
/35/	**Rexroth IndraDrive Antriebsregelgeräte Steuerteile CSB01, CSH01, CDB01** DOK-INDRV*-CSH********-PRRS-DE-P, R911295011 Diese Dokumentation ist eine Projektierungsbeschreibung zur Auswahl und Verwendung der Steuerteile CSB01, CDB01 und CSH01 für Antriebsregelgeräte der Produktfamilien Rexroth IndraDrive M und Rexroth IndraDrive C.		

Abb.1-8: XLC/MLC-Dokumentationsübersicht

Diagnose und Service		MLC	XLC
/26/	**Rexroth IndraWorks 13VRS IndraMotion Service Tool** DOK-IWORKS-IMST****V13-RERS-DE-P, R911337706 Die Dokumentation beschreibt das IndraMotion Service Tool (IMST). Es ist ein webbasiertes Diagnosewerkzeug, das zum Zugriff auf die Steuerungssysteme IndraMotion MLC L25, L45 oder L65 sowie MLP über eine Ethernet-Hochgeschwindigkeitsverbindung dient. Das IMST ermöglicht OEMs, Endanwendern und Kundenbetreuern von überall auf ein System zuzugreifen und Ferndiagnosen zu stellen. Voraussetzung ist ein PC mit der Internet Explorer Version 6 oder 7.	X	X
/22/	**Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnosen** DOK-XLCMLC-DIAG****V13-RERS-DE-P, R911336357 Diese Dokumentation enthält alle in den Steuerungssystemen IndraLogic XLC und IndraMotion MLC implementierten Steuerungsparameter.	X	X

Abb.1-9: XLC/MLC-Dokumentationsübersicht

Systemübersicht		MLC	XLC
/48/	**Rexroth IndraMotion for Printing 13VRS Systemübersicht** DOK-IM*PR*-SYSTEM**V13-PRRS-DE-P, R911336288 Das vorliegende Handbuch beschreibt das Produkt IndraMotion for Printing. Es stellt die Steuerungs-, Antriebs- und IO-Systeme vor und gibt eine Einführung in die Inbetriebnahme und Programmierung.	X	
/48/	**Rexroth IndraMotion for Packaging 13VRS Systemübersicht** DOK-IM*PA*-SYSTEM**V13-PRRS-DE-P, R911336286 Das vorliegende Handbuch beschreibt das Produkt IndraMotion for Packaging. Es stellt die Steuerungs-, Antriebs- und IO-Systeme vor und gibt eine Einführung in die Inbetriebnahme und Programmierung.	X	

Systemübersicht		MLC	XLC
/9/	**Rexroth IndraMotion MLC 13VRS Systemübersicht** DOK-MLC***-SYSTEM**V13-PRRS-DE-P, R911336309 Diese Dokumentation stellt eine Übersicht der möglichen Hardware-/Softwarekomponenten des Automationssystems IndraMotion MLC in der genannten Version dar. Die Dokumentation dient als Auswahlhilfe zur Zusammenstellung eines Systems.	X	
/9/	**Rexroth IndraLogic XLC 13VRS Systemübersicht** DOK-XLC***-SYSTEM**V13-PRRS-DE-P, R911336353 Diese Dokumentation stellt eine Übersicht der möglichen Hardware-/Softwarekomponenten des Automationssystems IndraLogic XLC in der genannten Version dar. Die Dokumentation dient als Auswahlhilfe zur Zusammenstellung eines Systems.		X

Abb.1-10: XLC/MLC-Dokumentationsübersicht

Erste Schritte		MLC	XLC
/25/	**Rexroth IndraMotion MLC 13VRS Die ersten Schritte** DOK-MLC***-F*STEP**V13-CORS-DE-P, R911336292 Diese Dokumentation beschreibt die "ersten Schritte" mit der IndraMotion MLC und RobotControl. Beschrieben werden die Hard- und Softwarevoraussetzungen und das Erstellen eines Projekts.	X	
/25/	**Rexroth IndraLogic XLC 13VRS Die ersten Schritte** DOK-XLC***-F*STEP**V13-CORS-DE-P, R911336349 Diese Dokumentation beschreibt die "ersten Schritte" mit der IndraLogic XLC. Beschrieben werden die Hard- und Softwarevoraussetzungen und das Erstellen eines Projekts.		X

Abb.1-11: XLC/MLC-Dokumentationsübersicht

1.4 Darstellung von Informationen

1.4.1 Sicherheitshinweise

Die Sicherheitshinweise, soweit sie in der vorliegenden Anwendungsdokumentation vorhanden sind, beinhalten bestimmte Signalwörter ("Gefahr", "Warnung", "Vorsicht", "Hinweis") und ggf. eine Signalgrafik (nach ANSI Z535.6-2006).

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

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

Die Sicherheitshinweise in dieser Dokumentation werden wie folgt dargestellt:

⚠ GEFAHR

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

⚠ WARNUNG

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

Zu dieser Dokumentation

⚠ VORSICHT

Bei Nichtbeachtung dieses Sicherheitshinweises können mittelschwere oder leichte Körperverletzung eintreten.

HINWEIS

Bei Nichtbeachtung dieses Sicherheitshinweises können Sachschäden eintreten.

1.4.2 Verwendete Symbole

Hinweis Hinweise werden wie folgt dargestellt:

 Das ist ein Hinweis für den Anwender.

Tipp Tipps werden wie folgt darstellt:

 Das ist ein Tipp für den Anwender.

1.4.3 Bezeichnungen und Abkürzungen

Begriff	**Erklärung**
GAT	**G**eneric **A**pplication **T**emplate
Achs-Interface	Das **Achs-Interface** bündelt und erweitert PLCopen-Bewegungs-Funktionsbausteine und stellt ein einfach zu bedienendes Interface für die Antriebs-Funktionalität zur Verfügung
Imc-Interface	Das **Imc-Interface** stellt ein einfach zu bedienendes Interface für die Steuerungs-Funktionalität zur Verfügung. Es enthält Steuersignale und Parameter für die Diagnose und Administration der Steuerung
TechInterface	Das TechInterface besteht aus den Komponenten • Achs-Interface und • Imc-Interface
Funktionsbaustein	stellt applikationsspezifische Funktionalitäten zur Verfügung
IndraMotion MLC	kompaktes Motion-Logic-System mit Motion-, Robot- und Logic-Control-Funktionalität
IndraLogic XLC	kompaktes SPS-System mit integrierter Motion-Funktionalität
IndraWorks Engineering Framework	Projektierungs- und Inbetriebnahmetool von Bosch Rexroth
IndraDrive	Antriebs-Seuergerät
Wizard	Assistent, der den Anwender durch mehrere Dialoge für eine ergonomische Dateneingabe führt
IEC	International Electrotechnical Commission (Normungsgremium für Elektrotechnik)

sercos	Das sercos (serial realtime communication System) interface ist eine weltweit genormte digitale Schnittstelle zur Kommunikation zwischen Steuerungen und Antrieben
XML	Die Extensible Markup Language (engl. für "erweiterbare Auszeichnungssprache"), abgekürzt XML, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdaten
CSV	Diese Dateiformat beschreibt den Aufbau einer Textdatei zur Speicherung oder zum Austausch einfach strukturierter Daten. Die Dateiendung .csv ist eine Abkürzung für "Comma-Separated Values"
IMC	IndraMotion Control
MLD-S	Motion Logic Drive (single drive)
MLD-M	Motion Logic Drive (multi drive)
D.h.	Das heißt
z.B.	zum Beispiel
ms	Millisekunde
mm	Millimeter
%	Prozent
U/min	Drehzahl

Abb.1-12: Verwendete Bezeichnungen und Abkürzungen

2 Wichtige Gebrauchshinweise

2.1 Bestimmungsgemäßer Gebrauch

2.1.1 Einführung

Die Produkte von Bosch Rexroth werden nach dem jeweiligen Stand der Technik entwickelt und gefertigt. Vor ihrer Auslieferung werden sie auf ihren betriebssicheren Zustand hin überprüft.

⚠ WARNUNG **Personen- und Sachschäden durch falschen Gebrauch der Produkte!**

Die Produkte sind für den Einsatz im industriellen Umfeld konzipiert und dürfen nur bestimmungsgemäß eingesetzt werden. Wenn die Produkte nicht bestimmungsgemäß eingesetzt werden, dann können Situationen entstehen, die Sach- und Personenbeschädigung nach sich ziehen.

☞ Für Schäden bei nicht bestimmungsgemäßem Gebrauch der Produkte leistet Bosch Rexroth als Hersteller keinerlei Gewährleistung, Haftung oder Schadensersatz; die Risiken bei nicht bestimmungsgemäßem Gebrauch der Produkte liegen allein beim Anwender.

Bevor Sie Bosch Rexroth Produkte einsetzen, müssen die folgenden Voraussetzungen erfüllt sein, um einen bestimmungsgemäßen Gebrauch der Produkte zu gewährleisten:

- Jeder, der in irgendeiner Weise mit einem unserer Produkte umgeht, muss die entsprechenden Sicherheitsvorschriften und den bestimmungsgemäßen Gebrauch lesen und verstehen
- Hardwareprodukte müssen in ihrem Originalzustand belassen werden; d. h. es dürfen keine baulichen Veränderungen an ihnen vorgenommen werden
- Softwareprodukte dürfen nicht dekompiliert und ihre Quellcodes dürfen nicht verändert werden
- Beschädigte oder fehlerhafte Produkte dürfen nicht eingebaut oder betrieben werden
- Es muss gewährleistet sein, dass die Produkte entsprechend den in der Dokumentation genannten Vorschriften installiert sind

2.1.2 Einsatz- und Anwendungsbereiche

Die IndraMotion MLC/IndraLogic XLC von Bosch Rexroth ist für Motion-/Logic-Anwendungen vorgesehen.

☞ Die IndraMotion MLC/IndraLogic XLC darf nur mit den in dieser Dokumentation angegebenen Zubehör- und Anbauteilen verwendet werden. Nicht ausdrücklich genannte Komponenten dürfen weder angebaut noch angeschlossen werden. Gleiches gilt für Kabel und Leitungen.

Der Betrieb darf nur in den ausdrücklich angegebenen Konfigurationen und Kombinationen der Komponenten und mit der in der jeweiligen Funktionsbeschreibung angegebenen und spezifizierten Soft- und Firmware erfolgen.

Wichtige Gebrauchshinweise

Die IndraMotion MLC/IndraLogic XLC ist für Logic-Anwendungen sowie den Einsatz in ein- und mehrachsigen Antriebs- und Steuerungsaufgaben entwickelt worden.

Für den applikationsspezifischen Einsatz der IndraMotion MLC/IndraLogic XLC stehen Gerätetypen mit unterschiedlicher Leistung und unterschiedlichen Schnittstellen zur Verfügung.

Typische Anwendungsbereiche der IndraMotion MLC/IndraLogic XLC sind:

- Handhabungs- und Montagesysteme
- Nahrungsmittel- und Verpackungsmaschinen
- Druck- und Verarbeitungsmaschinen
- Werkzeugmaschinen
- Pressen
- Hütten- und Walztechnik
- Fördertechnik
- Gebäudetechnik
- Sondermaschinen
- Holzbearbeitungsmaschinen

Die IndraMotion MLC/IndraLogic XLC darf nur unter den in dieser Dokumentation angegebenen Montage- und Installationsbedingungen, in der angegebenen Gebrauchslage und unter den angegebenen Umweltbedingungen (Temperatur, Schutzart, Feuchte, EMV u. a.) betrieben werden.

2.2 Nicht bestimmungsgemäßer Gebrauch

Die Verwendung der IndraMotion MLC/IndraLogic XLC außerhalb der vorgenannten Anwendungsgebiete oder unter anderen als den in der Dokumentation beschriebenen Betriebsbedingungen und angegebenen technischen Daten gilt als "nicht bestimmungsgemäß".

Die IndraMotion MLC/IndraLogic XLC darf nicht eingesetzt werden wenn:

- die IndraMotion MLC/IndraLogic XLC Betriebsbedingungen ausgesetzt wird, die die vorgeschriebenen Umgebungsbedingungen nicht erfüllen. Untersagt sind z. B. der Betrieb unter Wasser, unter extremen Temperaturschwankungen, extremen Maximaltemperaturen oder in explosionsgefährdeter Umgebung
- die beabsichtigten Anwendungen von Bosch Rexroth nicht freigegeben sind. Beachten Sie hierzu bitte unbedingt die Aussagen in den allgemeinen Sicherheitshinweisen!
- die Maschine oder Anlage über eine direkte ungesicherte Verbindung zum Internet verfügt. Die Ethernet-Schnittstelle der IndraMotion MLC/IndraLogic XLC bietet keinen sicheren Schutz gegen unautorisierten Zugriff. Deswegen sollte die Steuerung nur in abgesicherten Netzwerken eingesetzt werden (Security). Ansonsten besteht die Gefahr von Störungen bzw. einem Steuerungsausfall aufgrund unautorisierter Zugriffe

Der Betrieb nach den folgenden Normen und Richtlinien ist nicht erlaubt:

- Personenaufzüge (95/16/EG)
- ATEX (94/9/EG)
- Brennersteuerung (EN 298)
- Druckkessel (EN 764-7:2002/AC)

Weiterhin ist der Einsatz als sicherheitsbezogenes Teil von Steuerungen im Sinne der DIN EN ISO 13849 nicht bestimmungsgemäß. Die funktionale

Sicherheit muss durch zusätzliche sicherheitstechnisch zertifizierte Komponenten realisiert werden.

3 Sicherheitshinweise für elektrische Antriebe und Steuerungen

3.1 Begriffsdefinitionen

Anlage Eine Anlage besteht aus mehreren zu einem bestimmten Zweck und an einem bestimmten Ort miteinander verbundenen Geräten oder Systemen, die jedoch nicht als eine einzige Funktionseinheit in Verkehr gebracht werden sollen.

Elektrisches Antriebssystem Ein elektrisches Antriebssystem umfasst alle Bestandteile von der Netzeinspeisung bis zur Motorwelle; dazu zählen z. B. Elektromotor(en), Motorgeber, Versorgungs- und Antriebsregelgeräte, sowie Hilfs- und Zusatzkomponenten, wie Netzfilter, Netzdrossel und dazugehörige Leitungen und Kabel.

Anwender Ein Anwender ist eine Person, die ein in Verkehr gebrachtes Produkt installiert, in Betrieb nimmt oder verwendet.

Anwendungsdokumentation Eine Anwendungsdokumentation umfasst die gesamte Dokumentation, die dazu dient, den Anwender des Produkts über den Gebrauch und sicherheitsrelevante Inhalte für Projektierung, Einbau, Installation, Montage, Inbetriebnahme, Betrieb, Wartung, Reparatur, Außerbetriebnahme des Produkts zu informieren. Folgende Begriffe sind dafür ebenfalls üblich: Bedienungsanleitung, Betriebsanleitung, Inbetriebnahmeanleitung, Gebrauchsanleitung, Projektierungsanleitung, Anwendungsbeschreibung usw.

Elektrisches Betriebsmittel Ein elektrisches Betriebsmittel ist ein Gegenstand, der zum Erzeugen, Umwandeln, Fortleiten, Verteilen oder Anwenden von elektrischer Energie benutzt wird, wie z. B. Elektromotoren, Transformatoren, Schaltgeräte, Kabel, Leitungen, Stromverbrauchsgeräte, bestückte Leiterplatten, Einschübe, Schaltschränke usw.

Gerät Ein Gerät ist ein Endprodukt mit einer ihm eigenen Funktion, das für Anwender bestimmt ist und als eine einzelne Handelsware in Verkehr gebracht wird.

Hersteller Unter Hersteller ist eine natürliche oder juristische Person zu verstehen, welche die Verantwortung für die Auslegung und die Herstellung eines Produktes trägt, das in seinem Namen in den Verkehr gebracht wird. Der Hersteller kann Fertigerzeugnisse, Fertigteile oder Fertigelemente verwenden oder Arbeiten an Subunternehmer vergeben. Er muss jedoch immer die Oberaufsicht behalten und die notwendigen Befugnisse besitzen, um die Verantwortung für das Produkt übernehmen zu können.

Komponente Eine Komponente ist eine Kombination von Bauelementen mit vorgegebener Funktion, die Teil eines Betriebsmittels, Gerätes oder Systems sind. Komponenten des elektrischen Antriebs- und Steuerungssystems sind z. B. Versorgungsgeräte, Antriebsregelgeräte, Netzdrossel, Netzfilter, Motoren, Kabel, usw.

Maschine Unter Maschine ist die Gesamtheit von miteinander verbundenen Teilen oder Baugruppen zu verstehen, von denen mindestens eine(s) beweglich ist. Eine Maschine besteht somit aus entsprechenden Maschinenantriebselementen sowie Steuer- und Energiekreisen, die für eine bestimmte Anwendung zusammengefügt sind. Eine Maschine ist z. B. für die Verarbeitung, Behandlung, Fortbewegung oder Verpackung eines Materials bestimmt. Der Ausdruck "Maschine" deckt auch eine Zusammenstellung von Maschinen ab, die so angeordnet und gesteuert werden, dass sie als einheitliches Ganzes funktionieren.

Produkt Beispiele für ein Produkt: Gerät, Komponente, Bauteil, System, Software, Firmware u. a.

Sicherheitshinweise für elektrische Antriebe und Steuerungen

Projektierungsanleitung Eine Projektierungsanleitung ist Teil der Anwendungsdokumentation zur Hilfestellung bei der Auslegung und Planung von Systemen, Maschinen oder Anlagen.

Qualifiziertes Personal Im Sinne dieser Anwendungsdokumentation umfasst das qualifizierte Personal diejenigen Personen, die mit der Installation, Montage, Inbetriebnahme und Betrieb der Komponenten des elektrischen Antriebs- und Steuerungssystems sowie den damit verbundenen Gefahren vertraut sind und über die ihre Tätigkeit entsprechende Qualifikationen verfügen. Zu derartigen Qualifikationen gehören u. a.:

1) Eine Ausbildung oder Unterweisung bzw. Berechtigung, um Stromkreise und Geräte sicher ein- und auszuschalten, zu erden und zu kennzeichnen.

2) Eine Ausbildung oder Unterweisung für die Pflege und den Gebrauch angemessener Sicherheitsausrüstung

3) Eine Schulung in Erster Hilfe

Steuerungssystem Ein Steuerungssystem umfasst mehrere miteinander verbundene Steuerungskomponenten, die als eine einzige Funktionseinheit in Verkehr gebracht werden.

3.2 Grundsätzliche Hinweise

3.2.1 Benutzung und Weitergabe der Sicherheitshinweise

Installieren und betreiben Sie keine Komponenten des elektrischen Antriebs- und Steuerungssystems, bevor Sie alle mitgelieferten Unterlagen sorgfältig durchgelesen haben. Diese Sicherheitshinweise und alle anderen Benutzerhinweise sind vor jeder Arbeit mit diesen Komponenten durchzulesen. Sollten Ihnen keine Benutzerhinweise für die Komponenten zur Verfügung stehen, wenden Sie sich an Ihren zuständigen Vertriebspartner von Bosch Rexroth. Verlangen Sie die unverzügliche Übersendung dieser Unterlagen an den oder die Verantwortlichen für den sicheren Betrieb der Komponenten.

Bei Verkauf, Verleih und/oder anderweitiger Weitergabe der Komponente sind diese Sicherheitshinweise ebenfalls in der Landessprache des Anwenders mitzugeben.

Unsachgemäßer Umgang mit diesen Komponenten und Nichtbeachten der hier angegebenen Sicherheitshinweise sowie unsachgemäße Eingriffe in die Sicherheitseinrichtung können zu Sachschäden, Körperverletzung, elektrischem Schlag oder im Extremfall zum Tod führen.

3.2.2 Voraussetzungen für den sicheren Gebrauch

Lesen Sie vor der ersten Inbetriebnahme der Komponenten des elektrischen Antriebs- und Steuerungssystems folgende Hinweise, damit Sie Körperverletzungen und/oder Sachschäden vermeiden können. Sie müssen diese Sicherheitshinweise einhalten.

- Bei Schäden infolge von Nichtbeachtung der Sicherheitshinweise übernimmt Bosch Rexroth keine Haftung.
- Vor der Inbetriebnahme sind die Betriebs-, Wartungs- und Sicherheitshinweise durchzulesen. Wenn die Anwendungsdokumentation in der vorliegenden Sprache nicht einwandfrei verstanden wird, bitte beim Lieferanten anfragen und diesen informieren.
- Der einwandfreie und sichere Betrieb der Komponente setzt sachgemäßen und fachgerechten Transport, Lagerung, Montage und Installation sowie sorgfältige Bedienung und Instandhaltung voraus.

Sicherheitshinweise für elektrische Antriebe und Steuerungen

- Nur qualifiziertes Personal darf an Komponenten des elektrischen Antriebs- und Steuerungssystems oder in dessen Nähe arbeiten.
- Nur von Bosch Rexroth zugelassene Zubehör- und Ersatzteile verwenden.
- Sicherheitsvorschriften und -bestimmungen des Landes beachten, in welchem die Komponenten des elektrischen Antriebs- und Steuerungssystems betrieben werden.
- Komponenten des elektrischen Antriebs- und Steuerungssystems nur bestimmungsgemäß verwenden. Siehe dazu Kapitel "Bestimmungsgemäßer Gebrauch".
- Die in der vorliegenden Anwendungsdokumentation angegebenen Umgebungs- und Einsatzbedingungen müssen eingehalten werden.
- Anwendungen zur funktionalen Sicherheit sind nur zugelassen, wenn sie ausdrücklich und eindeutig in der Anwendungsdokumentation "Integrierte Sicherheitstechnik" angegeben sind. Ist dies nicht der Fall, sind sie ausgeschlossen. Funktionale Sicherheit beinhaltet Teile der Gesamtsicherheit, bei der Maßnahmen zur Risikominderung zum Personenschutz von elektrischen, elektronischen oder programmierbaren Steuerungen abhängig sind.
- Die in der Anwendungsdokumentation gemachten Angaben zur Verwendung der gelieferten Komponenten stellen nur Anwendungsbeispiele und Vorschläge dar.

  Der Maschinenhersteller und Anlagenerrichter muss für seine individuelle Anwendung die Eignung

  – der gelieferten Komponenten und die in dieser Anwendungsdokumentation gemachten Angaben zu ihrer Verwendung selbst überprüfen,
  – mit den für seine Anwendung geltenden Sicherheitsvorschriften und Normen abstimmen und die erforderlichen Maßnahmen, Änderungen, Ergänzungen durchführen.
- Die Inbetriebnahme der gelieferten Komponenten ist solange untersagt, bis festgestellt wurde, dass die Maschine oder Anlage, in der die Komponenten eingebaut sind, den länderspezifischen Bestimmungen, Sicherheitsvorschriften und Normen der Anwendung entspricht.
- Der Betrieb ist nur bei Einhaltung der nationalen EMV-Vorschriften für den vorliegenden Anwendungsfall erlaubt.
- Die Hinweise für eine EMV-gerechte Installation sind dem Abschnitt zur EMV in der zugehörigen Anwendungsdokumentation zu entnehmen.

  Die Einhaltung der durch die nationalen Vorschriften geforderten Grenzwerte liegt in der Verantwortung des Herstellers der Anlage oder Maschine.
- Die technischen Daten, die Anschluss- und Installationsbedingungen der Komponenten sind den zugehörigen Anwendungsdokumentationen zu entnehmen und unbedingt einzuhalten.

Länderspezifische Vorschriften, die vom Anwender zu berücksichtigen sind

- Europäische Länder: entsprechend Euronormen EN
- Vereinigte Staaten von Amerika (USA):
  – Nationale Vorschriften für Elektrik (NEC)
  – Nationale Vereinigung der Hersteller von elektrischen Anlagen (NEMA) sowie regionale Bauvorschriften
  – Vorschriften der National Fire Protection Association (NFPA)

Sicherheitshinweise für elektrische Antriebe und Steuerungen

- Kanada: Canadian Standards Association (CSA)
- Andere Länder:
  - International Organization for Standardization (ISO)
  - International Electrotechnical Commission (IEC)

3.2.3 Gefahren durch falschen Gebrauch

- Hohe elektrische Spannung und hoher Arbeitsstrom! Lebensgefahr oder schwere Körperverletzung durch elektrischen Schlag!
- Hohe elektrische Spannung durch falschen Anschluss! Lebensgefahr oder Körperverletzung durch elektrischen Schlag!
- Gefahrbringende Bewegungen! Lebensgefahr, schwere Körperverletzung oder Sachschaden durch unbeabsichtigte Bewegungen der Motoren!
- Gesundheitsgefahr für Personen mit Herzschrittmachern, metallischen Implantaten und Hörgeräten in unmittelbarer Umgebung elektrischer Antriebssysteme!
- Verbrennungsgefahr durch heiße Gehäuseoberflächen!
- Verletzungsgefahr durch unsachgemäße Handhabung! Körperverletzung durch Quetschen, Scheren, Schneiden, Stoßen!
- Verletzungsgefahr durch unsachgemäße Handhabung von Batterien!
- Verletzungsgefahr durch unsachgemäße Handhabung von unter Druck stehenden Leitungen!

3.3 Gefahrenbezogene Hinweise

3.3.1 Schutz gegen Berühren elektrischer Teile und von Gehäusen

☞ Dieser Abschnitt betrifft Komponenten des elektrischen Antriebs- und Steuerungssystems mit Spannungen **über 50 Volt.**

Werden Teile mit Spannungen größer 50 Volt berührt, können diese für Personen gefährlich werden und zu elektrischem Schlag führen. Beim Betrieb von Komponenten des elektrischen Antriebs- und Steuerungssystems stehen zwangsläufig bestimmte Teile dieser Komponenten unter gefährlicher Spannung.

Hohe elektrische Spannung! Lebensgefahr, Verletzungsgefahr durch elektrischen Schlag oder schwere Körperverletzung!

- Bedienung, Wartung und/oder Instandsetzung der Komponenten des elektrischen Antriebs- und Steuerungssystems darf nur durch qualifiziertes Personal erfolgen.
- Beachten Sie die allgemeinen Errichtungs- und Sicherheitsvorschriften zu Arbeiten an Starkstromanlagen.
- Stellen Sie vor dem Einschalten den festen Anschluss des Schutzleiters an allen elektrischen Komponenten entsprechend dem Anschlussplan her.

Sicherheitshinweise für elektrische Antriebe und Steuerungen

- Ein Betrieb, auch für kurzzeitige Mess- und Prüfzwecke, ist nur mit fest angeschlossenem Schutzleiter an den dafür vorgesehenen Punkten der Komponenten erlaubt.
- Trennen Sie elektrische Komponenten vom Netz oder von der Spannungsquelle, bevor Sie auf elektrische Teile mit Spannungen größer 50 V zugreifen. Sichern Sie die elektrische Komponente gegen Wiedereinschalten.
- Bei elektrischen Komponenten beachten:

  Warten Sie nach dem Abschalten grundsätzlich **30 Minuten**, damit sich spannungsführende Kondensatoren entladen können, bevor Sie auf eine elektrische Komponente zugreifen. Messen Sie die elektrische Spannung von spannungsführenden Teilen vor Beginn der Arbeiten, um Gefährdungen durch Berührung auszuschließen.
- Bringen Sie vor dem Einschalten die dafür vorgesehenen Abdeckungen und Schutzvorrichtungen für den Berührschutz an.
- Berühren Sie elektrische Anschlussstellen der Komponenten im eingeschalteten Zustand nicht.
- Ziehen Sie Stecker nicht unter Spannung ab oder stecken Sie diese nicht unter Spannung auf.
- Elektrische Antriebssysteme können unter bestimmten Voraussetzungen an Netzen betrieben werden, die durch allstromsensitive FI-Schutzeinrichtungen (RCD/RCM) abgesichert sind.
- Für Einbaugeräte ist der Schutz gegen Eindringen von Fremdkörpern und Wasser sowie gegen direktes Berühren durch ein äußeres Gehäuse, z. B. Schaltschrank, sicherzustellen.

Hohe Gehäusespannung und hoher Ableitstrom! Lebensgefahr, Verletzungsgefahr durch elektrischen Schlag!

- Erden oder verbinden Sie vor dem Einschalten und der Inbetriebnahme die Komponenten des elektrischen Antriebs- und Steuerungssystems mit dem Schutzleiter an den Erdungspunkten.
- Schließen Sie den Schutzleiter der Komponenten des elektrischen Antriebs- und Steuerungssystems stets fest und dauerhaft an das Versorgungsnetz an. Der Ableitstrom ist größer als 3,5 mA.
- Stellen Sie eine Schutzleiterverbindung mit einem Mindestquerschnitt gemäß nachfolgender Tabelle her. Bei einem Außenleiterquerschnitt kleiner 10 mm^2 ist alternativ auch ein Anschluss von zwei Schutzleitern zulässig, welche jeweils den gleichen Querschnitt wie die Außenleiter aufweisen.

Querschnitt Außenleiter	Mindestquerschnitt Schutzleiter Ableitstrom ≥ 3,5 mA	
	1 Schutzleiter	2 Schutzleiter
1,5 mm^2 (AWG 16)	10 mm^2 (AWG 8)	2 × 1,5 mm^2 (AWG 16)
2,5 mm^2 (AWG 14)		2 × 2,5 mm^2 (AWG 14)
4 mm^2 (AWG 12)		2 × 4 mm^2 (AWG 12)
6 mm^2 (AWG 10)		2 × 6 mm^2 (AWG 10)
10 mm^2 (AWG 8)		-

Sicherheitshinweise für elektrische Antriebe und Steuerungen

Querschnitt Außenleiter	Mindestquerschnitt Schutzleiter Ableitstrom ≥ 3,5 mA	
	1 Schutzleiter	2 Schutzleiter
16 mm² (AWG 6)	16 mm² (AWG 6)	-
25 mm² (AWG 4)		-
35 mm² (AWG 2)		-
50 mm² (AWG 1/0)	25 mm² (AWG 4)	-
70 mm² (AWG 2/0)	35 mm² (AWG 2)	-
...	...	...

Abb.3-1: Mindestquerschnitt der Schutzleiterverbindung

3.3.2 Schutzkleinspannung als Schutz gegen elektrischen Schlag

Schutzkleinspannung dient dazu, Geräte mit Basisisolierung an Kleinspannungskreise anschließen zu können.

An den Komponenten des elektrischen Antriebs- und Steuerungssystems von Bosch Rexroth sind alle Anschlüsse und Klemmen, die Spannungen von 5 bis 50 Volt führen, in Schutzkleinspannung ("Protective Extra Low Voltage - PELV") ausgeführt. An diese Anschlüsse dürfen Geräte angeschlossen werden, die mit Basisisolierung ausgestattet sind, wie beispielsweise Programmiergeräte, PCs, Notebooks, Anzeigegeräte.

Lebensgefahr, Verletzungsgefahr durch elektrischen Schlag! Hohe elektrische Spannung durch falschen Anschluss!

Werden Kleinspannungskreise von Geräten, die auch Spannungen und Stromkreise über 50 Volt beinhalten (z. B. den Netzanschluss), an Produkten von Bosch Rexroth angeschlossen, dann müssen die angeschlossenen Kleinspannungskreise die Anforderungen für Schutzkleinspannung ("Protective Extra Low Voltage - PELV") erfüllen.

3.3.3 Schutz vor gefährlichen Bewegungen

Gefährliche Bewegungen können durch fehlerhafte Ansteuerung von angeschlossenen Motoren verursacht werden. Die Ursachen können verschiedenster Art sein:

- unsachgemäße oder fehlerhafte Verdrahtung oder Verkabelung
- Bedienungsfehler
- falsche Eingabe von Parametern vor der Inbetriebnahme
- Fehler in den Messwertgebern und Signalgebern
- defekte Komponenten
- Fehler in der Software oder in der Firmware

Diese Fehler können unmittelbar nach dem Einschalten oder nach einer unbestimmten Zeitdauer im Betrieb auftreten.

Die Überwachungen in den Komponenten des elektrischen Antriebs- und Steuerungssystems schließen eine Fehlfunktion in den angeschlossenen Antrieben weitestgehend aus. Im Hinblick auf den Personenschutz, insbesondere auf die Gefahr von Körperverletzung und/oder Sachschaden, darf auf die-

sen Sachverhalt nicht allein vertraut werden. Bis zum Wirksamwerden der eingebauten Überwachungen ist auf jeden Fall mit einer fehlerhaften Antriebsbewegung zu rechnen, deren Maß von der Art der Steuerung und des Betriebszustandes abhängt.

Gefahrbringende Bewegungen! Lebensgefahr, Verletzungsgefahr, schwere Körperverletzung oder Sachschaden!

Für die Anlage oder Maschine mit ihren spezifischen Gegebenheiten, in welche die Komponenten des elektrischen Antriebs- und Steuerungssystems eingebaut werden, ist eine **Risikobeurteilung** zu erstellen.

Aus der Risikobeurteilung sind vom Anwender Überwachungen und anlagenseitig übergeordnete Maßnahmen für den Personenschutz vorzusehen. Die für die Anlage oder Maschine geltenden Sicherheitsbestimmungen sind hierbei mit einzubeziehen. Durch Ausschalten, Umgehen oder fehlendes Aktivieren von Sicherheitseinrichtungen können willkürliche Bewegungen der Maschine oder andere Fehlfunktionen auftreten.

Vermeidung von Unfällen, Körperverletzung und/oder Sachschaden:

- Halten Sie sich nicht im Bewegungsbereich der Maschine und Maschinenteilen auf. Verhindern Sie den unbeabsichtigten Zugang für Personen, z. B. durch
  - Schutzzaun
  - Schutzgitter
  - Schutzabdeckung
  - Lichtschranke
- Stellen Sie eine ausreichende Festigkeit der Schutzzäune und Schutzabdeckungen gegen die maximal mögliche Bewegungsenergie sicher.
- Ordnen Sie NOT-HALT-Schalter leicht zugänglich und schnell erreichbar an. Prüfen Sie die Funktion der NOT-HALT-Einrichtung vor der Inbetriebnahme. Unterlassen Sie den Betrieb der Maschine bei Fehlfunktion des NOT-HALT-Schalters.
- Stellen Sie sicher, dass es nicht zu einem unbeabsichtigten Anlauf kommt. Schalten Sie den Leistungsanschluss der Antriebe über Ausschalter/-taster frei oder verwenden Sie eine sichere Anlaufsperre.
- Bringen Sie vor dem Zugriff oder Zutritt in den Gefahrenbereich die Antriebe sicher zum Stillstand.
- Sichern Sie zusätzlich vertikale Achsen gegen Herabfallen oder Absinken nach Abschalten des Motors, z. B. durch
  - mechanische Verriegelung der vertikalen Achse,
  - externe Brems-/ Fang-/ Klemmeinrichtung oder
  - ausreichenden Gewichtsausgleich der Achse.
- Die serienmäßig gelieferte **Motor-Haltebremse** oder eine externe, vom Antriebsregelgerät angesteuerte Haltebremse **alleine ist nicht für den Personenschutz geeignet!**
- Schalten Sie die Komponenten des elektrischen Antriebs- und Steuerungssystems über den Hauptschalter spannungsfrei und sichern Sie diese gegen Wiedereinschalten bei:
  - Wartungsarbeiten und Instandsetzung
  - Reinigungsarbeiten
  - langen Betriebsunterbrechungen

Sicherheitshinweise für elektrische Antriebe und Steuerungen

- Vermeiden Sie den Betrieb von Hochfrequenz-, Fernsteuer- und Funkgeräten in der Nähe von Komponenten des elektrischen Antriebs- und Steuerungssystems und deren Zuleitungen. Wenn ein Gebrauch dieser Geräte unvermeidlich ist, prüfen Sie bei Erstinbetriebnahme des elektrischen Antriebs- und Steuerungssystems die Maschine oder Anlage auf mögliche Fehlfunktionen bei Betrieb solcher Hochfrequenz-, Fernsteuer- oder Funkgeräte in deren möglichen Gebrauchslagen. Eventuell ist eine spezielle EMV-Prüfung notwendig.

3.3.4 Schutz vor magnetischen und elektromagnetischen Feldern bei Betrieb und Montage

Magnetische und elektromagnetische Felder, die in unmittelbarer Umgebung von stromführenden Leitern oder Permanentmagneten von Elektromotoren bestehen, können eine ernste Gefahr für Personen mit Herzschrittmachern, metallischen Implantaten und Hörgeräten darstellen.

Gesundheitsgefahr für Personen mit Herzschrittmachern, metallischen Implantaten und Hörgeräten in unmittelbarer Umgebung elektrischer Komponenten!

- Personen mit Herzschrittmachern und metallischen Implantaten ist der Zugang zu folgenden Bereichen untersagt:
  - Bereiche, in denen Komponenten der elektrischen Antriebs- und Steuerungssysteme montiert, in Betrieb genommen und betrieben werden
  - Bereiche, in denen Motorenteile mit Dauermagneten gelagert, repariert oder montiert werden
- Besteht die Notwendigkeit für Träger von Herzschrittmachern derartige Bereiche zu betreten, so ist das zuvor von einem Arzt zu entscheiden. Die Störfestigkeit von implantierten Herzschrittmachern ist sehr unterschiedlich, somit bestehen keine allgemein gültigen Regeln.
- Personen mit Metallimplantaten oder Metallsplittern sowie mit Hörgeräten haben vor dem Betreten derartiger Bereiche einen Arzt zu befragen.

3.3.5 Schutz gegen Berühren heißer Teile

Heiße Oberflächen von Komponenten des elektrischen Antriebs- und Steuerungssystems. Verbrennungsgefahr!

- Vermeiden Sie das Berühren von heißen Oberflächen von z. B. Bremswiderständen, Kühlkörpern, Versorgungs- und Antriebsregelgeräten, Motoren, Wicklungen und Blechpaketen!
- Temperaturen der Oberflächen können während oder nach dem Betrieb je nach Betriebsbedingungen **über 60 °C** (140 °F) liegen.
- Lassen Sie die Motoren nach dem Abschalten ausreichend lange abkühlen, bevor Sie diese berühren. Abkühlzeiten **bis 140 Minuten** können erforderlich sein! Die erforderliche Abkühlzeit ist ungefähr fünfmal so groß wie die in den technischen Daten angegebene thermische Zeitkonstante.
- Lassen Sie Drosseln, Versorgungs- und Antriebsregelgeräte **15 Minuten** lang nach dem Abschalten abkühlen, bevor Sie diese berühren.
- Tragen Sie Schutzhandschuhe oder arbeiten Sie nicht an heißen Oberflächen.

- Für bestimmte Anwendungen sind nach den Sicherheitsvorschriften Maßnahmen zur Verhinderung von Verbrennungsverletzungen in der Endanwendung vom Hersteller der Maschine oder Anlage vorzunehmen. Diese Maßnahmen können beispielsweise sein: Warnhinweise an der Maschine oder Anlage, trennende Schutzeinrichtung (Abschirmung oder Absperrung) oder Sicherheitshinweise in der Anwendungsdokumentation.

3.3.6 Schutz bei Handhabung und Montage

Verletzungsgefahr durch unsachgemäße Handhabung! Körperverletzung durch Quetschen, Scheren, Schneiden, Stoßen!

- Beachten Sie die einschlägigen Vorschriften zur Verhütung von Unfällen (z. B. Unfallverhütungsvorschriften).
- Verwenden Sie geeignete Montage- und Transporteinrichtungen.
- Beugen Sie Einklemmungen und Quetschungen durch geeignete Vorkehrungen vor.
- Benutzen Sie nur geeignetes Werkzeug, sofern vorgeschrieben, Spezialwerkzeug.
- Setzen Sie Hebeeinrichtungen und Werkzeuge fachgerecht ein.
- Benutzen Sie geeignete Schutzausstattung (z. B. Schutzhelm, Schutzbrille, Sicherheitsschuhe, Schutzhandschuhe).
- Halten Sie sich nicht unter hängenden Lasten auf.
- Beseitigen Sie ausgelaufene Flüssigkeiten am Boden sofort, ansonsten besteht Sturzgefahr!

3.3.7 Schutz beim Umgang mit Batterien

Batterien bestehen aus aktiven Chemikalien in einem festen Gehäuse. Unsachgemäßer Umgang kann daher zu Verletzungen oder Sachschäden führen.

Verletzungsgefahr durch unsachgemäße Handhabung!

- Versuchen Sie nicht, leere Batterien durch Erhitzen oder andere Methoden zu reaktivieren (Explosions- und Ätzungsgefahr).
- Versuchen Sie nicht, Batterien aufzuladen, weil sie dabei auslaufen oder explodieren können.
- Werfen Sie Batterien nicht ins Feuer.
- Zerlegen Sie keine Batterie.
- Beschädigen Sie beim Wechsel der Batterie(n) nicht die elektrischen Bauteile in den Geräten.
- Verwenden Sie nur die für das Produkt vorgeschriebenen Batterietypen.

☞ Umweltschutz und Entsorgung! Die im Produkt enthaltenen Batterien sind im Sinne der gesetzlichen Bestimmungen als Gefahrengut beim Transport im Land-, Luft- und Seeverkehr anzusehen (Explosionsgefahr). Entsorgen Sie Altbatterien getrennt von anderem Abfall. Beachten Sie die nationalen Bestimmungen Ihres Landes.

Sicherheitshinweise für elektrische Antriebe und Steuerungen

3.3.8 Schutz vor unter Druck stehenden Leitungen

Flüssigkeits- und druckluftgekühlte Motoren und Komponenten können entsprechend den Angaben in den Projektierungsanleitungen zum Teil mit extern zugeführten und unter Druck stehenden Medien wie Druckluft, Hydrauliköl, Kühlflüssigkeit und Kühlschmiermittel versorgt werden. Unsachgemäßer Umgang mit den angeschlossenen Versorgungssystemen, Versorgungsleitungen oder Anschlüssen kann zu Verletzungen oder Sachschäden führen.

Verletzungsgefahr durch unsachgemäße Handhabung von unter Druck stehenden Leitungen!

- Versuchen Sie nicht, unter Druck stehende Leitungen zu trennen, zu öffnen oder zu kappen (Explosionsgefahr).
- Beachten Sie die Betriebsvorschriften der jeweiligen Hersteller.
- Lassen Sie vor Demontage von Leitungen, Druck und Medium ab.
- Benutzen Sie geeignete Schutzausrüstung (z. B. Schutzbrillen, Sicherheitsschuhe, Schutzhandschuhe).
- Beseitigen Sie ausgelaufene Flüssigkeiten am Boden sofort, ansonsten besteht Sturzgefahr!

☞ Umweltschutz und Entsorgung! Die für den Betrieb des Produktes verwendeten Medien können unter Umständen nicht umweltverträglich sein. Entsorgen Sie umweltschädliche Medien getrennt von anderem Abfall. Beachten Sie die nationalen Bestimmungen Ihres Landes.

3.4 Erläuterung der Signalwörter und der Signalgrafik

Die Sicherheitshinweise in der vorliegenden Anwendungsdokumentation beinhalten bestimmte Signalwörter (Gefahr, Warnung, Vorsicht, Hinweis) und ggf. eine Signalgrafik (nach ANSI Z535.6-2011).

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

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

⚠ GEFAHR

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

⚠ WARNUNG

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

⚠ VORSICHT

Bei Nichtbeachtung dieses Sicherheitshinweises können mittelschwere oder leichte Körperverletzung eintreten.

Sicherheitshinweise für elektrische Antriebe und Steuerungen

HINWEIS

Bei Nichtbeachtung dieses Sicherheitshinweises können Sachschäden eintreten.

4 Einführung

Mit dem **G**eneric **A**pplication **T**emplate (GAT) entwickeln Sie in kurzer Zeit gut strukturierte Applikationsprogramme für Motion und Logic Applikationen.

Das GAT ist ein einschaltfertiger Programmierrahmen indem Sie Ihren Applikationscode an vordefinierten Stellen eintragen. Durch die **dialogbasierte Codegenerierung** des GAT-Wizards erstellen Sie professionell gut strukturierte Applikationsprogramme mit vorbereiteten Elementen.

Vor der Programmierung Ihrer Applikation und Anwendung des GAT empfehlen wir folgendes Kapitel durchzulesen:

Grundlegendes Vorgehen bei der Automatisierung von Applikationen, Seite 39,

Je nach Komplexität und Anwendungsfall stehen folgende Varianten zur Verfügung:

GAT-Variante	Beschreibung	Verfügbar für Zielsystem
AxisInterface Beispielprojekt	Beispielprogramm zur Achsinterface Anwendung (vereinfachte Achskommandierung) Kap. 6.6.2 "Programm zum Achs-Interface" auf Seite 89	IndraLogic XLC IndraMotion MLC
GAT$_{compact}$	Nichtmodulares Beispielprogramm für **einfache** Applikationen mit wenigen Achsen Kap. 8 "GAT^{compact}" auf Seite 211 Kap. 9 "GAT^{compact} für IndraLogic XLC ohne sercos Interface" auf Seite 255	IndraLogic XLC IndraMotion MLC
GAT (zentral)	Programmrahmen für flexible **modulare** Maschinen-Konzepte mit **zentraler** Betriebsarten-Schrittkette Kap. 10 "GAT mit zentraler Zustandsmaschine" auf Seite 275	IndraMotion MLC
GAT (dezentral)	Programmrahmen für flexible **modulare** Maschinen-Konzepte mit **dezentraler** Betriebsarten-Schrittkette Kap. 11 "GAT mit dezentraler Zustandsmaschine" auf Seite 347	IndraMotion MLC

Abb.4-1: Verfügbare GAT-Varianten

GAT und GAT^{compact} nutzen einen auf PLCopen basierenden vereinfachten Zugriff auf Achsen (Achs-Interface) bzw. die Steuerung (**I**ndra**M**otion **C**ontrol-Interface). Zur Beschreibung von Achs- bzw. IMC-Interface[1] siehe Kap. 6 "ML_TechInterface.library" auf Seite 55.

Alle Ausprägungen des GAT sind integraler Bestandteil von IndraWorks.

GAT-Wizard

Mit dem GAT-Wizard führen Sie den GAT-Erstimport aus und editieren Ihr bestehendes GAT-Projekt. Durch die Dialoge des GAT-Wizard erstellen Sie in kurzer Zeit gut strukturierte Applikationsprogramme.

1) *IndraMotionControl Interface = IndraLogic Bibliothek mit Schnittstelle zur Steuerung, wie. z.B. Fehler löschen*

Einführung

Über den GAT-Button  in der IndraWorks Werkzeugleiste rufen Sie den GAT-Wizard auf.



Der GAT-Wizard bezieht sich immer auf die aktive Applikation im IndraWorks Projekt.

5 Automatisierung von Applikationen

5.1 Allgemeines

Bei der Automatisierung einer Maschine sind erfahrungsgemäß folgende Aufgaben zu lösen:

- Initialisierung
- Betriebsartenverwaltung
- Zyklische Bearbeitung
- Taskhandling/Rechenzeitaufteilung
- Diagnose/Fehlerhandling
- Visualisierung

Im Folgenden wird beschrieben, wie diese Aufgaben grundlegend zu lösen sind, auch unter Berücksichtigung der Besonderheiten von Motion-Applikationen. Hinsichtlich der Kommunikation zwischen Steuerungen und Antrieben steht in diesem Zusammenhang insbesondere der sercos Bus im Fokus. Task- und Fehlerhandling sowie die Visualisierung werden auch im Kontext des IndraWorks Engineering Frameworks betrachtet.

5.2 Initialisierung

Zu Beginn eines Maschinenhochlaufs müssen zumeist anlagenspezifische Parameter und Variablen in der Steuerung und in zugehörigen Antrieben gesetzt werden, um den Betrieb vorzubereiten. Dieser Vorgang wird Initialisierung genannt.

Dabei muss unterschieden werden zwischen Initialisierungen in Kommunikationsphase 2 (P2, Parametriermodus) und Kommunikationsphase 4 (BB, Betriebsmodus).

Kommunikationsphase 2 bedeutet, dass sich Steuerung und Antriebe im Parametriermodus befinden. In diesem Modus sind viele Parameter beschreibbar, die in der Kommunikationsphase 4 (Betriebsmodus) nicht mehr beschreibbar sind.

In der Kommunikationsphase 2 werden:

- üblicherweise die Kommunikationsparameter vom Master an den Antrieb übertragen (bei Feldbus- und sercos Geräten) und
- bei Bedarf die Funktionen "Antriebsparameter laden" und "Antriebsparameter sichern" ("File-Service") durchgeführt

Typischerweise werden bei der Initialisierung nachfolgende Aktivitäten durchgeführt, um die Maschine für den Betrieb vorzubereiten:

- Firmware der Antriebe überprüfen
- Vorhandene Hardware scannen
- Achsen konfigurieren
- Steuerungs- und Antriebsparameter überprüfen bzw. setzen
- SPS-Variablen/FB setzen/zurücksetzen
- Am Ende der Initialisierung Drehmoment auf Achse schalten
- Phasenumschaltung der Anlage in den Betriebsmodus (P4)

Automatisierung von Applikationen

5.3 Betriebsarten

5.3.1 Allgemeines

Typische Betriebsarten jeder Applikation sind **Automatik** und **Manual**. Darüber hinaus kann es weitere Betriebsarten geben (z. B. Cleaning).

Üblicherweise erfolgt ein Betriebsartenwechsel in Motion-Applikationen nur, wenn alle Achsen gestoppt sind.

5.3.2 Automatik

Die Automatik-Betriebsart in Motion-Applikationen ist im Allgemeinen dadurch gekennzeichnet, dass sich die Achsen koordiniert zueinander bewegen. So folgen oftmals eine oder mehrere Slave-Achsen einer Master-Achse geschwindigkeits- oder positionssynchron. Zusätzlich kann eine Slave-Achse dabei ein applikationsspezifisches Bewegungsprofil (Kurvenscheibe) abfahren. Die Achsen bewegen sich ohne Einwirkung des Bedieners.

Um ein besseres Handling bzw. eine bessere Übersicht zu erreichen, wird in komplexen Applikationen die Automatik-Betriebsart oftmals in mehrere Zustände unterteilt.

5.3.3 Manual

Die Manual-Betriebsart dient zum Einrichten oder Justieren der Maschine. Typisch ist, dass jede Achse mit einem Geschwindigkeitsprofil oder inkrementell bewegt (getippt, gejoggt) wird. Dabei können die Achsen auch synchronisiert zueinander verfahren werden. In Abhängigkeit von der Komplexität kann auch die Manual-Betriebsart in mehrere Zustände unterteilt sein.

5.4 Zyklische Bearbeitung

5.4.1 Allgemeines

Alle betriebsartenunabhängigen Automatisierungsaufgaben sollten in einen eigenen Programmteil zusammengefasst werden. Dieser kann dann, z. B. innerhalb einer zyklischen Task, abgearbeitet werden.

Im Folgenden werden typische Automatisierungsaufgaben für die zyklische Bearbeitung dargestellt.

5.4.2 Betriebsartenunabhängige Logik

Üblicherweise wird die betriebsartenunabhängige Logik in separaten Programmteilen behandelt. Diese wird im Allgemeinen zyklisch bearbeitet. In dieser zyklischen Task sollten alle betriebsartenunabhängigen Aufgaben programmiert werden (z. B. Tippen im Automatikbetrieb nicht zulassen).

5.4.3 Generierung und Anzeige von Diagnosemeldungen

Im täglichen Einsatz benötigt der Anwender aktuelle Informationen über den Zustand der Anlage (z. B. aktuelle Betriebsart, Temperatur, Hydraulikdruck, …). Hierzu müssen an zentraler Stelle die notwendigen Informationen gesammelt und angepasste, aussagekräftige Statusmeldungen erzeugt werden. Diese werden, z. B. zur Anzeige auf der Anlagen-HMI, bereitgestellt. Weitere Informationen siehe Kap. 5.7 "Visualisierung" auf Seite 49.

5.5 Taskhandling

5.5.1 Allgemeines

Bei der Automatisierung ist eine Teilaufgabe die sinnvolle Aufteilung der Applikation auf Tasks.

Gleichzeitig muss die Rechenzeit gemessen werden, um die Auslastung des Systems beurteilen zu können.

5.5.2 Grundsätze des Taskhandlings

Die folgenden Grundsätze sollten immer beachtet werden:

- so viele Tasks wie nötig, so wenig Tasks wie möglich
  - Multitasking erhöht die Komplexität des Systems
  - Taskwechsel kosten Rechenzeit
- SPS-Tasks sind grundsätzlich mit einem Watchdog zu betreiben (Ausnahme: freilaufende BackgroundTask). Ansonsten kann es zu unvorhergesehenem Systemverhalten kommen
  - Scharf: Watchdogzeit=Taskzykluszeit und Empfindlichkeit=1
  - Normal: Watchdogzeit=(Taskzykluszeit+50..100%) und Empfindlichkeit=2
  - Optionaler Watchdog im Anwendungsprogramm als Vorwarnung: Es ist mit Hilfe von Zeitmessfunktionen möglich, einen "Warn-Watchdog" mit z. B. 90% der Zeit des System-Watchdogs im Anwendungsprogramm zu implementieren. Damit kann eine Warnung generiert und Gegenmaßnahmen ergriffen werden
- Ein System ist optimal ausgelastet, wenn
  - die kleinstmögliche Steuerungsvariante für diese Aufgabe eingesetzt werden kann
  - alle Aufgaben der Steuerung rechtzeitig ausgeführt werden
  - In keinem jemals auftretenden Betriebszustand ein SPS-Task Watchdog auftritt (Worst case)

5.5.3 Empfohlene Taskaufteilung bei Verwendung von IndraWorks/ IndraLogic

Allgemeines

Wenn die Abarbeitung länger dauert als die eingestellte Watchdog-Zeit, und in IndraWorks/IndraLogic für diese Task der Watchdog aktiviert ist, so wird diese Task mit Fehler-Status (Ausnahmefehler) abgebrochen.

Dabei wird die Empfindlichkeit wie folgt mit eingerechnet:

- bei mehreren aufeinanderfolgenden Watchdog-Zeitüberschreitungen, gibt es:
  - Ausnahmefehler in Zyklus 1, wenn Empfindlichkeit 1 oder 2 ist
  - Ausnahmefehler in Zyklus 2, wenn Empfindlichkeit 3 ist
  - …
  - Ausnahmefehler in Zyklus (n-1), wenn Empfindlichkeit n ist
- Bei einmaliger Zeitüberschreitung gibt es einen Ausnahmefehler, wenn die aktuelle Zykluszeit größer ist als (Watchdogzeit * Empfindlichkeit)

  Z. B.: Aktuelle Zykluszeit = 10ms, Empfindlichkeit = 5 -> Ausnahmefehler, wenn die Task einmalig länger als 50ms läuft.

Automatisierung von Applikationen

Vor diesem Hintergrund werden folgende Tasks und die dazugehörigen zeitlichen Einstellungen empfohlen:

	Prio	Zykluszeit	Watchdogzeit	Empfindlichkeit des Watchdog
MotionTask (optional bei GAT^{compact})	2	= Motion Zykluszeit	T#2..10ms	1..2
PlcTask	~5	T#10..50ms	T#10..100ms	2..3
BackgroundTask (optional bei GAT^{compact})	~6	T#40..100ms	-	-

Abb.5-1:

Typische Aufgaben MotionTask

Grundsätzlich ist die MotionTask für besondere Aufgaben reserviert:

- Handling von Gantryachsen und Achsen mit besonderen Anforderungen
- Handling von Technologie-Funktionsbausteinen mit besonderen Zeitanforderungen (Wickler, Registerregler, Smartbelt, fliegende Säge…)
- Schnelle Fehlerreaktionen
- Handling Schnelle I/O

Typische Aufgaben PlcTask

Die PlcTask ist die zentrale Stelle für alle Standardaufgaben:

- Handling der Achsen, Betriebsarten-Umschaltung
- Maschinenabläufe
- Standard I/O
- Fehlerhandling
- Unterstützung der SI-Technik

Typische Aufgaben BackgroundTask

Die optionale BackgroundTask dient dazu, die Auslastung der PlcTask zu vermindern.

Alle zeitunkritischen Aufgaben können hier erledigt werden, z. B.:

- Generierung und Anzeige von Diagnosemeldungen
- Aufbereitung von Daten für die Visualisierung
- Komplexe Berechnungen (z. B. Kurvenscheiben) zur Vorbereitung des nächsten Produktwechsels

5.5.4 Messung der Zykluszeiten

Zykluszeitendialog der Steuerung

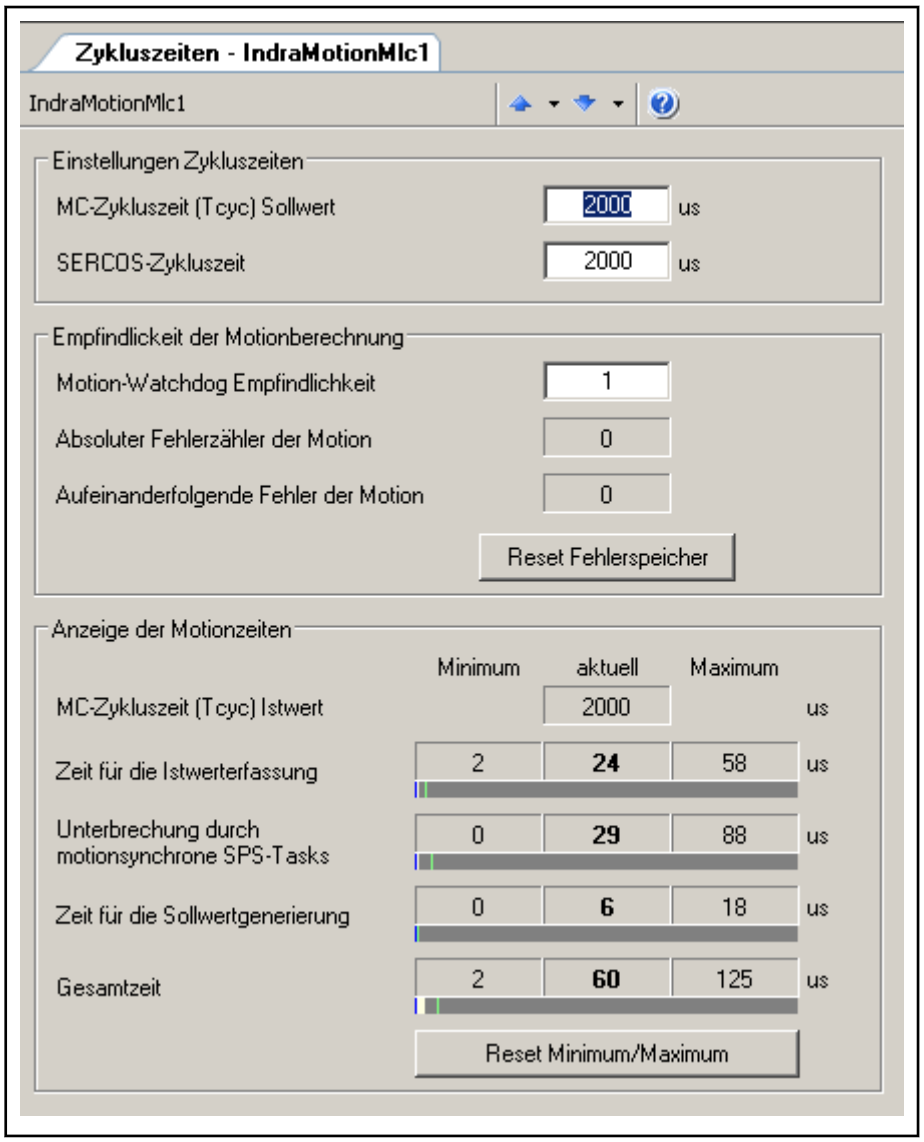


Abb.5-2: Aufteilung der Zykluszeiten von Motion-Control und SPS

Automatisierung von Applikationen

Task Execution Viewer der Steuerung

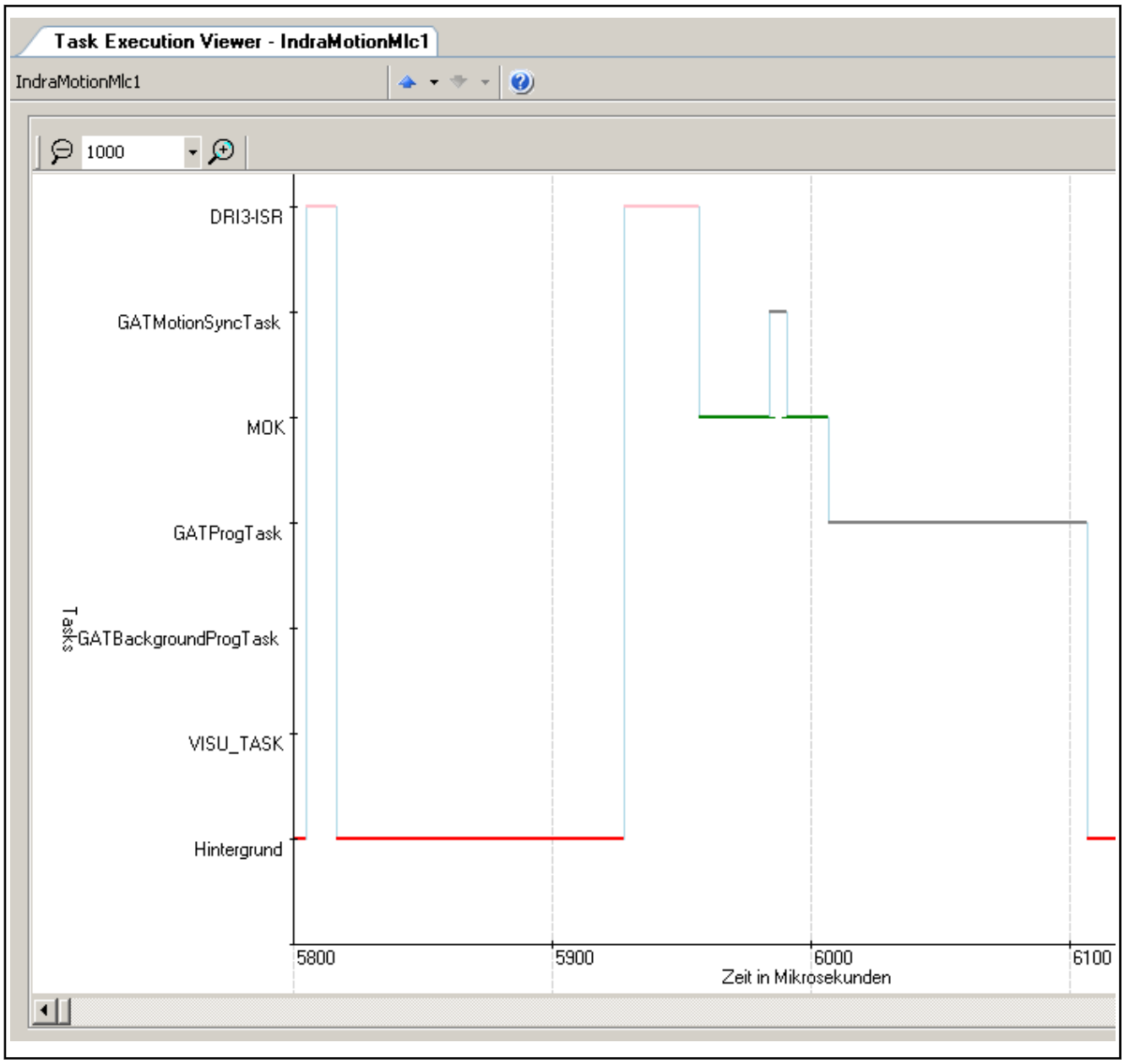


Abb.5-3: Darstellung des zeitlichen Ablaufs der SPS- und ausgewählter Systemtasks

Zeitmessung im SPS-Projekt

Um einzelne Codeabschnitte ausmessen zu können, stellt das Steuerungssystem hochauflösende Zeitmessfunktionen zur Verfügung:

- IL_HighResTimeTick und
- IL_HighResTimeDiff

in der Bibliothek RIL_UTILITIES.

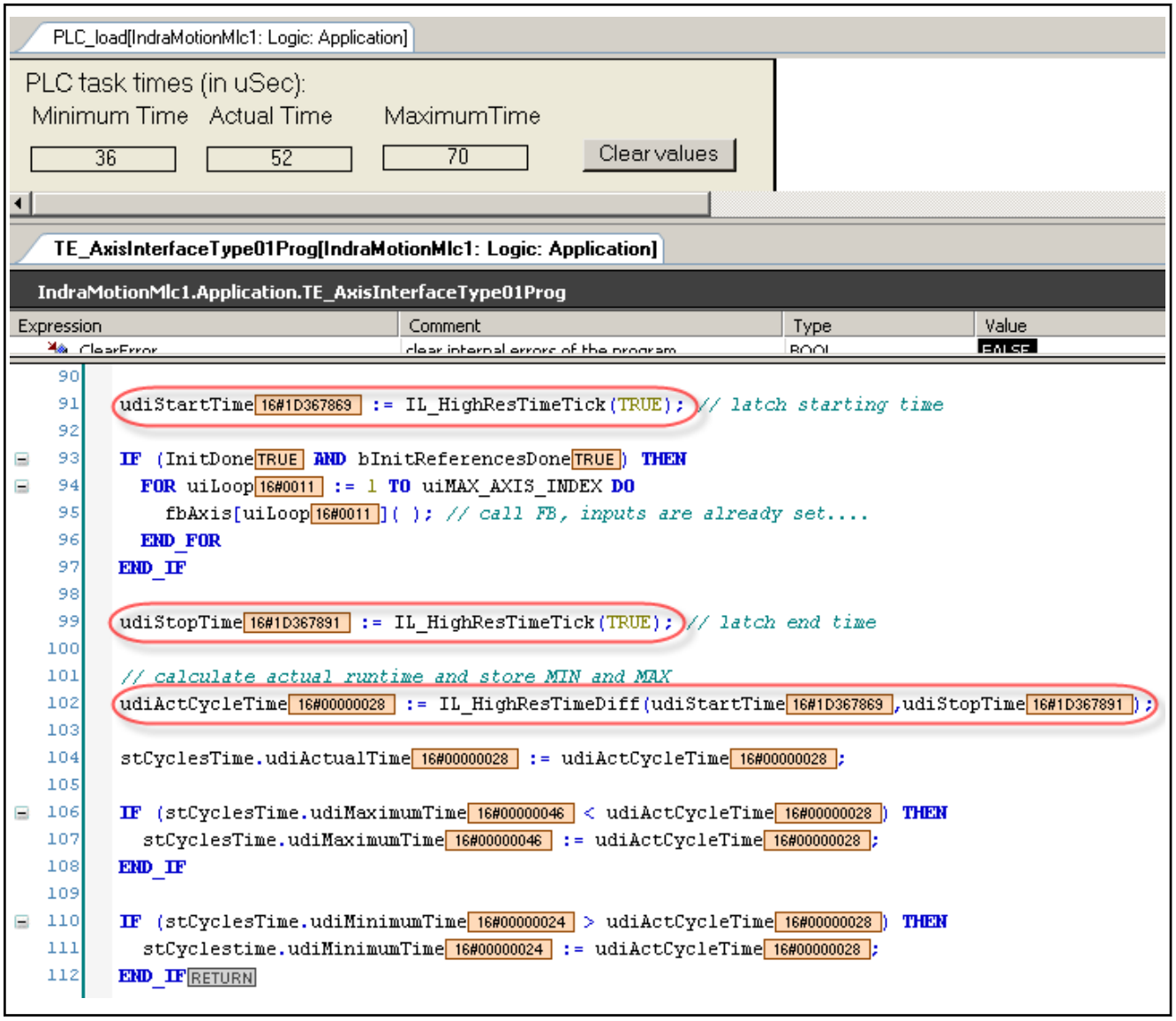


Abb.5-4: Zeitmessung im SPS-Projekt

Mit diesen Zeitmessfunktionen ist es möglich, einen "Warn-Watchdog" mit z. B. 90% der Zeit des System-Watchdogs im Anwendungsprogramm zu implementieren. Damit kann ein(e) Fehler/Warnung implementiert werden, der nicht zum Stillstand der Maschine führt. Im Gegensatz dazu stoppt der System-Watchdog die SPS.

5.5.5 Applikationsspezifischer Watchdog

Wird in der Taskkonfiguration von IndraLogic ein Watchdog angegeben, so wird das SPS-Programm abgebrochen, sobald dieser Watchdog anspricht. Somit kann nicht mehr applikativ, innerhalb des SPS-Programms, auf dieses Ereignis reagiert werden.

Durch Implementierung eines applikationsspezifischen Watchdogs, d. h. interne Zykluszeitmessung, kann im Falle einer Zykluszeitüberschreitung innerhalb des SPS-Programms noch reagiert werden. Die hierfür verwendbaren Funktionen sind in Kap. "Zeitmessung im SPS-Projekt" auf Seite 44 beschrieben.

Automatisierung von Applikationen

5.6 Fehlerhandling

5.6.1 Allgemeines

An industrielle Automatisierungssysteme werden besondere Anforderungen im Bereich der Fehlerbehandlung gestellt. So muss sichergestellt sein, dass beim Auftreten von Fehlern Schäden minimiert werden. Weiterhin sind Stillstandzeiten auf ein Minimum zu reduzieren, was mit aussagekräftigen Diagnosen und einem definiertem Wiederanlauf erreicht werden kann.

Im Weiteren wird das Fehlerhandling genauer beschrieben.

5.6.2 Grundlagen des Fehlerhandling

Das Fehlerhandling untergliedert sich in die folgenden 4 Teilbereiche:

1. Fehlererkennung, Seite 46

   Alle möglichen Fehlerquellen müssen abgefragt werden. Fehlerquellen sind z. B. Antriebe (Achs-Interface), Steuerung (IMC-Interface), Funktionsbausteinfehler, Peripheriefehler (I/O, Feldbus), anwenderdefinierbare Fehler (Materialstau…), SPS-Fehler…

2. Fehlerreaktion, Seite 47

   Im Fehlerfall ist der Schaden auf ein Minimum zu begrenzen, was im Regelfall durch eine definierte bestmögliche Stillsetzung erreicht wird. Einige Komponenten (wie z. B. Antriebe) bringen hierzu bereits eine Fehlerreaktion mit. Zusätzlich ist oft auch eine anwenderdefinierbare Fehlerreaktion notwendig (wie z. B. synchronisiertes Herunterfahren der Antriebe damit z. B. das Papier oder die Folie möglichst nicht reißt).

3. Fehlerdiagnose, Seite 48

   Um Stillstandszeiten gering zu halten, muss die Fehlerursache schnellstmöglich ermittelt werden. Bei verteilten Systemen sollte hierzu die Diagnose an einer zentralen Stelle bereitgestellt werden. Weiterhin sollten möglichst Klartextmeldungen angezeigt werden. Der angezeigte Fehler ist in einigen Fällen ein Folgefehler, der nur bedingt zur Fehlerursache führt. Daher ist ein Logbuch als zusätzliches Diagnosetool oft sehr hilfreich, um den eigentlichen Fehler-Verursacher zu finden und das Fehlerszenario besser zu analysieren.

4. Fehlerquittierung / Wiederanlauf nach Fehler, Seite 49

   Ist die Fehlerursache behoben, muss der Produktionsbetrieb schnellstmöglich wieder aufgenommen werden. Die Fehlerquittierung soll hierbei an nur einer zentralen Stelle erfolgen. Zusätzlich sind oft anwendungsseitig Schritte notwendig, um in den fehlerfreien Produktionsbetrieb zu kommen (z. B. Rücksetzen der anwenderdefinierbaren Fehler, Starten der virtuellen Leitachse…).

5.6.3 Fehlererkennung

Bei der Erstellung eines Anwendungsprogramms sind typischerweise folgende Fehlerquellen abzufragen und an einer zentralen Stelle dem Maschinenbediener bereitzustellen.

Fehlerquelle	Abfrage/Erkennung
Antrieb	• Ohne Achs-Interface: AxisData [AxisRef.Axis-No].Error Bit • Mit Achs-Interface: MB_AXIS_DIAGNOSIS.Error Bit
Steuerung	• IndraLogic XLC-/IndraMotion MLC-Steuerung ohne IMC-Interface: ML_CONTROLDATA.C_0_0023.Bit26 • IndraMotion MLD-Steuerung ohne IMC-Interface: P-0-0115.Bit13 • IndraLogic XLC-/IndraMotion MLC- und IndraMotion MLD-Steuerung mit IMC-Interface: MB_IMC_DIAG-NOSIS.Error
Funktionsbaustein	• Error-Ausgang des aufgerufenen FB
Peripherie	• Feldbusspezifisch, siehe z. B. Bibliothek RIL_Profi-busDP_02.lib
Anwendungsspezifisch	• Anwenderdefiniert

IMC-Interface **I**ndra**M**otion**C**ontrol Interface = IndraLogic Bibliothek mit Schnittstelle zur Steuerung, wie. z. B. Fehler löschen

5.6.4 Fehlerreaktion

Nach Fehlerauftritt folgt im Regelfall eine Fehlerreaktion, um Maschinen- und Material-Schäden zu vermeiden. Je nach Fehlerquelle wird vom Basissystem bereits eine Fehlerreaktion eingeleitet (siehe nachfolgende Tabelle). Ergänzend kann im Anwendungsprogramm eine weiterführende anwenderspezifische Fehlerreaktion umgesetzt werden (z. B. definiertes Herunterfahren einer virtuellen Leitachse).

Fehlerquelle	Reaktion
Antrieb	• Reaktion je nach Fehlerschweregrad: – F0..F2: Bestmögliche Stillsetzung -> gemäß eingestellter Parametrierung – F6..F7: Geschwindigkeitssollwert Nullschaltung – F8: Momentenfreischaltung • Nur fehlerbehaftete Achse wird stillgesetzt • Anwendungsprogramm läuft weiter
	Hinweis: Die Reaktion auf nichtfatale Antriebsfehler ist konfigurierbar (siehe auch Kapitel "Fehlerreaktion" der "Rexroth IndraMotion MLC xxVRS Funktionsbeschreibung") xx - Ausgabe

Automatisierung von Applikationen

Fehlerquelle	Reaktion
Steuerung	• IndraLogic XLC/IndraMotion MLC: Alle Achsen werden stillgesetzt -> Zwangshalt mit Verzögerungsrampe A-0-0224 (Schnellhalt Verzögerung) • IndraMotion MLD: Reaktion parametrierbar • Anwendungsprogramm läuft nicht in allen Fällen weiter: – sercos Unterbrechung: Anwendungsprogramm läuft weiter – Zykluszeitüberschreitung Motion Kernel: Anwendungsprogramm läuft nicht weiter
Funktionsbaustein	• Unterschiedlich, Reaktion gemäß FB-Dokumentation • Anwendungsprogramm läuft weiter
Peripherie	• Keine Reaktion • Anwendungsprogramm läuft weiter
Anwendungsspezifisch	• Reaktion ist anwenderspezifisch • Anwendungsprogramm läuft weiter

5.6.5 Fehlerdiagnose

Zur Bestimmung der Fehlerursache bieten die Komponenten des Steuerungssystems umfangreiche Diagnosemeldungen (siehe nachfolgende Tabelle). So beinhalten die Fehlermeldungen von Antrieben und Steuerung zusätzlich einen Schweregrad. Generell sollten Fehler mit dem höchsten Schweregrad zuerst angezeigt, Diagnosemeldungen an zentraler Stelle zusammengefasst und dem Maschinenbediener bereitgestellt werden, um eine schnelle Diagnose sicherzustellen.

Fehlerquelle	Fehler-Diagnose
Antrieb	• Ohne Achs-Interface: A-0-0023 (nur bei XLC/MLC), S-0-0095, S-0-0390 • Mit Achs-Interface: MB_AXIS_DIAGNOSIS.Message
Steuerung	• IndraLogic XLC/IndraMotion MLC ohne IMC-Interface – C-0-0650,C-0-0627 • IndraMotion MLD ohne IMC-Interface – P-0-0115; P1351 • IndraMotion MLC & IndraMotion MLD mit IMC-Interface – MB_IMC_DIAGNOSIS.Number – MB_IMC_DIAGNOSIS.Message
Funktionsbaustein	• ErrorID, ErrorIdent am FB-Ausgang

Fehlerquelle	Fehler-Diagnose
Peripherie	• Siehe Feldbus-/Inline-Dokumentation
Anwendungsspezifisch	• Anwendungsspezifisch

IMC-Interface IndraMotionControl Interface = IndraLogic Bibliothek mit Schnittstelle zur Steuerung, wie. z. B. Fehler löschen

5.6.6 Fehlerquittierung/Wiederanlauf nach Fehler

Nach Beseitigung der Fehlerursache muss der Fehler an der Fehlerquelle quittiert werden. Aus Anwendersicht wird ein **zentrales Fehler löschen** gefordert. Daher muss die Fehlerquittierung von der zentralen Stelle zu den Fehlerverursachern im Anwendungsprogramm umgesetzt werden. Die nachfolgende Tabelle zeigt hierbei, wie die Fehler an den entsprechenden Teilkomponenten gelöscht werden.

Fehlerquelle	Fehlerquittierung
Antrieb	• Ohne Achs-Interface: Funktionsbaustein MC_Reset • Mit Achs-Interface: MB_AXIS_ADMINISTRATION.ClearError
Steuerung	• IndraLogic XLC/IndraMotion MLC ohne IMC-Interface – Kommando C-0-1030 starten (über MB_Command) • IndraMotion MLD ohne IMC-Interface – MC_Reset Funktionsbaustein mit AxisRef = lokale Achse • IndraMotion MLC & IndraMotion MLD mit IMC-Interface – MB_IMC_ADMINISTRATION.ClearError
Funktionsbaustein	• Fallende Flanke an Execute/Enable FB-Eingang
Peripherie	• Siehe Feldbus-/Inline-Dokumentation
Anwendungsspezifisch	• Anwendungsspezifisch

IMC-Interface IndraMotionControl Interface = IndraLogic Bibliothek mit Schnittstelle zur Steuerung, wie. z. B. Fehler löschen

5.7 Visualisierung

5.7.1 Allgemeines

Eine Visualisierung sollte nicht nur die Daten bereitstellen, die zum Betrieb der Maschine nötig sind, sie sollte auch bei der Fehlerdiagnose unterstützen.

5.7.2 Typische Inhalte einer Visualisierung

Anzeige des Maschinen-Status

- Aktive Betriebsart
- Aktuelle Diagnosen/Diagnose-Historie (siehe Fehlererkennung, Seite 46)
- Weiterführende Informationen zur Unterstützung bei der Inbetriebnahme, z. B. Achszustände (Position, Geschwindigkeit, Status, …)
- Applikationsspezifische Maschinen-Daten

Automatisierung von Applikationen

Steuern der Maschinenzustände

- Umschaltung der Betriebsart (Automatik, Manual, …)
- Quittierung anstehender Fehler
- Umschaltung der Bedienbilder
- Bedienung applikationsspezifischer Maschinenfunktionalitäten

5.7.3 Bereitstellung der Daten durch die SPS

Allgemeines

Die zu visualisierenden Daten müssen in die Symboldatei eingetragen werden.

Symbolkonfiguration

Die Variablen können mit den folgenden Zugriffsberechtigungen konfiguriert werden:

- Kein Zugriff
- Lesend
- Schreibend
- Lesend und Schreibend

Die Symbolkonfiguration kann dialoggeführt über das Fenster "Symbolkonfiguration", über "Pragma Direktiven" im Variablen-Deklarationsbereich der POU oder über Variablen-Listen erfolgen.

Voraussetzung für die Symbolkonfiguration ist das Vorhandensein des SPS-Objekts "Symbolkonfiguration" unterhalb der SPS-Applikation im IndraWorks Projektbaum. Hierzu wird das Objekt aus dem Bibliotheksfenster "SPS Objekte/Allgemeine Objekte" unterhalb des Geräteknotens "Logic/Application" per Drag and Drop hinzugefügt.

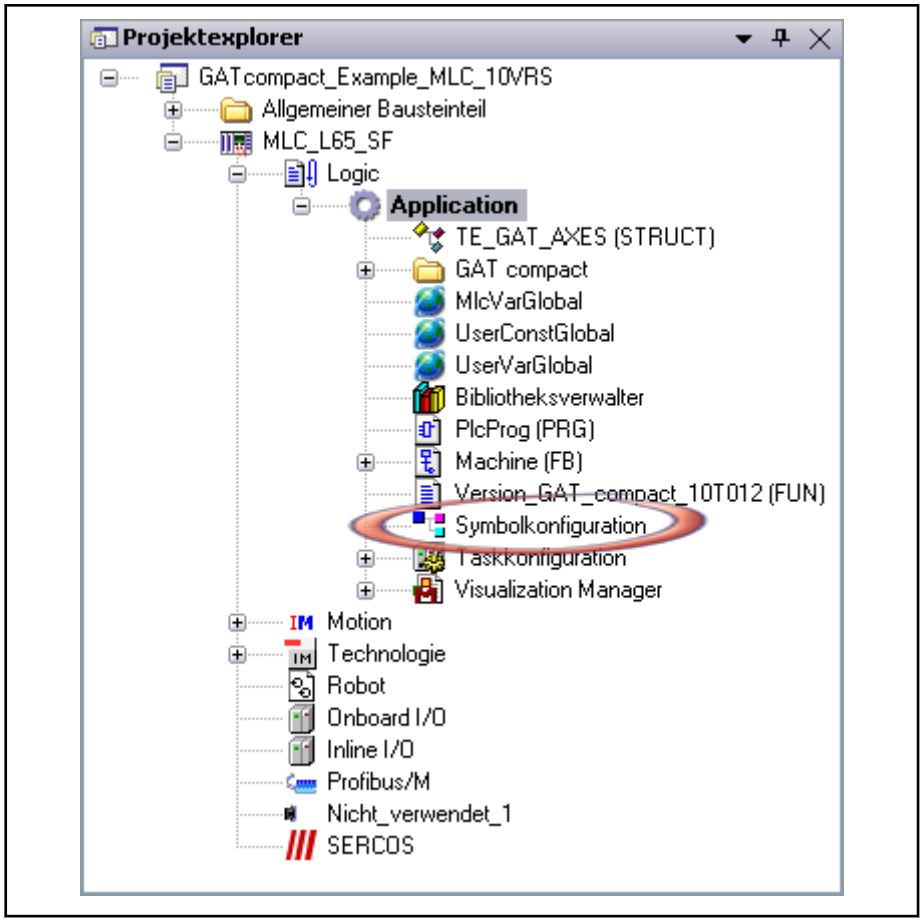


Abb.5-5: Symbolkonfiguration im Projektbaum

Dialoggeführte Symbolkonfiguration

Der Dialog für die Symbolkonfiguration wird über den neu erstellten Knoten "Symbolkonfiguration" im IndraWorks-Projektexplorer geöffnet.

Das Fenster der Symbolkonfiguration ist in zwei Bereiche gegliedert:

Verfügbare Variablen	Auflistung aller Variablen, die im SPS-Programm vorhanden sind, mit der jeweiligen maximalen Zugriffsberechtigungsstufe. Werden keine Variablen aufgelistet, wird über den Link "Aktualisieren" eine Aktualisierung ausgelöst. Hierbei ist zu beachten, dass das Projekt ohne Fehler übersetzbar sein muss
Ausgewählte Variablen	Auflistung aller Variablen, die für den symbolischen Zugriff konfiguriert wurden, mit der jeweiligen konfigurierten Zugriffsberechtigung. Mit Betätigen des Links "Ungültige Variablen anzeigen" werden alle Variablen, die nicht mehr im SPS-Projekt vorhanden sind, hervorgehoben

Die Auswahl bzw. das Löschen von Variablen erfolgt per Doppelklick auf das Element oder über die Schaltflächen **>** und **<**. Hierbei werden das selektierte Element und alle untergeordneten Elemente ausgewählt bzw. entfernt.

Automatisierung von Applikationen

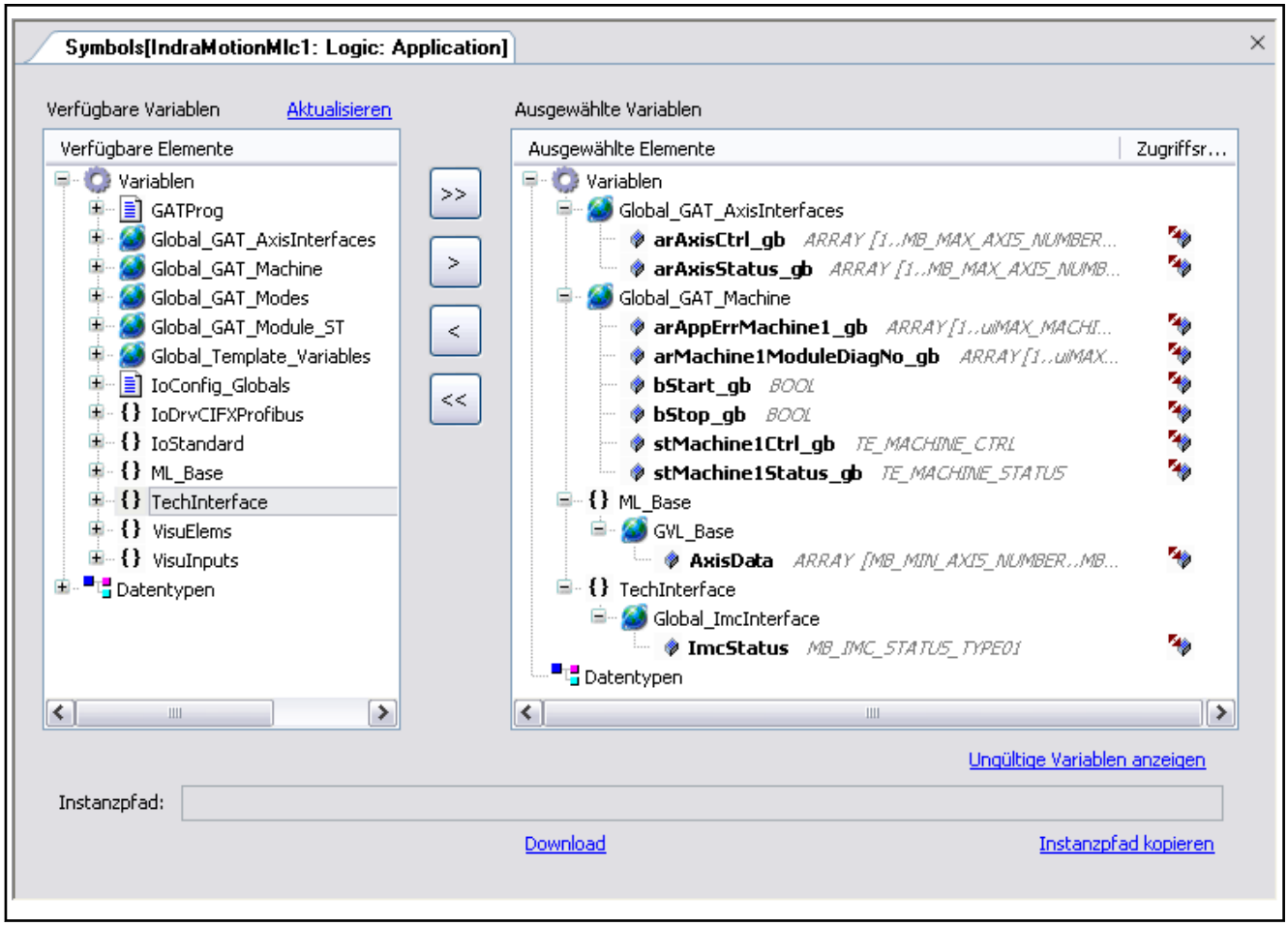


Abb.5-6: Dialoggeführte Symbolkonfiguration

Die Zugriffsrechte für die jeweilige Variable kann über einen Mausklick auf das Symbol in der Spalte "Zugriffsrechte" im Bereich "Ausgewählte Variablen" geändert werden.

Das Zugriffsrecht wird in der folgenden Reihenfolge weitergeschaltet:

- Lese und Schreibzugriff
- Schreibzugriff
- Lesezugriff

Symboleinträge für Variablen aus Globalen Variablenlisten, die aus einer Bibliothek stammen, müssen nach einer Versionsänderung der entsprechenden Bibliothek erneut in die Symbolkonfiguration eingefügt werden.

Symbolkonfiguration über Pragma

Die Symbolkonfiguration über "Pragma Direktiven" erfolgt direkt im Deklarationsbereich der Variablen, z. B. in Variablen-Listen oder in POUs.

Für die Pragma Konfiguration steht das Attribut **"{attribute 'symbol'}"** mit den Parametern für die Zugriffsberechtigung zur Verfügung. Die möglichen Konfigurationen sind in der folgenden Tabelle aufgelistet.

Pragma Direktive	Beschreibung
{attribute 'symbol' := 'none'}	Variablen sind für den Zugriff gesperrt und werden nicht in der Browsing-Liste des BR-WS Treibers angezeigt
{attribute 'symbol' := 'read'}	Variablen sind nur für den lesenden Zugriff freigegeben und werden in der Browsing-Liste des BR-WS Treibers angezeigt
{attribute 'symbol' := 'write'}	Variablen sind nur für den schreibenden Zugriff freigegeben und werden in der Browsing-Liste des BR-WS Treibers angezeigt
{attribute 'symbol'} {attribute 'symbol' := 'readwrite'}	Variablen sind für den lesenden und schreibenden Zugriff freigegeben und werden in der Browsing-Liste des BR-WS Treibers angezeigt

BR-WS Treiber WinStudio-Kommunikationstreiber von Bosch Rexroth

In Abhängigkeit der Position bzw. der Zeile in der das Pragma geschrieben wird, hat es für alle Variablen im Deklarationsbereich oder nur für eine Variable Gültigkeit:

- Damit das Pragma nur auf eine einzelne Variable wirkt, muss es unmittelbar in die Zeile vor der Variablen geschrieben werden
- Soll es auf alle Variablen im Deklarationsbereich wirken, muss es in der ersten Zeile der Deklarationsliste bzw. in der Zeile unmittelbar vor dem Deklarationsbereich (z. B. unmittelbar vor **VAR GLOBAL**…**END_VAR**) stehen. In diesem Fall können noch einzelne Variablen mit einer anderen Zugriffsberechtigung versehen werden

Mit dem Download der SPS-Applikation wird die Symbolinformation als XML-Datei im IndraWorks Projektunterverzeichnis "IndraLogic" erstellt und als Symbolapplikation in die Steuerung geladen

Kommunikations-Konfiguration

- Die Kommunikation zwischen der WinStudio-Applikation und der Steuerung erfolgt über den BR-WS Treiber. Hiermit können die Variablen aus der Symboldatei visualisiert werden
- Zur Optimierung der Kommunikations-Performance sollten die Variablen in verschiedene Treiber-Blätter aufgeteilt werden, welche in unterschiedlichen Intervallen aktualisiert werden

6 ML_TechInterface.library

6.1 Einführung und Übersicht

Das TechInterface besteht aus den Komponenten:

- Achs-Interface und
- IMC-Interface (Imc = IndraMotion Control)

Das **Achs-Interface** bündelt und erweitert PLCopen-Bewegungs-Funktionsbausteine und stellt ein einfach zu bedienendes Interface für die Antriebsfunktionalität zur Verfügung. Weniger Code und leistungsfähigere Kommandos beschleunigen die Programmentwicklung von Applikationen. Das Achs-Interface enthält Steuersignale und Parameter für die verschiedenen Betriebsarten der Leitachse und Folgeachse sowie Einstellmöglichkeiten für angewählte Prozesswerte.

Das **IMC-Interface** stellt ein einfach zu bedienendes Interface für die Steuerungsfunktionalität zur Verfügung. Das Imc-Interface enthält Steuersignale und Parameter für die Diagnose und Administration der Steuerung.

Das TechInterface wird vom IndraMotion MLC-Steuerungssystem als auch vom IndraMotion MLD-Antriebssystem unterstützt.

Die folgende Tabelle enthält die unterstützten Zielsysteme und Bibliotheken die die Funktionsbausteine und Datenstrukturen für das TechInterface enthalten:

System-Unterstützung	Drive Firmware-Unterstützung	Bibliothek
IndraMotion MLC ab 10VRS	ab MPx04VRS	ML_TechInterface.compiled-library
IndraMotion MLD-Antriebssystem ab 07VRS Siehe dazu Dokumentation R911322053 "Rexroth IndraMotion Achs-Interface für IndraLogic 1G"	ab MPx07VRS	MX_AxisInterface.lib MX_ImcInterface.lib

Abb.6-1: TechInterface Bibliotheken

Die Funktionsbausteine des TechInterface

Funktionsbaustein	Beschreibung
MB_AxisInitType01 (siehe Kap. 6.3.1 "MB_AxisInitType01" auf Seite 66)	wird zur Initialisierung des Achs-Interfaces für eine einzelne Achse benutzt. Der Funktionsbaustein muss nur einmal beim Programmstart oder bei jeder Phasenumschaltung vom Parametrier- in den Betriebs-Modus aufgerufen werden
MB_AxisInterfaceType01 (siehe Kap. 6.3.2 "MB_AxisInterfaceType01" auf Seite 68)	wird zur Konfiguration des Achs-Interfaces für eine einzelne Achse benutzt. Dieser Funktionsbaustein muss zyklisch (im Motion-Takt oder langsamer als der Motion-Takt) aufgerufen werden solange man sich im Betriebs-Modus befindet
MB_AxisInterfaceType02 (siehe Kap. 6.3.3 "MB_AxisInterfaceType02" auf Seite 71)	wird zur Konfiguration des Achs-Interfaces für Achsen benutzt, die koordinierte Bewegungen an einer Kinematik ausführen. (Robot Control)

ML_TechInterface.library

Funktionsbaustein	Beschreibung
MB_GetAxisInterfaceIndex (Funktion)	errechnet den passenden Index für die Nutzung im Achs-Interface Array aus der übergebenen Achsreferenz
MB_ImcInterface (Programm)	Funktionalität zur Ansteuerung und Bildung von Statusmeldungen der Steuerung

Abb.6-2: TechInterface

Das Achs-Interface wird als IndraMotion Programmier-Template oder als stand-alone Interface für die Achs-Funktionalität zur Verfügung gestellt.

Wenn es mit dem Programmier-Template GAT (Generic Application Template) bzw. GAT compact benutzt wird, muss sich der Anwender nicht mit Instanz-Aufrufen der Funktionsbausteine innerhalb des Projektes befassen. Diese Funktionalität ist komplett in das Template integriert und der Anwender muss nur ein paar Zeilen Code schreiben.

Wird hingegen das Achs-Interface als eigenständige Funktionalität benutzt, erfordert dies das Anlegen von Instanzen von beiden Funktionsbausteinen für jede Achse durch den Anwender. Dieser Abschnitt enthält Beschreibungen und Beispiele zur Nutzung des Achs-Interfaces als eigenständige Funktionalität.

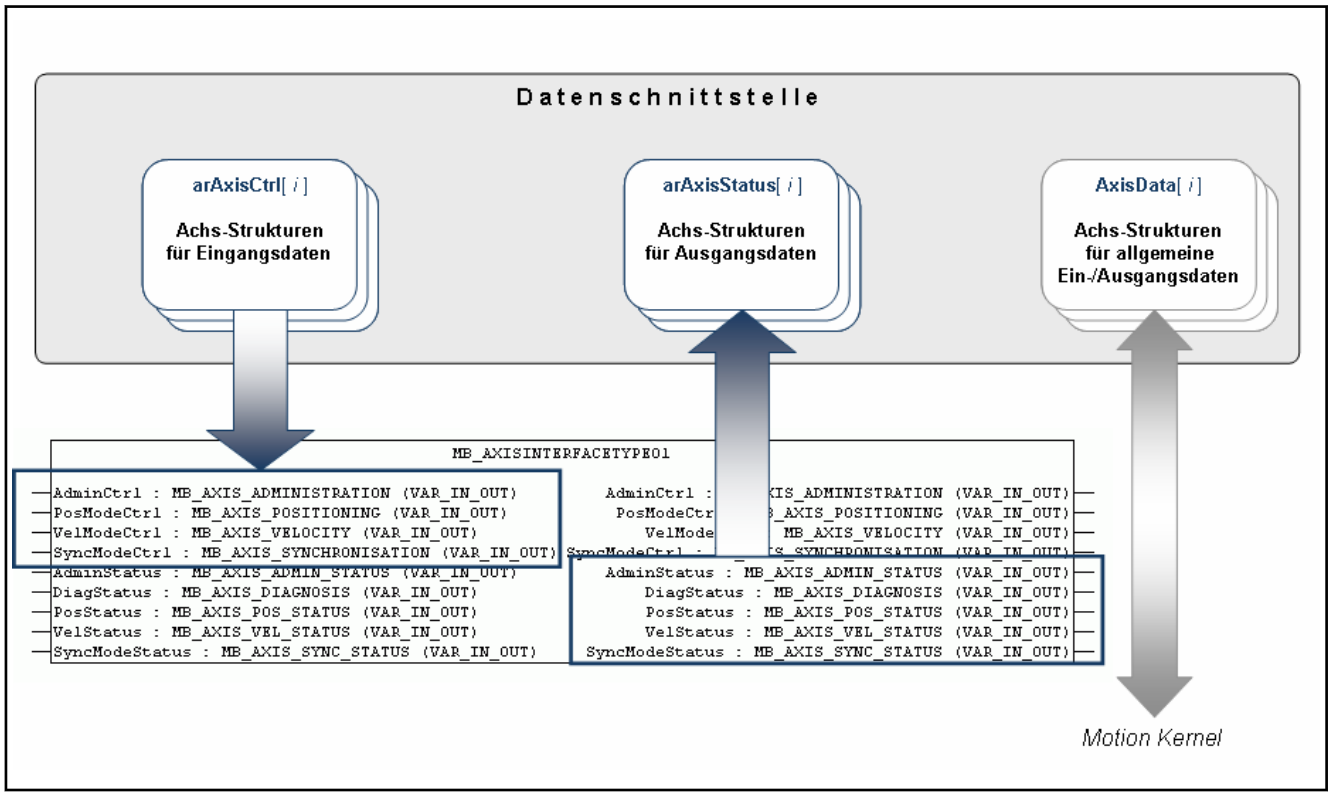


Abb.6-3: Achs-Interface Datenstruktur des Interface

Die folgende Tabelle zeigt das Anwender-Interface zur Achs-Interface Datenstruktur:

Anwender-Interface	Typ	Beschreibung
arAxisCtrl[ ]_gb	MB_AXIS_CONTROL_TYPE01 (MB_AXIS_CONTROL_TYPE02)	Steuerungsstruktur inklusive Sollwerte und Variablen zum Aktivieren der Betriebsarten
arAxisStatus[ ]_gb	MB_AXIS_STATUS_TYPE01 (MB_AXIS_STATUS_TYPE02)	Statusstruktur inklusive Diagnose-Info und Quittungen für die Betriebsarten
AxisData[ ]		Status- und Steuerungsstruktur inklusive Istwerte, Status-Bits und Zugriff auf die zyklisch konfigurierbaren Anwenderdaten (wird durch die MotionControl in der Firmware bearbeitet)
ImcCtrl[ ]	MB_IMC_CONTROL_TYPE01	Steuerungsstruktur inklusive Vorgabe des Kommunikations-Modus und Diagnosehandling
ImcStatus[ ]	MB_IMC_STATUS_TYPE01	Statusstruktur inklusive Status des Kommunikations-Modus und Diagnoseinformationen

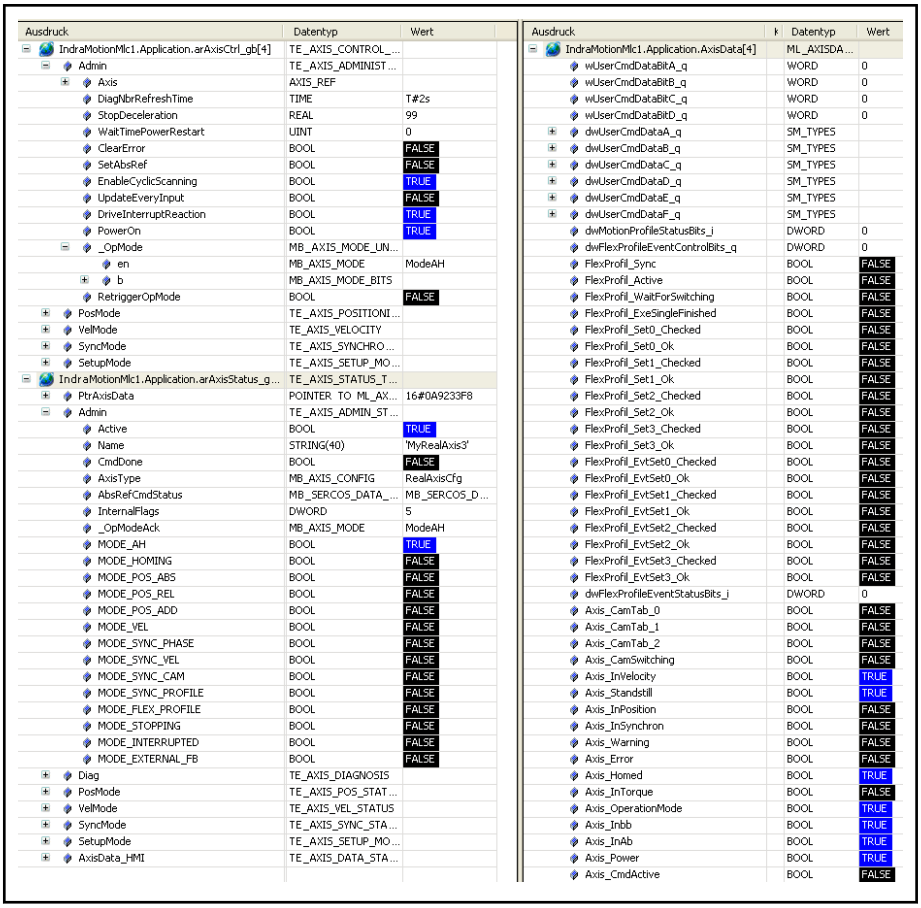
Abb.6-4: TechInterface – Anwender-Interface Datenstrukturen

Die Bezeichnung TYPE01 bzw. TYPE02 in den Namen der Funktionsbausteine und der Strukturen weist auf einen unterschiedlichen Funktionsumfang hin.



Zwischen den verschiedenen Achs-Interface Versionen "02VRS" (MLC 04VRS) und dem TechInterface "10VRS" (MLC10VRS) und neueren Versionen wurden Änderungen eingeführt, die zum Teil Anpassungen im Anwenderprogramm erfordern. Die notwendigen Anpassungen werden in Kap. 6.2 "TechInterface Erst-Konfiguration" auf Seite 62 beschrieben.

ML_TechInterface.library



Ausdruck	Datentyp	Wert
IndraMotionMlc1.Application.arAxisCtrl_gb[4]	TE_AXIS_CONTROL_...	
Admin	TE_AXIS_ADMINIST...	
Axis	AXIS_REF	
DiagNbrRefreshTime	TIME	T#2s
StopDeceleration	REAL	99
WaitTimePowerRestart	UINT	0
ClearError	BOOL	FALSE
SetAbsRef	BOOL	FALSE
EnableCyclicScanning	BOOL	TRUE
UpdateEveryInput	BOOL	FALSE
DriveInterruptReaction	BOOL	TRUE
PowerOn	BOOL	TRUE
_OpMode	MB_AXIS_MODE_UN...	
en	MB_AXIS_MODE	ModeAH
b	MB_AXIS_MODE_BITS	
RetriggerOpMode	BOOL	FALSE
PosMode	TE_AXIS_POSITIONI...	
VelMode	TE_AXIS_VELOCITY	
SyncMode	TE_AXIS_SYNCHRO...	
SetupMode	TE_AXIS_SETUP_MO...	
IndraMotionMlc1.Application.arAxisStatus_g...	TE_AXIS_STATUS_T...	
PtrAxisData	POINTER TO ML_AX...	16#0A9233F8
Admin	TE_AXIS_ADMIN_ST...	
Active	BOOL	TRUE
Name	STRING(40)	'MyRealAxis3'
CmdDone	BOOL	FALSE
AxisType	MB_AXIS_CONFIG	RealAxisCfg
AbsRefCmdStatus	MB_SERCOS_DATA_...	MB_SERCOS_D...
InternalFlags	DWORD	5
_OpModeAck	MB_AXIS_MODE	ModeAH
MODE_AH	BOOL	TRUE
MODE_HOMING	BOOL	FALSE
MODE_POS_ABS	BOOL	FALSE
MODE_POS_REL	BOOL	FALSE
MODE_POS_ADD	BOOL	FALSE
MODE_VEL	BOOL	FALSE
MODE_SYNC_PHASE	BOOL	FALSE
MODE_SYNC_VEL	BOOL	FALSE
MODE_SYNC_CAM	BOOL	FALSE
MODE_SYNC_PROFILE	BOOL	FALSE
MODE_FLEX_PROFILE	BOOL	FALSE
MODE_STOPPING	BOOL	FALSE
MODE_INTERRUPTED	BOOL	FALSE
MODE_EXTERNAL_FB	BOOL	FALSE
Diag	TE_AXIS_DIAGNOSIS	
PosMode	TE_AXIS_POS_STAT...	
VelMode	TE_AXIS_VEL_STATUS	
SyncMode	TE_AXIS_SYNC_STA...	
SetupMode	TE_AXIS_SETUP_MO...	
AxisData_HMI	TE_AXIS_DATA_STA...	

Ausdruck	k	Datentyp	Wert
IndraMotionMlc1.Application.AxisData[4]		ML_AXISDA...	
wUserCmdDataBitA_q		WORD	0
wUserCmdDataBitB_q		WORD	0
wUserCmdDataBitC_q		WORD	0
wUserCmdDataBitD_q		WORD	0
dwUserCmdDataA_q		SM_TYPES	
dwUserCmdDataB_q		SM_TYPES	
dwUserCmdDataC_q		SM_TYPES	
dwUserCmdDataD_q		SM_TYPES	
dwUserCmdDataE_q		SM_TYPES	
dwUserCmdDataF_q		SM_TYPES	
dwMotionProfileStatusBits_i		DWORD	0
dwFlexProfileEventControlBits_q		DWORD	0
FlexProfil_Sync		BOOL	FALSE
FlexProfil_Active		BOOL	FALSE
FlexProfil_WaitForSwitching		BOOL	FALSE
FlexProfil_ExeSingleFinished		BOOL	FALSE
FlexProfil_Set0_Checked		BOOL	FALSE
FlexProfil_Set0_Ok		BOOL	FALSE
FlexProfil_Set1_Checked		BOOL	FALSE
FlexProfil_Set1_Ok		BOOL	FALSE
FlexProfil_Set2_Checked		BOOL	FALSE
FlexProfil_Set2_Ok		BOOL	FALSE
FlexProfil_Set3_Checked		BOOL	FALSE
FlexProfil_Set3_Ok		BOOL	FALSE
FlexProfil_EvtSet0_Checked		BOOL	FALSE
FlexProfil_EvtSet0_Ok		BOOL	FALSE
FlexProfil_EvtSet1_Checked		BOOL	FALSE
FlexProfil_EvtSet1_Ok		BOOL	FALSE
FlexProfil_EvtSet2_Checked		BOOL	FALSE
FlexProfil_EvtSet2_Ok		BOOL	FALSE
FlexProfil_EvtSet3_Checked		BOOL	FALSE
FlexProfil_EvtSet3_Ok		BOOL	FALSE
dwFlexProfileEventStatusBits_i		DWORD	0
Axis_CamTab_0		BOOL	FALSE
Axis_CamTab_1		BOOL	FALSE
Axis_CamTab_2		BOOL	FALSE
Axis_CamSwitching		BOOL	FALSE
Axis_InVelocity		BOOL	TRUE
Axis_Standstill		BOOL	TRUE
Axis_InPosition		BOOL	FALSE
Axis_InSynchron		BOOL	FALSE
Axis_Warning		BOOL	FALSE
Axis_Error		BOOL	FALSE
Axis_Homed		BOOL	TRUE
Axis_InTorque		BOOL	FALSE
Axis_OperationMode		BOOL	TRUE
Axis_Inbb		BOOL	TRUE
Axis_InAb		BOOL	TRUE
Axis_Power		BOOL	TRUE
Axis_CmdActive		BOOL	FALSE

Abb.6-5: Überblick über die Datenstrukturen des Axis-Interface

Benutzen Sie die AxisNo der AXIS_REF-Struktur als Index für das Feld, z. B. arAxisCtrl_gb[MyVirtualAxis.AxisNo].Admin. usw.

EnableCyclicScanning

Die interne Handhabung einiger Sollwerte kann durch das arAxisCtrl_gb[ ].Admin.EnableCyclicScanning Element gesteuert werden. Wird "EnableCyclicScanning" auf TRUE gesetzt, werden einige Sollwerte der arAxisCtrl_gb[ ]-Struktur zyklisch gescannt und sofort wirksam, wenn sich ein Wert ändert.

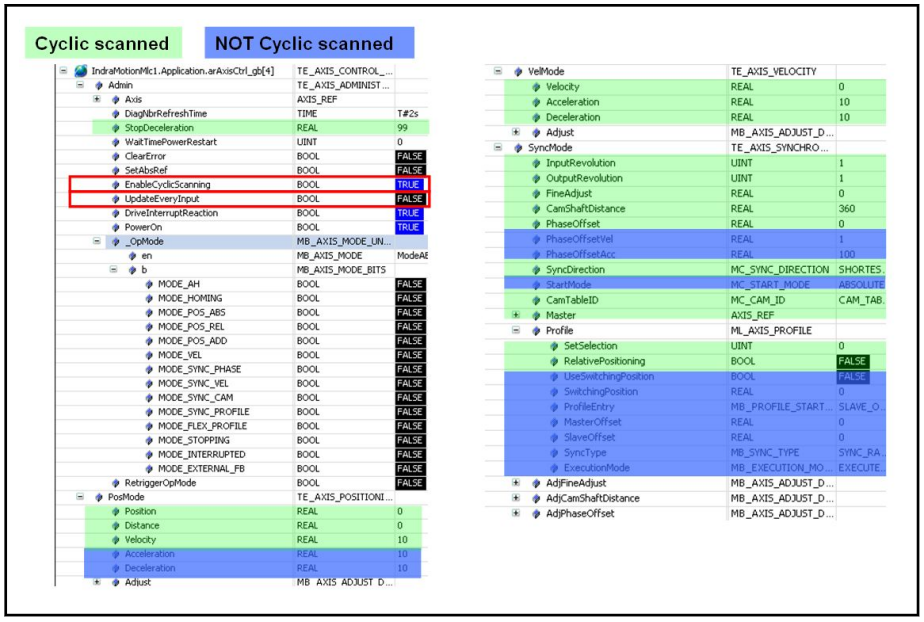


Abb.6-6: Zyklisch gescannte Elemente von arAxisCtrl_gb[ ]

UpdateEveryInput

Die oben gezeigten nicht zyklisch gescannten Daten werden bei einer Änderung eines zyklisch gescannten Elementes ebenfalls übernommen, wenn der Eingang "UpdateEveryInput" gesetzt wurde.

Beispiel: die Betriebsart "relatives Positionieren" wurde mit den oben gezeigten Werten Velocity=10 und Acceleration=10 gestartet. Um den nächsten Positioniervorgang mit einer geänderten Beschleunigung zu starten, wird der Eingang "PosMode. Acceleration" von 10 auf 20 geändert und danach die nächste Distance geschrieben. Mit dem Ändern der Distance wird auch die Beschleunigung übernommen.

- Bei der Aktivierung einer Betriebsart (.Admin._OpMode) werden, unabhängig von der Einstellung des "EnableCyclicScanning"-Eingangs, alle Eingangsdaten gelesen
- Wenn "EnableCyclicScanning" = TRUE, werden alle Eingangsdaten, die grün hervorgehoben sind, zyklisch gelesen. Das bedeutet, dass nach Aktivierung einer Betriebsart jede Änderung der Werte sofort gelesen wird
- Im Gegensatz dazu werden alle Eingangsdaten, die blau hervorgehoben sind, nicht zyklisch gescannt. Das bedeutet, dass die Werte nur gelesen werden, wenn eine Betriebsart aktiviert wird
- Wenn "UpdateEveryInput" = TRUE, werden alle Eingangsdaten, die blau hervorgehoben sind auch übernommen, wenn eines der zu dieser Betriebsart gehörenden zyklisch gescannten Elemente geändert wird
- Die Datenkonsistenz wird durch "EnableCyclicScanning" (FALSE→Daten schreiben→TRUE) erreicht

Für Inbetriebnahmezwecke stehen verschiedene IndraLogic-Visualisierungen, basierend auf den fünf Strukturelementen, die in diesem Abschnitt beschrieben werden, im Beispielprojekt zur Verfügung (Einzelheiten siehe Kap. 6.6.3 "Achs-Interface Visualisierungen" auf Seite 90).

Was ist neu in der Version 10VRS (MLC10VRS)?

- Zusammenführung Axis-Interface und Imc-Interface zum TechInterface:

ML_TechInterface.library

Im Rahmen der Umstellung auf IndraLogic-2G wurden die Bibliotheken ML_AxisInterface.lib und ML_ImcInterface.lib zu einer Bibliothek ML_TechInterface.compiled-library zusammengefasst

- Betriebsartenanwahl mit UNION:

  Mit IndraLogic 2G wird die Betriebsartenanwahl arAxisCtrl[MyRealAxis1.Axisno].Admin._OpMode mittels eines Vereinigungsdatentyps "UNION" über einen Aufzählungsdatentyp "ENUMERATION" und zusätzlich binär über den neuen Datentyp "BIT" implementiert. In IndraLogic-1G wurde eine ähnliche Funktionalität mittels "bitaccess" angeboten. Dabei kommt es aber zu Fehlzuweisungen
- Neuer Eingang "arAxisCtrl_gb[].Admin.UpdateEveryInput": Mit diesem Eingang können auch nicht zyklisch gescannte Eingänge bei aktivierter Betriebsart geändert werden
- Neue Betriebsart "ModeStopping": die Betriebsart "ModeStopping" hält die PLCopen statemachine im Zustand "Stopping", d. h. ein außerhalb des Achs-Interface aufgerufener PLCopen-Funktionsbaustein würde mit dem Fehler "Kommando im Zustand Stopping zurückgewiesen" beendet werden. Die Betriebsart "ModeAH" überführt jetzt die Achse in den PLCopen-Zustand "StandStill", d. h. ein außerhalb des Achs-Interface aufgerufener PLCopen-Funktionsbaustein wird akzeptiert und das Achs-Interface meldet "ModeExternalFB"
- Neuer Ausgang "arAxisStatus_gb[].Admin.AbsRefCmdStatus": mit diesem Ausgang kann der Status des über arAxisCtrl_gb[].Admin.SetAbsRef gestarteten Kommandos überprüft werden
- Neuer Ausgang "ImcStatus.PassiveMode": dieser Ausgang wird gesetzt, wenn unter Umgehung des Imc-Interface eine Phasenumschaltung z. B. über IndraWorks gestartet wird. Sobald mit einer weiteren Phasenumschaltung wieder in die an ImcCtrl.Admin._OpMode eingestellte Phase geschaltet wird, wird der Ausgang wieder gelöscht. Des Weiteren kann mit Hilfe des Eingangs "ImcCtrl.Admin.RetriggerOpMode" bei gesetztem Ausgang "PassiveMode" wieder in die eingestellte Zielphase geschaltet werden
- Unterstützung neuer Achstyp "Reglerachse": der in MLC10VRS eingeführte Achstyp "Reglerachse" wird als arAxisStatus_gb[].Admin.AxisType = ControlAxisCfg (16#0080) erkannt und soweit möglich unterstützt
- Komfortablere Anwendererweiterungen mit Hilfe der objektorientierten Spracherweiterungen von IndraLogic 2G:

  Es können jetzt auch die Unterstrukturen von arAxisCtrl_gb[] und arAxisStatus_gb[] durch den Anwender erweitert werden (siehe auch Kap. 6.7 "Achs-Interface Anwender-Erweiterung" auf Seite 103)

Was ist neu in der Version 11VRS (XLC11VRS/MLC11VRS)?

- Überwachung des Zustandes "Axis_Interrupted": das Achs-Interface überwacht das Bit "AxisData[].Axis_Interrupted" und wechselt bei TRUE im AxisStatus in "ModeInterrupted". Das Achs-Interface meldet damit die Übernahme der Kontrolle durch den Antrieb, z. B. durch die Sicherheitstechnik im Antrieb

  Durch Rücksetzen des neuen Eingangs "arAxisCtrl_gb[].Admin.DriveInterruptReaction" kann diese Überwachung deaktiviert werden. In der Defaulteinstellung ist die Überwachung aktiv

  Die Überwachung von Axis_Interrupted steht auch in der Vorgängerversion (ab MLC10V08) zur Verfügung

- Zyklisches Scannen von arAxisCtrl_gb[].SyncMode.SyncDirection (ab XLC/MLC11V04): Bei Verstellung des "SyncMode.PhaseOffset" wirkt bei Interpolation im Antrieb die eingestellte "SyncMode.SyncDirection". Bei Maschinen mit einer Materialbahn wird beim Aktivieren der Synchronbetriebsart in positive Richtung aufsynchronisiert. Bei späteren Verstellungen des "SyncMode.PhaseOffset" soll dieser auf dem kürzesten Weg verstellt werden. Eine Änderung von "SyncMode.SyncDirection" bei aktiver Betriebsart ist damit möglich
- Das Achs-Interface liest nun in Abhängigkeit des Achstyps und der Meldungsart jeweils die passenden Antriebs- oder Steuerungsparameter.

  IndraDrive mit Interpolation im Antrieb:

  – Gutmeldungen: Antriebsparameter werden gelesen, z. B.: "A0012 Betriebsbereit"
  – Fehler im Antrieb: Antriebsparameter werden gelesen, z. B.: "F2026 Leistung weg"
  – Fehler auf der Steuerung: Steuerungsparameter werden gelesen, z. B.: "F2229233 ungültige Leitachse"

  IndraDrive mit Interpolation in der Steuerung:

  – Gutmeldungen A-0-0020/A-0-0023 werden gelesen, z. B.: "Synchronized Motion"
  – Fehler im Antrieb: Antriebsparameter werden gelesen, z. B.: "F2026 Leistung weg"
  – Fehler auf der Steuerung: Steuerungsparameter werden gelesen, z. B.: "F2229233 ungültige Leitachse"

  SercosDrive und virtuelle Achsen:

  – Es werden immer die Steuerungsparameter gelesen

Was ist neu in der Version 12VRS (XLC12VRS/MLC12VRS)?

- Neuer Eingang "arAxisCtrl_gb[].Admin.PowerOn": Ist dieses Bit gesetzt (Defaultwert ist TRUE), so wird im AxisInterface der MC_Power intern aufgerufen. Ist der Eingang nicht gesetzt, muss der MC_Power durch das Anwendungsprogramm bedient werden
- Achs-Interface Beispielprojekt: zusätzlich zum "GAT compact" bietet der Template-Wizard jetzt ein spezielles Achs-Interface Beispielprojekt an
- Bildung von Diagnosenummer und Diagnosemeldung: ab Version 12VRS werden A-0-0020/A-0-0023 für IndraDrive Antriebe immer aus den Antriebsparametern S-0-0095/S-0-0390 gebildet. Das Achs-Interface liest jetzt als Default immer die Steuerungsparameter A-0-0020/A-0-0023. Über arAxisCtrl_gb[].Admin.DiagNbrRefreshTime kann das Verhalten modifiziert werden
- Initialisierung Imc-Interface Zielmodus: Der Zielmodus "ImcCtrl.Admin._OpMode" wird jetzt statt mit Null in Abhängigkeit der Parametrierung initialisiert:

  WENN aktueller Modus (C-0-0452) >= Hochlauf-Zielmodus (C-0-0450) DANN

  ImcCtrl.Admin._OpMode := Aktueller Modus (C-0-0452)

  ANSONSTEN

  ImcCtrl.Admin._OpMode := Hochlauf-Zielmodus (C-0-0450)

  Damit erreicht man mindestens den Hochlauf-Zielmodus (C-0-0450) bzw. bleibt im aktuellen Modus.

Diese Initialisierung erfolgt in der Bibliothek nur einmalig beim SPS-Programmstart und kann danach jederzeit überschrieben werden.

- Neu ab 12V08 und 13V04: Das Element "arAxisStatus_gb[].Admin.AxisFeatures" zeigt binär codiert die Eigenschaften der Achse an.

  Das Element "arAxisStatus_gb[].Admin.AxisType" wird dadurch ersetzt und sollte nicht mehr verwendet werden.

Was ist neu in der Version 13VRS (XLC13VRS/MLC13VRS)

- Ansteuerung von Achsen für koordinierte Bewegungen (Robot Control). Durch Verwendung des neuen Funktionsbausteins MB_AxisInterfaceType02 (statt MB_AxisInterfaceType01) können Achsen in der Betriebsart "Robot Control" betrieben werden.
- Unterstützung der Parameterebene mittels des Eingangs "arAxisCtrl_gb[].Admin.ParamLevel" und des Ausgangs "arAxisStatus_gb[].Admin.ParamLevelAck".
- Neu ab 12V08 und 13V04: Das Element "arAxisStatus_gb[].Admin.AxisFeatures" zeigt binär codiert die Eigenschaften der Achse an.

  Das Element "arAxisStatus_gb[].Admin.AxisType" wird dadurch ersetzt und sollte nicht mehr verwendet werden.



Zwischen den verschiedenen Achs-Interface Versionen "02VRS" (MLC 04VRS) und dem TechInterface "10VRS" (MLC10VRS) und höher wurden Änderungen eingeführt, die zum Teil Anpassungen im Anwenderprogramm erfordern. Die notwendigen Anpassungen werden in Kap. 6.2 "TechInterface Erst-Konfiguration" auf Seite 62 beschrieben.

6.2 TechInterface Erst-Konfiguration

Bevor die Funktionalität des TechInterfaces benutzt werden kann, muss dieses zuerst initialisiert werden. Die folgenden Schritte aktivieren das Achs-Interface in IndraLogic.



Viele der in diesem Abschnitt beschriebenen Schritte sind schon in den Programmiervorlagen "GAT bzw. GAT compact" enthalten. Diese Templates können in IndraLogic importiert und als Leitfaden für neue Projekte benutzt werden. Bei Verwendung der Vorlagen GAT bzw. GAT compact ist das Handling des Achs-Interface inklusive möglicher Erweiterungen durch den Anwender komplett lauffähig ausprogrammiert.

Erste Schritte bei Verwendung des Beispielprojektes GAT compact.

1. Ein neues Projekt erstellen und die passende Steuerung mit z.B. dem Namen "MyControl" einfügen. Im Folgenden die IP-Adresse der Steuerung eingeben. Falls reale Achsen an der Steuerung angeschlossen sind, diese unterhalb der Steuerung einfügen.
2. Klick mit rechter Maustaste auf die Steuerung. Im sich öffnenden Kontextmenü den Punkt "GAT" wählen.
3. Im Wizard die Variante "GAT compact" anwählen und mit **Fertigstellen** den Import starten. Der GAT-Wizard kann bei Beibehaltung der vorgegebenen Projektstruktur jederzeit wieder aufgerufen werden um Achsen hinzuzufügen, zu löschen usw. oder um Visualisierungsbeispiele aufzurufen. Der GAT-Wizard hat bereits zwei virtuelle Achsen als Starthilfe eingefügt. Falls reale Achsen vorhanden sind, sind diese über den GAT-Wizard hinzuzufügen.

4. Zum Übertragen des Motionprojektes in die Steuerung auf "MyControl" klicken und dann mittels "MyControl->Online schalten" die Übertragung starten. Anschließend über das Symbol "BB" in den Betriebsmodus umschalten.
5. Das Laden des SPS-Projektes wird über **Debug ▸ Einloggen auf Application...** gestartet. Anschließend die SPS über das Pfeil-Symbol starten.
6. Mit den mitgelieferten Visualisierungsbeispielen (System_Overview_All_Axis) sollte jetzt ein Verfahren der virtuellen Achsen möglich sein.

Erste Schritte bei Nutzung eines eigenen Programmes.

1. Öffnen Sie den Bibliotheksverwalter im Projektbaum unter "Logic->Application" und fügen Sie die Bibliothek **ML_TechInterface.compiled-library** zum aktuellen IndraLogic-Projekt hinzu.
2. Fügen Sie unterhalb des Knotens "Application" zwei "Globale Variablenlisten" hinzu und nennen Sie diese z.B. "**UserVarGlobal**" und "**UserConstGlobal**" (im GAT compact). Erzeugen Sie dann den folgenden Code:

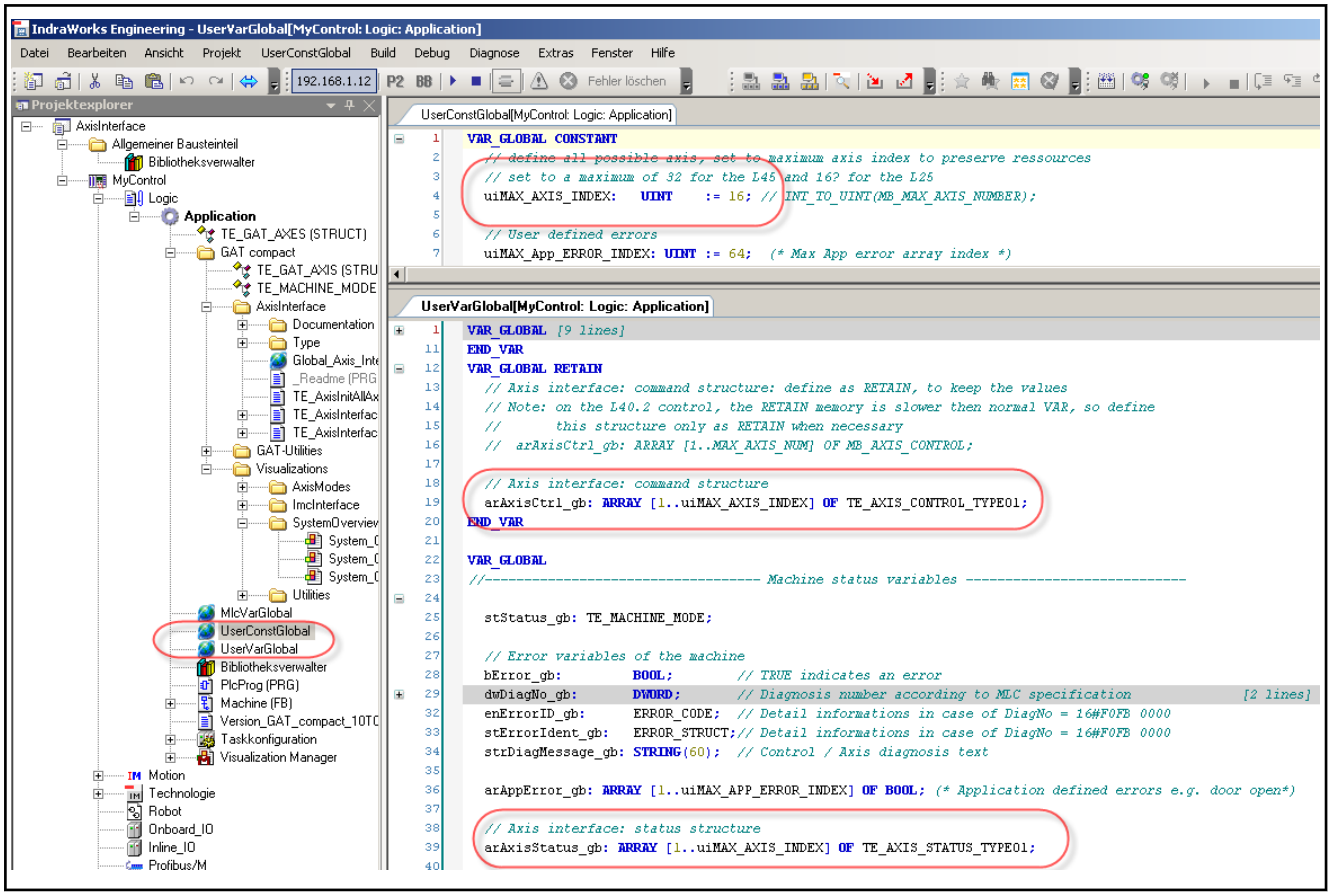


Abb.6-7: Globale Variablen des Achs-Interface

3. Legen Sie ein Programm (z.B. **TE_AxisInterfaceType01Prog** im GAT compact), das den Funktionsbaustein MB_AxisInterfaceType01 für MAX_AXIS_NUM Achsen aufruft, an. Dieser Funktionsbaustein interpretiert die Werte in "arAxisCtrl_gb" und aktualisiert die Werte in "arAxisStatus_gb". Möglich ist auch, den Funktionsbaustein in separaten Programmen aufzurufen, wenn die Struktur des Projekts dies erfordert.
4. Klicken Sie auf die "Taskkonfiguration" und legen Sie das TE_AxisInterfaceType01Prog als eine zyklisch aufzurufende PlcTask fest (z.B. Intervall: T#20ms).

5. Rufen Sie in der "Taskkonfiguration" das Programm "MB_ImcInterface" aus der Bibliothek TECHINTERFACE auf (z.B. Intervall: T#20ms oder auch in einer freilaufenden Task).
6. Legen Sie ein Initialisierungsprogramm an, das einmalig beim Start oder bei jeder Phasenumschaltung vom Parametrier- in den Betriebs-Modus aufgerufen wird bzw. integrieren Sie diesen Vorgang in das vorher angelegte Programm (z.B. **TE_AxisInterfaceType01Prog** im GAT compact).
   - Benutzen Sie den Funktionsbaustein MB_AxisInitType01 für die Grund-Initialisierung von arAxisCtrl_gb und arAxisStatus_gb für jede verwendete Achse

     Im GAT compact macht dies, in Abhängigkeit der Parametrierung, der Funktionsbaustein TE_AxisInitAllAxisType01
   - Die Struktur arAxisCtrl_gb[ ] kann als Retain-Daten definiert werden. In diesem Fall initialisieren Sie alle NICHT RETAIN Werte mit definierten Werten (bzgl. der definierten Werte siehe "STATE_FIRST_INIT" in TE_AxisInterfaceType01Prog).

     Ist die Struktur nicht als RETAIN definiert, müssen alle vom Defaultwert abweichenden Elemente initialisiert werden
7. Benutzen Sie nun arAxisCtrl_gb[ ], arAxisStatus_gb[ ] und AxisData[ ] zum Programmieren Ihrer Applikation.

Konvertierung von Achs-Interface Projekten

☞ Vor den folgenden Änderungen ist ein Archiv des IndraWorks-Projektes zu erstellen. Wurden Anpassungen an nicht zur Änderung vorgesehenen Programmteilen des Beispielprojektes vorgenommen, können diese verloren gehen.

Die folgenden Anpassungen zum Achs-Interface sind notwendig, um ein Projekt von Achs-Interface 02VRS (MLC 04VRS) umzusetzen.

1. Globale Variablen:

   Umbenennen der Strukturen
   - AXIS_CONTROL_TYPE01 in

     MB_AXIS_CONTROL_TYPE01 und
   - AXIS_STATUS_TYPE01 in

     MB_AXIS_STATUS_TYPE01

   Die Konstante MAX_AXIS_IF_CYCLES in Global_Axis_Interface wird nicht mehr benötigt.
2. Taskkonfiguration:

   Den Aufruf "ImcInterface()" in "MB_ImcInterface()" ändern.
3. Die folgenden Strukturelemente wurden verschoben bzw. umbenannt und müssen bei Verwendung im Anwenderprogramm angepasst werden:

- arAxisStatus[i].Active → arAxisStatus[i].Admin.Active
- arAxisCtrl[i].SyncMode.AdjCamShaftDist → arAxisCtrl[i].SyncMode.AdjCamShaftDistance
- arAxisCtrl[i].Comment → arAxisStatus[i].Admin.Name
- arAxisCtrl[i].SyncMode.InputRev → arAxisCtrl[i].SyncMode.InputRevolution
- arAxisCtrl[i].SyncMode.OutputRev → arAxisCtrl[i].SyncMode.OutputRevolution

- arAxisCtrl[i].SyncMode.CamShaftDist → arAxisCtrl[i].SyncMode.CamShaftDistance
- arAxisCtrl[i].SyncMode.AdjCamShaftDist → arAxisCtrl[i].SyncMode.AdjCamShaftDistance

4. Wurden die Strukturen arAxisCtrl bzw. arAxisStatus erweitert, müssen die Änderungen aus MB_AXIS_CONTROL_TYPE01 bzw. MB_AXIS_STATUS_TYPE01 in die Anwenderstrukturen übernommen werden. Alle Strukturen wurden von AXIS_xxx in MB_AXIS_xxx umbenannt.

Die folgenden Anpassungen zum Achs-Interface sind notwendig, um ein Projekt von Achs-Interface 02VRS (MLC 04VRS) auf das TechInterface 10VRS und höher (MLC 10VRS und höher) umzusetzen. Es wird davon ausgegangen, dass die Konvertierung des IndraWorks-Projektes bereits ausgeführt wurde.

Aufgrund des neuen BIT-Datentyps ändert sich der Zugriff auf "_OpMode" innerhalb der AxisCtrl-Struktur. In der AxisStatus-Struktur sind die Zugriffe auf "_OpMode" unverändert.

- Beispiel für Verwendung als Enumeration (Aufzählungsdatentyp):

  MLC 04VRS:

  ```
  arAxisCtrl[MyRealAxis1.Axisno].Admin._OpMode: = Mode-
  AH
  ```

  MLC 10VRS und höher:

  ```
  arAxisCtrl[MyRealAxis1.Axisno].Admin._OpMode.en:    =
  ModeAH
  ```

- Beispiel für Verwendung von BIT:

  02VRS:

  ```
  arAxisCtrl[MyRealAxis1.Axisno].Admin.MODE_SYNC_PHASE:
  = TRUE;
  ```

  MLC 10VRS und höher:

  ```
  arAxisCtrl[MyRealAxis1.Axisno].Admin._OpMode.b.MO-
  DE_SYNC_PHASE: = TRUE;
  ```

1. Vorgehensweise bei Verwendung als Enumeration (Aufzählungsdatentyp):
   - Suchen nach "Admin._OpMode"
   - Nur bei Fundstellen mit "arAxisCtrl[]" ersetzen durch "Admin._OpMode.en"
2. Vorgehensweise bei Verwendung von BIT:
   - Suchen nach "Admin.MODE_"
   - Nur bei Fundstellen mit "arAxisCtrl[]" ersetzen durch "Admin._OpMode.b.MODE_"

Ab dem MLC-Release 10V04 wird kein Achs-Interface Beispielprojekt mehr angeboten. Stattdessen kann das Beispielprojekt "GAT compact" als Einstiegsvorlage verwendet werden. Im Rahmen des Umbaus auf "GAT compact" wurden diverse Teile des Beispielprojektes umbenannt:

- arAxisStatus[i] → arAxisStatus_gb[i]
- arAxisCtrl[i] → arAxisCtrl_gb[i]
- Global_Axis_Interface → UserVarGlobal
- Global_Axis_Interface_Const → UserConstGlobal

ML_TechInterface.library

- MY_AxisInterface → TE_AxisInterface
- ML_AxisInitAllAxisType01 → TE_AxisInitAllAxisType01
- AxisInterfaceType01Prog und InitProg → zusammengefasst zu TE_AxisInterfaceType01Prog
- Strukturen MY_ .... → TE_...

6.3 Achs-Interface Funktionsbausteine

6.3.1 MB_AxisInitType01

Kurzbeschreibung

Der Funktionsbaustein MB_AxisInitType01 wird zur Initialisierung des Achs-Interfaces für eine einzelne Achse benutzt. Der Funktionsbaustein muss nur einmal beim Programmstart oder bei jeder Phasenumschaltung vom Parametrier- in den Betriebs-Modus aufgerufen werden.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
IndraMotion MLC ab 09VRS	ML_TechInterface.compiled-library
IndraMotion MLD ab MPx07 ohne Zusatzpaket MA	MX_AxisInterface.lib

Abb.6-8: Referenztabelle MB_AxisInitType01

Interfacebeschreibung

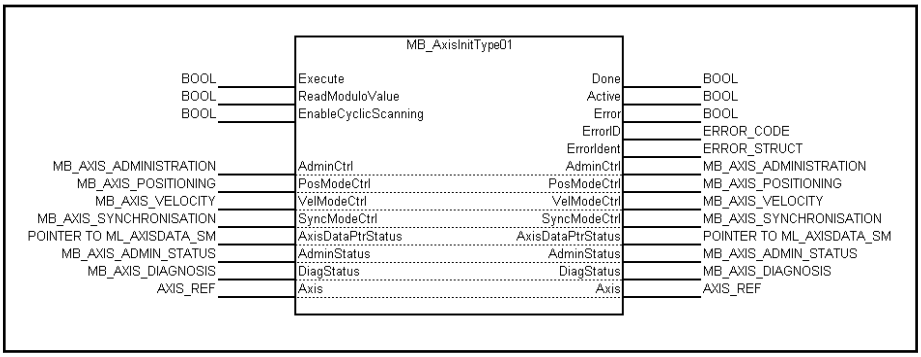


Abb.6-9: Funktionsbaustein MB_AxisInitType01

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	Execute	BOOL	Die Initialisierung wird durch eine positive Flanke an "Execute" gestartet
	ReadModuloValue	BOOL	Abhängig von der Parametrierung wird der Modulowert für rotarische Achsen in die Adjustment-Struktur von PosModeCtrl und SyncModeCtrl.AdjPhaseOffset geschrieben. Der Modulowert wird für alle anderen eingestellten Werte auf Null gesetzt. Der Standardwert ist TRUE
	EnableCyclicScanning	BOOL	Aktiviert die zyklische Überprüfung und Übertragung der geänderten Eingänge für die aktuelle Betriebsart - relevant für die Werte in arAxisCtrl_gb[ ] mit der Bemerkung: zyklisch überprüft UND übertragen bei Änderung. Der Standardwert ist TRUE

I/O-Typ	Name	Datentyp	Kommentar
VAR_OUTPUT	Done	BOOL	Wird gesetzt, wenn der FB die Bearbeitung beendet hat
	Active	BOOL	Wird gesetzt, wenn der FB aktiv ist (nicht im Leerlaufbetrieb)
	Error	BOOL	Zeigt an, dass ein Fehler in der FB-Instanz aufgetreten ist
	ErrorID	ERROR_CODE	Kurzer Hinweis zur Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Information zum Fehler
VAR_IN_OUT	Axis	AXIS_REF	Zu initialisierende Achse im Achs-Interface
	AdminCtrl	MB_AXIS_ADMINISTRATION	Verwaltung der Achse arAxisCtrl_gb[i].Admin anschließen
	PosModeCtrl	MB_AXIS_POSITIONING	Positionierungsbetriebsarten arAxisCtrl_gb[i].PosMode anschließen
	VelModeCtrl	MB_AXIS_VELOCITY	Betriebsart Geschwindigkeits-Steuerung arAxisCtrl_gb[i].VelMode anschließen
	SyncModeCtrl	MB_AXIS_SYNCHRONISATION	Synchronisations-Betriebsarten arAxisCtrl_gb[i].SyncMode anschließen
	AxisDataPtrStatus	POINTER TO ML_AXISDATA_SM	Zeiger auf AxisData dieser Achse arAxisStatus_gb[i].PtrAxisData anschließen
	AdminStatus	MB_AXIS_ADMIN_STATUS	Status Verwaltung der Achse arAxisStatus_gb[i].Admin anschließen
	DiagStatus	MB_AXIS_DIAGNOSIS	Diagnoseinformationen der Achse arAxisStatus_gb[i].Diag anschließen

Abb.6-10: I/O-Interface MB_AxisInitType01

☞ Es ist nicht möglich die komplette Instanz der MB_Axis_Control_Type01 und der MB_Axis_Status_Type01 Strukturen (z. B. arAxisCtrl_gb[ ] / arAxisStatus_gb[ ]) über einen Eingang dem Funktionsbaustein zu übergeben. Dies wurde vorgenommen, um der Anwender-Applikation spezielle Erweiterungen zum bestehenden Code zu ermöglichen. Nähere Details finden Sie unter Kap. 6.7 "Achs-Interface Anwender-Erweiterung" auf Seite 103.

Deshalb werden die benötigten Elemente von MB_Axis_Control_Type01 und MB_Axis_Status_Type01 als separate Eingänge übergeben.

Der Funktionsbaustein MB_AxisInitType01 initialisiert die folgenden Strukturelemente mit Standardwerten:

Strukturelement	Standard
AdminCtrl._OpMode	ModeAB
AdminCtrl.ClearError	FALSE
AdminCtrl.SetAbsRef	FALSE

ML_TechInterface.library

Strukturelement	Standard
AdminCtrl.EnableCyclicScanning	Eingang EnableCyclicScanning
AxisDataPtrStatus	ADR(AxisData[Axis.AxisNo])
AdminCtrl.Axis.CntrlNo	LOCAL_CNTRL
AdminCtrl.Axis.AxisNo	Eingang Axis.AxisNo
PosModeCtrl.Adjust.Enable	FALSE
VelModeCtrl.Adjust.Enable	FALSE
SyncModeCtrl.AdjCamShaftDistance.Enable	FALSE
SyncModeCtrl.AdjFineadjust.Enable	FALSE
SyncModeCtrl.AdjPhaseOffset.Enable	FALSE
DiagStatus.Number	0
DiagStatus.Message	reading not activated
AdminStatus.Active	TRUE für aktive Antriebe
AdminStatus.AxisFeatures	MB_AXIS_FEATURES, abhängig vom Achstyp
AdminStatus.Name	Achsname, abhängig von der Parametrierung

Abb.6-11: Durch den Funktionsbaustein initialisierte Strukturelemente

*Die oben aufgelisteten Strukturelemente werden nur initialisiert, wenn Eingang **ReadModulo = TRUE** ist und folgender Zustand existiert:*

- VelModeCtrl.Adjust.ModuloValue = 0
- SyncModeCtrl.AdjCamShaftDistance.ModuloValue = 0
- SyncModeCtrl.AdjFineadjust.ModuloValue = 0
- Wenn die Achswichtung auf Absolutformat gesetzt ist:
  - PosModeCtrl.Adjust.ModuloValue = 0
  - SyncModeCtrl.AdjPhaseOffset.ModuloValue = 0
- Wenn die Achswichtung auf Moduloformat gesetzt ist:
  - PosModeCtrl.Adjust.ModuloValue = Modulo value
  - SyncModeCtrl.AdjPhaseOffset.ModuloValue = Modolu value

Fehlerbehandlung Der Funktionsbaustein generiert keine speziellen Fehlercodes. Nur Fehlercodes der intern benutzten Funktionsbausteine zum Lesen der Parameter werden durchgereicht.

6.3.2 MB_AxisInterfaceType01

Kurzbeschreibung Der MB_AxisInterfaceType01 Funktionsbaustein wird zur Konfiguration des Achs-Interfaces für eine einzelne Achse benutzt. Dieser Funktionsbaustein muss zyklisch (im Motion-Takt oder langsamer als der Motion-Takt) aufgerufen werden solange man sich im Betriebs-Modus befindet.

ML_TechInterface.library

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
IndraMotion MLC ab 09VRS	ML_TechInterface.compiled-library
IndraMotion MLD ab MPx07 ohne Zusatzpaket MA	MX_AxisInterface.lib

Abb.6-12: Referenztabelle MB_AxisInterfaceType01

Interfacebeschreibung

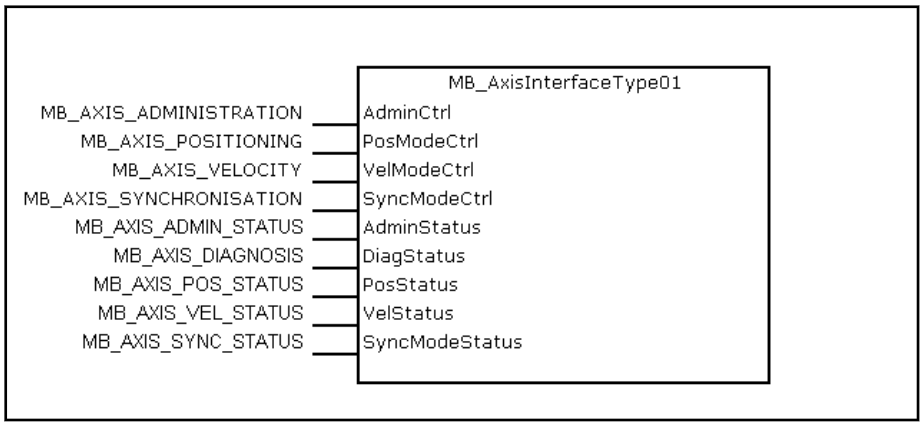


Abb.6-13: Funktionsbaustein MB_AxisInterfaceType01

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	AdminCtrl	REFERENCE TO MB_AXIS_ADMINISTRATION	Verwaltung der Achse, arAxisCtrl_gb[i].Admin anschließen
	PosModeCtrl	REFERENCE TO MB_AXIS_POSITIONING	Positionierungsbetriebsarten, arAxisCtrl_gb[i].PosMode anschließen
	VelModeCtrl	REFERENCE TO MB_AXIS_VELOCITY	Betriebsart Geschwindigkeits-Steuerung, arAxisCtrl_gb[i].VelMode anschließen
	SyncModeCtrl	REFERENCE TO MB_AXIS_SYNCHRONISATION	Synchronisations-Betriebsarten, arAxisCtrl_gb[i].SyncMode anschließen
	AdminStatus	REFERENCE TO MB_AXIS_ADMIN_STATUS	Status Verwaltung der Achse, arAxisStatus_gb[i].Admin anschließen
	DiagStatus	REFERENCE TO MB_AXIS_DIAGNOSIS	Status Diagnose der Achse, arAxisStatus_gb[i].Diag anschließen
	PosStatus	REFERENCE TO MB_AXIS_POS_STATUS	Status Positionierungsbetriebsarten, arAxisStatus_gb[i].PosMode anschließen
	VelStatus	REFERENCE TO MB_AXIS_VEL_STATUS	Status Geschwindigkeits-Betriebsart, arAxisStatus_gb[i].VelMode anschließen
	SyncModeStatus	REFERENCE TO MB_AXIS_SYNC_STATUS	Status Synchronisations-Betriebsart, arAxisStatus_gb[i].SyncMode anschließen

Abb.6-14: Schnittstellenvariablen MB_AxisInterfaceType01

ML_TechInterface.library

☞ Es ist nicht möglich die komplette Instanz der MB_Axis_Control_Type01 und der MB_Axis_Status_Type01 Strukturen (z. B. arAxisCtrl_gb[ ] / arAxisStatus_gb[ ]) über einen Eingang dem Funktionsbaustein zu übergeben. Dies wurde gemacht um der Anwender-Applikation spezielle Erweiterungen zum bestehenden Code zu ermöglichen. Nähere Details finden Sie unter Kap. 6.7 "Achs-Interface Anwender-Erweiterung" auf Seite 103.

Deshalb werden die benötigten Elemente von MB_Axis_Control_Type01 und MB_Axis_Status_Type01 als separate Eingänge übergeben.

Zur Performanceoptimierung sind die Strukturen statt als VAR_IN_OUT als VAR_INPUT mit REFERENCE TO angeschlossen. Damit reicht es einmalig die Eingänge zu initialisieren. Beim zyklischen Aufruf des Funktionsbausteins müssen damit die Strukturen nicht übergeben werden.

Der Funktionsbaustein überprüft die Eingänge von arAxisCtrl_gb[ ] und generiert intern die angeforderten Kommandos für die Achse. Die Ausgänge von arAxisStatus_gb[ ] werden aktualisiert in Abhängigkeit des Ergebnisses dieser Kommandos.

Zum Beispiel führt das Setzen von "arAxisCtrl_gb[ ].Admin._OpMode" von "ModeAb" auf "ModePosAbs" zu folgendem Ablauf:

- Überprüfen der erforderlichen Zustände zum Aktivieren eines Bewegungsbefehls, wie "Achse in Ab"
- Aktivierung des PLCopen-Funktionsbausteins MC_Power (wenn arAxisCtrl_gb.Admin.PowerOn = TRUE)
- Warten auf die Quittung, dass die Leistung der Achse zugeschaltet ist (AH/AF)
- Aktivierung des PLCopen-Funktionsbausteins MC_MoveAbsolute mit den Sollwerten von PosModeCtrl
- Quittieren des arAxisStatus_gb[ ].Admin._OpModeAck auf ModePosAbs (Bit MODE_POS_ABS)
- Scannen der Werte PosModeCtrl.Position und PosModeCtrl.Velocity und erneutes Aktivieren des MC_MoveAbsolute im Fall von Änderungen

---

☞ Wird der OpMode "arAxisCtrl_gb[].Admin._OpMode" mit "ModeCoordinated" und der Funktionsbaustein MB_AxisInterfaceType01 verwendet, führt es zur Fehlermeldung "OpMode" (Betriebsart wird nicht unterstützt).

Fehlerbehandlung

Der Funktionsbaustein erzeugt in DiagStatus.ErrorIdent.Additional1/Additional2 für die Tabelle "F_RELATED_TABLE", 16#0170 die folgenden Fehlermeldungen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR, (16#0005)	16#0006	16#0000	Ungültiger Zustand des Funktionsbausteins
DEVICE_ERROR, (16#0008)	16#1500	16#0000	Antrieb meldet einen Fehler in AxisData[ ]
INPUT_INVALID_ERROR, (16#0001)	16#1501	16#0000	Ungültige Betriebsart angewählt

ErrorID	Additional1	Additional2	Beschreibung
RESOURCE_ERROR, (16#0003)	16#1502	16#0000	Antrieb ist nicht in AB
INPUT_INVALID_ERROR, (16#0001)	16#1503	16#0000	Ungültige AdminCtrl.Axis.AxisNo
RESOURCE_ERROR, (16#0003)	16#150F	16#0000	AdminCtrl.PowerOn = FALSE UND Antrieb ist nicht in AH/AF
INPUT_INVALID_ERROR, (16#0010)	16#1510	OpMode	Betriebsart wird nicht unterstützt
INPUT_INVALID_ERROR, (16#0011)	16#1511	16#0000	Mindestens ein Eingang hat keine Referenz

Abb.6-15: Fehlercodes MB_AxisInterfaceType01

6.3.3 MB_AxisInterfaceType02

Kurzbeschreibung

Der MB_AxisInterfaceType02 Funktionsbaustein wird zur Konfiguration des Achs-Interfaces für eine einzelne Achse benutzt, welche im Rahmen einer Robot Control Applikation verwendet wird. Dieser Funktionsbaustein muss zyklisch (im Motion-Takt oder langsamer als der Motion-Takt) aufgerufen werden solange man sich im Betriebs-Modus befindet.

Zuordnung Zielsystem/Bibliothek

Zielsystem	Bibliothek
IndraMotion MLC ab 13VRS	ML_TechInterface.compiled-library

Abb.6-16: Referenztabelle AxisInterfaceType02

Interfacebeschreibung

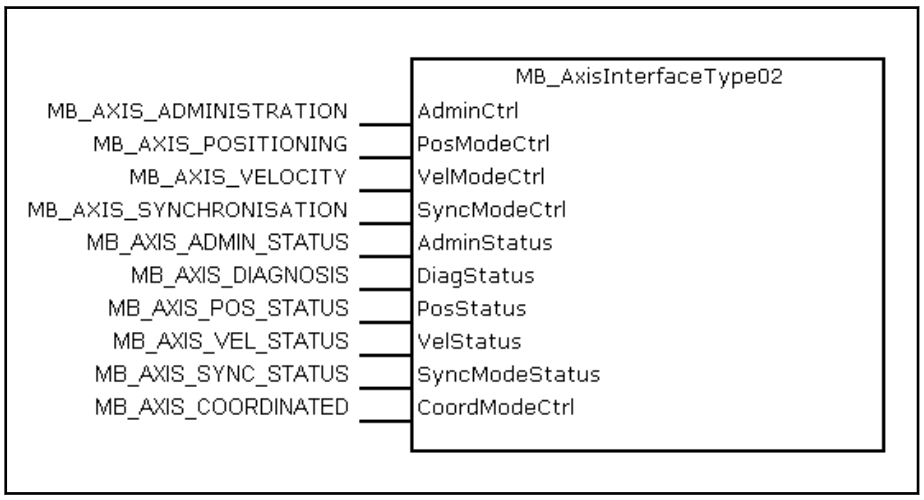


Abb.6-17: Funktionsbaustein MB_AxisInterfaceType02

ML_TechInterface.library

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	AdminCtrl	REFERENCE TO MB_AXIS_ADMINISTRATION	Verwaltung der Achse, arAxisCtrl_gb[i].Admin anschließen
	PosModeCtrl	REFERENCE TO MB_AXIS_POSITIONING	Positionierungsbetriebsarten, arAxisCtrl_gb[i].PosMode anschließen
	VelModeCtrl	REFERENCE TO MB_AXIS_VELOCITY	Betriebsart Geschwindigkeits-Steuerung, arAxisCtrl_gb[i].VelMode anschließen
	SyncModeCtrl	REFERENCE TO MB_AXIS_SYNCHRONISATION	Synchronisations-Betriebsarten, arAxisCtrl_gb[i].SyncMode anschließen
	AdminStatus	REFERENCE TO MB_AXIS_ADMIN_STATUS	Status Verwaltung der Achse, arAxisStatus_gb[i].Admin anschließen
	DiagStatus	REFERENCE TO MB_AXIS_DIAGNOSIS	Status Diagnose der Achse, arAxisStatus_gb[i].Diag anschließen
	PosStatus	REFERENCE TO MB_AXIS_POS_STATUS	Status Positionierungsbetriebsarten, arAxisStatus_gb[i].PosMode anschließen
	VelStatus	REFERENCE TO MB_AXIS_VEL_STATUS	Status Geschwindigkeits-Betriebsart, arAxisStatus_gb[i].VelMode anschließen
	SyncModeStatus	REFERENCE TO MB_AXIS_SYNC_STATUS	Status Synchronisations-Betriebsart, arAxisStatus_gb[i].SyncMode anschließen
	CoordModeCtrl	MB_AXIS_COORDINATED	CoordinatedMotion, arAxisCtrl_gb[i]. CoordMode anschließen

Abb.6-18: Schnittstellenvariablen MB_AxisInterfaceType02

☞ Der Funktionsbaustein MB_AxisInterfaceType02 kann nur auf der Steuerung IndraMotion MLC verwendet werden.

☞ Wird der OpMode "arAxisCtrl_gb[].Admin._OpMode" mit "ModeCoordinated" mit dem Funktionsbaustein MB_AxisInterfaceType01 verwendet, führt es zur Fehlermeldung "OpMode" (Betriebsart wird nicht unterstützt).

Der Funktionsbaustein besitzt alle Eingänge und Funktionen des Funktionsbausteins MB_AxisInterfaceType01 und zusätzlich den Eingang "CoordModeCtrl" für eine koordinierte Bewegung einer Achse.

Genereller Ablauf für eine koordinierte Bewegung mit TechInterface

- Betriebsart "MODE_COORDINATED" für alle an der entsprechenden Kinematik beteiligten Achsen mit dem Funktionsbaustein MB_AxisInterfaceType02 einstellen
- Kinematikprogramm(e) übersetzen und auf die Steuerung IndraMotion MLC laden
- Kinematikprogramm selektieren (mit dem Funktionsbaustein MB_IRDProgSelect)

- Kinematikprogramm starten (mit dem Funktionsbaustein MB_IRDProg-Start)
- Koordinierte Bewegung wird ausgeführt

  ...
- Kinematik stoppen (mit dem Funktionsbaustein MB_IRDProgStop)
- Betriebsart "MODE_COORDINATED" für alle an der entsprechenden Kinematik beteiligten Achsen mit dem Funktionsbaustein MB_AxisInterfaceType02 wieder wegnehmen



Die Kinematikprogramme können in IEC oder in der RCL (Robot Control Language) programmiert werden.

Fehlerbehandlung

Der Funktionsbaustein erzeugt in DiagStatus.ErrorIdent.Additional1/Additional2 für die Tabelle "F_RELATED_TABLE", 16#0170 die folgenden Fehlermeldungen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR, (16#0005)	16#0006	16#0000	Ungültiger Zustand des Funktionsbausteins
DEVICE_ERROR, (16#0008)	16#1500	16#0000	Antrieb meldet einen Fehler in AxisData[ ]
INPUT_INVALID_ERROR, (16#0001)	16#1501	16#0000	Ungültige Betriebsart angewählt
RESOURCE_ERROR, (16#0003)	16#1502	16#0000	Antrieb ist nicht in AB
INPUT_INVALID_ERROR, (16#0001)	16#1503	16#0000	Ungültige AdminCtrl.Axis.AxisNo
RESOURCE_ERROR, (16#0003)	16#150F	16#0000	AdminCtrl.PowerOn = FALSE UND Antrieb ist nicht in AH/AF
INPUT_INVALID_ERROR, (16#0011)	16#1511	16#0000	Mindestens ein Eingang hat keine Referenz

Abb.6-19: Fehlercodes MB_AxisInterfaceType02

6.4 Achsbetriebsarten

6.4.1 Dialogbasierte Achskommandierung

Vorteile

Zur komfortablen Achskommandierung steht für alle Programmiervorlagen von Bosch Rexroth die dialogbasierte Achskommandierung zur Verfügung. Hierbei können Sie in einem Dialog alle notwendigen Sollwerte der ausgewählten Achse eingeben, und den entsprechenden SPS Code hierfür erzeugen und einfügen lassen. Dieses Vorgehen spart Zeit, reduziert Fehler und Sie benötigen keine Kenntnis der genauen Syntax Regeln.

Aufruf

Der Dialog wird über das Kontextmenü "Achse kommandieren" aufgerufen. Die Funktion wird angeboten, wenn der Cursor in dem "Structured Text (ST)-Editor" an einer für die Achskommandierung zulässigen Stelle steht.

ML_TechInterface.library

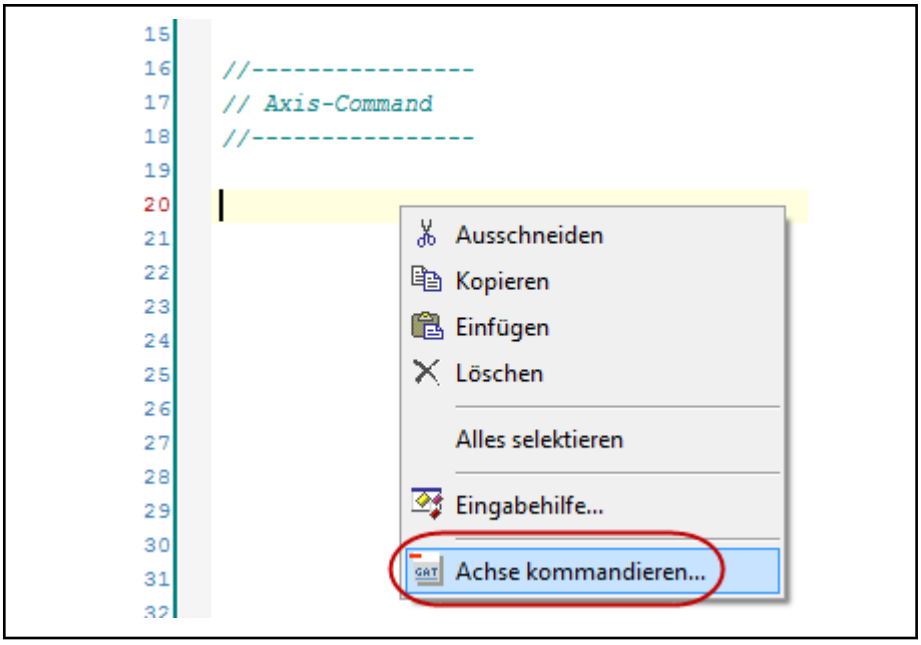


Abb.6-20: Aufruf der dialogbasierten Achskommandierung

Die in den nachfolgenden Kapiteln beschriebenen Regeln zur Erzeugung des SPS Codes werden hierbei automatisch berücksichtigt. Die nachfolgenden Kapitel dienen daher nur als Nachschlagewerk für spezielle Fälle, oder für die Interpretation des generierten Codes.

An folgenden Stellen im Programmcode wird die dialogbasierte Achskommandierung angeboten:

- Achs-Interface Beispielprojekt: Alle Programme, Funktionsbausteine und Funktionen
- GAT^{compact}: Machine FB
- GATzentral: In allen Modulen
- GATdezentral: In allen Units

Die Funktion hat folgende Merkmale:

Merkmale

- Vorgabe der Sollbetriebsart und der zugehörigen Sollwerte komfortabel über einen Dialog
- Sollwertvorgaben auch über SPS-Variablen möglich
- In den Eingabefeldern kann mit der F2 Taste die Eingabehilfe aufgerufen werden
- Optionales Hinzufügen von auskommentiertem Quittierungscode-Beispiel, zum Abfragen ob die Betriebsart aktiv ist, und Sollwerte erreicht sind
- Nach Betätigen von **Fertigstellen** wird der SPS Code an die Stelle des Cursors hinzugefügt
- Standardwerte (Initialwerte) sind im Dialog ersichtlich
- Keine Codeerzeugung für Elemente ohne Wertezuweisung (in diesem Falle wirken die Standardwerte bzw. die an anderer Stelle vorgegebenen Werte)

Beispiel

In dem folgenden Beispiel wird die Achse "MyVirtualAxis1" absolut an die vorgegebene Position "10" positioniert.

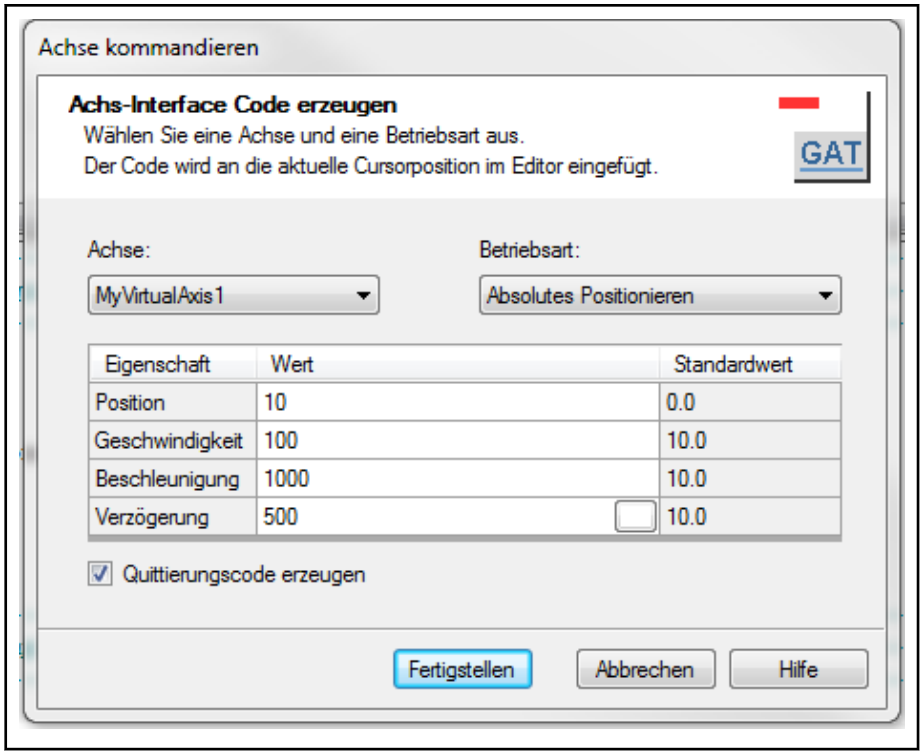


Abb.6-21: Beispiel für das absolute Positionieren der Achse "MyVirtualAxis1" über die dialogbasierte Achskommandierung

Mit **Fertigstellen** wird der folgende Programmcode erzeugt, und an die Cursorstelle eingefügt:

```
15
16  //----------------
17  // Axis-Command
18  //----------------
19  arAxisCtrl_gb[MyVirtualAxis1.AxisNo].PosMode.Position := 10;
20  arAxisCtrl_gb[MyVirtualAxis1.AxisNo].PosMode.Velocity := 100;
21  arAxisCtrl_gb[MyVirtualAxis1.AxisNo].PosMode.Acceleration := 1000;
22  arAxisCtrl_gb[MyVirtualAxis1.AxisNo].PosMode.Deceleration := 500;
23  arAxisCtrl_gb[MyVirtualAxis1.AxisNo].Admin._OpMode.en := ModePosAbs;
24  (* Code example to check if the commanded operation mode is acknowledged
25  IF ((arAxisStatus_gb[MyVirtualAxis1.AxisNo].Admin._OpModeAck = ModePosAbs) AND (arAxisStatus_gb[MyVirtualAxis1.AxisNo].Admin.CmdDone)) THEN
26    // commanded operation mode is acknowledged
27    ;
28  END_IF
29  *)
```

*Abb.6-22: Hinzugefügter Code nach **Fertigstellen***

Der hier gezeigte Programmcode entspricht der Syntax für das Achs-Interface Beispielprojekt. Für alle anderen GAT Varianten wird ein modifizierter Programmcode verwendet, da hierbei der Achscontainer "Axes" verwendet wird. Die dialogbasierte Achskommandierung erkennt automatisch alle GAT Varianten, und erzeugt immer den passenden Programmcode.

6.4.2 Überblick

Es gibt zwei Methoden um eine Betriebsart zu aktivieren:

1. **Auswahl über Enum-Werte**

   Zuweisen eines Wertes vom TYPE MB_AXIS_MODE an arAxisCtrl_gb[ ].Admin._OpMode.en:

   `arAxisCtrl_gb[].Admin._OpMode.en:= ModePosAbs;`

   - oder -

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModeAB;
```

2. **Benutzung des Bit-Zugriffs**

   Zuerst Löschen des "_OpMode" durch Programmierung...

   ```
   arAxisCtrl_gb[].Admin._OpMode.en:= ModeAB;
   ```

   dann Setzen eines Bits über die Bit-Zugriffs Funktionalität.

   ```
   arAxisCtrl_gb[].Admin._OpMode.b.MODE_POS_ABS:= TRUE;
   ```

☞ Das Setzen mehrerer Bits in `arAxisCtrl_gb[].Admin._OpMode` verursacht einen Fehler und der Antrieb schaltet in AB.

Bevor eine Betriebsart aktiviert werden kann, müssen jedem Attribut zuerst Werte zugewiesen werden. Alle Attribute haben Standardwerte. Einige haben Werte ungleich Null, während andere als 0 definiert sind und, aufgrund der speziellen Anforderungen, ihnen ein Wert zugewiesen werden muss.

☞ Nur die Attribute (z. B. Position, Geschwindigkeit), die benutzt werden oder deren Standardwert geändert wurde, müssen deklariert werden, bevor der aktuelle Betriebsarten-Wechsel ausgeführt wird.

Das folgende Beispiel zeigt die korrekte Reihenfolge zum Programmieren eines Betriebsarten-Wechsels:

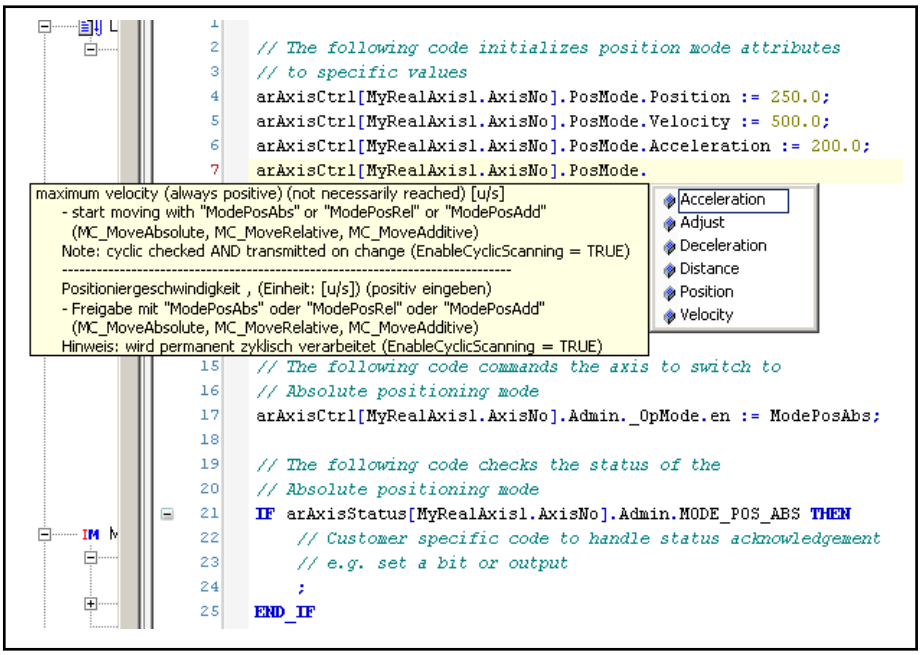


Abb.6-23: Beispiel eines absoluten Positionierungsbetrieb-Wechsels

☞ Die Status-Quittung (`arAxisStatus_gb[].Admin.`) für eine Betriebsart ist wie folgt implementiert.

Beispiel:

1. Die Status-Quittung gibt nur dann `TRUE` zurück, wenn das Kommando zum Umschalten der Achse auf absolutem Positionierungsbetrieb ausgeführt wurde:

   `arAxisStatus_gb[].Admin.MODE_POS_ABS`

2. Die Status-Quittung wird nur für die Betriebsart Antrieb Halt, das Referenzieren des Antriebs und für Positionierungsbetriebsarten unterstützt:

   `arAxisStatus_gb[].Admin.CmdDone`

   Die Positionierungsbetriebsart wird aktiviert:

   `arAxisCtrl_gb[].Admin._OpMode.b.MODE_POS_ABS`

   Der Ausgang wird auf `TRUE` gesetzt, wenn der Antrieb in den Positionierungsbetrieb schaltet und anfängt sich zu drehen:

   `arAxisStatus_gb[].Admin.MODE_POS_ABS`

   Hat der Antrieb die Zielposition erreicht, wird die Status-Quittung gesetzt:

   `arAxisStatus_gb[].Admin.CmdDone`

Der Eingang arAxisCtrl_gb.Admin.PowerOn steuert die interne Verwendung des MC_Power im Achs-Interface:

- PowerOn = TRUE (Standard): das Achs-Interface übernimmt die Ansteuerung des MC_Power. Bei Aktivierung einer Betriebsart wird intern zuerst der MC_Power aufgerufen und erst nach Bereitmeldung wird die eigentlich angewählte Betriebsart aktiviert. Bei Abschaltung (ModeAb) wird der MC_Power nach Anhalten des Antriebes weggenommen
- PowerOn = FALSE: das Applikationsprogramm muss den MC_Power bedienen. Bei Anwahl einer Betriebsart ohne vorherige Ansteuerung des MC_Power wird ein Fehler vom Achs-Interface ausgegeben

Der folgende Abschnitt beschreibt die Betriebsarten, die vom Achs-Interface Type01 unterstützt werden und die Attribute, die zugewiesen werden können. Es wird im Folgenden vom Standardwert des Eingangs "PowerOn" (TRUE) ausgegangen.

6.4.3 Antrieb Bereit

Die Aktivierung dieser Betriebsart schaltet den Antrieb in AB (Antrieb Bereit) und schaltet das Drehmoment ab. Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModeAB;
```

☞ Das Achs-Interface benutzt intern den MC_Power und MC_Stop PLCopen-Funktionsbaustein, um die Umschaltung durchzuführen.

ML_TechInterface.library

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl[ ]	Admin.StopDeceleration	REAL	100.0	Ja
	Admin.Axis	AXIS_REF		Nein

Abb.6-24: Attribute Antrieb Bereit

6.4.4 Antrieb Halt

Die Aktivierung dieser Betriebsart schaltet den Antrieb in AH (Antrieb Halt) unter Aufrechterhaltung des Drehmoments. Die Betriebsart "ModeAH" überführt die Achse in den PLCopen Zustand "StandStill", d. h. ein außerhalb des Achs-Interfaces aufgerufener PLCopen Funktionsbaustein wird akzeptiert und das Achs-Interface meldet "ModeExternalFB".

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModeAH;
```

oder

```
arAxisCtrl_gb[].Admin.MODE_AH:= TRUE;
```

☞ Das Achs-Interface benutzt intern den MC_Power und MC_Stop PLCopen-Funktionsbaustein, um die Umschaltung durchzuführen.

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl[ ]	Admin.StopDeceleration	REAL	100.0	Ja
	Admin.Axis	AXIS_REF		Nein
arAxisStatus_gb[]	Admin.MODE_AH	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

Abb.6-25: Attribute Antrieb Halt

6.4.5 Antrieb Stopp

Die Aktivierung dieser Betriebsart schaltet den Antrieb in AH (Antrieb Halt) unter Aufrechterhaltung des Drehmoments. Die Betriebsart ModeStopping hält die PLCopen statemachine im Zustand "Stopping", d. h. ein außerhalb des Achs-Interfaces aufgerufener PLCopen Funktionsbaustein würde mit dem Fehler "Kommando im Zustand Stopping zurückgewiesen" beendet werden.

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModeStopping;
```

oder

```
arAxisCtrl_gb[].Admin.MODE_STOPPING:= TRUE;
```

☞ Das Achs-Interface benutzt intern den MC_Power und MC_Stop PLCopen-Funktionsbaustein, um die Umschaltung durchzuführen.

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	Admin.StopDeceleration	REAL	100.0	Ja
	Admin.Axis	AXIS_REF		Nein
arAxisStatus_gb[]	Admin.MODE_STOPPING	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

Abb.6-26: Attribute Antrieb Stopp

6.4.6 Antrieb Referenzieren

Die Aktivierung dieser Betriebsart führt das Kommando zum Referenzieren des Antriebs aus. Es steuert Achsen mit inkrementellen Gebern beim Referenziervorgang.

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl[].Admin._OpMode.en:= ModeHoming;
```

oder

```
arAxisCtrl[].Admin._OpMode.b.MODE_HOMING:= TRUE;
```

Die folgende Quittung muss dem Betriebsarten-Kommando folgen und gibt TRUE zurück, wenn der Antrieb seine Referenzposition erreicht hat:

```
arAxisStatus[].Admin.CmdDone
```


Das Achs-Interface benutzt intern den MC_Power und MB_Home PLCopen-Funktionsbaustein, um die Umschaltung durchzuführen.

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl[ ]	Admin.Axis	AXIS_REF		Nein
arAxisStatus[ ]	Admin.MODE_HOMING	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

Abb.6-27: Attribute Antrieb referenzieren

6.4.7 Absolutes Positionieren

Die Aktivierung dieser Betriebsart führt eine absolute Bewegung auf eine vorher festgelegte Zielposition aus.

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModePosAbs;
```

oder

```
arAxisCtrl_gb[].Admin._OpMode.b.MODE_POS_ABS:= TRUE;
```

Die folgende Quittung muss dem Betriebsarten-Kommando folgen und gibt TRUE zurück, wenn der Antrieb seine Zielposition erreicht hat:

```
arAxisStatus_gb[].Admin.CmdDone
```

☞ Das Achs-Interface benutzt intern den MC_Power und MC_MoveAbsolute PLCopen-Funktionsbaustein, um die Umschaltung durchzuführen.

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	PosMode.Position	REAL	0.0	Ja
	PosMode.Velocity	REAL	10.0	Ja
	PosMode.Acceleration	REAL	10.0	Nein
	PosMode.Deceleration	REAL	10.0	Nein
	Admin.Axis	AXIS_REF		Nein
arAxisStatus_gb[ ]	Admin.MODE_POS_ABS	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

Abb.6-28: Attribute Absolutes Positionieren

6.4.8 Relatives Positionieren

Die Aktivierung dieser Betriebsart führt eine relative Bewegung auf eine vorher festgelegte Zielposition durch Addition der PosMode.Distance zur aktuellen Istposition aus.

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModePosRel;
```

oder

```
arAxisCtrl_gb[].Admin._OpMode.b.MODE_POS_REL:= TRUE;
```

Die folgende Quittung muss dem Betriebsarten-Kommando folgen und gibt `TRUE` zurück, wenn der Antrieb seine Zielposition erreicht hat:

```
arAxisStatus_gb[].Admin.CmdDone
```

☞ Das Achs-Interface benutzt intern den MC_Power und MC_MoveRelative PLCopen-Funktionsbaustein, um die Umschaltung durchzuführen.

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	PosMode.Distance	REAL	0.0	Ja
	PosMode.Velocity	REAL	10.0	Ja
	PosMode.Acceleration	REAL	10.0	Nein
	PosMode.Deceleration	REAL	10.0	Nein
	Admin.Axis	AXIS_REF		Nein

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisStatus_gb[ ]	Admin.MODE_POS_REL	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

Abb.6-29: Attribute Relatives Positionieren

6.4.9 Additives Positionieren

Die Aktivierung dieser Betriebsart führt eine relative Bewegung auf eine vorher festgelegte Zielposition durch Addition der PosMode.Distance zur aktuellen Zielposition aus.

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModePosAdd;
```

oder

```
arAxisCtrl_gb[].Admin._OpMode.b.MODE_POS_ADD:= TRUE;
```

Die folgende Quittung muss dem Betriebsarten-Kommando folgen und gibt TRUE zurück, wenn der Antrieb seine Zielposition erreicht hat:

```
arAxisStatus_gb[].Admin.CmdDone
```

☞ Das Achs-Interface benutzt intern den MC_Power und MC_MoveAdditive PLCopen-Funktionsbaustein, um die Umschaltung durchzuführen.

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	PosMode.Distance	REAL	0.0	Ja
	PosMode.Velocity	REAL	10.0	Ja
	PosMode.Acceleration	REAL	10.0	Nein
	PosMode.Deceleration	REAL	10.0	Nein
	Admin.Axis	AXIS_REF		Nein
arAxisStatus_gb[ ]	Admin.MODE_POS_ADD	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

Abb.6-30: Attribute Additives Positionieren

6.4.10 Geschwindigkeits-Steuerung

Die Aktivierung dieser Betriebsart führt Bewegungen mit einer vorher festgelegten Geschwindigkeit aus, bis ein Halt-Kommando gegeben wird.

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModeVel;
```

oder

```
arAxisCtrl_gb[].Admin._OpMode.b.MODE_VEL:= TRUE;
```

☞ Das Achs-Interface benutzt intern den MC_Power und MC_MoveVelocity PLCopen-Funktionsbaustein, um die Umschaltung durchzuführen.

ML_TechInterface.library

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	VelMode.Velocity	REAL	10.0	Ja
	VelMode.Acceleration	REAL	10.0	Ja
	VelMode.Deceleration	REAL	10.0	Ja
	Admin.Axis	AXIS_REF		Nein
arAxisStatus_gb[ ]	Admin.MODE_VEL	BOOL		entfällt

Abb.6-31: Attribute Geschwindigkeits-Steuerung

6.4.11 Betriebsart "Winkelsynchronisation"

Bei Aktivierung dieser Betriebsart wird der Antrieb mittels eines elektronischen Getriebes mit Feinabgleich betrieben. Diese Funktionalität führt eine Winkelsynchronisation zwischen Leitachse und angewählter Folgeachse durch.

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModeSyncPhase;
```

oder

```
arAxisCtrl_gb[].Admin._OpMode.b.MODE_SYNC_PHASE:= TRUE;
```

☞ Das Achs-Interface benutzt intern den MC_Power, MB_GearInPos, MC_GearOut und MB_PhasingSlave PLCopen-Funktionsbaustein, um die Umschaltung durchzuführen.

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	SyncMode.OutputRevolution	UINT	1	Ja
	SyncMode.InputRevolution	UINT	1	Ja
	SyncMode.Fineadjust	REAL	0.0	Ja
	SyncMode.SyncDirection	MC_SYNC_DIRECTION	SHORTEST_WAY	Ja
	SyncMode.StartMode	MC_START_MODE	ABSOLUTE	Nein
	SyncMode.Master	AXIS_REF		Ja
	Admin.Axis	AXIS_REF		Nein
	SyncMode.PhaseOffset	REAL	0.0	Ja
	SyncMode.PhaseOffsetVel	REAL	1.0	Nein
	SyncMode.PhaseOffsetAcc	REAL	100.0	Nein
arAxisStatus_gb[ ]	Admin.MODE_SYNC_PHASE	BOOL		entfällt
	SyncMode.PhasingSlaveDone	BOOL		entfällt

Abb.6-32: Attribute Betriebsart "Winkelsynchronisation"

6.4.12 Betriebsart "Geschwindigkeitssynchronisation"

Bei Aktivierung dieser Betriebsart wird der Antrieb mittels eines elektronischen Getriebes mit Feinabgleich betrieben. Diese Funktionalität führt eine Geschwindigkeitssynchronisation zwischen Leitachse und angewählter Folgeachse durch.

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModeSyncVel;
```

oder

```
arAxisCtrl_gb[].Admin._OpMode.b.MODE_SYNC_VEL:= TRUE;
```

☞ Das Achs-Interface benutzt intern den MC_Power, MC_GearIn und MC_GearOut PLCopen-Funktionsbaustein, um die Umschaltung durchzuführen.

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	SyncMode.OutputRevolution	UINT	1	Ja
	SyncMode.InputRevolution	UINT	1	Ja
	SyncMode.Fineadjust	REAL	0.0	Ja
	SyncMode.Master	AXIS_REF		Ja
	Admin.Axis	AXIS_REF		Nein
arAxisStatus_gb[ ]	Admin.MODE_SYNC_VEL	BOOL		entfällt

Abb.6-33: Attribute Betriebsart "Geschwindigkeitssynchronisation"

6.4.13 Betriebsart "Elektronische Kurvenscheibe"

Die Aktivierung dieser Betriebsart veranlasst den Antrieb dem Master unter Nutzung eines Kurvenscheiben-Profils, das im Antrieb gespeichert ist, zu folgen. Diese Betriebsart wird nur von IndraDrive-Antrieben unterstützt.

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModeSyncCam;
```

oder

```
arAxisCtrl_gb[].Admin._OpMode.b.MODE_SYNC_CAM:= TRUE;
```

☞ Das Achs-Interface benutzt intern den MC_Power, MC_CamIn, MC_CamOut und MB_PhasingSlave PLCopen-Funktionsbaustein, um die Umschaltung durchzuführen.

ML_TechInterface.library

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	SyncMode.OutputRevolution	UINT	1	Ja
	SyncMode.InputRevolution	UINT	1	Ja
	SyncMode.Fineadjust	REAL	0.0	Ja
	SyncMode.CamShaftDistance	REAL	360.0	Ja
	SyncMode.SyncDirection	MC_SYNC_DIRECTION	SHORTEST_WAY	Ja
	SyncMode.StartMode	MC_START_MODE	ABSOLUTE	Nein
	SyncMode.CamTableID	MC_CAM_ID	CAM_TABLE_1	Ja
	SyncMode.Master	AXIS_REF		Ja
	Admin.Axis	AXIS_REF		Nein
	SyncMode.PhaseOffset	REAL	0.0	Ja
	SyncMode.PhaseOffsetVel	REAL	1.0	Nein
	SyncMode.PhaseOffsetAcc	REAL	100.0	Nein
arAxisStatus_gb[ ]	SyncMode.PhasingSlaveDone	BOOL		
	SyncMode.CamActiveID	MC_CAM_ID		
	Admin.MODE_SYNC_CAM	BOOL		entfällt

Abb.6-34: Attribute Betriebsart "Elektronische Kurvenscheibe"

6.4.14 Betriebsart "Elektronisches Bewegungsprofil"

Die Aktivierung dieser Betriebsart schaltet im Antrieb die elektronische Kurvenscheibenfunktion ein. Die Kurvenscheibenprofile sind folgerichtig durch Nutzung der Parameter für das elektronische Bewegungsprofil definiert. Ein synchroner Positionssollwert für die Folgeachse wird durch die Position der Leitachse bestimmt. Diese Betriebsart kann nur bei virtuellen Achsen oder mit generischen Antrieben (Profil wird in der Steuerung berechnet) benutzt werden.

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModeSyncProfile;
```

oder

```
arAxisCtrl_gb[].Admin._OpMode.b.MODE_SYNC_PROFILE:= TRUE;
```

☞ Das Achs-Interface benutzt intern den MC_Power, MB_MotionProfile, MC_GearOut und MB_PhasingSlave PLCopen-Funktionsbaustein, um die Umschaltung durchzuführen.

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	SyncMode.OutputRevolution	UINT	1	Ja
	SyncMode.InputRevolution	UINT	1	Ja
	SyncMode.Fineadjust	REAL	0.0	Ja
	SyncMode.Profile.SetSelection	UINT	0	Ja
	SyncMode.Profile.RelativePositioning	BOOL	FALSE	Nein
	SyncMode.SyncDirection	MC_SYNC_DIRECTION	SHORTEST_WAY	Ja
	SyncMode.StartMode	MC_START_MODE	ABSOLUTE	Nein
	SyncMode.Master	AXIS_REF		Ja
	Admin.Axis	AXIS_REF		Nein
	SyncMode.PhaseOffset	REAL	0.0	Ja
	SyncMode.PhaseOffsetVel	REAL	1.0	Nein
	SyncMode.PhaseOffsetAcc	REAL	100.0	Nein
arAxisStatus_gb[ ]	SyncMode.PhasingSlaveDone	BOOL		entfällt
	SyncMode.ProfileActiveSet	UINT		entfällt
	Admin.MODE_SYNC_CAM	BOOL		entfällt

Abb.6-35: Attribute Betriebsart "Elektronisches Bewegungsprofil"

6.4.15 Betriebsart "Elektronisches FlexProfile"

Die Aktivierung dieser Betriebsart schaltet im Antrieb die elektronische Kurvenscheibenfunktion ein. Die Kurvenscheibenprofile sind folgerichtig durch Nutzung der Parameter für das elektronische FlexProfile definiert. Diese Betriebsart kann nur bei virtuellen Achsen oder mit generischen Antrieben (Profil wird in der Steuerung berechnet) benutzt werden. Auf dem antriebsbasierten IndraMotion MLD-System ist diese Betriebsart nicht verfügbar.

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModeFlexProfile;
```

oder

```
arAxisCtrl_gb[].Admin._OpMode.b.MODE_FLEX_PROFILE:= TRUE;
```

☞ Das Achs-Interface benutzt intern den MC_Power, ML_FlexProfile, MC_GearOut und MB_PhasingSlave PLCopen-Funktionsbaustein, um die Umschaltung durchzuführen.

ML_TechInterface.library

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	SyncMode.OutputRevolution	UINT	1	Ja
	SyncMode.InputRevolution	UINT	1	Ja
	SyncMode.Fineadjust	REAL	0.0	Ja
	SyncMode.Profile.SetSelection	UINT	0	Ja
	SyncMode.Profile.UseSwitchingPosition	BOOL	FALSE	Nein
	SyncMode.Profile.SwitchingPosition	REAL	0.0	Nein
	SyncMode.Profile.ProfileEntry	MB_PROFILE_START_POINT	SLAVE_ORIGIN_MASTER_ORIGIN	Nein
	SyncMode.Profile.MasterOffset	REAL	0.0	Nein
	SyncMode.Profile.SlaveOffset	REAL	0.0	Nein
	SyncMode.Profile.SyncType	MB_SYNC_TYPE	SYNC_RAMPIN_SHORTEST_WAY	Nein
	SyncMode.Profile.ExecutionMode	MB_EXECUTION_MODE	EXECUTE_CYCLIC	Nein
	SyncMode.Master	AXIS_REF		Ja
	Admin.Axis	AXIS_REF		Nein
	SyncMode.PhaseOffset	REAL	0.0	Ja
	SyncMode.PhaseOffsetVel	REAL	1.0	Nein
	SyncMode.PhaseOffsetAcc	REAL	100.0	Nein
arAxisStatus_gb[ ]	SyncMode.PhasingSlaveDone	BOOL		entfällt
	SyncMode.ProfileActiveSet	UINT		entfällt
	Admin.MODE_FLEX_PROFILE	BOOL		entfällt

Abb.6-36: Attribute Betriebsart "Elektronisches Flexprofile"

6.4.16 Betriebsart "Robot Control"

Die Aktivierung dieser Betriebsart schaltet den entsprechenden Antrieb in zyklische Lageregelung (antriebsgeführt). Des Weiteren wird die entsprechende Achse automatisch zu einer Kinematik gruppiert. Die so gruppierten Achsen können dann mittels eines IEC- bzw. RCL-Programms (Robot Control Language) eine koordinierte Bewegung ausführen.

Auf dem antriebsbasierten IndraMotion MLD-System ist diese Betriebsart nicht verfügbar (Voraussetzung hierfür ist der Funktionsbaustein MB_AxisInterfaceType02, der auf diesem System nicht zur Verfügung steht).

Auf der Steuerung IndraLogic XLC ist diese Betriebsart ebenfalls nicht verfügbar.

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModeCoordinated;
```

oder

```
arAxisCtrl_gb[].Admin._OpMode.b.MODE_COORDINATED:= TRUE;
```

Die Rückmeldung dieser Betriebsart TRUE erfolgt in der Achsstatusstruktur:

```
arAxisStatus_gb[].Admin.MODE_COORDINATED;
```

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	CoordMode.AxisGroupRef.CtrlNo	CONTROLS	LOCAL_CNTRL	Nein
	CoordMode.AxisGroupRef.Group-No	KIN_OBJECTS	NO_KINEMATICS	Nein

Abb.6-37: Attribute Betriebsart "Robot Control"

Deklaration:

z. B: Kinematik1: AXIS_GROUP_REF:= (CntrlNo:= LOCAL_CNTRL,Group-No:= KINEMATICS_1);

Zuweisung:

z. B: arAxisCtrl_gb[Antrieb1] .Ctrl.CoordMode.AxisGroupRef:= Kinematik1;

☞ Das Achs-Interfacee benutzt intern die PLCopen-Funktionsbausteine MC_Power, MC_AddAxisToGroup und MC_RemoveAxisFromGroup, um die Umschaltung durchzuführen.

6.4.17 Betriebsart "Antriebsgeführte Unterbrechung"

Diese Betriebsart dient zur Anzeige einer antriebsgeführten Unterbrechung.

Das Achs-Interface überwacht den Zustand des Antriebes mit dem Bit AxisData[].Axis_Interrupted und wechselt bei TRUE im AxisStatus in "ModeInterrupted". Das Achs-Interface meldet damit die Übernahme der Kontrolle durch den Antrieb, z. B. durch die Sicherheitstechnik im Antrieb.

Durch Rücksetzen des Eingangs arAxisCtrl_gb[].Admin.DriveInterruptReaction kann diese Überwachung deaktiviert werden. Standardeinstellung ist "Überwachung aktiv".

Eine Vorgabe dieser Betriebsart ist nicht vorgesehen. Falls trotzdem "ModeInterrupted" vorgegeben wird, wird wie bei "ModeExternalFb" der Antrieb in AH (MC_Stop ist NICHT aktiv) geschaltet.

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	Admin.DriveInterruptReaction	BOOL	TRUE	Ja
arAxisStatus_gb[ ]	Admin.MODE_INTERRUPTED	BOOL		entfällt

Abb.6-38: Attribute Betriebsart "Antriebsgeführte Unterbrechung"

Anwendungsfälle der Überwachung von Axis_Interrupted und des Eingangs DriveInterruptReaction:

- Anfahren bei Antrieb mit Bremse und Interpolation in der Steuerung: Bei einem großen Wert in S-0-0206 "Wartezeit Antrieb Ein" startet der Interpolator in der Steuerung bereits mit der Vorgabe von Sollwerten, ohne dass der Antrieb folgt. Dies kann am Ende der Wartezeit zu einem Sollwertsprung führen. Die Überwachung beim Anfahren ist immer aktiv, unabhängig vom Eingang "DriveInterruptReaction"
- Sicherheitstechnik oder antriebsinterne Kommandos (z. B. Bremsentest): "DriveInterruptReaction" = TRUE. Wenn in einer Betriebsart das Axis_Interrupted kommt, wird in arAxisStatus_gb[].Admin.MODE_INTERRUPTED gemeldet, bis eine Änderung der angewählten Betriebsart erfolgt. Bei steigender Flanke des "Axis_Interrupted" wird Geschwindigkeitsregelung mit Geschwindigkeit Null vorgegeben. Wird vor Ende der Unterbrechung bereits eine neue Betriebsart vorgegeben, geht es direkt nach Ende des "Axis_Interrupted" mit der neu angewählten Betriebsart weiter
- Anwendungen die Antriebsfunktionalität nutzen (z. B. Messtaster Schnellhalt): Die Überwachung durch das Achs-Interface ist hier nicht gewünscht und wird mit "DriveInterruptReaction" = FALSE abgeschaltet
- Temporäre Abschaltung: die Überwachung kann auch zeitweise abgeschaltet werden. Beispiel: beim Referenzieren mit Festanschlag kann nur mit "DriveInterruptReaction" = FALSE vom Festanschlag weggefahren werden

6.4.18 Betriebsart "Externer Funktionsbaustein"

Die Aktivierung dieser Betriebsart schaltet den Antrieb in AH (MC_Stop ist NICHT aktiv) und wartet auf einen externen Bewegungsbefehl, z. B. von einer Technologiefunktion.

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode.en:= ModeExternalFB;
```

oder

```
arAxisCtrl_gb[].Admin._OpMode.b.MODE_EXTERNAL_FB:= TRUE;
```

☞ Das Achs-Interface benutzt intern den MC_Power PLCopen-Funktionsbaustein, um die Umschaltung durchzuführen.

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	Admin.Axis	AXIS_REF		Nein
arAxisStatus_gb[ ]	Admin.MODE_EXTERNAL_FB	BOOL		entfällt

Abb.6-39: Attribute Betriebsart "Externer Funktionsbaustein"

6.5 Globale Variablen

Die Bibliothek ML_TechInterface (bei IndraMotion MLD MX_AxisInterface.lib und MX_ImcInterface.lib) enthält zwei Globale Variablenlisten:

1. **Global_MB_AxisInterfaceVars**.

   Diese Liste definiert Konstanten.

2. **Global_ImcInterface**.

   Diese Liste definiert die Steuer- und Status-Strukturen des Imc-Interface

- Steuer-Struktur: ImcCtrl
- Status-Struktur: ImcStatus

Global_MB_AxisInterfaceVars

```
1  VAR_GLOBAL CONSTANT
2      (* define masks for the sync operating modes *)
3      (* e.g. IF( (INT_TO_WORD(AdminStatus._OpMode.enAck) AND MASK_MODE_WITH_FINE_ADJ) <> 0 ) THEN *)
4          MASK_MODE_FINE_ADJUST    :WORD := 16#0BC0; (* SYNC_PHASE, SYNC_VEL, SYNC_CAM,
5                                                       SYNC_PROFILE, FLEX_PROFILE *)
6          MASK_MODE_PHASE_OFFSET   :WORD := 16#0B40; (* SYNC_PHASE, SYNC_CAM, SYNC_PROFILE,
7                                                       FLEX_PROFILE *)
8          MASK_MODE_CAM_SHAFT_DIST :WORD := 16#0100; (* SYNC_CAM *)
9
10     (* define masks for the axis config types *)
11     (* e.g. IF( (INT_TO_WORD(AdminStatus.AxisType) AND MASK_AXIS_COMMAND_POSSIBLE) <> 0 ) THEN *)
12         MASK_AXIS_COMMAND_POSSIBLE :WORD := 16#0098; (* RealAxisCfg, VirtualAxisCfg, ControlAxisCfg *)
13
```

Global_ImcInterface

```
1  VAR_GLOBAL
2
3    ImcCtrl:   MB_IMC_CONTROL_TYPE01; (* Instance in order to command the control *)
4    ImcStatus: MB_IMC_STATUS_TYPE01;  (* Instance for getting control status information *)
5
6  END_VAR
7
```

Abb.6-40: Globale Variablen des TechInterface

6.6 Erste Schritte Achs-Interface

6.6.1 Überblick

Das Programmiertemplate "Achs-Interface Beispielprojekt" dient als Beispielapplikation für das Achs-Interface.

Dieses Beispielprojekt kann jederzeit über den GAT-Wizard erzeugt werden:

1. Neues IndraWorks-Projekt anlegen.
2. Steuerung IndraMotion MLC einfügen.
3. Den GAT-Wizard über das Symbol links oben in der Menüleiste starten.
4. Achs-Interface-Beispielprojekt auswählen und mit der Schaltfläche **Fertigstellen** den Import des Projekts starten.

6.6.2 Programm zum Achs-Interface

*Das Programm **TE_AxisInterfaceType01Prog** des Beispielprojekts deckt die folgenden Punkte ab:*

- **Initialisierung des Achs-Interface**:

  Bei Erreichen des Parametriermodus (Phase 2) wird für alle konfigurierten Achsen mit Hilfe des Funktionsbausteins TE_AxisInitAllAxisType01

das Achs-Interface initialisiert. Bei Erreichen des Betriebsmodus (Phase 4) wird die Initialisierung erneut ausgeführt, um Änderungen in der Parametrierung zu erkennen

- **Zyklischer Aufruf des Achs-Interface**:

  Nach Abschluss der Initialisierung wird zyklisch das Achs-Interface aufgerufen. Dazu wird der Funktionsbaustein "TE_AxisInterface" aufgerufen. Dieser ist vom MB_AxisInterfaceType01 abgeleitet, um Anwendererweiterungen zu ermöglichen

- **Anwendererweiterungen des Achs-Interface**:

  Der Funktionsbaustein "TE_AxisInterface" enthält beispielhaft ausprogrammierte Anwendererweiterungen. Der Anwender kann den Achs-Interface Strukturen weitere Elemente hinzufügen um seine Anwendung optimal abzudecken

---

☞ Das Programm TE_AxisInterfaceType01Prog ist zum Integrieren des Achs-Interface in ein bestehendes Programm nützlich.

Für neue Projekte wird dringend empfohlen, die Importfunktion des GAT-Wizards für das Anlegen des Achs-Interfaces zu verwenden.

Um das Achs-Interface in ein bestehendes Projekt zu integrieren kann wie folgt vorgegangen werden:

1. Starten Sie den GAT-Wizard über das Symbol links oben in der Menüleiste.
2. Wählen Sie im erscheinenden "Template-Code importieren"-Dialog den Punkt "AxisInterface" aus.

   Das Achs-Interface-Beispielprojekt wird nun angelegt.

3. Mit den Strukturen "arAxisCtrl_gb", "arAxisStatus_gb" und "AxisData" können jetzt die Achsen angesteuert werden.
4. Mit den Strukturen "ImcCtrl" und "ImcStatus" kann die Steuerung bedient werden.

Im Programm "PlcProg" wird der Zugriff auf das Achs-Interface gezeigt. Dieser Code muss für das eigene Projekt entsprechend angepasst werden.

☞ Zur komfortablen Achskommandierung steht Ihnen die dialogbasierte Achskommandierung zur Verfügung. Mehr Details zu dem Thema finden Sie in Kap. 6.4.1 "Dialogbasierte Achskommandierung" auf Seite 73

6.6.3 Achs-Interface Visualisierungen

Überblick

Zum Achs- und Imc-Interface werden Visualisierungsmasken mitgeliefert, um ein vorgefertigtes und einfaches Interface zum Einstellen und Ansteuern der Systemachsen und der Steuerung zur Verfügung zu stellen.

Folgende Visualisierungen sind im GAT bzw. GAT compact enthalten:

Visualisierung	Beschreibung
System_Overview_All_Axis	Gesamtüberblick über alle definierten Achsen, einschließlich einfacher Diagnosen, Status und Tipp-Bedienelemente
Axis_Overview	Zeigt aktuelles Kommando und aktuelle Werte für Position und Geschwindigkeit, zusammen mit Navigation zur Positions-, Geschwindigkeits- und Synchronisations-Anzeige für die aktuelle Achse. Referenzieren und Abschalten der Achsen sind ebenfalls möglich
Position_mode	Überwachen des Positionierbetriebs
Velocity_mode	Überwachen des Geschwindigkeitsbetriebs
Synchronization_mode	Überwachen des Synchronisationsbetriebs
Profile_mode	Überwachen des Synchronisationsbetriebs mit Profilen
ImcInterface_Status_Control	IndraMotion MLC-Steuerung und Status-Interface der Betriebsarten, Quittungen und Systemdiagnose
PLC_load	Zeigt die minimale, aktuelle und maximale SPS-Task Zykluszeit für AxisInterface-Type01Prog an

Abb.6-41: Beispielprojekt Visualisierungen

Die folgenden globalen Variablen werden zum Steuern und für den Zugriff auf Systeminformationen innerhalb der Visualisierungen benutzt:

- arAxisCtrl_gb[ ]
- arAxisStatus_gb[ ]
- AxisData[ ] (MLC Standard)
- VisuAxisNo

Systemübersicht-Visualisierung

Die System_Overview_All_Axis Visualisierung erlaubt es dem Anwender jede, im Projekt konfigurierte Achse, schnell anzukoppeln. Die Default-Achsen, die angezeigt werden (MyVirtualAxis1 und MyVirtualAxis2), sind Teil des GAT ^{compact}-Beispielprojektes. Zusätzliche Achsen können im Offline-Betrieb zur Anzeige hinzugefügt werden.

Diese Anzeige liefert einen Gesamt-Systemstatus und ermöglicht das Löschen von Fehlern und das Setzen von Not-Halt. Einzelne Achs-Bedienelemente liefern den Achsnamen, Diagnosen, aktuelle Position und Geschwindigkeit zusammen mit den Betriebsarten-Statusanzeigen. Der SetupMode-Teil erlaubt es dem Anwender, einzelne Achsen zu joggen und zu positionieren, sowie virtuelle Achsen zu starten und zu stoppen.

ML_TechInterface.library

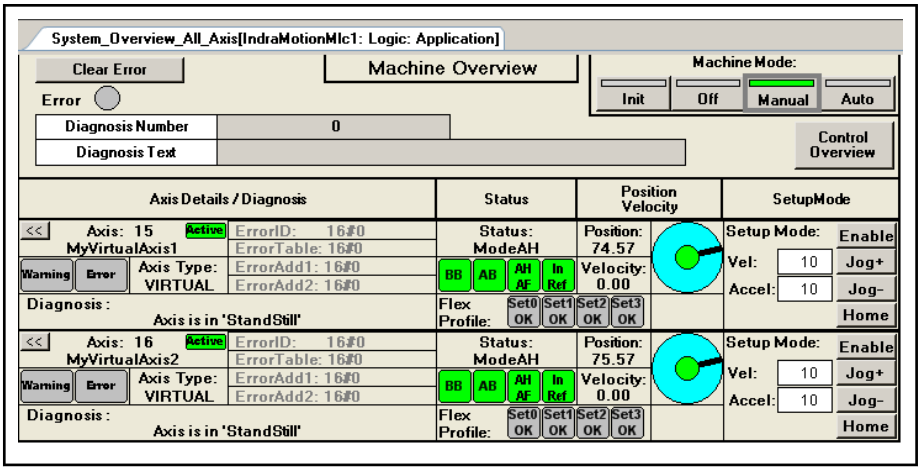


Abb.6-42: System_Overview_All_Axis Anzeige

Hinzufügen einer Achse zur Systemübersicht

Das Hinzufügen einer Achse zur Systemübersicht-Anzeige erfolgt wie das Hinzufügen einer neuen Visualisierung, durch Anwählen des entsprechenden Elements, der Festlegung des Achsnamens und der Achsnummer. Nachfolgend aufgeführte Schritte stellen die Vorgehensweise kurz dar:

1. Öffnen Sie das entsprechende IndraWorks-Projekt und wählen Sie den Reiter **Logic->Application->Visualizations->SystemOverview** an.
2. Mit IndraLogic im Offline-Betrieb Doppelklick auf die System_Overview_All_Axis Visualisierung.
3. Wählen Sie im Fenster Werkzeuge die Schaltfläche **Frame** an  und erzeugen Sie einen Grundriss unterhalb der letzten Achstabellenzeile, der der Zeilenhöhe und Zeilenbreite entspricht.

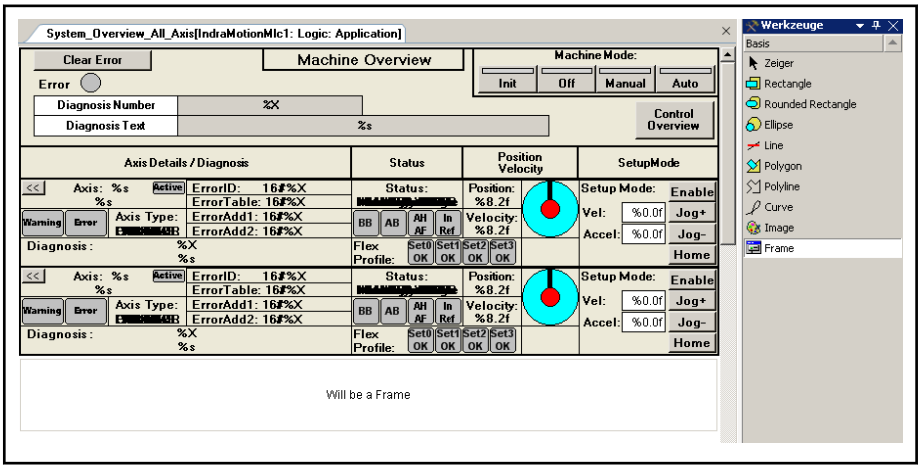


Abb.6-43: Frame

4. Mit rechter Maustaste auf den Frame klicken und "Frameauswahl" aktivieren. Wählen Sie das **System_Overview**-Element aus dem Visualisierungsauswahlfenster aus.

   Eine komplett neue Systemübersicht Achszeile erscheint.

5. Nach einem Klick in der neuen Visualisierung tragen Sie im "Eigenschaften"-Fenster als Wert für**m_Input_AxisName** den Namen der definierten Achse ein.

6. Das neue Achs-Interface für die Systemübersicht kann nun in der Grösse angepasst und positioniert werden.
7. Übersetzen Sie das IndraLogic-Projekt neu und gehen Sie Online.

Obige Schritte müssen für alle, zusätzlich zum Projekt hinzugefügten Achsen wiederholt werden.

Systemübersicht Navigation

Eine Einzelachs-Übersichtanzeige ist durch Klicken auf die **Achsname**-Schaltfläche, die sich unter der Details-Spalte in der Achstabelle befindet, erreichbar. Aus der Achsübersichtanzeige ist die Navigation zur Positions-, Geschwindigkeits- und Synchronisations-Betriebsartenanzeige möglich.

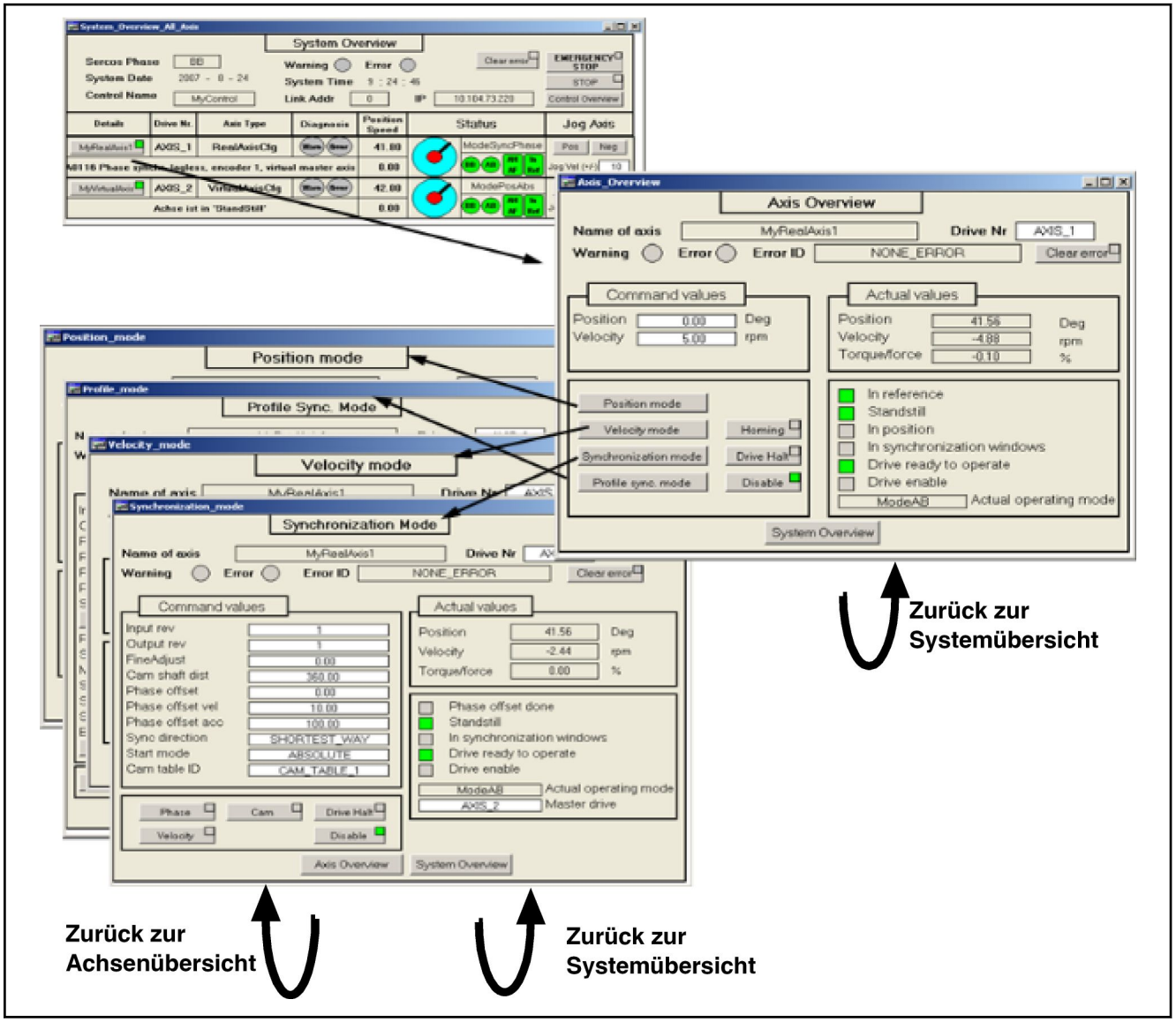


Abb.6-44: Systemübersicht Navigation

Achsübersicht-Visualisierung

Die Achsübersicht-Visualisierung bietet ein einfaches Interface für die aktuellen Soll- und Istwerte einer Achse. Sollwerte für den Positionierbetrieb und Geschwindigkeitsbetrieb werden bereitgestellt. Der Anwender kann in einem Projekt durch Klicken auf das **Drive Nr**-Feld und Eingabe der Nummer der entsprechenden Achse zwischen den Achsen hin und her wechseln.

ML_TechInterface.library

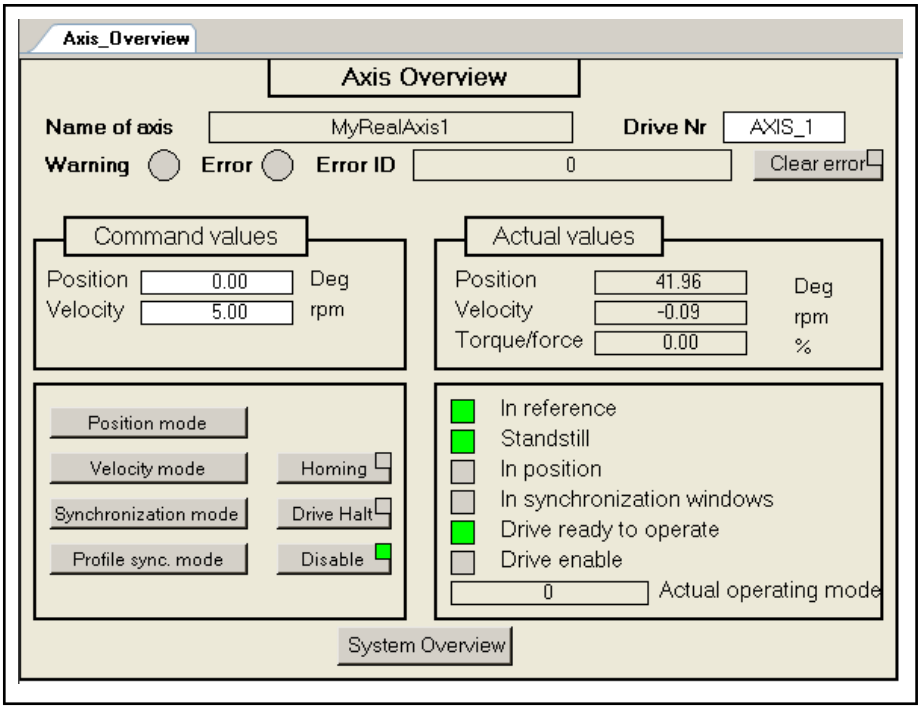


Abb.6-45: Achsübersicht-Visualisierung

Die Schaltflächen für die Betriebsarten Positionieren, Geschwindigkeitsregelung, Synchronisation und Synchronisation mit Profilen erlauben dem Anwender in die jeweilige Betriebsart zu schalten, in der spezielle Werte für die jeweilige Betriebsart eingegeben werden können. Die aktuelle Achse kann referenziert werden, es kann nach Antrieb Halt unter Aufrechterhaltung des Drehmoments (AH) geschaltet oder die Leistung abgeschaltet werden (Ab).

Die folgende Tabelle enthält die (allgemeinen) Achs-Interface-Strukturelemente, die als Teil der Visualisierung benutzt werden:

Visualisierungselement	Achs-Interface Strukturelement
Antriebs-Nr	VisuAxisNo
Kommando: Position	arAxisCtrl_gb[ ].PosMode.Position
Kommando: Geschwindigkeit	arAxisCtrl_gb[ ].PosMode.Velocity
Referenzieren-Button	arAxisCtrl_gb[ ].Admin._OpMode: = ModeHoming
Antrieb Halt Button	arAxisCtrl_gb[ ].Admin._OpMode: = ModeAh
Abschalt-Button	arAxisCtrl_gb[ ].Admin._OpMode: = ModeAb
Aktuell: Position	AxisData[ ].rActualPosition_i
Aktuell: Geschwindigkeit	AxisData[ ].rActualVelocity_i
Aktuell: Drehmoment/Kraft	AxisData[ ].rActualTorqueForce_i
LED: Referenziert	AxisData[ ].Axis_Homed
LED: im Stillstand	AxisData[ ].Axis_Standstill
LED: In-Position	AxisData[ ].Axis_InPosition
LED: In-Synchronisation Fenster	AxisData[ ].Axis_InSynchron
LED: Antrieb betriebsbereit	AxisData[ ].Axis_InAb

Visualisierungselement	Achs-Interface Strukturelement
LED: Antriebsfreigabe	AxisData[ ].Axis_Power
Aktuelle Betriebsart	arAxisStatus_gb[ ].Admin._OpModeAck

Abb.6-46: Achsübersicht Strukturelemente

Die arAxisCtrl_gb[ ] and arAxisStatus_gb[ ] Strukturen sind Teil der ML_TechInterface Bibliothek. Die AxisData [ ] Struktur ist Teil der ML_Base Bibliothek.

Positionierbetrieb-Visualisierung

Die Positionierbetrieb-Visualisierung wird benutzt, um Sollwerte für

- Position
- Weg
- Geschwindigkeit
- Beschleunigung
- Verzögerung

einzugeben.

Der Anwender kann die zu benutzende Positionierart auswählen (absolut, relativ oder additiv).

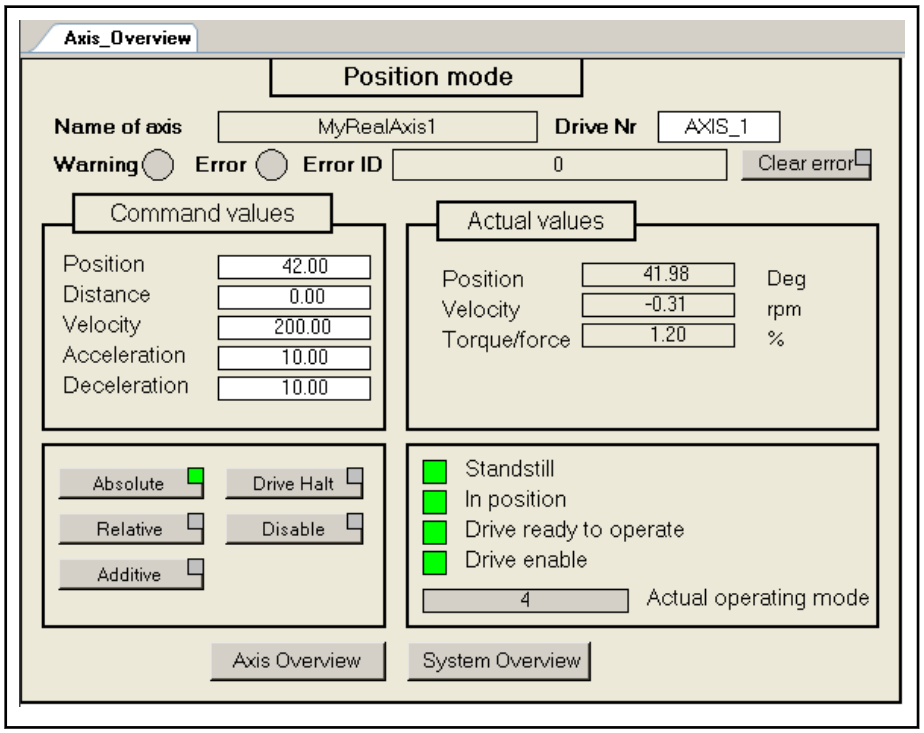


Abb.6-47: Positionierbetrieb-Visualisierung

Für nähere Hinweise siehe folgende Positionierarten:

Für absolut, siehe Kap. 6.4.7 "Absolutes Positionieren" auf Seite 79.

Für relativ, siehe Kap. 6.4.8 "Relatives Positionieren" auf Seite 80.

Für additiv, siehe Kap. 6.4.9 "Additives Positionieren" auf Seite 81.

Die folgende Tabelle enthält die (allgemeinen) Achs-Interface-Strukturelemente, die als Teil der Visualisierung benutzt werden:

ML_TechInterface.library

Visualisierungselement	Achs-Interface Strukturelement
Antriebs-Nr	VisuAxisNo
Kommando: Position	arAxisCtrl_gb[ ].PosMode.Position
Kommando: Weg	arAxisCtrl_gb[ ].PosMode.Distance
Kommando: Geschwindigkeit	arAxisCtrl_gb[ ].PosMode.Velocity
Kommando: Beschleunigung	arAxisCtrl_gb[ ].PosMode.Acceleration
Kommando: Verzögerung	arAxisCtrl_gb[ ].PosMode.Deceleration
Antrieb Halt Button	arAxisCtrl_gb[ ].Admin._OpMode: = ModeAh
Abschalt-Button	arAxisCtrl_gb[ ].Admin._OpMode: = ModeAb
Aktuell: Position	AxisData[ ].rActualPosition_i
Aktuell: Geschwindigkeit	AxisData[ ].rActualVelocity_i
Aktuell: Drehmoment/Kraft	AxisData[ ].rActualTorqueForce_i
LED: im Stillstand	AxisData[ ].Axis_Standstill
LED: In-Position	AxisData[ ].Axis_InPosition
LED: Antrieb betriebsbereit	AxisData[ ].Axis_InAb
LED: Antriebsfreigabe	AxisData[ ].Axis_Power
Aktuelle Betriebsart	arAxisStatus_gb[ ].Admin._OpModeAck

Abb.6-48: Positionierbetrieb Strukturelemente

Geschwindigkeitsbetrieb-Visualisierung

Die Geschwindigkeitsbetrieb-Visualisierung wird benutzt, um Sollwerte für

- Geschwindigkeit
- Beschleunigung
- Verzögerung

einzugeben.

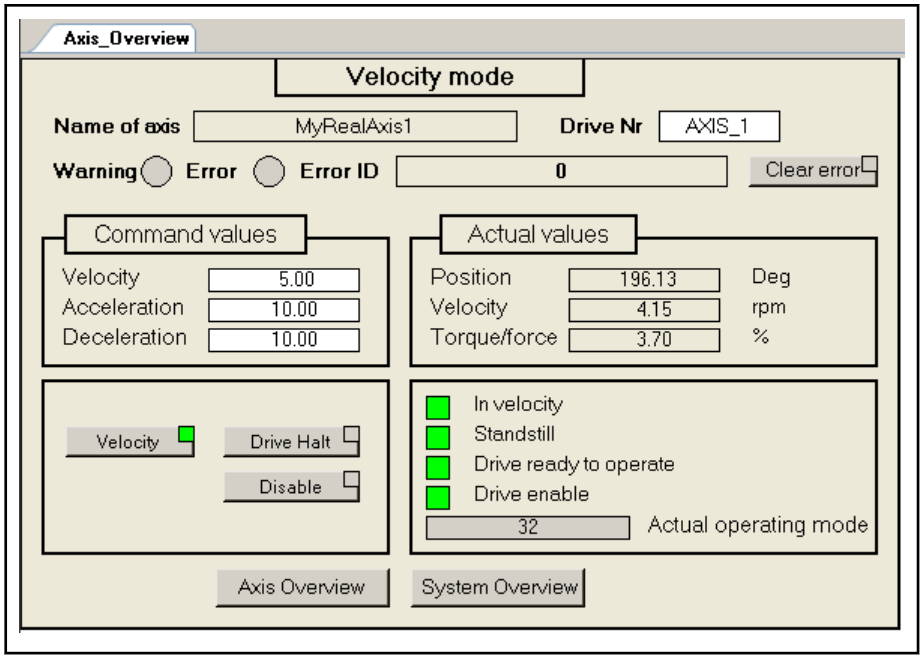


Abb.6-49: Geschwindigkeitsbetrieb-Visualisierung

Die folgende Tabelle enthält die (allgemeinen) Achs-Interface-Strukturelemente, die als Teil der Visualisierung benutzt werden:

Visualisierungselement	Achs-Interface Strukturelement
Antriebs-Nr	VisuAxisNo
Kommando: Geschwindigkeit	arAxisCtrl_gb[ ].VelMode.Velocity
Kommando: Beschleunigung	arAxisCtrl_gb[ ].VelMode.Acceleration
Kommando: Verzögerung	arAxisCtrl_gb[ ].VelMode.Deceleration
Antrieb Halt Button	arAxisCtrl_gb[ ].Admin._OpMode: = ModeAh
Abschalt-Button	arAxisCtrl_gb[ ].Admin._OpMode: = ModeAb
Aktuell: Position	AxisData[ ].rActualPosition_i
Aktuell: Geschwindigkeit	AxisData[ ].rActualVelocity_i
Aktuell: Drehmoment/Kraft	AxisData[ ].rActualTorqueForce_i
LED: In Geschwindigkeit	AxisData[ ].Axis_InVelocity
LED: im Stillstand	AxisData[ ].Axis_Standstill
LED: In-Position	AxisData[ ].Axis_InPosition
LED: Antrieb betriebsbereit	AxisData[ ].Axis_InAb
LED: Antriebsfreigabe	AxisData[ ].Axis_Power
Aktuelle Betriebsart	arAxisStatus_gb[ ].Admin._OpModeAck

Abb.6-50: Geschwindigkeitsbetrieb Strukturelemente

Synchronisationsbetrieb-Visualisierung

Die Synchronisationsbetrieb-Visualisierung wird benutzt, um die Funktionalität des elektronischen Getriebe des Antriebs zu steuern. Es wird vorausgesetzt, dass Feinabgleich- und Phasenoffset-Elemente benutzt werden. Das

ML_TechInterface.library

Drive Nr-Feld definiert die Folgeachse und das **Master drive**-Feld die Leitachse.

Der Anwender kann die zu benutzende Art des Synchronisationsbetriebs auswählen (Winkel-, Geschwindigkeits-Synchronisation, Kurvenscheibe oder Bewegungsprofil).

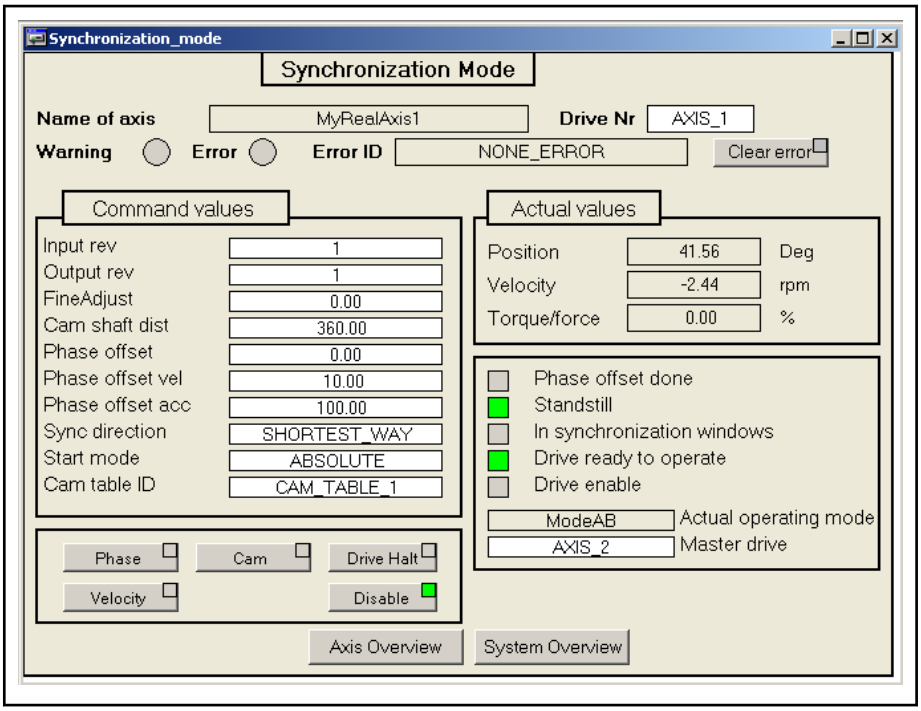


Abb.6-51: Synchronisationsbetrieb-Visualisierung

Für nähere Hinweise siehe folgende Synchronisationsbetriebsarten-Typen:

Für Winkelsynchronisation, siehe Kap. 6.4.11 "Betriebsart Winkelsynchronisation" auf Seite 82.

Für Geschwindigkeitssynchronisation, siehe Kap. 6.4.12 "Betriebsart Geschwindigkeitssynchronisation" auf Seite 83.

Für Kurvenscheibe, siehe Kap. 6.4.13 "Betriebsart Elektronische Kurvenscheibe" auf Seite 83.

Die folgende Tabelle enthält die (allgemeinen) Achs-Interface-Strukturelemente, die als Teil der Visualisierung benutzt werden:

Visualisierungselement	Achs-Interface Strukturelement
Antriebs-Nr	VisuAxisNo
Kommando: Eingangsumdrehungen	arAxisCtrl_gb[ ].SyncMode.InputRevolution
Kommando: Ausgangsumdrehungen	arAxisCtrl_gb[ ].SyncMode.InputRevolution
Kommando: Feinabgleich	arAxisCtrl_gb[ ].SyncMode.Fineadjust
Kommando: Nockenwellenhub	arAxisCtrl_gb[ ].SyncMode.CamShaftDistance
Kommando: Phasenoffset	arAxisCtrl_gb[ ].SyncMode.PhaseOffset
Kommando: Phasenoffset Geschwindigkeit	arAxisCtrl_gb[ ].SyncMode.PhaseOffsetVel
Kommando: Phasenoffset Beschleunigung	arAxisCtrl_gb[ ].SyncMode.PhaseOffsetAcc
Kommando: Synchronisationsrichtung	arAxisCtrl_gb[ ].SyncMode.SyncDirection

Visualisierungselement	Achs-Interface Strukturelement
Kommando: Synchronisations-Startbetrieb	arAxisCtrl_gb[ ].SyncMode.StartMode
Kommando: Kurvenscheiben-ID	arAxisCtrl_gb[ ].SyncMode.CamTableID
Button Phase	arAxisCtrl_gb[ ].Admin._OpMode: = ModeSyncPhase
Button Geschwindigkeit	arAxisCtrl_gb[ ].Admin._OpMode: = ModeSyncVelocity
Button Kurvenscheibe	arAxisCtrl_gb[ ].Admin._OpMode: = ModeSyncCam
Antrieb Halt Button	arAxisCtrl_gb[ ].Admin._OpMode: = ModeAh
Abschalt-Button	arAxisCtrl_gb[ ].Admin._OpMode: = ModeAb
Aktuell: Position	AxisData[ ].rActualPosition_i
Aktuell: Geschwindigkeit	AxisData[ ].rActualVelocity_i
Aktuell: Drehmoment/Kraft	AxisData[ ].rActualTorqueForce_i
LED: Phasenoffset ausgeführt	arAxisStatus_gb[ ].SyncMode.PhasingSlaveDone
LED: im Stillstand	AxisData[ ].Axis_Standstill
LED: In-Synchronisation Fenster	AxisData[ ].Axis_InSynchron
LED: Antrieb betriebsbereit	AxisData[ ].Axis_InAb
LED: Antriebsfreigabe	AxisData[ ].Axis_Power
Aktuelle Betriebsart	arAxisStatus_gb[ ].Admin._OpModeAck
Antrieb der Leitachse	arAxisCtrl_gb[ ].SyncMode.Master.AxisNo

Abb.6-52: Synchronisationsbetrieb Strukturelemente

Visualisierung – Synchronisationsbetrieb mit Profilen

Die Visualisierung "Synchronisationsbetrieb mit Profilen" wird benutzt, um die Funktionalität des elektronischen Getriebe des Antriebs mit Profilen zu steuern. Es wird vorausgesetzt, dass Feinabgleich- und Phasenoffset-Elemente benutzt werden. Das **Drive Nr**-Feld definiert die Folgeachse und das **Master drive**-Feld die Leitachse.

Der Anwender kann die zu benutzende Art des Synchronisationsbetriebs mit Profilen auswählen (FlexProfile oder MotionProfile).

ML_TechInterface.library

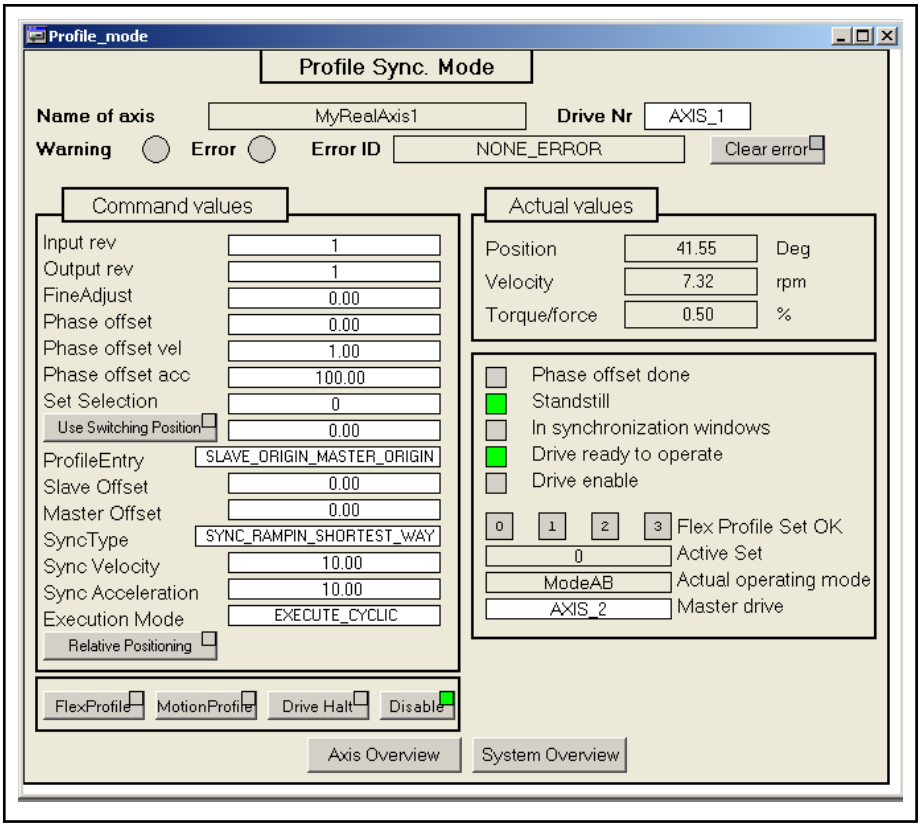


Abb.6-53: Visualisierung – Synchronisationsbetrieb mit Profilen

Für nähere Hinweise siehe folgende Synchronisationsbetriebsarten-Typen:

Für MotionProfile, siehe Kap. 6.4.14 "Betriebsart Elektronisches Bewegungsprofil" auf Seite 84.

Für FlexProfile, siehe Kap. 6.4.15 "Betriebsart Elektronisches FlexProfile" auf Seite 85.

Die folgende Tabelle enthält die (allgemeinen) Achs-Interface-Strukturelemente, die als Teil der Visualisierung benutzt werden:

Visualisierungselement	Achs-Interface Strukturelement
Antriebs-Nr	VisuAxisNo
Kommando: Eingangsumdrehungen	arAxisCtrl_gb[ ].SyncMode.InputRevolution
Kommando: Ausgangsumdrehungen	arAxisCtrl_gb[ ].SyncMode.InputRevolution
Kommando: Feinabgleich	arAxisCtrl_gb[ ].SyncMode.Fineadjust
Kommando: Phasenoffset	arAxisCtrl_gb[ ].SyncMode.PhaseOffset
Kommando: Phasenoffset Geschwindigkeit	arAxisCtrl_gb[ ].SyncMode.PhaseOffsetVel
Kommando: Phasenoffset Beschleunigung	arAxisCtrl_gb[ ].SyncMode.PhaseOffsetAcc
Kommando: Profilsatz-Auswahl	arAxisCtrl_gb[ ].SyncMode.Profile.SetSelection
Button Relative Positionierung	arAxisCtrl_gb[ ].SyncMode.Profile.RelativePositioning
Button MotionProfile	arAxisCtrl_gb[ ].Admin._OpMode: = ModeSyncProfile
Button FlexProfile	arAxisCtrl_gb[ ].Admin._OpMode: = ModeFlexProfile
Antrieb Halt Button	arAxisCtrl_gb[ ].Admin._OpMode: = ModeAh

Visualisierungselement	Achs-Interface Strukturelement
Abschalt-Button	arAxisCtrl_gb[ ].Admin._OpMode: = ModeAb
Aktuell: Position	AxisData[ ].rActualPosition_i
Aktuell: Geschwindigkeit	AxisData[ ].rActualVelocity_i
Aktuell: Drehmoment/Kraft	AxisData[ ].rActualTorqueForce_i
LED: Phasenoffset ausgeführt	arAxisStatus_gb[ ].SyncMode.PhasingSlaveDone
LED: im Stillstand	AxisData[ ].Axis_Standstill
LED: In-Synchronisation Fenster	AxisData[ ].Axis_InSynchron
LED: Antrieb betriebsbereit	AxisData[ ].Axis_InAb
LED: Antriebsfreigabe	AxisData[ ].Axis_Power
Aktuelle Betriebsart	arAxisStatus_gb[ ].Admin._OpModeAck
Antrieb der Leitachse	arAxisCtrl_gb[ ].SyncMode.Master.AxisNo

Abb.6-54: Synchronisationsbetrieb mit Profilen, Strukturelemente

Imc-Interface-Steuerung und Status-Visualisierung

Die ImcInterface_Status_Control - Visualisierung ermöglicht den Zugriff auf Steuerungsfunktionalitäten, wie z.B. Umschalten zwischen Parameter- und Betriebs-Modus, Fehler Löschen und Neustart der Phasenumschaltung. Die Visualisierung liefert auch den Steuerungsstatus der verschiedenen sercos Betriebsarten und Fehlerdiagnosen.

ML_TechInterface.library

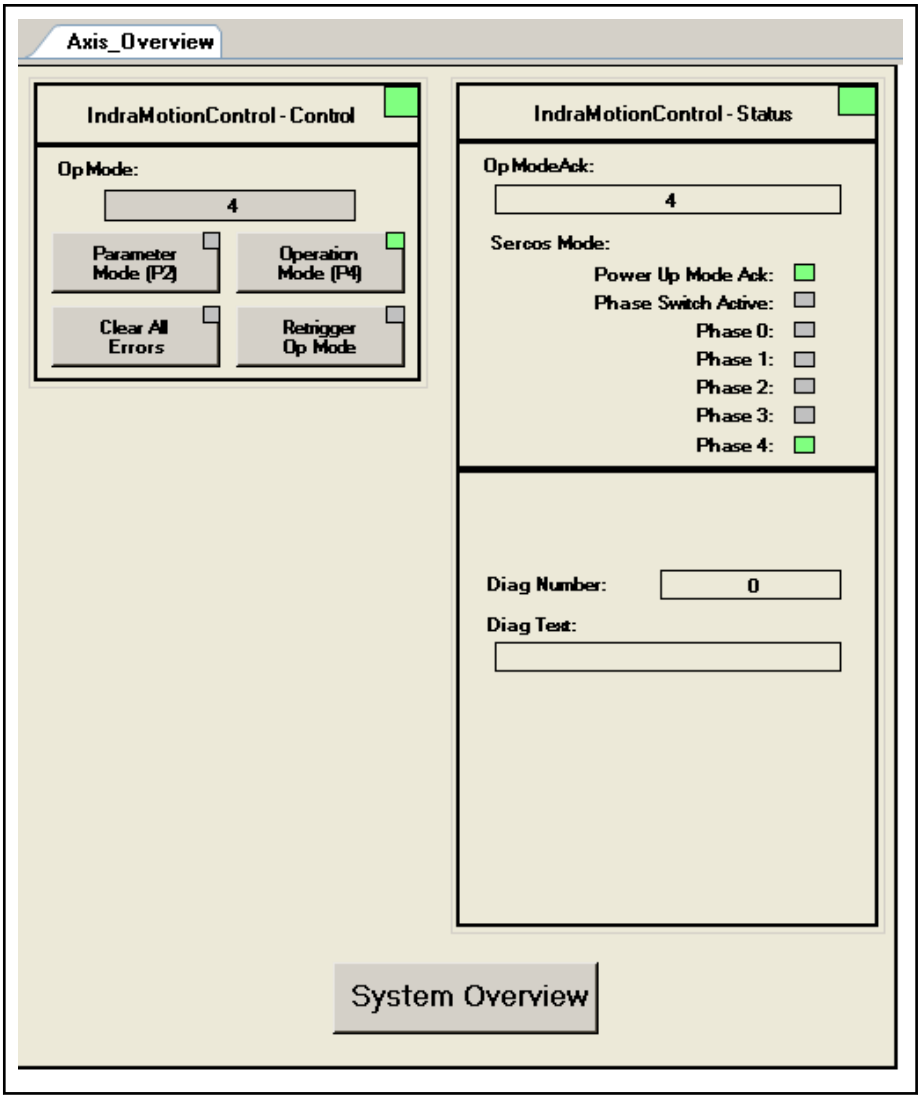


Abb.6-55: ImcInterface-Status und Steuerungs-Visualisierung

Die folgende Tabelle enthält die (allgemeinen) Achs-Interface-Strukturelemente, die als Teil der Visualisierung benutzt werden:

Visualisierungselement	Achs-Interface Strukturelement
IndraMotionControl - Steuerung	
Phase	ImcCtrl.Admin._OpMode
Parametrier-Modus (P2)	ImcCtrl.Admin._OpMode: = IMC_MODE_CTRL_P2
Betriebs-Modus (P4)	ImcCtrl.Admin._OpMode: = IMC_MODE_CTRL_BB
Alle Fehler löschen	ImcCtrl.Admin.ClearError
Neustart der Phasenumschaltung	ImcCtrl.Admin.RetriggerOpMode
IndraMotionControl - Status	
Phasen-Quittung	ImcStatus.Admin._OpModeAck
LED: Einschalt-Modus-Quittung	ImcStatus.Admin.PwrUpModeAck
LED: Phasenumschaltung aktiv	ImcStatus.Admin.PhaseSwitchActive

Visualisierungselement	Achs-Interface Strukturelement
LED: Phase 0	ImcStatus.Admin.ModeStatus_P0
LED: Phase 1	ImcStatus.Admin.ModeStatus_P1
LED: Phase 2	ImcStatus.Admin.ModeStatus_P2
LED: Phase 3	ImcStatus.Admin.ModeStatus_P3
LED: Phase 4	ImcStatus.Admin.ModeStatus_BB
LED: Lösche Fehler Quittung	ImcStatus.Diag.ClearErrorAck
LED: Fehler	ImcStatus.Diag.Error
LED: Warnung	ImcStatus.Diag.Warning
ErrorID	ImcStatus.Diag.ErrorID
Fehleridentifizierung: Tabelle	ImcStatus.Diag.ErrorIdent.Table
Fehleridentifizierung: Additional 1	ImcStatus.Diag.ErrorIdent.Additional1
Fehleridentifizierung: Additional2	ImcStatus.Diag.ErrorIdent.Additional2

Abb.6-56: ImcInterface-Strukturelemente

SPS-Auslastungs-Visualisierung

Die PLC_Load-Visualisierung wird benutzt, um die SPS-Auslastung zu überwachen. Es werden die minimale, aktuelle und maximale SPS-Zykluszeit für die zyklischen Aufrufe des Achs-Interface Funktionsbaustein im TE_AxisInterfaceType01Prog angezeigt.

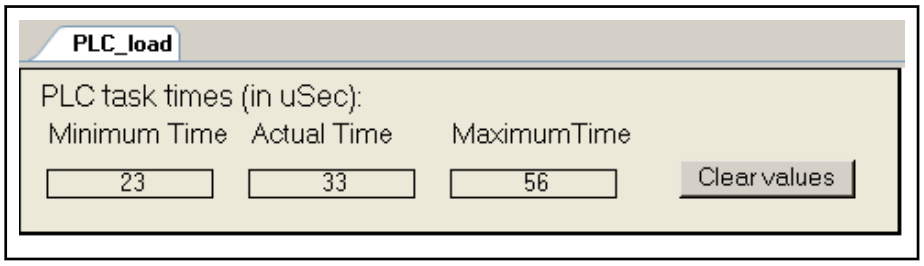


Abb.6-57: SPS-Auslastungs-Visualisierung

6.7 Achs-Interface Anwender-Erweiterung

6.7.1 Überblick

Die **arAxisCtrl_gb[ ]** und **arAxisStatus_gb[ ]** Strukturen können durch den Anwender erweitert werden, um das Achs-Interface an spezielle Applikationen anzupassen.

☞ Die **arAxisCtrl_gb[ ]**- und **arAxisStatus_gb[ ]**-Strukturen des Anwender-Interface sind

MB_Axis_Control_Type01 und MB_Axis_Status_Type01 Datenstrukturen (siehe Kap. 6.8 "Achs-Interface-Strukturen" auf Seite 109).

Da die Achs-Interface-Datenstruktur in der ML_TechInterface Bibliothek integriert und daher für den Anwender nicht zugänglich ist, ist es erforderlich eine neue Datenstruktur innerhalb des aktuellen Projektes entsprechend der bereits vorhandenen anzulegen, um zusätzliche Funktionalitäten hinzufügen zu können.

ML_TechInterface.library

Dieser Abschnitt zeigt, wie das Achs-Interface durch Hinzufügen der **MB_Phasing**-Funktionalität erweitert wird. Der MB_Phasing-Funktionsbaustein ermöglicht es, auf eine Leitachse einen Offset aufzuaddieren. Die Leitachse ist als Eingang mit dem **MC_CamIn** oder **MC_GearInPos** Synchronisations-Funktionsbaustein verbunden.

Die **MB_PhasingSlave**-Funktionalität ist mittels der

arAxisCtrl_gb[ ].SyncMode.PhaseOffset, (.PhaseOffsetVel), (.PhaseOffsetAcc) Elemente und dem

arAxisStatus_gb[ ].SyncMode.PhasingSlaveDone Ausgang in das Achs-Interface integriert.

Die folgenden neuen Ein- und Ausgänge werden definiert:

- arAxisCtrl_gb[ ].SyncMode.PhaseShift
- arAxisCtrl_gb[ ].SyncMode.PhaseShiftVel
- arAxisCtrl_gb[ ].SyncMode.PhaseShiftAcc
- arAxisStatus_gb[ ].SyncMode.PhasingMasterDone

☞ Der "arAxisCtrl_gb[ ].SyncMode.PhaseShift"-Eingang wird zyklisch gescannt und bei Änderung sofort aktiviert.

Im GAT bzw. GAT compact sind folgende anwenderspezifische Erweiterungen implementiert:

- Jog-Funktionalität und Homing als "SetupMode"
- Parameter A-0-2730 "Winkelverschiebung Tabellenanfang" als "SyncMode.AngleOffset"
- Parameter A-0-2790 "Aufsynchronisiergeschwindigkeit" als "SyncMode.SyncVelocity"
- Parameter A-0-2791 "Aufsynchronisierbeschleunigung" als "SyncMode.SyncAcceleration"
- Funktionsbaustein MB_Phasing() als SyncMode.PhaseShift, -.PhaseShiftVel und - .PhaseShiftAcc

Im Folgenden wird davon ausgegangen, dass die im GAT bzw. GAT compact bereits vorbereitete Struktur verwendet wird. Es wird nur beschrieben, welche Änderungen in den dort vorgegebenen POUs notwendig sind.

Hinweise zur Implementation von Anwendererweiterungen

Die Anwendererweiterungen wurden mit Hilfe der objektorientierten Erweiterungen von IndraLogic 2G implementiert. Dabei sind einige Besonderheiten zu beachten:

- Der Funktionsbaustein (FB) TE_AxisInterface ist vom Basis-FB MB_AxisInterfaceType01 abgeleitet. Über das Schlüsselwort "SUPER" kann der Basis-FB bzw. Methoden/Aktionen des Basis-FB aufgerufen werden. Zum Beispiel wird an diversen Stellen über SUPER^.mSetError(...); die Methode mSetError des FB MB_AxisInterfaceType01 aufgerufen um Fehler in das Diagnosesystem einzutragen
- Innerhalb des FB TE_AxisInterface kann auf die Daten von arAxisCtrl_gb über die Eingänge AdminCtrlExt, PosModeCtrlExt usw. bzw. auf arAxisStatus_gb über AdminStatusExt, DiagStatusExt usw. zugegriffen werden.

  Die Eingänge AdminCtrl (ohne Ext) usw. gehören zum Basis-FB und sollten nicht genutzt werden

- Die Eingänge des FB TE_AxisInterface sind als "REFERENCE TO" definiert. In der Methode mInitExtension werden die Referenzen einmalig initialisiert und müssen dann beim zyklischen Aufruf des FB nicht mehr übergeben werden
- Wenn die Basisstrukturen TE_AXIS_CONTROL_TYPE01 und TE_AXIS_STATUS_TYPE01 mit zusätzlichen Unterstrukturen erweitert werden sollen, sind folgende zusätzliche Schritte zu der weiter unten beschriebenen Vorgehensweise notwendig (siehe SetupMode und SetupModeAck als Beispiele):
  - Neue Elemente in TE_AXIS_CONTROL_TYPE01 bzw. TE_AXIS_STATUS_TYPE01 eintragen
  - Am FB TE_AxisInterface die zusätzlichen Eingänge als REFERENCE TO hinzufügen
  - In der Methode mInitExtension des FB TE_AxisInterface die Referenzen initialisieren

6.7.2 Erweitern der arAxisCtrl_gb[ ]-Struktur

Um die zusätzliche Funktionalität der MB_Axis_Control_Type01 Struktur hinzuzufügen, muss der Anwender eine neue Struktur anlegen, die Unterstrukturen von den bereits existierenden Struktur ableiten und dann die neuen Elemente hinzufügen. Im GAT/GAT compact ist eine Struktur TE_AXIS_CONTROL_TYPE01 und die Unterstrukturen TE_AXIS_ADMINISTRATION usw. bereits vorbereitet.

Nehmen Sie die folgenden Schritte vor, um die Funktionalität der TE_AXIS_SYNCHRONISATION Struktur zu erweitern:

1. Das entsprechende Projekt in IndraWorks öffnen.
2. Öffnen Sie die Struktur TE_AXIS_SYNCHRONISATION.

ML_TechInterface.library

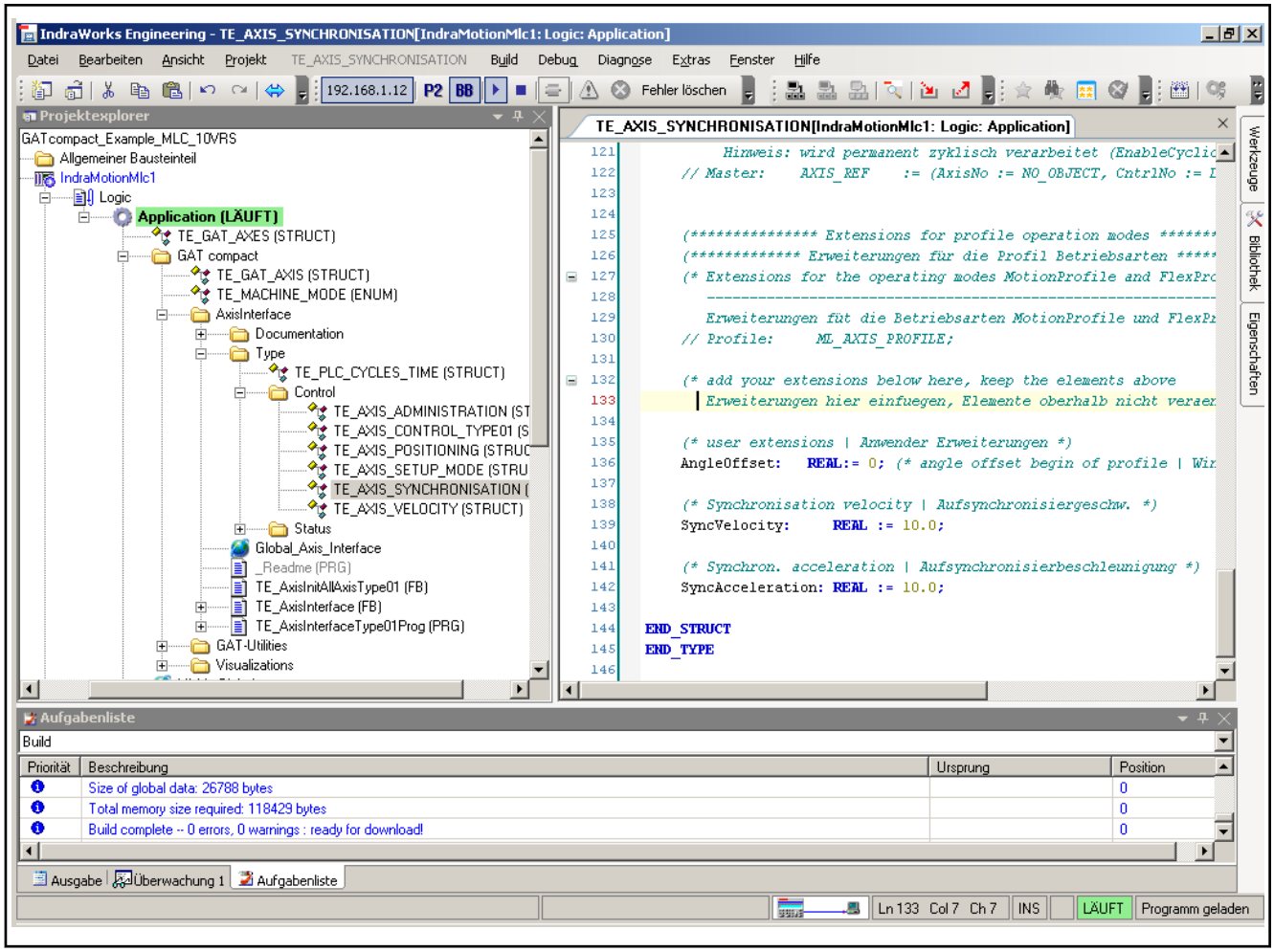


Abb.6-58: Öffnen der Struktur TE_AXIS_SYNCHRONISATION

3. Fügen Sie die folgenden drei neuen Element-Zuweisungen
   - **PhaseShift**
   - **PhaseShiftVel**
   - **PhaseShiftAcc**

   in "TE_AXIS_SYNCHRONISATION" unterhalb des Kommentars als REAL-Typ hinzu.

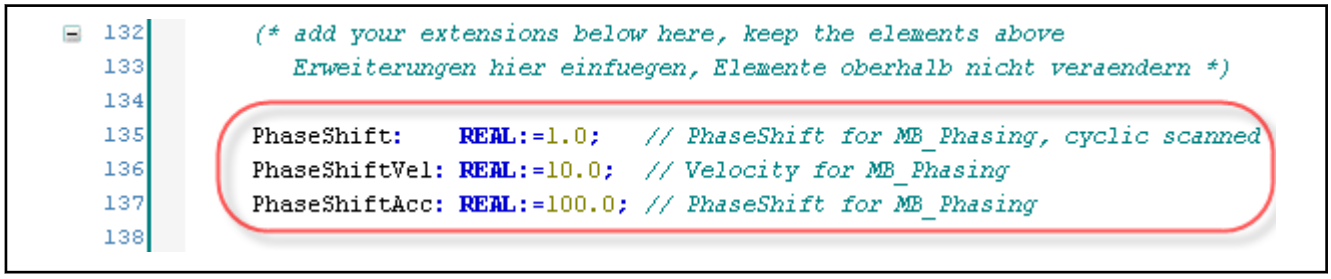


Abb.6-59: Neu zu TE_AXIS_SYNCHRONISATION hinzugefügte Elemente

6.7.3 Erweitern der arAxisStatus_gb[ ] Struktur

Erweitern Sie die **arAxisStatus_gb[ ]**-Struktur entsprechend den Schritten in "Erweitern der arAxisCtrl_gb[ ] Struktur". Die folgenden Schritte stellen die Vorgehensweise kurz dar:

1. Öffnen Sie das Objekt TE_AXIS_SYNC_STATUS.

2. Fügen Sie die folgenden neuen Element-Zuweisungen **PhasingMaster-Done** in TE_AXIS_SYNC_STATUS unterhalb des Kommentars als BO-OL-Typ hinzu.

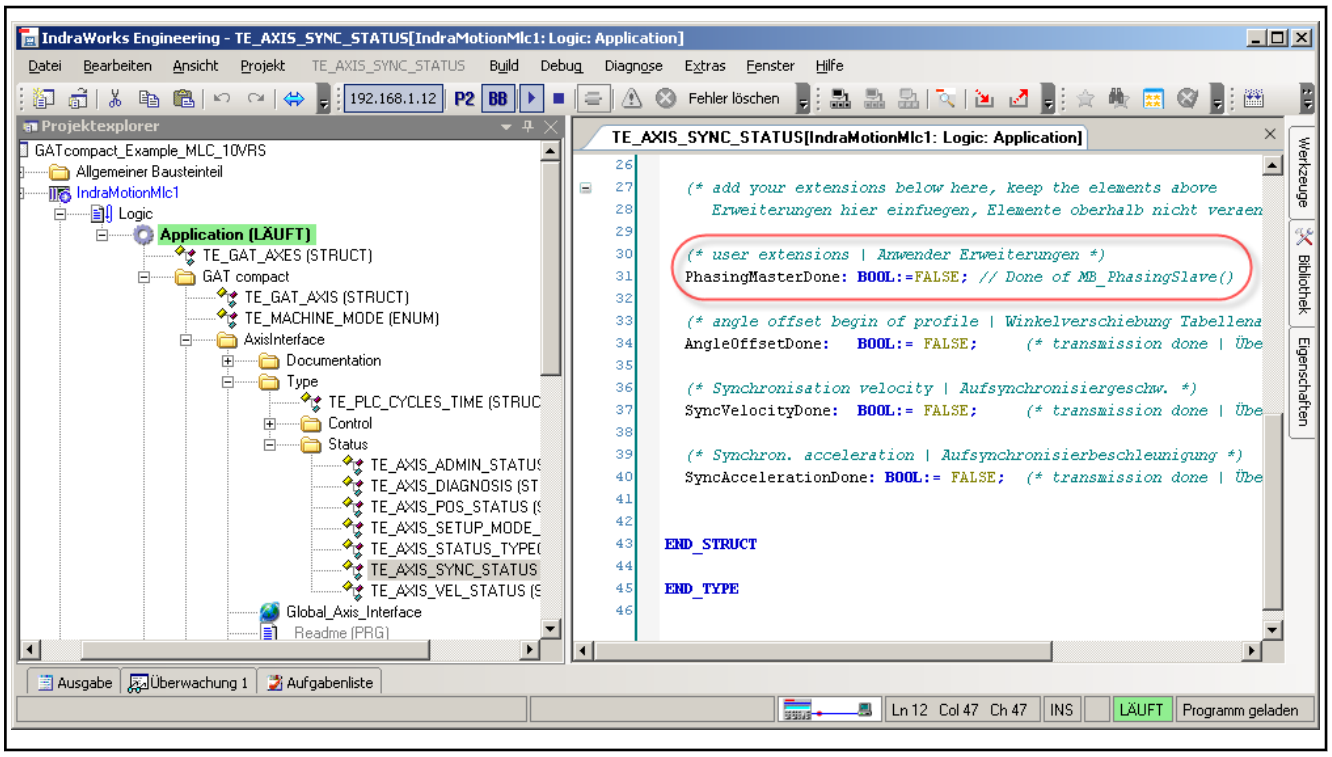


Abb.6-60: Neu zu TE_AXIS_SYNC_STATUS hinzugefügtes Element

6.7.4 Erweitern des Funktionsbausteines

Der letzte Schritt im Ablauf der Anwender-Erweiterung ist, den Funktionsbaustein so zu erweitern, dass die neuen Elemente benutzt werden können.

1. Im Projektbaum, mit rechtem Mausklick **TE_AxisInterface** und **Hinzufügen -> Aktion** wählen.
2. Nennen Sie die neue Aktion **actPhaseShift**.
3. Deklarieren Sie im FB TE_AxisInterface die folgenden Variablen:
   - fbAxisPhasing: MB_Phasing;
   - bExecPhasing: BOOL:= FALSE;
   - rOldPhaseShift: REAL:= 0.01; // alter Defaultwert ungleich 0 um eine Übertragung zu erzwingen
4. So könnte die Aktion ausprogrammiert werden.

ML_TechInterface.library

actPhaseShift[IndraMotionMlc1: Logic: Application: TE_AxisInterface]

```
1  //************************************************************
2  // Implements a PhasingMaster with the function block MB_Phasing
3  //************************************************************
4  //********* handle phase offset when sync-mode is active *******
5  IF( ((INT_TO_WORD(AdminStatusExt._OpModeAck) AND MASK_MODE_PHASE_OFFSET) <> 0) )THEN
6    IF( rOldPhaseShift <> SyncModeCtrlExt.PhaseShift) THEN
7      // start transmission
8    SyncModeStatusExt.PhasingMasterDone := FALSE;
9    bExecPhasing := TRUE;
10   END_IF
11   IF fbAxisPhasing.Done THEN
12     // finished GOOD transmission
13   SyncModeStatusExt.PhasingMasterDone := TRUE;
14   bExecPhasing := FALSE;
15     // remember old value
16     rOldPhaseShift := SyncModeCtrlExt.PhaseShift;
17   ELSIF (fbAxisPhasing.Error = TRUE) THEN
18     // finished BAD transmission
19     bExecPhasing := FALSE;
20     SUPER^.mSetError( Error      := TRUE,
21                       ErrorID    := fbAxisPhasing.ErrorID,
22                       ErrorIdent := fbAxisPhasing.ErrorIdent);
23   END_IF
24   // keep the FB always active to generate PhasingMasterDone
25   fbAxisPhasing(
26     Execute      := bExecPhasing,
27     PhaseShift   := SyncModeCtrlExt.PhaseShift,
28     Velocity     := SyncModeCtrlExt.PhaseShiftVel,
29     Acceleration := SyncModeCtrlExt.PhaseShiftAcc,
30     Axis         := AdminCtrlExt.Axis );
31 ELSE
32   SyncModeStatusExt.PhasingMasterDone := FALSE; // Reset, when no Sync-Mode active
33 END_IF // end of 'IF( (INT_TO_WORD(AdminStatusExt._OpMode.enAck) AND MASK_MODE_PHASE_OFFSET) <> 0 ) THEN'
```

Abb.6-61: Erweiterung des Anwenderprogramms

5. Rufen Sie im FB TE_AxisInterface die neue Aktion "actPhaseShift()" auf.
6. Übersetzen Sie das Projekt neu und überprüfen Sie es auf Programmierfehler.
7. Laden Sie das Projekt in die Steuerung.

Die neuen Eingangs- und Ausgangs-Elemente sind nun ein Teil der Achs-Interface-Struktur und können über die UserVarGlobal-Variablen betrachtet werden.

FineAdjust	REAL	0
CamShaftDistance	REAL	360
PhaseOffset	REAL	0
PhaseOffsetVel	REAL	1
PhaseOffsetAcc	REAL	100
SyncDirection	MC_SYNC_DIRECTION	MC_SYNC_DIRECTI...
StartMode	MC_START_MODE	MC_START_MODE.A...
CamTableID	MC_CAM_ID	MC_CAM_ID.CAM_T...
Master	AXIS_REF	
Profile	ML_AXIS_PROFILE	
AdjFineAdjust	MB_AXIS_ADJUST_DATA	
AdjCamShaftDistance	MB_AXIS_ADJUST_DATA	
AdjPhaseOffset	MB_AXIS_ADJUST_DATA	
AngleOffset	REAL	0
PhaseShift	REAL	0
PhaseShiftVel	REAL	10
PhaseShiftAcc	REAL	100
SyncVelocity	REAL	10
SyncAcceleration	REAL	10
IndraMotionMlc1.Application.UserVarGlobal.arAxisStatus_gb[1]	TE_AXIS_STATUS_TYPE01	
Admin	TE_AXIS_ADMIN_STATUS	
Diag	TE_AXIS_DIAGNOSIS	
PosMode	TE_AXIS_POS_STATUS	
VelMode	TE_AXIS_VEL_STATUS	
SyncMode	TE_AXIS_SYNC_STATUS	
CamActiveID	MC_CAM_ID	16#0000
ProfileActiveSet	UINT	16#0000
PhasingSlaveDone	BOOL	FALSE
AdjFineAdjust	MB_AXIS_ADJUST_STATUS	
AdjCamShaftDistance	MB_AXIS_ADJUST_STATUS	
AdjPhaseOffset	MB_AXIS_ADJUST_STATUS	
AngleOffsetDone	BOOL	TRUE
SyncVelocityDone	BOOL	TRUE
SyncAccelerationDone	BOOL	TRUE
PhasingMasterDone	BOOL	FALSE
PtrAxisData	POINTER TO ML_AXISDAT...	16#0A6B1B94
SetupMode	TE_AXIS_SETUP_MODE_S...	

Abb.6-62: arAxisCtrl_gb und arAxisStatus_gb Strukturen mit Anwender-Erweiterungen

6.8 Achs-Interface-Strukturen

6.8.1 Überblick

Das Achs-Interface stellt eine Datenschnittstelle zur komfortablen Ansteuerung der Achsen zur Verfügung. Die Datenstrukturen zum Achs-Interface werden im folgendem Abschnitt beschrieben.

☞ Das Achs-Interface im Programmier-Template GAT (Generic Application Template) bzw. GAT compact ist so vorbereitet, dass es durch den Anwender erweiterbar ist bzw. es sind einige Erweiterungen schon mit eingebaut. Um diese Erweiterungen zu ermöglichen, ist es nötig, eigene Strukturen im Anwenderprojekt zu definieren, die die Strukturen der Bibliothek erweitern. Die erweiterten Strukturen sind durch den Präfix "TE_" gekennzeichnet. Wenn also im GAT bzw. GAT compact eine Struktur, z.B. **TE_**AXIS_ADMIN_STATUS heisst, ist diese eine erweiterte Struktur der **MB_**AXIS_ADMIN_STATUS und Informationen zu der Struktur sind unter dem Namen **MB_**AXIS_ADMIN_STATUS zu finden.

ML_TechInterface.library

Die folgende Tabelle zeigt das Anwender-Interface zur Achs-Interface Datenstruktur:

Anwender-Interface	Typ	Beschreibung
arAxisCtrl_gb[ ]	MB_AXIS_CONTROL_TYPE01 (GAT:TE_AXIS_CONTROL_TYPE01 MB_AXIS_CONTROL_TYPE02 (GAT:TE_AXIS_CONTROL_TYPE02	Steuerungsstruktur inklusive Sollwerte und Variablen zum Aktivieren der Betriebsarten
arAxisStatus_gb[ ]	MB_AXIS_STATUS_TYPE01 (GAT:TE_AXIS_CONTROL_TYPE01) MB_AXIS_STATUS_TYPE02 (GAT:TE_AXIS_CONTROL_TYPE02)	Statusstruktur inklusive Diagnose-Info und Quittungen für die Betriebsarten
AxisData[ ]		Status- und Steuerungsstruktur inklusive Istwerte, Status-Bits und Zugriff auf die zyklisch konfigurierbaren Anwenderdaten (wird durch die MotionControl in der Firmware bearbeitet)

Abb.6-63: Achs-Interface - Anwender-Schnittstelle Datenstrukturen

6.8.2 MB_AXIS_STATUS_TYPE01 Struktur

MB_AXIS_STATUS_TYPE01

Kurzbeschreibung Auf die MB_AXIS_STATUS_TYPE01 Datenstruktur wird über die **arAxisStatus_gb[ ]** Instanz zugegriffen; sie enthält die folgenden Elemente:

```
TYPE MB_AXIS_STATUS_TYPE01 :

STRUCT
    (********* Outputs of the following axis - from motion control to logic control **********)

    Admin:   MB_AXIS_ADMIN_STATUS; (* status of the axis | Achs-Status *)

    Diag:    MB_AXIS_DIAGNOSIS; (* axis diagnosis information, error, warning....
      Diagnose Informationen der Achse, Fehler, Warnung.... *)

    PosMode: MB_AXIS_POS_STATUS; (* Status operating mode positioning (absolute, rel. and add.)
      Statusmeldungen Positionierbetrieb (absolut, relativ und additiv) *)

    VelMode: MB_AXIS_VEL_STATUS; (* Status operating mode velocity control
      Statusmeldungen Geschwindigkeitsregelung (Freilauf) *)

    SyncMode: MB_AXIS_SYNC_STATUS; (* Status operating mode synchronisation (phase, vel, and cam)
      Statusmeldungen Synchronbetriebsarten (Winkelsync., Geschwindigkeitssync. und Kurvensch.)*)

    PtrAxisData:  POINTER TO ML_AXISDATA_SM;
     (* Pointer to AxisData of this axis | Zeiger auf AxisData dieser Achse
      - Type .PtrAxisData^. for the intellisense menu | Eingabe .PtrAxisData^. fuer Auswahlmenue*)

(* add your extensions below here, keep the elements above
   Erweiterungen hier einfuegen, Elemente oberhalb nicht veraendern *)

(* user extensions | Anwender Erweiterungen *)


END_STRUCT

END_TYPE
```

Abb.6-64: MB_AXIS_STATUS_TYPE01

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

MB_AXIS_STATUS_TYPE01 Elemente

MB_AXIS_ADMIN_STATUS Struktur

Kurzbeschreibung Die MB_AXIS_ADMIN_STATUS Struktur ist eine Unterstruktur von MB_AXIS_STATUS_TYPE01 und enthält den Ausgangsstatus.

Das folgende Bild zeigt die aktuelle MB_AXIS_ADMIN_STATUS Struktur:

MB_AXIS_ADMIN_STATUS

```
STRUCT
    _OpModeAck: MB_AXIS_MODE:= ModeAb; (* acknowledge of the operating mode | Quittierung der Betriebsart
      - check ..._OpModeAck = ModeAB, ModeAH, ModePosAbs .... TO read the active operating mode
      - OR use bits ....MODE_AH, ....MODE_POS_ABS ....
      ------------------------------------------------------------------------------
      - Lesen der aktiven Betriebsart: ..._OpModeAck = ModeAB, ModeAH, ModePosAbs ....
      - oder BIT-Auswertung  ...MODE_AH, ...MODE_POS_ABS .... *)
    (* bit access to the OpMode variables *)
    {attribute 'relative_offset':='0'}
    (* AH, under torque OpMode(Ack).0 *)
    MODE_AH           :BIT := FALSE;
    MODE_HOMING       :BIT := FALSE;  (* AF, MB_Home OpMode(Ack).1 *)
    MODE_POS_ABS      :BIT := FALSE;  (* AF, MC_MoveAbsolute OpMode(Ack).2 *)
    MODE_POS_REL      :BIT := FALSE;  (* AF, MC_MoveRelative OpMode(Ack).3 *)
    MODE_POS_ADD      :BIT := FALSE;  (* AF, MC_MoveAdditive OpMode(Ack).4 *)
    MODE_VEL          :BIT := FALSE;  (* AF, MC_MoveVelocity OpMode(Ack).5 *)
    MODE_SYNC_PHASE   :BIT := FALSE;  (* AF, MB_GearInPos OpMode(Ack).6 *)
    MODE_SYNC_VEL     :BIT := FALSE;  (* AF, MC_GearIn OpMode(Ack).7 *)
    MODE_SYNC_CAM     :BIT := FALSE;  (* AF, MC_CamIn OpMode(Ack).8 *)
    MODE_SYNC_PROFILE :BIT := FALSE;  (* AF, MB_MotionProfile OpMode(Ack).9 *)
    { attribute 'hide' } MODE_TORQUE :BIT := FALSE;  (* AF, MC_TorqueControl OpMode(Ack).10 not implemented *)
    MODE_FLEX_PROFILE :BIT := FALSE;  (* AF, MB_FlexProfile OpMode(Ack).11 *)
    MODE_STOPPING     :BIT := FALSE;  (* AH, under torque stopping OpMode(Ack).12 *)
    MODE_COORDINATED  :BIT := FALSE;  (* AF, AddAxisToGroup OpMode(Ack).13*)
    MODE_INTERRUPTED  :BIT := FALSE;  (* only status: interrupted by a drive reaction e.g. safety OpMode(Ack).14 *)
    MODE_EXTERNAL_FB  :BIT := FALSE; (* ??, Switch to AH and wait for external move commands. [2 lines]
    (* TRUE: drive is activated | TRUE: Achse aktiviert [1 lines]
    Active:     BOOL:=FALSE;

    (* comment, name of axis | Kommentar, Name der Achse [1 lines]
    Name:       STRING(40);

    (* Last motion command is finished | Letztes Kommando ist abgeschlossen. [1 lines]
    CmdDone:    BOOL:=FALSE;

    (* Type of the axis | Achstyp [2 lines]
    { attribute 'hide' }AxisType:   MB_AXIS_CONFIG:=NoAxisConfig; // replaced by AxisFeatures from 13VRS on

    (* Status of command SetAbsRef (P-0-0012) | Status des Kommandos SetAbsRef (P-0-0012)
      - siehe auch MB_SERCOS_DATA_STATUS | - see also MB_SERCOS_DATA_STATUS *)
    AbsRefCmdStatus:    MB_SERCOS_DATA_STATUS:=MB_SERCOS_DATA_NOT_SET;

    (* Internal flags for the handling of different drives, e.g use P-0-0012 or S-0-0447 as Cmd AbsRef *)
    InternalFlags: DWORD:=0;

    (* parameter level 1, parameter A-0-0024 "Achszustand" bit 3 active *)
    (* TRUE:  parameter level 1 active *)
    (* FALSE: parameter level 1 not active *)
    ParamLevelAck: BOOL:=FALSE;

    (* Features of this axis (A-0-0004) | Funktionalität dieser Achse (A-0-0004) *)
    AxisFeatures: MB_AXIS_FEATURES := (wData:=0);
END_STRUCT
```

Abb.6-65: MB_AXIS_ADMIN_STATUS

Parametrierebene

Mit dem Ausgang "ParamLevelAck" wird angezeigt ob die Achse mittels des Eingangs "ParamLevel" in den Parametriermodus gesetzt wurde.

Durch Wegnehmen des Einganges wird die Achse wieder in den ursprünglichen Zustand gesetzt.



Ist der Parametriermodus aktiv, wird im Projektbaum bei der zugehörigen Achse "PM" angezeigt und in der AxisDatastruktur geht das Bit "OperationMode" weg.

ML_TechInterface.library

AxisFeatures

Das Element "AxisFeatures" gibt binär codiert die Eigenschaften der Achse an.

"AxisType" wird dadurch ersetzt und sollte zukünftig nicht mehr verwendet werden.

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

MB_AXIS_ADMIN_STATUS Elemente

MB_AXIS_MODE

Kurzbeschreibung Der MB_AXIS_MODE ist ein Aufzählungstyp der vom **_OpModeAck** Element, das in MB_AXIS_ADMIN_STATUS enthalten ist, verwendet wird.

Das folgende Bild zeigt den aktuellen MB_Axis_Mode Aufzählungstyp:

MB_AXIS_MODE

```
1  TYPE MB_AXIS_MODE :
2      (
3      (* enumeration access to the OpMode variables *)
4          ModeAB          := 16#0000, (* AB, free of torque  *)
5          ModeAH          := 16#0001, (* AH, under torque, end state standstill (AB->AH or AF->AH) *)
6          ModeHoming      := 16#0002, (* AH, MB_Home *)
7          ModePosAbs      := 16#0004, (* AF, MC_MoveAbsolute *)
8          ModePosRel      := 16#0008, (* AF, MC_MoveRelative *)
9          ModePosAdd      := 16#0010, (* AF, MC_MoveAdditive *)
10         ModeVel         := 16#0020, (* AF, MC_MoveVelocity *)
11         ModeSyncPhase   := 16#0040, (* AF, MB_GearInPos *)
12         ModeSyncVel     := 16#0080, (* AF, MC_GearIn *)
13         ModeSyncCam     := 16#0100, (* AF, MC_CamIn *)
14         ModeSyncProfile := 16#0200, (* AF, MB_MotionProfile *)
15         (* ModeTorque     := 16#0400, ; not implemented *) (* AF, MC_TorqueControl *)
16         ModeFlexProfile := 16#0800, (* AF, ML_FlexProfile *)
17         ModeStopping    := 16#1000, (* AH, under torque, end state stopping (AB->AH or AF->AH) *)
18         ModeInterrupted := 16#4000, (* ??, only Status:  Move comand interrupted by drive internal reaction *)
19         ModeExternalFB  := 16#8000  (* ??, Status:  Move comand aborted by external FB *)
20                                     (* ??, Control: Only Power is activated and then an *)
21                                     (*              external FB can overtake control *)
22     ) UINT;
23 END_TYPE
```

Abb.6-66: MB_Axis_Mode

MB_AXIS_FEATURES

Kurzbeschreibung MB_AXIS_FEATURES ist eine Struktur, die von der MB_AXIS_ADMIN_STATUS Struktur verwendet wird.

```
TYPE MB_AXIS_FEATURES :
STRUCT
  { attribute 'hide' } wData: WORD; // used only to set the info
  {attribute 'relative_offset':='0'}
   // bits from parameter A-0-0004 Axis type information is partly redundant to Status.Admin.AxisType
  IPO_CONTROL:       BIT;  // .0 FALSE: IPO Drive
  PACK_PROFILE:      BIT;  // .1 telegram konfiguration FALSE: device specific
  HYDRAULIC:         BIT;  // .2 FALSE: Electric drive
  REAL_AXIS:         BIT;  // .3 IndraDrive, HNC-axis, sercos drive
  VIRTUAL_AXIS:      BIT;  // .4 virtual axis
  ENCODER_AXIS:      BIT;  // .5 IndraDrive, HNC-axis, sercos drive
  LINK_AXIS:         BIT;  // .6 link axis
  CONTROLLER_AXIS:   BIT;  // .7 controller axis
  { attribute 'hide' } resbit8:          BIT;
  { attribute 'hide' } resbit9:          BIT;
  { attribute 'hide' } resbit10:         BIT;
  OPMODE_VEL:        BIT;  // .11 Operation mode velocity control supported
  OPMODE_POS:        BIT;  // .12 Operation mode positioning control supported
  OPMODE_TORQUE:     BIT;  // .13 Operation mode torque/force control supported
  OPMODE_SYNC:       BIT;  // .14 Operation mode synchronisation supported
  PARAM_LEVEL:       BIT;  // .15 param level supported (13VRS)
END_STRUCT
END_TYPE
```

Abb.6-67: MB_AXIS_FEATURES Struktur

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

MB_AXIS_DIAGNOSIS Struktur

Kurzbeschreibung Die MB_AXIS_DIAGNOSIS Struktur ist eine Unterstruktur von MB_AXIS_STATUS_TYPE01 und enthält Warnungs- und Fehler-Ausgänge. Das folgende Bild zeigt die aktuelle MB_AXIS_DIAGNOSIS Struktur:

ML_TechInterface.library

```
TYPE MB_AXIS_DIAGNOSIS :

STRUCT
    (********** Diagnosis outputs - from motion control to logic control *****************)
    (********** Diagnose-Ausgaenge - Motion Control zur Logic Control    *****************)

    Warning:       BOOL := FALSE; (* Warning, copy of | Warnung, Kopie von
      AxisData[].Axis_Warning *)

    Error:         BOOL := FALSE;
    (* Error -> AxisData[].Axis_Error OR Error from an FB in the Axis-Interface
       Fehler -> AxisData[].Axis_Error ODER Fehler von einem FB im Axis-Interface *)

    ErrorID:    ERROR_CODE;   (* Error type (internal used FB) | Fehlertyp (intern genutzter FB)
      - only valid when Error = TRUE | - gueltig wenn Error = TRUE *)

    ErrorIdent: ERROR_STRUCT; (* Error id. (internal used FB) | Fehler-Id. (intern genutzter FB)
      - only valid when Error = TRUE | - gueltig wenn Error = TRUE *)

    Number:    DWORD; (* diagnosis number of the axis | Diagnosenummer der Achse
      - only valid in operating mode | nur gueltig im Betriebsmodus
      - refresh rate controlled with | Auffrischrate gesteuert mit
        arAxisCtrl[].Admin.DiagNbrRefreshTime *)

    Message:    STRING(60); (* diagnosis message of the axis | Diagnosemeldung der Achse
      - only valid in operating mode    | nur gueltig im Betriebsmodus
      - refreshed when "Number" changes | Aufgefrischt bei Aenderung von "Number" *)

END_STRUCT

END_TYPE
```

Abb.6-68: MB_AXIS_DIAGNOSIS

Diagnoseanzeigen

Mit arAxisCtrl_gb[iColumn].Admin.DiagNbrRefreshTime wird die Aktualisierungsrate des Elementes "Number" gesteuert. Sobald sich das Element "Number" ändert, wird der Text "Message" neu gelesen. Mit arAxisCtrl_gb[].Admin.DiagNbrRefreshTime kann auch das Lesen der Diagnoseinformationen abgeschaltet werden bzw. es kann auch die Datenquelle der Diagnoseinformationen gewählt werden.

- DiagNbrRefreshTime = T#0ms: Bildung von Diagnosenummer und Diagnosemeldung abgeschaltet
- DiagNbrRefreshTime zwischen T#1ms und T#199ms (default T#100ms): Es werden immer die Steuerungsparameter A-0-0023/ A-0-0020 gelesen und als "Number" bzw. Text "Message"ausgegeben
- DiagNbrRefreshTime grösser gleich T#200ms: Das Achs-Interface liest in Abhängigkeit des Achstyp und der Meldungsart jeweils die passenden Antriebs- oder Steuerungsparameter

  IndraDrive mit Interpolation im Antrieb:

  – Gutmeldungen: Antriebsparameter werden gelesen. Bsp: A0012 Betriebsbereit
  – Fehler im Antrieb: Antriebsparameter werden gelesen. Bsp: F2026 Leistung weg
  – Fehler auf der Steuerung: Steuerungsparameter werden gelesen. Bsp: F2229233 ungültige Leitachse

  IndraDrive mit Interpolation in der Steuerung:

ML_TechInterface.library

- Gutmeldungen A-0-0020/A-0-0023 werden gelesen. Bsp: "Synchronized Motion"
- Fehler im Antrieb: Antriebsparameter werden gelesen. Bsp: F2026 Leistung weg
- Fehler auf der Steuerung: Steuerungsparameter werden gelesen. Bsp: F2229233 ungültige Leitachse

SercosDrive und virtuelle Achsen:

- Es werden immer die Steuerungsparameter gelesen

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit ML_ImcInterface.lib für IndraMotion MLC/MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

MB_AXIS_POS_STATUS Struktur

Kurzbeschreibung Die MB_AXIS_POS_STATUS Struktur ist eine Unterstruktur von MB_AXIS_STATUS_TYPE01 und enthält den Ausgangsstatus.

Das folgende Bild zeigt die aktuelle MB_AXIS_POS_STATUS Struktur:

```
TYPE MB_AXIS_POS_STATUS :

STRUCT
    (********** Outputs of the following axis - from motion control to logic control **********)
    (********** Achs-Ausgaenge - Motion Control zur Logic Control                    **********)

    Adjust: MB_AXIS_ADJUST_STATUS; (* -> outputs of the adjusting functionality for the position
      -> Ausgaenge fuer Tippbetrieb der Position *)

END_STRUCT

END_TYPE
```

Abb.6-69: MB_AXIS_POS_STATUS

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

MB_AXIS_ADJUST_STATUS

Kurzbeschreibung Die MB_AXIS_ADJUST_STATUS Struktur ist eine Unterstruktur von MB_AXIS_POS_STATUS und enthält den Ausgangsstatus.

Das folgende Bild zeigt die aktuelle MB_AXIS_POS_STATUS Struktur:

ML_TechInterface.library

```
TYPE MB_AXIS_ADJUST_STATUS :

STRUCT
    (********** Outputs of the following axis - motion control to logic control **********)
    (********** Achs-Ausgaenge - Motion Control zur Logic Control               **********)

    ModeAck:    BOOL := FALSE; (* Ackn. enable adjust mode | Quittierung Freigabe Tippbetrieb *)

    HighLimitAck: BOOL := FALSE; (* max. value reached |  max. Wert erreicht *)

    LowLimitAck:  BOOL := FALSE; (* min. value reached |  min. Wert erreicht *)

END_STRUCT

END_TYPE
```

Abb.6-70: MB_AXIS_ADJUST_STATUS

☞ Zum "Adjust" werden intern die Funktionsbausteine MB_ContinuousAdjustType01 und MB_IncrementalAdjustType01 aus der Bibliothek ML_TechBase verwendet.

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

MB_AXIS_VEL_STATUS Struktur

Kurzbeschreibung Die MB_AXIS_VEL_STATUS Struktur ist eine Unterstruktur von MB_AXIS_STATUS_TYPE01 und enthält den Ausgangsstatus der Geschwindigkeit.

Das folgende Bild zeigt die aktuelle MB_AXIS_VEL_STATUS Struktur:

```
TYPE MB_AXIS_VEL_STATUS :

STRUCT
    (********** Outputs of the following axis - from motion control to logic control **********)
    (********** Achs-Ausgaenge - Motion Control zur Logic Control                    **********)

    Adjust: MB_AXIS_ADJUST_STATUS; (* -> outputs of the adjusting functionality for the velocity
      -> Ausgaenge fuer Tippbetrieb der Geschwindigkeit *)

END_STRUCT

END_TYPE
```

Abb.6-71: MB_AXIS_VEL_STATUS

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

MB_AXIS_SYNC_STATUS Struktur

Kurzbeschreibung Die MB_AXIS_SYNC_STATUS Struktur ist eine Unterstruktur von MB_AXIS_STATUS_TYPE01 und enthält den Ausgangsstatus.

ML_TechInterface.library

Das folgende Bild zeigt die aktuelle MB_AXIS_SYNC_STATUS Struktur:

```
TYPE MB_AXIS_SYNC_STATUS :
STRUCT
    (********* Outputs of the following axis - from motion control to logic control **********)
    (********* Achs-Ausgaenge - Motion Control zur Logic Control                    **********)

    CamActiveID:      MC_CAM_ID; (* - ID of the active CamTable | - Nr. aktive Stuetzpunkt-Tab.*)

    ProfileActiveSet: UINT:= 0;  (* - Number of the active motion profile set 0 or 1
      - Nummer des aktiven Satzes 0 oder 1 des Bewegungsprofils *)

    PhasingSlaveDone: BOOL := FALSE; (* - Change of PhaseOffset is finished
      - Verstellung des PhaseOffset ist abgeschlossen  *)

    (*************** status for adjusting of synchronisation mode values ******************)
    (*************** Statusmeldungen, Tippen im Synchronbetrieb          ******************)

    AdjFineAdjust:       MB_AXIS_ADJUST_STATUS; (* adjusting | vertippen
      arAxisCntrl[].SyncMode.FineAdjust *)

    AdjCamShaftDistance: MB_AXIS_ADJUST_STATUS; (* adjusting | vertippen
      arAxisCntrl[].SyncMode.CamShaftDistance *)

    AdjPhaseOffset:      MB_AXIS_ADJUST_STATUS; (* adjusting | vertippen
      arAxisCntrl[].SyncMode.PhaseOffset *)

END_STRUCT
END_TYPE
```

Abb.6-72: MB_AXIS_SYNC_STATUS

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

6.8.3 MB_AXIS_CONTROL_TYPE01 Struktur

MB_AXIS_CONTROL_TYPE01

Kurzbeschreibung Auf die MB_AXIS_CONTROL_TYPE01 Datenstruktur wird über die **arAxisCtrl_gb[ ]** Instanz zugegriffen; sie enthält folgende Elemente:

ML_TechInterface.library

```
TYPE MB_AXIS_CONTROL_TYPE01 :

STRUCT
    Admin:      MB_AXIS_ADMINISTRATION; (* Administration of the axis | Verwaltung der Achse *)

    PosMode:    MB_AXIS_POSITIONING; (* operating mode positioning (absolute, rel. and additive)
      Positionierbetrieb (absolut, relativ und additiv) *)

    VelMode:    MB_AXIS_VELOCITY; (* operating mode velocity control | Geschwindigkeitsregelung *)

    SyncMode:   MB_AXIS_SYNCHRONISATION; (* operating mode synchronisation (phase, vel, and cam)
      Synchronbetriebsarten (Winkelsync., Geschwindigkeitssync. und Kurvenscheibe) *)

(* add your extensions below here, keep the elements above
   Erweiterungen hier einfuegen, Elemente oberhalb nicht veraendern *)

(* user extensions | Anwender Erweiterungen *)


END_STRUCT

END_TYPE
```

Abb.6-73: MB_AXIS_CONTROL_TYPE01

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

MB_AXIS_CONTROL_TYPE01 Elemente

MB_AXIS_ADMINISTRATION Struktur

Kurzbeschreibung Die MB_AXIS_ADMINISTRATION Struktur ist eine Unterstruktur von MB_AXIS_CONTROL_TYPE01 und enthält Steuerungsvariablen und Sollwerte.

Das folgende Bild zeigt die aktuelle MB_AXIS_ADMINISTRATION Struktur:

MB_AXIS_ADMINISTRATION

```
TYPE MB_AXIS_ADMINISTRATION :
STRUCT
  (* selection of the operating mode |  Betriebsarten-Anwahl
     - level controlled
     - set the corresponding variables for your operating mode before activating
     - set ..._OpMode.en.en := ModeAB, ModeAH, ModePosAbs .... to activate an operating mode
     - OR use bit variables ..._OpMode.en.b.MODE_AB, ..._OpMode.en.b.MODE_AH, _OpMode.b.MODE_POS_ABS.
       Note: only one bit is allowed at any time!
     ---------------------------------------------------------------------------
     - Pegel-sensitiv
     - vor Anwahl der Betriebsart die Eingangsdaten fuer die Betriebsart beschreiben
     - Betriebsarten-Anwahl: ..._OpMode.en.en := ModeAB, ModeAH, ModePosAbs ....  (ENUMs)
     - oder ..._OpMode.en.b.MODE_AB, ..._OpMode.en.b.MODE_AH, OpMode.b.MODE_POS_ABS.  (BIT-Variablen)
       Hinweis: immer nur 1 Eingang setzen! *)
  _OpMode:  MB_AXIS_MODE_UNION := ( en := ModeAb);

                      (* the reference to this axis | Achse adressieren/auswaehlen  *)
  Axis:   AXIS_REF := (AxisNo := NO_OBJECT, CntrlNo := LOCAL_CNTRL);

  (* Refresh time for arAxisStatus[].Diag.Number | Auffrischrate arAxisStatus[].Diag.Number
    - switched off, if zero           | - ausgeschaltet wenn Null
    - read drive para when >= T#200ms | - liest Antriebsparameter wenn >= T#200ms *)
  DiagNbrRefreshTime: TIME:=T#100MS;

  (* Deceleration when DRIVE_HALT is activated - always positive,
     (Unit: e.g. in [rad/s2] depending on parametrization)
     Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
     ---------------------------------------------------------------------------
     Freigabe-Verzoegerung bei Aktiv. ANTRIEB_HALT- positiver Wert,
     (Einheit: z.B. in [rad/s2] je nach Parametrierung)
     Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)
  StopDeceleration: REAL:=99.0;

  (* WaitTime restart of the MC_Power in [mSec] | Wartezeit Neustart des MC_Power in [mSec]
     - necessary for some drives e.g. Eco-Cs | - notwendig fuer einige Antriebe z.B. Eco-Cs *)
  WaitTimePowerRestart: UDINT:=0;

  (* clears axis errors with MC_Reset() | loescht Achsfehler mit MC_Reset()
     - edge controlled | flanken-sensitiv *)
  ClearError:   BOOL:=FALSE;

  (* Set absolute reference by command S-0-0447 (P-0-0012) | Absolutmass setzen mit Kommando S-0-0447 (P-0-0012)
     - edge controlled | flanken-sensitiv *)
  SetAbsRef:    BOOL:=FALSE;

  (* Enable the cyclic check and transmission of changed inputs for the actual operating mode
     - relevant for the values in AxisCtrl with "Note: cyclic checked AND transmitted on change"
     - level controlled
     ---------------------------------------------------------------------------------------------
     Zuschalten des bedarfsweisen Scannens von Eingangsdaten in der aktuellen Betriebsart
     - betrifft Groessen in arAxisCtrl mit Anm. "Note: cyclic checked AND transmitted on change"
     - Pegel-sensitiv *)
  EnableCyclicScanning:  BOOL:=TRUE;

  (* Enable the update on inputs, that are not cyclic checked, without leaving the active opmode
     - relevant for values in AxisCtrl without "Note: cyclic checked AND transmitted on change"
     - level controlled
     ---------------------------------------------------------------------------------------------
     Zuschalten der Aktualisierung, von nicht zyklisch gescannten Eingangsdaten, ohne die
     aktuelle Betriebsart zu verlassen
     - betrifft Groessen in arAxisCtrl ohne "Note: cyclic checked AND transmitted on change"
     - Pegel-sensitiv *)
  UpdateEveryInput:  BOOL:=FALSE;

  (* TRUE: When the drive is in ModeInterrupted (e.g. drive internal command) the axis will be stopped
     ---------------------------------------------------------------------------------------------
     TRUE: Im Falle von ModeInterrupted (z.B. internes Antriebskommando) wird die Achse angehalten *)
  DriveInterruptReaction:  BOOL:=TRUE;

  (* TRUE: MC_Power is internally activated when an _OpMode is selected
     FALSE: the user application has to handle the MC_Power before an _OpMode is selected
     ---------------------------------------------------------------------------------------------
     TRUE: der MC_Power wird intern aktiviert wenn _OpMode angewählt wird
     FALSE: die Anwenderapplikation muß den MC_Power freigeben, bevor eine Betriebsart angewählt wird *)
  PowerOn:  BOOL:=TRUE;

  (* Switch to parameter level 1, parameter A-0-0024 "Achszustand" bit 3 *)
  (* TRUE:  activate parameter level 1 *)
  (* FALSE: parameter level not activated *)
  ParamLevel: BOOL:=FALSE;

END_STRUCT
END_TYPE
```

Abb.6-74: MB_AXIS_ADMINISTRATION

Betriebsartenanwahl

Betriebsarten können durch jede der folgenden Methoden angewählt werden:

- **Anwahl über Enum-Werte**

  Zuweisen eines Wertes vom Typ MB_AXIS_MODE zu arAxisCtrl_gb[ ].Admin._OpMode

  z.B. **arAxisCtrl_gb[ ].Admin._OpMode.en :=ModePosAbs;** oder **arAxisCtrl_gb[].Admin._OpMode.en := ModeAB;**

- **Anwahl über Bit-Zugriff**

  Zuerst "_OpMode" durch **arAxisCtrl_gb[ ].Admin._OpMode := ModeAB;** löschen, dann über die Bit-Zugriffsfunktionalität nur ein Bit setzen

z.B. **arAxisCtrl_gb[ ].Admin._OpMode.b.MODE_POS_ABS := TRUE;**

☞ Das Setzen mehrerer Bits in **arAxisCtrl_gb[].Admin._OpMode** kann einen Fehler verursachen, der den Antrieb in AB schaltet.

Diagnoseanzeigen

Mit dem Element "DiagNbrRefreshTime" wird die Aktualisierungsrate von arAxisStatus_gb[].Diag.Number gesteuert. Sobald sich arAxisStatus_gb[]. Diag.Number ändert, wird arAxisStatus_gb[].Diag.Message neu gelesen.

Parametrierebene

Mit dem Eingang "ParamLevel" kann eine Achse separat in den Parametriermodus geschaltet werden (Parameter A-0-0024 "Achsstatus" Bit 3 gesetzt.

Damit können für diese einzelne Achse Parameter geändert werden, die sonst nur durch Herunterschalten des gesamtem sercos Rings in sercos Phase 2 geändert werden konnten.

Mit dem Ausgang "ParamLevelAck" wird angezeigt ob die Achse mittels des Eingangs "ParamLevel" in den Parametriermodus gesetzt wurde.

Durch Wegnehmen des Einganges wird die Achse wieder in den ursprünglichen Zustand gesetzt.

☞ Ist der Parametriermodus aktiv, wird im Projektbaum bei der zugehörigen Achse "PM" angezeigt und in der AxisDatastruktur geht das Bit "OperationMode" weg.

Bibliothek

ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit

MX_ImcInterface.lib für IndraMotion MLD

System

IndraMotion MLC/IndraMotion MLD

MB_AXIS_ADMINISTRATION Elemente

MB_Axis_Mode

Kurzbeschreibung

Dieser MB_AXIS_Mode_UNION ist ein Vereinigungsdatentyp (UNION) der vom **_OpMode**-Element, das in der MB_AXIS_ADMINISTRATION Struktur enthalten ist, benutzt wird. Die UNION enthält den Aufzählungstyp MB_AXIS_MODE und die Bitstruktur MB_AXIS_MODE_BITS. Die Steuerung des _OpMode ist über beide Datentypen möglich. Die beiden Datentypen liegen auf der gleichen Speicherstelle. Das folgende Bild zeigt den aktuellen MB_AXIS_MODE Datentyp:

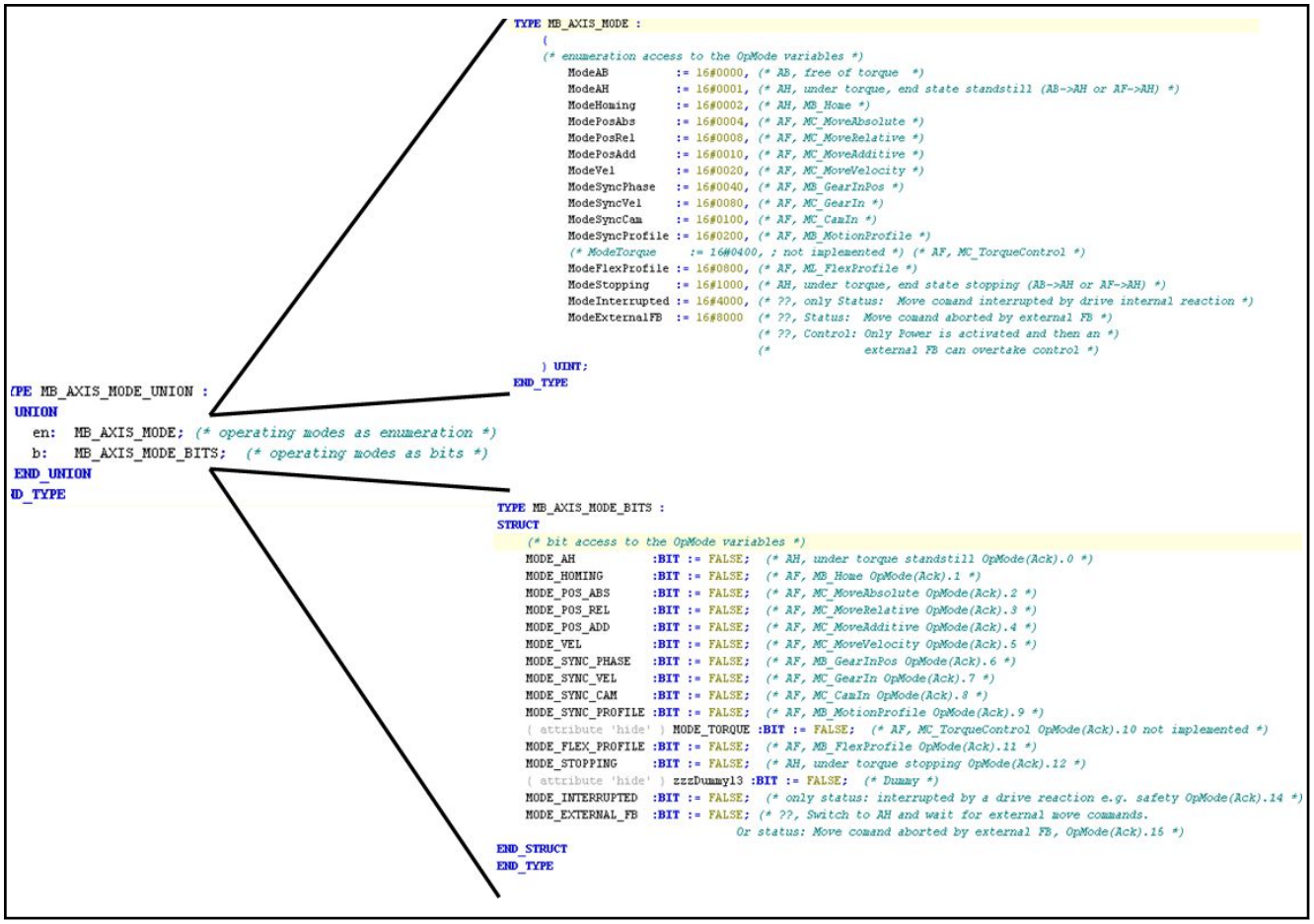


Abb.6-75: Datentyp MB_AXIS_MODE

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

Axis_Ref

Kurzbeschreibung Diese Axis_Ref Struktur ist Teil der "RIL_CommonTypes" Bibliothek und enthält Informationen über die entsprechende Achse. Diese Elemente rufen Instanzen der Aufzählungstypen CONTROLS und OBJECTS auf.

Das folgende Bild zeigt die aktuelle Struktur:

```
(* Structure of PLCopen element AXIS_REF. The AXIS_REF is a structure that contains information *)
(* on the corresponding axis.                                                                   *)
(* It is used as a VAR_IN_OUT in all Motion Control Function Blocks.                            *)
TYPE AXIS_REF :
STRUCT
    CntrlNo:    CONTROLS := LOCAL_CNTRL;        (* control reference number                    *)
    AxisNo:     OBJECTS  := NO_OBJECT           (* object reference number                     *)
END_STRUCT
END_TYPE
```

Abb.6-76: Axis_Ref Struktur

ML_TechInterface.library

MB_AXIS_POSITIONING Struktur

MB_AXIS_ADJUST_DATA

Kurzbeschreibung

Die Struktur MB_AXIS_ADJUST_DATA ist eine Unterstruktur von MB_AXIS_POSITIONING und enthält binäre Einstellungs-Kommandos und Sollwerte. Das folgende Bild zeigt die aktuelle MB_AXIS_ADJUST_DATA Struktur:

```
TYPE MB_AXIS_ADJUST_DATA :
STRUCT
    (*************** - binary adjust commands ******************)
    (*************** - Binaere Verstell-Eingaenge **************)

    Enable:          BOOL; (* Enable adjust mode  | Freigabe Tippbetrieb
      - level controlled                          | - Pegel-sensitiv
      - set before activating:                    | - vor Freigabe folgende Groessen beschreiben:
              ModuloValue, HighLimit, LowLimit, StepWidth, LowStepWidth,
              StepsPerSecond, LowStepsPerSec *)

    Inc:             BOOL; (* Adjust +            | Tippen +
      - level controlled when                     | - Pegel-sensitiv, wenn
              SelIncAdjust = FALSE (ML_ContinuousAdjustType01)
      - edge controlled when                      | - Flanken-sensitiv, wenn
              SelIncAdjust = TRUE (ML_IncrementalAdjustType01)
      - active, when Enable is TRUE               | - aktiv, wenn Enable = TRUE *)

    Dec:             BOOL; (* Adjust -            | Tippen -
      - level controlled when                     | - Pegel-sensitiv, wenn
              SelIncAdjust = FALSE (ML_ContinuousAdjustType01)
      - edge controlled when                      | - Flanken-sensitiv, wenn
              SelIncAdjust = TRUE (ML_IncrementalAdjustType01)
      - active, when Enable is TRUE               | - aktiv, wenn Enable = TRUE *)

    SelIncAdjust:    BOOL;  (* Incremental adjusting | Inkrementelle Verstellung
      - SelIncAdjust =
        TRUE:  one step width is adjusted on every rising edge of Inc or Dec (edge triggered)
        FALSE: the adjusting is executed as long Inc or Dec are TRUE (level triggered)
     ------------------------------------------------------------------------------
     - SelIncAdjust =
        TRUE:  Verstellung um 1 Schrittweite mit jeder steigenden Flanke an Inc oder Dec
        FALSE: kontinuierliche Verstellung, solange Inc oder Dec = TRUE *)
```

Abb.6-77: Struktur (Teil 1 von 2) MB_AXIS_ADJUST_DATA

```
    SelLowLevel:     BOOL; (* Reduced speed of changing | Reduzierte Verstellgeschwindigkeit
      - SelLowLevel = TRUE means:
        step width = StepWidth * LowStepWidth% / 100% and
        speed of changing = StepsPerSecond * LowStepsPerSec% / 100%
     ---------------------------------------------------------------------------------
      - SelLowLevel = TRUE bedeutet:
        Schrittweite = StepWidth * LowStepWidth% / 100% und
        Verstellgeschwindigkeit = StepsPerSecond * LowStepsPerSec% / 100% *)

    (*************** - setpoints of adjust | - Verstell-Variablen ******************)

    AdjModuloValue:  REAL:=0.0; (* ModuloValue for the adjusting | Modulo Wert fuer das Vertippen
      - choose zero for absolute scaling | - bei Absolut-Wichtung auf 0 setzen
      - read from Parameter by ML_AxisInitType01() *)

    HighLimit:       REAL; (* Upper limit of the adj. value | Max. Wert vertippte Zielgroesse *)

    LowLimit:        REAL; (* Lower limit of the adj. value | Min. Wert vertippte Zielgroesse *)

    StepWidth:       REAL; (* StepWidth | Schrittweite
      - s.a.: used to calculate the changing vel. | - s.o.: Berechnung Verstellgeschwindigkeit *)

    LowStepWidth:    INT; (* Reduced StepWidth in percent | Reduzierte Schrittweite in %
      - see above: used to calculate the changing velocity
        SelLowLevel = TRUE: step width = StepWidth * LowStepWidth% / 100%
     ---------------------------------------------------------------------------------
      - siehe oben: Berechnung der Verstellgeschwindigkeit
        SelLowLevel = TRUE: Schrittweite = StepWidth * LowStepWidth% / 100% *)

    StepsPerSecond:  REAL; (* Adjust steps per second |  Verstell-Schritte pro Sekunde
      - s.a.: used to calculate the changing vel. | - s.o.: Berechnung Verstellgeschwindigkeit *)

    LowStepsPerSec:  INT;  (* Reduced changing speed in % | Reduzierte Verstellgeschw. in %
      - see above: used to calculate the speed of changing
        when SelLowLevel = TRUE: speed of changing = StepsPerSecond * LowStepsPerSec% / 100%
     ---------------------------------------------------------------------------------
      - siehe oben: Berechnung der Verstellgeschwindigkeit
        wenn SelLowLevel = TRUE: Verstellgeschw. = StepsPerSecond * LowStepsPerSec% / 100% *)

END_STRUCT
END_TYPE
```

Abb.6-78: Struktur (Teil 2 von 2) MB_AXIS_ADJUST_DATA

☞ Zum "Adjust" werden intern die Funktionsbausteine MB_ContinuousAdjustType01 und MB_IncrementalAdjustType01 aus der Bibliothek ML_TechBase verwendet.

Bibliothek ML_AxisInterface.lib für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

MB_AXIS_POSITIONING

Kurzbeschreibung Die MB_AXIS_POSITIONING Struktur ist eine Unterstruktur von MB_AXIS_CONTROL_TYPE01 und enthält Sollwerte für den Positionierbetrieb. Das folgende Bild zeigt die aktuelle MB_AXIS_POSITIONING Struktur:

ML_TechInterface.library

```
TYPE MB_AXIS_POSITIONING :
STRUCT
    (*************** - setpoints of positioning mode ******************)
    (*************** - Positions-Sollwerte           ******************)

     Position:        REAL;
    (* position for an absolute move (in technical unit[u]) (negative or positive)
      - start moving with "ModePosAbs" (MC_MoveAbsolute)
      Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
      ------------------------------------------------------------------------
      Zielposition, absolut (Einheit: [u]) (negativ oder positiv)
      - Freigabe mit "ModePosAbs" (MC_MoveAbsolute)
      Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)

    Distance:        REAL;
    (* distance for a relative or additive move (in technical unit[u]) (negative or positive)
      - start moving with "ModePosRel" or "ModePosAdd" (MC_MoveRelative, MC_MoveAdditive)
      Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
      ------------------------------------------------------------------------
      Relativer / additiver Sollwert (Einheit: [u]) (negativ oder positiv )
      - Freigabe mit "ModePosRel" oder "ModePosAdd" (MC_MoveRelative, MC_MoveAdditive)
      Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)

    Velocity:        REAL:=10.0;
    (* maximum velocity (always positive) (not necessarily reached) [u/s]
      - start moving with "ModePosAbs" or "ModePosRel" or "ModePosAdd"
        (MC_MoveAbsolute, MC_MoveRelative, MC_MoveAdditive)
      Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
      ------------------------------------------------------------------------
      Positioniergeschwindigkeit , (Einheit: [u/s]) (positiv eingeben)
      - Freigabe mit "ModePosAbs" oder "ModePosRel" oder "ModePosAdd"
        (MC_MoveAbsolute, MC_MoveRelative, MC_MoveAdditive)
      Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)

    Acceleration:    REAL:=10.0;
    (* Value of the acceleration (always positive) (increasing energy of the motor) [u/s2]
      - start moving with "ModePosAbs" or "ModePosRel" or "ModePosAdd"
        (MC_MoveAbsolute, MC_MoveRelative, MC_MoveAdditive)
      ------------------------------------------------------------------------
      Beschleunigung (Einheit: [u/s2]) (nur positive Werte)
      - Freigabe mit "ModePosAbs" oder "ModePosRel" oder "ModePosAdd"
        (MC_MoveAbsolute, MC_MoveRelative, MC_MoveAdditive) *)

    Deceleration:    REAL:=10.0;
    (* Value of the deceleration (always positive) (decreasing energy of the motor) [u/s2]
      - start moving with ModePosAbs or ModePosRel or ModePosAdd
        (MC_MoveAbsolute, MC_MoveRelative, MC_MoveAdditive)
      ------------------------------------------------------------------------
      Verzoegerung, (Einheit: [u/s2]) (nur positive Werte)
      - Freigabe mit "ModePosAbs" oder "ModePosRel" oder "ModePosAdd"
        (MC_MoveAbsolute, MC_MoveRelative, MC_MoveAdditive) *)

    Adjust:          MB_AXIS_ADJUST_DATA;
    (* -> adjusting of the element Position
      - only possible, when absolute positioning mode is active
      ------------------------------------------------------------------------
      -> Manuelle Verstellung (Vertippen) der Position
      - wirksam bei freigegebenem Positionierbetrieb (absolut) *)

END_STRUCT
END_TYPE
```

Abb.6-79: MB_AXIS_POSITIONING

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

MB_AXIS_VELOCITY Struktur

Kurzbeschreibung Die MB_AXIS_VELOCITY Struktur ist eine Unterstruktur von MB_AXIS_CONTROL_TYPE01 und enthält Sollwerte für den Geschwindigkeitsbetrieb.

Das folgende Bild zeigt die aktuelle MB_AXIS_VELOCITY Struktur:

ML_TechInterface.library

MB_AXIS_VELOCITY

```
    (*************** - setpoints of velocity mode                    ************)
    (*************** - Sollwerte Geschwindigkeitsregelung (Freilauf) ************)
TYPE MB_AXIS_VELOCITY :
STRUCT

    (* Value of the maximum velocity (Unit: e.g. in [rpm] depending on parametrization)
      - start moving with "ModeVel" (MC_MoveVelocity)
      Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
      ------------------------------------------------------------------------------
      Geschwindigkeits-Sollwert, (Einheit: z.B. in [U/min] je nach Parametrierung)
      - Freigabe mit "ModeVel" (MC_MoveVelocity)
      Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)
    Velocity:        REAL:=0.0;

    (* Value of the acceleration (Unit: e.g. in [rad/s2] depending on
       parametrization) (always positive)
      - start moving with "ModeVel" (MC_MoveVelocity)
      Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
      ------------------------------------------------------------------------------
      Beschleunigung, (Einheit: z.B. in [rad/s2] je nach Parametrierung) (immer positiv)
      - Freigabe mit "ModeVel" (MC_MoveVelocity)
      Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)
    Acceleration:    REAL:=10.0;

    (* Value of the deceleration (Unit: e.g. in [rad/s2] depending on
       parametrization) (always positive)
      - start moving with "ModeVel" (MC_MoveVelocity)
      Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
      ------------------------------------------------------------------------------
      Verzoegerung, (Einheit: z.B. in [rad/s2] je nach Parametrierung) (immer positiv)
      - Freigabe mit "ModeVel" (MC_MoveVelocity)
      Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)
    Deceleration:    REAL:=10.0;

    (* -> adjusting of the element Velocity
       -> Manuelle Verstellung (Vertippen) der Geschwindigkeit *)
    Adjust:          MB_AXIS_ADJUST_DATA;

END_STRUCT
END_TYPE
```

Abb.6-80: MB_AXIS_VELOCITY

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

MB_AXIS_VELOCITY Elemente

MB_AXIS_ADJUST_DATA

Kurzbeschreibung Die Struktur MB_AXIS_ADJUST_DATA ist eine Unterstruktur von MB_AXIS_POSITIONING und enthält binäre Einstellungs-Kommandos und Sollwerte (siehe auch Kap. "MB_AXIS_ADJUST_DATA" auf Seite 122).

MB_AXIS_SYNCHRONISATION Struktur

Kurzbeschreibung Die MB_AXIS_SYNCHRONISATION Struktur ist eine Unterstruktur von MB_AXIS_CONTROL_TYPE01 und enthält Sollwerte und Einstellungen für den Synchronisationsbetrieb. Das folgende Bild zeigt die aktuelle MB_AXIS_SYNCHRONISATION Struktur:

ML_TechInterface.library

```
TYPE MB_AXIS_SYNCHRONISATION :
STRUCT
    (*************** - setpoint of synchronous motion ******************)

    InputRevolution: UINT:=1;
    (* Electronic gear input revolution
      - start moving with "ModeSyncPhase", "ModeSyncVel", "ModeSyncCam" or "ModeSyncProfile"
      - RatioNumerator for MC_GearIn, MB_GearInPos, MC_CamIn, MB_MotionProfile
      Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
      ------------------------------------------------------------------------------
      Elektronisches Getriebe, Eingangsumdrehungen
      - Freigabe mit "ModeSyncPhase", "ModeSyncVel", "ModeSyncCam" oder "ModeSyncProfile"
      - RatioNumerator fuer MC_GearIn, MB_GearInPos, MC_CamIn, MB_MotionProfile
      Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)

    OutputRevolution: UINT:=1;
    (* Electronic gear output revolution
      - start moving with "ModeSyncPhase", "ModeSyncVel", "ModeSyncCam" or "ModeSyncProfile"
      - RatioDenominator for MC_GearIn, MB_GearInPos, MC_CamIn, MB_MotionProfile
      Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
      ------------------------------------------------------------------------------
      Elektronisches Getriebe, Ausgangsumdrehungen
      - Freigabe mit "ModeSyncPhase", "ModeSyncVel", "ModeSyncCam" oder "ModeSyncProfile"
      - RatioDenominator fuer MC_GearIn, MB_GearInPos, MC_CamIn
      Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)

    FineAdjust:    REAL:=0.0;
    (* Master fine adjust
      - start moving with "ModeSyncPhase", "ModeSyncVel", "ModeSyncCam" or "ModeSyncProfile"
      - MasterFineadjust for MC_GearIn, MB_GearInPos, MC_CamIn, MB_MotionProfile
      Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
      ------------------------------------------------------------------------------
      Getriebe-Feinabgleich
      - Freigabe mit "ModeSyncPhase", "ModeSyncVel", "ModeSyncCam" oder "ModeSyncProfile"
      - MasterFineadjust fuer MC_GearIn, MB_GearInPos, MC_CamIn
      Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)

    CamShaftDistance: REAL:=360.0;
    (* Cam Shaft Distance
      - start moving with "ModeSyncCam"
      - CamShaftDist for MC_CamIn
      Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
      ------------------------------------------------------------------------------
      Kurvenscheibe, Hub
      - Freigabe mit "ModeSyncCam"
      - CamShaftDist fuer MC_CamIn
      Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)
```

Abb.6-81: Struktur (Teil 1 von 3) MB_AXIS_SYNCHRONISATION

```
    PhaseOffset:   REAL:=0.0; (* Phase offset between master and slave axis
      - start moving with "ModeSyncPhase", "ModeSyncCam" or "ModeSyncProfile"
      - PhaseShift for MB_PhasingSlave
      Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
      ------------------------------------------------------------------------------
      Winkelversatz (Lagesollwert additiv)
      - Freigabe mit "ModeSyncPhase", "ModeSyncCam" oder "ModeSyncProfile"
      - PhaseShift fuer MB_PhasingSlave
      Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)

    PhaseOffsetVel: REAL:=1.0;
    (* Velocity for the phase offset
      - start moving with "ModeSyncPhase", "ModeSyncCam" or "ModeSyncProfile"
      - Velocity for MB_PhasingSlave
      ------------------------------------------------------------------------------
      Aufsynchronisier-Geschwindigkeit
      - Freigabe mit "ModeSyncPhase", "ModeSyncCam" oder "ModeSyncProfile"
      - Velocity fuer MB_PhasingSlave *)

    PhaseOffsetAcc: REAL:=100.0;
    (* Acceleration for the phase offset
      - start moving with "ModeSyncPhase", "ModeSyncCam" or "ModeSyncProfile"
      - PhaseShift for MB_PhasingSlave
      ------------------------------------------------------------------------------
      Aufsynchronisier-Beschleunigung
      - Freigabe mit "ModeSyncPhase", "ModeSyncCam" oder "ModeSyncProfile"
      - PhaseShift fuer MB_PhasingSlave *)

    SyncDirection: MC_SYNC_DIRECTION:= SHORTEST_WAY;
    (* Synchronisation direction
      - values 'SHORTEST_WAY'; 'CATCH_UP' (always forward); 'SLOW_DOWN' (see RIL_CommonTypes.lib)
      - start moving with "ModeSyncPhase", "ModeSyncCam" or "ModeSyncProfile"
      - SyncMode for MB_GearInPos, MC_CamIn, MB_MotionProfile
      ------------------------------------------------------------------------------
      Aufsynchronisier-Richtung
      - Werte 'SHORTEST_WAY'; 'CATCH_UP' (positiv); 'SLOW_DOWN' (siehe RIL_CommonTypes.lib)
      - Freigabe mit "ModeSyncPhase", "ModeSyncCam" oder "ModeSyncProfile"
      - SyncMode fuer MB_GearInPos, MC_CamIn, MB_MotionProfile *)

    StartMode:     MC_START_MODE:= ABSOLUTE;
    (* Start mode of the synchronisation
      - values 'ABSOLUTE'; 'RELATIVE' (see RIL_CommonTypes.lib)
      - start moving with "ModeSyncPhase", "ModeSyncCam" or "ModeSyncProfile"
      - StartMode for MB_GearInPos, MC_CamIn, MB_MotionProfile
      ------------------------------------------------------------------------------
      Aufsynchronisier-Mode
      - Werte: 'ABSOLUTE'; 'RELATIVE' (siehe RIL_CommonTypes.lib)
      - Freigabe mit "ModeSyncPhase", "ModeSyncCam" oder "ModeSyncProfile"
      - StartMode fuer MB_GearInPos, MC_CamIn, MB_MotionProfile *)
```

Abb.6-82: Struktur (Teil 2 von 3) MB_AXIS_SYNCHRONISATION

ML_TechInterface.library

```
    CamTableID:    MC_CAM_ID:= CAM_TABLE_1;
    (* Number of the selected cam table
      - values 'CAM_TABLE_1'; 'CAM_TABLE_2' ... 'CAM_TABLE_8' (see RIL_CommonTypes.lib)
      - start moving with "ModeSyncCam"
      - CamTableID for MC_CamIn
      Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
      ------------------------------------------------------------------------
      Nummer Stuetzpunkt-Tabelle
      - Werte: 'CAM_TABLE_1'; 'CAM_TABLE_2' ... 'CAM_TABLE_8' (siehe RIL_CommonTypes.lib)
      - Freigabe mit "ModeSyncCam"
      - CamTableID fuer MC_CamIn
      Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)

    Master:    AXIS_REF    := (AxisNo := NO_OBJECT, CntrlNo := LOCAL_CNTRL);
    (* -> AXIS_REF of the master axis
      - "ModeSyncPhase", "ModeSyncVel", "ModeSyncCam", "ModeSyncProfile", "ModeFlexProfile"
      - Master for MC_GearIn, MB_GearInPos, MC_CamIn, MB_MotionProfile, ML_FlexProfile
      Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
      ------------------------------------------------------------------------
      -> AXIS_REF der Leitachse
      - "ModeSyncPhase", "ModeSyncVel", "ModeSyncCam", "ModeSyncProfile", "ModeFlexProfile"
      - Master fuer MC_GearIn, MB_GearInPos, MC_CamIn, MB_MotionProfile, ML_FlexProfile
         Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)


    (*************** Extensions for profile operation modes ******************)
    (************* Erweiterungen für die Profil Betriebsarten *****************)
    Profile:     ML_AXIS_PROFILE;
    (* Extensions for the operating modes MotionProfile and FlexProfile
       ------------------------------------------------------------------------
       Erweiterungen füt die Betriebsarten MotionProfile und FlexProfile *)

    (*************** adjusting of synchronisation mode values ******************)
    (*************** Vertippen im Synchronbetrieb             ******************)

    AdjFineAdjust:   MB_AXIS_ADJUST_DATA; (* adjusting | vertippen
      arAxisCntrl[].SyncMode.FineAdjust *)

    AdjCamShaftDistance: MB_AXIS_ADJUST_DATA; (* adjusting | vertippen
      arAxisCntrl[].SyncMode.CamShaftDistance *)

    AdjPhaseOffset:  MB_AXIS_ADJUST_DATA; (* adjusting | vertippen
      arAxisCntrl[].SyncMode.PhaseOffset *)

END_STRUCT
END_TYPE
```

Abb.6-83: Struktur (Teil 3 von 3) MB_AXIS_SYNCHRONISATION

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

MB_AXIS_SYNCHRONISATION Elemente

ML_AXIS_PROFILE

Kurzbeschreibung Die ML_AXIS_PROFILE-Struktur ist eine Unterstruktur von Abb. 6-81 "Struktur (Teil 1 von 3) MB_AXIS_SYNCHRONISATION" auf Seite 128 und enthält

ML_TechInterface.library

Sollwerte und Einstellungen für den Synchronisationsbetrieb. Das folgende Bild zeigt die aktuelle ML_AXIS_PROFILE Struktur:

```
TYPE ML_AXIS_PROFILE :
STRUCT
    (* Extensions for the operating modes MotionProfile and FlexProfile
       ------------------------------------------------------------------------
       Erweiterungen füt die Betriebsarten MotionProfile und FlexProfile *)

    SetSelection:  UINT:= 0;
    (* Number of the selected motion profile set 0 or 1
      - SetSelection for MB_MotionProfile, ML_FlexProfile
      Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
      ------------------------------------------------------------------------
      Anwahl des Satzes 0 oder 1 des Bewegungsprofils
      - SetSelection fuer MB_MotionProfile, ML_FlexProfile
         Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)

    RelativePositioning: BOOL:= FALSE;
    (* position data processing
      - FALSE: absolute,  TRUE: relative
      - RelativePositioning for MB_MotionProfile
      ------------------------------------------------------------------------
      Verarbeitung der Positionierdaten
      - FALSE: absolut,  TRUE: relativ
      - RelativePositioning fuer MB_MotionProfile *)

    (* All following: Extensions for the operating mode FlexProfile
      - all inputs have the same name as the inputs of ML_FlexProfile
      ------------------------------------------------------------------------
      Erweiterungen füt die Betriebsart FlexProfile
      - alle Eingängen haben die gleichen Namen wie die Eingänge von ML_FlexProfile *)

    UseSwitchingPosition: BOOL:=FALSE;
    (* TRUE: the switching to a different profile is done, when the master axis reaches
      the position of the input "SwitchingPosition".
      ------------------------------------------------------------------------
      TRUE: Umschaltung auf ein anderes Profil erst wenn die Leitachse die am Eingang
      "SwitchingPosition" angelegte Position erreicht. *)

    SwitchingPosition: REAL:=0.0;
    (* SwitchingPosition, where the switching to a different profile should be done.
      ------------------------------------------------------------------------
      Umschaltposition bzw. Anfangsposition, an der die Umschaltung auf ein anderes
      Profil bzw. der Profilstart erfolgen soll. *)
```

Abb.6-84: Struktur (Teil 1 von 2) ML_AXIS_PROFILE

ML_TechInterface.library

```
    (* Describes the interpretation of the profile beginning in case of a profile
       start or change of a profile
      - Values: 'SLAVE_ORIGIN_MASTER_ORIGIN'; 'SLAVE_CURRENT_MASTER_CURRENT',
          'SLAVE_ORIGIN_MASTER_CURRENT', 'SLAVE_CURRENT_MASTER_ORIGIN' (see ML_PLCopen.lib)
      ------------------------------------------------------------------------
      Beschreibt wie bei einem Profilstart oder Profilwechsel der Ursprung
      des Profils interpretiert werden soll
      - Werte: 'SLAVE_ORIGIN_MASTER_ORIGIN'; 'SLAVE_CURRENT_MASTER_CURRENT',
          'SLAVE_ORIGIN_MASTER_CURRENT', 'SLAVE_CURRENT_MASTER_ORIGIN' (siehe ML_PLCopen.lib) *)
    ProfileEntry:      MB_PROFILE_START_POINT := SLAVE_ORIGIN_MASTER_ORIGIN;

    (* Offset of the master axis | Offset der Leitachse *)
    MasterOffset: REAL := 0.0;

  (* Offset of the following axis | Offset der Folgeachse *)
    SlaveOffset: REAL := 0.0;

    (* Synchronisation type
      - Values: 'SYNC_RAMPIN_SHORTEST_WAY'; 'SYNC_RAMPIN_CATCH_UP',
          'SYNC_RAMPIN_SLOW_DOWN', 'SYNC_DIRECT' (see ML_PLCopen.lib)
      ------------------------------------------------------------------------
      Aufsynchronisier-Art
      - Werte: 'SYNC_RAMPIN_SHORTEST_WAY'; 'SYNC_RAMPIN_CATCH_UP',
       'SYNC_RAMPIN_SLOW_DOWN', 'SYNC_DIRECT' (siehe ML_PLCopen.lib) *)
  SyncType:    MB_SYNC_TYPE:= SYNC_RAMPIN_SHORTEST_WAY;

    (* Execution type
      - Values: 'EXECUTE_CYCLIC'; 'EXECUTE_SINGLE' (see ML_PLCopen.lib)
      ------------------------------------------------------------------------
      Abarbeitungstyp
      - Werte: 'EXECUTE_CYCLIC'; 'EXECUTE_SINGLE' (siehe ML_PLCopen.lib) *)
    ExecutionMode:    MB_EXECUTION_MODE := EXECUTE_CYCLIC;

END_STRUCT
END_TYPE
```

Abb.6-85: Struktur (Teil 2 von 2) ML_AXIS_PROFILE

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_AxisInterface.lib für IndraMotion MLD

Abhängigkeit MX_ImcInterface.lib für IndraMotion MLD

System IndraMotion MLC/IndraMotion MLD

6.8.4 MB_AXIS_STATUS_TYPE02 Struktur

MB_AXIS_STATUS_TYPE02

Kurzbeschreibung Auf die MB_AXIS_STATUS_TYPE02 Datenstruktur wird über die **arAxisCtrl_gb[ ]** Instanz zugegriffen; sie enthält die gleichen Elemente wie die MB_AXIS_STATUS_TYPE01 Datenstruktur:

```
TYPE MB_AXIS_STATUS_TYPE02 EXTENDS MB_AXIS_STATUS_TYPE01:

STRUCT
   (********* Outputs of the following axis - from motion control to logic control **********)

    (* Pointer to AxisData of this axis | Zeiger auf AxisData dieser Achse
    - Type .PtrAxisData^. for the intellisense menu | Eingabe .PtrAxisData^. fuer Auswahlmenue*)
    //PtrAxisData:  POINTER TO ML_AXISDATA_SM:=0;

    (* status of the axis | Achs-Status *)
    //Admin:   MB_AXIS_ADMIN_STATUS_TYPE02;

    (* axis diagnosis information, error, warning....
    Diagnose Informationen der Achse, Fehler, Warnung.... *)
    //Diag:    MB_AXIS_DIAGNOSIS;

    (* Status operating mode positioning (absolute, rel. and add.)
    Statusmeldungen Positionierbetrieb (absolut, relativ und additiv) *)
    //PosMode: MB_AXIS_POS_STATUS;

    (* Status operating mode velocity control
    Statusmeldungen Geschwindigkeitsregelung (Freilauf) *)
    //VelMode: MB_AXIS_VEL_STATUS;

    (* Status operating mode synchronisation (phase, vel, and cam)
    Statusmeldungen Synchronbetriebsarten (Winkelsync., Geschwindigkeitssync. und Kurvensch.)*)
    //SyncMode: MB_AXIS_SYNC_STATUS;

(* add your extensions below here, keep the elements above
   Erweiterungen hier einfuegen, Elemente oberhalb nicht veraendern *)

(* user extensions | Anwender Erweiterungen *)
END_STRUCT

END_TYPE
```

Abb.6-86: MB_AXIS_STATUS_TYPE02

6.8.5 MB_AXIS_CONTROL_TYPE02 Struktur

MB_AXIS_CONTROL_TYPE02

Kurzbeschreibung Auf die MB_AXIS_CONTROL_TYPE02 Datenstruktur wird über die **arAxisCtrl_gb[ ]** Instanz zugegriffen; sie enthält folgende Elemente:

```
TYPE MB_AXIS_CONTROL_TYPE02 EXTENDS MB_AXIS_CONTROL_TYPE01:

STRUCT
  //Admin:      MB_AXIS_ADMINISTRATION; (* Administration of the axis | Verwaltung der Achse *)

  //PosMode:    MB_AXIS_POSITIONING; (* operating mode positioning (absolute, rel. and additive)
  //Positionierbetrieb (absolut, relativ und additiv) *)

  //VelMode:    MB_AXIS_VELOCITY; (* operating mode velocity control | Geschwindigkeitsregelung *)

  //SyncMode:   MB_AXIS_SYNCHRONISATION; (* operating mode synchronisation (phase, vel, and cam)
  //Synchronbetriebsarten (Winkelsync., Geschwindigkeitssync. und Kurvenscheibe) *)

  CoordMode: MB_AXIS_COORDINATED; (* Coordianted motion *)

END_STRUCT

END_TYPE
```

Abb.6-87: Struktur MB_AXIS_CONTROL_TYPE02

ML_TechInterface.library



Die MB_AXIS_CONTROL_TYPE02 Datenstruktur wird von der Datenstruktur MB_AXIS_CONTROL_TYPE01 abgeleitet und enthält deswegen ebenso die Elemente:

- Admin
- PosMode
- VelMode
- SyncMode

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC

System IndraMotion MLC

MB_AXIS_CONTROL_TYPE02 Elemente

MB_AXIS_COORDINATED Struktur

Kurzbeschreibung Die MB_AXIS_COORDINATED Struktur ist eine Unterstruktur von MB_AXIS_CONTROL_TYPE02 und enthält die Unterstruktur AXIS_GROUP_REF.

Das folgende Bild zeigt die aktuelle MB_AXIS_COORDINATED Struktur:

```
        (* Command values multi-axis motions|Sollwerte Mehrachsbewegungen *)
TYPE MB_AXIS_COORDINATED :
STRUCT

    AxisGroupRef:        AXIS_GROUP_REF;  (* reference of kinematics | Referenz der Kinematik *)

END_STRUCT
END_TYPE
```

Abb.6-88: MB_AXIS_COORDINATED

6.9 IMC-Interface Strukturen

6.9.1 Überblick

Die Bibliotheken ML_ImcInterface für IndraMotion MLC-Steuerungen und das MX_ImcInterface für IndraMotion MLD-M-Antriebe bieten ein einfach zu verwendendes Interface für jedes System. Das Interface wird als IMC-Interface bezeichnet - (I)ndra (M)otion (C)ontrol Interface. Das IMC-Interface bietet die folgenden Funktionen, wenn es als Interface für das IndraMotion MLC- oder IndraMotion MLD-D-System verwendet wird:

- wählt IndraMotion MLC (IndraMotion MLD-M) Betriebsmodus, P0 (Phase 0), P2 (Parametermodus), BB (Betriebsmodus)
- führt den Befehl "Clear All Errors" (alle Fehler löschen) aus
- führt "RetriggerOpMode" aus, um es den Benutzern zu ermöglichen, den gewählten Betriebsmodus erneut auszuführen. Der Benutzer kann die Steuerung für den Betriebsmodus Imc-Interface zurücksetzen, wenn der sercos Ring getrennt wird
- Bei einer Phasenumschaltung über eine externe Instanz (z.B. IndraWorks) setzt das Imc-Interface den Ausgang "PassiveMode" und der angewählte Modus wird nicht angesteuert. Mit "RetriggerOpMode" oder durch zurückschalten in den angewählten Modus wird der "PassiveMode" wieder verlassen

☞ Die in diesem Kapitel verwendeten Bibliotheksnamen sind systemabhängig.

IndraMotion MLC-Steuerungssysteme verwenden die Bibliothek ML_TechInterface.compiled-library.

IndraMotion MLD-M-Antriebssysteme verwenden die Bibliothek MX_ImcInterface.lib.

Die folgende Tabelle listet das Benutzerinterface mit der Datenstruktur des Steuerungsinterface auf:

Benutzer-Interface	Typ	Beschreibung
ImcCtrl[ ]	MB_Imc_Control_Type01	Steuerungsstruktur, die die Eingaben für Betriebsmodus, Fehler löschen und erneutes "Auslösen des Betriebsmodus" bereitstellt
ImcStatus[ ]	MB_Imc_Status_Type01	Statusstruktur, die Bestätigungen und Diagnoseinformationen für den Betriebsmodus umfasst

Abb.6-89: Steuerungsinterface - Datenstrukturen Benutzerinterface

Um das Imc-Interface benutzen zu können, muss nur das Programm "MB_ImcInterface" zyklisch aufgerufen werden, z.B. in der PlcTask. Bei der Verwendung von GAT bzw. GAT compact ist der Programmaufruf bereits integriert.

6.9.2 MB_IMC_Status_Type01

Kurzbeschreibung Auf die MB_IMC_Status_Type01 Datenstruktur wird über die Instanz **ImcStatus[ ]** zugegriffen. Sie enthält die folgenden Elemente:

```
TYPE MB_IMC_STATUS_TYPE01 :
  STRUCT
    InOperation: BOOL:= FALSE;    (* TRUE: ImcInterface is operating *)

    Admin:   MB_IMC_ADMIN_STATUS; (* -> Status of the administration of the control *)

    Diag:    MB_IMC_DIAGNOSIS;    (* -> Control diagnosis information, error, warning..*)

    PassiveMode: BOOL := FALSE;   (* TRUE: External source controlling operation mode  *)

  END_STRUCT
END_TYPE
```

Abb.6-90: Struktur MB_IMC_Status_Type01

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_ImcInterface.lib für IndraMotion MLD-M

Abhängigkeit MX_AxisInterface.lib für IndraMotion MLD-M

System IndraMotion MLC/IndraMotion MLD-M

6.9.3 MB_IMC_Control_Type01

Kurzbeschreibung Auf die MB_IMC_Control_Type01 Datenstruktur wird über die Instanz **ImcCtrl[ ]** zugegriffen. Sie enthält die folgenden Elemente:

ML_TechInterface.library

```
TYPE MB_IMC_CONTROL_TYPE01 :
  STRUCT

    Admin:    MB_IMC_ADMINISTRATION;  (* -> Administration of the control *)

    UserDiag: MB_IMC_USER_DIAGNOSTIC; (* -> User controlled Diagnositics  *)

  END_STRUCT
END_TYPE
```

Abb.6-91: Struktur MB_IMC_Control_Type01

Bibliothek ML_TechInterface.compiled-library für IndraMotion MLC/MX_ImcInterface.lib für IndraMotion MLD-M

Abhängigkeit MX_AxisInterface.lib für IndraMotion MLD-M

System IndraMotion MLC/IndraMotion MLD-M

6.10 HowTo: Typische Anwenderaktivitäten

6.10.1 Zugriff auf Achsdaten

Die folgenden Daten sind verfügbar (Name = jeweiliger Achsname):

- AxisData[Name.AxisNo] => Achsdatenstruktur: Statusdaten der Firmware und Zugriff auf Achsdatencontainer
- arAxisCtrl_gb[Name.AxisNo] => Steuerstruktur des Achs-Interface
- arAxisStatus_gb[Name.AxisNo] => Statusstruktur des Achs-Interface
- DV_Axis[Name.AxisNo] => Zugriff auf Direktvariablen (Funktionsvariablen)

Die AxisData-Struktur befindet sich bei den globalen Variablen der Bibliothek ML_Base.

6.10.2 Achse hinzufügen

Eine reale Achse kann im IndraWorks-Projekt durch Auswahl des Antriebs (z. B. IndraDrive) aus der Bibliothek und Ziehen auf den sercos Knoten hinzugefügt werden.

Für eine hinzugefügte Achse muss ggf. eine Initialisierung bestimmter Strukturelemente des AxisInterface vorgenommen werden, z. B. Zuordnung zur gewünschten Masterachse einstellen:

arAxisCtrl_gb[Achsname.AxisNo].SyncMode.Master := vAxis1;

6.10.3 Erhöhung der maximalen Achsanzahl für Achsnummern größer 16

Sind Achsen in Ihrem Projekt enthalten deren logische Achsnummer größer '16' ist, so müssen Sie den Wert der globalen Konstanten "uiMAX_AXIS_INDEX" entsprechend anpassen.

"uiMAX_AXIS_INDEX" muss mindestens so groß sein wie die größte logische Achsnummer des Projektes. Die Defaulteinstellung ist '16'.

Die Konstante "uiMAX_AXIS_INDEX" finden Sie am schnellsten über die IndraWorks Suchfunktion (<Strg>+<F>) oder über folgende Ordnerstruktur:

Application/UserConstGlobal

6.10.4 Achse entfernen/umbenennen

Eine vorhandene Achse kann im IndraWorks-Projektbaum unterhalb des sercos Knotens gelöscht bzw. umbenannt werden.

Wird eine Achse umbenannt muss der Zugriff über die Control- u. Statusstrukturen "arAxisCtrl_gb[geänderterAchsname.AxisNo]" und "arAxisStatus_gb[geänderterAchsname.AxisNo]" innerhalb des SPS-Programmes angepasst werden.

6.10.5 Task anlegen

Generell ist bereits eine PlcTask mit dem Programm "PlcProg" angelegt.

Im Projektbaum kann im Ordner Taskkonfiguration mittels "Hinzufügen" eine neue Task angelegt werden.

Es muss dann folgendes festgelegt werden:

- die Taskpriorität (Priorität fällt mit steigender Ziffer)
- der Tasktyp (zyklisch, freilaufend, ereignisgesteuert)
- das Aufrufintervall
- die zugehörige Programmorganisationseinheit "POU"
- Verwendung eines Watchdogs mit Zeitintervall

Die Kommandierung der Achsen erfolgt in der Regel in der "PlcTask" im angehängten Programm "PlcProg". In diesem Programm erfolgt auch der Aufruf des AchsInterface-Bausteins "TE_AxisInterface".

Ist in besonderen Einzelfällen, eine zeitkritischere Kommandierung der Achsen gewünscht, so kann die Kommandierung in einer anderen Task (z. B. ereignisgesteuerte MotionTask) erfolgen. In diesem Fall sollte der Aufruf des AchsInterface-Bausteins "TE_AxisInterface" in dieser Task erfolgen.

6.10.6 Achs-Interface Erweiterungen

Das Achs-Interface erlaubt fast beliebige Erweiterungen der Achs-Interface-Strukturen. Es können zusätzliche Unterstrukturen eingefügt und auch die vorhandenen Unterstrukturen erweitert werden.

Die folgenden Strukturelemente sind als Anwendererweiterungen in der AxisCtrl-Struktur beispielhaft programmiert:

- SetupMode: zusätzliche Unterstruktur in der AxisCtrl-Struktur
  - Enable: Freigabe Einrichtbetrieb
  - JogPlus: Tippen +
  - JogMinus: Tippen -
  - Vel: Tipp-/Referenzfahrgeschwindigkeit
  - Accel: Tipp/Referenzfahr(brems)beschleunigung
  - Homing: aktiviert ModeHoming|SetAbsRef|ModePosAbs abhängig vom Achstyp/-konfiguration
- SyncMode: Erweiterung der vorhandenen Unterstruktur in der AxisCtrl-Struktur
  - AngleOffset: Winkelverschiebung Tabellenanfang -> der Parameter A-0-2730 wird mit diesem Wert beschrieben
  - PhaseShift: Offset zur Leitachse vor dem Getriebe (PhaseOffset: Offset zur Leitachse nach dem Getriebe)
  - PhaseShiftVel: Geschwindigkeit zur Verstellung des PhaseShift
  - PhaseShiftAcc: Beschleunigung zur Verstellung des PhaseShift

  - SyncVelocity: Aufsynchronisiergeschwindigkeit -> der Parameter A-0-2790 wird mit diesem Wert beschrieben
  - SyncAcceleration: Aufsynchronisierbeschleunigung -> der Parameter A-0-2791 wird mit diesem Wert beschrieben

Die folgenden Strukturelemente sind als Anwendererweiterungen in der Axis-Status-Struktur beispielhaft programmiert:

- SetupMode: zusätzliche Unterstruktur in der AxisStatus-Struktur
  - EnableAck: Einrichtbetrieb ist aktiv
  - Homingack: Homing ist abgeschlossen
- SyncMode: Erweiterung der vorhandenen Unterstruktur in der AxisStatus-Struktur
  - AngleOffsetDone: Übertragung Winkelverschiebung Tabellenanfang ist abgeschlossen -> der Parameter A-0-2730 wurde beschrieben
  - PhasingMasterDone: der PhaseShift wurde verfahren
  - SyncVelocityDone: Übertragung Aufsynchronisiergeschwindigkeit ist abgeschlossen -> der Parameter A-0-2790 wurde übertragen
  - SyncAccelerationDone: Übertragung Aufsynchronisierbeschleunigung ist abgeschlossen -> der Parameter A-0-2791 wurde übertragen

Der Code zu diesen Erweiterungen ist in den Aktionen des Bausteins TE_AxisInterface() im Ordner "AxisInterface" zu finden. Die dazugehörigen Strukturen sind in den Ordnern "AxisInterface/Type/Control" und "AxisInterface/Type/Status" zu finden.

Es können eigene Erweiterungen hinzugefügt werden (siehe dazu Kap. 6.7 "Achs-Interface Anwender-Erweiterung" auf Seite 103).

Bei der Einführung neuer Fehlercodes kann durch Erweiterung der Struktur "arAXIS_DIAG_MESSAGES" in der Variablenliste UserGLobalConst im Ordner "\AxisInterface" der Fehlernummer auch eine Diagnosemeldung für die SPS-Visualisierung zugeordnet werden.

6.10.7 Robot Control

Um Achsen für koordinierte Bewegungen ansteuern zu können, muss für das TechInterface der Funktionsbaustein MB_AxisInterfaceType**02** verwendet werden. Hierzu muss folgendes im GAT^{compact} angepasst werden:

- Die Struktur "TE_AXIS_CONTROL_Type01" muss in "TE_AXIS_CONTROL_TYPE02" umbenannt werden und um die Strukturvariable "CoordModeCtrl" vom Typ "MB_AXIS_COORDINATED" erweitert werden

Umbenennen der Struktur TE_AXIS_CONTROL_Type01 nach TE_AXIS_CONTROL_Type02:

```
//former type: TYPE TE_AXIS_CONTROL_TYPE01:
TYPE TE_AXIS_CONTROL_TYPE02:  // rename for RobotControl
STRUCT
    Admin:      TE_AXIS_ADMINISTRATION; (* Administration of the axis | Verwaltung der Achse *)

    PosMode:    TE_AXIS_POSITIONING; (* operating mode positioning (absolute, rel. and additive)
      Positionierbetrieb (absolut, relativ und additiv) *)

    VelMode:    TE_AXIS_VELOCITY; (* operating mode velocity control | Geschwindigkeitsregelung *)

    SyncMode:   TE_AXIS_SYNCHRONISATION; (* operating mode synchronisation (phase, vel, and cam)
      Synchronbetriebsarten (Winkelsync., Geschwindigkeitssync. und Kurvenscheibe) *)

    (* add your extensions below here, keep the elements above
       Erweiterungen hier einfuegen, Elemente oberhalb nicht veraendern *)
```

```
    (* user extensions | Anwender Erweiterungen *)

      // Jog/Homing functionality | Tipp/Einrichtfunktionen
    SetupMode:  TE_AXIS_SETUP_MODE;

    CoordModeCtrl:  MB_AXIS_COORDINATED;    // inserted for RobotControl

END_STRUCT
END_TYPE
```

- Umbenennen der "TE_AXIS_STATUS_Type01" Struktur in "TE_AXIS_STATUS_TYPE02"

Umbenennen der Struktur TE_AXIS_STATUS_Type01 nach TE_AXIS_STATUS_TYPE02:

```
    (********* Outputs of the following axis - from motion control to logic control **********)
    // former type: TYPE TE_AXIS_STATUS_TYPE01:
TYPE TE_AXIS_STATUS_TYPE02:     // Rename for RobotControl
  STRUCT
    (********* Outputs of the following axis - from motion control to logic control **********)

    PtrAxisData:  POINTER TO ML_AXISDATA_SM;
     (* Pointer to AxisData of this axis | Zeiger auf AxisData dieser Achse
      - Type .PtrAxisData^. for the intellisense menu | Eingabe .PtrAxisData^. fuer Auswahlmenue*)

    Admin:   TE_AXIS_ADMIN_STATUS; (* status of the axis | Achs-Status *)

    Diag:    TE_AXIS_DIAGNOSIS; (* axis diagnosis information, error, warning....
      Diagnose Informationen der Achse, Fehler, Warnung.... *)

    PosMode: TE_AXIS_POS_STATUS; (* Status operating mode positioning (absolute, rel. and add.)
      Statusmeldungen Positionierbetrieb (absolut, relativ und additiv) *)

    VelMode: TE_AXIS_VEL_STATUS; (* Status operating mode velocity control
      Statusmeldungen Geschwindigkeitsregelung (Freilauf) *)

    SyncMode: TE_AXIS_SYNC_STATUS; (* Status operating mode synchronisation (phase, vel, and cam)
      Statusmeldungen Synchronbetriebsarten (Winkelsync., Geschwindigkeitssync. und Kurvensch.)*)

    (* add your extensions below here, keep the elements above
       Erweiterungen hier einfuegen, Elemente oberhalb nicht veraendern *)
    (* user extensions | Anwender Erweiterungen *)
         (* Jog/Homing functionality | Tipp/Einrichtfunktionen *)
    SetupMode:  TE_AXIS_SETUP_MODE_STATUS;

    // AxisData for HMI (without data type "BIT")
      AxisData_HMI:    TE_AXIS_DATA_STATUS;

  END_STRUCT
END_TYPE
```

- Der Funktionsbaustein "TE_Axisinterface" muss mit dem neuen FB "MB_AxisInterfaceType02" erweitert werden. "TE_AxisInterface EXTENDS MB_AxisInterfaceType02"

Erweiterung TE_Axisinterface:

```
//former: FUNCTION_BLOCK TE_AxisInterface EXTENDS MB_AxisInterfaceType01
FUNCTION_BLOCK TE_AxisInterface EXTENDS MB_AxisInterfaceType02
(*#########################################################################################################
  #########################################################################################################

  General Header
  ---------------------------------------------------------------------------------------------------
  Shortdescription       : AxisInterface Example
```

- Die Methode "mInitExtensions()": Der zusätzliche Eingang "CoordModeCtrl" des FB "MB_AxisInterfaceType02" muss initialisiert werden

CoordModeCtrl initialisieren:

```
SUPER^.mInitExtension(index);

IF NOT bFirstInit THEN
     // User: Assigment of user defined axis interface inputs
     SetupMode      REF= arAxisCtrl_gb[index].SetupMode;
```

ML_TechInterface.library

```
    // inserted for RobotControl
    CoordModeCtrl  REF= arAxisCtrl_gb[index].CoordModeCtrl;
    // User: Assigment of user defined axis interface outputs
    SetupModeAck REF= arAxisStatus_gb[index].SetupMode;
    AxisData_HMI REF= arAxisStatus_gb[index].AxisData_HMI;

    // ================================================================
    // Do not touch the following lines
    // ================================================================

    // Set the references for the AxisInterface FB
    AdminCtrlExt  REF= arAxisCtrl_gb[index].Admin;
```

- Die Deklaration für die beiden Strukturvariablen "arAxisCtrl_gb" und "arAxisStatus_gb" muss auf die entsprechenden Type02-Strukturen geändert werden

Strukturvariablen anpassen:

```
VAR_GLOBAL
   // Axis interface: command structure: define as RETAIN, to keep the values
   // Note: on the L40.2 control, the RETAIN memory is slower then normal VAR, so define
   //       this structure only as RETAIN when necessary
   //  arAxisCtrl_gb: ARRAY [1..MAX_AXIS_NUM] OF MB_AXIS_CONTROL;

   // Axis interface: command structure
   //former: arAxisCtrl_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_CONTROL_TYPE01;
     arAxisCtrl_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_CONTROL_TYPE02;
END_VAR

VAR_GLOBAL
//---------------------------------- Machine status variables ----------------------------

  // Axis interface: status structure
  //former: arAxisStatus_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_STATUS_TYPE01;
  arAxisStatus_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_STATUS_TYPE02;
  bRemoteOn_gb   : BOOL := TRUE;  // TRUE: Internal IL visu is active
                                  // FALSE: PlcProg is active
END_VAR
```



Die Strukturen TE_AXIS_CONTROL_TYPE02, TE_AXIS_STATUS_TYPE02 befinden sich im Ordner: "AxisInterface\Type".

Die Strukturvariablen arAxisCtrl_gb, arAxisStatus_gb befinden sich in der globalen Variablenliste "UserVarGlobal".

Kinematik-Interface

Aufbauend auf das Achs-Interface steht das Kinematik-Interface als optionales Plug-in zur Verfügung. Über das Kinematik Interface können Kinematiken analog zu Achsen kommandiert werden. Fügen Sie hierzu das Kinematik Interface über den GAT Wizard als Plug-in hinzu.

Die Dokumentation des Kinematik Interface ist in dem Plug-in Konfigurationsordner enthalten.

Den Link zum Konfigurationsordner können Sie über den GAT Wizard wie folgt finden:

- Im GAT Wizard den Ordner "Plug-ins" selektieren
- Im Wizard Bedienfeld ist jetzt der Link "Plug-ins Konfigurationsordner öffnen" sichtbar

7 Kinematik-Interface

7.1 Einführung

Das **Kinematik-Interface** bündelt und erweitert Robot-Control-Funktionalitäten. Es stellt das Gegenstück zum **Achs-Interface** dar und dient der einfachen und komfortablen Bedienung einer Kinematik. So ist es möglich in kürzester Zeit gut strukturierte Applikationsprogramme für Robot-Control-Applikationen zu entwickeln.

Das Kinematik-Interface kann sowohl für Robot-Control V1 wie auch für Robot-Control V2 verwendet werden. Einige Betriebsarten sind allerdings nur für eine Robot-Control-Version zulässig. Nähere Details siehe Kap. 7.2.4 "Betriebsarten" auf Seite 182

Systemvoraussetzung

Für das Kinematik-Interface ist folgende Hard- und Software notwendig:

- Software: IndraMotion MLC 13VRS
- Firmware: IndraMotion ML* 13VRS
- *CML Steuerung:*
  - IndraControl L25 oder
  - IndraControl L45 oder
  - IndraControl L65
- Template: Achs-Interface Beispielprojekt oder GAT

Erstinbetriebnahme

Voraussetzung für das Kinematik-Interface ist, dass das Achs-Interface eingebunden und auf koordinierte Bewegungen (Robot-Control) angepasst wurde. Ist dies bereits geschehen, so können die folgenden Arbeitsschritte übersprungen werden.

Wurde das Projekt durch das Achs-Interface Beispielprojekt, GAT^{compact}, GAT mit zentraler Zustandsmaschine oder GAT mit dezentraler Zustandsmaschine erstellt sind die folgenden Schritte durchzuführen.

Achs-Interface Beispielprojekt

1. Die Struktur "TE_AXIS_CONTROL_Type**01**" muss in "TE_AXIS_CONTROL_TYPE**02**" umbenannt werden und um die Strukturvariable "CoordModeCtrl" vom Typ "MB_AXIS_COORDINATED" erweitert werden

   CoordModeCtrl: MB_AXIS_COORDINATED;

2. Die Struktur "TE_AXIS_STATUS_Type**01**" muss in "TE_AXIS_STATUS_TYPE**02**" umbenannt werden

   TE_AXIS_STATUS_TYPE02

3. Die Deklaration für die beiden Strukturvariablen "arAxisCtrl_gb" und "arAxisStatus_gb" muss auf die entsprechenden Type02-Strukturen geändert werden

   arAxisCtrl_gb: ARRAY [ ] OF TE_AXIS_CONTROL_TYPE02;
   arAxisStatus_gb: ARRAY [ ] OF TE_AXIS_STATUS_TYPE02;

4. Die Basisklasse des Funktionsbausteins "TE_Axisinterface" auf "MB_AxisInterfaceType**02**" ändern

Kinematik-Interface

TE_AxisInterface EXTENDS MB_AxisInterfaceType02

5. In der Methode "TE_Axisinterface.mInitExtensions()" den zusätzlichen Eingang "CoordModeCtrl" des FB "MB_AxisInterfaceType02" initialisieren

   CoordModeCtrl REF= arAxisCtrl_gb[index].CoordModeCtrl;

☞ Die Strukturen TE_AXIS_CONTROL_TYPE02, TE_AXIS_STATUS_TYPE02 befinden sich im Ordner: "AxisInterface\Type".

Die Strukturvariablen arAxisCtrl_gb, arAxisStatus_gb befinden sich in der globalen Variablenliste "UserVarGlobal".

GAT^{compact}

1. Die Struktur "TE_AXIS_CONTROL_Type**01**" muss in "TE_AXIS_CONTROL_TYPE**02**" umbenannt werden und um die Strukturvariable "CoordModeCtrl" vom Typ "MB_AXIS_COORDINATED" erweitert werden

   CoordModeCtrl: MB_AXIS_COORDINATED;

2. Die Struktur "TE_AXIS_STATUS_Type**01**" muss in "TE_AXIS_STATUS_TYPE**02**" umbenannt werden

   TE_AXIS_ STATUS_TYPE02

3. Die Deklaration für die beiden Strukturvariablen "arAxisCtrl_gb" und "arAxisStatus_gb" muss auf die entsprechenden Type02-Strukturen geändert werden

   arAxisCtrl_gb: ARRAY [ ] OF TE_AXIS_CONTROL_TYPE02;
   arAxisStatus_gb: ARRAY [ ] OF TE_AXIS_STATUS_TYPE02;

4. Die Basisklasse des Funktionsbausteins "TE_Axisinterface" auf "MB_AxisInterfaceType**02**" ändern

   TE_AxisInterface EXTENDS MB_AxisInterfaceType02

5. In der Methode "TE_Axisinterface.mInitExtensions()" den zusätzlichen Eingang "CoordModeCtrl" des FB "MB_AxisInterfaceType02" initialisieren

   CoordModeCtrl REF= arAxisCtrl_gb[ ].CoordModeCtrl;

6. Die "TE_GAT_AXIS"-Referenzen müssen auf die Type02-Strukturen angepasst werden.

   CTRL: REFERENCE TO TE_AXIS_CONTROL_TYPE02
   Status: REFERENCE TO TE_AXIS_STATUS_TYPE02

☞ Die Strukturen TE_AXIS_CONTROL_TYPE02, TE_AXIS_STATUS_TYPE02 befinden sich im Ordner: "GATcompact\AxisInterface\Type".

Die Strukturvariablen arAxisCtrl_gb, arAxisStatus_gb befinden sich in der globalen Variablenliste "UserVarGlobal".

Die Struktur TE_GAT_AXIS befindet sich im Ordner "GATcompact"

GAT mit zentraler Zustandsmaschine

1. Die Struktur "TE_AXIS_CONTROL_Type**01**" muss in "TE_AXIS_CONTROL_TYPE**02**" umbenannt werden und um die Strukturvariable "CoordModeCtrl" vom Typ "MB_AXIS_COORDINATED" erweitert werden

   CoordModeCtrl: MB_AXIS_COORDINATED;

2. Die Struktur "TE_AXIS_STATUS_Type**01**" muss in "TE_AXIS_STATUS_TYPE**02**" umbenannt werden

   TE_AXIS_STATUS_TYPE02

3. Die Basisklassen der Strukturen "TE_AXISINTERFACE_CONTROL" und "TE_AXISINTERFACE_STATUS" müssen ebenfalls entsprechend angepasst werden (Type01 -> Type02)

   TE_AXISINTERFACE_CONTROL EXTENDS TE_AXIS_CONTROL_TYPE02
   TE_AXISINTERFACE_STATUS EXTENDS TE_AXIS_STATUS_TYPE02

4. Die Basisklasse des Funktionsbausteins "TE_Axisinterface" auf "MB_AxisInterfaceType**02**" ändern

   TE_AxisInterface EXTENDS MB_AxisInterfaceType02

5. In der Methode "TE_Axisinterface.mInitExtensions()" den zusätzlichen Eingang "CoordModeCtrl" des FB "MB_AxisInterfaceType02" initialisieren

   CoordModeCtrl REF= arAxisCtrl_gb[ ].CoordModeCtrl;

☞ Die Strukturen TE_AXIS_CONTROL_TYPE02, TE_AXIS_STATUS_TYPE02 befinden sich im Ordner: "GAT\AxisInterface\Type".

Die Strukturen TE_AXISINTERFACE_CONTROL und TE_AXIS_STATUS_TYPE02 befinden sich im Ordner: "GAT\AxisInterface\Assignments"

GAT mit dezentraler Zustandsmaschine

1. Die Struktur "TE_AXIS_CONTROL_Type**01**" muss in "TE_AXIS_CONTROL_TYPE**02**" umbenannt werden und um die Strukturvariable "CoordModeCtrl" vom Typ "MB_AXIS_COORDINATED" erweitert werden

   CoordModeCtrl: MB_AXIS_COORDINATED;

2. Die Struktur "TE_AXIS_STATUS_Type**01**" muss in "TE_AXIS_STATUS_TYPE**02**" umbenannt werden

   TE_AXIS_STATUS_TYPE02

3. Die Deklaration für die beiden Strukturvariablen "arAxisCtrl_gb" und "arAxisStatus_gb" muss auf die entsprechenden Type02-Strukturen geändert werden

   arAxisCtrl_gb: ARRAY [ ] OF TE_AXIS_CONTROL_TYPE02;
   arAxisStatus_gb: ARRAY [ ] OF TE_AXIS_STATUS_TYPE02;

Kinematik-Interface

4. Die Basisklasse des Funktionsbausteins "TE_Axisinterface" auf "MB_AxisInterfaceType**02**" ändern

   TE_AxisInterface EXTENDS MB_AxisInterfaceType02

5. In der Methode "TE_Axisinterface.mInitExtensions()" den zusätzlichen Eingang "CoordModeCtrl" des FB "MB_AxisInterfaceType02" initialisieren

   CoordModeCtrl REF= arAxisCtrl_gb[ ].CoordModeCtrl;

6. Die Typen von "arAxisCtrl" und "arAxisStatus" müssen auf die Type02-Strukturen angepasst werden.

   arAxisCtrl: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_CONTROL_TYPE02;
   arAxisStatus: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_STATUS_TYPE02;

7. Die "TE_GAT_AXIS"-Referenzen müssen auf die Type02-Strukturen angepasst werden.

   CTRL: REFERENCE TO TE_AXIS_CONTROL_TYPE02
   Status: REFERENCE TO TE_AXIS_STATUS_TYPE02

☞ Die Strukturen TE_AXIS_CONTROL_TYPE02, TE_AXIS_STATUS_TYPE02 befinden sich im Ordner: "GAT\AxisInterface\Type".

Die Strukturvariablen arAxisCtrl_gb, arAxisStatus_gb befinden sich in der globalen Variablenliste "Global_Axis_Interfaces".

Die Typen arAxisCtrl und arAxisStatus befinden sich in "GAT \AxisInterface\TE_AxisInitAllAxisType01"

Die Struktur TE_GAT_AXIS befindet sich im Ordner "GAT\MachineUnits"

Integration Kinematik-Interface

Um das Kinematik-Interface zu integrieren kann wie folgt vorgegangen werden:

1. Starten Sie den GAT-Wizard über das Symbol  .
2. Wählen Sie den Knoten "Plugins"
3. Wählen Sie die Funktion "Plugin hinzufügen"
4. Wählen Sie das Plug-in "KinInterface" und bestätigen den Import mit **Übernehmen**

7.2 ML_KinTech.library

7.2.1 Einführung und Übersicht

Folgende Funktionsbausteine werden durch die ML_KinTech bereitgestellt:

Funktionsbaustein	Beschreibung
ML_KinematicsInitType01 (siehe Kap. "ML_KinematicsInitType01" auf Seite 171)	wird zur Initialisierung des Kinematik-Interfaces für jede Kinematik benutzt. Der Funktionsbaustein muss nur einmal beim Programmstart bzw. bei jeder Phasenumschaltung vom Parametrier- in den Betriebs-Modus aufgerufen werden
ML_KinematicsInterfaceType01 (siehe Kap. "ML_KinematicsInterfaceType01" auf Seite 174)	wird zur Konfiguration des Kinematik-Interfaces für jede Kinematik benutzt. Der Funktionsbaustein muss zyklisch aufgerufen werden, solange man sich im Betriebs-Modus befindet
ML_KinJogV1 (siehe Kap. "ML_KinJogV1" auf Seite 176)	dient zum Vertippen einzelner Achsen oder Achsgruppen einer Kinematik mit der Robot-Control V1.
ML_KinJog (siehe Kap. "ML_KinJog" auf Seite 178)	dient zum Vertippen einzelner Achsen oder Achsgruppen einer Kinematik mit der Robot-Control V2.

Abb.7-1: Funktionsbausteine der ML_KinTech

Das Kinematik-Interface wird als PlugIn zum IndraMotion Programmier-Template Achs-Interface zur Verfügung gestellt. Der strukturelle Aufbau und die Funktionsweise ähnelt daher der des Achs-Interfaces. Was bei dem Achs-Interface die Achse ist, ist bei dem Kinematik-Interface die Kinematik.

Die folgendes Abbildung zeigt die Datenstruktur des Kinematik-Interfaces:

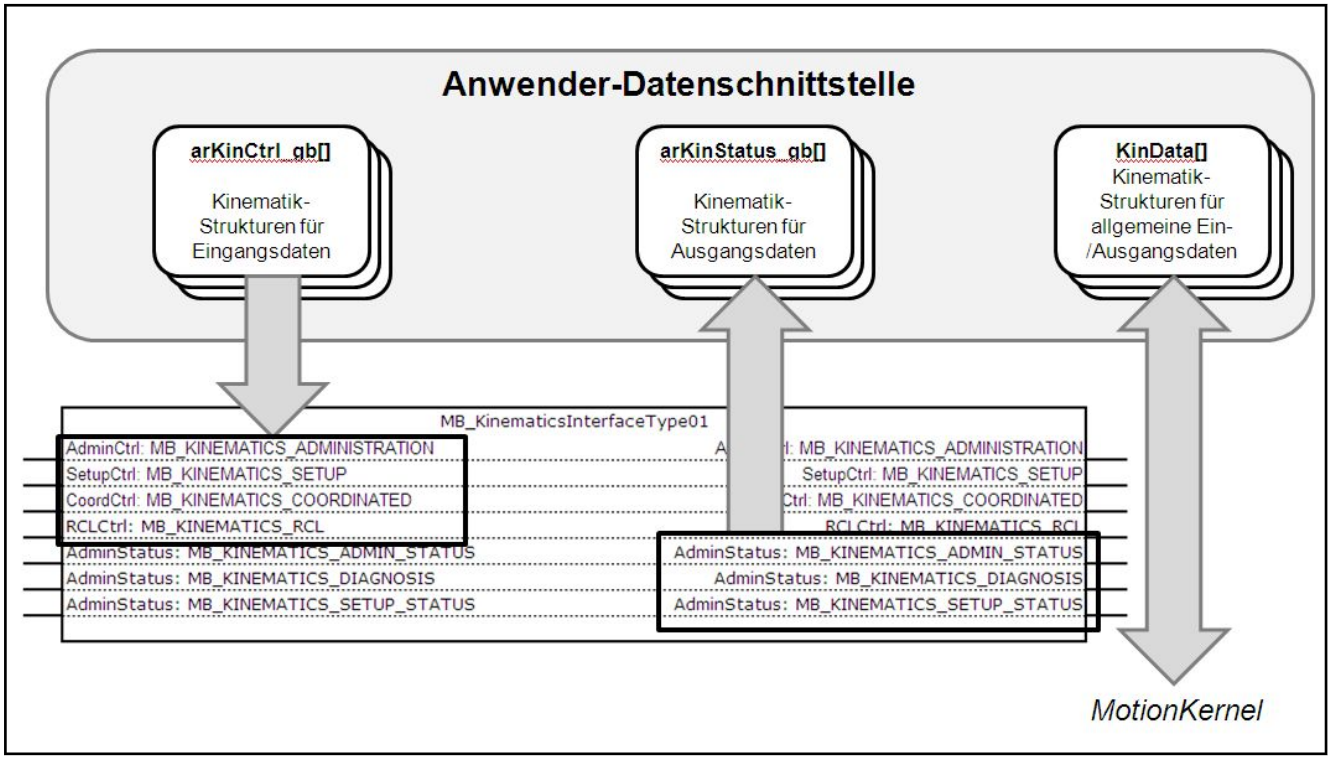


Abb.7-2: Kinematik-Interface - Datenstruktur des Interfaces

Kinematik-Interface

Die folgende Tabelle zeigt das Anwender-Interface zum Achs-Interface Datenstruktur:

Anwender-Interface	Typ	Beschreibung
arKinCtrl[ ]_gb	MB_KINEMATICS_CONTROL_TYPE01 (Anwender-Erweiterung: TE_KINEMATICS_CONTROL_TYPE01 MB_KINEMATICS_CONTROL_TYPE02 (Anwender-Erweiterung: TE_KINEMATICS_CONTROL_TYPE02	Steuerungsstruktur inklusive Sollwerte und Variablen zum Aktivieren der Betriebsarten
arKinStatus[ ]_gb	MB_KINEMATICS_STATUS_TYPE01 (Anwender-Erweiterung: TE_KINEMATICS_CONTROL_TYPE01) MB_KINEMATICS_STATUS_TYPE02 (Anwender-Erweiterung: TE_KINEMATICS_CONTROL_TYPE02)	Statusstruktur inklusive Diagnose-Info und Quittungen für die Betriebsarten
KinData[ ]		Status- und Steuerungsstruktur inklusive Istwerte und Status-Bits (wird durch die MotionControl in der Firmware bearbeitet)

Abb.7-3: Kinematik-Interface - Datenstruktur des Anwender-Interfaces

7.2.2 Kinematik-Interface-Strukturen

Überblick

Das Kinematik-Interface stellt eine Datenschnittstelle zur komfortablen Bedingung einer Kinematik zur Verfügung. Die Datenstrukturen zum Kinematik-Interface werden im folgendem Abschnitt beschrieben.

☞ Das Kinematik-Interface ist so vorbereitet, dass es durch den Anwender erweiterbar ist. Um diese Erweiterungen zu ermöglichen, ist es nötig, eigene Strukturen im Anwenderprojekt zu definieren, welche die Strukturen der Bibliothek erweitern. Die erweiterteten Strukturen sind durch den Präfix "TE_" gekennzeichnet. Wenn also eine Struktur, z. B. **TE_**KINEMATICS_ADMIN_STATUS heißt, ist diese eine erweiterte Struktur der **MB_**KINEMATICS_ADMIN_STATUS und Informationen zu der Struktur sind unter dem Namen **MB_**KINEMATICS_ADMIN_STATUS zu finden.

MB_KINEMATICS_CONTROL_TYPE01 Struktur

MB_KINEMATICS_CONTROL_TYPE01

Kurzbeschreibung Auf die MB_KINEMATICS_CONTROL_TYPE01-Datenstruktur wird über die arKinCtrl_gb[ ]-Instanz zugegriffen.

Sie enthält folgende Elemente:

MB_KINEMATICS_CONTROL_TYPE01

```
1  TYPE MB_KINEMATICS_CONTROL_TYPE01 :
2    STRUCT
3      Admin: MB_KINEMATICS_ADMINISTRATION;  (* Administration of the kinematics | Verwaltung der Kinematik *)
4
5      SetupMode: MB_KINEMATICS_SETUP;       (* Jog/Homing functionality | Tipp/Einrichtfunktionen  *)
6
7      CoordMode: MB_KINEMATICS_COORDINATED; (* operating mode coordinated motion (linear, circular, direct; absolute, relative) |
8                                               Koordinierbetriebsart (Linear, Zirkular, PTP) *)
9
10     RCLMode: MB_KINEMATICS_RCL;           (* operating mode robot control language (RCL) |
11                                              RCL-Betriebsart*)
12   END_STRUCT
13 END_TYPE
```

Abb.7-4: MB_KINEMATICS_CONTROL_TYPE01

Admin Zugriff auf Attribute der Verwaltung der Kinematik

SetupMode Zugriff auf Attribute der Tipp-/Einrichtfunktion

CoordMode Zugriff auf Attribute der Positionierbetriebsarten

RCLMode Zugriff auf Attribute der RCL-Betriebsart

Bibliothek ML_KinTech.compiled-library

System IndraMotion MLC

MB_KINEMATICS_CONTROL_TYPE01 Elemente

MB_KINEMATICS_ADMINISTRATION Struktur

Kurzbeschreibung Die MB_KINEMATICS_ADMINISTRATION-Struktur ist eine Unterstruktur von "MB_KINEMATICS_CONTROL_TYPE01" und enthält Steuerungsvariablen und Sollwerte für die Verwaltung der Kinematik.

Kinematik-Interface

Das folgende Bild zeigt die MB_KINEMATICS_ADMINISTRATION-Struktur:

MB_KINEMATICS_ADMINISTRATION

```
TYPE MB_KINEMATICS_ADMINISTRATION :
  STRUCT
    (* selection of the operating mode |  Betriebsarten-Anwahl
    - level controlled
    - set the corresponding variables for your operating mode before activating
    - use enumeration: e.g. ..._OpMode.en := ModeCoordAB
    - OR use bit variables: e.g. ..._OpMode.b.MODE_COORD_AB := TRUE
    Note: only one bit is allowed at any time!
    ------------------------------------------------------------------------------
    - Zustandsgesteuert
    - vor Anwahl der Betriebsart die Eingangsdaten fuer die Betriebsart beschreiben
    - Betriebsarten-Anwahl (ENUMs): z.B. ..._OpMode.en := ModeCoordAB
    - Betriebsarten-Anwahl (BIT-Variablen): z.B. ..._OpMode.b.MODE_COORD_AB := TRUE
    Hinweis: immer nur 1 Eingang setzen! *)
    _OpMode:  MB_KINEMATICS_MODE_UNION := ( en := ModeCoordAb);

    (* the reference to this kinematic | Kinematik-Referenz  *)
    Group:   AXIS_GROUP_REF := (CntrlNo := LOCAL_CNTRL, GroupNo := NO_KINEMATICS);

    (* Pointer to the transformation interface of this kinematic
    ------------------------------------------------------------------------------
    Zeiger auf das Transformations-Interface der Kinematik *)
    KinTransformPLC: MC_KIN_REF := 0;

    (* Refresh time for arKinStatus[].Diag.Number   | Auffrischrate von arKinStatus[].Diag.Number
    - switched off, if zero                         | - ausgeschaltet wenn Null
    - read kinematics para when >= T#0ms            | - liest Antriebsparameter wenn >= T#0ms *)
    DiagNbrRefreshTime: TIME := T#200MS;

    (* Brake deceleration - always positive,
    (Unit: e.g. in [rad/s2] depending on parametrization)
    ------------------------------------------------------------------------------
    Bremsverzoegerung - positiver Wert,
    (Einheit: z.B. in [rad/s2] je nach Parametrierung) *)
    StopDeceleration: REAL := 999.0;

    (* Deceleration jerk - always positive,
    (Unit: e.g. in [rad/s3] depending on parametrization)
    ------------------------------------------------------------------------------
    Verzoegerungsruck - positiver Wert,
    (Einheit: z.B. in [rad/s3] je nach Parametrierung) *)
    StopJerk: REAL := 9999.0;

    (* clears kinematics errors with MC_GroupReset() | loescht Kinematikfehler mit MC_GroupReset()
    - edge controlled                                | - flankengesteuert *)
    ClearError: BOOL := FALSE;

    (* Feedrate-Override                  | Vorschubkorrektur
    - is written to the Parameter K-0-0240  | - wird auf den Parameter K-0-0240 geschrieben *)
    VelocityOverride: REAL := 100;

    (* Acceleration-Override              | Beschleunigungskorrektur
    - is written to the Parameter K-0-0242  | - wird auf den Parameter K-0-0242 geschrieben *)
    AccelerationOverride: REAL := 100;
```

Abb.7-5: MB_KINEMATICS_ADMINISTRATION (Teil 1)

```
    (* Deceleration-Override                | Verzögerungskorrektur
    - is written to the Parameter K-0-0243  | - wird auf den Parameter K-0-0243 geschrieben *)
    DecelerationOverride: REAL := 100;

    (* Enable the cyclic check and transmission of changed inputs for the actual operating mode
    - relevant for the values in arKinCtrl_gb with note "Note: cyclic checked AND transmitted on change"
    - level controlled
    ------------------------------------------------------------------------------------------------
    Zuschalten des bedarfsweisen Scannens von Eingangsdaten in der aktuellen Betriebsart
    - betrifft Groessen in arKinCtrl_gb mit Anm. "Hinweis: wird permanent zyklisch verarbeitet"
    - Zustandsgesteuert *)
    EnableCyclicScanning:  BOOL := TRUE;

    (* Switch to parameter level 1, parameter A-0-0024 bit 3
    - TRUE:  activate parameter level 1
    - FALSE: deactivate parameter level 1
    ------------------------------------------------------------------------------------------------
    Umschalten der Achsen in den Parametriemodus 1, Parameter A-0-0024 Bit 3
    - TRUE: aktiviert den Parametriermodus
    - FALSE: deaktiviert den Parametriermodus *)
    ParamLevel: BOOL := FALSE;

    (* Restart the same OpMode again, e.g. ModeCoordPosLinAbs with same values
    - edge controlled
    ------------------------------------------------------------------------------------------------
    Neustat der gleichen Betriebsart, z.B. ModeCoordPosLinAbs mit gleichen Werten
    - flankengesteuert *)
    RetriggerOpMode: BOOL := FALSE;

    (* Configuration of a conveyor belt for Robot-Control V1
    ------------------------------------------------------------------------------------------------
    Konfiguration eines Transportbandes für Robot-Control V1 *)
    ConveyorBeltV1: ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF MB_KINEMATICS_CONVEYOR_BELT_V1;

    (* Configuration of a conveyor belt for Robot-Control V2
    ------------------------------------------------------------------------------------------------
    Konfiguration eines Transportbandes für Robot-Control V2 *)
    ConveyorBelt: ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF MB_KINEMATICS_CONVEYOR_BELT;

    (* Handling of RCL points | Handhabung von RCL-Punkten
    - set a single point      | - Setzen eines einzelnen Punktes
    - read a single point     | - Lesesn eines einzelnen Punktes
    - read a points list      | - Lesen einer Punkteliste
    - read a program list     | - Lesen einer Programmliste
    - ONLY Robot-Control V1   | - NUR mit Robot-Control V1 zulässig *)
    RCLPoint: MB_KINEMATICS_RCL_POINT;

    (* Handling of RCL programs | Handhabung von RCL-Programmen
    - load a program           | - Laden eines Programms
    - save a program           | - Speichern eines Programme
    - compile a programe       | - Kompilieren eines Programms
    - ONLY Robot-Control V1    | - NUR mit Robot-Control V1 zulässig*)
    RCLProg: MB_KINEMATICS_RCL_PROG;

    (* Activities after changing operation mode
    - NoAction: execute the latest block and command the next block
    - StopMovement: stop the latest block and start the next block
    - AbortMovement: abort the latest block and blend to the next block
    ------------------------------------------------------------------------------------------------
    Verhalten nach einem Betriebsartenwechsel
    - NoAction: führe den aktuellen Satz aus und kommandiere den nächsten Satz
    - StopMovement: stoppe den aktuellen Satz und starte den nächsten Satz
    - AbortMovement: unterbreche den aktuellen Block und verschleife auf den nächsten Satz *)
    ChangeOpMode: MB_KINEMATICS_ADMIN_CHANGE_OPMODE := NoAction;
  END_STRUCT
END_TYPE
```

Abb.7-6: MB_KINEMATICS_ADMINISTRATION (Teil 2)

OpMode Anwahl der Betriebsart

Nähere Informationen zum Betriebsartenwechsel siehe Kap. "Überblick" auf Seite 182



Das Setzen mehrerer Bits in `arKinCtrl_gb[].Admin._OpMode.b` verursacht einen Fehler.

Group Dieses Element enthält Informationen über die Steuerungs- und Kinematiknummer und dient den intern genutzten Funktionsbausteinen als Referenz.

KinTransformPLC Mithilfe dieses Elementes ist es möglich, benutzerdefinierte Tranformationen anzumelden. Legt man einen Transformations-Funktionsbaustein an, so wird dieser in der SPS angemeldet.

Kinematik-Interface

☞ Die Anmeldung einer benutzerdefinierten Transformation ist nur im Parametriermodus **P2** möglich. Wird versucht, eine Transformation im Betriebsmodus BB anzumelden, bleibt dies ohne Resultat.

☞ Wird das Kinematik-Interface erstmalig mit einer benutzerdefinierten Transformation gestartet so gibt die Steuerung den Fehler "Transformation nicht angemeldet" aus. Dies ist dadurch begründet, dass das IMC-Interface versucht in den Betriebsmodus zu schalten. Nach erfolgreicher Anmeldung der Transformation lässt sich der Fehler quittieren.

DiagNbrRefreshTime Mit diesem Element wird die Aktualisierungsrate von `arKinStatus_gb[].Diag` gesteuert.

StopDeceleration Der Wert dieses Elements gibt die Bremsverzögerung an, welche beim Stoppen der Kinematik wirkt.

☞ Nur für Robot-Control V2 zulässig.

StopJerk Der Wert dieses Elements gibt den Verzögerungsruck an, welcher beim Stoppen der Kinematik wirkt.

☞ Nur für Robot-Control V2 zulässig.

ClearError Mit diesem flankengesteuerten Element können Kinematik- und Achsfehler gelöscht werden. Ebenfalls wird mit steigender Flanke der Funktionsbaustein "MC_GroupReset" aufgerufen.

VelocityOverride Dieses Element gibt die Vorschubkorrektur an. Der Wert wird intern auf den Parameter K-0-0240 geschrieben.

AccelerationOverride Dieses Element gibt die Beschleunigungskorrektur an. Der Wert wird intern auf den Parameter K-0-0242 geschrieben.

DecelerationOverride Dieses Element gibt die Verzögerungskorrektur an. Der Wert wird intern auf den Parameter K-0-0243 geschrieben.

EnableCyclicScanning Dieses zustandsgesteuerte Element steuert das zyklische Lesen von `arKinCtrl_gb[].CoordMode.Point`. Mit Setzen des Elements wird mit jeder neuen Position eine neue Bewegung gestartet und somit ein neuer Satz in den Satzpuffer gestellt.

ParamLevel Mit diesem flankengesteuerten Element können alle, der Kinematik zugehörigen Achsen in den Parametriermodus versetzt werden. Durch Rücksetzen des Elements werden die Achsen wieder in den ursprünglichen Zustand versetzt.

☞ Ist der Parametriermodus aktiv, wird im Projektbaum bei den zugehörigen Achsen "PM" angezeigt.

RetriggerOpMode Dieses flankengesteuerte Element bewirkt einen Neustart der aktuell verwendeten Betriebsart. Wird es gesetzt, so schaltet das Kinematik-Interface intern nach "ModeCoordExternalFB" und anschließend wieder in die ursprüngliche Betriebsart. Somit ist beispielsweise ein mehrmaliges relatives Verfahren der Kinematik möglich, ohne Positionswerte zu ändern.

ConveyorBeltV1 Zugriff auf die Attribute zur Konfiguration von Bändern für Robot-Control V1

Nähere Informationen siehe Kap. "MB_KINEMATICS_CONVEYOR_BELT_V1" auf Seite 154

Nur zulässig für Robot-Control V1. Maximal sind 4 Bänder pro Kinematik konfigurierbar.

ConveyorBelt Zugriff auf die Attribute zur Konfiguration von Bändern für Robot-Control V2

Nähere Informationen siehe Kap. "MB_KINEMATICS_CONVEYOR_BELT" auf Seite 152

Nur zulässig für Robot-Control V2. Maximal sind 16 Bänder pro Kinematik konfigurierbar.

RCLPoint Zugriff auf Attribute zur Handhabung von RCL-Punkten

Nähere Informationen siehe Kap. "MB_KINEMATICS_RCL_POINT" auf Seite 155

Nur für Robot-Control V1 zulässig.

RCLProg Zugriff auf Attribute zur Handhabung von RCL-Programmen

Nähere Informationen siehe Kap. "MB_KINEMATICS_RCL_PROG" auf Seite 158

Nur für Robot-Control V1 zulässig.

ChangeOpMode Mithilfe dieses Elements kann das Verhalten nach einem Betriebsartenwechsel definiert werden.

Folgende Werte sind möglich:

- NoAction - bei einem Betriebsartenwechsel wird die vorher kommandierte Bewegung abgeschlossen
- StopMovement - bei einem Betriebsartenwechsel wird die aktuelle Bewegung mithilfe des Funktionsbausteins "MB_IRDProgStop" (Robot-Control V1) bzw. "MC_GroupStop" (Robot-Control V2) gestoppt

Bibliothek ML_KinTech.compiled-library

System IndraMotion MLC

MB_KINEMATICS_ADMINISTRATION Elemente

MB_KINEMATICS_MODE_UNION

Kurzbeschreibung Die Struktur MB_KINEMATICS_MODE_UNION ist ein Vereinigungsdatentyp (UNION), der vom **_OpMode**-Element, welches in der MB_KINEMATICS_ADMINISTRATION-Struktur enthalten ist, benutzt wird. Sie enthält den Aufzählungstyp MB_KINEMATICS_MODE und die Bitstruktur MB_KINEMATICS_MODE_BITS. Die Steuerung von _OpMode ist über beide Datentypen möglich, da dieses auf der gleichen Speicherstelle liegen.

Kinematik-Interface

Das folgende Bild zeigt den aktuellen MB_KINEMATICS_MODE- und MB_KINEMATICS_MODE_BITS-Datentyp:

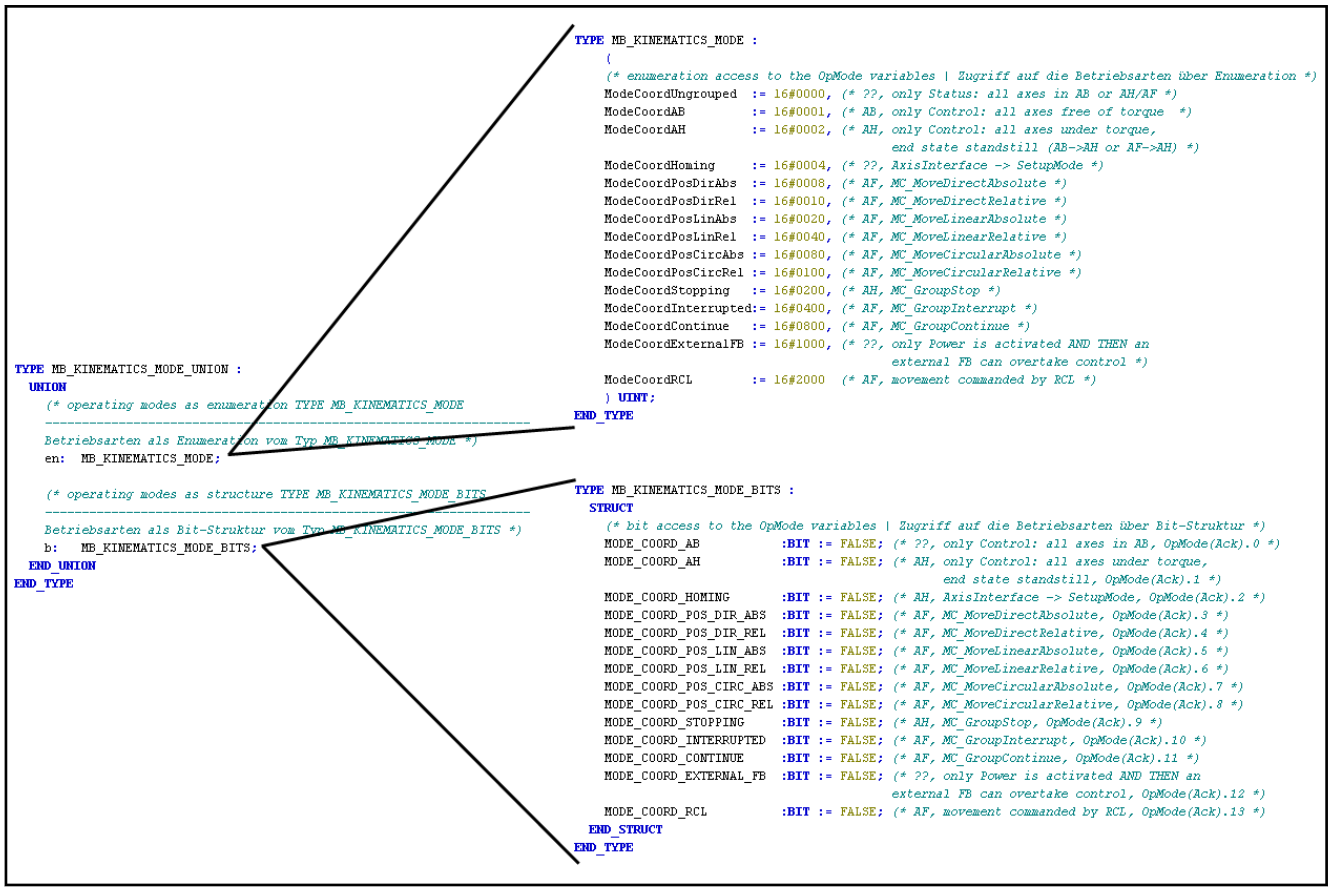


Abb.7-7: Datentypen MB_KINEMATICS_MODE _UNION

en Zugriff auf die Betriebsart über einen Aufzählungswert

b Zugriff auf die Betriebsart über eine Bitstruktur

Bibliothek ML_KinTech.compiled-library

System IndraMotion MLC

MB_KINEMATICS_CONVEYOR_BELT

Kurzbeschreibung Die Struktur MB_KINEMATICS_CONVEYOR_BELT dient der Konfiguration eines Bandes mit der Robot-Control V2.

Die in dieser Struktur enthaltene Elemente entsprechen den Eingängen der Funktionsbausteine

- ML_SetBeltConfiguration,
- ML_SetBeltErrorReaction.

Nähere Informationen sind der entsprechenden Dokumentation zu entnehmen.

Folgende Abbildung zeigt die Elemente der Struktur.

☞ Nur für Robot-Control V2 zulässig.

MB_KINEMATICS_CONVEYOR_BELT

```
TYPE MB_KINEMATICS_CONVEYOR_BELT :
  STRUCT
    (* Execute belt configuration     | Ausführen der Bandkonfiguration
    - edge controlled                 | - flankengesteuert *)
    Execute: BOOL := FALSE;

    (* Defines the position and rotation of the belt relative to the WCS
    ------------------------------------------------------------------------
    Definiert die Position und Verdrehung des Bandes relativ zum WCS *)
    ConveyorBeltOrigin: ARRAY [1..6] OF REAL := [6(0.0)];

    (* Defines the length of the belt synchronous traverse from the starting
    position
    ------------------------------------------------------------------------
    Definiert die Länge des bandsynchronen Verfahrbereichs ab der Startposition *)
    TotalLength: REAL := 0.0;

    (* Defines the length of the latest starting for band synchronous movement
    from the start position
    ------------------------------------------------------------------------
    Definiert die Länge des spätesten Starten einer bandsynchronen Bewegung ab
    der Startposition *)
    BeginLength: REAL := 0.0;

    (* Referenz of the belt axis
    ------------------------------------------------------------------------
    Referenz der Bandachse *)
    Axis: AXIS_REF;

    (* Defines the type of the error reaction
    ------------------------------------------------------------------------
    Definiert den Typ der Fehlerreaktion *)
    ErrorReactionType: ML_BELT_ERROR_REACTION_MODE := ML_BELT_ERROR_NO_SPECIFIC_REACTION;

    (* Defines the time for the desynchronization
    ------------------------------------------------------------------------
    Definiert die Zeit zum absynchronisieren *)
    ErrorReactionDesyncTime: REAL := 0.0;

    (* Defines the deceleration for the desynchronization
    ------------------------------------------------------------------------
    Definiert die Bremsverzögerung zum absynchronisieren *)
    ErrorReactionDeceleration: REAL := 0.0;

    (* Defines the jerk for the desynchronization
    ------------------------------------------------------------------------
    Definiert den Bremsruck zum absynchronisieren *)
    ErrorReactionJerk: REAL := 0.0;
  END_STRUCT
END_TYPE
```

Abb.7-8: MB_KINEMATICS_CONVEYOR_BELT

Bibliothek ML_KinTech.compiled-library

Kinematik-Interface

System IndraMotion MLC

MB_KINEMATICS_CONVEYOR_BELT_V1

Kurzbeschreibung Die Struktur MB_KINEMATICS_CONVEYOR_BELT_V1 dient der Konfiguration eines Bandes mit der Robot-Control V1.

Die in dieser Struktur enthaltene Elemente entsprechen den Eingängen des Funktionsbausteins

- ML_SetBeltParameter

Nähere Informationen sind der entsprechenden Dokumentation zu entnehmen.



Nur für Robot-Control V1 zulässig.

Folgende Abbildung zeigt die Elemente der Struktur:

MB_KINEMATICS_CONVEYOR_BELT_V1

```
TYPE MB_KINEMATICS_CONVEYOR_BELT_V1 :
  STRUCT
    (* Execute belt configuration     | Ausführen der Bandkonfiguration
    - edge controlled                 | - flankengesteuert *)
    Execute: BOOL := FALSE;

    (* Defines the belt synchronization algorithm
    ------------------------------------------------------------------------
    Definiert den Bandsynchronisationsalgorithmus *)
    BeltKind: ML_BELT_KIND := BK_4;

    (* Defines the starting position for belt synchronization
    ------------------------------------------------------------------------
    Definiert die Startposition ab der bandsynchron Verfahren werden kann *)
    Base: ARRAY [1..16] OF REAL := [16(0.0)];

    (* Defines the length of the belt synchronous traverse from the starting
    position
    ------------------------------------------------------------------------
    Definiert die Länge des bandsynchronen Verfahrbereichs ab der Startposition *)
    FullLength: REAL := 0.0;

    (* Defines the length of the latest starting for band synchronous movement
    from the start position
    ------------------------------------------------------------------------
    Definiert die Länge des spätesten Starten einer bandsynchronen Bewegung ab
    der Startposition *)
    BeginLength: REAL := 0.0;

    (* Referenz of the belt axis
    ------------------------------------------------------------------------
    Referenz der Bandachse *)
    Axis: AXIS_REF;
  END_STRUCT
END_TYPE
```

Abb.7-9: MB_KINEMATICS_CONVEYOR_BELT_V1

Bibliothek ML_KinTech.compiled-library

System IndraMotion MLC

MB_KINEMATICS_RCL_POINT

Kurzbeschreibung *Die Struktur MB_KINEMATICS_RCL_POINT dient dem*

- Schreiben eines einzelnen Punktes (ML_RoCoSetPoint),
- Lesen eines einzelnen Punktes (ML_RoCoGetPoint),
- Leser einer Punkteliste (ML_RoCoGetPointList),
- Lesen einer Programmliste (ML_RoCoDir).

Nähere Informationen entnehmen Sie der entsprechenden Dokumentation.

Kinematik-Interface

☞ Nur für Robot-Control V1 zulässig.

Folgende Abbildung zeigt die Elemente der Struktur:

MB_KINEMATICS_RCL_POINT

```
TYPE MB_KINEMATICS_RCL_POINT :
  STRUCT
    (* Execute process  | Ausführen der Bearbeitung
    - edge controlled   | - flankengesteuert *)
    Execute: BOOL;

    (* Type of process    | Art der Bearbeitung
    - write a point       | - schreiben eines Punktes
    - read a point        | - lesen eines Punktes
    - read a point list   | - lesen einer Punkteliste
    - read a program list | - lesen einer Programmliste *)
    Mode: MB_KINEMATICS_RCL_POINT_MODE := SetPoint;

    (* File name of the file
    - input of the function block ML_RoCoSetPoint
    - input of the function block ML_RoCoDir
    - input of the function block ML_RoCoGetPoint
    - input of the function block ML_RoCoGetPointList
    ------------------------------------------------------------------------
    Dateiname der zu lesenden Datei
    - Eingang des Funktionsbausteins ML_RoCoSetPoint
    - Eingang des Funktionsbausteins ML_RoCoDir
    - Eingang des Funktionsbausteins ML_RoCoGetPoint
    - Eingang des Funktionsbausteins ML_RoCoGetPointList *)
    FileName: ML_ROCO_NAME := '';

    (* Point name of the file
    - input of the function block ML_RoCoSetPoint
    - input of the function block ML_RoCoGetPoint
    ------------------------------------------------------------------------
    Punktename der zu lesenden Datei
    - Eingang des Funktionsbausteins ML_RoCoSetPoint
    - Eingang des Funktionsbausteins ML_RoCoGetPoint *)
    PointName: ML_ROCO_POINT_NAME := '';

    (* Number of dimensions
    - input of the function block ML_RoCoSetPoint
    - input of the function block ML_RoCoGetPoint
    ------------------------------------------------------------------------
    Anzahl der Dimension
    - Eingang des Funktionsbausteins ML_RoCoSetPoint
    - Eingang des Funktionsbausteins ML_RoCoGetPoint *)
    ArrayDimension: UDINT := 0;

    (* Index of each dimension
    - input of the function block ML_RoCoSetPoint
    - input of the function block ML_RoCoGetPoint
    ------------------------------------------------------------------------
    Index jeder Dimension
    - Eingang des Funktionsbausteins ML_RoCoSetPoint
    - Eingang des Funktionsbausteins ML_RoCoGetPoint *)
    PointIdx: ARRAY [1..ML_MAX_KIN_ARRAY_DIMENSIONS] OF DINT := [ML_MAX_KIN_ARRAY_DIMENSIONS(0)];
```

Abb.7-10: MB_KINEMATICS_RCL_POINT (Teil 1)

Kinematik-Interface

```
54      (* Index of each dimension
55      - input of the function block ML_RoCoDir
56      - input of the function block ML_RoCoGetPointList
57      ------------------------------------------------------------------------
58      Index jeder Dimension
59      - Eingang des Funktionsbausteins ML_RoCoDir
60      - Eingang des Funktionsbausteins ML_RoCoGetPointList *)
61      StartIdx: UDINT := 1;
62
63      (* Point structure with name and value
64      - input of the function block ML_RoCoSetPoint
65      ------------------------------------------------------------------------
66      Struktur eines Punktes mit Namen und Werte
67      - Eingang des Funktionsbausteins ML_RoCoSetPoint *)
68      Point: ML_ROCO_POINT;
69
70      (* Direction of the read-sequence(low to high, high to low)
71      - input of the function block ML_RoCoDir
72      ------------------------------------------------------------------------
73      Reihenfolge, in welcher gelesen wird (aufsteigend, absteigend)
74      - Eingang des Funktionsbausteins ML_RoCoDir *)
75      SearchDirection: BOOL := FALSE;
76    END_STRUCT
77  END_TYPE
```

Abb.7-11: MB_KINEMATICS_RCL_POINT (Teil 2)

Mode Dieses Element definiert die Art des Zugriffes.

Folgende Enumeration sind möglich:

- SetPoint: Schreiben eines einzelnen Punktes
- GetPoint: Lesen eines einzelnen Punktes
- GetPointList: Leser einer Punkteliste
- GetProgList: Lesen einer Programmliste

Bibliothek ML_KinTech.compiled-library

System IndraMotion MLC

MB_KINEMATICS_RCL_PROG

Kurzbeschreibung *Die Struktur MB_KINEMATICS_RCL_PROG dient dem*

- Laden eines RCL-Programms von der Compact Flash Karte in den internen Speicher der Steuerung,
- Speichern eines RCL-Programms vom internen Speicher der Steuerung auf der Compact Flash Karte,
- Kompilieren eines RCL-Programms in der Steuerung.

☞ Nur für Robot-Control V1 zulässig.

Für diese Aktionen werden die Parameter K-0-1060, K-0-1061 und K-0-1062 verwendet:

MB_KINEMATICS_RCL_PROG

```
1  TYPE MB_KINEMATICS_RCL_PROG :
2    STRUCT
3      (* Name of the RCL program which will be loaded or saved
4      - the name is written to the parameter K-0-0220 and K-0-0221
5      ------------------------------------------------------------------------
6      Name des RCL-Programms, welcher geladen oder gespeichert werden soll
7      - der Name wird auf den Parameter K-0-0220 und K-0-0221 geschrieben *)
8      ProgName: STRING[12] := '';
9
10     (* copy a RCl program from the Compact Flash card to the control's
11     internal memory
12     - edge controlled
13     ------------------------------------------------------------------------
14     Laden eines RCL-Programms von der Compact Flash Karte in den internen
15     Speicher der Steuerung
16     - flankengesteuert *)
17     ProgLoad: BOOL := FALSE;
18
19     (* save a RCl program from the control's internal memory to the
20     Compact Flash card
21     - edge controlled
22     ------------------------------------------------------------------------
23     Speichern eines RCL-Programms vom internen Speicher der Steuerung auf die
24     Compact Flash Karte
25     - flankengesteuert *)
26     ProgSave: BOOL := FALSE;
27
28     (* compile a RCL program at the controler
29     - edge controlled
30     ------------------------------------------------------------------------
31     Kompilieren eines RCL-Programms in der Steuerung
32     - flankengesteuert *)
33     ProgCompile: BOOL := FALSE;
34   END_STRUCT
35 END_TYPE
```

Abb.7-12: MB_KINEMATICS_RCL_PROG

ProgName In diesem Element steht der Name des RCL Programms, welches geladen oder gespeichert werden soll. Dieser Name wird intern auf die Parameter K-0-0220 und K-0-0221 geschrieben.

ProgLoad Dieses Element löst ein Kommando aus, welches die im Parameter K-0-0220 angegebene(n) Datei(en) von der Compact Flash-Karte in den inneren Speicher der Steuerung kopiert. Hierzu wird intern der Parameter K-0-1060 genutzt.

ProgSave Dieses Element löst ein Kommando aus, welches die im Parameter K-0-0220 angegebene(n) Datei(en) vom internen Speicher der Steuerung auf die Compact Flash-Karte kopiert. Hierzu wird intern der Parameter K-0-1061 genutzt.

Kinematik-Interface

ProgCompile Dieses Element kompiliert die im Parameter K-0-0221 angegebene QLL-Datei. Hierzu wird intern der Parameter K-0-1062 genutzt.

Bibliothek ML_KinTech.compiled-library

System IndraMotion MLC

MB_KINEMATICS_SETUP Struktur

Kurzbeschreibung Die MB_KINEMATICS_SETUP-Struktur ist eine Unterstruktur von MB_KINEMATICS_CONTROL_TYPE01 und enthält Steuerungsvariablen für das Vertippen einer Kinematik.

Das folgende Bild zeigt die MB_KINEMATICS_SETUP-Struktur:

MB_KINEMATICS_SETUP

```
TYPE MB_KINEMATICS_SETUP :
  STRUCT
    (* Enable setup mode   | Freigabe Tipp-/Einrichtbetrieb *)
    Enable: BOOL := FALSE;

    (* Coordination system | Koordinatensystem *)
    CoordSystem: ML_COORDSYS := CS_MACHINE_COORDINATES;

    (* Jog mode | Tippbetrieb *)
    JogMode: ML_KIN_E_JOG_VEL := KIN_JOG_STEP_SMALL;

    (* Jog +               | Tippen +
    - edge controlled when  | - flankengesteuert, wenn
            JogMode = KIN_JOG_CONT_*
    - level controlled then | - zustandsgesteuert, wenn
            JogMode = KIN_JOG_STEP_* *)
    JogPlus: ML_COORD_BITS;

    (* Jog -               | Tippen -
    - edge controlled when  | - flankengesteuert, wenn
            JogMode = KIN_JOG_CONT_*
    - level controlled then | - zustandsgesteuert, wenn
            JogMode = KIN_JOG_STEP_* *)
    JogMinus: ML_COORD_BITS;

    (* Coordination system    | Kurze Tippdistanz *)
    DistanceSmall: ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL := [MB_MAX_KIN_AXIS_NUMBER(1.0)];

    (* Coordination system    | Lange Tippdistanz *)
    DistanceLarge: ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL := [MB_MAX_KIN_AXIS_NUMBER(10.0)];

    (* Slow jog velocity      | Langsame Tippgeschwindigkeit *)
    VelocitySlow: ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL := [MB_MAX_KIN_AXIS_NUMBER(10.0)];

    (* Fast jog velocity      | Schnelle Tippgeschwindigkeit *)
    VelocityFast: ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL := [MB_MAX_KIN_AXIS_NUMBER(25.0)];

    (* Jog acceleration       | Tippbeschleunigung *)
    Acceleration: ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL := [MB_MAX_KIN_AXIS_NUMBER(370.0)];

    (* Jog deceleration       | Tippverzögerung *)
    Deceleration: ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL := [MB_MAX_KIN_AXIS_NUMBER(370.0)];

    (* Jog jerk               | Tippruck *)
    Jerk: ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL := [MB_MAX_KIN_AXIS_NUMBER(0.0)];
  END_STRUCT
END_TYPE
```

Abb.7-13: MB_KINEMATICS_SETUP

Enable Dieses Element aktiviert den Tipp-/Einrichtbetrieb.

CoordSystem Dieses Element definiert das Koordinatensystem, in dem vertippt werden soll.

Folgende Werte sind für Robot-Control V1 möglich:

- CS_MACHINE_COORDINATES
  (entspricht intern KIN_CORD_SYS_ORIGIN_COORD)
- CS_AXES_COORDINATES
  (entspricht intern KIN_CORD_SYS_JOINT_COORD)
- PRODUCT_COORDINATES_1

(entspricht intern KIN_CORD_SYS_WORKPIECE_COORD)

- CS_GRIPPER_COORDINATES

  (entspricht intern KIN_CORD_SYS_GRIPPER_COORD)

Folgende Werte sind für Robot-Control V2 möglich:

- CS_MACHINE_COORDINATES
- CS_AXES_COORDINATES
- CS_BASE_COORDINATES
- CS_PRODUCT_COORDINATES_1
- CS_PRODUCT_COORDINATES_2
- ...
- CS_PRODUCT_COORDINATES_16

JogMode Dieses Element definiert die Art des Vertippens (kleine/große inkrementelle Bewegung, langsame/schnelle kontinuierliche Bewegung).

JogPlus Dieses Element beinhaltet Bits für jede Achse der Kinematik. Ein Setzen des entsprechenden Bits löst eine Verfahrbewegung entsprechend der Vorgaben in positive Richtung aus.

JogMinus Dieses Element beinhaltet Bits für jede Achse der Kinematik. Ein Setzen des entsprechenden Bits löst eine Verfahrbewegung entsprechend der Vorgaben in negative Richtung aus.

DistanceSmall Dieses Element beinhaltet die kurze Distanz für das inkrementelle Tippen im Achs- und kartesischen Koordinatensystem. Bei Nutzung der Robot-Control V1 wird dieser Wert auf die Parameter A-0-0080 und K-0-0080 geschrieben.

DistanceLarge Dieses Element beinhaltet die lange Distanz für das inkrementelle Tippen im Achs- und kartesischen Koordinatensystem. Bei Nutzung der Robot-Control V1 wird dieser Wert auf die Parameter A-0-0081 und K-0-0081 geschrieben.

VelocitySlow Dieses Element beinhaltet die langsame Geschwindigkeit für das kontinuierliche Tippen im Achs- und kartesischen Koordinatensystem. Bei Nutzung der Robot-Control V1 wird dieser Wert auf die Parameter A-0-0082 und K-0-0082 geschrieben.

VelocityFast Dieses Element beinhaltet die schnelle Geschwindigkeit für das kontinuierliche Tippen im Achs- und kartesischen Koordinatensystem. Bei Nutzung der Robot-Control V1 wird dieser Wert auf die Parameter A-0-0083 und K-0-0083 geschrieben.

Acceleration Dieses Element beinhaltet die Beschleunigung für das inkrementelle und kontinuierliche Tippen im Achs- und kartesischen Koordinatensystem. Bei Nutzung der Robot-Control V1 wird dieser Wert auf die Parameter A-0-0084 und K-0-0084 geschrieben.

Deceleration Dieses Element beinhaltet die Bremsverzögerung für das inkrementelle und kontinuierliche Tippen im Achs- und kartesischen Koordinatensystem. Bei Nutzung der Robot-Control V1 wird dieser Wert auf die Parameter A-0-0085 und K-0-0085 geschrieben.

Jerk Dieses Element beinhaltet den maximalen Beschleunigungs- und Bremsruck für das inkrementelle und kontinuierliche Tippen im Achs- und kartesischen Koordinatensystem. Bei Nutzung der Robot-Control V1 wird dieser Wert auf die Parameter A-0-0086/87 und K-0-0086/87 geschrieben.

Bibliothek ML_KinTech.compiled-library

System IndraMotion MLC

Kinematik-Interface

MB_KINEMATICS_COORDINATED Struktur

Kurzbeschreibung

Die MB_KINEMATICS_COORDINATED Struktur ist eine Unterstruktur von MB_KINEMATICS_CONTROL_TYPE01 und enthält Steuerungsvariablen und Sollwerte für die Positionierbetriebsarten.

Die in dieser Struktur enthaltene Elemente entsprechen den Eingängen der Funktionsbausteine:

- ML_MoveLinearAbsolute,
- ML_MoveLinearRelative,
- ML_MoveDirectAbsolute,
- ML_MoveDirectRelative,
- ML_MoveCircularAbsolute.
- ML_MoveCircularRelative.

Nähere Informationen sind der entsprechenden Dokumentation zu entnehmen.

Das folgende Bild zeigt die MB_KINEMATICS_COORDINATED-Struktur:

MB_KINEMATICS_COORDINATED

```
TYPE MB_KINEMATICS_COORDINATED:
  STRUCT
    (* position for an absolute move (Unit: e.g. in [degree] depending on
    parametrization) (negative or positive)
    - start moving with positionings opmodes
    Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
    ------------------------------------------------------------------------
    Zielposition, absolut oder relativ (Einheit: z.B. in [Grad] je nach Parametrierung)
    (negativ oder positiv)
    - Freigabe mit den Positinierbetriebsarten
    Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)
    Point: ARRAY [1..16] OF REAL := [16(0.0)];

    (* maximum velocity (Unit: e.g. in [rpm] depending on parametrization)
    (always positive)
    - start moving with positionings opmodes
    ------------------------------------------------------------------------
    Positioniergeschwindigkeit , (Einheit: z.B. in [U/min] je nach Parametrierung)
    (positiv eingeben)
    - Freigabe mit Positinierbetriebsarten *)
    Velocity: REAL := 10.0;

    (* Value of the acceleration (Unit: e.g. in [rad/s2] depending on
    parametrization) (always positive)
    - start moving with positionings opmodes
    ------------------------------------------------------------------------
    Beschleunigung, (Einheit: z.B. in [rad/s2] je nach Parametrierung) (immer positiv)
    - Freigabe mit Positinierbetriebsarten *)
    Acceleration: REAL := 10.0;

    (* Value of the deceleration (Unit: e.g. in [rad/s2] depending on
    parametrization) (always positive)
    - start moving with Positinierbetriebsarten
    ------------------------------------------------------------------------
    Verzoegerung, (Einheit: z.B. in [rad/s2] je nach Parametrierung) (immer positiv)
    - Freigabe mit Positinierbetriebsarten *)
    Deceleration: REAL := 10.0;

    (* Value of the jerk (Unit: e.g. in [rad/s3] depending on parametrization)
    (always positive)
    - start moving with positionings opmodes
    ------------------------------------------------------------------------
    Ruck, (Einheit: z.B. in [rad/s3] je nach Parametrierung) (immer positiv)
    - Freigabe mit Positinierbetriebsarten *)
    Jerk: REAL := 100.0;

    (* defines in which coordinate system the points have to be specified
    (machine coordinates or joint coordinates)
    - start moving with positionings opmodes
    ------------------------------------------------------------------------
    Koordinatenssytem (Maschinen- oder Achs-Koordinaten)
    - Freigabe mit Positinierbetriebsarten *)
    CoordSystem: ML_COORDSYS := CS_MACHINE_COORDINATES;

    (* Define the slope type of the motion (BlockSlope, ProgramSlope or ContinuousSlope)
    - start moving with positionings opmodes
    ------------------------------------------------------------------------
    Slope-Art der Bewegung (BlockSlope, ProgramSlope or ContinuousSlope)
    - Freigabe mit Positinierbetriebsarten *)
    SlopeType: ML_SLOPE_TYPE := ST_BLOCK_SLOPE;

    (* Defines whether the motion ends at the end point or is blended (to or via)
    - start moving with positionings opmodes
    ------------------------------------------------------------------------
    definiert ob die Bewegung am Endpunkt endet oder überschliffen wird (to oder via)
    - Freigabe mit Positinierbetriebsarten *)
    SetMode: ML_SET_MODE := SM_TO;

    (* Defines the blending radius (Unit: [$] from the parameter A-0-0230)
    - start moving with positionings opmodes
    ------------------------------------------------------------------------
    Überschleifradius (Unit: [$] vom Parameter A-0-0230)
    - Freigabe mit Positinierbetriebsarten *)
    BlendingRadius: REAL := 0.0;

    (* position for an circular move (Unit: e.g. in [degree] depending on
    parametrization) (negative or positive)
    - start moving with circular positionings opmodes
    ------------------------------------------------------------------------
    Zielposition, zirkular (Einheit: z.B. in [Grad] je nach Parametrierung)
    (negativ oder positiv)
    - Freigabe mit Positinierbetriebsart "Zircular" *)
    CircularAuxPoint: ARRAY [1..16] OF REAL := [16(0.0)];
```

Abb.7-14: MB_KINEMATICS_COORDINATED (Teil 1)

Kinematik-Interface

```
    (* Defines the interpretation of the "AuxPoint" input
    - start moving with circular positionings opmodes
    ------------------------------------------------------------------------
    Definiert die Interpretation des Eingangs "AuxPoint"
    - Freigabe mit Positinierbetriebsart "Zircular" *)
    CircularCircMode: ML_CIRCMODES := CM_BOARDER;
  END_STRUCT
END_TYPE
```

Abb.7-15: MB_KINEMATICS_COORDINATED (Teil 2)

Bibliothek ML_KinTech.compiled-library

System IndraMotion MLC

MB_KINEMATICS_RCL Struktur

Kurzbeschreibung Die MB_KINEMATICS_RCL Struktur ist eine Unterstruktur von MB_KINEMATICS_CONTROL_TYPE01 und enthält Steuerungsvariable und Sollwerte für die Positionierbetriebsarten.

Die in dieser Struktur enthaltene Elemente entsprechen den Eingängen des Funktionsbausteins "MB_IRDProgSelect". Nähere Informationen entnehmen Sie der entsprechenden Dokumentation.

Nur für Robot-Control V1 zulässig.

Das folgende Bild zeigt die MB_KINEMATICS_RCL-Struktur:

MB_KINEMATICS_RCL

```
1  TYPE MB_KINEMATICS_RCL :
2    STRUCT
3      (* Name of the IRD program without extension
4      ------------------------------------------------------
5      Name des IRD-Programms ohne Dateinamen-Erweiterung  *)
6      ProgName: STRING[12] := '';
7    END_STRUCT
8  END_TYPE
```

Abb.7-16: MB_KINEMATICS_COORDINATED

Bibliothek ML_KinTech.compiled-library

System IndraMotion MLC

MB_KINEMATICS_STATUS_TYPE01 Struktur

MB_KINEMATICS_STATUS_TYPE01

Kurzbeschreibung Auf die MB_KINEMATICS_STATUS_TYPE01-Struktur wird über die arKinStatus_gb[ ]-Instanz zugegriffen.

Sie enthält folgende Elemente:

MB_KINEMATICS_STATUS_TYPE01

```
1  TYPE MB_KINEMATICS_STATUS_TYPE01 :
2    STRUCT
3      (* Pointer to KinData of this kinematic
4      - Type .PtrKinData^. for the intellisense menu
5      ------------------------------------------------------------
6      Zeiger auf KinData-Schnittstelle der Kinematik
7      - Eingabe .PtrKinData^. fuer Auswahlmenue*)
8      PtrKinData: POINTER TO ML_KINEMATICSDATA_SM := 0;
9
10     (* Status of the kinematic | Kinematik-Status *)
11     Admin: MB_KINEMATICS_ADMIN_STATUS;
12
13     (* kinematic diagnosis information
14     - error, warning, etc.
15     ------------------------------------------------------------
16     Diagnose Informationen der Kinematik
17     - Fehler, Warnung, usw. *)
18     Diag: MB_KINEMATICS_DIAGNOSIS;
19
20     (* Jog functionality | Tipp/Einrichtfunktionen *)
21     SetupMode: MB_KINEMATICS_SETUP_STATUS;
22   END_STRUCT
23 END_TYPE
```

Abb.7-17: MB_KINEMATICS_STATUS_TYPE01

PtrKinData Zugriff auf KinData-Schnittstelle der Kinematik

Admin Zugriff auf Status-Attribute der Kinematik-Verwaltung

Diag Zugriff auf Diagnoseausgaben

SetupMode Zugriff auf Status-Attribute der Tipp-/Einrichtfunktion

Bibliothek ML_KinTech.compiled-library

System IndraMotion MLC

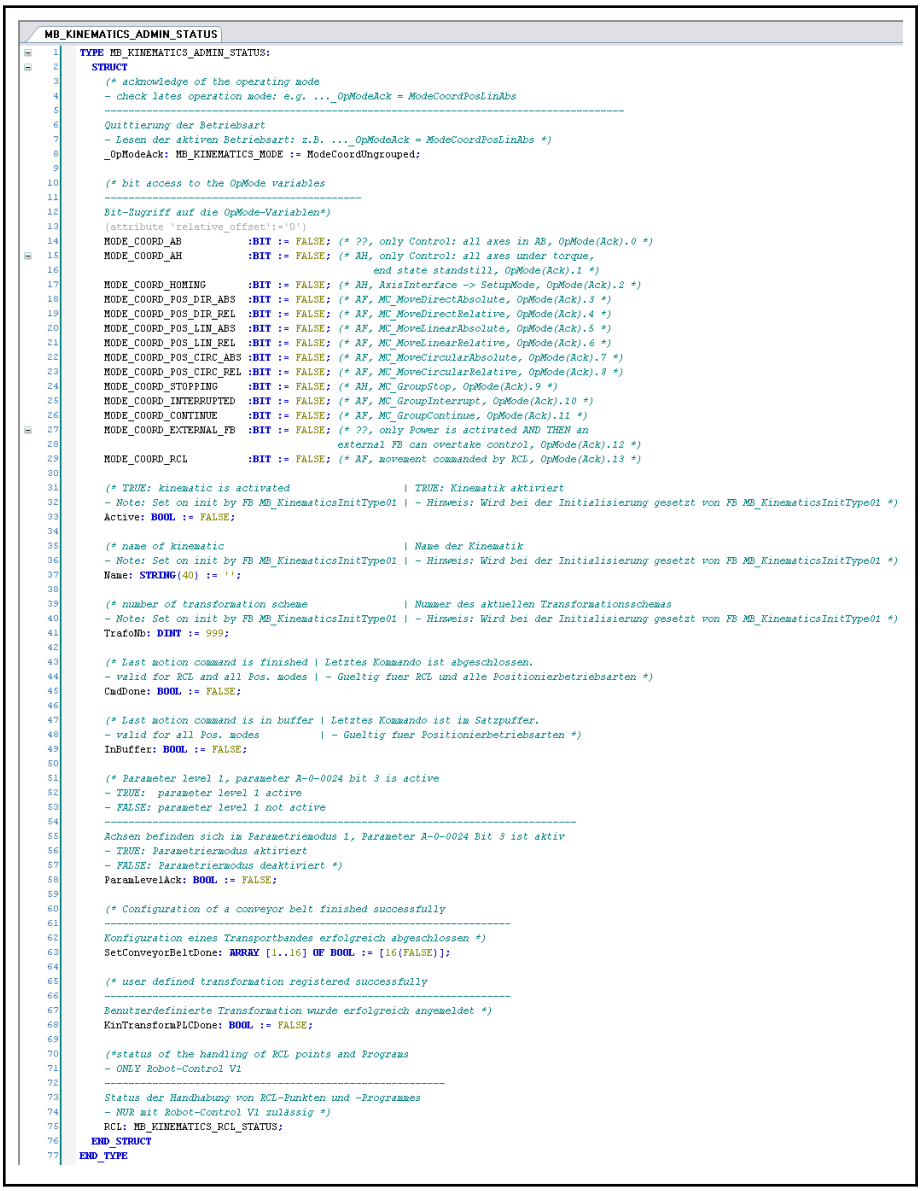
MB_KINEMATICS_STATUS_TYPE01 Elemente

MB_KINEMATICS_ADMIN_STATUS Struktur

Kurzbeschreibung Die MB_KINEMATICS_ADMIN_STATUS Struktur ist eine Unterstruktur von MB_KINEMATICS_STATUS_TYPE01 und enthält den Ausgangsstatus für die Verwaltung der Kinematik.

Kinematik-Interface

Das folgende Bild zeigt die MB_KINEMATICS_ADMIN_STATUS-Struktur:



```
MB_KINEMATICS_ADMIN_STATUS
TYPE MB_KINEMATICS_ADMIN_STATUS:
  STRUCT
    (* acknowledge of the operating mode
    - check lates operation mode: e.g. ..._OpModeAck = ModeCoordPosLinAbs
    ------------------------------------------------------------------------------------
    Quittierung der Betriebsart
    - Lesen der aktiven Betriebsart: z.B. ..._OpModeAck = ModeCoordPosLinAbs *)
    _OpModeAck: MB_KINEMATICS_MODE := ModeCoordUngrouped;

    (* bit access to the OpMode variables
    ----------------------------------
    Bit-Zugriff auf die OpMode-Variablen*)
    {attribute 'relative_offset':='0'}
    MODE_COORD_AB           :BIT := FALSE; (* ??, only Control: all axes in AB, OpMode(Ack).0 *)
    MODE_COORD_AH           :BIT := FALSE; (* AH, only Control: all axes under torque,
                                              end state standstill, OpMode(Ack).1 *)
    MODE_COORD_HOMING       :BIT := FALSE; (* AH, AxisInterface -> SetupMode, OpMode(Ack).2 *)
    MODE_COORD_POS_DIR_ABS  :BIT := FALSE; (* AF, MC_MoveDirectAbsolute, OpMode(Ack).3 *)
    MODE_COORD_POS_DIR_REL  :BIT := FALSE; (* AF, MC_MoveDirectRelative, OpMode(Ack).4 *)
    MODE_COORD_POS_LIN_ABS  :BIT := FALSE; (* AF, MC_MoveLinearAbsolute, OpMode(Ack).5 *)
    MODE_COORD_POS_LIN_REL  :BIT := FALSE; (* AF, MC_MoveLinearRelative, OpMode(Ack).6 *)
    MODE_COORD_POS_CIRC_ABS :BIT := FALSE; (* AF, MC_MoveCircularAbsolute, OpMode(Ack).7 *)
    MODE_COORD_POS_CIRC_REL :BIT := FALSE; (* AF, MC_MoveCircularRelative, OpMode(Ack).8 *)
    MODE_COORD_STOPPING     :BIT := FALSE; (* AH, MC_GroupStop, OpMode(Ack).9 *)
    MODE_COORD_INTERRUPTED  :BIT := FALSE; (* AF, MC_GroupInterrupt, OpMode(Ack).10 *)
    MODE_COORD_CONTINUE     :BIT := FALSE; (* AF, MC_GroupContinue, OpMode(Ack).11 *)
    MODE_COORD_EXTERNAL_FB  :BIT := FALSE; (* ??, only Power is activated AND THEN an
                                              external FB can overtake control, OpMode(Ack).12 *)
    MODE_COORD_RCL          :BIT := FALSE; (* AF, movement commanded by RCL, OpMode(Ack).13 *)

    (* TRUE: kinematic is activated                  | TRUE: Kinematik aktiviert
    - Note: Set on init by FB MB_KinematicsInitType01 | - Hinweis: Wird bei der Initialisierung gesetzt von FB MB_KinematicsInitType01 *)
    Active: BOOL := FALSE;

    (* name of kinematic                             | Name der Kinematik
    - Note: Set on init by FB MB_KinematicsInitType01 | - Hinweis: Wird bei der Initialisierung gesetzt von FB MB_KinematicsInitType01 *)
    Name: STRING(40) := '';

    (* number of transformation scheme               | Nummer des aktuellen Transformationsschemas
    - Note: Set on init by FB MB_KinematicsInitType01 | - Hinweis: Wird bei der Initialisierung gesetzt von FB MB_KinematicsInitType01 *)
    TrafoNb: DINT := 999;

    (* Last motion command is finished | Letztes Kommando ist abgeschlossen.
    - valid for RCL and all Pos. modes | - Gueltig fuer RCL und alle Positionierbetriebsarten *)
    CmdDone: BOOL := FALSE;

    (* Last motion command is in buffer | Letztes Kommando ist im Satzpuffer.
    - valid for all Pos. modes         | - Gueltig fuer Positionierbetriebsarten *)
    InBuffer: BOOL := FALSE;

    (* Parameter level 1, parameter A-0-0024 bit 3 is active
    - TRUE:  parameter level 1 active
    - FALSE: parameter level 1 not active
    ------------------------------------------------------------------------------------
    Achsen befinden sich im Parametriemodus 1, Parameter A-0-0024 Bit 3 ist aktiv
    - TRUE: Parametriermodus aktiviert
    - FALSE: Parametriermodus deaktiviert *)
    ParamLevelAck: BOOL := FALSE;

    (* Configuration of a conveyor belt finished successfully
    ------------------------------------------------------------------------
    Konfiguration eines Transportbandes erfolgreich abgeschlossen *)
    SetConveyorBeltDone: ARRAY [1..16] OF BOOL := [16(FALSE)];

    (* user defined transformation registered successfully
    ------------------------------------------------------------------------
    Benutzerdefinierte Transformation wurde erfolgreich angemeldet *)
    KinTransformPLCDone: BOOL := FALSE;

    (*status of the handling of RCL points and Programs
    - ONLY Robot-Control V1
    ------------------------------------------------------------
    Status der Handhabung von RCL-Punkten und -Programmes
    - NUR mit Robot-Control V1 zulässig *)
    RCL: MB_KINEMATICS_RCL_STATUS;
  END_STRUCT
END_TYPE
```

Abb.7-18: MB_KINEMATICS_ADMIN_STATUS

_OpModeAck Dieses Element gibt den Status der Betriebsart aus.

MODE_COORD_AB ... AH Diese Elemente dienen lediglich als Platzhalter und sind nur in der Control-Struktur zulässig.

MODE_COORD_HOMING ... RCL Diese Elemente geben den Status der Betriebsart als Bit aus. Ist keines der Bits gesetzt, so signalisiert dies den Status "ModeCoordUngrouped".

Active Dieses Element zeigt, ob die arKinCtrl_gb[ ]- und arKinStatus_gb[ ]-Struktur gültig (= TRUE) sind oder nicht (= FALSE). Es wird bei der Initialisierung des Kinematik-Interfaces gesetzt.

Name Dieses Element gibt den Namen der Kinematik an. Es wird bei der Initialisierung des Kinematik-Interfaces gesetzt.

TrafoNb Dieses Element gibt die Transformationsnummer der Kinematik an. Es wird bei der Initialisierung des Kinematik-Interfaces gesetzt.

CmdDone Dieses Element signalisiert, ob der **zuletzt** abgesetzte Befehl erfolgreich abgeschlossen wurde.

Nur gültig für die Betriebsarten

- ModeCoordContinue
- ModeCoordInterrupted
- ModeCoordPosDirAbs
- ModeCoordPosDirRel
- ModeCoordPosLinAbs
- ModeCoordPosLinRel
- ModeCoordPosCircAbs
- ModeCoordPosCircRel
- ModeCoordStopping
- ModeCoordRCL

InBuffer Dieses Element signalisiert, ob der **zuletzt** abgesetzte Befehl erfolgreich in den Satzpuffer gestellt wurde.

Nur gültig für die Betriebsarten

- ModeCoordContinue
- ModeCoordInterrupted
- ModeCoordPosDirAbs
- ModeCoordPosDirRel
- ModeCoordPosLinAbs
- ModeCoordPosLinRel
- ModeCoordPosCircAbs
- ModeCoordPosCircRel
- ModeCoordStopping

ParamLevelAck Dieses Element signalisiert, ob sich alle, der Kinematik zugehörigen Achsen, im Parametriermodus 1 befinden.

SetConveyorBeltDone Dieses Element signalisiert, ob die Konfiguration des Bandes erfolgreich abgeschlossen wurde.

KinTransformPLCDone Dieses Element signalisiert, ob die benutzerdefinierte Transformation erfolgreich angemeldet wurde.

RCL Zugriff auf Attribute zum Status zur Handhabung von RCL-Punkten und -Programmen

Nähere Informationen siehe Kap. "MB_KINEMATICS_RCL_STATUS" auf Seite 167

Bibliothek ML_KinTech.compiled-library

System IndraMotion MLC

MB_KINEMATICS_ADMIN_STATUS Elemente

MB_KINEMATICS_RCL_STATUS

Kurzbeschreibung Die MB_KINEMATICS_RCL_STATUS repräsentiert die Status-Strukturen zu MB_KINEMATICS_RCL_POINT und MB_KINEMATICS_RCL_PROG.

Kinematik-Interface

Das folgende Bild zeigt die MB_KINEMATICS_RCL_STATUS-Struktur:

MB_KINEMATICS_RCL_STATUS

```
TYPE MB_KINEMATICS_RCL_STATUS :
  STRUCT
    (* Status of the latest handling of RCL points
    - TRUE: processing was successful
    - FALSE: processing failed OR was not startet
    ------------------------------------------------
    Status der letzten Handhabung von RCL-Punkten
    - TRUE: Bearbeitung war erfolgreich
    - FALSE: Bearbeitung ist fehlgeschlagen oder wurde
    nicht gestartet *)
    PointModeDone: BOOL := FALSE;

    (* Shows the latest read points list
    ------------------------------------------------
    Zeigt die zuletzt gelesene Punkteliste *)
    PointList: ARRAY [1..16] OF ML_ROCO_POINT;

    (* Shows the latest read program list
    ------------------------------------------------
    Zeigt die zuletzt gelesene Programmliste *)
    ProgList: ARRAY [1..16] OF ML_ROCO_FILE_DIR;

    (* Shows the latest read point
    ------------------------------------------------
    Zeigt den zuletzt gelesenen Punkte *)
    Point: ML_ROCO_POINT;

    (* Status of K-0-0220 and K-0-0221
    - TRUE: set was successful
    - FALSE: set failed OR was not startet
    ------------------------------------------------
    Status des Setzens der Parameter K-0-0220 und
    K-0-0221
    - TRUE: Setzen war erfolgreich
    - FALSE: Setzen ist fehlgeschlagen oder wurde nicht
    gestartet *)
    ProgNameAck: BOOL := FALSE;

    (* Status of the copying from the Compact Flash card
    to the control's internal memory
    - TRUE: copying was successful
    - FALSE: copying failed OR was not startet
    ------------------------------------------------
    Status des Kopiervorgangs eines RCL-Programms von
    der Compact Flash Karte auf den internen Speicher
    der Steuerung
    - TRUE: Kopiervorgang war erfolgreich
    - FALSE: Kopiervorgang ist fehlgeschlagen oder wurde nicht
    gestartet *)
    ProgLoadAck: BOOL := FALSE;

```

Abb.7-19: Datentypen MB_KINEMATICS_RCL_STATUS (Teil 1)

```
52          (* Status of the copying from the control's internal
53          memory to the Compact Flash card
54          - TRUE: copying was successful
55          - FALSE: copying failed OR was not startet
56          ------------------------------------------------
57          Status des Kopiervorgangs eines RCL-Programms vom
58          internen Speicher der Steuerung auf der Compact Flash
59          Karte
60          - TRUE: Kopiervorgang war erfolgreich
61          - FALSE: Kopiervorgang ist fehlgeschlagen oder wurde nicht
62          gestartet *)
63          ProgSaveAck: BOOL := FALSE;
64
65          (* Status of the compile process of a RCL program
66          - TRUE: compile process was successful
67          - FALSE: compile process failed OR was not startet
68          ------------------------------------------------
69          Status des Kompiliervorgangs eines RCL-Programms
70          - TRUE: Kompiliervorgangs war erfolgreich
71          - FALSE: Kompiliervorgangs ist fehlgeschlagen oder wurde nicht
72          gestartet *)
73          ProgCompileAck: BOOL := FALSE;
74      END_STRUCT
75  END_TYPE
```

Abb.7-20: Datentypen MB_KINEMATICS_RCL_STATUS (Teil 2)

PointModeDone Dieses Element signalisiert, ob der zuletzt gestartet Befehl zum Schreiben oder Lesen eines Punktes oder einer Punkte-/Programmliste erfolgreich war (= TRUE) oder nicht (= FALSE).

PointList Dieses Element zeigt die zuletzt ausgelesene Punkteliste.

ProgList Dieses Element zeigt die zuletzt ausgelesene Programmliste.

Point Dieses Element zeigt den zuletzt ausgelesenen Punkt.

ProgNameAck Diese Element signalisiert, ob das Schreiben des unter `arKinCtrl_gb[].RCLMode.ProgName` angegebenen Namens erfolgreich war (= TRUE) oder nicht (= FALSE).

ProgLoadAck Diese Element signalisiert, ob das Laden des unter `arKinCtrl_gb[].RCLMode.ProgName` angegebenen Programms in den internen Speicher der Steuerung erfolgreich war (= TRUE) oder nicht (= FALSE).

ProgSaveAck Diese Element signalisiert, ob das Speichern des unter `arKinCtrl_gb[].RCLMode.ProgName` angegebenen Programms auf der Compact Flash Karte erfolgreich war (= TRUE) oder nicht (= FALSE).

ProgCompileAck Diese Element signalisiert, ob das Kompilieren des unter `arKinCtrl_gb[].RCLMode.ProgName` angegebenen Programms auf der Steuerung erfolgreich war (= TRUE) oder nicht (= FALSE).

Bibliothek ML_KinTech.compiled-library

System IndraMotion MLC

MB_KINEMATICS_DIAGNOSIS Struktur

Kurzbeschreibung

Die MB_KINEMATICS_DIAGNOSIS Struktur ist eine Unterstruktur von MB_KINEMATICS_STATUS_TYPE01 und enthält Warnungs- und Fehlerausgänge.

Das folgende Bild zeigt die MB_KINEMATICS_DIAGNOSIS-Struktur:

MB_KINEMATICS_DIAGNOSIS

```
1  TYPE MB_KINEMATICS_DIAGNOSIS :
2    STRUCT
3      (* Warning, copy of KinData[].Kin_Warning OR arAxisStatus_gb[].Diag.Warning
4      ------------------------------------------------------------------------------
5      Warnung, Kopie von KinData[].Kin_Warning  ODER arAxisStatus_gb[].Diag.Warning*)
6      Warning: BOOL := FALSE;
7
8      (* Error -> KinData[].Kin_Error OR Error from an FB in the KinInterface
9      ------------------------------------------------------------------------------
10     Fehler -> KinData[].Kin_Error ODER Fehler von einem FB im KinInterface *)
11     Error: BOOL := FALSE;
12
13     (* Error type (internal used FB)  | Fehlertyp (intern genutzter FB)
14     - only valid when Error = TRUE    | - gueltig wenn Error = TRUE *)
15     ErrorID: ERROR_CODE:=NONE_ERROR;
16
17     (* Error ID (internal used FB) | Fehler-ID (intern genutzter FB)
18     - only valid when Error = TRUE | - gueltig wenn Error = TRUE *)
19     ErrorIdent: ERROR_STRUCT := ( Table:=   NO_TABLE_USED, Additionall:=0, Additional2:=0) ;
20
21     (* Diagnosis number of the kinematic  | Diagnosenummer der Kinematik
22     - only valid in operating mode        | - nur gueltig im Betriebsmodus
23     - refresh rate controlled with        | - Auffrischrate gesteuert mit
24                    arKinCtrl[].Admin.DiagNbrRefreshTime *)
25     Number: DWORD := 0;
26
27     (* Diagnosis message of the kinemaitc | Diagnosemeldung der Kinematik
28     - only valid in operating mode        | - nur gueltig im Betriebsmodus
29     - refreshed when "Number" changes     | - Aufgefrischt bei Aenderung von "Number" *)
30     Message: STRING(60) := '';
31   END_STRUCT
32 END_TYPE
```

Abb.7-21: MB_KINEMATICS_DIAGNOSIS

Mit `arKinCtrl_gb[].Admin.DiagNbrRefreshTime` wird die Aktualisierungsrate des Elementes "Number" gesteuert. Sobald sich das Element "Number" ändert, wird der Text "Message" neu gelesen. Mit `arKinCtrl_gb[].Admin.DiagNbrRefreshTime` kann auch das Lesen der Diagnoseinformationen abgeschaltet werden (DiagNbrRefreshTime := T#0ms).

Warning Dieses Element stellt eine Kopie von `KinData[].Kin_Warning` oder von `arAxisStatus_gb[].Diag.Warning` dar und signalisiert, dass die Kinematik oder deren zugehörigen Achsen eine Warnung ausgibt.

Error Dieses Element signalisiert, dass die Kinematik, deren zugehörige Achsen oder ein interner Funktionsbaustein des Kinematik-Interfaces einen Fehler meldet.

ErrorID Dieses Element gibt den Fehlertyp der Kinematik, einer zugehörigen Achse oder eines internen Funktionsbausteins aus.

ErrorIdent Dieses Element gibt die Fehler-ID der Kinematik, einer zugehörigen Achse oder eines internen Funktionsbausteins aus.

Number Dieses Element gibt die Diagnosenummer der Kinematik, einer zugehörigen Achse oder eines internen Funktionsbausteins aus.

Message Dieses Element gibt die Diagnosemeldung der Kinematik, einer zugehörigen Achse oder eines internen Funktionsbausteins aus.

Bibliothek ML_KinTech.compiled-library

System IndraMotion MLC

MB_KINEMATICS_SETUP_STATUS Struktur

Kurzbeschreibung

Die MB_KINEMATICS_SETUP_STATUS-Struktur ist eine Unterstruktur von MB_KINEMATICS_STATUS_TYPE01 und enthält Steuerungsvariablen für das Vertippen einer Kinematik.

Das folgende Bild zeigt die MB_KINEMATICS_SETUP_STATUS-Struktur:

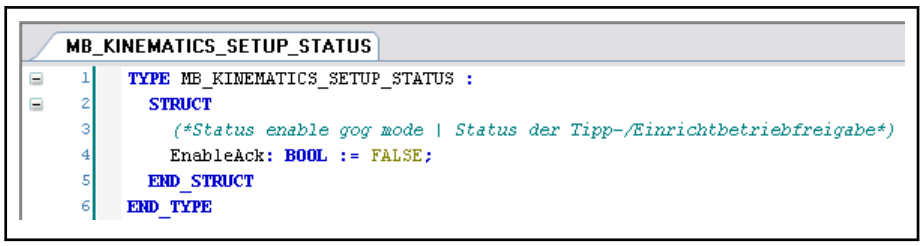


Abb.7-22: MB_KINEMATICS_SETUP_STATUS

EnableAck Dieses Element signalisiert, ob der Tipp-/Einrichtbetrieb aktiv ist (= TRUE) oder nicht (= FALSE).

Bibliothek ML_KinTech.compiled-library

System IndraMotion MLC

7.2.3 Kinematik-Interface Funktionsbausteine

ML_KinematicsInitType01

Kurzbeschreibung

Der Funktionsbaustein **ML_KinematicsInitType01** wird zur Initialisierung des Kinematik-Interfaces für jede Kinematik benutzt. Der Funktionsbaustein muss nur einmal beim Programmstart bzw. bei jeder Phasenumschaltung vom Parametrier- in den Betriebs-Modus aufgerufen werden.

Schnittstellenbeschreibung

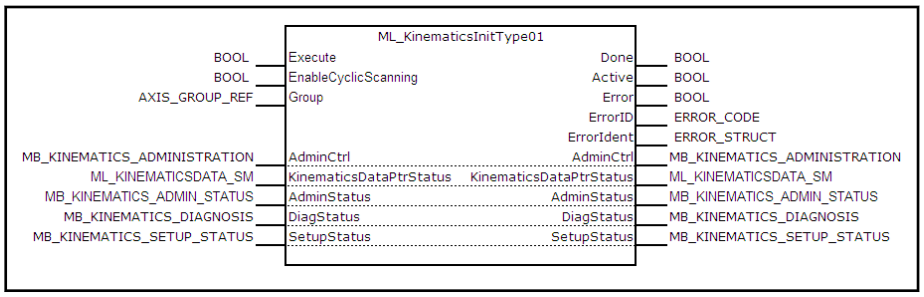


Abb.7-23: Funktionsbaustein ML_KinematicsInitType01

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	Execute	BOOL	Start der Initialisierung durch positive Flanke
	EnableCyclicScanning	BOOL	Zyklische Überprüfung und Übertragung der geänderten Eingänge für die aktuelle Betriebsart
	Group	BOOL	Zu initialisierende Kinematik im Kinematik-Interface
VAR_OUTPUT	Done	BOOL	Wird gesetzt, wenn der FB die Bearbeitung beendet hat
	Active	BOOL	Wird gesetzt, wenn der FB aktiv ist (nicht im Leerlaufbetrieb)
	Error	BOOL	Zeigt an, dass ein Fehler in der FB-Instanz aufgetreten ist
	ErrorID	ERROR_CODE	Kurzer Hinweis zur Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Information zum Fehler

Kinematik-Interface

I/O-Typ	Name	Datentyp	Kommentar
VAR_IN_OUT	AdminCtrl	MB_KINEMATICS_ADMINISTRATION	Verwaltung der Achse arKinCtrl_gb[ ].Admin
	KinematicsDataPtrStatus	ML_KINEMATICSDATA_SM	Zeiger auf KinData dieser Kinematik arKinStatus_gb[ ].PtrAxisData
	AdminStatus	MB_KINEMATICS_ADMIN_STATUS	Status Verwaltung der Kinematik arKinStatus_gb[ ].Admin
	DiagStatus	MB_KINEMATICS_DIAGNOSIS	Diagnoseinformationen der Kinematik arKinStatus_gb[ ].Diag
	SetupStatus	MB_KINEMATICS_SETUP_STATUS	Status Tip-/Einrichtfunktion der Kinematik arKinStatus_gb[ ].SetupMode

Abb.7-24: Schnittstellenvariablen ML_KinematicsInitType01

☞ Es ist nicht möglich anstelle der einzelnen Strukturelemente "AdminCtrl", "KinematicsDataPtrStatus", etc. die kompletten Strukturen "MB_Kinematics_Control_Type01" und "MB_Kinematics_Status_Type01" dem Funktionsbaustein zu übergeben. Somit könnten in der Anwender-Applikation keine speziellen Erweiterungen zum bestehenenden Code hinzugefügt werden. Deshalb werden die benötigten Elemente von "MB_Kinematics_Control_Type01" und "MB_Kinematics_Status_Type01" als separate Eingänge übergeben.

Nähere Details siehe Kap. 7.2.7 "Kinematik-Interface Anwender-Erweiterung" auf Seite 205.

☞ Der Funktionsbaustein ML_KinematicsInitType01 kann nur auf der Steuerung IndraMotion MLC verwendet werden.

Funktionsbeschreibung

Der Funktionsbaustein ML_KinematicsInitType01 initialisiert die folgenden Strukturelemente mit Standardwerten:

Strukturelement	Standard
AdminCtrl._OpMode	ModeCoordAB
AdminCtrl.ClearError	FALSE
AdminCtrl.EnableCyclicScanning	Eingang EnableCyclicScanning
KinematicsDataPtrStatus	ADR(KinData[Group.GroupNo])
AdminCtrl.Group	Eingang Group
AdminStatus._OpModeAck	ModeCoordUngrouped
AdminStatus.Active	FALSE
AdminStatus.Name	' '
AdminStatus.TrafoNb	999
AdminStatus.CmdDone	FALSE
AdminStatus.InBuffer	FALSE

Strukturelement	Standard
AdminStatus.ParamLevelAck	FALSE
AdminStatus.SetConveyorBeltDone[1]	FALSE
AdminStatus.SetConveyorBeltDone[2]	FALSE
AdminStatus.SetConveyorBeltDone[3]	FALSE
AdminStatus.SetConveyorBeltDone[4]	FALSE
AdminStatus.KinTransformPLCDone	FALSE
AdminStatus.RCL.PointModeDone	FALSE
AdminStatus.RCL.ProgNameAck	FALSE
AdminStatus.RCL.ProgLoadAck	FALSE
AdminStatus.RCL.ProgSaveAck	FALSE
AdminStatus.RCL.ProgCompileAck	FALSE
DiagStatus.Error	FALSE
DiagStatus.ErrorID	NONE_ERROR
DiagStatus.ErrorIdent.Table	NO_TABLE_USED
DiagStatus.ErrorIdent.Additional1	0
DiagStatus.ErrorIdent.Additional2	0
DiagStatus.Number	0
DiagStatus.Message	'reading not activated'
SetupStatus.EnableAck	FALSE

Abb.7-25: Durch ML_KinematicsInitType01 initialisierte Strukturelemente

Der Funktionsbaustein ML_KinematicsInitType01 liest folgende Parameter aus und schreibt die Werte auf die folgenden Strukturelemente:

K-Parameter (Name)	Strukturelement
K-0-0031 (Transformationsschema)	AdminStatus.TrafoNb
K-0-0002 (Kinematikname)	AdminStatus.Name
K-0-0008 (Achskonfigurationsliste)	nur für interne Zwecke benötigt

Abb.7-26: Durch ML_KinematicsInitType01 ausgelesene Parameter

EnableCyclicScanning: Der Eingang "EnableCyclicScanning" aktiviert die zyklische Überprüfung und Übertragung der geänderten Eingänge für die aktuelle Betriebsart. Dies ist relevant für die Werte in `arKinCtrl_gb[]` mit der Bemerkung: "Zyklisch überprüft und übertragen bei Änderung.". Der Standardwert ist TRUE.

Group: Der Eingang "Group" gibt an, für welche Kinematik die Initialisierung durchgeführt werden soll.

Kinematik-Interface

Fehlerbehandlung

Der Funktionsbaustein erzeugt in DiagStatus.ErrorIdent.Additional1/Additional2 für die Tabelle "F_RELATED_TABLE", 16#0170 die folgenden Fehlermeldungen:

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR, (16#0005)	16#0006	16#0000	Ungültiger Zustand des Funktionsbausteins
INPUT_INVALID_ERROR, (16#0001)	16#2600	16#0000	ungültige Kinematiknummer "Group.GroupNo"
INPUT_INVALID_ERROR, (16#0001)	16#2600	16#0001	ungültige Steuerungsnummer "Group.CntrlNo". Diese muss auf LOCAL_CNTRL gesetzt sein.

Abb.7-27: Fehlercodes ML_KinematicsInitType01

ML_KinematicsInterfaceType01

Kurzbeschreibung

Der Funktionsbaustein **ML_KinematicsInterfaceType01** wird zur Konfiguration des Kinematik-Interfaces für jede Kinematik benutzt. Dieser Funktionsbaustein muss zyklisch aufgerufen werden, solange man sich im Betriebs-Modus befindet.

Interfacebeschreibung

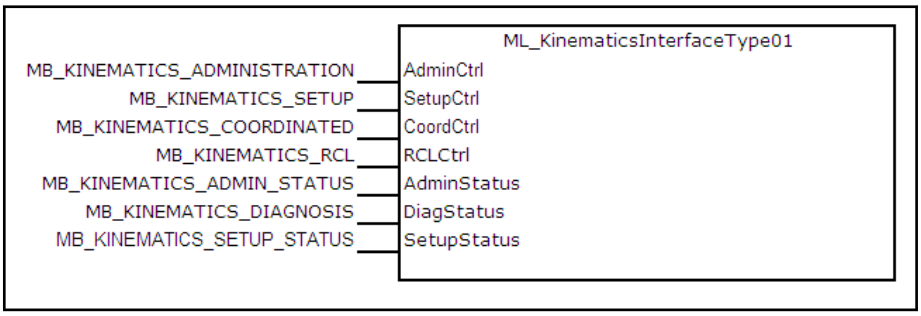


Abb.7-28: Funktionsbaustein ML_KinematicsInterfaceType01

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	AdminCtrl	REFERENCE TO MB_KINEMATICS_ADMINISTRATION	Verwaltung der Achse arKinCtrl_gb[ ].Admin
	SetupCtrl	REFERENCE TO MB_KINEMATICS_SETUP	Tip-/Einrichtbetrieb arKinCtrl_gb[ ].SetupMode
	CoordCtrl	REFERENCE TO MB_KINEMATICS_COORDINATED	Betriebsart koordinierte Bewegung arKinCtrl_gb[ ].CoordMode
	RCLCtrl	REFERENCE TO MB_KINEMATICS_RCL	Betriebsarten RCL arKinCtrl_gb[ ].RCLMode
	AdminStatus	REFERENCE TO MB_KINEMATICS_ADMIN_STATUS	Status Verwaltung der Achse arKinStatus_gb[ ].Admin
	DiagStatus	REFERENCE TO MB_KINEMATICS_DIAGNOSIS	Status Diagnose der Achse arKinStatus_gb[ ].Diag
	SetupStatus	REFERENCE TO MB_KINEMATICS_SETUP_STATUS	Status Tip-/Einrichtbetrieb arKinStatus_gb[ ].SetupMode

Abb.7-29: Schnittstellenvariablen ML_KinematicsInterfaceType01

Es ist nicht möglich anstelle der einzelnen Strukturelemente "AdminCtrl", "SetupCtrl", etc. die kompletten Strukturen "MB_Kinematics_Control_Type01" und "MB_Kinematics_Status_Type01" dem Funktionsbaustein zu übergeben. Somit könnten in der Anwender-Applikation keine speziellen Erweiterungen zum bestehenenden Code hinzugefügt werden. Deshalb werden die benötigten Elemente von "MB_Kinematics_Control_Type01" und "MB_Kinematics_Status_Type01" als separate Eingänge übergeben.

Nähere Details siehe Kap. 7.2.7 "Kinematik-Interface Anwender-Erweiterung" auf Seite 205.

Zur Performanceoptimierung sind die Strukturen statt als VAR_IN_OUT als VAR_INPUT mit REFERENCE TO angeschlossen. Damit reicht es einmalig die Eingänge zu initialisieren. Beim zyklischen Aufruf des Funktionsbausteins müssen damit die Strukturen nicht übergeben werden.

Der Funktionsbaustein ML_KinematicsInterfaceType01 kann nur auf der Steuerung IndraMotion MLC verwendet werden.

Kinematik-Interface

Fehlerbehandlung **Der Funktionsbaustein erzeugt in DiagStatus.ErrorIdent.Additional1/Additional2 für die Tabelle "F_RELATED_TABLE", 16#0170 die folgenden Fehlermeldungen:**

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR, (16#0005)	16#0006	16#0000	Ungültiger interner Zustand
STATE_MACHINE_ERROR, (16#0005)	16#0006	16#0001	Ungültiger Zustand des Funktionsbausteins
RESOURCE_ERROR, (16#0003)	16#2600	16#0001	ungültige Steuerungsnummer "Group.CntrlNo". Diese muss auf LOCAL_CNTRL gesetzt sein.
DEVICE_ERROR, (16#0008)	16#2601	16#0000	Kinematik meldet eines Fehler in der KinData-Struktur
INPUT_INVALID_ERROR, (16#0001)	16#2602	Nummer des OpMode	ungültige Betriebsart ausgewählt
INPUT_INVALID_ERROR, (16#0001)	16#2603	Nummer des OpMode	ungültige, nicht unterstützte Betriebsart aktiviert
INPUT_RANGE_ERROR, (16#0006)	16#2604	16#0000	mindesten eine Eingangsreferenz (AdminCtrl, AdminStatus, usw.) ist nicht gesetzt
ACCESS_ERROR, (16#0004)	16#2605	16#0000	referenzieren der Achse(n) nicht möglich und nicht notwendig
ACCESS_ERROR, (16#0004)	16#2605	Nummer der zu referenzierenden Achse	referenzieren der Achse(n) nicht möglich
RESOURCE_ERROR, (16#0003)	16#2606	16#0000	Betriebsart wurde während des SetupModes verändert
OTHER_ERROR, (16#7FFE)	16#2607	16#0000	Kinematik konnte nicht gruppiert werden
INPUT_INVALID_ERROR, (16#0001)	16#2609	16#0000	unzulässiges Koordinatensystem ausgewählt
RESOURCE_ERROR, (16#0003)	16#26FF	16#0000	ungültige Robot-Control-Version

Abb.7-30: Fehlercodes ML_KinematicsInterfaceType01

ML_KinJogV1

Kurzbeschreibung Der Funktionsbaustein **ML_KinJogV1** dient dem Vertippen einzelner Achsen oder Achsgruppen einer Kinematik. Dieser Funktionsbaustein muss zyklisch aufgerufen werden.



Nur für Robot-Control V1 zulässig.

Schnittstellenbeschreibung

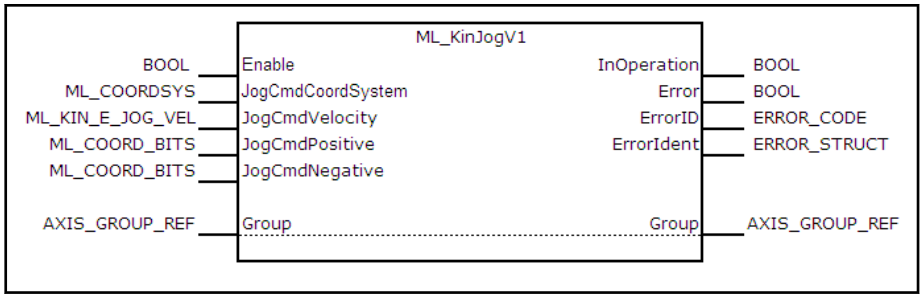


Abb.7-31: Funktionsbaustein ML_KinJogV1

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Bausteins
	JogCmdCoordSystem	ML_COORDSYS	Angabe des Tipp-Koordinatensystems
	JogCmdVelocity	ML_KIN_E_JOG_VEL	Angabe des Tipp-Geschwindigkeit
	JogCmdPositive	ML_COORD_BITS	Enthält pro Koordinate ein Bit, über das die Koordinaten in positiver Richtung vertippt werden kann
	JogCmdNegative	ML_COORD_BITS	Enthält pro Koordinate ein Bit, über das die Koordinaten in negativer Richtung vertippt werden kann
VAR_OUTPUT	InOperation	BOOL	Der Baustein arbeitet an seiner eigentlichen Aufgabe
	Error	BOOL	Zeigt an, dass ein Fehler in der FB-Instanz aufgetreten ist
	ErrorID	ERROR_CODE	Kurzer Hinweis zur Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Information zum Fehler
VAR_IN_OUT	Group	BOOL	Referenz zur Kinematik, welche vertippt werden soll

Abb.7-32: Schnittstellenvariablen ML_KinJogV1

Name	Typ	Min-Wert	Max-Wert	Standardwert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
JogCmdCoordSystem	ML_COORDSYS			KIN_CORD_SYS_JOINT_COORD	kontinuierlich
JogCmdVelocity	ML_KIN_E_JOG_VEL			KIN_JOG_STOP	kontinuierlich
JogCmdPositive	ML_COORD_BITS			16(FALSE)	kontinuierlich
JogCmdNegative	ML_COORD_BITS			16(FALSE)	kontinuierlich

Abb.7-33: Eingangsverhalten ML_KinJogV1

Der Funktionsbaustein ML_KinJogV1 kann nur auf der Steuerung IndraMotion MLC verwendet werden.

Funktionsbeschreibung

Mithilfe des Funktionsbausteins ML_KinJog ist es möglich, eine Kinematik mit der Robot-Control V1 zu vertippen. Hierbei erfolgt das Vertippen über die KinData-Schnittstelle `KinData[].stJogCmd_q`. Nach dem erfolgreichen Start des Funktionsbausteins gruppiert dieser automatisch die zur Kinematik zugehörigen Achsen.

Kinematik-Interface

JogCmdCoordSystem Dieser Eingang gibt das Koordinatensystem an, in welchem vertippt werden soll.

Der Eingang kann folgende Werte annehmen:

- KIN_CORD_SYS_JOINT_COORD: Achs-Koordinatensystem
- KIN_CORD_SYS_ORIGIN_COORD: Welt-Koordinatensystem
- KIN_CORD_SYS_WORKPIECE_COORD: Werkstück-Koordinatensystem
- KIN_CORD_SYS_GRIPPER_COORD: Greifer-Koordinatensystem

JogCmdVelocity Dieser Eingang gibt die Art des Vertippens an (inkrementell langsam und schnell, kontinuierlich langsam und schnell).

Der Eingang kann folgende Werte annehmen:

- KIN_JOG_STOP
  Tippen deaktiviert
- KIN_JOG_STEP_SMALL
  inkrementelles Tippen in kleinen Schritten
- KIN_JOG_STEP_LARGE
  inkrementelles Tippen in großen Schritten
- KIN_JOG_CONT_SLOW
  kontinuierliches Tippen mit langsamer Geschwindigkeit
- KIN_JOG_CONT_LARGE
  kontinuierliches Tippen mit schneller Geschwindigkeit

JogCmdPositive Dieser Eingang aktiviert die Verfahrbewegung in positive Richtung für die entsprechende Achse.

JogCmdNegative Dieser Eingang aktiviert die Verfahrbewegung in negative Richtung für die entsprechende Achse.

Fehlerbehandlung **Der Funktionsbaustein erzeugt in ErrorIdent.Additional1/Additional2 für die Tabelle "F_RELATED_TABLE", 16#0170 die folgenden Fehlermeldungen:**

ErrorID	Additional1	Additional2	Beschreibung
RESOURCE_ERROR, (16#0003)	16#26FF	16#0000	ungültige Robot-Control-Version

Abb.7-34: Fehlercodes ML_KinJogV1

ML_KinJog

Kurzbeschreibung Der Funktionsbaustein **ML_KinJog** dient dem Vertippen einzelner Achsen oder Achsgruppen einer Kinematik. Dieser Funktionsbaustein muss zyklisch aufgerufen werden.

☞ Nur für Robot-Control V2 zulässig.

Schnittstellenbeschreibung

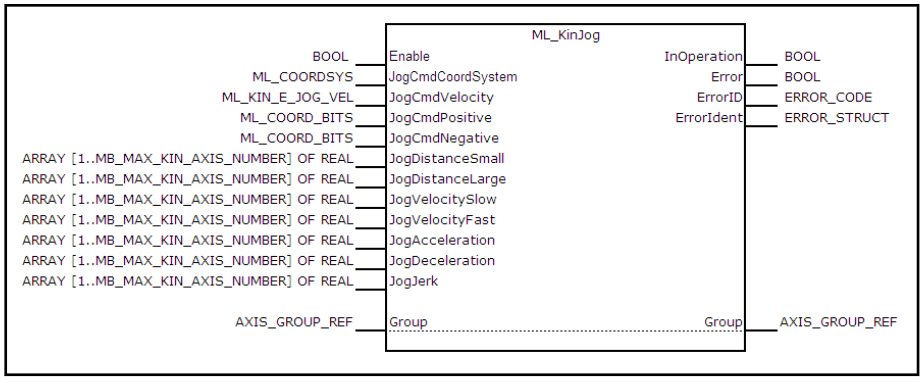


Abb.7-35: Funktionsbaustein ML_KinJog

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	Enable	BOOL	Bearbeitungsfreigabe des Bausteins
	JogCmdCoordSystem	ML_COORDSYS	Angabe des Tipp-Koordinatensystems
	JogCmdVelocity	ML_KIN_E_JOG_VEL	Angabe des Tipp-Geschwindigkeit
	JogCmdPositive	ML_COORD_BITS	Enthält pro Koordinate ein Bit, über das die Koordinaten in positiver Richtung vertippt werden kann
	JogCmdNegative	ML_COORD_BITS	Enthält pro Koordinate ein Bit, über das die Koordinaten in negativer Richtung vertippt werden kann
	JogDistanceSmall	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Angabe der kurzen Tipp-Distanz für jede Achse
	JogDistanceLarge	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Angabe der langen Tipp-Distanz für jede Achse
	JogVelocitySlow	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Angabe der langsamen Tipp-Geschwindigkeit für jede Achse
	JogVelocityFast	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Angabe der schnellen Tipp-Geschwindigkeit für jede Achse
	JogAcceleration	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Angabe der Beschleunigung für den Tipp-Betrieb für jede Achse
	JogDeceleration	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Angabe der Bremsverzögerung für den Tipp-Betrieb für jede Achse
	JogJerk	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Angabe des Rucks für den Tipp-Betrieb für jede Achse

Kinematik-Interface

I/O-Typ	Name	Datentyp	Kommentar
VAR_OUTPUT	InOperation	BOOL	Der Baustein arbeitet an seiner eigentlichen Aufgabe
	Error	BOOL	Zeigt an, dass ein Fehler in der FB-Instanz aufgetreten ist
	ErrorID	ERROR_CODE	Kurzer Hinweis zur Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Information zum Fehler
VAR_IN_OUT	Group	BOOL	Referenz zur Kinematik, welche vertippt werden soll

Abb.7-36: Schnittstellenvariablen ML_KinJog

Name	Typ	Min-Wert	Max-Wert	Standardwert	Übernahme
Enable	BOOL			FALSE	kontinuierlich
JogCmdCoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	kontinuierlich
JogCmdVelocity	ML_KIN_E_JOG_VEL			KIN_JOG_STOP	kontinuierlich
JogCmdPositive	ML_COORD_BITS			16(FALSE)	kontinuierlich
JogCmdNegative	ML_COORD_BITS			16(FALSE)	kontinuierlich
JogDistanceSmall	ARRAY [1..MB_MAX_KIN_NUMBER] OF REAL	>0.0		1.0	kontinuierlich
JogDistanceLarge	ARRAY [1..MB_MAX_KIN_NUMBER] OF REAL	>0.0		10.0	kontinuierlich
JogVelocitySlow	ARRAY [1..MB_MAX_KIN_NUMBER] OF REAL	>0.0		10.0	kontinuierlich
JogVelocityFast	ARRAY [1..MB_MAX_KIN_NUMBER] OF REAL	>0.0		25.0	kontinuierlich
JogAcceleration	ARRAY [1..MB_MAX_KIN_NUMBER] OF REAL	>0.0		370.0	kontinuierlich
JogDeceleration	ARRAY [1..MB_MAX_KIN_NUMBER] OF REAL	>0.0		370.0	kontinuierlich
JogJerk	ARRAY [1..MB_MAX_KIN_NUMBER] OF REAL	>0.0		0.0	kontinuierlich

Abb.7-37: Eingangsverhalten ML_KinJog

☞ Der Funktionsbaustein ML_KinJog kann nur auf der Steuerung IndraMotion MLC verwendet werden.

Funktionsbeschreibung

Mithilfe des Funktionsbausteins ML_KinJog ist es möglich, eine Kinematik mit der Robot-Control V2 zu vertippen. Hierbei entspricht das Vertippen im kartesischen Koordinatenraum einer linearen Verfahrbewegung, wohingegen das Vertippen im Achskoordinatenraum einer PTP Verfahrbewegung gleicht. Nach dem erfolgreichen Start des Funktionsbausteins gruppiert dieser automatisch die zur Kinematik zugehörigen Achsen.

JogCmdCoordSystem

Dieser Eingang gibt das Koordinatensystem an, in welchem vertippt werden soll.

Der Eingang kann folgende Werte annehmen:

- CS_MACHINE_COORDINATES: Welt-Koordinatensystem
- CS_AXES_COORDINATES: Achs-Koordinatensystem
- CS_BASE_COORDINATES: Basis-Koordinatensystem
- CS_PRODUCT_COORDINATES_1: Produkt-Koordinatensystem 1
- CS_PRODUCT_COORDINATES_2: Produkt-Koordinatensystem 2
- ...
- CS_PRODUCT_COORDINATES_16: Produkt-Koordinatensystem 16

JogCmdVelocity

Dieser Eingang gibt die Art des Vertippens an (inkrementell kurz und lang, kontinuierlich langsam und schnell).

Der Eingang kann folgende Werte annehmen:

- KIN_JOG_STOP

  Tippen deaktiviert
- KIN_JOG_STEP_SMALL

  inkrementelles Tippen in kleinen Schritten
- KIN_JOG_STEP_LARGE

  inkrementelles Tippen in großen Schritten
- KIN_JOG_CONT_SLOW

  kontinuierliches Tippen mit langsamer Geschwindigkeit
- KIN_JOG_CONT_LARGE

  kontinuierliches Tippen mit schneller Geschwindigkeit

JogCmdPositive

Dieser Eingang aktiviert die Verfahrbewegung in positive Richtung für die entsprechende Achse.

JogCmdNegative

Dieser Eingang aktiviert die Verfahrbewegung in negative Richtung für die entsprechende Achse.

JogDistanceSmall, JogDistanceLarge, JogVelocitySlow, JogVelocityFast, JogAcceleration, JogDeceleration, JogJerk

Diese Eingänge definieren die Tipp-Distanz, Tipp-Geschwindigkeit, Tipp-Beschleunigung sowie Tipp-Ruck für inkrementelles und kontinuierliches Tippen. Deren Einheiten sind abhängig von "JogCmdCoordSystem" und entsprechen den Einheiten der Kinematik (bei kartesischem Koordinatensystem) bzw. den Einheiten der Achsen (bei Achskoordinatensystem).

☞ Das kontinuierliche Tippen basiert auf dem Absetzen mehrerer Verfahrbefehle, solange das entsprechende Bit des Eingangs "JogCmdPositive" bzw. "JogCmdNegative" gesetzt ist. Bewegt man sich mit der Kinematik auf den Rand des Arbeitsraumes zu so kann es frühzeitig zu der Fehlermeldung kommen, dass der kommandierte Punkt durch die Kinematik nicht erreichbar ist. In diesem Fall ist der Fehler zu quittieren und mit verminderter Geschwindigkeit bzw. durch inkrementelles Tippen weiter zum gewünschten Punkt zu fahren.

Kinematik-Interface

Fehlerbehandlung **Der Funktionsbaustein erzeugt in ErrorIdent.Additional1/Additional2 für die Tabelle "F_RELATED_TABLE", 16#0170 die folgenden Fehlermeldungen:**

ErrorID	Additional1	Additional2	Beschreibung
RESOURCE_ERROR, (16#0003)	16#26FF	16#0000	ungültige Robot-Control-Version

Abb.7-38: Fehlercodes ML_KinJog

7.2.4 Betriebsarten

Überblick

Es gibt zwei Methoden um eine Betriebsart zu aktivieren:

1. **Auswahl über Enum-Werte**
   Zuweisen eines Wertes vom TYPE MB_KINEMATICS_MODE:
   `arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordAH;`
2. **Benutzung des Bit-Zugriffs**
   Setzen eines Bits über die Bit-Zugriffs Funktionalität.
   `arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_AH:= TRUE;`

☞ Es darf immer nur 1 Bit in `arKinCtrl_gb[].Admin._OpMode` gesetzt sein. Das Setzen mehrerer Bits verursacht einen Fehler im Kinematik-Interface.

Bevor eine Betriebsart aktiviert werden kann, müssen jedem Attribut zuerst Werte zugewiesen werden. Alle Attribute haben Stadardwerte. Einige haben Werte ungleich Null, während andere als 0 definiert sind und, aufgrund der speziellen Anforderungen, ihnen ein Wert zugewiesen werden muss.

☞ Nur die Attribute (z. B. Point, Velocity), die benutzt werden oder deren Standardwert geändert wurde, müssen definiert werden, bevor der aktuelle Betriebsartenwechsel ausgeführt wird.

Der folgende Abschnitt beschreibt die Betriebsarten, die vom Kinematik-Interface unterstützt werden und die Attribute, die zugewiesen werden können.

☞ Das Kinematik-Interface nutzt intern das Achs-Interface, um einige Umschaltungen durchzuführen (z. B. gruppieren / degruppieren aller Achsen, referenzieren von Achsen) .

Kinematik Bereit, Antriebe Bereit

Die Aktivierung dieser Betriebsart degruppiert die Kinematik und alle, der Kinematik zugehörigen Achsen werden in AB (Antrieb Bereit) geschalten. In dieser Betriebsart ist das Kinematik-Interface deaktiviert und die Kinematik befindet sich im PLCOpen-Zustand "StandStil incomplete". Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordAB;
```

oder

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_AB := TRUE;
```

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Ja
	Admin.StopDeceleration	REAL	999	Ja
	Admin.StopJerk	REAL	9999	Ja
arKinStatus_gb[ ]	Admin.MODE_COORD_AB	BOOL		entfällt

Abb.7-39: Attribute Kinematik Bereit, Antriebe Bereit

Bei der Anwahl der Betriebsart "ModeCoordAB" gibt das Kinematik-Interface in der Statusstruktur die Betriebsart "ModeCoordUngrouped" aus!

Kinematik Bereit, Antriebe Halt

Die Aktivierung dieser Betriebsart degruppiert die Kinematik und alle, der Kinematik zugehörigen Achsen werden in AH (Antrieb Halt) geschaltet. In dieser Betriebsart ist das Kinematik-Interface deaktiviert und die Kinematik befindet sich im PLCOpen-Zustand "StandStil incomplete". Die Achsen werden in den PLCOpen-Zustand "StandStil" überführt, d. h. sie sind mithilfe des Achs-Interfaces oder separater PLCOpen-Funktionsbausteine kommandierbar. Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordAH;
```

oder

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_AH := TRUE;
```

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Ja
	Admin.StopDeceleration	REAL	999.0	Ja
	Admin.StopJerk	REAL	9999.0	Ja
arKinStatus_gb[ ]	Admin.MODE_COORD_AH	BOOL		entfällt

Abb.7-40: Attribute Kinematik Bereit, Antriebe Halt

Bei der Anwahl der Betriebsart "ModeCoordAH" gibt das Kinematik-Interface in der Status-Struktur die Betriebsart "ModeCoordUngrouped" aus!

Kinematik Referenzieren

Die Aktivierung dieser Betriebsart führt das Referenzieren aller, der Kinematik zugehörigen Achsen aus. Dieses Referenzieren erfolgt über den Setup-Mode der Achs-Interface. Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordHoming;
```

Kinematik-Interface

oder

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_HOMING := TRUE;
```

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Ja
	Admin.HomingSequence	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	MB_MAX_KIN_AXIS_NUMBER(0.0)	Ja
arKinStatus_gb[ ]	Admin.MODE_COORD_HOMING	BOOL		entfällt
	Admin.HomingAck	BOOL		entfällt

Abb.7-41: Attribute Kinematik Referenzieren

☞ Ist die Kinematik bei Anwahl der Betriebsart "ModeCoordHoming" bereits gruppiert, so ist ein erneutes Referenzieren nicht möglich und das Kinematik-Interface gibt eine Fehlermeldung aus.

Absolutes PTP Positionieren

Mit Aktivierung dieser Betriebsart wird eine absolute PTP-Bewegung auf eine vorher festgelegt Zielposition ausgeführt. Die Kinematik wird gruppiert und geht in den PLCOpen-Zustand "GroupMoving" über. Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordPosDirAbs;
```

oder

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_POS_DIR_ABS  :=
TRUE;
```

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Ja
	CoordMode.Point	ARRAY [1..16] OF REAL	16(0.0)	Ja
	CoordMode.Velocity	REAL	10.0	Nein
	CoordMode.Acceleration	REAL	10.0	Nein
	CoordMode.Deceleration	REAL	10.0	Nein
	CoordMode.Jerk	REAL	100	Nein
	CoordCoordModeCtrl.CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	Nein
	CoordMode.SlopeType	ML_SLOPE_TYPE	ST_BLOCK_SLOPE	Nein
	CoordMode.SetMode	ML_SET_MODE	SM_TO	Nein
	CoordMode.BlendingRadius	REAL	0.0	Nein
arKinStatus_gb[ ]	Admin.MODE_COORD_POS_DIR_ABS	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt
	Admin.InBuffer	BOOL		entfällt

Abb.7-42: Attribute Absolutes PTP Positionieren

Relatives PTP Positionieren

Mit Aktivierung dieser Betriebsart wird eine relative PTP-Bewegung um eine vorher festgelegt Positionsdifferenz ausgeführt. Die Kinematik wird gruppiert und geht in den PLCOpen-Zustand "GroupMoving" über. Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordPosDirRel;
```

oder

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_POS_DIR_REL  := TRUE;
```

Kinematik-Interface

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Ja
	CoordMode.Point	ARRAY [1..16] OF REAL	16(0.0)	Ja
	CoordMode.Velocity	REAL	10.0	Nein
	CoordMode.Acceleration	REAL	10.0	Nein
	CoordMode.Deceleration	REAL	10.0	Nein
	CoordMode.Jerk	REAL	100	Nein
	CoordCoordModeCtrl.CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	Nein
	CoordMode.SlopeType	ML_SLOPE_TYPE	ST_BLOCK_SLOPE	Nein
	CoordMode.SetMode	ML_SET_MODE	SM_TO	Nein
	CoordMode.BlendingRadius	REAL	0.0	Nein
arKinStatus_gb[ ]	Admin.MODE_COORD_POS_DIR_REL	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt
	Admin.InBuffer	BOOL		entfällt

Abb.7-43: Attribute Relatives PTP Positionieren

Absolutes Lineares Positionieren

Mit Aktivierung dieser Betriebsart wird eine absolute Linearbewegung auf eine vorher festgelegt Zielposition ausgeführt. Die Kinematik wird gruppiert und geht in den PLCOpen-Zustand "GroupMoving" über. Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordPosLinAbs;
```

oder

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_POS_LIN_ABS  := TRUE;
```

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Ja
	CoordMode.Point	ARRAY [1..16] OF REAL	16(0.0)	Ja
	CoordMode.Velocity	REAL	10.0	Nein
	CoordMode.Acceleration	REAL	10.0	Nein
	CoordMode.Deceleration	REAL	10.0	Nein
	CoordMode.Jerk	REAL	100.0	Nein
	CoordMode.CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	Nein
	CoordMode.SlopeType	ML_SLOPE_TYPE	ST_BLOCK_SLOPE	Nein
	CoordMode.SetMode	ML_SET_MODE	SM_TO	Nein
	CoordMode.BlendingRadius	REAL	0.0	Nein
arKinStatus_gb[ ]	Admin.MODE_COORD_POS_LIN_ABS	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt
	Admin.InBuffer	BOOL		entfällt

Abb.7-44: Attribute Absolutes Lineares Positionieren

Relatives Lineares Positionieren

Mit Aktivierung dieser Betriebsart wird eine relative Linearbewegung um eine vorher festgelegt Positionsdifferenz ausgeführt. Die Kinematik wird gruppiert und geht in den PLCOpen-Zustand "GroupMoving" über. Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordPosLinRel;
```

oder

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_POS_LIN_REL  := TRUE;
```

Kinematik-Interface

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Ja
	CoordMode.Point	ARRAY [1..16] OF REAL	16(0.0)	Ja
	CoordMode.Velocity	REAL	10.0	Nein
	CoordMode.Acceleration	REAL	10.0	Nein
	CoordMode.Deceleration	REAL	10.0	Nein
	CoordMode.Jerk	REAL	100.0	Nein
	CoordMode.CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	Nein
	CoordMode.SlopeType	ML_SLOPE_TYPE	ST_BLOCK_SLOPE	Nein
	CoordMode.SetMode	ML_SET_MODE	SM_TO	Nein
	CoordMode.BlendingRadius	REAL	0.0	Nein
arKinStatus_gb[ ]	Admin.MODE_COORD_POS_LIN_REL	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt
	Admin.InBuffer	BOOL		entfällt

Abb.7-45: Attribute Relatives Lineares Positionieren

Absolutes Zirkulares Positionieren

Mit Aktivierung dieser Betriebsart wird eine absolute Zirkularbewegung auf eine vorher festgelegt Zielposition und um einen ebenfalls festgelegten Zwischenpunkt ausgeführt. Die Kinematik wird gruppiert und geht in den PLCOpen-Zustand "GroupMoving" über. Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordPosCircAbs;
```

oder

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_POS_CIRC_ABS :=
TRUE;
```

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Ja
	CoordMode.AuxPoint	ARRAY [1..16] OF REAL	16(0.0)	Nein
	CoordMode.Point	ARRAY [1..16] OF REAL	16(0.0)	Ja
	CoordMode.Velocity	REAL	10.0	Nein
	CoordMode.Acceleration	REAL	10.0	Nein
	CoordMode.Deceleration	REAL	10.0	Nein
	CoordMode.Jerk	REAL	100.0	Nein
	CoordMode.CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	Nein
	CoordMode.SlopeType	ML_SLOPE_TYPE	ST_BLOCK_SLOPE	Nein
	CoordMode.SetMode	ML_SET_MODE	SM_TO	Nein
	CoordMode.BlendingRadius	REAL	0.0	Nein
	CoordMode.CircularCircMode	ML_CIRCMODES	CM_BOARDER	Nein
arKinStatus_gb[ ]	Admin.MODE_COORD_POS_CIRC_ABS	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt
	Admin.InBuffer	BOOL		entfällt

Abb.7-46: Attribute Absolutes Zirkulares Positionieren

Relatives Zirkulares Positionieren

Mit Aktivierung dieser Betriebsart wird eine relative Zirkularbewegung um eine vorher festgelegt Positionsdifferenz und um eine ebenfalls festgelegte Zwischenpositionsdifferenz ausgeführt. Die Kinematik wird gruppiert und geht in den PLCOpen-Zustand "GroupMoving" über. Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordPosCircRel;
```

oder

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_POS_CIRC_REL := TRUE;
```

Kinematik-Interface

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Ja
	CoordMode.CircularAuxPoint	ARRAY [1..16] OF REAL	16(0.0)	Nein
	CoordMode.Point	ARRAY [1..16] OF REAL	16(0.0)	Ja
	CoordMode.Velocity	REAL	10.0	Nein
	CoordMode.Acceleration	REAL	10.0	Nein
	CoordMode.Deceleration	REAL	10.0	Nein
	CoordMode.Jerk	REAL	100.0	Nein
	CoordMode.CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	Nein
	CoordMode.SlopeType	ML_SLOPE_TYPE	ST_BLOCK_SLOPE	Nein
	CoordMode.SetMode	ML_SET_MODE	SM_TO	Nein
	CoordMode.BlendingRadius	REAL	0.0	Nein
	CoordMode.CircularCircMode	ML_CIRCMODES	CM_BOARDER	Nein
arKinStatus_gb[ ]	Admin.MODE_COORD_POS_CIRC_REL	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt
	Admin.InBuffer	BOOL		entfällt

Abb.7-47: Attribute Relatives Zirkulares Positionieren

Kinematik Stopp

Die Aktivierung dieser Betriebsart stoppt die Kinematik unter Aufrechterhaltung der Gruppierung. Die Kinematik geht in den PLCOpen-Zustand "Stopping" über. Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordStopping;
```

oder

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_STOPPING := TRUE;
```

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Ja
	AdminCtrl.StopDeceleration	REAL	999.0	Ja
	AdminCtrl.StopJerk	REAL	9999.0	Ja

Element	Name	Typ	Standard	Zyklisch gescannt
arKinStatus_gb[ ]	Admin.MODE_COORD_STOPP-ING	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt
	Admin.InBuffer	BOOL		entfällt

Abb.7-48: Attribute Kinematik Stopp

☞ Die Betriebsart "ModeCoordStopping" ist nur für Robot-Control V2 verfügbar. Wird sie mit Robot-Control V1 angewählt, so gibt das Kinematik-Interface einen Fehler aus!

Betriebsart "Interrupted"

Die Aktivierung dieser Betriebsart unterbricht die Bewegung der Kinematik unter Aufrechterhaltung der Gruppierung. Die Kinematik bleibt im PLCOpen-Zustand "GroupMoving". Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordInterrupted;
```

oder

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_INTERRUPTED := TRUE;
```

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	Admin.ChangeOpMode	MB_KINEMATICS_AD-MIN_CHANGE_OPMODE	NoAction	Ja
	CoordMode.Deceleration	REAL	10.0	Nein
	CoordMode.Jerk	REAL	100.0	Nein
arKinStatus_gb[ ]	Admin.MODE_COORD_INTER-RUPTED	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt
	Admin.InBuffer	BOOL		entfällt

Abb.7-49: Attribute Betriebsart "Interrupted"

☞ Die Betriebsart "ModeCoordInterrupted" ist nur für Robot-Control V2 verfügbar. Wird sie mit Robot-Control V1 angewählt, so gibt das Kinematik-Interface einen Fehler aus!

Betriebsart "Continue"

Die Aktivierung dieser Betriebsart startet eine zuvor unterbrochene Bewegung. Die Kinematik befindet sich im PLCOpen-Zustand "GroupMoving". Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordContinue;
```

oder

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_CONTINUE := TRUE;
```

Kinematik-Interface

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Ja
arKinStatus_gb[ ]	Admin.MODE_COORD_CONTINUE	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt
	Admin.InBuffer	BOOL		entfällt

Abb.7-50: Attribute Betriebsart "Continue"

☞ Die Betriebsart "ModeCoordContinue" ist nur für Robot-Control V2 verfügbar. Wird sie mit Robot-Control V1 angewählt, so gibt das Kinematik-Interface einen Fehler aus!

Betriebsart "Externer Funktionsbaustein"

Die Aktivierung dieser Betriebsart gruppiert die Kinematik und wartet auf einen externen Bewegungsfehl. Die Kinematik geht in den PLCOpen-Zustand "StandStill Complete" über. Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordExternalFB;
```

oder

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_EXTERNAL_FB  := TRUE;
```

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Ja
arKinStatus_gb[ ]	Admin.MODE_COORD_EXTERNAL_FB	BOOL		entfällt

Abb.7-51: Attribute Betriebsart "Externer Funktionsbaustein"

Betriebsart "RCL"

Die Aktivierung dieser Betriebsart gruppiert die Kinematik und startet das unter `arKinCtrl_gb[].RCLMode.ProgName` angegebene RCL-Programm. Die Kinematik geht in den PLCOpen-Zustand "ProgrammedMotion" über. Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordRCL;
```

oder

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_RCL := TRUE;
```

Die folgende Tabelle enthält die Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Ja
	RCLCtrl.ProgName	STRING	' '	Nein
arKinStatus_gb[ ]	Admin.MODE_COORD_RCL	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

Abb.7-52: Attribute Betriebsart "RCL"

☞ Die Betriebsart "ModeCoordRCL" ist nur für Robot-Control V1 verfügbar. Wird sie mit Robot-Control V2 angewählt, so gibt das Kinematik-Interface einen Fehler aus!

7.2.5 Globale Variablen

Die Bibliothek ML_KinTech enthält die globale Variablenliste **Global_MB_KinInterfaceVars**.

Die Bibliothek definiert folgende Konstanten:

1. **arK8ConfigList_gb**:

   Diese Array-Konstante beinhaltet Informationen über die Achsen, welche der Kinematik zugeordnet sind:

   - uiNumberOfKinAxis - Gesamtanzahl der Achsen
   - uiNumberOfMoveAxis - Anzahl der Positionier- und Orientierungsachsen
   - uiNumberOfBeltAxis - Anzahl der Bandachsen
   - arAxisType - Achstypen (Positionier-, Orientier-, Bandachse oder keine Achse)
   - arAxisNumber - Achsnummern

2. **arRocoKernel_gb**:

   Diese Array-Konstante beinhaltet Informationen über die Robot-Control-Version der Kinematik

   - 1 - Robot-Control V1
   - 2 - Robot Control V2

3. **arpAxisCtrlAdmin_gb**:

   Diese Pointer-Konstante zeigt auf die AdminCtrl-Struktur des Achs-Interfaces ("MB_AXIS_ADMINISTRATION"). Sie ermöglicht den parallelen Zugriff auf das Achs-Interface sowohl aus dem Kinematik-Interface als auch von einer Anwender-Applikation.

4. **arpAxisCtrlCoordMode_gb**:

   Diese Pointer-Konstante zeigt auf die CoordModeCtrl-Struktur des Achs-Interfaces ("MB_AXIS_COORDINATED"). Sie ermöglicht den parallelen Zugriff auf das Achs-Interface sowohl aus dem Kinematik-Interface als auch von einer Anwender-Applikation..

5. **arpAxisStatusAdmin_gb**:

   Diese Pointer-Konstante zeigt auf die AdminStatus-Struktur des Achs-Interfaces ("MB_AXIS_ADMIN_STATUS"). Sie ermöglicht den parallelen

Zugriff auf das Achs-Interface sowohl aus dem Kinematik-Interface als auch von einer Anwender-Applikation.

6. **arpAxisStatusDiag_gb**:

   Diese Pointer-Konstante zeigt auf die DiagStatus-Struktur des Achs-Interfaces ("MB_AXIS_DIAGNOSIS"). Sie ermöglicht den parallelen Zugriff auf das Achs-Interface sowohl aus dem Kinematik-Interface als auch von einer Anwender-Applikation.

7.2.6 Erste Schritte Kinematik-Interface

Überblick

Wie beim Programmiertemplate "Achs-Interface Beispielprojekt" liefert auch das Kinematik-Interface ein äquivalentes Beispielprojekt, welches beim Einbinden des PlugIns automatisch erstellt wird. Dieses Beispielprojekt ist unter dem Reiter **Logic** ▸ **Application** ▸ **AxisInterface** ▸ **Plugins** ▸ **KinInterface** zu finden.

Programm zum Kinematik-Interface

Das Programm **TE_KinInterfaceType01Prog** besitzt folgende Aufgaben:

- **Initialisierung des Kinematik-Interfaces**:

  Bei Erreichen des Parametriermodus (Phase 2) wird für alle konfigurierten Kinematiken mithilfe des Funktionsbausteins **TE_KinematicsInitAllKinematicsType01** das Kinematik-Interface initialisiert. Bei Erreichen des Betriebsmodus (Phase 4) wird die Initialisierung erneut ausgeführt, um Änderungen in der Parametrierung zu erkennen

- **Zyklischer Aufruf des Kinematik-Interfaces**:

  Nach erfolgreicher Initialisierung (signalisiert durch "TE_KinInterfaceType01Prog.InitDone") wird das Kinematik-Interface zyklisch aufgerufen. Dazu wird der Funktionsbaustein "TE_KinematicsInterface" aufgerufen, welcher vom Funktionsbaustein "MB_KinInterfaceType01" abgeleitet ist und somit Anwendererweiterungen ermöglicht.

- **Anwendererweiterungen des Kinematik-Interfaces**:

  Der Funktionsbaustein "TE_KinematicsInterface" gibt dem Anwender die Möglichkeit den Strukturen des Kinematik-Interfaces weitere Elemente hinzuzufügen, um somit seine Anwendung optimal zu erweitern. Nähere Hinweise siehe Kap. 7.2.7 "Kinematik-Interface Anwender-Erweiterung" auf Seite 205

Kinematik-Interface Visualisierungen

Überblick

Zum Kinematik-Interface werden Visualisierungsmasken mitgeliefert, um ein vorgefertigtes und einfaches Interface zum Einstellen und Ansteuern der Kinematiken zur Verfügung zu stellen.

Folgende Visualisierungen sind enthalten:

Visualisierung	Beschreibung
SystemOverview	
• KinSystem_Overview_All	Gesamtüberblick über alle definierten Kinematiken, einschließlich einfacher Diagnosen, Status und Koordinaten
KinematicsModes	

Visualisierung	Beschreibung
• Kinematics_Overview	Zeigt aktuelle Betriebsart und aktuellen Status des Kinematik-Interfaces, zusammen mit Navigationsschaltflächen zu allen weiteren Visualisierungen.
• KinHoming_mode	Kommandierung und Überwachen der Referenzierbetriebsart
• KinPosition_mode	Kommandierung und Überwachen der Positionierbetriebsarten
• KinRCL_mode	Kommandierung und Überwachen der Betriebsart RCL
Utilities	
• ConveyorBelt	Konfiguration eines oder mehrerer Bänder
• RCLPoint	Setzen und lesen eines RCL-Punktes, Lesen einer RCL-Punkteliste und -Programmliste
• RCLProg	Laden, speichern und kompilieren eines RCL-Programms
• Setup	Kommandierung und Überwachen des Tipp-Betriebes

Abb.7-53: Kinematik-Interface Visualisierungen

Die folgenden globalen Variablen werden zum Steuern und für den Zugriff auf Systeminformationen innerhalb der Visualisierungen benutzt:

- arKinCtrl_gb[ ]
- arKinStatus_gb[ ]
- KinData[ ]
- VisuKinematicsNo

Systemübersicht Visualisierung

Die Visualisierung **KinSystem_Overview_All** erlaubt es dem Anwender jede, im Projekt konfigurierte Kinematik schnell anzukoppeln. Die angezeigte Standard-Kinematik besitzt keine Referenz. Wie diese Referenz gesetzt werden muss, wird im Folgenden beschrieben. Zusätzliche Kinematiken können im Offline-Betrieb zur Anzeige hinzugefügt werden.

Die Visualisierung liefert einen Gesamt-Systemstatus und ermöglicht das Löschen von Fehlern. Einzelne Kinematik-Bedienelemente liefern den Kinematiknamen und die Kinematiknummer, einige Diagnosen sowie die aktuelle Position in Weltkoordinaten zusammen mit den Betriebsarten-Statusanzeigen.

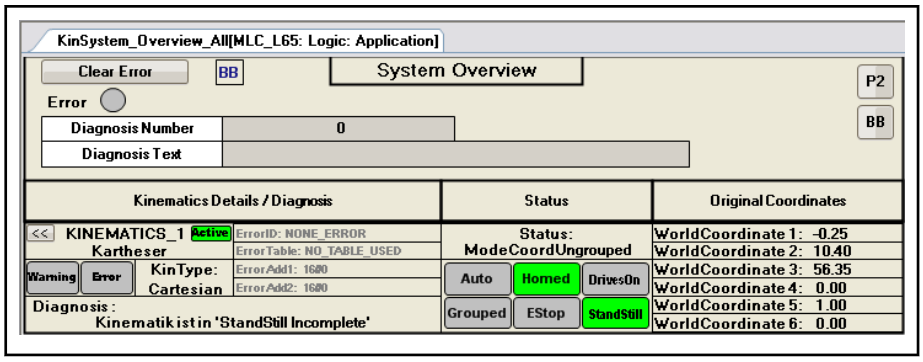


Abb.7-54: KinSystem_Overview_All Anzeige

Clear Error Mit dieser Schaltfäche wird ein "Clear Error" über das IMC-Interface ausgelöst.

Es werden keine Kinematik- oder Achsfehler gelöscht. Diese müssen separat in den jeweiligen Kinematik- bzw. Achsübersichten gelöscht werden.

Kinematik-Interface

P2 Mit dieser Schaltfäche kann in den Parametriermodus P2 umgeschaltet werden.

BB Mit dieser Schaltfäche kann in den Betriebsmodus BB umgeschaltet werden.

Hinzufügen einer Kinematik zur Systemübersicht

Das Hinzufügen einer Kinematik zur Systemübersichtsanzeige erfolgt wie das Hinzufügen einer neuen Visualisierung, durch Anwählen des entsprechenden Elements und der Festlegung der Kinematikreferenz. Nachfolgend aufgeführt sind zwei Vorgehensweisen, wie eine neue Kinematik der Systemübersicht hinzugefügt werden kann:

Möglichkeit 1: Erstellen einer Kinematik-Visualisierung mittels Frame

1. Öffnen Sie das entsprechende IndraWorks-Projekt und wählen Sie den Reiter **Logic** ▸ **Application** ▸ **AxisInterface** ▸ **Plugins** ▸ **KinInterface** ▸ **Visualizations** ▸ **SystemOverview** an.
2. Mit IndraLogic im Offline-Betrieb Doppelklick auf die Visualisierung "KinSystem_Overview_All".
3. Wählen Sie im Fenster **Werkzeuge** die Schaltfläche **Frame** an und erzeugen Sie einen Grundriss unterhalb der letzten Kinematik in entsprechender Größe.

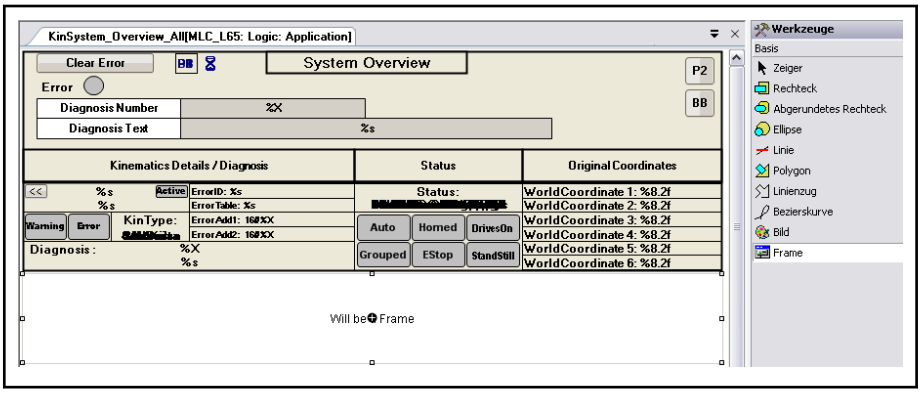


Abb.7-55: Hinzufügen einer Kinematik zur Systemübersicht

4. Mit rechter Maustaste auf den Frame klicken und "Frameauswahl" aktivieren. Wählen Sie das **KinSystem_Overview**-Element aus dem Visualisierungsauswahlfenster aus.

   Eine komplett neue Systemübersicht für die Kinematik erscheint.

5. Öffnen Sie das Eigenschaften-Fenster für den neu eingefügten Frame und tragen Sie unter **Referenzierte Visualisierungen** ▸ **Referenzierte Visualisierungen** ▸ **KinSystem_Overview** ▸ **m_Input_KinRef** den Namen der Kinematik ein. In gleicher Art und Weise kann so auch die Referenz der ersten Kinematik angepasst werden.

   Die Systemübersicht steht nun für die angelegte Kinematik zur Verfügung.

Möglichkeit 2: Kopieren einer bestehenden Kinematikvisualisierung

1. Öffnen Sie das entsprechende IndraWorks-Projekt und wählen Sie den Reiter **Logic** ▸ **Application** ▸ **AxisInterface** ▸ **Plugins** ▸ **KinInterface** ▸ **Visualizations** ▸ **SystemOverview** an.
2. Mit IndraLogic im Offline-Betrieb Doppelklick auf die Visualisierung "KinSystem_Overview_All".

3. Kopieren Sie die bereits vorhandene Kinematik-Visualisierung und fügen Sie sie beliebig oft ein.
4. Öffnen Sie das Eigenschaften-Fenster für die neu eingefügten Kinematikvisualisierungen und tragen Sie unter **Referenzierte Visualisierungen ▸ Referenzierte Visualisierungen ▸ KinSystem_Overview ▸ m_Input_KinRef** den Namen der Kinematik ein. In gleicher Art und Weise kann so auch die Referenz der ersten Kinematik angepasst werden.

   Die Systemübersicht steht nun für die angegebene Kinematik zur Verfügung.

☞ Eine der oben genannten Möglichkeit muss für alle, zusätzlich zum Projekt hinzugefügte Kinematiken wiederholt werden.

Systemübersicht Navigation

Aus der Kinematik-Systemübersicht ist es über << möglich, zur Visualisierung der Einzelkinematik "Kinematics_Overview" zu gelangen. Diese Schaltfläche befindet sich links in der Spalte "Kinematics Details / Diagnosis". Aus der Einzelkinematikanzeige sind alle weiteren Visualisierungen erreichbar.

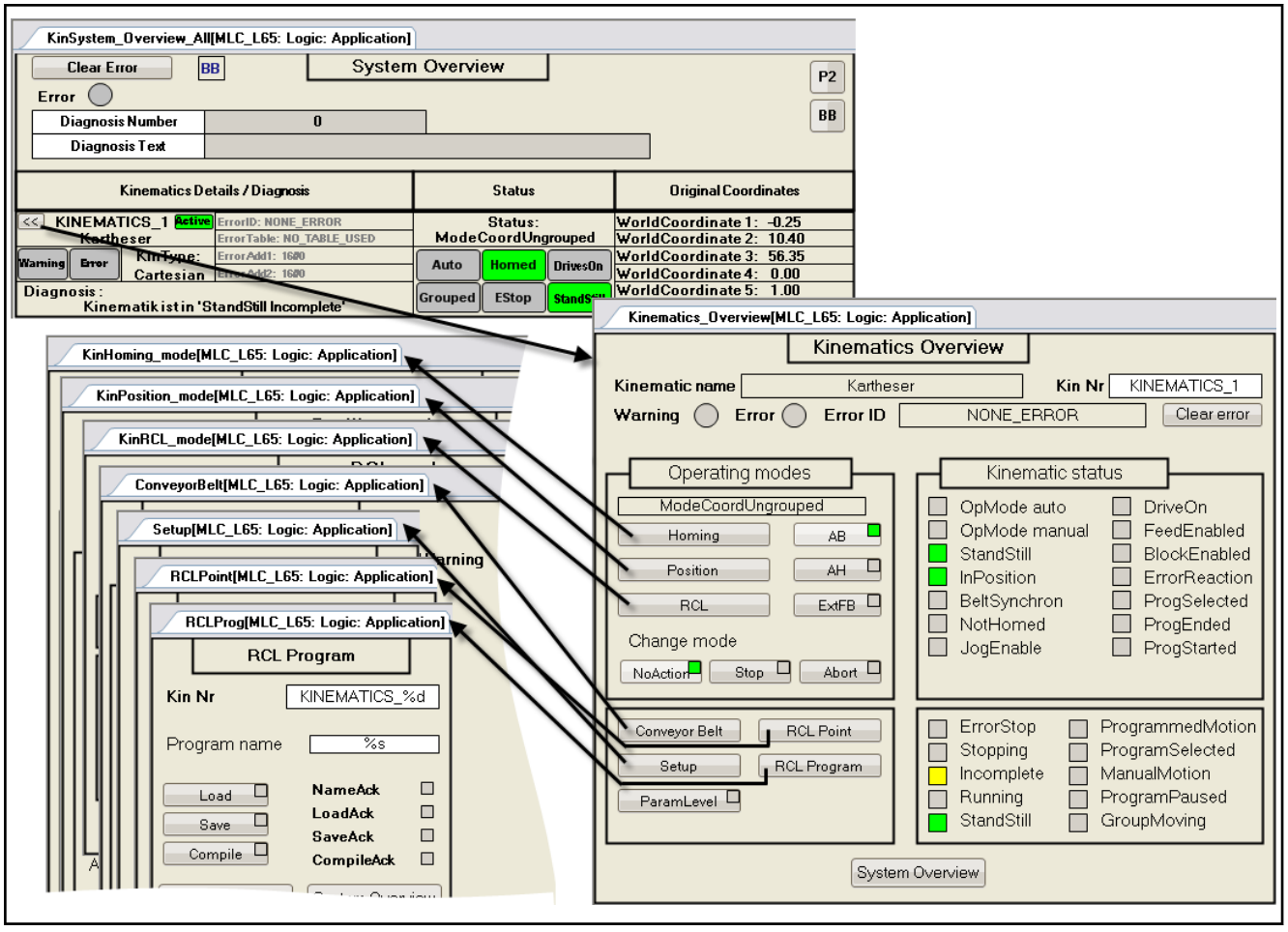


Abb.7-56: Systemübersicht Navigation

☞ Jede Visualisierung bietet die Möglichkeit durch Eingabe einer Kinematiknummer im Feld **Kin Nr** zwischen verschiedenen Kinematiken hin und her zu wechseln.

Kinematikübersicht Visualisierung

Die Visualisierung **Kinematics_Overview** stellt übersichtlich die wichtigsten Funktionen des Kinematik-Interfaces und Diagnosen der KinData-Schnittstelle dar.

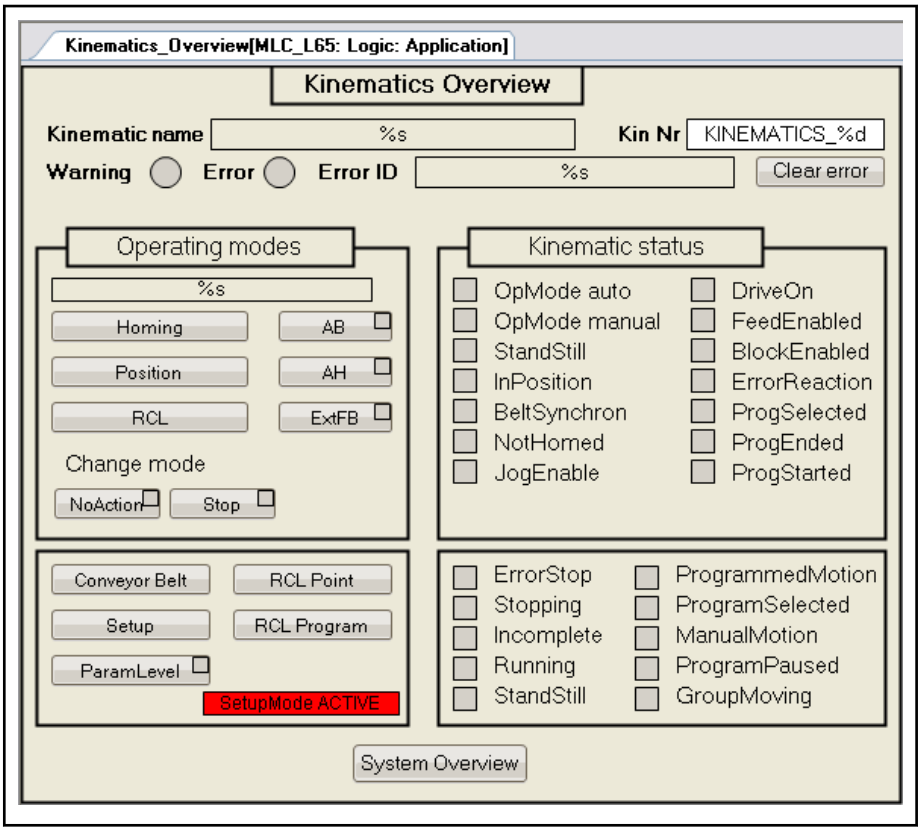


Abb.7-57: Kinematikübersicht Visualisierung

Über die Schaltflächen **Homing**, **Position** und **RCL** kann die Visualisierung für die entsprechende Betriebsart umgeschalten werden, in welcher spezielle Werte für die jeweilige Betriebsart eingegeben werden können. Die Schaltflächen **AB**, **AH** und **ExtFB** erlauben eine direkte Umschaltung in die entsprechende Betriebsart. Mithilfe der Schaltflächen **NoAction** und **Stop** kann die Reaktion nach einem Betriebsartenwechsel definiert werden.

Über die Schaltflächen **Conveyor Belt**, **Setup**, **RCL Point** und **RCL Program** öffnen sich die entsprechenden, unter "Utilities" abgelegten Visualisierungen zum Konfigurieren eines Bandes, Vertippen der Kinematik, Bearbeiten eines RCL-Punktes sowie zur Handhabung von RCL-Programmen. Ist der Tipp-Betrieb aktiviert, so wird dies über die Meldung "SetupMode ACTIVE" signalisiert.

Der rechte Teil der Visualisierung stellt die Diagnosedaten der KinData-Schnittstelle dar und zeigt den aktuellen PLCOpen-Zustand der Kinematik an.

☞ In Abhängigkeit davon, welche Robot-Control-Version genutzt wird, werden einige Anzeigeelemente ausgeblendet. Somit sind nur Kinematik-Interface-Funktionalitäten, welche in der entsprechenden Robot-Control-Version verfügbar sind, anwählbar.

Referenzierübersicht (Homing) Visualisierung

Die Visualisierung **KinHoming_mode** dient zum Referenzierung von Kinematikachsen.

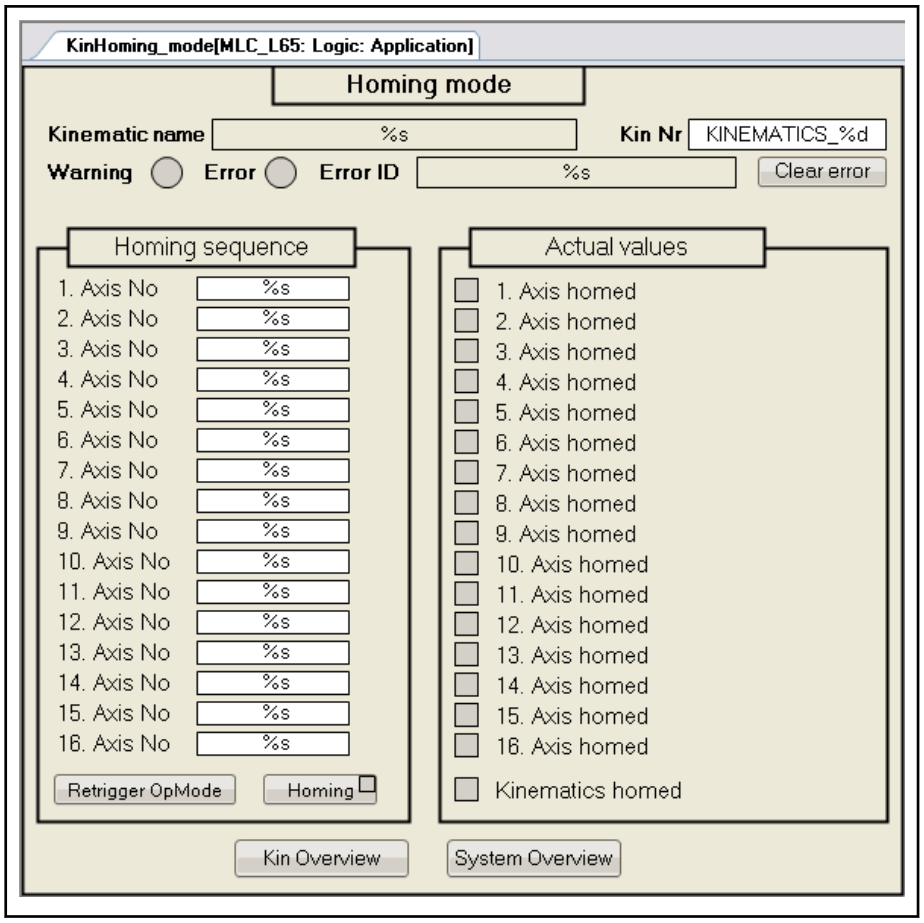


Abb.7-58: Referenzierübersicht Visualisierung

Die Reihenfolge der zu referenzierenden Achsen kann beliebig gewählt werden. Standardmäßig sind alle, der Kinematik zugeordneten Achsen entsprechend der Konfigurationsreihenfolge eingetragen. Als Abbruchbedingung für die Homing-Sequenz steht die Achsnummer 0.

Im rechten Teil der Visualisierung ist der Referenzstatus jeder angegebenen Achse sowie der Kinematik dargestellt. Ist die Achse bzw. die Kinematik in Referenz, so färbt sich das Anzeigeelement grün.

☞ Das Anzeigeelement für "Kinematics homed" wird erst grün, wenn die Homing-Sequenz fehlerfrei abgearbeitet wurde, auch wenn alle Achsen bereits referenziert sind.

Positionierübersicht (Position) Visualisierung

Die Visualisierung **KinPosition_mode** dient dem Positionieren der Kinematik.

Kinematik-Interface

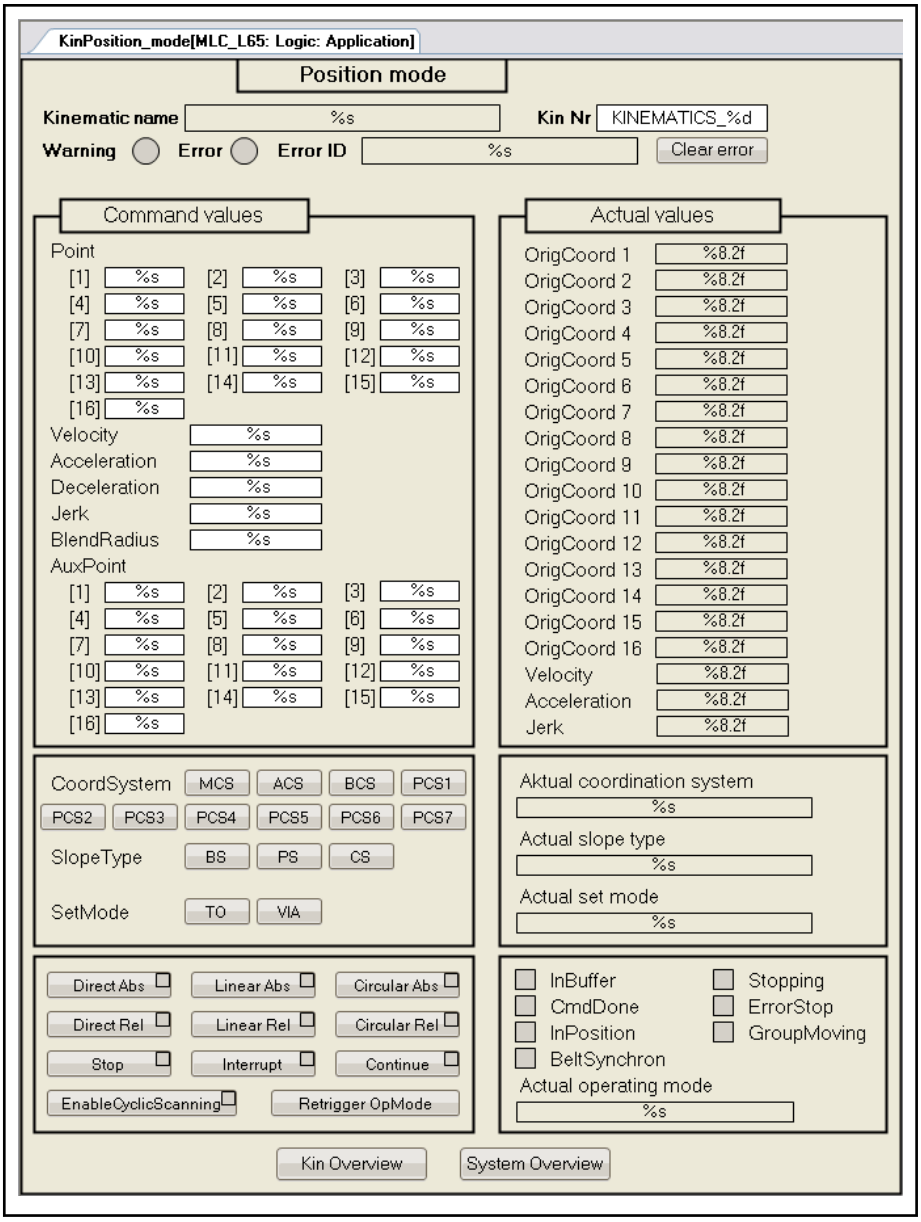


Abb.7-59: Positionierübersicht Visualisierung

Eingegeben bzw. ausgewählt werden kann die

- Position
- Geschwindigkeit
- Beschleunigung
- Verzögerung
- Ruck
- Überschleifradius
- Zwischenposition
- Koordinatensystem
- Slope-Art der Bewegung
- Art des Überschleifens

Über die Schaltflächen **DirectAbs**, **LinearAbs**, etc. kann die entsprechende Positionierbetriebsart gewählt werden. Nähere Details sind der Betriebsartenbeschreibung zu entnehmen.

Im rechten Teil der Visualisierung wird die aktuelle Position in Weltkoordinaten, die parametrierten Randbedingungen, wichtige Statuselemente des Kinematik-Interfaces und der KinData-Schnittstelle sowie die aktuelle Betriebsart angezeigt.

☞ In Abhängigkeit davon, welche Robot-Control-Version genutzt wird, werden einige Anzeigeelemente ausgeblendet. Somit sind nur Kinematik-Interface-Funktionalitäten, welche in der entsprechenden Robot-Control-Version verfügbar sind, anwählbar.

Robot-ControlLanguage (RCL) Visualisierung

Die Visualisierung **KinRCL_mode** dient dem Auswählen und Starten eines RCL-Programms.

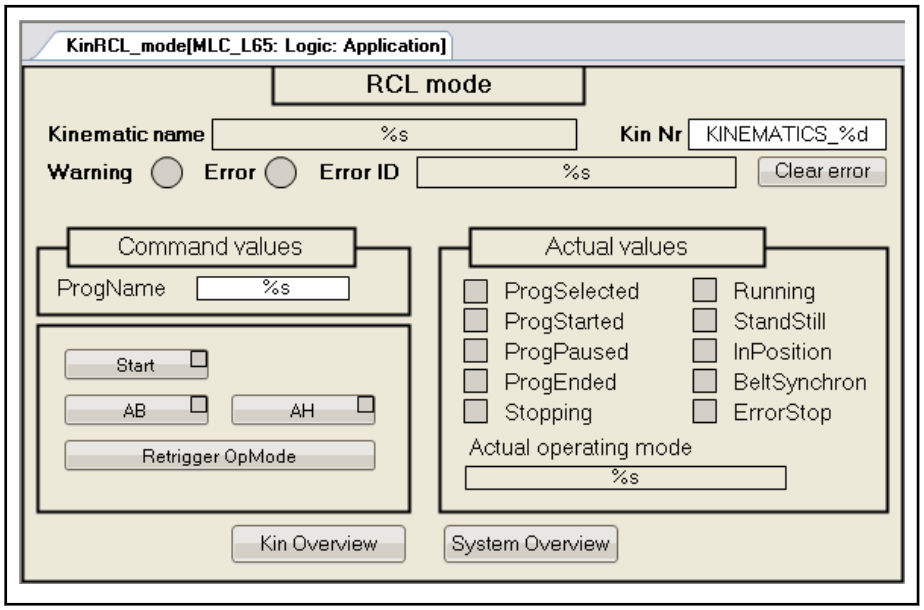


Abb.7-60: Robot-ControlLanguage Visualisierung

Durch Angabe des RCL-Programm-Namens im Feld "ProgName" kann das Programm über die Schaltfläche **Start** gestartet werden. Gestoppt wird das RCL-Programm durch einen Betriebsartenwechsel, welcher hier mit den Schaltflächen **AB** oder **AH** angestoßen werden kann.

Im rechten Teil der Visualisierung werden wichtige Statuselemente des Kinematik-Interfaces und der KinData-Schnittstelle sowie die aktuelle Betriebsart angezeigt.

☞ Der RCL-Programm-Name im Feld "ProgName" ist **ohne** Dateinamenserweiterung (*.ird) anzugeben.

☞ Nur für Robot-Control V1 zulässig.

Bandkonfiguration (ConveyorBelt) Visualisierung

Die Visualisierung **ConveyorBelt** dient der Konfiguration aller, der Kinematik zugehörigen Bänder.

Kinematik-Interface

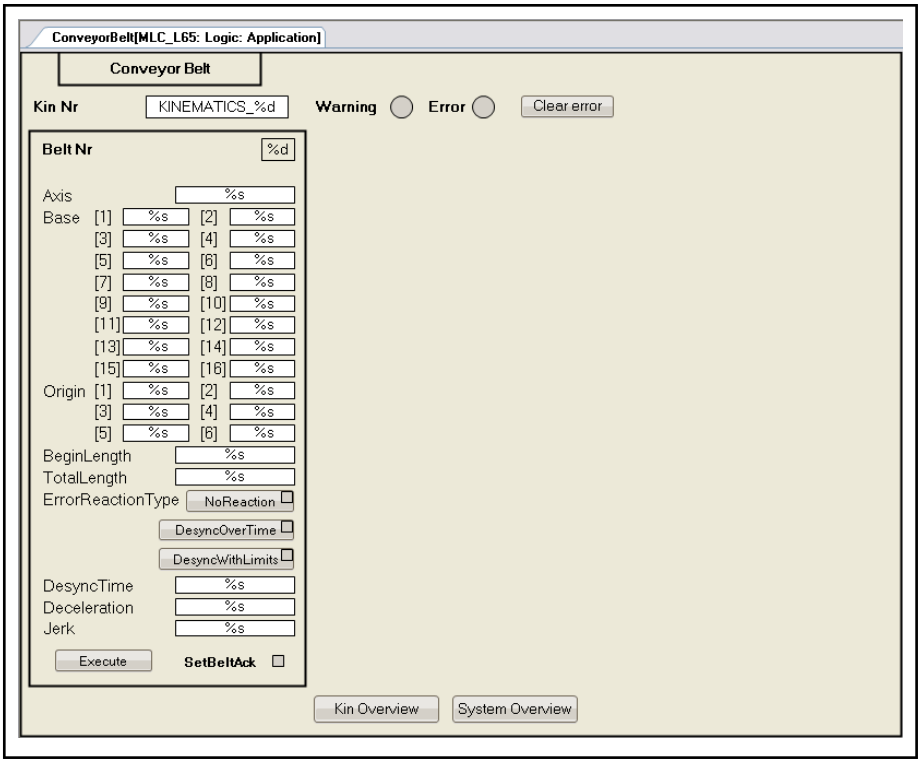


Abb.7-61: Bandkonfiguration Visualisierung

Durch Angabe der Bandkonfiguration und der Bandfehlerreaktion kann das entsprechende Band parametriert werden. Entsprechend der Anzahl der Bänder ist die Bandkonfigurationsübersicht zu vervielfachen. Dies erfolgt in gleicher Art und Weise wie das Hinzufügen einer Kinematik zur Systemübersicht (siehe Kap. "Systemübersicht Visualisierung" auf Seite 195)

Als referenzierte Visualisierung ist hierbei eine fortlaufende Bandnummer anzugeben. Mithilfe dieser Bandnummer wird intern auf die Kinematik-Interface-Struktur `arKinCtrl_gb[].Admin.ConveyorBelt` zugegriffen.

☞ Die Konfiguration des Bandes erfolgt intern mit dem Funktionsbaustein "ML_SetBeltParameter" (Robot-Control V1) bzw. "ML_SetBeltConfiguration" (Robot-Control V2). Nähere Informationen hierzu ist der entsprechenden Dokumentation zu entnehmen.

☞ In Abhängigkeit davon, welche Robot-Control-Version genutzt wird, werden einige Anzeigeelemente ausgeblendet. Somit sind nur Kinematik-Interface-Funktionalitäten, welche in der entsprechenden Robot-Control-Version verfügbar sind, anwählbar.

Tipp-/Einrichtbetrieb (Setup) Visualisierung

Die Visualisierung **Setup** bietet die Möglichkeit, die Kinematik zu vertippen.

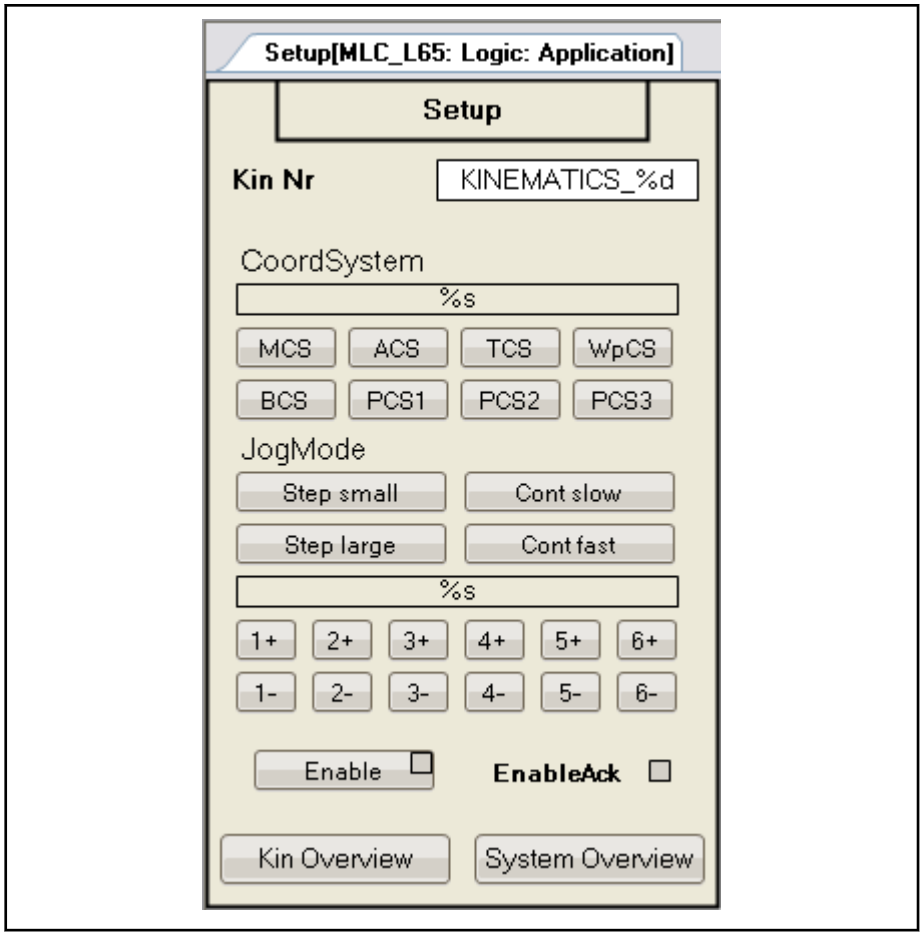


Abb.7-62: Tipp-/Einrichtbetrieb Visualisierung

Über die Schaltflächen unter "CoordSystem" wird das Koordinatensystem definiert, in welchem vertippt werden soll. In gleicher Art und Weise kann die Schrittweite bzw. die Verfahrgeschwindigkeit unter "JogMode" festgelegt werden. Mit den Schaltflächen **1+** bis **6-** ist es möglich, die ersten 6 Achsen der Kinematik in positive und negative Richtung entsprechend der Einstellungen zu verfahren.

☞ Bandachsen gehören nicht zur Achsgruppe und können daher nicht vertippt werden.

☞ In Abhängigkeit davon, welche Robot-Control-Version genutzt wird, werden einige Anzeigeelemente ausgeblendet. Somit sind nur Kinematik-Interface-Funktionalitäten, welche in der entsprechenden Robot-Control-Version verfügbar sind, anwählbar.

Robot Control Language Punkt (RCLPoint) Visualisierung

Über die Visualisierung **RCLPoint** kann ein RCL-Punkt gelesen oder geschrieben bzw. eine RCL-Punktelist oder -Programmliste gelesen werden.

Kinematik-Interface

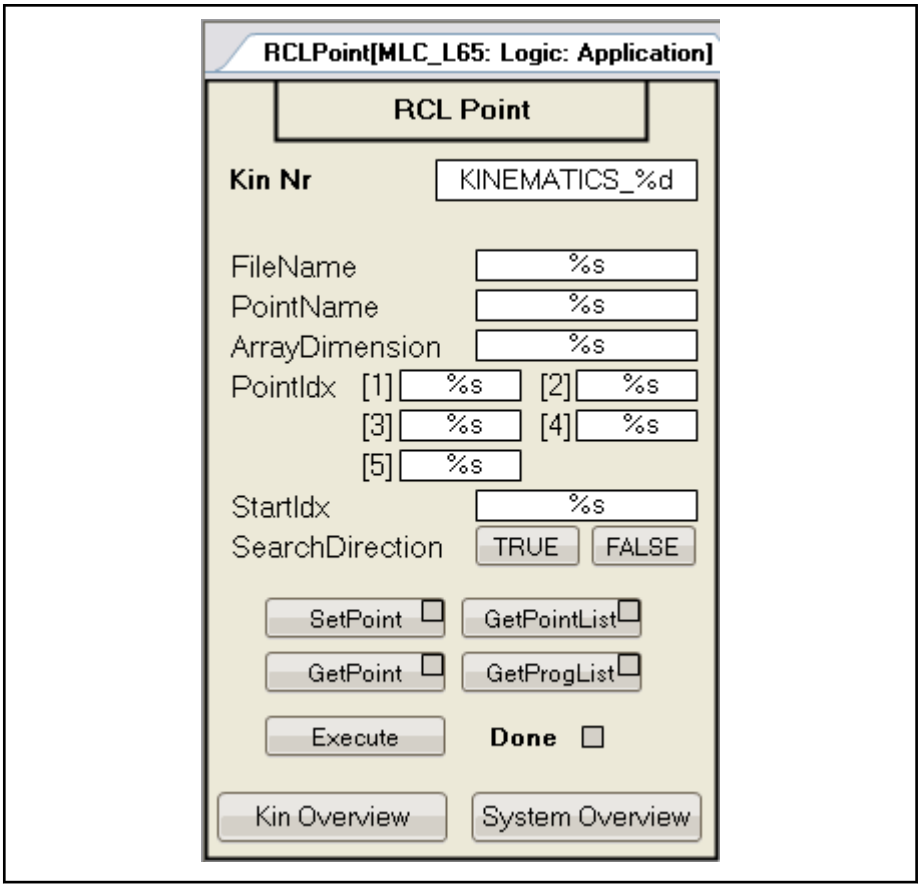


Abb.7-63: Robot Control Language Punkt Visualisierung

Durch die in dieser Visualisierung anzugebenden Werte ist es möglich, einen Punkt (**GetPoint**), eine Punkteliste (**GetPointListe**) oder Programmliste (**GetProgList**) auszulesen. Für das Beschreiben eines Punkte (**SetPoint**) ist weiterhin im Anwenderprogramm das Kinematik-Interface-Strukturelement `arKinCtrl_gb[].Admin.RCLPoint.Point` zu beschreiben.

☞ Die Angabe des Dateinamens im Feld "FileName" muss **mit** Dateinamenserweiterung (*.pkt) erfolgen.

☞ Nur für Robot-Control V1 zulässig.

Robot Control Language Programm (RCLProg) Visualisierung

*Über die Visualisierung **RCLProg** kann eine RCL-Datei kopiert werden:*

- Von der Compact Flash Karte in den internen Speicher der Steuerung
- Vom internen Speicher der Steuerung in die Compact Flash Karte

oder

*Über die Visualisierung **RCLProg** kann eine RCL-Datei kompiliert werden:*

- Auf der Steuerung

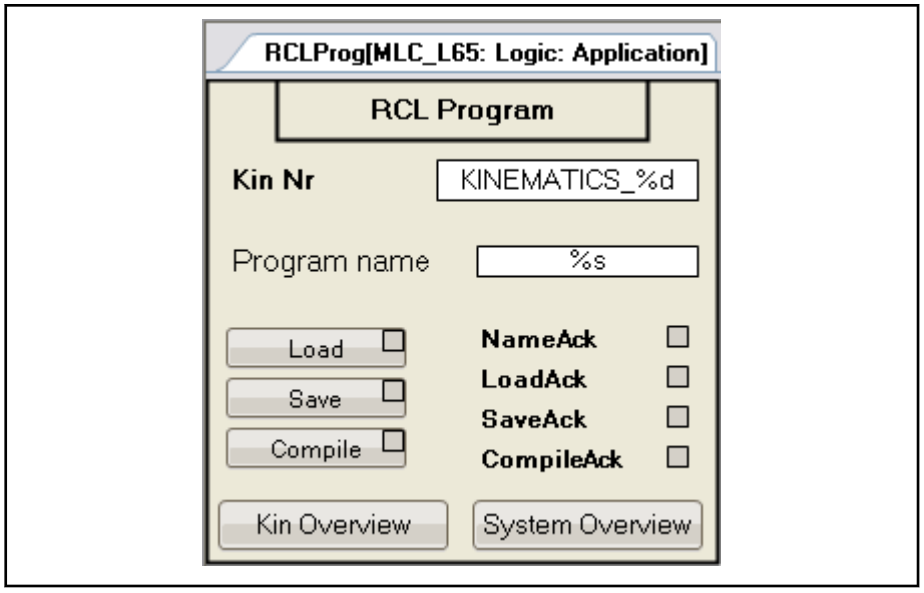


Abb.7-64: Robot Control Language Programm Visualisierung

☞ Die Angabe des Dateinamens im Feld "Program name" muss **mit** Dateinamenserweiterung (z. B. *.qll) erfolgen.

☞ Nur für Robot-Control V1 zulässig.

7.2.7 Kinematik-Interface Anwender-Erweiterung

Überblick

Die `arKinCtrl_gb[]` und `arKinStatus_gb[]` Strukturen können durch den Anwender erweitert werden, um das Kinematik-Interface an spezielle Applikationen anzupassen.

Dieser Abschnitt zeigt, wie das Kinematik-Interface durch Hinzufügen einer neuen Funktionalität erweitert wird. Es wird davon ausgegangen, dass die im Kinematik-Interface bereits vorbereiteten Erweiterungsstrukturen verwendet werden. Es wird im Folgenden nur beschrieben, welche Änderungen in den POUs und Strukturen notwendig sind.

Hinweise zur Implementation von Anwendererweiterungen

Die Anwendererweiterungen wurden mit Hilfe der objektorientierten Erweiterungen von IndraLogic 2G implementiert.

Dabei sind einige Besonderheiten zu beachten:

- Der Funktionsbaustein **TE_KinematicsInterface** ist vom Basis-Funktionsbaustein "MB_KinInterfaceType01" abgeleitet. Über das Schlüsselwort "SUPER" kann der Basis-Funktionsbaustein bzw. dessen Methoden/ Aktionen aufgerufen werden. Zum Beispiel wird an diversen Stellen über `SUPER^.mSetError(...);` die Methode mSetError von MB_KinInterfaceType01 aufgerufen, um Fehler in das Diagnosesystem einzutragen
- Innerhalb des Funktionsbaustein **TE_KinematicsInterface** kann auf die Daten von arKinCtrl_gb über die Eingänge AdminCtrlExt, CoordModeCtrlExt usw. bzw. auf arAxisStatus_gb über AdminStatusExt, DiagStatusExt usw. zugegriffen werden

Kinematik-Interface

☞ Die Eingänge AdminCtrl (ohne *Ext) usw. gehören zum Basis-Funktionsbaustein und sollten nicht genutzt werden, da hier keine Anwender-Erweiterungen zur Verfügung stehen.

- Die Eingänge des Funktionsbaustein TE_KinematicsInterface sind als "REFERENCE TO" definiert. In der Methode "mInitExtension" werden die Referenzen einmalig initialisiert und müssen dann beim zyklischen Aufruf des Funktionsbausteins nicht mehr übergeben werden.

Erweitern der Kinematik-Interface-Struktur

Wenn die Strukturen TE_KINEMATICS_CONTROL_TYPE01 und TE_KINEMATICS_STATUS_TYPE01 mit zusätzlichen Unterstrukturen erweitert werden sollen, sind folgende zusätzliche Schritte zu der weiter unten beschriebenen Vorgehensweise notwendig:

1. Neue Elemente in TE_KINEMATICS_CONTROL_TYPE01 bzw. TE_KINEMATICS_STATUS_TYPE01 eintragen
2. Am FB TE_KinematicsInterface die zusätzlichen Eingänge als REFERENCE TO hinzufügen
3. In der Methode "mInitExtension" des FB TE_KinematicsInterface die Referenzen initialisieren

☞ Wenn die Unterstrukturen TE_KINEMATICS_ADMINISTRATION, TE_KINEMATICS_COORDINATED, usw. erweitert werden sollen, ist nur Schritt 1 des zuvor beschriebenen Arbeitsablaufes für die jeweilige Struktur notwendig. In dieser Struktur ist die Erweiterung nach `(* add your extensions below here | Erweiterungen hier einfuegen*)` einzufügen. Danach steht der neu angelegte Ein- oder Ausgang über das entsprechende *Ext-Strukturelement zur Verfügung.

Erweitern des Funktionsbausteins TE_KinematicsInterface

Der letzte Schritt im Ablauf der Anwender-Erweiterung ist, den Funktionsbaustein TE_KinematicsInterface so zu erweitern, dass der Programmcode der Anwender-Erweiterung eingefügt werden kann.

1. Im Projektbaum, mit rechtem Mausklick **TE_KinematicsInterface** und **Hinzufügen ▸ Aktion** wählen.
2. Geben Sie der neuen Aktion einen zu Ihrer Erweiterung passenden Namen
3. Fügen Sie ihren Programmcode in die neue Aktion ein.

   Nutzen Sie für den Zugriff auf die Strukturen `arKinCtrl_gb[]` und `arKinStatus_gb[]` die Eingänge mit dem Suffix *Ext (z. B. `AdminCtrlExt`, `DiagStatusExt`, usw).
4. Rufen Sie im Funktionsbaustein TE_KinematicsInterface die neue Aktion auf.
5. Übersetzen Sie das Projekt neu und überprüfen Sie es auf Programmierfehler.
6. Laden Sie das Projekt in die Steuerung.

Die neuen Eingangs- und Ausgangs-Elemente sind nun ein Teil der Kinematik-Interface-Struktur und können über die UserVarGlobal-Variablen betrachtet werden.

7.3 Anwendungsbeispiele

7.3.1 Sequenzielle Programmierung mit dem Kinematik-Interface

Unter Beachtung gewisser Randbedingungen ist es möglich, mit dem Kinematik-Interface einen sequenziellen Bewegungsablauf zu realisieren. Anhand eines Beispiels wird im Folgenden erläutert, wie dies möglich ist.

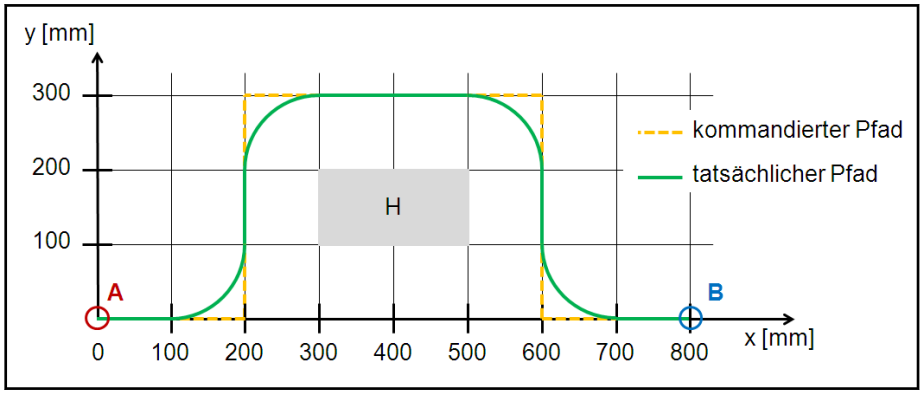


Abb.7-65: Sequenzielle Programmierung mit dem Kinematik-Interface - Bewegungspfad

Eine Kinematik (in diesem Fall ein Kartheser) soll von einem Punkt A zu einem Punkt B um ein Hindernis H fahren. Hierzu wird der in obiger Abbildung dargestellte Pfad vorgegeben.

Die hieraus resultierenden Positionen sind:

- A(0,0)
- P1(200,0)
- P2(200,300)
- P3(600,300)
- P4(600,0)
- B(800,0)

Die Positionen sollen mittels linearer, absoluter Verfahrbewegungen überschleifend angefahren werden. Um dies mithilfe des Kinematik-Interfaces zu realisieren sind folgende Programmzeilen notwendig. Die Grenzwerte für Geschwindigkeit, Beschleunigung, Verzögerung und Ruck sind hierbei frei gewählt.

Deklarationsteil

```
VAR
    uiState:       UINT := STATE_IDLE;
    bStartMove:    BOOL := FALSE;
    rVelocity:     REAL := 0.0;
    rAcceleration: REAL := 0.0;
    rDeceleration: REAL := 0.0;
    rJerk:         REAL := 0.0;
    arPoint1:      ARRAY[1..16]OF REAL := [200.0, 0.0, 14(0.0)];
    arPoint2:      ARRAY[1..16]OF REAL := [200.0, 300.0, 14(0.0)];
    arPoint3:      ARRAY[1..16]OF REAL := [600.0, 300.0, 14(0.0)];
    arPoint4:      ARRAY[1..16]OF REAL := [600.0, 0.0, 14(0.0)];
    arPointB:      ARRAY[1..16]OF REAL := [800.0, 0.0, 14(0.0)];
END_VAR

VAR CONSTANT
    STATE_IDLE:  UINT := 1;
    STATE_MOVE1: UINT := 2;
    STATE_MOVE2: UINT := 3;
    STATE_MOVE3: UINT := 4;
    STATE_MOVE4: UINT := 5;
    STATE_MOVE5: UINT := 6;
END_VAR
```

Kinematik-Interface

Programmteil

```
CASE uiState OF
  STATE_IDLE:
    IF bStartMove THEN
      bStartMove := FALSE;
      uiState    := STATE_MOVE1;
    END_IF

  STATE_MOVE1:
    (* Verhalten nach einem Betriebsartenwechsel festlegen; muss bei
       überschleifenden Bewegungen immer auf "NoAction" stehen *)
    arKinCtrl_gb[1].Admin.ChangeOpMode := NoAction;

    (* arPoint1 = Position P1 festlegen *)
    arKinCtrl_gb[1].CoordMode.Point := arPoint1;

    (* Grenzwerte festlegen *)
    arKinCtrl_gb[1].CoordMode.Velocity     := 100;
    arKinCtrl_gb[1].CoordMode.Acceleration := 1000;
    arKinCtrl_gb[1].CoordMode.Deceleration := 1000;
    arKinCtrl_gb[1].CoordMode.Jerk         := 10000;

    (* Koordinatensystem festlegen *)
    arKinCtrl_gb[1].CoordMode.CoordSystem := CS_MACHINE_COORDINATES;

    (* SetMode festlegen *)
    arKinCtrl_gb[1].CoordMode.SetMode := SM_VIA;

    (* SlopeType festlegen *)
    arKinCtrl_gb[1].CoordMode.SlopeType := ST_CONTINUOUS_SLOPE;

    (* Überschleifradius festlegen *)
    arKinCtrl_gb[1].CoordMode.BlendingRadius := 100;

    (* Lineares Verfahrkommando absetzen;
       Satz wird in den Satzpuffer gestellt *)
    arKinCtrl_gb[1].Admin._OpMode.en := ModeCoordPosLinAbs;

    (* vor nächstem Schritt: Ausgang "InBuffer" abfragen *)
    IF arKinStatus_gb[1].Admin.InBuffer THEN
      uiState := STATE_MOVE2;
    END_IF

  STATE_MOVE2:
    (* arPoint2 = Position P2 festlegen *)
    arKinCtrl_gb[1].CoordMode.Point := arPoint2;

    (* es müssen nur die Eingangsstrukturen beschrieben werden, welche
       sich geändert haben *)
    arKinCtrl_gb[1].CoordMode.Velocity := 20;

    (* ein weiteres Absetzen eines linearen Verfahrkommandos ist
       nicht notwendig, da dies durch die Änderung des
       Positionswertes automatisch geschieht *)

    (* vor nächstem Schritt: Ausgang "InBuffer" abfragen *)
    IF arKinStatus_gb[1].Admin.InBuffer THEN
      uiState := STATE_MOVE3;
    END_IF

  STATE_MOVE3:
    (* arPoint3 = Position P3 festlegen *)
    arKinCtrl_gb[1].CoordMode.Point := arPoint3;

    (* vor nächstem Schritt: Ausgang "InBuffer" abfragen *)
    IF arKinStatus_gb[1].Admin.InBuffer THEN
      uiState := STATE_MOVE4;
    END_IF

  STATE_MOVE4:
    (* arPoint4 = Position P4 festlegen *)
    arKinCtrl_gb[1].CoordMode.Point := arPoint4;

    (* vor nächstem Schritt: Ausgang "InBuffer" abfragen *)
    IF arKinStatus_gb[1].Admin.InBuffer THEN
      uiState := STATE_MOVE5;
    END_IF

  STATE_MOVE5:
    (* arPointB = Position B festlegen *)
    arKinCtrl_gb[1].CoordMode.Point := arPointB;
```

```
    (* Ende *)
    uiState := STATE_IDLE;

END_CASE
```

Die Satzvorbereitung, d. h. die Abarbeitung der Schrittkette erfolgt erheblich schneller als die Ausführung der Bewegung. Somit ist es möglich, dass die Schrittkette bereits im STATE_IDLE steht, obwohl die Kinematik noch in der Bewegung ist.



Die Status-Ausgänge des Kinematik-Interfaces beziehen sich immer auf den **zuletzt** abgesetzten Verfahrbefehl, in diesem Beispiel auf die Bewegung zum Punkt B. Es gibt somit keine Möglichkeit den Status der vorhergehenden Bewegungen auszuwerten.

7.3.2 Anmeldung einer benutzerdefinierten Transformation

Bei der Verwendung einer benutzerdefinierten Transformation, d. h. der Transformationsnummer 1100 ist es notwendig vor der Phasenschaltung nach BB die Transformation in der SPS anzumelden. Diese Anmeldung kann in einfacher Art und Weise über das Kinematik-Interface realisiert werden. Hierfür ist es lediglich notwendig, den Transformations-Funktionsbaustein an den Eingang "Admin.KinTransform" des Kinematik-Interfaces anzulegen. Folgendes Kommando startet die Anmeldung der Transformation ("fbTransformation" entspricht einer Instanz des Transformations-Funktionsbausteins):

```
arKinCtrl_gb[1].Admin.KinTransformPLC := fbTransformation;
```



Die Anmeldung ist nur im Parametriermodus **P2** möglich. Wird in einer anderen Phase eine Transformation an den Eingang des Kinematik-Interfaces angelegt, so startet die Anmeldung erst wenn Phase 2 erreicht ist.



Wird das Kinematik-Interface erstmalig mit einer benutzerdefinierten Transformation angemeldet, so gibt die Steuerung den Fehler "Transformation nicht angemeldet" aus. Dies ist dadurch begründet, dass das IMC-Interface versucht in den Betriebsmodus zu schalten. Nach erfolgreicher Anmeldung der Transformation lässt sich der Fehler quittieren.

War die Anmeldung der Transformation in der SPS erfolgreich, so wird dies über den Statusausgang `arKinStatus_gb[1].Admin.KinTransformPLCDone` signalisiert.

8 GAT^{compact}

8.1 Überblick GAT^{compact}

Allgemeines

Das Generic Application Template^{compact} ist ein Beispielprojekt, welches auf IEC 61131-3 basiert. Es wird mit dem IndraWorks Engineering Framework für IndraLogic XLC und IndraMotion MLC zur Verfügung gestellt, und lässt sich Wizard-unterstützt laden und in Betrieb nehmen. Es unterstützt auch die Besonderheiten von Motion-Applikationen, wie zum Beispiel die Handhabung von Achsen.

Grundidee

Die Anwendung des GAT^{compact} beruht auf dem Ansatz, dass eine einfache Maschine über die Betriebsarten "Auto" und "Manual" verfügen muss. Außerdem soll die Handhabung von Achsen beispielhaft demonstriert werden.

Voraussetzungen zur Anwendung

Der Anwender muss wissen, welche Aktivitäten die zu realisierende Maschine im Automatik-Betrieb und welche im Manual-Betrieb ausführen soll.

8.2 Erste Schritte – GAT^{compact}

8.2.1 Einfache Motion-Applikation erstellen

Vorbereitung

Mit der nachfolgenden Beschreibung können sie in wenigen Minuten über den GAT Wizard eine Motion Logic Applikation mit 2 Achsen erstellen.

Sie benötigen hierfür eine IndraLogic XLC oder IndraMotion MLC Steuerung, reale Antriebe sind nicht erforderlich. Grundlage ist das "GAT compact" Beispielprojekt.

Aufgabenstellung

- Im Automatik-Betrieb einer Maschine soll sich eine Achse "Master" mit konstanter Geschwindigkeit kontinuierlich bewegen. Die Geschwindigkeit soll über eine Variable veränderbar sein.
- Eine zweite Achse "Slave" soll sich dabei winkelsynchron zur ersten Achse bewegen.
- Nachfolgende Abbildung zeigt schematisch die Aufgabenstellung.

GATcompact

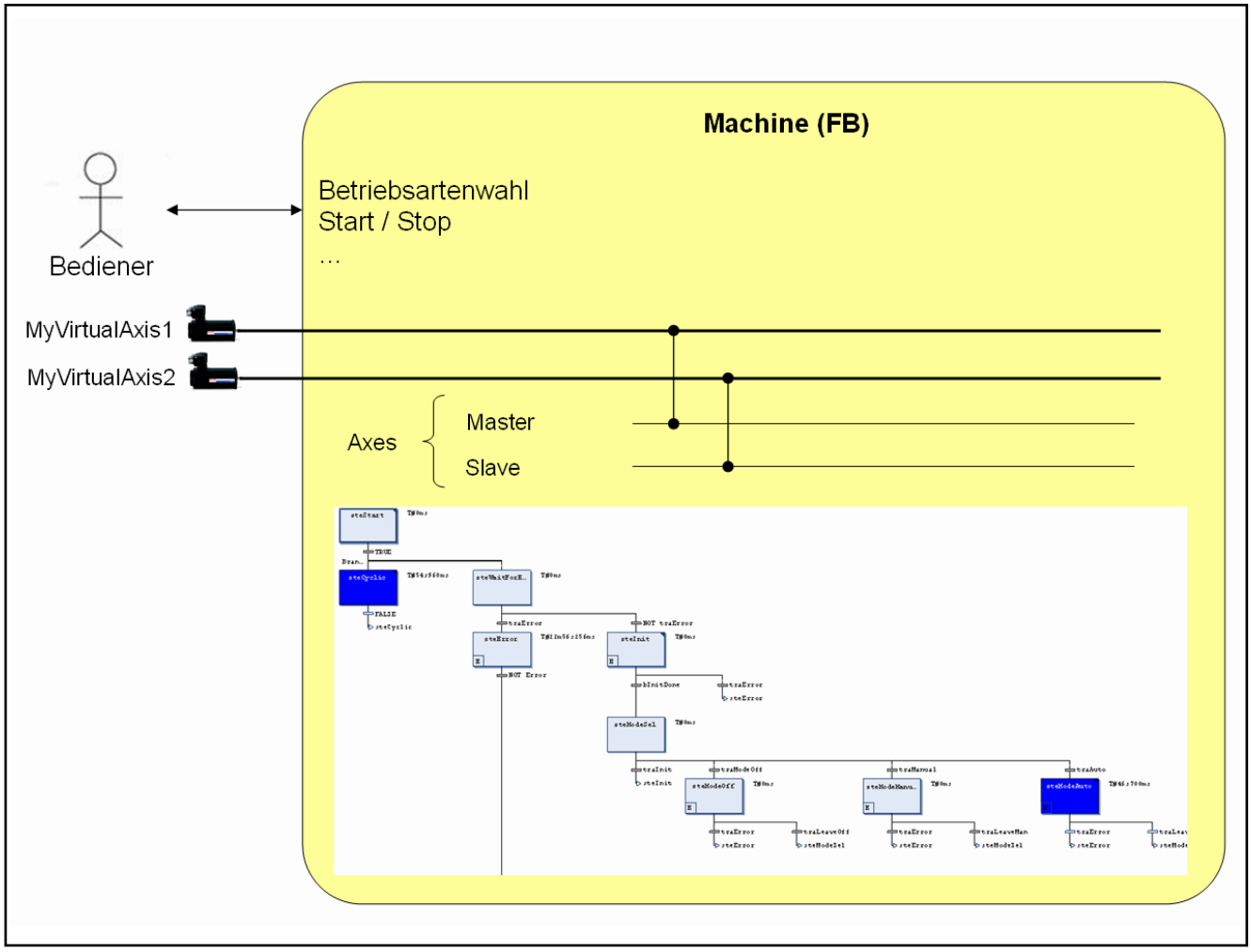


Abb.8-1: Schematisierte Aufgabenstellung

Arbeitsablauf

- IndraWorks öffnen und neues Projekt erstellen.
- IndraLogic XLC oder IndraMotion MLC Steuerung aus der Bibliothek in den Projektbaum hinzufügen.
- GAT zur XLC/MLC Steuerung hinzufügen. Hierfür das "GAT" Kontext-menü der Steuerung nutzen, oder den GAT-Button  in der IndraWorks Toolbar anklicken.
- Über nachfolgenden Dialog die Option "GAT compact" auswählen und mit "Fertigstellen" abschließen.

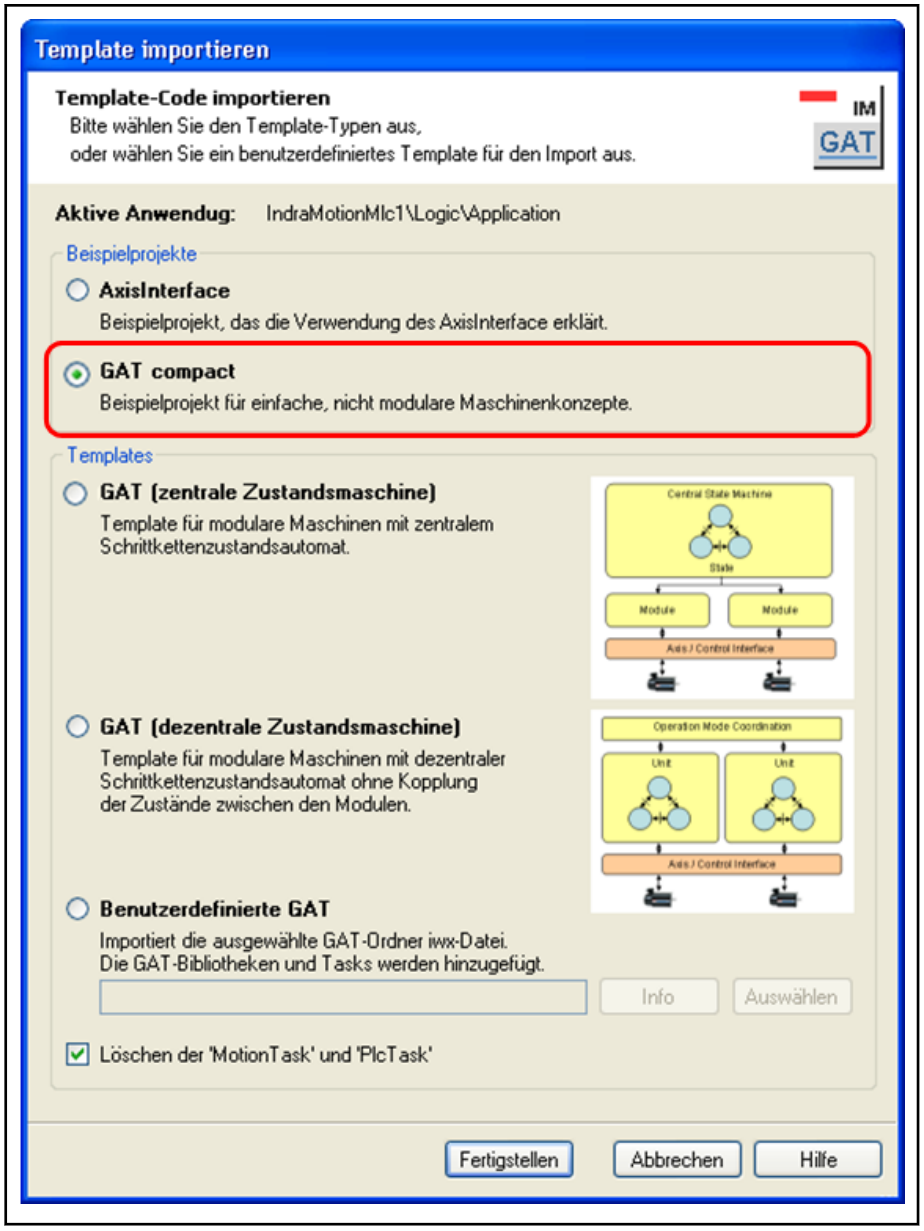


Abb.8-2: GAT-Wizard Startdialog

Durch die Auswahl des GAT^{compact} wird ein lauffähiges und vorkonfiguriertes Projekt importiert. Dieses Projekt enthält bereits 2 virtuelle Achsen.

Zur Realisierung der Aufgabe ist hinsichtlich der Definition von Betriebsarten kein weiterer Aufwand nötig, da die Betriebsart "Automatik" bereits in der Vorkonfiguration enthalten ist.

Durch den Erstimport ist bereits in den Aktionen der "AUTO" Betriebsart Beispielcode vorhanden:

- In der Aktion "actModeAuto_entry" werden die Achsen gestartet. Diese Aktion wird einmalig bei Aktivierung der "AUTO" Betriebsart ausgeführt. Im Beispielcode wir hierbei der Achse "MyAxis1" der Geschwindigkeitsmode (ModeVel) vorgegeben. Weiterhin wird die Achse "MyAxis2" in die Betriebsart Winkelsynchronisation (ModeSyncPhase) geschaltet.
- Die Aktion "actModeAuto_active" wird permanent zyklisch bearbeitet, wenn die "AUTO" Betriebsart aktiv ist. Hierbei werden die Sollgeschwindigkeit der Achse und die angezeigte Geschwindigkeit kopiert.

GATcompact

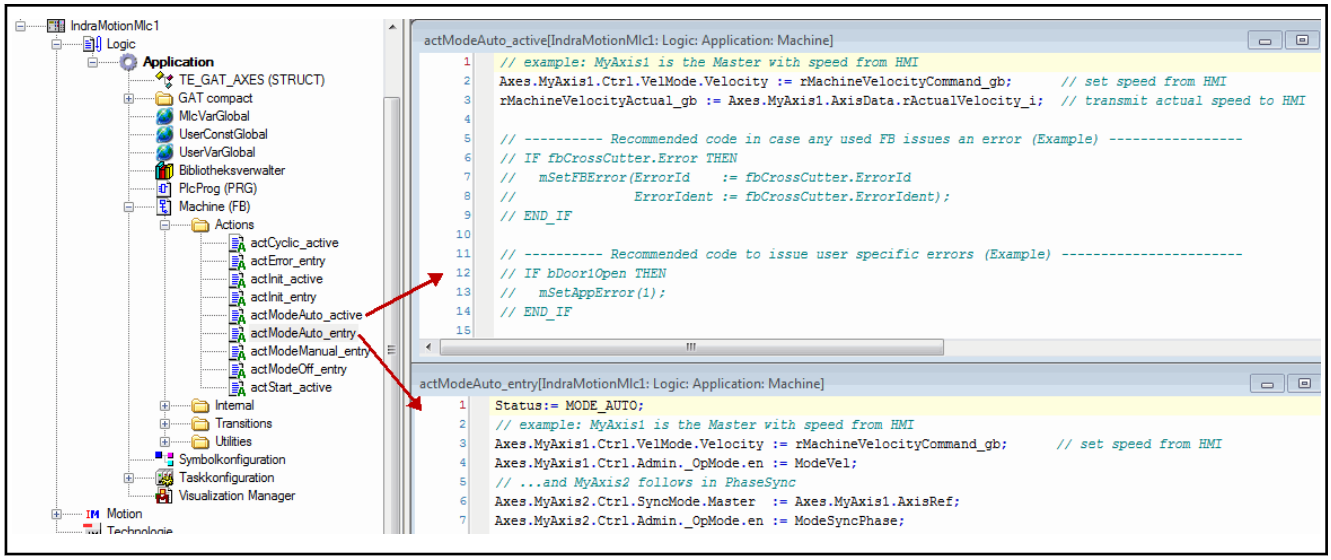


Abb.8-3: Ansteuern der Achsen

Führen Sie folgende Schritte aus um die Anpassungen umzusetzen:

- Achse "MyAxis1" nach "Master" umbenennen. Hierzu den GAT Wizard wie folgt verwenden und **Übernehmen** anklicken.



Abb.8-4: Umbenennung der Achse über "Eigenschaften"

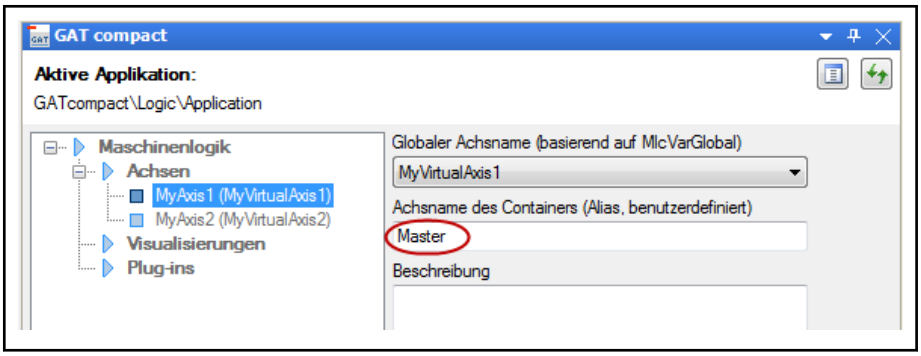


Abb.8-5: Umbenennen der Achse

- In gleicher Art und Weise den Name der zweiten GAT^{compact}-Achse von "MyAxis2" nach "Slave" umbenennen. Die nachfolgende Abbildung zeigt die Ansicht nach den Anpassungen.

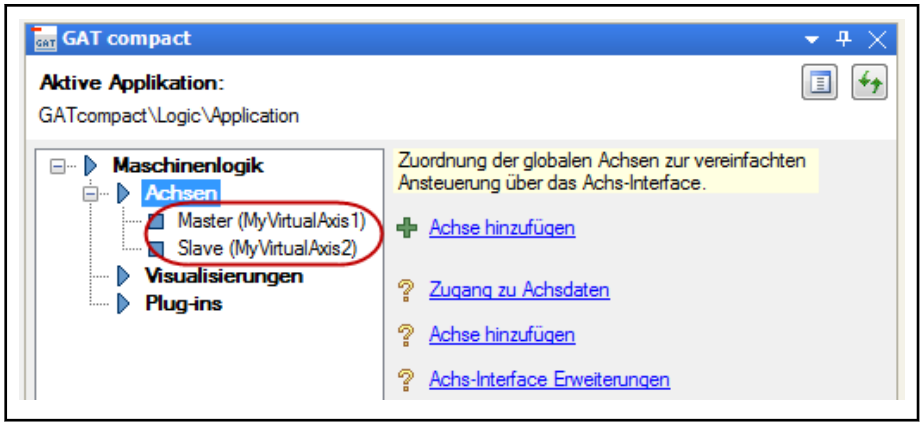


Abb.8-6: Achsübersicht nach Anpassungen

- Durch den GAT^{compact}-Wizard werden nur die Codeänderungen hinsichtlich der Deklaration der GAT^{compact}-Achsen vorgenommen. Passen Sie daher in den Aktionen "actAutoMode_entry" und "actAutoMode_active" den Programmcode mit den geänderten Achsnamen an. Das kann durch "Suchen und Ersetzen" erfolgen oder wird durch die IndraLogic-Intellisense-Funktionalität unterstützt, wenn die Achsnamen unmittelbar in den Aktionen geändert werden. Löschen Sie dazu den Punkt hinter dem Bezeichner "Axes" und fügen ihn wieder hinzu. Daraufhin werden alle Elemente der Struktur "Axes" angezeigt. So können Sie z.B. durch Fokussierung der "Slave"-Achse und <Enter> den Bezeichner "MyAxis2" durch "Slave" ersetzen (siehe nachfolgende Abbildung).

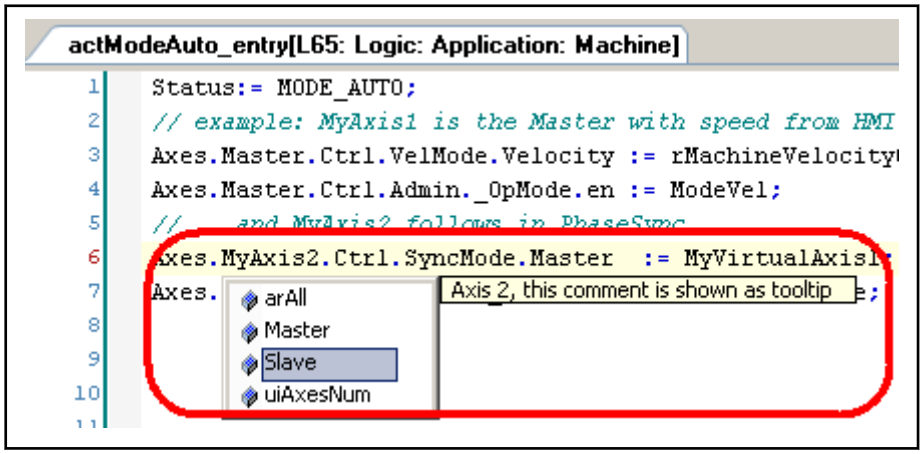


Abb.8-7: Intellisense Funktion

- Nachfolgend das Ergebnis nach den Anpassungen:

GATcompact

actModeAuto_active[GATcompact: Logic: Application: Machine]

```
1  // example: MyAxis1 is the Master with speed from HMI
2  Axes.Master.Ctrl.VelMode.Velocity := rMachineVelocityCommand_gb;      // set speed from
3  rMachineVelocityActual_gb := Axes.Master.AxisData.rActualVelocity_i;  // transmit actual
4
5  // ---------- Recommended code in case any used FB issues an error (Example) ----------
```

actModeAuto_entry[GATcompact: Logic: Application: Machine]

```
1  Status:= MODE_AUTO;
2  // example: MyAxis1 is the Master with speed from HMI
3  Axes.Master.Ctrl.VelMode.Velocity := rMachineVelocityCommand_gb;      // set speed from
4  Axes.Master.Ctrl.Admin._OpMode.en := ModeVel;
5  // ...and MyAxis2 follows in PhaseSync
6  Axes.Slave.Ctrl.SyncMode.Master  := Axes.Master.AxisRef;
7  Axes.Slave.Ctrl.Admin._OpMode.en := ModeSyncPhase;
```

Abb.8-8: Implementierung der Achsansteuerung für das Beispiel

Inbetriebnahme Visualisierung Anpassen

Zur komfortablen Achskommandierung steht Ihnen zusätzlich die dialogbasierte Achskommandierung zur Verfügung. Mehr Details zu dem Thema finden Sie in Kap. 6.4.1 "Dialogbasierte Achskommandierung" auf Seite 73.

Die neuen Achsnamen müssen in der Inbetriebnahmevisualisierung wie folgt angepasst werden:

- Inbetriebnahmevisualisierung über den GAT Wizard öffnen. Hierzu im GAT Wizard Baum den Knoten "Visualisierung" anwählen und "Visualisierung VI Logic öffnen" anklicken.

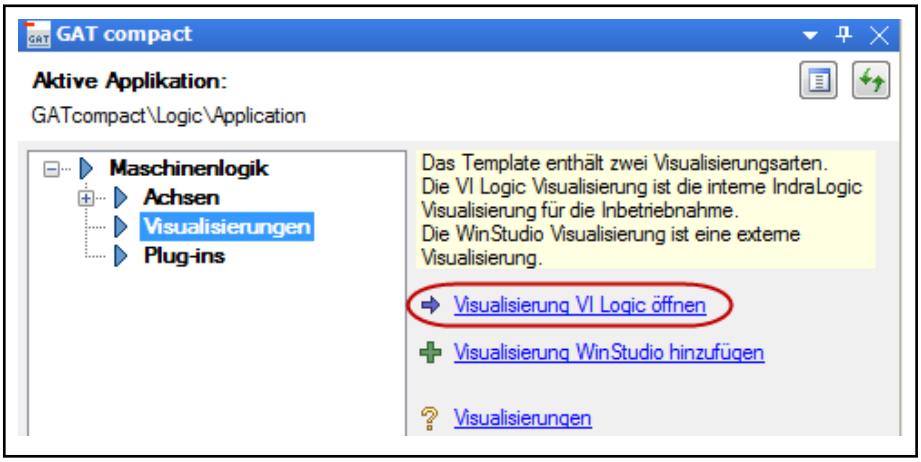


Abb.8-9: Visualisierung über GAT Wizard öffnen

- Das in der folgenden Abbildung markierte Visualisierungsobjekt anklicken und im Eingabefeld "Referenzierte Visualisierungen" den Achsnamen wie folgt anpassen. Damit wirkt diese Achsvisualisierung auf die Achse "Master".

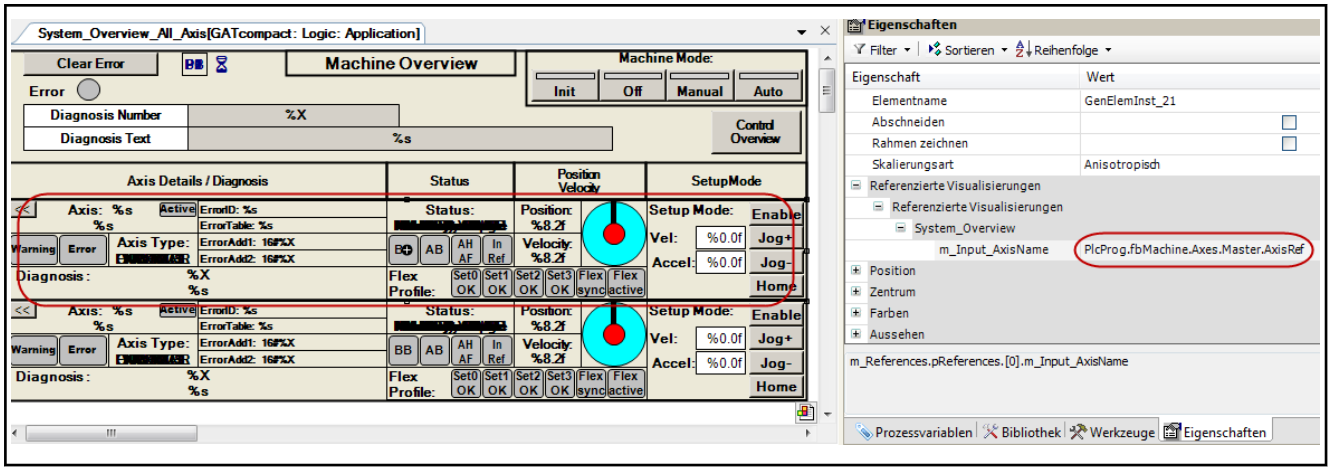


Abb.8-10: Anpassung der Achsvisualisierung für die MasterAxis

Nachfolgend der Text für das Eingabefeld:

`PlcProg.fbMachine.Axes.Master.AxisRef`

- Das darunter liegende Visualisierungsobjekt in gleicher Weise für die "Slave" Achse anpassen.

Nachfolgend der Text für das Eingabefeld der Slave Achse:

`PlcProg.fbMachine.Axes.Slave.AxisRef`

Damit ist die Programmierung abgeschlossen, und das Projekt kann geladen und gestartet werden.

Konfiguration und SPS Projekt Laden und Starten

Wenn alle Änderungen korrekt ausgeführt sind, kann das Projekt fehlerfrei übersetzt und geladen werden.

Nach dem Herunterladen der Konfiguration (1) und des SPS-Projektes (2) kann die SPS gestartet werden (3) (siehe nachfolgende Abbildung).



Abb.8-11: Herunterladen (Konfiguration/SPS-Projekt) und Starten der SPS

Öffnen Sie die Inbetriebnahme Visualisierung über den GAT Wizard Baum unter dem Knoten "Visualisierung" durch die Schaltfläche "Visualisierung VI Logic".

GATcompact

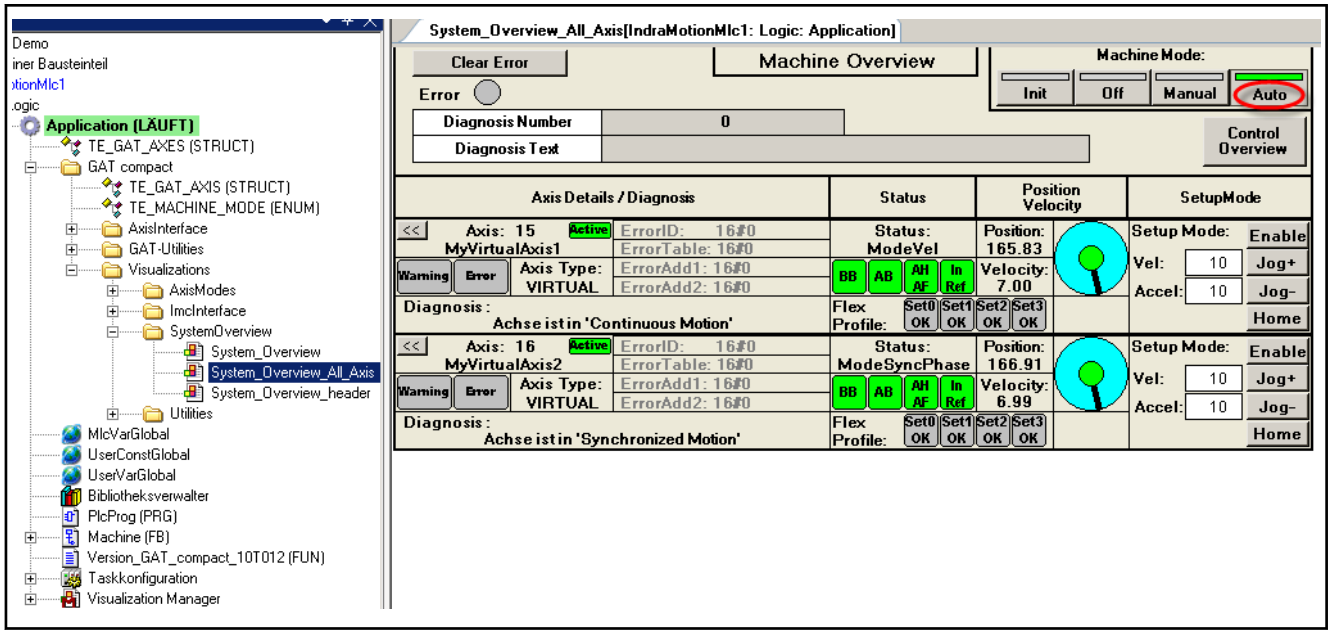


Abb.8-12: Inbetriebnahme Visualisierung

Wählen Sie über **Auto** die Automatik Betriebsart an. In der Betriebsart **Auto** bewegt sich die Achse "Master" im Achscontainer des GAT^{compact} kontinuierlich mit 7U/min (7U/min ist der Defaultwert der Variablen "rMachineVelocity-Command_gb"). Die Achse "Slave" im Achscontainer des GAT^{compact} folgt dieser winkelsynchron.

Betätigen Sie **Manual** um in den Manual Mode zu schalten, und damit die Achsen zu stoppen. Im Manual Mode können Sie anschließend über die jeweiligen Enable Buttons den Setup Modus aktivieren. Im Setup Modus können Sie dann die Achsen über die Jog Tasten vertippen, und über den Home Button referenzieren.

8.3 Initialisierung

Die Initialisierung wird zu Beginn durchlaufen. Nach erfolgreicher Initialisierung befindet sich die Maschine im Betriebsmodus.

Zu Beginn der Initialisierung wird die Anlage nach "Coldstart" (Parametriermodus/P2) kommandiert. Im zweiten Schritt der Initialisierung wird die Anlage nach "Warmstart" (BB) kommandiert.

Befindet sich die Maschine bereits im Betriebsmodus, kann durch Anwahl der Initialisierung die Maschine nach "Coldstart" kommandiert werden. Dieser wird solange aufrecht erhalten, bis eine Betriebsart des Betriebsmodus gewählt wird.

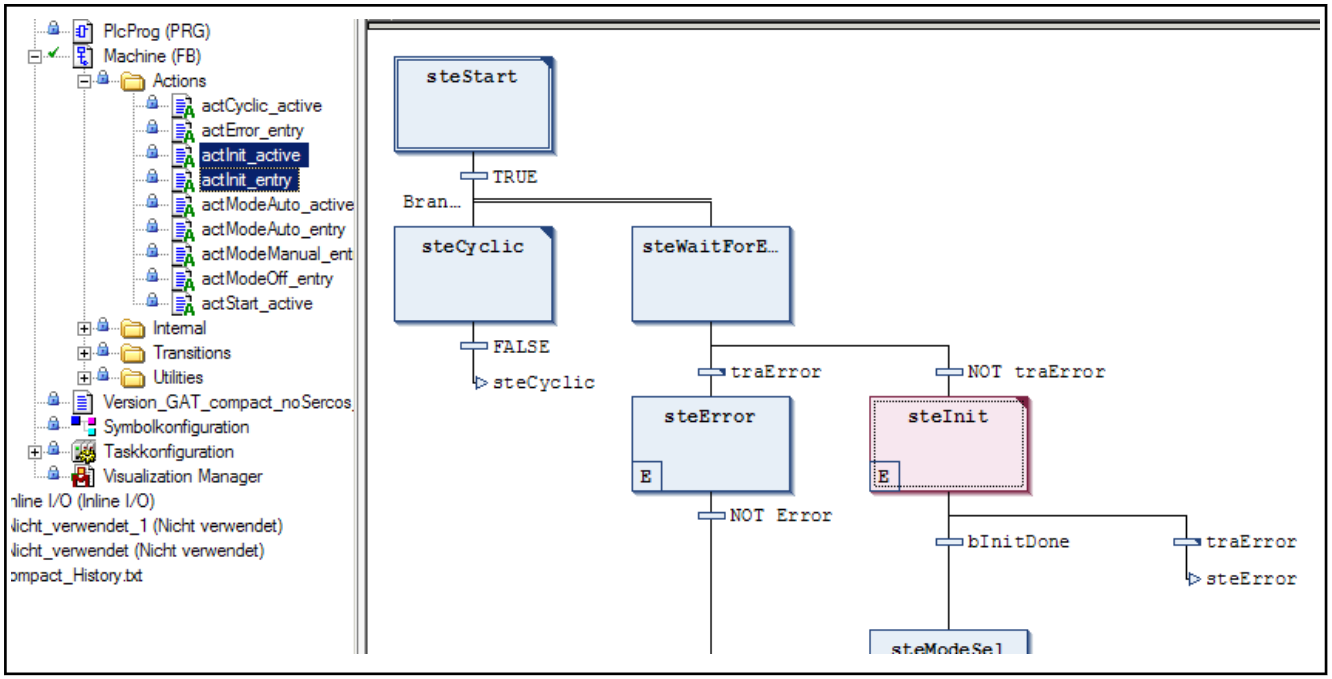


Abb.8-13: Init-Schritt im Machine SFC

Sofern kein Fehler ansteht, wird die Initialisierung im Schritt "steInit" durchgeführt. Zu Beginn des Schrittes wird die Eingangsaktion "steInit_entry" einmalig abgearbeitet.

Im Falle eines Fehlers während der Initialisierung wird in den Fehlerzustand gewechselt. Nach dem Löschen des Fehlers wird die Initialisierung an der Stelle fortgesetzt, an der diese unterbrochen wurde.

Wenn das Bit "bInitDone" gesetzt ist, wird die Initialisierung verlassen und mit dem nächsten Schritt fortgesetzt.

GATcompact

actInit_active[IndraMotionMlc1: Logic: Application: Machine]

```
1  CASE uiStateInit OF
2    uiSTATE_INIT_PREPARE_COLDSTART:
3    bInitDone  := FALSE;  // Init is not done
4    bAllAxesOff := TRUE;
5
6    // switch all axes off to ModeAb
7    FOR uiLoopIndex := 1 TO uiMAX_AXIS_INDEX DO
8      // for all active axes, but only when added by the wizard (<>NO_OBJECT)
9      IF Axes.arAll[uiLoopIndex].Status.Admin.Active                AND
10        Axes.arAll[uiLoopIndex].AxisRef.AxisNo <> NO_OBJECT        AND
11        Axes.arAll[uiLoopIndex].Status.Admin._OpModeAck <> ModeAb
12     THEN
13       bAllAxesOff := FALSE; // at least one is under Power.....
14     END_IF
15   END_FOR
16
17   IF (bAllAxesOff = TRUE)
18   THEN
19     uiStateInit := uiSTATE_INIT_SWITCH_TO_COLDSTART;
20   END_IF
21
22
23   uiSTATE_INIT_SWITCH_TO_COLDSTART:
24   ImcCtrl.Admin._OpMode := IMC_MODE_CTRL_P2;  //Switch your MLC to P2 (Parameter mode)
25
26   IF (ImcStatus.Admin._OpModeAck = IMC_MODE_STATUS_P2)
27   THEN
28     uiStateInit := uiSTATE_INIT_COLDSTART;
29   END_IF
30
31
32   uiSTATE_INIT_COLDSTART:
33   IF (Ctrl <> MODE_INIT)
34   THEN
35     // ************************************************
36     // Now we reached P2 (paramter mode)
37     // Add your code for writing the parameters (in P2) here
38     // ************************************************
39     uiStateInit := uiSTATE_INIT_WARMSTART;
40   END_IF
41
42   uiSTATE_INIT_WARMSTART:
43   ImcCtrl.Admin._OpMode := IMC_MODE_CTRL_BB;  //Switch your MLC to BB (Operation mode)
44
45   IF (ImcStatus.Admin._OpModeAck = IMC_MODE_STATUS_BB) AND
46    (TE_AxisInterfaceTypeOlProg.InitDone = TRUE)
47   THEN
48     // ************************************************
49     // Now we reached BB (operating mode)
50     // Add your code for writing the parameters (in BB) here
51     // ************************************************
```

Abb.8-14: Implementierung der Initialisierung in der Aktion "actInit_active"

Im Schritt "actInit_active" passiert folgendes:

1. Zu Beginn wird geprüft, ob alle aktiven Achsen bereit und momentenfrei (Ab) sind (Zeile 7 bis 15). Andernfalls kann die Maschine nicht nach "Coldstart" (P2) kommandiert werden.
2. Dann wird die Maschine in den Zustand "Coldstart" (P2) kommandiert (Zeile 23 bis 29).
3. Erreicht die Maschine den Zustand "Coldstart", wird der Zustand solange aufrecht gehalten, bis eine Betriebsart des Betriebsmodus (z. B. ModeOff, ModeAuto, ModeManual) kommandiert wird. Dies ist die Stelle an der die Parameter, die nur in P2 änderbar sind, in die Steuerung und in die Antriebe geschrieben werden können (Zeile 32 bis 40).
4. Anschließend wird die Maschine in den Zustand "Warmstart" kommandiert (Zeile 39). Dies ist die Stelle an der die Parameter, die in P4 änderbar sind, zur Initialisierung in die Steuerung und in die Antriebe geschrieben werden können (Zeile 42 bis 51).
5. Wenn die Initialisierung beendet ist, wird das Bit "bInitDone" auf "TRUE" gesetzt und somit die Initialisierung verlassen. Die Maschine ist nun bereit für den Betrieb in den Betriebsarten "Manual" oder "Auto".

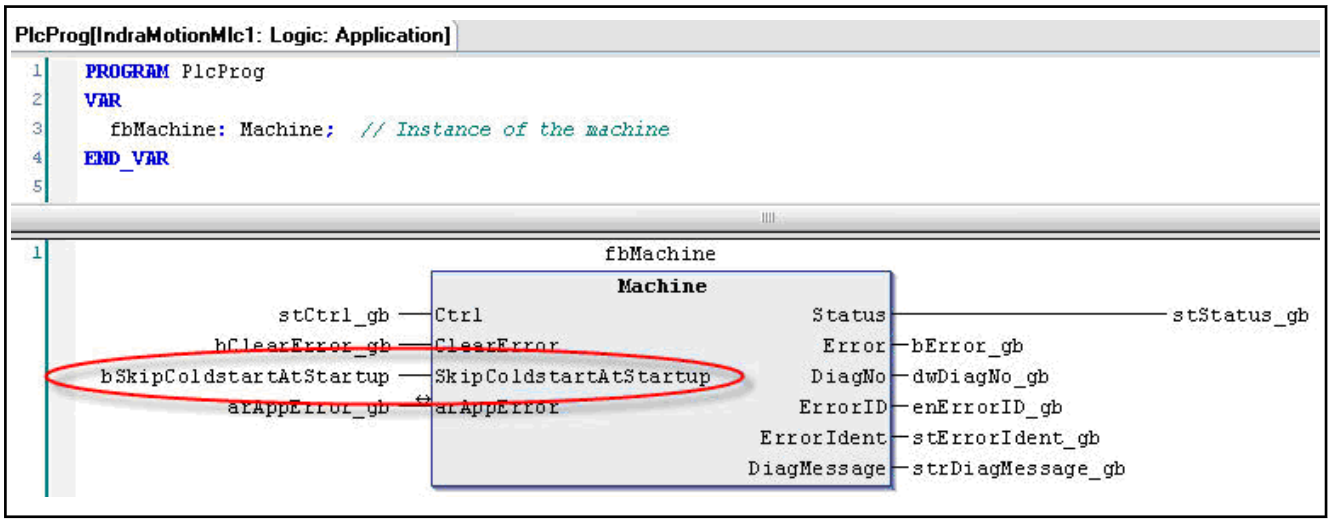


Abb.8-15: Eingang "SkipColdStartAtStartup" am FB "Machine"

Soll die Initialisierung beim erstmaligen Starten der SPS nicht durchlaufen werden, lässt sich dies über das Bit "bSkipColdstartAtStartup" einstellen. Es wird also in den Zustand "Warmstart" gesprungen, "Coldstart" wird nicht abgearbeitet. Somit entfällt das Umschalten in P2 und wieder zurück nach BB. Diese Einstellung spart Zeit beim Hochlauf. Jedoch sollte sichergestellt sein, dass die Aktionen, die im Zustand "Coldstart" abgearbeitet werden (z. B. Parameter schreiben), nicht nötig sind.

8.4 Betriebsarten

Nach erfolgreicher Initialisierung verzweigt der Ablauf in eine der 3 Betriebsarten (siehe nachfolgende Abbildung):

- ModeOff (Alle Antriebe momentenfrei)
- ModeManual (Maschine einrichten, Achsen können getippt werden)
- ModeAuto (zyklischer Automatikbetrieb)

GATcompact

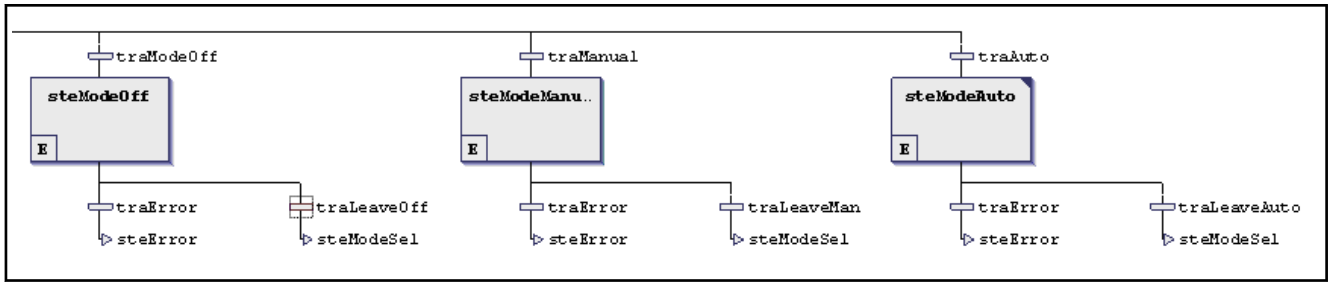


Abb.8-16: Betriebsarten

(siehe Kap. 8.3 "Initialisierung" auf Seite 218)

Im GAT^{compact} ist als Zielbetriebsart nach erstmaligem Start der Maschine "ModeManual" vorinitialisiert:

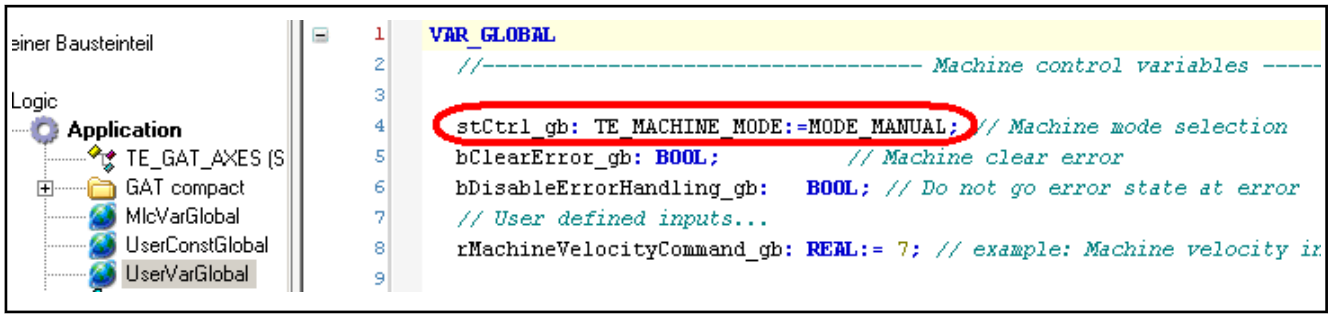


Abb.8-17: Einstellung der Zielbetriebsart nach erstmaligem Start der Maschine

Entry-/Active-Aktion

Mit der Aktivierung einer Betriebsart wird einmalig eine spezielle "entry"-Aktion ausgeführt. Die nachfolgende Tabelle gibt dazu eine Übersicht, zeigt wie jede Betriebsart aktiviert und was in der Betriebsart getan wird.

	Bedingung zur Aktivierung	Einmalige ausgeführte Aktion (zu Beginn)	Zyklisch ausgeführte Aktion
ModeOff **(Antriebe momentenfrei)**	Ctrl= MODE_OFF Abfrage erfolgt in **traModeOff** bzw. **traLeaveOff**	**actModeOff_entry** • Setzt den Status auf "MODE_OFF" • Setzt alle aktiven Achsen auf "ModeAb" (Achsen sind momentenfrei)	Keine zyklische Aktion vordefiniert
ModeManual **(Maschine einrichten, Vertippen der Achsen)**	Ctrl= MODE_MANUAL Abfrage erfolgt in **traManual** bzw. **traLeaveManual**	**actModeManual_entry** • Setzt den Status auf "MODE_MANUAL" • Setzt alle aktiven Achsen auf Mode "ModeAH" (Achse momentenbehaftet mit v=0)	Keine zyklische Aktion vordefiniert Typische Aktivitäten: Einrichten und Justieren der Maschine, z. B. Joggen ausgewählter Achsen
ModeAuto **(zyklischer Automatikbetrieb)**	Ctrl= MODE_AUTO Abfrage erfolgt in **traAuto** bzw. **traLeaveAuto**	**actModeAuto_entry** • Setzt den Status auf "SMODE_AUTO" • Kommandiert "MyVirtualAxis1" beispielhaft in den Achs-Geschwindigkeits-Mode und synchronisiert "MyVirtualAxis2" ohne Phasenoffset auf	**actModeAuto_active** z. B. Bearbeitung von Prozessreglern, Überwachung des Produktionsprozesses (Material vorhanden?, taktsynchrones Ein-/Auskoppeln)

Parallel zur Bearbeitung einer aktiven Betriebsart wird ein permanentes Fehlerscanning ausgeführt. Im Falle des Auftretens eines Fehlers wird in die Fehlerbehandlung verzweigt (siehe Kap. 8.7.1 "Fehlererkennung und Diagnose" auf Seite 227).

Um Betriebsarten hinzuzufügen, siehe Kap. 8.10.5 "Betriebsart hinzufügen" auf Seite 247.

8.5 Zyklische Bearbeitung

Im GAT^{compact} ist der zyklische Programmteil aktuell leer. Allgemeine Informationen siehe Kap. 5.4 "Zyklische Bearbeitung" auf Seite 40.

8.6 Taskhandling

8.6.1 Taskkonfiguration GAT^{compact}

Die Beispielapplikation GAT^{compact} beschränkt sich bewusst auf eine SPS-Task (PlcTask).

Nachfolgend ist die Standard-Einstellung der PlcTask im GAT^{compact} gezeigt:

	Prio	Zykluszeit	Watchdogzeit	Empfindlichkeit des Watchdog
PlcTask	5	T#10ms	T#20ms	3

Abb.8-18:

GATcompact

Eine schnelle MotionTask und eine niederpriore BackgroundTask können bei entsprechendem Bedarf im GAT^{compact} ergänzt werden.

8.6.2 Hinzufügen einer MotionTask

Siehe auch Kap. "Typische Aufgaben MotionTask" auf Seite 42.

	Prio	Zykluszeit	Watchdogzeit	Empfindlichkeit des Watchdog
MotionTask	2	= sercos Zykluszeit	T#2..10ms	1..2

Um eine MOTION-synchrone Task hinzuzufügen, sind folgende Schritte notwendig:

- Taskkonfiguration, hinzufügen der Task "MotionTask", Einstellungen siehe Abb. 8-19 "Konfiguration der MotionTask" auf Seite 225
- Hinzufügen POU "MotionProg" und Aufruf von "MotionProg" in der Konfiguration der MotionTask
- Am FB "Machine" die Methode "mMotionCyclic" hinzufügen. Diese im "MotionProg" aufrufen. Mit Hilfe der Methode "mMotionCyclic" sind alle Variablen (z. B. zentraler Achseinstiegspunkt Axes….) in der Methode verfügbar.

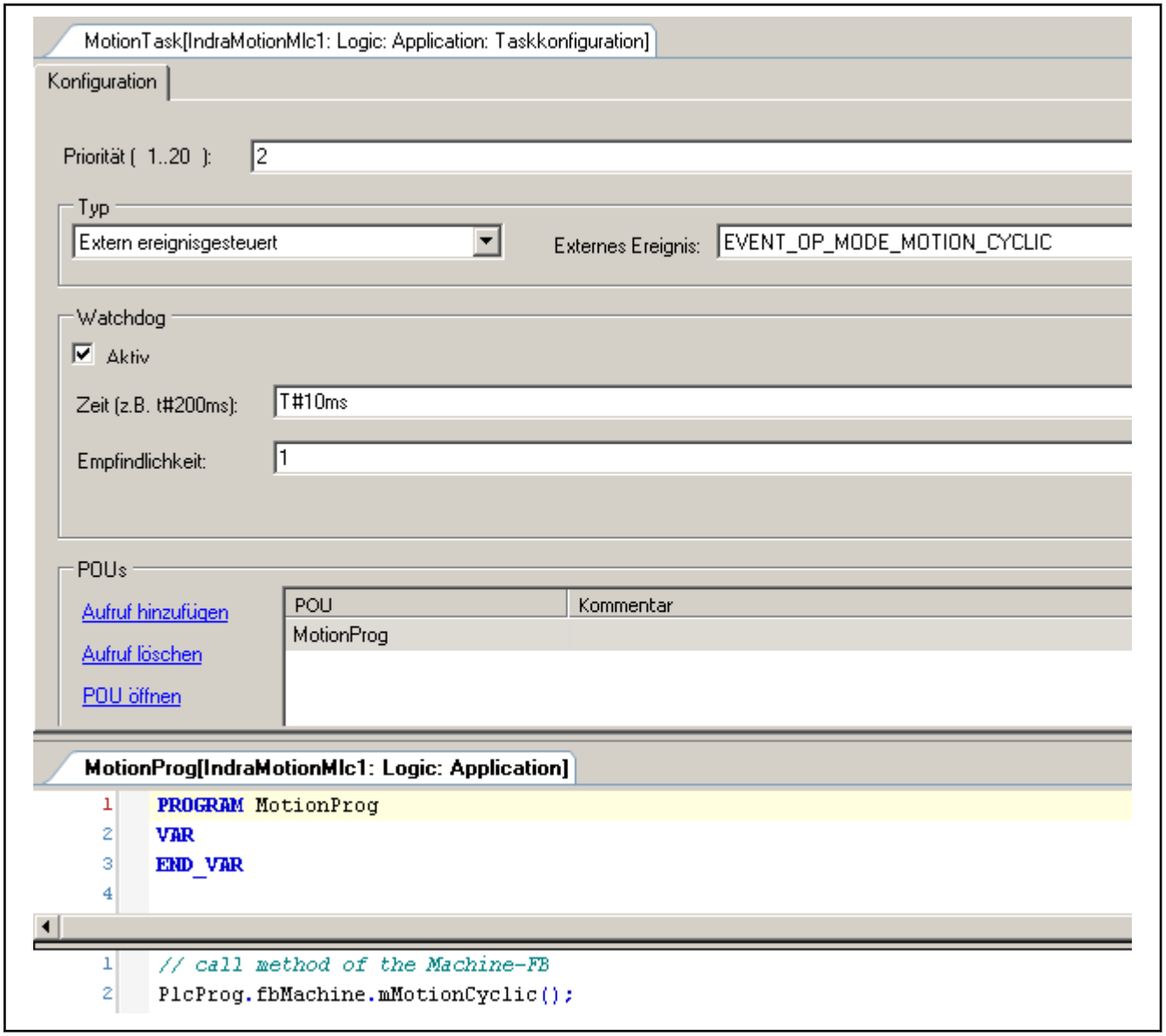


Abb.8-19: Konfiguration der MotionTask

Um aus der MotionTask auf Achsen konsistent zuzugreifen, muss auch das Achs-Interface in der MotionTask aufgerufen werden. Alle Achsen in der MotionTask zu bearbeiten, würde die MotionTask unnötigerweise belasten, deshalb wird die folgende Vorgehensweise empfohlen.

Folgende Schritte sind notwendig:

- Die Struktur "TE_AXIS_CONTROL_TYPE01" (im Ordner "GAT^{compact}/AxisInterface/Type/Control" um das Element "MotionSync" (Typ BOOL) erweitern
- Im Init-Teil (z. B. "actInit_entry") das neue Element "MotionSync" für die ausgewählten Achsen auf TRUE setzen.

  Beispiel: ST Axes.MyVirtualAxis1.Ctrl.MotionSync := TRUE;
- TE_AxisInterfaceType01Prog():
  - Programm um Abfrage des Flags MotionSync = FALSE ergänzen

GATcompact

TE_AxisInterfaceType01Prog[MLC_L25: Logic: Application]

```
1   PROGRAM TE_AxisInterfaceType01Prog
2   (*#################################################################
92
93  IF (InitDone AND bInitReferencesDone) THEN
94    FOR uiLoop := 1 TO uiMAX_AXIS_INDEX DO
95      IF arAxisCtrl_gb[uiLoop].MotionSync = FALSE THEN
96        fbAxis[uiLoop]( ); // call FB, inputs are already set....
97      END_IF
98    END_FOR
99  END_IF
100
```

Abb.8-20: Abfrage des Flags MotionSync = FALSE

– Aktion "actMotionTask" hinzufügen. Die Aufrufschleife des Funktionsbausteines "fbAxis()" kopieren (siehe Abb. 8-20 "Abfrage des Flags MotionSync = FALSE" auf Seite 226). Neue Schleifenvariable "uiLoopMotionSync" anlegen (nicht uiLoop verwenden!), Abfrage Flag "MotionSync" auf TRUE ändern.

Ergebnis:

actMotionTask[MLC_L25: Logic: Application: TE_AxisInterfaceType01Prog]

```
1  // handle the motion sync axes
2  IF (InitDone AND bInitReferencesDone) THEN
3    FOR uiLoopMotionSync := 1 TO uiMAX_AXIS_INDEX DO
4      IF arAxisCtrl_gb[uiLoopMotionSync].MotionSync = TRUE THEN
5        fbAxis[uiLoopMotionSync]( ); // call FB, inputs are already set....
6      END_IF
7    END_FOR
8  END_IF
```

Abb.8-21: Aktion "actMotionTask"

– Aufruf der Aktion "actMotionTask" im MotionProg einfügen (TE_AxisInterfaceType01Prog.actMotionTask();)

8.6.3 Hinzufügen einer BackgroundTask

Falls die Laufzeit der PlcTask zu groß wird, kann durch Auslagern von zeitunkritischen Aufgaben in eine Backgroundtask die PlcTask entlastet werden. Zu bedenken ist dabei, dass durch die zusätzlichen Taskwechsel PlcTask <> BackgroundTask das Gesamtsystem höher belastet wird (siehe auch Kap. "Typische Aufgaben BackgroundTask" auf Seite 42).

	Prio	Zykluszeit	Watchdogzeit	Empfindlichkeit des Watchdog
BackgroundTask	~6	T#40ms	-	-

Um eine BackgroundTask hinzuzufügen sind folgende Schritte notwendig:

- Taskkonfiguration, hinzufügen der Task "BackgroundTask", Einstellungen siehe Abb. 8-22 "Konfiguration der BackgroundTask" auf Seite 227
- Hinzufügen POU "BackgroundProg" und Aufruf von "BackgroundProg" in der Konfiguration der BackgroundTask

- Am FB "Machine" die Methode "mBackground" hinzufügen. Diese im "BackgroundProg" aufrufen. Mit Hilfe der Methode "mBackground" sind alle Variablen in der Methode verfügbar

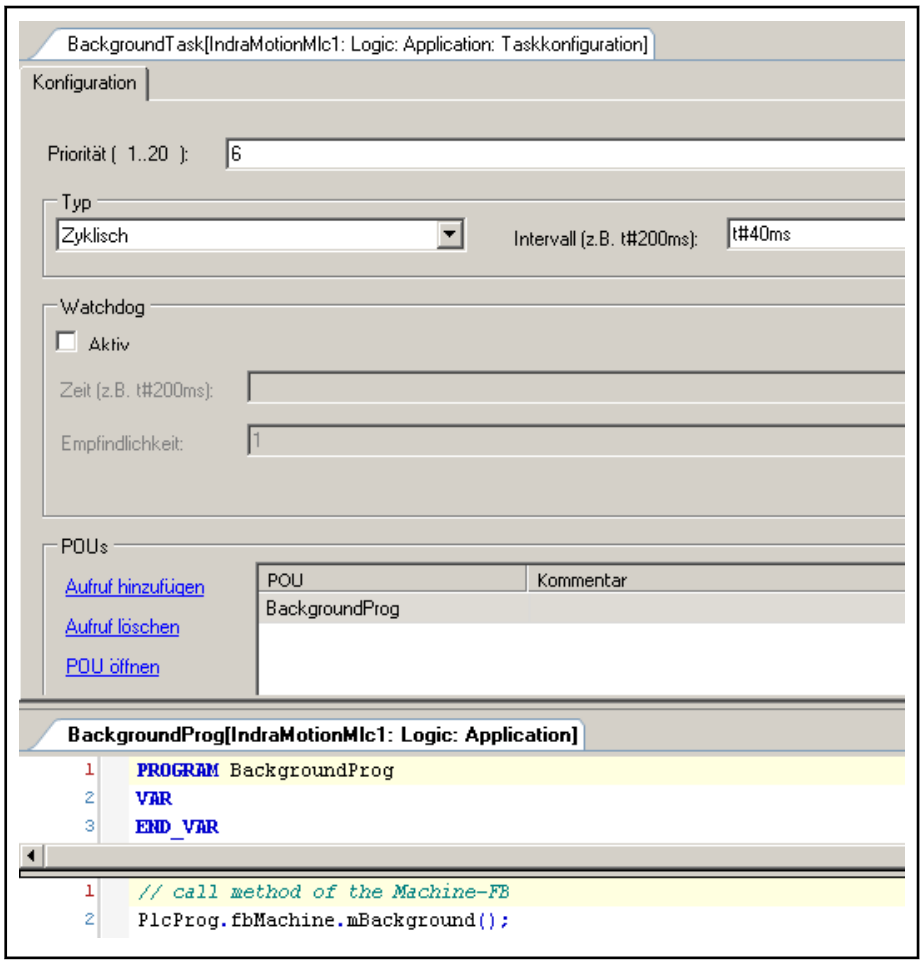


Abb.8-22: Konfiguration der BackgroundTask

8.7 Fehlerhandling

8.7.1 Fehlererkennung und Diagnose

Die Programmiervorlage GAT^{compact} enthält vordefinierte Funktionalitäten zum Erkennen, Behandeln und Löschen von Fehlern an **einer zentralen Stelle**. Fehler der verschiedenen Fehlerquellen (wie z. B. Steuerungsfehler, Applikationsfehler, FB-Fehler…) werden automatisch durch den maschinenübergreifenden FB "Machine" in der Aktion "mErrorHandling" zusammengeführt und über die Diagnoseausgänge des FB "Machine" bereitgestellt.

Falls mehrere Fehler gleichzeitig anstehen, wird der schwerwiegendste angezeigt. Stehen gleichzeitig mehrere Fehler mit dem gleichen Schweregrad an, so wird der zuletzt erkannte Fehler ausgegeben.

GATcompact

Der FB "Machine" enthält die nachfolgend beschriebenen Ausgänge für Diagnosezwecke:

Element	Typ	Beschreibung
Error	BOOL	TRUE signalisiert Fehler
DiagNo	DWORD	Nummer gemäß Dokumentation "Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnosen" mit den folgenden Erweiterungen: • 16#F0AE xxxx: Applikations-Fehler (z. B. geöffnete Schutztür) (xxxx = frei durch den Anwender definierbar) • 16#F0FB 0000: FB-Fehler (z. B. Fehler an einem TechFB) -> Details siehe ErrorID/ErrorIdent • 16#F0AA 0000: GAT interner Fehler -> Details siehe ErrorID/ErrorIdent
ErrorID	ERROR_CODE	Detaildiagnose im Fall von FB- + GAT-Fehlern (DiagNo = 16#F0FB/AA 0000)
ErrorIdent	ERROR_STRUCT	Detaildiagnose im Fall von FB- + GAT-Fehlern (DiagNo = 16#F0FB/AA 0000)
DiagMessage	STRING(60)	Enthält bei Antriebs- und Steuerungsfehlern den Diagnosetext

Abb.8-23:

Der Ausgang **DiagNo** ist zur Nutzung an einem HMI vorgesehen. Die Fehlertexte werden dort der numerischen Information in beliebigen Sprachen zugeordnet.

DiagMessage, ErrorID und **ErrorIdent** sind zusätzliche Informationen, die hauptsächlich bei der Inbetriebnahme durch den Programmierer z. B. über eine IndraLogic-Visualisierung angezeigt werden können.

Hinweis zu applikationsspezifischen Fehlern

Durch Aufrufen der Methode mSetAppError(Number) können Applikationsfehler (wie z. B. geöffnete Schutztür) abgesetzt werden. Die übergebene "Number" am Eingang der Methode entspricht hierbei der Applikationsfehlernummer.

Das zentrale Diagnosesystem im FB "Machine" überwacht hierbei zyklisch die abgesetzten Anwenderfehler und erzeugt den Applikationsfehler DiagNo = Fehler 16#F0AE xxxx (xxxx = Applikationsfehlernummer), wenn ein Applikationsfehler abgesetzt wurde.

In der globalen Variablenliste "UserConstGlobal" können den Applikationsfehlern Diagnosetexte zugeordnet werden, die dann in der SPS-Visualisierung angezeigt werden.

Für Applikationsfehler in der Maschine sind Nummern im Bereich 1 … uiMAX_APP_ERROR_INDEX gültig. Die Konstante "uiMAX_APP_ERROR_INDEX" ist mit dem Wert 64 vorbelegt. Bei Bedarf kann diese Konstante in der Globalen Variablenliste "UserConstGlobal" unterhalb des "Application"-Ordners angepasst werden.

Codebeispiel zum Absetzen eines Applikationsfehlers:

```
// Recommended code to issue user specific errors (Example)
IF bDoor1Open THEN
    mSetAppError(1);
END_IF
```

Hinweis zu Fehler-Schweregrad

Das Steuerungs-Diagnosesystem definiert 10 Schweregrade "Severity" (siehe IndraLogic XLC-/IndraMotion MLC-Diagnosesystem).

Der Schweregrad hat Einfluss auf die Fehlerreaktion:

- DiagNo = 16#F0xx xxxx ist der geringste Schweregrad (z. B. Applikationsfehler, FB-Fehler)
- DiagNo = 16#F9xx xxxx ist der höchste Schweregrad (fataler Systemfehler)

Es wird der Fehler mit dem höchsten Schweregrad angezeigt, wenn mehrere Fehler gleichzeitig anstehen (siehe Kap. 5.6.4 "Fehlerreaktion" auf Seite 47).

Hinweis zu Funktionsbaustein-Fehler

Treten Fehler an aufgerufenen Funktionsbausteinen (z. B. Technologie-FB) auf, so wird empfohlen die Methode "mSetFBError(ErrorID, ErrorIdent)" aufzurufen und die "ErrorID" und "ErrorIdent" Ausgänge des Fehler verursachenden FB an die Methode zu übergeben. Damit wird sichergestellt, dass der Fehler an das Diagnosesystem des FB "Machine" weitergeleitet wird und die "ErrorID" und "ErrorIdent" an den Diagnoseausgängen des FB "Machine" bereitgestellt wird.

Codebeispiel zum Absetzen eines Funktionsbaustein-Fehlers:

```
// Recommended code a used FB issues an error (Example)
IF fbCrossCutter.Error THEN
    mSetFBError(ErrorID:=fbCrossCutter.ErrorID,
                ErrorIdent:=fbCrossCutter.ErrorIdent);
END_IF
```

8.7.2 Fehlerreaktion

Die Fehlerreaktion hat die Aufgabe Schäden aufgrund von Fehlern des Steuerungs- und Antriebssystems möglichst zu vermeiden. Daher wird bei Achs- und Steuerungsfehlern ab Schweregrad 2 automatisch eine Fehlerreaktion durch die Steuerungsfirmware und die realen Achsen durchgeführt (siehe Kap. 5.6.4 "Fehlerreaktion" auf Seite 47).

Bei allen anderen Fehlern mit kleineren Schweregraden (wie z. B. Applikationsfehler, FB-Fehler…) wird keine automatische Fehlerreaktion durch die Steuerungsfirmware und die realen Achsen durchgeführt.

Daher enthält das GAT^{compact} eine Funktionalität, um Fehler zu erkennen und eine anwenderspezifische Fehlerreaktion durchzuführen.

Allgemeine GAT Fehlerreaktion

Mit der Standardeinstellung ist die GAT^{compact} Fehlerreaktion eingeschaltet. Im Fehlerfall wird dadurch die angewählte Betriebsart unterbrochen (Ausnahme: Initialisierung) und die Fehlerreaktion ausgeführt. Die Fehlerreaktion ist in der Aktion "actError_entry" des Funktionsbausteines "Machine" programmiert und kann applikationsspezifisch angepasst werden.

Für den Standardfall ist in dieser Aktion das Verzögern aller Achsen mit dem "ModeAH" programmiert (ModeAH = Verzögern bis zum Stillstand und Drehmoment bleibt eingeschaltet):

GATcompact

```
// ----------------------- Error reaction example 1 --------------------
// Decelerate all module axes to standstill and hold power (e.g. hanging axes)
// --> switch all axes to ModeAH
// alternative: set to ModeAb to switch the axes OFF to Ab
// ---------------------------------------------------------------------
FOR uiLoopIndex := 1 TO uiMAX_AXIS_INDEX DO
  // for all active axes, but only when added by the wizard (<>NO_OBJECT)
  IF Axes.arAll[uiLoopIndex].Status.Admin.Active         AND
     Axes.arAll[uiLoopIndex].AxisRef.AxisNo <> NO_OBJECT     THEN
    Axes.arAll[uiLoopIndex].Ctrl.Admin._OpMode.en := ModeAH; // or set ModeAb
    // reset SetupMode Inputs to avoid hanging buttons in the
    // System_Overview_All_Axis visualisation
    Axes.arAll[uiLoopIndex].Ctrl.SetupMode.Enable := FALSE;
    Axes.arAll[uiLoopIndex].Ctrl.SetupMode.JogMinus := FALSE;
    Axes.arAll[uiLoopIndex].Ctrl.SetupMode.JogPlus := FALSE;
    Axes.arAll[uiLoopIndex].Ctrl.SetupMode.Homing := FALSE;
  END_IF
END_FOR
```

Abb.8-24: Standardfall für die Fehlerreaktion in der Aktion "actError_entry"

Wenn in dem obigen Beispiel "ModeAH" durch "ModeAb" ersetzt wird, dann erfolgt nach der Verzögerung im Stillstand die Momentenfreischaltung der realen Achsen.

☞ Für hydraulische Achsen wird eine Stillsetzung über "ModeAb" empfohlen.

☞ Bei **"hängenden Achsen"** darf keine Momentenfreischaltung erfolgen, um ein Absenken der Achsen zu vermeiden. Bei diesen Anwendungen wird empfohlen, alle Achsen in "ModeAH" zu schalten.

Oftmals besteht die Anforderung, Achsen im **Synchronverbund** zu verzögern, um z. B. das Reißen einer Materialbahn zu vermeiden. Bei diesen Anwendungen wird empfohlen, alle Achsen im Synchronverbund zu belassen, und nur den Master zu verzögern. Dieser Anwendungsfall ist im folgenden Beispiel dargestellt:

Beispiel für **synchrone Stillsetzung** der Achsen in der Modulaktion "actError_entry" (nur nicht synchrone Achsen werden über "ModeAH" verzögert).

```
// ------------------------ Error reaction example 2 ---------------------
// Keep all axes in synchronization with the master axis
// and decelerate only the master axis
// -----------------------------------------------------------------------
FOR uiLoopIndex := 1 TO uiMAX_AXIS_INDEX DO
  // for all active axes, but only when added by the wizard (<>NO_OBJECT)
  IF Axes.arAll[uiLoopIndex].Status.Admin.Active         AND
    Axes.arAll[uiLoopIndex].AxisRef.AxisNo <> NO_OBJECT     THEN
    IF (Axes.arAll[uiLoopIndex].AxisData.SynchronizedMotion) THEN
    ; // Axis is in synchronized motion --> no reaction as the master axis
    // decelerates to standstill and the slave axis follows in synchronization
    ELSE
      Axes.arAll[uiLoopIndex].Ctrl.Admin._OpMode.en := ModeAH;
      // Axis is not in synchronized motion --> stop axis
  END_IF
    // reset SetupMode Inputs to avoid hanging buttons in the
    // System_Overview_All_Axis visualisation
    Axes.arAll[uiLoopIndex].Ctrl.SetupMode.Enable := FALSE;
    Axes.arAll[uiLoopIndex].Ctrl.SetupMode.JogMinus := FALSE;
    Axes.arAll[uiLoopIndex].Ctrl.SetupMode.JogPlus := FALSE;
    Axes.arAll[uiLoopIndex].Ctrl.SetupMode.Homing := FALSE;
  END_IF
END_FOR
```

Abb.8-25: Beispiel für die synchrone Stillsetzung der Achsen in der Aktion "actError_entry"

Für die synchrone Stillsetzung sind bei realen Achsen folgende zusätzliche Einstellungen notwendig:

- Modulbusreaktion abschalten
- Fehlerreaktion der Antriebe mit "Bestmöglichen Stillsetzung = 30s NC Reaktion" konfigurieren

---

☞ Bei der Einstellung "Bestmöglichen Stillsetzung = 30s NC Reaktion" einer realen Achse wird bei dem IndraLogic-XLC/IndraMotion MLC-Steuerungssystem die fehlerbehaftete Achse bei einem F2-Fehler mit der Notstopp-Rampe A-0-0224 verzögert. Alle anderen fehlerfreien Achsen bleiben unbeeinflusst.

Abschalten der allgemeinen GAT Fehlerreaktion

Für Sonderfälle kann die allgemeine GAT Fehlerreaktion abgeschaltet werden.

Ändern Sie dazu die Standardeinstellung der Variablen "bDisableErrorHandling_gb" über die Variablendeklaration wie folgt:

- Globale Variablenliste: "UserVarGlobal"
- Deklaration wie folgt anpassen: "bDisableErrorHandling_gb: BOOL := TRUE;"

Bei abgeschalteter Fehlerreaktion wird die aktuelle Betriebsart nicht unterbrochen. Fehler werden jedoch weiterhin angezeigt, und können auch gelöscht werden.

GAT Fehlerreaktion während der Initialisierung

Mit der Standardeinstellung ist die GAT^{compact} Fehlerreaktion während der Initialisierung abgeschaltet. Auftretende Fehler werden jedoch angezeigt und können gelöscht werden.

Für Sonderfälle kann die Fehlerreaktion während der Initialisierung eingeschaltet werden.

Ändern Sie dazu die Standardeinstellung der Variablen "bEnableErrorReactionInit_gb" über die Variablendeklaration wie folgt:

- Globale Variablenliste: "UserVarGlobal"
- Deklaration wie folgt anpassen: "bEnableErrorReactionInit_gb: BOOL := TRUE;"

Wenn die Fehlerreaktion eingeschaltet ist, dann wird die Initialisierung bei Fehlerauftritt unterbrochen. Nachfolgend wird dann die programmierte Fehlerreaktion ausgeführt. Die Fehlerreaktion ist in der Aktion "**actError_entry**" des Funktionsbausteines "Machine" programmiert.

☞ Die Fehlerreaktion während der Initialisierung wird nur dann ausgeführt, wenn die allgemeine GAT Fehlerreaktion eingeschaltet ist (bDisableErrorHandling_gb = FALSE).

8.7.3 Fehlerlöschen

Über eine positive Flanke am Eingang **"ClearError"** des FB "Machine" wird das zentrale Fehlerlöschen automatisch durchgeführt. Der FB leitet dieses Kommando an alle unterlagerten Komponenten (wie z. B. Imc-Interface[1)] , AxisInterface …) weiter um das gesamte System in den fehlerfreien Zustand zu überführen. Die aktiv angewählte Betriebsart wird nach Fehlerlöschen wieder bearbeitet, falls diese zur Fehlerreaktion verlassen wurde.

8.8 Visualisierung

8.8.1 Interne Visualisierung zur Inbetriebnahmeunterstützung

Allgemeines

Das GAT[compact] beinhaltet vordefinierte IndraLogic-Visualisierungsobjekte, die bei der Inbetriebnahme unterstützen und erweiterte Diagnosemöglichkeiten bieten.

System_Overview

Das Visualisierungselement "System_Overview" ist für einzelne Achsen instanziierbar.

In der Visualisierung "System_overview_All_Axis" ist dies exemplarisch für die Achsen "MyVirtualAxis1" und "MyVirtualAxis2" umgesetzt.

1) *IndraMotionControl Interface = IndraLogic Bibliothek mit Schnittstelle zur Steuerung, wie. z. B. Fehler löschen*

GATcompact

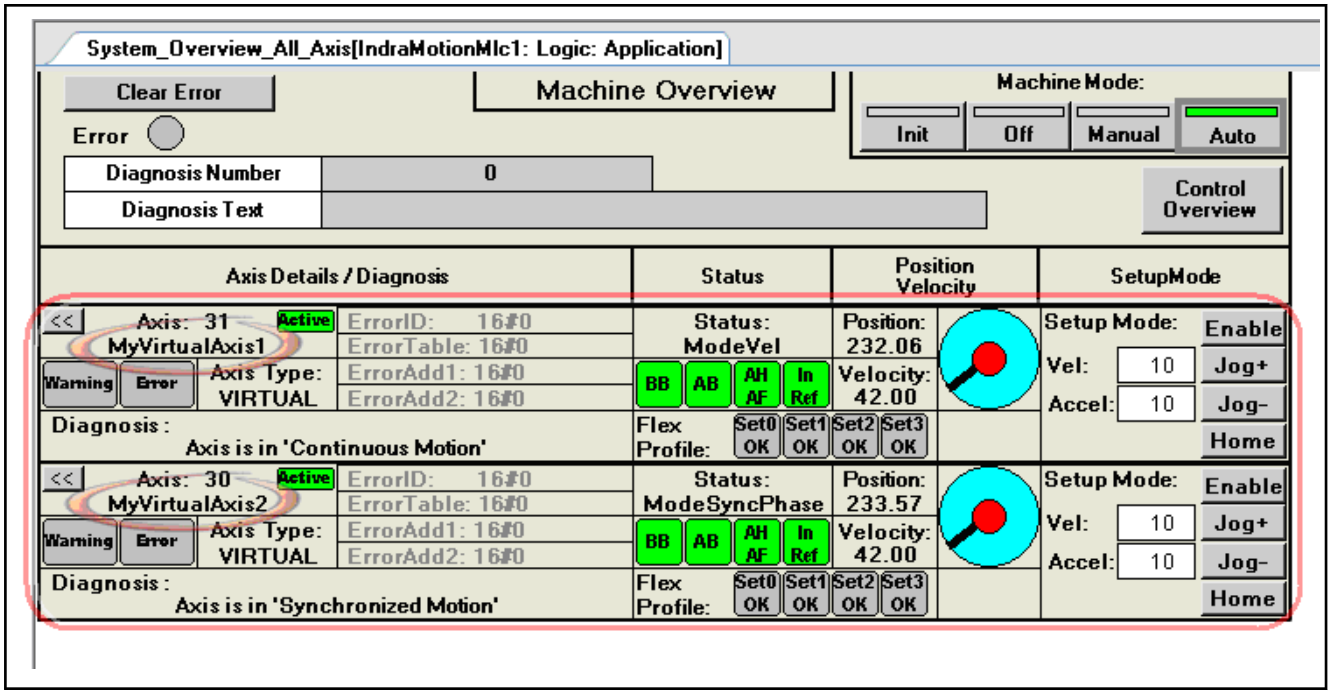


Abb.8-26: Interne Visualisierung zur Inbetriebnahmeunterstützung

Über das Objekt "Frame" kann in der Visualisierung eine Instanz des Visualisierungsobjekts "System_Overview" angelegt werden.

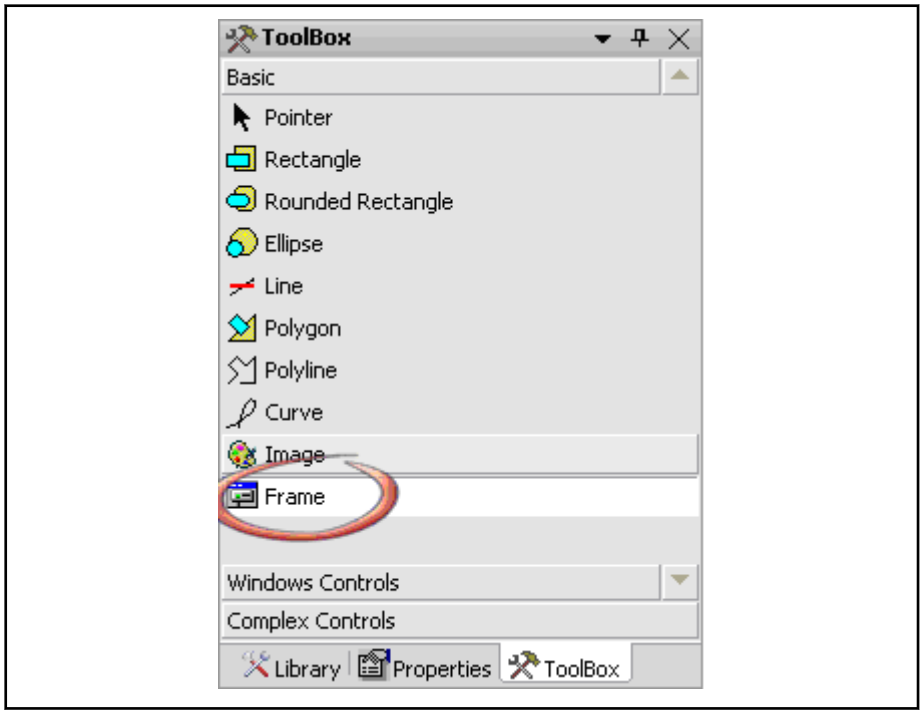


Abb.8-27: Anlegen einer Instanz des Visualisierungsobjekts "System_Overview"

GATcompact

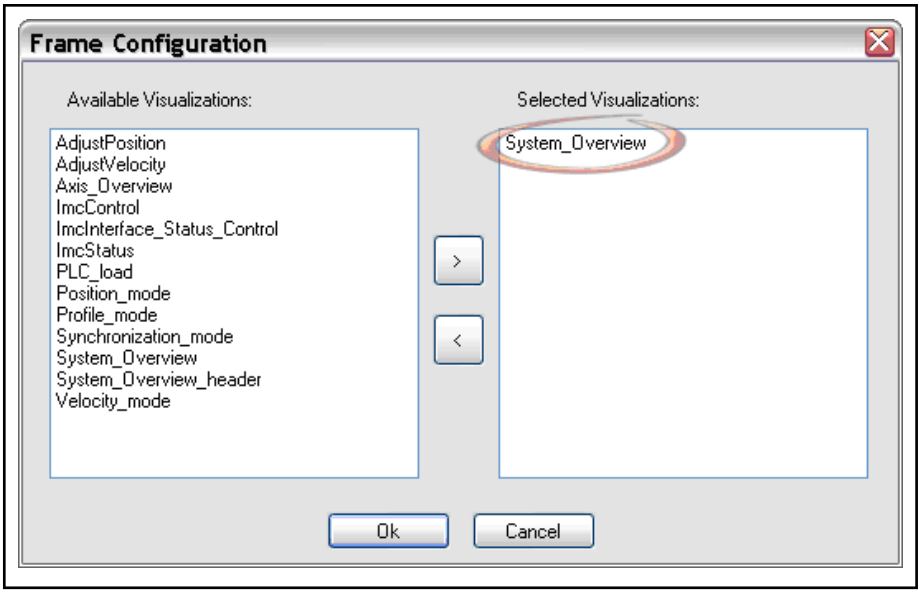


Abb.8-28: Anlegen einer Instanz des Visualisierungsobjekts "System_Overview"

Unter Properties – References muss nun die entsprechende Achsreferenz eingetragen werden.

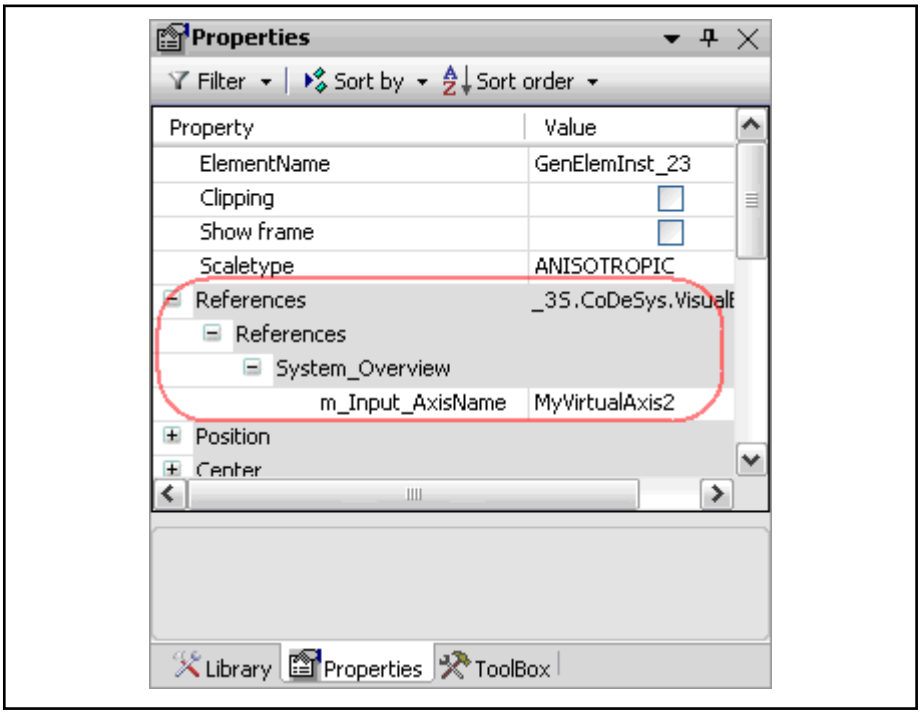


Abb.8-29: Anlegen einer Instanz des Visualisierungsobjekts "System_Overview"

8.8.2 Externe Visualisierung mit separaten HMI (WinStudio-Visualisierung)

Allgemeines

Für das GAT^{compact} steht eine vordefinierte WinStudio-Visualisierung zur Verfügung, die über den Wizard in das Projekt eingefügt werden kann.

Zur Bearbeitung der WinStudio-Visualisierung wird mindestens folgende Entwicklungsumgebungslizenz oder größer benötigt:

WinStudio-Entwicklungslizenz, max. 1.500 Variablen (Mat.-Nr. R911323596).

Zur Ausführung auf einem VEP wird mindestens folgende Runtime-Lizenz benötigt:

WinStudio-Runtimelizenz für Geräte mit Windows CE als Betriebssystem, max. 1.500 Variablen (Mat.-Nr. R911323620).

Einfügen der WinStudio-Visualisierung

Um die WinStudio-Visualisierung dem Projekt hinzufügen zu können, muss ein entsprechendes Visualisierungsgerät im Projekt vorhanden sein.

Fügen Sie mittels "Drag and Drop" ein VEP aus der Gerätebibliothek in den Projektbaum ein und konfigurieren Sie es wie folgt:

- Gerätetyp: VEPxx (CE)
- Applikationstyp: WinStudio-Anwendung

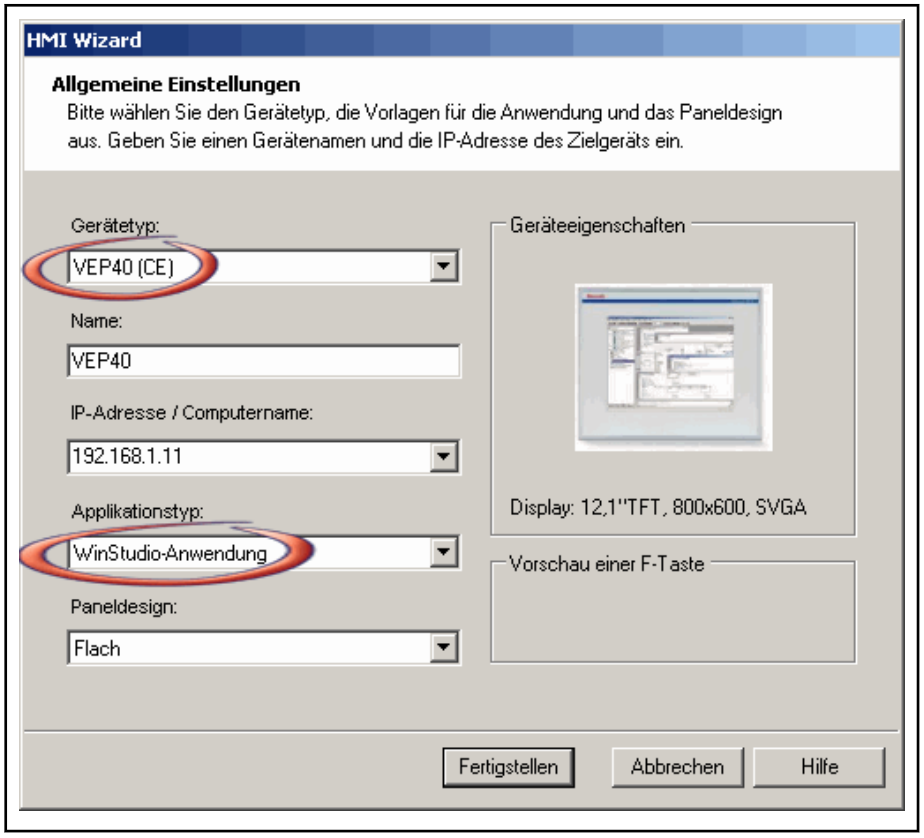


Abb.8-30: Visualisierungsgerät einfügen

Wenn ein entsprechendes Visualisierungsgerät vorhanden ist, wird über folgenden Link im Wizard die WinStudio-Visualisierung eingefügt:

GATcompact

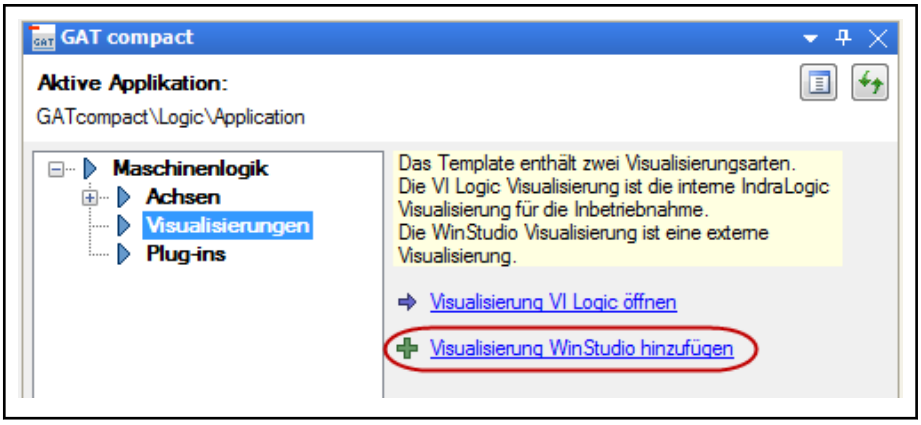


Abb.8-31: WinStudio-Visualisierung einfügen

Im nächsten Dialog des Wizard werden alle Visualisierungsgeräte des Projekts angeboten, deren Konfiguration der Vorgabe entspricht.

Nach Auswahl des entsprechenden Visualisierungsgerätes kann mittels **Übernehmen** das Einfügen der WinStudio-Visualisierung ausgeführt werden.

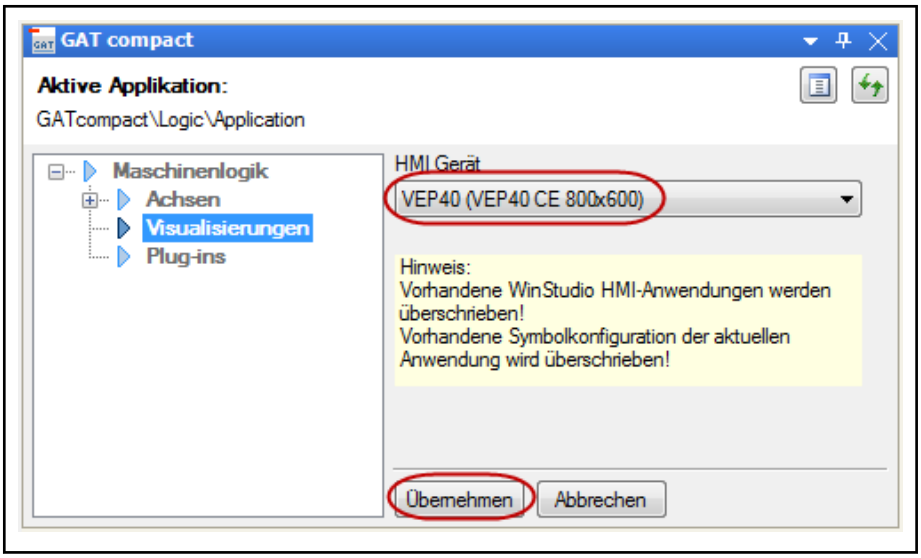


Abb.8-32: Visualisierungsgerät auswählen – WinStudio-Visualisierung einfügen

☞ Bestehende Visualisierungsdaten des Visualisierungsgerätes und die Symbolkonfiguration der Steuerung werden durch das Einfügen der WinStudio-Visualisierung überschrieben!

Nach dem Einfügen können die Visualisierungsdaten bearbeitet werden.

Die Konfiguration der Steuerungsadresse erfolgt im "Eigenschaften"-Dialog des BR-WS Treibers.

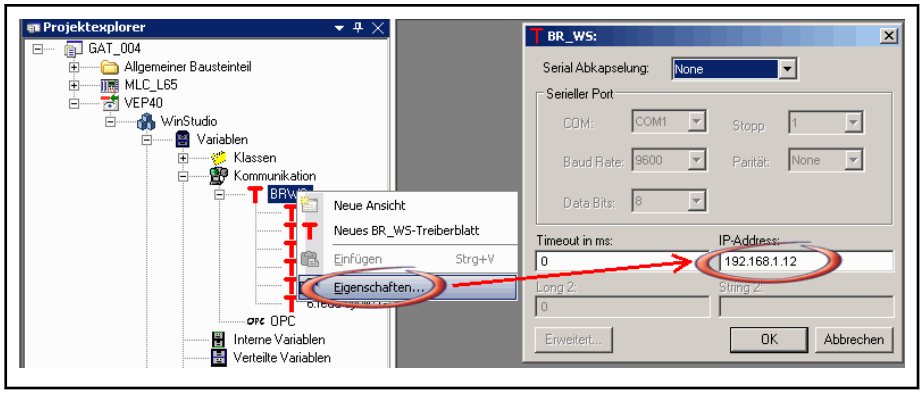


Abb.8-33: Einstellen der IP-Adresse der Steuerung

Im Kontextmenü des WinStudio-Knotens kann nun über  "Anwendung Starten" die Emulation der Winstudio-Visualisierung gestartet werden.

Dialog-Struktur

Die Bedienbilder teilen sich in 3 Teilbereiche auf:

- Kopfzeile mit Statusinformationen
- Hauptbildbereich mit menüabhängigen Anzeigen und Funktionstasten
- Fußzeile mit Tasten zur Bedienbildumschaltung

GATcompact

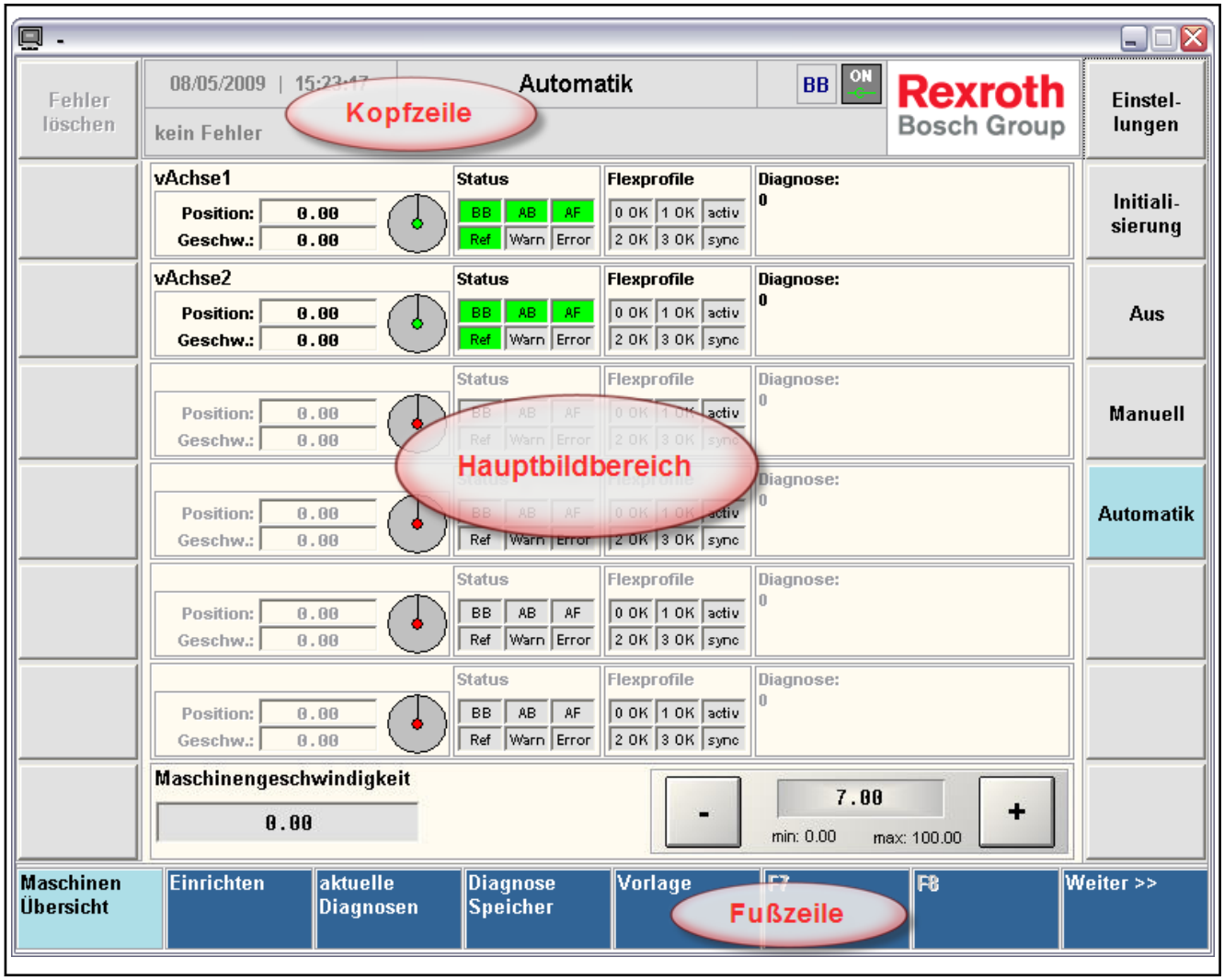


Abb.8-34: WinStudio-Visualisierung

*Die **Kopfzeile** beinhaltet:*

- Betriebsartenanzeige
- Globale Funktionstasten zum Fehlerlöschen und Öffnen des "Einstellungen"-Dialogs (Sprachumschaltung, Einstellung der Dezimalstellen für numerische Anzeigen)
- Anzeige aktuell anstehender Maschinen-Diagnosen
- Anzeige der aktuellen sercos Phase (inklusive aktiver Phasenumschaltung)
- Anzeige des Kommunikationsstatus

Der **Hauptbildbereich** beinhaltet die bedienbildabhängigen Anzeigen und Funktionstasten (z. B. Achsinformationen, Betriebsartenumschaltung, …).

Die **Fußzeile** beinhaltet die Tasten zur Bedienbildumschaltung.

Erweitern der Visualisierung

Bedienbild einfügen

Zum Erweitern der Visualisierung können neue Bedienbilder mittels Kopieren der vorhanden "Template"-Bilder und Einfügen unter neuem Namen angelegt werden.

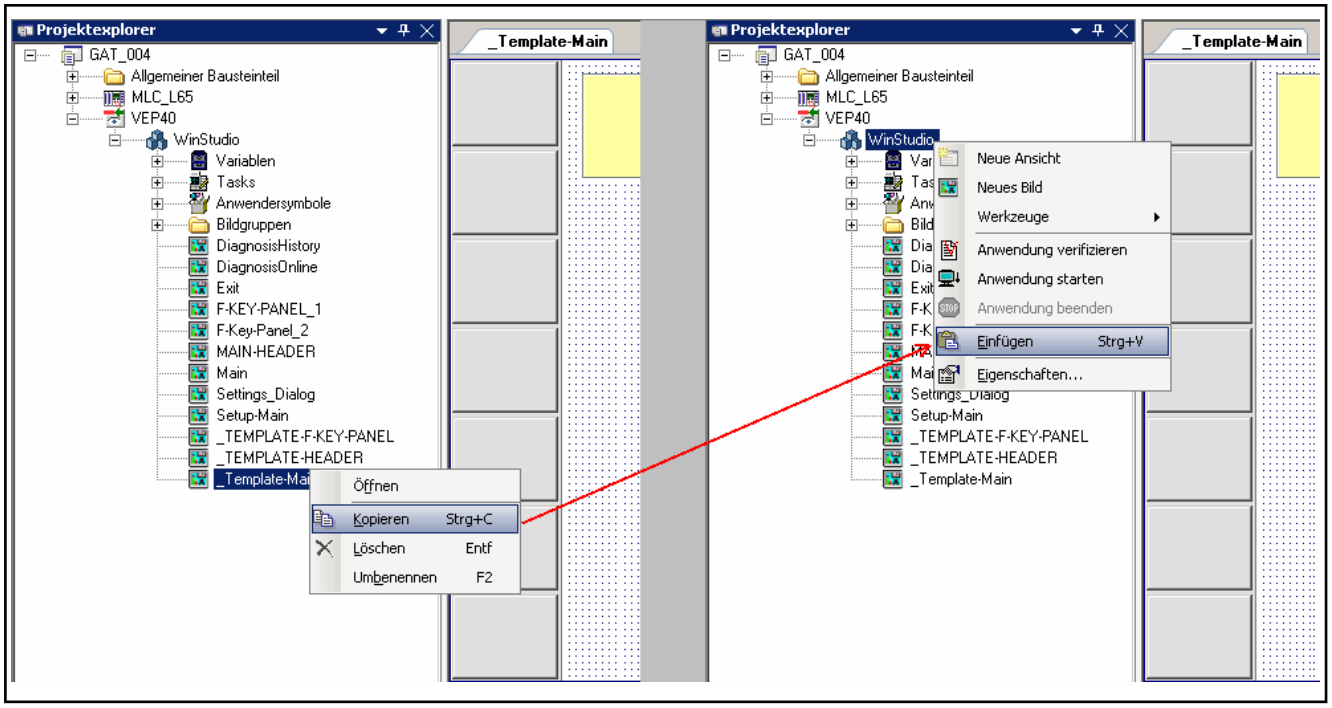


Abb.8-35: Bedienbild duplizieren

Die Elemente zum Anzeigen der Achsdaten sind gruppierte Einzelobjekte, die durch "Kopieren" und "Einfügen" dupliziert werden können. Zum Umkonfigurieren des eingefügten Objekts auf eine andere Achse muss nun über die Replace-Funktion des jeweiligen Objekts der Array-Index der Achse angepasst werden (siehe nachfolgende Abbildungen).

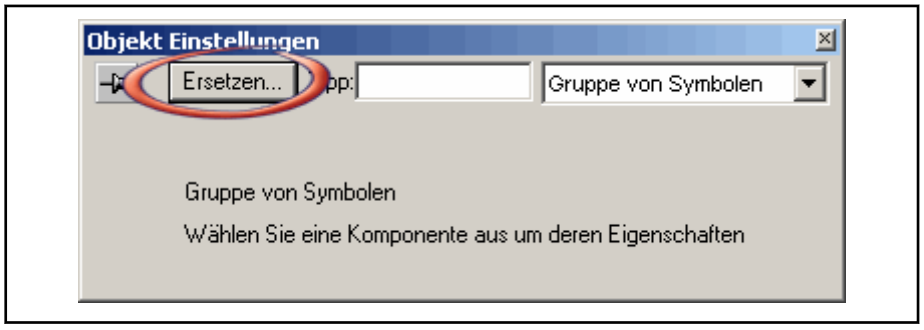


Abb.8-36: Umkonfigurieren des eingefügten Objekts

GATcompact

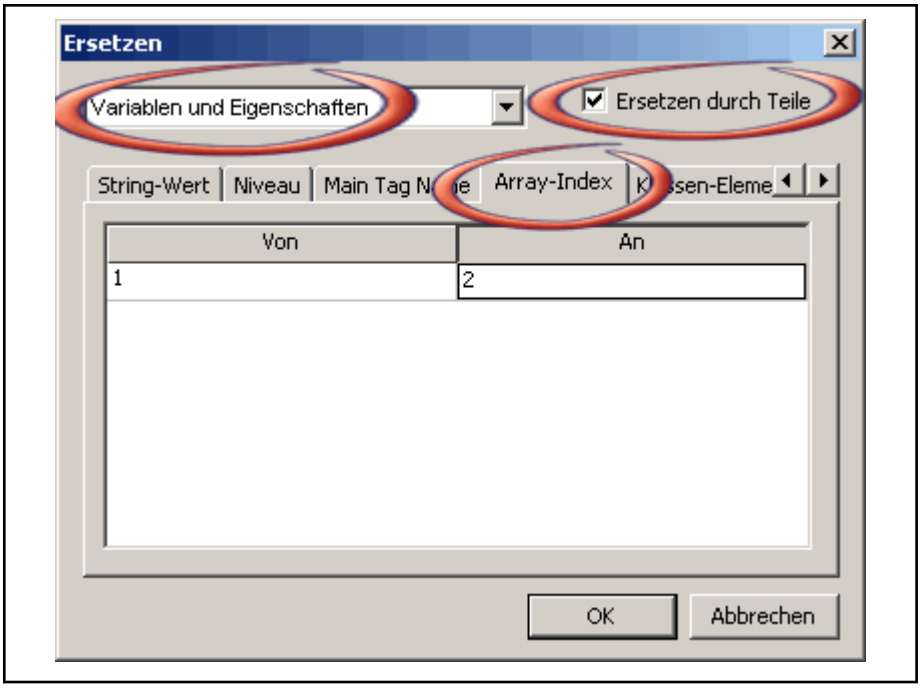


Abb.8-37: Umkonfigurieren des eingefügten Objekts

Fußzeile erweitern

Die Tasten zur Bedienbildumschaltung sind als parametrierbare Grafik-Objekte angelegt (linked Symbol) und verfügen über folgende Übergabe-Parameter:

- Caption -> Beschriftung der Taste
- Disable -> Eingang zum Sperren der Tastenfunktionalität (ausgegraute Beschriftung)
- ScreenName -> Bildname des zu öffnenden Bedienbildes (bzw. Bildgruppe)
- ScreenNumber -> Nummer, die dem Bild zugewiesen werden soll (zum farblichen hervorheben der jeweiligen Taste)

Kommunikationsanbindung

Die Kommunikation zwischen Visualisierung und Steuerung erfolgt symbolisch über den BR-WS Treiber. Hierzu ist es erforderlich die Symboleinträge für die zu visualisierenden SPS-Variablen zu erzeugen (siehe hierzu Kap. "Kommunikations-Konfiguration" auf Seite 53).

Die benötigte Symbolkonfiguration der vordefinierten WinStudio-Visualisierung beinhaltet folgende Einträge:

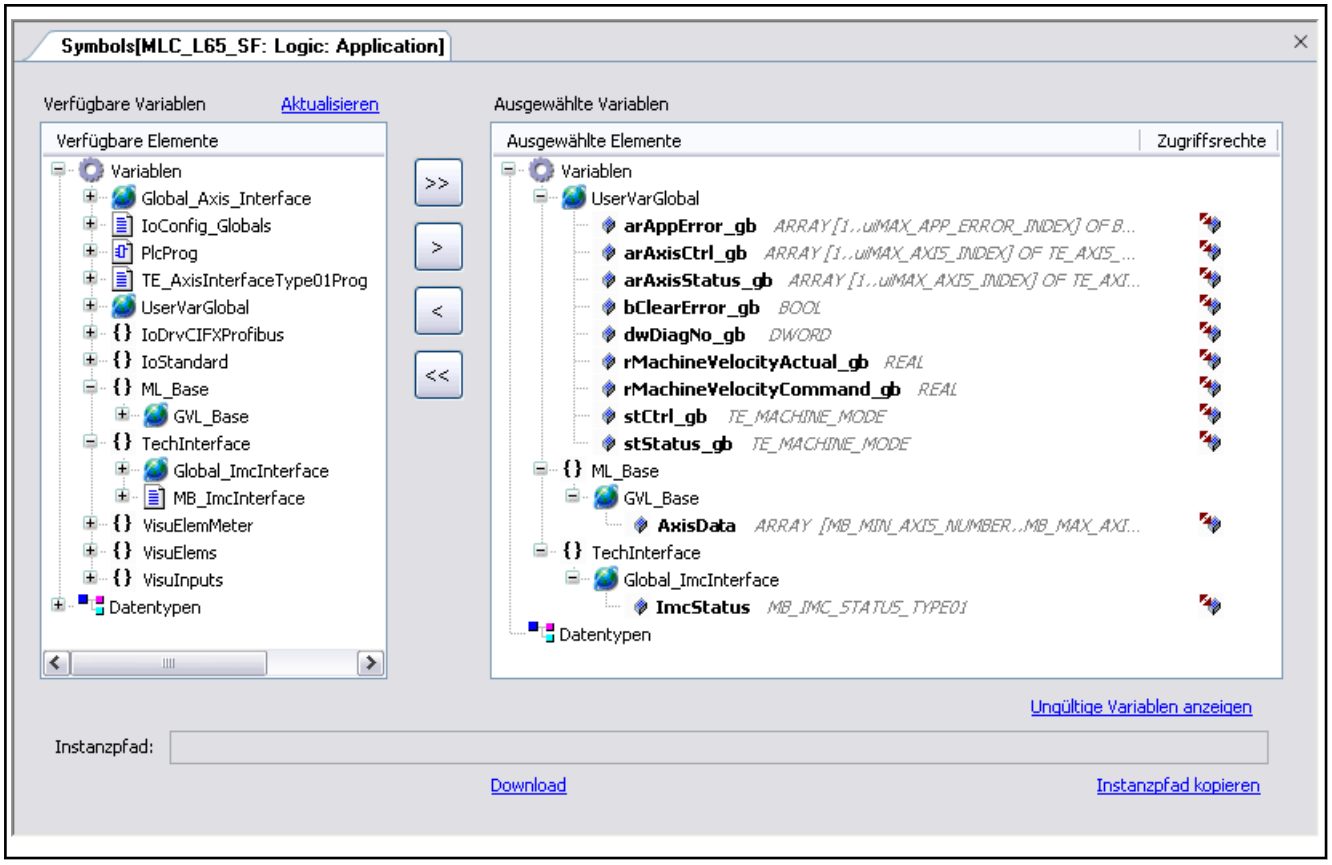


Abb.8-38: Symbolkonfiguration der vordefinierten WinStudio-Visualisierung

Symboleinträge für Variablen aus Globalen Variablenlisten die aus einer Bibliothek stammen, müssen nach einer Versionsänderung der entsprechenden Bibliothek erneut in die Symbolkonfiguration eingefügt werden.

Einstellen der IP-Adresse der Steuerung:

Die Konfiguration der Steuerungsadresse erfolgt im "Eigenschaften"-Dialog des BR-WS Treibers:

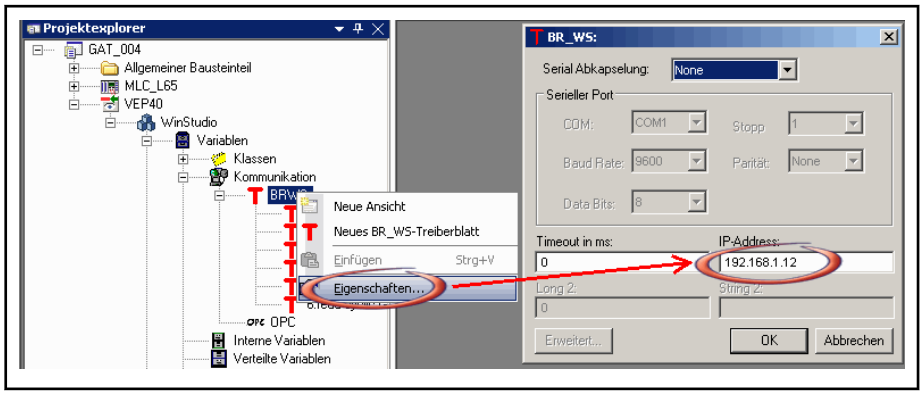


Abb.8-39: Einstellen der IP-Adresse der Steuerung

Um die Kommunikation zu strukturieren, wurden die Variablen in verschiedene Treiberblätter aufgeteilt:

1. **read cyclic fast**: Variablen deren Werte zyklisch in kurzen Intervallen gelesen werden sollen (z. B. Diagnosenummern, Achspositionen, …)

2. **read only once**: Variablen die einmalig nach dem Start der Visualisierung gelesen werden (z. B. Achsnamen)
3. **read after switching phase**: Variablen die nur einmalig nach einer Phasenumschaltung gelesen werden (z. B. ob die Achse aktiviert wurde)
4. **read - write cyclic**: Variablen, deren Werte durch die Visualisierung sowohl gelesen als auch geschrieben werden sollen (z. B. Maschinengeschwindigkeit, Achsgeschwindigkeit für Einrichtbetrieb, …)
5. **write only**: Variablen die durch die Visualisierung nur geschrieben werden sollen (Triggersignal zum Fehlerlöschen, …)

Zum Erweitern eines Treiberblattes kann die offline Symboldatei (*.XML) mit dem BR-WS Browser durchsucht und die entsprechende Variable selektiert werden.

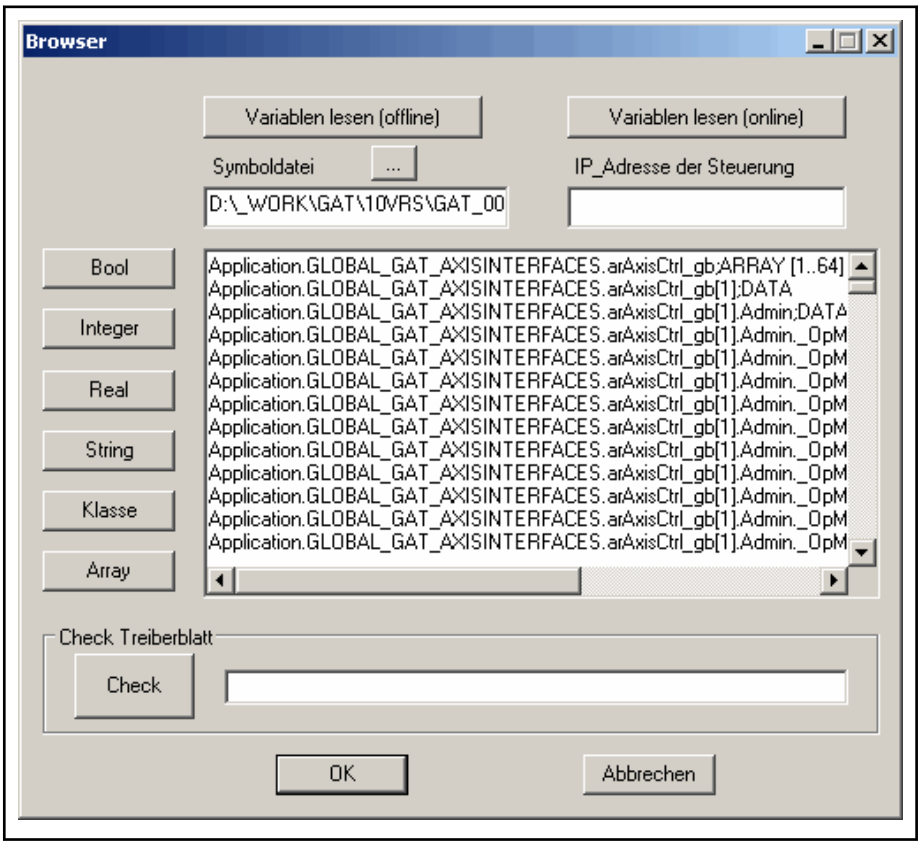


Abb.8-40: Erweitern des Treiberblattes

Mehrsprachigkeit

Die Mehrsprachigkeit der Visualisierung wurde mit der in IndraWorks integrierten Übersetzungsfunktionalität realisiert.

Hierbei sind die Übersetzungsdaten bereits für Steuerungs- und Antriebs-Diagnosen, sowie den verwendeten Texten der Visualisierung vorbereitet. Die Übersetzungsdaten können mit IndraWorks modifiziert bzw. ergänzt werden.

Antriebsdiagnosen sind hierbei in deutsch, englisch, französisch, spanisch und türkisch enthalten.

Alle andere Texte sind in deutsch und englisch enthalten.

8.9 Schnittstellen zum GAT^{compact}

8.9.1 Allgemeines

Die Schnittstellendaten des GAT^{compact} sind weitgehend in der Variablenliste "UserVarGlobal" zusammengefasst. Ein Teil der Daten wird über Bibliotheken bereitgestellt.

Im Folgenden sind die Daten aus Bibliotheken in der Spalte "Name der Variablen" gekennzeichnet. Alle anderen Daten befinden sich in der Variablenliste "UserVarGlobal".

8.9.2 Daten zur Zustandsmaschine

I/O-Typ	Name	Datentyp	Beschreibung
I	stCtrl_gb	TE_MACHINE_MODE_CTRL	Vorgabe für die Betriebsarten der Maschine
I	bSkipColdstartAtStartup_gb	BOOL	Beim Starten des GAT^{compact}-Programms wird eine Umschaltung in den Parametriermodus (P2) zur Durchführung der zur Applikation nötigen Parametrierung ausgeführt. Mit TRUE kann diese Phasenumschaltung übersprungen werden, wenn die Applikation keine Parametrierung in P2 erfordert
O	stStatus_gb	TE_MACHINE_MODE_STATUS	Rückmeldung aktuelle Betriebsart der Maschine

I Eingang zum Beschreiben
O Ausgang zum Lesen

8.9.3 Daten zum Fehlerhandling

I/O-Typ	Name	Datentyp	Beschreibung
I	bClearError_gb	BOOL	TRUE = Globales Fehlerlöschen
I	bDisableErrorHandling_gb	BOOL	TRUE = bei einem Fehler wird nicht in den Zustand "ERROR" gewechselt. Die Fehleranzeige wird weiterhin ausgeführt und auch Fehlerlöschen ist weiterhin möglich
O	bError_gb	BOOL	TRUE = globaler Fehler steht an
O	dwDiagNo_gb	DWORD	Nummer gemäß Dokumentation "Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnosen" mit den folgenden Erweiterungen: DiagNo = 16#F0AE xxxx: Applikations-Fehler (z. B. geöffnete Schutztür) DiagNo = 16#F0FB 0000: FB-Fehler DiagNo = 16#F0AA xxxx: GAT-interner Fehler
O	enErrorID_gb	ERROR_CODE	Detailinformation, wenn DiagNo = 16#F0FB 0000/16#F0AA xxxx
O	stErrorIdent_gb	ERROR_STRUCT	Detailinformation, wenn DiagNo = 16#F0FB 0000/16#F0AA xxxx

GATcompact

I/O-Typ	Name	Datentyp	Beschreibung
O	strDiagMessage_gb	STRING(60)	Diagnosetext der Steuerung/Achse
O	arAppError_gb	ARRAY OF BOOL	Applikationsspezifische Fehler (z. B. Schutztür geöffnet)

I Eingang zum Beschreiben
O Ausgang zum Lesen

8.9.4 Daten der Achsen

I/O-Typ	Name	Datentyp	Beschreibung
I	arAxisCtrl_gb	ARRAY OF TE_AXIS_CONTROL_TYPE01	Vorgaben für die Achsen
O	arAxisStatus_gb	ARRAY OF TE_AXIS_STATUS_TYPE01	Rückmeldungen der Achsen
I/O	AxisData (GVL_Base aus der Bibliothek ML_Base)	ARRAY OF ML_AXISDATA_SM	Firmwarestruktur mit Ein- und Ausgängen der Achsen

I Eingang zum Beschreiben
O Ausgang zum Lesen
I/O Struktur mit Ein- und Ausgängen

☞ Elemente der arAxisCtrl_gb werden im FB "Machine" beschrieben und können deshalb von außen nicht mehr genutzt werden.

8.9.5 Daten der Steuerung

I/O-Typ	Name	Datentyp	Beschreibung
I	ImcCtrl (Global_ImcInterface aus der Bibliothek ML_TechInterface)	MB_IMC_CONTROL_TYPE01	Vorgaben für die Steuerung
O	ImcStatus (Global_ImcInterface aus der Bibliothek ML_TechInterface	MB_IMC_STATUS_TYPE01	Rückmeldungen der Steuerung

I Eingang zum Beschreiben
O Ausgang zum Lesen

☞ Elemente der ImcCtrl_gb werden im FB "Machine" beschrieben und können deshalb von außen nicht mehr genutzt werden.

8.10 HowTo: Typische Anwenderaktivitäten

8.10.1 Zugriff auf Achsdaten

Zum Zugriff auf Achsdaten ist im FB "Machine" als zentraler Zugriffspunkt die Struktur **Axes** definiert. Über diese Struktur sind alle relevanten Daten für eine Achse adressierbar.

Die Struktur ist so aufgebaut, das über die Namen der Achsen auf die Achsdaten zugegriffen wird und damit gut lesbarer Programmcode entsteht.

Die folgenden Daten sind unterhalb von "Axes" verfügbar:

- Axes.Name.AxisData => Achsdatenstruktur: Statusdaten der Firmware und Zugriff auf Achsdaten-Container

- Axes.Name.Ctrl => Steuerstruktur des Achs-Interface
- Axes.Name.Status => Statusstruktur des Achs-Interface
- Axes.Name.DV => Zugriff auf Direktvariablen (Funktionsvariablen)
- Axes.Name.AxisRef => Achsreferenzstruktur für FB-Aufrufe (z. B. MB_ReadParameter)

Auf die vorher genannten Daten kann ebenfalls über das Array Axes.arAll[] zugegriffen werden. Das ist vorteilhaft für Schleifenprogrammierung (Beispiel siehe "actModeOff_entry").

Es wird bei beiden Varianten über Referenzen auf die gleichen Speicherstellen zugegriffen.

Beispiel: Folgende Zugriffe sind absolut gleichwertig und liefern das gleiche Ergebnis:

- Axes.Name.AxisData.rActualPosition_i
- Axes.arAll[Name.AxisNo].AxisData.rActualPosition_i

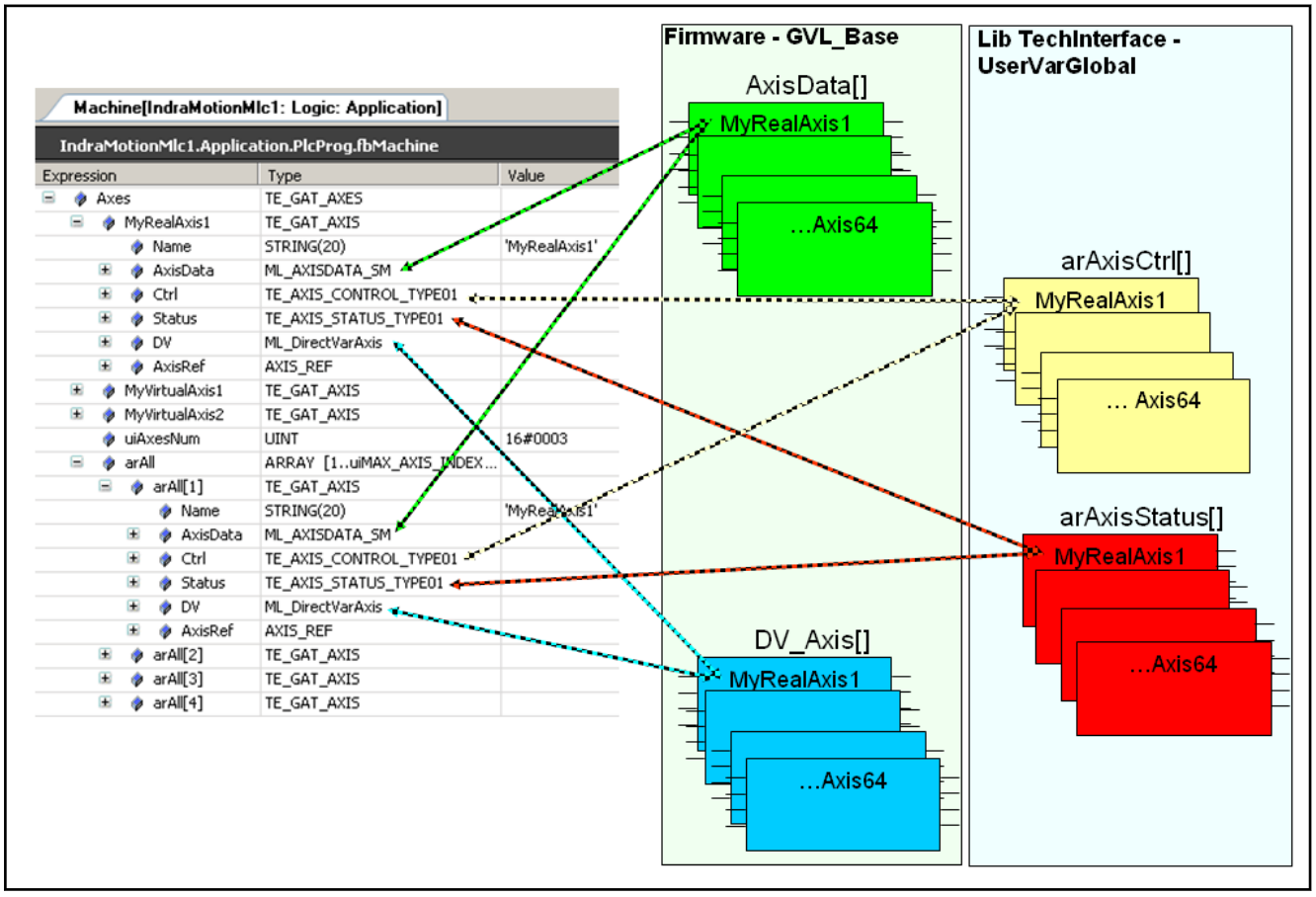


Abb.8-41: Achsdaten-Übersicht

8.10.2 Achse hinzufügen

Zum Hinzufügen einer Achse sind folgende Schritte durchzuführen:

- Einfügen der Achse in das IndraWorks-Projekt (auf den sercos Knoten ziehen…)
- Bei Nutzung der Visualisierung "SystemOverview_All_Axis": einen Frame für die neue Achse einfügen und in den Eigenschaften die Referenz auf die neue Achse setzen (siehe auch Kap. "System_Overview" auf Seite 232)

- Über den GAT-Wizard die Achse hinzufügen

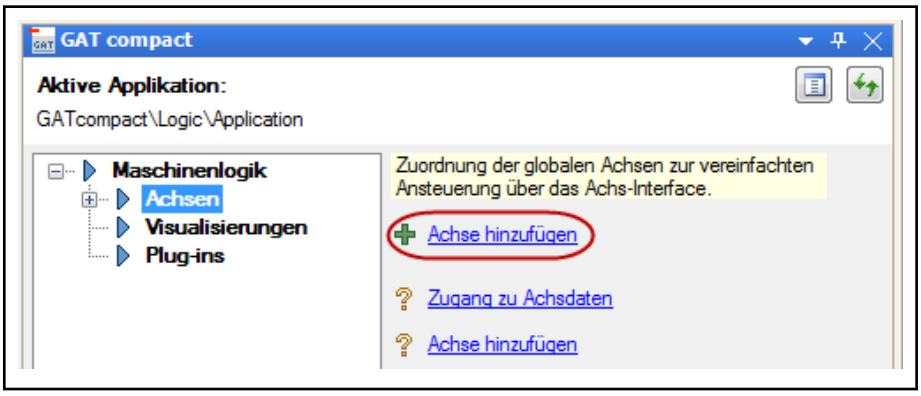


Abb.8-42: Achse über den GAT-Wizard hinzufügen

8.10.3 Erhöhung der maximalen Achsanzahl für Achsnummern größer 16

Sind Achsen in Ihrem Projekt enthalten deren logische Achsnummer größer '16' ist, so müssen Sie den Wert der globalen Konstanten "uiMAX_AXIS_INDEX" entsprechend anpassen.

"uiMAX_AXIS_INDEX" muss mindestens so groß sein wie die größte logische Achsnummer des Projektes.

Die Standardeinstellung ist "uiMAX_AXIS_INDEX := 16"

Die Konstante "uiMAX_AXIS_INDEX" finden Sie am schnellsten über die IndraWorks Such-Funktion (<STRG>+<F>) oder über folgende Ordnerstruktur:

Application/UserConstGlobal

8.10.4 Achse entfernen/umbenennen

☞ Der Wizard ändert nur die Namen in der Achs-Deklaration. Wird die Achse bereits in der Implementation bzw. internen Visualisierung verwendet, so muss der neue Achsname durch "Suchen/Ersetzen" an diesen Stellen angepasst werden.

Zum Entfernen/Umbenennen einer Achse sind folgende Schritte durchzuführen:

- Löschen/Umbenennen der Achse aus dem IndraWorks-Projekt
- Bei Nutzung der Visualisierung "SystemOverview_All_Axis": den Frame für die gelöschte Achse entfernen bzw. in den Eigenschaften die Referenz auf die Achse ändern
- Über den GAT-Wizard die Achse entfernen/umbenennen (Umbenennen durch "Eigenschaften", siehe nachfolgende Abbildung)

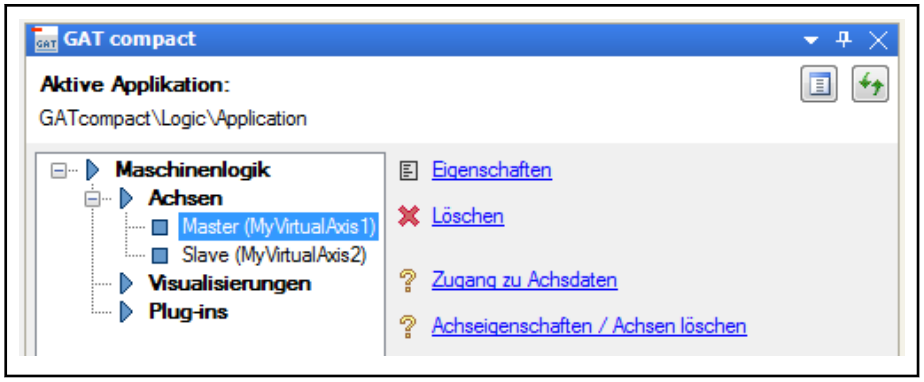


Abb.8-43: Achse mit dem GAT-Wizard entfernen/umbenennen

8.10.5 Betriebsart hinzufügen

GAT^{compact} bringt vordefiniert die folgenden Betriebsarten mit:

- OFF
- MANUAL
- AUTO

Im Folgenden soll beispielhaft zusätzlich die Betriebsart **"CLEANING"** (Reinigung der Maschine) implementiert werden.

Dazu sind folgende Schritte notwendig:

- TYPE "TE_MACHINE_MODE" (im Ordner GAT^{compact}): erweitern um das Element **MODE_CLEANING**
- *FB "Machine":*
  1. Die Transition "traAuto" kopieren, in "traCleaning" umbenennen, und in "traCleaning" MODE_AUTO durch MODE_CLEANING ersetzen, Ergebnis: (Ctrl= MODE_CLEANING;)
  2. Die Transition "traLeaveAuto" kopieren, in "traLeaveCleaning" umbenennen, und in "traLeaveCleaning" MODE_AUTO durch MODE_CLEANING ersetzen, Ergebnis: NOT(Ctrl= MODE_CLEANING;)
  3. Parallel zum "ModeAuto" einen Zweig "ModeCleaning" einfügen
  4. Am "ModeCleaning" die oben genannten Transitionen verwenden
  5. Am "ModeCleaning" eine Eingangsaktion "actModeCleaning_entry()" hinzufügen. In der Aktion als erste Zeile den folgenden Code einfügen: Status:= MODE_CLEANING;
  6. Nach den Transitionen die gleichen Sprungziele wie bei "ModeAuto" einfügen

GATcompact

Ergebnis der Erweiterung:

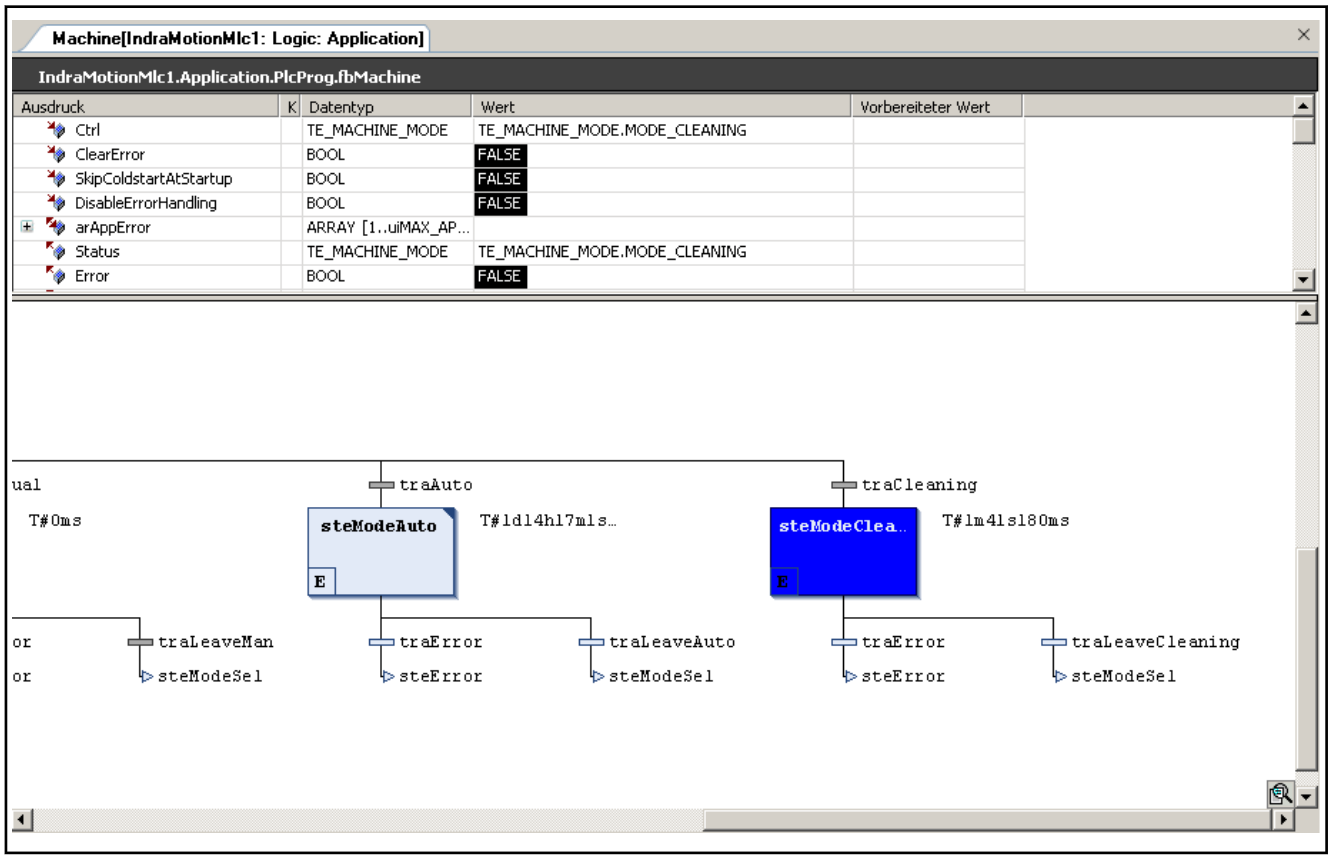


Abb.8-44: Ergebnis nach Betriebsart hinzufügen

In der Aktion "actModeCleaning_entry()" bzw. in einer optionalen Aktion "actModeCleaning_active()" kann das gewünschte Verhalten implementiert werden.

8.10.6 Abarbeitung in MotionTask

Für die Abarbeitung in der MotionTask kann der FB "Machine" um die Methode **"mMotionCyclic"** erweitert werden. Aus dieser Methode kann auf alle Variablen des FB (z. B. zentraler Achseinstiegspunkt Axes….) zugegriffen werden.

Wichtig ist dabei, dass auch die hier angesteuerten Achsen in der MotionTask aktualisiert werden (siehe Kap. 8.6.2 "Hinzufügen einer MotionTask" auf Seite 224).

8.10.7 Achs-Interface Erweiterungen

Das GAT^{compact} erlaubt fast beliebige Erweiterungen der Achs-Interface-Strukturen. Es können zusätzliche Unterstrukturen eingefügt und auch die vorhandenen Unterstrukturen erweitert werden.

Die folgenden Strukturelemente sind als Anwendererweiterungen in der AxisCtrl-Struktur beispielhaft programmiert:

- SetupMode: zusätzliche Unterstruktur in der AxisCtrl-Struktur
  - Enable: Freigabe Einrichtbetrieb
  - JogPlus: Tippen +
  - JogMinus: Tippen -
  - Vel: Tipp-/Referenzfahrgeschwindigkeit

  - Accel: Tipp/Referenzfahr(brems)beschleunigung
  - Homing: aktiviert ModeHoming|SetAbsRef|ModesPosAbs abhängig vom Achstyp/-konfiguration
- SyncMode: Erweiterung der vorhandenen Unterstruktur in der AxisCtrl-Struktur
  - AngleOffset: Winkelverschiebung Tabellenanfang -> der Parameter A-0-2730 wird mit diesem Wert beschrieben
  - PhaseShift: Offset zur Leitachse vor dem Getriebe (PhaseOffset: Offset zur Leitachse nach dem Getriebe)
  - PhaseShiftVel: Geschwindigkeit zur Verstellung des PhaseShift
  - PhaseShiftAcc: Beschleunigung zur Verstellung des PhaseShift
  - SyncVelocity: Aufsynchronisiergeschwindigkeit -> der Parameter A-0-2790 wird mit diesem Wert beschrieben
  - SyncAcceleration: Aufsynchronisierbeschleunigung -> der Parameter A-0-2791 wird mit diesem Wert beschrieben

Die folgenden Strukturelemente sind als Anwendererweiterungen in der Axis-Status-Struktur beispielhaft programmiert:

- SetupMode: zusätzliche Unterstruktur in der AxisStatus-Struktur
  - EnableAck: Einrichtbetrieb ist aktiv
  - Homingack: Homing ist abgeschlossen
- SyncMode: Erweiterung der vorhandenen Unterstruktur in der AxisStatus-Struktur
  - AngleOffsetDone: Übertragung Winkelverschiebung Tabellenanfang ist abgeschlossen -> der Parameter A-0-2730 wurde beschrieben
  - PhasingMasterDone: der PhaseShift wurde verfahren
  - SyncVelocityDone: Übertragung Aufsynchronisiergeschwindigkeit ist abgeschlossen -> der Parameter A-0-2790 wurde übertragen
  - SyncAccelerationDone: Übertragung Aufsynchronisierbeschleunigung ist abgeschlossen -> der Parameter A-0-2791 wurde übertragen

Der Code zu diesen Erweiterungen ist in den Aktionen des Bausteins TE_AxisInterface() im Ordner "GAT^{compact}/AxisInterface" zu finden. Die dazugehörigen Strukturen sind in den Ordnern " GAT^{compact}/AxisInterface/Type/Control" und "GAT^{compact}/AxisInterface/Type/Status" zu finden.

Es können eigene Erweiterungen hinzugefügt werden (siehe dazu Kap. 6.7 "Achs-Interface Anwender-Erweiterung" auf Seite 103).

Bei der Einführung neuer Fehlercodes kann durch Erweiterung der Struktur "arAXIS_DIAG_MESSAGES" in der Variablenliste GatConstGLobal im Ordner "\GAT compact\GAT-Utilities" der Fehlernummer auch eine Diagnosemeldung für die SPS-Visualisierung zugeordnet werden.

8.10.8 Robot Control

Um Achsen für koordinierte Bewegungen ansteuern zu können, muss für das TechInterface der Funktionsbaustein MB_AxisInterfaceType**02** verwendet werden. Hierzu muss folgendes im GAT^{compact} angepasst werden:

- Die Struktur "TE_AXIS_CONTROL_Type01" muss in "TE_AXIS_CONTROL_TYPE02" umbenannt werden und um die Strukturvariable "CoordModeCtrl" vom Typ "MB_AXIS_COORDINATED" erweitert werden

GATcompact

Umbenennen der Struktur TE_AXIS_CONTROL_Type01 nach TE_AXIS_CONTROL_Type02:

```
//former type: TYPE TE_AXIS_CONTROL_TYPE01:
TYPE TE_AXIS_CONTROL_TYPE02:  // rename for RobotControl
STRUCT
    Admin:      TE_AXIS_ADMINISTRATION; (* Administration of the axis | Verwaltung der Achse *)

    PosMode:    TE_AXIS_POSITIONING; (* operating mode positioning (absolute, rel. and additive)
      Positionierbetrieb (absolut, relativ und additiv) *)

    VelMode:    TE_AXIS_VELOCITY; (* operating mode velocity control | Geschwindigkeitsregelung *)

    SyncMode:   TE_AXIS_SYNCHRONISATION; (* operating mode synchronisation (phase, vel, and cam)
      Synchronbetriebsarten (Winkelsync., Geschwindigkeitssync. und Kurvenscheibe) *)

    (* add your extensions below here, keep the elements above
       Erweiterungen hier einfuegen, Elemente oberhalb nicht veraendern *)

    (* user extensions | Anwender Erweiterungen *)

      // Jog/Homing functionality | Tipp/Einrichtfunktionen
    SetupMode:  TE_AXIS_SETUP_MODE;

    CoordModeCtrl:  MB_AXIS_COORDINATED;    // inserted for RobotControl

END_STRUCT
END_TYPE
```

- Umbenennen der "TE_AXIS_STATUS_Type01" Struktur in "TE_AXIS_STATUS_TYPE02"

Umbenennen der Struktur TE_AXIS_STATUS_Type01 nach TE_AXIS_STATUS_TYPE02:

```
    (********* Outputs of the following axis - from motion control to logic control **********)
    // former type: TYPE TE_AXIS_STATUS_TYPE01:
TYPE TE_AXIS_STATUS_TYPE02:     // Rename for RobotControl
  STRUCT
    (********* Outputs of the following axis - from motion control to logic control **********)

    PtrAxisData:  POINTER TO ML_AXISDATA_SM;
     (* Pointer to AxisData of this axis | Zeiger auf AxisData dieser Achse
      - Type .PtrAxisData^. for the intellisense menu | Eingabe .PtrAxisData^. fuer Auswahlmenue*)

    Admin:   TE_AXIS_ADMIN_STATUS; (* status of the axis | Achs-Status *)

    Diag:    TE_AXIS_DIAGNOSIS; (* axis diagnosis information, error, warning....
      Diagnose Informationen der Achse, Fehler, Warnung.... *)

    PosMode: TE_AXIS_POS_STATUS; (* Status operating mode positioning (absolute, rel. and add.)
      Statusmeldungen Positionierbetrieb (absolut, relativ und additiv) *)

    VelMode: TE_AXIS_VEL_STATUS; (* Status operating mode velocity control
      Statusmeldungen Geschwindigkeitsregelung (Freilauf) *)

    SyncMode: TE_AXIS_SYNC_STATUS; (* Status operating mode synchronisation (phase, vel, and cam)
      Statusmeldungen Synchronbetriebsarten (Winkelsync., Geschwindigkeitssync. und Kurvensch.)*)

    (* add your extensions below here, keep the elements above
       Erweiterungen hier einfuegen, Elemente oberhalb nicht veraendern *)
    (* user extensions | Anwender Erweiterungen *)
          (* Jog/Homing functionality | Tipp/Einrichtfunktionen *)
    SetupMode:  TE_AXIS_SETUP_MODE_STATUS;

    // AxisData for HMI (without data type "BIT")
      AxisData_HMI:    TE_AXIS_DATA_STATUS;

  END_STRUCT
END_TYPE
```

- Der Funktionsbaustein "TE_Axisinterface" muss mit dem neuen FB "MB_AxisInterfaceType02" erweitert werden. "TE_AxisInterface EXTENDS MB_AxisInterfaceType02"

GATcompact

Erweiterung TE_Axisinterface:

```
//former: FUNCTION_BLOCK TE_AxisInterface EXTENDS MB_AxisInterfaceType01
FUNCTION_BLOCK TE_AxisInterface EXTENDS MB_AxisInterfaceType02
(*##############################################################################################
  ##############################################################################################

  General Header
  ----------------------------------------------------------------------------------------------
  Shortdescription       : AxisInterface Example
```

- Die Methode "mInitExtensions()": Der zusätzliche Eingang "CoordModeCtrl" des FB "MB_AxisInterfaceType02" muss initialisiert werden

CoordModeCtrl initialisieren:

```
SUPER^.mInitExtension(index);

IF NOT bFirstInit THEN
     // User: Assigment of user defined axis interface inputs
     SetupMode      REF= arAxisCtrl_gb[index].SetupMode;
     // inserted for RobotControl
     CoordModeCtrl  REF= arAxisCtrl_gb[index].CoordModeCtrl;
     // User: Assigment of user defined axis interface outputs
     SetupModeAck REF= arAxisStatus_gb[index].SetupMode;
     AxisData_HMI REF= arAxisStatus_gb[index].AxisData_HMI;

     // ================================================================
     // Do not touch the following lines
     // ================================================================

     // Set the references for the AxisInterface FB
     AdminCtrlExt  REF= arAxisCtrl_gb[index].Admin;
```

- Die Deklaration für die beiden Strukturvariablen "arAxisCtrl_gb" und "arAxisStatus_gb" muss auf die entsprechenden Type02-Strukturen geändert werden

Strukturvariablen anpassen:

```
VAR_GLOBAL
   // Axis interface: command structure: define as RETAIN, to keep the values
   // Note: on the L40.2 control, the RETAIN memory is slower then normal VAR, so define
   //       this structure only as RETAIN when necessary
   //  arAxisCtrl_gb: ARRAY [1..MAX_AXIS_NUM] OF MB_AXIS_CONTROL;

   // Axis interface: command structure
   //former: arAxisCtrl_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_CONTROL_TYPE01;
     arAxisCtrl_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_CONTROL_TYPE02;
END_VAR

VAR_GLOBAL
//---------------------------------- Machine status variables ----------------------------

  // Axis interface: status structure
  //former: arAxisStatus_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_STATUS_TYPE01;
  arAxisStatus_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_STATUS_TYPE02;
  bRemoteOn_gb  : BOOL := TRUE;  // TRUE: Internal IL visu is active
                                 // FALSE: PlcProg is active
END_VAR
```



Die Strukturen TE_AXIS_CONTROL_TYPE02, TE_AXIS_STATUS_TYPE02 befinden sich im Ordner: "AxisInterface\Type".

Die Strukturvariablen arAxisCtrl_gb, arAxisStatus_gb befinden sich in der globalen Variablenliste "UserVarGlobal".

Kinematik-Interface

Aufbauend auf das Achs-Interface steht das Kinematik-Interface als optionales Plug-in zur Verfügung. Über das Kinematik Interface können Kinematiken analog zu Achsen kommandiert werden. Fügen Sie hierzu das Kinematik Interface über den GAT Wizard als Plug-in hinzu.

Die Dokumentation des Kinematik Interface ist in dem Plug-in Konfigurationsordner enthalten.

GATcompact

Den Link zum Konfigurationsordner können Sie über den GAT Wizard wie folgt finden:

- Im GAT Wizard den Ordner "Plug-ins" selektieren
- Im Wizard Bedienfeld ist jetzt der Link "Plug-ins Konfigurationsordner öffnen" sichtbar

8.11 GAT^{compact} Kurzübersicht

8.11.1 Strukturübersicht

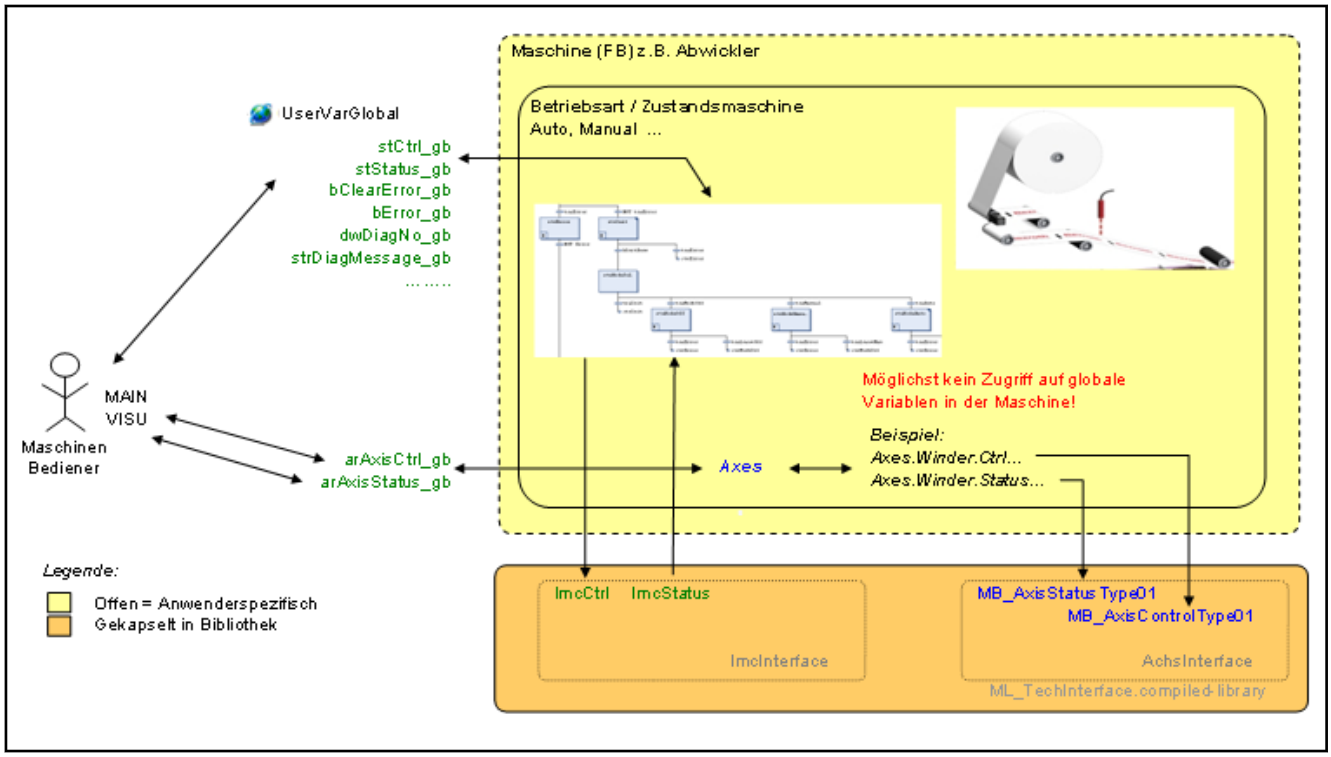


Abb.8-45: GAT^{compact} Strukturübersicht

8.11.2 Übersicht der wichtigsten globalen Variablen der Maschine

Siehe globale Variablenliste **„UserVarGlobal"**:

Variable	Beschreibung	Zugriff
stCtrl_gb	Steuerung der Maschine (Betriebsartenanwahl, Standard-Betriebs-art)	RD/WR
stStatus_gb	Status der Maschine (aktuelle Betriebsart)	RD
bClearError_gb	Fehlerlöschen	RD/WR
bDisableErrorHandling_gb	Bei Fehler nicht in den Fehlerzustand wechseln	RD/WR
bSkipColdstartAtStartup_gb	Beim Programmstart wird der "ColdStart" (Umschaltung in P2) nicht ausgeführt	RD/WR
bError_gb	TRUE zeigt einen Fehler an	RD
dwDiagNo_gb	Diagnosenummer entsprechend der IndraLogic XLC-/IndraMotion MLC-Dokumentation	RD

Variable	Beschreibung	Zugriff
enErrorID_gb	Detailinformationen wenn dwDiagNo_gb = 16#F0FB 0000 (Funktionsbausteinfehler)	RD
stErrorIdent_gb	Detailinformationen wenn dwDiagNo_gb = 16#F0FB 0000 (Funktionsbausteinfehler)	RD
strDiagMessage_gb	Steuerungs- bzw. Achsdiagnosetext	RD
arAppError_gb	Array of BOOL mit anwenderspezifischen Fehlerflags der Maschine	RD

RD Lesen
WR Schreiben

8.11.3 Übersicht der wichtigsten globalen Variablen der Achsen

Siehe globale Variablenliste **„UserVarGlobal”**:

Variable	Beschreibung	Zugriff
arAxisCtrl_gb	Array mit den Achs-Interface Kommando-Daten	RD/WR
arAxisStatus_gb	Array mit den Achs-Interface Status-Daten	RD

RD Lesen
WR Schreiben

8.11.4 Übersicht der Methoden und Aktionen der Maschine

Aktion/Methode	Aufruf	Zugehörige Task	Beschreibung/Beispiel
actCyclic_active	In jedem Zustand	PlcTask	Übergreifende Logik, z. B. Sicherheitstechnik
actError_entry	Für einen Zyklus beim Übergang in den Zustand MODE_ERROR	PlcTask	Fehlerreaktion, z. B. Umschalten in ModeAH (Stillstand mit Leistung)
actInit_entry	Für einen Zyklus beim Übergang in den Zustand MODE_INIT	PlcTask	Vorbereiten des Zustands MODE_INIT, z. B. Abschalten aller Achsen um die Phasenumschaltung in P2 zu ermöglichen
actInit_active	Nur im Zustand MODE_INIT	PlcTask	Umschaltung in den Parametriermodus P2, mögliches Beschreiben von Parametern in P2. Beim Verlassen des MODE_INIT Umschaltung in den Betriebsmodus (bb), mögliches Beschreiben von Parametern beim Erreichen von bb
actModeOff_entry	Für einen Zyklus beim Übergang in den Zustand MODE_OFF	PlcTask	Vorbereiten des Zustands MODE_OFF, typischerweise Abschalten aller Achsen (ModeAb)
actModeManual_entry	Für einen Zyklus beim Übergang in den Zustand MODE_MANUAL	PlcTask	Vorbereiten des Zustands MODE_MANUAL, typischerweise Anhalten aller Achsen (ModeAH)
actModeAuto_entry	Für einen Zyklus beim Übergang in den Zustand MODE_AUTO	PlcTask	Vorbereiten des Zustands MODE_AUTO, z. B. Aktivierung einer Synchronbetriebsart (z. B. ModeSyncVel) für alle Folgeachsen und Freigabe einer Leitachse (z. B. ModeVel)

GATcompact

Aktion/Methode	Aufruf	Zugehörige Task	Beschreibung/Beispiel
actModeAuto_active	Nur im Zustand MODE_AUTO	PlcTask	z. B. Überwachungen im Automatik-Mode, Aufruf von Prozessregler Funktionsbausteinen
FB_init	Einmalig vor SPS-Programmstart/Reset	keine bestimmte Taskzuordnung	Systemintern, darf nicht modifiziert werden

8.11.5 Zugriff auf Achsen innerhalb der Maschine

☞ Zugriff immer über **Axes.<Achsname>**.

Beispiel: Zugriff auf Achsen

```
 // Achse in Geschwindigkeitsregelung kommandieren
Axes.MyAxis1.Ctrl.Admin._OpMode.en := ModeVel;

// Wenn Achse die Geschwindigkeit erreicht hat
IF (Axes.MyAxis1.Status.PtrAxisData^.Axis_InVelocity = TRUE) THEN;
```

9 GAT^{compact} für IndraLogic XLC ohne sercos Interface

9.1 Überblick GAT^{compact}

Allgemeines

Das Generic Application Template (GAT^{compact} für IndraLogic XLC ohne sercos Interface ist ein offenes Beispielprojekt, welches auf IEC 61131-3 basiert. Es wird mit dem IndraWorks Engineering Framework zur Verfügung gestellt, und lässt sich Wizard-unterstützt laden und in Betrieb nehmen.

Das GAT^{compact} für IndraLogic XLC ohne sercos Interface enthält folgende Teile:

- Machine FB mit Betriebsarten-Schrittkette
- Fehlerbehandlung
- Inbetriebnahme Visualisierung
- Vorbereitete Maschinen-Visualisierung (HMI)

9.2 Erste Schritte GAT^{compact}

Erstimport

Mit folgenden Schritten fügen Sie das GAT^{compact} in Ihr Steuerungsprojekt hinzu:

- IndraLogic XLC Steuerung zu Projekt hinzufügen
- GAT-Button  in der Werkzeugleiste anklicken
- GAT^{compact} auswählen und **Fertigstellen** anklicken. Der Erstimport wird ausgeführt
- SPS Applikation einloggen (falls Steuerungshardware verfügbar)
- SPS Applikation starten (falls Steuerungshardware verfügbar)
- Visualisierung "System_Overview_header" öffnen. Hiermit können Sie Betriebsarten anwählen, Fehler löschen und Steuerungsdiagnosen anzeigen
- SPS Ausloggen, und - wie nachfolgend beschrieben - Applikationscode hinzufügen

Applikationscode-Implementierung

Nach dem Erstimport steht Ihnen der fertig vorbereiteter GAT^{compact} Programmrahmen zur Verfügung.

In dem Funktionsbaustein **Machine** (FB) können Sie jetzt Ihren Programmcode an die vorbereiteten Stellen eintragen, und die Ein- und Ausgänge des FB erweitern.

- Schritt **steCyclic**: Fügen Sie hier den Applikationscode ein, der permanent zyklisch bearbeitet wird
- Schritte **steModeOff**, **steModeManual** und **steModeAuto**: Fügen Sie hier den Applikationscode ein, der nur bei der angewählten Betriebsart zyklisch bearbeitet wird
- Schritt **steError**: Fügen Sie hier den Applikationscode ein, der im Fehlerfall bearbeitet wird
- Schritt **steInit**: Fügen Sie hier den Applikationscode ein, der bei der Initialisierung bearbeitet wird
- In der Deklaration des Machine (FB) können Sie weitere Eingänge, Ausgänge und lokale Variablen hinzufügen

GATcompact für IndraLogic XLC ohne sercos Interface

9.3 Initialisierung

Nach Einschalten der Steuerung wird die Initialisierung durchlaufen. Hierbei wird der Schritt **steInit** aktiv und der zugehörige Code bearbeitet. Die Initialisierung ist in 2 Schritte unterteilt:

- Coldstart (Steuerung in Parametriermodus)
- Warmstart (Steuerung ist Betriebsbereit bb)

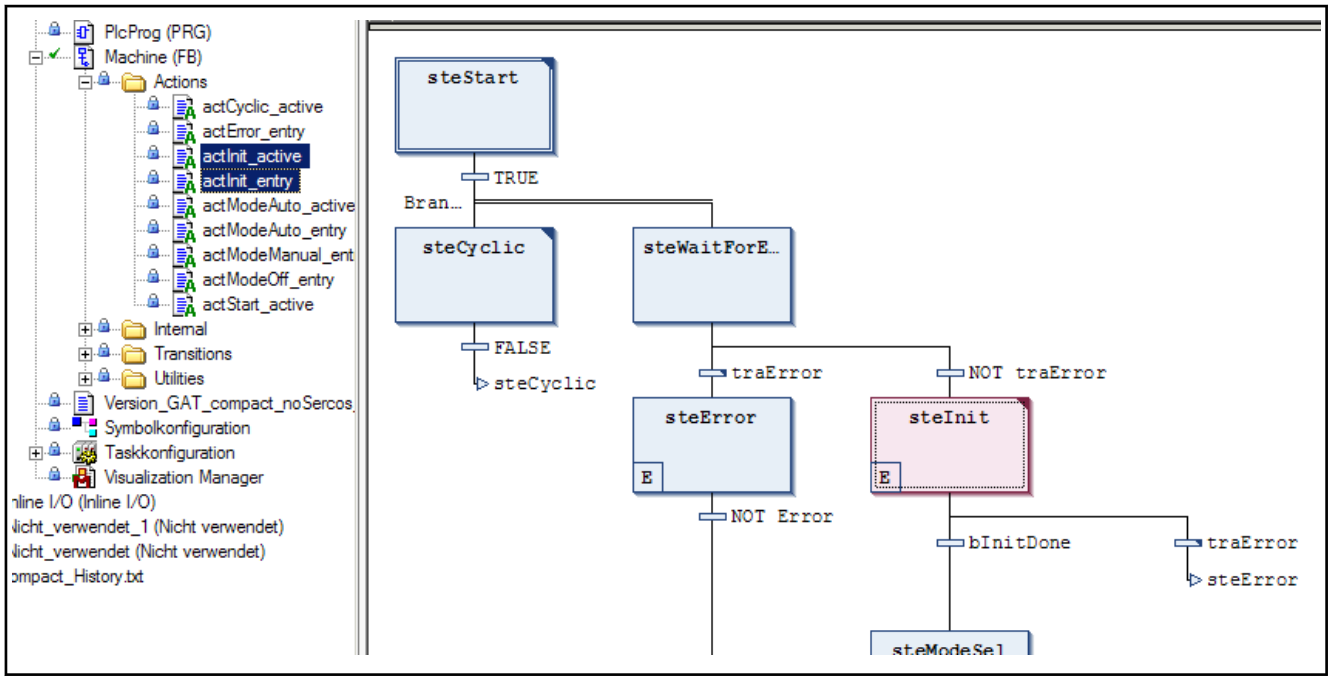


Abb.9-1: Init-Schritt im Machine SFC

Sofern kein Fehler ansteht, wird die Initialisierung im Schritt "steInit" durchgeführt. Zu Beginn des Schrittes wird die Eingangsaktion "steInit_entry" einmalig abgearbeitet.

Im Falle eines Fehlers während der Initialisierung wird in den Fehlerzustand gewechselt. Nach dem Löschen des Fehlers wird die Initialisierung an der Stelle fortgesetzt, an der diese unterbrochen wurde.

Wenn das Bit "bInitDone" gesetzt ist, wird die Initialisierung verlassen und mit dem nächsten Schritt fortgesetzt.

Im Schritt "actInit_active" ist folgender Ablauf umgesetzt:

1. Die Steuerung wird in den Parametermodus geschaltet (Zeile 12).
2. Nach Erreichen des Parametermodus ist der Zustand "Coldstart" erreicht. In diesem Zustand können Sie Parameter beschreiben welche im Betriebsmodus geschützt sind (Zeile 20 bis 25).
3. Der Zustand "Coldstart" bleibt solange aktiv, bis eine Betriebsart <> "Init" vorgegeben wird (Zeile 26). Ergänzen Sie die Weiterschaltung in der IF-Abfrage, falls das Schreiben der Parameter mehrere Zyklen erfordert.
4. Die Steuerung wird in den Betriebsmodus (bb) geschaltet (Zeile 33).
5. Der Betriebsmodus wurde von der Steuerung quittiert. Initialisieren Sie Variablen oder Parameter (Zeile 36 bis 39). Die Initialisierung wird mit dem setzen von "bInitDone" auf "TRUE" verlassen und die angewählte Betriebsart wird aktiv.

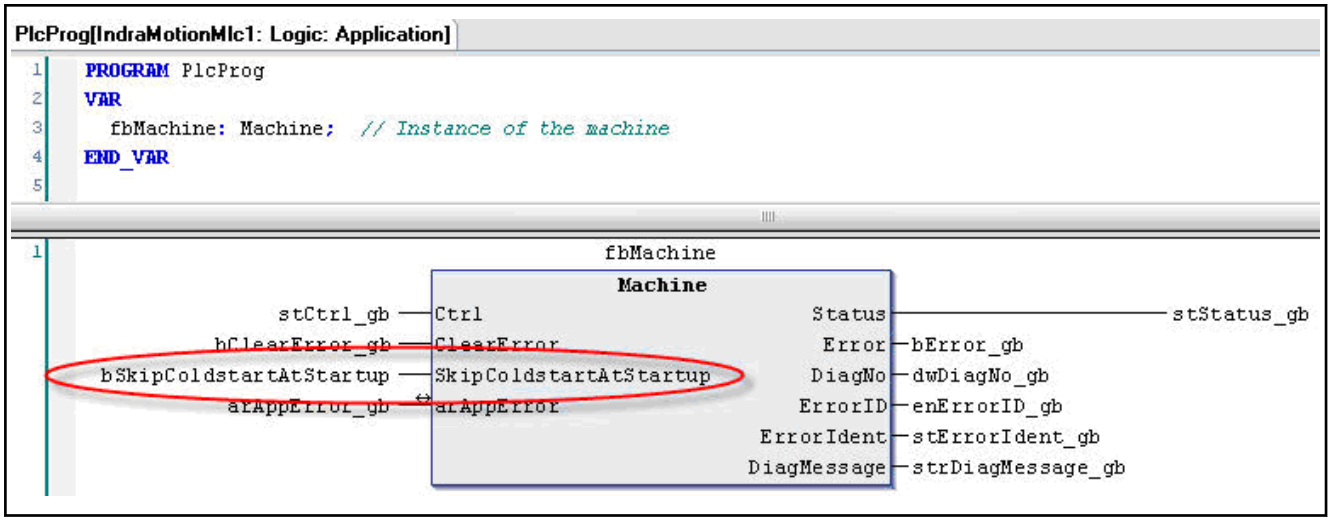


Abb.9-2: Eingang "SkipColdStartAtStartup" am FB "Machine"

Soll die Initialisierung beim erstmaligen Starten der SPS nicht durchlaufen werden, lässt sich dies über das Bit "bSkipColdstartAtStartup" einstellen. Es wird also in den Zustand "Warmstart" gesprungen, "Coldstart" wird nicht abgearbeitet. Diese Einstellung spart Zeit beim Hochlauf. Jedoch sollte sichergestellt sein, dass die Aktionen, die im Zustand "Coldstart" abgearbeitet werden (z. B. Parameter schreiben), nicht nötig sind.

9.4 Betriebsarten

Nach erfolgreicher Initialisierung verzweigt der Ablauf in eine der 3 Betriebsarten (siehe nachfolgende Abbildung):

- ModeOff
- ModeManual (Maschine einrichten)
- ModeAuto (zyklischer Automatikbetrieb)

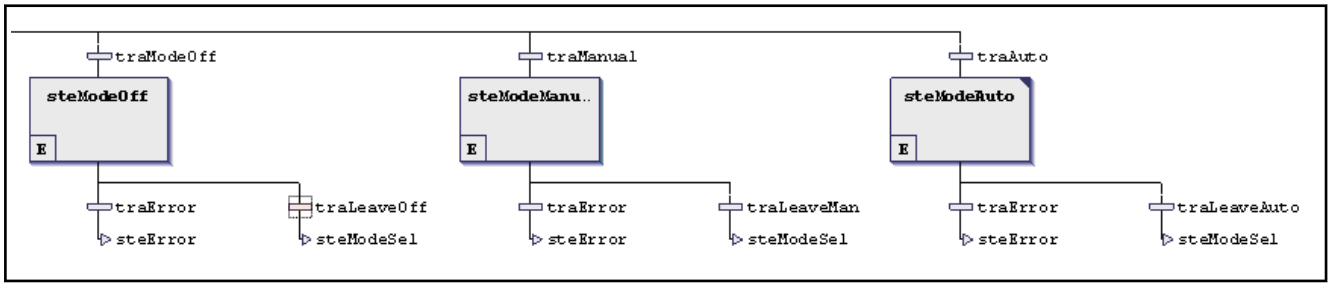


Abb.9-3: Betriebsarten

(siehe Kap. 9.3 "Initialisierung" auf Seite 256)

GATcompact für IndraLogic XLC ohne sercos Interface

Im GAT^{compact} ist als Zielbetriebsart nach erstmaligem Start der Maschine "ModeManual" vorinitialisiert:

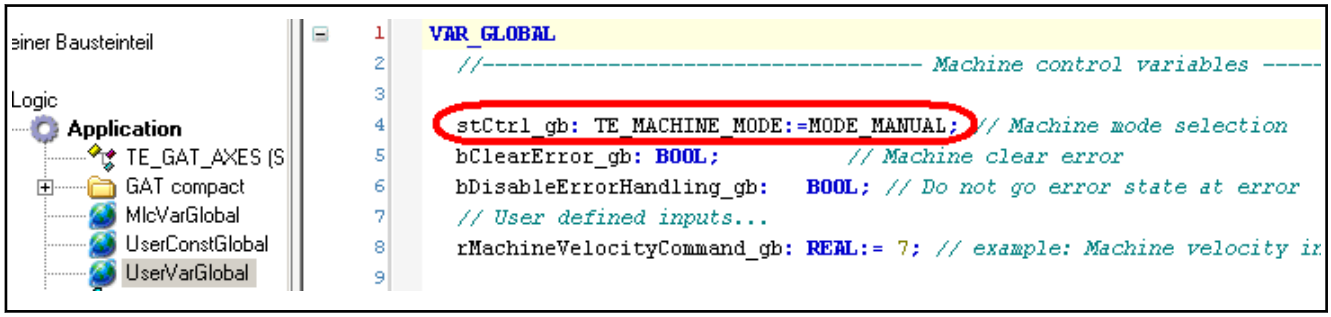


Abb.9-4: Einstellung der Zielbetriebsart nach erstmaligem Start der Maschine

Entry-/Active-Aktion

Mit der Aktivierung einer Betriebsart wird einmalig eine spezielle "entry"-Aktion ausgeführt. Die nachfolgende Tabelle enthält dazu eine Übersicht, wie jede Betriebsart aktiviert und was in der Betriebsart bearbeitet wird.

	Bedingung zur Aktivierung	**Einmalige ausgeführte Aktion (zu Beginn)**	**Zyklisch ausgeführte Aktion**
ModeOff	Ctrl= MODE_OFF Abfrage erfolgt in **traModeOff** bzw. **traLeaveOff**	**actModeOff_entry** • Setzt den Status auf "MODE_OFF"	Keine zyklische Aktion vordefiniert
ModeManual **(Maschine einrichten)**	Ctrl= MODE_MANUAL Abfrage erfolgt in **traManual** bzw. **traLeaveManual**	**actModeManual_entry** • Setzt den Status auf "MODE_MANUAL"	Keine zyklische Aktion vordefiniert Typische Aktivitäten: Einrichten und Justieren der Maschine
ModeAuto **(zyklischer Automatikbetrieb)**	Ctrl= MODE_AUTO Abfrage erfolgt in **traAuto** bzw. **traLeaveAuto**	**actModeAuto_entry** • Setzt den Status auf "MODE_AUTO"	**actModeAuto_active** z. B. Bearbeitung des Automatik Prozesses

Um Betriebsarten hinzuzufügen, siehe Kap. 9.10.1 "Betriebsart hinzufügen" auf Seite 271.

9.5 Zyklische Bearbeitung

Im GAT^{compact} wird im zyklischen Programmteil **actCyclic** das Steuerungs-Interface und die zentrale Fehlerbehandlung aufgerufen.

9.6 Taskhandling

9.6.1 Taskkonfiguration GAT^{compact}

Die Beispielapplikation GAT^{compact} beschränkt sich bewusst auf eine SPS-Task (PlcTask).

Nachfolgend ist die Standard-Einstellung der PlcTask im GAT^{compact} gezeigt

	Prio	**Zykluszeit**	**Watchdogzeit**	**Empfindlichkeit des Watchdog**
PlcTask	5	T#10ms	T#20ms	3

Abb.9-5:

Weitere Tasks können bei Bedarf im GAT^{compact} ergänzt werden.

9.6.2 Hinzufügen einer BackgroundTask

Falls die Laufzeit der PlcTask zu groß wird, kann durch Auslagern von zeitunkritischen Aufgaben in eine Backgroundtask die PlcTask entlastet werden. Zu bedenken ist dabei, dass durch die zusätzlichen Taskwechsel PlcTask <> BackgroundTask das Gesamtsystem höher belastet wird (siehe auch Kap. "Typische Aufgaben BackgroundTask" auf Seite 42).

	Prio	Zykluszeit	Watchdogzeit	Empfindlichkeit des Watchdog
BackgroundTask	~6	T#40ms	-	-

Um eine BackgroundTask hinzuzufügen sind folgende Schritte notwendig:

- Taskkonfiguration, hinzufügen der Task "BackgroundTask", Einstellungen siehe Abb. 9-6 "Konfiguration der BackgroundTask" auf Seite 259
- Hinzufügen POU "BackgroundProg" und Aufruf von "BackgroundProg" in der Konfiguration der BackgroundTask
- Am FB "Machine" die Methode "mBackground" hinzufügen. Diese im "BackgroundProg" aufrufen. Mit Hilfe der Methode "mBackground" sind alle Variablen in der Methode verfügbar

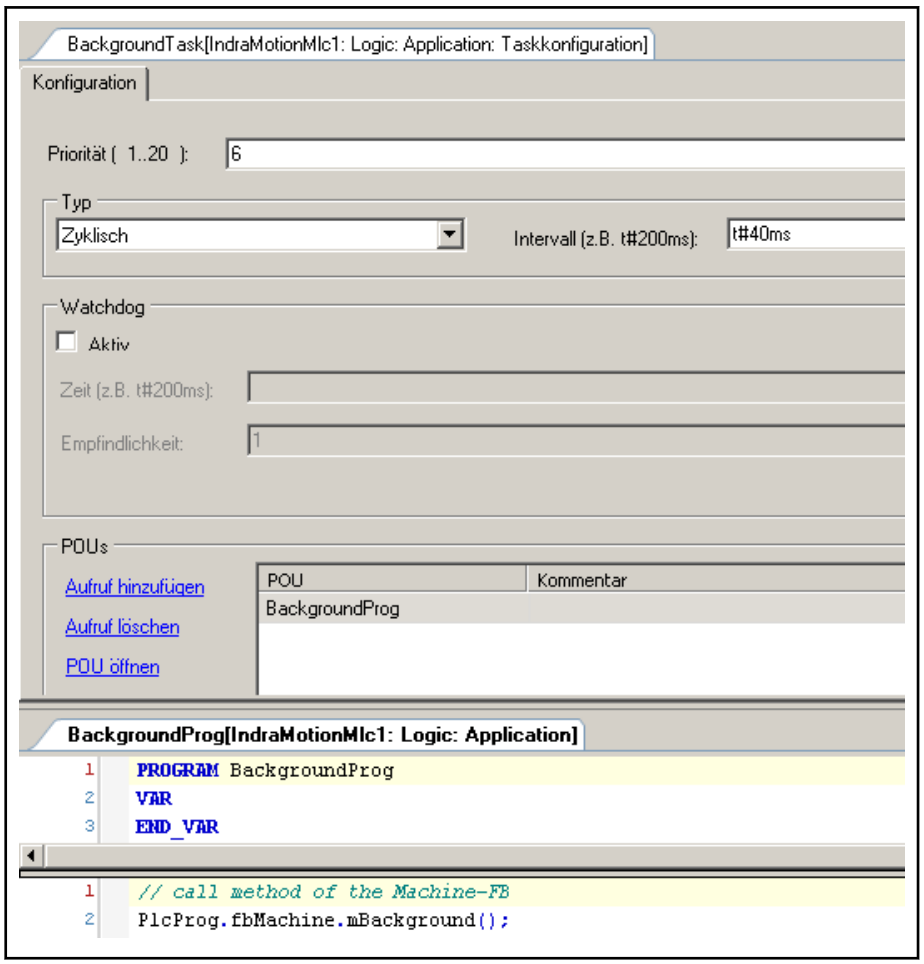


Abb.9-6: Konfiguration der BackgroundTask

GATcompact für IndraLogic XLC ohne sercos Interface

9.7 Fehlerhandling

9.7.1 Fehlererkennung und Diagnose

Die Programmiervorlage GAT^{compact} enthält vordefinierte Funktionalitäten zum Erkennen, Behandeln und Löschen von Fehlern an **einer zentralen Stelle**. Fehler der verschiedenen Fehlerquellen (wie z. B. Steuerungsfehler, Applikationsfehler, FB-Fehler…) werden automatisch durch den maschinenübergreifenden FB "Machine" in der Aktion "mErrorHandling" zusammengeführt und über die Diagnoseausgänge des FB "Machine" bereitgestellt.

Falls mehrere Fehler gleichzeitig anstehen, wird der schwerwiegendste angezeigt. Stehen gleichzeitig mehrere Fehler mit dem gleichen Schweregrad an, so wird der zuletzt erkannte Fehler ausgegeben.

Der FB "Machine" enthält die nachfolgend beschriebenen Ausgänge für Diagnosezwecke:

Element	Typ	Beschreibung
Error	BOOL	TRUE signalisiert Fehler
DiagNo	DWORD	*Nummer gemäß Dokumentation "Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnosen" mit den folgenden Erweiterungen:* • 16#F0AE xxxx: Applikations-Fehler (z. B. geöffnete Schutztür) (xxxx = frei durch den Anwender definierbar) • 16#F0FB 0000: FB-Fehler (z. B. Fehler an einem TechFB) -> Details siehe ErrorID/ErrorIdent • 16#F0AA 0000: GAT interner Fehler -> Details siehe ErrorID/ErrorIdent
ErrorID	ERROR_CODE	Detaildiagnose im Fall von FB- + GAT-Fehlern (DiagNo = 16#F0FB/AA 0000)
ErrorIdent	ERROR_STRUCT	Detaildiagnose im Fall von FB- + GAT-Fehlern (DiagNo = 16#F0FB/AA 0000)
DiagMessage	STRING(60)	Enthält bei Steuerungsfehlern den Diagnosetext

Abb.9-7:

Der Ausgang **DiagNo** ist zur Nutzung an einem HMI vorgesehen. Die Fehlertexte werden dort der numerischen Information in beliebigen Sprachen zugeordnet.

DiagMessage, ErrorID und **ErrorIdent** sind zusätzliche Informationen, die hauptsächlich bei der Inbetriebnahme durch den Programmierer, z. B. über eine IndraLogic-Visualisierung, angezeigt werden können.

Hinweis zu applikationsspezifischen Fehlern

Durch Aufrufen der Methode "mSetAppError(Number)" können Applikationsfehler (wie z. B. geöffnete Schutztür) abgesetzt werden. Die übergebene "Number" am Eingang der Methode entspricht hierbei der Applikationsfehlernummer.

Das zentrale Diagnosesystem im FB "Machine" überwacht hierbei zyklisch die abgesetzten Anwenderfehler und erzeugt den Applikationsfehler DiagNo = Fehler 16#F0AE xxxx (xxxx = Applikationsfehlernummer), wenn ein Applikationsfehler abgesetzt wurde.

In der globalen Variablenliste "UserConstGlobal" können den Applikationsfehlern Diagnosetexte zugeordnet werden, die dann in der SPS-Visualisierung angezeigt werden.

Für Applikationsfehler in der Maschine sind Nummern im Bereich 1 … uiMAX_APP_ERROR_INDEX gültig. Die Konstante "uiMAX_APP_ERROR_INDEX" ist mit dem Wert 64 vorbelegt. Bei Bedarf kann diese Konstante in der

Globalen Variablenliste "UserConstGlobal" unterhalb des "Application"-Ordners angepasst werden.

Codebeispiel zum Absetzen eines Applikationsfehlers:

```
// Recommended code to issue user specific errors (Example)
IF bDoor1Open THEN
    mSetAppError(1);
END_IF
```

Hinweis zu Fehler-Schweregrad

Das Steuerungs-Diagnosesystem definiert 10 Schweregrade "Severity" (siehe IndraLogic XLC-/IndraMotion MLC-Diagnosesystem).

Der Schweregrad hat Einfluss auf die Fehlerreaktion:

- DiagNo = 16#F0xx xxxx ist der geringste Schweregrad (z. B. Applikationsfehler, FB-Fehler)
- DiagNo = 16#F9xx xxxx ist der höchste Schweregrad (fataler Systemfehler)

Es wird der Fehler mit dem höchsten Schweregrad angezeigt, wenn mehrere Fehler gleichzeitig anstehen (siehe Kap. 5.6.4 "Fehlerreaktion" auf Seite 47).

Hinweis zu Funktionsbaustein-Fehler

Treten Fehler an aufgerufenen Funktionsbausteinen (z. B. Technologie-FB) auf, so wird empfohlen die Methode "mSetFBError(ErrorID, ErrorIdent)" aufzurufen und die "ErrorID"- und "ErrorIdent"-Ausgänge des Fehler verursachenden FB an die Methode zu übergeben. Damit wird sichergestellt, dass der Fehler an das Diagnosesystem des FB "Machine" weitergeleitet wird und die "ErrorID" und "ErrorIdent" an den Diagnoseausgängen des FB "Machine" bereitgestellt wird.

Codebeispiel zum Absetzen eines Funktionsbaustein-Fehlers:

```
// Recommended code if a used FB issues an error (Example)
IF fbCrossCutter.Error THEN
    mSetFBError(ErrorID:=fbCrossCutter.ErrorID,
                ErrorIdent:=fbCrossCutter.ErrorIdent);
END_IF
```

9.7.2 Fehlerreaktion

Die Fehlerreaktion hat die Aufgabe Schäden aufgrund von Fehlern des Steuerungssystems möglichst zu vermeiden. In Abhängigkeit des Schweregrades führt daher die Steuerungsfirmware automatisch eine Fehlerreaktion aus (siehe Kap. 5.6.4 "Fehlerreaktion" auf Seite 47).

Bei Fehlern mit Schweregrad kleiner 2 (wie z. B. Applikationsfehler, FB-Fehler…) wird keine automatische Fehlerreaktion durch die Steuerungsfirmware durchgeführt.

Daher enthält das GAT^{compact} eine Funktionalität, um Fehler zu erkennen und eine anwenderspezifische Fehlerreaktion durchzuführen.

Allgemeine GAT Fehlerreaktion

Mit der Standardeinstellung ist die GAT^{compact} Fehlerreaktion eingeschaltet. Im Fehlerfall wird dadurch die angewählte Betriebsart unterbrochen (Ausnahme: Initialisierung) und die Fehlerreaktion ausgeführt. Für die Fehlerreaktion ist die Aktion "**actError_entry**" des Funktionsbausteines "Machine" vorgesehen, und kann applikationsspezifisch angepasst werden.

Abschalten der allgemeinen GAT Fehlerreaktion

Für Sonderfälle kann die allgemeine GAT Fehlerreaktion abgeschaltet werden.

Ändern Sie dazu die Standardeinstellung der Variablen "bDisableErrorHandling_gb" über die Variablendeklaration wie folgt:

- Globale Variablenliste: "UserVarGlobal"
- Deklaration wie folgt anpassen: "bDisableErrorHandling_gb: BOOL := TRUE;"

Bei abgeschalteter Fehlerreaktion wird die aktuelle Betriebsart nicht unterbrochen. Fehler werden jedoch weiterhin angezeigt, und können auch gelöscht werden.

GAT Fehlerreaktion während der Initialisierung

Mit der Standardeinstellung ist die GAT^{compact} Fehlerreaktion während der Initialisierung abgeschaltet. Auftretende Fehler werden jedoch angezeigt und können gelöscht werden.

Für Sonderfälle kann die Fehlerreaktion während der Initialisierung eingeschaltet werden.

Ändern Sie dazu die Standardeinstellung der Variablen "bEnableErrorReactionInit_gb" über die Variablendeklaration wie folgt:

- Globale Variablenliste: "UserVarGlobal"
- Deklaration wie folgt anpassen: "bEnableErrorReactionInit_gb: BOOL := TRUE;"

Wenn die Fehlerreaktion eingeschaltet ist, dann wird die Initialisierung bei Fehlerauftritt unterbrochen. Nachfolgend wird dann die programmierte Fehlerreaktion ausgeführt. Die Fehlerreaktion ist in der Aktion "**actError_entry**" des Funktionsbausteines "Machine" programmiert.

☞ Die Fehlerreaktion während der Initialisierung wird nur dann ausgeführt, wenn die allgemeine GAT Fehlerreaktion eingeschaltet ist (bDisableErrorHandling_gb = FALSE).

9.7.3 Fehlerlöschen

Über eine positive Flanke am Eingang "ClearError" des FB "Machine" wird das zentrale Fehlerlöschen automatisch durchgeführt. Der FB leitet dieses Kommando an alle unterlagerten Komponenten (wie z. B. Imc-Interface[1)] …) weiter um das gesamte System in den fehlerfreien Zustand zu überführen. Die aktiv angewählte Betriebsart wird nach Fehlerlöschen wieder bearbeitet, falls diese zur Fehlerreaktion verlassen wurde.

9.8 Visualisierung

9.8.1 Interne Visualisierung zur Inbetriebnahmeunterstützung

Allgemeines

Das GAT^{compact} beinhaltet vordefinierte IndraLogic-Visualisierungsobjekte, die bei der Inbetriebnahme unterstützen und erweiterte Diagnosemöglichkeiten bieten.

System_Overview

Das Visualisierungselement "System_Overview" ist eine Inbetriebnahme-Visualisierung zur Betriebsarten-Anwahl, Diagnose und Fehlerquittierung.

1) *IndraMotionControl Interface = IndraLogic Bibliothek mit Schnittstelle zur Steuerung, wie. z. B. Fehler löschen*

9.8.2 Externe Visualisierung mit separaten HMI (WinStudio-Visualisierung)

Allgemeines

Für das GAT^{compact} steht eine vordefinierte WinStudio-Visualisierung zur Verfügung, die über den Wizard in das Projekt eingefügt werden kann.

☞ Zur Bearbeitung der WinStudio-Visualisierung wird mindestens folgende Entwicklungsumgebungslizenz oder größer benötigt:

WinStudio-Entwicklungslizenz, max. 1.500 Variablen (Mat.-Nr. R911323596).

Zur Ausführung auf einem VEP wird mindestens folgende Runtime-Lizenz benötigt:

WinStudio-Runtimelizenz für Geräte mit Windows CE als Betriebssystem, max. 1.500 Variablen (Mat.-Nr. R911323620).

Einfügen der WinStudio Visualisierung

☞ Um die WinStudio-Visualisierung dem Projekt hinzufügen zu können, muss ein entsprechendes Visualisierungsgerät im Projekt vorhanden sein.

Fügen Sie mittels "Drag and Drop" ein VEP aus der Gerätebibliothek in den Projektbaum ein und konfigurieren Sie es wie folgt:

- Gerätetyp: VEPxx (CE)
- Applikationstyp: WinStudio-Anwendung

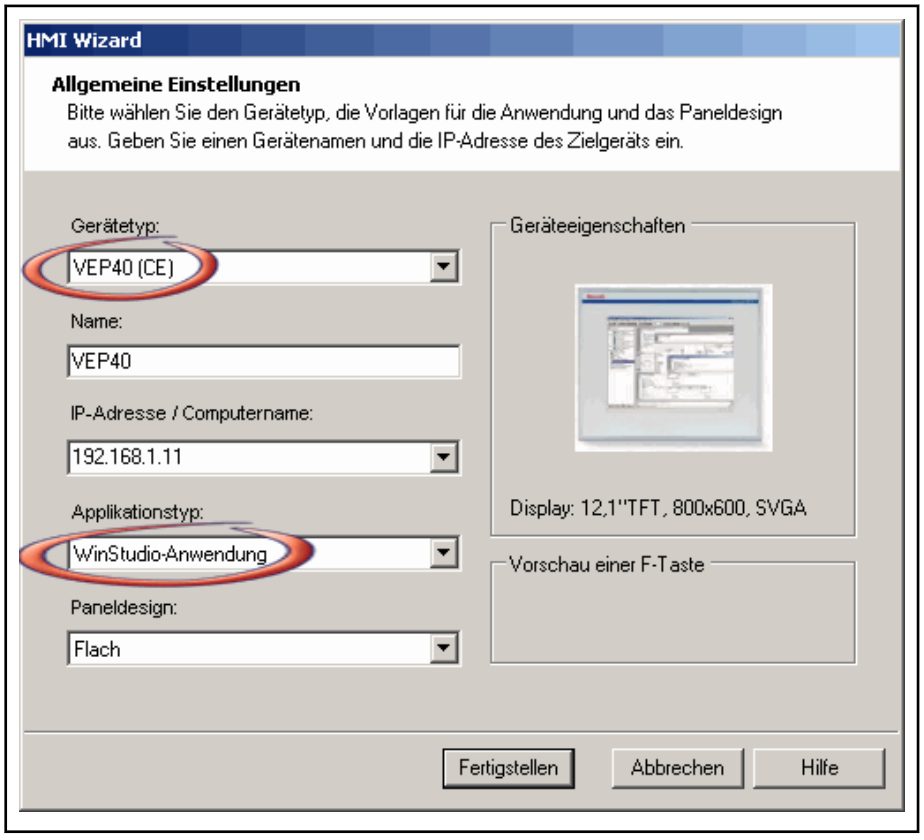


Abb.9-8: Visualisierungsgerät einfügen

GATcompact für IndraLogic XLC ohne sercos Interface

Wenn ein entsprechendes Visualisierungsgerät vorhanden ist, wird über folgenden Link im Wizard die WinStudio-Visualisierung eingefügt:

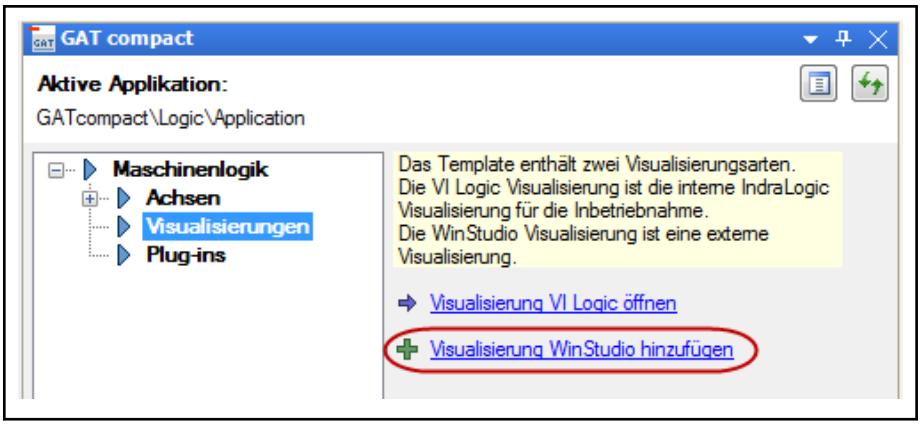


Abb.9-9: WinStudio-Visualisierung einfügen

Im nächsten Dialog des Wizard werden alle Visualisierungsgeräte des Projekts angeboten, deren Konfiguration der Vorgabe entspricht.

Nach Auswahl des entsprechenden Visualisierungsgerätes kann mittels **Übernehmen** das Einfügen der WinStudio-Visualisierung ausgeführt werden.

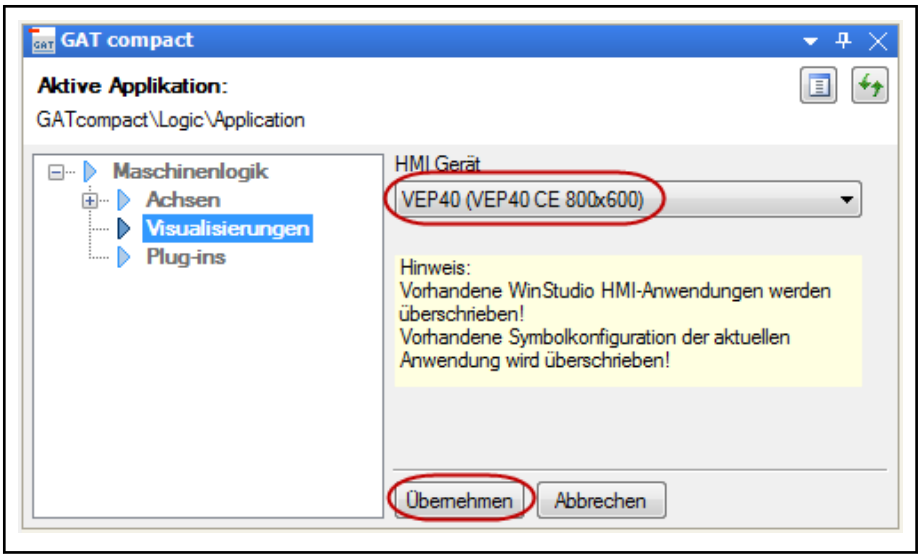


Abb.9-10: Visualisierungsgerät auswählen – WinStudio-Visualisierung einfügen

☞ Bestehende Visualisierungsdaten des Visualisierungsgerätes und die Symbolkonfiguration der Steuerung werden durch das Einfügen der WinStudio-Visualisierung überschrieben!

Nach dem Einfügen können die Visualisierungsdaten bearbeitet werden.

Die Konfiguration der Steuerungsadresse erfolgt im "Eigenschaften"-Dialog des BR-WS Treibers.

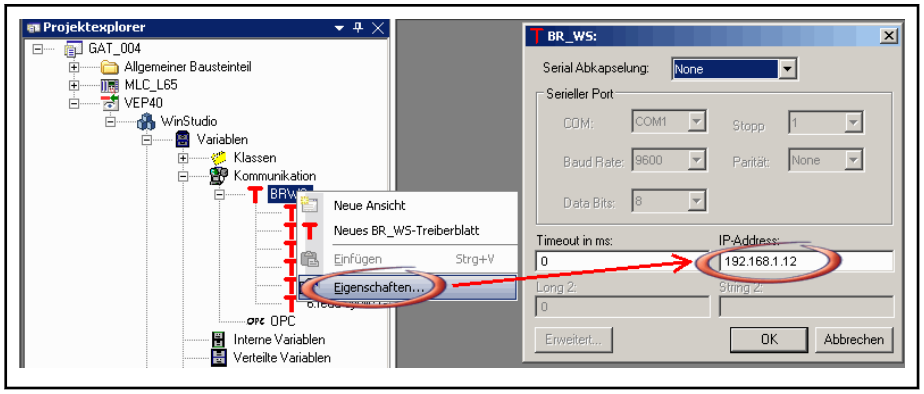


Abb.9-11: Einstellen der IP-Adresse der Steuerung

Im Kontextmenü des WinStudio-Knotens kann nun über  "Anwendung Starten" die Emulation der Winstudio Visualisierung gestartet werden.

Dialog-Struktur

Die Bedienbilder teilen sich in 3 Teilbereiche auf:

- Kopfzeile mit Statusinformationen
- Hauptbildbereich mit menüabhängigen Anzeigen und Funktionstasten
- Fußzeile mit Tasten zur Bedienbildumschaltung

GATcompact für IndraLogic XLC ohne sercos Interface

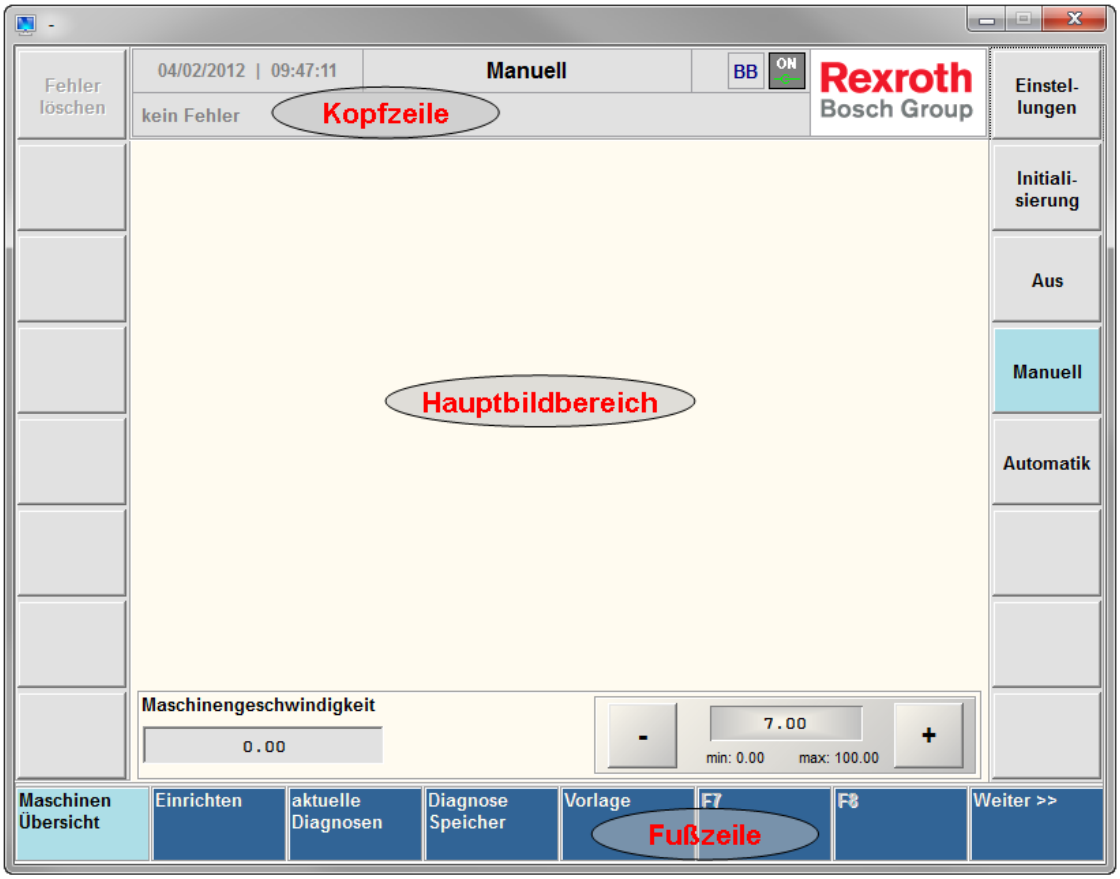


Abb.9-12: WinStudio-Visualisierung

Die Kopfzeile beinhaltet:

- Betriebsartenanzeige
- Globale Funktionstasten zum Fehlerlöschen und Öffnen des "Einstellungen"-Dialogs (Sprachumschaltung, Einstellung der Dezimalstellen für numerische Anzeigen)
- Anzeige aktuell anstehender Maschinen-Diagnosen
- Anzeige des Kommunikationsstatus

Der **Hauptbildbereich** beinhaltet die bedienbildabhängigen Anzeigen und Funktionstasten (z. B. Betriebsartenumschaltung, …).

Die **Fußzeile** beinhaltet die Tasten zur Bedienbildumschaltung.

Erweitern der Visualisierung

Bedienbild einfügen

Zum Erweitern der Visualisierung können neue Bedienbilder mittels Kopieren der vorhanden "Template"-Bilder und Einfügen unter neuem Namen angelegt werden.

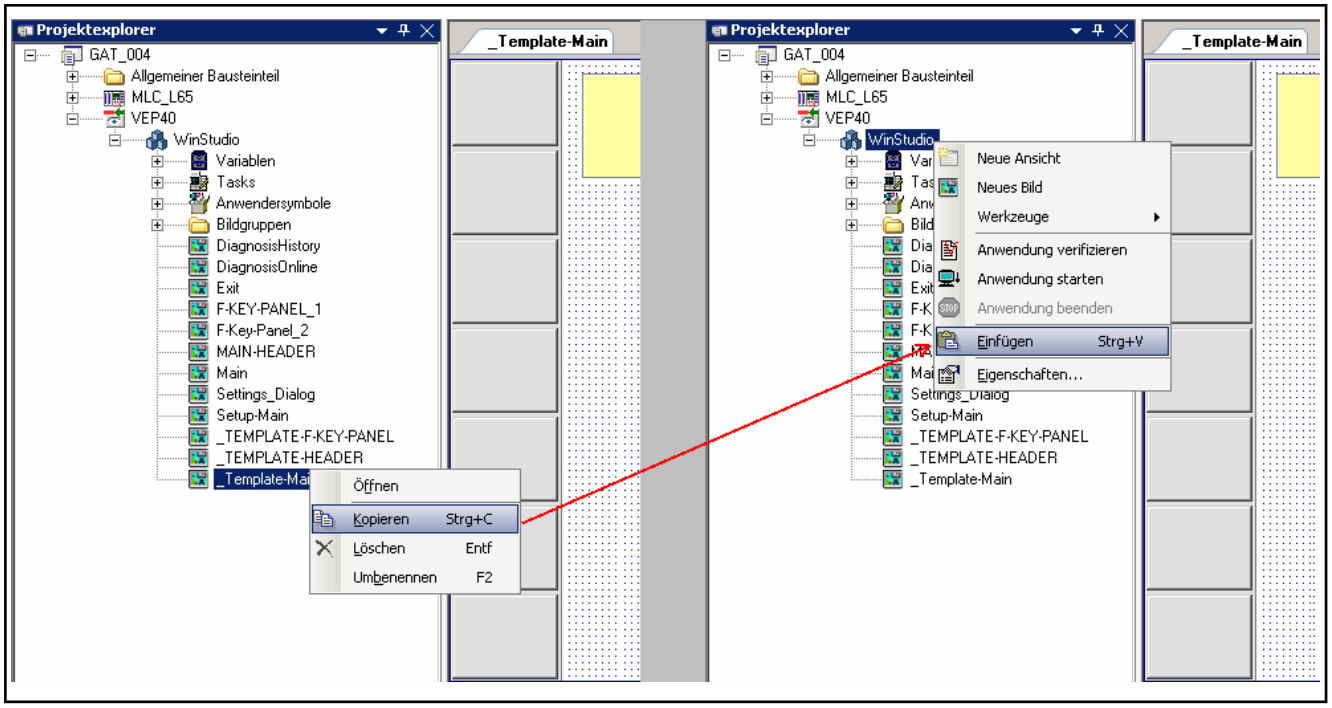


Abb.9-13: Bedienbild duplizieren

Fußzeile erweitern

Die Tasten zur Bedienbildumschaltung sind als parametrierbare Grafik-Objekte angelegt (linked Symbol) und verfügen über folgende Übergabe-Parameter:

- Caption -> Beschriftung der Taste
- Disable -> Eingang zum Sperren der Tastenfunktionalität (ausgegraute Beschriftung)
- ScreenName -> Bildname des zu öffnenden Bedienbildes (bzw. Bildgruppe)
- ScreenNumber -> Nummer, die dem Bild zugewiesen werden soll (zum farblichen hervorheben der jeweiligen Taste)

Kommunikationsanbindung

Die Kommunikation zwischen Visualisierung und Steuerung erfolgt symbolisch über den BR-WS Treiber. Hierzu ist es erforderlich, die Symboleinträge für die zu visualisierenden SPS-Variablen zu erzeugen (siehe hierzu Kap. "Kommunikations-Konfiguration" auf Seite 53).

Die benötigte Symbolkonfiguration der vordefinierten WinStudio-Visualisierung beinhaltet folgende Einträge:

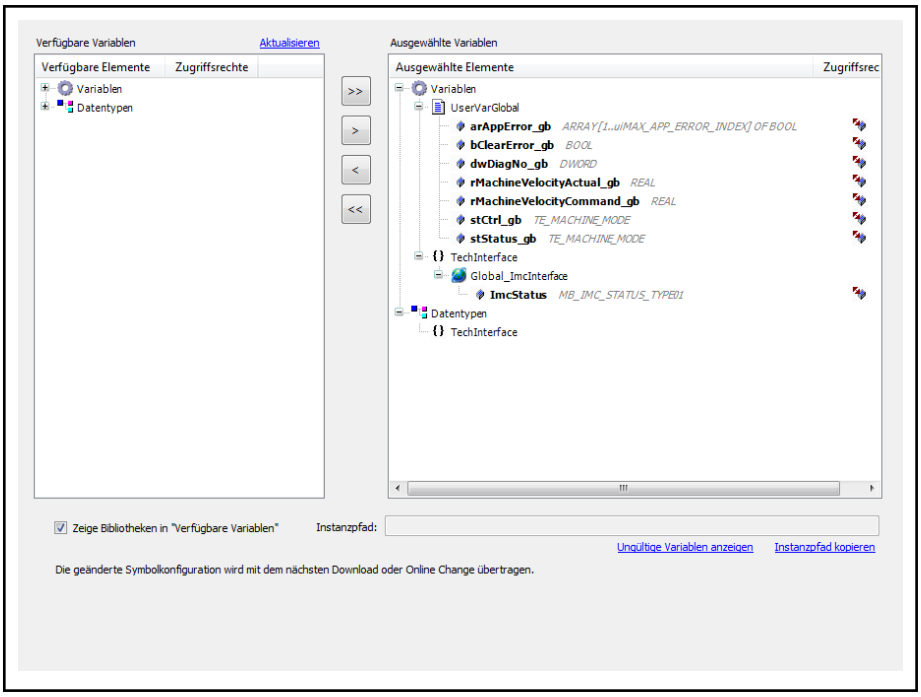


Abb.9-14: Symbolkonfiguration der vordefinierten WinStudio-Visualisierung

Symboleinträge für Variablen aus Globalen Variablenlisten die aus einer Bibliothek stammen, müssen nach einer Versionsänderung der entsprechenden Bibliothek erneut in die Symbolkonfiguration eingefügt werden.

Einstellen der IP-Adresse der Steuerung:

Die Konfiguration der Steuerungsadresse erfolgt im "Eigenschaften"-Dialog des BR-WS Treibers:

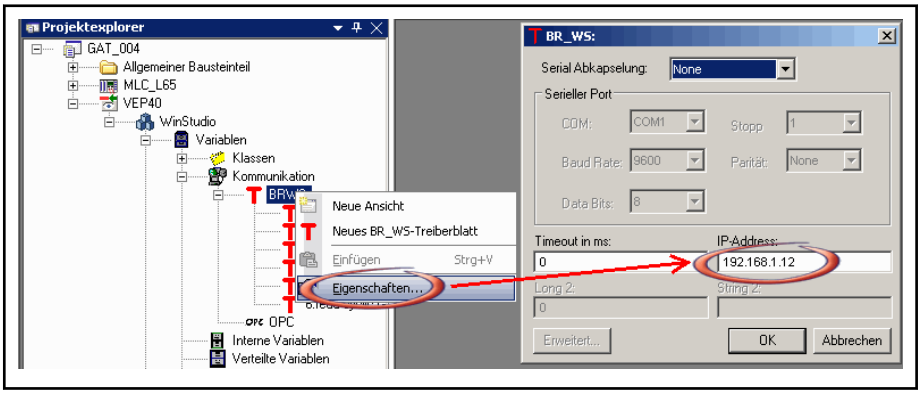


Abb.9-15: Einstellen der IP-Adresse der Steuerung

Um die Kommunikation zu strukturieren, wurden die Variablen in verschiedene Treiberblätter aufgeteilt:

1. **read cyclic fast**: Variablen deren Werte zyklisch in kurzen Intervallen gelesen werden sollen (z. B. Diagnosenummern, Achspositionen, …)
2. **read only once**: Variablen die einmalig nach dem Start der Visualisierung gelesen werden (z. B. Achsnamen)

3. **read after switching phase**: Variablen die nur einmalig nach einer Phasenumschaltung gelesen werden (z. B. ob die Achse aktiviert wurde)
4. **read - write cyclic**: Variablen, deren Werte durch die Visualisierung sowohl gelesen als auch geschrieben werden sollen (z. B. Maschinengeschwindigkeit, Achsgeschwindigkeit für Einrichtbetrieb, …)
5. **write only**: Variablen die durch die Visualisierung nur geschrieben werden sollen (Triggersignal zum Fehlerlöschen, …)

Zum Erweitern eines Treiberblattes kann die offline Symboldatei (*.XML) mit dem BR-WS Browser durchsucht und die entsprechende Variable selektiert werden.

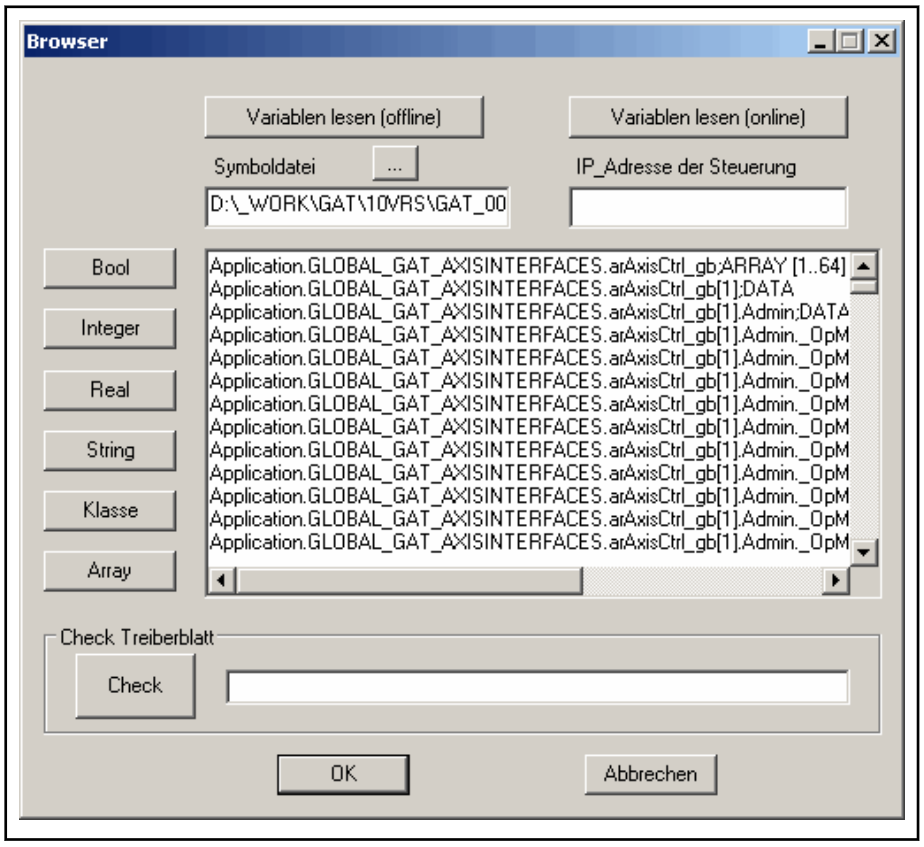


Abb.9-16: Erweitern des Treiberblattes

Mehrsprachigkeit

Die Mehrsprachigkeit der Visualisierung wurde mit der in IndraWorks integrierten Übersetzungsfunktionalität realisiert.

Hierbei sind die Übersetzungsdaten bereits für Steuerungs- und Antriebs-Diagnosen, sowie den verwendeten Texten der Visualisierung vorbereitet. Die Übersetzungsdaten können mit IndraWorks modifiziert bzw. ergänzt werden.

Antriebsdiagnosen sind hierbei in deutsch, englisch, französisch, spanisch und türkisch enthalten.

Alle andere Texte sind in deutsch und englisch enthalten.

9.9 Schnittstellen zum GAT^{compact}

9.9.1 Daten der Machine

Siehe globale Variablenliste **„UserVarGlobal”**:

GATcompact für IndraLogic XLC ohne sercos Interface

I/O-Typ	Name	Datentyp	Beschreibung
I	stCtrl_gb	TE_MACHINE_MODE_CTRL	Vorgabe für die Betriebsarten der Maschine
I	bSkipColdstartAtStartup_gb	BOOL	Beim Starten des GAT^{compact}-Programms wird eine Umschaltung in den Parametriermodus (P2) zur Durchführung der zur Applikation nötigen Parametrierung ausgeführt. Mit TRUE kann diese Phasenumschaltung übersprungen werden, wenn die Applikation keine Parametrierung in P2 erfordert
O	stStatus_gb	TE_MACHINE_MODE_STATUS	Rückmeldung aktuelle Betriebsart der Maschine

I Eingang zum Beschreiben
O Ausgang zum Lesen

9.9.2 Daten zum Fehlerhandling

Siehe globale Variablenliste **„UserVarGlobal”**:

I/O-Typ	Name	Datentyp	Beschreibung
I	bClearError_gb	BOOL	TRUE = Globales Fehlerlöschen
I	bDisableErrorHandling_gb	BOOL	TRUE = bei einem Fehler wird nicht in den Zustand "ERROR" gewechselt. Die Fehleranzeige wird weiterhin ausgeführt und auch Fehlerlöschen ist weiterhin möglich
I	bEnableErrorReactionInit_gb	BOOL	TRUE = bei einem Fehler während der Initialisierung wird in den Zustand "ERROR" gewechselt, wenn bDisableErrorHandling_gb = FALSE ist
I	bLogError_gb	BOOL	TRUE = GAT-, FB- und applikationsspezifische Fehler werden im Steuerungsdisplay angezeigt, und im Fehler- und Diagnosespeicher der Steuerung gespeichert
O	bError_gb	BOOL	TRUE = globaler Fehler steht an
O	dwDiagNo_gb	DWORD	Diagnosenummer gemäß Dokumentation "Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnosen": DiagNo = 16#F0AE xxxx: Applikations-Fehler (z. B. geöffnete Schutztür) DiagNo = 16#F0FB 0000: FB-Fehler DiagNo = 16#F0AA xxxx: GAT-interner Fehler
O	enErrorID_gb	ERROR_CODE	Detailinformation, wenn DiagNo = 16#F0FB 0000/16#F0AA xxxx
O	stErrorIdent_gb	ERROR_STRUCT	Detailinformation, wenn DiagNo = 16#F0FB 0000/16#F0AA xxxx
O	strDiagMessage_gb	STRING(60)	Diagnosetext der Steuerung/Achse
O	arAppError_gb	ARRAY OF BOOL	Applikationsspezifische Fehler (z. B. Schutztür geöffnet)

I Eingang zum Beschreiben
O Ausgang zum Lesen

9.9.3 Daten der Steuerung

I/O-Typ	Name	Datentyp	Beschreibung
I	ImcCtrl (Global_ImcInterface aus der Bibliothek ML_TechInterface)	MB_IMC_CONTROL_TYPE01	Vorgaben für die Steuerung
O	ImcStatus (Global_ImcInterface aus der Bibliothek ML_TechInterface	MB_IMC_STATUS_TYPE01	Rückmeldungen der Steuerung

I Eingang zum Beschreiben
O Ausgang zum Lesen



Elemente der ImcCtrl_gb werden im FB "Machine" beschrieben und können deshalb von außen nicht mehr genutzt werden.

9.10 HowTo: Typische Anwenderaktivitäten

9.10.1 Betriebsart hinzufügen

GAT^{compact} bringt vordefiniert die folgenden Betriebsarten mit:

- OFF
- MANUAL
- AUTO

Im Folgenden soll beispielhaft zusätzlich die Betriebsart **"CLEANING"** (Reinigung der Maschine) implementiert werden.

Dazu sind folgende Schritte notwendig:

- TYPE "TE_MACHINE_MODE" (im Ordner GAT^{compact}): erweitern um das Element **MODE_CLEANING**
- *FB "Machine":*
  1. Die Transition "traAuto" kopieren, in "traCleaning" umbenennen, und in "traCleaning" MODE_AUTO durch MODE_CLEANING ersetzen, Ergebnis: (Ctrl= MODE_CLEANING;)
  2. Die Transition "traLeaveAuto" kopieren, in "traLeaveCleaning" umbenennen, und in "traLeaveCleaning" MODE_AUTO durch MODE_CLEANING ersetzen, Ergebnis: NOT(Ctrl= MODE_CLEANING;)
  3. Parallel zum "ModeAuto" einen Zweig "ModeCleaning" einfügen
  4. Am "ModeCleaning" die oben genannten Transitionen verwenden
  5. Am "ModeCleaning" eine Eingangsaktion "actModeCleaning_entry()" hinzufügen. In der Aktion als erste Zeile den folgenden Code einfügen: Status:= MODE_CLEANING;
  6. Nach den Transitionen die gleichen Sprungziele wie bei "ModeAuto" einfügen

GATcompact für IndraLogic XLC ohne sercos Interface

Ergebnis der Erweiterung:

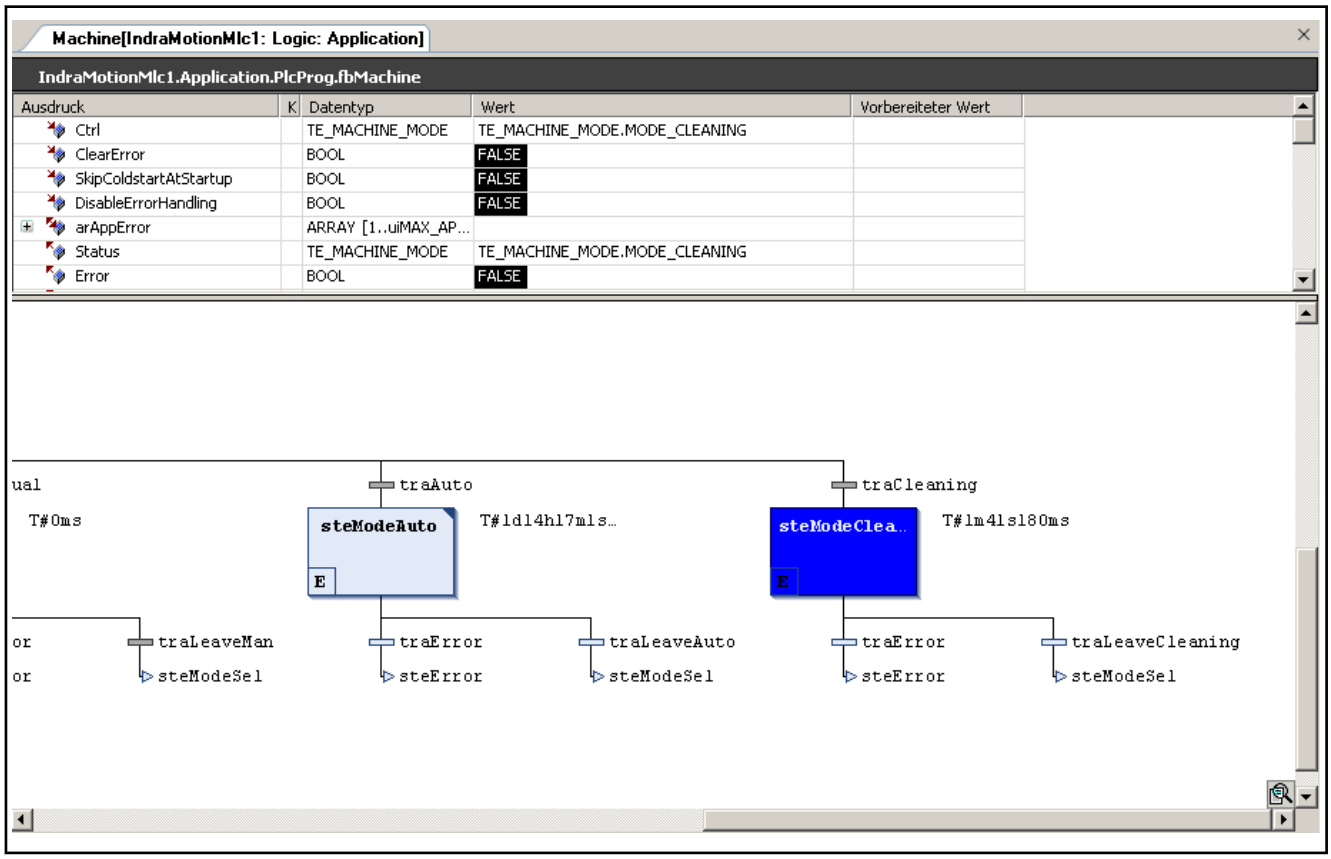


Abb.9-17: Ergebnis nach Betriebsart hinzufügen

In der Aktion "actModeCleaning_entry()" bzw. in einer optionalen Aktion "actModeCleaning_active()" kann das gewünschte Verhalten implementiert werden.

9.11 GAT^{compact} Kurzübersicht

9.11.1 Strukturübersicht

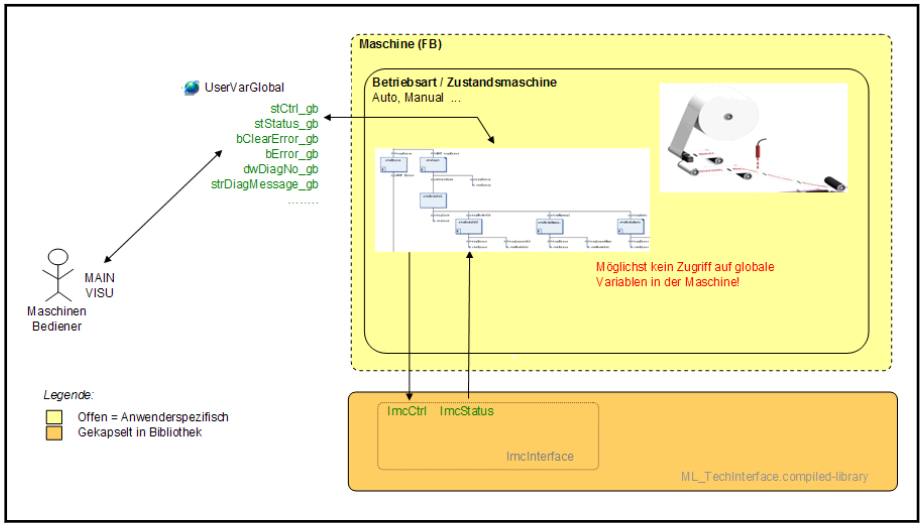


Abb.9-18: GAT^{compact} Strukturübersicht

9.11.2 Übersicht der wichtigsten globalen Variablen der Maschine

Siehe globale Variablenliste **„UserVarGlobal"**:

Variable	Beschreibung	Zugriff
stCtrl_gb	Steuerung der Maschine (Betriebsartenanwahl, Standardbetriebs-art)	RD/WR
stStatus_gb	Status der Maschine (aktuelle Betriebsart)	RD
bClearError_gb	Fehlerlöschen	RD/WR
bDisableErrorHandling_gb	Bei Fehler nicht in den Fehlerzustand wechseln	RD/WR
bSkipColdstartAtStartup_gb	Beim Programmstart wird der "ColdStart" (Umschaltung in P2) nicht ausgeführt	RD/WR
bError_gb	TRUE zeigt einen Fehler an	RD
dwDiagNo_gb	Diagnosenummer entsprechend der IndraLogic XLC-/IndraMotion MLC-Dokumentation	RD
enErrorID_gb	Detailinformationen wenn dwDiagNo_gb = 16#F0FB 0000 (Funkti-onsbausteinfehler)	RD
stErrorIdent_gb	Detailinformationen wenn dwDiagNo_gb = 16#F0FB 0000 (Funkti-onsbausteinfehler)	RD
strDiagMessage_gb	Steuerungs- bzw. Achsdiagnosetext	RD
arAppError_gb	Array of BOOL mit anwenderspezifischen Fehlerflags der Maschi-ne	RD

RD Lesen
WR Schreiben

GATcompact für IndraLogic XLC ohne sercos Interface

9.11.3 Übersicht der Methoden und Aktionen der Maschine

Aktion/Methode	Aufruf	Zugehörige Task	Beschreibung/Beispiel
actCyclic_active	In jedem Zustand	PlcTask	Übergreifende Logik, z. B. Sicherheitstechnik
actError_entry	Für einen Zyklus beim Übergang in den Zustand MODE_ERROR	PlcTask	Applikationsspezifische Fehlerreaktion (Stillstand mit Leistung)
actInit_entry	Für einen Zyklus beim Übergang in den Zustand MODE_INIT	PlcTask	Vorbereiten des Zustands MODE_INIT
actInit_active	Nur im Zustand MODE_INIT	PlcTask	Umschaltung in den Parametriermodus , mögliches Beschreiben von Parametern in P2. Beim Verlassen des MODE_INIT Umschaltung in den Betriebsmodus (bb), mögliches Beschreiben von Parametern beim Erreichen von bb
actModeOff_entry	Für einen Zyklus beim Übergang in den Zustand MODE_OFF	PlcTask	Vorbereiten des Zustands MODE_OFF
actModeManual_entry	Für einen Zyklus beim Übergang in den Zustand MODE_MANUAL	PlcTask	Vorbereiten des Zustands MODE_MANUAL
actModeAuto_entry	Für einen Zyklus beim Übergang in den Zustand MODE_AUTO	PlcTask	Vorbereiten des Zustands MODE_AUTO
actModeAuto_active	Nur im Zustand MODE_AUTO	PlcTask	z. B. Überwachungen im Automatik-Mode, Aufruf von Prozessregler Funktionsbausteinen
FB_init	Einmalig vor SPS-Programmstart/Reset	keine bestimmte Taskzuordnung	Systemintern, darf nicht modifiziert werden

10 GAT mit zentraler Zustandsmaschine

10.1 Allgemeines

Das Generic Application Template ist ein in IndraWorks integriertes Softwarewerkzeug zum Erstellen von SPS-Applikationen. Ausgehend von einem Programmrahmen entsprechend IEC61131-3, ermöglicht es durch dialoggeführte Anwendungserstellung die einfache und schnelle Entwicklung von Applikationen. Es unterstützt auch die Besonderheiten von Motion-Applikationen, wie zum Beispiel die Handhabung von Achsen.

☞ Das GAT mit zentraler Betriebsarten-Schrittkette erfordert eine IndraMotion MLC-Steuerung. Die Steuerung IndraLogic XLC wird derzeit nicht durch das GAT unterstützt.

10.2 Überblick

10.2.1 Grundidee

Die Anwendung des GAT beruht auf dem Ansatz, dass eine Applikation auf einer Maschine läuft, die eine oder mehrere funktionale Einheiten in unterschiedlichen Betriebsarten koordiniert.

Dieser Ansatz soll an einem Beispiel verdeutlicht werden.

Für eine einfache Verpackungsmaschine, bestehend aus einer Zuführeinheit, einer Vereinzelungseinheit und einer Verpackungseinheit (siehe nachfolgende Abbildung) soll eine Applikation erstellt werden.

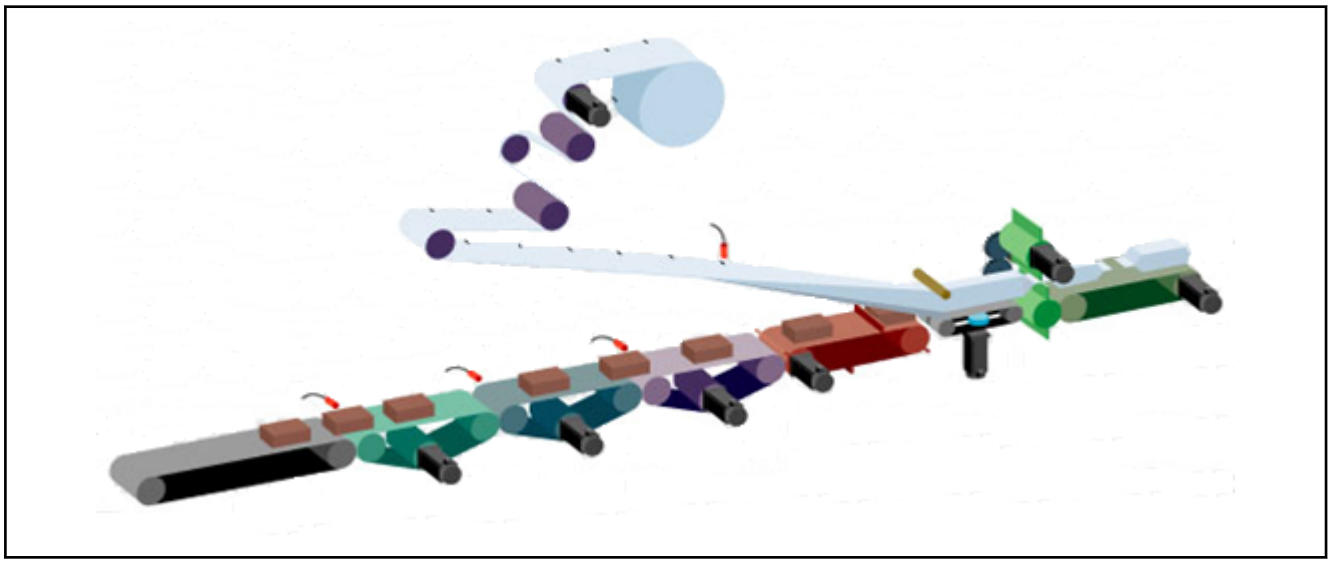


Abb.10-1: Beispiel: Horizontale Schlauchbeutelmaschine (HFFS)

Zur Anwendung des GAT wird diese Verpackungsmaschine in 3 funktionale Einheiten (Module) aufgeteilt.

GAT mit zentraler Zustandsmaschine

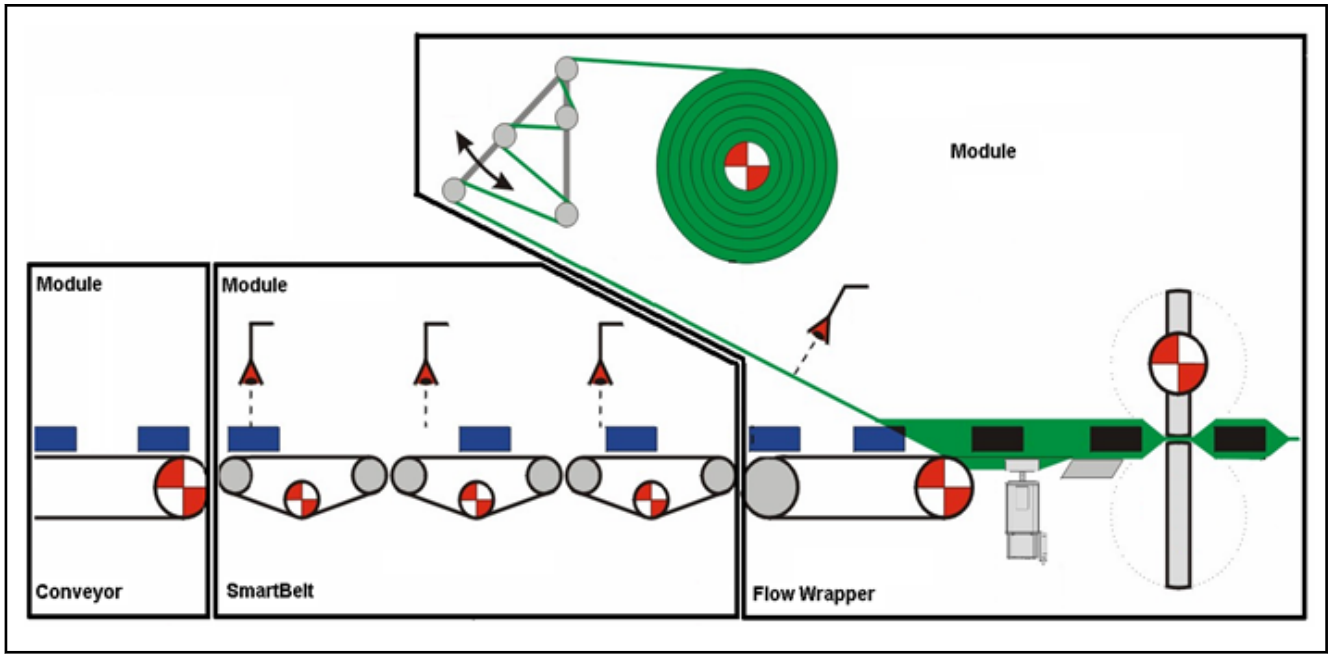


Abb.10-2: Funktionale Einheiten der horizontalen Schlauchbeutelmaschine

Im Ergebnis ergibt sich nachfolgende Softwarearchitektur, die unter Nutzung des GAT direkt in die entsprechende Software der realen Maschine umgesetzt werden kann.

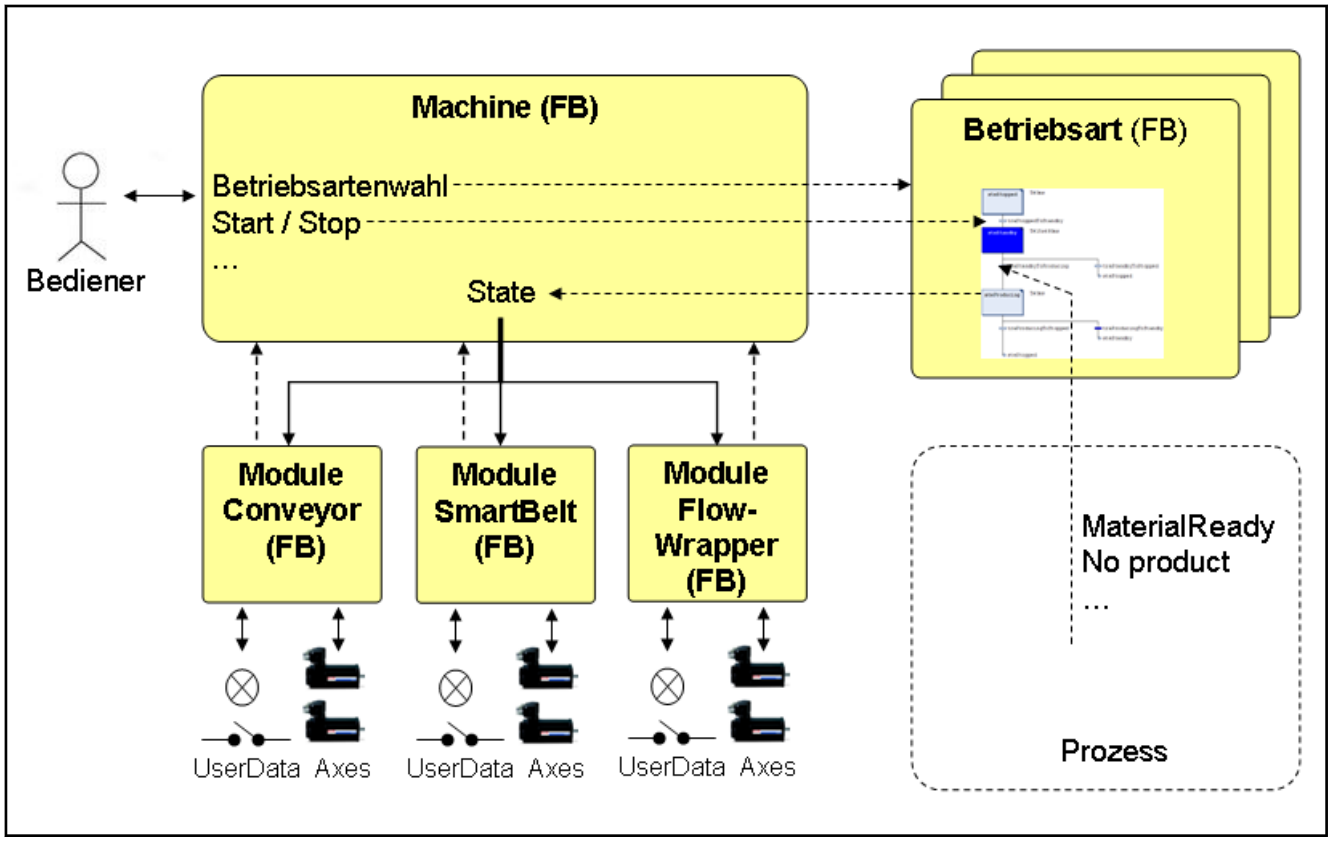


Abb.10-3: Softwarearchitektur unter Nutzung des GAT

10.2.2 Voraussetzungen zur Anwendung

Wie im vorhergehenden Kapitel bereits ansatzweise aufgezeigt, sollen hier die Voraussetzungen vor der Verwendung des GAT noch einmal zusammengefasst werden.

Bevor eine Motion-Applikation mit dem GAT entwickelt wird, muss der Anwender hinsichtlich seiner Applikation folgende Fragen klären:

- Welche Betriebsarten sollen realisiert werden? Welche Zustände sind in der jeweiligen Betriebsart nötig?
- Aus welchen funktionalen Einheiten (Modulen) setzt sich die Maschine zusammen. Welche Achsen und I/O gehören zu der jeweiligen funktionalen Einheit?

Mit diesen Informationen kann der Wizard in kürzester Zeit das Code-Gerüst der konkreten Motion-Applikation erzeugen.

Im Folgenden wird beschrieben, welche Teile zum GAT gehören, wie diese Teile zusammenarbeiten und wo der Anwender zur Realisierung seiner konkreten Applikation Ergänzungen vornehmen muss.

10.3 Erste Schritte – GAT mit zentraler Zustandsmaschine

10.3.1 Einfache Motion-Applikation erstellen

Allgemeines

Mit der nachfolgenden Beschreibung können sie in wenigen Minuten über den GAT Wizard eine Motion Logic Applikation mit 2 Modulen und 2 Achsen erstellen.

Sie benötigen hierfür eine IndraMotion MLC Steuerung, reale Antriebe sind nicht erforderlich. Grundlage ist das GAT mit zentraler Betriebsarten Schrittkette.

Aufgabenstellung

- Eine Maschine soll aus zwei funktionalen Einheiten (Modulen) bestehen.
- Ausgehend von einem Stopp-Zustand soll sich in der ersten funktionalen Einheit eine Achse im Automatik-Betrieb in einem produzierenden Zustand mit 100U/min kontinuierlich bewegen.
- In der anderen Einheit soll eine zweite Achse im produzierenden Automatik-Betrieb winkelsynchron zur Achse im ersten Modul arbeiten. Nachfolgende Abbildung zeigt schematisch die Aufgabenstellung.

GAT mit zentraler Zustandsmaschine

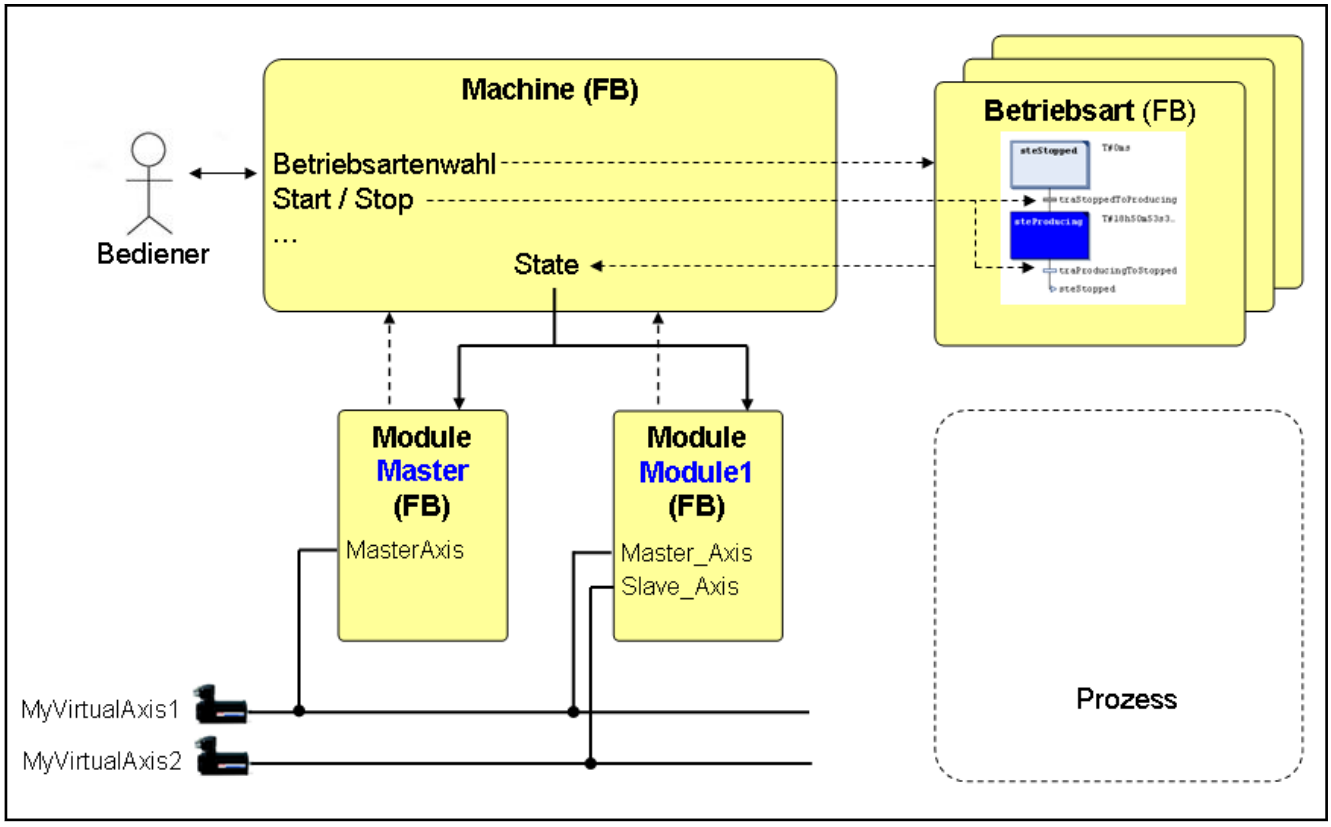


Abb.10-4: Schematisierte Aufgabenstellung

Arbeitsablauf

- IndraWorks öffnen und neues Projekt erstellen.
- MLC Steuerung aus der Bibliothek in den Projektbaum hinzufügen.
- GAT zur MLC Steuerung hinzufügen. Hierfür das "GAT"-Kontextmenü der Steuerung nutzen, oder den GAT-Button  in der IndraWorks Toolbar anklicken.
- Über nachfolgenden Dialog die Option "GAT (zentrale Zustandsmaschine)" auswählen und Schaltfläche "Fertigstellen" anklicken.

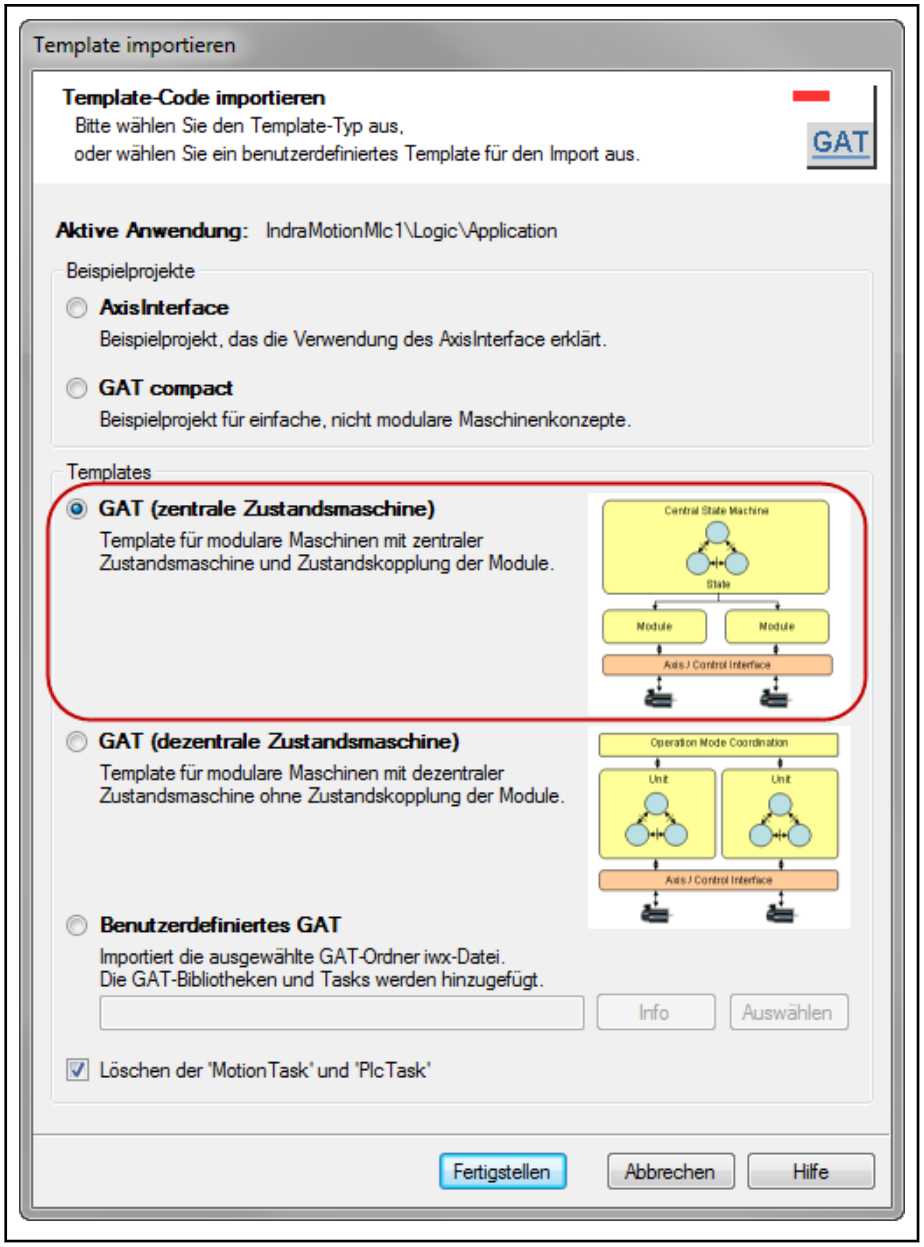


Abb.10-5: GAT mit zentraler Zustandsmaschine importieren

Durch die Auswahl des "GAT mit zentraler Zustandsmaschine" wird ein lauffähiges und vorkonfiguriertes Projekt importiert. Dieses Projekt enthält bereits 2 virtuelle Achsen.

AUTO-Betriebsart

Im GAT ist bereits eine AUTO-Betriebsart mit den Zuständen "Stopped" und "Producing" vordefiniert. Der Wechsel von Zustand "Stopped" in den Zustand "Producing" findet statt, wenn die globale Variable "bStart_gb" gesetzt wird.

GAT mit zentraler Zustandsmaschine

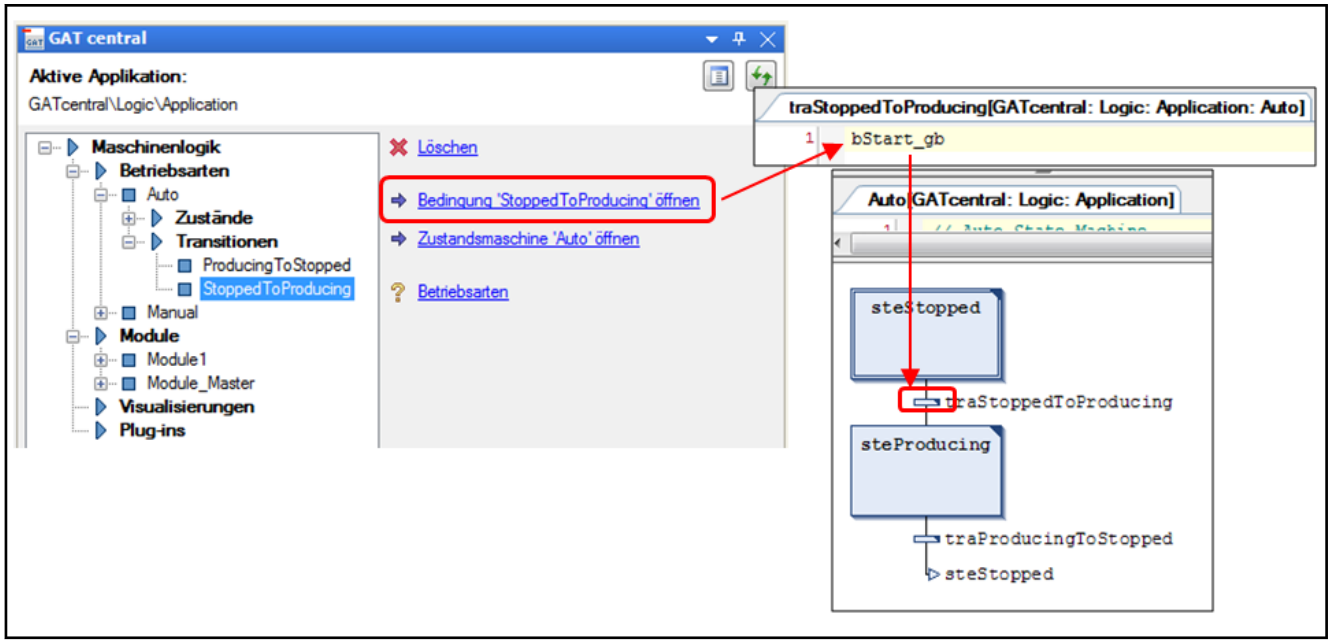


Abb.10-6: Transition bearbeiten

Zur Realisierung der Aufgabe ist hinsichtlich der Definition von Betriebsarten oder Zuständen kein weiterer Aufwand nötig.

Module_Master anlegen

Durch den GAT Erstimport ist bereits ein Modul (Module1) vorhanden. Zur Realisierung der Aufgabenstellung wird ein weiteres Modul "Module_Master" benötigt. Dieses Modul enthält eine Achse "MasterAxis" welche sich im produzierenden Zustand bewegen soll, und im nicht produzierenden Zustand anhalten soll.

Mit folgenden Schritten legen Sie dieses Modul an:

- Im GAT Verzeichnisbaum den Knoten "Module" anklicken und anschließend "Modul hinzufügen" anwählen

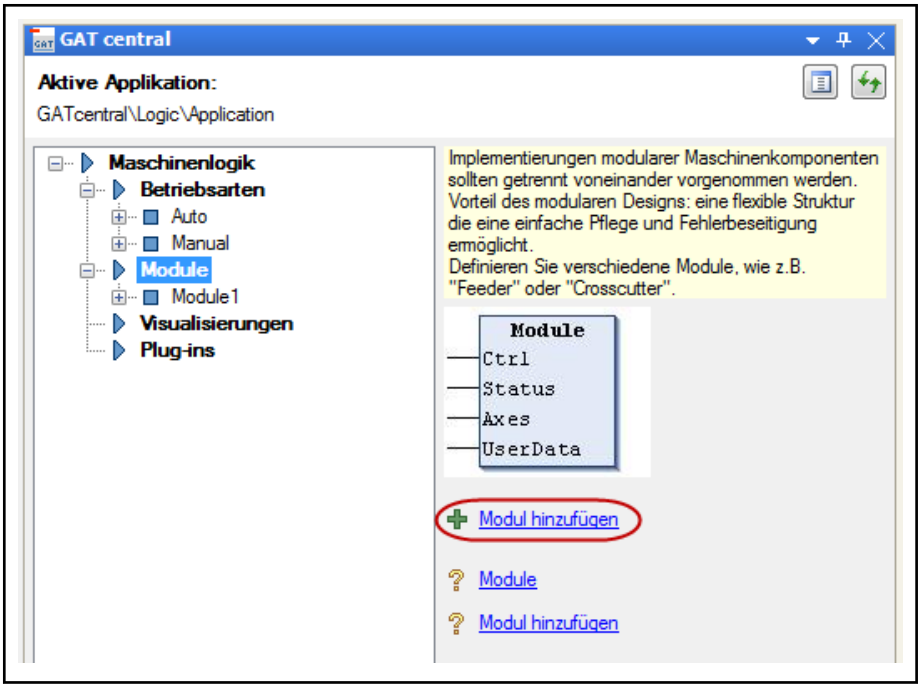


Abb.10-7: Modul hinzufügen

- In dem nachfolgenden Dialog den Namen "Module_Master" wie folgt eintragen und Schaltfläche "Übernehmen" anklicken.

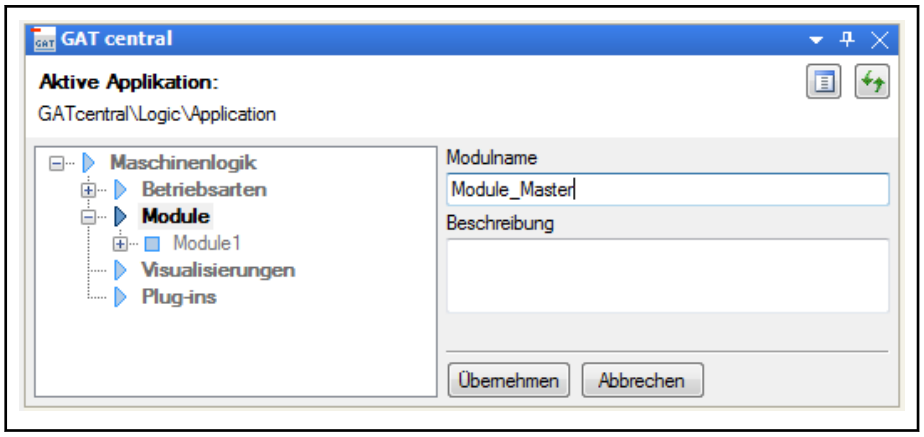


Abb. 10-8: Modulname festlegen

- Die Achse "MasterAxis" für das Modul "Module_Master" über den GAT-Wizard wie folgt hinzufügen und Schaltfläche "Übernehmen" anklicken.

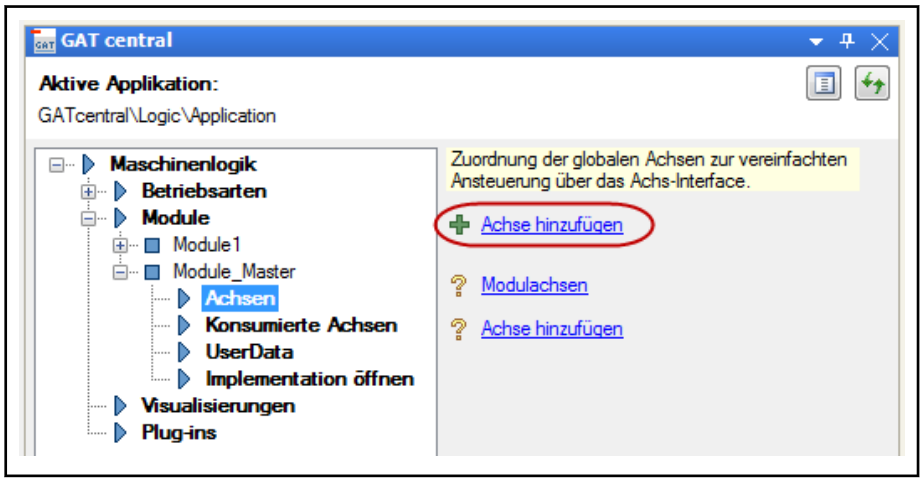


Abb. 10-9: Hinzufügen einer Achse mit dem GAT-Wizard

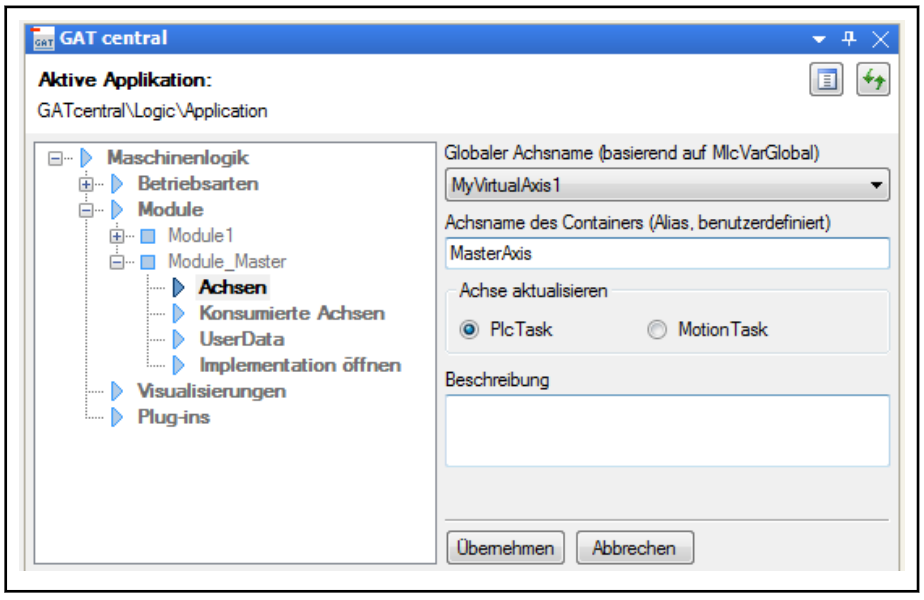


Abb. 10-10: Zuweisung der Modulachse zur global konfigurierten Achse

GAT mit zentraler Zustandsmaschine

- Den Programmcode für den Automatikbetrieb in den Zuständen "Producing" und "Stopped" wie folgt hinzufügen. Hierzu den Ordner "Implementierung öffnen" anwählen, und über die Links "Auto_Producing" und "Auto_Stopped" die Codestellen öffnen. Tragen Sie dort den Programmcode aus folgender Abbildung ein. Damit wird im Zustand "Auto_Stopped" die "MasterAxis" bis zum Stillstand verzögert (durch ModeAH). Weiterhin wird im Zustand "Auto_Producing" die "MasterAxis" in Geschwindigkeitsregelung mit 100U/min gestartet (durch ModeVel).

Zur komfortablen Achskommandierung steht Ihnen zusätzlich die dialogbasierte Achskommandierung zur Verfügung. Mehr Details zu dem Thema finden Sie in Kap. 6.4.1 "Dialogbasierte Achskommandierung" auf Seite 73.

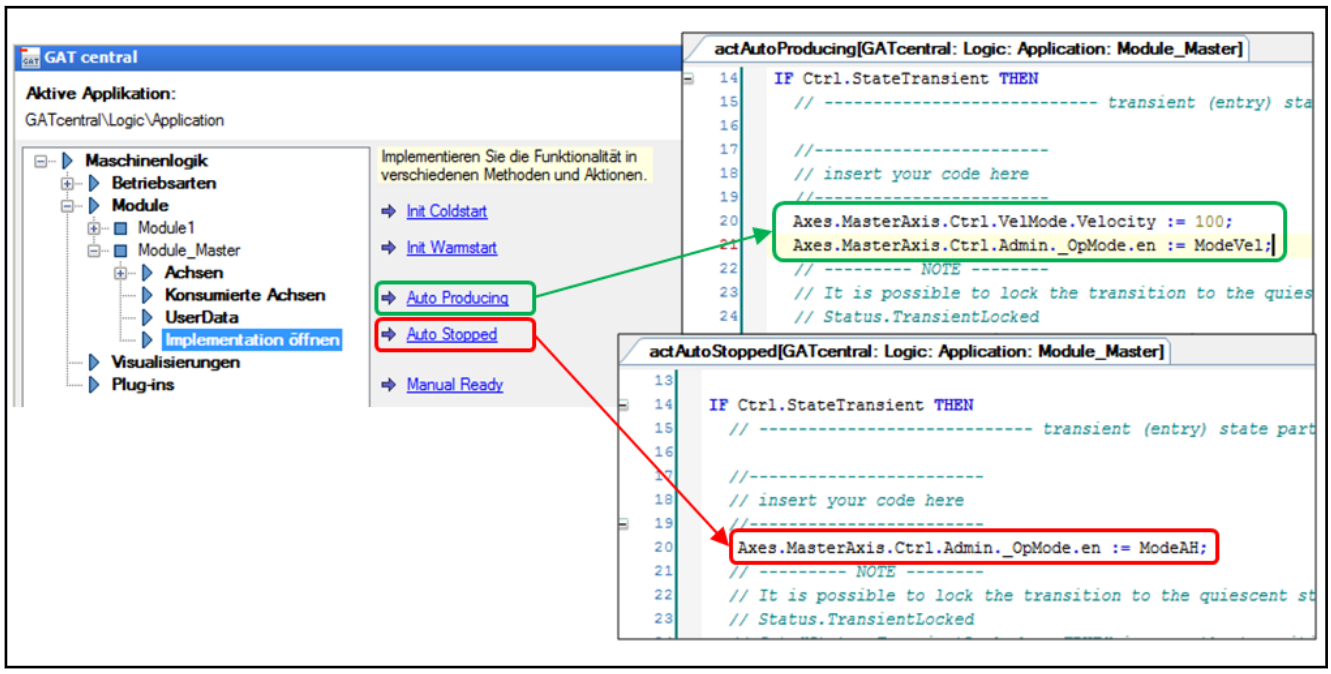


Abb.10-11: Implementierung des Codes in die Aktionen der Zustände

Hinweis zum Achscontainer "Axes"

Die Referenzierung einer Achse innerhalb eines Moduls erfolgt immer über "Axes". "Axes" ist in jedem Modul der Name des Achscontainers. Durch Eingabe eines "." (Punkt) nach Eingabe von "Axes" werden alle Achsen angezeigt, die im Achscontainer enthalten sind (siehe nachfolgende Abbildung). Auf gleiche Art und Weise erfolgt der Zugriff auf alle I/O-Daten des Moduls über "UserData".

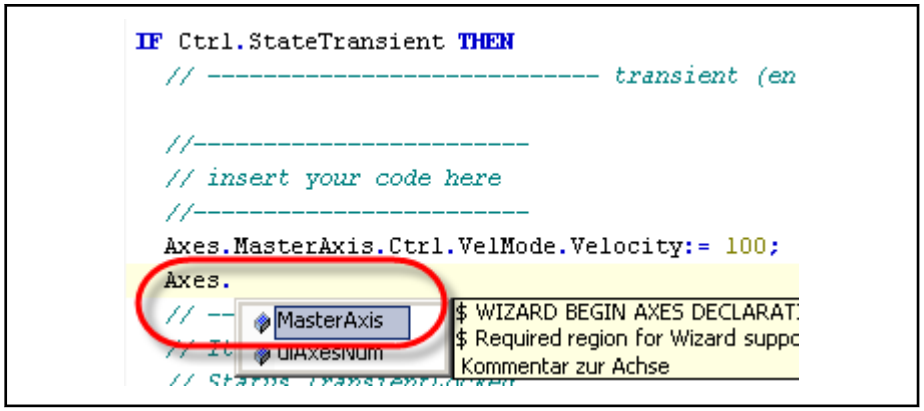


Abb.10-12: Achskommandierung und Intellisense

Des Weiteren werden z. B. durch Eingabe des Punkt-Operators nach dem Achsnamen alle Elemente angezeigt und zur Auswahl gestellt, die Daten zu dieser Achse beinhalten (siehe nachfolgende Abbildung). Die nächste Ebene wird durch Eingabe eines Punktes basierend auf dem Namen der vorhergehenden Ebene erreicht.

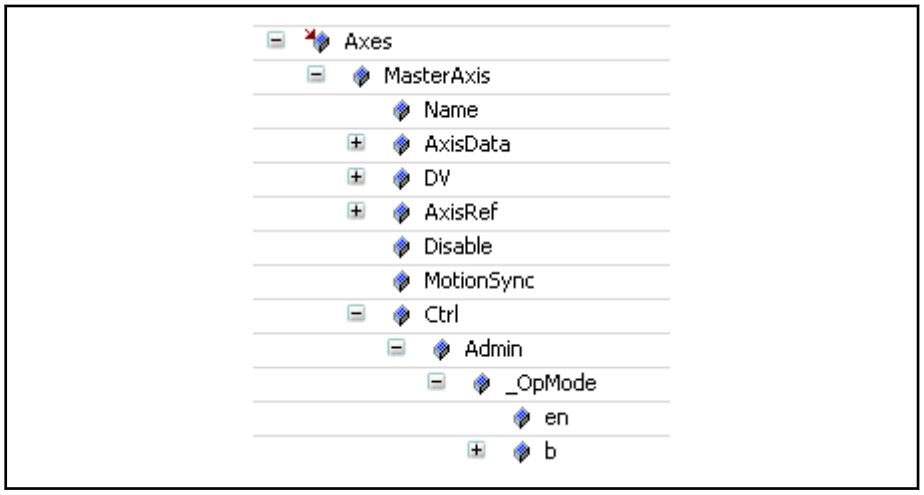


Abb.10-13: Verfügbare Strukturelemente einer Achse

Module1 anpassen

Module1 ist bereits durch den GAT Erstimport enthalten. Dieses Modul wird mit den folgenden Schritten angepasst. Hierbei soll sich in der Betriebsart "Automatik" die "Slave" Achse winkelsynchron zu der "Master" Achse bewegen. Die Master Achse wird bereits in dem "Module_Master" in der Automatik Betriebsart gestartet bzw. gestoppt.

- "Master_Axis" als konsumierte Achse über den GAT Wizard wie folgt hinzufügen.

GAT mit zentraler Zustandsmaschine

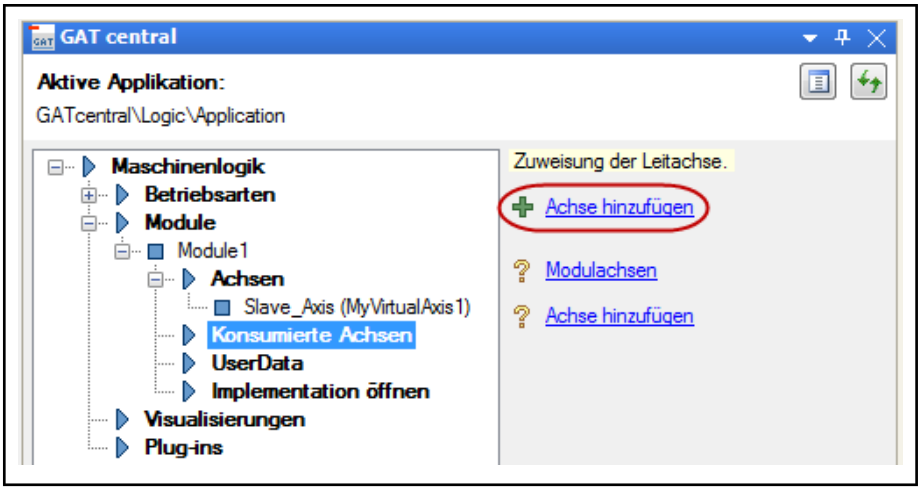


Abb. 10-14: Konsumierte Achse hinzufügen

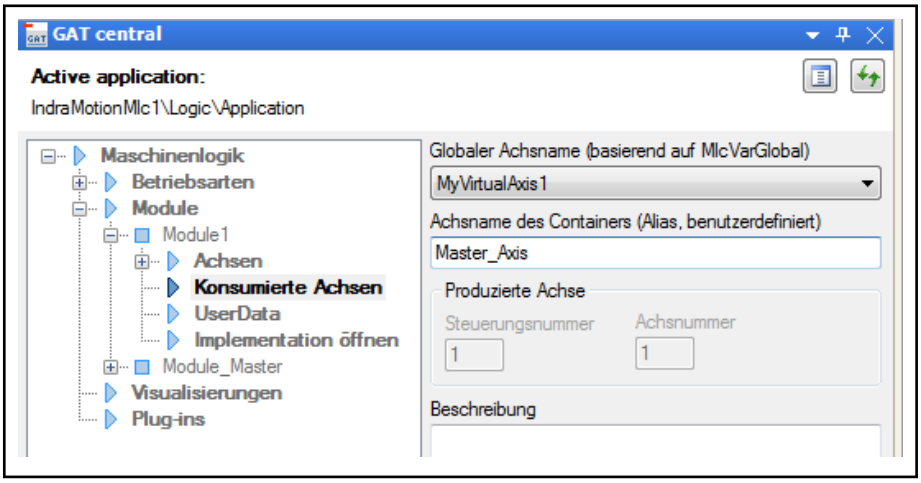


Abb. 10-15: Konsumierte Master Achse hinzufügen

☞ Die "Master_Axis" wird als "konsumierte Achse" in das Module1 hinzugefügt, da diese Achse in diesem Modul nur als Leitachse verwendet wird.

- Achse "MyAxis1" über den GAT Wizard entfernen.

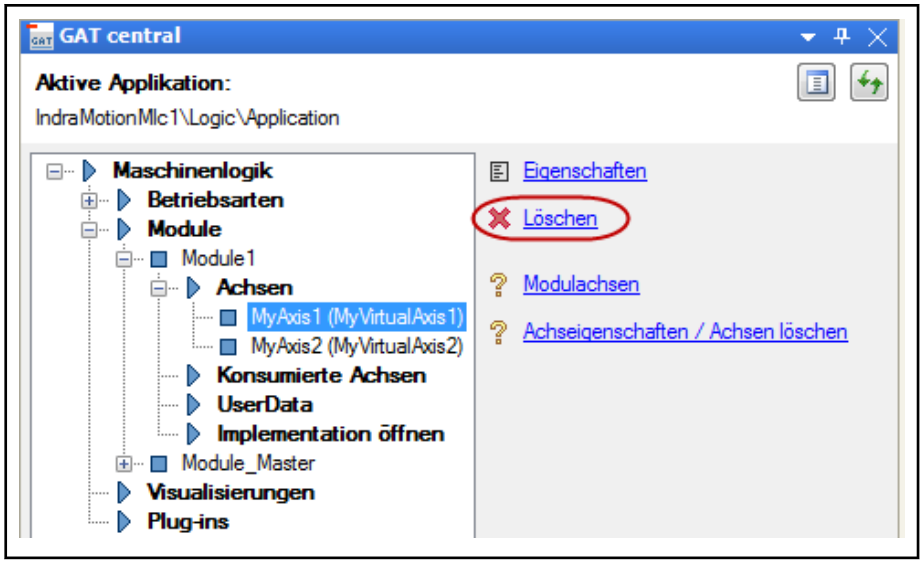


Abb.10-16: Achse "MyAxis1" über den GAT Wizard entfernen

- Achse "MyAxis2" in "Slave_Axis" umbenennen.

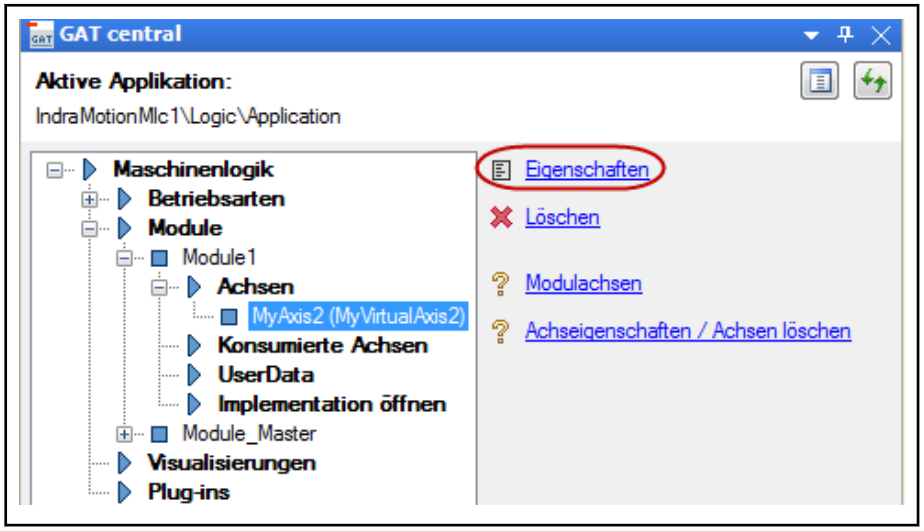


Abb.10-17: Bearbeitung der konfigurierten Achsen

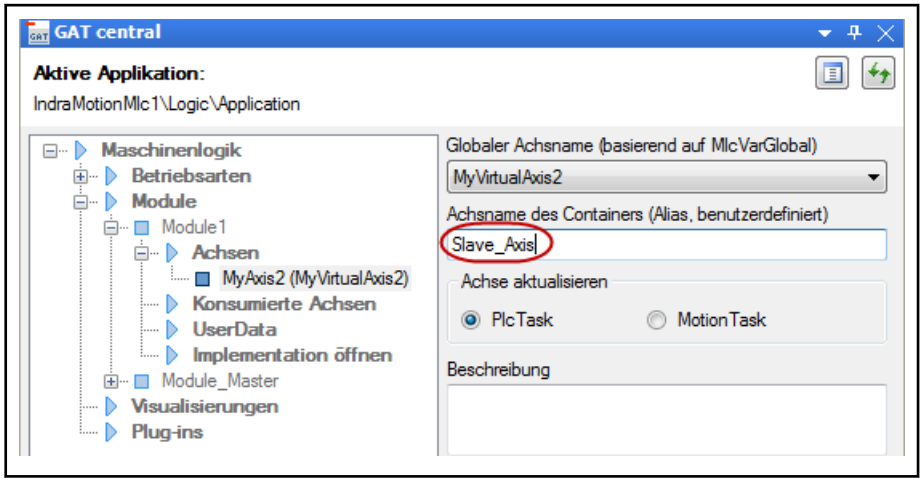


Abb.10-18: Achse umbenennen

GAT mit zentraler Zustandsmaschine

- Den Programmcode für den Automatikbetrieb in den Zuständen "Producing" und "Stopped" wie folgt hinzufügen. Hierzu den Ordner "Implementierung öffnen" anwählen, und über die Links "Auto_Producing" und "Auto_Stopped" die Codestellen öffnen. Tragen Sie dort den Programmcode aus folgender Abbildung ein. Damit wird im Zustand "Auto_Stopped" die "Slave_Axis" bis zum Stillstand verzögert (durch ModeAH). Weiterhin wird im Zustand "Auto_Producing" die "Slave_Axis" mit der winkelsynchronen Betriebsart synchron zur "Master_Axis" gestartet (durch ModeSyncPhase).

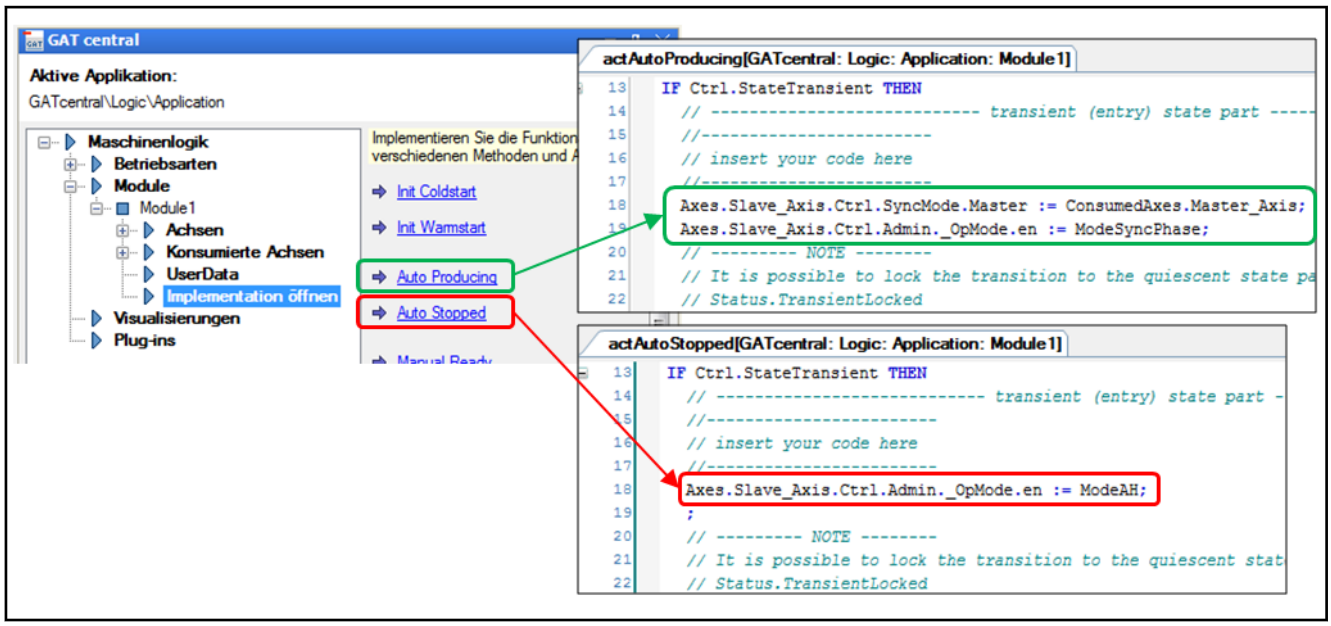


Abb.10-19: Implementierung des Codes in der Aktion des Zustands "Auto_Producing"

Inbetriebnahme Visualisierung Anpassen

Die neuen Achsnamen müssen in der Inbetriebnahmevisualisierung wie folgt angepasst werden:

- Inbetriebnahmevisualisierung über den GAT Wizard öffnen.

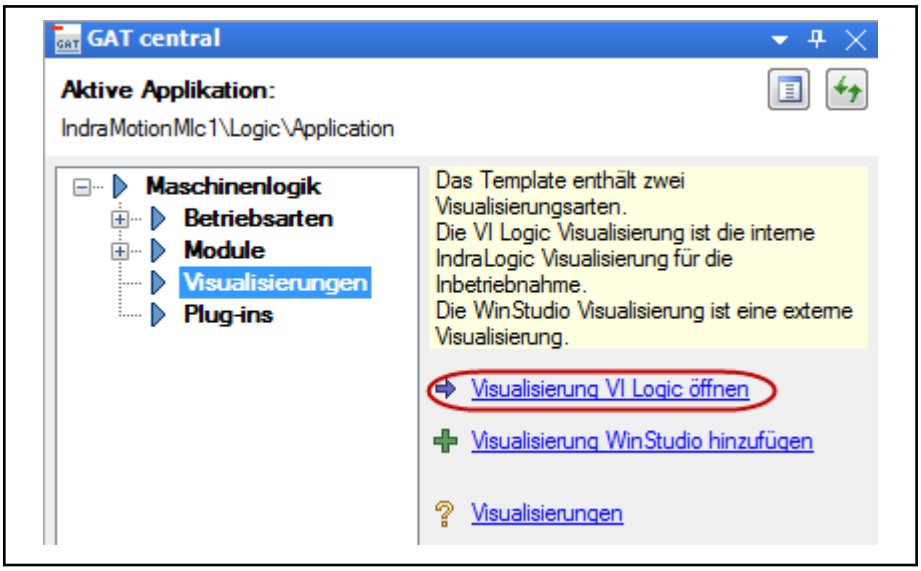


Abb.10-20: Öffnen der Inbetriebnahme Visualisierung über den GAT Wizard

- Das in der folgenden Abbildung markierte Visualisierungsobjekt anklicken und im Eingabefeld "Referenzierte Visualisierungen" den Modul- und Achsnamen wie folgt anpassen. Damit wirkt diese Achsvisualisierung auf die "MasterAxis" des Modules "Modul_Master".

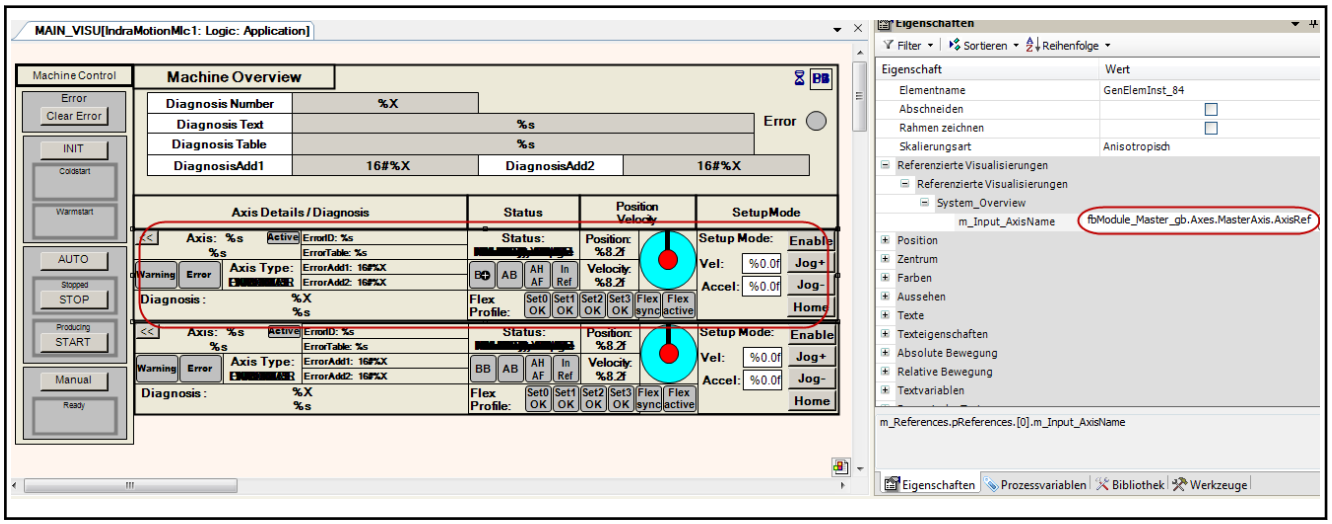


Abb. 10-21: Anpassung der Achsvisualisierung für die MasterAxis

Nachfolgend der Text für das Eingabefeld:

`fbModule_Master_gb.Axes.MasterAxis.AxisRef`

- Das in der folgenden Abbildung markierte Visualisierungsobjekt anklicken und im Eingabefeld "Referenzierte Visualisierungen" den Modul- und Achsnamen wie folgt anpassen. Damit wirkt diese Achsvisualisierung auf die "Slave_Axis" des Modules "Module1".

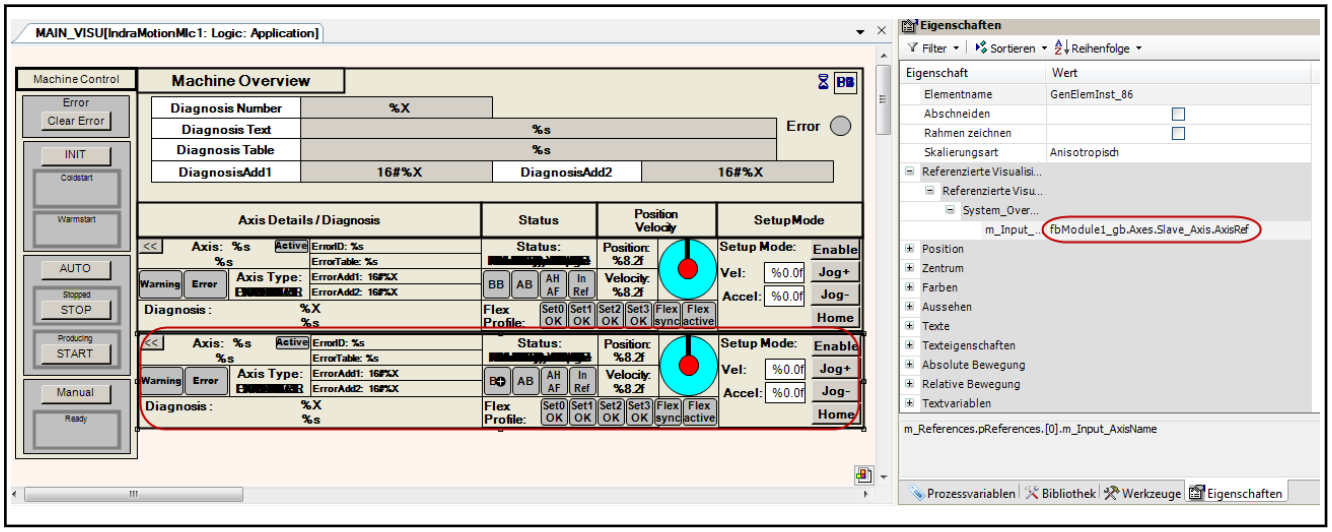


Abb. 10-22: Anpassung der Achsvisualisierung für die SlaveAxis

Nachfolgend der Text für das Eingabefeld:

`fbModule1_gb.Axes.Slave_Axis.AxisRef`

Die Programmierung ist damit abgeschlossen.

GAT mit zentraler Zustandsmaschine

Konfiguration und SPS Projekt Laden und Starten

Nach dem Herunterladen der Konfiguration (1) und des SPS-Projektes (2) können Sie die SPS starten (3):



Abb.10-23: Herunterladen (Konfiguration/SPS-Projekt) und Starten der SPS

Über die im vorherigen Schritt vorbereitete Inbetriebnahmevisualisierung "MAIN_VISU" können Sie jetzt die Betriebszustände beobachten und steuern.

Hierbei können Sie folgende Schritte durchführen, nachdem die Initialisierung durchlaufen wurde:

- Durch die Defaulteinstellung befindet sich die Steuerung nach der Initialisierung in der "AUTO" Betriebsart. Klicken Sie die Schaltfläche **START**. Dadurch wird der Zustandswechsel von "Stopped" nach "Producing" ausgelöst. Damit wird die Master-Achse in Geschwindigkeitsregelung gestartet und die Slave-Achse startet winkelsynchron zur Master Achse. Die Schaltfläche **START** schreibt hierbei auf die zuvor beschriebene globale Variable "bStart_gb".
- Klicken Sie die Schaltfläche **STOP**. Dadurch wird der Zustandswechsel von "Producing" nach "Stopped" ausgelöst. Damit werden Master- und Slave-Achse gestoppt. Die Schaltfläche **Stop** schreibt hierbei auf die globale Variable "bStop_gb".
- In dem "Stopped" Zustand kann die Betriebsart verändert werden. Klicken Sie hierzu auf die Schaltfläche **Manual**. In dieser Betriebsart können Sie über den SetupMode die Achsen vertippen oder referenzieren. Aktivieren Sie den SetupMode durch die jeweiligen Enable Buttons der Visualisierung. Anschließend können Sie über die Jog Buttons die Achse vertippen, und über den Home Button die Achse referenzieren.

10.4 Funktionsprinzip

Um die Motion-Applikation einfach und effektiv zu erstellen, ist der Programm-Code des GAT zweigeteilt. Ein Teil befindet sich in einer geschlossenen Bibliothek (ML_TechTemplate.library), der andere Teil ist offen zugänglich.

In dem offenen Teil platziert der Anwender mit Unterstützung des Wizards den für die Anwendung speziellen Code. Der Codeteil in der Bibliothek organisiert den Aufruf von Betriebsarten und Zuständen, Modulen und Achsen. Die Bibliothek arbeitet generisch, d. h. auch wenn neue Betriebsarten, Zustände, Module oder Achsen hinzukommen, sich deren Namen ändern oder diese gelöscht werden, ist die Konsistenz dieser Elemente sichergestellt.

Dieses Funktionsprinzip führt zu folgenden Vorteilen:

- Erstellung von kompaktem und übersichtlichem Anwendercode:

  -> Durch die Verwendung einfacher Schnittstellen und klarer Strukturen basierend auf einer optimierten Programmvorlage
- Erhöhung der Robustheit:

  -> Durch den Einsatz getesteter und optimierter Codeteile
- Verbesserung der Wartbarkeit:

  -> Durch den einfachen Austausch der Bibliothek im Falle von Fehlern

10.5 Initialisierung

Die Initialisierung wird zu Beginn des Maschinenhochlaufs durchlaufen. Nach erfolgreicher Initialisierung befindet sich die Maschine im Betriebsmodus.

Die Initialisierung erfolgt in zwei Schritten:

1. Coldstart: Schalten der Steuerung in den Parametriermodus (P2)
2. Warmstart: Schalten der Steuerung in den Betriebsmodus (BB)

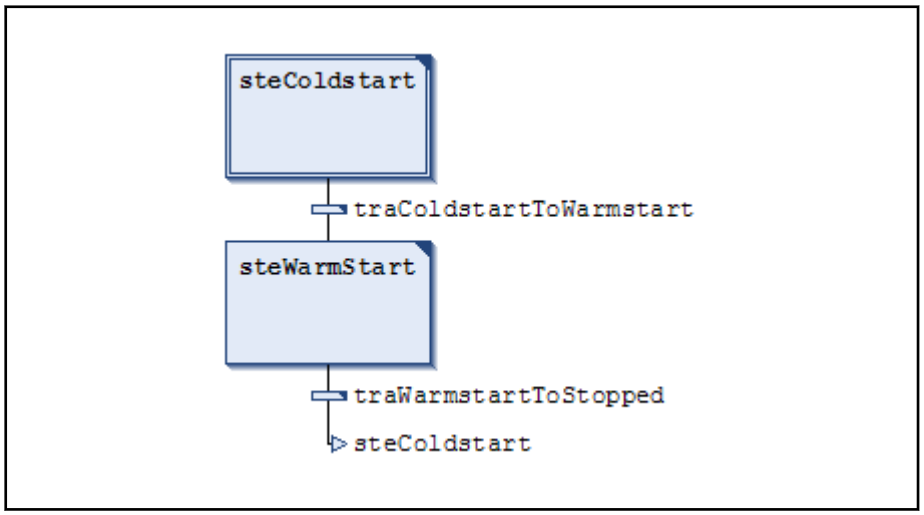


Abb.10-24: Zustandsmaschine des OpMode "_Init"

SkipColdstartAtStartup/mOneTimeInit

Nach SPS-Projekt Download bzw. einem SPS-Reset wird per Standard der Zustand "Coldstart" der Betriebsart "Init" aktiviert. Insbesondere bei der Maschinen-Inbetriebnahme kann dieses Verhalten störend sein, da das Umschalten auf "Coldstart = Parametermodus" zeitaufwändig ist. Über die Einstellung **"SkipColdstartAtStartup = TRUE"** kann daher für diese Fälle das Umschalten auf "Coldstart" vermieden werden. Bei dieser Einstellung wird nach dem SPS-Download bzw. Reset der "Coldstart" übersprungen, und nur der "Warmstart" bearbeitet. Wird jedoch die Betriebsart Init der Maschine über den Eingang "OpMode" vorgegeben, so wird auch bei der Einstellung "SkipColdStartAtStartup = TRUE" der "Coldstart" durchlaufen.

Veränderungen an der Einstellung **"SkipColdstartAtStartup"** können in der Zuweisung der Methode "mOneTimeInit" am Funktionsbaustein "Machine" vorgenommen werden (siehe nachfolgende Abbildung). Nach dem Download des modifizierten SPS-Projektes werden die neuen Einstellungen wirksam.

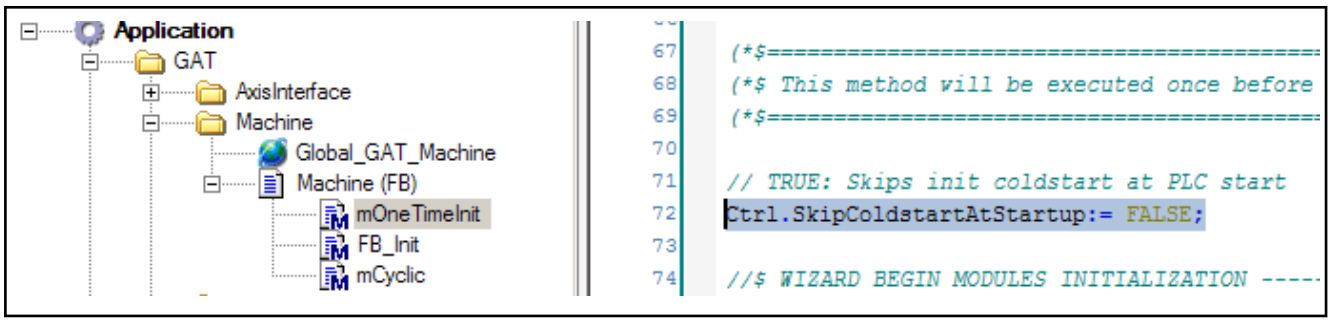


Abb.10-25: Konfiguration der Initialisierung über das Strukturelement "SkipColdstartAtStartup"

☞ In der Zustandsmaschine (SFC) für die Initialisierung (siehe "GAT/OpModes/System/_Init") dürfen keine Zustände/Transitionen entfernt werden!

GAT mit zentraler Zustandsmaschine

10.6 Betriebsarten

10.6.1 Allgemeines

Das GAT bringt bereits die vordefinierten Betriebsarten "_Init", "Auto" und "Manual" mit. Die Betriebsart "_Init" für die Initialisierung der Maschine wurde bereits im vorhergehenden Kapitel behandelt.

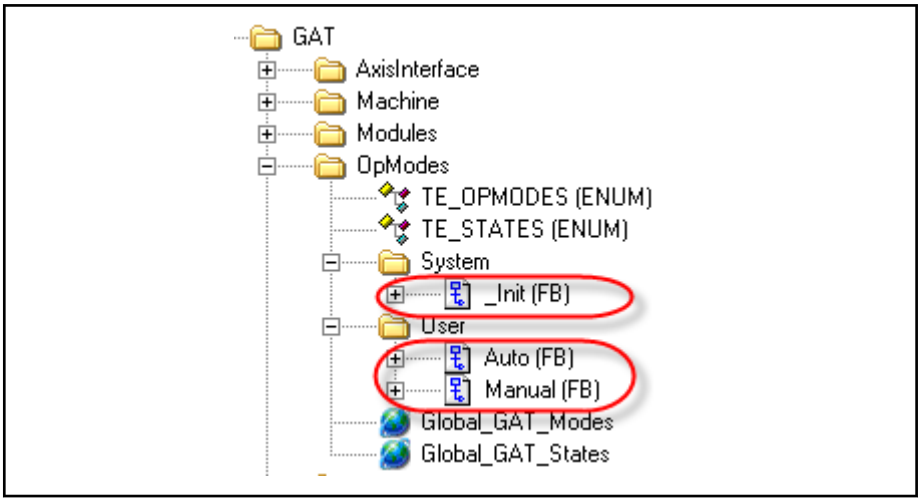


Abb.10-26: Konfigurierte Betriebsarten

Bei den Betriebsarten wird generell unterschieden zwischen

- **System-Betriebsarten**

  wie "_Init", die durch das System bereitgestellt und den Anwender nicht verändert werden können

- **User-Betriebsarten**

  wie "Auto", "Manual" die durch den Anwender modifiziert werden können

Alle Kap. 10.7 "Module" auf Seite 294 einer GAT-Anwendung befinden sich in der Regel in derselben Betriebsart.

In Abhängigkeit von der konkreten Applikation kann der Anwender bis zu 8 weitere User-Betriebsarten definieren (siehe Kap. 10.12.4 "Betriebsart hinzufügen" auf Seite 324).

Die Betriebsarten werden durch die Maschinen-Organisationseinheit in der ML_TechTemplate.library verwaltet. Es ist deshalb nicht nötig, dass der Anwender eine Betriebsart innerhalb des GAT aufruft.

10.6.2 Zustände einer Betriebsart

Um die Motion-Applikation übersichtlich zu strukturieren, ist es möglich, in einer Betriebsart mehrere Zustände zu definieren (siehe Kap. 10.12.5 "Anwahl der gewünschten Betriebsart nach Fehlerlöschen" auf Seite 324).

So ist zum Beispiel die vordefinierte User-Betriebsart "Auto" durch die Kombination der Zustände "Stopped" und "Producing" entsprechend der nachfolgenden Abbildung gekennzeichnet. Wenn es die konkrete Applikation erfordert, kann der Anwender weitere Zustände hinzufügen, umbenennen oder löschen.

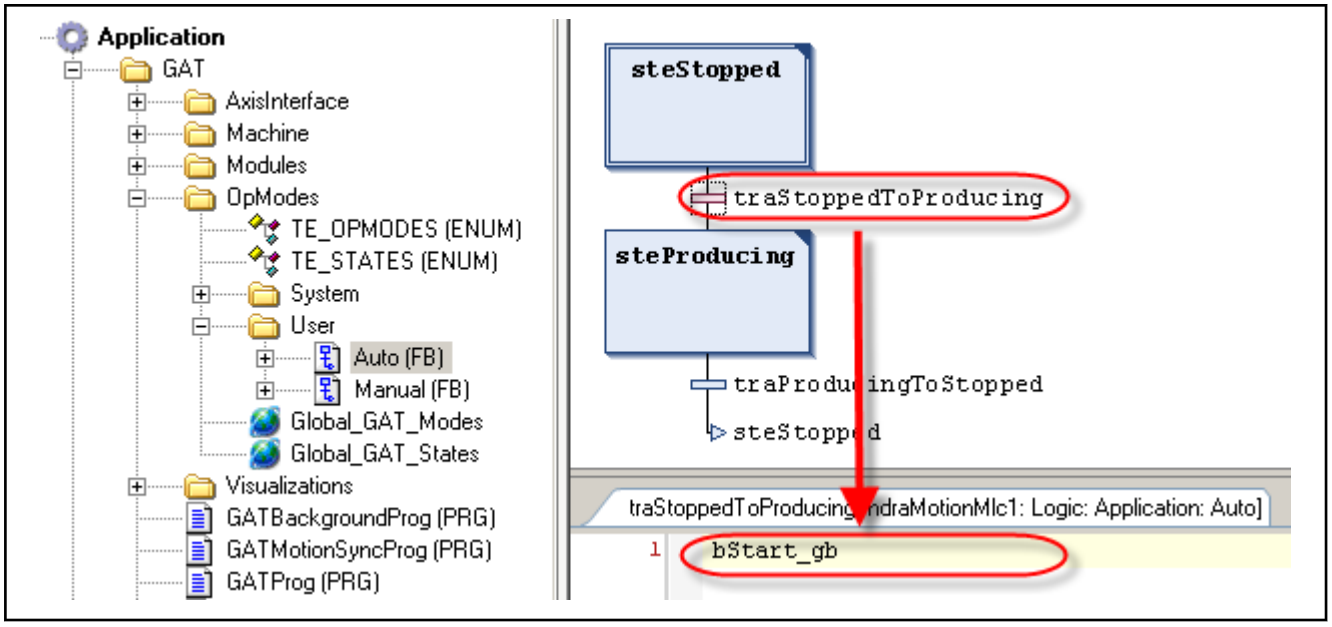


Abb.10-27: Zustände und Transitionen von Betriebsarten

Ein Zustandswechsel innerhalb einer Betriebsart erfolgt, wenn die Transitionsbedingung, die dem aktiven Zustand folgt, TRUE ist. Durch den Anwender können alle Transitionen mit einer applikationsspezifischen Bedingung versehen werden. So findet z. B. in der vordefinierten User-Betriebsart "Auto" ein Wechsel vom Zustand "Stopped" zum Zustand "Producing" statt, wenn der Zustand "Stopped" aktiv ist und die globale Variable "bStart_gb" den Wert TRUE hat (siehe vorhergehende Abbildung).

10.6.3 Einstellen der Zielbetriebsart nach Initialisierung

Mit der Variable **fbMachine_gb.Ctrl.Init_Successor** wird die Zielbetriebsart nach der Initialisierung festgelegt. Standard-Wert ist **OPMODE_AUTO**. Das GAT verbleibt in dieser Betriebsart bis über **fbMachine_gb.Ctrl.OpMode** eine andere Betriebsart kommandiert wird. Nachfolgende Abbildung zeigt, wo diese Elemente im GAT zu finden sind. Zur Erstellung weiterer Betriebsarten siehe Kap. 10.12.4 "Betriebsart hinzufügen" auf Seite 324.

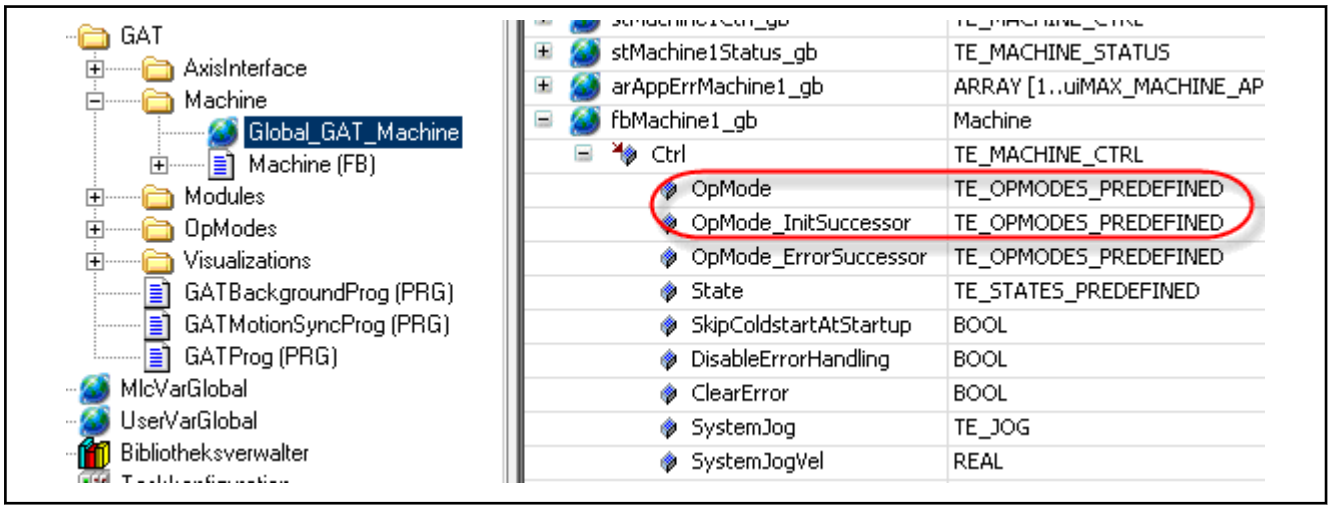


Abb.10-28: Einstellen der Zielbetriebsart nach Initialisierung

GAT mit zentraler Zustandsmaschine

10.6.4 Wechsel der Betriebsart

Ein Wechsel der Betriebsart wird eingeleitet, wenn **fbMachine_gb.Ctrl.OpMode** ein Wert (eine Betriebsart) zugewiesen wird, der sich vom Wert der Variable **fbMachine_gb.Status.OpMode** unterscheidet. Das ist aber nur möglich, wenn in der aktuellen Betriebsart der Initialzustand (STATE_<OPMODE>_STOPPED) aktiv ist. Dann erfolgt der Übergang vom Initialzustand der aktuellen Betriebsart zum Initialzustand der neuen Betriebsart.

mOpModeChangeAllowed

Optional kann durch die Methode "**mOpModeChangeAllowed**" des Funktionsbausteins "Machine" die Betriebsartenumschaltung blockiert werden. Diese Methode wird automatisch vom GAT aufgerufen, wenn ein Betriebsartenwechsel eingeleitet wird. Der Betriebsartenwechsel wird blockiert solange die Methode FALSE als Rückgabewert liefert. Liefert die Methode TRUE als Rückgabewert, so wird der Betriebsartenwechsel nicht blockiert (Standardverhalten).

Programm:

```
METHOD mOpModeChangeAllowed : BOOL
// Method is called in case of operation mode change
// mOpModeChangeAllowed := FALSE blocks the operation mode switching
// mOpModeChangeAllowed := TRUE allows the operation mode switching

// Declaration
VAR_INPUT
  OpMode:  TE_OPMODES_PREDEFINED;  // Current operation mode
END_VAR

// Implementation
 (*
IF (OpMode=OPMODE_AUTO) THEN
  // Switching from AUTO Mode to another mode is allowed only if the ServiceKey is ena
  mOpModeChangeAllowed :=  bServiceKeyEnabled;
END_IF
 *)

// END Implementation
```

☞ Eine Umschaltung in die Betriebsart "Init" (und damit Parametriermodus) ist systembedingt nur dann zulässig wenn alle Achsen den Zustand Ab bzw. AH quittieren. Vor Umschaltung von einer User-Betriebsart auf die Betriebsart "Init" müssen somit alle Achsen in den Zustand Ab bzw. AH kommandiert werden. Eine entsprechende Achskommandierung im transienten Teil des Initialzustandes "Stopped" wird daher empfohlen.

10.6.5 Anwenderaktivitäten bei Verlassen einer Betriebsart

mExitOpMode

In speziellen Anwendungsfällen müssen Aktivitäten bei Verlassen einer Betriebsart ausgeführt werden (z. B. zum Rücksetzen von SPS-Variablen). Für diesen Anwendungsfall sind bereits Funktionalitäten vorbereitet, die wie folgt genutzt werden können:

Zu jedem Modul-FB/Machine-FB kann die Methode "**mExitOpMode**" hinzugefügt werden. Diese Methode wird bei Verlassen einer Betriebsart für einen Zyklus vom GAT aufgerufen. Über den Eingang "OpMode" der Methode wird die verlassene Betriebsart von dem GAT übergeben.

Daher wird folgende Deklaration der Methode benötigt:

```
METHOD mExitOpMode
// Method is called for one cycle on operation mode exit

// Declaration
VAR_INPUT
 OpMode: TE_OPMODES_PREDEFINED; // Old operation mode
END_VAR
```

```
// Implementation
(*
CASE OpMode OF
 OPMODE_AUTO:
  ; // Activities in order to reset all variables of operation mode AUTO
 OPMODE_MANUAL:
  ; // Activities in order to reset all variables of operation mode MANUAL
ELSE
;
END_CASE
*)
// END Implementation
```

Innerhalb der Methode kann dann die Betriebsart abgefragt und entsprechende SPS-Variablen zurückgesetzt werden.

10.6.6 Anwenderaktivitäten bei Start einer Betriebsart

mEntryOpMode

In speziellen Anwendungsfällen müssen Aktivitäten bei Start einer Betriebsart ausgeführt werden (z. B. zum Setzen von SPS-Variablen). Für diesen Anwendungsfall sind bereits Funktionalitäten vorbereitet, die wie folgt genutzt werden können:

Zu jedem Modul-FB/Machine-FB kann die Methode "**mEntryOpMode**" hinzugefügt werden. Diese Methode wird bei Start einer Betriebsart für einen Zyklus vom GAT aufgerufen. Über den Eingang "OpMode" der Methode wird die neu gestartete Betriebsart vom GAT übergeben.

Daher wird folgende Deklaration der Methode benötigt:

```
METHOD mEntryOpMode
// Method is called for one cycle on operation mode entry

// Declaration
VAR_INPUT
 OpMode: TE_OPMODES_PREDEFINED;  // New operation mode
END_VAR

 // Implementation
 (*
 CASE OpMode OF
  OPMODE_AUTO:
   ; // Activities in order to set all variables of operation mode AUTO
  OPMODE_MANUAL:
   ; // Activities in order to set all variables of operation mode MANUAL
 ELSE
 ;
 END_CASE
 *)
 // END Implementation
```

Innerhalb der Methode kann dann die Betriebsart abgefragt und entsprechende SPS-Variablen zurückgesetzt werden.

10.6.7 Überbrücken der zentralen Betriebsart

Beim Betriebsartenwechsel werden in der Regel alle Module in die neue Betriebsart geschaltet. Unter Umständen kann es jedoch sinnvoll sein, dass nicht alle Module einer Maschine der zentral vorgegebenen Betriebsart folgen, weil z. B. Wartungsarbeiten am Modul durchgeführt werden.

mForceOpMode(), mReleaseOpMode()

Es ist möglich, ein oder mehrere Module in eine andere Betriebsart zu schalten, als durch **fbMachine_gb.Ctrl.OpMode** des Funktionsbausteins "Machine" vorgegeben wird. Hierfür stehen die Methoden "mForceOpMode()" und "mReleaseOpMode()" der Module zur Verfügung.

Beide Funktionen werden direkt am Modul aufgerufen:

```
fbModule1_gb.mForceOpMode(OPMODE_MANUAL); // Switch to OpMode "Manual"

fbModule1_gb.mReleaseOpMode(); // Switch back to global OpMode
```

Das manuelle Schalten einzelner Module ist nur in einer benutzerdefinierten Betriebsart möglich. Bei der Umschaltung der Betriebsart gelten dieselben Regeln, wie beim Umschalten der zentralen Betriebsart der Maschine (siehe Kap. 10.6.4 "Wechsel der Betriebsart" auf Seite 292). Die Umschaltung in die angegebene Betriebsart erfolgt also erst, wenn ein Schalten der Betriebsart durch die Machine erlaubt wird.

Behandelt die Maschine einen Fehler sind immer alle Module betroffen. Beim Durchlaufen der Initialisierung (OpMode _Init), wird die manuelle Überbrückung der Betriebsart für alle Module aufgehoben.



Das Überbrücken der zentralen Betriebsart kann zu ungewollten Veränderungen im Verhalten der Gesamtmaschine führen, wenn Abhängigkeiten zwischen den einzelnen Modulen bestehen.

Name	Parameter	Rückgabewert	Beschreibung
mForceOpMode	OpMode	BOOL	Versetzt das Modul in die in OpMode angegebene Betriebsart
mReleaseOpMode		BOOL	Versetzt das Modul in die Betriebsart der Maschine

Abb.10-29: Funktionen zum manuellen Überbrücken der Betriebsart eines Moduls

10.7 Module

10.7.1 Allgemeines

Ein Modul im GAT repräsentiert eine funktionale Einheit der Maschine.

Es ist nicht nötig, dass der Anwender ein Modul innerhalb des GAT aufruft. Dies erfolgt automatisch durch in der ML_TechTemplate-Bibliothek lokalisierte Maschinen-Organisationseinheit.

Kap. 10.12.8 "Neues Modul hinzufügen" auf Seite 326 beschreibt, wie unter Verwendung des Wizards Module hinzugefügt und entfernt werden können.

Das bereitgestellte GAT enthält bereits das Modul **Module1** (siehe nachfolgende Abbildung). Dieser Name kann unter Verwendung des GAT-Wizard einfach geändert werden (siehe Kap. 10.12.8 "Neues Modul hinzufügen" auf Seite 326).

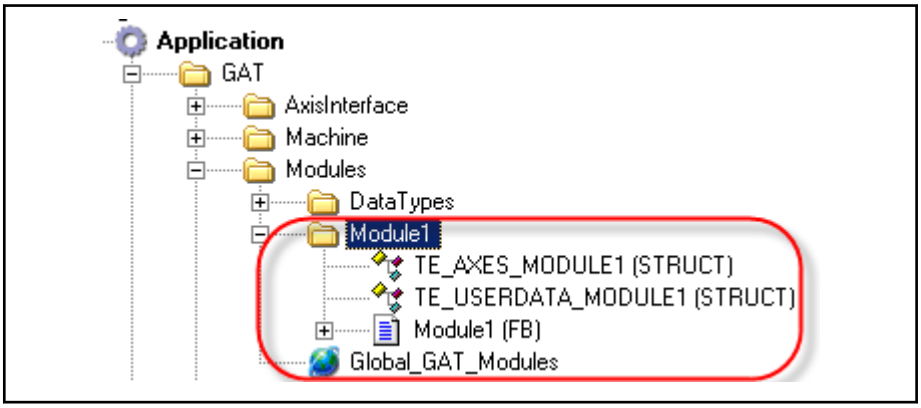


Abb.10-30: Modulordner im Projektverzeichnis

10.7.2 Regeln zur Erstellung

Um eine einfache Wartung und die Wiederverwendbarkeit eines GAT-Moduls zu erreichen, soll es so gestaltet werden, dass:

- Abhängigkeiten zwischen der Modulen gering gehalten werden
- innerhalb eines Moduls keine globalen Variablen verwendet werden

Um das zu erreichen, stellt jedes Modul die nachfolgende Schnittstelle bereit.

10.7.3 Modulschnittstelle

Allgemeines

Nachfolgende Abbildung zeigt die einheitliche Modulschnittstelle.

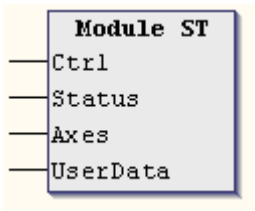


Abb.10-31: Modulschnittstelle

Um eine effektive Datenhaltung bzw. performantes Datenhandling zu ermöglichen, sind alle Elemente der Modulschnittstelle als Referenzen ausgeführt, d. h. die im Modul benötigten Daten werden nicht in das Modul kopiert, sondern per Referenz an das Modul übergeben.

Im Folgenden werden die Bestandteile dieser Schnittstelle beschrieben.

Control- und Status-Struktur

Jedes Modul verfügt über eine Variable zum Steuern (**Ctrl**) und eine Variable zur Statusmeldung (**Status**) (siehe nachfolgende Abbildung).

```
Ctrl:   REFERENCE TO TE_CMI_CONTROL;
Status: REFERENCE TO TE_CMI_STATUS;
```

Abb.10-32: Modul Control- und Status-Struktur

Name und Typ dieser Variablen sind in jedem Modul identisch. Kap. 10.11.4 "Daten zu den Modulen" auf Seite 318 gibt einen Überblick, welche Elemente zu diesen Strukturen gehören, welche von diesen Elementen durch den Anwender genutzt werden können und welche Elemente der internen Organisationseinheit der Maschine vorbehalten sind.

Modulachsen

Zur Schnittstelle jedes Moduls gehört ein Achs-Container (**Axes**). Alle Achsen dieses Containers werden in dem Modul aktiv kommandiert.



Passiv genutzte Achsen (wie z. B. Leitachsen) sollten aus Performancegründen **nicht** im Achs-Container **Axes** konfiguriert werden.

Konfigurieren Sie **passiv** genutzte Achsen im **Consumed Axes** Achs-Container.

Der Name des Achs-Containers ist für jedes Modul gleich, der Inhalt (Typ) aber unterschiedlich, da nicht in jedem Modul dieselben Achsen verwendet werden.

GAT mit zentraler Zustandsmaschine

Jede Achse im Achs-Container (Modulachse) ist vom Typ TE_GAT_AXIS, welcher auf dem Typ TE_BASE_MODULE_AXIS basiert (siehe Bibliothek ML_TechTemplate). Die nachfolgende Tabelle zeigt alle Elemente, die im Strukturtype TE_GAT_AXIS verfügbar sind.

Element	Typ	Beschreibung
Name	STRING(20)	Name der Achse als Zeichenkette (wird bei der Initialisierung des Systems initialisiert, kann durch den Anwender modifiziert werden)
AxisData	REFERENCE TO ML_AXISDATA_SM	Referenz auf das entsprechende Element im globalen Array **AxisData** (siehe Bibliothek **ML_Base**)
DV	REFERENCE TO ML_DirectVarAxis	Im Falle einer lokalen Achse: Referenz auf das entsprechende Element im globalen Array **DV_Axis** (siehe Bibliothek **ML_Base**) Im Falle einer Verbundachse (Achse einer anderen Steuerung): Dummyelement
GV	REFERENCE TO ML_GlobalVarAxis	Referenz auf das entsprechende Element im globalen Array **GV_Axis** (siehe Bibliothek **ML_Base**)
AxisRef	AXIS_REF	Achsreferenz
Disable	BOOL	TRUE: Der entsprechende Axis-Interface-Datensatz wird nicht mehr aktualisiert FALSE: Standard **Eine Änderung des Wertes dieser Variable wird erst wirksam im INIT Mode (STATE_INIT_COLDSTART oder STATE_INIT_WARMSTART)**
MotionSync	BOOL	TRUE: Die Achsdaten dieser Achse werden in der Motion-Task aktualisiert FALSE: Die Achsdaten dieser Achse werden in der PlcTask aktualisiert
Ctrl	REFERENCE TO TE_AXISINTERFACE_CONTROL	Referenz auf die Steuerstruktur des Achs-Interfaces der entsprechenden Achse (Referenz auf den entsprechenden Eintrag des globalen Arrays **arAxisCtrl_gb**, siehe globale Variablenliste **Global_GAT_AxisInterfaces** im Ordner **AxisInterface**
Status	REFERENCE TO TE_AXISINTERFACE_STATUS	Referenz auf die Statusstruktur des Achs-Interfaces der entsprechenden Achse (Referenz auf den entsprechenden Eintrag des globalen Arrays **arAxisStatus_gb**, siehe globale Variablenliste **Global_GAT_AxisInterfaces** im Ordner **AxisInterface**

Abb.10-33: Alle mit dem Type TE_GAT_AXIS verfügbaren Elemente. Innerhalb des Moduls erfolgt der Zugriff auf ein Element über Axes.<AxisName>.<Element>

Die Optionen, eine Achse innerhalb der GAT-Programmierumgebung zu deaktivieren (Element **Disable**) oder der MotionTask zuzuweisen (Element **MotionSync**), können innerhalb des GAT-Wizard gewählt werden, wenn eine Modulachse definiert oder umbenannt wird (siehe Kap. 10.12.3 "Achse entfernen/umbenennen" auf Seite 322).

> Eine Achse muss der MotionTask zugewiesen werden, wenn eine Aktivität innerhalb der MotionTask läuft, die diese Achse verwendet!

Über den Achs-Container **Axes** sind alle relevanten Daten einer Achse adressierbar.

Der Container ist so aufgebaut, das über die Namen der Achsen auf die Achsdaten zugegriffen wird und damit gut lesbarer Programmcode entsteht.

Nachfolgende Tabelle zeigt, wie man auf Achsen im Achs-Container innerhalb und außerhalb eines Moduls zugreift.

Zugriff auf eine Modulachse (z. B. Achse "vAxis" im Achs-Container des Moduls "Module1")	Aktualisierung der Achsdaten der Motion-Task zuweisen	Disable Achse
Zugriff außerhalb des Moduls	stAxesModule1_gb.vAxis.MotionSync := TRUE;	stAxesModule1_gb.vAxis.Disable := TRUE;
Zugriff innerhalb des Moduls	Axes.vAxis.MotionSync := TRUE;	Axes.vAxis.Disable := TRUE;

Unter Nutzung des Wizard wird in der Methode "mOneTimeInit" (diese Methode wird während der Initialisierung des GAT nur einmal abgearbeitet) der übergeordneten Maschine eine physische Achse (reale, virtuelle,…) der entsprechen Achse im Modul zugewiesen (siehe nachfolgende Abbildung und Kap. 10.12.2 "Achse hinzufügen" auf Seite 322).

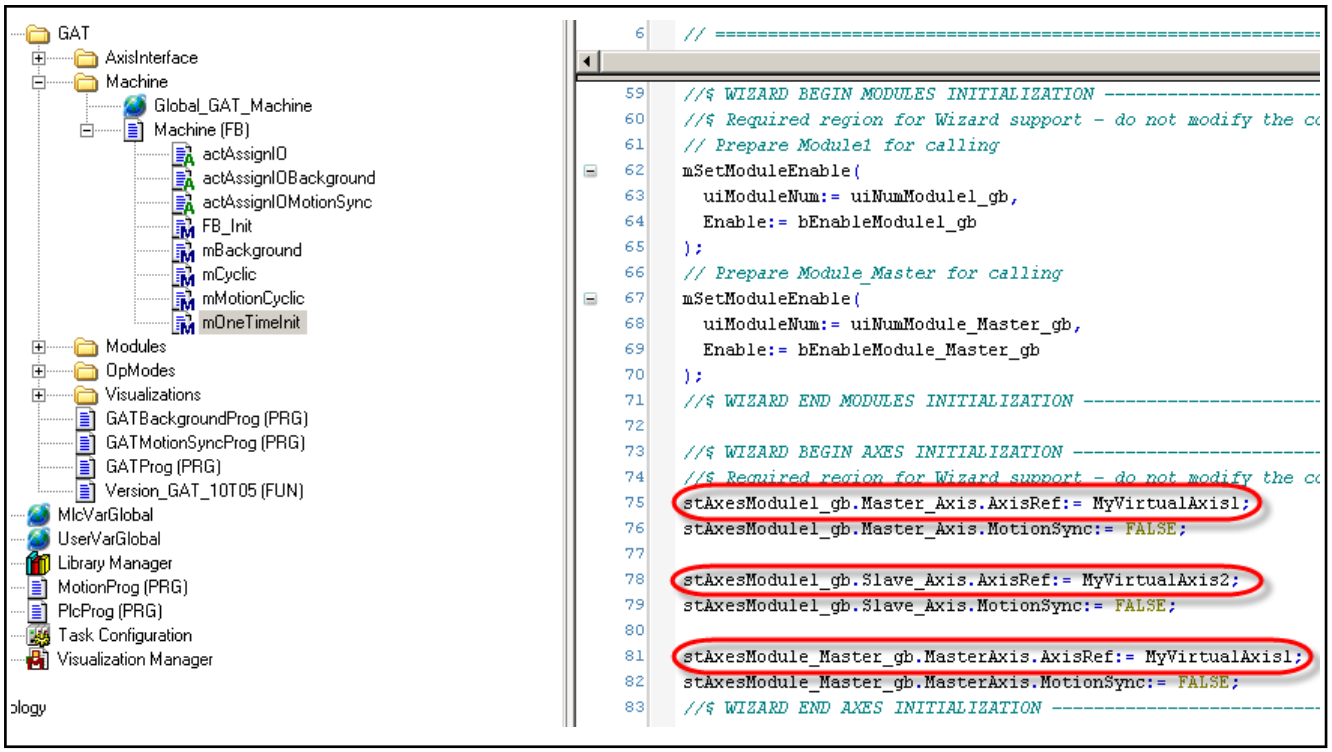


Abb.10-34: Achszuweisung in der Methode "mOneTimeInit"

Ein Beispiel aus der oberen Abbildung:

Die globale, in IndraWorks erzeugte Achse "MyVirtualAxis1" wird der Achse "Master_Axis" des Moduls "Module1" zugewiesen.

Passive Achsen (Consumed Axes)

Konfigurieren Sie alle passiv genutzten Achsen (wie z. B. Leitachsen) eines Moduls in dem **ConsumedAxes** Achs-Container.

GAT mit zentraler Zustandsmaschine

Führen Sie hierzu mit dem GAT Wizard folgende Schritte aus:

- Öffnen Sie den GAT-Wizard (GAT-Button  in der IndraWorks Toolbar)
- Selektieren Sie den **Consumed Axes** Knoten unterhalb des Moduls
- Funktion "Achse hinzufügen" wählen und den Dialogabfragen folgen

Als passiv genutzte Achsen können auch Encoder-, Link- und C2C Consumed-Achsen verwendet werden.

*Durch die Nutzung des **Consumed Axes** Achs-Containers werden folgende Vorteile erreicht:*

- **Performancevorteil**, da das AxisInterface für die passiven Achsen nicht aufgerufen wird
- **Erhöhte Benutzerfreundlichkeit**, da schneller ersichtlich wird, in welchem Modul welche Achsen aktiv kommandiert werden

Beispiel für die Nutzung einer Consumed-Achse im Programmcode eines Modules:

```
// Example: MySlaveAxis1 will be synchrozied with VirtualMaster 1 consumed axis
Axes.MySlaveAxis1.Ctrl.SyncMode.Master := ConsumedAxes.VirtualMaster1;
Axes.MySlaveAxis1.Ctrl.Admin._OpMode.en := ModeSyncPhase;
```

Bei einer C2C Consumed-Achse können die Achsdaten (wie z. B. Position, Geschwindigkeit) über die Variable **GV_LinkAxis[]** in dem Modul gelesen werden.

Bei allen anderen Achstypen können die Achsdaten der Consumed-Achse über die Variable **DV_Axis[]** gelesen werden.

Eine Consumed-Achse entspricht dem Datentyp AXIS_REF mit den Elementen "CntrlNo" und "AxisNo".

Modulspezifische I/O-Daten

Um ein Modul wieder zu verwenden, darf in dem Modul nicht auf globale Daten zugegriffen werden. Alle I/O, die im Modul benötigt werden, werden über den I/O-Container **UserData** zur Verfügung gestellt. Der Name des I/O-Containers ist für jedes Modul gleich, der Inhalt (Typ) aber unterschiedlich, da nicht in jedem Modul dieselben I/O-Daten verwendet werden.

Die Variable "UserData" ist eine Datenstruktur deren Elemente der Anwender definieren muss.

Nachfolgende Abbildung zeigt, wo die I/O-Daten für das im GAT enthaltene Modul "Module1" zu definieren sind:

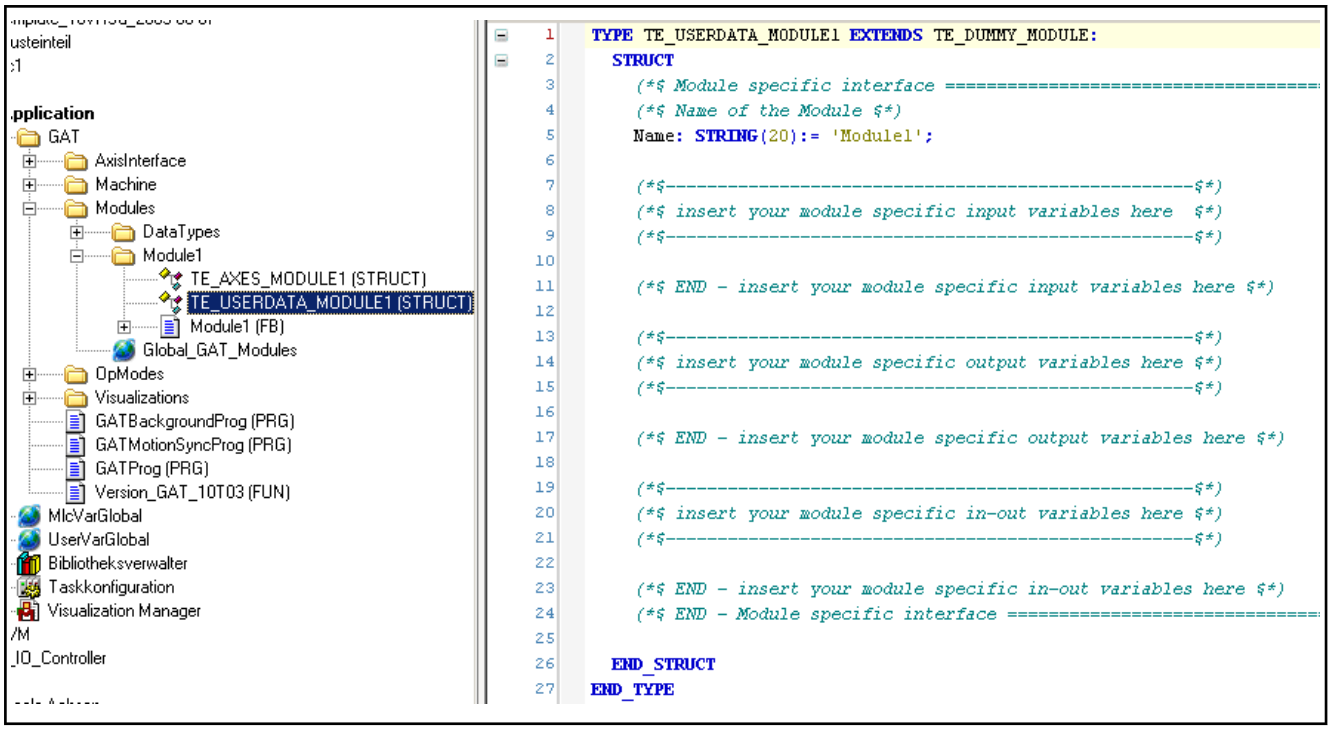


Abb. 10-35: Modulspezifische I/O-DATEN

Modulinstanzen

Von einem bestehenden Modul können weitere Instanzen angelegt werden, die sich nur durch die zugeordneten IndraWorks-Achsen und der zugeordneten I/O unterscheiden. Die Modul-Instanziierung wird vom GAT-Wizard unterstützt und ist in Kap. 10.12.9 "Weitere Instanz eines Moduls hinzufügen" auf Seite 327 beschrieben.

10.7.4 Zustände im Modul

Beim Hinzufügen oder Löschen eines Zustands innerhalb einer Betriebsart sorgt der Wizard dafür, dass in jedem Modul eine zustandskorrespondierende Aktion angelegt wird. Diese Aktion wird nur bearbeitet, wenn der entsprechende Zustand in der zugehörigen Betriebsart aktiv ist. So wird die Aktion "actAutoProducing" im Modul "Module1" bearbeitet, wenn den Zustand "Producing" in der Betriebsart "AUTO" aktiv ist.

Jede zustandsbezogene Aktion innerhalb eines Moduls kann unterteilt werden in einen **transienten** und einen **stationären** Teil.

Der transiente Teil der zustandsbezogenen Aktion wird bestimmt durch die Bedingung **Ctrl.StateTransient**. Hier sollte der Anwender die Aktivitäten koordinieren, die zeitlich begrenzt nötig sind, um diesen Zustand einzunehmen (z. B. Temperaturregler im Zustand AUTO_PRODUCING einschalten). Wenn das nicht in einem SPS-Zyklus möglich ist, kann der Anwender durch Setzten von **Status.TransientLocked:= TRUE** das Verbleiben in diesem Teil erreichen (z. B. bis Solltemperatur des Temperaturreglers im Zustand AUTO_PRODUCING erreicht ist).

Ist **Ctrl.StateTransient=FALSE** dann wird der stationäre Teil der zustandsbezogenen Aktion bearbeitet. Hier sollte der Anwender alle Aktivitäten codieren, die ausgeführt werden sollen, wenn dieser Zustand bereits eingenommen ist (z. B. taktsynchrones Ein- und Auskoppeln bei fehlenden Produkten). Durch

das Setzen von **Status.StateLocked:=TRUE** kann der Wechsel in einen anderen Zustand dieser Betriebsart verhindert werden, auch wenn die entsprechende globale Transitionsbedingung erfüllt ist.

10.7.5 Modulkoordination

Da innerhalb eines Moduls keine Informationen über andere Module vorhanden sind, muss an übergeordneter Stelle die Koordinierung der Module erfolgen.

Durch die Maschinen-Organisationseinheit der Bibliothek ML_TechTemplate.library wird die Methode **"mCyclic"** der Maschine in jedem SPS-Zyklus aufgerufen (siehe nachfolgende Abbildung). In dieser zyklisch aufgerufenen Methode sollen I/O-Informationen zwischen den Modulen und von den Modulen zu den I/O-Punkten der Maschine ausgetauscht werden.

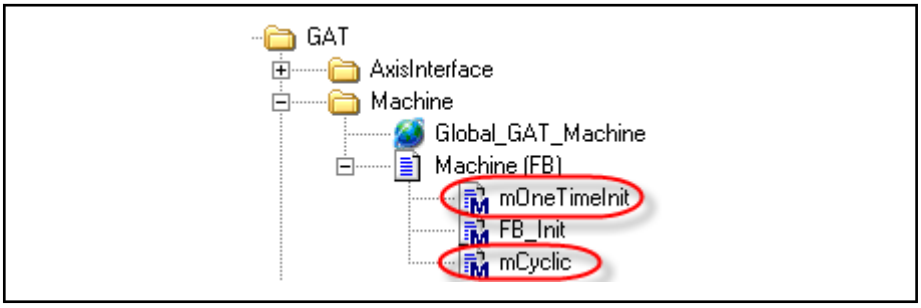


Abb.10-36: Methode "mCyclic" der Maschinen-Organisationseinheit

Da sich die Modulkonfiguration bzw. die Zuweisung einer globalen Achse an eine entsprechende Modulachse zur Laufzeit nicht ändert, erfolgen diese Aktivitäten in der Methode **"mOneTimeInit"** der Maschine. Diese Methode wird einmalig zur Initialisierung des GAT durch die Maschinen-Organisationseinheit aufgerufen (siehe vorhergehende Abbildung).

Der in der Methode "mOneTimeInit" enthaltene Code wird durch den Wizard verwaltet. Beim Hinzufügen oder Entfernen von Modulen oder Modulachsen werden entsprechende Einträge hinzugefügt oder entfernt.

10.8 Taskverwaltung

10.8.1 Taskkonfiguration GAT

Innerhalb der SPS laufen mehrere Tasks quasi parallel. Jede dieser Tasks führt eigenen SPS-Code aus. Die Abarbeitung der Tasks durch den Prozessor der Steuerung wird anhand von Taskprioritäten und der Zykluszeit ermittelt.

Im GAT sind die MotionTask, die PlcTask und die BackgroundTask standardmäßig enthalten.

	Prio	Zykluszeit	Watchdogzeit	Empfindlichkeit des Watchdog	Programmaufruf
GATMotionSyncTask	2	= Motion Zykluszeit	T#2ms	1	GATMotionSyncProg
GATProgTask	5	T#10ms	T#20ms	2	GATProg
GATBackgroundProgTask	6	T#40ms	-	-	GATBackgroundProg

10.8.2 GATMotionSyncTask

Um aus der MotionTask auf Achsen konsistent zuzugreifen, muss auch das Achs-Interface in der MotionTask aufgerufen werden. Alle Achsen in der MotionTask zu bearbeiten, würde die MotionTask unnötigerweise belasten. Deshalb kann beim Hinzufügen einer Achse über den GAT-Wizard die Abarbeitung des Achs-Interface der Plc- oder MotionTask zugeordnet werden.

Siehe auch Kap. "Typische Aufgaben MotionTask" auf Seite 42.

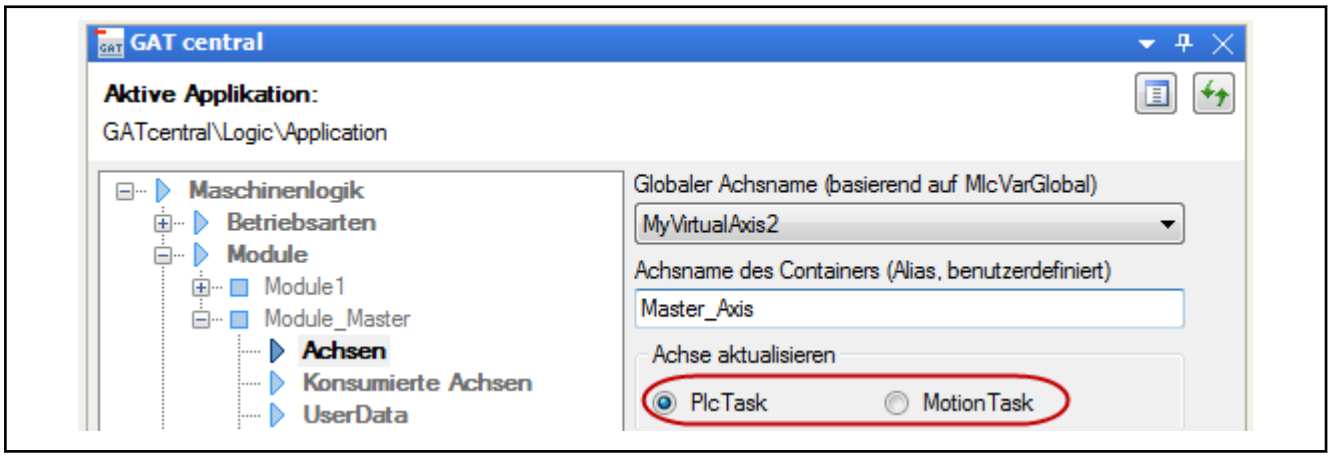


Abb. 10-37: Zuordnung der Abarbeitung des Achs-Interface

Grundsätzlich sollten die Achsdaten in der PlcTask aktualisiert werden.

Eine Achse muss der MotionTask zugewiesen werden, wenn eine Aktivität innerhalb der MotionTask läuft, die diese Achse verwendet! (z. B. Ein Prozessregler wird in der MotionTask bearbeitet. Dann müssen alle Achsen, die dem Prozessregler zugeordnet sind, auch in der MotionTask aktualisiert werden!)

Für die Abarbeitung in der MotionTask bringt jedes Modul/Maschine die Methode **"mMotionCyclic"** mit. Aus dieser Methode kann auf alle Variablen des Moduls/Maschine (z. B. zentraler Achseinstiegspunkt Axes….) zugegriffen werden.

10.8.3 GATBackgroundProgTask

Falls die Laufzeit der PlcTask zu groß wird, kann durch Auslagern von zeitunkritischen Aufgaben in eine Backgroundtask die PlcTask entlastet werden. Zu bedenken ist dabei, dass durch die zusätzlichen Taskwechsel PlcTask <> BackgroundTask das Gesamtsystem höher belastet wird (siehe auch Kap. "Typische Aufgaben BackgroundTask" auf Seite 42).

Für die Abarbeitung in der BackgroundTask bringt jedes Modul die Methode **"mBackground"** mit. Aus dieser Methode kann auf alle Variablen des Moduls zugegriffen werden.

10.9 Fehlerbehandlung

10.9.1 Fehlererkennung und Diagnose

Die Programmiervorlage GAT enthält vordefinierte Funktionalitäten zum Erkennen, Behandeln und Löschen von Fehlern an **einer zentralen Stelle**. Fehler der verschiedenen Fehlerquellen (wie z. B. Steuerungsfehler, Applikationsfehler, FB-Fehler…) werden automatisch durch den maschinenübergreif-

enden Funktionsbaustein "Machine" zusammengeführt und über die Statusstruktur des Funktionsbausteins "Machine" bereitgestellt.

Falls mehrere Fehler gleichzeitig anstehen, wird der schwerwiegendste angezeigt. Stehen gleichzeitig mehrere Fehler mit dem gleichen Schweregrad an, so wird der zuletzt erkannte Fehler ausgegeben.

Die Maschinen-Statusstruktur enthält die nachfolgend beschriebenen Strukturelemente für Diagnosezwecke:

Element	Typ	Beschreibung
Error	BOOL	TRUE signalisiert Fehler
DiagNo	DWORD	Nummer gemäß Dokumentation "Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnosen" mit den folgenden Erweiterungen: • 16#F0**AE** xxxx: Applikations-Fehler (z. B. geöffnete Schutztür) (xxxx = frei durch den Anwender definierbar) • 16#F0**FB** 0000: FB-Fehler (z. B. Fehler an einem TechFB) -> Details siehe ErrorID/ErrorIdent • 16#F0**AA** 0000: GAT interner Fehler -> Details siehe ErrorID/ErrorIdent • 16#F0**B1...B9** xxxx: Plugin Zusatzdiagnose Fehler (z. B. durch PROFIBUS Diagnose Plug-in)
ErrorID	ERROR_CODE	Detaildiagnose im Fall von FB + GAT Fehlern (DiagNo = 16#F0FB/AA 0000)
ErrorIdent	ERROR_STRUCT	Detaildiagnose im Fall von FB + GAT Fehlern (DiagNo = 16#F0FB/AA 0000)
DiagMessage	STRING(60)	Enthält bei Antriebs- und Steuerungsfehlern den Diagnosetext

Das Element **DiagNo** ist zur Nutzung an einem HMI vorgesehen. Die Fehlertexte werden dort der numerischen Information in beliebigen Sprachen zugeordnet.

DiagMessage, ErrorID und **ErrorIdent** sind zusätzliche Informationen die hauptsächlich bei der Inbetriebnahme durch den Programmierer, z. B. über eine IndraLogic-Visualisierung angezeigt werden können.

Hinweis zu applikationsspezifischen Fehlern

mSetAppError

Durch Aufrufen der Methode mSetAppError(Number) können Applikationsfehler (wie z. B. geöffnete Schutztür) abgesetzt werden. Die übergebene "Number" am Eingang der Methode entspricht hierbei der Applikationsfehlernummer.

Das zentrale Diagnosesystem im Modul/Maschine überwacht hierbei zyklisch die abgesetzten Applikationsfehler und erzeugt den Applikationsfehler DiagNo = Fehler 16#F0AE xxxx (xxxx = Applikationsfehlernummer) wenn ein Applikationsfehler abgesetzt wurde.

Für Applikationsfehler in Modulen sind Nummern im Bereich 1 … uiMAX_MODULE_APP_ERROR_INDEX gültig. Die Konstante "uiMAX_MODULE_APP_ERROR_INDEX" ist mit dem Wert 64 vorbelegt. Bei Bedarf kann diese Konstante in der Globalen Variablenliste "Global_GAT_Machine" unterhalb des "Machine"-Ordners angepasst werden.

Für Applikationsfehler in der Maschine sind Nummern im Bereich 1 … uiMAX_MACHINE_APP_ERROR_INDEX gültig. Die Konstante "uiMAX_MA-

CHINE_APP_ERROR_INDEX" ist mit dem Wert 64 vorbelegt. Bei Bedarf kann diese Konstante in der Globalen Variablenliste "Global_GAT_Machine" unterhalb des "Machine"-Ordners angepasst werden.

Codebeispiel zum Absetzen eines Applikationsfehlers:

```
// Recommended code to issue user specific errors (Example)
IF bDoor1Open THEN
    mSetAppError(1);
END_IF
```

Hinweis zu Fehler-Schweregrad

Das Steuerungs-Diagnosesystem definiert 10 Schweregrade "Severity" (siehe IndraMotion MLC-Diagnosesystem). Der Schweregrad hat Einfluss auf die Fehlerreaktion (siehe Kap. 5.6.4 "Fehlerreaktion" auf Seite 47).

- DiagNo = 16#F0xx xxxx ist der geringste Schweregrad (z. B. Applikationsfehler, FB-Fehler)
- DiagNo = 16#F9xx xxxx ist der höchste Schweregrad (fataler Systemfehler)

Es wird der Fehler mit dem höchsten Schweregrad angezeigt wenn mehrere Fehler gleichzeitig anstehen.

Hinweis zu Funktionsbaustein-Fehler

mSetFBError

Treten Fehler an aufgerufenen Funktionsbausteinen (z. B. Technologie-FB) auf, so wird empfohlen die Methode "mSetFBError(ErrorID, ErrorIdent)" aufzurufen und die "ErrorID" und "ErrorIdent" Ausgänge des Fehler verursachenden FB an die Methode zu übergeben. Damit wird sichergestellt, dass der Fehler an das Diagnosesystem des Moduls/Maschine weitergeleitet wird und die "ErrorID" und "ErrorIdent" an den Diagnoseausgängen des Moduls/Maschine bereitgestellt wird.

Codebeispiel zum Absetzen eines Funktionsbaustein-Fehlers:

```
// Recommended code if a used FB issues an error (Example)
IF fbCrossCutter.Error THEN
    mSetFBError(ErrorID:=fbCrossCutter.ErrorID,
                ErrorIdent:=fbCrossCutter.ErrorIdent);
END_IF
```

10.9.2 Fehlerreaktion

Die Fehlerreaktion hat die Aufgabe Schäden aufgrund von Fehlern des Steuerungs- und Anriebssystemes möglichst zu vermeiden. Daher wird bei Achs- und Steuerungsfehlern ab Schweregrad 2 automatisch eine Fehlerreaktion durch die Steuerungsfirmware und die realen Achsen durchgeführt (siehe Kap. 5.6.4 "Fehlerreaktion" auf Seite 47).

Bei allen anderen Fehlern mit kleineren Schweregraden (wie z. B. Applikationsfehler, FB-Fehler…) wird keine automatische Fehlerreaktion durch die Steuerungsfirmware und die realen Achsen durchgeführt.

Daher enthält das GAT eine Funktionalität, um Fehler zu erkennen und eine anwenderspezifische Fehlerreaktion durchzuführen.

Allgemeine GAT Fehlerreaktion

Mit der Standardeinstellung ist die GAT Fehlerreaktion eingeschaltet. Im Fehlerfall wird dadurch die angewählte Betriebsart unterbrochen (Ausnahme: Initialisierung) und die Fehlerreaktion ausgeführt. Die Fehlerreaktion ist in der Aktion "actError" in jedem Modul des GAT enthalten. Hierbei ist jedes Modul für die Fehlerreaktion der zugehörigen Komponenten (wie z. B. Achsen) verantwortlich.

Standardfall für die Fehlerreaktion in der Modulaktion actError (alle Modulachsen verzögern):

```
// ------------------------ Error reaction example 1 ---------------------
// Decelerate all module axes to standstill --> switch all axes to ModeAH
// ------------------------------------------------------------------------
FOR uiLoopIndex := 1 TO Axes.uiAxesNum DO
  IF mGetAxis(AxisNumber:=uiLoopIndex, PointerToAxis => pCurrentAxis) THEN
    IF pCurrentAxis^.AxisData.Axis_InAb THEN
      pCurrentAxis^.Ctrl.Admin._OpMode.en := ModeAH;
      // reset SetupMode Inputs to avoid hanging buttons in the
      // System_Overview_All_Axis visualisation
      pCurrentAxis^.Ctrl.SetupMode.Enable := FALSE;
      pCurrentAxis^.Ctrl.SetupMode.JogMinus := FALSE;
      pCurrentAxis^.Ctrl.SetupMode.JogPlus := FALSE;
      pCurrentAxis^.Ctrl.SetupMode.Homing := FALSE;
    END_IF
  END_IF
END_FOR
```

Wenn in dem obigen Beispiel "ModeAH" durch "ModeAb" ersetzt wird, dann erfolgt nach der Verzögerung im Stillstand die Momentenfreischaltung der realen Achsen.



Für hydraulische Achsen wird eine Stillsetzung über "ModeAb" empfohlen.



Bei **"hängenden Achsen"** darf keine Momentenfreischaltung erfolgen, um ein Absenken der Achsen zu vermeiden. Bei diesen Anwendungen wird empfohlen, alle Achsen in "ModeAH" zu schalten.

Oftmals besteht die Anforderung, Achsen im **Synchronverbund** zu verzögern, um z. B. das Reißen einer Materialbahn zu vermeiden. Bei diesen Anwendungen wird empfohlen, alle Achsen im Synchronverbund zu belassen, und nur den Master zu verzögern. Dieser Anwendungsfall ist im folgenden Beispiel dargestellt:

*Beispiel für **synchrone Stillsetzung** der Achsen in der Modulaktion "actError" (nur nicht synchrone Modulachsen werden über "ModeAH" verzögert):*

```
// ------------------ Error reaction example 2 ---------------------
// Keep all axes in synchronization with the master axis
// and decelerate only the master axis
// -----------------------------------------------------------------
FOR uiLoopIndex := 1 TO Axes.uiAxesNum DO
  IF mGetAxis(AxisNumber:=uiLoopIndex, PointerToAxis => pCurrentAxis) THEN
    // reset SetupMode Inputs to avoid hanging buttons in the
    // System_Overview_All_Axis visualisation
    pCurrentAxis^.Ctrl.SetupMode.Enable := FALSE;
    pCurrentAxis^.Ctrl.SetupMode.JogMinus := FALSE;
    pCurrentAxis^.Ctrl.SetupMode.JogPlus := FALSE;
    pCurrentAxis^.Ctrl.SetupMode.Homing := FALSE;
    IF (pCurrentAxis^.AxisData.SynchronizedMotion) THEN
      ; // Axis is in sync motion --> no reaction as the master axis
        // decelerates to standstill and the slave axis follows in sync
    ELSE
      IF pCurrentAxis^.AxisData.Axis_InAb THEN
        pCurrentAxis^.Ctrl.Admin._OpMode.en := ModeAH;
        ; // Axis is not in sync motion --> command to stop
      END_IF
    END_IF
  END_IF
END_FOR
```

Für die synchrone Stillsetzung sind bei realen Achsen folgende zusätzliche Einstellungen notwendig:

- Modulbusreaktion abschalten

- Fehlerreaktion der Antriebe mit "Bestmöglichen Stillsetzung = 30s NC Reaktion" konfigurieren

---

☞ Bei der Einstellung "Bestmöglichen Stillsetzung = 30s NC Reaktion" einer realen Achse wird bei dem IndraLogic-XLC/IndraMotion MLC-Steuerungssystem die fehlerbehaftete Achse bei einem F2-Fehler mit der Notstopp-Rampe A-0-0224 verzögert. Alle anderen fehlerfreien Achsen bleiben unbeeinflusst.

Abschalten der allgemeinen GAT Fehlerreaktion

*Für Sonderfälle kann die allgemeine GAT Fehlerreaktion abgeschaltet werden. Ändern Sie dazu die Standardeinstellung der Variablen "**stMachine1Ctrl_gb**" über die Variablendeklaration wie folgt:*

- Globale Variablenliste: "Global_GAT_Machine"
- Bestehende Deklaration der Variablen stMachine1Ctrl_gb mit folgendem Code ersetzen:

```
stMachine1Ctrl_gb: TE_MACHINE_CTRL := (DisableErrorHandling := TRUE);
```

Bei abgeschalteter Fehlerreaktion wird die aktuelle Betriebsart nicht unterbrochen. Fehler werden jedoch weiterhin angezeigt, und können auch gelöscht werden.

GAT Fehlerreaktion während der Initialisierung

Mit der Standardeinstellung ist die GAT Fehlerreaktion während der Initialisierung abgeschaltet. Auftretende Fehler werden jedoch angezeigt und können gelöscht werden.

Für Sonderfälle kann die Fehlerreaktion während der Initialisierung eingeschaltet werden.

*Ändern Sie dazu die Standardeinstellung der Variablen "**stMachine1Ctrl_gb**" über die Variablendeklaration wie folgt:*

- Globale Variablenliste: "Global_GAT_Machine"
- Deklaration wie folgt anpassen:

```
stMachine1Ctrl_gb: TE_MACHINE_CTRL := (EnableErrorReactionInit := TRUE);
```

Wenn die Fehlerreaktion eingeschaltet ist, dann wird die Initialisierung bei Fehlerauftritt unterbrochen. Nachfolgend wird dann die programmierte Fehlerreaktion ausgeführt. Die Fehlerreaktion ist in der Aktion "**actError**" der jeweiligen "Module" programmiert.

☞ Die Fehlerreaktion während der Initialisierung wird nur dann ausgeführt, wenn die allgemeine GAT Fehlerreaktion eingeschaltet ist (DisableErrorHandling = FALSE).

Fehlerfilterung

Nicht jeder Fehler an der Maschine muss zwangsläufig zum Stillsetzen der gesamten Anlage führen. Fehler können daher optional zentral durch die Maschine sowie dezentral durch die einzelnen Module behandelt werden. Hierfür steht die Methode "mIsCriticalError" am Funktionsbaustein "Machine" und an jedem Modul-Funktionsbaustein zur Verfügung.

Wird von der der Maschine ein Fehler erkannt, so werden die folgenden Schritte abgearbeitet:

1. Die Diagnosenummer (DiagNo) und ggf. die Achse, die den Fehler verursacht hat werden ermittelt.
2. Es wird mIsCriticalError (DiagNo, AxisRef) des Funktionsbausteins "Maschine" aufgerufen.

   Gibt mIsCriticalError() FALSE zurück, wird der Fehler ignoriert.

Gibt mIsCriticalError() TRUE zurück, wird im nächsten Schritt fortgesetzt.

3. An jedem aktiven Modul wird mIsCritialError(DiagNo, AxisRef) aufgerufen.

   Geben alle Module für diesen Aufruf FALSE zurück, wird der Fehler ignoriert.

   Gibt mindestens ein Modul TRUE zurück, wird die normale Fehlerbehandlung fortgesetzt.

Standardmäßig gibt "mIsCritalError()" TRUE zurück, d. h. jeder Fehler wird als kritisch eingestuft. Durch Implementieren der Methode "mIsCriticalError" unterhalb des Funktionsbausteins "Machine" oder an den Modul-Funktionsbausteinen kann das Verhalten beeinflusst werden.

Beispiel: Implementierung der Methode "mIsCriticalError()" am Funktionsbaustein "Machine":

```
METHOD mIsCriticalError : BOOL
VAR_INPUT
    DiagNo: DWORD;     // diagnositic number
    Axis: AXIS_REF;    // axis
END_VAR

CASE DiagNo OF
    // My uncritical errors
    16#F0AE0001,
    16#F0AE0002: mIsCriticalError := FALSE;
ELSE
    // All other errors
    mIsCriticalError := TRUE;
END_CASE
```

Beispiel: Implementierung der Methode mIsCriticalError() am einem Modul-Funktionsbaustein:

```
METHOD mIsCriticalError : BOOL
VAR_INPUT
    DiagNo: DWORD;     // diagnositic number
    Axis: AXIS_REF;    // axis
END_VAR

mIsCriticalError := TRUE;

IF Axis.AxisNo = 0 THEN
    // error with no axis supplied
    mIsCriticalError := FALSE;
ELSIF mIsMyAxis(Axis) THEN
    // error at my own axis
    IF DiagNo=16#FFAA0002 THEN
        mIsCriticalError := FALSE;
    END_IF
END_IF
```

10.9.3 Fehler löschen

Über eine positive Flanke am Strukturelement **"ClearError"** der Maschinen-Steuerstruktur wird das zentrale Fehlerlöschen automatisch durchgeführt. Der FB "Machine" leitet dieses Kommando an alle unterlagerten Komponenten (wie z. B. Imc-Interface[1)] , Module, AxisInterface …) weiter um das gesamte System in den fehlerfreien Zustand zu überführen. Durch das Strukturelement "enOpModeCtrl_ErrorSuccessor" (siehe Steuerstruktur der Maschine) wird bestimmt, welche Betriebsart nach Fehlerlöschen aktiv wird.

1) *IndraMotionControl Interface = IndraLogic Bibliothek mit Schnittstelle zur Steuerung, wie. z. B. Fehler löschen*

10.9.4 Fehler protokollieren

*Ist in der Maschinen Steuerstruktur das Strukturelement **"LogError"** gesetzt (TRUE), werden zusätzlich folgende GAT-spezifischen Fehler im Fehler-/ Diagnosespeicher der Steuerung eingetragen:*

- 16#F0**AE** xxxx: Applikations-Fehler (z. B. geöffnete Schutztür) (xxxx = frei durch den Anwender definierbar)
- 16#F0**FB** 0000: Funktionsbaustein-Fehler (z. B. Fehler an einem Technologie-Funktionsbaustein)
- 16#F0**AA** 0000: GAT interner Fehler
- 16#F0**B1...B9** xxxx: Plugin Zusatzdiagnose-Fehler (z. B. durch PROFIBUS Diagnose-Plugin)

☞ Wenn "LogError" auf TRUE gesetzt ist, werden die GAT-spezifischen Fehler auch am Steuerungsdisplay und an den Diagnoseparametern der Steuerung angezeigt.

10.10 Visualisierung

10.10.1 Interne Visualisierung zur Inbetriebnahmeunterstützung

Allgemeines

Das GAT beinhaltet vordefinierte IndraLogic-Visualisierungsobjekte, die bei der Inbetriebnahme unterstützen und erweiterte Diagnosemöglichkeiten bieten.

AXIS_GAT

Das Visualisierungselement "AXIS_GAT" ist für einzelne Achsen instanziierbar.

In der Visualisierung "MAIN_VISU" ist dies exemplarisch für die Achsen "MyVirtualAxis1" und "MyVirtualAxis2" umgesetzt.

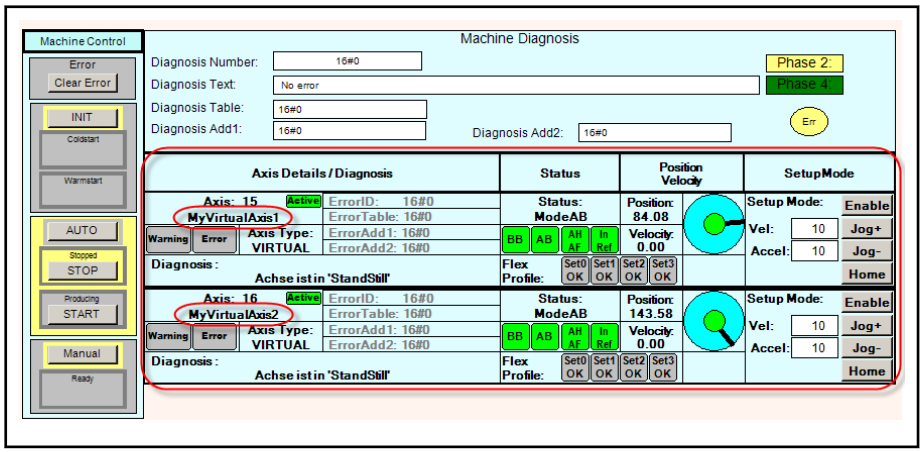


Abb.10-38: Interne Visualisierung zur Inbetriebnahmeunterstützung

Über das Objekt "Frame" kann in der Visualisierung eine Instanz des Visualisierungsobjekts "AXIS_GAT" angelegt werden.

GAT mit zentraler Zustandsmaschine

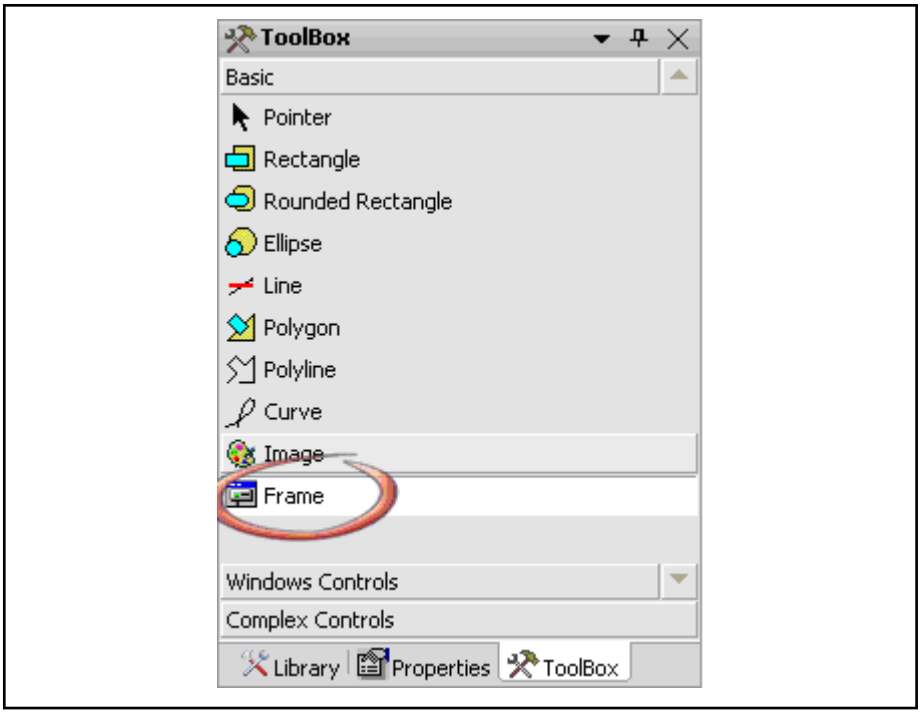


Abb.10-39: Anlegen einer Instanz des Visualisierungsobjekts "AXIS_GAT"

Unter Properties – References muss nun die entsprechende Achsreferenz eingetragen werden.

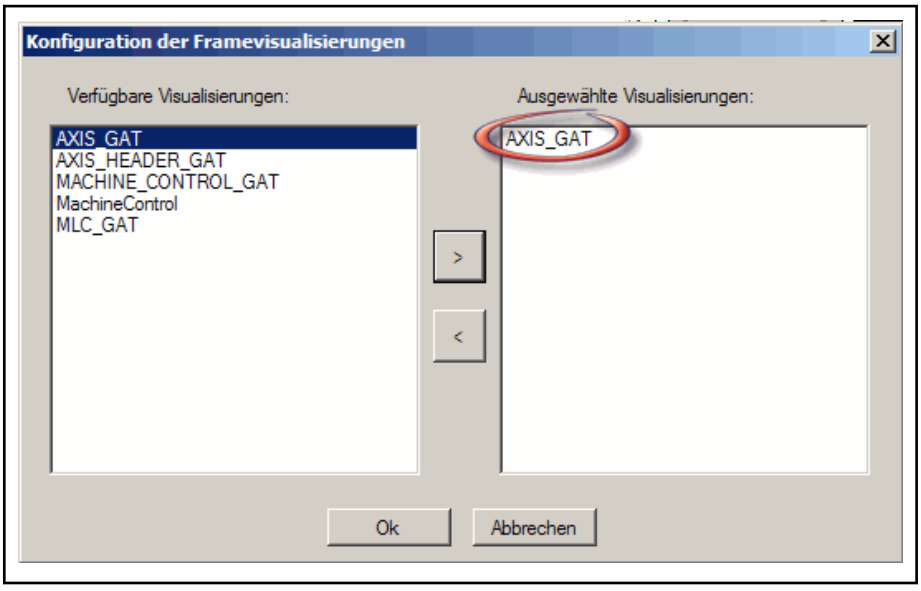


Abb.10-40: Anlegen einer Instanz des Visualisierungsobjekts "AXIS_GAT"

10.10.2 Externe Visualisierung mit separaten HMI (WinStudio-Visualisierung)

Allgemeines

Für das GAT steht eine vordefinierte WinStudio-Visualisierung zur Verfügung, die über den Wizard in das Projekt eingefügt werden kann.

Zur Bearbeitung der WinStudio-Visualisierung wird mindestens folgende Entwicklungsumgebungslizenz oder größer benötigt:

WinStudio-Entwicklungslizenz, max. 1.500 Variablen (Mat.-Nr. R911323596)

Zur Ausführung auf einem VEP wird mindestens folgende Runtime-Lizenz benötigt:

WinStudio-Runtimelizenz für Geräte mit Windows CE als Betriebssystem, max. 1.500 Variablen (Mat.-Nr. R911323620)

Einfügen der WinStudio-Visualisierung

Um die WinStudio-Visualisierung dem Projekt hinzufügen zu können, muss ein entsprechendes Visualisierungsgerät im Projekt vorhanden sein.

Fügen Sie mittels "Drag and Drop" ein VEP aus der Gerätebibliothek in den Projektbaum ein und konfigurieren Sie es wie folgt:

- Gerätetyp: VEPxx (CE)
- Applikationstyp: WinStudio-Anwendung

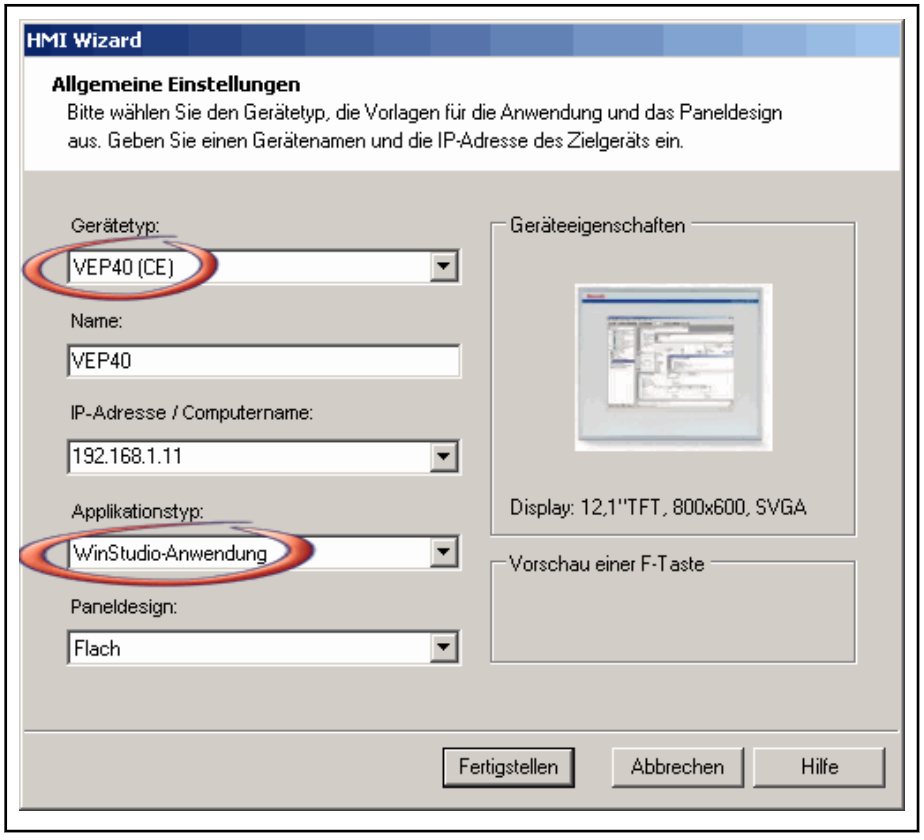


Abb.10-41: Visualisierungsgerät einfügen

GAT mit zentraler Zustandsmaschine

Wenn ein entsprechendes Visualisierungsgerät vorhanden ist, wird über folgenden Link im Wizard die WinStudio-Visualisierung eingefügt:

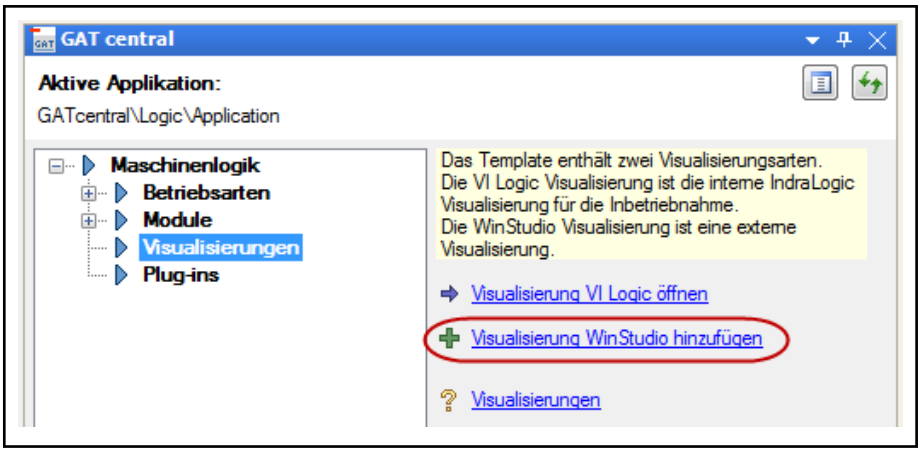


Abb.10-42: WinStudio-Visualisierung einfügen

Im nächsten Dialog des Wizard werden alle Visualisierungsgeräte des Projekts angeboten, deren Konfiguration der Vorgabe entspricht.

Nach Auswahl des entsprechenden Visualisierungsgerätes kann mittels der Schaltfläche **Übernehmen** das Einfügen der WinStudio-Visualisierung ausgeführt werden.

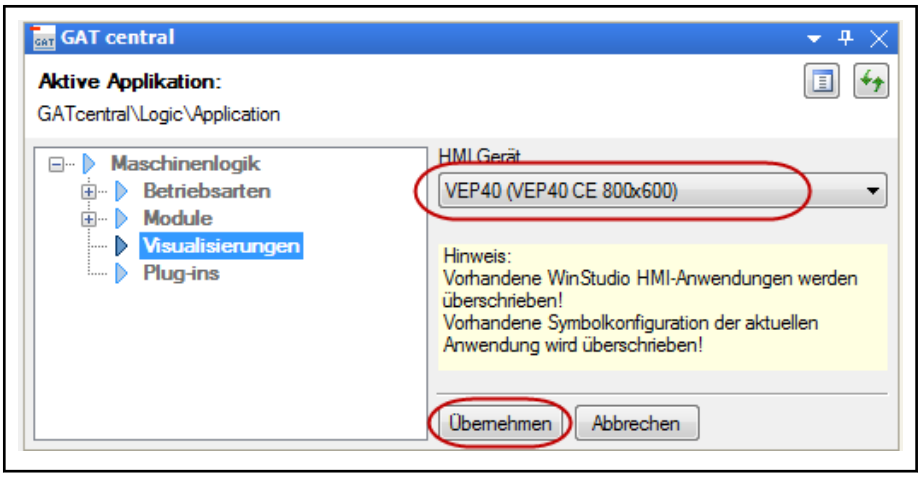


Abb.10-43: Visualisierungsgerät auswählen – WinStudio-Visualisierung einfügen

Bestehende Visualisierungsdaten des Visualisierungsgerätes und die Symbolkonfiguration der Steuerung werden durch das Einfügen der WinStudio-Visualisierung überschrieben!

Nach dem Einfügen können die Visualisierungsdaten bearbeitet werden.

Die Konfiguration der Steuerungsadresse erfolgt im "Eigenschaften"-Dialog des BR-WS-Treibers:

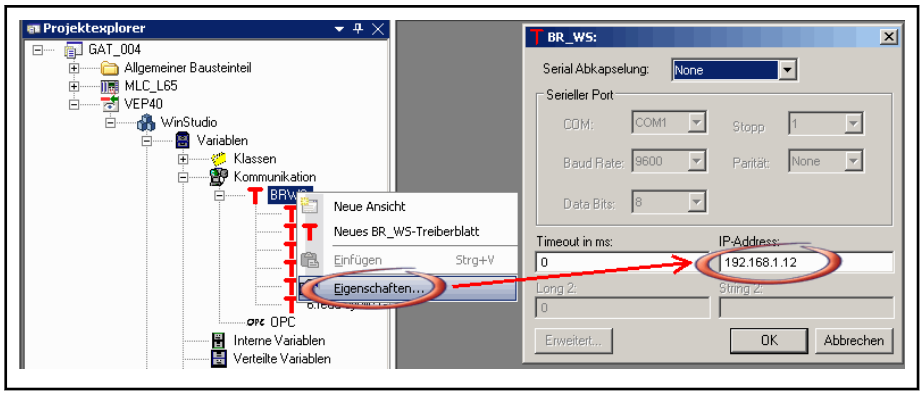


Abb.10-44: Einstellen der IP-Adresse der Steuerung

Im Kontextmenü des WinStudio-Knotens kann nun über  "Anwendung Starten" die Emulation der Winstudio-Visualisierung gestartet werden.

Dialog-Struktur

Die Bedienbilder teilen sich in 3 Teilbereiche auf:

- Kopfzeile mit Statusinformationen
- Hauptbildbereich mit menüabhängigen Anzeigen und Funktionstasten
- Fußzeile mit Tasten zur Bedienbildumschaltung

GAT mit zentraler Zustandsmaschine

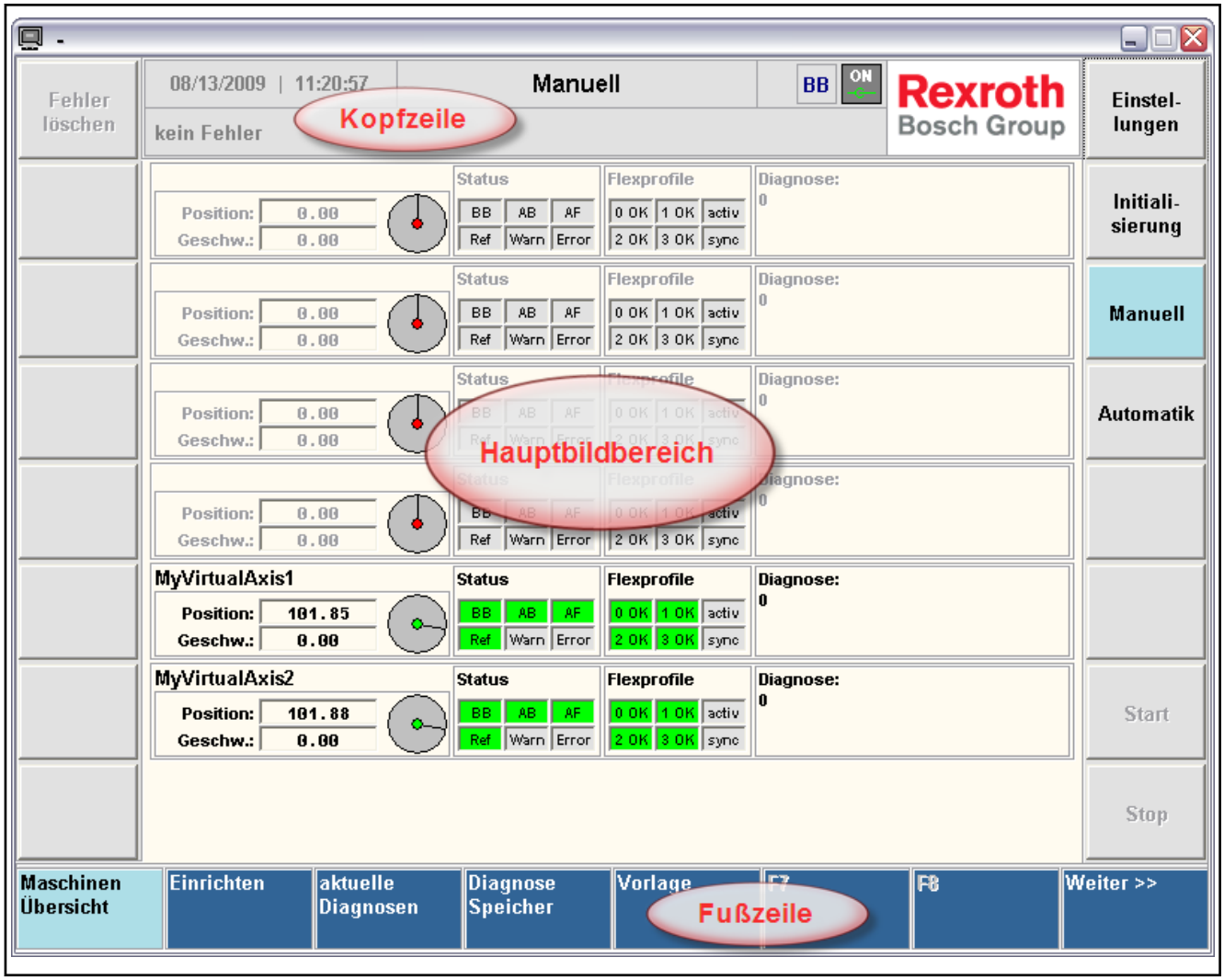


Abb. 10-45: WinStudio-Visualisierung

*Die **Kopfzeile** beinhaltet:*

- Betriebsartenanzeige
- Globale Funktionstasten zum "Fehlerlöschen" und Öffnen des "Einstellungen"-Dialogs (Sprachumschaltung, Einstellung der Dezimalstellen für numerische Anzeigen)
- Anzeige aktuell anstehender Maschinen-Diagnosen
- Anzeige der aktuellen sercos Phase (inklusive aktiver Phasenumschaltung)
- Anzeige des Kommunikationsstatus

Der **Hauptbildbereich** beinhaltet die bedienbildabhängigen Anzeigen und Funktionstasten (z. B. Achsinformationen, Betriebsartenumschaltung, …)

Die **Fußzeile** beinhaltet die Tasten zur Bedienbildumschaltung.

Erweitern der Visualisierung

Bedienbild einfügen

Zum Erweitern der Visualisierung können neue Bedienbilder mittels Kopieren der vorhanden "Template"-Bilder und Einfügen unter neuem Namen angelegt werden:

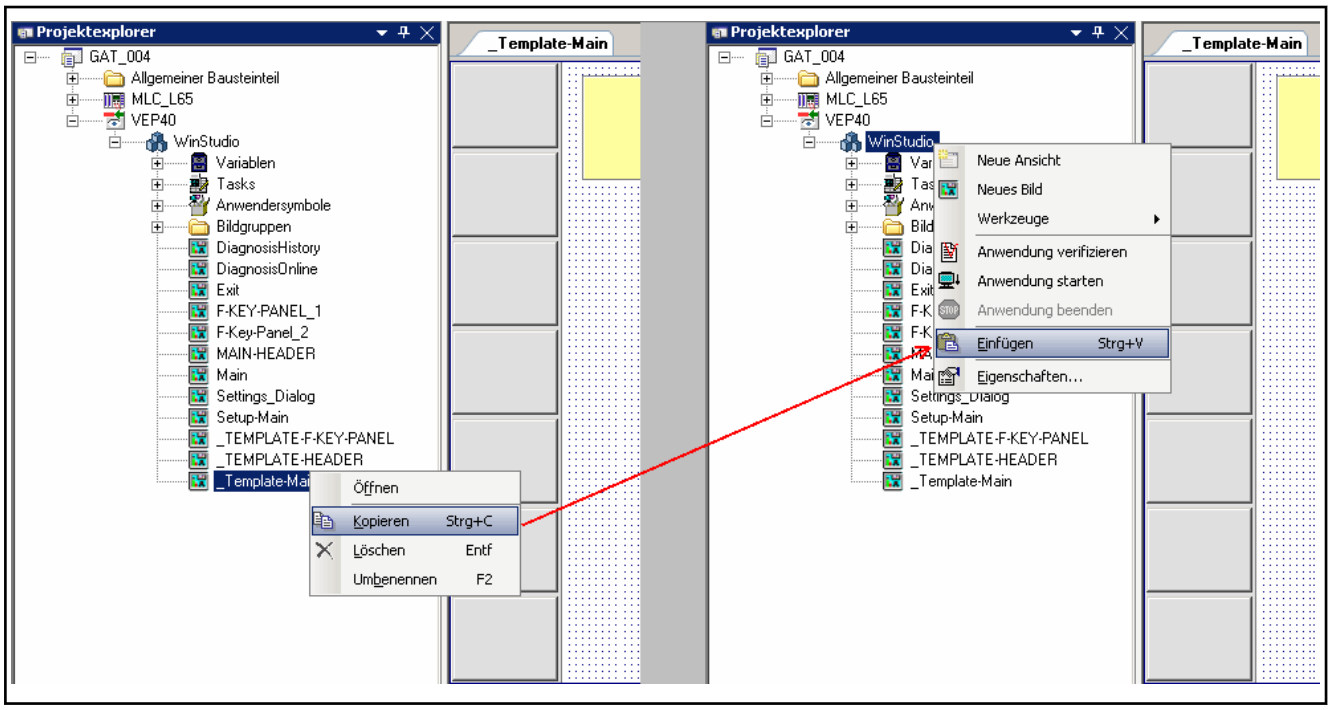


Abb. 10-46: Bedienbild duplizieren

Die Elemente zum Anzeigen der Achsdaten sind gruppierte Einzelobjekte, die durch "Kopieren" und "Einfügen" dupliziert werden können.

Zum Umkonfigurieren des eingefügten Objekts auf eine andere Achse muss nun über die Ersetzen-Funktion des jeweiligen Objekts der Array-Index der Achse angepasst werden:

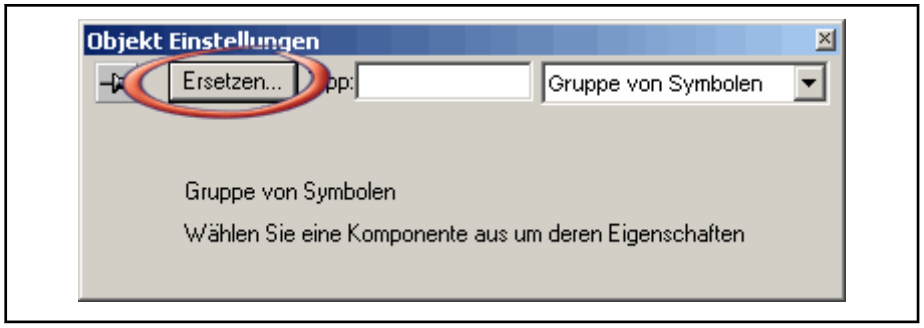


Abb. 10-47: Umkonfigurieren des eingefügten Objekts

GAT mit zentraler Zustandsmaschine

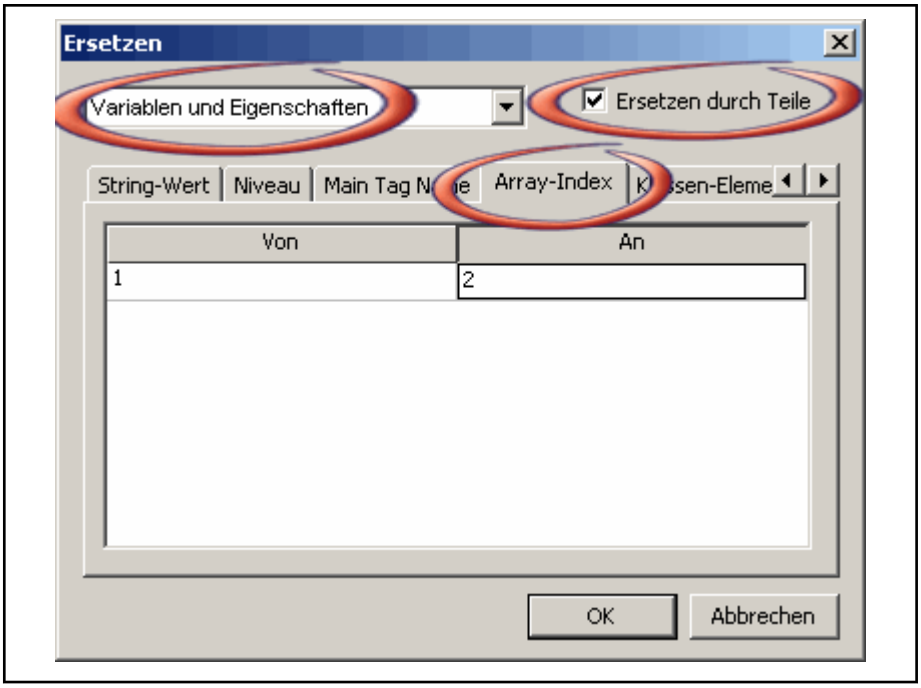


Abb.10-48: Umkonfigurieren des eingefügten Objekts

Fußzeile erweitern

Die Tasten zur Bedienbildumschaltung sind als parametrierbare Grafik-Objekte angelegt (linked Symbol) und verfügen über folgende Übergabe-Parameter:

- Caption -> Beschriftung der Taste
- Disable -> Eingang zum Sperren der Tastenfunktionalität (ausgegraute Beschriftung)
- ScreenName -> Bildname des zu öffnenden Bedienbildes (bzw. Bildgruppe)
- ScreenNumber -> Nummer, die dem Bild zugewiesen werden soll (zum farblichen hervorheben der jeweiligen Taste)

Kommunikationsanbindung

Die Kommunikation zwischen Visualisierung und Steuerung erfolgt über den BR-WS Treiber. Hierzu ist es erforderlich, die Symboleinträge für die zu visualisierenden SPS-Variablen zu erzeugen (siehe hierzu Kap. "Kommunikations-Konfiguration" auf Seite 53).

Die benötigte Symbolkonfiguration der vordefinierten WinStudio-Visualisierung beinhaltet folgende Einträge:

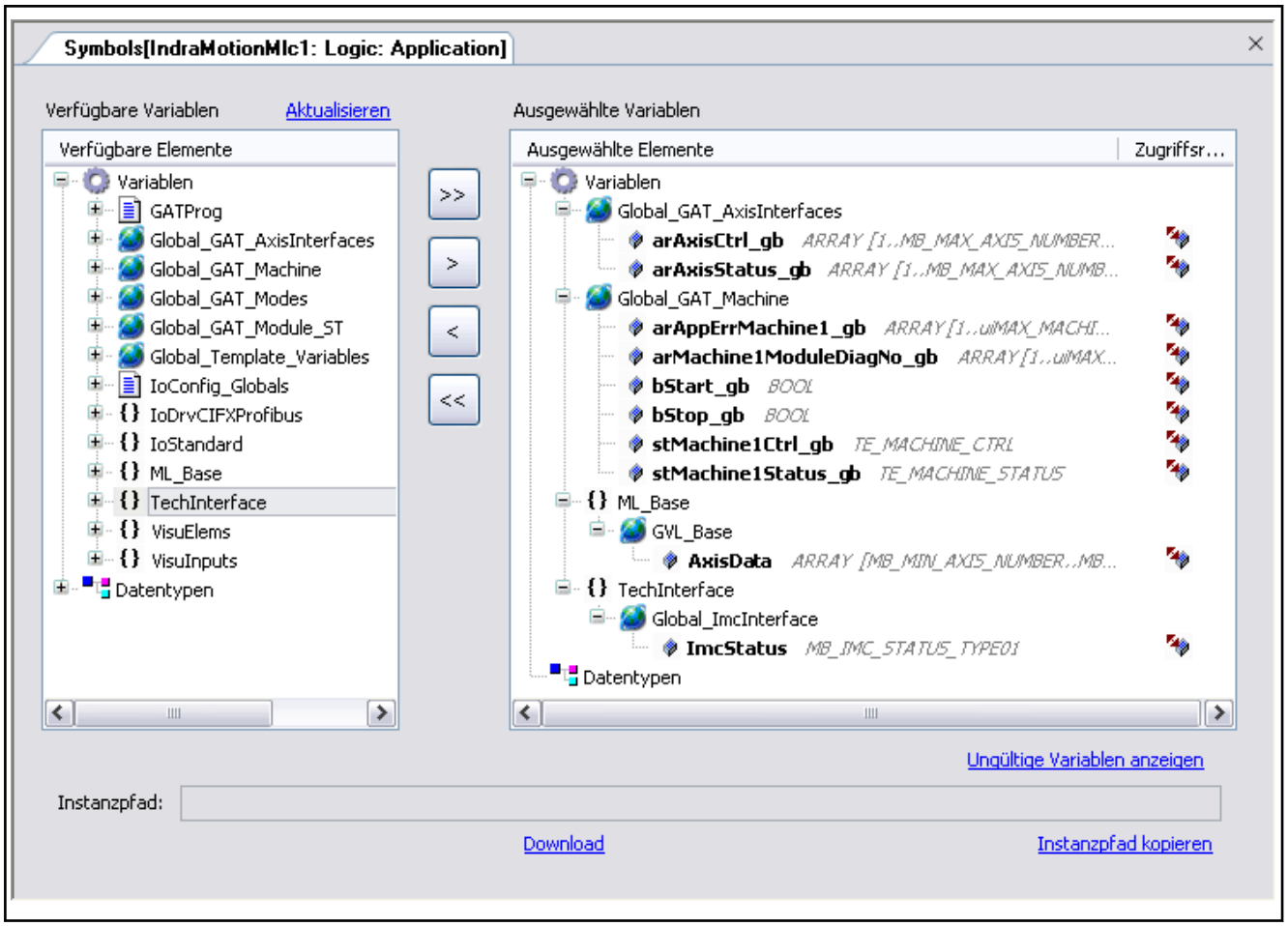


Abb. 10-49: Symbolkonfiguration der vordefinierten WinStudio-Visualisierung

Symboleinträge für Variablen aus Globalen Variablenlisten die aus einer Bibliothek stammen, müssen nach einer Versionsänderung der entsprechenden Bibliothek erneut in die Symbolkonfiguration eingefügt werden.

Einstellen der IP-Adresse der Steuerung:

Die Konfiguration der Steuerungsadresse erfolgt im "Eigenschaften"-Dialog des BR-WS Treibers:

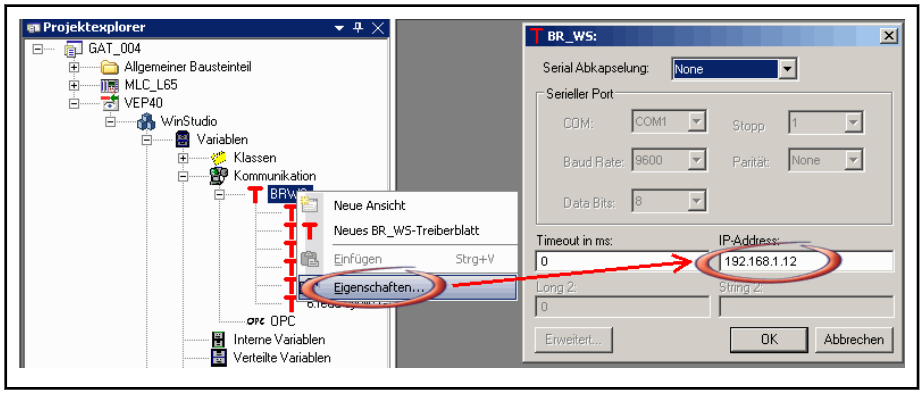


Abb. 10-50: Einstellen der IP-Adresse der Steuerung

GAT mit zentraler Zustandsmaschine

Um die Kommunikation zu strukturieren, wurden die Variablen in verschiedene Treiberblätter aufgeteilt:

1. read cyclic fast: -> Variable deren Werte zyklisch in kurzen Intervallen gelesen werden sollen (z. B. Diagnosenummern, Achspositionen, …)
2. read only once: -> Variablen die einmalig nach dem Start der Visualisierung gelesen werden (z. B. Achsnamen)
3. read after switching phase: Variablen die nur einmalig nach einer Phasenumschaltung gelesen werden (z. B. ob die Achse aktiviert wurde)
4. read - write cyclic: -> Variablen, deren Werte durch die Visualisierung sowohl gelesen als auch geschrieben werden sollen (z. B. Maschinengeschwindigkeit, Achsgeschwindigkeit für Einrichtbetrieb, …)
5. write only: -> Variablen die durch die Visualisierung nur geschrieben werden sollen (Triggersignal zum Fehlerlöschen, …)

Zum Erweitern eines Treiberblattes kann die offline Symboldatei (*.XML) mit dem BR-WS Browser durchsucht werden.

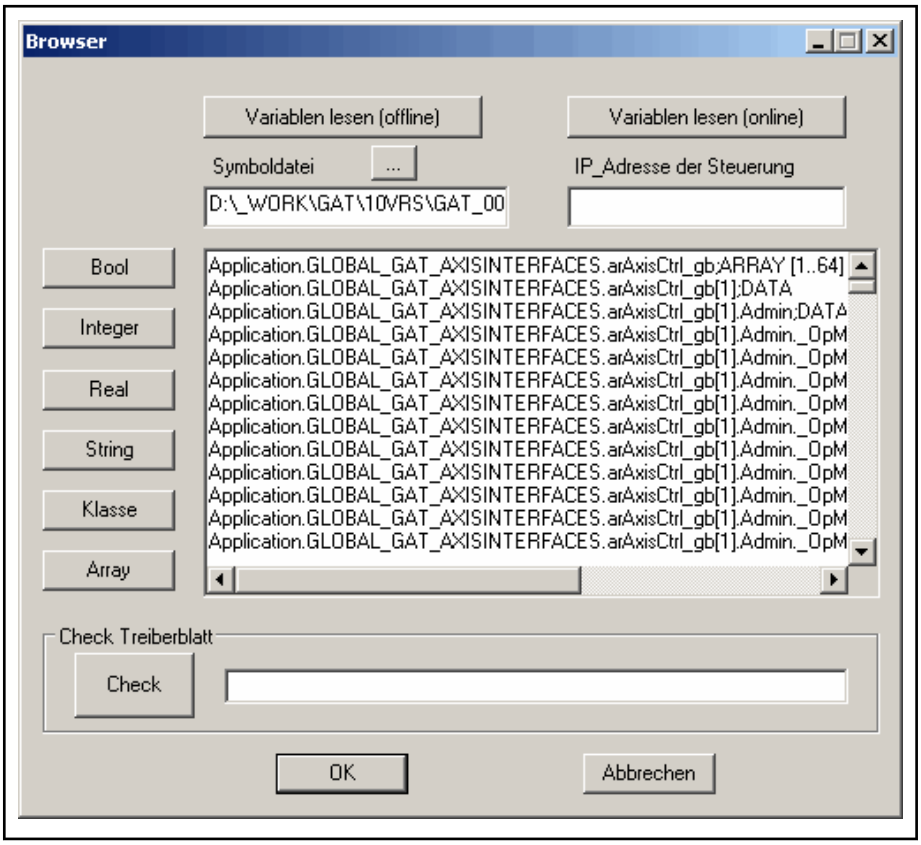


Abb.10-51: Erweitern des Treiberblattes

Mehrsprachigkeit

Die Mehrsprachigkeit der Visualisierung wurde mit der in IndraWorks integrierten Übersetzungsfunktionalität realisiert.

Hierbei sind die Übersetzungsdaten bereits für Steuerungs- und Antriebs-Diagnosen, sowie den verwendeten Texten der Visualisierung vorbereitet. Die Übersetzungsdaten können mit IndraWorks modifiziert bzw. ergänzt werden.

Antriebsdiagnosen sind hierbei in deutsch, englisch, französisch, spanisch und türkisch enthalten.

Alle andere Texte sind in deutsch und englisch enthalten.

10.11 Schnittstellen zum GAT

10.11.1 Allgemeines

Die Schnittstellendaten des GAT werden in diversen Variablenlisten bereitgestellt. Ein Teil der Daten wird über Bibliotheken bereitgestellt.

10.11.2 Daten zur Maschine

Die Eingangsdaten der Maschine sind in der Struktur stMachine1Ctrl_gb (Global_GAT_MachineControl) vom Typ TE_MACHINE_CTRL zusammengefasst.

Die folgende Tabelle beschreibt die Elemente der Struktur stMachine1Ctrl_gb.

I/O-Typ	Name	Datentyp	Beschreibung
I	OpMode	TE_OPMODES_PREDEFINED	Vorgabe Betriebsart der Maschine
I	OpMode_InitSuccessor	TE_OPMODES_PREDEFINED	Betriebsart der Maschine nach erfolgreicher Initialisierung
I	OpMode_ErrorSuccessor	TE_OPMODES_PREDEFINED	Betriebsart der Maschine nach erfolgreichem Fehlerlöschen
I*)	StateCtrl	TE_STATES_PREDEFINED	Intern verwendet/Zustände der Maschine
I	SkipColdstartAtStartup	BOOL	Beim Starten des GAT-Programms wird eine Umschaltung in den Parametriermodus (P2) zur Durchführung der zur Applikation nötigen Parametrierung ausgeführt. Mit TRUE kann diese Phasenumschaltung übersprungen werden, wenn die Applikation keine Parametrierung in P2 erfordert
I	DisableErrorHandling	BOOL	TRUE = bei einem Fehler wird nicht in den Zustand "ERROR" gewechselt. Die Fehleranzeige wird weiterhin ausgeführt und auch Fehlerlöschen ist weiterhin möglich
I	ClearError	BOOL	Globales Fehlerlöschen mit positiver Flanke
I	LogError	BOOL	TRUE = GAT spezifische Fehler (Applikations-, FB-, GAT- und Plugin-Fehler) werden zusätzlich in dem Fehler-/Diagnosespeicher der Steuerung abgespeichert.

*) Darf vom Anwender nicht beschrieben werden. Das Setzen dieses Eingangs ist der Maschinen-Organisationseinheit vorbehalten.

I Eingang zum Beschreiben

Die Ausgangsdaten der Maschine sind in der Struktur stMachine1Status_gb (Global_GAT_MachineControl) vom Typ TE_MACHINE_STATUS zusammengefasst.

Die folgende Tabelle beschreibt die Elemente der Struktur stMachine1Status_gb.

GAT mit zentraler Zustandsmaschine

I/O-Typ	Name	Datentyp	Beschreibung
O	OpMode	TE_OPMODES_PREDEFINED	Rückmeldung aktuelle Betriebsart der Maschine
O	State	TE_STATES_PREDEFINED	Intern verwendet/Rückmeldung aktueller Zustand der Maschine
O	Error	BOOL	TRUE = globaler Fehler steht an
O	ErrorID	ERROR_CODE	Detailinformation wenn DiagNo = 16#F0FB 0000/16#F0AA xxxx
O	ErrorIdent	ERROR_STRUCT	Detailinformation wenn DiagNo = 16#F0FB 0000/16#F0AA xxxx
O	DiagNo	DWORD	Nummer gemäß Dokumentation "Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnosen" mit den folgenden Erweiterungen: DiagNo = 16#F0AE xxxx: Applikations-Fehler (z. B. geöffnete Schutztür) DiagNo = 16#F0FB 0000: FB-Fehler DiagNo = 16#F0AA xxxx: GAT-interner Fehler DiagNo = 16#F0B1...B9 xxxx: Plugin Zusatzdiagnose Fehler (z. B. durch PROFIBUS Diagnose Plugin)
O	DiagMessage	STRING(60)	Diagnosetext der Steuerung/Achse

O Ausgang zum Lesen

10.11.3 Daten zum Fehlerhandling

Die Ein- und Ausgänge zum Fehlerhandling sind in den oben genannten Strukturen zur Maschine enthalten.

10.11.4 Daten zu den Modulen

Die Module werden von der Maschinen-Organisationseinheit gesteuert. Auf die Daten der Module kann zu Diagnosezwecken von extern zugegriffen werden.

Die Eingangsdaten aller Module sind im Array arModuleCtrl vom Typ TE_CMI_CONTROL innerhalb des Funktionsbausteines bMachine1_gb (Global_GAT_MachineControl) zusammengefasst.

Die folgende Tabelle beschreibt die Elemente der Struktur Global_GAT_MachineControl.fbMachine1_gb.arModuleCtrl.

I/O-Typ	Name	Datentyp	Beschreibung
I	Enable	BOOL	TRUE = Start der Bearbeitung des Moduls
I	ClearError	BOOL	TRUE = Fehlerlöschen des Moduls
I*)	State	TE_STATES_PREDEFINED	Vorgabe-Zustand des Moduls

I/O-Typ	Name	Datentyp	Beschreibung
I*)	StateTransient	BOOL	TRUE = Modul befindet sich im Übergang zum kommandierten Zustand
I*)	DisableTransient	BOOL	TRUE = Deaktivierung des Moduls wurde gestartet

I Eingang zum Beschreiben
*) Darf vom Anwender nicht beschrieben werden. Das Setzen dieses Eingangs ist der Maschinen-Organisationseinheit vorbehalten.

Elemente der arModuleCtrl werden im FB "Machine" beschrieben und können deshalb von außen nur gelesen werden.

Die Ausgangsdaten aller Module sind im Array arModuleStatus vom Typ TE_CMI_STATUS innerhalb des Funktionsbausteins bMachine1_gb (Global_GAT_MachineControl) zusammengefasst.

Die folgende Tabelle beschreibt die Elemente der Struktur Global_GAT_MachineControl.fbMachine1_gb.arModuleStatus.

I/O-Typ	Name	Datentyp	Beschreibung
O*)	InOperation	BOOL	TRUE = das Modul wird bearbeitet
O*)	State	TE_STATES_PREDEFINED	Rückmeldung Zustand des Moduls
O	StateLocked	BOOL	TRUE = das Weiterschalten auf den nachfolgenden Zustand ist durch den Anwender gesperrt worden
O	TransientLocked	BOOL	TRUE = Der Übergang in den stationären Teil des kommandierten Zustandes wurde durch den Anwender gesperrt
O*)	TransientState	TE_STATES_PREDEFINED	Anzeige des Zustands in den das Modul gerade wechselt (kommandierter Zustand ist noch nicht erreicht, Aktivitäten dafür laufen)
O	DisableLocked	BOOL	TRUE = Aktivitäten zur Deaktivierung des Moduls laufen noch
O*)	Error	BOOL	TRUE = Modulfehler steht an
O*)	ErrorID	ERROR_CODE	Detailinformation wenn DiagNo = 16#F0FB 0000
O*)	ErrorIdent	ERROR_STRUCT	Detailinformation wenn DiagNo = 16#F0FB 0000

GAT mit zentraler Zustandsmaschine

I/O-Typ	Name	Datentyp	Beschreibung
O*)	DiagNo	DWORD	Nummer gemäß Dokumentation "Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnosen" mit den folgenden Erweiterungen: DiagNo = 16#F0AE xxxx: Applikations-Fehler (z. B. geöffnete Schutztür) DiagNo = 16#F0FB 0000: FB-Fehler DiagNo = 16#F0AA xxxx: GAT-interner Fehler 16#F0B1...B9 xxxx: Plugin Zusatzdiagnose Fehler (z. B. durch PROFIBUS Diagnose Plugin)
O*)	DiagMessage	STRING(60)	Diagnosetext der Steuerung/Achse

O Ausgang zum Lesen
*) Darf vom Anwender nicht beschrieben werden. Das Setzen dieses Ausgangs ist der Maschinen-Organisationseinheit vorbehalten.

10.11.5 Daten der Achsen

I/O-Typ	Name	Datentyp	Beschreibung
I	arAxisCtrl_gb (Global_GAT_AxisInterfaces)	ARRAY OF TE_AXIS_CONTROL_TYPE01	Vorgaben für die Achsen
O	arAxisStatus_gb (Global_GAT_AxisInterfaces)	ARRAY OF TE_AXIS_STATUS_TYPE01	Rückmeldungen der Achsen
I/O	AxisData (GVL_Base aus der Bibliothek ML_Base)	ARRAY OF ML_AXISDATA_SM	Firmwarestruktur mit Ein- und Ausgängen der Achsen

I Eingang zum Beschreiben
O Ausgang zum Lesen
I/O Struktur mit Ein- und Ausgängen

☞ Elemente der arAxisCtrl_gb werden im FB "Machine" beschrieben und können deshalb von außen nicht mehr genutzt werden.

10.11.6 Daten der Steuerung

I/O-Typ	Name	Datentyp	Beschreibung
I	ImcCtrl (Global_ImcInterface aus der Bibliothek ML_TechInterface)	MB_IMC_CONTROL_TYPE01	Vorgaben für die Steuerung
O	ImcStatus (Global_ImcInterface aus der Bibliothek ML_TechInterface	MB_IMC_STATUS_TYPE01	Rückmeldungen der Steuerung

I Eingang zum Beschreiben
O Ausgang zum Lesen

☞ Elemente der ImcCtrl_gb werden im FB "Machine" beschrieben und können deshalb von außen nicht mehr genutzt werden.

10.12 HowTo: Typische Anwenderaktivitäten

10.12.1 Zugriff auf Achsdaten

Zum Zugriff auf Achsdaten ist in jedem Modul als zentraler Zugriffspunkt die Struktur **Axes** für alle dem Modul zugeordneten Achsen definiert. Über diese Struktur sind alle relevanten Daten für eine Achse adressierbar.

Die Struktur ist so aufgebaut, das über den Namen der Achse auf die Achsdaten zugegriffen wird und damit gut lesbarer Programmcode entsteht.

Die folgenden Daten sind unterhalb von "Axes" verfügbar:

- Axes.<Name>.AxisData => Achsdatenstruktur: Statusdaten der Firmware und Zugriff auf Achsdatencontainer
- Axes.<Name>.Ctrl => Steuerstruktur des Achs-Interface
- Axes.<Name>.Status => Statusstruktur des Achs-Interface
- Axes.<Name>.DV => Zugriff auf Direktvariablen (Funktionsvariablen)
- Axes.<Name>.AxisRef => Achsreferenzstruktur für FB-Aufrufe (z. B. MB_ReadParameter)

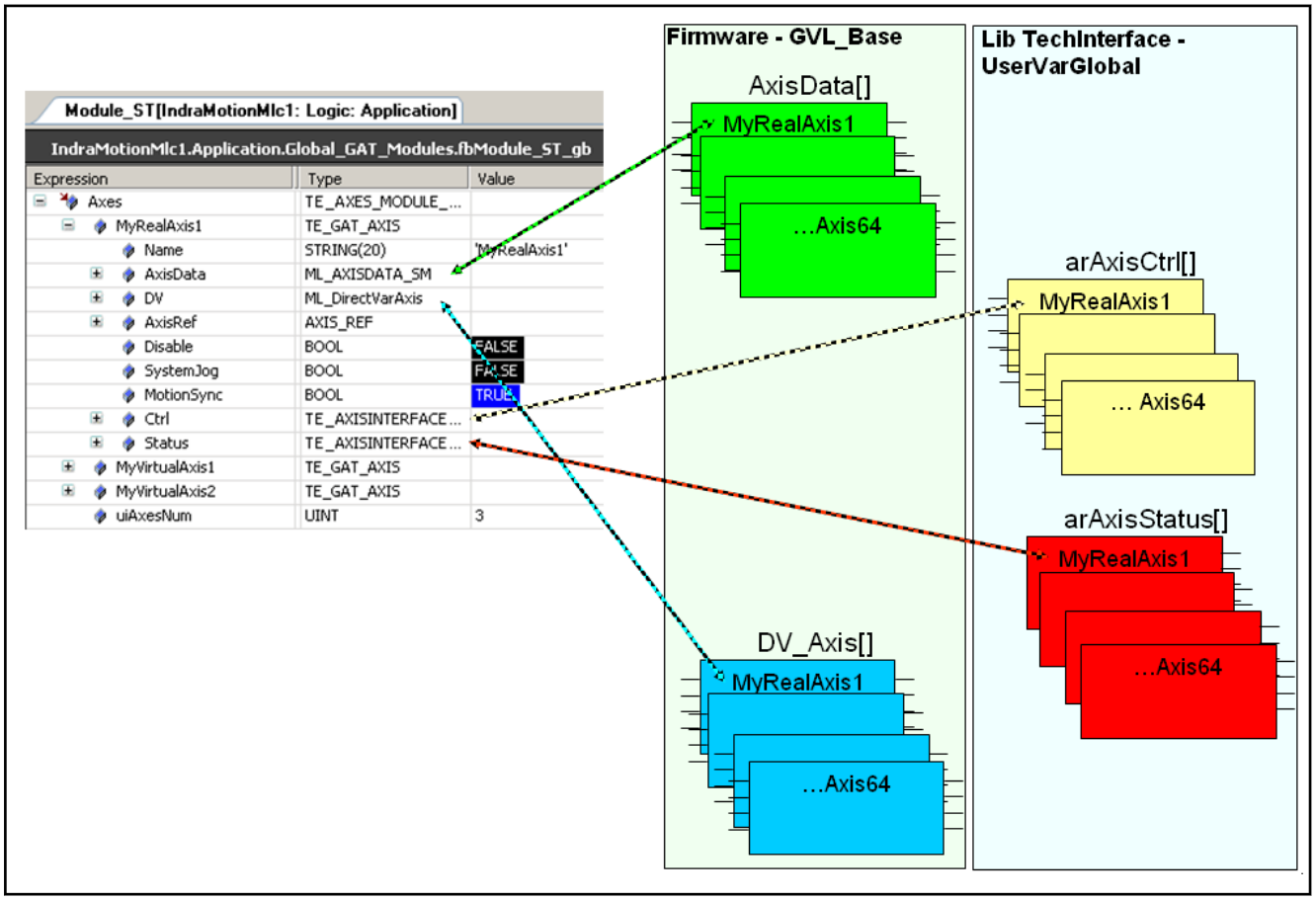


Abb. 10-52: Übersicht Achsdatenzugriffe

Zur Programmierung von Schleifen über alle Achsen eines Moduls bringt jedes Modul die Methode **mGetAxis()** und das Element **Axes.uiAxesNum** mit.

Beispiel: Alle Achsen eines Moduls in "ModeAb" schalten:

Deklaration:

```
pCurrentAxis:       POINTER TO TE_GAT_AXIS;
uiLoopIndex:        UINT; // For loop index
```

GAT mit zentraler Zustandsmaschine

Implementation:

```
// ------------------------ Error reaction example 1 --------------------
// Decelerate all module axes to standstill and turn power off
// --> switch all axes off to ModeAb
// ----------------------------------------------------------------------
FOR uiLoopIndex := 1 TO Axes.uiAxesNum DO
   IF mGetAxis(AxisNumber:=uiLoopIndex, PointerToAxis => pCurrentAxis) THEN
      pCurrentAxis^.Ctrl.Admin._OpMode.en := ModeAb;
   END_IF
END_FOR
```

10.12.2 Achse hinzufügen

Zum Hinzufügen einer Achse sind folgende Schritte durchzuführen:

- Einfügen der Achse in das IndraWorks-Projekt (auf den sercos Knoten ziehen…)
- Bei Nutzung der Visualisierung "SystemOverview_All_Axis": einen Frame für die neue Achse einfügen und in den Eigenschaften die Referenz auf die neue Achse setzen (siehe auch Kap. 8.10.2 "Achse hinzufügen" auf Seite 245)
- Über den GAT-Wizard die Achse hinzufügen (siehe nachfolgende Abbildung)

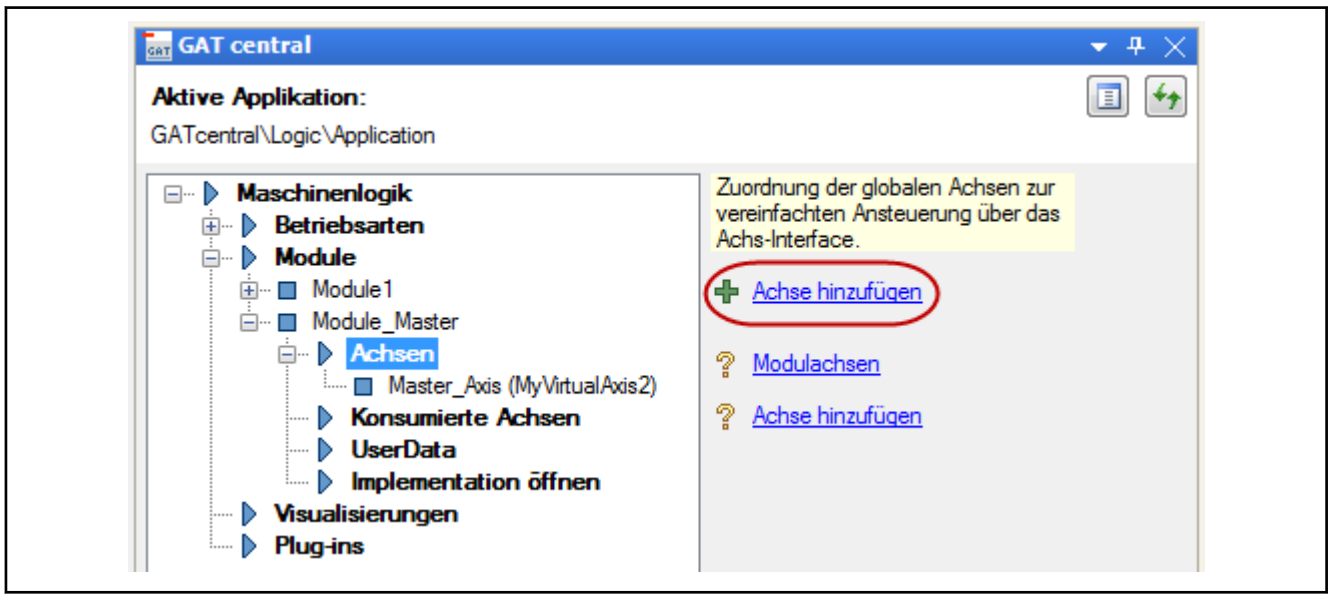


Abb.10-53: Achse über den GAT-Wizard hinzufügen

10.12.3 Achse entfernen/umbenennen

☞ Der Wizard ändert nur die Namen in der Achs-Deklaration. Wird die Achse bereits in der Implementation bzw. internen Visualisierung verwendet, so muss der neue Achsname durch "Suchen/Ersetzen" an diesen Stellen angepasst werden.

Zum Entfernen/Umbenennen einer Achse sind folgende Schritte durchzuführen:

- Löschen/Umbenennen der Achse aus dem IndraWorks-Projekt
- Bei Nutzung der Visualisierung "SystemOverview_All_Axis": den Frame für die gelöschte Achse entfernen bzw. in den Eigenschaften die Referenz auf die Achse ändern

- Über den GAT-Wizard die Achse entfernen/umbenennen (über "Eigenschaften" kann der Achsnahme verändert werden, siehe nachfolgende Abbildung)

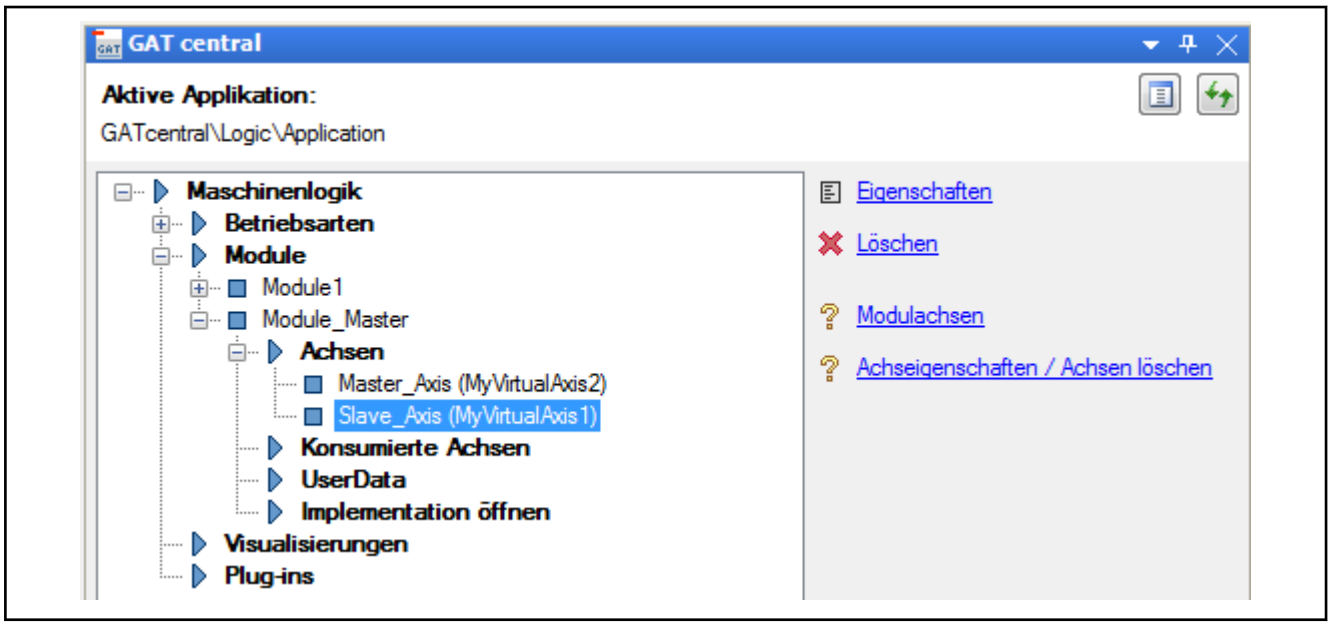


Abb.10-54: Achse mit dem GAT-Wizard entfernen/umbenennen

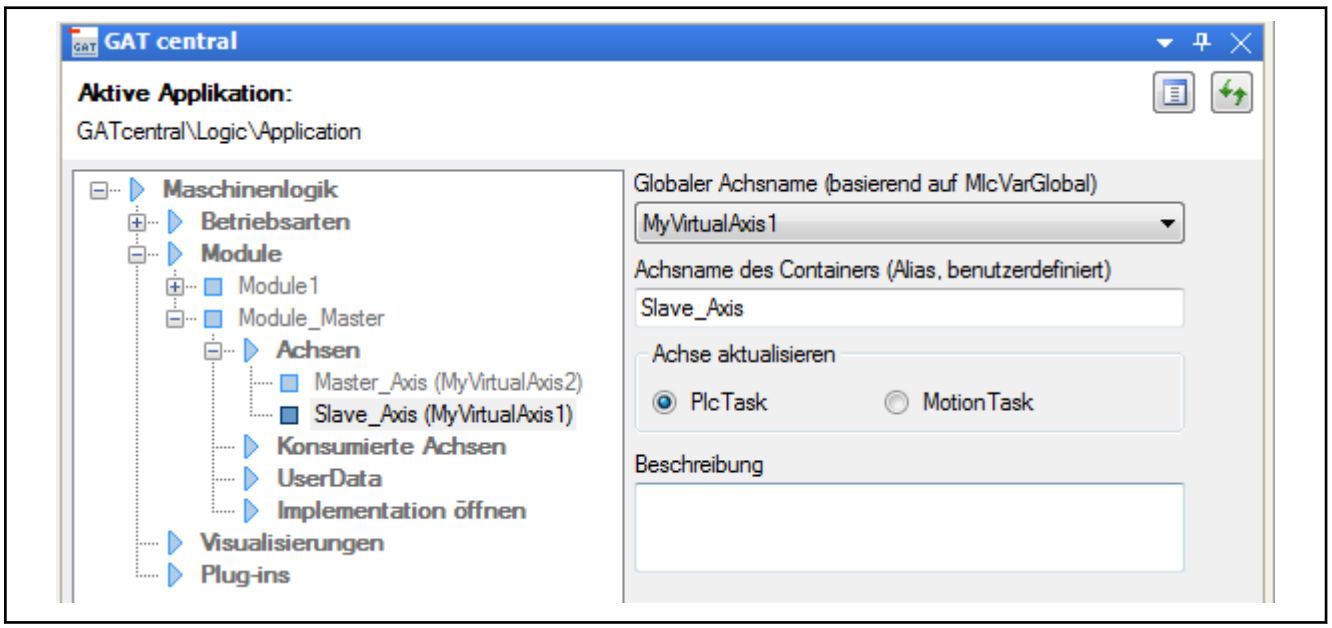


Abb.10-55: Achse mit dem GAT-Wizard entfernen/umbenennen

GAT mit zentraler Zustandsmaschine

10.12.4 Betriebsart hinzufügen

Der GAT-Wizard erlaubt das komfortable Hinzufügen von Betriebsarten zur Maschine. Über den Wizard können die Betriebsarten auch gelöscht oder umbenannt werden.



Abb.10-56: Betriebsart mit dem GAT-Wizard hinzufügen

10.12.5 Anwahl der gewünschten Betriebsart nach Fehlerlöschen

Über die Variable **fbMachine_gb.Ctrl. OpMode_ErrorSuccessor** kann das Verhalten der Applikation nach Fehlerlöschen beeinflusst werden. Der Wert dieser Variablen kann zu jeder Zeit durch den Anwender geändert werden. Unmittelbar nach Fehlerlöschen wird die Variable durch die Maschinen-Organisationseinheit ausgewertet.

Per Voreinstellung ist der Wert dieser Variable "STATE_UNDEFINED". Wird unter dieser Randbedingung ein Fehler zurückgesetzt, verlässt die Applikation den Fehlerzustand und geht über den Zustand STATE_INIT_WARMSTART in den STOPPED-Zustand der Betriebsart über, die zuvor aktiv war.

Der Zustand "PRODUCING" der Betriebsart "AUTO" war z. B. vor Auftreten eines Fehlers aktiv. Wird der Fehler gelöscht, wechselt die Applikation über den Zustand STATE_INIT_WARMSTART in den Zustand STATE_AUTO_STOPPED.

In dem Fall, dass der Wert der Variablen **fbMachine_gb.Ctrl. OpMode_ErrorSuccessor** eine bestimmte Betriebsart ist, dann wird die Applikation nach Fehlerlöschen in den STOPPED-Zustand dieser Betriebsart kommandiert.

Z. B. Vor Auftreten eines Fehlers war der Zustand "PRODUCING" der Betriebsart "AUTO" aktiv. Außerdem gilt "fbMachine_gb.Ctrl. OpMode_ErrorSuccessor=OPMODE_MANUAL". Nach Löschen des Fehlers wechselt die Applikation direkt in den Zustand STATE_MANUAL_STOPPED.

Der Initialzustand von INIT ist STATE_INIT_COLDSTART.

Tritt ein Fehler auf, während sich die Applikation im Zustand STATE_INIT_COLDSTART bzw. STATE_INIT_WARSTART befindet, kehrt die Applikation nach Fehlerlöschen in exakt diesen Zustand zurück.

10.12.6 Übergeordnete Logik zur Betriebsartenvorgabe

In machen Anwendungsfällen ist die Anwahl einer Betriebsart der Maschine von zusätzlichen Bedingungen abhängig. Die empfohlene Stelle um diese Logik zu implementieren ist im Programm "GATProg" vor Aufruf des "Machine"-FB. Hier können die Bedingungen abgefragt und die Betriebsart über die globale Variable **stMachine1Ctrl_gb.OpMode** vorgegeben werden.

10.12.7 Zustände und Transitionen hinzufügen

Sind die auszuführenden Aktivitäten innerhalb einer Betriebsart sehr komplex, empfiehlt sich die Unterteilung in mehrere Zustände. Dadurch wird die Applikation übersichtlich strukturiert und für Dritte leichter verständlich.

Der GAT-Wizard ermöglicht das einfache Hinzufügen von Zuständen und Bedingungen zwischen Zuständen (Transitionen).



Abb.10-57: Zustände hinzufügen

Beim Hinzufügen von Zuständen innerhalb einer Betriebsart wird unterschieden, ob ein neuer Zustand nach einem fokussierten Zustand oder nach einer fokussierten Bedingung (Transition) eingefügt werden soll. Außerdem ist es möglich, Verzweigungen zu erstellen (siehe vorhergehende Abbildung).

In allen zu diesem Zeitpunkt existierenden Modulen werden bei der Generierung eines neuen Zustands zustandskorrespondierende Aktionen angelegt.

GAT mit zentraler Zustandsmaschine

10.12.8 Neues Modul hinzufügen

Der GAT-Wizard erlaubt das komfortable Hinzufügen von Modulen zur Maschine. Über den Wizard können die Module auch gelöscht oder umbenannt werden.

Für alle in diesem Moment existierenden Zustände werden in dem neuen Modul zustandskorrespondierende Aktionen angelegt.

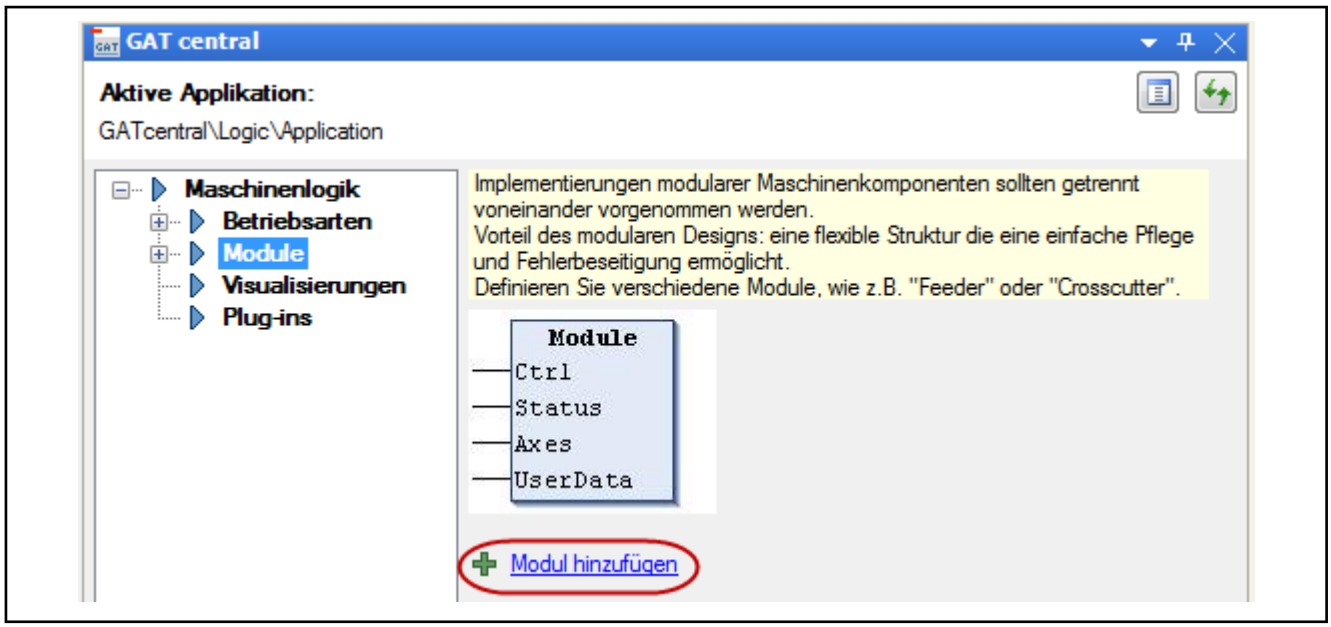


Abb. 10-58: Modul hinzufügen

10.12.9 Weitere Instanz eines Moduls hinzufügen

Der GAT-Wizard erlaubt das komfortable Hinzufügen weiterer Instanzen von existierenden Modulen zur Maschine.

Die Schnittstellen bleiben identisch, es müssen jedoch andere globale Achsen (an "Axes") und I/O-Daten (an "UserData") zugewiesen werden.

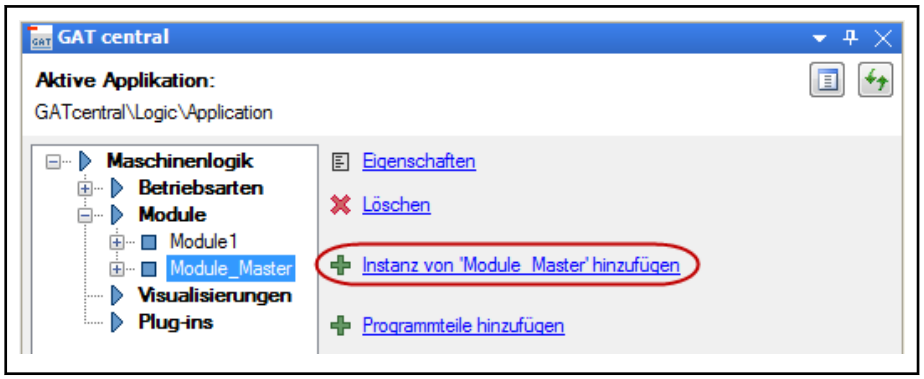


Abb.10-59: Weitere Instanz eines Moduls hinzufügen

10.12.10 Modul deaktivieren

Um Module von der Bearbeitung auszuschließen, ist es möglich, diese zu deaktivieren. Ist ein Modul deaktiviert, so wird der Modulcode nicht mehr bearbeitet. Über die Methode "mSetModuleEnable()" können Module aktiviert bzw. deaktiviert werden.

Per Standard wird ein Modul beim Hinzufügen über den GAT-Wizard in der Methode "mOneTimeInit" des "Machine"-FB aktiviert:

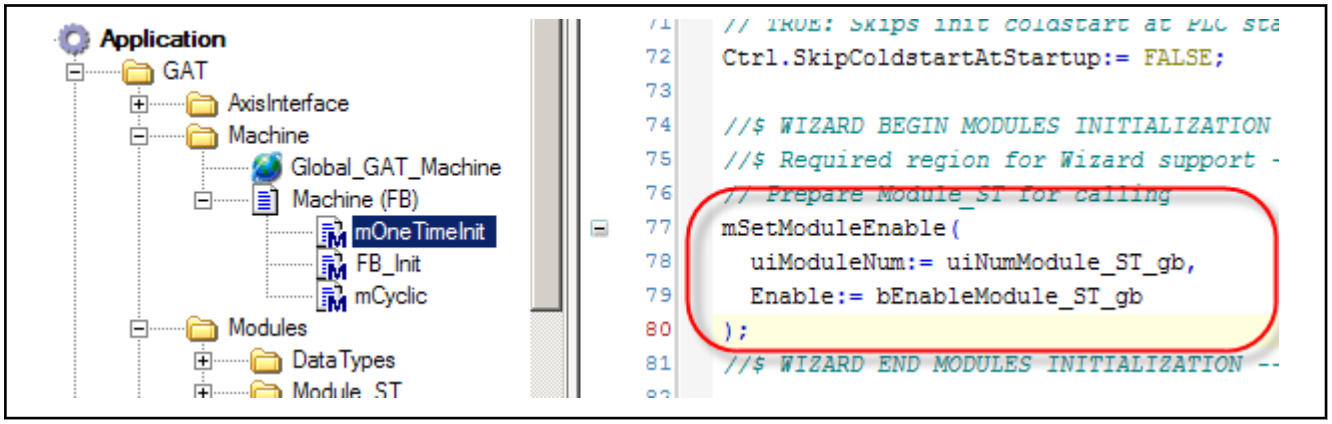


Abb.10-60: Modul aktivieren/deaktivieren

Ein Modul kann entweder direkt nach erneutem Herunterladen des SPS-Projektes, oder auch nachträglich zur Laufzeit deaktiviert werden. Durch Verändern des Initialwertes der "Enable"-Variable auf FALSE (siehe nachfolgende Abbildung) wird direkt nach dem SPS-Download das Modul deaktiviert. Durch dieses Vorgehen ist es beispielsweise möglich, auch für verschiedene Maschinenvarianten nur ein Projekt zu pflegen, und nicht vorhandene Module einfach abzuschalten.

GAT mit zentraler Zustandsmaschine

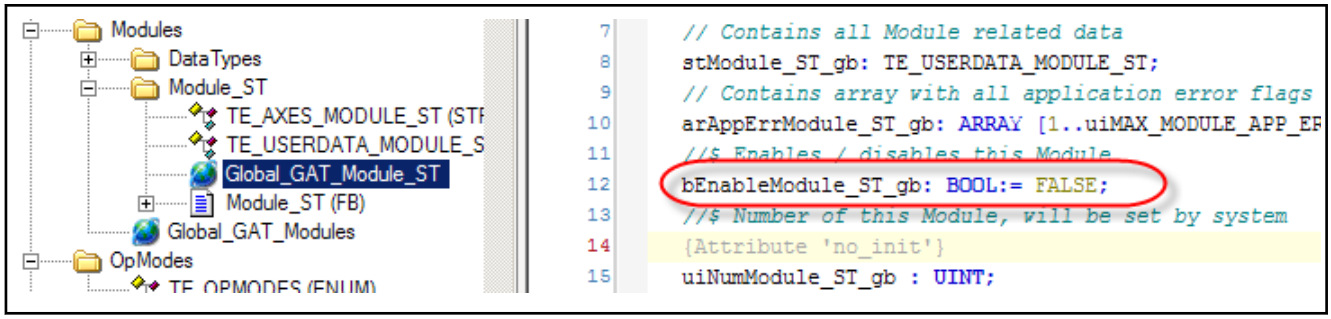


Abb.10-61: Modul aktivieren/deaktivieren durch Veränderung des Initialwertes

Auch bei laufendem Programm kann das Modul zur Laufzeit deaktiviert werden.

Hierbei ist folgende Vorgehensweise zu beachten:

- Zunächst muss die Maschine (und damit auch alle zugeordneten Module) durch den Anwender in den sicheren Initialzustand (z. B. Zustand "STOPPED" der Automatikbetriebsart) gebracht werden. Aus sicherheitstechnischen Gründen wird das Kommando zum Deaktivieren des Moduls nicht ausgeführt, wenn der sichere Initialzustand der Maschine nicht erreicht ist
- Über die Methode "mSetModuleEnable(Enable:=FALSE…)" wird anschließend aus dem Applikationsprogramm die Funktion zur Modul-Deaktivierung angestoßen
- Das GAT setzt das Steuerelement "DisableTransient" in der Modul-Steuerstruktur für mindestens einen Zyklus auf TRUE
- Im deaktivierten Modul wird darauf die Aktion "actDisable" bearbeitet. In dieser Aktion kann der notwendige Code zum Deaktivieren des Moduls implementiert werden. Die Aktion "actDisable" wird mindestens für einen Zyklus bearbeitet. Dauert die Modul-Deaktivierung mehrere Zyklen, so kann die Bearbeitung durch Setzen des Modul-Statuselementes "DisableLocked := TRUE" verlängert werden. Die Aktion "actDisable" wird solange bearbeitet bis "DisableLocked" wieder auf FALSE zurückgesetzt wird. Anschließend wird der Modulecode nicht mehr bearbeitet

10.12.11 Wiederverwendung von Modulen

Das GAT ist so aufgebaut, dass entwickelte funktionale Einheiten (Module) auch in anderen Projekten, die auf dem GAT aufbauen, wieder verwendet werden können.

Dazu ist einfach der Ordner des gewünschten Moduls auszuwählen und die dazugehörigen Daten unter Nutzung des Kontextmenüs zu exportieren:

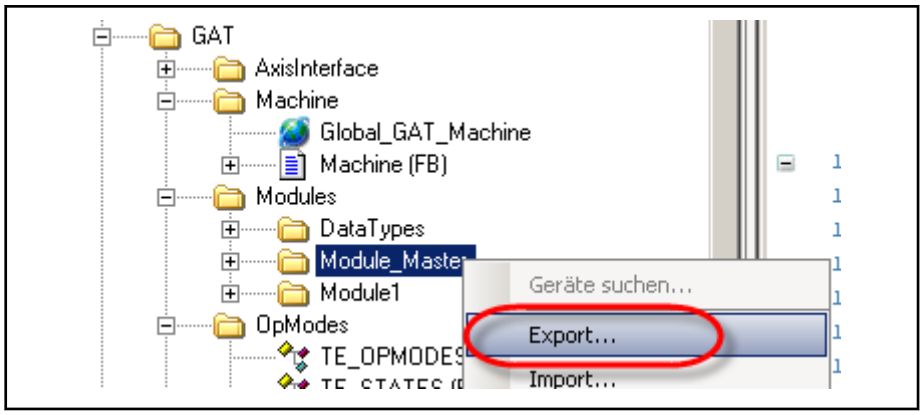


Abb.10-62: Modul exportieren

Im nachfolgenden Dialog sind alle Elemente auszuwählen.

Die beim Export erzeugte Datei ist in das neue Projekt zu importieren.

Dazu ist der Ordner "Modules" zu fokussieren und im dazugehörigen Kontextmenü die "Import"-Funktionalität zu nutzen:

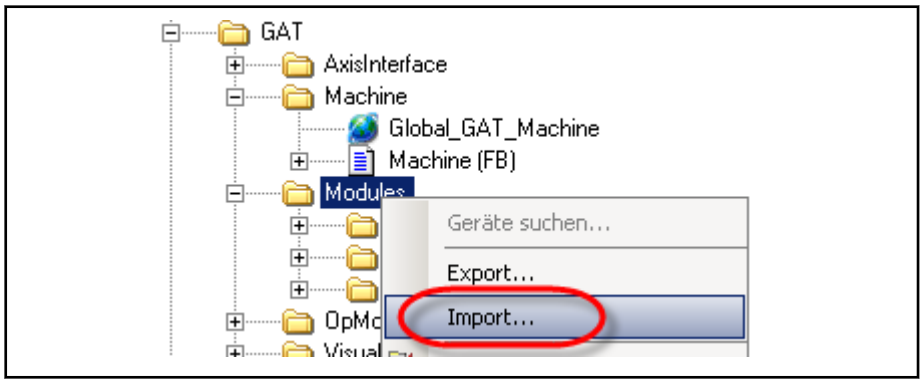


Abb.10-63: Modul importieren

GAT mit zentraler Zustandsmaschine

Bei der Verwendung (Import) des Moduls in einem anderen Projekt ist darauf zu achten, dass alle Zustände, die im Modul enthalten sind, auch in der neuen Maschine (Projekt) definiert sind:

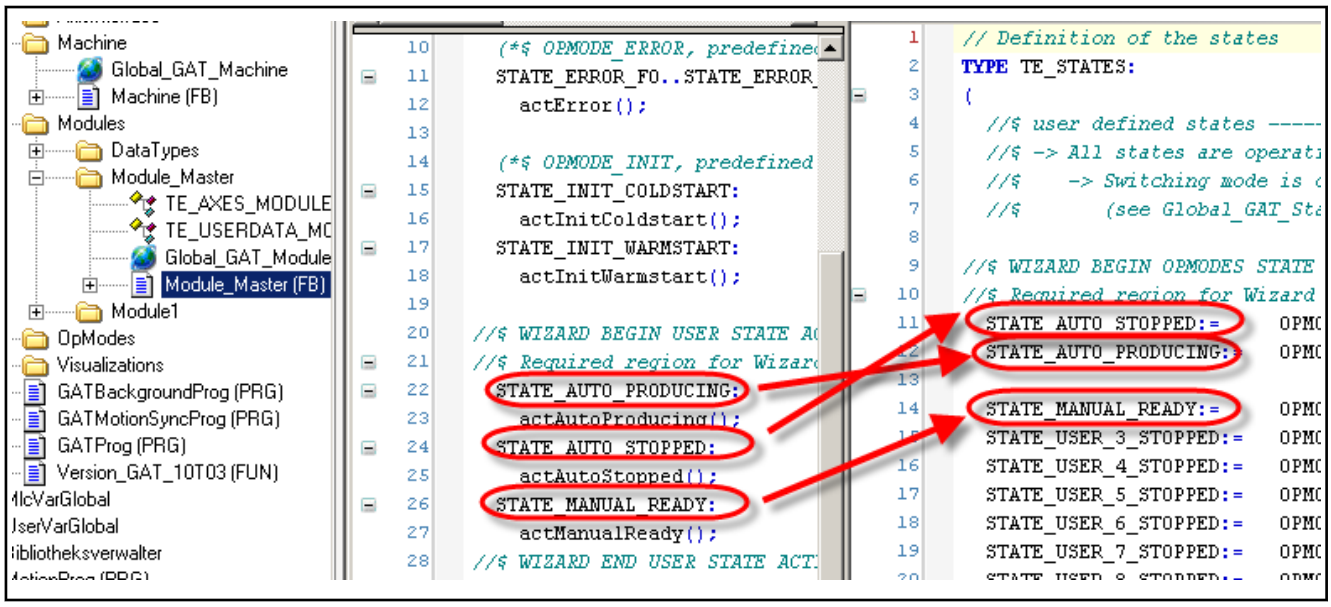


Abb. 10-64: Deklaration der Zustände

Nach dem Import des Moduls muss weiterhin die Methode "mOneTimeInit" des FB "Machine" mit einem zugehörigen Eintrag ergänzt werden, um das Modul zu aktivieren/deaktivieren:

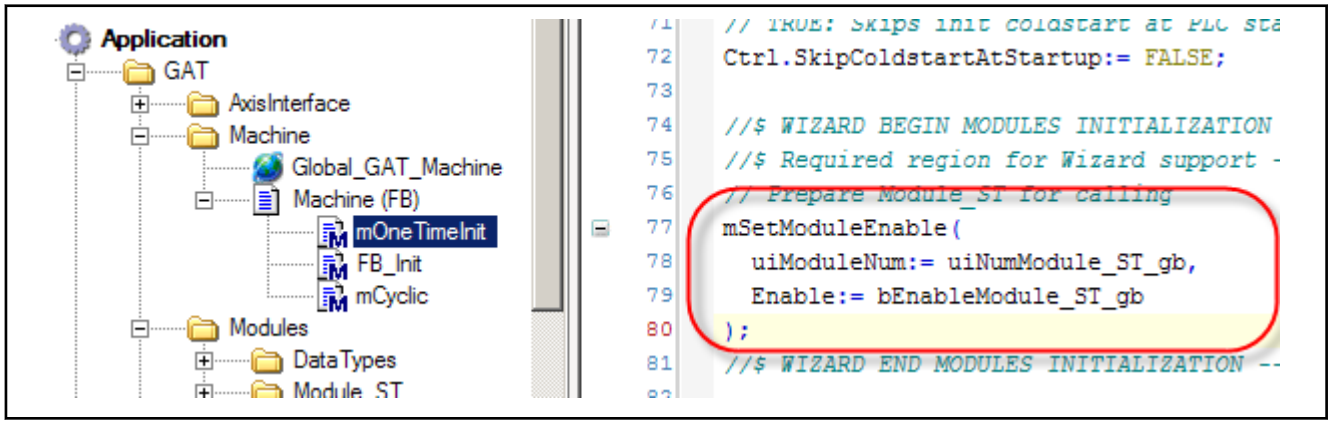


Abb. 10-65: Erweiterung der Methode "mOneTimeInit" des FB "Machine" nach Import eines Modules

10.12.12 Verwendung von globalen und Adressvariablen in Modulen

Wegen der Mehrfachverwendung eines Moduls innerhalb einer Applikation (Instanziierung), bzw. der Wiederverwendung in einer anderen Applikation, darf in einem Modul kein direkter Zugriff auf eine globale Variable oder Adressvariable erfolgen.

Alle globalen und Adressvariablen, die im Modul verwendet werden sollen, müssen als Interfacevariablen im Daten-Container **"UserData"** des Moduls deklariert werden. Die Zuweisung der globalen oder Adressvariable an das entsprechende Element im UserData-Container erfolgt einmalig in der Maschine. Der Zugriff auf das Element innerhalb des Moduls erfolgt durch **UserData.<Element>**.

Dieses Handling soll nachfolgend an einem Beispiel gezeigt werden:

In der globalen Variablenliste **MlcVarGlobal** sind zwei Variablen deklariert, ein globaler Eingang und ein globaler Ausgang:

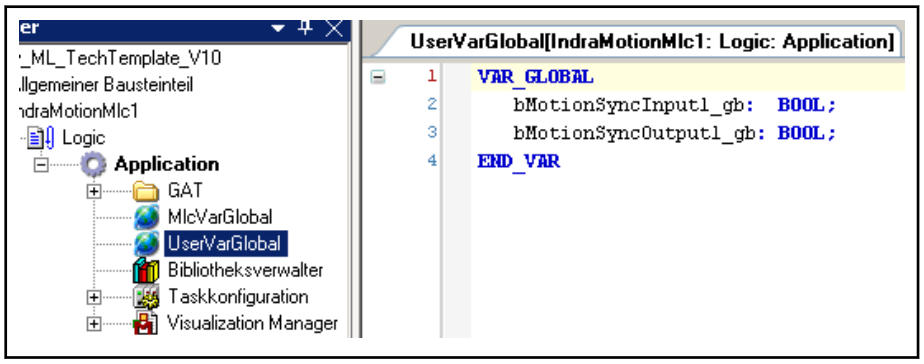


*Abb.10-66: Deklaration je einer booleschen Eingangs- und Ausgangsvariable in der globalen Variablenliste **UserVarGlobal***

Bei Bedarf können diese Variablen zusätzlich auf physikalische Adressen gemappt werden.

Im konkreten Fall werden die globale Eingangsvariable auf das erste Bit der Onboard-Eingänge und die globale Ausgangsvariable auf das erste Bit der Onboard-Ausgänge der Steuerung gemappt:

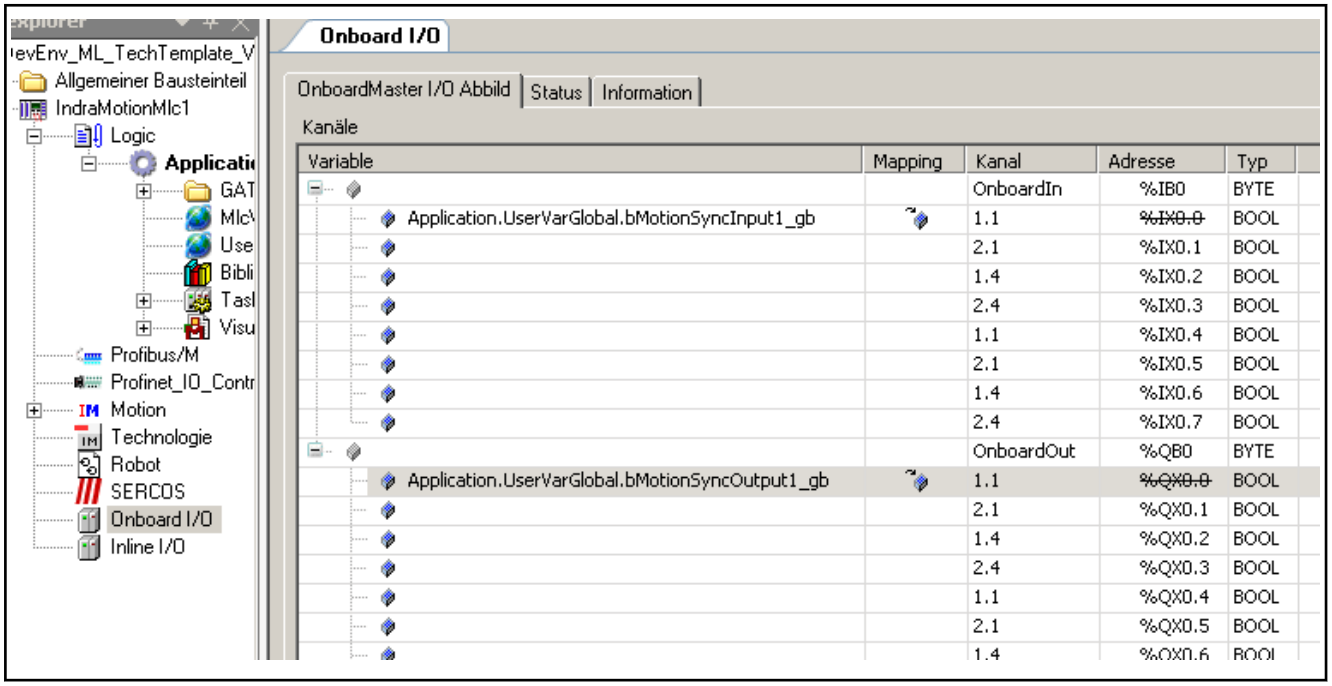


Abb.10-67: Mapping von booleschen globalen Variablen auf Onboard-Ein-/Ausgänge

Im Modul, in dem die globalen Variablen genutzt werden sollen, müssen in der UserData-Datenstruktur Referenzvariablen deklariert werden.

GAT mit zentraler Zustandsmaschine

Die Referenzvariablen müssen dem Typ der zu nutzenden globalen Variablen entsprechen:

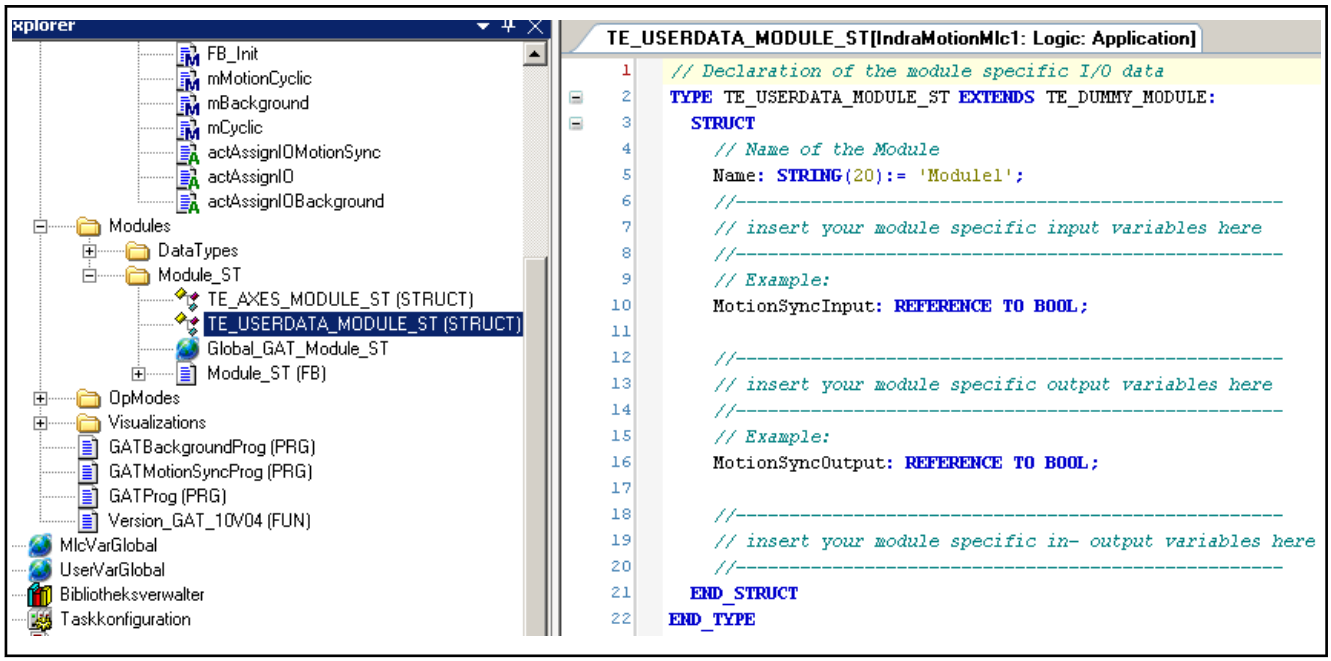


Abb.10-68: Deklaration von Referenzvariablen im Modul für die globalen Variablen, die im Modul genutzt werden sollen

Zuletzt muss die Verbindung zwischen den globalen Variablen und den entsprechenden Interfacevariablen (Referenzvariablen) des Moduls hergestellt werden. Grundsätzlich erfolgt die Aktualisierung von globalen bzw. Adressvariablen taskbezogen. Deshalb gibt es im FB "Machine" taskbezogene Aktionen, in denen die Verknüpfung der globalen und Adressvariablen mit den entsprechenden Modulvariablen erfolgt.

So müssen die folgenden taskbezogene Aktionen miteinander verbunden werden:

- **MotionSyncTask** zu aktualisierende Variablen in der Aktion **actAssignIOMotionSync**
- **PlcTask** zu aktualisierende Variablen in der Aktion **actAssignIO**
- **BackgroundTask** zu aktualisierende Variablen in der Aktion **actAssignIOBackground**

Die nachfolgende Abbildung zeigt, wie die eingangs deklarierten globalen Ein- bzw. Ausgangsvariablen mit den entsprechenden Modulvariablen verknüpft werden:

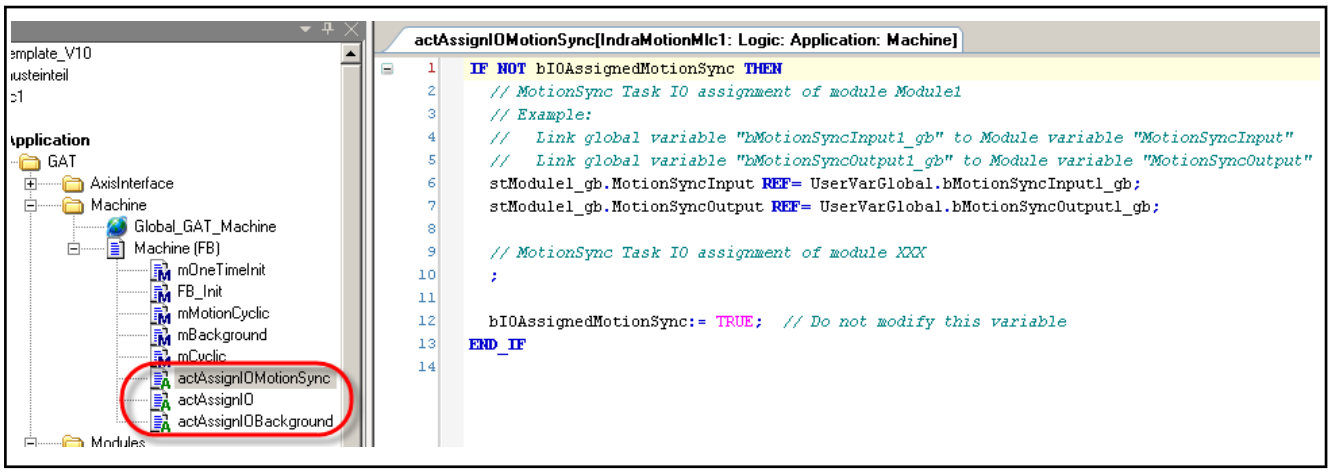


*Abb. 10-69: Verknüpfung des globalen Eingangs bzw. Ausgangs aus der globalen Variablenliste **UserVarGlobal** mit den entsprechenden Variablen im Datencontainer des Moduls **Module1***

Da diese Verknüpfung in der Aktion **actAssignIOMotionSync** erfolgt, werden diese Variablen mit der **MotionSyncTask** aktualisiert.

Basierend auf den vorangegangenen Deklarationen und Zuweisungen erfolgt jetzt der Zugriff auf die globale oder Adressvariable innerhalb des Moduls über den Datencontainer **UserData**:

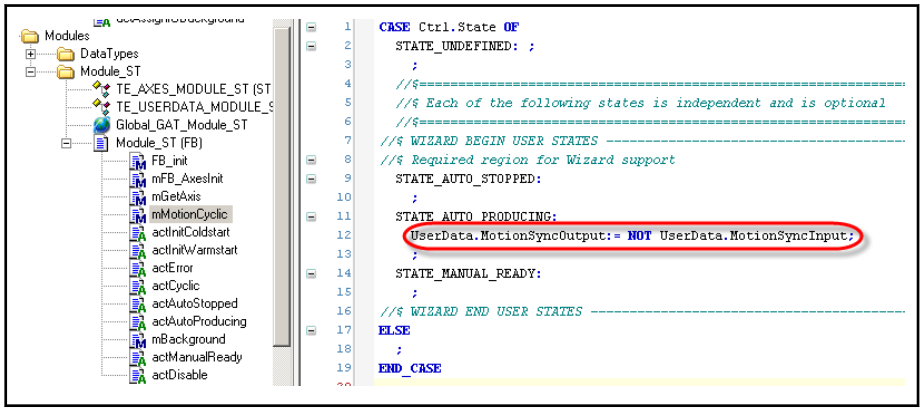


*Abb. 10-70: Zugriff auf globale bzw. Adressvariable innerhalb des Moduls über den modulbezogenen Datencontainer **UserData***

10.12.13 Abarbeitung in MotionTask

Für die Abarbeitung in der MotionTask bringt jedes Modul/Maschine die Methode **"mMotionCyclic"** mit. Aus dieser Methode kann auf alle Variablen des Moduls/Maschine (z. B. zentraler Achseinstiegspunkt Axes….) zugegriffen werden.

Wichtig ist dabei, dass auch die hier angesteuerten Achsen in der MotionTask aktualisiert werden (siehe auch Kap. 5.5 "Taskhandling" auf Seite 41).

10.12.14 Achs-Interface Erweiterungen

Das GAT erlaubt fast beliebige Erweiterungen der Achs-Interface-Strukturen. Es können zusätzliche Unterstrukturen eingefügt und die vorhandenen Unterstrukturen erweitert werden.

Die folgenden Strukturelemente sind als Anwendererweiterungen in der AxisCtrl-Struktur beispielhaft programmiert:

- SetupMode: zusätzliche Unterstruktur in der AxisCtrl-Struktur
  - Enable: Freigabe Einrichtbetrieb
  - JogPlus: Tippen +
  - JogMinus: Tippen –
  - Vel: Tipp-/Referenzfahrgeschwindigkeit
  - Accel: Tipp/Referenzfahr(brems)beschleunigung
  - Homing: aktiviert ModeHoming|SetAbsRef|ModesPosAbs abhängig vom Achstyp/-konfiguration
- SyncMode: Erweiterung der vorhandenen Unterstruktur in der AxisCtrl-Struktur
  - AngleOffset: Winkelverschiebung Tabellenanfang -> der Parameter A-0-2730 wird mit diesem Wert beschrieben
  - PhaseShift: Offset zur Leitachse vor dem Getriebe (PhaseOffset: Offset zur Leitachse nach dem Getriebe)
  - PhaseShiftVel: Geschwindigkeit zur Verstellung des PhaseShift
  - PhaseShiftAcc: Beschleunigung zur Verstellung des PhaseShift
  - SyncVelocity: Aufsynchronisiergeschwindigkeit -> der Parameter A-0-2790 wird mit diesem Wert beschrieben
  - SyncAcceleration: Aufsynchronisierbeschleunigung -> der Parameter A-0-2791 wird mit diesem Wert beschrieben

Die folgenden Strukturelemente sind als Anwendererweiterungen in der Axis-Status-Struktur beispielhaft programmiert:

- SetupMode: zusätzliche Unterstruktur in der AxisStatus-Struktur
  - EnableAck: Einrichtbetrieb ist aktiv
  - Homingack: Homing ist abgeschlossen
- SyncMode: Erweiterung der vorhandenen Unterstruktur in der AxisStatus-Struktur
  - AngleOffsetDone: Übertragung Winkelverschiebung Tabellenanfang ist abgeschlossen -> der Parameter A-0-2730 wurde beschrieben
  - PhasingMasterDone: der PhaseShift wurde verfahren
  - SyncVelocityDone: Übertragung Aufsynchronisiergeschwindigkeit ist abgeschlossen -> der Parameter A-0-2790 wurde übertragen
  - SyncAccelerationDone: Übertragung Aufsynchronisierbeschleunigung ist abgeschlossen -> der Parameter A-0-2791 wurde übertragen

Der Code zu diesen Erweiterungen ist in den Aktionen des Bausteins TE_AxisInterface() im Ordner "GAT/AxisInterface" zu finden. Die dazugehörigen Strukturen sind in den Ordnern "GAT/AxisInterface/Type/Control" und "GAT/AxisInterface/Type/Status" zu finden.

Es können eigene Erweiterungen hinzugefügt werden (siehe dazu Kap. 6.7 "Achs-Interface Anwender-Erweiterung" auf Seite 103).

10.12.15 Robot Control

Um Achsen für koordinierte Bewegungen ansteuern zu können, muss für das TechInterface der Funktionsbaustein MB_AxisInterfaceType**02** verwendet werden.

Hierzu muss folgendes im GATzentral angepasst werden:

- Die Struktur "TE_AXIS_CONTROL_Type01" muss in "TE_AXIS_CONTROL_TYPE02" umbenannt werden und um die Strukturvariable "CoordModeCtrl" vom Typ "MB_AXIS_COORDINATED" erweitert werden

Umbenennen der Struktur TE_AXIS_CONTROL_Type01 nach TE_AXIS_CONTROL_Type02:

```
//former type: TYPE TE_AXIS_CONTROL_TYPE01:
TYPE TE_AXIS_CONTROL_TYPE02:  // rename for RobotControl
STRUCT
    Admin:      TE_AXIS_ADMINISTRATION; (* Administration of the axis | Verwaltung der Achse *)

    PosMode:    TE_AXIS_POSITIONING; (* operating mode positioning (absolute, rel. and additive)
      Positionierbetrieb (absolut, relativ und additiv) *)

    VelMode:    TE_AXIS_VELOCITY; (* operating mode velocity control | Geschwindigkeitsregelung *)

    SyncMode:   TE_AXIS_SYNCHRONISATION; (* operating mode synchronisation (phase, vel, and cam)
      Synchronbetriebsarten (Winkelsync., Geschwindigkeitssync. und Kurvenscheibe) *)

    (* add your extensions below here, keep the elements above
       Erweiterungen hier einfuegen, Elemente oberhalb nicht veraendern *)

    (* user extensions | Anwender Erweiterungen *)

      // Jog/Homing functionality | Tipp/Einrichtfunktionen
    SetupMode:  TE_AXIS_SETUP_MODE;

    CoordModeCtrl:  MB_AXIS_COORDINATED;    // inserted for RobotControl

END_STRUCT
END_TYPE
```

- Umbenennen der "TE_AXIS_STATUS_Type01" Struktur in "TE_AXIS_STATUS_TYPE02"

Umbenennen der Struktur TE_AXIS_STATUS_Type01 nach TE_AXIS_STATUS_TYPE02:

```
    (********* Outputs of the following axis - from motion control to logic control **********)
    // former type: TYPE TE_AXIS_STATUS_TYPE01:
TYPE TE_AXIS_STATUS_TYPE02:     // Rename for RobotControl
  STRUCT
    (********* Outputs of the following axis - from motion control to logic control **********)

    PtrAxisData:  POINTER TO ML_AXISDATA_SM;
     (* Pointer to AxisData of this axis | Zeiger auf AxisData dieser Achse
      - Type .PtrAxisData^. for the intellisense menu | Eingabe .PtrAxisData^. fuer Auswahlmenue*)

    Admin:   TE_AXIS_ADMIN_STATUS; (* status of the axis | Achs-Status *)

    Diag:    TE_AXIS_DIAGNOSIS; (* axis diagnosis information, error, warning....
      Diagnose Informationen der Achse, Fehler, Warnung.... *)

    PosMode: TE_AXIS_POS_STATUS; (* Status operating mode positioning (absolute, rel. and add.)
      Statusmeldungen Positionierbetrieb (absolut, relativ und additiv) *)

    VelMode: TE_AXIS_VEL_STATUS; (* Status operating mode velocity control
      Statusmeldungen Geschwindigkeitsregelung (Freilauf) *)

    SyncMode: TE_AXIS_SYNC_STATUS; (* Status operating mode synchronisation (phase, vel, and cam)
      Statusmeldungen Synchronbetriebsarten (Winkelsync., Geschwindigkeitssync. und Kurvensch.)*)
```

GAT mit zentraler Zustandsmaschine

```
    (* add your extensions below here, keep the elements above
       Erweiterungen hier einfuegen, Elemente oberhalb nicht veraendern *)
    (* user extensions | Anwender Erweiterungen *)
          (* Jog/Homing functionality | Tipp/Einrichtfunktionen *)
    SetupMode:  TE_AXIS_SETUP_MODE_STATUS;

    // AxisData for HMI (without data type "BIT")
      AxisData_HMI:    TE_AXIS_DATA_STATUS;

  END_STRUCT
END_TYPE
```

- Der Funktionsbaustein "TE_Axisinterface" muss mit dem neuen FB "MB_AxisInterfaceType02" erweitert werden. "TE_AxisInterface EXTENDS MB_AxisInterfaceType02"

Erweiterung TE_Axisinterface:

```
//former: FUNCTION_BLOCK TE_AxisInterface EXTENDS MB_AxisInterfaceType01
FUNCTION_BLOCK TE_AxisInterface EXTENDS MB_AxisInterfaceType02
(*#########################################################################################
  #########################################################################################

  General Header
  -----------------------------------------------------------------------------------------
  Shortdescription       : AxisInterface Example
```

- Die Methode "mInitExtensions()": Der zusätzliche Eingang "CoordModeCtrl" des FB "MB_AxisInterfaceType02" muss initialisiert werden

CoordModeCtrl initialisieren:

```
SUPER^.mInitExtension(index);

IF NOT bFirstInit THEN
     // User: Assigment of user defined axis interface inputs
     SetupMode      REF= arAxisCtrl_gb[index].SetupMode;
     // inserted for RobotControl
     CoordModeCtrl  REF= arAxisCtrl_gb[index].CoordModeCtrl;
     // User: Assigment of user defined axis interface outputs
     SetupModeAck REF= arAxisStatus_gb[index].SetupMode;
     AxisData_HMI REF= arAxisStatus_gb[index].AxisData_HMI;

     // ================================================================
     // Do not touch the following lines
     // ================================================================

     // Set the references for the AxisInterface FB
     AdminCtrlExt  REF= arAxisCtrl_gb[index].Admin;
```

- Die Deklaration für die beiden Strukturvariablen "arAxisCtrl_gb" und "arAxisStatus_gb" muss auf die entsprechenden Type02-Strukturen geändert werden

Strukturvariablen anpassen:

```
VAR_GLOBAL
   // Axis interface: command structure: define as RETAIN, to keep the values
   // Note: on the L40.2 control, the RETAIN memory is slower then normal VAR, so define
   //       this structure only as RETAIN when necessary
   //  arAxisCtrl_gb: ARRAY [1..MAX_AXIS_NUM] OF MB_AXIS_CONTROL;

   // Axis interface: command structure
   //former: arAxisCtrl_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_CONTROL_TYPE01;
     arAxisCtrl_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_CONTROL_TYPE02;
END_VAR

VAR_GLOBAL
//---------------------------------- Machine status variables ----------------------------

  // Axis interface: status structure
  //former: arAxisStatus_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_STATUS_TYPE01;
  arAxisStatus_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_STATUS_TYPE02;
  bRemoteOn_gb   : BOOL := TRUE;  // TRUE: Internal IL visu is active
                                  // FALSE: PlcProg is active
END_VAR
```



Die Strukturen TE_AXIS_CONTROL_TYPE02, TE_AXIS_STATUS_TYPE02 befinden sich im Ordner: "AxisInterface\Type".

Die Strukturvariablen arAxisCtrl_gb, arAxisStatus_gb befinden sich in der globalen Variablenliste "UserVarGlobal".

Kinematik-Interface

Aufbauend auf das Achs-Interface steht das Kinematik-Interface als optionales Plug-in zur Verfügung. Über das Kinematik Interface können Kinematiken analog zu Achsen kommandiert werden. Fügen Sie hierzu das Kinematik Interface über den GAT Wizard als Plug-in hinzu.

Die Dokumentation des Kinematik Interface ist in dem Plug-in Konfigurationsordner enthalten.

Den Link zum Konfigurationsordner können Sie über den GAT Wizard wie folgt finden:

- Im GAT Wizard den Ordner "Plug-ins" selektieren
- Im Wizard Bedienfeld ist jetzt der Link "Plug-ins Konfigurationsordner öffnen" sichtbar

10.12.16 Generierung eigener Programmrahmen auf Basis des GAT

Eine auf Basis des GAT entwickelte Applikation kann wiederum als Vorlage (Programmrahmen) zur Entwicklung einer anderen, komplexeren Applikation dienen. Auf diese Weise kann ein Anwender z. B. für unterschiedliche Maschinentypen jeweils einen eigenen Programmrahmen zur Verfügung stellen, der sich aber von dem Standard-GAT durch umfangreiche Weiterentwicklungen unterscheidet (z. B. zusätzliche Betriebsarten und Module). Zur Arbeit am weiterentwickelten Programmrahmen kann dann wieder der GAT-Wizard genutzt werden.

Zur Generierung einer eigenen Vorlage auf Basis einer Applikation, die mit dem GAT entwickelt wurde, muss folgendermaßen vorgegangen werden:

1. Gesamten GAT-Ordner exportieren, alle Checkboxen angewählt lassen (*.iwx Datei erzeugen). Im Export-Dialog ist ein Eingabefeld "Beschreibung" vorhanden. Der hier eingetragene Text kann bei der Import-Funktion des GAT-Wizard angezeigt werden.

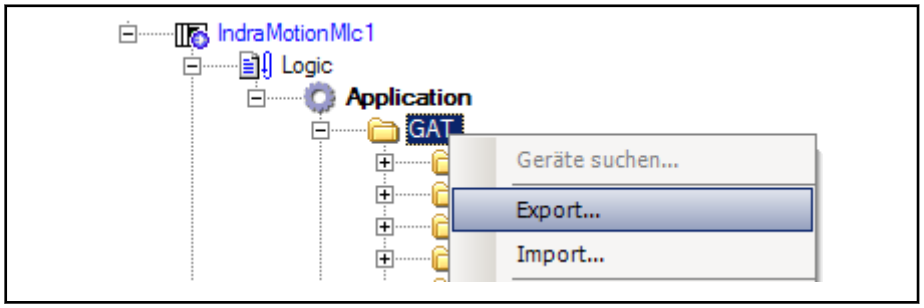


Abb. 10-71: GAT-Ordner exportieren

2. Die im Schritt 1 erzeugte *.iwx Datei beinhaltet das angepasste/ergänzte Template und kann nun beim Neuanlegen von Projekten über den GAT-Wizard wie folgt genutzt werden:

GAT mit zentraler Zustandsmaschine

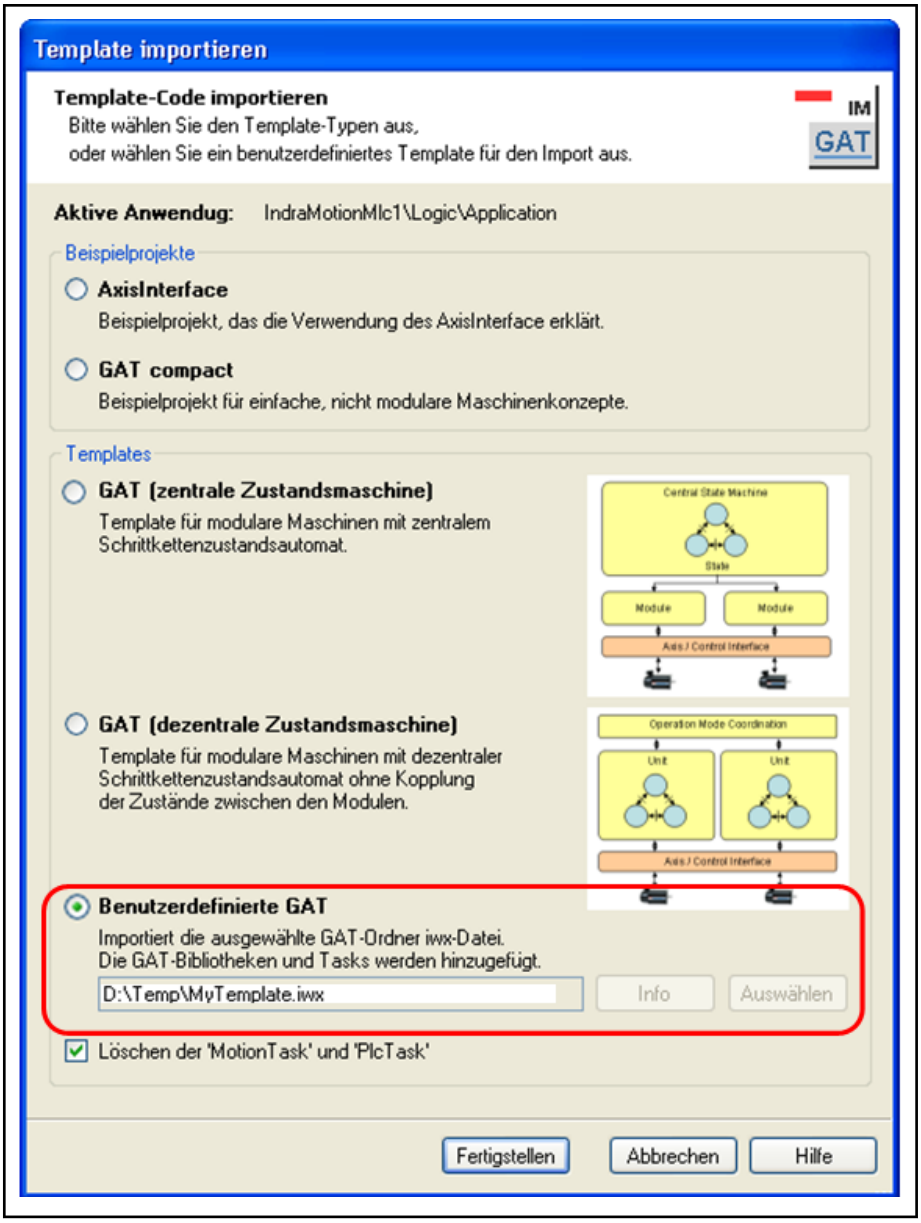


Abb. 10-72: Benutzerdefiniertes Template importieren

10.12.17 PackML V3.0 Programmrahmen basierend auf GAT

Mit der Installation wird auch ein vorgefertigter Programmrahmen bereitgestellt, der die in PackML V3.0 definierten Modes und States abbildet.

Der PackML V3.0 Programmrahmen kann über die Auswahl **Benutzerdefiniertes GAT** nach Aufruf des GAT-Wizard geladen werden, indem die Datei "GAT-PackML-V3.iwx" zum Import ausgewählt wird. Die Datei "GAT-PackML-V3.iwx" wird im Rahmen der IndraWorks Installation mit ausgeliefert, und ist im entsprechenden GAT-Wizard Verzeichnis abgelegt.

Alle PackML V3.0 Definitionen findet man unterhalb des GAT-Ordners im Ordner "PackML_V3":

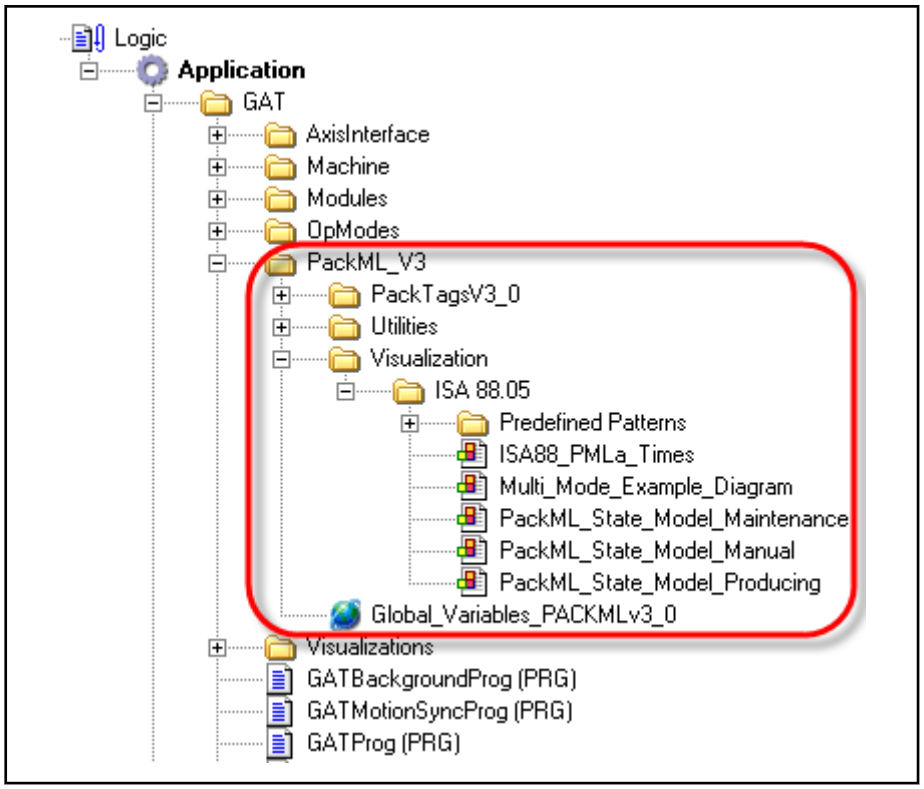


Abb.10-73: Ort der PackML V3.0 Definitionen im vordefinierten Programmrahmen

Neben den in PackML V3.0 definierten Modes und States beinhaltet diese Programmvorlage auch die in dem Standard definierten Visualisierungen:

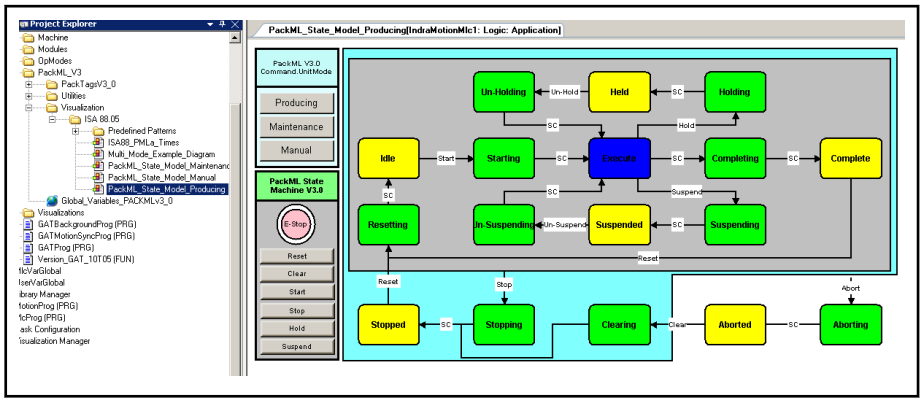


*Abb.10-74: PackML V3.0 Visualisierung für den Mode **Producing** innerhalb der vorgefertigten Programmvorlage*

Da diese Programmvorlage auf dem GAT beruht, gelten alle in diesem Dokument zum GAT genannten Regeln und Möglichkeiten zur Bearbeitung.

Um **transiente** PackML-Zustände (<statename>ing) zu implementieren, muss der entsprechende Code im **transienten** Teil einer zustandskorrenspondierenden GAT-Aktion platziert werden (siehe Kap. 10.7.4 "Zustände im Modul" auf Seite 299).

Um **stationäre** PackML-Zustände zu implementieren, muss der entsprechende Code im **stationären** Teil einer zustandskorrenspondierenden GAT-Aktion platziert werden (siehe Kap. 10.7.4 "Zustände im Modul" auf Seite 299).

Für weiterführende Informationen zu PackML V3.0 siehe www.isa.org.

10.12.18 Zeitkritische applikationsspezifische Fehlerreaktion

In der Standard-Auslieferung des GAT werden die Betriebsarten und Zustandsmaschinen im Kontext der "GATProgTask" (10ms zyklische Task) bearbeitet. Ein Fehler wird damit in diesem relativ langen Zeitintervall erkannt und behandelt.

Für Applikationen, die eine schnellere Fehlerreaktion erfordern, wird empfohlen die Betriebsarten und Zustandsmaschinen im Kontext der "GATMotionSyncTask" (zyklische Bearbeitung synchron zum Motion-Zyklus) zu bearbeiten. Das Zeitintervall zwischen Fehlererkennung und Fehlerreaktion beträgt dann ein Motion-Zyklus.

Folgende Schritte sind hierzu notwendig:

1. Löschen der "GATProgTask" in der Taskkonfiguration.
2. Zusätzlicher Programmaufruf "GATProg" in der "GATMotionSyncTask".

10.12.19 GAT Fehlerreaktion für F0 Fehler abschalten

Bei Fehlern mit dem Schweregrad "F0" erfolgt steuerungsseitig keine automatische Fehlerreaktion und Stillsetzung der Achsen durch die Steuerungsfirmware. Das GAT erkennt jedoch den Fehler, unterbricht die aktuell angewählte Betriebsart, und verzweigt in die Fehlerbehandlung. Mit den Standardeinstellungen des GAT werden in dieser Fehlerbehandlung die Achsen mit Rampe per "ModeAH"-Befehl verzögert.

*Soll das GAT auf F0-Steuerungsfehler nicht reagieren, so muss der nachfolgende Code in die Methode "**mCyclic**" des "Machine" FB implementiert werden:*

```
Ctrl.DisableErrorHandling :=  TRUE;  // Turn error reaction off

MB_IMCInterface(); // Refresh control diagnosis number

IF (TE_GetErrorSeverity(IMCStatus.Diag.Number) > 16#F0) THEN
  // Turn error reaction on if errors > F0 appear
  Ctrl.DisableErrorHandling :=  FALSE;
END_IF
```

10.12.20 Update von Version 10/11 auf 12/13

Ein Update des GAT von Version 10/11 auf 12/13 ist ohne Änderungen im Applikationsprogramm möglich, da die Funktionen und Schnittstellen kompatibel sind.

Der GAT-Wizard in Version 12/13 erfordert die aktuelle Steuerungsfirmware Version 12VRS/13VRS. Daher ist ein Update der Steuerungsfirmware bei

Steuerungen mit Firmware 10/11 durchzuführen, wenn der GAT-Wizard verwendet werden soll.

Mit folgenden Schritten wird die Steuerungsfirmware und die Steuerungskonfiguration aktualisiert:

1. Firmwareupdate auf Version 12/13 über Firmwareverwaltung durchführen
2. In der Steuerungskonfiguration Firmware-Version 12VRS/13VRS auswählen
3. Im Bibliotheksverwalter die Bibliotheken ML_TechInterface und ML_TechTemplate der Version 12/13 auswählen (per Platzhalter TECHINTERFACE, bzw. TECHTEMPLATE)

10.13 GAT Kurzübersicht

10.13.1 Strukturübersicht

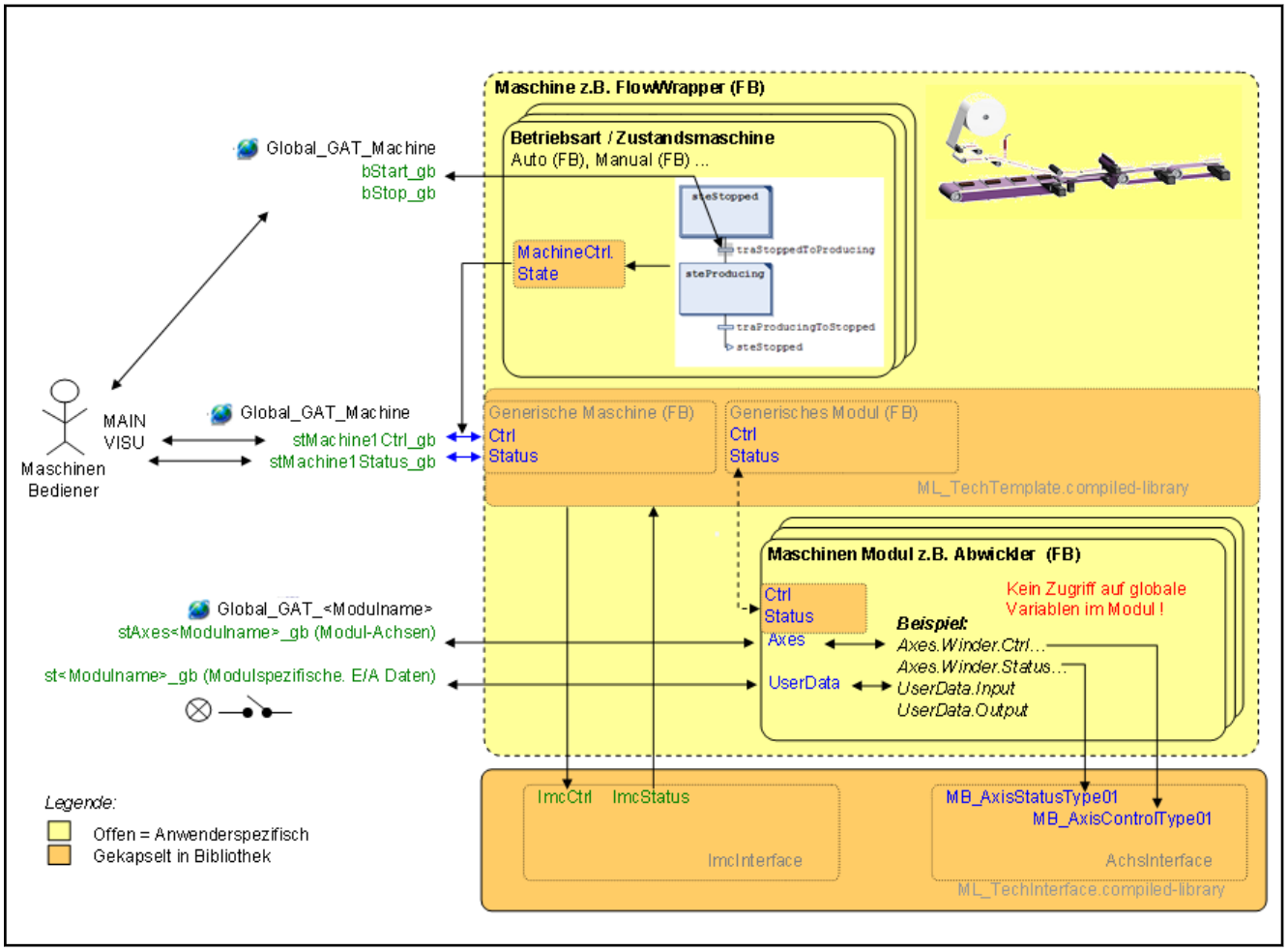


Abb. 10-75: GAT Strukturübersicht

GAT mit zentraler Zustandsmaschine

10.13.2 Übersicht der wichtigsten globalen Variablen der Maschine

Siehe globale Variablenliste **„GAT/Machine/Global_GAT_Machine”**:

Variable	Beschreibung	Zugriff
stMachine1Ctrl_gb	Steuerung der Maschine (z. B. Betriebsartenanwahl, zentrales "Fehlerlöschen", Nachfolge-Betriebsart nach Initialisierung/Fehler Reset, Abschalten des Errorhandling, Initialisierungs-Einstellungen)	RD/WR
stMachine1Status_gb	Status der Maschine (z. B. aktuelle Betriebsart und Zustand, Error und Diagnoseausgänge)	RD
arAppErrMachine1_gb	Array of BOOL mit anwenderspezifischen Fehlerflags der Maschine	RD
arMachine1ModuleDiagNo_gb	Array mit den Diagnosenummern aller Module	RD
bStart_gb	Startsignal der Auto-Betriebsart/Übergangstransition von AUTO_STOPPED nach AUTO_PRODUCING	RD/WR
bStop_gb	Stoppsignal der Auto Betriebsart/Übergangstransition von AUTO_PRODUCING nach AUTO_STOPPED	RD/WR

RD Lesen
WR Schreiben

10.13.3 Übersicht der wichtigsten globalen Variablen der Module

Siehe globale Variablenliste **„GAT/Moduls/Global_GAT_<Modulname>”**:

Variable	Beschreibung	Zugriff
st<Modulname>_gb	Modulspezifische I/O	RD/WR
arAppErr<Modulname>_gb	Array of BOOL mit anwenderspezifischen Fehlerflags des Moduls	RD
stAxes<Modulname>_gb	Container mit Modulachsen	RD

RD Lesen
WR Schreiben

10.13.4 Übersicht der wichtigsten globalen Variablen der Achsen

Siehe globale Variablenliste **„GAT/AxisInterface/ Global_GAT_AxisInterfaces”**:

Variable	Beschreibung	Zugriff
arAxisCtrl_gb	Array mit den Achs-Interface Kommando-Daten	RD/WR
arAxisStatus_gb	Array mit den Achs-Interface Status-Daten	RD

RD Lesen
WR Schreiben

10.13.5 Übersicht der Methoden und Aktionen der Maschine

Aktion/Methode	Aufruf	Zugehörige Task	Beschreibung/Beispiel
mMotionCyclic	In jedem Zustand	GATMotionSyncTask	Zeitkritische modulübergreifende Logik im Kontext der GATMotionSyncTask (z. B. modulübergreifende zeitkritische Überwachungen)
mCyclic	In jedem Zustand	GATProgTask	Zyklische modulübergreifende Logik im Kontext der GATProgTask
mBackground	In jedem Zustand	GATBackgroundProgTask	Nicht zeitkritische modulübergreifende Hintergrundaktivitäten im Kontext der GATBackgroundProgTask (z. B. HMI Daten Aufbereitung, Statistik)
mOneTimeInit	Bei Initialisierung des GAT für einen Zyklus. Ausführung vor "ColdStart"	GATProgTask	Module Aktivieren/Deaktivieren, Einstellungen für "SkipColdStart" initialisieren, Zuordnung Modulachsen zu den globalen IndraWorks-Achsen
mExitOpMode	Für einen Zyklus bei verlassen einer Betriebsart	GATProgTask	Methode ist optional: Aktivitäten bei Verlassen einer Betriebsart (z. B. Variablen zurücksetzen)
mEntryOpMode	Für einen Zyklus bei Aktivierung einer Betriebsart	GATProgTask	Methode ist optional: Aktivitäten bei Aktivierung einer Betriebsart (z. B. Variablen initialisieren)
mOpModeChangeAllowed	Vor Betriebsartenwechsel	GATProgTask	Methode ist optional: Die Methode wir vor Betriebsartenumschaltung vom GAT aufgerufen. Der Betriebsartenwechsel wird blockiert, solange der Rückgabewert = FALSE ist
mIsCriticalError	Im Fehlerfall	GATProgTask	Wird im Fehlerfall aufgerufen und erlaubt das Abweisen eines Fehlers für die Maschine
actAssignIOMotionSync	Einmalig nach SPS-Programmstart/Reset	GATMotionSyncTask	E/A-Zuordnung an die "GATMotionSyncTask"
actAssignIO	Einmalig nach SPS-Programmstart/Reset	GATProgTask	E/A-Zuordnung an die "GATProgTask"
actAssignIOBackground	Einmalig nach SPS-Programmstart/Reset	GATBackgroundProgTask	E/A-Zuordnung an die "GATBackgroundProgTask"
FB_init	Einmalig vor SPS-Programmstart/Reset	keine bestimmte Taskzuordnung	Systemintern, darf nicht modifiziert werden

GAT mit zentraler Zustandsmaschine

10.13.6 Übersicht der Methoden und Aktionen der Module

Aktion/Methode	Aufruf	Zugehörige Task	Beschreibung/Beispiel
actInitColdstart	Nur im Zustand INIT_COLDSTART	GATProgTask	Initialisieren/Schreiben von Parametern im Parametermodus (P2)
actInitWarmstart	Nur im Zustand INIT_WARMSTART	GATProgTask	Initialisieren/Schreiben von Parametern, Leistungszuschaltung der Achsen
actAutoStopped	Nur im Zustand AUTO_STOPPED	GATProgTask	Achsbewegung anhalten
actAutoProducing	Nur im Zustand AUTO_PRODUCING	GATProgTask	Achsbewegung starten
actManualReady	Nur im Zustand MANUAL_READY	GATProgTask	Überwachungen beim Vertippen der Achsen
actError	Nur im Zustand ERROR_F0.. ERROR_F9	GATProgTask	Zusätzliche anwenderspezifische Fehlerreaktion (z. B. synchronisiertes Verzögern der Achsen)
mMotionCyclic	In jedem Zustand	GATMotionSyncTask	Zyklischer Aufruf im Kontext der GATMotionSyncTask. Aufruf von zeitkritischen Prozessregler-FB (z. B. RegisterController)
mCyclic	In jedem Zustand	GATProgTask	Zyklischer Aufruf im Kontext der GATProgTask für modulspezifische zyklische Logik
mBackground	In jedem Zustand	GATBackgroundProgTask	Zyklischer Aufruf im Kontext der GATBackgroundProgTask für nicht zeitkritische Hintergrundaktivitäten (z. B. HMI Daten-Aufbereitung, Statistik)
mIsMyAxis	Kein automatischer Aufruf	Keine bestimmte Taskzuordnung	Prüft, ob eine Achse zum Modul gehört
mIsCriticalError	Im Fehlerfall	GATProgTask	Wird im Fehlerfall aufgerufen und erlaubt das Abweisen eines Fehlers für dieses Modul
mForceOpMode	Kein automatischer Aufruf	Keine bestimmte Taskzuordnung	• Setzt das Modul in eine andere Betriebsart als durch die Maschine vorgegeben wird • Setzt die Betriebsart des Moduls auf die Betriebsart der Maschine zurück
mExitOpMode	Für einen Zyklus bei verlassen einer Betriebsart	GATProgTask	Methode ist optional: Aktivitäten bei Verlassen einer Betriebsart (z. B. Variablen zurücksetzen)
mEntryOpMode	Für einen Zyklus bei Aktivierung einer Betriebsart	GATProgTask	Methode ist optional: Aktivitäten bei Aktivierung einer Betriebsart (z. B. Variablen initialisieren)
actDisable	In jedem Zustand beim deaktivieren eines Moduls bis die Deaktivierung abgeschlossen ist	GATProgTask	Zurücksetzen von Variablen bzw. Modul-Ausgängen bei Modul-Deaktivierung

Aktion/Methode	Aufruf	Zugehörige Task	Beschreibung/Beispiel
mGetAxis	Kein automatischer Aufruf	keine bestimmte Taskzuordnung	Systemintern, darf nicht modifiziert werden. Liefert den Pointer zu einer Modul-Achse
mFB_AxesInit	Bei Achsinitialisierung	GATProgTask	Systemintern, darf nicht modifiziert werden
FB_init	Einmalig vor SPS-Programmstart/Reset	keine bestimmte Taskzuordnung	Systemintern, darf nicht modifiziert werden

10.13.7 Zugriff auf Achsen innerhalb des Moduls

☞ Zugriff immer über **Axes.<Achsname>.**

Beispiel:

```
 // Achse in Geschwindigkeitsregelung kommandieren
Axes.MyAxis1.Ctrl.Admin._OpMode.en := ModeVel;

// Wenn Achse die Geschwindigkeit erreicht hat
IF (Axes.MyAxis1.Status.PtrAxisData^.Axis_InVelocity = TRUE) THEN;
```

10.13.8 Zugriff auf I/O-Daten innerhalb des Moduls

☞ Zugriff immer über **UserData.<Variable>**. Nicht auf globale Variablen zugreifen!

Beispiel:

```
IF (UserData.bInput1=TRUE) THEN // Wenn „bInput1“ gesetzt ist…
  UserData.bOutput1:=TRUE; // Ausgang „bOutput1“ setzen
END_IF
```

11 GAT mit dezentraler Zustandsmaschine

11.1 Allgemeines

Das **G**eneric **A**pplication **T**emplate (GAT) mit dezentraler Betriebsarten-Schrittkette ist ein einfacher, skalierbarer und einschaltfertiger IndraWorks-Programmrahmen auf Basis der IEC61131-3, der bereits alle relevanten Komponenten zur Steuerung einer Maschine und deren Antriebe enthält.

Der in IndraWorks integrierte GAT-Wizard beschleunigt die Einarbeitung, da der Anwender sofort an alle relevanten Codestellen geführt wird.

Das GAT beinhaltet bereits alle wichtige Basisfunktionen einer Maschine, wie Betriebsartenverwaltung, Fehlerbehandlung, Diagnosesystem, Produkt- und Maschinendatenverwaltung, bis hin zur vereinfachten Motion Kommandierung über das Achs-Interface. Durch die vorbereiteten Visualisierungen ist das GAT bereits in der Grundausführung intuitiv bedienbar.

Durch definierte Schnittstellen und Wizardunterstützung sind Anpassungen und Ergänzungen leicht möglich.

Das GAT mit dezentraler Betriebsarten-Schrittkette erfordert eine IndraMotion MLC-Steuerung.

11.2 Überblick

11.2.1 Grundidee

Die Anwendung des GAT beruht auf dem Ansatz, dass eine Applikation auf einer Maschine läuft, die eine oder mehrere funktionale Einheiten (im folgenden "Unit" genannt) koordiniert. Im einfachsten Fall wird den Units die angewählte Betriebsart weitergeleitet. Es besteht jedoch auch die Möglichkeit, den Units jeweils unterschiedliche Betriebsarten vorzugeben.

Das GAT besteht aus folgenden Elementen:

GAT mit dezentraler Zustandsmaschine

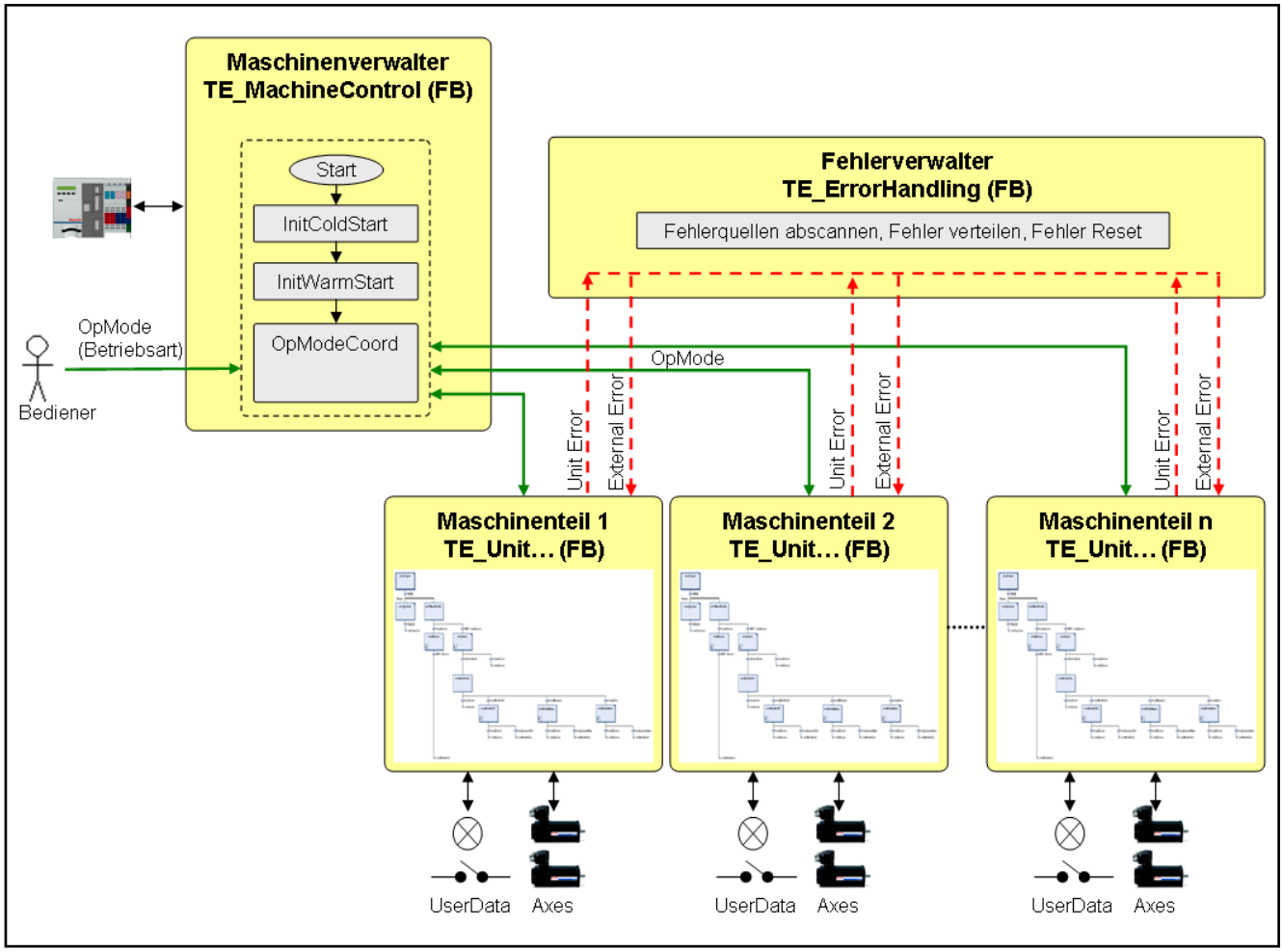


Abb.11-1: Softwarearchitektur des GAT mit dezentraler Zustandsmaschine

Maschinenverwalter

Der Maschinenverwalter (Funktionsbaustein **TE_MachineControl**) leitet die von dem Bediener angewählte Betriebsart an die jeweiligen Units weiter. Während der System-Initialisierung kommandiert er die Phasenumschaltung der Steuerung, und führt dabei alle Units durch die jeweiligen Initialisierungsschritte. Nach Abschluss der Initialisierung wird im einfachsten Fall allen Units die vom Bediener vorgegebene Betriebsart weitergegeben.

Der Maschinenverwalter ist in folgende Teile unterteilt:

- Ablaufsteuerung für die Initialisierung (Action **actOPModeSFC**)
- Datenverwalter für Visualisierung- und Feldbusanbindung, mit Maschinen und Produktdatenverwaltung (Methode **mDataHandler**)
- Hauptbetriebsarten-Bestimmung (Methode **mOpModeSet**)
- Betriebsarten-Verteilung zu den jeweiligen Units (Methode **mOpModeDistribution**)
- Permanent zyklische Bearbeitung (Methode **mCyclic**)

Fehlerverwalter

Der Fehlerverwalter (Funktionsbaustein **TE_ErrorHandling**) ist die zentrale Stelle an der alle Fehler gesammelt, und weiterverteilt werden. Weiterhin wird von dem Fehlerverwalter das zentrale Fehlerlöschen ausgeführt.

Der Fehlerverwalter ist hierbei in folgende Teile unterteilt:

- Im Teil **Fehlererkennung** werden alle bekannten Fehlerquellen (wie z. B. Steuerungs-, Unit-, Feldbus-, Inline-Fehler) abgescannt (Methode **mScanError**)
- Im Teil **Fehlerverteilung** werden die Fehlerinformationen an alle Units verteilt. Hierbei ist es auch möglich diese nur selektiv an bestimmte Units weiterzuleiten (Methode **mDistributeError**)
- Im Teil **Fehlerquittierung** wird der Auftrag zum Fehler löschen an alle unterlagerten Fehlerquellen weitergegeben (Methode **mClearError**)

Units Ein Maschinenteil, im folgenden Unit genannt, ist ein gekapselter Teil der Maschine um die Software zu modularisieren (Funktionsbaustein **TE_Unit**...). Alle Maschinenteile haben eine einheitliche Schnittstelle, um einen einfachen Austausch und Wartung sicherzustellen.

Jede Unit fragt die Fehlerinformationen der zugehörigen Komponenten (wie z. B. Achsen) ab und leitet diese Information an den Fehlerverwalter weiter. Zusätzlich wird in der Unit die **Fehlerreaktion** ausgeführt.

11.2.2 Voraussetzungen zur Anwendung

Wie im vorhergehenden Kapitel bereits ansatzweise aufgezeigt, sollen hier die Voraussetzungen vor der Verwendung des GAT noch einmal zusammengefasst werden.

Bevor eine Motion-Applikation mit dem GAT entwickelt wird, muss der Anwender hinsichtlich seiner Applikation folgende Fragen klären:

- Welche Betriebsarten sollen realisiert werden?
- Aus welchen funktionalen Einheiten (Units) setzt sich die Maschine zusammen. Welche Achsen und I/O gehören zu der jeweiligen funktionalen Einheit?

Mit diesen Informationen kann der Wizard in kürzester Zeit das Code-Gerüst der konkreten Motion-Applikation erzeugen.

Im Folgenden wird beschrieben, welche Teile zum GAT gehören, wie diese Teile zusammenarbeiten und wo der Anwender zur Realisierung seiner konkreten Applikation Ergänzungen vornehmen muss.

11.3 Erste Schritte – GAT mit dezentraler Zustandsmaschine

11.3.1 Einfache Motion-Applikation erstellen

Vorbereitung

Mit der nachfolgenden Beschreibung können sie in wenigen Minuten über den GAT Wizard eine Motion Logic Applikation mit 2 Units und 2 Achsen erstellen.

Sie benötigen hierfür eine IndraMotion MLC Steuerung, reale Antriebe sind nicht erforderlich. Grundlage ist das GAT mit dezentraler Betriebsarten Schrittkette.

Aufgabenstellung

- Eine Maschine soll aus zwei funktionalen Einheiten (Units) bestehen.
- In der Unit mit dem Namen "UnitMaster" soll sich in der "Automatik"-Betriebsart die Unitachse "Master_Axis" kontinuierlich mit 100U/min bewegen.
- In der Unit mit dem Namen "Unit1" soll sich in der "Automatik"-Betriebsart die Unitachse "Slave_Axis" winkelsynchron zu der kontinuierlich bewegenden Achse "Master_Axis" bewegen.

GAT mit dezentraler Zustandsmaschine

Nachfolgende Abbildung zeigt schematisch die Aufgabenstellung.

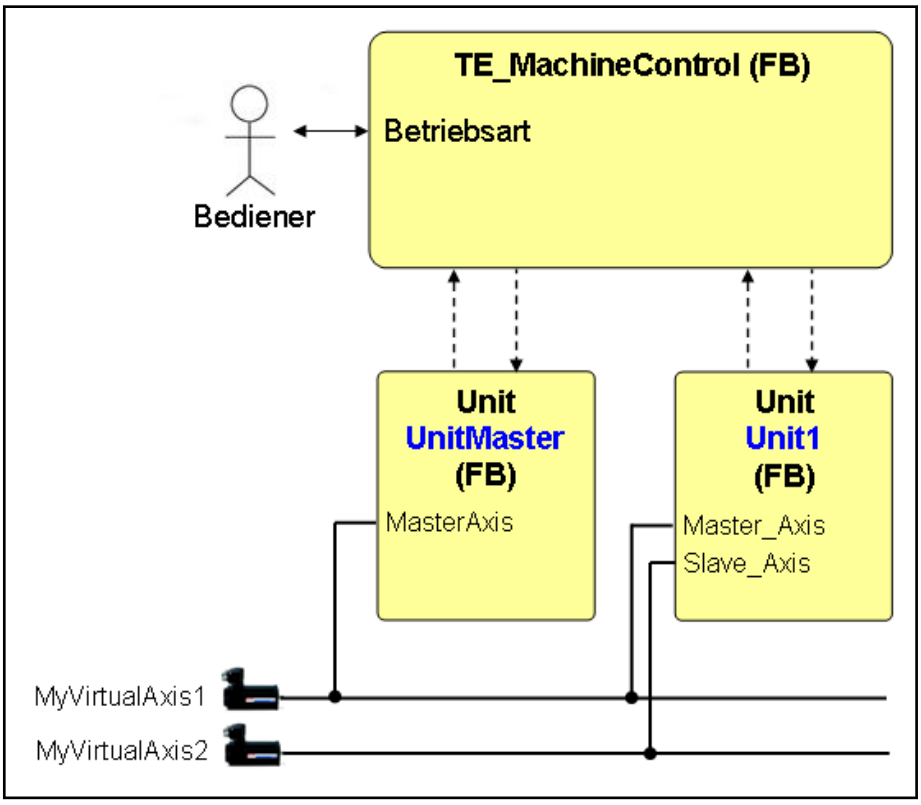


Abb.11-2: Schematisierte Aufgabenstellung

Arbeitsablauf

- IndraWorks öffnen und neues Projekt erstellen.
- MLC Steuerung aus der Bibliothek in den Projektbaum hinzufügen.
- GAT zur MLC Steuerung hinzufügen. Hierfür das "GAT"-Kontextmenü der Steuerung nutzen, oder in der IndraWorks Toolbar  anklicken.
- Über nachfolgenden Dialog die Option "GAT (dezentrale Zustandsmaschine)" auswählen und "Fertigstellen" anklicken.

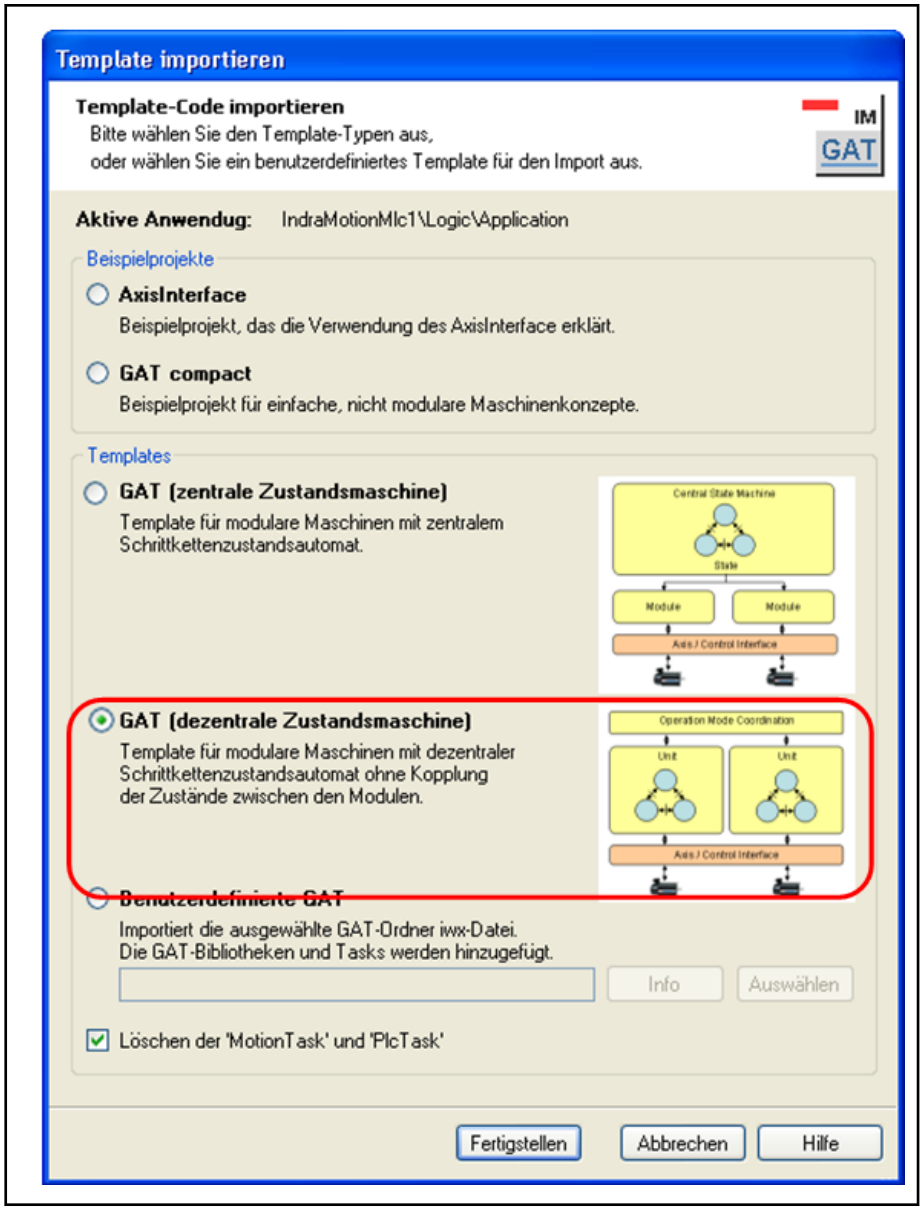


Abb.11-3: GAT mit dezentraler Zustandsmaschine importieren

Durch die Auswahl des "GAT mit dezentraler Zustandsmaschine" wird ein lauffähiges und vorkonfiguriertes Projekt importiert. Dieses Projekt enthält bereits 2 virtuelle Achsen.

Zur Realisierung der Aufgabe ist hinsichtlich der Definition von Betriebsarten kein weiterer Aufwand nötig, da die Betriebsart "Automatik" bereits in der Vorkonfiguration enthalten ist.

UnitMaster anlegen

Legen Sie im ersten Schritt die Unit "Unit_Master" an. Diese Unit enthält eine Achse ("MasterAxis") welche sich in der Betriebsart "Automatik" mit 100U/min bewegen soll.

Gehen Sie hierbei wie folgt vor:

- Über den GAT Wizard eine neue Unit hinzufügen. Hierzu im GAT Wizard den Knoten "Units" selektieren. Anschließend die Schaltfläche "Unit hinzufügen" anklicken.

GAT mit dezentraler Zustandsmaschine

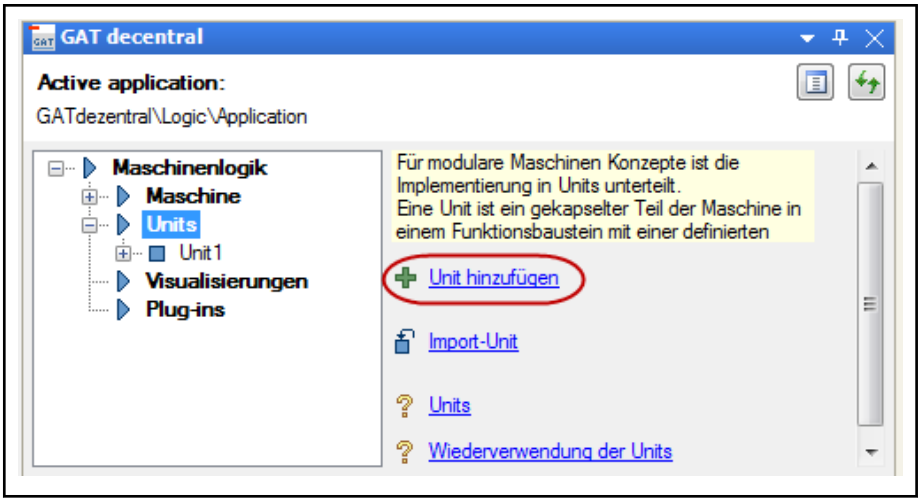


Abb.11-4: Unit hinzufügen

- In dem nachfolgenden Dialog den Namen "Unit_Master" und die weiteren Einstellungen wie folgt eintragen und Schaltfläche "Übernehmen" anklicken.

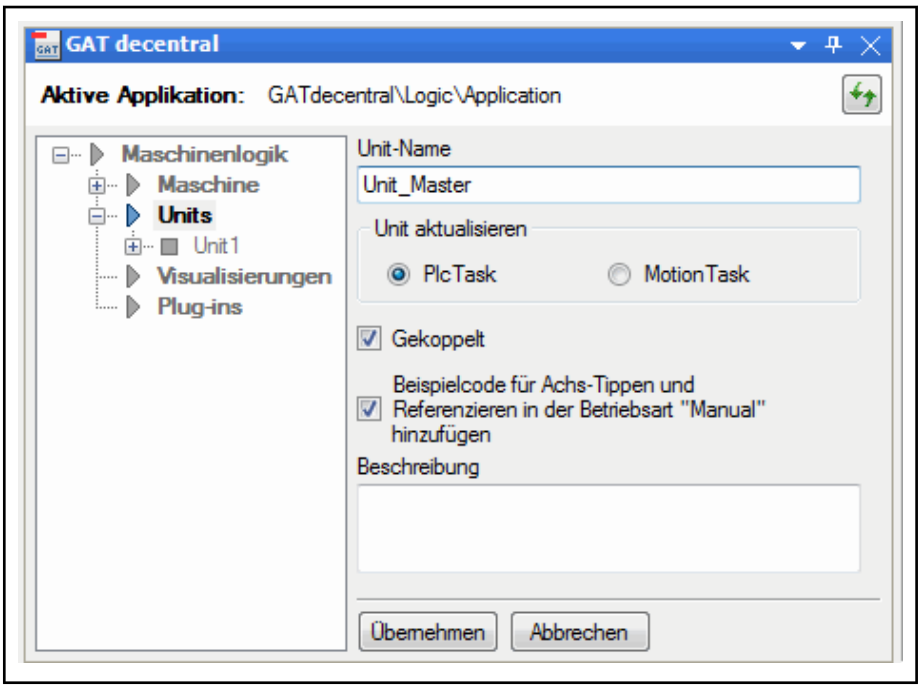


Abb.11-5: Unitname festlegen

- Die Achse "MasterAxis" für die Unit "Unit_Master" über den GAT Wizard wie folgt hinzufügen.

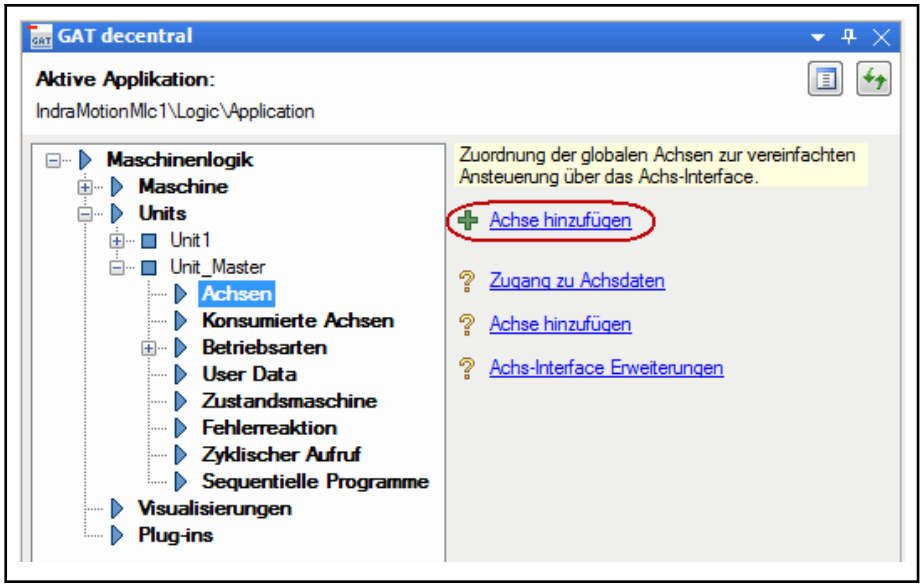


Abb.11-6: Hinzufügen einer Unitachse mit dem GAT-Wizard

- Im nachfolgenden Dialog den Achsnamen "MasterAxis" eintragen, und der global konfigurierten Achse "MyVirtualAxis1" zuweisen.

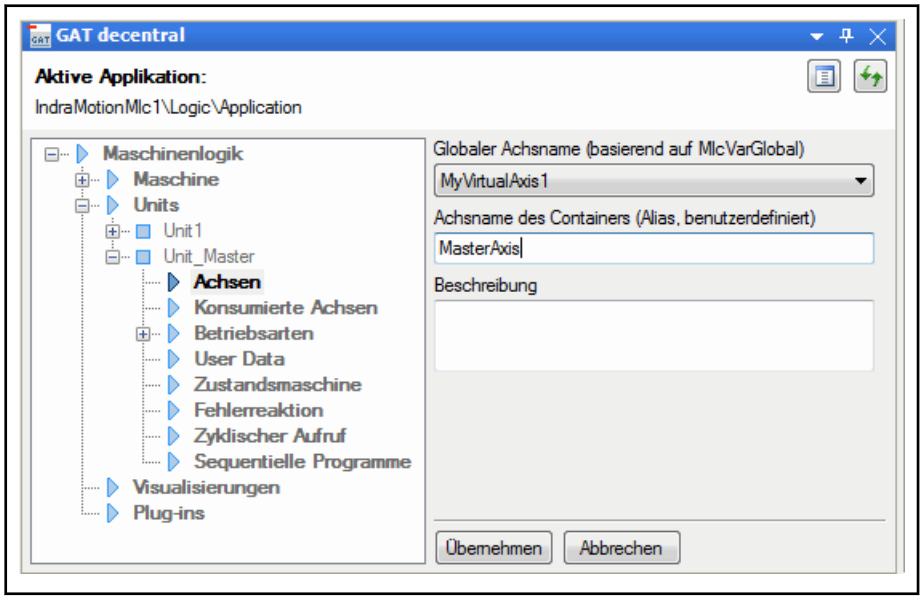


Abb.11-7: Zuweisung der Unitachse zur global konfigurierten Achse

- Den Programmcode für den Automatikbetrieb wie folgt hinzufügen. Hierzu den "Auto" Ordner anwählen, und über den Link "Implementierung öffnen" die Codestelle öffnen.

GAT mit dezentraler Zustandsmaschine

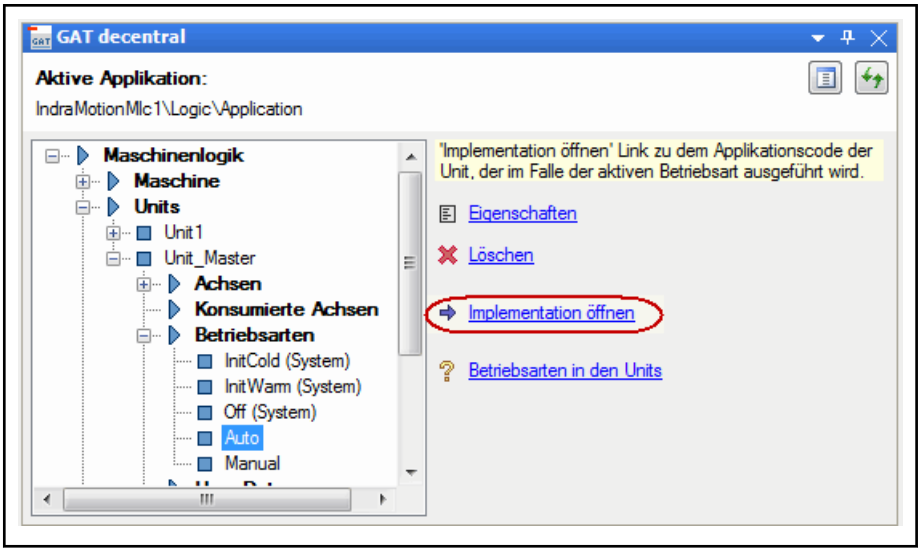


Abb.11-8: Codestelle für die "Automatik"-Betriebsart über den GAT-Wizard finden

- Folgenden Programmcode einfügen. Hierbei wird zunächst die Sollgeschwindigkeit 100U/min, und dann die Betriebsart "Geschwindigkeitsregelung" (ModeVel) vorgegeben.



Zur komfortablen Achskommandierung steht Ihnen zusätzlich die dialogbasierte Achskommandierung zur Verfügung. Mehr Details zu dem Thema finden Sie in Kap. 6.4.1 "Dialogbasierte Achskommandierung" auf Seite 73.

Sollgeschwindigkeit und Betriebsart vorgeben:

```
//-----------------------
// insert your code here
//-----------------------
Axes.MasterAxis.Ctrl.VelMode.Velocity := 100;
Axes.MasterAxis.Ctrl.Admin._OpMode.en := ModeVel;
```

Hinweis zum Achscontainer "Axes"

Die Referenzierung einer Achse innerhalb einer Unit erfolgt immer über "Axes". "Axes" ist in jeder Unit der Name des Achscontainers. Durch Eingabe eines "." (Punkt) nach Eingabe von "Axes" werden alle Achsen angezeigt, die im Achscontainer enthalten sind (siehe nachfolgende Abbildung). Auf gleiche Art und Weise erfolgt der Zugriff auf alle I/O-Daten der Unit über "UserData".

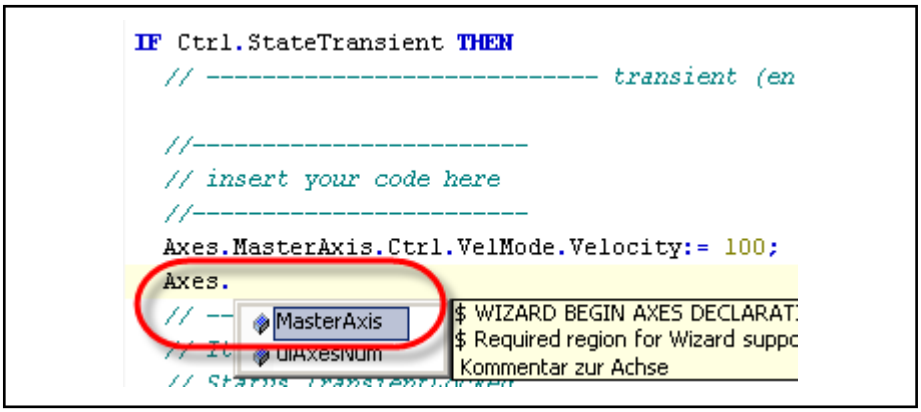


Abb.11-9: Achskommandierung innerhalb einer Unit und Intellisense

Im Weiteren werden z. B. durch Eingabe des Punkt-Operators nach dem Achsnamen alle Elemente angezeigt und zur Auswahl gestellt, die Daten zu

dieser Achse beinhalten (siehe nachfolgende Abbildung). Die nächste Ebene wird durch Eingabe eines Punktes basierend auf dem Namen der vorhergehenden Ebene erreicht.

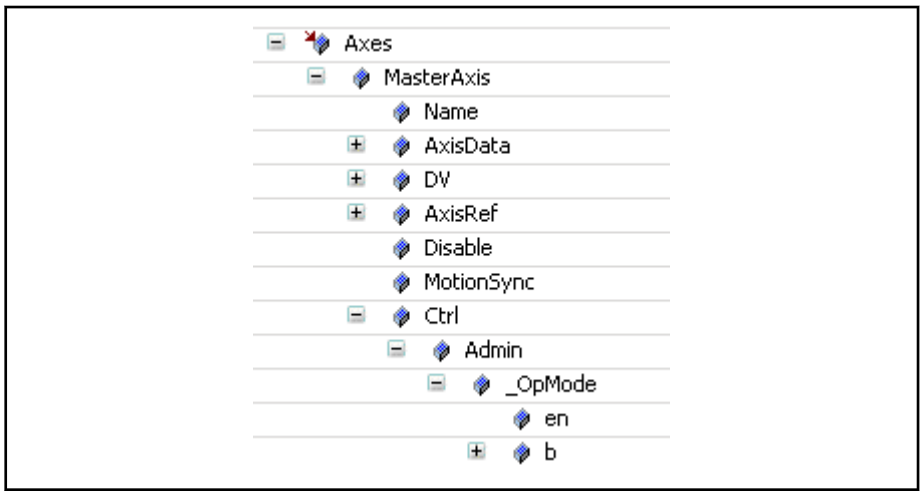


Abb.11-10: Verfügbare Strukturelemente einer Unitachse

Unit1 anpassen

Unit1 ist bereits durch den GAT Erstimport enthalten. Diese Unit wird mit den folgenden Schritten angepasst. Hierbei soll sich in der Betriebsart "Automatik" die "Slave" Achse winkelsynchron zu der "Master" Achse bewegen. Die Master Achse wird bereits in der Unit "Unit_Master" in der Automatik Betriebsart gestartet.

- "MasterAxis" als konsumierte Achse über den GAT Wizard wie folgt hinzufügen.

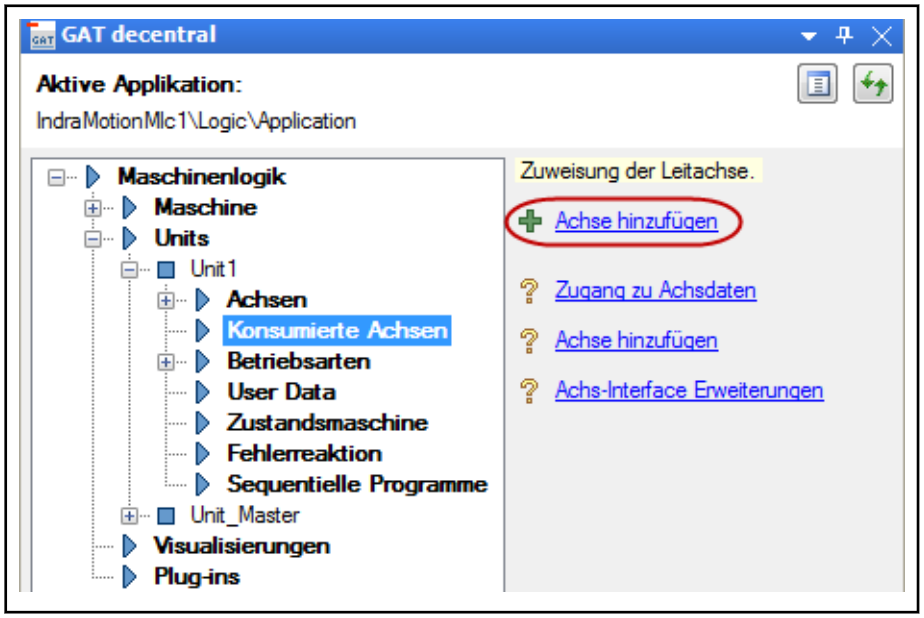


Abb.11-11: "MasterAxis" als konsumierte Achse hinzufügen

GAT mit dezentraler Zustandsmaschine

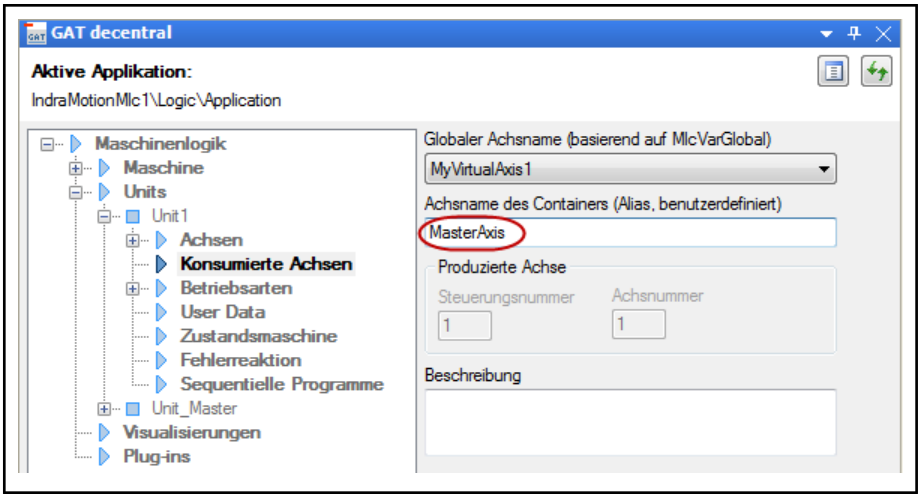


Abb.11-12: "MasterAxis" als konsumierte Achse hinzufügen

Die "MasterAxis" wird als "konsumierte Achse" in das Unit1 hinzugefügt, da diese Achse in dieser Unit nur als Leitachse verwendet wird.

- Achse "MyAxis1" über den GAT Wizard entfernen.

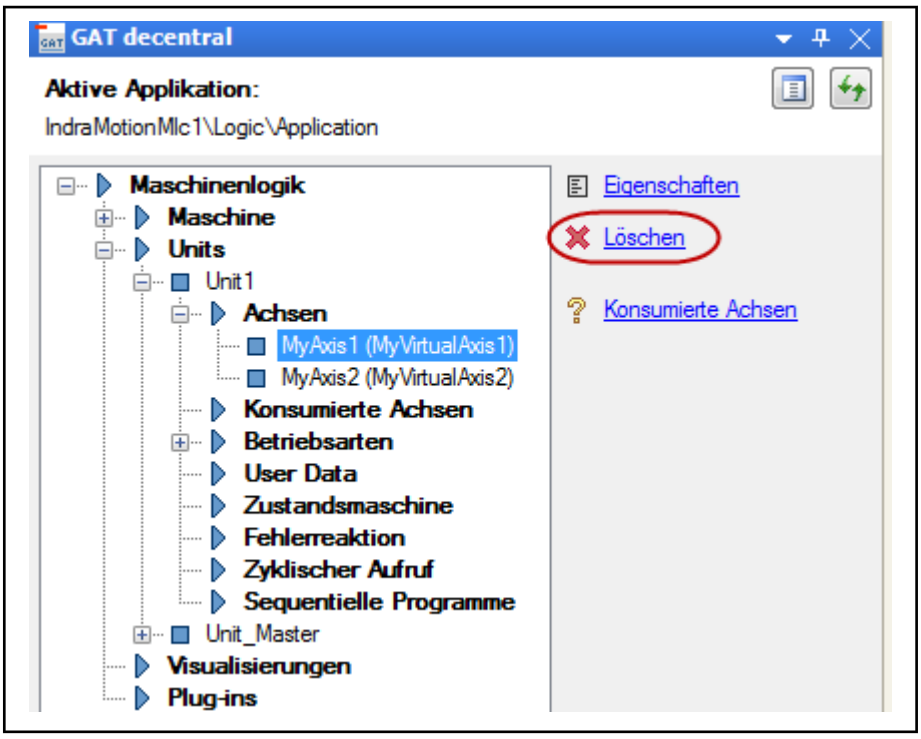


Abb.11-13: Löschen der nicht benötigten Achse "MyAxis1"

- Achse "MyAxis2" in "SlaveAxis" umbenennen.

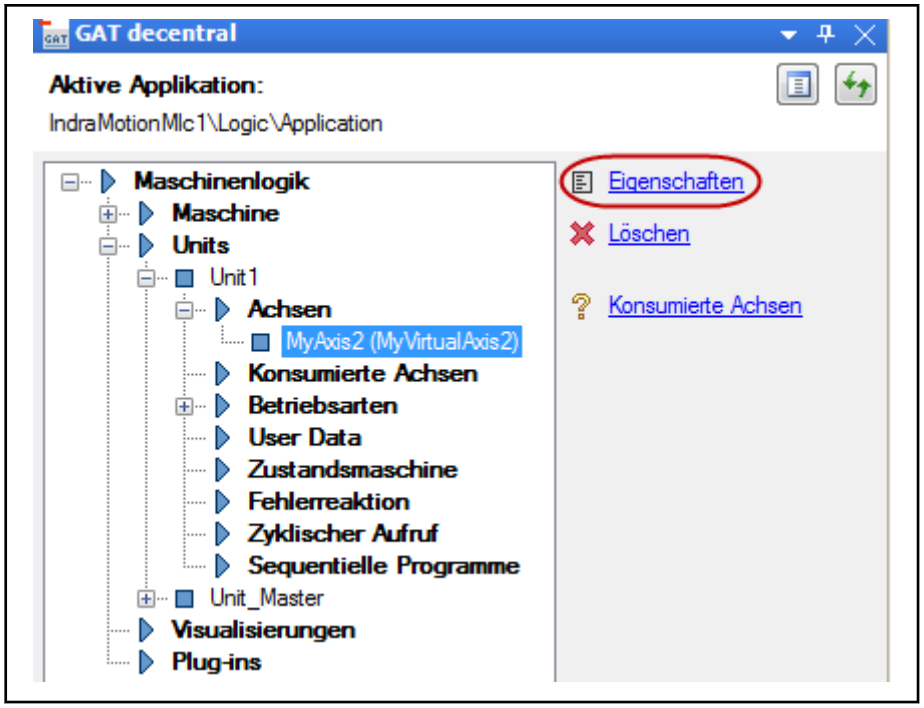


Abb.11-14: Achse "MyAxis2" in "SlaveAxis" umbenennen

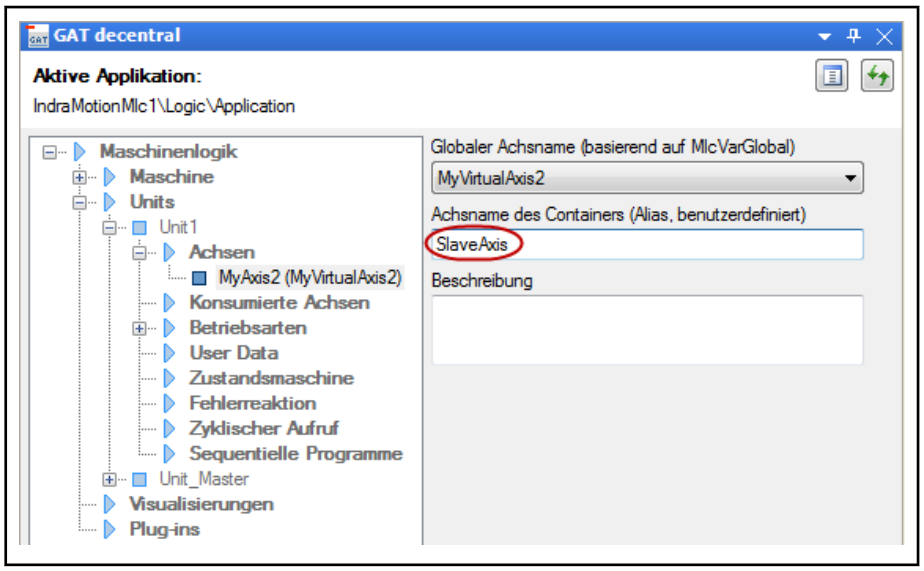


Abb.11-15: Achse "MyAxis2" in "SlaveAxis" umbenennen

- In der Automatik Betriebsart den bestehenden Beispielcode aus dem Erstimport löschen und durch den folgenden Programmcode ersetzen. Damit wird die Achse "SlaveAxis" winkelsynchron zu der Achse "MasterAxis" kommandiert.

GAT mit dezentraler Zustandsmaschine

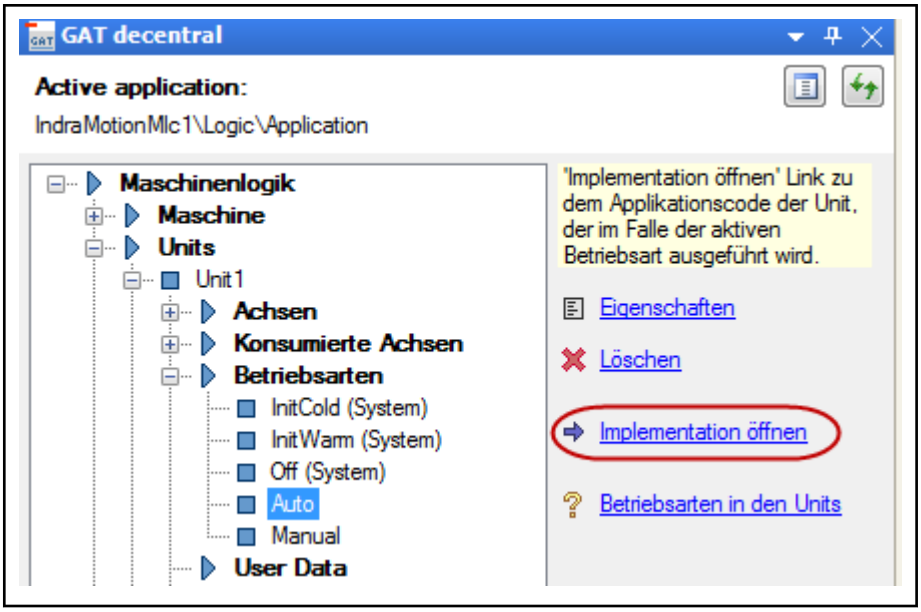


Abb.11-16: "Automatik"-Betriebsart der Unit1 öffnen

SlaveAxis synchron zu MasterAxis

```
//------------------------
// insert your code here
//------------------------
Axes.SlaveAxis.Ctrl.SyncMode.Master :=  ConsumedAxes.MasterAxis;
Axes.SlaveAxis.Ctrl.Admin._OpMode.en := ModeSyncPhase;
```

Inbetriebnahme Visualisierung Anpassen

Die neuen Achsnamen müssen in der Inbetriebnahmevisualisierung wie folgt angepasst werden:

- Inbetriebnahmevisualisierung über den GAT Wizard öffnen.

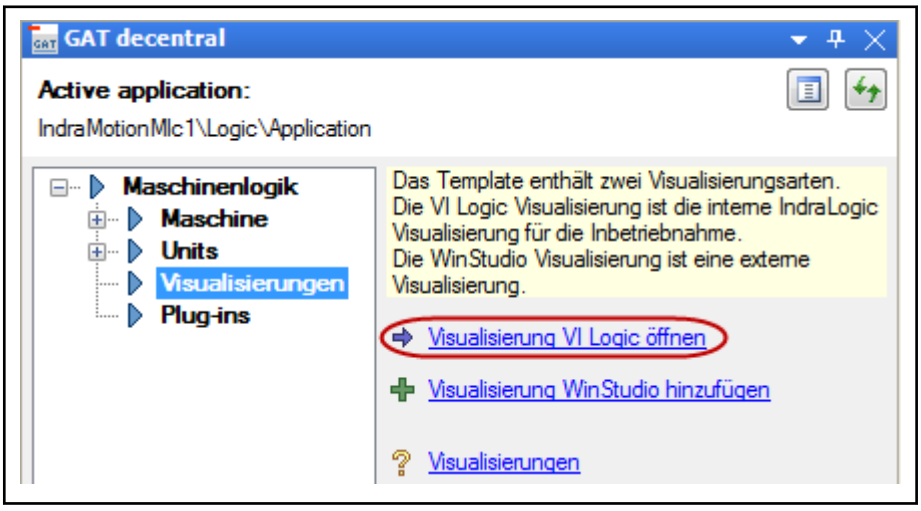


Abb.11-17: Öffnen der Inbetriebnahme Visualisierung über den GAT Wizard

- Das in der folgenden Abbildung markierte Visualisierungsobjekt anklicken und im Eingabefeld "Referenzierte Visualisierungen" den Unit- und Achsnamen wie folgt anpassen. Damit wirkt diese Achsvisualisierung auf die "MasterAxis" der Unit "Unit_Master".

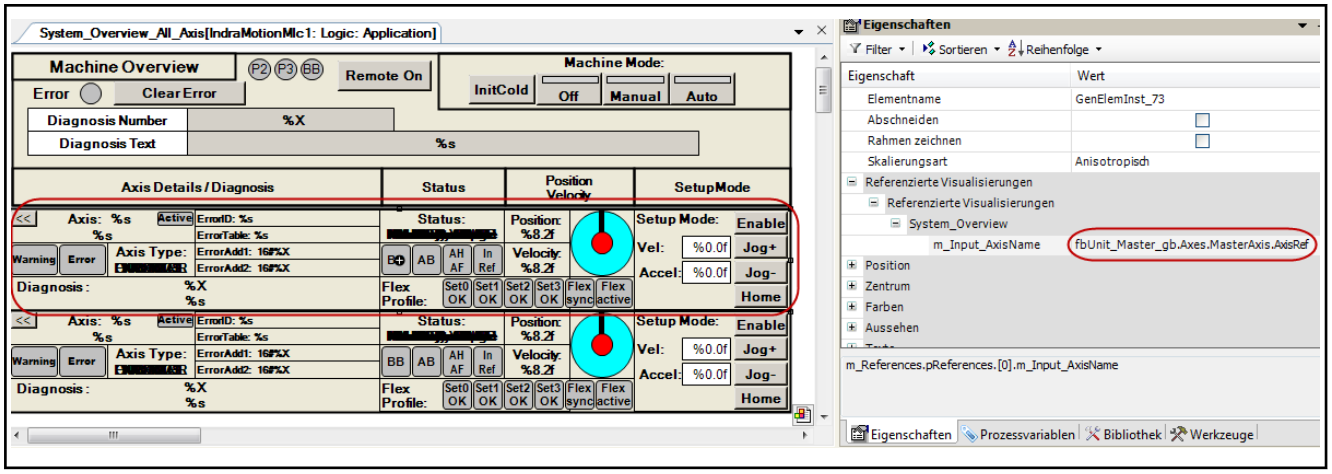


Abb.11-18: Anpassung der Achsvisualisierung für die MasterAxis

`fbUnit_Master_gb.Axes.MasterAxis.AxisRef`

- Das in der folgenden Abbildung markierte Visualisierungsobjekt anklicken und im Eingabefeld "Referenzierte Visualisierungen" den Unit- und Achsnamen wie folgt anpassen. Damit wirkt diese Achsvisualisierung auf die "SlaveAxis" der Unit "Unit1".

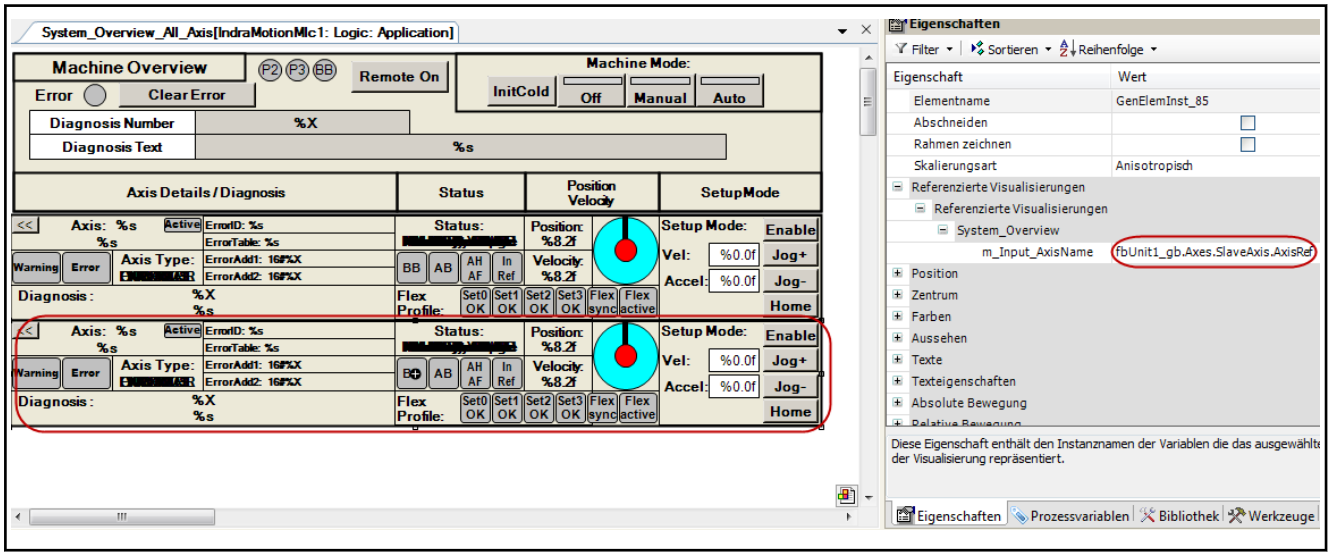


Abb.11-19: Anpassung der Achsvisualisierung für die SlaveAxis

`fbUnit1_gb.Axes.SlaveAxis.AxisRef`

Die Programmierung ist damit abgeschlossen.

Konfiguration und SPS Projekt Laden und Starten

Nach dem Herunterladen der Konfiguration (1) und des SPS-Projektes (2) kann die SPS gestartet (3) werden:



Abb.11-20: Herunterladen (Konfiguration/SPS-Projekt) und Starten der SPS

Nach Start des SPS-Projektes kann über die vordefinierte internen Visualisierung "System_Overview_All_Axis" das GAT bedient werden. Die **Remote On-**

Schaltfläche der Visualisierung muss dafür zunächst betätigt werden, um die Kontrolle über das GAT zu übernehmen.

Anschließend kann durch die **Auto**-Schaltfläche die Betriebsart "Automatik" aktiviert werden. Nun bewegen sich die Achsen entsprechend.

Bei Auswahl der "Manual"-Betriebsart, werden die Achsen angehalten, und der Setup-Mode der jeweiligen Achse kann über die **Enable**-Schaltfläche aktiviert werden. Ein Vertippen oder Referenzieren der Achse ist bei aktiviertem "Setup"-Mode über die jeweiligen Schaltflächen der Visualisierung möglich.

11.4 Funktionsprinzip

Um die Motion-Applikation einfach und effektiv zu erstellen, ist der Programmcode des GAT zweigeteilt.

Programmcode der sich bei unterschiedlichen Applikationen verallgemeinern lässt, befindet sich in den Basistypen und Basisbausteinen der GAT-Bibliothek (ML_TechTemplate.compiled-library).

Applikationsspezifischer Programmcode platziert der Anwender mit Unterstützung des Wizards an die dafür vorgesehene Stelle im Applikationsprogramm.

Dieses Funktionsprinzip führt zu folgenden Vorteilen:

- Erstellung von kompaktem und übersichtlichem Anwendercode:
  - Durch die Verwendung einfacher Schnittstellen und klarer Strukturen basierend auf einer optimierten Programmvorlage
- Erhöhung der Robustheit:
  - Durch den Einsatz getesteter und optimierter Codeteile
- Verbesserung der Wartbarkeit:
  - Durch den einfachen Austausch der Bibliothek bei Fehlerbehebungen

11.5 Initialisierung

Die Initialisierung wird zu Beginn des Maschinenhochlaufs durchlaufen, und besteht aus 2 Schritten:

- Im ersten Schritt wird der "Coldstart" (Parametriermodus/P2) aktiv, und an alle Units weitergegeben. Der applikationsspezifische Programmcode zur Initialisierung befindet sich in der Aktion "**actInit**" der jeweiligen Unit, und wird in diesem Zustand bearbeitet. Die Weiterschaltung zum nächsten Schritt erfolgt, wenn der Parametriermodus/P2 der Steuerung aktiv ist, und alle Units die Weiterschaltung freigeben (siehe nachfolgenden Hinweis).
- Im zweiten Schritt wird der "Warmstart" (Betriebsmodus/BB) aktiv, und an alle Units weitergegeben. Die Initialisierung ist abgeschlossen, wenn der Betriebsmodus/BB der Steuerung aktiv ist, und alle Units die Weiterschaltung freigeben (siehe Hinweis unten). Anschließend wird die vom Maschinenbediener angewählte Betriebsart aktiv. Durch Anwahl der Initialisierungs Betriebsarten kann jederzeit die Initialisierung erneut ausgeführt werden.

Durch Nutzung der Funktionalität "**LockOpModeAcknowledge**" innerhalb der Unit kann ein Weiterschalten zum nächsten Initialisierungs Schritt verriegelt bzw. freigegeben werden.

Durch setzen von "**LockOpModeAcknowledge := TRUE**" kann die Weiterschaltung zum nächsten Schritt für mehrere Zyklen blockiert werden, um beispielsweise Antriebsparameter zu lesen oder zu schreiben.

SkipColdstartAtStartup

Nach dem SPS-Projekt Download bzw. einem SPS-Reset wird per Standard der "Coldstart" der Init Betriebsart aktiviert. Insbesondere bei der Maschinen-Inbetriebnahme kann dieses Verhalten störend sein, da das Umschalten auf "Coldstart" = Parametermodus zeitaufwändig ist. Über die Einstellung "**SkipColdstartAtStartup = TRUE**" kann daher für diese Fälle das Umschalten auf "Coldstart" vermieden werden. Bei dieser Einstellung wird nach dem SPS-Download bzw. Reset der "Coldstart" übersprungen, und nur der "Warmstart" bearbeitet.

Veränderungen an der Einstellung "**SkipColdstartAtStartup**" werden in der Zuweisung der Methode "**mOneTimeInit**" am TE_MachineControl FB vorgenommen (siehe nachfolgende Abbildung). Nach Download des modifizierten SPS-Projektes werden die neuen Einstellungen wirksam.

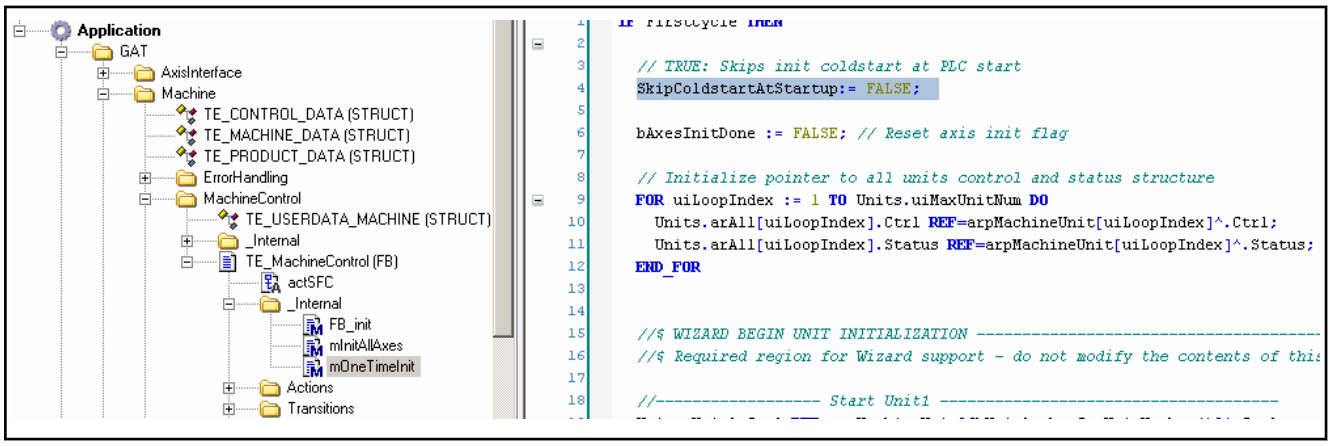


Abb.11-21: Konfiguration der Initialisierung über das Strukturelement "SkipColdstartAtStartup"

11.6 Betriebsarten

Allgemeines

Das GAT beinhaltet beim Erstimport bereits die vordefinierten Hauptbetriebsarten "MODEOFF", "AUTO" und "MANUAL". Die Initialisierung wurde bereits im vorhergehenden Kapitel behandelt.

Grundsätzlich werden die Betriebsarten wie folgt unterschieden:

- **Hauptbetriebsarten** werden an alle **gekoppelten** Units weitergegeben, und sind dort als Zweig in der Betriebsarten-Schrittkette enthalten
- **Teilmaschinenspezifische-Betriebsarten** sind jeweils spezifische Betriebsarten für **ungekoppelte** Units, und sind dort als Zweig in der jeweiligen Betriebsarten-Schrittkette enthalten

Zielbetriebsart nach Initialisierung/ Betriebsarten-Wechsel

Nach Abschluss der Initialisierung wird die vom Bediener vorgegebene Hauptbetriebsart an alle gekoppelten Units weitergeleitet und damit wirksam.

GAT mit dezentraler Zustandsmaschine

Für alle ungekoppelten Units wird über den Programmcode in der Methode "mOpmodeDistribution" des Funktionsbausteins "TE_MachineControl" für jede Unit individuell eine teilmaschinenspezifische Betriebsart vorgegeben.

Betriebsarten-Quittierung

Die angewählte Hauptbetriebsart wird am Status-Ausgang der Maschine (Variable "stMachine_Status_gb") erst dann quittiert, wenn alle Units (gekoppelt und ungekoppelt) die vorgegebene Betriebsart quittieren.

Über die Funktionalität "**LockOpModeAcknowledge**" hat jede Unit individuell die Möglichkeit, die Quittierung der Betriebsart zu verzögern.

11.7 Units

11.7.1 Allgemeines

Betriebsarten-Schrittkette

Jede Unit enthält eine Betriebsarten-Schrittkette (actOPModeSFC), die zyklisch aufgerufen wird. Je nach vorgegebener Betriebsart wird der entsprechende Schritt der Schrittkette aktiv, und der dazugehörige SPS-Code wird bearbeitet (z. B. für Automatik-Betrieb).

Lokale Fehlerbehandlung

Im Fehlerfall wir die Betriebsart unterbrochen und der SFC verzweigt in eine Fehlerbehandlung. Jedes Maschinenteil ist hierbei für die Fehlerreaktion der ihm zugeordneten Komponenten (wie z. B. zugehörige Achsen) verantwortlich. Bei Fehlerauftritt innerhalb eines Maschinenteils wird dieses an den übergeordneten Fehlerverwalter weitergegeben, um diese Information zu verteilen. Optional kann die Weiterleitung des lokalen Fehlers an den übergeordneten Fehlerverwalter in einem Maschinenteil unterdrückt werden (siehe "Lokale Fehlerbehandlung in einer Unit" auf Seite 370).

Achshandling

Sind Achsen in der Unit vorhanden, werden diese beim Hochlauf initialisiert und das Achs-Interface zur Laufzeit aufgerufen. Systembedingt wird das Achs-Interface für die jeweiligen Unitachsen automatisch in der gleichen Task wie die zugehörige Unit aufgerufen.

Über den Wizard können sehr einfach Units in das Applikationsprogramm hinzugefügt, gelöscht und modifiziert werden.

Grundsätzlich werden die Units wie folgt unterschieden:

Gekoppelte Units

Allen gekoppelten Units wird automatisch die vom Bediener vorgegebene Hautptbetriebsart weitergeleitet. In gekoppelten Units sind daher alle Hauptbetriebsarten als Zweig in der Betriebsarten-Schrittkette enthalten.

Die Betriebsarten-Verteilung an die Units erfolgt in der Methode **mOpmodeDistribution** des TE_MachineControl. Hier wird durch die Anweisung "mSetUnitsOpModes(OpMode := OpMode, CoupledUnitsOnly := TRUE);" die Hauptbetriebsart an alle gekoppelten Units weitergeleitet.

Bei gekoppelten Units besteht die Möglichkeit, zur Laufzeit die Betriebsarten-Kopplung aufzuheben und wieder herzustellen. Beispielsweise kann für Wartungszwecke durch diese Funktion eine Unit aus dem Betriebsarten-Verbund herausgelöst werden, wobei alle anderen Units weiter im Automatikbetrieb produzieren.

Durch die Anweisung "Units.<UnitName>.Ctrl.OpModeCoupled := FALSE;" innerhalb des TE_MachineControl wird die Betriebsarten-Kopplung aufgehoben. In diesem Fall wird die an der Maschine angewählte Hauptbetriebsart nicht mehr an diese Unit weitergegeben. Dadurch kann in der Methode **mOpmodeDistribution** durch das Anwenderprogramm eine beliebige Hauptbetriebsart der Unit vorgegeben werden.

Durch die Anweisung "Units.<UnitName>.Ctrl.OpModeCoupled := TRUE;" wird die Betriebsarten-Kopplung der Unit wieder hergestellt, und die Unit erhält wieder automatisch die vorgegebene Hauptbetriebsart.

Ungekoppelte Units

Ungekoppelten Units wird nach Abschluss der Initialisierung nicht automatisch die vom Bediener vorgegebene Hautptbetriebsart weitergeleitet. Die Betriebsart wird über die Methode "mOpmodeDistribution" im Funktionsbaustein "TE_MachineControl" durch das Applikationsprogramm vorgegeben.

Nach dem GAT-Erstimport steht bereits eine gekoppelte Unit **"Unit1"** zur Verfügung (siehe nachfolgende Abbildung). Der Unit-Name kann unter Verwendung des GAT-Wizard modifiziert werden.

GAT mit dezentraler Zustandsmaschine

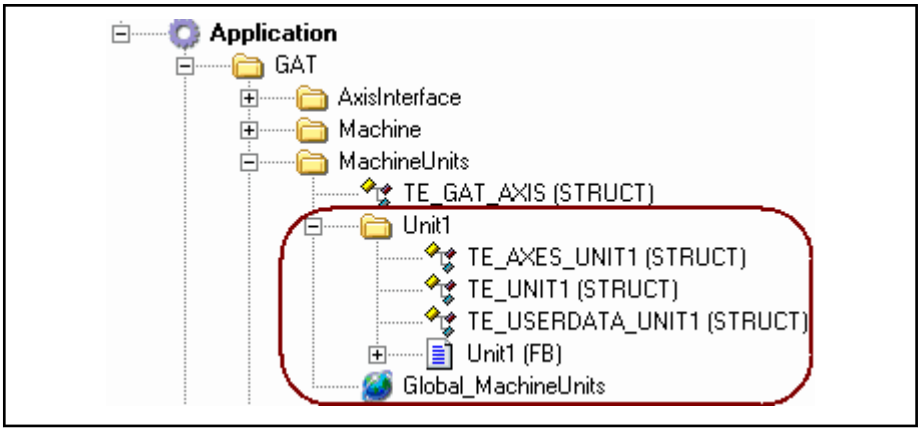


Abb.11-22: Unit-Ordner im Projektverzeichnis

11.7.2 Regeln zur Erstellung

Um eine einfache Wartung und die Wiederverwendbarkeit einer GAT-Unit zu erreichen, soll sie so gestaltet werden, dass:

- Abhängigkeiten zwischen den Units gering gehalten werden
- innerhalb einer Unit keine globalen Variablen verwendet werden

Um das zu erreichen, stellt jede Unit die nachfolgende Schnittstelle bereit.

11.7.3 Unit-Schnittstelle

Allgemeines

Nachfolgende Abbildung zeigt die einheitliche Unit-Schnittstelle.

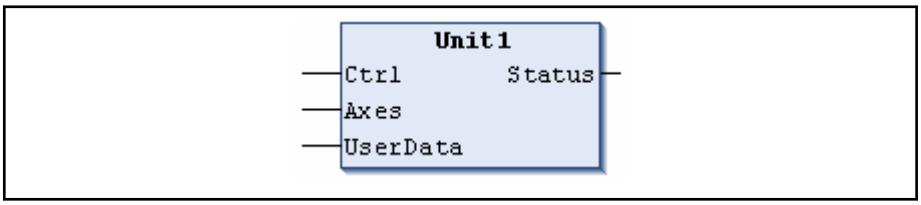


Abb.11-23: Unit-Schnittstelle

Um ein performantes Datenhandling zu ermöglichen, sind die Elemente "Axes" und "UserData" der Unitschnittstelle als Referenzen ausgeführt, d. h. die in der Unit benötigten Daten werden nicht in die Unit kopiert, sondern per Referenz an die Unit übergeben.

Im Folgenden werden die Bestandteile dieser Schnittstelle beschrieben.

Control- und Status-Struktur

Jede Unit verfügt über eine Variable zum Steuern (**Ctrl**) und eine Variable zur Statusmeldung (**Status**). Diese Elemente sind bereits am Basis-Baustein der GAT-Bibliothek enthalten, und werden auch von dort verwaltet.

Unitachsen

Zur Schnittstelle jeder Unit gehört ein Achs-Container (**Axes**). Alle Achsen dieses Containers werden in der Unit aktiv kommandiert.

☞ Passiv genutzte Achsen (wie z. B. Leitachsen) sollten aus Performancegründen **nicht** in den Achs-Container **Axes** konfiguriert werden.

Konfigurieren Sie **passiv** genutzte Achsen in dem **Consumed Axes** Achs-Container.

Der Name des Achs-Containers ist für jedes Modul gleich, der Inhalt (Typ) aber unterschiedlich, da nicht in jeder Unit dieselben Achsen verwendet werden.

Jede Achse im Achs-Container (Unitachse) ist vom Typ TE_GAT_AXIS, welcher auf dem Typ TE_BASE_MODULE_AXIS basiert (siehe Bibliothek ML_TechTemplate). Die nachfolgende Tabelle zeigt alle Elemente, die im Strukturtype TE_GAT_AXIS verfügbar sind.

Element	Typ	Beschreibung
Name	STRING(20)	Name der Achse als Zeichenkette (wird bei der Initialisierung des Systems initialisiert, kann durch den Anwender modifiziert werden)
AxisData	REFERENCE TO ML_AXISDATA_SM	Referenz auf das entsprechende Element im globalen Array **AxisData** (siehe Bibliothek **ML_Base**)
DV	REFERENCE TO ML_DirectVarAxis	Im Falle einer lokalen Achse: Referenz auf das entsprechende Element im globalen Array **DV_Axis** (siehe Bibliothek **ML_Base**) Im Falle einer Verbundachse (Achse einer anderen Steuerung): Dummyelement
GV	REFERENCE TO ML_GlobalVarAxis	Referenz auf das entsprechende Element im globalen Array **GV_Axis** (siehe Bibliothek **ML_Base**)
AxisRef	AXIS_REF	Achsreferenz
Disable	BOOL	TRUE: Der entsprechende AxisInterface-Datensatz wird nicht mehr aktualisiert FALSE: Standard **Eine Änderung des Wertes dieser Variable wird erst wirksam im INIT-Mode (STATE_INIT_COLDSTART oder STATE_INIT_WARMSTART)**
Ctrl	REFERENCE TO TE_AXISINTERFACE_CONTROL	Referenz auf die Steuerstruktur des Achs-Interfaces der entsprechenden Achse (Referenz auf den entsprechenden Eintrag des globalen Arrays **arAxisCtrl_gb**, siehe globale Variablenliste **Global_GAT_AxisInterfaces** im Ordner **AxisInterface**
Status	REFERENCE TO TE_AXISINTERFACE_STATUS	Referenz auf die Statusstruktur des Achs-Interfaces der entsprechenden Achse (Referenz auf den entsprechenden Eintrag des globalen Arrays **arAxisStatus_gb**, siehe globale Variablenliste **Global_GAT_AxisInterfaces** im Ordner **AxisInterface**

Abb.11-24: Alle mit dem Type TE_GAT_AXIS verfügbaren Elemente. Innerhalb der Unit erfolgt der Zugriff auf ein Element über Axes.<AxisName>.<Element>

Über den Achs-Container **Axes** sind alle relevanten Daten einer Achse adressierbar.

Der Container ist so aufgebaut, das über die Namen der Achsen auf die Achsdaten zugegriffen wird und damit gut lesbarer Programmcode entsteht.

Nachfolgende Tabelle zeigt, wie man auf Achsen im Achs-Container innerhalb und außerhalb einer Unit zugreift.

GAT mit dezentraler Zustandsmaschine

Zugriff auf eine Unitachse (z. B. Achse "vAxis" im Achs-Container der Unit "Unit1")	Achse deaktivieren
Zugriff außerhalb der Unit	stAxesUnit1_gb.vAxis.Disable := TRUE;
Zugriff innerhalb der Unit	Axes.vAxis.Disable := TRUE;

Der GAT-Wizard stellt die Zuordnung der Unit-Achse, und der in IndraWorks konfigurierten Achse (reale, virtuelle,…) her. Hierzu wird in der Methode "mOneTimeInit" des FB TE_MachineControl der Unit-Achse die entsprechende physische Achse zugewiesen (siehe nachfolgende Abbildung und Kap. 11.13.2 "Achse hinzufügen" auf Seite 394).

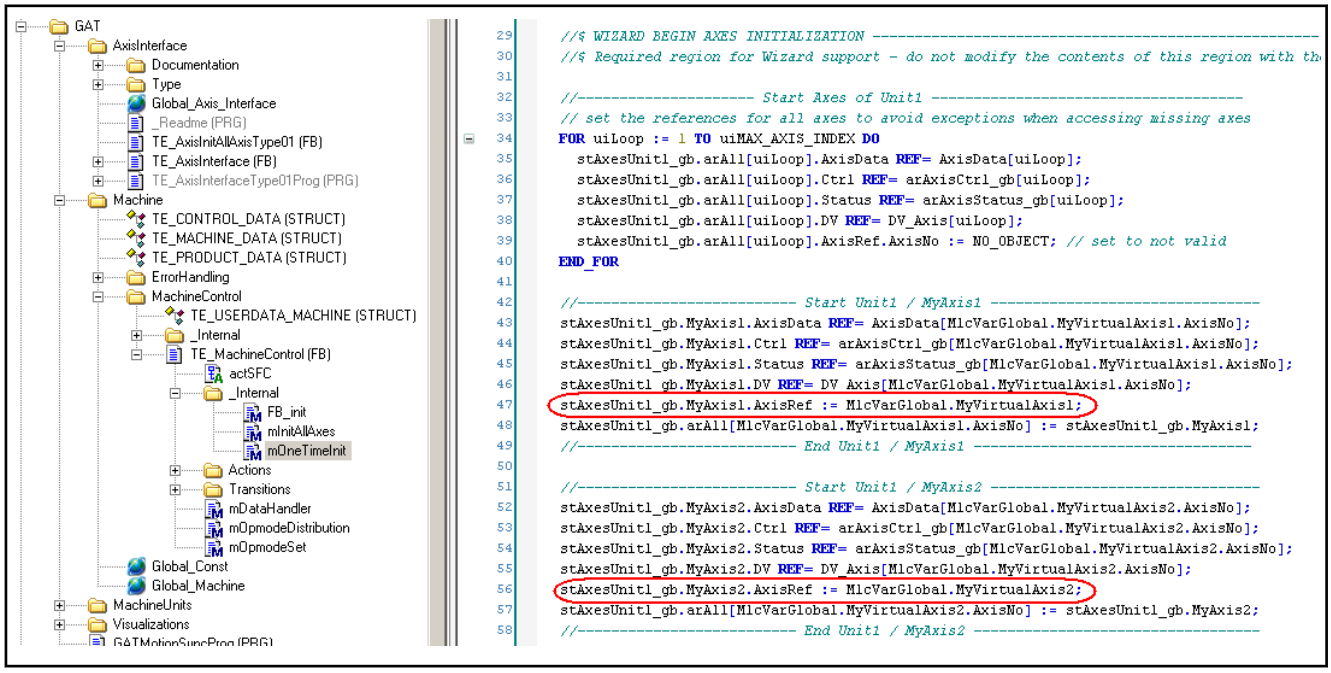


Abb.11-25: Achszuweisung in der Methode "mOneTimeInit" im TE_MachineControl

Ein Beispiel aus der oberen Abbildung: Die globale, in IndraWorks erzeugte Achse "MyVirtualAxis1" wird der Achse "MyAxis1" der Unit "Unit1" zugewiesen.

Passive Achsen (Consumed Axes)

Konfigurieren Sie alle passiv genutzten Achsen (wie z. B. Leitachsen) einer Unit in dem **Consumed Axes** Achs-Container.

Führen Sie hierzu mit dem GAT Wizard folgende Schritte aus:

- Öffnen Sie den GAT Wizard (GAT-Button 

in der IndraWorks Toolbar)
- Selektieren Sie den **Consumed Axes** Knoten unterhalb der Unit
- Funktion "Achse hinzufügen" wählen und Dialogabfragen folgen

Als passiv genutzte Achsen können auch Encoder-, Link- und C2C Consumed-Achsen verwendet werden.

Durch die Nutzung des **Consumed Axes** Achs-Container werden folgende Vorteile erreicht:

- **Performancevorteil**, da das AxisInterface für die Consumed-Achsen nicht aufgerufen wird
- **Erhöhte Benutzerfreundlichkeit**, da schneller ersichtlich wird, in welcher Unit welche Achsen aktiv kommandiert werden

Beispiel für die Nutzung einer Consumed Achse im Programmcode einer Unit:

Programm:

```
// Example: MySlaveAxis1 will be synchrozied with VirtualMaster 1 consumed axi
Axes.MySlaveAxis1.Ctrl.SyncMode.Master := ConsumedAxes.VirtualMaster1;
Axes.MySlaveAxis1.Ctrl.Admin._OpMode.en := ModeSyncPhase;
```

Bei einer C2C Consumed-Achse können die Achsdaten (wie z. B. Position, Geschwindigkeit) über die Variable **GV_LinkAxis[]** in einer Unit gelesen werden.

Bei allen anderen Achstypen können die Achsdaten der Consumed-Achse über die Variable **DV_Axis[]** gelesen werden.

Eine Consumed-Achse entspricht dem Datentyp AXIS_REF mit den Elementen "CntrlNo" und "AxisNo".

Unitspezifische I/O-Daten

Um Wiederverwendbarkeit und Mehrfach-Instanziierbarkeit der Unit sicherzustellen, darf in einer Unit nicht auf globale Daten zugegriffen werden. Alle I/O, die in einer Unit benötigt werden, werden über den E/A-Container **UserData** zur Verfügung gestellt. Der Name des E/A-Containers ist für jede Unit gleich, der Inhalt (Typ) aber unterschiedlich, da nicht in jeder Unit dieselben I/O-DATEN verwendet werden.

Die Variable UserData ist eine Datenstruktur deren Elemente der Anwender definiert.

Nachfolgende Abbildung zeigt, wo die I/O-DATEN für die im GAT enthaltene Unit1 definiert sind:

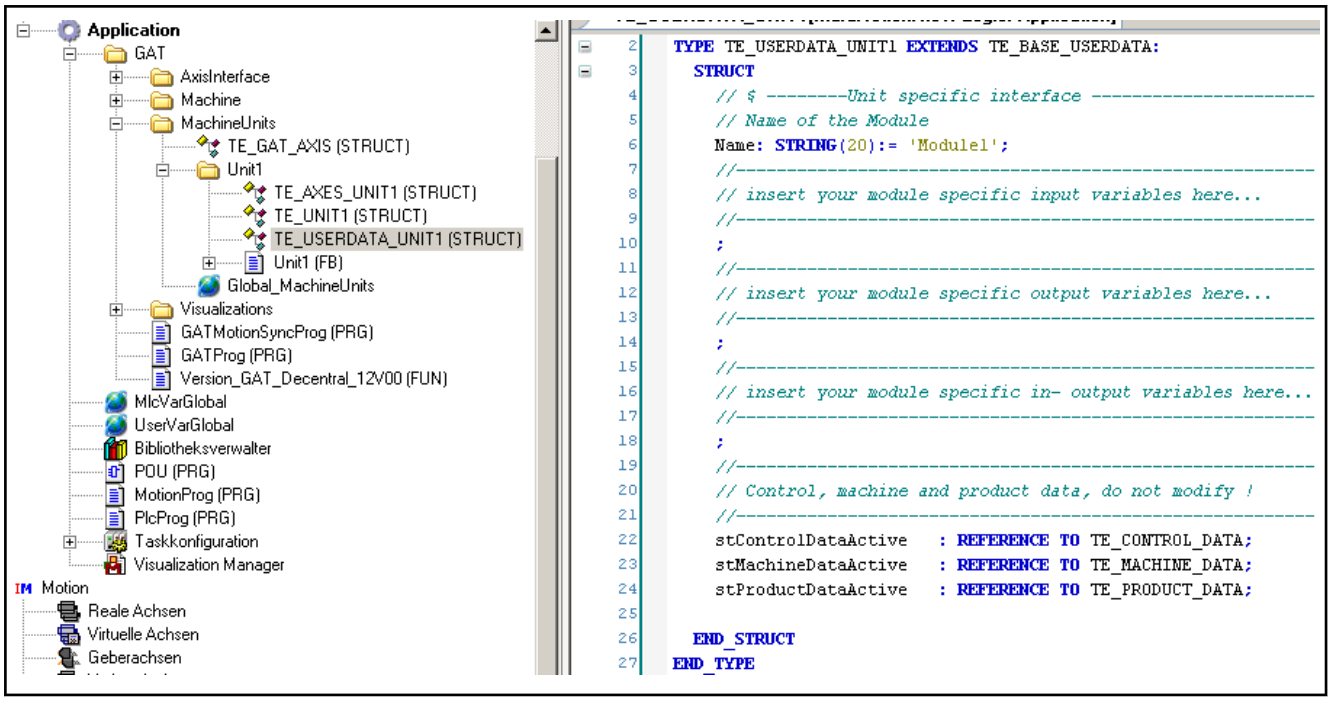


Abb.11-26: Unitspezifische I/O-DATEN

Unitinstanzen

Von einer bestehenden Unit können weitere Instanzen angelegt werden, die sich nur durch die zugeordneten IndraWorks-Achsen und der zugeordneten I/O unterscheiden. Die Unit-Instanziierung wird vom GAT-Wizard unterstützt.

GAT mit dezentraler Zustandsmaschine

11.7.4 Betriebsarten in der Unit

Bei Hinzufügen einer neuen Hauptbetriebsart sorgt der Wizard dafür, dass in allen gekoppelten Units ein korrespondierender Zweig in der Betriebsarten-Schrittkette angelegt wird. Dieser Schritt wird nur dann bearbeitet, wenn die zugehörige Hauptbetriebsart angewählt wird.

Über die Funktionalität "LockOpModeAcknowledge" hat jede Unit individuell die Möglichkeit die Betriebsartenquittierung zu verzögern. Damit kann beispielsweise auch ein Weiterschalten der Initialisierungsschritte verzögert werden, wenn die Aktivitäten dafür mehrere Zyklen erfordern (z. B. Schreiben von Parametern).

In der Methode "mOpModeAcknowledge" der Unit wird die Betriebsartenquittierung durchgeführt. Diese Methode wird vom Wizard erstellt und bei Änderungen der Hauptbetriebsarten modifiziert.

11.7.5 Sequentielle Programmierung

Einführung in die Sequentielle Programmierung

Ziel der sequentiellen Programmierung ist es, die Ablaufprogrammierung, insbesondere die Automatikabläufe, innerhalb der IndraMotion MLC zu vereinfachen. Die sequentiellen Befehle bedienen sich den Sprachmitteln der IEC - 61131 (Structured Text). Dadurch ist keine zusätzliche Programmierumgebung notwendig, da der in IndraWorks integrierte Editor auch für die sequentielle Programmierung genutzt werden kann. Die sequentielle Programmierung ist vollständig in das GAT (mit dezentraler Zustandsmaschine) integriert. Für die Konfiguration der sequentiellen Programme wurde der GAT-Wizard erweitert. Der Anwender kann mit Hilfe des Wizards sequentielle Programme anlegen, bzw. auch wieder entfernen lassen. Durch das Anlegen eines sequentiellen Programms werden automatisch die entsprechenden Achsen für die Verwendung der sequentiellen Befehle vorbereitet.

Interaktion der Sequentiellen Programmierung mit dem GAT

Sequentielle Programme sind genau einer Betriebsart einer Unit zugeordnet. Vor der Benutzung muss eine Programm- und Achskonfiguration vorgenommen werden. Die Konfiguration der Sequentiellen Programmierung wird durch den GAT-Wizard wie folgt durchgeführt.

- Unterhalb des Unit <UnitName>-Ordners werden die zu einer Betriebsart zugehörigen sequentiellen Programme in einem Ordner (Mode <Betriebsart>) abgelegt. In den Programmen wird der sequentielle Ablauf programmiert.
- Jedes sequentielle Programm ist einer eigenen Task zugeordnet. Die Tasks sind in der Taskkonfiguration zu finden.
- Um die sequentiellen Programmen starten und stoppen zu können, wird jeweils eine Instanz des Bausteins "SeqTaskControl" benötigt. Die Instanzen dieses Bausteins werden in der globalen Variablenliste Global <UnitName> der entsprechenden Unit eingefügt. Standardmäßig wird im Eingangs- bzw. Ausgangsschritt der verwendeten Betriebsart der Start- und Stop-Befehl des sequentiellen Programms eingefügt.

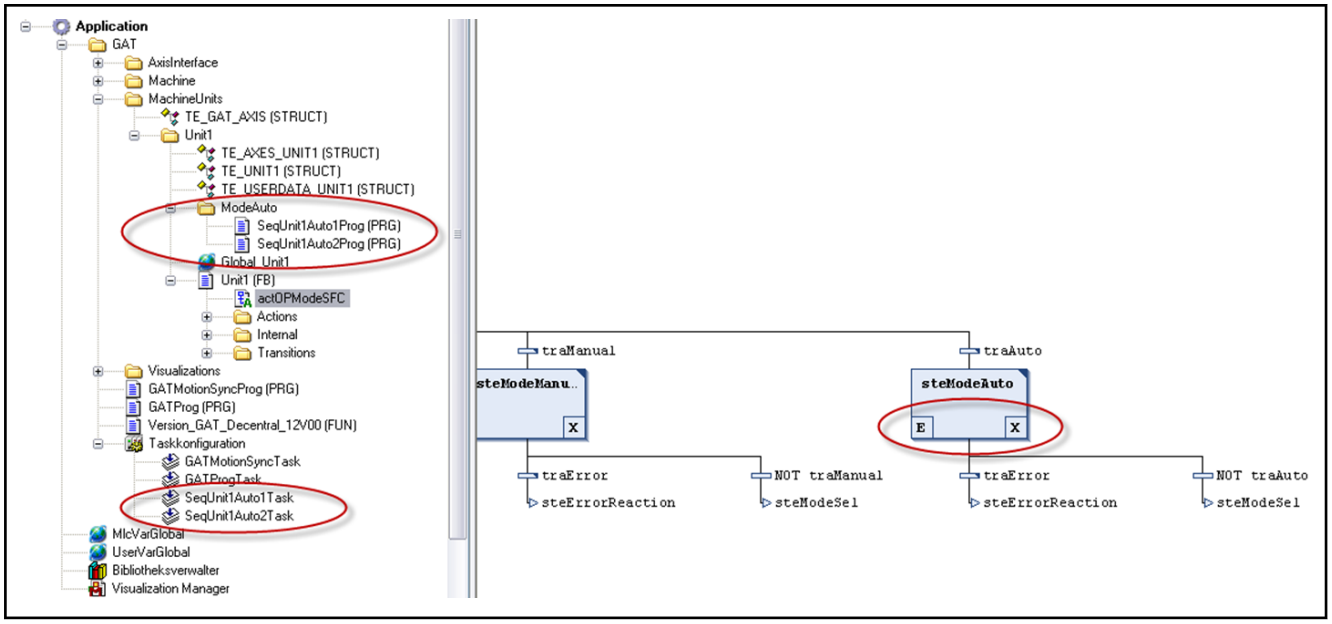


Abb.11-27: Ein- und Ausgangsaktion durch sequentielle Programmierung

- Der GAT-Wizard stellt die Zuordnung der Unit-Achsen zur sequentiellen Programmierung her. Hierzu werden in der Methode "mOneTimeInit" des FB TE_MachineControl die Achsreferenzen der Unitachsen dem Feld Seq[…] zugeordnet
- Die sequentiellen Achshandler werden motionsynchron im "GATMotionSyncProg" augerufen
- Im Ausgangsschritt der Betriebsart wird durch den GAT-Wizard für jede Achse der Befehl Seq[…].Disable() eingetragen. Der Befehl sorgt dafür, dass der Zugriff der sequentiellen Programmierung auf die Achsen beim Verlassen der Betriebsart beendet wird

Weiterführende Hinweise zur sequentiellen Programmierung

Weiterführende Hinweise zur sequentiellen Programmierung wie z. B. die Funktionsbausteine, Befehle und Schnittstellen sind in der Funktionsbeschreibung zur der sequentiellen Programmierung enthalten (siehe Dokumentation "Rexroth IndraLogic XLC IndraMotion MLC xxVRS Sequentielle Programmierung").

xx- Version

11.7.6 Unitkoordination

Da innerhalb einer Unit keine Informationen über andere Units vorhanden sind, muss die Unitkoordination an einer übergeordneter Stelle erfolgen.

Für die übergeordnete Unitkoordination sind die folgenden Funktionsbausteine vorgesehen:

- **TE_ErrorHandling** koordiniert die Fehler-Weiterverteilung zwischen den Units. In der Grundausführung werden alle Fehler an alle Units weitergegeben
- **TE_MachineControl** behandelt alle nicht fehlerbezogenen unitübergreifenden Aktivitäten, wie beispielsweise die übergreifende Betriebsarten-Verwaltung

Der Funktionsbaustein **TE_MachineControl** beinhaltet bereits die Methode **"mCyclic"**, die in jedem Zyklus aufgerufen wird. In dieser zyklisch aufgerufenen Methode sollen E/A-Informationen zwischen den Units und von den Units zu den E/A-Punkten der Maschine ausgetauscht werden.

GAT mit dezentraler Zustandsmaschine

Da sich die Unitkonfiguration bzw. die Zuweisung einer globalen Achse an eine entsprechende Unitachse zur Laufzeit nicht ändert, erfolgen diese Aktivitäten in der Methode **"mOneTimeInit"** des TE_MachineControl. Diese Methode wird zur Initialisierung des GAT durch die Maschinen-Organisationseinheit aufgerufen.

Der in der Methode "mOneTimeInit" enthaltene Code wird durch den Wizard verwaltet. Beim Hinzufügen oder Entfernen von Units oder Unitachsen werden entsprechende Einträge hinzugefügt oder entfernt.

11.8 Taskverwaltung

11.8.1 Taskkonfiguration GAT

Im GAT sind folgende Tasks standardmäßig enthalten:

	Prio	Zykluszeit	Watchdogzeit	Empfindlichkeit des Watchdog	Programmaufruf
GATMotionSync-Task	2	=Motion Zykluszeit	T#2ms	1	GATMotionSyncProg
GATProgTask	5	T#10ms	T#20ms	2	GATProg

Abb.11-28:

11.8.2 Hinzufügen einer BackgroundTask

Falls die Laufzeit der PlcTask zu groß wird, kann durch Auslagern von zeitunkritischen Aufgaben in eine Backgroundtask die PlcTask entlastet werden. Zu bedenken ist dabei, dass durch die zusätzlichen Taskwechsel PlcTask <> BackgroundTask das Gesamtsystem höher belastet wird (siehe auch Kap. 5.5 "Taskhandling" auf Seite 41).

	Prio	Zykluszeit	Watchdogzeit	Empfindlichkeit des Watchdog
BackgroundTask	~6	T#40ms	-	-

Um eine BackgroundTask hinzuzufügen sind die folgenden Schritte notwendig:

- Taskkonfiguration, hinzufügen der Task "BackgroundTask", Einstellungen siehe Tabelle oben
- Hinzufügen des "BackgroundProg" (Programm) und Aufruf von "BackgroundProg" in der Konfiguration der "BackgroundTask"
- Am FB TE_MachineControl die Methode "mBackground" hinzufügen. Anschließend diese Methode im "BackgroundProg" aufrufen. Programmzeile mit Aufruf: fbTE_MachineControl_gb.mBackground(). In der Methode "mBackground" sind alle Variablen des FB "TE_MachineControl" verfügbar.

11.9 Fehlerbehandlung

11.9.1 Fehlererkennung und Diagnose

Die Programmiervorlage GAT enthält vordefinierte Funktionalitäten zum Erkennen, Behandeln und Löschen von Fehlern. Die grundsätzliche Vorgehensweise ist im Folgenden beschrieben.

Lokale Fehlerbehandlung in einer Unit

Die Fehlerbehandlung ist lokal in den Units enthalten. Hierbei ist jede Unit für die Fehlerreaktion der ihr zugeordneten Komponenten zuständig. Hat bei-

spielsweise ein Antrieb der Unit einen Fehler, so hat auch die Unit einen lokalen Fehler. Dieser lokale Fehler wird standardmäßig durch Setzen des "Error"-Ausgangs an den Fehlerverwalter weitergegeben, damit auch die anderen Units auf den Fehler reagieren können.

Fehlerweiterleitung an Fehlerverwalter

Im Fehlerfall wird automatisch das "Error"-Bit im Unit-Statusausgang gesetzt. Ist das in bestimmten Fällen nicht gewünscht, so kann durch das Hinzufügen der Methode **mForwardLocalError** in der Unit die Fehlerweiterleitung unterdrückt werden. Die Methode **mForwardLocalError** der Unit wird durch das GAT im Fehlerfall automatisch aufgerufen, und erhält zusätzlich die Diagnosenummer des erkannten lokalen Fehlers über die "DiagNo". In dem Applikationscode können Sie die "DiagNo" auswerten, und basierend auf dieser Information den Fehler weiterleiten (mForwardLocalError := TRUE) oder nicht weiterleiten (mForwardLocalError := FALSE).

*Bei dem Hinzufügen der Methode **mForwardLocalError** muss folgende Schnittstelle deklariert sein:*

```
METHOD mForwardLocalError : BOOL
VAR_INPUT
      DiagNo:   DWORD;   // error number
END_VAR
```

Auch wenn der Fehler nicht weitergeleitet wird, so erfolgt in der Unit eine lokale Fehlerreaktion unmittelbar noch im gleichen Zyklus, wenn die allgemeine GAT Fehlerreaktion eingeschaltet ist.

Lokale Fehlerquellen

Als lokale Fehlerquellen innerhalb einer Unit werden berücksichtigt:

- Achsfehler (z. B. Schleppfehler einer Achse)
- Applikationsspezifische Fehler (z. B. geöffnete Schutztür)
- Funktionsbaustein-Fehler (z. B. durch Fehler eines aufgerufenen Funktionsbausteins)

Unitübergreifende Fehlerbehandlung

Der Funktionsbaustein **TE_ErrorHandling** überwacht **unitübergreifend** alle Fehlerquellen an einer zentralen Stelle. Wird ein Fehler erkannt, wird dieser an alle Units als **externer** Fehler weitergeleitet (Uniteingang "Ctrl.ExtError" und "Ctrl.ExtErrDiagNo"). Optional kann die Fehlerverteilung auf die jeweiligen Units im Applikationsprogramm auch unterdrückt werden. Hierzu ist die Methode **mDistributeError** des "TE_Errorhandling" entsprechend anzupassen. In den jeweiligen Units erfolgt darauf eine entsprechende Fehlerreaktion im nächsten Zyklus.

Weiterhin koordiniert der Funktionsbaustein **TE_ErrorHandling** auch das Quittieren des Fehlers und leitet diese Information von zentraler Stelle an alle Units weiter. Die Units sind dann für das Ausführen der Fehlerquittierung an den zugeordneten Komponenten verantwortlich.

Falls mehrere Fehler gleichzeitig anstehen, wird der schwerwiegendste angezeigt. Stehen gleichzeitig mehrere Fehler mit dem gleichen Schweregrad an, so wird der zuletzt erkannte Fehler ausgegeben.

Am Ausgang des **TE_ErrorHandling** wird über die Variable **stErr_Handling_Status_gb** die Diagnose der Maschine ausgegeben:

GAT mit dezentraler Zustandsmaschine

Element	Typ	Beschreibung
Error	BOOL	TRUE signalisiert Fehler
DiagNo	DWORD	Nummer gemäß Dokumentation "Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnosen" mit den folgenden Erweiterungen: • 16#F0**AE** xxxx: Applikations-Fehler (z. B. geöffnete Schutztür) (xxxx = frei durch den Anwender definierbar) • 16#F0**FB** 0000: FB-Fehler (z. B. Fehler an einem TechFB) -> Details siehe ErrorID/ErrorIdent • 16#F0**AA** 0000: GAT-interner Fehler -> Details siehe ErrorID/ErrorIdent • 16#F0**B1...B9** xxxx: Plugin Zusatzdiagnose Fehler (z. B. durch PROFIBUS Diagnose Plugin)
ErrorID	ERROR_CODE	Detaildiagnose im Fall von FB- + GAT-Fehlern (DiagNo = 16#F0FB/AA 0000)
ErrorIdent	ERROR_STRUCT	Detaildiagnose im Fall von FB- + GAT-Fehlern (DiagNo = 16#F0FB/AA 0000)
DiagMessage	STRING(60)	Enthält bei Antriebs- und Steuerungsfehlern den Diagnosetext

Das Element **DiagNo** ist zur Nutzung an einem HMI vorgesehen. Die Fehlertexte werden dort der numerischen Information in beliebigen Sprachen zugeordnet.

DiagMessage, ErrorID und **ErrorIdent** sind zusätzliche Informationen, die hauptsächlich bei der Inbetriebnahme durch den Programmierer z. B. über eine IndraLogic-Visualisierung angezeigt werden können.

Hinweis zu applikationsspezifischen Fehlern

mSetAppError

Durch Aufrufen der Methode "mSetAppError(Number)" können Applikationsfehler (wie z. B. geöffnete Schutztür) abgesetzt werden. Die übergebene "Number" am Eingang der Methode entspricht hierbei der Applikationsfehlernummer.

Das zentrale Diagnosesystem in der Unit überwacht hierbei zyklisch die abgesetzten Applikationsfehler und erzeugt den Applikationsfehler DiagNo = Fehler 16#F0AE xxxx (xxxx = Applikationsfehlernummer) wenn ein Applikationsfehler abgesetzt wurde.

Für Applikationsfehler in den Units sind Nummern im Bereich 1 … uiMAX_MODULE_APP_ERROR_INDEX gültig. Die Konstante "uiMAX_MODULE_APP_ERROR_INDEX" ist mit dem Wert 64 vorbelegt. Bei Bedarf kann diese Konstante in der Globalen Variablenliste "Global_Const" unterhalb des "Machine"-Ordners angepasst werden.

Für Applikationsfehler in der Maschine sind Nummern im Bereich 1 … uiMAX_MACHINE_APP_ERROR_INDEX gültig. Die Konstante "uiMAX_MACHINE_APP_ERROR_INDEX" ist mit dem Wert 64 vorbelegt. Bei Bedarf kann diese Konstante in der Globalen Variablenliste "Global_Const" unterhalb des "Machine"-Ordners angepasst werden.

Codebeispiel zum Absetzen eines Applikationsfehlers:

```
// Recommended code to issue user specific errors (Example)
IF bDoor1Open THEN
    mSetAppError(1);
END_IF
```

Hinweis zu Fehler-Schweregrade

Das Steuerungs-Diagnosesystem definiert 10 Schweregrade "Severity" (siehe IndraMotion MLC-Diagnosen).

Der Schweregrad hat Einfluss auf die Fehlerreaktion:

- DiagNo = 16#F0xx xxxx ist der geringste Schweregrad (z. B. Applikations-Fehler, FB-Fehler)
- DiagNo = 16#F9xx xxxx ist der höchste Schweregrad (fataler Systemfehler)

Es wird der Fehler mit dem höchsten Schweregrad angezeigt, wenn mehrere Fehler gleichzeitig anstehen (siehe Kap. 5.6.4 "Fehlerreaktion" auf Seite 47).

Hinweis zu Funktionsbaustein-Fehler

mSetFBError

Treten Fehler an aufgerufenen Funktionsbausteinen (z. B. Technologie-FB) auf, so wird empfohlen die Methode "mSetFBError(ErrorID, ErrorIdent)" aufzurufen und die "ErrorID" und "ErrorIdent" Ausgänge des Fehler verursachenden FB an die Methode zu übergeben. Damit wird sichergestellt, dass der Fehler an das Diagnosesystem der Units weitergeleitet wird, und die "ErrorID" und "ErrorIdent" an den Diagnoseausgängen der Unit bereitgestellt wird.

Codebeispiel zum Absetzen eines Funktionsbaustein-Fehlers:

```
// Recommended code a used FB issues an error (Example)
IF fbCrossCutter.Error THEN
    mSetFBError(ErrorID:=fbCrossCutter.ErrorID,
                ErrorIdent:=fbCrossCutter.ErrorIdent);
END_IF
```

11.9.2 Fehlerreaktion

Die Fehlerreaktion hat die Aufgabe Schäden aufgrund von Fehlern des Steuerung- und Antriebssystems möglichst zu vermeiden. Daher wird bei Achs- und Steuerungsfehlern ab Schweregrad 2 automatisch eine Fehlerreaktion durch die Steuerungsfirmware und die realen Achsen durchgeführt (siehe Kap. 5.6.4 "Fehlerreaktion" auf Seite 47).

Bei allen anderen Fehlern mit kleineren Schweregraden (wie z. B. Applikationsfehler, FB-Fehler…) wird keine automatische Fehlerreaktion durch die Steuerungsfirmware und die realen Achsen durchgeführt.

Daher enthält das GAT eine Funktionalität, um Fehler zu erkennen und eine anwenderspezifische Fehlerreaktion durchzuführen.

Allgemeine GAT Fehlerreaktion

Mit der Standardeinstellung ist die GAT Fehlerreaktion eingeschaltet. Dadurch wird im Fehlerfall die angewählte Betriebsart unterbrochen (Ausnahme: Initialisierung) und die Fehlerreaktion ausgeführt. Die Fehlerreaktion ist in der Aktion "actErrorReaction_entry" in jeder Unit des GAT enthalten. Hierbei ist jede Unit für die Fehlerreaktion der zugehörigen Komponenten (wie z. B. Achsen) verantwortlich.

Standardfall für die Fehlerreaktion in der Unitaktion actErrorReaction_entry (alle Achsen verzögern):

```
// ----------------------- Error reaction example 1 --------------------
// Decelerate all module axes to standstill --> switch all axes to ModeAH
// ----------------------------------------------------------------------
FOR uiLoopIndex := 1 TO Axes.uiAxesNum DO
  IF mGetAxis(AxisNumber:=uiLoopIndex, PointerToAxis => pCurrentAxis) THEN
    IF pCurrentAxis^.AxisData.Axis_InAb THEN
      pCurrentAxis^.Ctrl.Admin._OpMode.en := ModeAH;
      // reset SetupMode Inputs to avoid hanging buttons in the
      // System_Overview_All_Axis visualisation
      pCurrentAxis^.Ctrl.SetupMode.Enable := FALSE;
      pCurrentAxis^.Ctrl.SetupMode.JogMinus := FALSE;
      pCurrentAxis^.Ctrl.SetupMode.JogPlus := FALSE;
      pCurrentAxis^.Ctrl.SetupMode.Homing := FALSE;
```

```
    END_IF
  END_IF
END_FOR
```



Für hydraulische Achsen wird eine Stillsetzung über "ModeAb" empfohlen.

Wenn in dem obigen Beispiel "ModeAH" durch "ModeAb" ersetzt wird, dann erfolgt nach der Verzögerung im Stillstand die Momentenfreischaltung der realen Achsen.



Bei **"hängenden Achsen"** darf keine Momentenfreischaltung erfolgen, um ein Absenken der Achsen zu vermeiden. Bei diesen Anwendungen wird empfohlen, alle Achsen in "ModeAH" zu schalten.

Oftmals besteht die Anforderung, Achsen im **Synchronverbund** zu verzögern, um z. B. das Reißen einer Materialbahn zu vermeiden. Bei diesen Anwendungen wird empfohlen, alle Achsen im Synchronverbund zu belassen, und nur den Master zu verzögern. Dieser Anwendungsfall ist im folgenden Beispiel dargestellt:

Beispiel für synchrone Stillsetzung der Achsen in der Unitaktion "actErrorReaction_entry" (nur nicht synchrone Achsen werden über "ModeAH" verzögert):

```
// ------------------ Error reaction example 2 ---------------------
// Keep all axes in synchronization with the master axis
// and decelerate only the master axis
// -----------------------------------------------------------------
FOR uiLoopIndex := 1 TO Axes.uiAxesNum DO
  IF mGetAxis(AxisNumber:=uiLoopIndex, PointerToAxis => pCurrentAxis) THEN
    // reset SetupMode Inputs to avoid hanging buttons in the
    // System_Overview_All_Axis visualisation
    pCurrentAxis^.Ctrl.SetupMode.Enable := FALSE;
    pCurrentAxis^.Ctrl.SetupMode.JogMinus := FALSE;
    pCurrentAxis^.Ctrl.SetupMode.JogPlus := FALSE;
    pCurrentAxis^.Ctrl.SetupMode.Homing := FALSE;
    IF (pCurrentAxis^.AxisData.SynchronizedMotion) THEN
      ; // Axis is in sync motion --> no reaction as the master axis
        // decelerates to standstill and the slave axis follows in sync
    ELSE
      IF pCurrentAxis^.AxisData.Axis_InAb THEN
        pCurrentAxis^.Ctrl.Admin._OpMode.en := ModeAH;
        ; // Axis is not in sync motion --> command to stop
      END_IF
    END_IF
  END_IF
END_FOR
```

Für die synchrone Stillsetzung sind bei realen Achsen folgende zusätzliche Einstellungen notwendig:

- Modulbusreaktion abschalten
- Fehlerreaktion der Antriebe mit "Bestmöglichen Stillsetzung = 30s NC Reaktion" konfigurieren



Bei der Einstellung "Bestmöglichen Stillsetzung = 30s NC Reaktion" einer realen Achse wird bei dem IndraLogic-XLC/IndraMotion MLC-Steuerungssystem die fehlerbehaftete Achse bei einem F2-Fehler mit der Notstopp-Rampe A-0-0224 verzögert. Alle anderen fehlerfreien Achsen bleiben unbeeinflusst.

Abschalten der allgemeinen GAT Fehlerreaktion

In Sonderfällen kann die allgemeine GAT Fehlerreaktion für jede Unit einzeln abgeschaltet werden. Das Abschalten der Fehlerreaktion kann beispielswei-

se für eine Service-Betriebsart genutzt werden, um z. B. Achsen auch im Fehlerzustand zu bewegen. Durch setzen des Bit "DisableErrorReaction" im Unit Steuerwort wird für diese Unit die allgemeine GAT Fehlerreaktion abgeschaltet.

Implementieren Sie den dafür erforderlichen Programmcode wie folgt:

- Stelle: Funktionsbaustein "TE_MachineControl", Aktion "actInitCold-Start"
- Ergänzen Sie hier den Code wie folgt:

Abschaltung der allgemeinen GAT Fehlerreaktion für alle Units:

```
FOR uiLoopIndex := 1 TO Units.uiMaxUnitNum DO
  Units.arAll[uiLoopIndex].Ctrl.DisableErrorReaction := TRUE;
END_FOR
```

Abschaltung der allgemeinen GAT Fehlerreaktion für eine Unit (<UnitName> mit dem Namen der Unit ersetzen):

```
Units.<UnitName>.Ctrl.DisableErrorReaction := TRUE;
```

Die Fehlerreaktion der Units kann auch nach der Initialisierung in dem Funktionsbaustein "TE_MachineControl" ein- und ausgeschaltet werden.

GAT Fehlerreaktion während der Initialisierung

Mit der Standardeinstellung ist die GAT Fehlerreaktion während der Initialisierung abgeschaltet. Auftretende Fehler werden jedoch angezeigt und können gelöscht werden. Für Sonderfälle kann die Fehlerreaktion während der Initialisierung eingeschaltet werden.

Implementieren Sie den dafür erforderlichen Programmcode wie folgt:

- Stelle: Funktionsbaustein "TE_MachineControl", Aktion "actInitCold-Start"
- Ergänzen Sie hier den Code wie folgt:

Einschaltung der GAT Fehlerreaktion während der Initialisierung für alle Units:

Programm:

```
FOR uiLoopIndex := 1 TO Units.uiMaxUnitNum DO
  Units.arAll[uiLoopIndex].Ctrl.EnableErrorReactionInit := TRUE;
END_FOR
```

Einschaltung der GAT Fehlerreaktion für eine Unit (<UnitName> mit dem Namen der Unit ersetzen):

```
Units.<UnitName>.Ctrl.EnableErrorReactionInit := TRUE;
```

☞ Die Fehlerreaktion während der Initialisierung wird nur dann ausgeführt, wenn die allgemeine GAT Fehlerreaktion eingeschaltet ist (DisableErrorHandling_gb = FALSE).

11.9.3 Fehlerlöschen

Über eine positive Flanke am Element **"ClearAllError"** der Variablen **"stErr_Handling_Ctrl_gb"** (Eingang des "TE_ErrorHandling") wird das zentrale Fehlerlöschen ausgelöst, und automatisch an alle unterlagerten Komponenten (wie z. B. Imc-Schnittstelle[1)], Units, Axisschnittstelle …) weitergeleitet.

1) *IndraMotionControl Interface = IndraLogic Bibliothek mit Schnittstelle zur Steuerung, wie. z. B. Fehler löschen*

Ist der fehlerfreie Zustand erreicht, wird die vom Bediener angewählte Betriebsart wieder aktiv.

11.9.4 Fehler protokollieren

*Ist in der Errorhandling Steuerstruktur (Variable "stErr_Handling_Ctrl_gb") das Strukturelement **"LogError"** gesetzt (TRUE), werden zusätzlich folgende GAT spezifische Fehler in dem Fehler-/Diagnosespeicher der Steuerung eingetragen:*

- 16#F0**AE** xxxx: Applikations-Fehler (z. B. geöffnete Schutztür) (xxxx = frei durch den Anwender definierbar)
- 16#F0**FB** 0000: Funktionsbaustein-Fehler (z. B. Fehler an einem Technologie-Funktionsbaustein)
- 16#F0**AA** 0000: GAT interner Fehler
- 16#F0**B1...B9** xxxx: Plugin Zusatzdiagnose-Fehler (z. B. durch PROFIBUS Diagnose-Plugin)



Wenn "LogError" auf TRUE gesetzt ist, werden die GAT-spezifischen Fehler auch am Steuerungsdisplay und an den Diagnoseparametern der Steuerung angezeigt.

11.10 Produkt- und Maschinendaten-Verwaltung

11.10.1 Einleitung

Das GAT stellt vorbereitete Funktionen zur Verfügung um Maschinen- bzw. Produkt-Daten in einem Datensatz vorzubereiten, und zu einem genau definierten Zeitpunkt zu aktivieren. Maschinen- und Produktdaten werden hier jeweils getrennt betrachtet und verwaltet. Das nachfolgend beschriebene Datenhandling ist in der Methode "mDataHandler" des "TE_MachineControl" als Open Source Konzept umgesetzt, und kann entsprechend erweitert oder modifiziert werden.

11.10.2 Maschinendaten-Verwaltung

Maschinendaten sind die Daten die sich auf die Maschine beziehen und im Betrieb verändert werden können. Beispiele hierfür sind Produktionsgeschwindigkeit, Beschleunigungs- und Verzögerungsrampen...

Diese Maschinendaten werden über die Variable "stMachineDataEdit_gb" der Variablenliste "Global_Machine" editiert/vorbereitet. Der Inhalt dieser Variablen ist anwendungsspezifisch und wird über die Datenstruktur "TE_MACHINE_DATA" vorgegeben.

Durch eine positive Flanke am Eingang "stControlDataActive_gb.bSetMachineData" werden diese Daten auf die Variable "stMachineDataActive_gb" geschrieben und damit wirksam. Die Quittierung erfolgt über "stControlDataActive_gb.bSetMachineDataDone".

Umgekehrt wird durch positive Flanke am Eingang "stControlDataActive_gb.bGetMachineData" der aktive Datensatz auf den editierbaren Datensatz "stMachineDataEdit_gb" kopiert, und über "stControlDataActive_gb.bGetMachineDataDone" quittiert.

11.10.3 Produktdaten-Verwaltung

Produktdaten sind die Daten die das Produkt beschreiben und zur Herstellung erforderlich sind. Beispiele hierfür sind Produktlänge, Rezepturdaten...

Diese Produktdaten werden über die Variable "stProductDataEdit_gb" der Variablenliste "Global_Machine" editiert/vorbereitet. Der Inhalt dieser Variab-

len ist anwendungsspezifisch und wird über die Datenstruktur "TE_PRODUCT_DATA" vorgegeben.

Durch eine positive Flanke am Eingang "stControlDataActive_gb.bSetProductData" werden diese Daten auf die Variable "stProductDataActive_gb" geschrieben und damit wirksam. Die Quittierung erfolgt über "stControlDataActive_gb.bSetProductDataDone".

Umgekehrt wird durch positive Flanke am Eingang "stControlDataActive_gb.bGetMachineData" der aktive Datensatz auf den editierbaren Datensatz "stProductDataEdit_gb" kopiert, und über "stControlDataActive_gb.bGetProductDataDone" quittiert.

11.11 Visualisierung

11.11.1 Umschalten der Visualisierung

Im GAT sind Funktionen vorbereitet um die Maschine wahlweise über die integrierte Inbetriebnahme- oder eine externe Visualisierung zu steuern. Hierbei darf jedoch zum gleichen Zeitpunkt nur eine Visualisierung aktiv sein. Um dies sicherzustellen, ist jeweils für die integrierte Inbetriebnahme- und der externen Visualisierung eine eigene Variable vorgesehen.

Die integrierte Inbetriebnahme-Visualisierung nutzt die Variable "**stControlDataVisuIL_gb**" aus der globalen Variablenliste "GlobalMachine".

Die externe Visualisierung (z. B. durch Feldbus oder Kunden HMI) nutzt die Variable "**stControlDataCustomer_gb**" aus der globalen Variablenliste "GlobalMachine".

Im Anwendungsprogramm ist die Variable "**stControlDataActive_gb**" wirksam, die entweder die integrierte oder die externe Visualisierung referenziert. Die integrierte Inbetriebnahme-Visualisierung kann über die Schaltfläche **Remote On** die Referenz umschalten und somit die Kontrolle übernehmen oder abgeben.

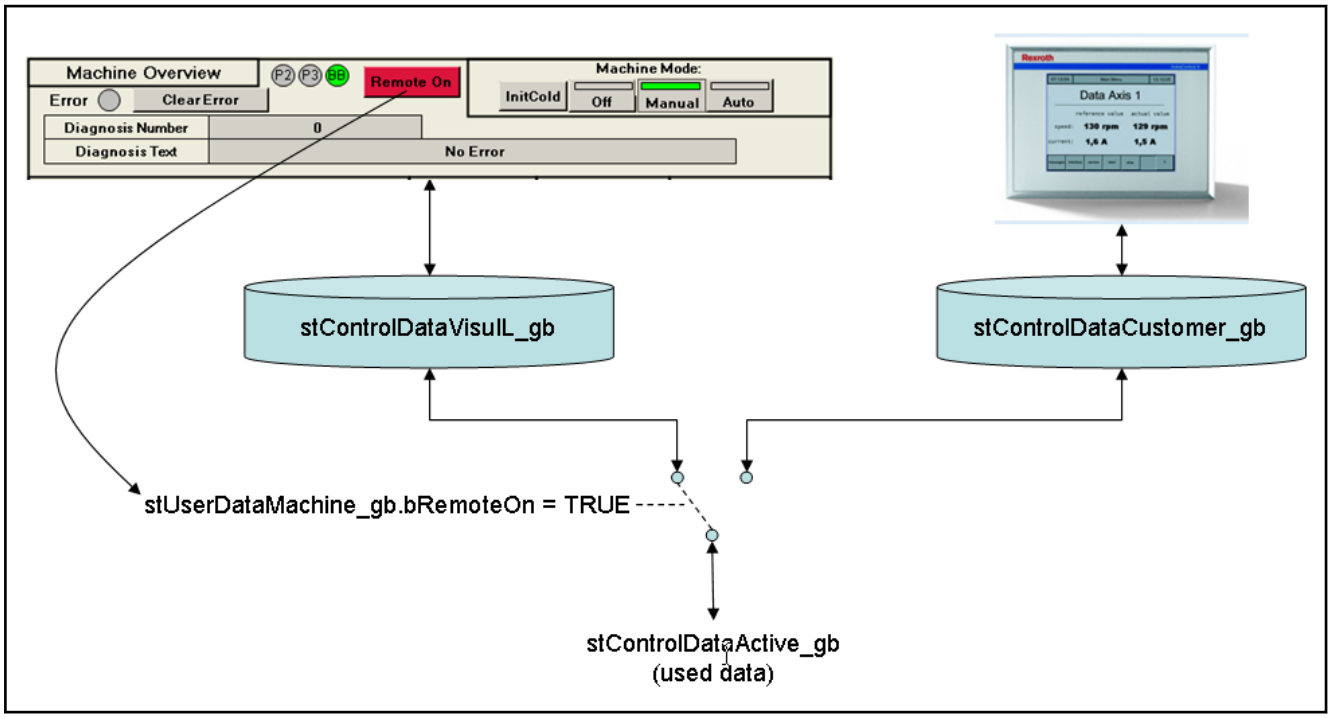


Abb. 11-29: Umschaltung zwischen Inbetriebnahme- und externer Visualisierung

GAT mit dezentraler Zustandsmaschine

11.11.2 Interne Visualisierung zur Inbetriebnahmeunterstützung

Allgemeines

Das GAT beinhaltet vordefinierte IndraLogic-Visualisierungsobjekte, die bei der Inbetriebnahme unterstützen und erweiterte Diagnosemöglichkeiten bieten.

System_Overview

Das Visualisierungselement "System_Overview" ist für einzelne Achsen instanziierbar.

In der Visualisierung "System_overview_All_Axis" ist dies exemplarisch für die Achsen "MyVirtualAxis1" und "MyVirtualAxis2" umgesetzt.

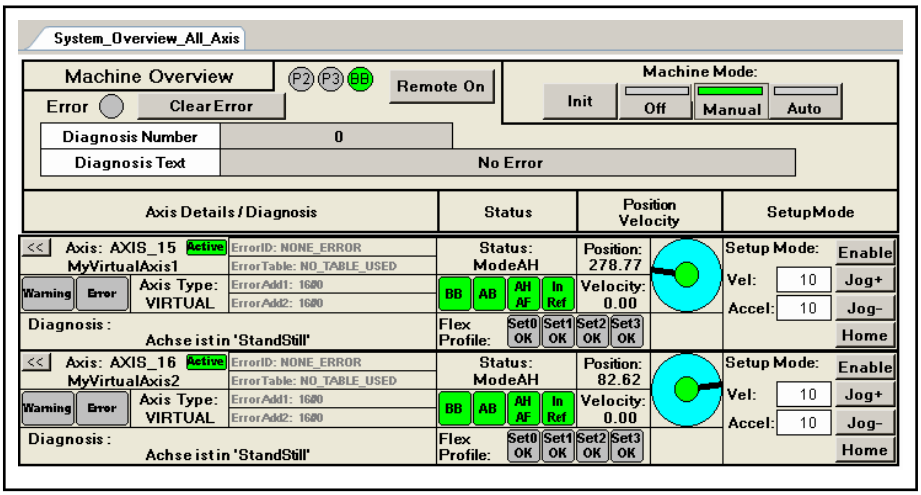


Abb.11-30: Interne Visualisierung zur Inbetriebnahmeunterstützung

Die Inbetriebnahme-Visualisierung wird erst mit Betätigen der Schaltfläche **Remote On** wirksam.

Über das Objekt "Frame" kann in der Visualisierung eine Instanz des Visualisierungsobjekts "System_Overview" angelegt werden.

GAT mit dezentraler Zustandsmaschine

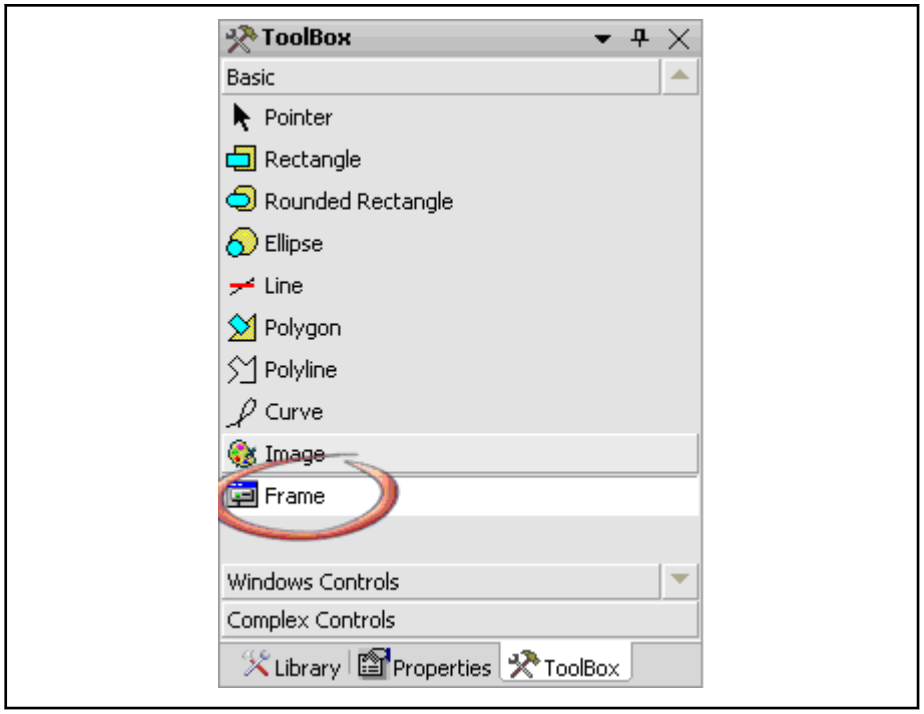


Abb.11-31: Anlegen einer Instanz des Visualisierungsobjekts "System_Overview"

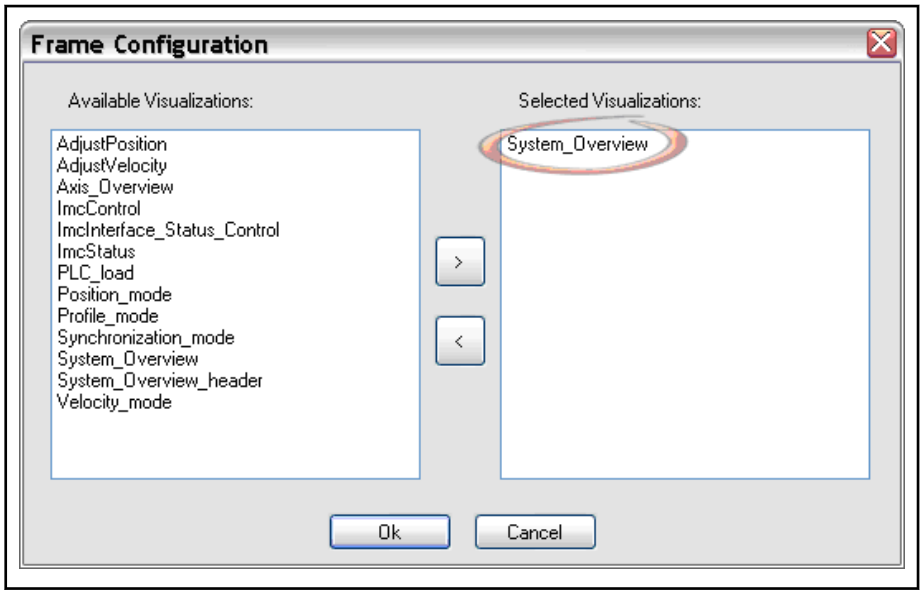


Abb.11-32: Anlegen einer Instanz des Visualisierungsobjekts "System_Overview"

GAT mit dezentraler Zustandsmaschine

Unter Properties – References muss nun die entsprechende Achsreferenz eingetragen werden:

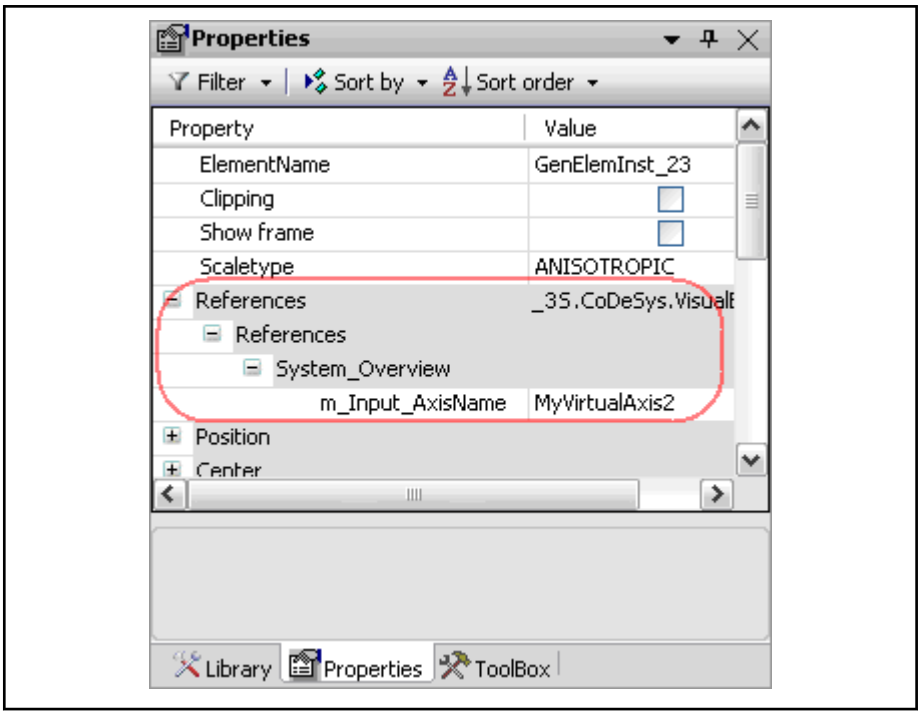


Abb.11-33: Anlegen einer Instanz des Visualisierungsobjekts "System_Overview"

11.11.3 Externe Visualisierung mit separaten HMI (WinStudio-Visualisierung)

Allgemeines

Für das GAT steht eine vordefinierte WinStudio-Visualisierung zur Verfügung, die über den Wizard in das Projekt eingefügt werden kann.

☞ Zur Bearbeitung der WinStudio-Visualisierung wird mindestens folgende Entwicklungsumgebungslizenz oder größer benötigt:

WinStudio-Entwicklungslizenz, max. 1.500 Variablen (Mat.-Nr. R911323596)

Zur Ausführung auf einem VEP wird mindestens folgende Runtime-Lizenz benötigt:

WinStudio-Runtimelizenz für Geräte mit Windows CE als Betriebssystem, max. 1.500 Variablen (Mat.-Nr. R911323620)

Einfügen der WinStudio-Visualisierung

☞ Um die WinStudio-Visualisierung dem Projekt hinzufügen zu können, muss ein entsprechendes Visualisierungsgerät im Projekt vorhanden sein.

Fügen Sie mittels "Drag and Drop" ein VEP aus der Gerätebibliothek in den Projektbaum ein und konfigurieren es wie folgt:

- Gerätetyp: VEPxx (CE)
- Applikationstyp: WinStudio-Anwendung

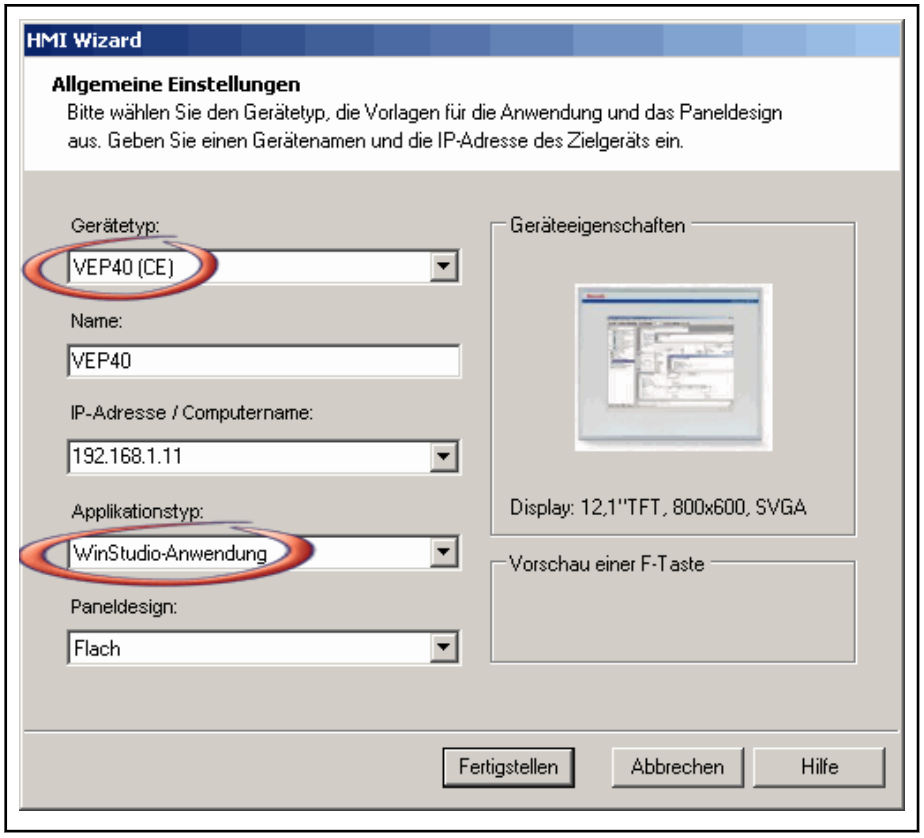


Abb.11-34: Visualisierungsgerät einfügen

Wenn ein entsprechendes Visualisierungsgerät vorhanden ist, wird über folgenden Link im Wizard die WinStudio-Visualisierung eingefügt:

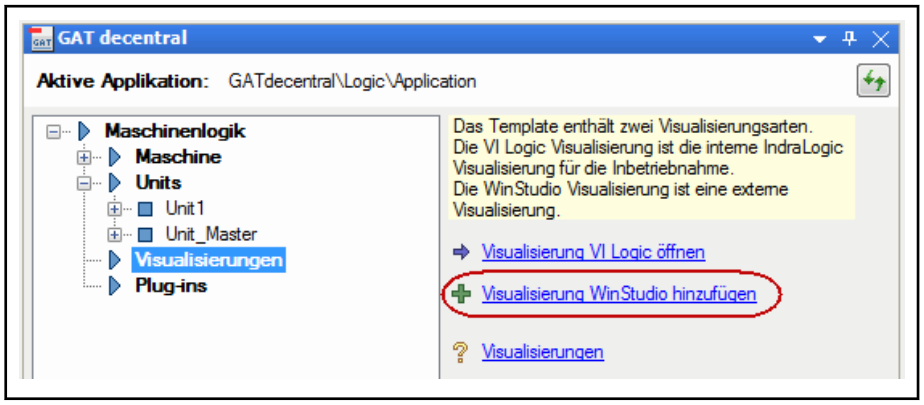


Abb.11-35: WinStudio-Visualisierung einfügen

Im nächsten Dialog des Wizard werden alle Visualisierungsgeräte des Projekts angeboten, deren Konfiguration der Vorgabe entspricht.

GAT mit dezentraler Zustandsmaschine

Nach Auswahl des entsprechenden Visualisierungsgerätes kann mittels **Übernehmen** das Einfügen der WinStudio-Visualisierung ausgeführt werden:

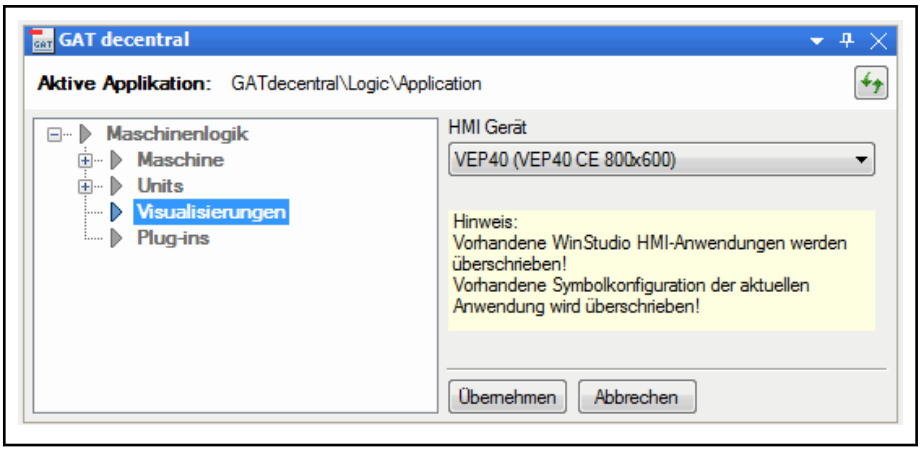


Abb.11-36: Visualisierungsgerät auswählen – WinStudio-Visualisierung einfügen

Bestehende Visualisierungsdaten des Visualisierungsgerätes und die Symbolkonfiguration der Steuerung werden durch das Einfügen der WinStudio-Visualisierung überschrieben!

Nach dem Einfügen können die Visualisierungsdaten bearbeitet werden.

Die Konfiguration der Steuerungsadresse erfolgt im "Eigenschaften"-Dialog des BR-WS Treibers:

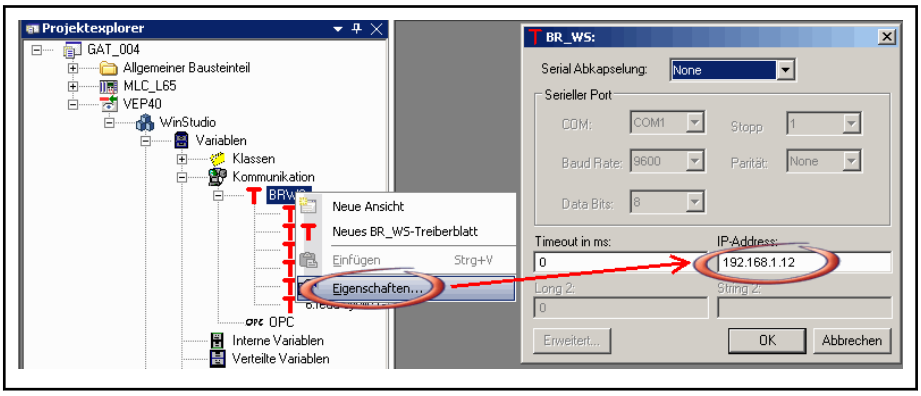


Abb.11-37: Einstellen der IP-Adresse der Steuerung

Im Kontextmenü des WinStudio-Knotens kann nun über  "Anwendung Starten" die Emulation der Winstudio-Visualisierung gestartet werden.

Dialog-Struktur

Die Bedienbilder teilen sich in 3 Teilbereiche auf:

- Kopfzeile mit Statusinformationen
- Hauptbildbereich mit menüabhängigen Anzeigen und Funktionstasten
- Fußzeile mit Tasten zur Bedienbildumschaltung

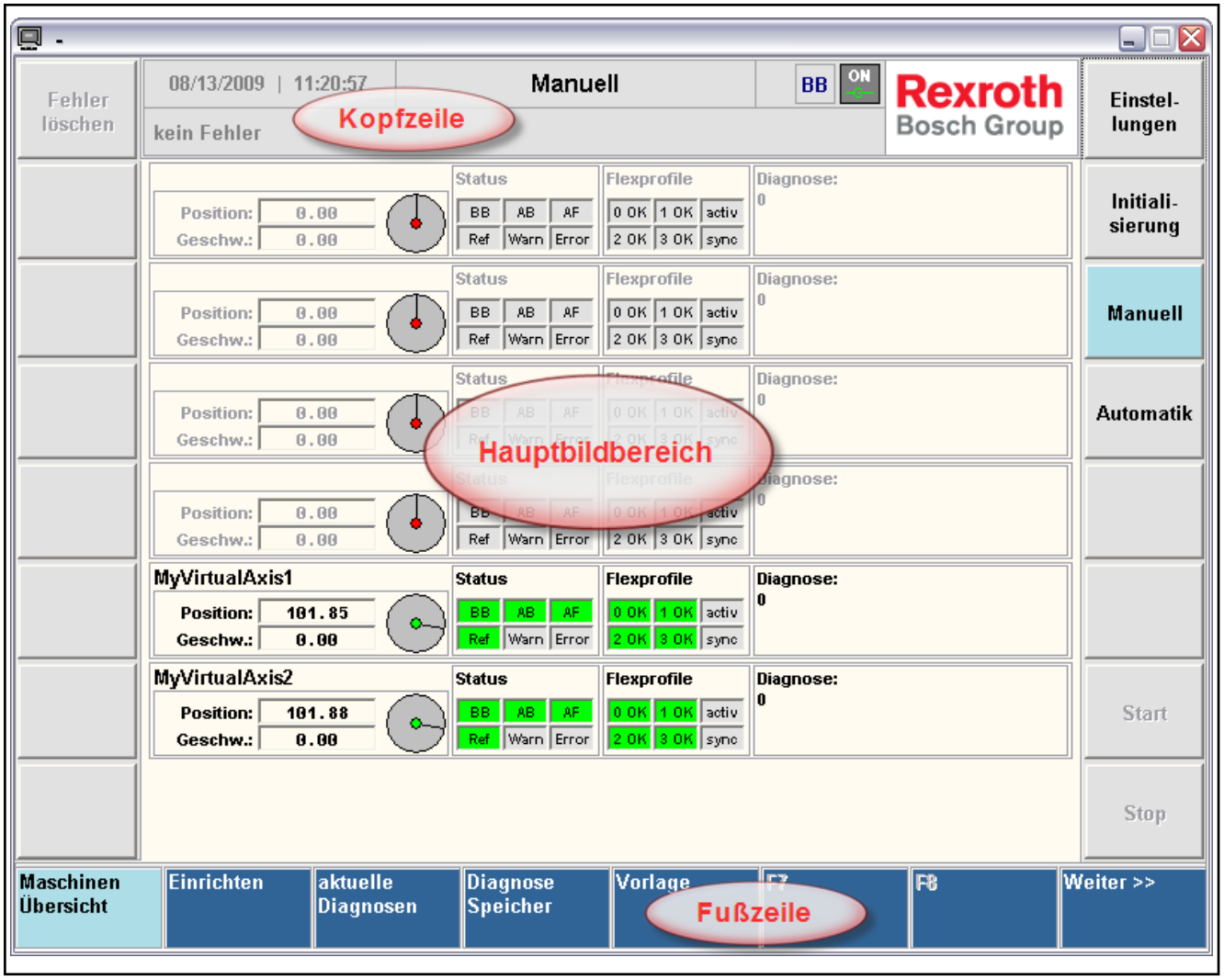


Abb.11-38: WinStudio-Visualisierung

*Die **Kopfzeile** beinhaltet:*

- Betriebsartenanzeige
- Globale Funktionstasten zum "Fehlerlöschen" und Öffnen des "Einstellungen"-Dialogs (Sprachumschaltung, Einstellung der Dezimalstellen für numerische Anzeigen)
- Anzeige aktuell anstehender Maschinen-Diagnosen
- Anzeige der aktuellen sercos Phase (inklusive aktiver Phasenumschaltung)
- Anzeige des Kommunikationsstatus

Der **Hauptbildbereich** beinhaltet die bedienbildabhängigen Anzeigen und Funktionstasten (z. B. Achsinformationen, Betriebsartenumschaltung, …).

Die **Fußzeile** beinhaltet die Tasten zur Bedienbildumschaltung.

Erweitern der Visualisierung

Bedienbild einfügen

GAT mit dezentraler Zustandsmaschine

Zum Erweitern der Visualisierung können neue Bedienbilder mittels Kopieren der vorhanden "Template"-Bilder und Einfügen unter neuem Namen angelegt werden:

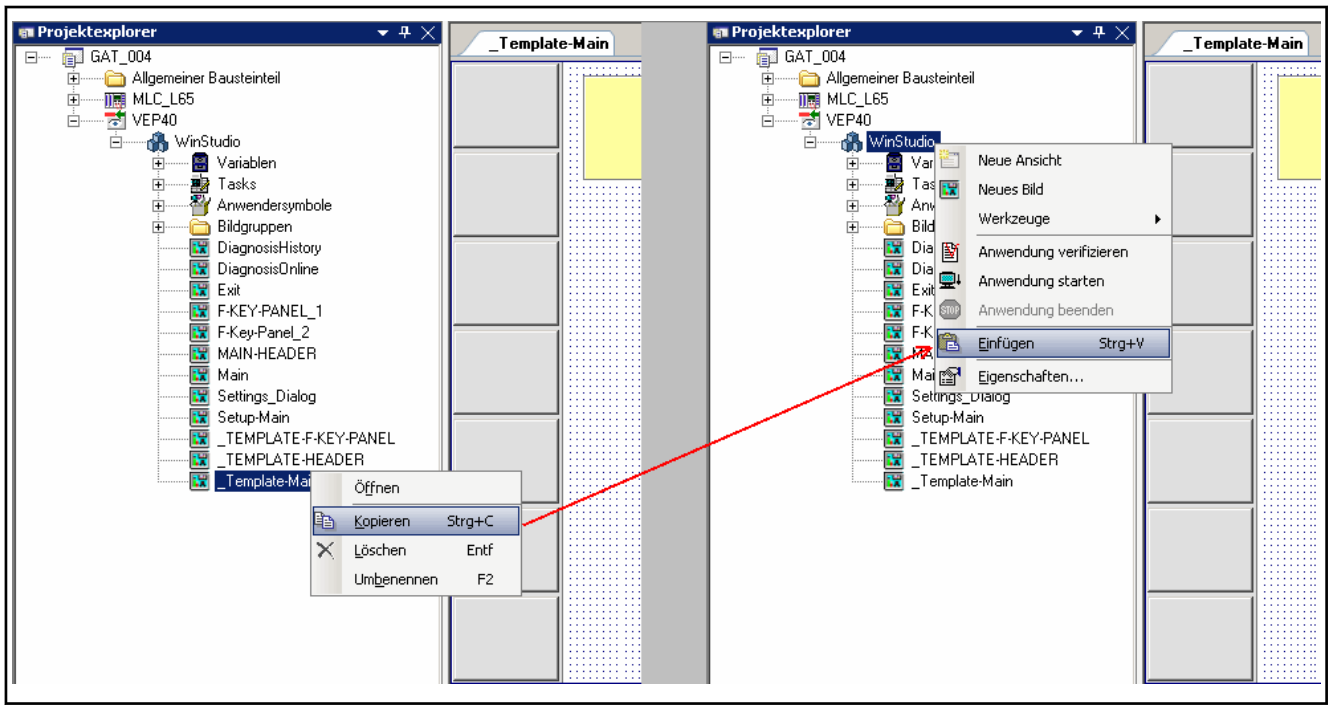


Abb.11-39: Bedienbild duplizieren

Die Elemente zum Anzeigen der Achsdaten sind gruppierte Einzelobjekte, die durch "Kopieren" und "Einfügen" dupliziert werden können. Zum Umkonfigurieren des eingefügten Objekts auf eine andere Achse muss nun über die Ersetzen-Funktion des jeweiligen Objekts der Array-Index der Achse angepasst werden (siehe nachfolgende Abbildungen).

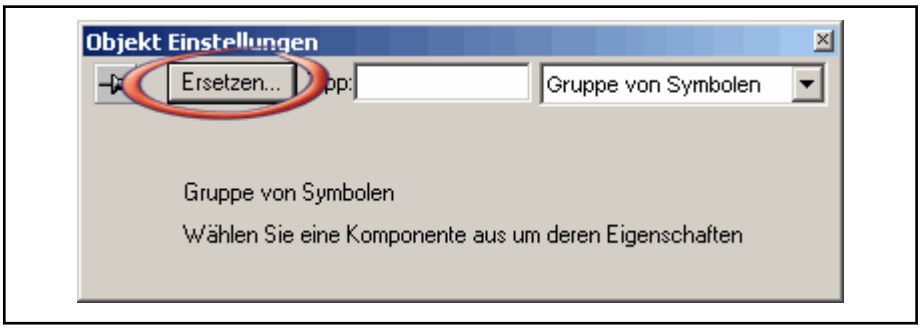


Abb.11-40: Umkonfigurieren des eingefügten Objekts

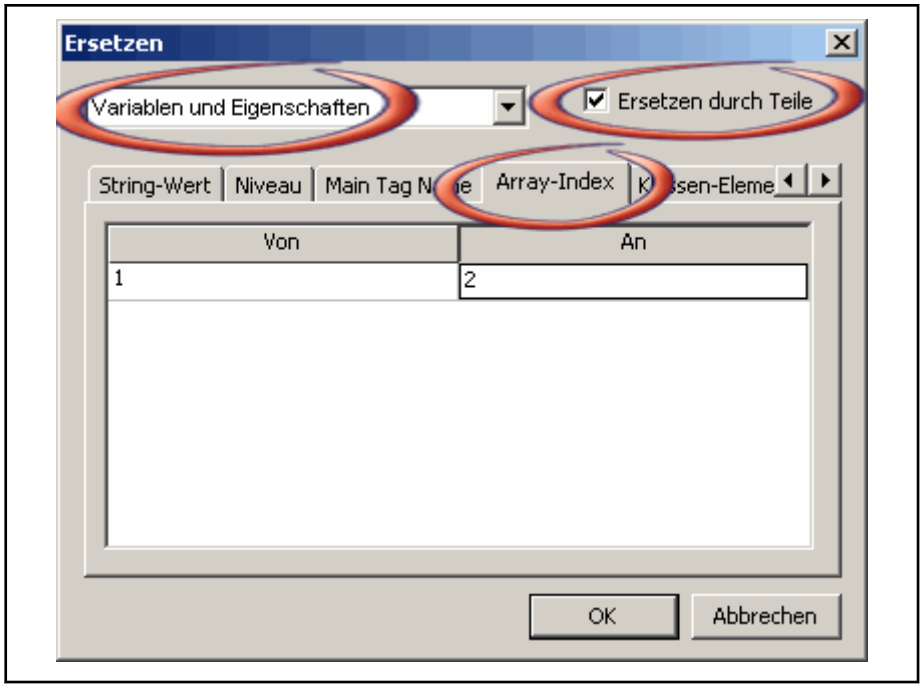


Abb.11-41: Umkonfigurieren des eingefügten Objekts

Fußzeile erweitern

Die Tasten zur Bedienbildumschaltung sind als parametrierbare Grafik-Objekte angelegt (linked Symbol) und verfügen über folgende Übergabe-Parameter:

- Caption -> Beschriftung der Taste
- Disable -> Eingang zum Sperren der Tastenfunktionalität (ausgegraute Beschriftung)
- ScreenName -> Bildname des zu öffnenden Bedienbildes (bzw. Bildgruppe)
- ScreenNumber -> Nummer, die dem Bild zugewiesen werden soll (zum farblichen hervorheben der jeweiligen Taste)

Erweitern der maximalen Achsanzahl

Die Visualisierung ist nach der Importierung für 16 Achsen ausgelegt. Dies entspricht der maximalen Achsanzahl der IndraControl L25. Benötigt man mehr Achsen, müssen die entsprechenden Treiberblätter der Visualisierung aufgerufen werden.

GAT mit dezentraler Zustandsmaschine

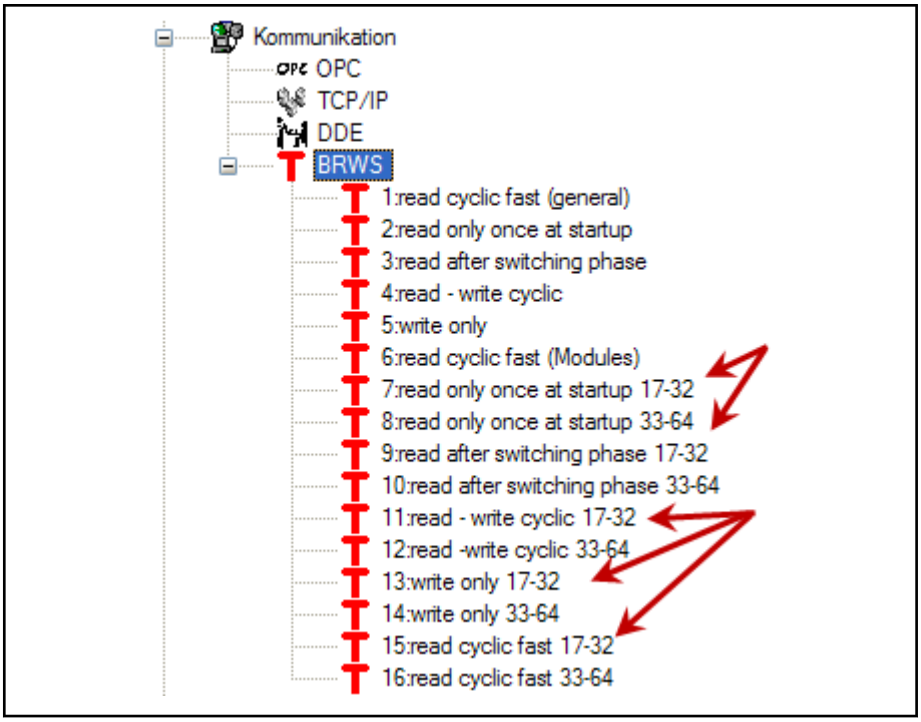


Abb.11-42: Übersicht Treiberblätter

Mit einem Doppelklick auf das entsprechende Treiberblatt öffnen sich die Einstellungen. Die Lese- und Schreibtrigger-Variablen sind bei den inaktiven Treiberblättern leer und müssen um die Einträge aus den aktiven Treiberblättern ergänzt werden.

Beispiel anhand des Treiberblatts "read only once at startup 17-32":

Dort muss als Triggervariable der Eintrag "stCom.TrigOnce" vom Treiberblatt "read only once at startup" in das Treiberblatt "read only once at startup 17-32" kopiert werden.

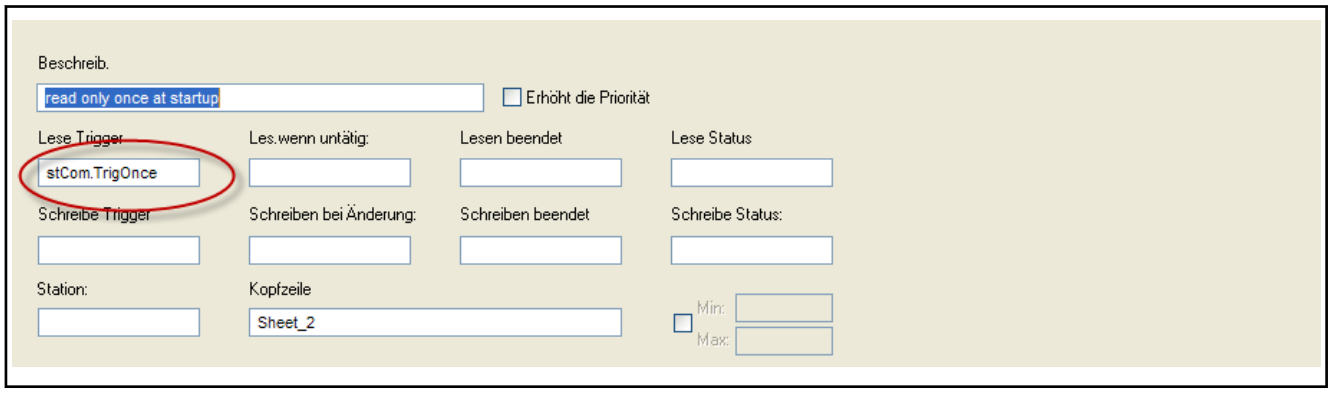


Abb.11-43: Treiberblatteinstellungen des aktiven Treiberblattes

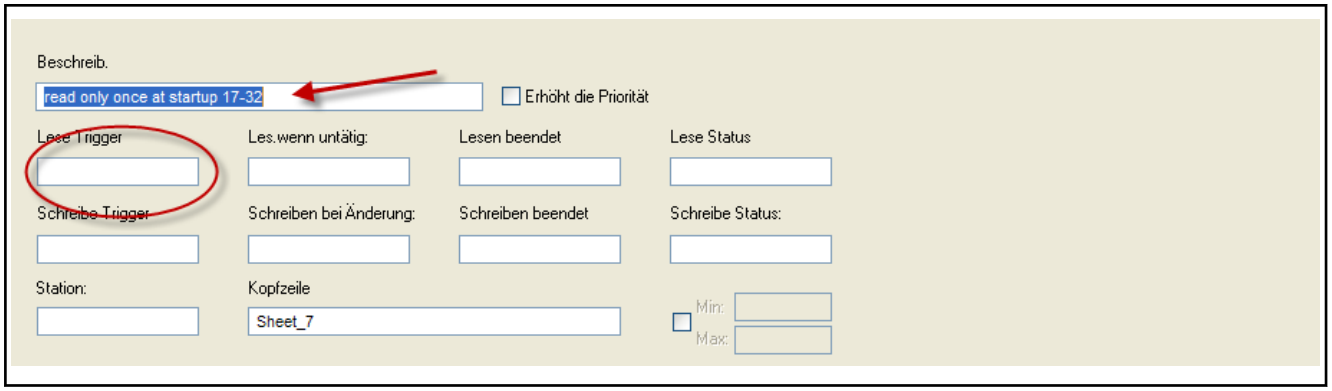


Abb.11-44: Treiberblatteinstellungen für Erweiterung auf 32 Achsen

Die passenden Einträge müssen für alle Treiberblätter abgeglichen werden. Die Zahlen hinter den Treiberblattnamen geben an, für welche Achsen diese Treiberblätter angelegt wurden.

Kommunikationsanbindung

Die Kommunikation zwischen Visualisierung und Steuerung erfolgt symbolisch über den BR-WS Treiber. Hierzu ist es erforderlich, die Symboleinträge für die zu visualisierenden SPS-Variablen zu erzeugen (siehe hierzu Kap. "Symbolkonfiguration" auf Seite 50).

Die benötigte Symbolkonfiguration der vordefinierten WinStudio-Visualisierung beinhaltet folgende Einträge:

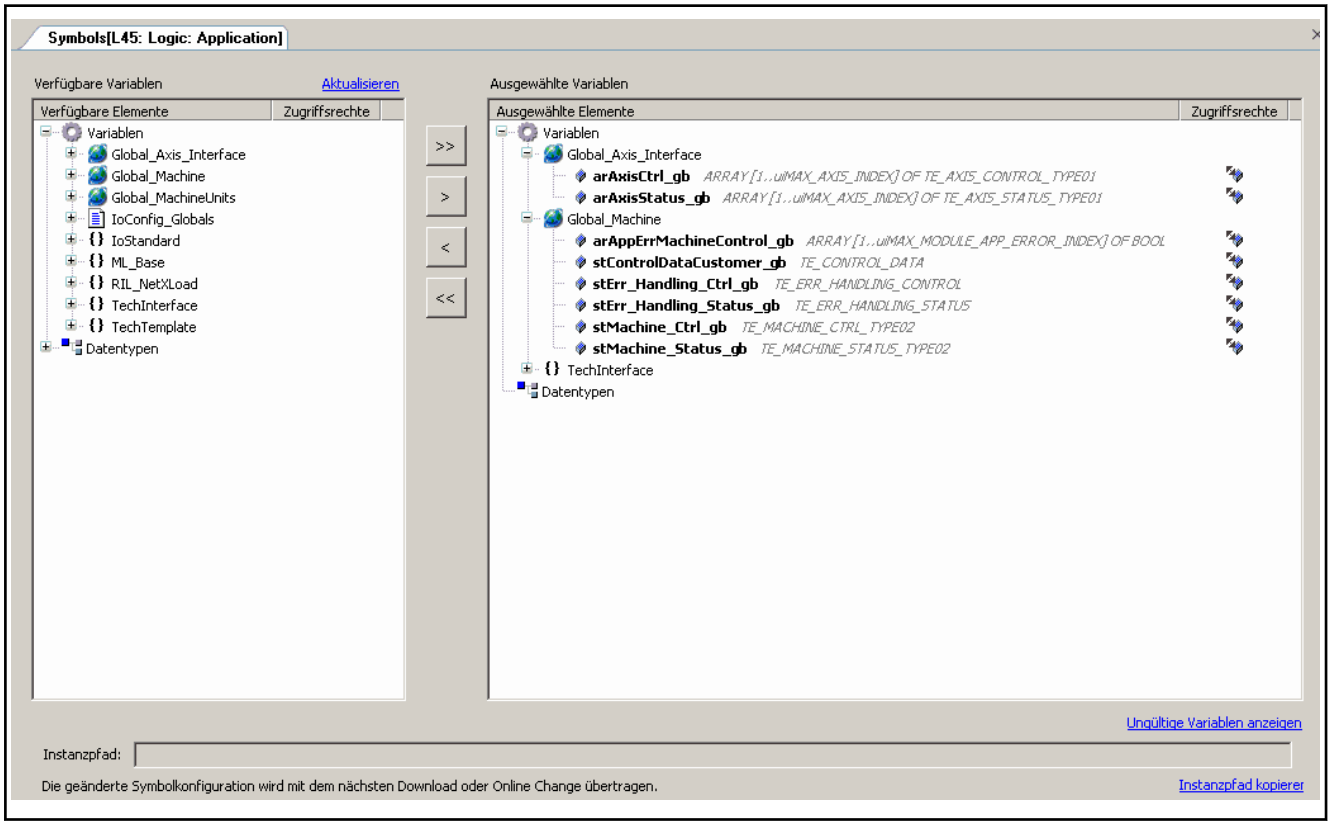


Abb.11-45: Symbolkonfiguration der vordefinierten WinStudio-Visualisierung

GAT mit dezentraler Zustandsmaschine



Symboleinträge für Variablen aus Globalen Variablenlisten die aus einer Bibliothek stammen, müssen nach einer Versionsänderung der entsprechenden Bibliothek erneut in die Symbolkonfiguration eingefügt werden.

Einstellen der IP-Adresse der Steuerung:

Die Konfiguration der Steuerungsadresse erfolgt im "Eigenschaften-Dialog" des BR-WS Treibers:

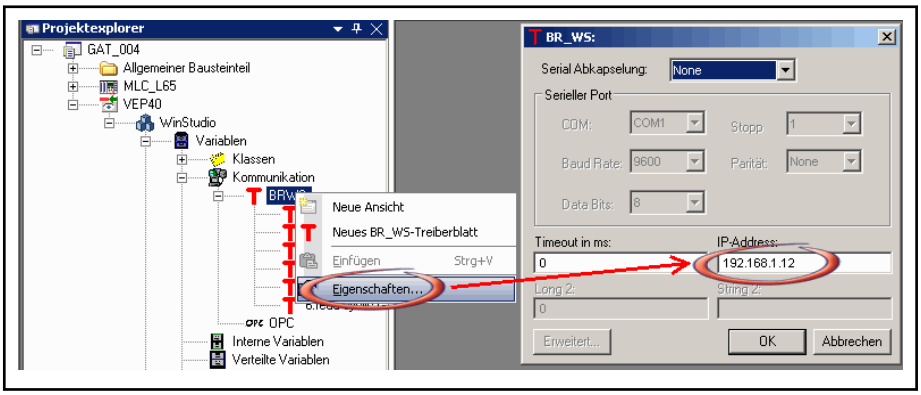


Abb.11-46: Einstellen der IP-Adresse der Steuerung

Um die Kommunikation zu strukturieren wurden die Variablen in verschiedene Treiberblätter aufgeteilt:

1. read cyclic fast: -> Variable deren Werte zyklisch in kurzen Intervallen gelesen werden sollen (z. B. Diagnosenummern, Achspositionen, …)
2. read only once: -> Variablen die einmalig nach dem Start der Visualisierung gelesen werden (z. B. Achsnamen)
3. read after switching phase: Variablen die nur einmalig nach einer Phasenumschaltung gelesen werden (z. B. ob die Achse aktiviert wurde)
4. read - write cyclic: -> Variablen, deren Werte durch die Visualisierung sowohl gelesen als auch geschrieben werden sollen (z. B. Maschinengeschwindigkeit, Achsgeschwindigkeit für Einrichtbetrieb, …)
5. write only: -> Variablen die durch die Visualisierung nur geschrieben werden sollen (Triggersignal zum Fehlerlöschen, …)

Zum Erweitern eines Treiberblattes kann die offline Symboldatei (*.XML) mit dem BR-WS Browser durchsucht und die entsprechende Variable selektiert werden.

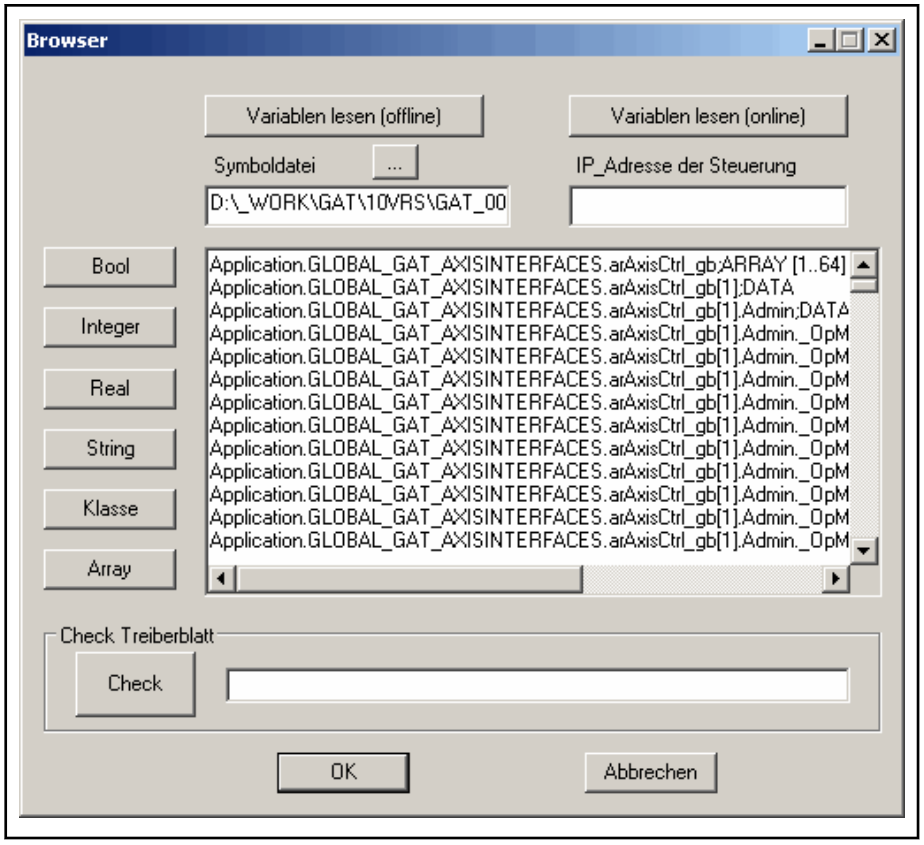


Abb.11-47: Erweitern des Treiberblattes

Mehrsprachigkeit

Die Mehrsprachigkeit der Visualisierung wurde mit der in IndraWorks integrierten Übersetzungsfunktionalität realisiert.

Hierbei sind die Übersetzungsdaten bereits für Steuerungs- und Antriebs-Diagnosen, sowie den verwendeten Texten der Visualisierung vorbereitet. Die Übersetzungsdaten können mit IndraWorks modifiziert bzw. ergänzt werden.

Antriebsdiagnosen sind hierbei in deutsch, englisch, französisch, spanisch und türkisch enthalten.

Alle andere Texte sind in deutsch und englisch enthalten.

11.12 Schnittstellen zum GAT mit dezentraler Zustandsmaschine

11.12.1 Allgemeines

Die Schnittstellendaten des GAT werden in Variablenlisten bereitgestellt. Ein Teil der Daten wird über Bibliotheken bereitgestellt.

11.12.2 Daten des Maschinenverwalters (TE_MachineControl)

TE_MachineControl Eingangsdaten

Die Eingangsdaten des TE_MachineControl sind in der Struktur **stMachineCtrl_gb** (Variablenliste "Global_Machine") vom Typ **TE_MACHINE_CTRL_TYPE02** zusammengefasst.

GAT mit dezentraler Zustandsmaschine

Die folgende Tabelle beschreibt die Elemente der Eingangsstruktur:

I/O-Typ	Name	Datentyp	Beschreibung
I	Enable	BOOL	Bearbeitungsfreigabe
I	ClearError	BOOL	Positiver Flanke führt lokales Fehlerlöschen aus

I Eingang (schreib- und lesbar)

Abb.11-48:

TE_MachineControl Ausgangsdaten

Die Ausgangsdaten der "TE_MachineControl" sind in der Struktur **stMachineStatus_gb** (Variablenliste "Global_Machine") vom Typ **TE_MACHINE_STATUS_TYPE02** zusammengefasst.

Die folgende Tabelle beschreibt die Elemente der Ausgangsstruktur:

I/O-Typ	Name	Datentyp	Beschreibung
O	InOperation	BOOL	Bearbeitung läuft
O	OpMode	TE_OPMODES_PREDEFINED_TYPE02	Quittierung der aktuellen Betriebsart der Maschine
O	Error	BOOL	Lokaler Fehler steht an
O	ErrorID	ERROR_CODE	Detailinformation wenn DiagNo = 16#F0FB 0000 oder16#F0AA xxxx
O	ErrorIdent	ERROR_STRUCT	Detailinformation wenn DiagNo = 16#F0FB 0000 oder16#F0AA xxxx
O	DiagNo	DWORD	Nummer gemäß Dokumentation "Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnosen" mit den folgenden Erweiterungen: DiagNo = 16#F0AE xxxx: Applikations-Fehler (z. B. geöffnete Schutztür) DiagNo = 16#F0FB 0000: FB-Fehler DiagNo = 16#F0AA xxxx: GAT-interner Fehler DiagNo = 16#F0B1...B9 xxxx: Plugin Zusatzdiagnose Fehler (z. B. durch PROFIBUS Diagnose Plugin)
O	DiagMessage	STRING(60)	Diagnosetext der Steuerung/Achse

O Ausgang (lesbar)

Abb.11-49:

11.12.3 Daten der zentralen Fehlerverwaltung (TE_ErrorHandling)

TE_ErrorHandling Eingangsdaten

Die Eingangsdaten des "TE_MachineControl" sind in der Struktur **stErr_Handling_Ctrl_gb** (Variablenliste "Global_Machine") vom Typ **TE_ERR_HANDLING_CONTROL** zusammengefasst.

Die folgende Tabelle beschreibt die Elemente der Eingangsstruktur:

I/O-Typ	Name	Datentyp	Beschreibung
I	Enable	BOOL	Bearbeitungsfreigabe
I	ClearAllError	BOOL	Positiver Flanke führt **globales** Fehlerlöschen aus
I	LogError	BOOL	TRUE = GAT spezifische Fehler (Applikations-, FB-, GAT- und Plugin-Fehler) werden zusätzlich im Fehler-/Diagnosespeicher der Steuerung abgespeichert.

I Eingang (schreib- und lesbar)

Abb.11-50:

TE_ErrorHandling Ausgangsdaten

Die Ausgangsdaten des "TE_MachineControl" sind in der Struktur **stErr_Handling_Status_gb** (Variablenliste "Global_Machine") vom Typ **TE_ERR_HANDLING_CONTROL** zusammengefasst.

Die folgende Tabelle beschreibt die Elemente der Ausgangsstruktur:

I/O-Typ	Name	Datentyp	Beschreibung
O	InOperation	BOOL	Bearbeitung läuft
O	Error	BOOL	Globaler Fehler steht an
O	ErrorID	ERROR_CODE	Detailinformation wenn DiagNo = 16#F0FB 0000/16#F0AA xxxx
O	ErrorIdent	ERROR_STRUCT	Detailinformation wenn DiagNo = 16#F0FB 0000/16#F0AA xxxx
O	DiagNo	DWORD	Nummer gemäß Dokumentation "Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnosen" mit den folgenden Erweiterungen: DiagNo = 16#F0AE xxxx: Applikations-Fehler (z. B. geöffnete Schutztür) DiagNo = 16#F0FB 0000: FB-Fehler DiagNo = 16#F0AA xxxx: GAT-interner Fehler DiagNo = 16#F0B1...B9 xxxx: Plugin Zusatzdiagnose Fehler (z. B. durch PROFIBUS Diagnose Plugin)
O	DiagMessage	STRING(60)	Diagnosetext der Steuerung/Achse

O Ausgang (lesbar)

Abb.11-51:

11.12.4 Daten der Units

Unit-Eingangsdaten

Jede Unit hat als Eingang eine Steuer-Datenstruktur (Ctrl) die vom Betriebsartenverwalter (TE_MachineControl) und dem Fehlerverwalter (TE_ErrorHandling) vorgegeben wird. Die Elemente ClearError, ExtError, und ExtErrDiagNo werden vom Fehlerverwalter (TE_ErrorHandling) vorgegeben. Alle anderen Elemente werden vom Betriebsartenverwalter (TE_MachineControl) vorgegeben.

GAT mit dezentraler Zustandsmaschine

Die Elemente der Eingangsdatenstruktur der Unit TE_UNIT_CONTROL sind in folgender Tabelle aufgeführt:

I/O-Typ	Name	Datentyp	Beschreibung
I	Enable	BOOL	Bearbeitungsfreigabe
I	ClearError	BOOL	Fehlerquittierung über positive Flanke (von TE_ErrorHandling)
I	OpMode	TE_OPMODES_PREDEFINED_TYPE02	Vorgegebene Betriebsart
I	DisableErrorReaction	BOOL	Deaktiviert die Fehlerreaktion
I	EnableErrorReactionInit	BOOL	Aktiviert die Fehlerreaktion während der Initialisierung
I	OpModeCoupled	BOOL	TRUE = gekoppelte Unit FALSE = ungekoppelte Unit
I	ExtError	BOOL	Externer Fehler (von TE_ErrorHandling)
I	ExtErrDiagNo	DWORD	Externe Fehlernummer (von TE_ErrorHandling)

I Eingang (schreib- und lesbar)

Abb.11-52:

Unit-Ausgangsdaten

Jede Unit hat als Ausgang eine Status-Datenstruktur (Status) die vom Betriebsartenverwalter (TE_MachineControl) und dem Fehlerverwalter (TE_ErrorHandling) ausgewertet wird. Alle Error...- und Diag...-Elemente werden vom Fehlerverwalter (TE_ErrorHandling) ausgewertet. Alle anderen Elemente werden vom Betriebsartenverwalter (TE_MachineControl) ausgewertet.

Die Elemente der Ausgangsdatenstruktur der Unit TE_UNIT_STATUS sind in folgender Tabelle aufgeführt:

I/O-Typ	Name	Datentyp	Beschreibung
O	InOperation	BOOL	Bearbeitung läuft
O	OpMode	TE_OPMODES_PREDEFINED_TYPE02	Quittierung der aktuellen Betriebsart
O	Error	BOOL	Lokaler Fehler
O	ErrorID	ERROR_CODE	Detailinformation wenn DiagNo = 16#F0FB 0000/16#F0AA xxxx
O	ErrorIdent	ERROR_STRUCT	Detailinformation wenn DiagNo = 16#F0FB 0000/16#F0AA xxxx

I/O-Typ	Name	Datentyp	Beschreibung
O	DiagNo	DWORD	Nummer gemäß Dokumentation "Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnosen" mit den folgenden Erweiterungen: DiagNo = 16#F0AE xxxx: Applikations-Fehler (z. B. geöffnete Schutztür) DiagNo = 16#F0FB 0000: FB-Fehler DiagNo = 16#F0AA xxxx: GAT-interner Fehler DiagNo = 16#F0B1...B9 xxxx: Plugin Zusatzdiagnose Fehler (z. B. durch PROFIBUS Diagnose Plugin)
O	DiagMessage	STRING(60)	Diagnosetext

O Ausgang (lesbar)

Abb.11-53:

11.12.5 Daten der Achsen

I/O-Typ	Name	Datentyp	Beschreibung
I	arAxisCtrl_gb (Global_Axis_Interfaces)	ARRAY OF TE_AXIS_CONTROL_TYPE01	Vorgaben für die Achsen
O	arAxisStatus_gb (Global_Axis_Interfaces)	ARRAY OF TE_AXIS_STATUS_TYPE01	Rückmeldungen der Achsen
I/O	AxisData (GVL_Base aus der Bibliothek ML_Base)	ARRAY OF ML_AXISDATA_SM	Firmwarestruktur mit Ein- und Ausgängen der Achsen

I Eingang (schreib- und lesbar)
O Ausgang (lesbar)
I/O Struktur mit Ein- und Ausgängen

☞ Elemente der arAxisCtrl_gb werden von den Units beschrieben, und können deshalb von außen nicht mehr genutzt werden.

11.12.6 Daten der Steuerung

I/O-Typ	Name	Datentyp	Beschreibung
I	ImcCtrl (Global_ImcInterface aus der Bibliothek ML_TechInterface)	MB_IMC_CONTROL_TYPE01	Vorgaben für die Steuerung
O	ImcStatus (Global_ImcInterface aus der Bibliothek ML_TechInterface	MB_IMC_STATUS_TYPE01	Rückmeldungen der Steuerung

I Eingang (schreib- und lesbar)
O Ausgang (lesbar)

☞ Elemente der ImcCtrl_gb werden im "TE_MachineControl" beschrieben, und können deshalb von außen nicht mehr genutzt werden.

GAT mit dezentraler Zustandsmaschine

11.13 HowTo: Typische Anwenderaktivitäten

11.13.1 Zugriff auf Achsdaten

Zum Zugriff auf Achsdaten ist in jeder Unit als zentraler Zugriffspunkt die Struktur **Axes** für alle der Unit zugeordneten Achsen definiert. Über diese Struktur sind alle relevanten Daten für eine Achse adressierbar.

Die Struktur ist so aufgebaut, das über den Namen der Achse auf die Achsdaten zugegriffen wird und damit gut lesbarer Programmcode entsteht.

Die folgenden Daten sind unterhalb von "Axes" verfügbar:

- Axes.<Name>.AxisData => Achsdatenstruktur: Statusdaten der Firmware und Zugriff auf Achsdaten-Container
- Axes.<Name>.Ctrl => Steuerstruktur des Achs-Interface
- Axes.<Name>.Status => Statusstruktur des Achs-Interface
- Axes.<Name>.DV => Zugriff auf Direktvariablen (Funktionsvariablen)
- Axes.<Name>.AxisRef => Achsreferenzstruktur für FB-Aufrufe (z. B. MB_ReadParameter)

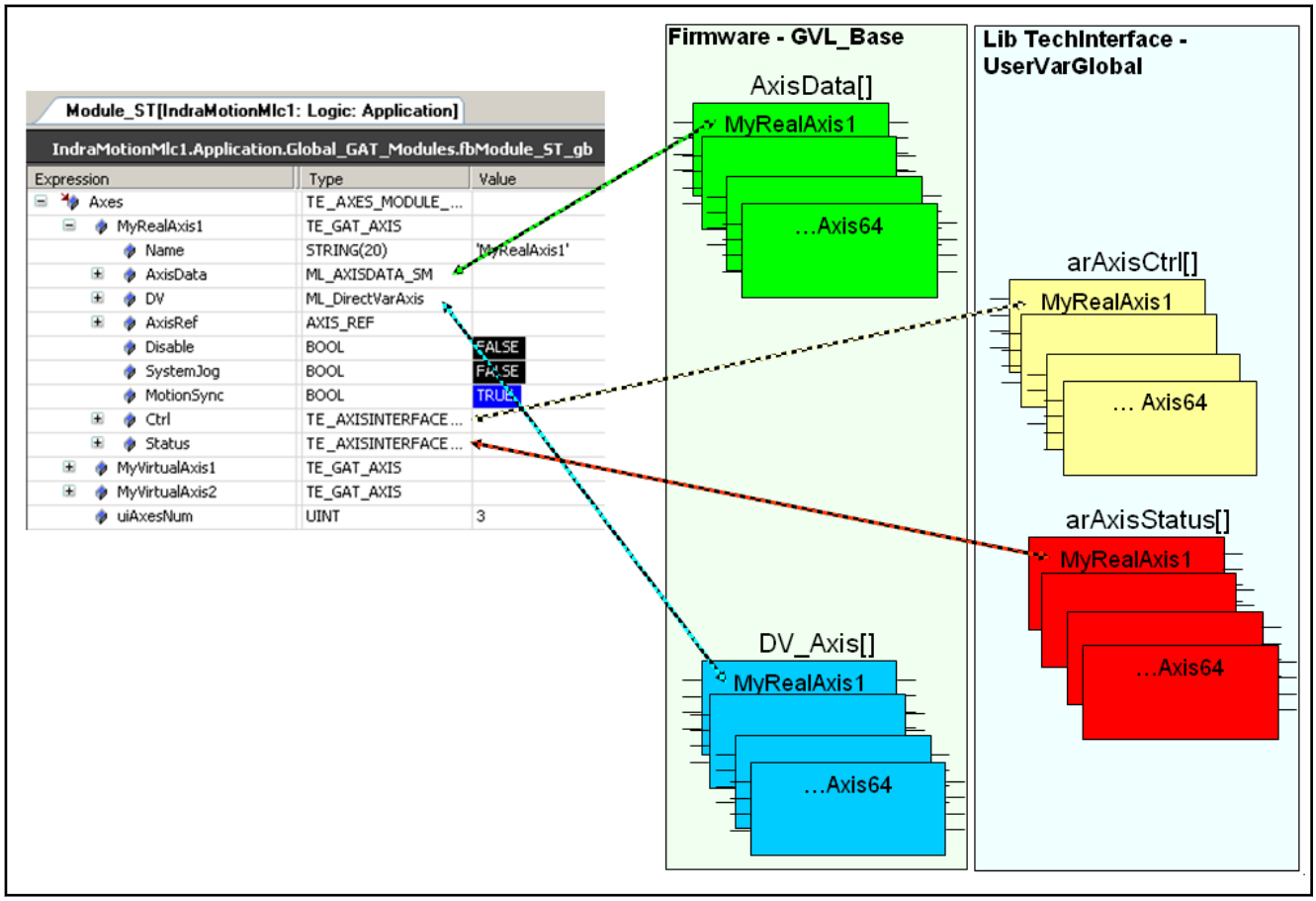


Abb.11-54: Übersicht Achsdatenzugriffe

11.13.2 Achse hinzufügen

Zum Hinzufügen einer Achse sind folgende Schritte durchzuführen:

- Einfügen der Achse in das IndraWorks-Projekt (auf den sercos Knoten ziehen…)

- Bei Nutzung der Visualisierung "SystemOverview_All_Axis": einen Frame für die neue Achse einfügen und in den Eigenschaften die Referenz auf die neue Achse setzen (siehe auch Kap. 8.8.1 "Interne Visualisierung zur Inbetriebnahmeunterstützung" auf Seite 232)
- Über den GAT-Wizard die Unitachse hinzufügen (siehe nachfolgende Abbildung)

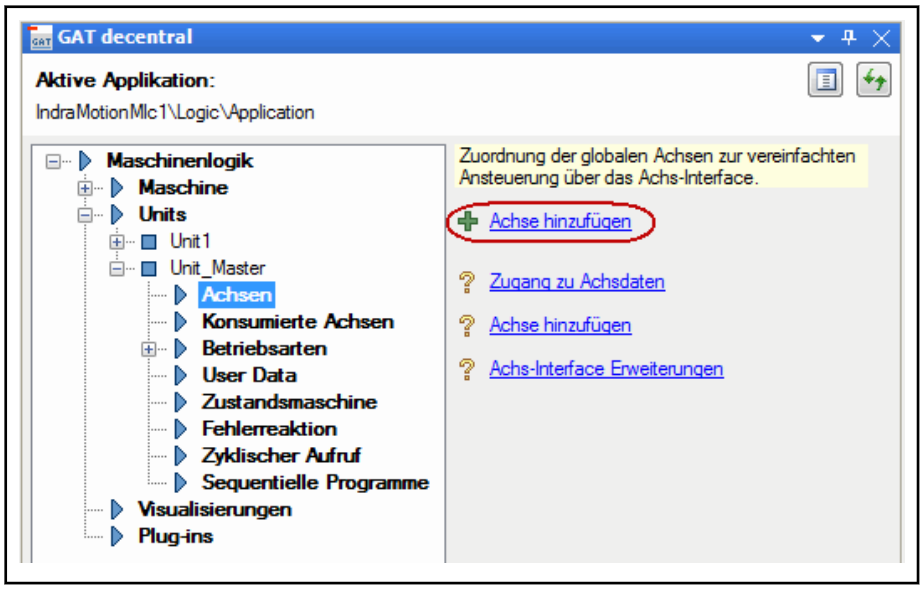


Abb.11-55: Achse über den GAT-Wizard hinzufügen

11.13.3 Achse entfernen/umbenennen

Der Wizard ändert nur die Namen in der Achs-Deklaration. Wird die Achse bereits in der Implementation bzw. internen Visualisierung verwendet, so muss der neue Achsname durch "Suchen/Ersetzen" an diesen Stellen angepasst werden.

Zum Entfernen/Umbenennen einer Achse sind folgende Schritte durchzuführen:

- Löschen/Umbenennen der Achse aus dem IndraWorks-Projekt
- Bei Nutzung der Visualisierung "SystemOverview_All_Axis": den Frame für die gelöschte Achse entfernen bzw. in den Eigenschaften die Referenz auf die Achse ändern
- Über den GAT-Wizard die Achse entfernen/umbenennen

11.13.4 Hauptbetriebsart hinzufügen

Der GAT-Wizard erlaubt das komfortable Hinzufügen von Hauptbetriebsarten der Maschine. Über den Wizard können die Betriebsarten auch gelöscht oder umbenannt werden.

GAT mit dezentraler Zustandsmaschine

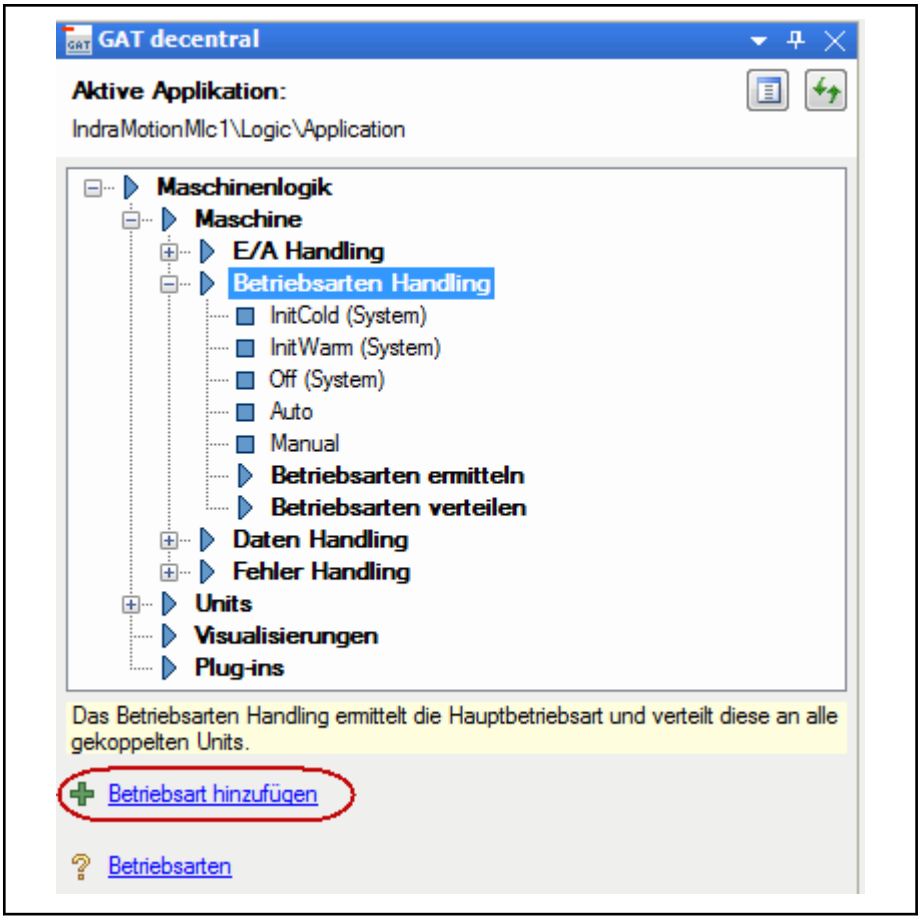


Abb.11-56: Betriebsart mit dem GAT-Wizard hinzufügen

11.13.5 Neue Unit hinzufügen

Der GAT-Wizard erlaubt das komfortable Hinzufügen von Units zur Maschine. Über den Wizard können die Units auch gelöscht oder umbenannt werden.

Für alle existierenden Hauptbetriebsarten werden in der neuen gekoppelten Unit korrespondierende Zweige im Betriebsarten-SFC angelegt.

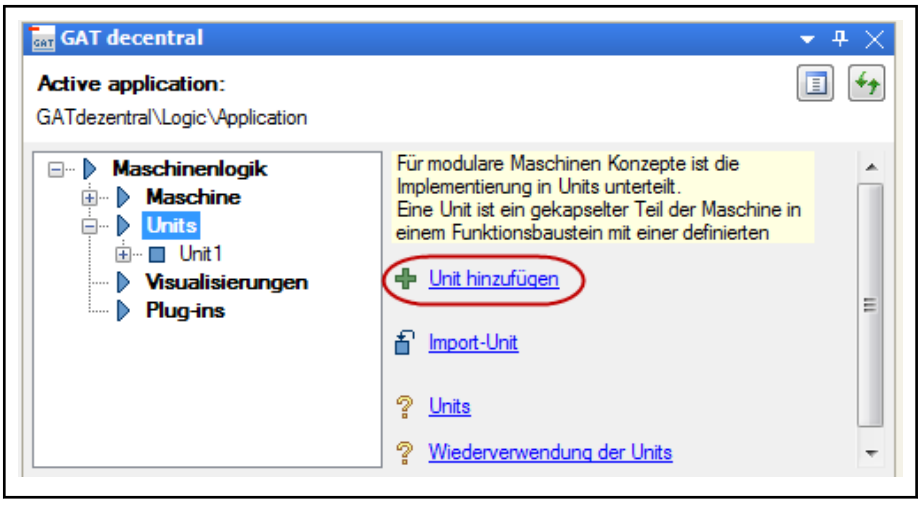


Abb.11-57: Unit mit dem GAT-Wizard hinzufügen

11.13.6 Weitere Instanz einer Unit hinzufügen

Der GAT-Wizard erlaubt das komfortable Hinzufügen weiterer Instanzen von existierenden Units zur Maschine:

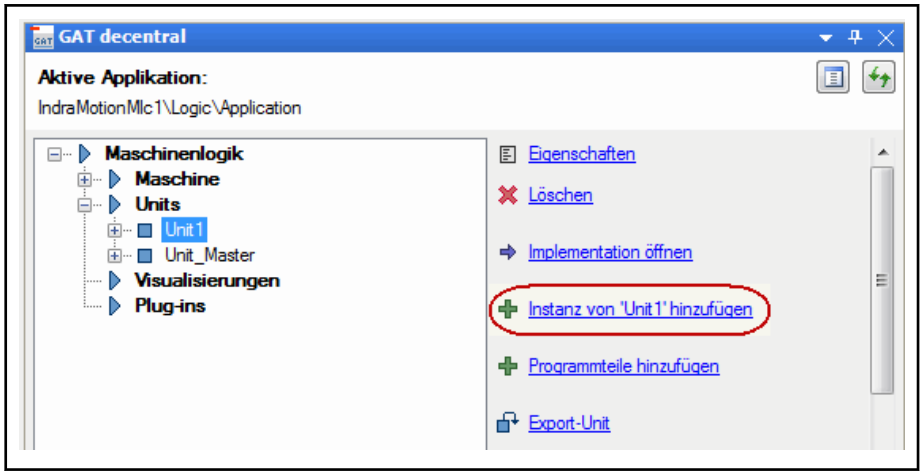


Abb.11-58: Weitere Instanz einer Unit mit dem GAT-Wizard hinzufügen

Die Schnittstellen bleiben identisch, es müssen jedoch andere globale Achsen (an "Axes") und I/O-Daten (an "UserData") zugewiesen werden.

11.13.7 Units deaktivieren

In der Aktion **actInitColdStart** des FB "TE_MachineControl" werden über die Anweisung "mEnableUnits(Enable := TRUE);" alle Units über den Enable-Eingang aktiviert (siehe nachfolgende Abbildung, Markierung 1):

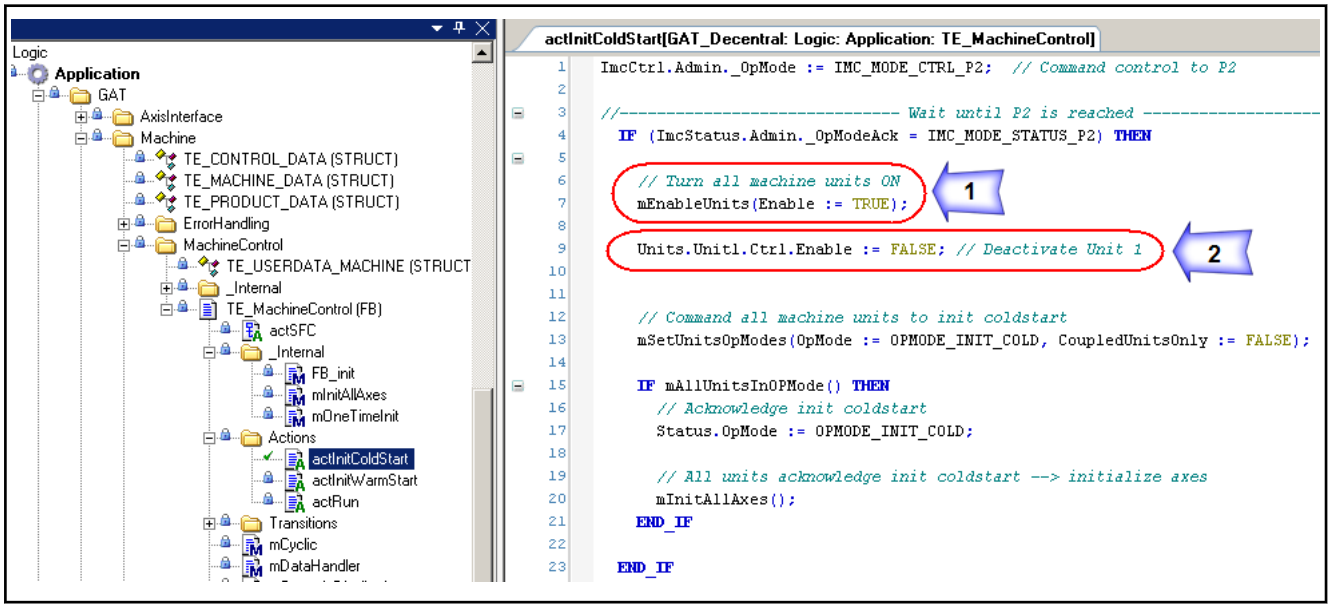


Abb.11-59: Beispiel "Unit1 wird deaktiviert"

Nach der o.g. Anweisung können einzelne Units über die Anweisung "Units.<UnitName>.Ctrl.Enable := FALSE" gezielt deaktiviert werden (siehe obige Abbildung, Markierung 2). Der Programmcode von deaktivierten Units ist von der Bearbeitung ausgeschlossen. Ein Deaktivieren der Units ist jederzeit möglich. Der Status-Ausgang "InOperation = FALSE" zeigt, dass diese Unit deaktiviert ist.

GAT mit dezentraler Zustandsmaschine

11.13.8 Wiederverwendung von Units

Das GAT ist so aufgebaut, dass entwickelte Units auch in anderen Projekten, die auf dem GAT aufbauen, wieder verwendet werden können.

Dazu ist der Ordner der gewünschten Units auszuwählen, und die dazugehörigen Daten unter Nutzung des Kontextmenüs zu exportieren (siehe nachfolgende Abbildung):

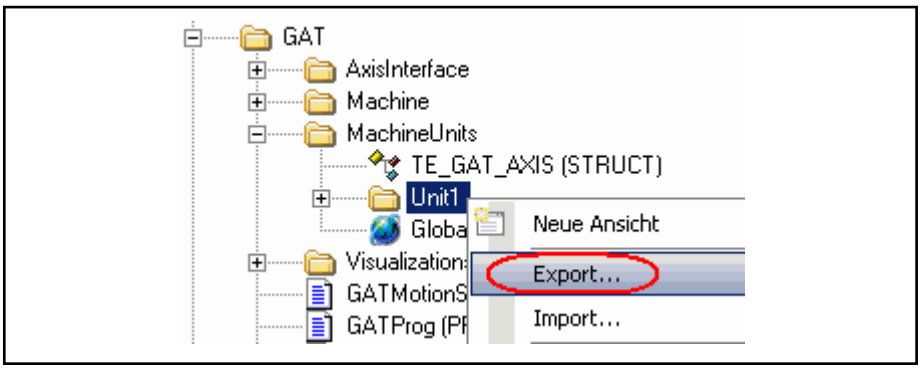


Abb.11-60: Unit exportieren

Im nachfolgenden Dialog sind alle Elemente für den Export auszuwählen.

Die beim Export erzeugte Datei ist im neuen Projekt zu importieren. Dazu ist der Ordner "MachineUnits" zu fokussieren und im dazugehörigen Kontextmenü die "Import"-Funktionalität zu nutzen (siehe nachfolgende Abbildung):

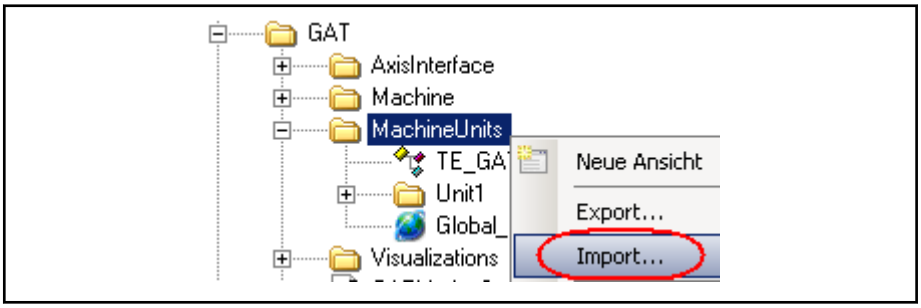


Abb.11-61: Unit importieren

In der Datenstruktur **TE_MACHINE_UNITS** muss die importierte Unit als Element aufgenommen und das Element **uiMaxUnitNum** um 1 erhöht werden.

In der nachfolgenden Abbildung ist exemplarisch ein Beispiel für eine Unit mit dem Namen "MyImportUnit" dargestellt:

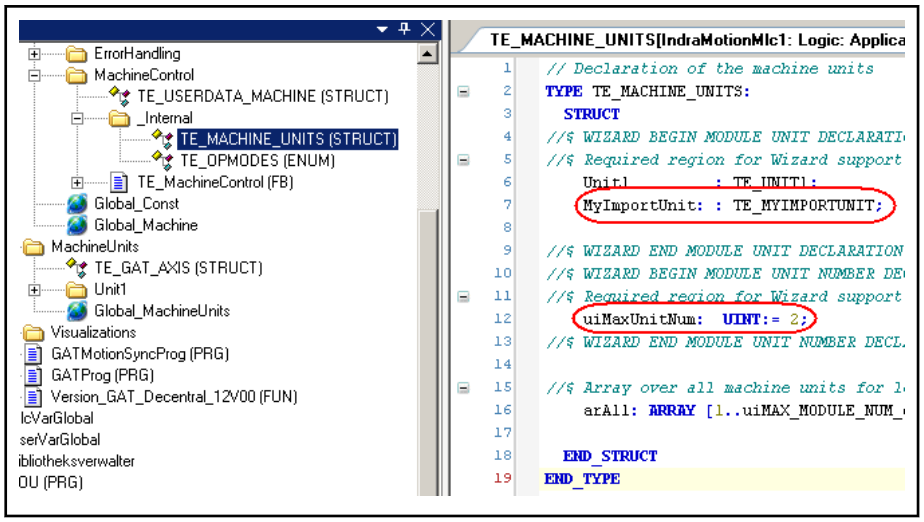


Abb.11-62: Exemplarische Anpassung der Datenstruktur "TE_MACHINE_UNITS" nach Unit-Import

Der Aufruf der importierten Unit-Instanz muss im Programm "GATProg" bzw. "GATMotionSyncProg" hinzugefügt werden.

In der Methode "mOneTimeInit" des FB TE_MachineControl sind die Unit-Referenzen für die neue Instanz hinzuzufügen. In diesem Bereich muss auch die Zuweisung für gekoppelte (Units.Unit1.Ctrl.OpModeCoupled := TRUE; // Coupled unit) bzw. ungekoppelte (Units.Unit1.Ctrl.OpModeCoupled := FALSE; // Uncoupled unit) Units eingetragen werden.

Weiterhin ist im ELSE-Zweig dieser Methode zusätzlich die Referenz für das Element der UserData-Struktur hinzuzufügen (Units.<UnitName>.UserData.stControlDataActive REF= stControlDataActive_gb;).

In der Methode "mOneTimeInit" des FB "TE_ErrorHandling" sind die Unit-Referenzen für die neue Instanz hinzuzufügen.

Im Fall einer ungekoppelten Unit muss weiterhin noch die Methode "mOpModeDistribution" mit dem Teil für die ungekoppelte Unit ergänzt werden.

Nach dem Import eine gekoppelten Unit ist zu prüfen ob alle Hauptbetriebsarten der Maschine auch im SFC der Unit enthalten sind. Ist dies nicht der Fall müssen die fehlenden Betriebsarten im SFC-Zweig hinzugefügt werden.

11.13.9 Verwendung von globalen und Adressvariablen in Units

Wegen der Mehrfachverwendung einer Unit innerhalb einer Applikation (Instanziierung) bzw. der Wiederverwendung in einer anderen Applikation darf in einer Unit kein direkter Zugriff auf eine globale Variable oder Adressvariable erfolgen.

Alle globalen und Adressvariablen, die in der Unit verwendet werden sollen, müssen als Interfacevariablen im Daten-Container **"UserData"** der Unit deklariert werden. Der Zugriff auf das Element innerhalb der Unit erfolgt durch **UserData.<Element>**.

GAT mit dezentraler Zustandsmaschine

11.13.10 Achs-Interface Erweiterungen

Das GAT erlaubt Erweiterungen der Achs-Interface-Strukturen. Es können zusätzliche Unterstrukturen eingefügt und die vorhandenen Unterstrukturen erweitert werden.

Die folgenden Strukturelemente sind als Anwendererweiterungen in der AxisCtrl-Struktur beispielhaft programmiert:

- SetupMode: zusätzliche Unterstruktur in der AxisCtrl-Struktur
  - Enable: Freigabe Einrichtbetrieb
  - JogPlus: Tippen +
  - JogMinus: Tippen –
  - Vel: Tipp-/Referenzfahrgeschwindigkeit
  - Accel: Tipp/Referenzfahr(brems)beschleunigung
  - Homing: aktiviert ModeHoming|SetAbsRef|ModesPosAbs abhängig vom Achstyp/-konfiguration
- SyncMode: Erweiterung der vorhandenen Unterstruktur in der AxisCtrl-Struktur
  - AngleOffset: Winkelverschiebung Tabellenanfang -> der Parameter A-0-2730 wird mit diesem Wert beschrieben
  - PhaseShift: Offset zur Leitachse vor dem Getriebe (PhaseOffset: Offset zur Leitachse nach dem Getriebe)
  - PhaseShiftVel: Geschwindigkeit zur Verstellung des PhaseShift
  - PhaseShiftAcc: Beschleunigung zur Verstellung des PhaseShift
  - SyncVelocity: Aufsynchronisiergeschwindigkeit -> der Parameter A-0-2790 wird mit diesem Wert beschrieben
  - SyncAcceleration: Aufsynchronisierbeschleunigung -> der Parameter A-0-2791 wird mit diesem Wert beschrieben

Die folgenden Strukturelemente sind als Anwendererweiterungen in der Axis-Status-Struktur beispielhaft programmiert:

- SetupMode: zusätzliche Unterstruktur in der AxisStatus-Struktur
  - EnableAck: Einrichtbetrieb ist aktiv
  - Homingack: Homing ist abgeschlossen
- SyncMode: Erweiterung der vorhandenen Unterstruktur in der AxisStatus-Struktur
  - AngleOffsetDone: Übertragung Winkelverschiebung Tabellenanfang ist abgeschlossen -> Parameter A-0-2730 wurde beschrieben
  - PhasingMasterDone: der PhaseShift wurde verfahren
  - SyncVelocityDone: Übertragung Aufsynchronisiergeschwindigkeit ist abgeschlossen -> Parameter A-0-2790 wurde übertragen
  - SyncAccelerationDone: Übertragung Aufsynchronisierbeschleunigung ist abgeschlossen -> Parameter A-0-2791 wurde übertragen

Der Code zu diesen Erweiterungen ist in den Aktionen des Bausteins "TE_AxisInterface()" im Ordner "GAT/AxisInterface" zu finden. Die dazugehörigen Strukturen sind in den Ordnern "GAT/AxisInterface/Type/Control" und "GAT/AxisInterface/Type/Status" zu finden.

Es können eigene Erweiterungen hinzugefügt werden (siehe dazu Kap. 6.7 "Achs-Interface Anwender-Erweiterung" auf Seite 103).

11.13.11 Robot Control

Um Achsen für koordinierte Bewegungen ansteuern zu können, muss für das TechInterface der Funktionsbaustein MB_AxisInterfaceType**02** verwendet werden. Hierzu muss folgendes im GAT^{compact} angepasst werden:

- Die Struktur "TE_AXIS_CONTROL_Type01" muss in "TE_AXIS_CONTROL_TYPE02" umbenannt werden und um die Strukturvariable "CoordModeCtrl" vom Typ "MB_AXIS_COORDINATED" erweitert werden

Umbenennen der Struktur TE_AXIS_CONTROL_Type01 nach TE_AXIS_CONTROL_Type02:

```
//former type: TYPE TE_AXIS_CONTROL_TYPE01:
TYPE TE_AXIS_CONTROL_TYPE02:  // rename for RobotControl
STRUCT
    Admin:      TE_AXIS_ADMINISTRATION; (* Administration of the axis | Verwaltung der Achse *)

    PosMode:    TE_AXIS_POSITIONING; (* operating mode positioning (absolute, rel. and additive)
      Positionierbetrieb (absolut, relativ und additiv) *)

    VelMode:    TE_AXIS_VELOCITY; (* operating mode velocity control | Geschwindigkeitsregelung *)

    SyncMode:   TE_AXIS_SYNCHRONISATION; (* operating mode synchronisation (phase, vel, and cam)
      Synchronbetriebsarten (Winkelsync., Geschwindigkeitssync. und Kurvenscheibe) *)

    (* add your extensions below here, keep the elements above
       Erweiterungen hier einfuegen, Elemente oberhalb nicht veraendern *)

    (* user extensions | Anwender Erweiterungen *)

      // Jog/Homing functionality | Tipp/Einrichtfunktionen
    SetupMode:  TE_AXIS_SETUP_MODE;

    CoordModeCtrl:  MB_AXIS_COORDINATED;    // inserted for RobotControl

END_STRUCT
END_TYPE
```

- Umbenennen der "TE_AXIS_STATUS_Type01" Struktur in "TE_AXIS_STATUS_TYPE02"

Umbenennen der Struktur TE_AXIS_STATUS_Type01 nach TE_AXIS_STATUS_TYPE02:

```
    (********* Outputs of the following axis - from motion control to logic control **********)
    // former type: TYPE TE_AXIS_STATUS_TYPE01:
TYPE TE_AXIS_STATUS_TYPE02:     // Rename for RobotControl
  STRUCT
    (********* Outputs of the following axis - from motion control to logic control **********)

    PtrAxisData:  POINTER TO ML_AXISDATA_SM;
     (* Pointer to AxisData of this axis | Zeiger auf AxisData dieser Achse
      - Type .PtrAxisData^. for the intellisense menu | Eingabe .PtrAxisData^. fuer Auswahlmenue*)

    Admin:   TE_AXIS_ADMIN_STATUS; (* status of the axis | Achs-Status *)

    Diag:    TE_AXIS_DIAGNOSIS; (* axis diagnosis information, error, warning....
      Diagnose Informationen der Achse, Fehler, Warnung.... *)

    PosMode: TE_AXIS_POS_STATUS; (* Status operating mode positioning (absolute, rel. and add.)
      Statusmeldungen Positionierbetrieb (absolut, relativ und additiv) *)

    VelMode: TE_AXIS_VEL_STATUS; (* Status operating mode velocity control
      Statusmeldungen Geschwindigkeitsregelung (Freilauf) *)

    SyncMode: TE_AXIS_SYNC_STATUS; (* Status operating mode synchronisation (phase, vel, and cam)
      Statusmeldungen Synchronbetriebsarten (Winkelsync., Geschwindigkeitssync. und Kurvensch.)*)

    (* add your extensions below here, keep the elements above
       Erweiterungen hier einfuegen, Elemente oberhalb nicht veraendern *)
    (* user extensions | Anwender Erweiterungen *)
          (* Jog/Homing functionality | Tipp/Einrichtfunktionen *)
    SetupMode:  TE_AXIS_SETUP_MODE_STATUS;
```

GAT mit dezentraler Zustandsmaschine

```
    // AxisData for HMI (without data type "BIT")
      AxisData_HMI:    TE_AXIS_DATA_STATUS;

  END_STRUCT
END_TYPE
```

- Der Funktionsbaustein "TE_Axisinterface" muss mit dem neuen FB "MB_AxisInterfaceType02" erweitert werden. "TE_AxisInterface EXTENDS MB_AxisInterfaceType02"

Erweiterung TE_Axisinterface:

```
//former: FUNCTION_BLOCK TE_AxisInterface EXTENDS MB_AxisInterfaceType01
FUNCTION_BLOCK TE_AxisInterface EXTENDS MB_AxisInterfaceType02
(*#########################################################################################################
  #########################################################################################################

  General Header
  -------------------------------------------------------------------------------------------
  Shortdescription       : AxisInterface Example
```

- Die Methode "mInitExtensions()": Der zusätzliche Eingang "CoordModeCtrl" des FB "MB_AxisInterfaceType02" muss initialisiert werden

CoordModeCtrl initialisieren:

```
SUPER^.mInitExtension(index);

IF NOT bFirstInit THEN
     // User: Assigment of user defined axis interface inputs
     SetupMode      REF= arAxisCtrl_gb[index].SetupMode;
     // inserted for RobotControl
     CoordModeCtrl  REF= arAxisCtrl_gb[index].CoordModeCtrl;
     // User: Assigment of user defined axis interface outputs
     SetupModeAck REF= arAxisStatus_gb[index].SetupMode;
     AxisData_HMI REF= arAxisStatus_gb[index].AxisData_HMI;

     // ================================================================
     // Do not touch the following lines
     // ================================================================

     // Set the references for the AxisInterface FB
     AdminCtrlExt  REF= arAxisCtrl_gb[index].Admin;
```

- Die Deklaration für die beiden Strukturvariablen "arAxisCtrl_gb" und "arAxisStatus_gb" muss auf die entsprechenden Type02-Strukturen geändert werden

Strukturvariablen anpassen:

```
VAR_GLOBAL
   // Axis interface: command structure: define as RETAIN, to keep the values
   // Note: on the L40.2 control, the RETAIN memory is slower then normal VAR, so define
   //       this structure only as RETAIN when necessary
   //  arAxisCtrl_gb: ARRAY [1..MAX_AXIS_NUM] OF MB_AXIS_CONTROL;

   // Axis interface: command structure
   //former: arAxisCtrl_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_CONTROL_TYPE01;
     arAxisCtrl_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_CONTROL_TYPE02;
END_VAR

VAR_GLOBAL
//---------------------------------- Machine status variables ----------------------------

  // Axis interface: status structure
  //former: arAxisStatus_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_STATUS_TYPE01;
  arAxisStatus_gb: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_STATUS_TYPE02;
  bRemoteOn_gb  : BOOL := TRUE;  // TRUE: Internal IL visu is active
                                 // FALSE: PlcProg is active
END_VAR
```



Die Strukturen TE_AXIS_CONTROL_TYPE02, TE_AXIS_STATUS_TYPE02 befinden sich im Ordner: "AxisInterface\Type".

Die Strukturvariablen arAxisCtrl_gb, arAxisStatus_gb befinden sich in der globalen Variablenliste "UserVarGlobal".

Kinematik-Interface

Aufbauend auf das Achs-Interface steht das Kinematik-Interface als optionales Plug-in zur Verfügung. Über das Kinematik Interface können Kinematiken analog zu Achsen kommandiert werden. Fügen Sie hierzu das Kinematik Interface über den GAT Wizard als Plug-in hinzu.

Die Dokumentation des Kinematik Interface ist in dem Plug-in Konfigurationsordner enthalten.

Den Link zum Konfigurationsordner können Sie über den GAT Wizard wie folgt finden:

- Im GAT Wizard den Ordner "Plug-ins" selektieren
- Im Wizard Bedienfeld ist jetzt der Link "Plug-ins Konfigurationsordner öffnen" sichtbar

11.13.12 Generierung eines eigenen Programmrahmens auf Basis des GAT

Eine auf Basis des GAT entwickelte Applikation kann wiederum als Vorlage (Programmrahmen) zur Entwicklung einer anderen, komplexeren Applikation dienen. Auf diese Weise kann ein Anwender z. B. für unterschiedliche Maschinentypen jeweils einen eigenen Programmrahmen zur Verfügung stellen, der sich aber von dem Standard GAT durch umfangreiche Weiterentwicklungen unterscheidet (z. B. zusätzliche Betriebsarten und Units). Zur Arbeit am weiterentwickelten Programmrahmen kann dann wieder der GAT-Wizard genutzt werden.

Zur Generierung einer eigenen Vorlage auf Basis einer Applikation, die mit dem GAT entwickelt wurde, muss folgendermaßen vorgegangen werden:

1. Gesamten GAT-Ordner exportieren, alle Checkboxen angewählt lassen (*.iwx Datei erzeugen). Im Export Dialog ist ein Eingabefeld "Beschreibung" vorhanden. Der hier eingetragene Text kann bei der Import-Funktion des GAT-Wizards angezeigt werden.

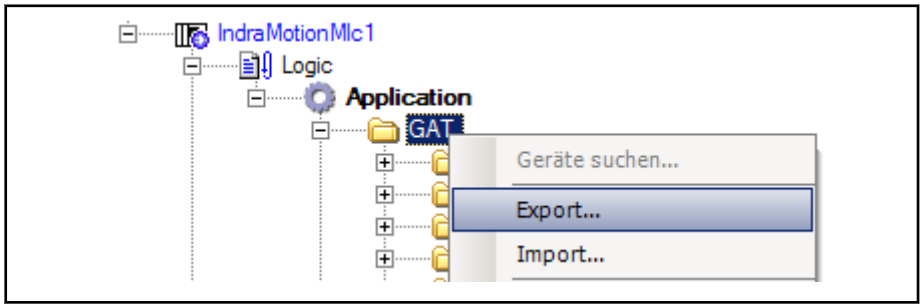


Abb. 11-63: GAT-Ordner exportieren

2. Die im Schritt 1 erzeugte *.iwx Datei beinhaltet das angepasste/ergänzte Template und kann nun beim Neuanlegen von Projekten über den GAT-Wizard wie folgt genutzt werden:

GAT mit dezentraler Zustandsmaschine

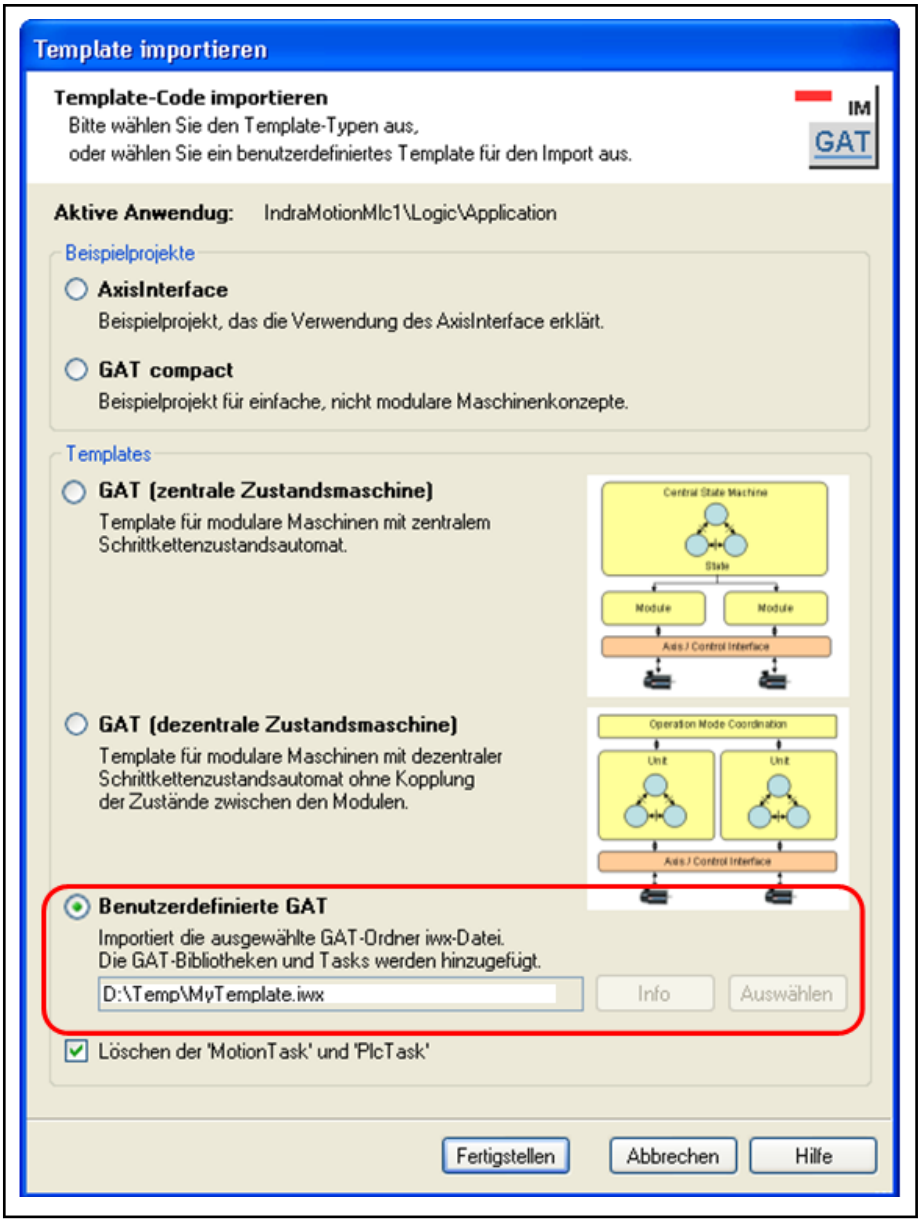


Abb.11-64: Benutzerdefiniertes Template importierten

11.14 GAT – dezentrale Zustandsmaschine Kurzübersicht

11.14.1 Strukturübersicht

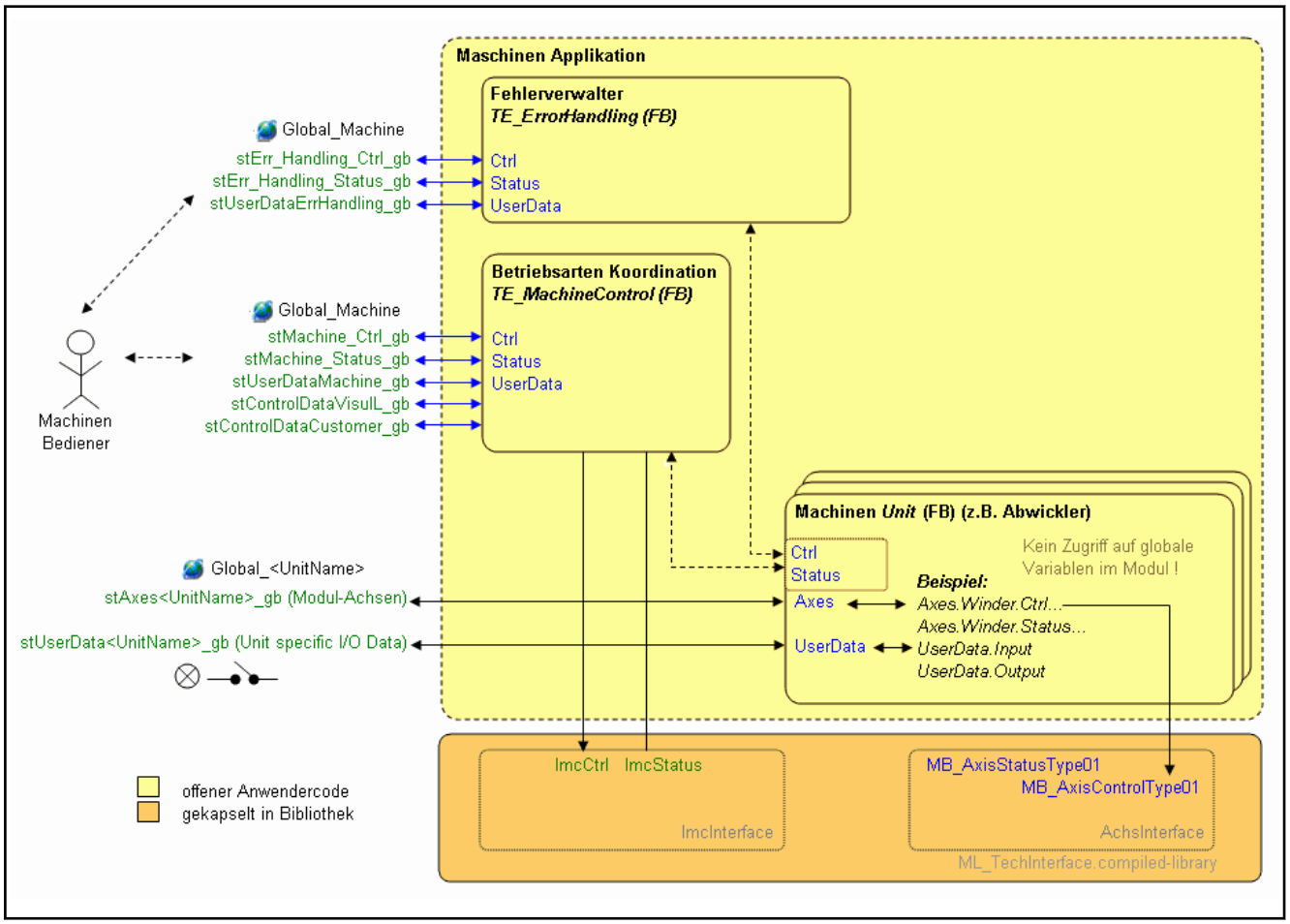


Abb.11-65: Strukturübersicht GAT mit dezentraler Zustandsmaschine

11.14.2 Übersicht der wichtigsten globalen Variablen der Maschine und der Fehlerverwaltung

Die nachfolgende Tabelle zeigt die wichtigsten globalen Variablen der Maschine und der Fehlerbehandlung.

Die Variablen sind in der globale Variablenliste **„GAT/Machine/Global_Machine"** enthalten.

Variable	Beschreibung	Zugriff
stControlDataVisuIL_gb	Zentrale Maschinensteuerung (z. B. Betriebsartenanwahl), Vorgabe durch integrierte Inbetriebnahme Visualisierung	RD/WR
stControlDataCustomer_gb	Zentrale Maschinensteuerung (z. B. Betriebsartenanwahl), Vorgabe durch externe Visualisierung	RD/WR
stControlDataActive_gb	Wirksame Daten der zentralen Maschinensteuerung (z. B. Betriebsartenanwahl)	RD/WR
stStatusData_gb	Statusdaten der Machine für HMI	RD/WR

GAT mit dezentraler Zustandsmaschine

Variable	Beschreibung	Zugriff
stExchangeData_gb	Enthält Daten zum Datenaustausch zwischen den Units	RD/WR
stMachineDataEdit_gb	Vorbereitete maschinenspezifische Daten	RD/WR
stMachineDataActive_gb	Wirksame maschinenspezifische Daten	RD
stProductDataEdit_gb	Vorbereitete produktspezifische Daten	RD/WR
stProductDataActive_gb_gb	Wirksame produktspezifische Daten	RD
stMachine1Status_gb	Status der Maschine (z. B. Ausgabe der aktuellen Betriebsart)	RD
stUserDataMachine_gb	Anwenderspezifische I/O-Daten der Maschine	RD/WR
arAppErrMachineControl_gb	Anwenderspezifische Fehlerflags der Maschine	RD
stErr_Handling_Ctrl_gb	Zentrale Fehlerverwaltung Steuerstruktur (z. B. für globales Fehlerlöschen)	RD/WR
stErr_Handling_Status_gb	Zentrale Fehlerverwaltung Statusstruktur (z. B. Ausgabe des zentralen Fehlerbits)	RD
stUserDataErrHandling_gb	Anwenderspezifische I/O-Daten der Fehlerbehandlung	RD/WR

RD Lesen
WR Schreiben

11.14.3 Übersicht der wichtigsten globalen Variablen der Units

Die nachfolgende Tabelle zeigt die wichtigsten globalen Variablen der Units.

Die Variablen sind in der globale Variablenliste **„GAT/MachineUnits/ Global_<Unitname>”** enthalten.

Variable	Beschreibung	Zugriff
fb<Unitname>_gb.Ctrl	Unit-Steuerstruktur (z. B. Betriebsarten-Vorgabe)	RD/WR
fb<Unitname>_gb.Status	Unit-Statusstruktur (z. B. Betriebsarten-Quittierung)	RD
stUserData<Unitname>_gb	Unitspezifische I/O	RD/WR
arAppErr<Unitname>_gb	Array mit anwenderspezifischen Fehlerflags der Unit	RD
stAxes<Unitname>_gb	Container mit Unitachsen	RD

RD Lesen
WR Schreiben

11.14.4 Übersicht der wichtigsten globalen Variablen der Achsen

Die nachfolgende Tabelle zeigt die wichtigsten globalen Variablen der Achsen.

Die Variablen sind in der globale Variablenliste **„GAT/ AxisInterface/ Global_GAT_AxisInterfaces”** enthalten.

Variable	Beschreibung	Zugriff
arAxisCtrl_gb	Array mit den Achs-Interface Kommando-Daten	RD/WR
arAxisStatus_gb	Array mit den Achs-Interface Status-Daten	RD

RD Lesen
WR Schreiben

11.14.5 Übersicht der Methoden und Aktionen des TE_MachineControl

Die nachfolgende Tabelle zeigt die wichtigesten Methoden und Aktionen des FB "TE_MachineControl".

Aktion/Methode	Aufruf	Beschreibung/Beispiel
actOPModeSFC	permanent zyklisch	Betriebsarten-SFC
mCyclic	permanent zyklisch	Unitübergreifende Logik
mDataHandler	permanent zyklisch	Datenhandling/Umschaltung zwischen integrierter Inbetriebnahme- und externer Visualisierung, Maschinen- und Produktdatenverwaltung
mOpModeDistrubution	permanent zyklisch	Betriebsartenverteilung zu den Units
mOpModeSet	permanent zyklisch	Ermittlung der Hauptbetriebsart
mOneTimeInit	Bei Initialisierung des GAT für einen Zyklus. Ausführung vor "ColdStart"	Einstellungen für "SkipColdstartAtStartup" initialisieren Zuordnung Unitachsen zu den globalen IndraWorks-Achsen Referenzen initialisieren
mInitAllAxes	Bei Verlassen von" InitCold-Start" und im "InitWarmStart"	Achs-Interface initialisieren
mErrorHandling	permanent zyklisch	Lokale Fehlerquellen abscannen, lokales Fehler löschen
FB_init	Einmalig vor SPS-Programmstart/Reset	Systemintern, darf nicht modifiziert werden

11.14.6 Übersicht der Methoden und Aktionen des TE_ErrorHandling

Die nachfolgende Tabelle zeigt die wichtigsten Methoden und Aktionen des FB "TE_ErrorHandling".

Aktion/Methode	Aufruf	Beschreibung/Beispiel
mScanError	permanent zyklisch	Fehlerquellen abscannen
mDistributeError	permanent zyklisch	Fehlerverteilung auf Units
mClearError	permanent zyklisch	Zentrales Fehlerlöschen
mOneTimeInit	Bei Initialisierung des GATs für einen Zyklus. Ausführung vor ColdStart	Einstellungen für "SkipColdstartAtStartup" initialisieren Zuordnung Unitachsen zu den globalen IndraWorks-Achsen Referenzen initialisieren
FB_init	Einmalig vor SPS-Programmstart/Reset	Systemintern, darf nicht modifiziert werden

GAT mit dezentraler Zustandsmaschine

11.14.7 Übersicht der Methoden und Aktionen der Units

Die nachfolgende Tabelle zeigt die wichtigesten Methoden und Aktionen der Units.

Aktion/Methode	Aufruf	Beschreibung/Beispiel
actOPModeSFC	permanent zyklisch	Betriebsarten-SFC
mErrorHandling	permanent zyklisch	Lokale Fehlerquellen abscannen, lokales Fehler löschen
mRefreshAxes	permanent zyklisch	Achs-Interface Daten aktualisieren
mOpModeAcknowledge	permanent zyklisch	Betriebsartenquittierung
FB_init	Einmalig vor SPS-Programmstart/Reset	Systemintern, darf nicht modifiziert werden

11.14.8 Zugriff auf Achsen innerhalb der Unit

☞ Zugriff immer über **Axes.<Achsname>.**

Beispiel:

```
 // Achse in Geschwindigkeitsregelung kommandieren
Axes.MyAxis1.Ctrl.Admin._OpMode.en := ModeVel;

// Wenn Achse die Geschwindigkeit erreicht hat
IF (Axes.MyAxis1.Status.PtrAxisData^.Axis_InVelocity = TRUE) THEN;
```

11.14.9 Zugriff auf I/O-Daten innerhalb der Unit

☞ Zugriff immer über **UserData.<Variable>**. Nicht direkt auf globale Variablen zugreifen!

Beispiel:

```
IF (UserData.bInput1=TRUE) THEN // Wenn „bInput1“ gesetzt ist…
  UserData.bOutput1:=TRUE; // Ausgang „bOutput1“ setzen
END_IF
```

12 GAT Plug-ins

12.1 Einleitung

GAT Plug-ins sind optional nachladbare Funktionserweiterungen für alle GAT-Varianten. GAT Plug-Ins ermöglichen den Programmrahmen auf die wichtigsten Funktionen zu reduzieren und optionale Funktionen nur bei Bedarf nachzuladen.

Das GAT Plug-in Konzept ist durch folgende Merkmale gekennzeichnet:

- **Einfach**: Plug-in Einbindung durch GAT-Wizard Dialoge ohne Programmieraufwand
- **Flexibel**: Durch das "open source" Konzept ist der Code modifizierbar
- **Erweiterbar**: Plug-ins können von Anwendern selbst erstellt, ausgetauscht und wiederverwendet werden

Ein Plug-In besteht aus folgenden Teilen:

- **SPS Objekte**, wie z. B. Programme, Funktionsbausteine, Datenstrukturen, Visualisierungen
- **Konfigurationsdaten**, um die Aufrufstelle und den Aufrufcode festzulegen
- **Anwenderdokumentation**

12.2 Plug-in hinzufügen

Sie können ein Plug-in mit folgenden Schritten in ein GAT-Projekt einfügen:

- "Plug-ins" Knoten im GAT-Wizard selektieren
- Funktion "Plug-in Hinzufügen" wählen
- Plug-in auswählen, und mit **Übernehmen** hinzufügen

12.3 Plug-in entfernen

Mit folgenden Schritten können Sie ein Plug-in entfernen:

- "Plug-ins" Knoten im GAT-Wizard expandieren
- Plug-in selektieren
- "Plug-in entfernen" auswählen

12.4 Plug-in installieren

Plug-ins können jederzeit auch nachträglich auf dem Engineering PC installiert werden.

Mit den nachfolgenden Schritte wird ein Plug-in installiert, und damit über den GAT-Wizard auswählbar:

- GAT -Wizard öffnen
- Knoten "Plug-ins" selektieren
- Link "Plug-ins Konfigurationsverzeichnis öffnen" anklicken
- Neuer Plug-in Ordner unterhalb des geöffneten Konfigurationsverzeichnis hinzufügen. Im Plug-in Ordner müssen die nachfolgenden Dateien enthalten sein.

Erforderliche Plug-in Dateien

Beschreibung	Dateiname
IndraWorks Export Datei mit allen notwendigen SPS Objekten des Plug-in	<Plug-inName>.iwx
Konfigurationsdatei	<Plug-inName>Config.txt
Anwenderdokumentation	<Plug-inName>.pdf oder <Plug-inName>.txt

Abb.12-1: Erforderliche Plug-in Dateien

12.5 Plug-in selbst erstellen

Mit folgender Vorgehensweise können Plug-ins auch von Anwendern selbst erstellt werden:

Exportdatei erzeugen

Die Exportdatei enthält alle SPS Objekte (z. B. Programme, Funktionsbausteine, Strukturen, Visualisierungen) des Plug-in.

Mit folgenden Schritten wird die Exportdatei erzeugt:

- Neuer Plug-in Ordner (Name: <Plug-inName>) in IndraWorks anlegen
- Alle SPS Objekte des Plug-in in den Plug-in Ordner verschieben
- Plug-in Ordner mit IndraWorks exportieren (Dateiname: <Plug-inName>.iwx)

Konfigurationsdatei erzeugen

Der GAT Wizard benötigt Plug-in spezifische Informationen, wie z. B. den Plug-in Aufrufcode, Aufrufstelle, Kurzbeschreibung.

Die Konfiguration besteht aus folgenden Teilbereichen (Regionen):

- Plug-in Kurzbeschreibung (gelbes Feld im GAT-Wizard)
- Link zur Plug-in Dokumentation
- Plug-in Aufrufstelle und Aufrufcode
- Zusätzliche Bibliotheken
- Zusätzliche SPS Tasks

Die einzelnen Regionen sind im folgenden beschrieben.

Kurzbeschreibung

Die Kurzbeschreibung dient zur ersten Vorab-Information des Benutzers. Beschreiben Sie hier in wenigen Worten die Funktionalität des Plug-in. Der Text erscheint in einem Textfeld des GAT Wizard, wenn das Plug-in ausgewählt wird.

Die Region ist durch folgende Textmarken gekennzeichnet:

- "WIZARD BEGIN PLUGIN SHORT DESCRIPTION" definiert den Anfang der Region
- "WIZARD END PLUGIN SHORT DESCRIPTION" definiert das Ende der Region

Kurzbeschreibung

```
//$ WIZARD BEGIN PLUGIN SHORT DESCRIPTION -----------
My short description...
//$ WIZARD END PLUGIN SHORT DESCRIPTION -------------
```

Link zu Dokumentation

Der Wizard öffnen die in dieser Region angegebene Dokumentations-Datei, wenn über den GAT Wizard die Plug-in Dokumentation aufgerufen wird. Die Dokumentation kann in allen gängigen Formaten erstellt werden (z. B. *.pdf, *.txt, etc.)

Die Region ist durch folgende Textmarken gekennzeichnet:

- "WIZARD BEGIN PLUGIN DOCUMENTATION LINK" definiert den Anfang der Region
- "WIZARD END PLUGIN DOCUMENTATION LINK" definiert das Ende der Region

Es bestehen hierbei folgende Möglichkeiten:

- Dateiname eintragen (z. B. MyPlugin.pdf), dann wird die Datei im lokalen Plug-in Verzeichnis aufgerufen
- Vollständiger Link eintragen, dann wird die über den Link verfügbare Datei aufgerufen. Der Link muss entweder mit "http:\\" oder "\\" beginnen.

Link zu Dokumentation

```
//$ WIZARD BEGIN PLUGIN DOCUMENTATION LINK ----------------------
//$ File in the PLUGINfolder or a hyperlink (starting with 'http')
MyPlugInDocu.pdf
//$ WIZARD END PLUGIN DOCUMENTATION LINK ------------------------
```

Zur Erstellung der Datei können beliebige Textbearbeitungsprogramme oder Editoren verwendet werden.

Vorzugsweise wird die Dokumentation in englisch mit folgendem Aufbau erstellt:

- Tabelle mit Autor, Datum, Version, Änderungsverfolgung
- Beschreibung
- Schnittstellenbeschreibung
- Abhängigkeiten / notwendige Bibliotheken

Aufrufcode und Aufrufstelle

In der hier beschriebenen Region wird die Aufrufstelle und der Aufrufcode vorgegeben.

Ein Plug-in kann an mehreren Aufrufstellen jeweils unterschiedlichen Aufrufcode hinzufügen. Daher können beliebig viele dieser Regionen in der Konfigurationsdatei enthalten sein.

Die Region ist durch folgende Textmarken gekennzeichnet:

- "WIZARD BEGIN PLUGIN CALL UP" definiert den Anfang der Region
- "WIZARD END PLUGIN CALL UP" definiert das Ende der Region

Aufrufcode und Aufrufstelle

```
//$ WIZARD BEGIN PLUGIN CALL UP ---------------------------------------
//$ BEGIN PLUGIN PATH TO POU ------------------------------------
//$ path to the ST pou where the PLUGIN call has to be inserted
//$ controls the availability of the PLUGIN in the GAT wizard
Enter the path to the POU where the PLUGIN call has to be inserted ...
//$ END PLUGIN PATH TO POU --------------------------------------

//$ BEGIN PLUGIN CALL -------------------------------------------
//$ Code region to be inserted in the configured structured text pou
//$ This code is used if there is no other special code for a specific
//$ GAT type is available
Enter code to call and connect the PLUGIN ...
//$ END PLUGIN CALL ---------------------------------------------
//$ WIZARD END PLUGIN CALL UP -----------------------------------------
```

Die Region ist in zwei Teile gegliedert.

Der erste Teil "PLUGIN PATH TO POU" beschreibt an welcher Stelle der Aufrufcode hinzugefügt wird. Hierbei wird folgende Syntaxregel verwendet:

GAT Plug-ins

Programm:

```
/Gat-Typ = Path to code position | InsertLocation=Implementation/Declaration | InsertPosition=End/Top |
InsertMark="Markierungstext"
```

Folgende GAT-Typen werden hierbei unterstützt:

- GatCentral
- GatDecentral
- GatCompact
- AxisInterface



Durch den Syntax #AllUnits# und #AllModules# in dem "Path to code position" kann der Aufrufcode in alle Units bzw. alle Module des GAT hinzugefügt werden.

Beispiele:

- GatCentral = #AllModules#\actCyclic: Aufrufcode wird in die Aktion "actCyclic" aller Module hinzugefügt
- GatCentral = #AllModules#: Aufrufcode wird in allen Modul Funktionsbausteinen hinzugefügt
- GatDecentral = #AllUnits#\Actions\actOff: Aufrufcode wird in die Aktion "actOff" aller Units hinzugefügt
- GatDecentral = #AllUnits#: Aufrufcode wird in allen Unit Funktionsbausteinen hinzugefügt

Die weiteren nachfolgenden Elemente steuern wo der Aufrufcode hinzugefügt wird:

- "InsertLocation" beschreibt ob der Aufrufcode in dem Implementations- oder Deklarationsteil des SPS Objektes eingefügt wird
- "InsertPosition" beschreibt ob der Aufrufcode am Ende (End) oder Anfang (Top) eingefügt wird
- "InsertMark" ist optional und beschreibt eine Textmarke. Vor- (InsertPosition=Top) bzw. nach (InsertPosition=End) dieser Textmarke wird der Aufrufcode hinzugefügt.

Der zweite Teil "BEGIN PLUGIN CALL" beschreibt den Aufrufcode.

Hierbei gelten folgende Regeln:

- Der allgemeine Aufrufcode wird hinzugefügt, wenn kein GAT-Typ-spezifischer Aufrufcode enthalten ist
- Der allgemeine Aufrufcode ist durch den Syntax "BEGIN PLUGIN CALL" gekennzeichnet
- Der GAT Typ-spezifische Aufrufcode wird hinzugefügt, wenn er in der Konfigurationsdatei enthalten ist
- *Der GAT Typ-spezifische Aufrufcode ist durch folgenden Syntax gekennzeichnet:*
  - "BEGIN PLUGIN CALL GATCENTRAL" für GATcentral
  - "BEGIN PLUGIN CALL GATDECENTRAL" für GATdecentral
  - "BEGIN PLUGIN CALL GATCOMPACT" für GATcompact Beispielprojekt
  - "BEGIN PLUGIN CALL AXISINTERFACE" für AxisInterface Beispielprojekt

Folgendes Beispiel fügt am Ende der Implementierung der Methode mCyclic des FBs Machine den Aufrufcode "Test" hinzu:

```
//$ WIZARD BEGIN PLUGIN CALL UP ------------------------------------------------
//$ BEGIN PLUGIN PATH TO POU ---------------------------------------------------
//$ path to the ST pou where the PLUGIN call has to be inserted
//$ controls the availability of the PLUGIN in the GAT wizard
GatCentral = Gat\Machine\mCyclic | InsertLocation=Implementation | InsertPosition=End
//$ END PLUGIN PATH TO POU -----------------------------------------------

//$ BEGIN PLUGIN CALL ----------------------------------------------------
//$ Code region to be inserted in the configured structured text pou
//$ This code is used if there is no other special code for a specific
//$ GAT type is available
Test
//$ END PLUGIN CALL ------------------------------------------------------
//$ WIZARD END PLUGIN CALL UP --------------------------------------------------
```

Zusätzliche Bibliotheken

Zusätzliche SPS-Bibliotheken, die nicht bereits im Bibliothekenverwalter enthalten sind, können über die hier beschriebene Region hinzugefügt werden.

Die Region ist durch folgende Textmarken gekennzeichnet:

- "WIZARD BEGIN PLUGIN LIBRARIES" definiert den Anfang der Region
- "WIZARD END PLUGIN LIBRARIES" definiert das Ende der Region

Die Vorgabe der Bibliothek kann entweder über einen Platzhalter, oder direkt erfolgen.

Zusätzliche Bibliotheken

```
//$ WIZARD BEGIN PLUGIN LIBRARIES ----------------------------------------------
//$ add required libraries by PlaceholderName or fixed
Optional: Enter additional libraries which are required by the PLUGIN...
//$ example PlacehoderName: RIL_PROFIBUSDP_02 = RIL_ProfibusDP_02, * (System)
//$ example fixed: RIL_ProfibusDP_02, 12.2.0.0 (System)
//$ WIZARD END PLUGIN LIBRARIES ------------------------------------------------
```

Zusätzliche SPS Tasks

Zusätzliche SPS-Tasks, die nicht bereits in der Taskkonfiguration enthalten sind, können über die hier beschriebene Region hinzugefügt werden.

Die Region ist durch folgende Textmarken gekennzeichnet:

- "WIZARD BEGIN PLUGIN TASKS" definiert den Anfang der Region
- "WIZARD END PLUGIN TASKS" definiert das Ende der Region

Weiterhin muss vorab jede einzelne Task über IndraWorks exportiert werden. Diese Exportdatei (*.iwx) muss im lokalen Plug-in Verzeichnis abgelegt werden. In der Konfigurationsdatei wird anschließend der Name der Exportdatei angegeben. Für jede Task muss jeweils eine eigene Exportdatei erzeugt und eingetragen werden.

Zusätzliche SPS Tasks

```
//$ WIZARD BEGIN PLUGIN TASKS --------------------------------------------------
//$ imports the additional defined tasks of the plugin
//$ taskname and filename are equal
Optional: Enter additional tasks which are required by the PLUGIN
Example:
PlugInTask1.iwx
//$ WIZARD END PLUGIN TASKS ----------------------------------------------------
```

Beispiel Konfigurationsdatei

Das nachfolgende Beispiel zeigt eine vollständige Konfigurationsdatei.

Durch "Copy & Paste" des nachfolgenden Textes in einen Texteditor können Sie leicht eine Konfigurationsdatei erstellen.

Alternativ können Sie auch von bestehenden Plug-ins die Konfigurationsdatei kopieren und entsprechend modifizieren.

GAT Plug-ins

Beispiel Konfigurationsdatei

```
//$ WIZARD BEGIN PLUGIN SHORT DESCRIPTION -------------------------------------
Enter PLUGIN short description here...
//$ WIZARD END PLUGIN SHORT DESCRIPTION ---------------------------------------

//$ WIZARD BEGIN PLUGIN DOCUMENTATION LINK ------------------------------------
//$ File in the PLUGINfolder or a hyperlink (starting with 'http')
Enter documentation file name here...
//$ WIZARD END PLUGIN DOCUMENTATION LINK --------------------------------------

//$ ---------------------------------------------------------------------------
//$ WIZARD BEGIN PLUGIN CALL UP -----------------------------------------------
//$ ---------------------------------------------------------------------------

//$ BEGIN PLUGIN PATH TO POU --------------------------------------------------
//$ path to the ST pou where the PLUGIN call has to be inserted
//$ controls the availability of the PLUGIN in the GAT wizard
Enter the path to the POU where the PLUGIN call has to be inserted ...
Examples:
GatCentral = Gat\Machine\Machine\mCyclic | InsertLocation=Implementation | InsertPosition=End
GatDecentral = ...
GatCompact = ...
AxisInterface = ...
//$ END PLUGIN PATH TO POU ----------------------------------------------

//$ BEGIN PLUGIN CALL ---------------------------------------------------
//$ Code region to be inserted in the configured structured text pou
//$ This code is used if there is no other special code for a specific
//$ GAT type is available
Enter code to call and connect the PLUGIN ...
//$ END PLUGIN CALL -----------------------------------------------------

//$ BEGIN PLUGIN CALL GATCENTRAL ----------------------------------------
//$ This section is only required in case special code is required
//$ for GATcentral
Optional : Enter code to call and connect the PLUGIN with GATcentral...
//$ END PLUGIN CALL GATCENTRAL ------------------------------------------

//$ BEGIN PLUGIN CALL GATDECENTRAL --------------------------------------
//$ This section is only required in case special code is required
//$ for GATdecentral
Optional : Enter code to call and connect the PLUGIN with GATdecentral...
//$ END PLUGIN CALL GATDECENTRAL ----------------------------------------

//$ BEGIN PLUGIN CALL GATCOMPACT ----------------------------------------
//$ This section is only required in case special code is required
//$ for GATcompact example project
Optional : Enter code to call and connect the PLUGIN with GATcompact...
//$ END PLUGIN CALL GATCOMPACT------------------------------------------

//$ BEGIN PLUGIN CALL AXISINTERFACE -------------------------------------
//$ This section is only required in case special code is required
//$ for the axis interface example project
Optional : Enter code to call and connect the PLUGIN with AxisInterface...
//$ END PLUGIN CALL AXISINTERFACE ---------------------------------------

//$ ---------------------------------------------------------------------------
//$ WIZARD END PLUGIN CALL UP -------------------------------------------------
//$ ---------------------------------------------------------------------------

//$ WIZARD BEGIN PLUGIN LIBRARIES ---------------------------------------------
//$ add required libraries by PlaceholderName or fixed
Optional: Enter additional libraries which are required by the PLUGIN...
//$ example PlacehoderName: RIL_PROFIBUSDP_02 = RIL_ProfibusDP_02, * (System)
//$ example fixed: RIL_ProfibusDP_02, 12.2.0.0 (System)
//$ WIZARD END PLUGIN LIBRARIES -----------------------------------------------

//$ WIZARD BEGIN PLUGIN TASKS -------------------------------------------------
//$ imports the additional defined tasks of the plugin
//$ taskname and filename are equal
Optional: Enter additional tasks which are required by the PLUGIN and
Example:
PlugInTask1.iwx
//$ WIZARD END PLUGIN TASKS ---------------------------------------------------
```

13 Service und Support

Für Ihre schnelle und optimale Unterstützung verfügen wir über ein dichtes weltweites Servicenetz. Unsere Experten stehen Ihnen mit Rat und Tat zur Seite. Sie erreichen uns täglich **rund um die Uhr - auch am Wochenende und an Feiertagen**.

Service Deutschland

Unser technologieorientiertes Competence Center in Lohr deckt alle Belange rund um den Service für elektrische Antriebe und Steuerungen ab.

Die **Service Helpdesk & Hotline** erreichen Sie unter:

Telefon:	**+49 9352 40 5060**
Fax:	**+49 9352 18 4941**
E-Mail:	service.svc@boschrexroth.de
Internet:	http://www.boschrexroth.com

Auf unseren Internetseiten finden Sie ergänzende Hinweise zu Service, Reparatur (z.B. Anlieferadressen) und Training.

Service weltweit

Außerhalb Deutschlands nehmen Sie bitte zuerst Kontakt mit Ihrem Ansprechpartner auf. Die Hotline-Rufnummern entnehmen Sie bitte den Vertriebsadressen im Internet.

Vorbereitung der Informationen

Wir können Ihnen schnell und effizient helfen, wenn Sie folgende Informationen bereithalten:

- detaillierte Beschreibung der Störung und der Umstände
- Angaben auf dem Typenschild der betreffenden Produkte, insbesondere Typenschlüssel und Seriennummern
- Ihre Kontaktdaten (Telefon-, Faxnummer und E-Mail-Adresse)

Index

A
Abkürzungen ... 18
Achsbetriebsarten ... 73
Achs-Interface Betriebsarten ... 75
Achs Interface Erweiterungen GAT ... 334
Achs-Interface Erweiterungen GAT dezentral ... 400
Achs-Interface Funktionsbausteine ... 66
Achs-Interface-Strukturelemente
Achsübersicht-Visualisierung ... 94
Geschwindigkeitsbetrieb-Visualisierung ... 97
ImcInterface-Status ... 102
ImcInterface-Steuerung ... 102
Positionierbetrieb-Visualisierung ... 95
Synchronisationsbetrieb mit Profilen ... 100
Synchronisationsbetrieb-Visualisierung ... 98
Achs-Interface Strukturen
Axis_Mode ... 120
MB_AXIS_ADJUST_DATA ... 122, 127
MB_AXIS_ADMIN_STATUS ... 111
MB_AXIS_ADMINISTRATION ... 118
MB_AXIS_CONTROL_TYPE01 - Struktur ... 117
MB_AXIS_CONTROL_TYPE02 ... 133
MB_AXIS_CONTROL_TYPE02 - Struktur ... 133
MB_AXIS_Coordinated - Struktur ... 134
MB_AXIS_DIAGNOSIS ... 113
MB_AXIS_FEATURES ... 112
MB_AXIS_MODE ... 112
MB_AXIS_POS_STATUS ... 115
MB_AXIS_POSITIONING ... 122, 123
MB_AXIS_STATUS_TYPE01 ... 110
MB_AXIS_STATUS_TYPE02 ... 132
MB_AXIS_STATUS_TYPE02 - Struktur ... 132
MB_AXIS_SYNC_STATUS ... 116
MB_AXIS_SYNCHRONISATION ... 127
MB_AXIS_VEL_STATUS ... 116
MB_AXIS_VELOCITY ... 125
ML_AXIS_PROFILE ... 130
Achs-Interface-Strukturen ... 109
Achs-Interface Visualisierungen ... 90
actAssignIO ... 343
actAssignIOBackground ... 343
actAssignIOMotionSync ... 343
actAutoProducing ... 344
actAutoStopped ... 344
actDisable ... 328, 344
actError ... 303, 344
actInitColdstart ... 344
actInitWarmstart ... 344
actManualReady ... 344
actOpModeSFC ... 361
Adressvariablen in der Unit ... 399
Antriebssystem ... 25
Anwender-Erweiterung ... 103, 205
arAxisCtrl_gb[ ] ... 105
arStatusCtrl_gb[ ] ... 106

A
...Anwender-Erweiterung
Funktionsbaustein ... 107, 206
Interface-Strukturen erweitern ... 206
Überblick ... 205
Anwendungsbeispiele ... 207
Anmeldung einer benutzerdefinierten Transformation ... 209
Sequenzielle Programmierung mit dem Kinematik-Interface ... 207
Applikationsspezifischer Watchdog ... 45
arAppErrMachine1_gb ... 342
arAxisCtrl_gb ... 320
arAxisStatus_gb ... 320
arMachine1ModuleDiagNo_gb ... 342
Automatik Betriebsart allgemein ... 40
Automatisierung von Applikationen ... 39
AXIS_GAT ... 307
Axis_Ref ... 121

B
Benutzerdefiniertes GAT ... 337
Benutzerdefiniertes GAT dezentral ... 403
bestimmungsgemäßer Gebrauch ... 21
Einsatz- und Anwendungsbereiche ... 21
Stand der Technik ... 21
Betriebsarten ... 182
Absolutes Lineares Positionieren ... 186
Absolutes Positionieren ... 79
Absolutes PTP Positionieren ... 184
Absolutes Zirkulares Positionieren ... 188
Additives Positionieren ... 81
Antrieb Bereit ... 77
Antrieb Halt ... 78
Antrieb Referenzieren ... 79
Antriebsgeführte Unterbrechung ... 87
Antrieb Stop ... 78
Continue ... 191
Elektronische Kurvenscheibe ... 83
Elektronisches Bewegungsprofil ... 84
Elektronisches FlexProfile ... 85
Externer Funktionsbaustein ... 88, 192
Geschwindigkeits-Steuerung ... 81
Geschwindigkeitssynchronisation ... 83
Interrupt ... 191
Kinematik Bereit ... 182, 183
Kinematik Referenzieren ... 183
Kinematik Stopp ... 190
RCL ... 192
Relatives Lineares Positionieren ... 187
Relatives Positionieren ... 80
Relatives PTP Positionieren ... 185
Relatives Zirkulares Positionieren ... 189
Robot Control ... 86
Winkelsynchronisation ... 82

Index

B
Betriebsarten allgemein ... 40
Betriebsarten GAT ... 290
Betriebsarten in der Unit ... 368
Betriebsartenunabhängige Logik ... 40
Betriebsartenwechsel GAT ... 292
Bezeichnungen und Abkürzungen ... 18
Bibliothek
ML_KinTech.library ... 144
ML_TechInterface.library ... 55
bStart_gb ... 342
bStop_gb ... 342

C
ClearError ... 306

D
Das Beispielprojekt GAT compact ... 211
Das Beispielprojekt GAT compact für IndraLogic XLC ohne sercos ... 255
Dialogbasierte Achskommandierung ... 73
DisableErrorHandling GATcentral ... 305
DisableErrorReaction GATdecentral ... 374
DisableLocked ... 328
DisableTransient ... 328
DriveInterruptReaction ... 87

E
Einführung ... 37
Elektrisches Antriebssystem ... 25
Empfohlene Taskaufteilung bei Verwendung von IndraWorks/IndraLogic ... 41
Typische Aufgaben BackgroundTask ... 42
Typische Aufgaben MotionTask ... 42
Typische Aufgaben PlcTask ... 42
EnableCyclicScanning ... 58
Erste Schritte Achs-Interface ... 89
Erste Schritte Kinematik-Interface ... 194

F
Fehlerdiagnose ... 48
Fehlererkennung ... 46
Fehlerhandling allgemein ... 46
Fehler protokollieren ... 307, 376
Fehlerquittierung/Wiederanlauf nach Fehler ... 49
Fehlerreaktion ... 47
Fehlerreaktion GAT ... 303, 373
Funktionsprinzip GAT ... 288, 360

G
GATBackgroundProg ... 301
GATBackgroundProgTask ... 301
GAT compact
Abarbeitung in MotionTask ... 248

G
...GATcompact
Achse entfernen/umbenennen ... 246
Achse hinzufügen ... 245
Achs-Interface Erweiterungen ... 248
Betriebsarten ... 221
Betriebsart hinzufügen ... 247
Daten der Achsen ... 244
Daten der Steuerung ... 244
Daten zum Fehlerhandling ... 243
Daten zur Zustandsmaschine ... 243
Erhöhung der maximalen Achsanzahl für Achsnummern größer 16 ... 246
Externe Visualisierung mit separaten HMI (WinStudio-Visualisierung) ... 234
Fehlererkennung und Diagnose ... 227
Fehlerhandling ... 227
Fehlerlöschen ... 232
Fehlerreaktion ... 229
Hinzufügen einer BackgroundTask ... 226
Hinzufügen einer MotionTask ... 224
Initialisierung ... 218
Schnittstellen zum GAT ... 243
Strukturübersicht ... 252
Taskhandling ... 223
Taskkonfiguration ... 223
Übersicht der Methoden und Aktionen der Maschine ... 253
Übersicht der wichtigsten globalen Variablen der Achsen ... 253
Übersicht der wichtigsten globalen Variablen der Maschine ... 252
Visualisierung ... 232
Zugriff auf Achsdaten ... 244
Zugriff auf Achsen innerhalb der Maschine 254
Zyklische Bearbeitung ... 223
GAT compact
Erste Schritte ... 211
Erste Schritte, einfache Motion-Applikation erstellen ... 211
GAT compact für IndraLogic XLC ohne sercos
Betriebsarten ... 257
Betriebsart hinzufügen ... 271
Daten der Machine ... 269
Daten der Steuerung ... 271
Daten zum Fehlerhandling ... 270
Externe Visualisierung mit separaten HMI (WinStudio-Visualisierung) ... 263
Fehlererkennung und Diagnose ... 260
Fehlerhandling ... 260
Fehlerlöschen ... 262
Fehlerreaktion ... 261
Hinzufügen einer BackgroundTask ... 259
Initialisierung ... 256
Schnittstellen zum GAT ... 269
Strukturübersicht ... 273
Taskhandling ... 258

Index

G

...GATcompact für IndraLogic XLC ohne sercos
Taskkonfiguration ... 258
Übersicht der Methoden und Aktionen der Maschine ... 274
Übersicht der wichtigsten globalen Variablen der Maschine ... 273
Visualisierung ... 262
Zyklische Bearbeitung ... 258
GAT mit dezentraler Zustandsmaschine... 347
Achse entfernen/umbenennen ... 395
Achse hinzufügen ... 394
Achs-Interface Erweiterungen ... 400
Betriebsarten ... 361
Betriebsart hinzufügen ... 395
Daten der Achsen ... 393
Daten der Steuerung ... 393
Daten der Units ... 391
Daten der zentralen Fehlerverwaltung (TE_ErrorHandling) ... 390
Daten des Maschinenverwalters (TE_MachineControl) ... 389
Erste Schritte ... 349
Erste Schritte, einfache Motion-Applikation erstellen ... 349
Erste Schritte, Vorbereitung ... 349
Externe Visualisierung mit separaten HMI (WinStudio-Visualisierung) ... 380
Fehlerbehandlung ... 370
Fehlererkennung und Diagnose ... 370
Fehlerlöschen ... 375
Fehlerreaktion ... 373
Funktionsprinzip ... 360
Generierung eigener Programmrahmen auf Basis des GAT ... 403
Grundidee ... 347
Hinzufügen einer BackgroundTask ... 370
HowTo: Typische Anwenderaktivitäten ... 394
Initialisierung ... 360
Interne Visualisierung zur Inbetriebnahmeunterstützung ... 378
Maschinendaten-Verwaltung ... 376
Neue Unit hinzufügen ... 396
Produktdaten-Verwaltung ... 376
Produkt- und Maschinendaten-Verwaltung 376
Schnittstellen zum GAT mit dezentraler Zustandsmaschine ... 389
Strukturübersicht ... 405
Taskverwaltung ... 370
Übersicht der Methoden und Aktionen der Units ... 408
Übersicht der Methoden und Aktionen des TE_ErrorHandling ... 407
Übersicht der Methoden und Aktionen des TE_MachineControl ... 407
Übersicht der wichtigsten globalen Variablen der Achsen ... 406

G

...GAT mit dezentraler Zustandsmaschine
Übersicht der wichtigsten globalen Variablen der Maschine und der Fehlerbehandlung ... 405
Übersicht der wichtigsten globalen Variablen der Units ... 406
Umschalten der Visualisierung ... 377
Units ... 363
Units deaktivieren ... 397
Verwendung von globalen und Adressvariablen in Units ... 399
Visualisierung ... 377
Voraussetzungen zur Anwendung ... 349
Weitere Instanz einer Unit hinzufügen ... 397
Wiederverwendung von Units ... 398
Zugriff auf Achsdaten ... 394
Zugriff auf Achsen innerhalb der Unit ... 408
Zugriff auf I/O-Daten innerhalb der Unit ... 408
GAT mit zentraler Zustandsmaschine... 275
Abarbeitung in MotionTask ... 333
Achse entfernen/umbenennen ... 322
Achse hinzufügen ... 322
Achs-Interface Erweiterungen ... 334
Anwahl der gewünschten Betriebsart nach Fehlerlöschen ... 324
Betriebsarten ... 290
Betriebsart hinzufügen ... 324
Daten der Achsen ... 320
Daten der Steuerung ... 320
Daten zu den Modulen ... 318
Daten zum Fehlerhandling ... 318
Daten zur Maschine ... 317
Erste Schritte ... 277
Erste Schritte, einfache Motion-Applikation erstellen ... 277
Externe Visualisierung mit separaten HMI (WinStudio-Visualisierung) ... 308
Fehlerbehandlung GAT ... 301
Funktionsprinzip ... 288
GAT Fehlerreaktion für F0 Fehler abschalten ... 340
GAT Kurzübersicht, Übersicht der Methoden und Aktionen der Module ... 344
GAT Strukturübersicht ... 341
Generierung eigener Programmrahmen auf Basis des GAT ... 337
HowTo: Typische Anwenderaktivitäten ... 321
Initialisierung ... 289
Interne Visualisierung zur Inbetriebnahmeunterstützung ... 307
Modul deaktivieren ... 327
Module ... 294
Neues Modul hinzufügen ... 326
PackML V3.0 Programmrahmen basierend auf GAT ... 338
Schnittstellen zum GAT ... 317

Index

G

...GAT mit zentraler Zustandsmaschine
Taskverwaltung 300
Überblick 275
Übergeordnete Logik zur Betriebsartenvorgabe 325
Übersicht der Methoden und Aktionen der Maschine 343
Übersicht der wichtigsten globalen Variablen der Achsen 342
Übersicht der wichtigsten globalen Variablen der Maschine 342
Übersicht der wichtigsten globalen Variablen der Module 342
Update von Version 10/11 auf 12/13 340
Verwendung von globalen und Adressvariablen in Modulen 330
Visualisierung 307
Weitere Instanz eines Moduls hinzufügen . 327
Wiederverwendung von Modulen 328
Zeitkritische applikationsspezifische Fehlerreaktion 340
Zugriff auf Achsdaten 321
Zugriff auf Achsen innerhalb des Moduls … 345
Zugriff auf I/O-Daten innerhalb des Moduls 345
Zustände und Transitionen hinzufügen 325
GATMotionSyncProg..... 301
GATMotionSyncTask..... 301
GAT Plug-in..... 409
Einleitung 409
Plug-in entfernen 409
Plug-in hinzufügen 409
Plug-in nachträglich installieren 409
Plug-in selbst erstellen 410
GATProg..... 325
Generierung und Anzeige von Diagnosemeldungen..... 40
Gliederung der Dokumentation..... 11
Global_GAT_MachineControl..... 319
Global_ImcInterface..... 88
Global_MB_AxisInterfaceVars..... 88
Global_MB_KinInterfaceVars..... 193
Globale Variablen..... 88, 193
Globale Variablen in der Unit..... 399
Grundlagen des Fehlerhandling..... 46
Grundsätze des Taskhandlings..... 41

H

Hinzufügen einer Achse zur Systemübersicht…. 92
Hinzufügen einer Kinematik zur Systemübersicht..... 196

I

ImcCtrl..... 320
IMC-Interface Strukturen..... 134

I

ImcStatus..... 320
Init..... 289
Initialisierung..... 39
Initialisierung GAT..... 289, 360

K

Kinematik-Interface Betriebsarten
Überblick 182
Kinematik-Interface Funktionsbausteine..... 171
Kinematik-Interface Strukturen
MB_KINEMATICS_CONTROL_TYPE01 Struktur 146
MB_KINEMATICS_STATUS_TYPE01 Struktur 164
Kinematik-Interface-Strukturen..... 146
Kinematics_Conveyor_Belt 152
Kinematics_Conveyor_Belt_V1 154
Kinematics_Mode_Union 151
Kinematics_RCL_Point 155
Kinematics_RCL_Prog 158
Kinematics_RCL_Status 167
MB_KINEMATICS_ADMIN_STATUS 165
MB_KINEMATICS_ADMINISTRATION 147
MB_KINEMATICS_CONTROL_TYPE01 … 146
MB_KINEMATICS_COORDINATED 162
MB_KINEMATICS_DIAGNOSIS 170
MB_KINEMATICS_RCL 164
MB_KINEMATICS_SETUP 160
MB_KINEMATICS_SETUP_STATUS 171
MB_KINEMATICS_STATUS_TYPE01 164
Kinematik-Interface Visualisierungen..... 194
KinSystem_Overview_All..... 195
Konfiguration..... 62
Konvertierung von Achs-Interface Projekten…… 64

M

Manual Betriebsart allgemein..... 40
MB_AXIS_ADJUST_DATA..... 122, 127
MB_AXIS_ADJUST_STATUS..... 115
MB_AXIS_ADMIN_STATUS..... 111
MB_AXIS_ADMINISTRATION..... 118
MB_AXIS_CONTROL_TYPE01..... 117, 134
MB_AXIS_CONTROL_TYPE02..... 133, 134
MB_AXIS_Coordinated - Struktur..... 134
MB_AXIS_DIAGNOSIS..... 113
MB_AXIS_FEATURES..... 112
MB_AXIS_MODE..... 112
MB_AXIS_POS_STATUS..... 115
MB_AXIS_POSITIONING..... 122, 123
MB_AXIS_STATUS_TYPE01..... 110
MB_AXIS_STATUS_TYPE02..... 132
MB_AXIS_SYNC_STATUS..... 116
MB_AXIS_SYNCHRONISATION..... 127
MB_AXIS_VEL_STATUS..... 116
MB_AXIS_VELOCITY..... 125

Index

M

MB_AxisInitType01 ... 66
MB_AxisInterfaceType01 ... 68, 70, 72
MB_AxisInterfaceType02 ... 71, 72, 73
MB_IMC_Control_Type01 ... 135
MB_IMC_Status_Type01 ... 135
MB_IRDProgSelect ... 72
MB_IRDProgStart ... 73
MB_KINEMATICS_ADMIN_STATUS ... 165
MB_KINEMATICS_ADMINISTRATION ... 147
MB_KINEMATICS_COORDINATED ... 162
MB_KINEMATICS_DIAGNOSIS ... 170
MB_KINEMATICS_RCL ... 164
MB_KINEMATICS_SETUP ... 160
MB_KINEMATICS_SETUP_STATUS ... 171
MB_Phasing ... 104
MB_PhasingSlave ... 104
mBackground ... 301, 343, 344
mClearError ... 349
mCyclic ... 343, 344
mDataHandler ... 376
mDistributeError ... 349
mEntryOpMode ... 293, 343, 344
mErrorHandling ... 370
Messung der Zykluszeiten ... 43
  Task Execution Viewer der Steuerung ... 44
  Zeitmessung im SPS-Projekt ... 44
  Zykluszeitendialog der Steuerung ... 43
mExitOpMode ... 292, 343, 344
mFB_AxesInit ... 345
mForceOpMode ... 293, 344
mGetAxis ... 321, 345
mIsCriticalError ... 305, 343, 344
mIsMyAxis ... 344
ML_AXIS_PROFILE ... 130
ML_KinematicsInitType01 ... 171, 172
ML_KinematicsInterfaceType01 ... 174, 175
ML_KinJog ... 178, 180
ML_KinJogV1 ... 176, 177
ML_Kintech.library ... 144
ML_TechInterface.library ... 55
mMotionCyclic ... 301, 333, 343, 344
ModeAB ... 77
ModeAH ... 78
ModeCoordAB ... 182
ModeCoordAH ... 183
ModeCoordContinue ... 191
ModeCoordExternalFB ... 192
ModeCoordHoming ... 183
ModeCoordInterrupt ... 191
ModeCoordPosCircAbs ... 188
ModeCoordPosCircRel ... 189
ModeCoordPosDirAbs ... 184
ModeCoordPosDirRel ... 185
ModeCoordPosLinAbs ... 186
ModeCoordPosLinRel ... 187
ModeCoordRCL ... 192

M

ModeCoordStopping ... 190
ModeExternalFB ... 88
ModeFlexProfile ... 85
ModeHoming ... 79
ModeInterrupted ... 87
ModePosAbs ... 79
ModePosAdd ... 81
ModePosRel ... 80
ModeStopping ... 78
ModeSyncCam ... 83
ModeSyncPhase ... 82
ModeSyncProfile ... 84
ModeSyncVel ... 83
ModeVel ... 81
Modulachsen ... 295
Modularisierungsregeln ... 295, 364
Module ... 294
Modulinstanzen ... 299
Modulkoordination ... 300
Modulschnittstelle ... 295
Modulspezifische I/O-Daten ... 298
Modul Wiederverwendung ... 328
mOneTimeInit ... 289, 330, 343, 361
mOpModeChangeAllowed ... 292, 343
mReleaseOpMode ... 293
mScanError ... 349
mSetAppError ... 302, 372
mSetFBError ... 303, 373
mSetModuleEnable ... 327

N

nicht bestimmungsgemäßer Gebrauch ... 22
  Folgen, Haftungsausschluss ... 21

O

OPMODE_AUTO ... 291
OpMode_ErrorSuccessor ... 324
OPMODE_INIT_COLD ... 360
OPMODE_INIT_WARM ... 360

P

PackML V3.0 ... 338
PELV ... 30
PowerOn ... 77
Programm zum Achs-Interface ... 89
Programm zum Kinematik-Interface ... 194

R

Robot Control ... 73, 86, 138, 249, 335, 401

S

Schnittstellen zum GAT ... 317
Schnittstellen zum GAT dezentral ... 389
Schutzkleinspannung ... 30

Index

S
Sequentielle Programmierung ... 368
Sicherheitshinweise ... 17
Sicherheitshinweise für elektrische Antriebe und Steuerungen ... 25
SkipColdstartAtStartup ... 289
stControlDataActive_gb ... 377
stControlDataCustomer_gb ... 377
stControlDataVisuIL_gb ... 377
stMachine1Ctrl_gb ... 342
stMachine1Status_gb ... 342
Support
siehe Service-Hotline ... 415
System_Overview_All_Axis ... 91
Systemübersicht Navigation ... 93, 197

T
Taskhandling allgemein ... 41
Taskkonfiguration GAT ... 300
Taskkonfiguration GAT dezentral ... 370
TE_AxisInterfaceType01Prog ... 89
TE_BASE_MODULE_AXIS ... 295
TE_BaseUnit ... 363
TE_CMI_CONTROL ... 318
TE_CMI_STATUS ... 319
TE_DummyMode ... 290
TE_DummyModule ... 294
TE_ErrorHandling ... 370
TE_GAT_AXIS ... 295
TE_KinInterfaceType01Prog ... 194
TE_Machine ... 332
TE_MACHINE_CTRL ... 317
TE_MACHINE_DATA ... 376
TE_MACHINE_STATUS ... 317
TE_MachineControl ... 369
TE_OPMODES ... 361
TE_PRODUCT_DATA ... 376
TE_UNIT_CONTROL ... 364
TE_UNIT_STATUS ... 364
TechInterface
Einführung und Übersicht ... 55
TechInterface Erst-Konfiguration ... 62
Typische Inhalte einer Visualisierung ... 49
Anzeige des Maschinen-Status ... 49
Steuern der Maschinenzustände ... 50

U
Unitachsen ... 364
Unitinstanzen ... 367
Unitkoordination ... 369

U
Units ... 363
Unit-Schnittstelle ... 364
Unitspezifische I/O-Daten ... 367
Unit Wiederverwendung ... 398
UpdateEveryInput ... 59
UserData ... 330, 345

V
Verwendete Symbole ... 18
Visualisierung ... 49
Achsübersicht ... 93
Bandkonfiguration ... 201
Geschwindigkeitsbetrieb ... 96
Imc-Interface-Status ... 101
Imc-Interface-Steuerung ... 101
Kinematikübersicht ... 198
Positionierbetrieb ... 95
Positionierübersicht ... 199
Referenzierübersicht ... 199
Robot-ControlLanguage ... 201
Robot Control Language Programm ... 204
Robot Control Language Punkt ... 203
SPS-Auslastung ... 103
Synchronisationsbetrieb ... 97
Synchronisationsbetrieb mit Profilen ... 99
Systemübersicht ... 91, 195
Tipp-/Einrichtbetrieb ... 202
Überblick ... 90, 194
Visualisierung, Bereitstellung der Daten durch die SPS ... 50
Kommunikations-Konfiguration ... 53
Symbolkonfiguration ... 50
Visualisierung GAT Dezentral ... 377

W
Wichtige Gebrauchshinweise ... 21

Z
Zeitkritische applikationsspezifische Fehlerreaktion ... 340
Zielbetriebsart nach Initialisierung des GAT ... 291
Zu dieser Dokumentation ... 11
Darstellung von Informationen ... 17
Gültigkeit der Dokumentation ... 11
Zustände einer Betriebsart ... 290
Zustände im Modul ... 299
Zyklische Bearbeitung allgemein ... 40

Notizen

Rexroth
Bosch Group

Bosch Rexroth AG
Electric Drives and Controls
Postfach 13 57
97803 Lohr, Deutschland
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr, Deutschland
Tel. +49 9352 18 0
Fax +49 9352 18 8400
www.boschrexroth.com/electrics

R911336369

DOK-XLCMLC-TF*GAT**V13-AP02-DE-P