

# **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

# Inhaltsverzeichnis

1	D	Definitionen5			
	1.1	Abkürz	ungen	5	
	1.2	Beariffs	serklärungen	5	
2	וח		Jotion Control Platform	7	
2				7	
	2.1	Was is	t die ISG Motion Control Platform ?	1	
	2.2	Elemer	te 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	Sicherł	neitskonzept, 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	Realisi	erungsdetails 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	Α	nhang <sup>,</sup>	1: Best Practise bei der SPS-Anwendungsprogrammierung	33	
-	31	Grunds	sätzliches	33	
	3.2	W/acan	tliches in Kürze	33	
	J.Z	321	Verhalten der Execute" und Done" Fin / Ausgänge der PL Copen FR	<i>33</i>	
		3.2.1 3.2.2	Knacknunkt: Auftragedurchsetzung und -quittierung	33 22	
	2.0	J.Z.Z	Ind Tricke zur SDS Anwondungenrogrommingung	00 05	
	ა.ა ე_₄	Tipps t		30	
	3.4	i ipps z		36	
4	A	nhang 2	2: HelloWorld mit der ISG Motion Control Platform	37	

kernel Industrielle Steuerungstechnik GmbH

	4.1	"Multip	rog"-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	"CoDe	Sys"-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	Li	iteratur	verzeichnis	59
~				~~
b	A	mang		60
	6.1	Anregu	ungen, Korrekturen und neueste Dokumentation	60
	S	tichwor	rtverzeichnis	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-Entwicklungsumge- bung	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
СМ	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 entspre- chend der PLCopen-Spezifikation, sowie weitere FB, die Aufgaben der Bewegungser- zeugung 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 entspre- chend der PLCopen-Spezifikation, sowie weitere FB, die Aufgaben der Bewegungser- zeugung 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 entspre- chend der PLCopen-Spezifikation, sowie weitere FB, die Aufgaben der Bewegungser- zeugung 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).

# i

### 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 instanziert 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 imweiteren 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:

😓 CoDeSys - McpPLCopenP1.lib						
<u>D</u> atei <u>B</u> earbeiten <u>P</u> rojekt <u>E</u> infügen E <u>x</u> tras <u>O</u> nline <u>F</u> enster <u>H</u> ilfe						
Bausteine Mcp_P1 MCV FBs MCV Functions MCV Util FBs MCV Util FBs MCV Util FBs MCV Depen_P1 Multiple Axis MCV_P1_Platform (FB) Bau Dat Visu Res						
JUNLIN	E JUB JEESEN					

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	
MC_ReadBoolParameter*		MC_MoveVelocity	MC_Phasing	
MC_WriteParameter		MC_Home	MC_GearInPos*	
MC_WriteBoolParameter*		MC_Stop		
MC_ReadActualPosition		MC_PositionProfile*		
MC_Reset		MC_VelocityProfile*		
MC_TouchProbe		MC_AccelerationProfile*		
MC_AbortTrigger		MC_TorqueControl*		
MC_ReadDigitalInput*		MC_MoveContinuous*		
MC_ReadDigitalOutput*		MC_Halt		
MC_WriteDigitalOutput*				
MC_SetPosition				
MC_SetOverride				
MC_ReadActualVelocity*				
MC_ReadActualTorque*				
MC_DigitalCamSwitch*				

### 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:

- a) Die generischen FB "MCV\_Axis" der PLCopen-Achsen werden in der ISG-MCP instanziert und sind im FB MCV\_P1\_PLATFORM implementiert.
- b) 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.
- 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\_P1\_PLATFORM muss das HLI (Schnittstelle zum MC) initialisiert sein und die Instanz des FB MCV\_PlatformBase die erfolgreiche Initialisierung der MCP melden.

兔 CoDeSys - Frame_PLCopenP1.pro*	- [MAIN (PRG-ST)]	
🎭 <u>D</u> atei <u>B</u> earbeiten <u>P</u> rojekt <u>E</u> infügen E	<u>x</u> tras <u>O</u> nline <u>F</u> enster <u>H</u> ilfe	- 8 ×
12 <b></b>	🖁 🔏 🖻 📾 🙀 🙀	
Bausteine Application MAIN (PRG) UserInitialisations (FUN)	0001       PROCRAM MAIN         0002       VAR         0003       Hli :: MCV_HliInterface; (* HLI interface - have         0004       PlatformBase :: MCV_PlatformBase; (* FLC platform - have         0005       Pl_Platform :: MCV_Pl_Platform; (* Motion platform - have         0006       UserInitialisationDone :: BOOL := FALSE; (* Error at initialisation         0007       HliInitError :: BOOL := FALSE; (* Initialisations that an         0008       UserInitialisationDone :: BOOL := FALSE; (* Initialisations that an         0010       (* Request description of the HLI from the CNC *)         0001       (* Request description of the HLI finished successful and if         0001       (* Check if initialization of HII finished successful and if         0003       errors occured during initialization phase. *)         0006       THEN         0007       FlatformBase();         0018       THEN         0019       PlatformBase.cone = TRUE THEN         0012       (* Do the initialization ve do once the PIC starts up. *)         0012       IF UserInitialisationDone := UserInitialisations(dummy:=TRUE);         0014       (* Get the result of the user defined initialization *)         0015       END_IF;         0016       P1_PLATFORM();         0022       (* Insert y	to call until s to call each PI i of HLI interfere necessary for
	Tunlementation der Task 'Standard'	
	Uberprifen 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)	
📄 Bau 📲 Da 📮 Vis 끓 Re		>
	I [Z.: 1, Sp.: 1	ONLINE ÜB LESEN

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

🏩 MULTIPROG - Frame_PlcopenP1 - [Main:Mai	in] 📃 🗖 🔀
Datei Bearbeiten Ansicht Projekt Code Objekte	Layout Online Extras Eenster ?
🗋 🗅 🚅 🖬 🖪 🖧 🎒 🕺 🛍 🛍 🗠 🗠 🎯	. Q 🖪 🗆 🔨 😨 🗔 🖽 🗃 👪 22 📻 🛛 🕸 🏔 🖽
🛛 🔁 🍓 🝓 🔄 🔍 🖓 👉 🛤 🕮 👐 🞕	
Projektbaum-Fenster	<pre>1 (* Programm von ISG *) 2 3 PlatformBase(); 4 5 IF PlatformBase.Done = TRUE THEN 6 P1_PLATFORM(); 7 END_IF; 8 9</pre>
	E Main:Main
Variable Hi Watch 1 Watch 2 \	able APDE/Art
Drücken Sie F1 für Hilfe	Zeile 7, Spalte 1 D: >2GB

# 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\_ar-ray\_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-/Ausgabevariable 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 Achsgruppenabbbildung 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 instanziert 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.

🕏 CoDeSys - Frame_PLCopenP1P4.pr	o - [MAIN (PRG-ST)]	
🎭 Datei Bearbeiten Projekt Einfügen Ex	tras <u>O</u> nline <u>F</u> enster <u>H</u> ilfe	- 8 ×
" <mark>````````````````````````````````````</mark>		
Bausteine Application MAIN (PRG) UserInitialisations (FUN)	0001       PROGRAM MAIN         0002       VAR         0003       Hli       : MCV_HliInterface;       (* HLI interface - have         0004       PlatformBase : MCV_PlatformBase;       (* FLC platform - have         0005       P1_Platform : MCV_P1_Platform;       (* Motion platform - have         0006       F4_Platform : MCV_P4_Platform;       (* Motion platform - have         0007       0007       : BOOL := FALSE;       (* Error at initialisation         0009       UserInitialisationDone : BOOL := FALSE;       (* Initailisations that at a	to call to call to call to call to call n of HLI re neces
	0001 (* Request description of the HLI from the CNC *) 0002 Hli(Start := TRUE);	<u> </u>
	<pre>10003 10004 10004 10005 10006 10006 10007 10010 10010 10010 10010 10010 10010 10010 10010 10011 10011 10012 10012 10013 (* 10014 (* Do the initialization we do once the PLC starts up. *) 10015 1016 (* Get the result of the user defined initialization *) 1017 1018 1016 (* Get the result of the user defined initialization *) 1017 1018 1019 1019 1019 1019 1019 1010 1010</pre>	
	0025         (* Insert your PLC application code after this comment *)           0026         (* *)           0027         *)	~
	<pre>Lade Bibliothek 'C:\isgnc\plc\McpPLCopenP4.lib'</pre>	
Bau Ca 🛱 Vis 🛺 Re		>
	[ [Z.: 21, Sp.: 18   ONLINE	ÜB LESEN

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

kernel

😩 MULTIPROG - Frame_PlcopenP1P4 - [Main:N	lain]
Datei Bearbeiten Ansicht Projekt Code Objekte	Layout Online Extras Eenster ?
0 🗳 🖬 🔃 🚭 👗 🖻 🖻 🗠 🗠 🔍	Q 🖪 🗖 🌂 😧 🗔 🔜 🗃 🚳 🚜 22 🔙 🛛 🕸 🏦
<b>1 1 1 1 1 1 1 1</b>	·····································
Projektbaum-Fenster	<pre>1 (* Programm von ISG *) 2 3 PlatformBase(); 4 5 IF PlatformBase.Done = TRUE THEN 6 P1_PLATFORM(); 7 P4_PLATFORM(); 8 END_IF; 9 10</pre>
Coningulation: In C_40  PCWin32: PCOS_NT  PCWin32: PCOS_NT  Tasks  Tasks  Coningulation  Tasks  Coningulation  Tasks  Coningulation  Coningulation  Configuration  Configu	
(* <b>6 9 6</b>	🔲 Main:Main 📑 Global_Vari
Variable Wert Hi ??? Watch 1 (Watch 2 ) Watch 3 )	<b>()</b> Code $\langle$ Fehler $\rangle$ Warnungen $\lambda$ Infos $\lambda$ SPS-Fehler $\lambda$ Di
Drücken Sie F1 für Hilfe	Zeile 10, Spalte 1 D: >2GB //

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 **gA-xesGroupRef** 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	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	Тур	Verwendung	В	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		%MV1.44					
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 Setzten 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 einkompiliert.

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.

kernel

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\_ER-ROR 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 Anhieb 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\_ER-ROR 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 beauftragt 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-Typ*)
    bf type
                     : WORD;
   behebungs klasse : WORD;
                                (*Fehlerbehebungsklasse*)
    reaktions klasse : WORD;
                                (*Fehlerreaktionsklasse*)
    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\_PLATFORM instanziert 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\_PLATFORM instanziert, 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

- Endschalterüberwachung (nicht wirksam bei Moduloachsen): wird eine Verfahrwegsgrenze 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 werde.

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", "Com- mandAborted" 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 vorzugehen.

```
IF MC MoveVelocity Active > 0 THEN
                             := AxisReference,
 MC MoveVelocity( Axis
                   Execute
                                := Execute_MoveVelocity,
                   Velocity := Velocity_MoveVelocity,
                   Acceleration := Acceleration MoveVelocity,
                   Deceleration := Deceleration MoveVelocity,
                   Jerk
                              := Jerk MoveVelocity,
                                := Direction MoveVelocity );
                   Direction
  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;
                                    = 2
  ΙF
       MC MoveVelocity Active
                                            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).

infügen		×
<u>N</u> ame:		OK
Main	- Sprache	Abbrechen
Programm	C A <u>W</u> L	Hilfe
C Funktion C Funktions <u>b</u> austein	⊙ S <u>I</u> ○ A <u>S</u> ○ <u>F</u> BD	
C Aktion	C KOP	<u>R</u> eserve verwenden

Abb. 16: Anlegen des Programms Main in ST

nfügen		
<u>N</u> ame:		0K
HelloWorld		Abbrechen
Тур	Sprache	Abbiechen
Programm	C AWL	<u>H</u> ilfe
C Funktion	C SI	
C Funktions <u>b</u> austein	C AS • FBD	
$oldsymbol{C}$ Aktion	C KOP	<u>R</u> eserve verwenden
C Transition	C MAS	
C Schritt		

Abb. 17: Anlegen von Programm HelloWorld in FBD

Nach dem Anlegen erscheinen die Programme im Projektbaum unter "Logische POEs"

MULTIPROG 4.6 Suite - HelloWorld - [H	elloWo	orld:He	lloWor	ld]									
Datei Bearbeiten Ansicht Projekt Cod	e O <u>n</u> li	ine E <u>x</u> t	ras <u>?</u>										Ð×
<u> </u> D 📽 🛛   B, ⊕   X 🖻 E   ≤	2	•	1	3 🗆	×	<b>(</b>	<b>.</b>	ß	?? 18	?.?			2 🎬
<b>1 1 1 1 1 1</b>	n Kuno	输入	} <b>«</b> •••			曲	<b>1</b>	l	- 8		нној н	⊢⊩⊣⊢	o ihr
Projektbaum-Fenster 🗾 🔀	-												-1
Projekt : D:\cc\v262.kmc\plc\▲     in → → → Bibliotheken	2		15	22	25	22	2	2	33	2	15	8	
⊡…© Datentypen ⊡…© Logische POEs		27	123	3		82	12		82	32	13	33	
	- 13		8	08	20	69		20	2		<b>x</b> 3	2	
HelloWorldT	12	12	e.	12		8	2	<u>.</u>	8	2	28	85	
		\$	12		·		·			*	•		
												•	
	Ф.н	lelloWor	ld:										

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

<b>FB-</b> Тур	Instanzname	Bemerkungen
MCV_PlatformBase	MCV_PlatformBase_1	Stellt Verbindung zu MC her
MCV_P1_PLATFORM	MCV_P1_PLATFORM_1	Aktualisiert zyklisch die Achsreferen- zen

#### 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	<b>HelloWorld</b>	verwendeten FB
------------------	----------	-------------------	----------------

PLCopen-FB	Instanzname	Bemerkungen
MC_Power	MC_Power_1	Dient zum Setzen der Regler- und Vor- schubfreigabe
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 instanzierten Funktionsblöcke im Programm HelloWorld



Abb. 20: Im Programm HelloWorld instanzierte 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-Varibalen

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	

MULTIPROG 4.6 Suite - HelloWorld	[HelloWorld¥:HelloWorld]			
Datei Bearbeiten Ansicht Projekt	<u>C</u> ode O <u>n</u> line E <u>x</u> tras <u>?</u>			
	≌ ≅ @ @ <b>⊡</b> □	🔨 😨 🗟 🗃  🕷	1 22	🖽 👗 🔯 f
	鄒 🗠 稽 🖷 🖷 📗 🖤	, =     ←   ☆ = •	OHF HHF [KHH] []	~~ 다 다 나
Projektbaum-Fenster	Name	Typ	Verwendung	BA Anfangsw
Bibliotheken	🖂 Default			
	EnablePower	BOOL	VAR	
	EnablePositive	BOOL	VAR	3236
	EnableNegative	BOOL	VAR	
HelloWorldT	StartMotion	BOOL	VAR	
HelloWorldV	EnableReadActPos	BOOL	VAR	
HelloWorld	Position	REAL	VAR	
	Distance	REAL	VAR	100000.0
🕂 💮 Hardwarestruktur	Velocity	REAL	VAR	10000.0
	Acceleration	REAL	VAR	2000.0
	Deceleration	REAL	VAR	2000.0
	Jerk	REAL	VAR	2000.0
	Done	BOOL	VAR	
	MC_Power_1	MC_Power	VAR	
	MC_ReadActualPosition_1	MC_ReadActualPosition	VAR	
	MC_MoveRelative_1	MC_MoveRelative	VAR	
	g_array_axis_ref	ARRAY_AXIS_REF	VAR_EXTERNAL	
	Main:Main HelloWorld	t 🔝 HelloWorldV:		

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 Hello-World aufruft, müssen zusätzlich die erforderlichen "Globalen Variablen" am Besten in einer eigenen Gruppe zur besseren Übersichtlichkeit angelegt werden.

🛱 MULTIPROG - HelloWorld - [Global_Variables:Configuration.PCWin32]								
Datei Bearbeiten Ansicht Projekt Code Layout Online Extras Fenster ?								
┗┏┏┓┓,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,								
· · · · · · · · · · · · · · · · · · ·								
Projektbaum-Fenster 📃 🗾								
Projekt : D:\cc\v263 sbv pc285\		Name	Тур	Verwendung	В	Adresse	Anfangsw	
🕂 🦳 Bibliotheken		E MCP	•					
🕂 🧰 Datentypen		MAX_RESET_RETRIALS	UDINT	VAR_GLOBAL	Τ		50000	
🗏 🔄 Logische POEs		MAX_RESET_WAIT_CYCLES	UDINT	VAR_GLOBAL	Γ		1000000	
		MAX_RETRIALS	UDINT	VAR_GLOBAL	Γ		0	
i HelloWorldT		g_order_id	UDINT	VAR_GLOBAL			1	
HelloWorldV*		g_axis_idx_offset	INT	VAR_GLOBAL	<u>!</u> .		0	
HelloWorld*		g_array_axis_ref	ARRAY_AXIS_REF	VAR_GLOBAL	Γ			
		gAxesGroupRef	ARRAY_AXES_GROUP	VAR_GLOBAL	T			
E Configuration : IPC 40		hli	HIGH_LEVEL_INTERFACE	VAR_GLOBAL		%MB3.0		
PCWin32 : PCOS NT		MAX_USED_INSTANCES	INT	VAR_GLOBAL	T		16	
Tasks		NR_CYCLES_CHK_MC_RUNS	UINT	VAR_GLOBAL	t		10	
Global Variables*		NR_MAX_PLC_CYCLES_CH	UINT	VAR_GLOBAL	T		1000	
ID Configuration								
		gHwines	MCV_ARRAY_HVV_INCS	VAR_GLOBAL	Т			
		gKeyDirCur	MCV_ARRAY_KEY_DIR	VAR_GLOBAL	T			
			MOV ADDAXC KDC DID	VAR ALABAL	1			
🕐 🖻 💴 📶		Global_Varia						

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.

PCWin32		PCWin32	_ _  ×
Status: Ein		Status: Stop	
<u>S</u> top	Kalt	Stop	<u>K</u> alt
<u>R</u> ücksetzen	<u>₩</u> arm	<u>R</u> ücksetzen	<u>W</u> arm
	Heiß		Неі <u>в</u>
S <u>e</u> nden	Hochladen	S <u>e</u> nden	Hochjaden
Eehler	Info	Eehler	<u>I</u> nfo
S <u>c</u> hließen	Hilfe	S <u>c</u> hließen	Hilfe

#### 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:

0 10 10 N	MC_Powe	r i	20	22	23	20	122		64	MC_R	eadA	ctual	Position	13. 192	122	÷.	64
g_array_axis_ref[0]-	- Axis	Axisg_a	array_	axis_rel	f[0]	g_ari	ray_a:	xis_ref	T0]—	Axis	. <u> </u>	2010-002	- Axis-	g	array	_axis_	ref[0]
19 - 19 - 19 Carl					- R										•	1.	
EnablePower-	Enable	Status 🛶				Ena	ableRe	adActi	Pos-	Ena	ble		Valid	-•			
EnablePositive -	Enable Positive	Error -9	8.50	83	-			1.5	0				Error		353	10	194
			33	13	-28				39						22		89
EnableNegative-	Enable_Negative	ErrorID -0										E	rrorID	-			
(		and the state of the	81	22	135	<u>.</u>	15	12	32							12	32
	Dr				• • • •							1	osition			00E+0	100.
	3	DriveReady												0.	00000	DOL	
SI N. K. S		1	-	83	132	10	2	38	335	10	12	38	33	<b>3</b> 8		38	33
				83	13	12		52	53	:8			33	58			37
	MC MayoRal	stive of	1	32	13		1	32	191	15		33	121	28		33	191
	MC MoveRe	elative															
g array aris ref[0]-	- Avis	Avia	1 arrai	avis	reff01			1	SS.		12		S.		12		S.
g_anay_axa_re[o]	1 Iola	T Inia		y_ania_	101[0]												
StartMotion-	Execute	Done D	Done		•••				200								
Distance-	Distance Com	mandAborted 🗝	)	14	73		25		85	10			35	10	5		85
1.0000000E+005	and the	124	12	12	135		35		32	10	125		32	10	123		32
1 000000E ±00/	Velocity	Error -•															
Acceleration-	Acceleration	ErrorID -	1	22	-99		•		69		•	•	69		•		<u></u>
2.0000000E+003				2.2		22			2.	22	121		25	22	121		22
Deceleration-	Deceleration	(5.)															
2.0000000E+003	Taula	10	200	63	- 63	8		18	68		18	18	69	63	$ \mathbf{k} $	08	68
2.0000000E ±003	Jerk																
			1	1.5	568	:	27	1	852	:1	22		852	20	22		832

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-/Ausgabevariablen überschrieben werden. Zuerst wird die Regler- und Vorschubfreigabe für den Antrieb der Achse gegeben.





### 4.1.11 Schritt 11: Setzen der Freigabe für PLCopen-FB

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

orcen/Uberschreiben	Breakpoint
StartMotion	<u>S</u> etzen
Weit	<u>R</u> ücksetzen
• TRUE C FALSE	<u>A</u> lle rücksetzen
Forcen Forcen zurück Überschreiben Forceliste rücksetzen	Angez. Wert Stangard Dezimal Hexadezimal Binär REAL-Werte Breite: Nachkomma: 15 7
Schließen Info	Hilfe

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\_MoveReltive\_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.

	a a a a	MC_Powe	r	20		22	83	20		68	- 64	MC_R	eadA	ctuali	Position	1.0		68	69
6	g_array_axis_ref[0]—	- Axis	Axis	-g_ar	ray_a	xis_re	f[0]	g_ari	ray_a	xis_rel	f[0]—	- Axis	s —		- Axis-	-g.	array	_axis	_ref[0
	EnablePower-	Enable	Status 🛶	Ċ.		3	-	Ena	ableRe	adAct	Pos-	Ena	ble		Valid	-	•	18	28
	1 EnablePositive —	Enable Positive	Error			13	6			1.	1	- Aller			Error	L.	•	18	69
	1 EnableNegative—	Enable_Negative	ErrorID -0	•		12	53		22	3	82			E	rrorID	-	22	3	82
	1	Di	riveEnabled	e.	8		15		15	12	37			P	osition	P(	ositior	1	
		3	DriveReady 🗕		- 25	82	100		18		87					1.		UUE +1	
				2	1		102	1	ŝ.	18	385		ŝ.	38	365	28	ŝ.	1	105
	a w w w		23 N.	23	1	83	102	8	2	1	35	38	2	3	35	22		1	
		MC_MoveRel	ative <u>1</u>	2		112	135	2	1	2	121	26	1		23.	20	4	2	121
2	g_array_axis_ref[0]—	- Axis	Axis		_array	_axis_	ref[0]			52	33 <b>.</b>			62	33 <b>.</b>		1.2	62	÷.
	StartMotion-	Execute	Done	-0	one		12			1.	S.		.2		S.		12	12	S.
	Distance	Distance Com	mandAborted	-• <sup>1</sup>	12	14	73	2		3	35		25	9	85	13	35	3	85
	Velocity-	Velocity	Error	•	120	24	-33		1		35	10		2	84	10			34
	Acceleration-	Acceleration	ErrorID	-•			-0		•		69		•		6	19	•		8
	Deceleration	Deceleration		80		10	9		×.	18	68	83	×.	1	69	83	1	1	68
	2.0000000E+003 Jerk—	Jerk		80		1	9	•	•	18	69	63	•	1	6	83	1	12	6
	2.0000000E+003	the state of the s		1	12	23	53		$\mathcal{R}$		82	<u>.</u>	22	3	82	<u>.</u>	22		852
					1.0														

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:

Neuer Baustein		×
<u>N</u> ame des Bausteins:	HelloWorld	ОК
_ Typ des Bausteins	<u> </u>	Abbrechen
Programm	C A <u>W</u> L	
C Funktions <u>b</u> lock	C KOP	
C Eunktion	C FU <u>P</u>	
	⊂ <u>A</u> S	
BOOL	C SI	
	CFC	

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 in	n Programm	HelloWorld	verwendeten FB
------------------	------------	------------	----------------

PLCopen-FB	Instanzname	Bemerkungen
MC_Power	MC_Power_1	Dient zum Setzen der Regler- und Vor- schubfreigabe
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 aktuel- Ien Position.

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



Abb. 34: Im Programm HelloWorld instanzierte 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.



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-Varibalen

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:



CoDeSys - HelloWorld.pro - [HelloWo	rld (PRG-CFC)] Evtrar Oplina Fanctar Hilfa	
Bausteine         0000	Image: Second	
	<pre>g Jerk : REAL := 2000; Done : BOOL; StartMotion : BOOL; END_VAR MC_Power_1 EnablePower EnablePower EnablePositive DriveEnabled EnablePositive DriveReady EnableNegative Enable_Negative Status g_array_axis_ref[0] Axis P Error ID Axis</pre>	<u> </u>
	MC_MoveRelative_1 MC_MoveRelative_1 MC_MoveRelative MC_MoveRelative MC_MoveRelative MC_MoveRelative Done Distance Distance Distance CommandAborted Velocity Velocity Acceleration Deceleration Deceleration Jerk Jerk g_array_axis_ref[0] MC_MoveRelative_1 MC_Move	)one
1	MC_ReadActualPosition_13 MC_ReadActualPosition MC_ReadActualPosition G_array_axis_ref[0] Axis ErrorID Position Position Axis	
	degröße: 47969 Bytes	

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

### 4.2.6 Schritt 6: Programm HelloWorld in Programm MAIN einfügen

In der Beispielapplikation Frame\_PLCopenP1.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:

🎭 CoDeSys - HelloWorld.pro*	* - [MAIN (PRG-ST)]
😒 Datei Bearbeiten Projekt	Einfügen Extras Online Eenster Hilfe
Bausteine Application MAIN (PRG) UserInitialisations (FUN)	Image: Construction of the second

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:

CoDeSys - HelloWorld.pro* - [Taskkonfiguration]	
🔀 Datei Bearbeiten Projekt Einfügen Extras Online Fenster	Hilfe
<u> </u>	
Ressourcen     Bibliothek hli.lib 31.10.0     Bibliothek hli.lib 31.10.0     B'	Taskeigenschaften         Name:       Standard         Priorität (031):       0         Typ       C         Zyklisch       C         Ereilaufend       C         Ereilgnisgesteuert       C         Eigenschaften       Eigenschaften         Ereignis:       SercosTick

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. Erkenntlich 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 haben einen Fehler gefunden, 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

