



DOKUMENTATION ISG-kernel

SPS-Bibliothek **ISG Motion Control Platform für PLCopen**

Kurzbezeichnung:
MCP-INTRO

© Copyright
ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart
Alle Rechte vorbehalten
www.isg-stuttgart.de
support@isg-stuttgart.de

Dokumentation Version: 1.03
13.12.2023

Inhaltsverzeichnis

| | | |
|----------|---|-----------|
| 1 | Definitionen | 5 |
| 1.1 | Abkürzungen | 5 |
| 1.2 | Begriffserklärungen | 5 |
| 2 | Die ISG Motion Control Platform | 7 |
| 2.1 | Was ist die ISG Motion Control Platform ? | 7 |
| 2.2 | Elemente der ISG Motion Control Platform | 7 |
| 2.2.1 | HLI-Bibliothek – Speicherschnittstelle zur ISG-MCE | 8 |
| 2.2.2 | Plattform-Bibliothek | 8 |
| 2.2.3 | Motion-Bibliothek – PLCopen Part1 | 9 |
| 2.2.3.1 | PLCopen Funktionsbausteine | 11 |
| 2.2.3.2 | Funktionsbaustein MCV_Axis | 12 |
| 2.2.3.3 | Funktionsbaustein MCV_P1_PLATFORM | 13 |
| 2.2.4 | Achsgruppen-Bibliothek – PLCopen Part4 | 16 |
| 2.2.4.1 | Funktionsbaustein MCV_AxesGroup | 18 |
| 2.2.4.2 | Funktionsbaustein MCV_P4_PLATFORM | 19 |
| 2.2.4.3 | PLCopen Funktionsbausteine | 22 |
| 2.2.5 | Globale Variablen | 23 |
| 2.3 | Sicherheitskonzept, Einhaltung der EN775 | 25 |
| 2.3.1 | Grundsätzliches zum softwaretechnischen Sicherheitskonzept | 25 |
| 2.3.2 | Ist-Geschwindigkeitsüberwachung | 25 |
| 2.3.3 | Bidirektionale Kongruenzprüfung der HLI Speicher Schnittstelle | 25 |
| 2.3.4 | Priorität des FB MC_Stop | 25 |
| 2.3.5 | Geschwindigkeitsüberwachung während aktiver Drehmomentbegrenzung | 26 |
| 2.3.6 | Verhindern von unbeabsichtigten MCFB-Bewegungen im T1-Mode | 26 |
| 2.3.7 | Sicherheitskleingeschwindigkeit für nicht referenzierte Achsen | 26 |
| 2.4 | Realisierungsdetails innerhalb der ISG-MCP | 27 |
| 2.4.1 | Hochlauf | 27 |
| 2.4.2 | Grundsätzliches zur Arbeitsweise der PLCopen FB | 27 |
| 2.4.3 | Realisierung der FB | 27 |
| 2.4.4 | Interaktion der FB mit dem FBSD, Fehlerhandling | 30 |
| 2.4.4.1 | Fehlerbehandlung auf FBSD Ebene | 30 |
| 2.4.4.2 | Fehlerbehandlung auf FB Ebene | 31 |
| 2.4.4.3 | Definierte Fehler auf FB-Ebene | 31 |
| 2.4.4.4 | Achsfehler aus dem Motion Controller | 31 |
| 2.4.5 | Versionierung | 32 |
| 2.4.5.1 | Versionsüberprüfung durch FB | 32 |
| 2.4.6 | Weitere allgemeine Systemeigenschaften | 32 |
| 3 | Anhang 1: Best Practise bei der SPS-Anwendungsprogrammierung | 33 |
| 3.1 | Grundsätzliches | 33 |
| 3.2 | Wesentliches in Kürze | 33 |
| 3.2.1 | Verhalten der „Execute“ und „Done“ Ein / Ausgänge der PLCopen-FB | 33 |
| 3.2.2 | Knackpunkt: Auftragsdurchsetzung und -quittierung | 33 |
| 3.3 | Tipps und Tricks zur SPS-Anwendungsprogrammierung | 35 |
| 3.4 | Tipps zur Laufzeitoptimierung | 36 |
| 4 | Anhang 2: HelloWorld mit der ISG Motion Control Platform | 37 |

| | | |
|----------|--|-----------|
| 4.1 | „Multiprog“-Programmierbeispiel..... | 37 |
| 4.1.1 | Schritt 1: Einfügen der erforderlichen Bibliotheken..... | 37 |
| 4.1.2 | Schritt 2: Anlegen von SPS-Programm Main und HelloWorld | 38 |
| 4.1.3 | Schritt 3: Implementation von Programm Main..... | 39 |
| 4.1.4 | Schritt 4: Programm HelloWorld: Instanzieren der PLCopen FB..... | 40 |
| 4.1.5 | Schritt 5: Anbinden der Achse an die PLCopen FB..... | 41 |
| 4.1.6 | Schritt 6: Belegen der Baustein-IN/OUT-Varibalen..... | 42 |
| 4.1.7 | Schritt 7: Zuordnung der Programme zu einer Task..... | 44 |
| 4.1.8 | Schritt 8: Anlegen der erforderlichen globalen Variablen..... | 44 |
| 4.1.9 | Schritt 9: Projekt senden, Kalt starten..... | 45 |
| 4.1.10 | Schritt 10: Setzen der Freigaben für die Achse | 46 |
| 4.1.11 | Schritt 11: Setzen der Freigabe für PLCopen-FB | 46 |
| 4.1.12 | Schritt 12: Fertig, Achse ist verfahren!..... | 47 |
| 4.2 | „CoDeSys“-Programmierbeispiel | 48 |
| 4.2.1 | Schritt 1: Erforderliche Bibliotheken..... | 48 |
| 4.2.2 | Schritt 2: Anlegen des Applikationsprogrammes HelloWorld..... | 49 |
| 4.2.3 | Schritt 3: Programm HelloWorld: Instanzieren der PLCopen FB..... | 50 |
| 4.2.4 | Schritt 4: Anbinden der Achse an die PLCopen FB..... | 51 |
| 4.2.5 | Schritt 5: Belegen der Baustein-IN/OUT-Varibalen..... | 52 |
| 4.2.6 | Schritt 6: Programm HelloWorld in Programm MAIN einfügen | 54 |
| 4.2.7 | Schritt 7: Zuordnung der Programme zu einer Task..... | 55 |
| 4.2.8 | Schritt 8: Applikation übersetzen, Einloggen , Starten..... | 56 |
| 4.2.9 | Schritt 9: Setzen der Freigaben für die Achse | 57 |
| 4.2.10 | Schritt 10: Fertig, Achse ist verfahren!..... | 58 |
| 5 | Literaturverzeichnis | 59 |
| 6 | Anhang | 60 |
| 6.1 | Anregungen, Korrekturen und neueste Dokumentation..... | 60 |
| | Stichwortverzeichnis..... | 61 |

Abbildungsverzeichnis

| | | |
|----------|--|----|
| Abb. 1: | Der SPS-Anwendungsprogrammierer sieht die ISG-MCP als einzige Programmierschnittstelle..... | 7 |
| Abb. 2: | Übersicht über die Motion Bibliothek McpPLCopenP1.lib in CoDeSys | 9 |
| Abb. 3: | Struktureller Aufbau der Motion Bibliothek McpPLCopenP1.zwt in Multiprog | 10 |
| Abb. 4: | | 12 |
| Abb. 5: | SPS-Basisprogramm für Motion-Applikationen in CoDeSys | 14 |
| Abb. 6: | Programm Main mit Instanz des FB MCV_P1_PLATFOrm wird als erstes Programm in der Task aufgerufen | 15 |
| Abb. 7: | Übersicht über die Motion Bibliothek McpPLCopenP4.lib in CoDeSys | 17 |
| Abb. 8: | Struktureller Aufbau der Motion Bibliothek McpPLCopenP4.zwt..... | 18 |
| Abb. 9: | SPS-Basisprogramm für Achsgruppen-Applikationen in CoDeSys-Umgebung | 20 |
| Abb. 10: | Programm Main mit Instanz des FB MCV_P4_PLATFOrm in Multiprog-Entwicklungsumgebung | 21 |
| Abb. 11: | Erforderliche globale Variablen zur Verwendung der ISG-MCP im Multiprog-Entwicklungsumgebung..... | 24 |
| Abb. 12: | Zustände innerhalb eines FB..... | 28 |
| Abb. 13: | Error-handler versorgt die achsspezifischen FBSD Arbeitsdaten | 31 |
| Abb. 14: | Verstopfungszustand falls ein FB nicht mehr aufgerufen wird | 34 |
| Abb. 15: | Erforderliche Bibliotheken in Projekt einbinden | 37 |
| Abb. 16: | Anlegen des Programms Main in ST | 38 |
| Abb. 17: | Anlegen von Programm HelloWorld in FBD | 38 |
| Abb. 18: | Einordnung der Programme Main und HelloWorld in Projektbaum..... | 38 |
| Abb. 19: | Implementation von Programm Main..... | 39 |
| Abb. 20: | Im Programm HelloWorld instanzierte PLCopen-FB | 40 |
| Abb. 21: | Anbinden der ersten Achse im System an PLCopen-FB über g_array_axis_ref[0]..... | 41 |
| Abb. 22: | Deklaration der Variablen im Variablenarbeitsblatt | 43 |
| Abb. 23: | Ein-/Ausgabevariablen an PLCopen-FB angeschlossen..... | 43 |
| Abb. 24: | Einfügen der Instanzen der Programme Main und HelloWorld in Task..... | 44 |
| Abb. 25: | Globale Variablen werden in entsprechender Resource angelegt | 44 |
| Abb. 26: | Kontrolldialog zum Senden, starten der SPS-Applikation | 45 |
| Abb. 27: | Zustand des Programmes HelloWorld nach dem Programmstart | 45 |
| Abb. 28: | Setzen von Regler- und Vorschubfreigabe an MC_Power_1 | 46 |
| Abb. 29: | Setzen der Eingangsvariable StartMotion um die Bewegung zu starten..... | 46 |
| Abb. 30: | Zustand nach Ende der Bewegung | 47 |
| Abb. 31: | Erforderliche Bibliotheken in Projekt eingebunden..... | 48 |
| Abb. 32: | Definition des Programms HelloWorld..... | 49 |
| Abb. 33: | Einordnung der Programme Main und HelloWorld in Projektbaum..... | 49 |
| Abb. 34: | Im Programm HelloWorld instanzierte PLCopen-FB | 50 |
| Abb. 35: | Anbinden der ersten Achse im System an PLCopen-FB über g_array_axis_ref[0]..... | 51 |
| Abb. 36: | Ein-/Ausgabevariablen an PLCopen-FB angeschlossen..... | 53 |
| Abb. 37: | Einfügen von Programm HelloWorld in Programm MAIN..... | 54 |
| Abb. 38: | Programm MAIN ist der Task Standard zugewiesen..... | 55 |
| Abb. 39: | Programm HelloWorld nach dem Starten der Applikation | 56 |
| Abb. 40: | Setzen von Regler- und Vorschubfreigabe an MC_Power_1 | 57 |
| Abb. 41: | Zustand am Ende der Bewegung | 58 |

1 Definitionen

1.1 Abkürzungen

| | |
|----------|--|
| AXHLI | Achsspezifisches High-Level-Interface |
| CM | Continuous Motion (Endlosdrehen) |
| DM | Discrete Motion (Positionieren) |
| FB | Function Block (Funktionsbaustein) |
| FBSD | FB-State Diagram |
| HLI | High-Level-Interface zwischen MC und PLC |
| MC | Motion Controller |
| MCP | Motion Control Platform |
| MCE | Motion Control Engine |
| MC-FB | Motion Controller Function Block |
| NL-Slope | Nicht-Linearer Slope |
| PCS | Part program coordinate system; Teileprogrammkoordinatensystem |
| PLC | Programmable Logic Control |
| POE | Programmorganisationseinheit |
| SAI | Single Axis Interpolator |

1.2 Begriffserklärungen

| | |
|-------------------|---|
| Achsgruppe | Ein Verbund von Achsen, die durch einen Kanal eine Bewegung auf einer Raumkurve koordiniert durchführen können unter Einhaltung vorgegebener Werte für die Geschwindigkeit, Beschleunigung und Ruck auf dieser Raumkurve. |
| CoDeSys | SPS-Programmiersystem der Fa. 3S Smart Software Solutions |
| Funktionssatz | Internes Beauftragungsformat des ISG Motion-Controllers. |
| HLI-Bibliothek | Zugriff auf die Speicherschnittstelle zur ISG-MCE. |
| ISG-MCE | Damit ist der ISG NC-Kern gemeint, der im Zusammenhang mit dieser Dokumentation auch als „Motion Control Engine“ bezeichnet wird. |
| Kanal | Einheit, die Achsbewegungen einer Achsgruppe koordiniert. |
| MC-FB | Bezeichnet die SPS-Funktionsbausteine, die zur Beauftragung des ISG-MC verwendet werden. |
| Multiprog | SPS-Programmiersystem der Fa. KW-Software |
| Motion-Bibliothek | SPS-Softwareapplikation, die Funktionsbausteine zur Bewegung von Achsen entsprechend der PLCopen-Spezifikation, sowie weitere FB, die Aufgaben der Bewegungserzeugung übernehmen, enthält. |

| | |
|-------------------|---|
| Achsgruppe | Ein Verbund von Achsen, die durch einen Kanal eine Bewegung auf einer Raumkurve koordiniert durchführen können unter Einhaltung vorgegebener Werte für die Geschwindigkeit, Beschleunigung und Ruck auf dieser Raumkurve. |
| CoDeSys | SPS-Programmiersystem der Fa. 3S Smart Software Solutions |
| Funktionssatz | Internes Beauftragungsformat des ISG Motion-Controllers. |
| HLI-Bibliothek | Zugriff auf die Speicherschnittstelle zur ISG-MCE. |
| ISG-MCE | Damit ist der ISG NC-Kern gemeint, der im Zusammenhang mit dieser Dokumentation auch als „Motion Control Engine“ bezeichnet wird. |
| Kanal | Einheit, die Achsbewegungen einer Achsgruppe koordiniert. |
| MC-FB | Bezeichnet die SPS-Funktionsbausteine, die zur Beauftragung des ISG-MC verwendet werden. |
| Motion-Bibliothek | SPS-Softwareapplikation, die Funktionsbausteine zur Bewegung von Achsen entsprechend der PLCopen-Spezifikation, sowie weitere FB, die Aufgaben der Bewegungserzeugung übernehmen, enthält. |

| | |
|-------------------|---|
| Achsgruppe | Ein Verbund von Achsen, die durch einen Kanal eine Bewegung auf einer Raumkurve koordiniert durchführen können unter Einhaltung vorgegebener Werte für die Geschwindigkeit, Beschleunigung und Ruck auf dieser Raumkurve. |
| Funktionssatz | Internes Beauftragungsformat des ISG Motion-Controllers. |
| HLI-Bibliothek | Zugriff auf die Speicherschnittstelle zur ISG-MCE. |
| ISG-MCE | Damit ist der ISG NC-Kern gemeint, der im Zusammenhang mit dieser Dokumentation auch als „Motion Control Engine“ bezeichnet wird. |
| Kanal | Einheit, die Achsbewegungen einer Achsgruppe koordiniert. |
| MC-FB | Bezeichnet die SPS-Funktionsbausteine, die zur Beauftragung des ISG-MC verwendet werden. |
| Multiprog | SPS-Programmiersystem der Fa. KW-Software |
| Motion-Bibliothek | SPS-Softwareapplikation, die Funktionsbausteine zur Bewegung von Achsen entsprechend der PLCopen-Spezifikation, sowie weitere FB, die Aufgaben der Bewegungserzeugung übernehmen, enthält. |

Obligatorischer Hinweis zu Verweisen auf andere Dokumente

Zwecks Übersichtlichkeit wird eine verkürzte Darstellung der Verweise (Links) auf andere Dokumente bzw. Parameter gewählt, z.B. [PROG] für Programmieranleitung oder P-AXIS-00001 für einen Achsparameter.

Technisch bedingt funktionieren diese Verweise nur in der Online-Hilfe (HTML5, CHM), allerdings nicht in PDF-Dateien, da PDF keine dokumentenübergreifenden Verlinkungen unterstützt.

2 Die ISG Motion Control Platform

2.1 Was ist die ISG Motion Control Platform ?

Die ISG-MCP ist eine SPS-Bibliothek, die je nach Kundenwunsch auch in IEC 61131-Source ausgeliefert werden kann. Sie ermöglicht dem SPS-Anwendungsprogrammierer die Programmierung von Bewegungsaufgaben nach der PLCopen-Spezifikation innerhalb einer IEC 61131 SPS. Sämtliche zur Bewegungserzeugung intern notwendige Funktionen bleiben dabei dem SPS-Anwendungsprogrammierer verborgen, wie z.B.:

- Interpolation
- Lageregelung
- Bedienung der Antriebsschnittstellen usw.

Die ISG-MCP stellt die in der PLCopen-Spezifikation [1] definierten Funktionsbausteine, Datenstrukturen und Zustandsmodelle zur Verfügung.

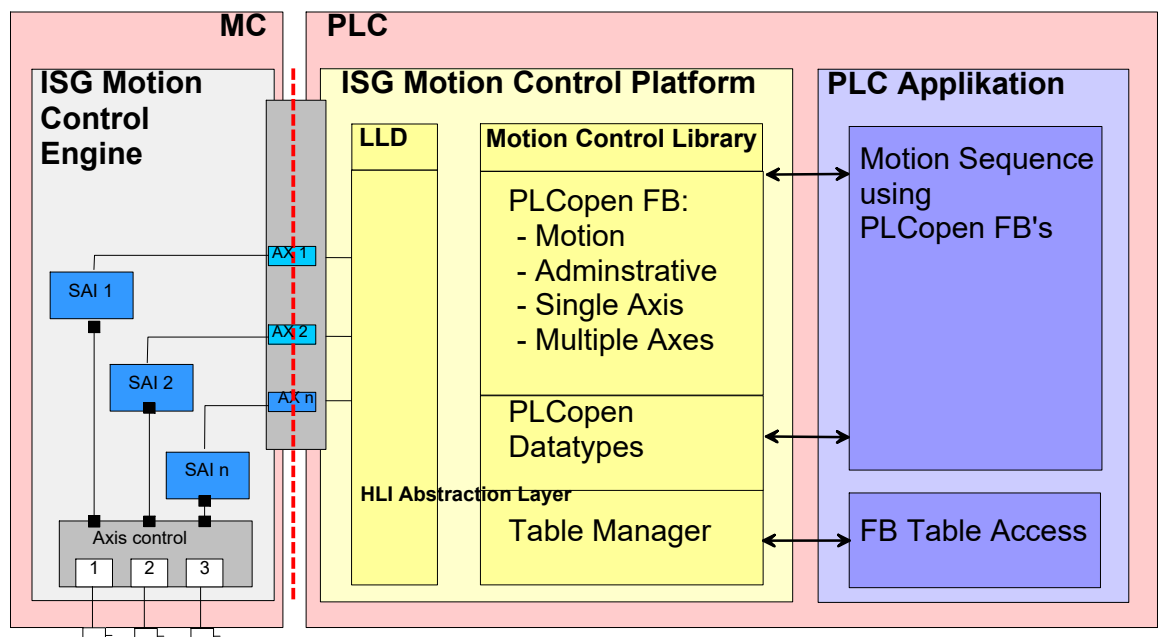


Abb. 1: Der SPS-Anwendungsprogrammierer sieht die ISG-MCP als einzige Programmierschnittstelle.

2.2 Elemente der ISG Motion Control Platform

Die ISG Motion Control Platform umfasst verschiedene SPS Anwenderbibliotheken. Diese beinhalten: FB und Datentypen nach den PLCopen-Spezifikationen und solche, die von Steuerungshersteller spezifiziert wurden. Durch das vorgestellte Präfix können diese Elemente leicht unterschieden werden:

- Alle mit dem Präfix **MC_** gekennzeichneten Elemente sind in den verschiedenen Teilen der PLCopen-Spezifikation aufgeführt.
- Diejenigen Elemente mit dem Präfix **MCV_** sind durch den Steuerungshersteller spezifiziert worden.

2.2.1 HLI-Bibliothek – Speicherschnittstelle zur ISG-MCE

Ein Bestandteil der ISG-MCP ist die Anwenderbibliothek hli.lib.

Sie enthält die Definition der Speicherschnittstelle HLI zur ISG-MCE. Über diese Schnittstelle setzen die PLCopen-FB die Kommandos zur Bewegung ihrer zugeordneten Achse ab und erhalten Meldungen der ISG-MCE bezüglich jeder Achse.

In der Multiprog-Umgebung wird für Zugriffe auf das HLI die globale Variable **hli** als %M3.xxx-Variable in der SPS-Applikation angelegt.

Eine SPS-Applikation in der CoDeSys-Umgebung muss eine Instanz des FB **MCV_HliInterface** als allerersten Baustein aufrufen, der global angelegte Zeiger zum Zugriff auf die Bereiche des HLI initialisiert (siehe Frame_PLCopenP1).



Hinweis

Erst nach erfolgreicher Initialisierung dürfen Programme und FB aus den nachfolgend beschriebenen Anwenderbibliotheken aufgerufen werden.

2.2.2 Plattform-Bibliothek

In der Anwenderbibliothek McpBase.lib sind die Datenstrukturen definiert, die im Rahmen der PLCopen-Spezifikationen als Referenz die Objekte darstellen, durch deren Anwendung die Bewegungsaufgaben gelöst werden sollen.

Die Referenzen sind als globale Variablen bereits in der Bibliothek vorhanden.

Die Variablen müssen in der SPS-Applikation als globale Variable angelegt werden.

Weiterer zentraler Bestandteil dieser Bibliothek ist der FB **MCV_PlatformBase**, der in jeder SPS-Applikation instanziiert werden muss, die Bewegungsaufgaben auf der Basis der PLCopen-Spezifikationen löst.

Dieser FB übernimmt die Aufgabe die Strukturen auf die Referenzen zu initialisieren und die Konsistenz der Schnittstelle HLI auf Seiten der MCE und der SPS zu prüfen. Erst wenn dieser FB seinen Ausgang „Done“ auf TRUE gesetzt hat, können Bewegungsaufträge erfolgreich über die FB der nachfolgend aufgeführten Motion-Bibliothek an den MC abgesetzt werden (siehe Frame_PLCopenP1).

2.2.3 Motion-Bibliothek – PLCopen Part1

In der Anwenderbibliothek McpPLCopenP1.lib sind neben FB, die der PLCopen-Spezifikation Part 1 entsprechen, auch FB definiert, die zusätzliche Funktionalität abdecken und zur Realisierung einer Applikation eingesetzt werden müssen. Diese Bibliothek wird im weiteren Motion-Bibliothek genannt.



Versionshinweis

Der Versionsumfang unterscheidet sich je nach verwendeter SPS-Plattform!

Das nachfolgende Bild zeigt den strukturellen Aufbau der Motion-Bibliothek. Anschließend werden die wesentlichen Elemente dieser Bibliothek näher erläutert:

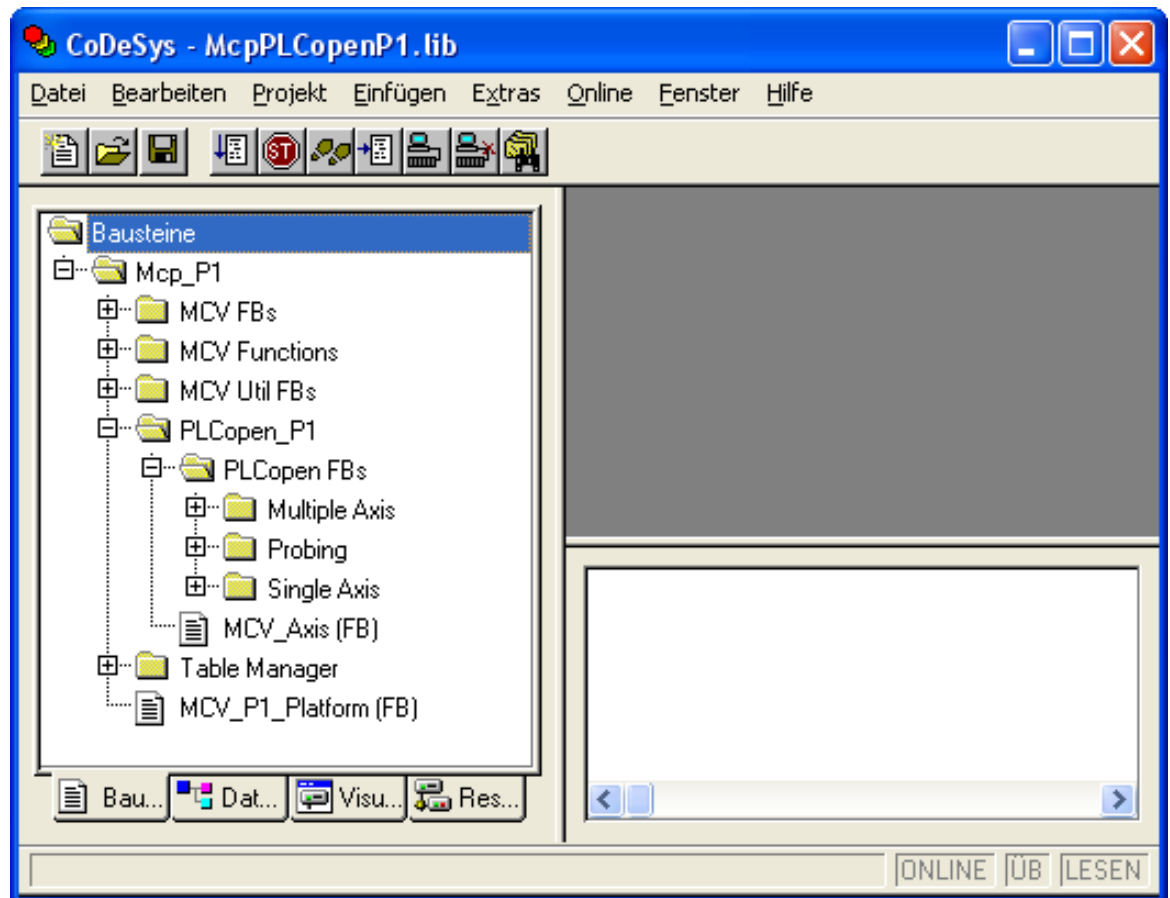


Abb. 2: Übersicht über die Motion Bibliothek McpPLCopenP1.lib in CoDeSys

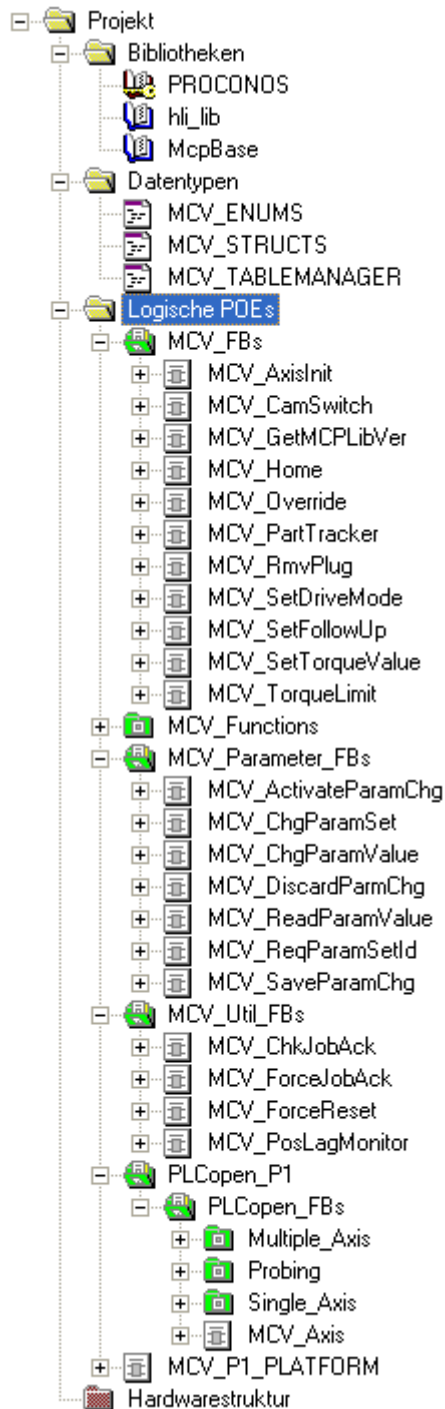


Abb. 3: Struktureller Aufbau der Motion Bibliothek McpPLCopenP1.zwt in Multiprog

2.2.3.1 PLCopen Funktionsbausteine

In der PLCopen-Spezifikation Part 1 werden die dort definierten FB entsprechend ihrer Verwendung unterteilt in:

- administrative und
- bewegungsbezogene FB.

Innerhalb dieser beiden Bereiche wird eine weitere Unterscheidung bezüglich der Anwendung getroffen, nämlich auf:

- eine (single axis) oder
- mehrere (multiple axis) Achsen.

Die nachfolgende Tabelle ist entsprechend organisiert und zeigt die Funktionsblöcke nach PLCopen-Spezifikation Part 1.



Hinweis

Die kursiv gedruckten und mit einem * versehenen FB sind nicht in der Motionbibliothek Part1 implementiert. Jedoch kann es in den Bibliotheken FB geben, die eine ähnliche Funktionalität besitzen, aber durch den Steuerungshersteller spezifiziert wurden.

Einteilung der PLCopen-FB Part1 in administrative und bewegungsbezogene FB

| Administrative | | Motion | |
|-------------------------------|-------------------|--------------------------------|----------------------|
| Single Axis | Multiple Axis | Single Axis | Multiple Axis |
| MC_Power | MC_CamTableSelect | MC_MoveAbsolute | MC_CamIn |
| MC_ReadStatus | | MC_MoveRelative | MC_CamOut |
| MC_ReadAxisError | | MC_MoveAdditive | MC_GearIn |
| MC_ReadParameter | | MC_MoveSuperimposed | MC_GearOut |
| <i>MC_ReadBoolParameter*</i> | | MC_MoveVelocity | MC_Phasing |
| MC_WriteParameter | | MC_Home | <i>MC_GearInPos*</i> |
| <i>MC_WriteBoolParameter*</i> | | MC_Stop | |
| MC_ReadActualPosition | | <i>MC_PositionProfile*</i> | |
| MC_Reset | | <i>MC_VelocityProfile*</i> | |
| MC_TouchProbe | | <i>MC_AccelerationProfile*</i> | |
| MC_AbortTrigger | | <i>MC_TorqueControl*</i> | |
| <i>MC_ReadDigitalInput*</i> | | <i>MC_MoveContinuous*</i> | |
| <i>MC_ReadDigitalOutput*</i> | | MC_Halt | |
| <i>MC_WriteDigitalOutput*</i> | | | |
| MC_SetPosition | | | |
| MC_SetOverride | | | |
| <i>MC_ReadActualVelocity*</i> | | | |
| <i>MC_ReadActualTorque*</i> | | | |
| <i>MC_DigitalCamSwitch*</i> | | | |

2.2.3.2 Funktionsbaustein MCV_Axis

Aktualisiert werden die Daten einer Struktur `AXIS_REF` durch den FB `MCV_Axis`, der als Ein-/Ausgabevariable eine Struktur `AXIS_REF` besitzt. Dieser FB übernimmt zusätzlich folgende Aufgaben:

- Anmeldung einer Achse an der MCE über das HLI. Dies geschieht durch Setzen des Flags „`plc_present_w`“ auf dem achsspezifischen HLI-Bereich
- Anmeldung der SPS über das HLI, damit die SPS Spindel-Reset, Reglerfreigabe, Vorschubfreigabe und Antrieb EIN für eine spezifische Achse an die MCE kommandieren kann.
- Bei der Initialisierung wird die Konsistenz des HLI verifiziert, indem die Versionskennung und die Größe des HLI überprüft wird.
- Übernahme der Fehlermeldungen, die von der MCE achsspezifisch gemeldet werden.

In jeder SPS-Applikation, die `PLCopen-Part1` FB der ISG-MCP benutzt, muss für jede verwendete Achse eine Instanz dieses FB angelegt sein, und diesem eine Struktur `AXIS_REF` in der Form `g_array_axis_ref[i]` als `VAR_IN_OUT`-Parameter zugewiesen werden.

Um dies zu gewährleisten enthält die ISG-MCP den FB `MCV_P1_PLATFORM` (siehe Kap. Funktionsbaustein `MCV_P1_PLATFORM` [▶ 13]), der in einem Programm einer SPS-Applikation aufgerufen werden muss. Damit ist gewährleistet, dass die Arbeitsdaten einer Achse in jedem SPS-Zyklus aktualisiert werden.

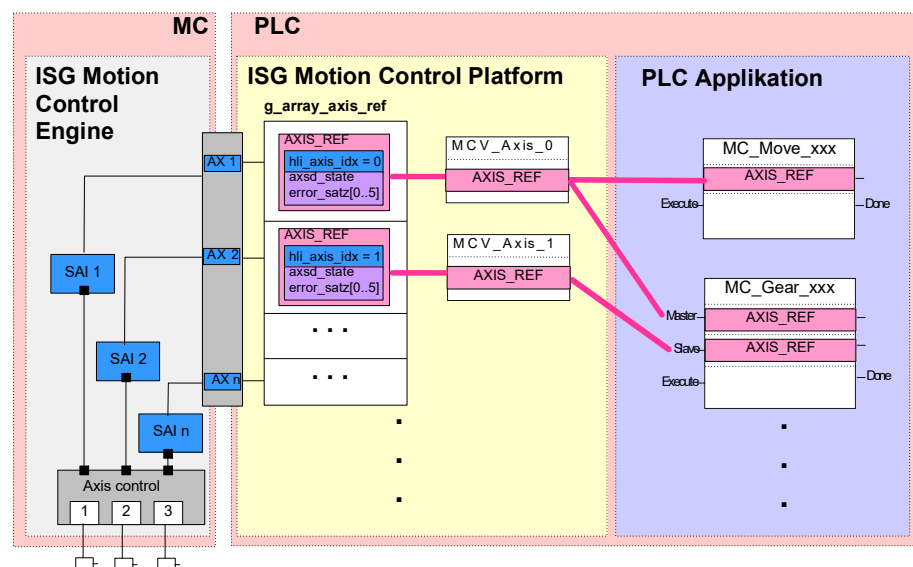


Abb. 4: Bereitstellung der `AXIS_REF` über den FB „`MCV_Axis`“



Programmierbeispiel

Deklaration und Aufruf in ST:

Deklaration in ST:

```
cam_in_1 : MC_CamIn;
```

Aufruf in ST:

```
cam_in_1 (Master:= g_array_axis_ref[0], Slave := g_array_axis_ref [1]);
```

2.2.3.3 Funktionsbaustein MCV_P1_PLATFORM

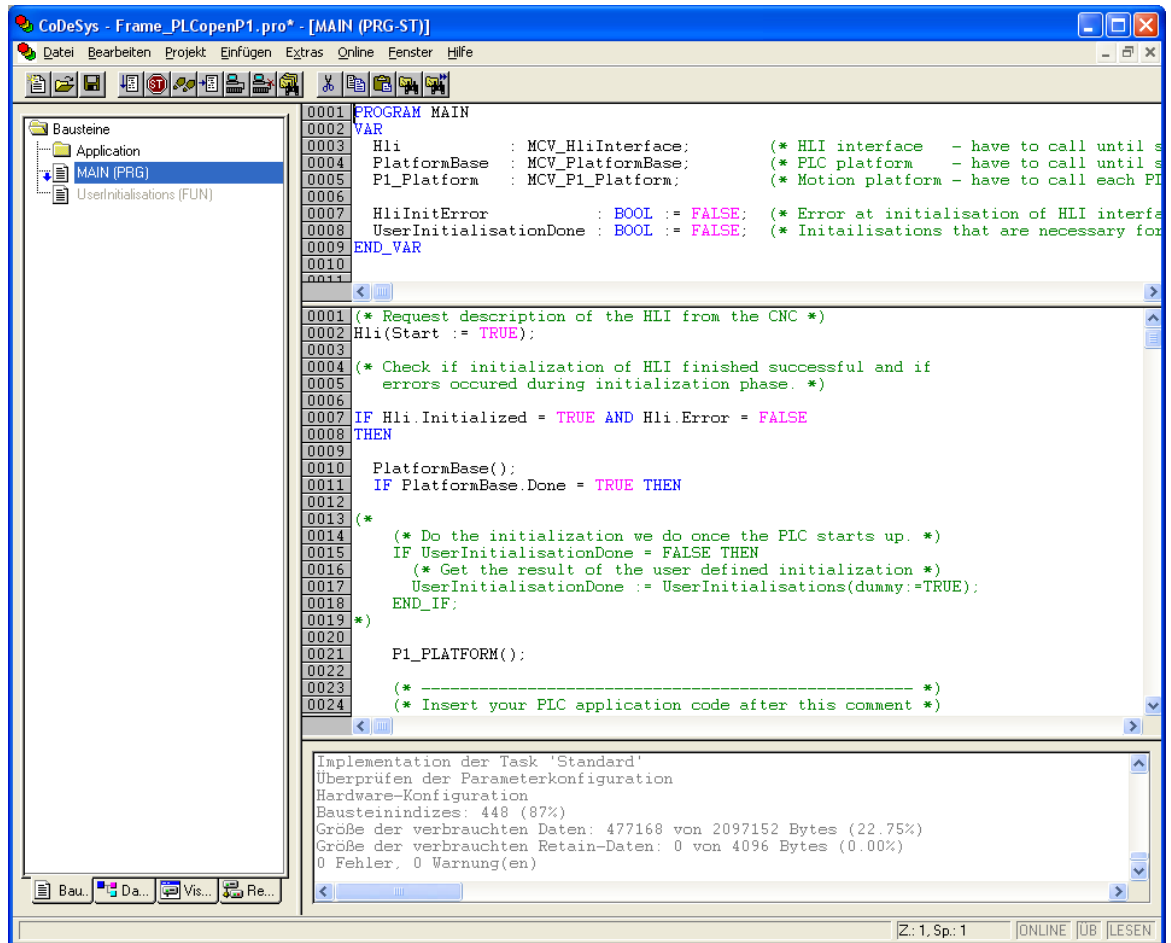
Für die MCP wurde folgende Festlegung getroffen:



Hinweis

Festlegung für die MCP:

- Die generischen FB „MCV_Axis“ der PLCopen-Achsen werden in der ISG-MCP instanziiert und sind im FB MCV_P1_PLATFORM implementiert.
- In jeder SPS-Applikation, die Bewegungsaufgaben unter Verwendung von FB nach den PLCopen-Spezifikationen Part 1 und 2 löst, muss zyklisch genau eine Instanz des FB MCV_P1_PLATFORM vor der Berechnung der FB zur Lösung der Bewegungsaufgabe durchgerechnet werden.
- Für die Instanzierung und den Aufruf aller PLCopen-FB, die zur Programmierung der Applikation (z.B. Bewegungsablauf) dienen, hat der Anwendungsprogrammierer in einem Applikationsprogramm zu sorgen.
- Vor dem erstmaligen Aufruf der Instanz von MCV_P1_PLATFORM muss das HLI (Schnittstelle zum MC) initialisiert sein und die Instanz des FB MCV_PlatformBase die erfolgreiche Initialisierung der MCP melden.



The screenshot shows the CoDeSys IDE with a project named 'Frame_PLCopenP1.pro'. The main window displays a ladder logic program for 'PROGRAM MAIN'. The program includes variable declarations for HLI interface, platform base, and motion platform, along with initialization logic for HLI and user-defined functions. A status window at the bottom provides implementation details for the 'Standard' task.

```

0001 PROGRAM MAIN
0002 VAR
0003   Hli           : MCV_HliInterface;      (* HLI interface - have to call until s
0004   PlatformBase : MCV_PlatformBase;      (* PLC platform - have to call until s
0005   P1_Platform  : MCV_P1_Platform;      (* Motion platform - have to call each PI
0006
0007   HliInitError   : BOOL := FALSE; (* Error at initialisation of HLI interfe
0008   UserInitialisationDone : BOOL := FALSE; (* Initialisations that are necessary for
0009 END_VAR
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024
0001 (* Request description of the HLI from the CNC *)
0002 Hli(Start := TRUE);
0003
0004 (* Check if initialization of HLI finished successful and if
0005    errors occurred during initialization phase. *)
0006
0007 IF Hli.Initialized = TRUE AND Hli_Error = FALSE
0008 THEN
0009
0010   PlatformBase();
0011   IF PlatformBase.Done = TRUE THEN
0012
0013      (*
0014      (* Do the initialization we do once the PLC starts up. *)
0015      IF UserInitialisationDone = FALSE THEN
0016      (* Get the result of the user defined initialization *)
0017      UserInitialisationDone := UserInitialisations(dummy:=TRUE);
0018      END_IF;
0019      *)
0020
0021   P1_PLATFORM();
0022
0023   (* ----- *)
0024   (* Insert your PLC application code after this comment *)
    
```

Implementation der Task 'Standard'
 Überprüfen der Parameterkonfiguration
 Hardware-Konfiguration
 Bausteinindizes: 448 (87%)
 Größe der verbrauchten Daten: 477168 von 2097152 Bytes (22.75%)
 Größe der verbrauchten Retain-Daten: 0 von 4096 Bytes (0.00%)
 0 Fehler, 0 Warnung(en)

Abb. 5: SPS-Basisprogramm für Motion-Applikationen in CoDeSys

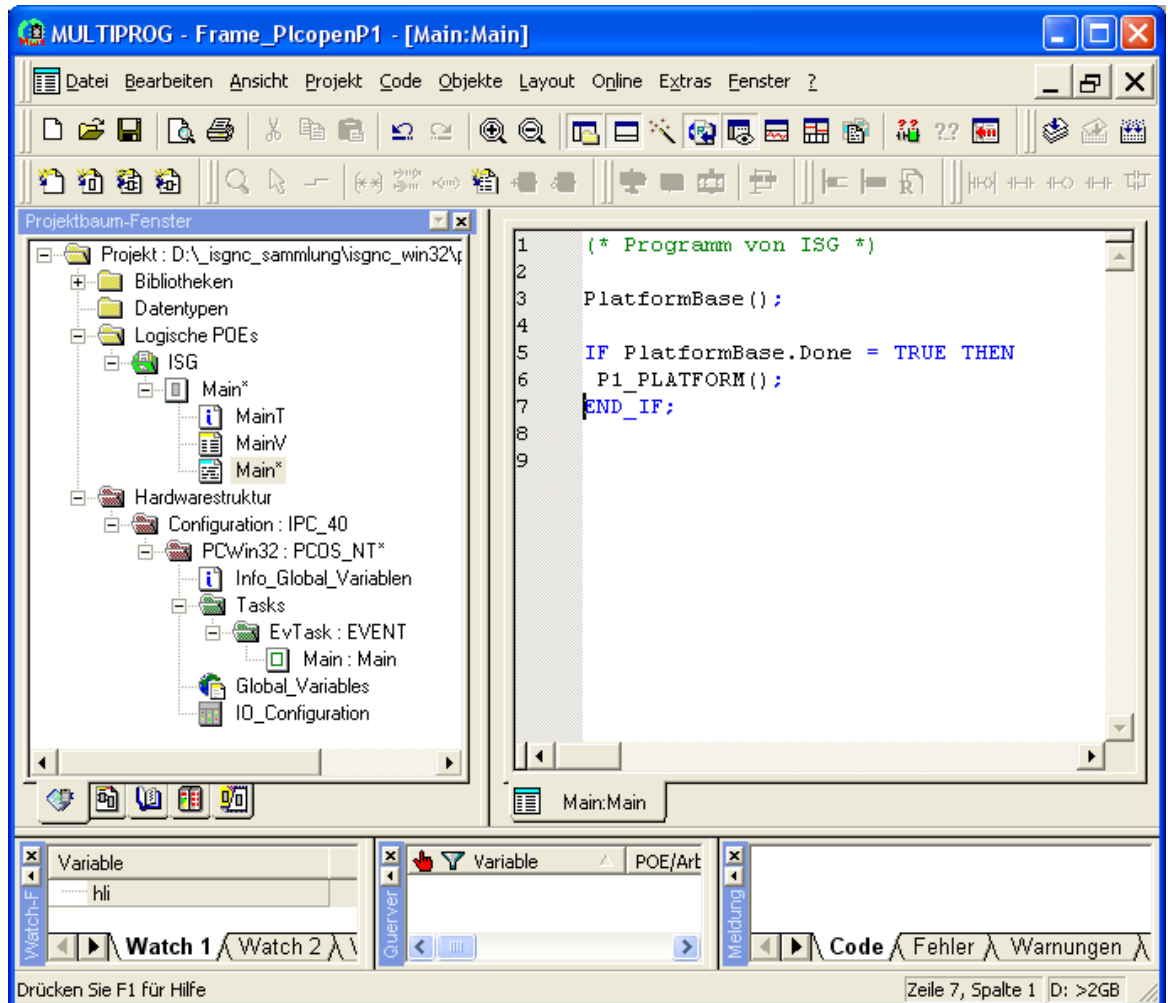


Abb. 6: Programm Main mit Instanz des FB MCV_P1_PLATFORM wird als erstes Programm in der Task aufgerufen

Im Funktionsblock MCV_P1_PLATFORM wird in der Initialisierungsphase jeder Achse eine Struktur AXIS_REF zugeordnet, die als Elemente des global definierten Feldes **g_array_axis_ref** vorliegen.

2.2.4

Achsgruppen-Bibliothek – PLCopen Part4

In der Anwenderbibliothek McpPLCopenP4.lib sind neben FB, die der PLCopen-Spezifikation Part 4 entsprechen, auch FB definiert, die zusätzliche Funktionalität abdecken und zur Realisierung einer Applikation eingesetzt werden müssen. Diese Bibliothek wird im weiteren Achsgruppen-Bibliothek genannt. Der Versionsumfang kann sich je nach verwendeter SPS-Plattform unterscheiden.

Das nachfolgende Bild zeigt den strukturellen Aufbau der Motion-Bibliothek. Anschließend werden die wesentlichen Elemente dieser Bibliothek näher erläutert.

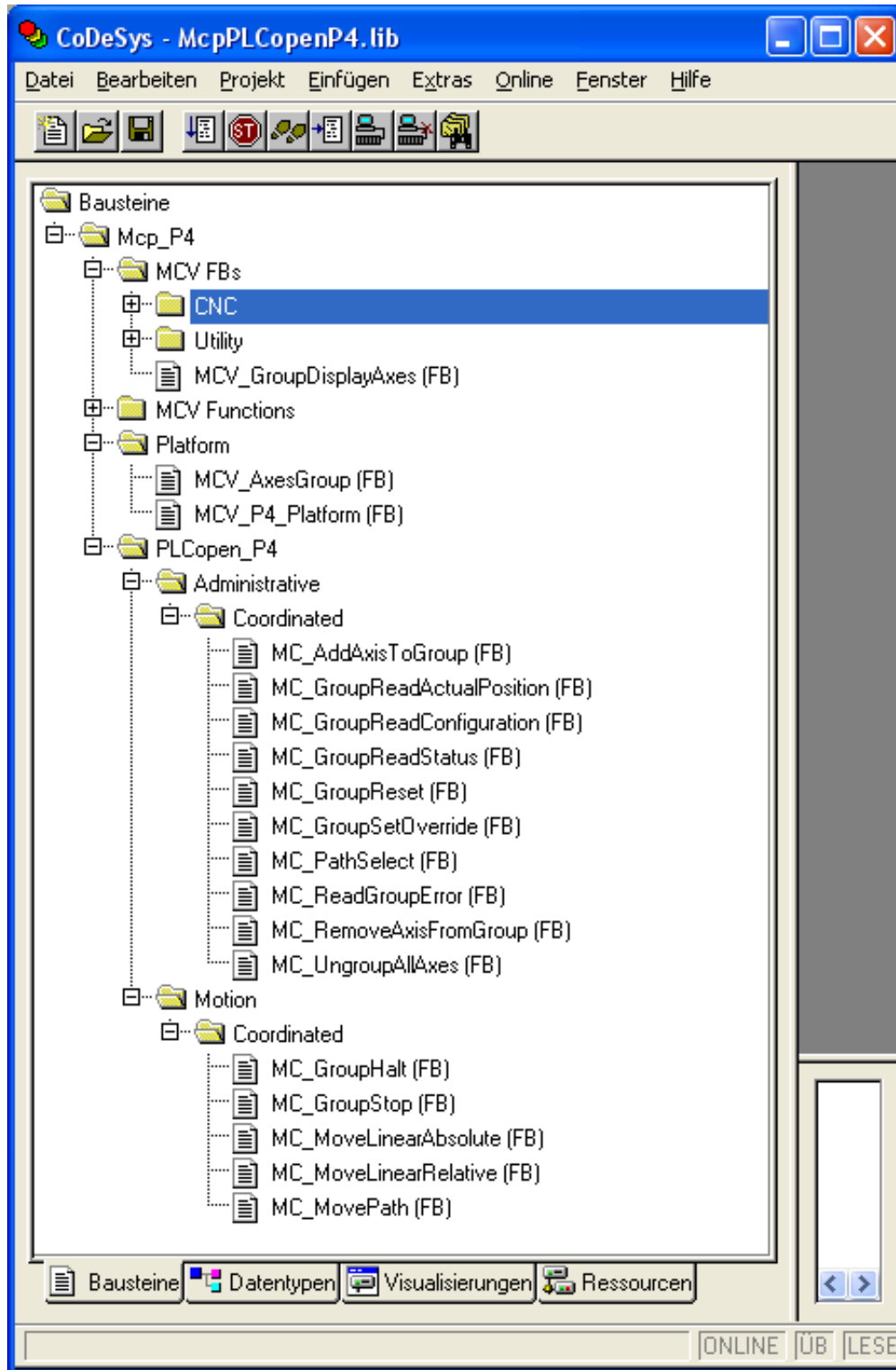


Abb. 7: Übersicht über die Motion Bibliothek McpPLCopenP4.lib in CoDeSys

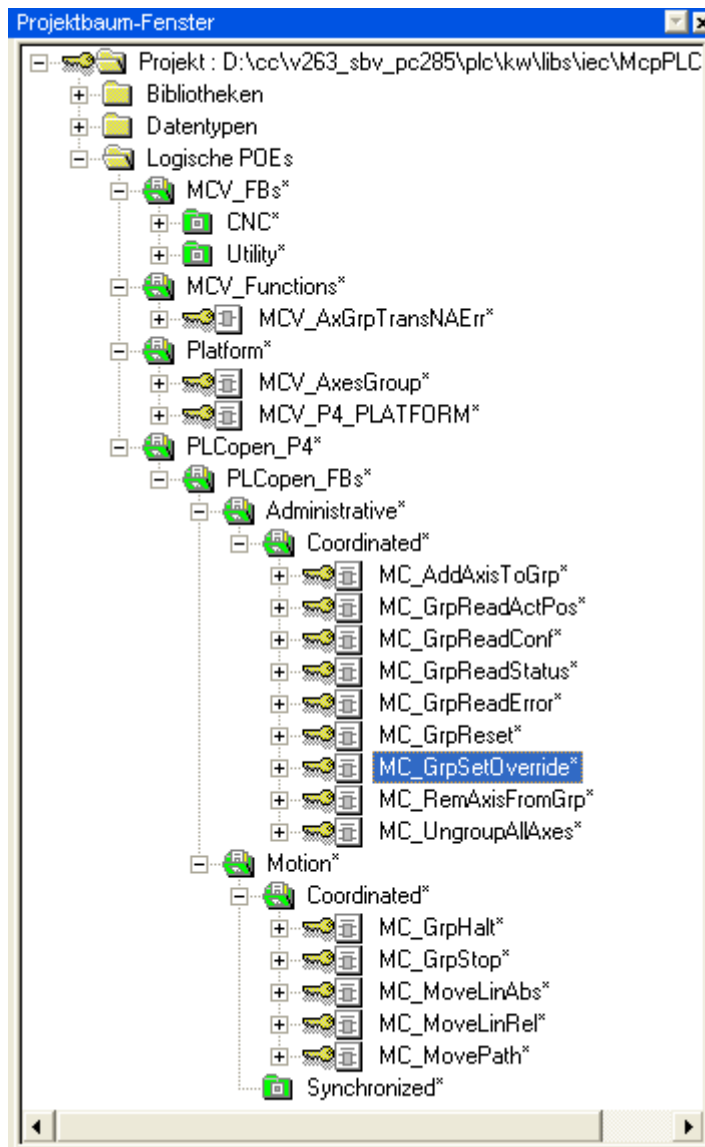


Abb. 8: Struktureller Aufbau der Motion Bibliothek McpPLCopenP4.zwt

2.2.4.1

Funktionsbaustein MCV_AxesGroup

Aktualisiert werden die Daten einer Struktur AXES_GROUP_REF durch den FB MCV_Axes-Group, der als Ein-/Ausgabeveriable eine Struktur AXES_GROUP_REF besitzt. Dieser FB übernimmt zusätzlich folgende Aufgaben:

- Anmeldung einer Achsgruppe an der MCE über das HLI. Dies geschieht durch Setzen des Flags „plc_present_w“ auf dem kanalspezifischen HLI-Bereich
- Bei der Initialisierung wird überprüft, ob einer Achsgruppe bereits Achsen zugeordnet sind. Ist dies der Fall, werden diese Achsen der SPS-internen Achsgruppenabbildung hinzugefügt, ohne dass eine MC_AddAxisToGroup beauftragt werden muss.
- Übernahme der Fehlermeldungen, die von der MCE kanalspezifisch gemeldet werden.

In jeder SPS-Applikation, die PLCopen-Part4 FB der ISG-MCP benutzt, muss für jede verwendete Achsgruppe eine Instanz dieses FB angelegt sein, und diesem eine Struktur AXES_GROUP_REF in der Form **gAxesGroupRef[i]** als VAR_IN_OUT-Parameter zugewiesen werden.

Um dies zu gewährleisten enthält die ISG-MCP den FB `MCV_P4_PLATFORM` [▶ 19], der in einem Programm einer SPS-Applikation aufgerufen werden muss. Damit ist gewährleistet, dass die Arbeitsdaten einer Achse in jedem SPS-Zyklus aktualisiert werden.

2.2.4.2 Funktionsbaustein `MCV_P4_PLATFORM`



Hinweis

Festlegung für die MCP

- a) Die generischen FB „`MCV_AxesGroup`“ der PLCopen-Achsgruppen werden in der ISG-MCP instanziiert und sind im FB `MCV_P4_PLATFORM` implementiert.
- b) In jeder SPS-Applikation, die Bewegungsaufgaben unter Verwendung von FB nach den PLCopen-Spezifikationen Part 4 löst, muss zyklisch genau eine Instanz des FB `MCV_P4_PLATFORM` vor der Berechnung der FB zur Lösung der Bewegungsaufgabe durchrechnen.
- c) Für die Instanzierung und den Aufruf aller PLCopen-FB, die zur Programmierung der Applikation (z.B. Bewegungsablauf) dienen, hat der Anwendungsprogrammierer in einem Applikationsprogramm zu sorgen.
- d) Vor dem erstmaligen Aufruf der Instanz von `MCV_P4_PLATFORM` muss das HLI (Schnittstelle zum MC) initialisiert sein und die Instanz des FB `MCV_PlatformBase` muss die erfolgreiche Initialisierung der MCP melden.

```

0001 PROGRAM MAIN
0002 VAR
0003   Hli           : MCV_HliInterface;      (* HLI interface - have to call
0004   PlatformBase : MCV_PlatformBase;     (* PLC platform - have to call
0005   P1_Platform   : MCV_P1_Platform;     (* Motion platform - have to call
0006   P4_Platform   : MCV_P4_Platform;     (* Motion platform - have to call
0007
0008   HliInitError   : BOOL := FALSE;      (* Error at initialisation of HLI
0009   UserInitialisationDone : BOOL := FALSE; (* Initialisations that are neces
0010 END_VAR
0011
0001 (* Request description of the HLI from the CNC *)
0002 Hli(Start := TRUE);
0003
0004 (* Check if initialization of HLI finished successful and if
0005    errors occurred during initialization phase. *)
0006
0007 IF Hli.Initialized = TRUE AND Hli.Error = FALSE
0008 THEN
0009
0010   PlatformBase();
0011   IF PlatformBase.Done = TRUE THEN
0012
0013   (*
0014      (* Do the initialization we do once the PLC starts up. *)
0015      IF UserInitialisationDone = FALSE THEN
0016         (* Get the result of the user defined initialization *)
0017         UserInitialisationDone := UserInitialisations(dummy:=TRUE);
0018      END_IF;
0019   *)
0020
0021   P1_PLATFORM();
0022   P4_PLATFORM();
0023
0024   (* ----- *)
0025   (* Insert your PLC application code after this comment *)
0026   (* ----- *)
0027
    
```

Lade Bibliothek 'C:\isgnc\plc\McpPLCopenP4.lib'

Z: 21, Sp.: 18 ONLINE ÜB LESEN

Abb. 9: SPS-Basisprogramm für Achsgruppen-Applikationen in CoDeSys-Umgebung

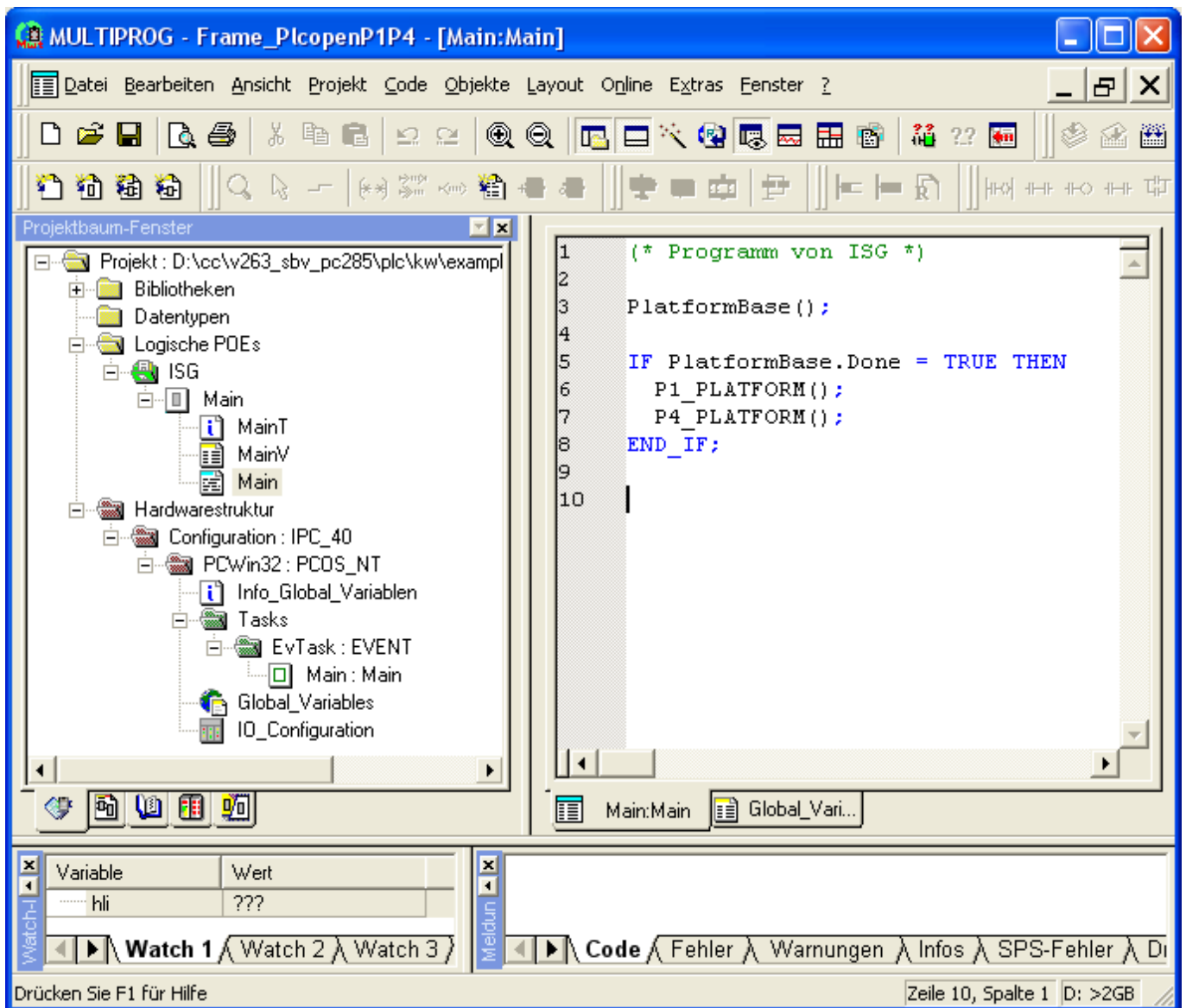


Abb. 10: Programm Main mit Instanz des FB MCV_P4_PLATFORM in Multiprog-Entwicklungsumgebung

Im Funktionsblock MCV_P4_PLATFORM wird in der Initialisierungsphase jeder Achsgruppe eine Struktur AXES_GROUP_REF zugeordnet, die als Elemente des global definierten Feldes **gAxesGroupRef** vorliegen.

2.2.4.3 PLCopen Funktionsbausteine

In der PLCopen-Spezifikation Part 4 werden die dort definierten FB entsprechend ihrer Verwendung in administrative und bewegungsbezogene FB unterteilt.

Innerhalb dieser beiden Bereiche wird weiter unterschieden, ob ein FB sich nur auf die Achsgruppe bezieht (coordinated) oder ob durch den FB eine Funktionalität im Zusammenspiel mit Komponenten außerhalb der Achsgruppe beauftragt (synchronized) wird.

Die nachfolgende Tabelle ist entsprechend organisiert und zeigt die Funktionsblöcke nach PLCopen-Spezifikation Part 4.



Hinweis

Die mit einem * versehenen FB sind nicht in der Motionbibliothek Part4 implementiert. Jedoch kann es in den Bibliotheken FB geben, die eine ähnliche Funktionalität besitzen, aber durch den Steuerungshersteller spezifiziert wurden.

Einteilung der PLCopen-FB Part4 in administrative und bewegungsbezogene FB

| Administrative | Motion | |
|---------------------------------|--------------------------|-----------------------|
| | Coordinated | Synchronized |
| MC_AddAxisToGroup | MC_GroupHome* | MC_SyncAxisToGroup* |
| MC_RemoveAxisFromGroup | MC_GroupStop | MC_SyncGroupToAxis* |
| MC_UngroupAllAxes | MC_GroupHalt | MC_TrackConveyorBelt* |
| MC_GroupReadConfiguration | MC_GroupInterrupt* | MC_TrackRotaryTable* |
| MC_GroupEnable* | MC_GroupContinue* | |
| MC_GroupDisable | MC_MoveLinearAbsolute | |
| MC_SetKinTransform* | MC_MoveLinearRelative | |
| MC_SetCartesianTransform* | MC_MoveCircularAbsolute* | |
| MC_SetCoordinateTransform* | MC_MoveCircularRelative* | |
| MC_ReadKinTransform* | MC_MoveDirectAbsolute* | |
| MC_ReadCartesianTransform* | MC_MoveDirectRelative* | |
| MC_ReadCoordinateTransform* | MC_MovePath | |
| MC_GroupSetPosition* | | |
| MC_GroupReadActualPosition | | |
| MC_GroupReadActualVelocity* | | |
| MC_GroupReadActualAcceleration* | | |
| MC_GroupReadStatus | | |
| MC_GroupReadError | | |
| MC_GrpReset | | |
| MC_PathSelect | | |
| MC_GroupSetOverride | | |
| MC_SetDynCoordTransform* | | |

2.2.5 Globale Variablen

Je nach SPS-Entwicklungssystem kann es erforderlich sein, dass globale Daten, die in den Anwenderbibliotheken verwendet werden, in der SPS-Applikation definiert werden müssen.

Zur Verwendung der ISG-MCP müssen in einem SPS-Projekt für eine Ressource die unter der Gruppe ISG MCP aufgeführten globalen Variablen definiert sein:

| Name | Typ | Verwendung | B | Adresse | Anfangsw... |
|-------------------------------|----------------------|------------|---|-----------|-------------|
| ISG MCP | | | | | |
| MAX_RESET_RETRIALS | UDINT | VAR_GLOBAL | | | 50000 |
| MAX_RESET_WAIT_CYCLES | UDINT | VAR_GLOBAL | | | 1000000 |
| MAX_RETRIALS | UDINT | VAR_GLOBAL | | | 0 |
| g_order_id | UDINT | VAR_GLOBAL | | | 1 |
| g_axis_idx_offset | INT | VAR_GLOBAL | ! | | 0 |
| g_array_axis_ref | ARRAY_AXIS_REF | VAR_GLOBAL | | | |
| gAxesGroupRef | ARRAY_AXES_GROUP_REF | VAR_GLOBAL | | | |
| hli | HIGH_LEVEL_INTERFACE | VAR_GLOBAL | | %MB3.0 | |
| MAX_USED_INSTANCES | INT | VAR_GLOBAL | | | 3 |
| NR_CYCLES_CHK_MC_RUNS | UINT | VAR_GLOBAL | | | 10 |
| NR_MAX_PLC_CYCLES_CHK_MC_RUNS | UINT | VAR_GLOBAL | | | 1000 |
| System Variables | | | | | |
| PLCMODE_ON | BOOL | VAR_GLOBAL | | %MX 1.0.0 | |
| PLCMODE_RUN | BOOL | VAR_GLOBAL | | %MX 1.0.1 | |
| PLCMODE_STOP | BOOL | VAR_GLOBAL | | %MX 1.0.2 | |
| PLCMODE_HALT | BOOL | VAR_GLOBAL | | %MX 1.0.3 | |
| PLCDEBUG_BPSET | BOOL | VAR_GLOBAL | | %MX 1.1.4 | |
| PLCDEBUG_FORCE | BOOL | VAR_GLOBAL | | %MX 1.2.0 | |
| PLCDEBUG_POWERFLOW | BOOL | VAR_GLOBAL | | %MX 1.2.3 | |
| PLC_TICKS_PER_SEC | INT | VAR_GLOBAL | | %MW 1.44 | |
| PLC_SYS_TICK_CNT | DINT | VAR_GLOBAL | | %MD 1.52 | |

Abb. 11: Erforderliche globale Variablen zur Verwendung der ISG-MCP im Multiprog-Entwicklungsumgebung

2.3 Sicherheitskonzept, Einhaltung der EN775

2.3.1 Grundsätzliches zum softwaretechnischen Sicherheitskonzept

Softwaretechnische Sicherheitsfunktionen im obigen Sinne sind grundsätzlich innerhalb der ISG-MCE bzw. der MCP realisiert.

Der sichere Zustand ist grundsätzlich der Default-Zustand, d.h. dieser sichere Default-Zustand kann über speziell dafür vorgesehene, sicherheitsrelevante Funktionsbausteine nur *ausgeschaltet* werden.

Die erforderlichen Funktionsbausteine zur Deaktivierung sind **nicht** Bestandteil der ISG Lieferung.

Da es sich beim HLI um eine speichergekoppelte Schnittstelle handelt, ist für die Kommunikation sicherheitsrelevanter Kommandos eine Realisierung gewählt worden, die sicherstellt, dass das einmalige Ausschalten einer sicherheitsrelevanten Funktion, das durch Setzen einer Speicherstelle beauftragt wird, nicht anstehen bleibt, falls die SPS den entsprechenden Sicherheitsfunktionsbaustein nicht mehr aufruft.

2.3.2 Ist-Geschwindigkeitsüberwachung

In der ISG-MCE ist eine istwertseitige Geschwindigkeitsüberwachung realisiert.

Der Grenzwert für die Geschwindigkeitsüberwachung ist für Linearachsen fest auf 250 mm/s eingekompiliert.

Für Rundachsen oder Spindeln ist eine Parametrierbarkeit wegen des Bauteildurchmessers erforderlich. Deshalb wird der Grenzwert per Getriebeparameter „vb_monitor“ (P-AXIS-00311) in der ACHS_MDS-Liste eingestellt. Ist der Parameter in der Liste nicht enthalten oder „0“, so wird die Ist-Geschwindigkeit auf „vb_not_referenced“ (P-AXIS-00268) überwacht.

- Im SAI wird die Fehlermeldung 60241 (P-ERR-60241) „Programmierte Geschwindigkeit überschreitet Überwachungsgrenze“ ausgegeben, falls ein Fahrauftrag mit höherer Geschwindigkeit beauftragt wird.
- Die Geschwindigkeitsüberwachung ist in der ISG-MCE per Default aktiv und kann mit einem speziellen Sicherheitsfunktionsbaustein, der zyklisch aufzurufen ist, deaktiviert werden.
- Die Ist-Geschwindigkeit wird in der BF LR zyklisch im Systemtakt überwacht. Bei Überschreitung der zulässigen Ist-Geschwindigkeit wird die Fehlermeldung 70225 (P-ERR-70225) „Max. Ist-Geschwindigkeit während aktiver Geschw.-Überwachung überschritten“ ausgegeben und die Achse durch Schließen der Bremse und Wegnahme der Reglerfreigabe gestoppt.

2.3.3 Bidirektionale Kongruenzprüfung der HLI Speicher Schnittstelle

Die ISG FB beauftragen die ISG-MCE über das sogenannte HLI. Da es sich um eine speichergekoppelte Schnittstelle handelt, ist eine speicherdeckungsgleiche (=kongruente) Nachdefinition dieser Schnittstelle in der SPS die Voraussetzung für eine einwandfreie Funktion des Gesamtsystems. Deshalb ist innerhalb der MCP Bibliothek eine Kongruenzprüfung realisiert, die sicherstellt, dass die HLI-Schnittstelle nicht mit einer fehlerhaften HLI-Nachbildung betrieben wird.

2.3.4 Priorität des FB MC_Stop

Das HLI der ISG-MCE hat **pro Achse genau eine** Beauftragungs- und Quittierungsschnittstelle für MC-Aufträge. Beauftragungs- und Quittierungsschnittstelle sind jeweils als Verbrauchsdatum realisiert. Daraus ergibt sich, dass pro SPS-Zyklus maximal 1 FB-Auftrag abgesetzt werden kann. Wird innerhalb eines SPS-Zyklus ein zweiter FB angetriggert, so kann dieser nicht durchgesetzt werden. Falls der (innerhalb eines SPS-Zyklus) nachfolgende Auftrag ein MC_Stop ist, wird der bereits im Beauftragungsspeicher stehende durch diesen MC_Stop überschrieben, so dass der MC_Stop immer die höchste Priorität hat.

2.3.5 Geschwindigkeitsüberwachung während aktiver Drehmomentbegrenzung

Für die Ist-Geschwindigkeitsüberwachung während aktiver Drehmomentbegrenzung soll eine zusätzliche Überwachungsfunktion realisiert werden. Diese ist im ACHS_MDS-Parameter „vb_torq_limit_max“ (P-AXIS-00314) parametrierbar.

- Die Ist-Geschwindigkeit während aktiver Drehmomentbegrenzung wird in der BF LR ebenfalls zyklisch überwacht. Bei Überschreitung der zulässigen Ist-Geschwindigkeit wird die Fehlermeldung „Max. Ist-Geschwindigkeit während aktiver Drehmomentbegrenzung überschritten“ (P-ERR-70220) ausgegeben und die Achse durch Schließen der Bremse und Wegnahme der Reglerfreigabe gestoppt.

2.3.6 Verhindern von unbeabsichtigten MCFB-Bewegungen im T1-Mode

Einrichtbetriebsarten:

- T1-Mode = Einrichtbetrieb mit reduzierter Geschwindigkeit
- T2-Mode = Einrichtbetrieb ohne reduzierter Geschwindigkeit

Die ISG-MCE hat keine direkte Kenntnis über die Betriebsart der Anlage. Der SPS-Programmierer, dem diese Informationen zur Verfügung stehen, kann mit dem Sicherheitsfunktionsbaustein den sicheren Zustand (=Stillstand) verlassen. Für die Einhaltung der EN775 besteht folgende Forderung:

Im kritischen Zustand, d.h. ein Bediener befindet sich im Gefahrenbereich der Anlage und es herrscht Betriebsart T1 oder T2, dürfen Bewegungen nur aufgrund bewusster Bedienhandlungen (bei einem Roboter: Zustimmungstaste + START Taste) beauftragt werden können.

In diesem Zustand muss das Loslassen der Starttaste nur zum Feedhold, nicht aber zum Auftragsabbruch führen.

Bei einem Betriebsartwechsel von Automatik auf T1 oder T2 fallen i.d.R. ohnehin die Freigaben der Antriebe ab.

Wird die Zustimmungstaste gedrückt, dann werden PLCopen Aufträge gepuffert, die Bewegung beginnt jedoch noch nicht.

Der SPS-Programmierer kann dann das Drücken der Starttaste zur Versorgung des Sicherheitsfunktionsbausteins verwenden, um die gewohnte Funktionalität der Starttaste zu gewährleisten.

2.3.7 Sicherheitskleingeschwindigkeit für nicht referenzierte Achsen

Falls eine Achse noch nicht referenziert/justiert ist, bzw. den Referenzpunkt/Justage verloren hat, ist damit der Maßbezug zu einem maschinenfesten Punkt ebenfalls verloren und somit kann keine Softwareendschalter-Überwachung stattfinden.

In diesem Zustand können die Achsen deshalb nicht absolut positioniert werden. Es sind ausschließlich MC_MoveRelative und MC_MoveVelocity möglich. Diese relativen Positionierarten werden mit reduzierter Geschwindigkeit ausgeführt, die im ACHS_MDS Parameter

```
getriebe[0].vb_not_referenced
```

(P-AXIS-00268) konfiguriert werden kann.

2.4 Realisierungsdetails innerhalb der ISG-MCP

In diesem Kapitel sollen diejenigen Aspekte der ISG-MCP Realisierung diskutiert werden, bei denen das Verhalten der FB allein mit der PLCopen-Spezifikation 1.0 nicht genau vorhersagbar sind.

Gründe hierfür können sein:

- Unzureichende Festlegungen in der PLCopen-Spezifikation 1.0
- Eigenschaften der Motion Control Engine

2.4.1 Hochlauf

Beim Kaltstart des SPS-Programms werden verschiedene Initialisierungen der ISG-MCP durchlaufen.

Während dieser Initialisierungsphase befindet sich der AXSD im Zustand 0 (INIT_STATE).

Sämtliche FB der MCP verhalten sich während dieser Phase passiv, sprich sie greifen nicht schreibend auf das HLI zu. Falls ein FB einen Auftrag absetzt, bevor die Initialisierungsphase beendet ist, z.B. durch die Initialisierung eines „Enable“ Inputs mit „TRUE“, meldet zeigt der FB den Fehler ERR_PO_AX_TNA_INIT_STATE (P-ERR-44013) an.

2.4.2 Grundsätzliches zur Arbeitsweise der PLCopen FB

Die PLCopen FB besitzen intern keine Funktionalität zur Interpolation. Vielmehr dienen diese lediglich dazu Bewegungsaufträge an die Motion Control Engine abzusetzen und entsprechende Quittierungen entgegen zu nehmen.

2.4.3 Realisierung der FB

Die PLCopen FB bilden intern folgendes Zustandsdiagramm ab.

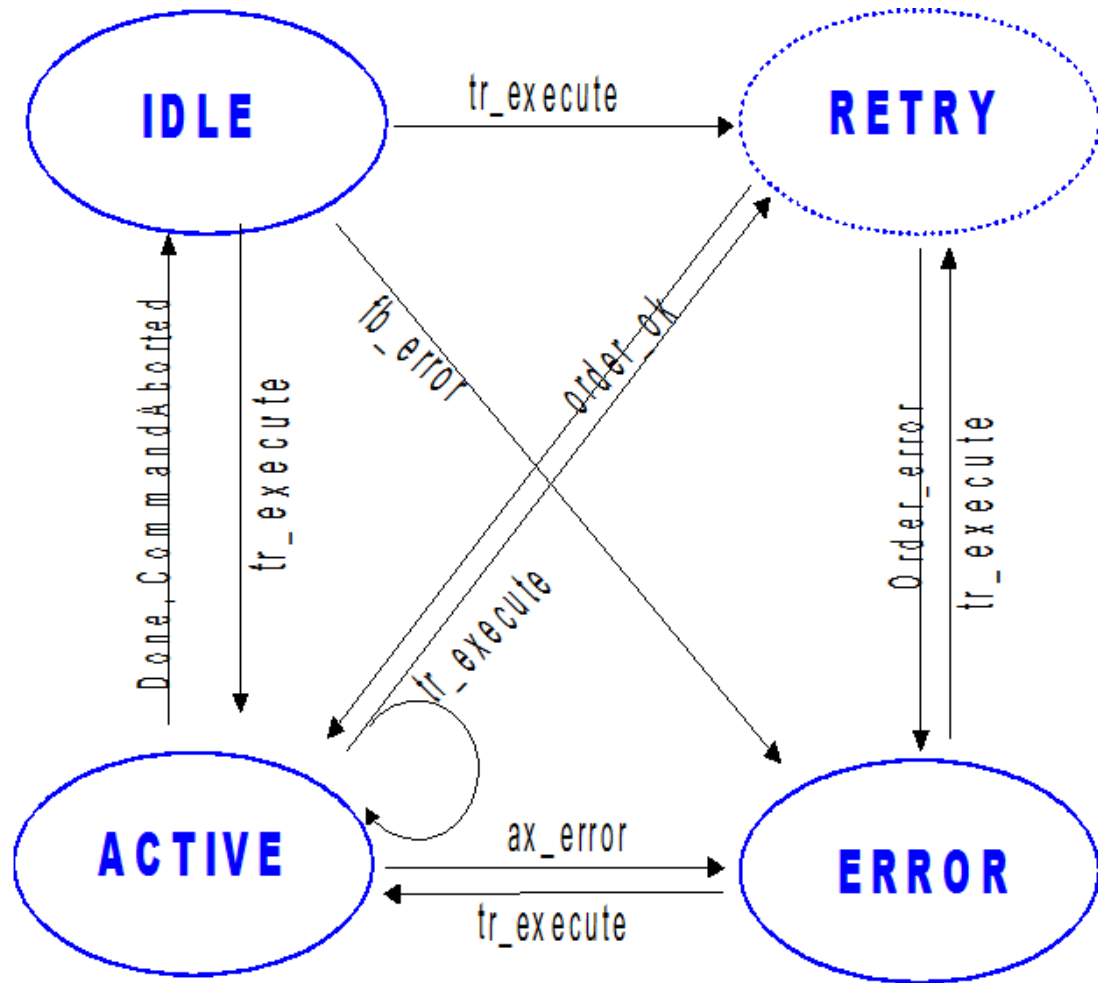


Abb. 12: Zustände innerhalb eines FB.

Der nachfolgend aufgezeigte Zustandsautomat in IEC 61131-3 Structured Text zeigt das Gerüst für die Realisierung der Beauftragung innerhalb eines FB. Die einzelnen Aktionen sind dabei im Pseudo-Code gehalten.

```
(*=====*)
(*      Zustandsverteiler für die HLI Beauftragung      *)
(*=====*)
CASE fb_state OF
(*=====*)
(*=====*)
FB_IDLE,
FB_ERROR: (* *)
IF ( tr_execute.Q = TRUE OR retry ) THEN
  (* checking of the FB's input parameters *)
  (* check whether transition is allowed *)
  (* try to send the MC order. *)
  (* IF (sendorder = OK) THEN fb_state := FB_ACTIVE; END_IF *)
END_IF

(*=====*)
(*=====*)
ACTIVE: (* *)
IF ( tr_execute.Q = TRUE OR retry) THEN
  (* check whether FB's ax_ref connection has changed since idle state*)
  (* checking of the FB's input parameters *)
  (* check whether transition is allowed *)
  (* try to send send the MC order. *)
END_IF
(* collection of Acknowledge *)
(* IF (Acknowledge = OK) THEN fb_state := FB_IDLE; END_IF *)

(*=====*)
(*=====*)
ELSE
(*// default: Unerlaubter Zustand *)
END_CASE;
```

Die FBs kennen keine Reset-Transition, vielmehr wird nach einem vorangegangenen FB_ERROR nach erneutem Antriggern des FBs einfach versucht die neue Beauftragung durchzusetzen. Deshalb unterscheidet der Zustandsverteiler für die HLI Beauftragung nicht zwischen FB_IDLE und FB_ERROR.

Der Zustand FB_RETRY tritt nicht als expliziter Zustand im Zustandsverteiler auf sondern wird in den Zuständen FB_IDLE, FB_ERROR und FB_ACTIVE jeweils in einer Variablen gehalten für den Fall, dass der FB im entsprechenden Zustand nicht auf Antrieb eine Beauftragung durchsetzen kann.

Die eindeutige Auftragsnummer wird als globales SPS Datum gehalten und von den FB **nur zum Zeitpunkt nach einer erfolgreichen Beauftragung aus den Zuständen FB_IDLE und FB_ERROR heraus** inkrementiert, und in seinen Instanzdaten vermerkt.

Bei einer Beauftragung aus dem Zustand FB_ACTIVE heraus wird ein neuer Auftrag mit derselben Auftragsnummer an den MC geschickt und ein FB-interner Zähler, wie viele Aufträge einer Auftragsnummer unterwegs sind, inkrementiert.

Diese Methode erspart eine aufwendige Verwaltung der Auftragsnummern die beauftrag wurden und der zugehörigen eingegangenen Quittierungen.

Wenn ein FB angetriggert wird, überprüft er, ob der Zustandsübergang, der durch die Ausführung des FB im FBSD ausgelöst **würde** (konjunktiv!), im momentanen Zustand des FBSD überhaupt erlaubt ist („transition allowed“). Falls dies nicht der Fall ist, wird der Auftrag gar nicht erst abgesetzt sondern der FB meldet einen Beauftragungsfehler (= Fehler auf FB Ebene). Der Achszustand ändert sich in einem solchen Fall niemals, da ja überhaupt gar kein Auftrag abgesetzt wurde. Näheres dazu finden Sie im folgenden Kapitel.

2.4.4 Interaktion der FB mit dem FBSD, Fehlerhandling

Die in der PLCopen Spezifikation beschriebene Zustandsmaschine „FB state behaviour“ bezieht sich immer auf eine Achse. Deshalb macht es Sinn diesen Achszustand in den achsspezifischen Arbeitsdaten der SPS zu halten und allen FB jeweils über die AXIS_REF-Struktur zur Verfügung zu stellen. Ebenso wird der Index der korrespondierenden Ax-HLI Schnittstelle in der AXIS_REF-Struktur vermerkt.

Da der Achszustand in den achsspezifischen Arbeitsdaten der SPS gehalten wird, ist klar, dass dieser FBSD zunächst nur den Beauftragungszustand enthalten kann.

Dies wird auch durch folgende PLCopen Regel zum Ausdruck gebracht:

```
The axis is always in one of the defined state (see diagram below). Any motion command is a transition that changes the state of the axis ...
```

Bei der Beauftragung eines FB muss dieser anhand des aktuellen Zustands des FBSD die Zulässigkeit der Beauftragung überprüfen.

2.4.4.1 Fehlerbehandlung auf FBSD Ebene

An dieser Stelle könnte man nun zu der Ansicht gelangen, dass der FB bei einer unzulässigen Beauftragung den FBSD in einen Fehlerzustand versetzt, weil es sich ja um einen Beauftragungszustandsautomaten handelt. Dem ist jedoch nach PLCopen Spezifikation **nicht** so, denn für den FBSD gilt:

```
Note 3: The transition Error refers to errors from the axis and axis control, and not from the Function Block instances. These axis errors may also be reflected in the output of the Function Blocks 'FB instances errors'.
```

Das bedeutet: ein Bewegungs-FB versetzt den FBSD niemals in den Zustand ERROR, sondern nur ein Fehler, der vom Motion-Controller gemeldet wird. Dazu ist es erforderlich, dass ein achsspezifischer Handler-Prozess die Fehlermeldungen vom HLI entnimmt und in den achsspezifischen Arbeitsdaten, sprich im FBSD, ablegt. Die einzelnen FB sehen den Fehlerzustand der Achse dann über ihre AXIS_REF.

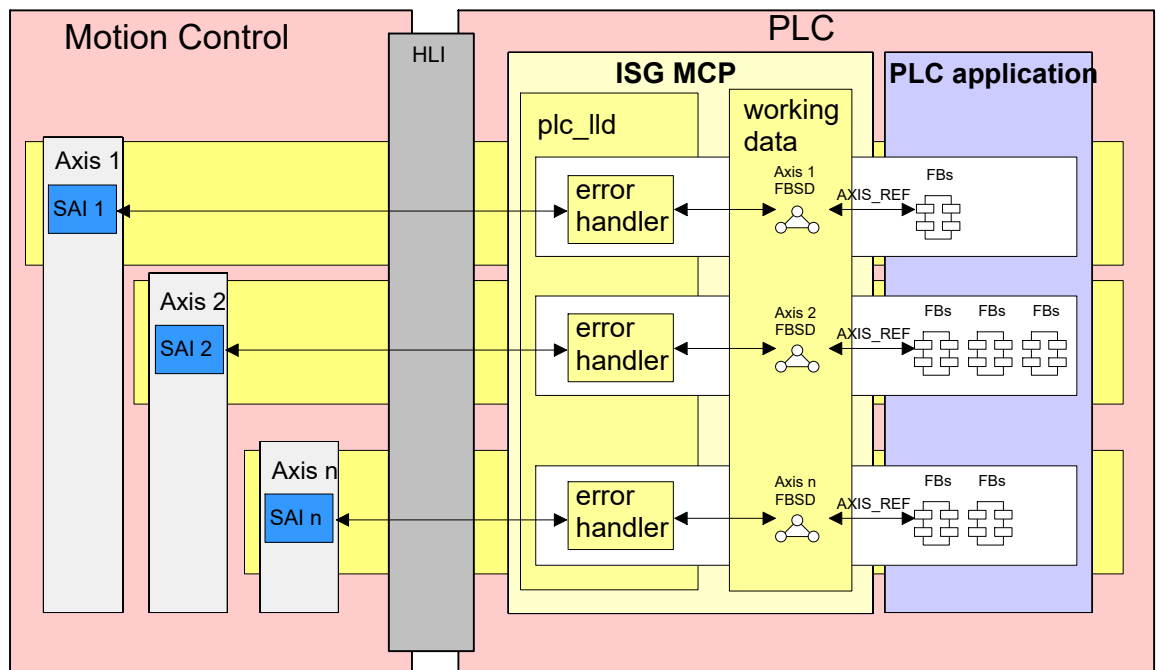


Abb. 13: Error-handler versorgt die achsspezifischen FBSD Arbeitsdaten

2.4.4.2 Fehlerbehandlung auf FB Ebene

Die Fehlerausgänge der einzelnen FB beziehen sich auf Fehler, die im Rahmen der Beauftragung einer FB Instanz aufgetreten sind, nicht auf die Fehler der Achse. Deshalb versetzt eine fehlerhafte Beauftragung nur den beauftragten FB in einen Fehlerzustand. Der fehlerhafte Auftrag selbst wird vom FB gar nicht erst beauftragt, so dass die Achse (FBSD) erst gar nicht in einen Fehler gebracht wird. Andere FB Instanzen, die auf derselben Achse sitzen, merken davon also nichts.

2.4.4.3 Definierte Fehler auf FB-Ebene

Die spezifischen Fehler, die in den einzelnen FB auftreten können, sind in der Diagnoseanleitung (DIAG) beschrieben.

2.4.4.4 Achsfehler aus dem Motion Controller

Der Motion Controller nutzt die achsspezifischen Schnittstellen des HLI, um Meldungen über eine Achse für die SPS bereitzustellen. Die Informationen werden dabei über eine Datenstruktur des Typs HLI_ERROR_SATZ ausgetauscht.

```

TYPE
  HLI_ERROR_SATZ :
  STRUCT
    error_id          : UDINT;  (*Fehlernummer*)
                                (*Systemzeit beim Auftreten des Fehlers*)
    fb_zeitangabe    : HLI_FB_ZEITANGABE;
    bf_type          : WORD;    (*BF-Typ*)
    behebangs_klasse : WORD;    (*Fehlerbehebungs-klasse*)
    reaktions_klasse : WORD;    (*Fehlerreaktions-klasse*)
    reserved         : WORD;
  END_STRUCT;
END_TYPE
    
```

Diesen achsspezifischen HLI-Bereich überprüfen Instanzen des FB MCV_Axis in jedem SPS-Zyklus, da diese im Programm MCV_P1_PLATTFORM instanziiert sind und entsprechend den Erläuterungen in Kapitel 1.2.3.3 [► 13]. dieses Programm als erstes in die SPS-Task eingebunden werden muss. Die MCV_Axis-Instanz entnimmt **jede** neu aufgetretene Meldung und überträgt diese in die AXIS_REF-Struktur der zugeordneten Achse, die ein Feld für 6 Datenstrukturen des Typs HLI_ERROR_SATZ enthält.

Ist eine Meldung als Fehler klassifiziert setzt die MCV_Axis-Instanz den aktuellen Zustand des Achszustandsdiagramm (AXSD) auf **ERROR_STOP**.



Hinweis

Fehlermeldungen sind alle diejenigen Meldungen, bei denen der Wert der Variablen **behebungs_klasse > 0** ist.

Ist der Wert von behebungs_klasse = 0, ist die Meldung eine Warnung.

Der Zustand **ERROR_STOP** wird von den anderen PLCopen-FB-Instanzen detektiert, denen dieselbe Achse zugeordnet ist. In der Folge setzen diese ihre Ausgangsvariable „Error“ auf TRUE und an der Ausgangsvariable „ErrorID“ wird der Wert **1** (ERR_PLC_AX_MC, siehe P-ERR-40001) angezeigt.

2.4.5 Versionierung

Die Versionsinformation wird entsprechend den durch das SPS-Laufzeitsystem vorgegebenen Rahmenbedingungen und der Anforderung auch ohne Entwicklungssystem oder laufende SPS-Applikation verfügbar zu sein, zum einen im Namen der SPS-Bibliotheken abgelegt und zum anderen sind Funktionsbausteine implementiert, die in der Applikation die Überprüfung der Versionsstände der einzelnen Bestandteile (SPS-Bibliotheken, HLI-Definitionen auf MC- bzw. SPS-Seite) der ISG-MCP durchführen.

2.4.5.1 Versionsüberprüfung durch FB

Die Implementierung der Versionsüberwachung durch FB beruht auf dem Prinzip, dass jede Komponente der ISG-MCP die Version derjenigen Schnittstellen und SPS-Bibliotheken überprüft, von denen sie selbst direkt abhängt. Deshalb sind in der **HLI-Bibliothek** und in der **Motion-Bibliothek** einige FB implementiert. Diese führen die Überprüfung der Komponenten durch, oder sie liefern die Version der Bibliothek.

Generell muss sich der Anwender nicht um diese Überprüfung kümmern, da diese bereits im Hochlauf der SPS-Applikation durch die Instanzen des FB MCV_Axis durchgeführt wird. Diese MCV_Axis-Instanzen sind im Programm MCV_P1_PLATTFORM instanziiert, welches zyklisch aufgerufen wird.

Treten Inkonsistenzen auf, werden die anderen FB-Instanzen der Applikation darüber in Kenntnis gesetzt und diese zeigen an ihrem Ausgang „Error“ FALSE und am Ausgang „ErrorID“ eine spezifische Fehlerkennung.

2.4.6 Weitere allgemeine Systemeigenschaften

- Endschalteüberwachung (nicht wirksam bei Moduloachsen): wird eine Fahrwegsgrenze erkannt, so wird mit den Werten der Beschleunigung im Eilgang gebremst (Grenzbeschleunigung an der Stromgrenze, siehe P-AXIS-00004 bzw. P-AXIS-00005, P-AXIS-00006), nicht mit der Standard-Beschleunigung.
- Zustand Discrete Motion (MC_MoveRelative) bricht Zustand „Continuous Motion“ mit hoher Drehzahl ab, so dass der Bremsweg mehr als einen Modulobereich beträgt. Das zu erwartende Verhalten geht aus der PLCopen-Spezifikation nicht hervor. Die ISG-MCP verhält sich bei Modulo-Achsen wie folgt: Die Zielposition wird im Augenblick des Auftragsabbruchs vermerkt. Es wird über so viele Umdrehungen auf die Position gebremst, dass die Beschleunigungsvorgaben eingehalten werden. Bei Linearachsen wird der gesamte Bremsweg rückwärts gefahren, weil es nur eine mathematische Möglichkeit gibt, um die Zielposition zu erreichen.

3 Anhang 1: Best Practise bei der SPS-Anwendungsprogrammierung

3.1 Grundsätzliches

Die PLCopen-Spezifikation definiert ein gewisses Verhalten der PLCopen-FB. Die genaue Kenntnis der PLCopen-Spezifikation ist Grundvoraussetzung für eine erfolgreiche SPS-Anwendungsprogrammierung unter Verwendung der PLCopen-FB und der ISG-MCP. Des Weiteren wird für die folgenden Ausführungen ein sicherer Umgang mit dem verwendeten SPS-Programmiersystem vorausgesetzt.

3.2 Wesentliches in Kürze

3.2.1 Verhalten der „Execute“ und „Done“ Ein / Ausgänge der PLCopen-FB

Ein PLCopen-FB wertet nur die **AUF - FLANKE** des „Execute“-Signals aus.

D.h. bevor ein FB erneut beauftragt werden kann, muss er mindestens einmal mit „Execute“ = FALSE aufgerufen werden!

Das „Done“-Signal eines PLCopen-FB wird **nur** aufgrund der **AB - FLANKE** des „Execute“-Signals gelöscht.

D.h. wenn z.B. der „Done“-Ausgang eines FB auf den „Execute“-Eingang eines zweiten FB gelegt wird, kann sich im Zusammenhang mit einer Triggerbeauftragung folgendes Problem ergeben:

Wenn der erste FB getriggert wird und „Execute“ wird FALSE bevor „Done“ TRUE wird, so bleibt dieses „Done“ am ersten FB solange stehen, bis dessen „Execute“ eine neuerliche AB - FLANKE des „Execute“-Signals durchläuft. Da der zweite FB direkt mit dem ersten verbunden ist, kann auch sein „Execute“ erst dann wieder eine Auf-Flanke detektieren, wenn der erste FB eine komplette Triggerung durchgemacht hat. Bis dahin ist er jedoch **blockiert!**

3.2.2 Knackpunkt: Auftragsdurchsetzung und -quittierung

Das HLI des ISG MC hat **pro Achse genau eine** Beauftragungs- und Quittierungsschnittstelle für MC-Aufträge. Beauftragungs- und Quittierungsschnittstelle sind jeweils als Verbrauchsdatum realisiert. Daraus ergibt sich, dass pro SPS-Zyklus maximal 1 FB-Auftrag abgesetzt werden kann.

Wird innerhalb eines SPS-Zyklus ein zweiter FB angetriggert, so meldet dieser den FB-spezifischen Fehler: **4 (FB_ERR_MC_DID_NOT_TAKE_ORDER)** weil die Beauftragungsschnittstelle durch den vorangegangenen Auftrag blockiert ist.

Deshalb gilt folgende Regel:

Es darf max. ein PLCopen-FB Auftrag pro Achse u. SPS-Zyklus abgesetzt werden.

Obwohl pro Achse und SPS-Zyklus maximal ein Auftrag durchgesetzt werden kann, können aus SPS-Sicht auch mehrere Aufträge „unterwegs“ sein, für die auf Quittierungen gewartet wird.

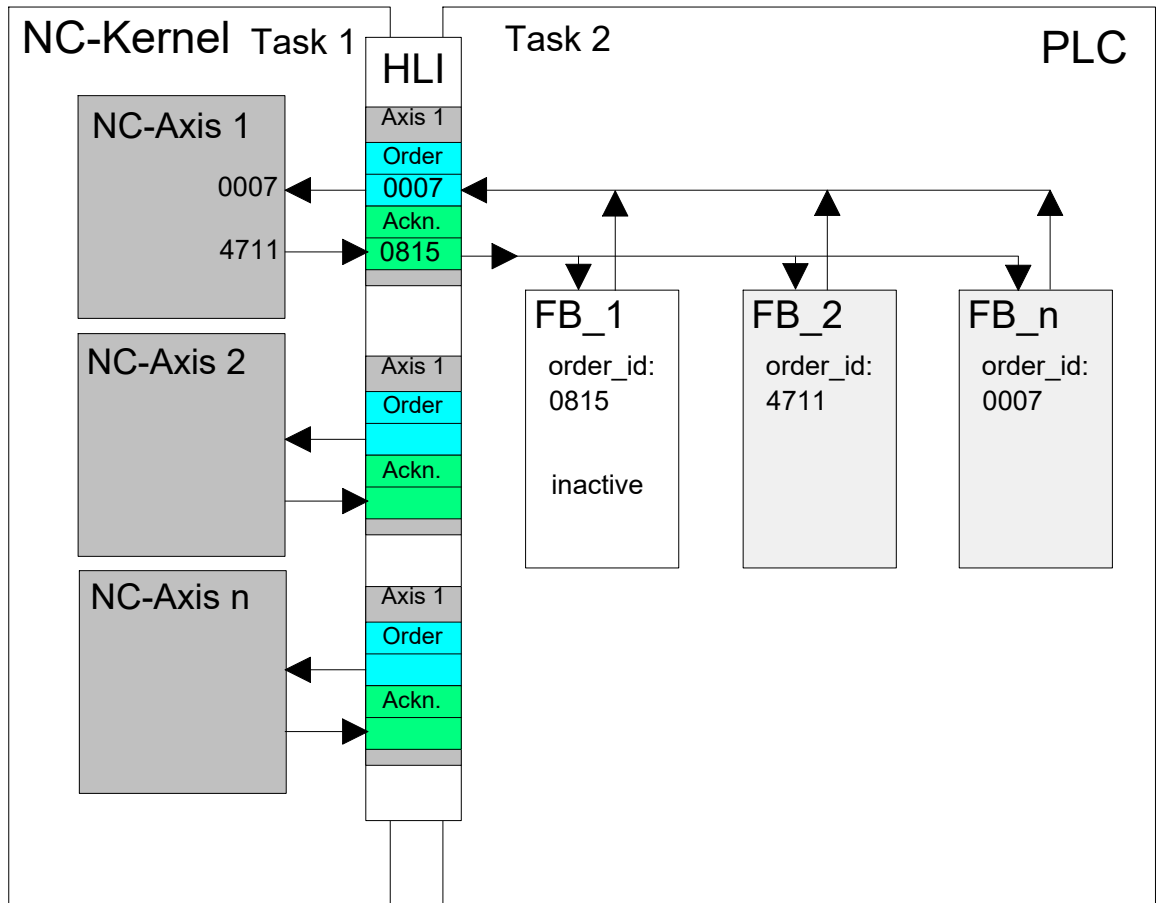


Abb. 14: Verstopfungszustand falls ein FB nicht mehr aufgerufen wird

Jede FB Instanz entnimmt nur diejenigen Quittierungen (Auftragsnummern), die sie beauftragt hat. Ein Verstopfungszustand ergibt sich falls ein FB, nachdem er eine Beauftragung abgeschickt hat, vor Erhalt „seiner“ Quittierung nicht mehr aufgerufen wird. Dann können auch die anderen, noch aktiven, keine Quittierung mehr erhalten.

Eine solche Situation kann nur durch einen Fehler im SPS-Anwendungsprogramm oder durch ungeordnetes Herunterfahren und starten der SPS bei laufender NC eintreten.

Deshalb gilt folgende weitere wichtige Regel:

Ein PLCopen-FB muss nach einer Beauftragung solange aufgerufen werden, bis er einen seiner Ausgänge „Done“, „CommandAborted“ oder „Error“ setzt.

Zur Detektierung eines solchen Problems kann im SPS-Programm eine Überwachungsfunktion eingebaut werden, die überprüft, ob ein und dieselbe Quittierung mehr als 2 SPS-Durchläufe lang auf dem HLI anliegt. Dies kann bei einem fehlerfreien SPS-Programm nicht sein.

3.3

Tipps und Tricks zur SPS-Anwendungsprogrammierung

Typischerweise erfolgt die SPS-Anwendungsprogrammierung in Schrittfolgen. Werden dabei die oben aufgeführten Beauftragungs- und Aufrufarten gemischt, kann es leicht zu einem unerwarteten Verhalten des Anwendungsprogramms kommen.

Bezüglich der **Beauftragungsart** PLCopen-FB kann unterschieden werden:

- Triggerbeauftragung („Execute“ nur ein Takt langes TRUE)
- Pegelbeauftragung („Execute“ liegt mindestens an bis „Done“ = TRUE)

Der SPS-Anwendungsprogrammierer sollte sich vorab Gedanken machen welche der beiden Beauftragungsarten er verwenden möchte und sollte diese innerhalb eines Projektes beibehalten.

Bezüglich der **Aufrufart** PLCopen-FB kann unterschieden werden:

- Aufruf immer aller FB in jedem SPS-Zyklus
- Aufruf nur der jeweils relevanten FB pro Schritt

Auch hier sollte sich der SPS-Anwendungsprogrammierer vorab Gedanken machen welche der beiden Aufrufarten er verwenden möchte und sollte auch diese innerhalb eines Projektes beibehalten.

3.4 Tipps zur Laufzeitoptimierung

Bei größeren Applikationen mit vielen Achsen kann es notwendig werden, die Instanzen der Funktionsbausteine zeitoptimiert aufzurufen, d.h. die FB-Instanzen sollen nur dann ausgeführt werden, wenn sie auch benötigt werden.

Der folgende kurze Codeabschnitt in StructuredText soll die Technik verdeutlichen, mit der die Laufzeit einer Applikation verringert werden kann. Als Beispiel wird hier der Funktionsbaustein „MC_MoveVelocity“ verwendet.

Mit der Variablen „MC_MoveVelocity_Active“ wird der Funktionsbaustein gesteuert. Sie kann dabei die folgenden Werte annehmen:

| Wert | Bedeutung |
|------|--|
| 0 | Funktionsbaustein wird nicht ausgeführt. |
| 1 | Funktionsbaustein wird ausgeführt, der Eingang „Execute“/„Enable“ ist TRUE. |
| 2 | Funktionsbaustein wird ausgeführt, bis die entsprechend abgefragte Quittierung (z.B. „Done“, „CommandAborted“ etc.) gesetzt ist. „Execute“/„Enable“ ist FALSE. |

Zu Beginn wird „MC_MoveVelocity_Active“ in Abhängigkeit vom Eingang „Execute“/„Enable“ auf 1 gesetzt. Damit wird der Funktionsbaustein „MC_MoveVelocity“ ausgeführt. Solange der Eingang nicht zurück gesetzt wird, bleibt dieser Zustand erhalten.

```
IF ( Execute_MoveVelocity = TRUE ) AND
  ( MC_MoveVelocity_Active = 0 ) THEN
  MC_MoveVelocity_Active := 1;
END_IF;
```

Erst, wenn der Eingang „Execute“/„Enable“ auf FALSE gesetzt wird, ändert sich der Wert von „MC_MoveVelocity_Active“ auf 2. Der Funktionsbaustein wird jetzt noch solange gerechnet, bis eine Quittierung am Ausgang anliegt (hier: „Done“ oder „CommandAborted“). Im Fehlerfall, der hier nicht berücksichtigt ist, ist entsprechend ähnlich zu vorgehen.

```
IF MC_MoveVelocity_Active > 0 THEN
  MC_MoveVelocity( Axis           := AxisReference,
                  Execute         := Execute_MoveVelocity,
                  Velocity        := Velocity_MoveVelocity,
                  Acceleration    := Acceleration_MoveVelocity,
                  Deceleration    := Deceleration_MoveVelocity,
                  Jerk             := Jerk_MoveVelocity,
                  Direction       := Direction_MoveVelocity );

  AxisReference           := MC_MoveVelocity_0.Axis;
  InVelocity_MoveVelocity := MC_MoveVelocity_0.InVelocity;
  CommandAborted_MoveVelocity := MC_MoveVelocity_0.CommandAborted;
  Error_MoveVelocity      := MC_MoveVelocity_0.Error;
  ErrorID_MoveVelocity    := MC_MoveVelocity_0.ErrorID;

  IF MC_MoveVelocity_Active = 2 AND
    ( InVelocity_MoveVelocity = TRUE OR
      CommandAborted_MoveVelocity = TRUE ) THEN
    MC_MoveVelocity_Active := 0;
  ELSIF Execute_MoveVelocity = FALSE THEN
    MC_MoveVelocity_Active := 2;
  END_IF;
END_IF;
```

4 Anhang 2: HelloWorld mit der ISG Motion Control Platform

4.1 „Multiprog“-Programmierbeispiel

Als Aufgabe soll die 1. Achse im System durch Vorgabe von Strecken relativ zur bisherigen Position bewegt werden. Zusätzlich soll die aktuelle Position der Achse angezeigt werden.

4.1.1 Schritt 1: Einfügen der erforderlichen Bibliotheken

Zur Lösung einer Bewegungsaufgabe unter Verwendung von PLCopen-FB müssen die SPS-Bibliotheken

hli_lib.mwt, Nachbildung des Speichers zwischen SPS und MC

McpBase.mwt, stellt Verbindung zu MC her, Bereitstellung von Datenstrukturen und FB die in weiteren Bibliotheken verwendet werden

McpPLCopenP1.mwt, FB nach PLCopen Spezifikation Part 1 und 2



Abb. 15: Erforderliche Bibliotheken in Projekt einbinden

4.1.2

Schritt 2: Anlegen von SPS-Programm Main und HelloWorld

In diesem Beispiel wird das Programm Main in Structured Text (ST) programmiert und das Programm HelloWorld als Funktionsblockdiagramm (FBD).

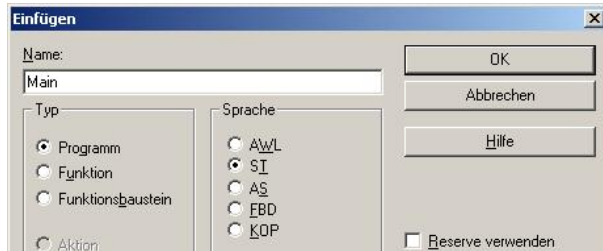


Abb. 16: Anlegen des Programms Main in ST

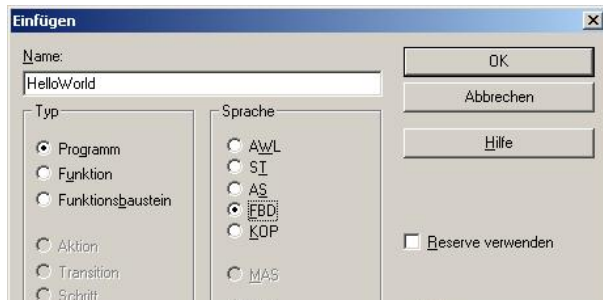


Abb. 17: Anlegen von Programm HelloWorld in FBD

Nach dem Anlegen erscheinen die Programme im Projektbaum unter „Logische POEs“

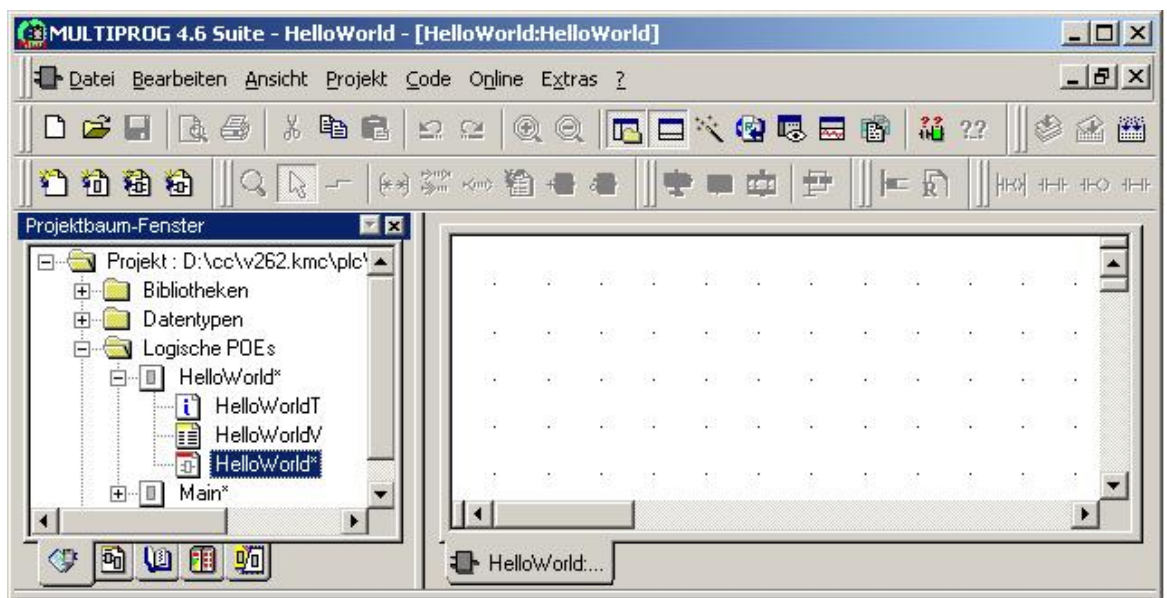


Abb. 18: Einordnung der Programme Main und HelloWorld in Projektbaum

4.1.3 Schritt 3: Implementation von Programm Main

Zur Lösung von Bewegungsaufgaben mit FB nach der PLCopen Spezifikation Part1 ist es erforderlich, dass eine Instanz des MCE-Plattform-FB **MCV_PlatformBase** und eine Instanz des Motion-Plattform-FB **MCV_P1_PLATFORM** zyklisch aufgerufen werden.

Aus diesem Grund wird im Programm Main von jedem FB eine Instanz angelegt wie in der Tabelle aufgeführt:

Instanzen der im Programm Main verwendeten FB

| FB-Typ | Instanzname | Bemerkungen |
|------------------|--------------------|--|
| MCV_PlatformBase | MCV_PlatformBase_1 | Stellt Verbindung zu MC her |
| MCV_P1_PLATFORM | MCV_P1_PLATFORM_1 | Aktualisiert zyklisch die Achsreferenzen |

Und der nachfolgende Code in ST implementiert

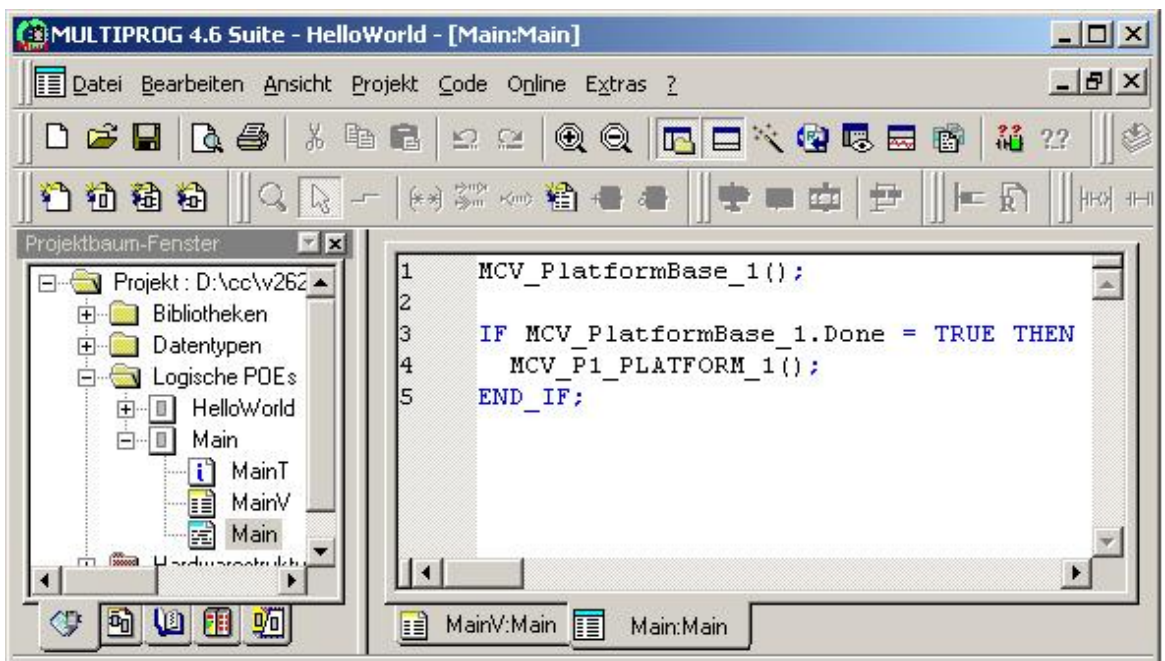


Abb. 19: Implementation von Programm Main

4.1.4 Schritt 4: Programm HelloWorld: Instanzieren der PLCopen FB

Die nachfolgend aufgeführten Instanzen von FB werden zur Lösung der Aufgabe benötigt und im Programm HelloWorld angelegt, in dem die Bewegungsaufgabe implementiert werden soll.

Instanzen der im Programm HelloWorld verwendeten FB

| PLCopen-FB | Instanzname | Bemerkungen |
|-----------------------|-------------------------|--|
| MC_Power | MC_Power_1 | Dient zum Setzen der Regler- und Vorschubfreigabe |
| MC_ReadActualPosition | MC_ReadActualPosition_1 | Zeigt die Position der Achse an der OUT-Variable „Position“ |
| MC_MoveRelative | MC_MoveRelative_1 | Bewegt die Achse um den Wert der IN-Variable „Distance“ relative zur aktuellen Position. |

Darstellung der instanziierten Funktionsblöcke im Programm HelloWorld

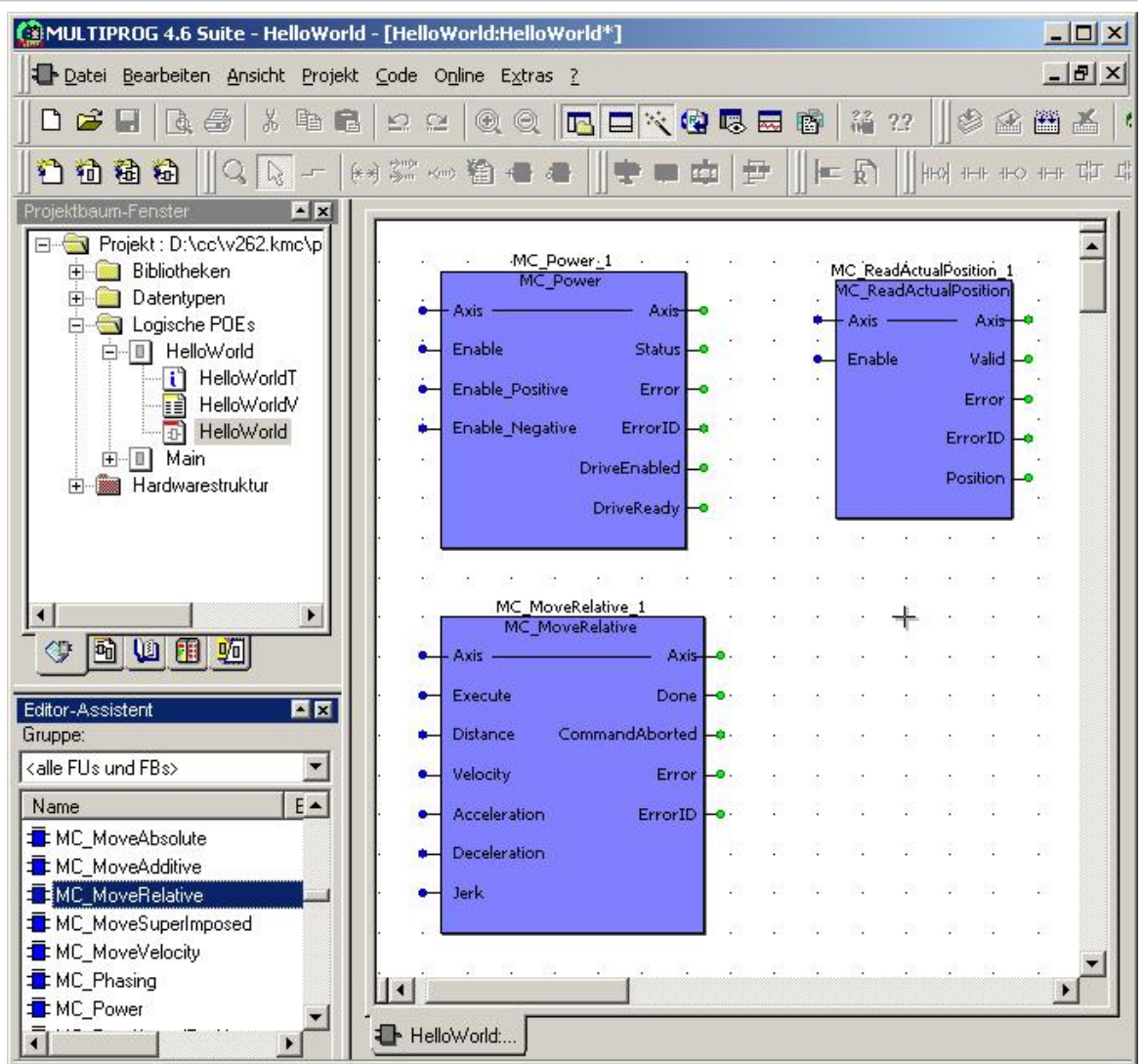


Abb. 20: Im Programm HelloWorld instanziierte PLCopen-FB

4.1.5

Schritt 5: Anbinden der Achse an die PLCopen FB

Jetzt wird die Achsreferenz `g_array_axis_ref[0]`, die auf die erste Achse im System verweist, an alle PLCopen-FB angelegt.

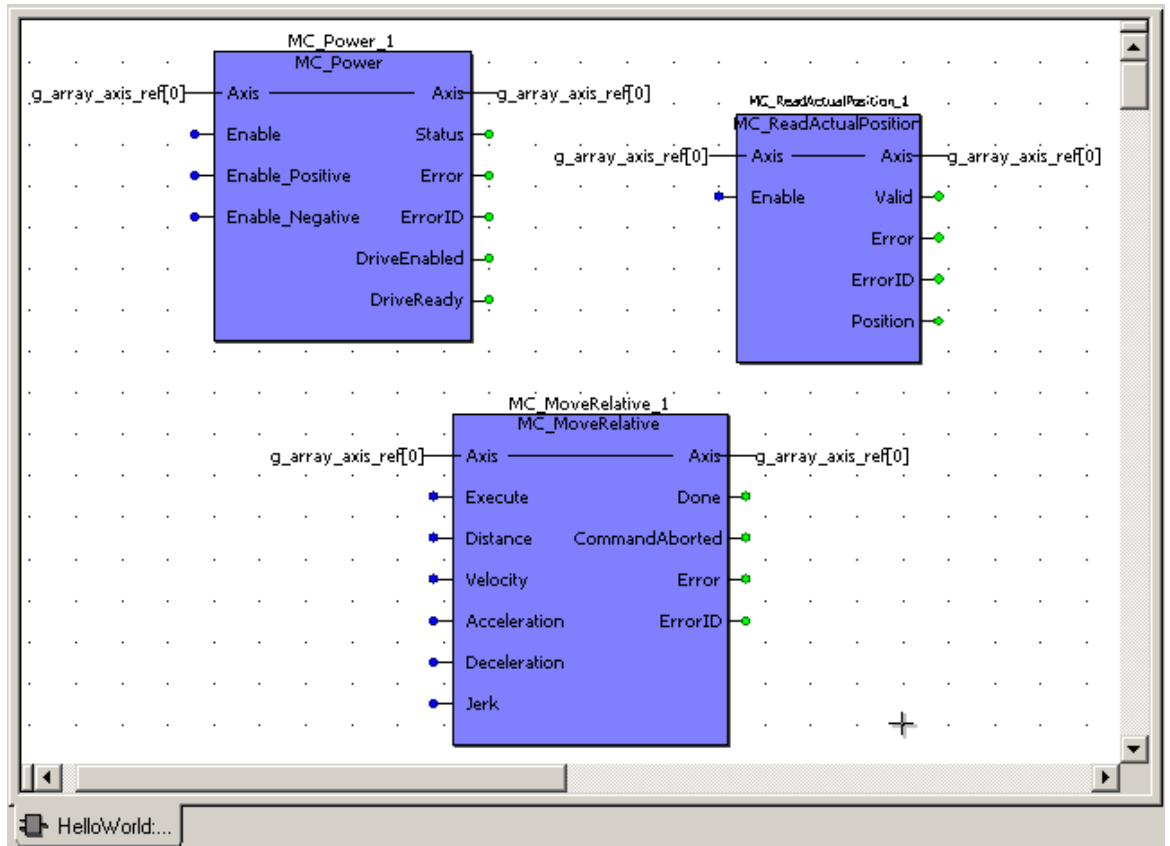


Abb. 21: Anbinden der ersten Achse im System an PLCopen-FB über `g_array_axis_ref[0]`

4.1.6 Schritt 6: Belegen der Baustein-IN/OUT-Variablen

An die benötigten Ein-/Ausgänge der PLCopen-FB werden Variablen angeschlossen, die später im Betrieb mit Werten beschrieben werden können und so die Bewegung kommandieren. Die Initialisierungswerte können der Tabelle entnommen werden:

Variablen zur Verbindung mit Ein-/Ausgängen der PLCopen-FB

| Variable | Datentyp | Initialisierungswert |
|--|----------|----------------------|
| Instanz MC_Power_1 | | |
| EnablePower | BOOL | FALSE |
| EnablePositive | BOOL | FALSE |
| EnableNegative | BOOL | FALSE |
| Instanz MC_ReadActualPosition_1 | | |
| EnableReadActPos | BOOL | TRUE |
| Position | REAL | |
| Instanz MC_MoveRelative_1 | | |
| Distance | REAL | 100000.0 |
| Velocity | REAL | 10000.0 |
| Acceleration | REAL | 2000.0 |
| Deceleration | REAL | 2000.0 |
| Jerk | REAL | 2000.0 |
| Done | BOOL | |

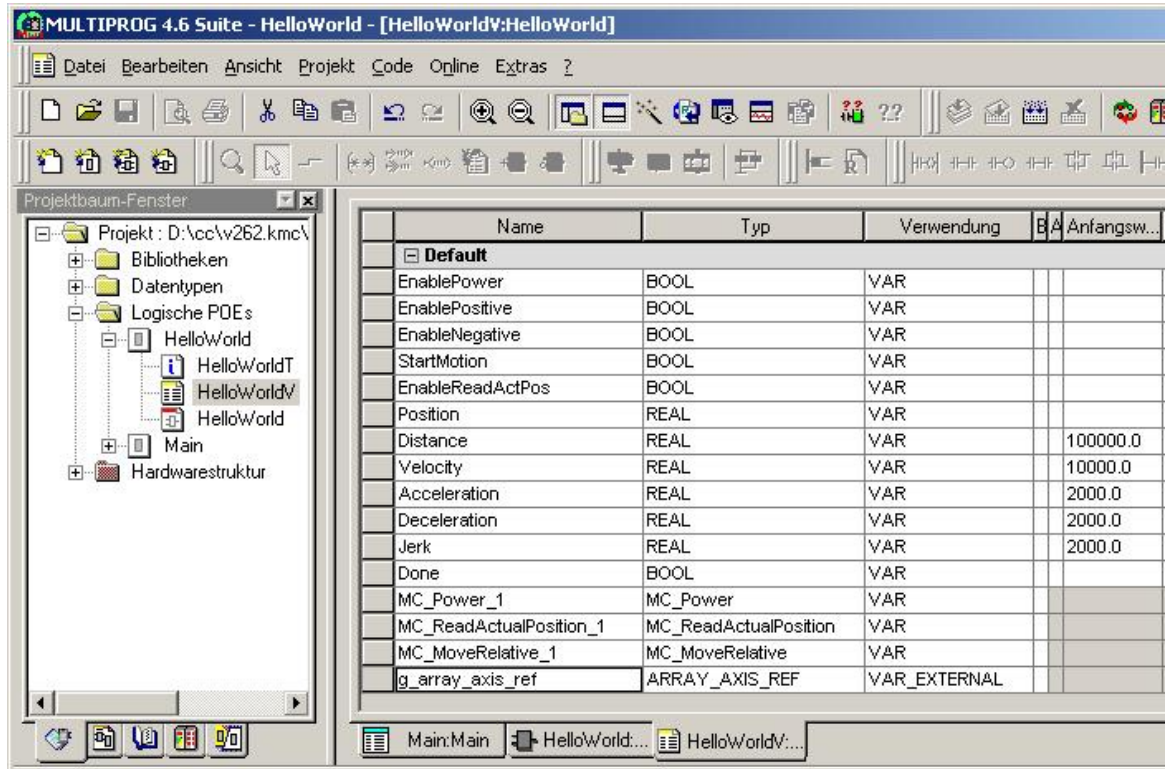


Abb. 22: Deklaration der Variablen im Variablenarbeitsblatt

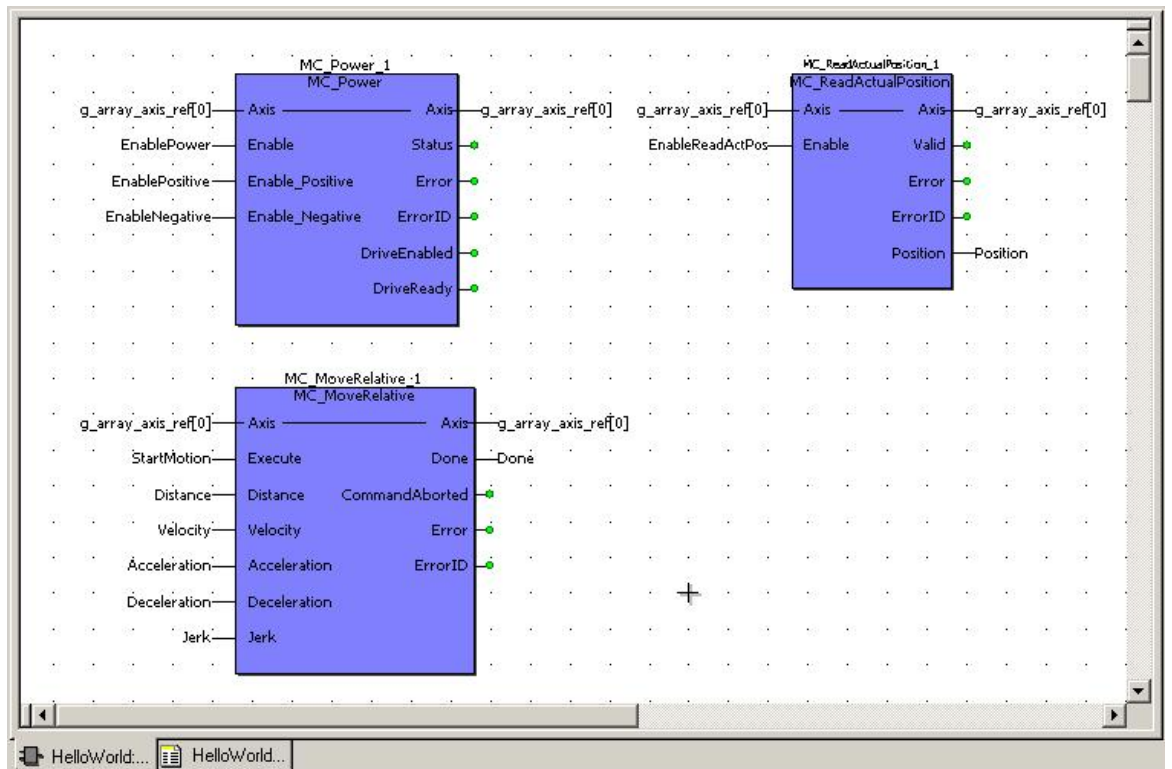


Abb. 23: Ein-/Ausgabevariablen an PLCOpen-FB angeschlossen

4.1.7 Schritt 7: Zuordnung der Programme zu einer Task

Von jedem Programm wird eine Instanz einer Task zugeordnet. Dabei ist es unbedingt erforderlich, dass das Programm Main vor dem Applikationsprogramm HelloWorld aufgerufen wird, da dort die Funktionsblöcke aufgerufen werden, die zur korrekten Funktion der MCE erforderlich sind.

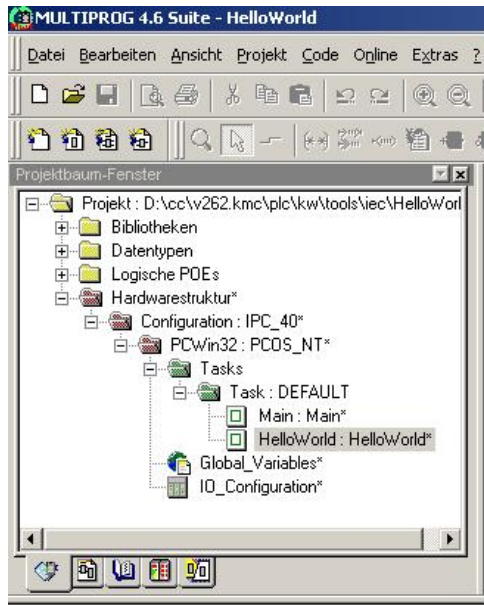


Abb. 24: Einfügen der Instanzen der Programme Main und HelloWorld in Task

4.1.8 Schritt 8: Anlegen der erforderlichen globalen Variablen

In der Resource, in der auch die Task definiert wurde, die die beiden Programme Main und HelloWorld aufruft, müssen zusätzlich die erforderlichen „Globalen Variablen“ am Besten in einer eigenen Gruppe zur besseren Übersichtlichkeit angelegt werden.

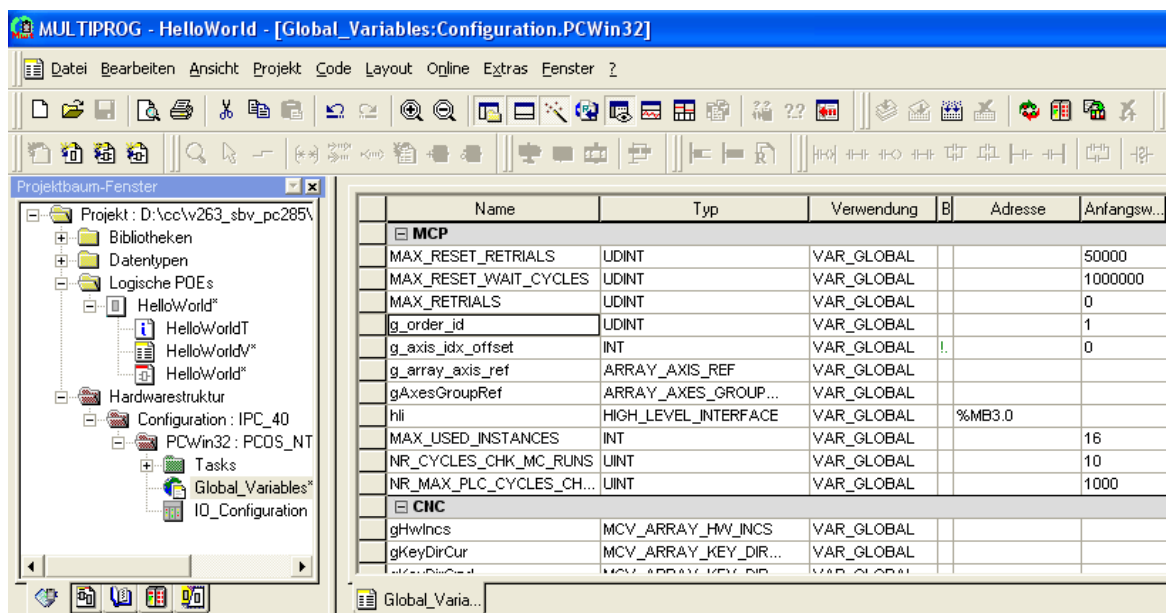


Abb. 25: Globale Variablen werden in entsprechender Resource angelegt

4.1.9 Schritt 9: Projekt senden, Kalt starten

Nach der erfolgreichen Kompilierung des Projektes wird dieses an die Ressource gesendet und über den Button „Kalt“ gestartet.



Abb. 26: Kontrolldialog zum Senden, starten der SPS-Applikation

Nach dem Starten des Projektes werden im FBD die einzelnen Werte der an die Funktionsblöcke angeschlossenen Variablen dargestellt:

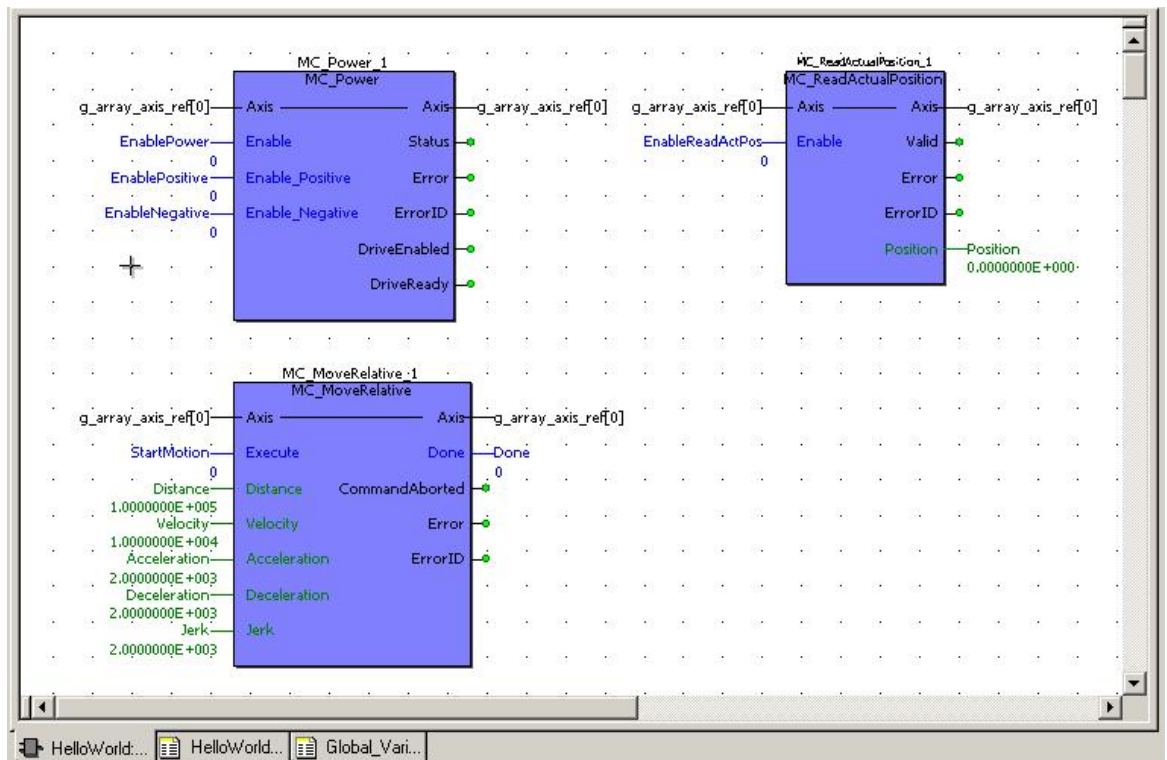


Abb. 27: Zustand des Programmes HelloWorld nach dem Programmstart

4.1.10 Schritt 10: Setzen der Freigaben für die Achse

Im Debug-Modus können die Werte für die Ein-/Ausgabeveriablen überschrieben werden. Zuerst wird die Regler- und Vorschubfreigabe für den Antrieb der Achse gegeben.

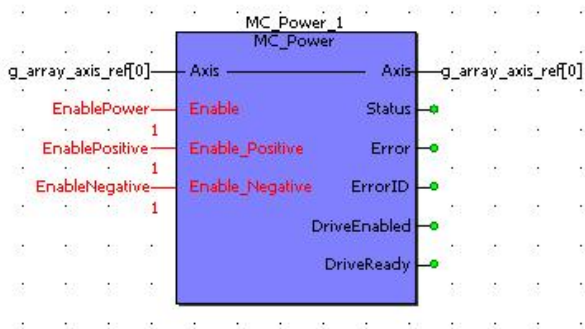


Abb. 28: Setzen von Regler- und Vorschubfreigabe an MC_Power_1

4.1.11 Schritt 11: Setzen der Freigabe für PLCopen-FB

Um die Bewegung auszulösen, muss die Eingabeveriable **StartMotion** auf **TRUE** gesetzt werden.

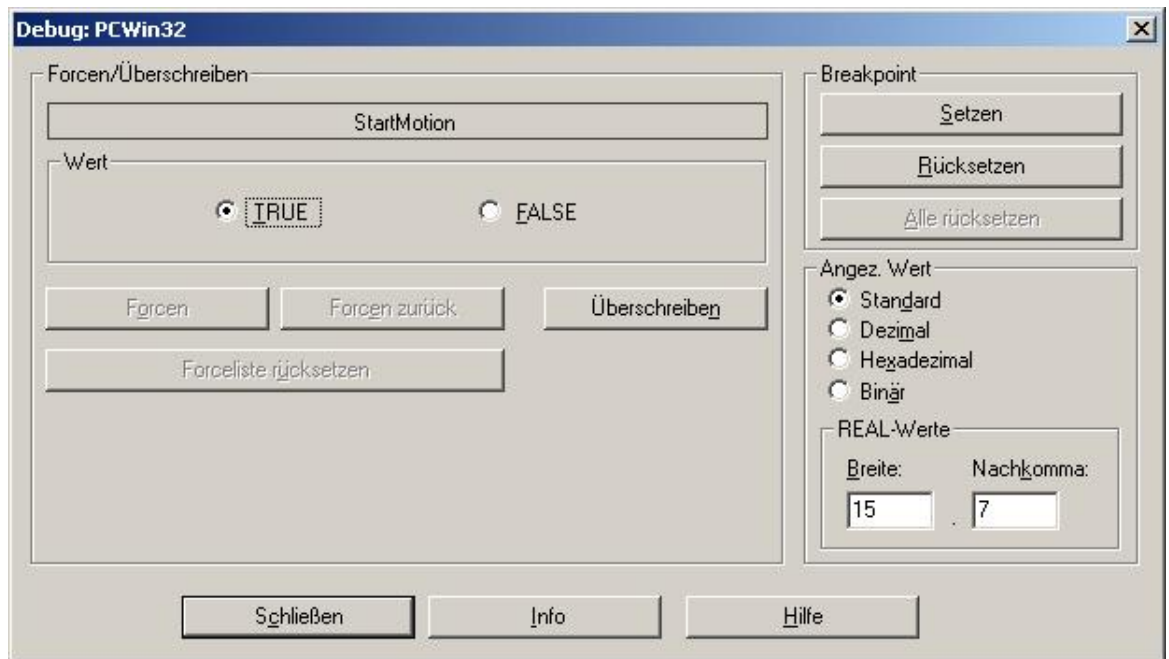


Abb. 29: Setzen der Eingangsvariable StartMotion um die Bewegung zu starten

4.1.12 Schritt 12: Fertig, Achse ist verfahren!

Nachdem die Variable StartMotion auf TRUE gesetzt wurde beginnt die Bewegung der 1. Achse und am FB MC_ReadActualPosition_1 kann die aktuelle Istposition der Achse abgelesen werden.

Das Ende der Bewegung wird durch das Setzen des Ausgangs „Done“ am FB MC_MoveRelative_1 angezeigt. Dieser Ausgang bleibt solange TRUE, bis eine fallende Flanke am Eingang „Execute“ detektiert wird. Dies wird erreicht indem die Variable über wieder auf FALSE gesetzt wird.

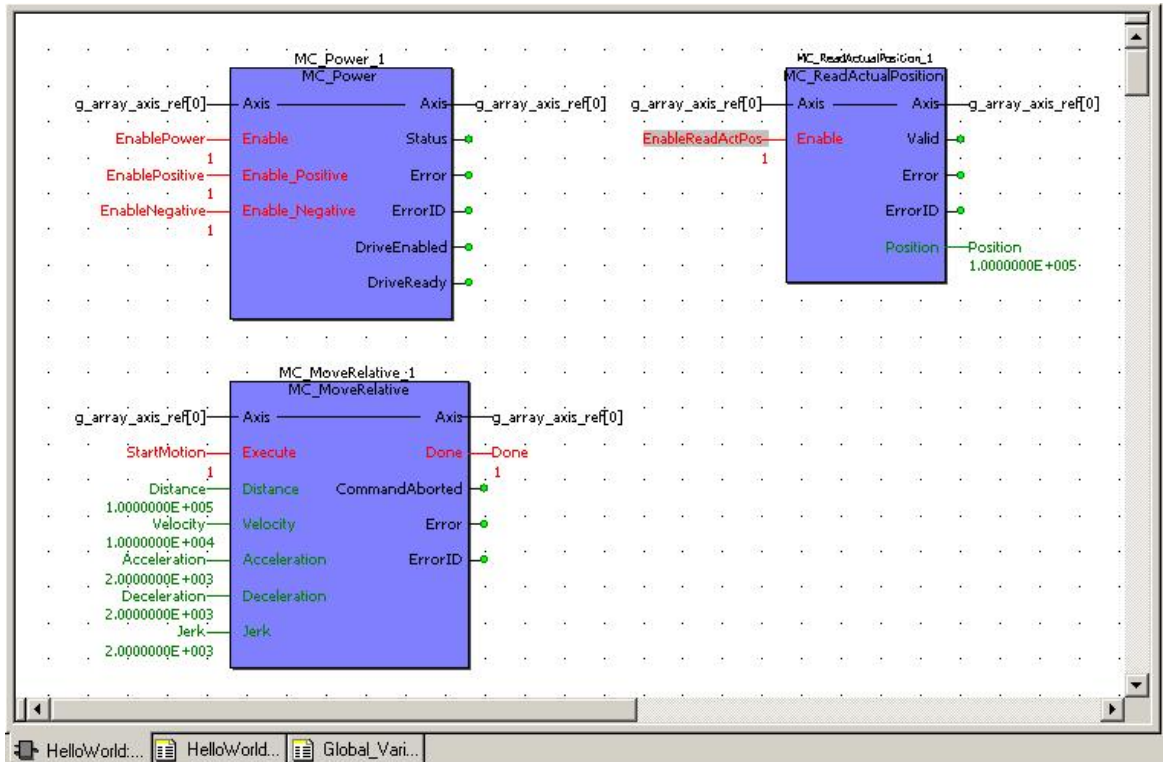


Abb. 30: Zustand nach Ende der Bewegung

4.2 „CoDeSys“-Programmierbeispiel

Auf der Basis des SPS-Projektes `Frame_PLCopenP1.pro`, wird gezeigt wie eine einfache Bewegungsaufgabe gelöst wird. In dieser Rahmenapplikation werden im Programm `MAIN` alle Programme und Funktionsbausteine aufgerufen, die die Verbindung zum Motion Controller aufbauen und die Arbeitsdaten der SPS initialisieren.

4.2.1 Schritt 1: Erforderliche Bibliotheken

Zur Lösung einer Bewegungsaufgabe unter Verwendung von `PLCopen-FB` müssen die SPS-Bibliotheken

- **STANDARD.LIB**, gehört zum SPS-Laufzeitsystem
- **hli_rts_lib.lib**, enthält Utility-FB
- **hli.lib**, Nachbildung des Speichers zwischen SPS und MC
- **McpBase.lib**, stellt Verbindung zu MC her, Bereitstellung von Datenstrukturen und FB die in weiteren Bibliotheken verwendet werden
- **McpPLCopenP1.lib**, FB nach `PLCopen Spezifikation Part 1 und 2`

in der Applikation eingebunden sein. Im Beispielprogramm `Frame_PLCopenP1.pro` ist dies bereits der Fall.

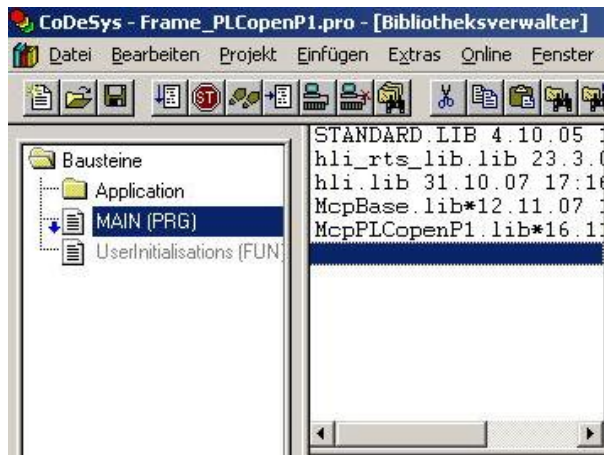


Abb. 31: Erforderliche Bibliotheken in Projekt eingebunden

Dieses Beispielprogramm wird geöffnet und als `HelloWorld.pro` gespeichert, bevor die weiteren Arbeitsschritte zur Lösung der Aufgabe durchgeführt werden.

4.2.2

Schritt 2: Anlegen des Applikationsprogrammes HelloWorld

Die Bewegungsaufgabe wird im Programm HelloWorld implementiert. Zur Implementierung soll der freigraphische Funktionsplaneditor (CFC) verwendet werden. Entsprechend diesen Vorgaben wird das Programm im bereits vorhandenen Ordner „Application“ angelegt:



Abb. 32: Definition des Programms HelloWorld



Abb. 33: Einordnung der Programme Main und HelloWorld in Projektbaum

4.2.3 Schritt 3: Programm HelloWorld: Instanzieren der PLCopen FB

Die nachfolgend aufgeführten Instanzen von FB werden zur Lösung der Aufgabe benötigt und im Programm HelloWorld angelegt, in dem die Bewegungsaufgabe implementiert werden soll.

Instanzen der im Programm HelloWorld verwendeten FB

| PLCopen-FB | Instanzname | Bemerkungen |
|-----------------------|-------------------------|--|
| MC_Power | MC_Power_1 | Dient zum Setzen der Regler- und Vorschubfreigabe |
| MC_ReadActualPosition | MC_ReadActualPosition_1 | Zeigt die Position der Achse an der OUT-Variable „Position“ |
| MC_MoveRelative | MC_MoveRelative_1 | Bewegt die Achse um den Wert der IN-Variable „Distance“ relative zur aktuellen Position. |

Die nachfolgende Darstellung zeigt den Variablen-Definitionsbereich in dem die erforderlichen Funktionsblöcke definiert wurden und ihre Erscheinung im Funktionsplaneditor:

The screenshot shows the 'HelloWorld (PRG-CFC)' editor with the following code in the variable declaration section:

```

0001 PROGRAM HelloWorld
0002 VAR
0003   MC_Power_1           : MC_Power;
0004   MC_MoveRelative_1   : MC_MoveRelative;
0005   MC_ReadActualPosition_1 : MC_ReadActualPosition;
0006 END_VAR
0007
    
```

Below the code, three function blocks are instantiated in the function plan editor:

- MC_Power_1 (Instance 0):**
 - Inputs: Enable, Enable_Positive, Enable_Negative, Axis
 - Outputs: DriveEnabled, DriveReady, Status, Error, ErrorID, Axis
- MC_ReadActualPosition_1 (Instance 1):**
 - Input: Axis
 - Outputs: Valid, Error, ErrorID, Position, Axis
- MC_MoveRelative_1 (Instance 2):**
 - Inputs: Execute, Distance, Velocity, Acceleration, Deceleration, Jerk, Axis
 - Outputs: Done, CommandAborted, Error, ErrorID, Axis

Abb. 34: Im Programm HelloWorld instanziierte PLCopen-FB

4.2.4

Schritt 4: Anbinden der Achse an die PLCopen FB

Jetzt wird die Achsreferenz `g_array_axis_ref[0]`, die auf die erste Achse im System verweist, an alle PLCopen-FB angelegt.

The screenshot shows the CoDeSys IDE with the following details:

- Project Explorer (left):** Shows a tree view with 'Bausteine' containing 'Application', 'HelloWorld (PRG)', 'MAIN (PRG)', and 'UserInitialisations (FUN)'.
- Code Editor (top right):** Contains the following Ladder Logic (LLD) code:


```

0001 PROGRAM HelloWorld
0002 VAR
0003   MC_Power_1           : MC_Power;
0004   MC_MoveRelative_1   : MC_MoveRelative;
0005   MC_ReadActualPosition_1 : MC_ReadActualPosition;
0006 FND VAR
            
```
- Function Block Diagram (center):** Three function blocks are shown, each with an input box labeled `g_array_axis_ref[0]` connected to its 'Axis' parameter:
 - MC_Power_1 (0):** Parameters include Enable, Enable_Positive, Enable_Negative, and Axis. Outputs include DriveEnabled, DriveReady, Status, Error, ErrorID, and Axis.
 - MC_MoveRelative_1 (1):** Parameters include Execute, Distance, Velocity, Acceleration, Deceleration, Jerk, and Axis. Outputs include Done, CommandAborted, Error, ErrorID, and Axis.
 - MC_ReadActualPosition_1 (2):** Parameters include Enable and Axis. Outputs include Valid, Error, ErrorID, and Position.
- Code Size (bottom):** A status bar indicates 'Codegröße: 47969 Bytes'.
- Bottom Bar:** Includes buttons for 'ONLINE', 'ÜB', and 'LESEN'.

Abb. 35: Anbinden der ersten Achse im System an PLCopen-FB über `g_array_axis_ref[0]`

4.2.5 Schritt 5: Belegen der Baustein-IN/OUT-Variablen

An die benötigten Ein-/Ausgänge der PLCopen-FB werden Variablen angeschlossen, die später im Betrieb mit Werten beschrieben werden können und so die Bewegung kommandieren. Die Initialisierungswerte können der Tabelle entnommen werden:

Variablen zur Verbindung mit Ein-/Ausgängen der PLCopen-FB

| Variable | Datentyp | Initialisierungswert |
|--|----------|----------------------|
| Instanz MC_Power_1 | | |
| EnablePower | BOOL | FALSE |
| EnablePositive | BOOL | FALSE |
| EnableNegative | BOOL | FALSE |
| Instanz MC_ReadActualPosition_1 | | |
| EnableReadActPos | BOOL | TRUE |
| Position | REAL | |
| Instanz MC_MoveRelative_1 | | |
| Distance | REAL | 100000.0 |
| Velocity | REAL | 10000.0 |
| Acceleration | REAL | 2000.0 |
| Deceleration | REAL | 2000.0 |
| Jerk | REAL | 2000.0 |
| Done | BOOL | |

Im Variablen-Definitionsbereich sind nun die Variablen angelegt und teilweise mit Initialwerten vorbelegt worden. Im Funktionsplaneditor erkennt man, dass die Variablen bereits an die entsprechenden Ein-/Ausgangspins der Funktionsblöcke angeschlossen wurden:

The screenshot displays the CoDeSys development environment for a 'HelloWorld' program. The top window shows the source code for the 'PROGRAM HelloWorld' with the following variable declarations:

```

0001 PROGRAM HelloWorld
0002 VAR
0003   MC_Power_1           : MC_Power;
0004   MC_MoveRelative_1   : MC_MoveRelative;
0005   MC_ReadActualPosition_1 : MC_ReadActualPosition;
0006
0007   (* Variables connected to FB pins *)
0008   EnablePower          : BOOL;
0009   EnablePositive       : BOOL;
0010   EnableNegative       : BOOL;
0011
0012   EnableReadActPos     : BOOL := TRUE;
0013   Position             : REAL;
0014
0015   Distance             : REAL := 100000;
0016   Velocity             : REAL := 10000;
0017   Acceleration         : REAL := 2000;
0018   Deceleration         : REAL := 2000;
0019   Jerk                 : REAL := 2000;
0020   Done                 : BOOL;
0021
0022   StartMotion          : BOOL;
0023 END_VAR
    
```

The bottom window shows a graphical representation of the program's connection to three PLCopen function blocks (FBs):

- MC_Power_1 (FB 0):** Receives inputs `EnablePower`, `EnablePositive`, and `EnableNegative`. It outputs `DriveEnabled`, `DriveReady`, `Status`, `Error`, `ErrorID`, and `Axis`. The `Axis` output is connected to `g_array_axis_ref[0]`.
- MC_MoveRelative_1 (FB 1):** Receives inputs `StartMotion`, `Distance`, `Velocity`, `Acceleration`, `Deceleration`, and `Jerk`. It outputs `Done`, `CommandAborted`, `Error`, `ErrorID`, and `Axis`. The `Done` output is connected to a variable labeled `Done` (2), and the `Axis` output is connected to `g_array_axis_ref[0]`.
- MC_ReadActualPosition_1 (FB 3):** Receives inputs `EnableReadActPos` and `Axis`. It outputs `Valid`, `Error`, `ErrorID`, and `Position`. The `Position` output is connected to a variable labeled `Position` (4).

The status bar at the bottom indicates a code size of 47969 Bytes and provides buttons for `ONLINE`, `ÜB`, and `LESEN`.

Abb. 36: Ein-/Ausgabevariablen an PLCopen-FB angeschlossen

4.2.6

Schritt 6: Programm HelloWorld in Programm MAIN einfügen

In der Beispielapplikation Frame_PLCCopenP1.pro ist bereits das Programm MAIN angelegt gewesen, das alle zur korrekten Funktion der MCE erforderlichen Funktionsbausteine und deren Aufruf beinhaltet. Damit das Programm HelloWorld ausgeführt wird, wird es nun nach dem Kommentar „Insert your PLC application code after this comment“ eingefügt:

```

0001 PROGRAM MAIN
0002 VAR
0003   Hli           : MCV_HliInterface;      (* HLI interface - have to call until sets In:
0004   PlatformBase : MCV_PlatformBase;      (* PLC platform - have to call until sets Do
0005   P1_Platform   : MCV_P1_Platform;      (* Motion platform - have to call each PLC cycle
0006
0007   HliInitError   : BOOL := FALSE;      (* Error at initialisation of HLI interface *)
0008   UserInitialisationDone : BOOL := FALSE; (* Initialisations that are necessary for applic
0009 END_VAR
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
0100
0101
0102
0103
0104
0105
0106
0107
0108
0109
0110
0111
0112
0113
0114
0115
0116
0117
0118
0119
0120
0121
0122
0123
0124
0125
0126
0127
0128
0129
0130
0131
0132
0133
0134
0135
0136
0137
0138
0139
0140
0141
0142
0143
0144
0145
0146
0147
0148
0149
0150
0151
0152
0153
0154
0155
0156
0157
0158
0159
0160
0161
0162
0163
0164
0165
0166
0167
0168
0169
0170
0171
0172
0173
0174
0175
0176
0177
0178
0179
0180
0181
0182
0183
0184
0185
0186
0187
0188
0189
0190
0191
0192
0193
0194
0195
0196
0197
0198
0199
0200
0201
0202
0203
0204
0205
0206
0207
0208
0209
0210
0211
0212
0213
0214
0215
0216
0217
0218
0219
0220
0221
0222
0223
0224
0225
0226
0227
0228
0229
0230
0231
0232
0233
0234
0235
0236
0237
0238
0239
0240
0241
0242
0243
0244
0245
0246
0247
0248
0249
0250
0251
0252
0253
0254
0255
0256
0257
0258
0259
0260
0261
0262
0263
0264
0265
0266
0267
0268
0269
0270
0271
0272
0273
0274
0275
0276
0277
0278
0279
0280
0281
0282
0283
0284
0285
0286
0287
0288
0289
0290
0291
0292
0293
0294
0295
0296
0297
0298
0299
0300
0301
0302
0303
0304
0305
0306
0307
0308
0309
0310
0311
0312
0313
0314
0315
0316
0317
0318
0319
0320
0321
0322
0323
0324
0325
0326
0327
0328
0329
0330
0331
0332
0333
0334
0335
0336
0337
0338
0339
0340
0341
0342
0343
0344
0345
0346
0347
0348
0349
0350
0351
0352
0353
0354
0355
0356
0357
0358
0359
0360
0361
0362
0363
0364
0365
0366
0367
0368
0369
0370
0371
0372
0373
0374
0375
0376
0377
0378
0379
0380
0381
0382
0383
0384
0385
0386
0387
0388
0389
0390
0391
0392
0393
0394
0395
0396
0397
0398
0399
0400
0401
0402
0403
0404
0405
0406
0407
0408
0409
0410
0411
0412
0413
0414
0415
0416
0417
0418
0419
0420
0421
0422
0423
0424
0425
0426
0427
0428
0429
0430
0431
0432
0433
0434
0435
0436
0437
0438
0439
0440
0441
0442
0443
0444
0445
0446
0447
0448
0449
0450
0451
0452
0453
0454
0455
0456
0457
0458
0459
0460
0461
0462
0463
0464
0465
0466
0467
0468
0469
0470
0471
0472
0473
0474
0475
0476
0477
0478
0479
0480
0481
0482
0483
0484
0485
0486
0487
0488
0489
0490
0491
0492
0493
0494
0495
0496
0497
0498
0499
0500
0501
0502
0503
0504
0505
0506
0507
0508
0509
0510
0511
0512
0513
0514
0515
0516
0517
0518
0519
0520
0521
0522
0523
0524
0525
0526
0527
0528
0529
0530
0531
0532
0533
0534
0535
0536
0537
0538
0539
0540
0541
0542
0543
0544
0545
0546
0547
0548
0549
0550
0551
0552
0553
0554
0555
0556
0557
0558
0559
0560
0561
0562
0563
0564
0565
0566
0567
0568
0569
0570
0571
0572
0573
0574
0575
0576
0577
0578
0579
0580
0581
0582
0583
0584
0585
0586
0587
0588
0589
0590
0591
0592
0593
0594
0595
0596
0597
0598
0599
0600
0601
0602
0603
0604
0605
0606
0607
0608
0609
0610
0611
0612
0613
0614
0615
0616
0617
0618
0619
0620
0621
0622
0623
0624
0625
0626
0627
0628
0629
0630
0631
0632
0633
0634
0635
0636
0637
0638
0639
0640
0641
0642
0643
0644
0645
0646
0647
0648
0649
0650
0651
0652
0653
0654
0655
0656
0657
0658
0659
0660
0661
0662
0663
0664
0665
0666
0667
0668
0669
0670
0671
0672
0673
0674
0675
0676
0677
0678
0679
0680
0681
0682
0683
0684
0685
0686
0687
0688
0689
0690
0691
0692
0693
0694
0695
0696
0697
0698
0699
0700
0701
0702
0703
0704
0705
0706
0707
0708
0709
0710
0711
0712
0713
0714
0715
0716
0717
0718
0719
0720
0721
0722
0723
0724
0725
0726
0727
0728
0729
0730
0731
0732
0733
0734
0735
0736
0737
0738
0739
0740
0741
0742
0743
0744
0745
0746
0747
0748
0749
0750
0751
0752
0753
0754
0755
0756
0757
0758
0759
0760
0761
0762
0763
0764
0765
0766
0767
0768
0769
0770
0771
0772
0773
0774
0775
0776
0777
0778
0779
0780
0781
0782
0783
0784
0785
0786
0787
0788
0789
0790
0791
0792
0793
0794
0795
0796
0797
0798
0799
0800
0801
0802
0803
0804
0805
0806
0807
0808
0809
0810
0811
0812
0813
0814
0815
0816
0817
0818
0819
0820
0821
0822
0823
0824
0825
0826
0827
0828
0829
0830
0831
0832
0833
0834
0835
0836
0837
0838
0839
0840
0841
0842
0843
0844
0845
0846
0847
0848
0849
0850
0851
0852
0853
0854
0855
0856
0857
0858
0859
0860
0861
0862
0863
0864
0865
0866
0867
0868
0869
0870
0871
0872
0873
0874
0875
0876
0877
0878
0879
0880
0881
0882
0883
0884
0885
0886
0887
0888
0889
0890
0891
0892
0893
0894
0895
0896
0897
0898
0899
0900
0901
0902
0903
0904
0905
0906
0907
0908
0909
0910
0911
0912
0913
0914
0915
0916
0917
0918
0919
0920
0921
0922
0923
0924
0925
0926
0927
0928
0929
0930
0931
0932
0933
0934
0935
0936
0937
0938
0939
0940
0941
0942
0943
0944
0945
0946
0947
0948
0949
0950
0951
0952
0953
0954
0955
0956
0957
0958
0959
0960
0961
0962
0963
0964
0965
0966
0967
0968
0969
0970
0971
0972
0973
0974
0975
0976
0977
0978
0979
0980
0981
0982
0983
0984
0985
0986
0987
0988
0989
0990
0991
0992
0993
0994
0995
0996
0997
0998
0999
1000
    
```

Abb. 37: Einfügen von Programm HelloWorld in Programm MAIN

4.2.7

Schritt 7: Zuordnung der Programme zu einer Task

In der Beispielapplikation Frame_PLCopenP1.pro ist das Programm MAIN bereits einer Task zugeordnet. Die Darstellung zeigt die entsprechenden Einstellungen:

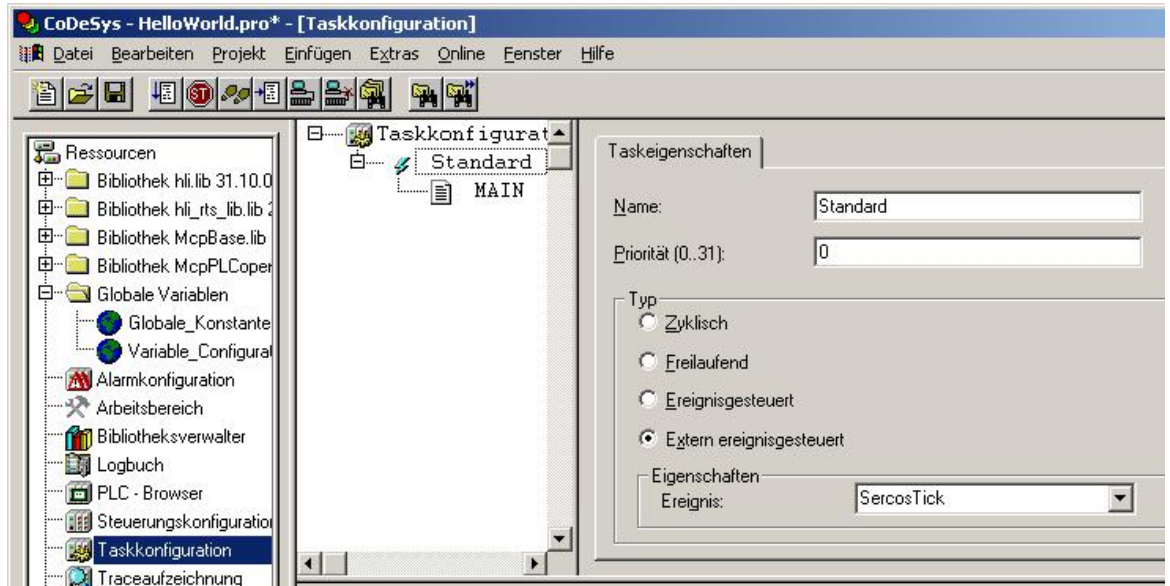


Abb. 38: Programm MAIN ist der Task Standard zugewiesen

4.2.8

Schritt 8: Applikation übersetzen, Einloggen , Starten

Es erfolgt das Übersetzen und das Einloggen der Applikation. Nach dem Start der Applikation sind die Initialwerte an den Variablen bereits sichtbar:

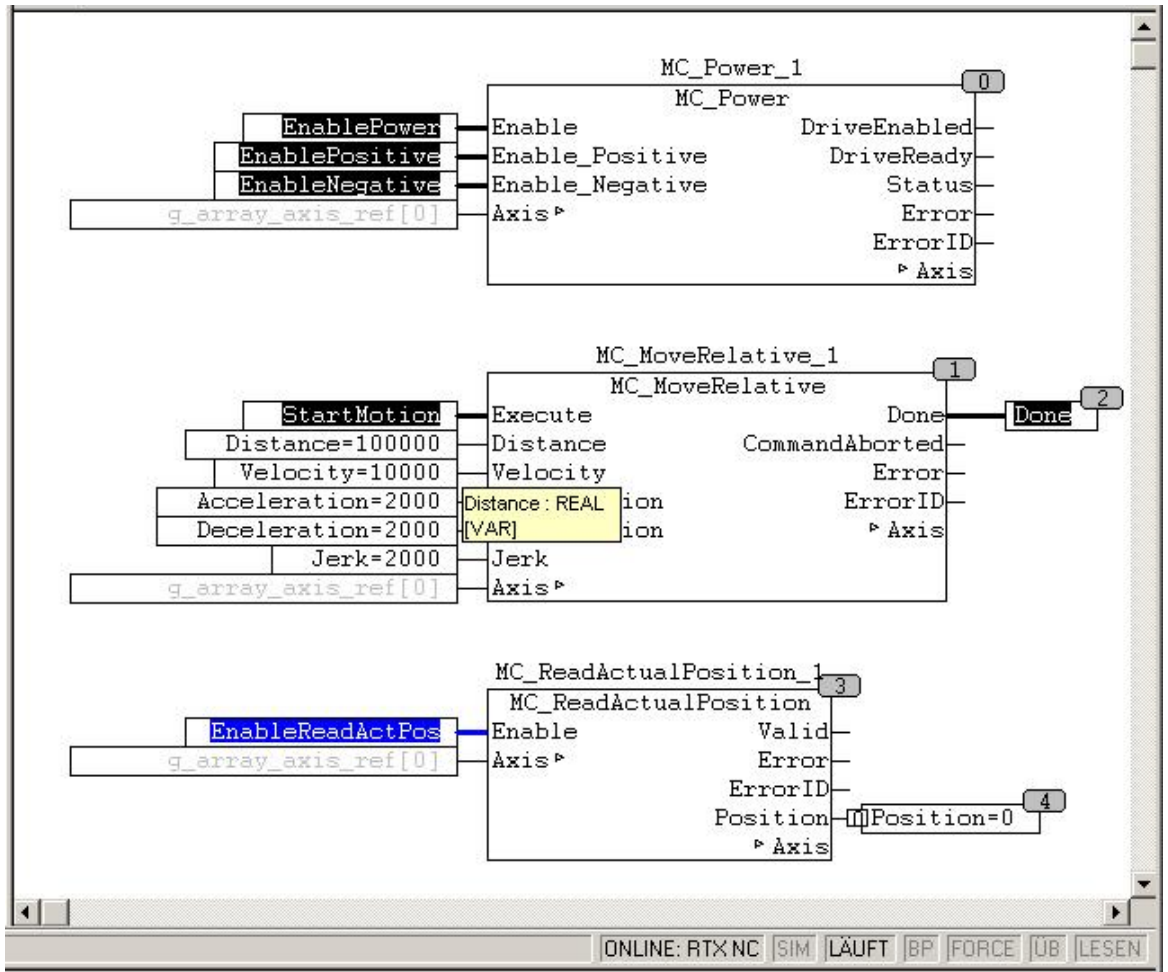


Abb. 39: Programm HelloWorld nach dem Starten der Applikation

4.2.9

Schritt 9: Setzen der Freigaben für die Achse

Zuerst wird die Regler- und Vorschubfreigabe für den Antrieb der Achse gegeben. Dies erfolgt durch Überschreiben oder Forcen der Eingangsvariablenwerte am Funktionsblock MC_Power_1.

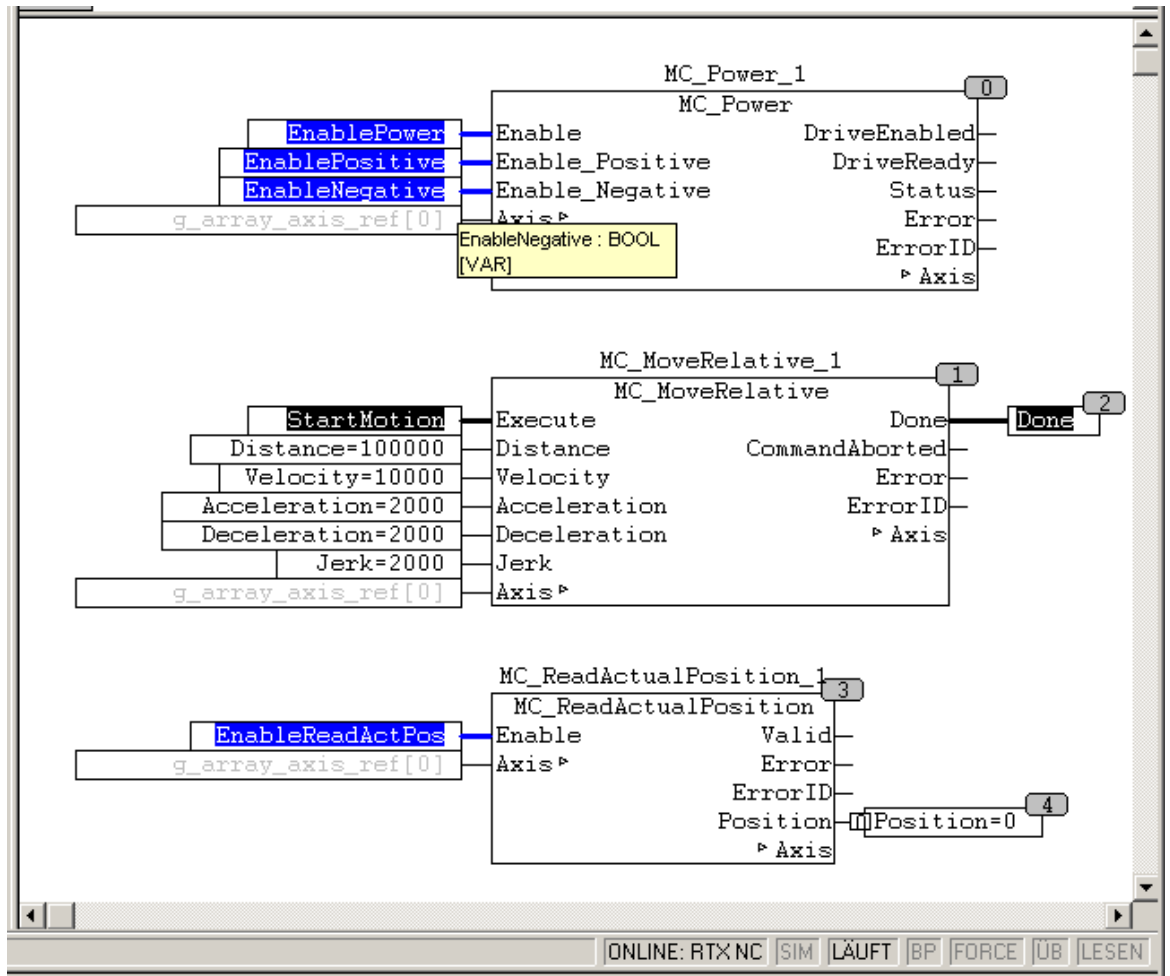


Abb. 40: Setzen von Regler- und Vorschubfreigabe an MC_Power_1

4.2.10 Schritt 10: Fertig, Achse ist verfahren!

Die Verfahrbewegung der Achse wird ausgelöst indem der Wert der Variable StartMotion am Funktionsblock MC_MoveRelative_1 auf TRUE gesetzt wird. Die aktuelle Istposition der Achse kann nun an der Variable „Position“ am FB MC_ReadActualPosition_1 abgelesen werden.

Die Darstellung zeigt den Zustand der Funktionsblöcke und Variablen am Ende der Bewegung. Erkennlich ist das Ende der Bewegung, weil die Variable „Done“ nun den Wert TRUE besitzt. Dieser Ausgang bleibt solange TRUE, bis eine fallende Flanke am Eingang „Execute“ detektiert wird. Dies wird erreicht indem die Variable über wieder auf FALSE gesetzt wird.

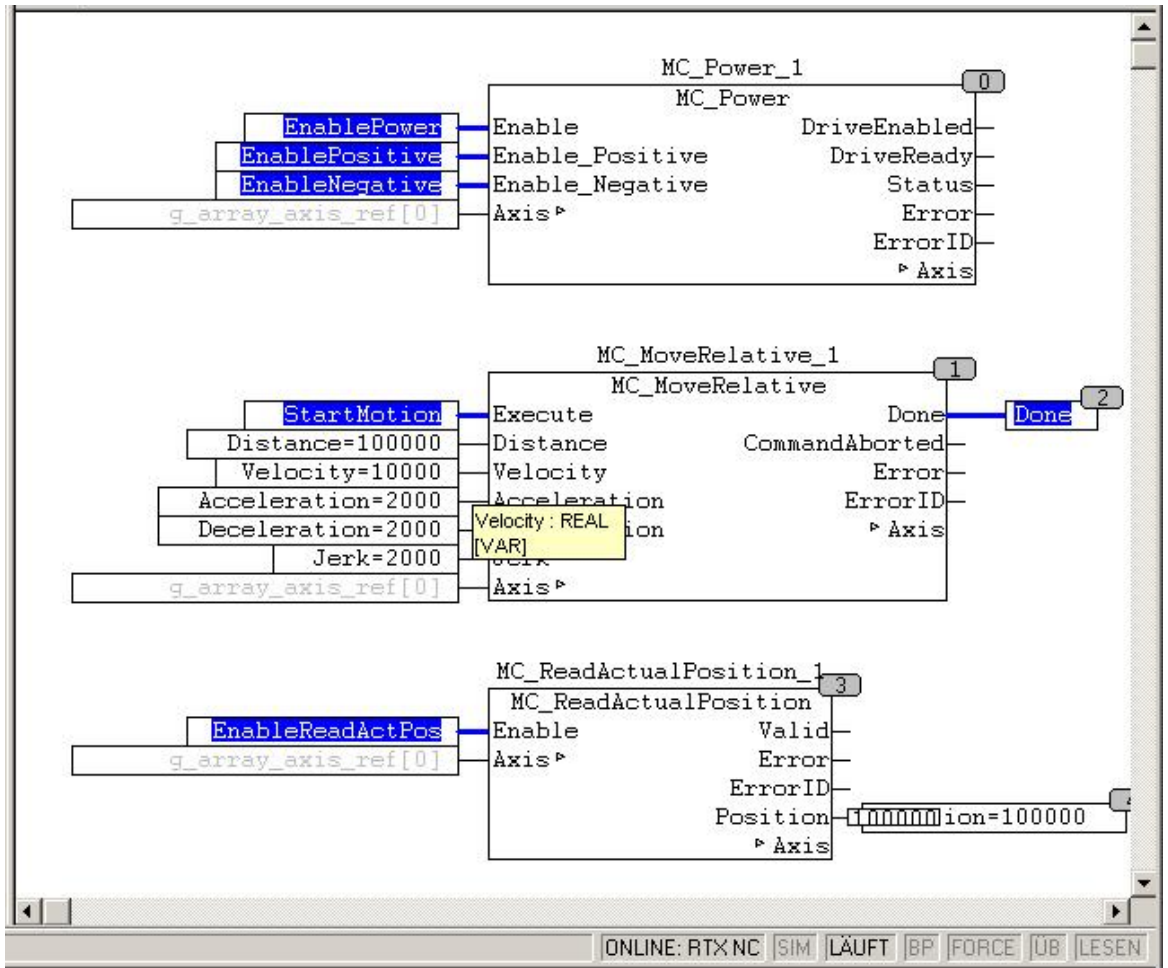


Abb. 41: Zustand am Ende der Bewegung

5 Literaturverzeichnis

[1] PLCopen-Spezifikation: TC2 Task Force Motion Control “Function Blocks for motion control”
Version 1.0, vom 23.Nov.2001

[2] Dokumentation CNC SPS Steuerungsgesamtsystem

[3] Das PLCopen Compliance Statement V1.0 von ISG ist auf der PLCopen Homepage
(www.plcopen.org) zu finden

6 Anhang

6.1 Anregungen, Korrekturen und neueste Dokumentation

Sie finden Fehler, haben Anregungen oder konstruktive Kritik? Gerne können Sie uns unter documentation@isg-stuttgart.de kontaktieren. Die aktuellste Dokumentation finden Sie in unserer Onlinehilfe (DE/EN):



QR-Code Link: <https://www.isg-stuttgart.de/documentation-kernel/>

Der o.g. Link ist eine Weiterleitung zu:

<https://www.isg-stuttgart.de/fileadmin/kernel/kernel-html/index.html>



Hinweis

Mögliche Änderung von Favoritenlinks im Browser:

Technische Änderungen der Webseitenstruktur betreffend der Ordnerpfade oder ein Wechsel des HTML-Frameworks und damit der Linkstruktur können nie ausgeschlossen werden.

Wir empfehlen, den o.g. „QR-Code Link“ als primären Favoritenlink zu speichern.

PDFs zum Download:

DE:

<https://www.isg-stuttgart.de/produkte/softwareprodukte/isg-kernel/dokumente-und-downloads>

EN:

<https://www.isg-stuttgart.de/en/products/softwareproducts/isg-kernel/documents-and-downloads>

E-Mail: documentation@isg-stuttgart.de

Stichwortverzeichnis



© Copyright
ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart
Alle Rechte vorbehalten
www.isg-stuttgart.de
support@isg-stuttgart.de

