1 CXA_MotionInterface.library

1.1 Einführung und Übersicht

Allgemeines

Die SPS-Bibliotheken CXA_MotionInterface und CXA_MotionInterfaceUser und das dazugehörige Programm-Template stellen Funktionen zur Ansteuerung von Achsen und Kinematiken zur Verfügung.

Alternativ zu den Einzel-Funktionen der SPS-Bibliotheken CXA_Motion bzw. CXA_PLCopen ist das MotionInterface ein fertiger Programmrahmen und stellt ein einfach zu bedienendes Interface für die Achs- bzw. Kinematik-Funktionalität zur Verfügung.

Weniger Code und leistungsfähigere Kommandos beschleunigen die Programmentwicklung von Applikationen.

*Zur Nutzung der Bibliothek muss die App **rexroth-motion** installiert sein.*

Komponenten des MotionInterface

- CXA_MotionInterface.compiled-library: Basisbibliothek mit den Basis-Funktionsbausteinen und Basis-Strukturen.
- CXA_MotionInterfaceUser.library: offene Bibliothek mit erweiterten Funktionsbausteinen, erweiterten Strukturen, Programmen und Visualisierungen. Hier sind auch die globalen Variablen der Interfaces instanziiert. Hier sind Anpassungen / Erweiterungen der Interfaces durch den Anwender möglich.
- Programm-Template "ctrlX CORE Axis/Kin-Interface": enthält den Aufruf der notwendigen Programme und Beispielcode. Das Template wird angeboten beim Anlegen eines neuen Projektes im **ctrlX PLC Engineering**.

Achs-Interface

Das Achs-Interface enthält in der Kontrollstruktur Steuersignale und Parameter für die verschiedenen Betriebsarten der Achsen. In der Statusstruktur werden Istwerte, Statusbits, Diagnoseinformationen und Quittungen für die Betriebsarten zur Verfügung gestellt.

Die Strukturen werden als Arrays zur Verfügung gestellt. Dies erlaubt FOR-Schleifen über alle Achsen. Die Adressierung der Achsen erfolgt über den Index in den Strukturen. Die Zuordnung des Achs-Index zum Achs-Namen kann automatisch erfolgen oder durch den Anwender definiert werden (siehe Programm-Template).

Tab. 1: Zuordnung Anwender-Interface zur Achs-Interface-Datenstruktur:

Anwender-Interface	Typ	Beschreibung
arAxisCtrl[ ]_gb	TE_AXIS_CONTROL_TYPE01	Steuerungsstruktur inklusive Sollwerte und Variablen zum Aktivieren der Betriebsarten
arAxisStatus[ ]_gb	TE_AXIS_STATUS_TYPE01	Statusstruktur inklusive Diagnoseinfo, Quittungen für die Betriebsarten, Istwerte und Statusbits

Kinematik-Interface

Das Kinematik-Interface enthält in der Kontrollstruktur Steuersignale und Parameter für die verschiedenen Betriebsarten der Kinematiken. In der Statusstruktur werden Istwerte, Statusbits, Diagnoseinformationen und Quittungen für die Betriebsarten zur Verfügung gestellt.

Die Strukturen werden als Arrays zur Verfügung gestellt. Dies erlaubt FOR-Schleifen über alle Kinematiken. Die Adressierung der Kinematiken erfolgt über den Index in den Strukturen. Die Zuordnung des Kinematik-Index zum Kinematik-Namen kann automatisch erfolgen oder durch den Anwender definiert werden (siehe Programm-Template).

Das Kinematik-Interface nutzt intern das Achs-Interface. Es ist also zwingend erforderlich auch das Achs-Interface aufzurufen.

Tab. 2: Zuordnung Anwender-Interface zur Kinematik-Interface-Datenstruktur:

Anwender-Interface	Typ	Beschreibung
arKinCtrl[ ]_gb	TE_KINEMATICS_CONTROL_TYPE01	Steuerungsstruktur inklusive Sollwerte und Variablen zum Aktivieren der Betriebsarten
arKinStatus[ ]_gb	TE_KINEMATICS_STATUS_TYPE01	Statusstruktur inklusive Diagnoseinfo, Quittungen für die Betriebsarten, Istwerte und Statusbits

IMC-Interface

Das IMC-Interface (**I**nterface-**M**otion-**C**ontrol) enthält in der Kontrollstruktur Steuersignale für die App **rexroth-motion**. In der Statusstruktur werden Statusbits und Diagnoseinformationen der App **rexroth-motion** zur Verfügung gestellt.

Das IMC-Interface wird intern von Achs- und Kinematik-Interface genutzt um bei Erreichen des Zustandes "Running" die Initialisierung zu starten.

Das Programm MB_ImcInterface wird bereits vom Achs-Interface intern aufgerufen. Ein zusätzlicher Aufruf ist nur notwendig, wenn das IMC-Interface ohne Achs-Interface genutzt wird.

Tab. 3: Zuordnung Anwender-Interface zur IMC-Interface-Datenstruktur:

Anwender-Interface	Typ	Beschreibung
ImcCtrl	MB_IMC_CONTROL_TYPE01	Steuerungsstruktur zur Vorgabe des Motion Modus und zum Fehler löschen
ImcStatus	MB_IMC_STATUS_TYPE01	Statusstruktur mit dem aktuellen Motion Modus und Diagnoseinformationen

1.2 MotionInterface - Erstkonfiguration

Bevor die Funktionalität des MotionInterfaces benutzt werden kann, muss dieses zuerst initialisiert werden. Die folgenden Schritte aktivieren das Achs- und Kinematik-Interface in ctrlX PLC Engineering.

Viele der in diesem Abschnitt beschriebenen Schritte sind schon im Programm-Template "ctrlX CORE Axis/Kin-Interface" enthalten. Dieses Template kann in ctrlX PLC Engineering importiert und als Leitfaden für neue Projekte benutzt werden. Bei Verwendung der Vorlage "ctrlX CORE Axis/Kin-Interface" ist das Handling des Achs- und Kinematik-Interface inklusive möglicher Erweiterungen durch den Anwender komplett lauffähig ausprogrammiert.

Erste Schritte bei Verwendung des Beispielprojektes "ctrlX CORE Axis/Kin-Interface" .

1. Im ctrlX PLC Engineering ein neues Projekt anlegen. Im Dialog das Template "ctrlX CORE Axis/Kin-Interface" auswählen. Weitere Schritte der Dialogabfolge fertigstellen.
2. Das eingefügte Programm ist auf einer "ctrlX COREvirtual" komplett lauffähig. Für die Aktualisierung der "ctrlX COREvirtual" oder der Nutzung realer Hardware rechte Maustaste auf das Device und "Gerät aktualisieren". Gewünschte Steuerung "ctrlX CORE" oder "ctrlX COREvirtual" auswählen.
3. Doppelklick auf das Device und die Kommunikationseinstellungen vornehmen. Sie können die SPS-Applikation einloggen, starten und über die Visualisierungen "OverviewAxes" bzw. "OverviewKinematics" die Funktion testen. Die Zuordnung AchsName <> AchsIndex bzw. KinName <> KinIndex erfolgt per Default automatisch.
4. Im Bibliotheksverwalter die Bibliothek CXA_MotionInterfaceUser auswählen und die Biblitotheksparameter "MOTIF_CONFIG" öffnen. Hier können Sie die Arrays an die Anzahl Ihrer Achsen bzw. Kinematiken anpassen. Wählen Sie einen anderen Konfigurationsmodus aus, um die Zuordnung AchsName <> AchsIndex bzw. KinName <> KinIndex selbst zu definieren.
5. Benutzen Sie nun arAxisCtrl_gb[ ], arAxisStatus_gb[ ], arKinCtrl_gb[ ], arKinStatus_gb[ ] zum Programmieren Ihrer Applikation.

Erste Schritte bei Nutzung eines eigenen Programmes.

1. Öffnen Sie den Bibliotheksverwalter im Projektbaum unter "Logic->Application" und fügen Sie die Bibliotheken **CXA_MotionInterface.compiled-library** und **CXA_MotionInterfaceUser.library** zum aktuellen ctrlX PLC Engineering Projekt hinzu.
2. Rufen Sie z.B. im PlcProg die Programme **TE_AxisInterfaceMainProg** und **TE_KinInterfaceMainProg** aus der Bibliothek **CXA_MotionInterfaceUser.library** auf. In der Dokumentation der Bibliothek **CXA_MotionInterfaceUser.library** ist entsprechender Beispielcode hinterlegt im Programm "KinInterfaceUser/_Examples/Example_KinIfApplicationPart". Alternativ im ctrlX PLC Engineering ein neues Projekt mit dem Template "ctrlX CORE Axis/Kin-Interface" anlegen und mit Export/Import die benötigten Teile in das eigene Programm übertragen.

3. Im Bibliotheksverwalter die Bibliothek CXA_MotionInterfaceUser auswählen und die Biblitotheksparameter "MOTIF_CONFIG" öffnen. Hier können Sie die Arrays an die Anzahl Ihrer Achsen bzw. Kinematiken anpassen. Wählen Sie einen anderen Konfigurationsmodus aus, um die Zuordnung Achsname <> Achsindex bzw. KinName <> KinIndex selbst zu definieren.
4. Optional: wenn sie zur Inbetriebnahme der Achsen die mitgelieferten Visualisierungen nutzen möchten, legen Sie eine neue Visualisierung mit dem Namen "OverviewAxes" an.
   - Oben legen Sie einen neuen Frame an und wählen dafür die Visualisierung "OverviewAxesHeader" aus der Bibliothek **CXA_MotionInterfaceUser.library** aus.
   - Darunter für jede Achse einen Frame "OverviewOneAxis" und übergeben als "m_Input_AxisIndex" den jeweiligen AchsIndex.
5. Optional: wenn sie zur Inbetriebnahme der Kinematiken die mitgelieferten Visualisierungen nutzen möchten, legen Sie eine neue Visualisierung mit dem Namen "OverviewKinematics" an.
   - Oben legen Sie einen neuen Frame an und wählen dafür die Visualisierung "OverviewKinematicsHeader" aus der Bibliothek **CXA_MotionInterfaceUser.library** aus.
   - Darunter für jede Achse einen Frame "OverviewOneKinematics" und übergeben als "m_Input_KinIndex" den jeweiligen KinIndex.
6. Benutzen Sie nun arAxisCtrl_gb[ ], arAxisStatus_gb[ ], arKinCtrl_gb[ ], arKinStatus_gb[ ] zum Programmieren Ihrer Applikation.

1.3 Achs-Interface

1.3.1 Einführung und Übersicht

Das **Achs-Interface** bündelt und erweitert PLCopen-Bewegungsfunktionsbausteine und stellt ein einfach zu bedienendes Interface für die Antriebsfunktionalität zur Verfügung.

Weniger Code und leistungsfähigere Kommandos beschleunigen die Programmentwicklung von Applikationen.

Das Achs-Interface enthält Steuersignale und Parameter für die verschiedenen Betriebsarten der Achsen sowie Einstellmöglichkeiten für angewählte Prozesswerte.

Tab. 4: Programmorganisationseinheiten des Achs-Interface in der Bibliothek CXA_MotionInterface

Anwendungsgebiet (Ordner in der Bibliothek CXA_MotionInterface)	
POUs	Beschreibung
AxisInterface/POUs	

➚ , *Seite*	Wird zur Initialisierung des Achs-Interfaces für eine einzelne Achse benutzt. Der Funktionsbaustein muss nur einmal beim Programmstart oder bei jeder Modusumschaltung von Configuration in Running aufgerufen werden.
➚ , *Seite*	Konfiguration des Achs-Interfaces für eine einzelne Achse. Der Funktionsbaustein muss zyklisch (im Motion-Takt oder langsamer als der Motion-Takt) aufgerufen werden, solange man sich im Betriebsmodus befindet
AxisInterface/DUTs	
	Informationen zu den Datenstrukturen siehe Online-Dokumentation in der Bibliothek **CXA_MotionInterface** im Ordner "AxisInterface/DUTs".
AxisInterface/GlobalVariables	
	Informationen zu den globalen Variablen siehe Online-Dokumentation in der Bibliothek **CXA_MotionInterface** im Ordner "AxisInterface/GlobalVariables".

Das Achs-Interface wird als Programmiertemplate oder als standalone-Interface für die Achsfunktionalität zur Verfügung gestellt.

Wenn es mit dem Programmiertemplate "ctrlX CORE Axis-/Kin-Interface" benutzt wird, muss sich der Anwender nicht mit Instanz-Aufrufen der Funktionsbausteine innerhalb des Projektes befassen. Diese Funktionalität ist komplett in das Template integriert und der Anwender muss nur ein paar Zeilen Code schreiben.

Wird hingegen das Achs-Interface als eigenständige Funktionalität benutzt, erfordert dies das Anlegen von Instanzen von beiden Funktionsbausteinen für jede Achse durch den Anwender.

Tab. 5: Programmorganisationseinheiten des Achs-Interface in der Bibliothek CXA_MotionInterfaceUser

Anwendungsgebiet (Ordner in der Bibliothek CXA_MotionInterfaceUser)	
POUs	Beschreibung
AxisInterfaceUser/POUs	
TE_AxisInitAllAxes	Initialisierung des Achs-Interfaces für alle Achsen. Ruft intern den ➚ , *Seite* für jede Achse auf
TE_AxisInterface erweitert ➚ , *Seite*	Hier kann das Achs-Interface für eine einzelne Achse durch den Anwender erweitert werden ➚ *Kapitel 1.3.7 „Achs-Interface Anwender-Erweiterung “ auf Seite 36*. Der Funktionsbaustein muss zyklisch (im Motion-Takt oder langsamer als der Motion-Takt) aufgerufen werden, solange man sich im Betriebsmodus befindet

CXA_MotionInterface.library

Achs-Interface

TE_AxisInterfaceMainProg	Das Hauptprogramm führt bei Erreichen des Modus "Running" die Initialisierung aus und nach erfolgreicher Initialisierung wird der TE_AxisInterface für alle Achsen aufgerufen
TE_GetAxisInterfaceIndex	Liefert den Index einer Achse, anhand des Achs-Namen
	Weitere Informationen siehe Online-Dokumentation in der Bibliothek **CXA_MotionInterfaceUser** im Ordner "AxisInterfaceUser/POUs".
AxisInterfaceUser/DUTs	
	Informationen zu den Datenstrukturen siehe Online-Dokumentation in der Bibliothek **CXA_MotionInterfaceUser** im Ordner "AxisInterfaceUser/DUTs".
AxisInterfaceUser/GlobalVariables	
Global_AxisInterface	Hier ist das eigentliche Achs-Interface mit den Arrays **arAxisCtrl_gb** und **arAxisStatus_gb** zu finden. Die weiteren Variablen werden intern bzw. von den Visualisierungen genutzt. Siehe auch ↗ *Kapitel 1.3.4 „ Achs-Interface - Globale Variablen “ auf Seite 27*
AxisInterfaceUser/Visualizations	
↗ *Kapitel 1.3.6.4 „Achs-Interface Visualisierungen “ auf Seite 34*	Inbetriebnahmevisualisierungen, z.B. zum Vertippen der Achsen
AxisInterfaceUser/_AxifDebug	
Bei Problemen kann mit Hilfe der POUs in diesem Ordner der Ablauf der Motion-Befehle aufgezeichnet werden.	Informationen zu diesem Debug-Feature siehe Online-Dokumentation in der Bibliothek **CXA_MotionInterfaceUser** im Ordner "AxisInterfaceUser/_AxifDebug". Ein HowTo ist in der Dokumentation des Programmes "TE_AxIfDebugProg" zu finden.
AxisInterfaceUser/_Examples	
PROGRAM Example_AxIfApplicationPart	Beispielcode zur Anwendung des Achs-Interfaces. Im Programm-Template "ctrlX CORE Axis-/Kin-Interface" ist dieser Beispielcode auch enthalten.

Projektierungshinweis/Laufzeitbedarf

Für jede Achse des Achs-Interface wird der Achs-Interface-Funktionsbaustein aufgerufen. Dieser Aufruf benötigt Laufzeit der SPS. Diese Laufzeit variiert je nach Achstyp und Achsbetriebsart.

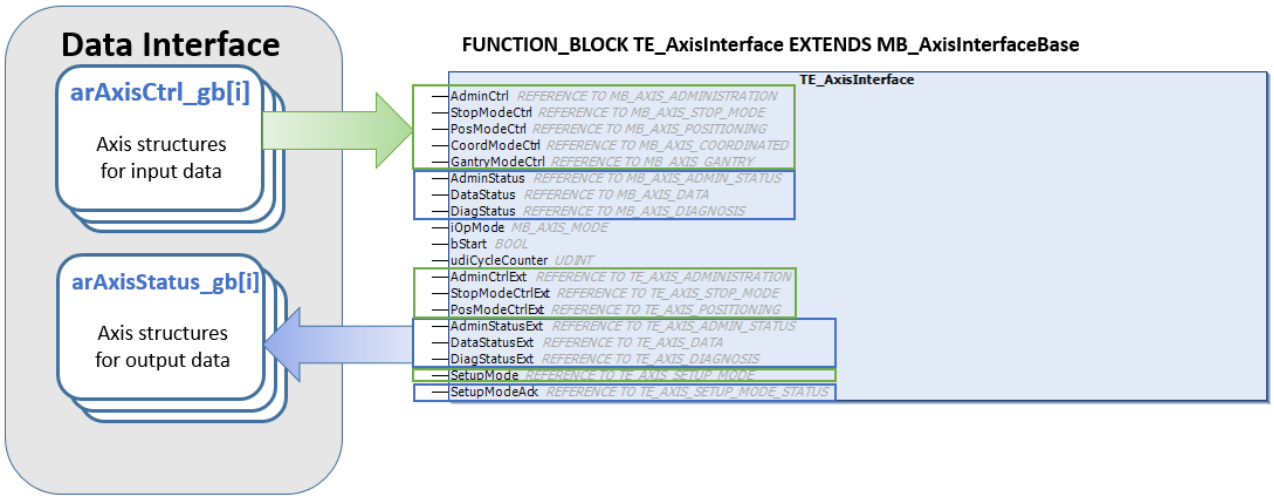


Abb. 1: Achs-Interface Datenstruktur des Interface

Tab. 6: Zuordnung Anwender-Interface zur Achs-Interface-Datenstruktur:

Anwender-Interface	Typ	Beschreibung
arAxisCtrl[ ]_gb	TE_AXIS_CONTROL_TYPE01	Steuerungsstruktur inklusive Sollwerte und Variablen zum Aktivieren der Betriebsarten
arAxisStatus[ ]_gb	TE_AXIS_STATUS_TYPE01	Statusstruktur inklusive Diagnoseinfo, Quittungen für die Betriebsarten, Istwerten und Statusinformation der App **rexroth-motion**.

Achs-Interface

Watch 1

Expression	Type	Value	Prep
Global_AxisInterface.arAxisCtrl_gb[0]	TE_AXIS_CONTROL_TYPE01		
Admin	TE_AXIS_ADMINISTRATION		
Config	MB_AXIS_ADMIN_CONFIG		
ClearError	BOOL	FALSE	
_OpMode	MB_AXIS_MODE	ModePosAbs	
_OpModeBits	TE_AXIS_MODE_BITS		
MODE_AB	BOOL	FALSE	
MODE_AH	BOOL	FALSE	
MODE_COORDINATED	BOOL	FALSE	
MODE_EXTERNAL_FB	BOOL	FALSE	
MODE_GANTRY	BOOL	FALSE	
MODE_HOMING	BOOL	FALSE	
MODE_POS_ABS	BOOL	TRUE	
MODE_POS_ADD	BOOL	FALSE	
MODE_POS_REL	BOOL	FALSE	
MODE_XX_USER_0	BOOL	FALSE	
RetriggerOpMode	BOOL	FALSE	
StopMode	TE_AXIS_STOP_MODE		
StopJerk	LREAL	0	
StopDeceleration	LREAL	99	
PosMode	TE_AXIS_POSITIONING		
DynValues	MB_AXIS_DYN_VALUES		
Distance	LREAL	42	
Position	LREAL	0	
Velocity	LREAL	10	
CoordMode	MB_AXIS_COORDINATED		
KinName	STRING(15)	'Mover'	
GantryMode	MB_AXIS_GANTRY		
Master	AXIS_REF		
AxisName	STRING(15)	'X_Axis'	
SetupMode	TE_AXIS_SETUP_MODE		
Enable	BOOL	FALSE	
JogPlus	BOOL	FALSE	
JogMinus	BOOL	FALSE	
Vel	LREAL	10	
DynValues	MB_AXIS_DYN_VALUES		
JogIncr	BOOL	FALSE	
StepWidth	LREAL	1	

Watch 2

Expression	Type	Value
Global_AxisInterface.arAxisStatus_gb[0]	TE_AXIS_STATU...	
Admin	TE_AXIS_ADMI...	
_OpModeAck	MB_AXIS_MODE	ModePosAbs
Active	BOOL	TRUE
Name	STRING(15)	'MoverX'
AxisFeatures	MB_AXIS_FEAT...	
LastCmdId	ULINT	71
ActiveCmdId	ULINT	71
ActiveCmdSource	STRING(50)	'AxIf.mPosModeAbs...
ActiveCmdStatus	STRING(50)	'ACTIVE'
CmdDone	BOOL	FALSE
LastCmdStatus	STRING(50)	'ACTIVE'
_OpModeAckBits	TE_AXIS_MODE...	
Data	TE_AXIS_DATA	
Aborting	BOOL	FALSE
ActualAcceleration	LREAL	0
ActualPosition	LREAL	791.860000000009...
ActualTorque	LREAL	0
ActualVelocity	LREAL	-10
CoordinatedMotion	BOOL	FALSE
Disabled	BOOL	FALSE
DiscreteMotion	BOOL	TRUE
DistLeft	LREAL	0
ErrorStop	BOOL	FALSE
PLCopenState	STRING(50)	'DISCRETE_MOTION'
StandStill	BOOL	FALSE
StandStillPending	BOOL	FALSE
Diag	TE_AXIS_DIAGN...	
Error	BOOL	FALSE
ErrorID	ERROR_CODE	DEVICE_ERROR
ErrorIdent	ERROR_STRUCT	
NumberMain	DWORD	16#00000000
NumberDetail	DWORD	16#0C000000
Message	STRING(60)	''
SetupMode	TE_AXIS_SETUP...	
EnableAck	BOOL	FALSE

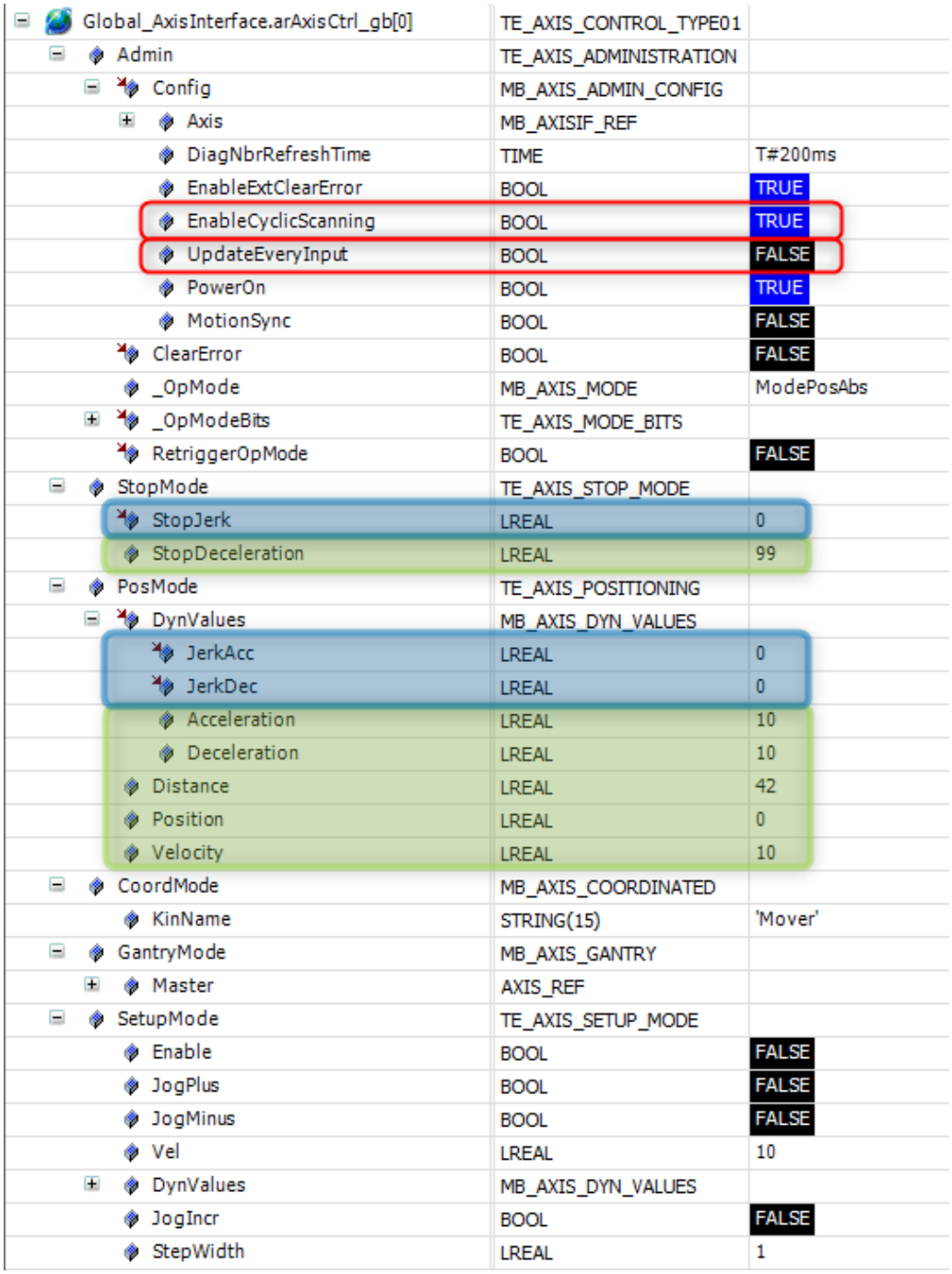
Abb. 2: Überblick über die Datenstrukturen des Achs-Interface

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

EnableCyclicScanning

Die interne Handhabung einiger Sollwerte kann durch das arAxisCtrl_gb[ ].Admin.Config.EnableCyclicScanning Element gesteuert werden.

Wird "EnableCyclicScanning" auf TRUE gesetzt, werden einige Sollwerte der arAxisCtrl_gb[ ]-Struktur zyklisch gescannt und sofort wirksam, wenn sich ein Wert ändert.



Global_AxisInterface.arAxisCtrl_gb[0]	TE_AXIS_CONTROL_TYPE01	
Admin	TE_AXIS_ADMINISTRATION	
Config	MB_AXIS_ADMIN_CONFIG	
Axis	MB_AXISIF_REF	
DiagNbrRefreshTime	TIME	T#200ms
EnableExtClearError	BOOL	TRUE
EnableCyclicScanning	BOOL	TRUE
UpdateEveryInput	BOOL	FALSE
PowerOn	BOOL	TRUE
MotionSync	BOOL	FALSE
ClearError	BOOL	FALSE
_OpMode	MB_AXIS_MODE	ModePosAbs
_OpModeBits	TE_AXIS_MODE_BITS	
RetriggerOpMode	BOOL	FALSE
StopMode	TE_AXIS_STOP_MODE	
StopJerk	LREAL	0
StopDeceleration	LREAL	99
PosMode	TE_AXIS_POSITIONING	
DynValues	MB_AXIS_DYN_VALUES	
JerkAcc	LREAL	0
JerkDec	LREAL	0
Acceleration	LREAL	10
Deceleration	LREAL	10
Distance	LREAL	42
Position	LREAL	0
Velocity	LREAL	10
CoordMode	MB_AXIS_COORDINATED	
KinName	STRING(15)	'Mover'
GantryMode	MB_AXIS_GANTRY	
Master	AXIS_REF	
SetupMode	TE_AXIS_SETUP_MODE	
Enable	BOOL	FALSE
JogPlus	BOOL	FALSE
JogMinus	BOOL	FALSE
Vel	LREAL	10
DynValues	MB_AXIS_DYN_VALUES	
JogIncr	BOOL	FALSE
StepWidth	LREAL	1

Abb. 3: Zyklisch gescannte Elemente von arAxisCtrl_gb[ ] sind hervorgehoben

UpdateEveryInput

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

Beispiel: die Betriebsart "relatives Positionieren" wurde mit den oben gezeigten Werten Velocity=10 und JerkAcc=0 gestartet.

Um den nächsten Positioniervorgang mit einer geänderten Beschleunigung zu starten, wird der Eingang "PosMode.DynValues.JerkAcc" von 0 auf 100 geändert und danach die nächste Distance geschrieben.

Mit dem Ändern der Distance wird auch der Ruck übernommen.

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

Für Inbetriebnahmezwecke stehen verschiedene Visualisierungen, basierend auf den Strukturelementen, die in diesem Abschnitt beschrieben werden, in der Bibliothek CXA_MotioninterfaceUser zur Verfügung.

Was ist neu bzw. geändert gegenüber der Version für MLC/MLD

- Es wurden Teile des ereignisgesteuerten Achs-Interface (Funktionsbaustein MB_AxisInterfaceType11) übernommen. Die Strukturelemente sind zum Teil als Properties implementiert. Die Unterstrukturen sind dann als Funktionsbausteine anstatt Strukturen implementiert um Properties nutzen zu können. In einer Struktur ist eine Methode arAxisCtrl_gb[].Admin.mTriggerMoveCmd() implementiert.
- Die Betriebsartenanwahl arAxisCtrl_gb[].Admin._OpMode ist nicht mehr als "UNION" implementiert sondern als Properties umgesetzt. Bei der Ansteuerung über Bits (_OpModeBits) ist damit keine Mehrfachanwahl mehr möglich.
- Selten verwendete Elemente von arAxisCtrl_gb[].Admin wurden in arAxisCtrl_gb[].Admin.Config verschoben (siehe Tabelle unten).
- Werte vom Typ REAL werden jetzt generell als LREAL in den Strukturen definiert.
- Es gibt keine AxisData[] Struktur. Die aktuellen Istwerte und einige Statusbits sind in arAxisStatus_gb[].Data zu finden.

Tab. 7: Folgende Code-Änderungen sind bei einer Portierung von MLC/MLD mindestens notwendig (Suchen/Ersetzen).

Code MLC/MLD	Ersetzen durch
Kontrollstruktur arAxisCtrl_gb[]	
_OpMode.en	_OpMode
_OpMode.b	_OpModeBits
Admin.StopDeceleration	StopMode.StopDeceleration
Admin.Axis	Admin.Config.Axis
Admin.DiagNbrRefreshTime	Admin.Config.DiagNbrRefreshTime
Admin.EnableExtClearError	Admin.Config.EnableExtClearError
Admin.EnableCyclicScanning	Admin.Config.EnableCyclicScanning
Admin.UpdateEveryInput	Admin.Config.UpdateEveryInput
Admin.PowerOn	Admin.Config.PowerOn
PosMode.Acceleration	PosMode.DynValues.Acceleration
PosMode.Deceleration	PosMode.DynValues.Deceleration
SetupMode.Accel	SetupMode.DynValues.Acceleration
Kontrollstruktur arAxisStatus_gb[]	
Admin.MODE_AH	Admin._OpModeAckBits.MODE_AH
Admin.MODE_COORDINATED	Admin._OpModeAckBits.MODE_COORDINATED
Admin.MODE_EXTERNAL_FB	Admin._OpModeAckBits.MODE_EXTERNAL_FB
Admin.MODE_POS_ABS	Admin._OpModeAckBits.MODE_POS_ABS
Admin.MODE_POS_ADD	Admin._OpModeAckBits.MODE_POS_ADD
Admin.MODE_POS_REL	Admin._OpModeAckBits.MODE_POS_REL

Diese Liste der Code-Änderungen ist nicht vollständig. Bei der Portierung ist eine generelle Überprüfung des Programm-Codes notwendig.

1.3.2 Achs-Interface - Funktionsbausteine

1.3.2.1 MB_AxisInit

Kurzbeschreibung

Der Funktionsbaustein MB_AxisInit wird zur Initialisierung des Achs-Interfaces (↗ *, Seite ,* für eine einzelne Achse benutzt.

Der Funktionsbaustein muss nur einmal beim Programmstart oder bei jeder Modusumschaltung von "Configuration" in "Running" aufgerufen werden. In der Vorlage "ctrlX CORE Axis/Kin-Interface" ist dies bereits implementiert.

Schnittstellenbeschreibung

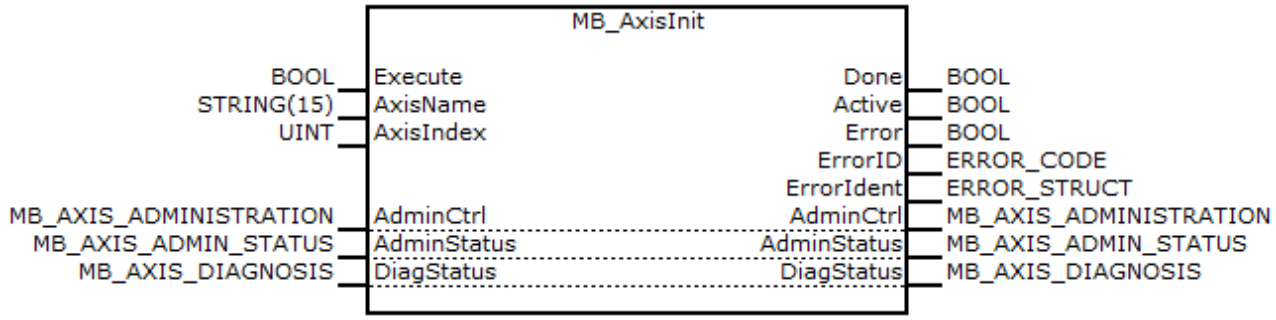


Abb. 4: Funktionsbaustein MB_AxisInit

Tab. 8: Schnittstellenvariablen MB_AxisInit

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	Execute	BOOL	Die Initialisierung wird durch eine positive Flanke an "Execute" gestartet
	AxisName	STRING(15)	Name der Achse
	AxisIndex	UINT	Index in den AchsInterface Strukturen
VAR_OUTPUT	Done	BOOL	Wird gesetzt, wenn der FB die Bearbeitung beendet hat
	Active	BOOL	Wird gesetzt, wenn der FB aktiv ist (nicht im Leerlaufbetrieb)
	Error	BOOL	Zeigt an, dass ein Fehler in der FB-Instanz aufgetreten ist
	ErrorID	ERROR_CODE	Kurzer Hinweis zur Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Information zum Fehler
VAR_IN_OUT	AdminCtrl	MB_AXIS_ADMINISTRATION	Verwaltung der Achse arAxisCtrl_gb[i].Admin anschließen
	AdminStatus	MB_AXIS_ADMIN_STATUS	Status Verwaltung der Achse arAxisStatus_gb[i].Admin anschließen
	DiagStatus	MB_AXIS_DIAGNOSIS	Diagnoseinformationen der Achse arAxisStatus_gb[i].Diag anschließen

Es ist nicht möglich die komplette Instanz der Strukturen (z. B. arAxisCtrl_gb[ ] / arAxisStatus_gb[ ]) über einen Eingang dem Funktionsbaustein zu übergeben.

Dies wurde vorgenommen, um der Anwender-Applikation spezielle Erweiterungen zum bestehenden Code zu ermöglichen. Nähere Details finden Sie unter ➚ , Seite .

Deshalb werden die benötigten Elemente von TE_AXIS_CONTROL_TYPE01 und TE_AXIS_STATUS_TYPE01 als separate Eingänge übergeben.

Der Funktionsbaustein MB_AxisInit initialisiert die folgenden Strukturelemente mit Standardwerten:

Tab. 9: Durch den Funktionsbaustein initialisierte Strukturelemente

Strukturelement	Standard
AdminCtrl._OpMode	ModeAB
AdminCtrl.Config.Axis.AxisNo	AxisIndex
AdminCtrl.Config.Axis.AxisName	AxisName
AdminStatus.Active	TRUE
AdminStatus.Name	AxisName
AdminStatus.Active	TRUE für aktive Achse
AdminStatus.AxisFeatures	Aabhängig vom Achstyp, siehe MB_AXIS_FEATURES in der Bibliothek.

Fehlerbehandlung:

Die Fehlercodes des intern benutzten Funktionsbausteines DL_ReadNode zum Lesen von Datalayer Knoten werden durchgereicht. Der Funktionsbaustein kann die folgenden Fehlercodes erzeugen:

Tab. 10: Fehlercodes des Funktionsbausteins MB_AxisInit

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230101	String AxisName ist zu kurz oder zu lang
STATE_MACHINE_ERROR	16#0A0F0107	16#0C230103	Fehler im Ablauf des Funktionsbaustein

1.3.2.2 MB_AxisInterfaceBase

Kurzbeschreibung

Der Funktionsbaustein MB_AxisInterfaceBase wird zur Konfiguration des Achs-Interfaces für eine einzelne Achse benutzt.

Dieser Funktionsbaustein muss zyklisch (im Motion-Takt oder langsamer als der Motion-Takt) aufgerufen werden solange man sich im Modus "Running" befindet. In der Vorlage "ctrlX CORE Axis/Kin-Interface" ist dies bereits implementiert.

Interfacebeschreibung

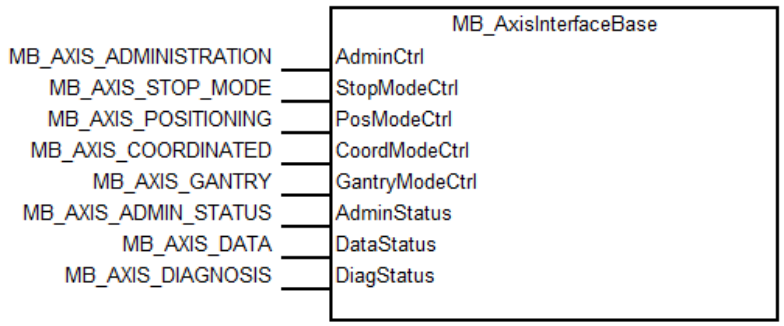


Abb. 5: Funktionsbaustein MB_AxisInterfaceBase

Tab. 11: Schnittstellenvariablen MB_AxisInterfaceBase

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	AdminCtrl	REFERENCE TO MB_AXIS_ADMINISTRATION	Verwaltung der Achse, arAxisCtrl_gb[i].Admin anschließen
	StopModeCtrl	REFERENCE TO MB_AXIS_STOP_MODE	Stoppen der Achse, arAxisCtrl_gb[i].StopMode anschließen
	PosModeCtrl	REFERENCE TO MB_AXIS_POSITIONING	Positionierungsbetriebsarten, arAxisCtrl_gb[i].PosMode anschließen
	CoordModeCtrl	REFERENCE TO MB_AXIS_COORDINATED	Kinematikbetrieb, arAxisCtrl_gb[i].CoordMode anschließen
	GantryModeCtrl	REFERENCE TO MB_AXIS_GANTRY	Betriebsart Gantry, arAxisCtrl_gb[i].GantryMode anschließen
	AdminStatus	REFERENCE TO MB_AXIS_ADMIN_STATUS	Status Verwaltung der Achse, arAxisStatus_gb[i].Admin anschließen
	DataStatus	REFERENCE TO MB_AXIS_DATA	Istwerte und Status der Achse, arAxisStatus_gb[i].Data anschließen
	DiagStatus	REFERENCE TO MB_AXIS_DIAGNOSIS	Status Diagnose der Achse, arAxisStatus_gb[i].Diag anschließen

Es ist nicht möglich die komplette Instanz der Strukturen (z. B. arAxisCtrl_gb[ ]/arAxisStatus_gb[ ]) über einen Eingang dem Funktionsbaustein zu übergeben.

Dies wurde gemacht um der Anwender-Applikation spezielle Erweiterungen zum bestehenden Code zu ermöglichen. Nähere Details finden Sie unter ➚ , Seite .

Deshalb werden die benötigten Elemente von TE_AXIS_CONTROL_TYPE01 und TE_AXIS_STATUS_TYPE01 als separate Eingänge übergeben.

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

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

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

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

Fehlerbehandlung:

Die Fehlercodes der intern benutzten Funktionsbausteine (z.B. DL_ReadNode zum Lesen von Datalayer Knoten) und der intern benutzten Funktionen ((z.B. ML_AxsPower) werden durchgereicht. Der Funktionsbaustein kann die folgenden Fehlercodes erzeugen:

Tab. 12: Fehlercodes des Funktionsbausteins MB_AxisInterfacebase

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230110	Mindestens einer der Funktionsbaustein-Eingänge ist nicht initialisiert
DEVICE_ERROR	16#0A0F0107	16#0C230111	Achse ist im ErrorStop

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#0A0F0107	16#0C230113	Fehler im Ablauf des Funktionsbaustein
RESOURCE_ERROR	16#0A0F0107	16#0C230115	Ctrl.Admin.PowerOn ist auf FALSE eingestellt, eine Betriebsart wurde angewählt aber der Antrieb ist nicht in AH/AF
RESOURCE_ERROR	16#0A0F0107	16#0C230116	Mit der Methode Ctrl.Admin.mTriggerMoveCmd() wurde in ModeAb eine Betriebsart angewählt
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230117	OpMode wird von dem Funktionsbaustein in dieser Variante nicht unterstützt
STATE_MACHINE_ERROR	16#0A0F0107	16#0C230118	Fehler im Ablauf des Funktionsbaustein - Power
STATE_MACHINE_ERROR	16#0A0F0107	16#0C230119	Fehler im Ablauf des Funktionsbaustein - Reset
STATE_MACHINE_ERROR	16#0A0F0107	16#0C23011A	Fehler im Ablauf des Funktionsbaustein - Stop

1.3.3 Achs-Interface - Betriebsarten

1.3.3.1 Überblick

Es gibt drei Methoden um eine Betriebsart zu aktivieren:

- **Auswahl über ENUM-Werte**
  Zuweisen eines Wertes vom TYPE MB_AXIS_MODE an arAxisCtrl_gb[ ].Admin._OpMode:
  ```
  arAxisCtrl_gb[].Admin._OpMode:= ModePosAbs;
  ```
  - oder -
  ```
  arAxisCtrl_gb[].Admin._OpMode:= ModeAB;
  ```
- **Benutzung des Bit-Zugriffs**
  Setzen eines Bits über die Bit-Zugriffs Funktionalität.
  ```
  arAxisCtrl_gb[].Admin._OpMode-
  Bits.MODE_POS_ABS:= TRUE;
  ```
  Löschen des "_OpMode" durch Bit-Zugriffs Funktionalität ist auch möglich.

`arAxisCtrl_gb[].Admin._OpMode-Bits.MODE_POS_ABS := FALSE;` Durch das Bit-Löschen wird MODE_AB aktiviert.

Auch möglich: `arAxisCtrl_gb[].Admin._OpMode-Bits.MODE_AB := TRUE;`

- **Benutzung der Methode Ctrl.Admin.mTriggerMoveCmd()**

  Siehe auch "DemoBufferedAxisCommands" in der Vorlage "ctrlX CORE Axis/Kin-Interface"

  Diese Methode setzt sofort in dem Kontext dieses Aufrufes den Befehl an die Motion-Firmware ab. Damit dies funktioniert, muss die Achse bereits freigegeben sein, z.B. ModeAH und CmdDone abfragen.

  Aufruf: `arAxisCtrl_gb[uiAxisIndex].Admin.mTrig-gerMoveCmd(_OpMode:= ModePosAbs, Buf-fered:=TRUE, UserID:='my text');`

  - Übergabeparameter **_Opmode** = Wert vom TYPE MB_AXIS_MODE
  - Übergabeparameter **Buffered** = TRUE: Kommando wird erst aktiv, wenn das vorhergehende Kommando abgeschlossen ist
  - Übergabeparameter **Buffered** = FALSE: das vorhergehende Kommando wird abgebrochen
  - Übergabeparameter **UserID** = String (max. 25 byte). Wird als "Source" übergeben bei Aufruf von Motionkommandos. Bei Fehlern kann so die Quelle des Befehls identifiziert werden.
  - Returnwert der Methode: "cmdID" des abgesetzten Motion-kommandos. Bei Fehler wird 16#FFFFFFFFFFFFFFFF zurückgegeben.

Nur mit der Methode arAxisCtrl_gb[uiAxisIndex].Admin.mTrigger-MoveCmd() können auch gepufferte Befehle abgesetzt werden. Es ist damit auch möglich mehrere Befehle in einem Zyklus abzu-setzen (Buffered = TRUE).

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

Nur die Attribute (z. B. Position, Geschwindigkeit), die benutzt werden oder deren Standardwert geändert wurde, müssen dekla-riert werden, bevor der aktuelle Betriebsartenwechsel ausgeführt wird.

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

Beispiel:

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

  `arAxisStatus_gb[].Admin.MODE_POS_ABS`
- *Die Status-Quittung wird sofort beim Absetzen eines neuen Kommandos zurückgesetzt:*

  `arAxisStatus_gb[].Admin.CmdDone`

  Die Positionierungsbetriebsart wird aktiviert:

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

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

  `arAxisStatus_gb[].Admin.MODE_POS_ABS`

  Hat der Antrieb die Zielposition erreicht, wird die Statusquittung gesetzt:

  `arAxisStatus_gb[].Admin.CmdDone`

Der Eingang arAxisCtrl_gb.Admin.PowerOn steuert die interne Verwendung des **ML_AxsPower** (Bibliothek CXA_Motion) im Achs-Interface:

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

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

1.3.3.2 Antrieb Bereit

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

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

oder

```
arAxisCtrl_gb[].Admin._OpModeBits.MODE_AB:= TRUE;
```

Das AchsInterface benutzt intern die Funktionen ML_AxsPower und ML_AxsAbort (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.

Attribute Antrieb Bereit

Tab. 13: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl[ ]	StopMode.StopDeceleration	LREAL	99.0	Ja
	StopMode.StopJerk	LREAL	0.0	Nein
arAxis-Status_gb[]	Admin._OpModeAck-Bits.MODE_AB	BOOL		entfällt

1.3.3.3 Antrieb Halt

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

Folgendes Kommando aktiviert die Betriebsart:

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

oder

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

Das AchsInterface benutzt intern die Funktionen ML_AxsPower und ML_AxsAbort (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.

Attribute Antrieb Halt

Tab. 14: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl[ ]	StopMode.StopDeceleration	LREAL	99.0	Ja
	StopMode.StopJerk	LREAL	0.0	Nein
arAxis-Status_gb[]	Admin._OpModeAck-Bits.MODE_AH	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

1.3.3.4 Absolutes Positionieren

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

Folgendes Kommando aktiviert die Betriebsart:

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

oder

```
arAxisCtrl_gb[].Admin._OpModeBits.MODE_POS_ABS :=
TRUE;
```

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

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

Das Achs-Interface benutzt intern die Funktionen ML_AxsPower und ML_AxsPosAbs (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.

Attribute Absolutes Positionieren

Tab. 15: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	PosMode.Position	LREAL	0.0	Ja
	PosMode.Velocity	LREAL	10.0	Ja
	PosMode.DynValues.Acceleration	LREAL	10.0	Ja
	PosMode.DynValues.Deceleration	LREAL	10.0	Ja
	PosMode.DynValues.JerkAcc	LREAL	0.0	Nein
	PosMode.DynValues.JerkDec	LREAL	0.0	Nein
arAxisStatus_gb[ ]	Admin._OpModeAckBits.MODE_POS_ABS	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

1.3.3.5 Relatives Positionieren

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

Folgendes Kommando aktiviert die Betriebsart:

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

oder

```
arAxisCtrl_gb[].Admin._OpModeBits.MODE_POS_REL :=
TRUE;
```

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

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

Das Achs-Interface benutzt intern die Funktionen ML_AxsPower und ML_AxsPosRel (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.

Attribute Relatives Positionieren

Tab. 16: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	PosMode.Distance	LREAL	0.0	Ja
	PosMode.Velocity	LREAL	10.0	Ja
	PosMode.DynValues.Acceleration	LREAL	10.0	Ja
	PosMode.DynValues.Deceleration	LREAL	10.0	Ja
	PosMode.DynValues.JerkAcc	LREAL	0.0	Nein
	PosMode.DynValues.JerkDec	LREAL	0.0	Nein
arAxis-Status_gb[ ]	Admin._OpModeAck-Bits.MODE_POS_REL	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

1.3.3.6 Additives Positionieren

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

Folgendes Kommando aktiviert die Betriebsart:

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

oder

```
arAxisCtrl_gb[].Admin._OpModeBits.MODE_POS_ADD:=
TRUE;
```

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

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

Das Achs-Interface benutzt intern die Funktionen ML_AxsPower und ML_AxsPosAdd (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.

Attribute Additives Positionieren

Tab. 17: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	PosMode.Distance	LREAL	0.0	Ja
	PosMode.Velocity	LREAL	10.0	Ja
	PosMode.DynValues.Acceleration	LREAL	10.0	Ja
	PosMode.DynValues.Deceleration	LREAL	10.0	Ja
	PosMode.DynValues.JerkAcc	LREAL	0.0	Nein
	PosMode.DynValues.JerkDec	LREAL	0.0	Nein
arAxis-Status_gb[ ]	Admin._OpModeAck-Bits.MODE_POS_ADD	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

1.3.3.7 Betriebsart "Robot-Control"

Die Aktivierung dieser Betriebsart schaltet den entsprechenden Antrieb in zyklische Lageregelung (antriebsgeführt). Des Weiteren wird die entsprechende Achse automatisch zu einer Kinematik gruppiert. Die so gruppierten Achsen können dann eine koordinierte Bewegung ausführen.

Folgendes Kommando aktiviert die Betriebsart:

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

oder

```
arAxisCtrl_gb[].Admin._OpModeBits.MODE_COORDI-
NATED:= TRUE;
```

Die Rückmeldung dieser Betriebsart `TRUE` erfolgt in der Achsstatusstruktur:

```
arAxisStatus_gb[].Admin._OpModeAckBits.MODE_COOR-
DINATED;
```

Das Achs-Interface benutzt intern die Funktionen ML_AxsPower, ML_AxsAddToKin und ML_AxsRemoveFromKin (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.

Attribute Betriebsart Robot-Control

Tab. 18: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	CoordMode.KinName	STRING(15)		Nein

Deklaration:

z. B: Kin1: MB_AXESGROUPIF_REF :=(KinName:='Kin1',GroupNo:=0);

Zuweisung:

z. B: arAxisCtrl_gb[uiAxisIndex].CoordMode.KinName := Kin1.KinName;

1.3.3.8 Betriebsart "Gantry"

Die Aktivierung dieser Betriebsart schaltet den entsprechenden Antrieb in zyklische Lageregelung (antriebsgeführt). Des Weiteren wird die entsprechende Achse automatisch einem Gantry-Master angekoppelt. Die Achse folgt dem Gantry-Master im PLCopen Zustand "SYNCHRONIZED_MOTION".

Folgendes Kommando aktiviert die Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode := ModeGantry
```

oder

```
arAxisCtrl_gb[].Admin._OpModeBits.MODE_GANTRY:=
TRUE;
```

Die Rückmeldung dieser Betriebsart `TRUE` erfolgt in der Achsstatusstruktur:

```
arAxisStatus_gb[].Admin._OpModeAck-
Bits.MODE_GANTRY;
```

Das Achs-Interface benutzt intern die Funktionen ML_AxsPower, ML_AxsAddToGantry und ML_AxsRemoveFromGantry (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.

Attribute Betriebsart Gantry

Tab. 19: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxisCtrl_gb[ ]	GantryMode.Master	AXIS_REF		Nein

Deklaration:

z. B: Master1: AXIS_REF :=(AxisName:='Axis1');

Zuweisung:

z. B: arAxisCtrl_gb[uiAxisIndex].GantryMode.Master := Master1;

1.3.3.9 Betriebsart "Externer Funktionsbaustein"

Wenn sich die Achse bereits in einer Betriebsart befindet, dann wird bei der Aktivierung dieser Betriebsart kein Bewegungsbefehl vom Achs-Interface ausgeführt.

Wird die Betriebsart von ModeAb auf ModeExternalFB geändert, dann schaltet das Achs-Interface den Antrieb in AH und wartet auf einen externen Bewegungsbefehl, z. B. von einer Technologiefunktion.

Folgendes Kommando aktiviert die Betriebsart:

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

oder

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

Das Achs-Interface benutzt intern die Funktion ML_AxsPower (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.

Attribute Betriebsart Externer Funktionsbaustein

Tab. 20: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxis-Status_gb[ ]	Admin._OpModeAck-Bits.MODE_EXTERNAL_FB	BOOL		entfällt

1.3.3.10 Benutzerdefinierte Betriebsarten

Bei benutzerdefinierbaren Betriebsarten werden die Motion-Kommandos in der anwenderspezifischen Erweiterung in der Bibiliothek **CXA_MotionInteraceUser** festgelegt. Damit können Anwender eigene Betriebsarten und deren Motion-Kommandos frei implementieren. Es stehen insgesamt 10 benutzerdefinierte Betriebsarten (ModeUserXx_User_0...ModeUserXx_User_9) zur Verfügung. Für ModeUserXx_User_0 ist bereits eine Beispielimplementation vorhanden.

Wenn sich die Achse bereits in einer Betriebsart befindet, dann wird bei der Aktivierung dieser Betriebsart kein Bewegungsbefehl vom Achs-Interface ausgeführt.

Wird die Betriebsart von ModeAb auf Mode-UserXx_User_0...ModeUserXx_User_9 geändert, dann schaltet das Achs-Interface den Antrieb in AH und wartet auf einen Bewegungsbefehl aus der Erweiterung im Applikationsteil.

Folgendes Kommando aktiviert die erste benutzerdefinierte Betriebsart:

```
arAxisCtrl_gb[].Admin._OpMode := Mode-
UserXx_User_0;
```

oder

```
arAxisCtrl_gb[].Admin._OpMode-
Bits.MODE_XX_USER_0:= TRUE;
```

Das Achs-Interface benutzt intern die Funktion ML_AxsPower (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.

Attribute der benutzerdefinierten Betriebsarten

Tab. 21: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxis-Status_gb[ ]	Admin._OpModeAck-Bits.MODE_XX_USER_0...9	BOOL		entfällt

1.3.4 Achs-Interface - Globale Variablen

Die Bibliothek CXA_MotionInterfaceUser enthält im Ordner "AxisInterfaceUser/GlobalVariables" die globale Variablenliste "Global_AxisInterface".

Diese Liste enthält die folgenden Strukturen/Variablen:

Name	Type	Beschreibung
arAxisCtrl_gb[ ]	ARRAY [] OF TE_AXIS_CONTROL_TYPE01	Kontrollstruktur des AchsInterface
arAxisStatus_gb[ ]	ARRAY [] OF TE_AXIS_STATUS_TYPE01	Statusstruktur des AchsInterface
arAxisIdx_gb	ARRAY [] OF UINT	Nicht lückende Liste der Achsen
uiNofAxis_gb	UINT	Anzahl der aktiven Achsen
VisuAxisNo	INT	Umschaltung des Achsindex in den Visualisierungen
bRemoteOn_gb	BOOL	TRUE: Visualisierung ist aktiv

1.3.5 Achs-Interface - Strukturen

1.3.5.1 Überblick

Das Achs-Interface stellt eine Datenschnittstelle zur komfortablen Ansteuerung der Achsen zur Verfügung.

Informationen zu den Datenstrukturen siehe Online-Dokumentation in den Bibliotheken **CXA_MotionInterface** im Ordner "AxisInterface/DUTs" bzw. in **CXA_MotionInterfaceUser** im Ordner "AxisInterfaceUser/DUTs".

*Das Programm-Template "ctrlX CORE Axis/Kin-Interface" ist so vorbereitet, dass es durch den Anwender erweiterbar ist bzw. es sind einige Erweiterungen schon mit eingebaut. Um diese Erweiterungen zu ermöglichen, ist es nötig, eigene Strukturen im Anwenderprojekt zu definieren, die die Strukturen der Bibliothek erweitern. Die erweiterteten Strukturen sind durch den Präfix "TE_" gekennzeichnet. Wenn also imProgramm-Template "ctrlX CORE Axis/Kin-Interface" eine Struktur, z.B. **TE_**AXIS_ADMIN_STATUS heisst, ist diese eine erweiterte Struktur der **MB_**AXIS_ADMIN_STATUS.*

*Informationen zu der Struktur sind in der Online-Dokumentation unter dem Namen **MB_**AXIS_ADMIN_STATUS in der Bibliothek **CXA_MotionInterface** und unter dem Namen TE_AXIS_ADMIN_STATUS in der Bibliothek **CXA_MotionInterfaceUser** zu finden.*

1.3.6 Achs-Interface - Beispielprogramm

1.3.6.1 Überblick

In diesem Kapitel soll ein Überblick über die als offener Code verfügbaren Teile des Achs-Interface gegeben werden.

Die offenen Programmteile werden mit den folgenden Elementen geliefert:

- Das Programmiertemplate "ctrlX CORE Axis/Kin-Interface" dient als Beispielapplikation für das Achs-Interface. Siehe auch ↗ *Kapitel 1.2 „MotionInterface - Erstkonfiguration “ auf Seite 2*
- Die Bibliothek **CXA_MotionInterfaceUser.library** kann durch den Anwender verändert werden um das Achs-Interface an die jeweilige Applikation anzupassen. Siehe auch ↗ *Kapitel 1.3.7 „Achs-Interface Anwender-Erweiterung “ auf Seite 36*.

1.3.6.2 Programmiertemplate "ctrlX CORE Axis/Kin-Interface"

Das Programmiertemplate "ctrlX CORE Axis/Kin-Interface" deckt die folgenden Punkte zum Achs-Interface ab:

- **PlcProg**:

  Aufruf des TE_AxisInterfaceMainProg() - Initialisierung und Zyklischer Aufruf des Achs-Interface.

Beispielcode zur Verwendung des AchsInterface. Dieser Code muss für das eigene Projekt entsprechend angepasst werden.

- **MotionProg**:

  Hier wird die Methode TE_AxisInterfaceMainProg.mMotion-Task() aufgerufen. In diesem Takt werden die Istwerte in arAxis-Status_gb[].Data aufgefrischt. Falls die Methode nicht aufgerufen wird, werden die Istwerte im Takt des PlcProg aktualisiert.
- **GlobalAxisDefines**:

  Wird nur benötigt, wenn MOTIF_CONFIG.CONFIG_MODE auf GLOB_VAR eingestellt ist.

  Hier werden die Achsen als Konstanten vom Typ MB_AXISIF_REF definiert und in einer Liste an TE_AxisInterfaceMainProg() übergeben. Die Konstanten müssen für das eigene Projekt entsprechend angepasst werden.
- **DemoBufferedAxisCommands**:

  Beispielcode mit einer Ablaufprogrammierung und dem Absetzen gepufferter Achskommandos. Dieser Code muss für das eigene Projekt entsprechend angepasst werden.
- **OverViewAxes**:

  Visualisierung zur Bedienung des Achs-Interface während der Inbetriebnahmephase. Durch Klicken auf Felder mit "<<" kann in weitere Bilder abgetaucht werden.
- **Version_AxisKinInterface**:

  Änderungshistorie und Disclaimer

1.3.6.3 Bibliothek CXA_MotionInterfaceUser.library"

Diese offene Bibliothek dient dazu die Funktionsbausteine und Strukturen der Basisbibliothek **CXA_MotionInterface** zu erweitern. Programme und Visualisierungen werden hier zur Verfügung gestellt. Hier sind auch die globalen Variablen der Interfaces instanziiert. Mit dieser Bibliothek sind Anpassungen / Erweiterungen der Interfaces durch den Anwender möglich.

Wie man die Anpassungen ausführen kann, ist hier ↗ *Kapitel 1.3.7 „Achs-Interface Anwender-Erweiterung “ auf Seite 36* beschrieben

In diesem Kapitel werden die POUs des AchsInterface aus dem Ordner "AxisInterfaceUser/POUs" beschrieben.

TE_AxisInterfaceMainProg

Das Programm **TE_AxisInterfaceMainProg** deckt die folgenden Punkte ab:

- **Initialisierung des Achs-Interface**:

  Bei Erreichen des Modus "Running" wird das Achs-Interface mit Hilfe des Funktionsbausteins **TE_AxisInitAllAxis** initialisiert. Bei erfolgreicher Initialisierung wird der Ausgang "InitDone" gesetzt, bei Fehlern der Ausgang "Error".
- **Zyklischer Aufruf des Achs-Interface**:

  Nach erfolgreicher Initialisierung wird der Funktionsbaustein (FB) **TE_AxisInterface** zyklisch aufgerufen.
- **Methode TE_AxisInterfaceMainProg.mMotionTask()**:

Wird die Methode aus einer schnelleren MotionTask aufgerufen, werden die Elemente in "arAxisStatus_gb[].Data" im schnelleren Takt aktualisiert. Der Aufruf des FB **TE_AxisInterface** kann mit Hilfe der Steuer-Variable "arAxisCtrl_gb[].Admin.Config.MotionSync" in die schnellere Task verschoben werden.

Wird die Methode nicht aufgerufen, erfolgen alle Aktualisierungen im Takt der PLC-Task.

Tab. 22: Schnittstellenvariablen TE_AxisInterfaceMainProg

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	ClearError	BOOL	Fehler löschen wird durch eine positive Flanke an "ClearError" gestartet
	AxisCfgIdx	POINTER TO ARRAY [] OF MB_AXISIF_REF	Konfigurationsliste für die Indizes der Achsen (nur für Konfigurationsmodus "GLOB_VAR")
VAR_OUTPUT	InitDone	BOOL	Wird gesetzt, wenn das Programm die Initialisierung erfolgreich beendet hat
	Error	BOOL	Zeigt an, dass ein Fehler im Programm aufgetreten ist
	ErrorID	ERROR_CODE	Kurzer Hinweis zur Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Information zum Fehler

Fehlerbehandlung: die Fehlercodes des intern benutzten Funktionsbausteines TE_AxisInitAllAxes werden durchgereicht. Das Programm kann die folgenden Fehlercodes erzeugen:

Tab. 23: Fehlercodes des Programmes TE_AxisInterfaceMainProg

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#0A0F0107	16#0C230190	Fehler im Ablauf des Programmes

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

*Siehe auch **Example_AxIfApplicationPart** im Ordner "AxisInterfaceUser/_Examples"*

TE_AxisInitAllAxes

Der Funktionsbaustein **TE_AxisInitAllAxes** initialisiert die Achs-Interface Strukturen.

Die Initialisierung kann gesteuert werden mit Hilfe der Parameterliste "MOTIF_CONFIG".

Es gibt die folgenden Möglichkeiten:

- AUTO = MOTIF_CONFIG.CONFIG_MODE: Es wird der Datalayer Knoten "motion/axs/" ausgelesen und die Achsen in der dort gefundenen Reihenfolge in die Achs-Interface Strukturen eingeordnet.
- GLOB_VAR = MOTIF_CONFIG.CONFIG_MODE: Die Achsen werden anhand des globalen Arrays "AXIF_CONFIG_INDEXES" in die Achs-Interface Strukturen eingeordnet.

Tab. 24: Schnittstellenvariablen TE_AxisInitAllAxes

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	Execute	BOOL	Die Initialisierung wird durch eine positive Flanke an "Execute" gestartet
	AxisCfgIdx	POINTER TO ARRAY [] OF MB_AXISIF_REF	Konfigurationsliste für die Indizes der Achsen (nur für Konfigurationsmodus "GLOB_VAR")
VAR_OUTPUT	Done	BOOL	Wird gesetzt, wenn der FB die Bearbeitung beendet hat
	Active	BOOL	Wird gesetzt, wenn der FB aktiv ist (nicht im Leerlaufbetrieb)
	Error	BOOL	Zeigt an, dass ein Fehler im Programm aufgetreten ist
	ErrorID	ERROR_CODE	Kurzer Hinweis zur Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Information zum Fehler

Fehlerbehandlung: die Fehlercodes des intern benutzten Funktionsbausteines MB_AxisInit werden durchgereicht. Der Funktionsbaustein kann die folgenden Fehlercodes erzeugen:

Tab. 25: Fehlercodes des Funktionsbausteines TE_AxisInitAllAxes

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230180	Eingang AxisIndex ausserhalb des gültigen Bereiches [MOTIF_CONFIG.MIN_AXIS_INDEX..MOTIF_CONFIG.MAX_AXIS_INDEX]
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230181	Unbekannter Konfigurationsmodus (MOTIF_CONFIG.CFG_MODE_AXS)

Achs-Interface

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230182	Pointer AxisCfgIdx zur globalen Variablen ist 0 (MOTIF_CONFIG.CFG_MODE_AXS = GLOB_VAR)
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230183	Achsindex ausserhalb des gültigen Bereiches (MOTIF_CONFIG.CFG_MODE_AXS = GLOB_VAR)
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230184	Achsindex doppelt (MOTIF_CONFIG.CFG_MODE_AXS = GLOB_VAR)
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230185	Achsname doppelt (MOTIF_CONFIG.CFG_MODE_AXS = GLOB_VAR)
STATE_MACHINE_ERROR	16#0A0F0107	16#0C230188	Fehler im Ablauf des Funktionsbausteines

*Das Programm TE_AxisInterfaceMainProg ruft diesen Funktionsbaustein **TE_AxisInitAllAxes** bereits auf.*

TE_AxisInterface

Der Funktionsbaustein **TE_AxisInterface** erweitert den **MB_AxisInterfaceBase** und bearbeitet im zyklischen Betrieb die Achs-Interface Strukturen.

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

Tab. 26: Schnittstellenvariablen TE_AxisInterface

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT Steuer-Struktur	AdminCtrlExt	REFERENCE TO TE_AXIS_ADMINISTRATION	Referenz zur Steuer-Struktur Admin
	StopModeCtrlExt	REFERENCE TO TE_AXIS_STOP_MODE	Referenz zur Steuer-Struktur Stop Mode
	PosModeCtrlExt	REFERENCE TO TE_AXIS_POSITIONING	Referenz zur Steuer-Struktur Positioning

I/O-Typ	Name	Datentyp	Kommentar
	SetupMode	REFERENCE TO TE_AXIS_SETUP_MODE	Referenz zur Steuer-Struktur SetupMode
VAR_INPUT Status-Struktur	AdminStatusExt	REFERENCE TO TE_AXIS_ADMIN_STATUS	Referenz zur Status-Struktur Admin
	DataStatusExt	REFERENCE TO TE_AXIS_DATA	Referenz zur Status-Struktur Data
	DiagStatusExt	REFERENCE TO TE_AXIS_DIAG-NOSIS	Referenz zur Status-Struktur Diag
	SetupModeAck	REFERENCE TO TE_AXIS_SETUP_MODE_STATUS	Referenz zur Status-Struktur SetupMode

Fehlerbehandlung: die Fehlercodes der intern benutzten Funktionsbausteine werden durchgereicht. Der Funktionsbaustein kann die folgenden Fehlercodes erzeugen:

Tab. 27: Fehlercodes des Funktionsbausteines TE_AxisInterface

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#0A0F0107	16#0C2301A0	Fehler im Ablauf des Funktionsbausteines

*Das Programm TE_AxisInterfaceMainProg ruft diesen Funktionsbaustein **TE_AxisInterface** bereits auf.*

TE_GetAxisInterfaceIndex

Die Funktion **TE_GetAxisInterfaceIndex** liefert den Index einer Achse in den Achs-Interface-Strukturen definiert durch den Achs-Namen.

Tab. 28: Schnittstellenvariablen TE_GetAxisInterfaceIndex

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	AxisName	STRING(15)	Name der gesuchten Achse
Return	TE_GetAxisInterfaceIndex	UINT	Index der gesuchten Achse "16#FFFF" (-1) wenn die Achse nicht gefunden wurde

Fehlerbehandlung: Die Funktion gibt "16#FFFF" (-1) zurück, wenn die Achse nicht gefunden wurde.

1.3.6.4 Achs-Interface Visualisierungen

1.3.6.4.1 Überblick

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

Folgende Visualisierungen sind Programmiertemplate "ctrlX CORE Axis/Kin-Interface" und in der Bibliothek **CXA_MotionInterfaceUser.library** enthalten:

Beispielprojekt Visualisierungen

Tab. 29: Beispielprojekt Visualisierungen

Visualisierung	Beschreibung
OverviewAxes (im Template-Project)	Gesamtüberblick über alle definierten Achsen, einschließlich einfacher Diagnosen, Status und Tipp-Bedienelemente
Axis_Overview	Zeigt aktuelles Kommando und aktuelle Werte für Position und Geschwindigkeit, zusammen mit Navigation zur Positions-Anzeige für die aktuelle Achse. Referenzieren und Abschalten der Achsen sind ebenfalls möglich
Position_mode	Überwachen des Positionierbetriebs

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

- arAxisCtrl_gb[ ]
- arAxisStatus_gb[ ]
- VisuAxisNo

1.3.6.4.2 Systemübersicht-Visualisierung

Die OverviewAxes Visualisierung erlaubt es dem Anwender jede, im Projekt konfigurierte Achse, schnell anzukoppeln. Zusätzliche Achsen können im Offline-Betrieb zur Anzeige hinzugefügt werden.

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

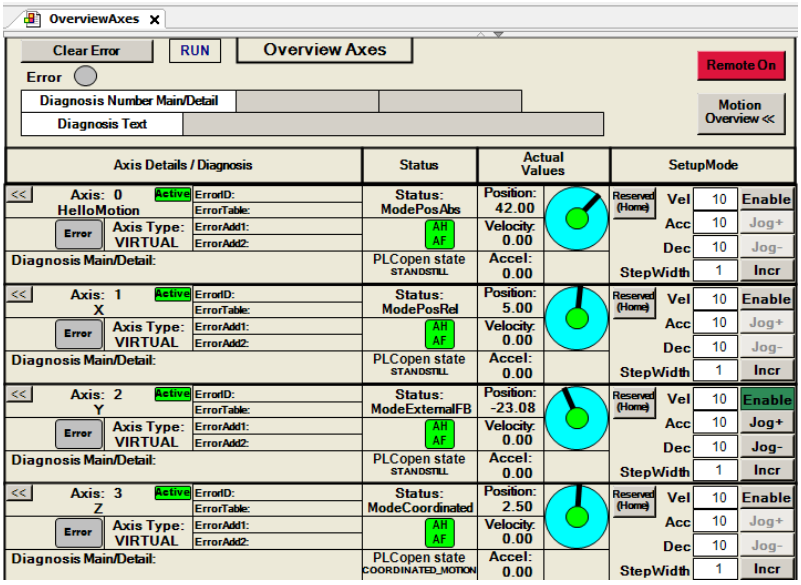


Abb. 6: OverviewAxes

Hinzufügen einer Achse zur Systemübersicht

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

1. Mit ctrlX PLC Engineering im Offline-Betrieb Doppelklick auf die "OverviewAllAxes" Visualisierung.
2. Wählen Sie im Fenster Visualisierungswerkzeuge die Schaltfläche *„Frame“* an und erzeugen Sie einen Grundriss unterhalb der letzten Achstabellenzeile, der der Zeilenhöhe und Zeilenbreite entspricht.
3. Mit rechter Maustaste auf den Frame klicken und "Frameauswahl" aktivieren. Wählen Sie das **OverviewOneAxis**-Element aus dem Visualisierungsauswahlfenster im Ordner **CXA_MotionInterfaceUser/AxisInterfaceUser/Visualizations/SystemOverview**aus.

   ➡ Eine komplett neue Systemübersicht Achszeile erscheint.
4. Nach einem Klick in der neuen Visualisierung tragen Sie im "Eigenschaften"-Fenster als Wert für**m_Input_AxisIndex** den Index der definierten Achse ein.
5. Das neue Achs-Interface für die Systemübersicht kann nun in der Grösse angepasst und positioniert werden.
6. Übersetzen Sie das SPS-Projekt neu und gehen Sie Online.

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

Systemübersicht Navigation

Eine Einzelachs-Übersichtanzeige ist durch Klicken auf die Schaltfläche mit zwei Pfeilen "<<", die sich unter der Details-Spalte in der Achstabelle befindet, erreichbar. Aus der Achsübersichtanzeige ist die Navigation zur Positions-Betriebsartenanzeige möglich.

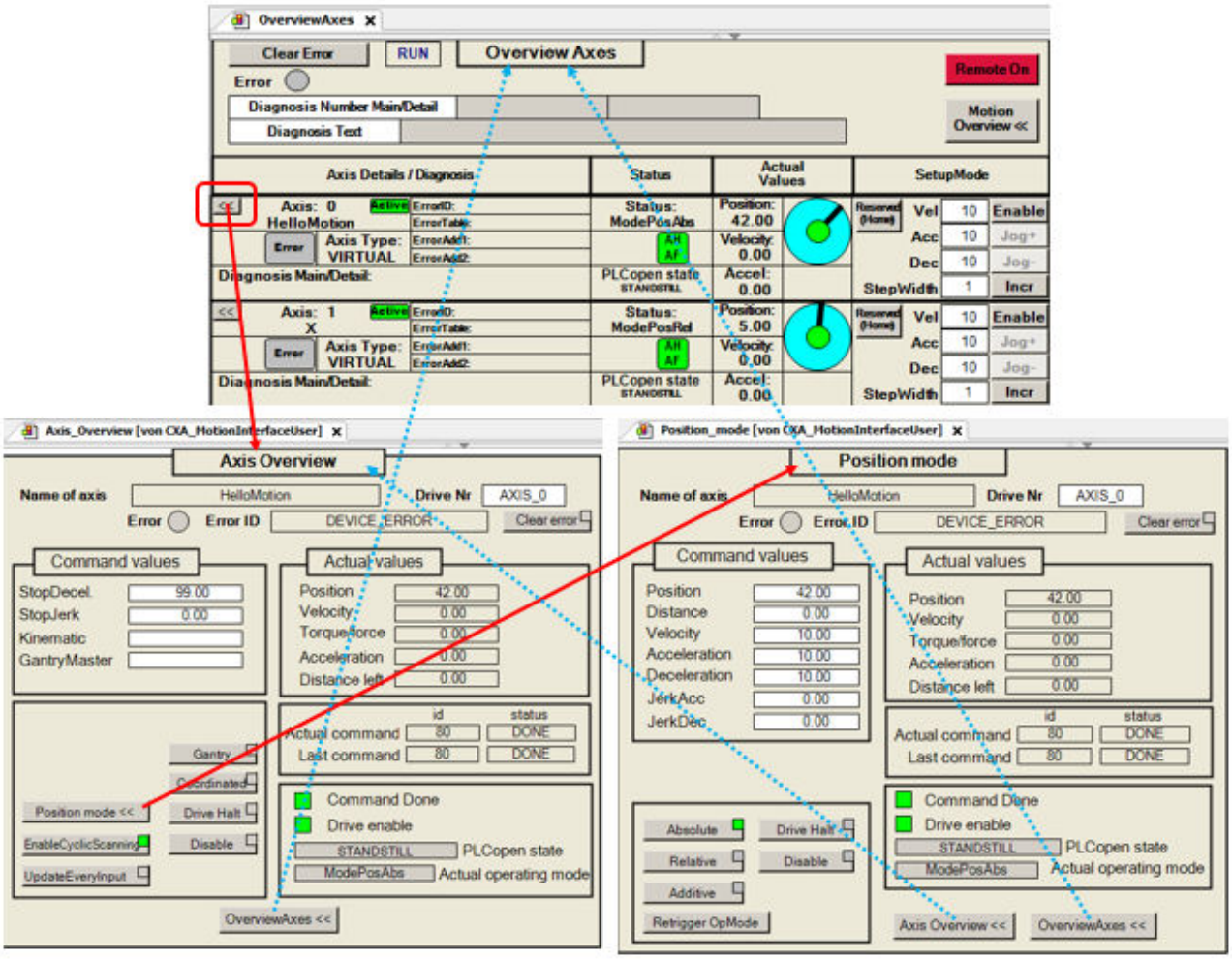


Abb. 7: Systemübersicht Navigation

1.3.7 Achs-Interface Anwender-Erweiterung

1.3.7.1 Überblick

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

*Die **arAxisCtrl_gb[ ]**- und **arAxisStatus_gb[ ]**-Strukturen des Anwender-Interface sind als Basistypen mit dem Präfix "MB_" in der geschlossenen Bibliothek **CXA_MotionInterface.compiled-library** definiert und daher für den Anwender nicht zugänglich.*

*Um Erweiterungen zu ermöglichen, ist es nötig, eigene Strukturen zu definieren, die die Strukturen "MB_" erweitern. Die erweiterteten Strukturen sind durch den Präfix "TE_" gekennzeichnet und befinden sich in der offenen Bibliothek **CXA_MotionInterfaceUser.library**.*

Empfohlene Vorgehensweise

Um die Anwendererweiterungen auszuführen, ist es notwendig die Bibliothek **CXA_MotionInterfaceUser.library** anzupassen. Zur Nachvollziehbarkeit ist es notwendig und dringend empfohlen der angepassten Bibliothek einen neuen Namen zu geben, z.B. CXA_MotionInterface**MyCompany**.library. Im Folgenden wird als Bibliotheksname **CXA_MotionInterfaceMyCompany.library** verwendet.

Arbeitsablauf

1. Im Bibliotheksverwalter die **CXA_MotionInterfaceUser.library** selektieren. Rechte Maustaste -> "Bibliothek exportieren" anwählen. Speicherort wählen und einen neuen Namen z.B. CXA_MotionInterface**MyCompany**.library vergeben. Nachträgliches Umbenennen ist ebenfalls möglich.
2. Mit einer zweiten Instanz von ctrlX PLC Engineering die Bibliothek **CXA_MotionInterfaceMyCompany.library** öffnen. In den Projektinformationen das Feld "freigegeben" abwählen, die weiteren Felder anpassen und in den Eigenschaften den Schlüssel "Placeholder" löschen.
3. Im Anwendungsprogramm (erste Instanz von ctrlX PLC Engineering) im Bibliotheksverwalter die **CXA_MotionInterfaceUser.library** entfernen und dafür **CXA_MotionInterfaceMyCompany.library** einbinden.
4. Anpassungen in der Bibliothek vornehmen. Am Ende ausführen: "Datei"->"Projekt speichern und ins Bibiliotheksrepository installieren".
5. Im Anwendungsprogramm (erste Instanz von ctrlX PLC Engineering) die Anpassungen testen. Debuggen im Code aus der Bibliothek ist auch möglich.
6. Schritte 4. und 5. wiederholen bis die Funktion fehlerfrei ist.

*Sobald ein Update der **CXA_MotionInterfaceUser.library** zur Verfügung steht, können Änderungen mit "Projekt"->"Vergleichen" in die **CXA_MotionInterfaceMyCompany.library** übernommen werden.*

*Die **CXA_MotionInterfaceUser.library** verwendet Fehlercodes mit "CXA_TABLE". Diese sind in der Produktdokumentation zu finden. Wenn in den Anwendererweiterungen weitere Fehlercodes benötigt werden, können diese frei definiert werden, müssen aber mit "USER1_TABLE..USER10_TABLE" gemeldet werden.*

ML_AxsGetIpoValues

Dieser Abschnitt zeigt, wie das Achs-Interface durch Hinzufügen der **ML_AxsGetIpoValues**-Funktionalität erweitert wird. Die Funktion ML_AxsGetIpoValues ermöglicht es, die interpolierten Werte einer Achse abzufragen. Die Istwerte sind bereits in der Basis-Struktur als z.B. "ActualPosition" vorhanden.

Die folgenden neuen Ein- und Ausgänge werden definiert:

- arAxisCtrl_gb[ ].Admin.EnableReadIpo
- arAxisStatus_gb[ ].Data.IpoPosition
- arAxisStatus_gb[ ].Data.IpoVelocity

Im Programm-Template "ctrlX CORE Axis/Kin-Interface" sind folgende anwenderspezifische Erweiterungen implementiert:

- Jog-Funktionalität als "SetupMode"
- "RetriggerOpMode" um eine Betriebsart erneut anzusteuern z.B. MoveRelative mit der gleichen Distanz
- Eine benutzerdefinierte Betriebsart "MODE_XX_USER_0"

Im Folgenden wird davon ausgegangen, dass die im Programm-Template "ctrlX CORE Axis/Kin-Interface" bereits vorbereitete Struktur verwendet wird. Es wird nur beschrieben, welche Änderungen in den dort vorgegebenen POUs notwendig sind.

Hinweise zur Implementation von Anwendererweiterungen

Die Anwendererweiterungen wurden mit Hilfe der objektorientierten Erweiterungen von ctrlX PLC Engineering implementiert. Dabei sind einige Besonderheiten zu beachten:

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

  Die Eingänge AdminCtrl (ohne Ext) usw. gehören zum Basis-FB und sollten nicht genutzt werden
- Die Eingänge des FB TE_AxisInterface sind als "REFERENCE TO" definiert. In der Methode mInitExtension werden die Referenzen einmalig initialisiert und müssen dann beim zyklischen Aufruf des FB nicht mehr übergeben werden

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

1.3.7.2 Erweitern der arAxisCtrl_gb[ ]-Struktur

Um die zusätzliche Funktionalität der arAxisCtrl_gb[] Struktur hinzuzufügen, muss der Anwender eine neue Struktur anlegen, die Unterstrukturen von den bereits existierenden Struktur ableiten und dann die neuen Elemente hinzufügen. Im Programm-Template "ctrlX CORE Axis/Kin-Interface" ist eine Struktur TE_AXIS_CONTROL_TYPE01 und die Unterstrukturen TE_AXIS_ADMINISTRATION usw. bereits vorbereitet.

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

1. Mit ctrlX PLC Engineering die Bibliothek **CXA_MotionInterfaceMyCompany.library** öffnen
2. Öffnen Sie die Struktur TE_AXIS_ADMINISTRATION, Ordner AxisInterfaceUser/DUTs/Control.
3. Deklarieren Sie die folgende Variable
   - `EnableReadIpo: BOOL:=TRUE;`

1.3.7.3 Erweitern der arAxisStatus_gb[ ] Struktur

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

1. Öffnen Sie die Struktur TE_AXIS_DATA, Ordner AxisInterfaceUser/DUTs/Status.
2. Deklarieren Sie die folgenden Variablen
   - `IpoPosition: LREAL;`
   - `IpoVelocity: LREAL;`

1.3.7.4 Erweitern des Funktionsbausteines

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

1. Deklarieren Sie im FB TE_AxisInterface die folgende Variable:

```
stAxsGetIpoValuesData: ML_AxsGetIpoValues-
Data;
```

2. Im FB TE_AxisInterface könnte die Funktion so ausprogrammiert werden:

```
IF AdminCtrlExt.EnableReadIpo = TRUE THEN

stAxsGetIpoValuesData.In.AxisName :=
AdminCtrlExt.Config.Axis.AxisName;

ML_AxsGetIpoValues(stAxsGetIpoValuesData); //
call motion function

 DataStatusExt.IpoPosition := stAxsGetIpoVa-
luesData.Out.Position;

DataStatusExt.IpoVelocity := stAxsGetIpoValu-
esData.Out.Velocity;

END_IF
```

3. Übersetzen Sie das Projekt neu und überprüfen Sie es auf Programmierfehler.
4. Laden Sie das Projekt in die Steuerung.

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

Watch 1

Expression	Type	Value
Global_AxisInterface.arAxisCtrl_gb[0]	TE_AXIS_CONTROL_TYPE01	
Admin	TE_AXIS_ADMINISTRATION	
Config	MB_AXIS_ADMIN_CONFIG	
ClearError	BOOL	FALSE
_OpMode	MB_AXIS_MODE	ModePosRel
_OpModeBits	TE_AXIS_MODE_BITS	
RetriggerOpMode	BOOL	FALSE
EnableReadIpo	BOOL	TRUE
StopMode	TE_AXIS_STOP_MODE	
PosMode	TE_AXIS_POSITIONING	
CoordMode	MB_AXIS_COORDINATED	
GantryMode	MB_AXIS_GANTRY	
SetupMode	TE_AXIS_SETUP_MODE	
Global_AxisInterface.arAxisStatus_gb[0]	TE_AXIS_STATUS_TYPE01	
Admin	TE_AXIS_ADMIN_STATUS	
Data	TE_AXIS_DATA	
Aborting	BOOL	FALSE
ActualAcceleration	LREAL	0
ActualPosition	LREAL	307.52000000000004
ActualTorque	LREAL	0
ActualVelocity	LREAL	10
CoordinatedMotion	BOOL	FALSE
Disabled	BOOL	FALSE
DiscreteMotion	BOOL	TRUE
DistLeft	LREAL	0
ErrorStop	BOOL	FALSE
PLCopenState	STRING(50)	'DISCRETE_MOTION'
StandStill	BOOL	FALSE
StandStillPending	BOOL	FALSE
IpoPosition	LREAL	307.40000000000003
IpoVelocity	LREAL	10
Diag	TE_AXIS_DIAGNOSIS	
SetupMode	TE_AXIS_SETUP_MODE_STATUS	

Abb. 8: arAxisCtrl_gb und arAxisStatus_gb Strukturen mit Anwender-Erweiterungen

1.3.7.5 Anwendung der benutzerdefinierten Betriebsarten

Bei benutzerdefinierbaren Betriebsarten werden die Motion-Kommandos in der anwenderspezifischen Erweiterung in der Bibliothek **CXA_MotionInterfaceMyCompany.library** implementiert.

Die folgenden Schritte zeigen an einem Beispiel wie Sie die benutzerdefinierten Betriebsarten nutzen können.

1. Fügen Sie eine neue Datenstruktur für die Sollwerte der benutzerdefinierten Betriebsart (Name z. B. TE_AXIS_USERMODE1) unterhalb des Ordners AxisInterface/Type/Control hinzu.
2. Definieren Sie die Elemente dieser Datenstruktur (z. B. Vel: REAL, Acc:REAL, DEC: REAL).
3. Fügen Sie am Ende der Datenstruktur TE_AXIS_CONTROL_TYPE01 oder TE_AXIS_CONTROL_TYPE02 ein neues Element vom Typ der o.g. Datenstruktur ein (Beispiel: UserMode1: TE_AXIS_USERMODE1;).
4. Fügen Sie am Ende der VAR_INPUT Deklaration im FB TE_AxisInterface einen neuen Eingang ein (Beispiel: MyModeUser1Ext : REFERENCE TO TE_AXIS_USERMODE1;).
5. Initialisieren Sie die Referenz des neu angelegten Einganges in der Methode mInitExtension des TE_AxisInterface (Beispiel: MyModeUser1Ext REF= arAxisCtrl_gb[index].UserMode1;).
6. Programmieren Sie das gewünschte Motion-Kommando in der Methode **mUserModes**. Die Methode **mUserModes** des TE_AxisInterface überschreibt die Methode des Basisbausteines MB_AxisInterfaceBase und wird von diesem aufgerufen.
7. Übersetzen Sie das Projekt neu und überprüfen Sie es auf Programmierfehler.
8. Laden Sie das Projekt in die Steuerung und starten Sie die SPS.
9. Jetzt können Sie die neue benutzdefinierte Betriebsart anwählen und das Motion-Kommando wird mit den übergebenen Sollwerten wirksam.

1.3.8 HowTo: Typische Anwenderaktivitäten

1.3.8.1 Zugriff auf Achsdaten

Die folgenden Daten sind verfügbar (Name = jeweiliger Achsname):

- arAxisCtrl_gb[Name.AxisNo] => Steuerstruktur des Achs-Interface
- arAxisStatus_gb[Name.AxisNo] => Statusstruktur des Achs-Interface
- arAxisStatus_gb[Name.AxisNo].Data => Istwerte und Statusinformationen
-

Azyklische Zugriffe auf Achsdaten sind über den ctrlX DataLayer mit den Funktionsbausteinen DL_ReadNode und DL_WriteNode möglich.

1.3.8.2 Anpassung der maximalen Achsanzahl

Die Achsstrukturen können an die tatsächlich vorhandene Achsanzahl angepasst werden.

Die Bibliothek **CXA_MotionInterfaceUser** erlaubt Anpassungen über die Bibliotheksparameter "MOTIF_CONFIG".

Mit den Konstanten "MIN_AXIS_INDEX" und "MAX_AXIS_INDEX" kann die Grösse der Strukturen passend zur Anwendung gewählt werden.

Wird das Kinematik-Interface verwendet, darf die Konstante "MAX_AXIS_INDEX" nicht grösser als "Global_MB_KinInterfaceVars.MB_KINIF_MAX_AXIS_INDEX" gewählt werden. Es kommt ansonsten zu einem Initialisierungsfehler beim Start des SPS-Programmes.

1.3.8.3 Anpassung der Zuordnung Achsname<>AchsIndex

Das AchsInterface arbeitet mit einem AchsIndex zur Adressierung in den Achsstrukturen. Die Motion-Firmware arbeitet mit dem Achsnamen. Die Zuordnung Achsname<>AchsIndex kann mit verschiedenen Methoden erfolgen.

Die Bibliothek **CXA_MotionInterfaceUser** erlaubt eine Auswahl der Methode über die Bibliotheksparameter "MOTIF_CONFIG".

Zuordnung Achsname<>AchsIndex in MOTIF_CONFIG.CFG_MODE_AXS

- AUTO: Auslesen des DataLayer Knotens "motion/axs" und Zuweisung des AchsIndex in der hier vorgefundenen Reihenfolge
- GLOB_VAR: in der Application wird eine Liste von "MB_AXISIF_REF" definiert und an das Programm "TE_AxisInterfaceMainProg" übergeben. Siehe "GlobalAxisDefines" im Programm-Template "ctrlX CORE Axis/Kin-Interface".

1.3.8.4 Achse hinzufügen

Eine Achse kann in der Bedienoberfläche im Bereich Motion angelegt werden oder auch z.B. aus dem SPS-Programm erzeugt werden.

Für eine hinzugefügte Achse muss ggf. eine Initialisierung bestimmter Strukturelemente des Achs-Interface vorgenommen werden. Dies geschieht bei Verwendung des Programm-Template "ctrlX CORE Axis/Kin-Interface" automatisch beim Erreichen des Zustandes "Running". Die Zuordnung Achsname<>AchsIndex muss ergänzt werden, wenn MOTIF_CONFIG.CFG_MODE_AXS = GLOB_VAR konfiguriert ist. Siehe auch oben ➚ *weitere Informationen auf Seite 43*.

1.3.8.5 Achse entfernen/umbenennen

Eine vorhandene Achse kann in der Bedienoberfläche im Bereich Motion oder über diverse Schnittstellen gelöscht bzw. umbenannt werden.

Wird eine Achse umbenannt, muss der Zugriff über die Control- u. Statusstrukturen "arAxisCtrl_gb[geänderterAchsname.AxisNo]" und "arAxisStatus_gb[geänderterAchsname.AxisNo]" innerhalb des SPS-Programmes angepasst werden.

Die Zuordnung Achsname<>AchsIndex muss angepasst werden, wenn MOTIF_CONFIG.CFG_MODE_AXS = GLOB_VAR konfiguriert ist. Siehe auch oben ➚ *weitere Informationen auf Seite 43*.

1.3.8.6 Achs-Interface Erweiterungen

Das Achs-Interface erlaubt fast beliebige Erweiterungen der Achs-Interface-Strukturen. Es können zusätzliche Unterstrukturen eingefügt und auch die vorhandenen Unterstrukturen erweitert werden.

Die folgenden Strukturelemente sind als Anwendererweiterungen in der AxisCtrl-Struktur beispielhaft programmiert:

- Admin: Erweiterung der vorhandenen Unterstruktur in der AxisCtrl-Struktur
  - ReTriggerOpMode: Neustart der eingestellten Betriebsart, z. B. MoveRelative mit der gleichen Distance starten
- SetupMode: zusätzliche Unterstruktur in der AxisCtrl-Struktur
  - Enable: Freigabe Einrichtbetrieb
  - JogPlus: Tippen +
  - JogMinus: Tippen -
  - Vel: Tippgeschwindigkeit
  - DynValues: Tipp(brems)beschleunigung und Ruck.
  - JogIncr: TRUE -> inkrementelles Tippen. Eine positive Flanke an JogPlus oder JogMinus vertippt eine Schrittweite
  - StepWidth: Schrittweite für das inkrementelle Tippen

Die folgenden Strukturelemente sind als Anwendererweiterungen in der AxisStatus-Struktur beispielhaft programmiert:

- SetupMode: zusätzliche Unterstruktur in der AxisStatus-Struktur
  - EnableAck: Einrichtbetrieb ist aktiv

Der Code zu diesen Erweiterungen ist in den Aktionen des Bausteins TE_AxisInterface() im Ordner "AxisInterfaceUser/POUS" zu finden. Die dazugehörigen Strukturen sind in den Ordnern "AxisInterfaceUser/DUTs/Control" und "AxisInterfaceUser/DUTs/Status" zu finden.

Es können eigene Erweiterungen hinzugefügt werden (siehe dazu ➚ *, Seite*).

1.4 Kinematik-Interface

1.4.1 Einführung und Übersicht

Das **Kinematik-Interface** bündelt und erweitert PLCopen-Bewegungsfunktionsbausteine und stellt ein einfach zu bedienendes Interface für die Kinematikfunktionalität zur Verfügung.

Weniger Code und leistungsfähigere Kommandos beschleunigen die Programmentwicklung von Applikationen.

Das Kinematik-Interface enthält Steuersignale und Parameter für die verschiedenen Betriebsarten der Kinematiken.

Tab. 30: Programmorganisationseinheiten des Kinematik-Interface in der Bibliothek CXA_MotionInterface

Anwendungsgebiet (Ordner in der Bibliothek CXA_MotionInterface)	
POUs	Beschreibung
KinInterface/POUs	
↗ , *Seite*	Wird zur Initialisierung des Kinematik-Interfaces für jede Kinematik benutzt. Der Funktionsbaustein muss nur einmal beim Programmstart bzw. bei jeder Phasenumschaltung vom Parametriermodus in den Betriebsmodus aufgerufen werden
↗ , *Seite*	Wird zur Konfiguration des Kinematik-Interfaces für jede Kinematik benutzt. Der Funktionsbaustein muss zyklisch (im Motion-Takt oder langsamer als der Motion-Takt) aufgerufen werden, solange man sich im Betriebsmodus befindet
KinInterface/DUTs	
	Informationen zu den Datenstrukturen siehe Online-Dokumentation in der Bibliothek **CXA_MotionInterface** im Ordner "KinInterface/DUTs".
KinInterface/GlobalVariables	
	Informationen zu den globalen Variablen siehe Online-Dokumentation in der Bibliothek **CXA_MotionInterface** im Ordner "KinInterface/GlobalVariables".

CXA_MotionInterface.library

Kinematik-Interface

Kinematik-Interface wird als Programmiertemplate oder als standalone-Interface für die Kinematikfunktionalität zur Verfügung gestellt.

Wenn es mit dem Programmiertemplate "ctrlX CORE Axis-/Kin-Interface" benutzt wird, muss sich der Anwender nicht mit Instanz-Aufrufen der Funktionsbausteine innerhalb des Projektes befassen. Diese Funktionalität ist komplett in das Template integriert und der Anwender muss nur ein paar Zeilen Code schreiben.

Wird hingegen das Kinematik-Interface als eigenständige Funktionalität benutzt, erfordert dies das Anlegen von Instanzen von beiden Funktionsbausteinen für jede Kinematik durch den Anwender.

Das Kinematik-Interface nutzt intern das Achs-Interface. Es ist also zwingend erforderlich auch das Achs-Interface aufzurufen.

Tab. 31: Programmorganisationseinheiten des Achs-Interface in der Bibliothek CXA_MotionInterfaceUser

Anwendungsgebiet (Ordner in der Bibliothek CXA_MotionInterfaceUser)	
POUs	Beschreibung
KinInterfaceUser/POUs	
TE_KinematicsInitAllKinematics	Initialisierung des Kinematik-Interfaces für alle Kinematiken. Ruft intern den ↗ *, Seite* für jede Kinematik auf
TE_KinematicsInterface erweitert ↗ *, Seite*	Hier kann das Kinematik-Interface für eine einzelne Kinematik durch den Anwender erweitert werden ↗ *Kapitel 1.4.7 „Kinematik-Interface Anwender-Erweiterung “ auf Seite 75*. Der Funktionsbaustein muss zyklisch (im Motion-Takt oder langsamer als der Motion-Takt) aufgerufen werden, solange man sich im Betriebsmodus befindet
TE_KinInterfaceMainProg	Das Hauptprogramm führt bei Erreichen des Modus "Running" die Initialisierung aus und nach erfolgreicher Initialisierung wird der TE_KinematicsInterface für alle Kinematiken aufgerufen
	Weitere Informationen siehe Online-Dokumentation in der Bibliothek **CXA_MotionInterfaceUser** im Ordner "KinInterfaceUser/POUs".
KinInterfaceUser/DUTs	
	Informationen zu den Datenstrukturen siehe Online-Dokumentation in der Bibliothek **CXA_MotionInterfaceUser** im Ordner "KinInterfaceUser/DUTs".
KinInterfaceUser/GlobalVariables	

Global_Kinematics_Interface	Hier ist das eigentliche Kinematik-Interface mit den Arrays **arKinCtrl_gb** und **arKinStatus_gb** zu finden. Die weiteren Variablen werden intern bzw. von den Visualisierungen genutzt. Siehe auch ↗ *Kapitel 1.4.4 „Kinematik-Interface - Globale Variablen “ auf Seite 66*
KinInterfaceUser/Visualizations	
↗ *Kapitel 1.4.6.4 „Kinematik-Interface Visualisierungen “ auf Seite 73*	Inbetriebnahmevisualisierungen, z.B. zum Vertippen der Achsen
KinInterfaceUser/_Examples	
PROGRAM Example_KinIfApplicationPart	Beispielcode zur Anwendung des Kinematik-Interfaces. Im Programm-Template "ctrlX CORE Axis-/Kin-Interface" ist dieser Beispielcode auch enthalten.

Projektierungshinweis/Laufzeitbedarf

Für jede Kinematik des Kinematik-Interface wird der Kinematik-Interface-Funktionsbaustein aufgerufen. Dieser Aufruf benötigt Laufzeit der SPS. Diese Laufzeit variiert je nach Kinematiktyp und Kinematikbetriebsart. Zusätzlich muss für jede Achse der Achs-Interface-Funktionsbaustein aufgerufen werden.

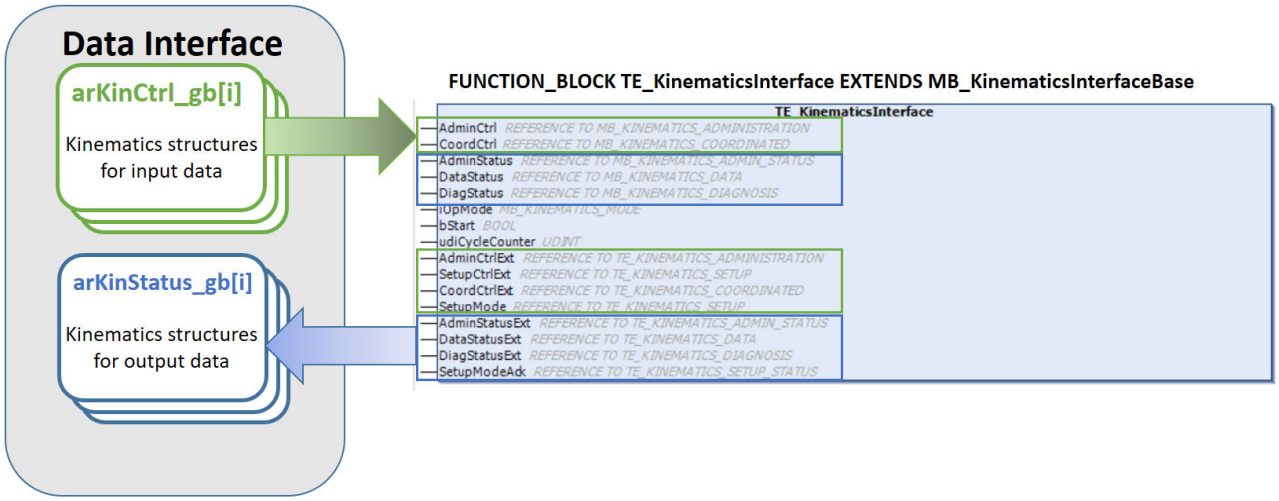
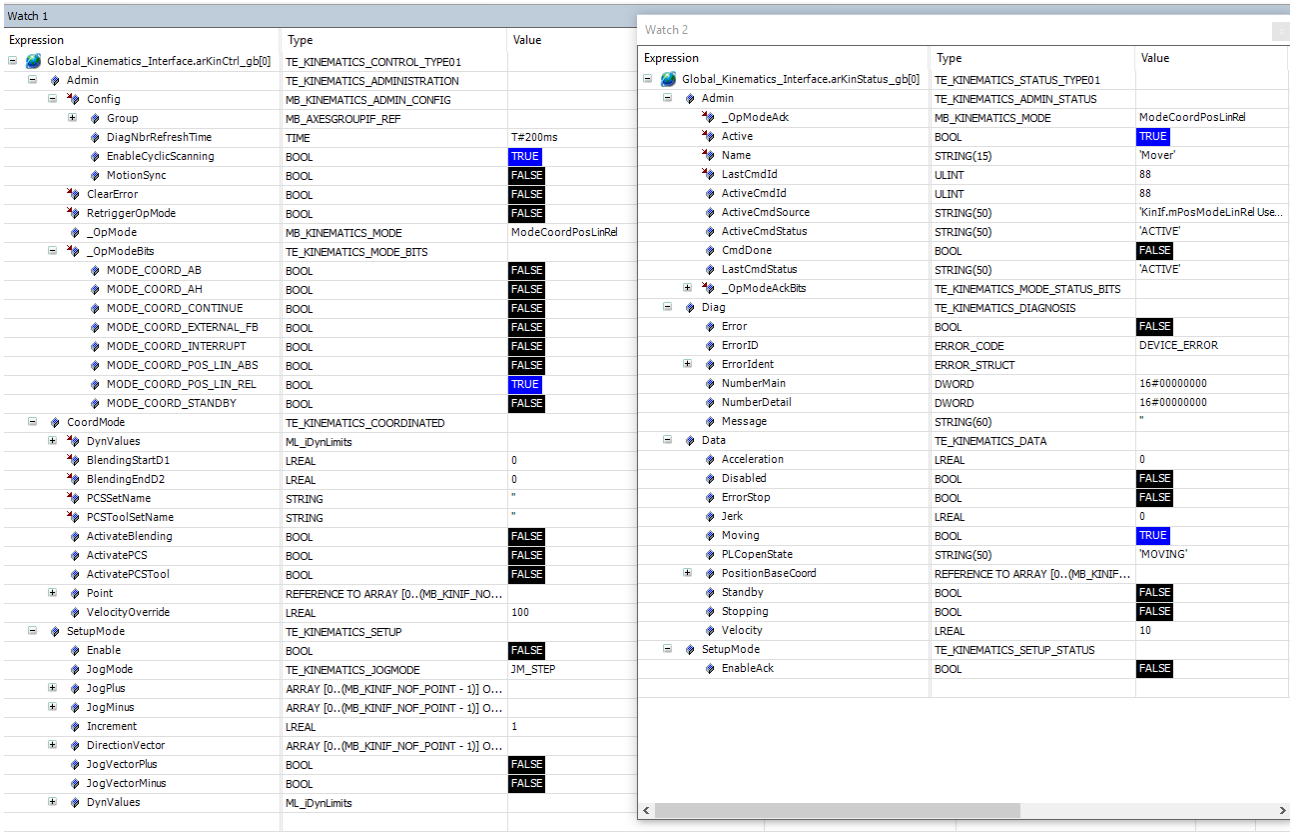


Abb. 9: Kin-Interface Datenstruktur des Interface

Tab. 32: Zuordnung Anwender-Interface zur Kinematik-Interface-Datenstruktur:

Anwender-Interface	Typ	Beschreibung
arKinCtrl[ ]_gb	TE_KINEMATICS_CONTROL_TYPE01	Steuerungsstruktur inklusive Sollwerte und Variablen zum Aktivieren der Betriebsarten
arKinStatus[ ]_gb	TE_KINEMATICS_STATUS_TYPE01	Statusstruktur inklusive Diagnoseinfo, Quittungen für die Betriebsarten, Istwerten und Statusinformation der App **rexroth-motion**.

Kinematik-Interface



Watch 1

Expression	Type	Value
Global_Kinematics_Interface.arKinCtrl_gb[0]	TE_KINEMATICS_CONTROL_TYPE01	
Admin	TE_KINEMATICS_ADMINISTRATION	
Config	MB_KINEMATICS_ADMIN_CONFIG	
Group	MB_AXESGROUPIF_REF	
DiagNbrRefreshTime	TIME	T#200ms
EnableCyclicScanning	BOOL	TRUE
MotionSync	BOOL	FALSE
ClearError	BOOL	FALSE
RetriggerOpMode	BOOL	FALSE
_OpMode	MB_KINEMATICS_MODE	ModeCoordPosLinRel
_OpModeBits	TE_KINEMATICS_MODE_BITS	
MODE_COORD_AB	BOOL	FALSE
MODE_COORD_AH	BOOL	FALSE
MODE_COORD_CONTINUE	BOOL	FALSE
MODE_COORD_EXTERNAL_FB	BOOL	FALSE
MODE_COORD_INTERRUPT	BOOL	FALSE
MODE_COORD_POS_LIN_ABS	BOOL	FALSE
MODE_COORD_POS_LIN_REL	BOOL	TRUE
MODE_COORD_STANDBY	BOOL	FALSE
CoordMode	TE_KINEMATICS_COORDINATED	
DynValues	ML_iDynLimits	
BlendingStartD1	LREAL	0
BlendingEndD2	LREAL	0
PCSSetName	STRING	''
PCSToolSetName	STRING	''
ActivateBlending	BOOL	FALSE
ActivatePCS	BOOL	FALSE
ActivatePCSTool	BOOL	FALSE
Point	REFERENCE TO ARRAY [0..(MB_KINIF_NO...	
VelocityOverride	LREAL	100
SetupMode	TE_KINEMATICS_SETUP	
Enable	BOOL	FALSE
JogMode	TE_KINEMATICS_JOGMODE	JM_STEP
JogPlus	ARRAY [0..(MB_KINIF_NOF_POINT - 1)] O...	
JogMinus	ARRAY [0..(MB_KINIF_NOF_POINT - 1)] O...	
Increment	LREAL	1
DirectionVector	ARRAY [0..(MB_KINIF_NOF_POINT - 1)] O...	
JogVectorPlus	BOOL	FALSE
JogVectorMinus	BOOL	FALSE
DynValues	ML_iDynLimits	

Watch 2

Expression	Type	Value
Global_Kinematics_Interface.arKinStatus_gb[0]	TE_KINEMATICS_STATUS_TYPE01	
Admin	TE_KINEMATICS_ADMIN_STATUS	
_OpModeAck	MB_KINEMATICS_MODE	ModeCoordPosLinRel
Active	BOOL	TRUE
Name	STRING(15)	'Mover'
LastCmdId	ULINT	88
ActiveCmdId	ULINT	88
ActiveCmdSource	STRING(50)	'KinIf.mPosModeLinRel Use...
ActiveCmdStatus	STRING(50)	'ACTIVE'
CmdDone	BOOL	FALSE
LastCmdStatus	STRING(50)	'ACTIVE'
_OpModeAckBits	TE_KINEMATICS_MODE_STATUS_BITS	
Diag	TE_KINEMATICS_DIAGNOSIS	
Error	BOOL	FALSE
ErrorID	ERROR_CODE	DEVICE_ERROR
ErrorIdent	ERROR_STRUCT	
NumberMain	DWORD	16#00000000
NumberDetail	DWORD	16#00000000
Message	STRING(60)	''
Data	TE_KINEMATICS_DATA	
Acceleration	LREAL	0
Disabled	BOOL	FALSE
ErrorStop	BOOL	FALSE
Jerk	LREAL	0
Moving	BOOL	TRUE
PLCopenState	STRING(50)	'MOVING'
PositionBaseCoord	REFERENCE TO ARRAY [0..(MB_KINIF...	
Standby	BOOL	FALSE
Stopping	BOOL	FALSE
Velocity	LREAL	10
SetupMode	TE_KINEMATICS_SETUP_STATUS	
EnableAck	BOOL	FALSE

Abb. 10: Überblick über die Datenstrukturen des Kinematik-Interface

Benutzen Sie die GroupNo der MB_AXESGROUPIF_REF-Struktur als Index für das Feld, z. B. arKinCtrl_gb[Mover.GroupNo].Admin. usw.

EnableCyclicScanning

Die interne Handhabung einiger Sollwerte kann durch das arKinCtrl_gb[ ].Admin.Config.EnableCyclicScanning Element gesteuert werden.

Wird "EnableCyclicScanning" auf TRUE gesetzt, werden einige Sollwerte der arKinCtrl_gb[ ]-Struktur zyklisch gescannt und sofort wirksam, wenn sich ein Wert ändert.

Global_Kinematics_Interface.arKinCtrl_gb[0]	TE_KINEMATICS_CONTROL_TYPE01	
Admin	TE_KINEMATICS_ADMINISTRATION	
Config	MB_KINEMATICS_ADMIN_CONFIG	
Group	MB_AXESGROUPIF_REF	
DiagNbrRefreshTime	TIME	T#200ms
EnableCyclicScanning	BOOL	TRUE
MotionSync	BOOL	FALSE
ClearError	BOOL	FALSE
RetriggerOpMode	BOOL	FALSE
_OpMode	MB_KINEMATICS_MODE	ModeCoordPosLinRel
_OpModeBits	TE_KINEMATICS_MODE_BITS	
CoordMode	TE_KINEMATICS_COORDINATED	
DynValues	ML_iDynLimits	
Velocity	LREAL	10
Acceleration	LREAL	10
Deceleration	LREAL	10
JerkAcc	LREAL	0
JerkDec	LREAL	0
BlendingStartD1	LREAL	0
BlendingEndD2	LREAL	0
PCSSetName	STRING	''
PCSToolSetName	STRING	''
ActivateBlending	BOOL	FALSE
ActivatePCS	BOOL	FALSE
ActivatePCSTool	BOOL	FALSE
Point	REFERENCE TO ARRAY [0..(MB_KINI...	
VelocityOverride	LREAL	100
SetupMode	TE_KINEMATICS_SETUP	

Abb. 11: Zyklisch gescannte Elemente von arKinCtrl_gb[ ] sind hervorgehoben

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

Für Inbetriebnahmezwecke stehen verschiedene IndraLogic-Visualisierungen, basierend auf den Strukturelementen, die in diesem Abschnitt beschrieben werden, in der Bibliothek CXA_MotioninterfaceUser zur Verfügung.

Was ist neu bzw. geändert gegenüber der Version für MLC/MLD

- Es wurden Teile des ereignisgesteuerten Kinematik-Interface (Funktionsbaustein MB_KinematicsInterfaceType12) übernommen. Die Strukturelemente sind zum Teil als Properties implementiert. Die Unterstrukturen sind dann als Funktionsbausteine anstatt Strukturen implementiert um Properties nutzen zu können. In einer Struktur ist eine Methode arKinCtrl_gb[].Admin.mTriggerMoveCmd() implementiert.
- Die Betriebsartenanwahl arKinCtrl_gb[].Admin._OpMode ist nicht mehr als "UNION" implementiert sondern als Properties umgesetzt. Bei der Ansteuerung über Bits (_OpModeBits) ist damit keine Mehrfachanwahl mehr möglich.
- Selten verwendete Elemente von arKinCtrl_gb[].Admin wurden in arKinCtrl_gb[].Admin.Config verschoben (siehe Tabelle unten).
- Werte vom Typ REAL werden jetzt generell als LREAL in den Strukturen definiert.
- Es gibt keine KinData[] Struktur. Die aktuellen Istwerte und einige Statusbits sind in arKinStatus_gb[].Data zu finden.

Tab. 33: Folgende Code-Änderungen sind bei einer Portierung von MLC/MLD mindestens notwendig (Suchen/Ersetzen).

Code MLC/MLD	Ersetzen durch
Kontrollstruktur arKinCtrl_gb[]	
_OpMode.en	_OpMode
ModeCoordInterrupted	ModeCoordInterrupt
ModeCoordStopping	ModeCoordStandby
_OpMode.b	_OpModeBits
MODE_COORD_INTERRUPTED	MODE_COORD_INTERRUPT
MODE_COORD_STOPPING	MODE_COORD_STANDBY
Admin.Group	Admin.Config.Group
Admin.DiagNbrRefreshTime	Admin.Config.DiagNbrRefreshTime
Admin.EnableCyclicScanning	Admin.Config.EnableCyclicScanning
CoordMode.Velocity	CoordMode.DynValues.Velocity
CoordMode.Acceleration	CoordMode.DynValues.Acceleration
CoordMode.Deceleration	CoordMode.DynValues.Deceleration
CoordMode.Jerk	CoordMode.DynValues.JerkAcc CoordMode.DynValues.JerkDec
CoordMode.BlendingRadius	CoordMode.BlendingStartD1 CoordMode.BlendingEndD2
SetupMode.Velocity[]	SetupMode.DynValues.Velocity

Code MLC/MLD	Ersetzen durch
SetupMode.Acceleration[]	SetupMode.DynValues.Acceleration
SetupMode.Deceleration[]	SetupMode.DynValues.Deceleration
SetupMode.Jerk[]	SetupMode.DynValues.JerkAcc SetupMode.DynValues.JerkDec
SetupMode.Distance[]	SetupMode.Increment
Kontrollstruktur arKinStatus_gb[]	
Admin.MODE_COORD_UNGROUPED	Admin._OpModeAck-Bits.MODE_COORD_UNGROUPED
Admin.MODE_COORD_CONTINUE	Admin._OpModeAckBits.MODE_COORD_CONTINUE
Admin.MODE_COORD_INTERRUPTED	Admin._OpModeAckBits.MODE_COORD_INTERRUPT
Admin.MODE_COORD_EXTERNAL_FB	Admin._OpModeAck-Bits.MODE_COORD_EXTERNAL_FB
Admin.MODE_COORD_POS_LIN_ABS	Admin._OpModeAck-Bits.MODE_COORD_POS_LIN_ABS
Admin.MODE_COORD_POS_LIN_REL	Admin._OpModeAck-Bits.MODE_COORD_POS_LIN_REL
Admin.MODE_COORD_STOPPING	Admin._OpModeAck-Bits.MODE_COORD_STANDBY

Diese Liste der Code-Änderungen ist nicht vollständig. Bei der Portierung ist eine generelle Überprüfung des Programm-Codes notwendig.

1.4.2 Kinematik-Interface - Funktionsbausteine

1.4.2.1 MB_KinematicsInit

Kurzbeschreibung

Der Funktionsbaustein **MB_KinematicsInit** wird zur Initialisierung des Kinematik-Interfaces für jede Kinematik benutzt.

Der Funktionsbaustein muss nur einmal beim Programmstart oder bei jeder Modusumschaltung von "Configuration" in "Running" aufgerufen werden. In der Vorlage "ctrlX CORE Axis/Kin-Interface" ist dies bereits implementiert.

Kinematik-Interface

Schnittstellenbeschreibung

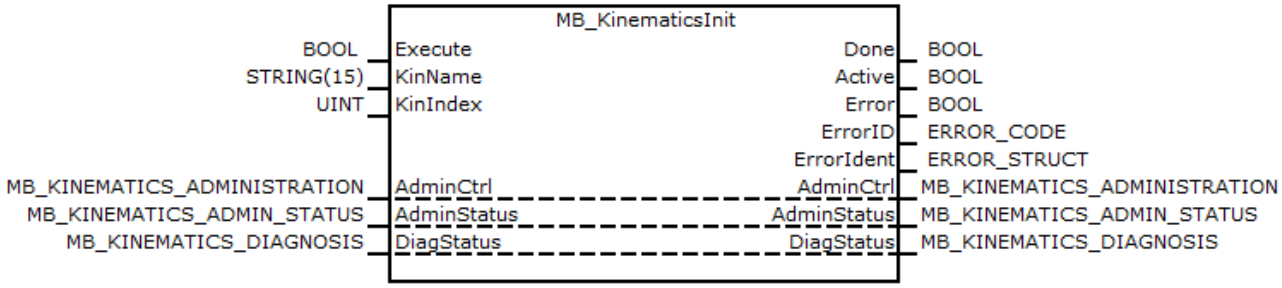


Abb. 12: Funktionsbaustein MB_KinematicsInit

Tab. 34: Schnittstellenvariablen MB_KinematicsInit

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	Execute	BOOL	Start der Initialisierung durch positive Flanke
	KinName	STRING(15)	Name der zu initialisierenden Kinematik
	KinIndex	UINT	Index in den Kinematik-Interface Strukturen
VAR_OUTPUT	Done	BOOL	Wird gesetzt, wenn der FB die Bearbeitung beendet hat
	Active	BOOL	Wird gesetzt, wenn der FB aktiv ist (nicht im Leerlaufbetrieb)
	Error	BOOL	Zeigt an, dass ein Fehler in der FB-Instanz aufgetreten ist
	ErrorID	ERROR_CODE	Kurzer Hinweis zur Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Information zum Fehler
VAR_IN_OUT	AdminCtrl	MB_KINEMATICS_ADMINISTRATION	Verwaltung der Kinematik arKinCtrl_gb[ ].Admin
	AdminStatus	MB_KINEMATICS_ADMIN_STATUS	Status Verwaltung der Kinematik arKinStatus_gb[ ].Admin
	DiagStatus	MB_KINEMATICS_DIAGNOSIS	Diagnoseinformationen der Kinematik arKinStatus_gb[ ].Diag

Es ist nicht möglich anstelle der einzelnen Strukturelemente "AdminCtrl", "AdminStatus", etc. die kompletten Strukturen "MB_Kinematics_Control_Type01" und "MB_Kinematics_Status_Type01" dem Funktionsbaustein zu übergeben.

Dies wurde vorgenommen, um der Anwender-Applikation spezielle Erweiterungen zum bestehenden Code zu ermöglichen.

Deshalb werden die benötigten Elemente von TE_KINEMATICS_CONTROL_TYPE01 und TE_KINEMATICS_STATUS_TYPE01 als separate Eingänge übergeben.

Funktionsbeschreibung

Tab. 35: Der Funktionsbaustein MB_KinematicsInit initialisiert die folgenden Strukturelemente mit Standardwerten:

Administration Control	
AdminCtrl._OpMode	ModeCoordAB
AdminCtrl.Config.Group.GroupNo	KinIndex
AdminCtrl.Config.Group.KinName	KinName
Administration Status	
AdminStatus.Active	TRUE für aktive Kinematik
AdminStatus.Name	KinName

Fehlerbehandlung

Die Fehlercodes der intern benutzten Funktionsbausteine DL_ReadNode und DL_BrowseNode zum Lesen von Datalayer Knoten werden durchgereicht. Der Funktionsbaustein kann die folgenden Fehlercodes erzeugen:

Tab. 36: Fehlercodes MB_KinematicsInit

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230140	Stringlänge KinName ausserhalb des gültigen Bereiches [1..15]
STATE_MACHINE_ERROR	16#0A0F0107	16#0C230141	Fehler im Ablauf des Funktionsbaustein
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230142	Zu viele Achsen für diese Kinematik konfiguriert
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230143	KinIndex ausserhalb des gültigen Bereiches [1..MB_KINIF_MAX_KIN_NUMBER]

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230144	Ungültiger Pointer arpAdminAxisCtrl_gb für mindestens eine Achse

1.4.2.2 MB_KinematicsInterfaceBase

Kurzbeschreibung

Der Funktionsbaustein **MB_KinematicsInterfaceBase** wird zur Konfiguration des Kinematik-Interfaces für jede Kinematik benutzt.

Dieser Funktionsbaustein muss zyklisch (im Motion-Takt oder langsamer als der Motion-Takt) aufgerufen werden solange man sich im Modus "Running" befindet. In der Vorlage "ctrlX CORE Axis/Kin-Interface" ist dies bereits implementiert.

Schnittstellenbeschreibung

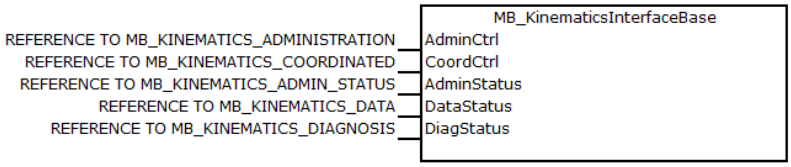


Abb. 13: Funktionsbaustein MB_KinematicsInterfaceBase

Tab. 37: Schnittstellenvariablen MB_KinematicsInterfaceBase

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	AdminCtrl	REFERENCE TO MB_KINEMATICS_ADMINISTRATION	Verwaltung der Kinematik arKinCtrl_gb[ ].Admin
	CoordCtrl	REFERENCE TO MB_KINEMATICS_COORDINATED	Betriebsart koordinierte Bewegung arKinCtrl_gb[ ].CoordMode
	AdminStatus	REFERENCE TO MB_KINEMATICS_ADMIN_STATUS	Status Verwaltung der Kinematik arKinStatus_gb[ ].Admin
	DataStatus	REFERENCE TO MB_KINEMATICS_DATA	Istwerte und Status der Kinematik arKinStatus_gb[ ].Data
	DiagStatus	REFERENCE TO MB_KINEMATICS_DIAGNOSIS	Status Diagnose der Achse arKinStatus_gb[ ].Diag

Es ist nicht möglich anstelle der einzelnen Strukturelemente "AdminCtrl", "CoordCtrl", etc. die kompletten Strukturen über einen Eingang dem Funktionsbaustein zu übergeben.

Dies wurde gemacht um der Anwender-Applikation spezielle Erweiterungen zum bestehenden Code zu ermöglichen.

Deshalb werden die benötigten Elemente von "TE_Kinematics_Control_Type01" und "TE_Kinematics_Status_Type01" als separate Eingänge übergeben.

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

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

Zum Beispiel führt das Setzen von "arKinCtrl_gb[ ].Admin._OpMode" von "ModeCoordAb" auf "ModeCoordPosLinAbs" zu folgendem Ablauf:

- Alle zugeordneten Achsen werden über das Achs-Interface in "COORDINATED_MOTION" geschaltet
- Aktivierung der Funktion **ML_KinEnable**
- Warten auf die Quittung, dass die Kinematik bereit ist
- Aktivierung der Funktion **ML_KinMoveLinAbs** mit den Sollwerten von CoordModeCtrl
- Quittieren des arKinStatus_gb[ ].Admin._OpModeAck auf ModeCoordPosLinAbs (Bit MODE_COORD_POS_LIN_ABS)
- Scannen der Werte in CoordModeCtrl.Position[] und erneutes Aktivieren des **ML_KinMoveLinAbs** im Fall von Änderungen

Fehlerbehandlung

Die Fehlercodes der intern benutzten Funktionsbausteine (z.B. DL_ReadNode zum Lesen von Datalayer Knoten) und der intern benutzten Funktionen ((z.B. ML_KinEnable) werden durchgereicht. Der Funktionsbaustein kann die folgenden Fehlercodes erzeugen:

Tab. 38: Fehlercodes des Funktionsbausteins MB_KinematicsInterfacebase

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230150	Mindestens einer der Funktionsbaustein-Eingänge ist nicht initialisiert
DEVICE_ERROR	16#0A0F0107	16#0C230151	Kinematik ist im ErrorStop
STATE_MACHINE_ERROR	16#0A0F0107	16#0C230153	Fehler im Ablauf des Funktionsbaustein

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230154	Ungültiger Eingang AdminCtrl.Config.Group.GroupNo
OTHER_ERROR	16#0A0F0107	16#0C230155	Fehler beim Gruppieren der Kinematik
RESOURCE_ERROR	16#0A0F0107	16#0C230156	Mit der Methode Ctrl.Admin.mTriggerMoveCmd() wurde in ModeCoordUngrouped eine Betriebsart angewählt
IINPUT_INVALID_ERROR	16#0A0F0107	16#0C230157	OpMode wird von dem Funktionsbaustein in dieser Variante nicht unterstützt
INPUT_RANGE_ERROR	16#0A0F0107	16#0C230158	Globale Achs-Pointer sind ungültig
STATE_MACHINE_ERROR	16#0A0F0107	16#0C230159	Fehler im Ablauf des Funktionsbaustein - Reset
STATE_MACHINE_ERROR	16#0A0F0107	16#0C23015A	Fehler im Ablauf des Funktionsbaustein - Stop

1.4.3 Kinematik-Interface - Betriebsarten

1.4.3.1 Überblick

Der folgende Abschnitt beschreibt die Betriebsarten, die vom Kinematik-Interface Basis Typ unterstützt werden und die Attribute, die zugewiesen werden können.

Es gibt drei Methoden um eine Betriebsart zu aktivieren:

- **Auswahl über ENUM-Werte**

  Zuweisen eines Wertes vom TYPE MB_KINEMATICS_MODE an

  arKinCtrl_gb[ ].Admin._OpMode:

  ```
  arKinCtrl_gb[].Admin._OpMode:= ModeCoordPosLinAbs;
  ```

  - oder -

  ```
  arKinCtrl_gb[].Admin._OpMode:= ModeCoordAB;
  ```

- **Benutzung des Bit-Zugriffs**

  Setzen eines Bits über die Bit-Zugriffs Funktionalität.

  ```
  arKinCtrl_gb[].Admin._OpModeBits.MODE_COORD_POS_LIN_ABS:= TRUE;
  ```

Löschen des "_OpMode" durch Bit-Zugriffs Funktionalität ist auch möglich.

`arKinCtrl_gb[].Admin._OpMode-Bits.MODE_COORD_POS_LIN_ABS := FALSE;` Durch das Bit-Löschen wird MODE_COORD_AB aktiviert.

Auch möglich: `arKinCtrl_gb[].Admin._OpMode-Bits.MODE_COORD_AB := TRUE;`

- **Benutzung der Methode Ctrl.Admin.mTriggerMoveCmd()**

  Siehe auch "DemoKinematicsCommands" in der Vorlage "ctrlX CORE Axis/Kin-Interface"

  Diese Methode setzt sofort in dem Kontext dieses Aufrufes den Befehl an die Motion-Firmware ab. Damit dies funktioniert, muss die Kinematik bereits freigegeben sein, z.B. ModeCoordStandby und CmdDone abfragen.

  Aufruf: `arKinCtrl_gb[uiKinIndex].Admin.mTrigger-MoveCmd(_OpMode:= ModeCoordPosLinAbs, UserID:='my text');`

  - Übergabeparameter **_Opmode** = Wert vom TYPE MB_KINEMATICS_MODE
  - Übergabeparameter **UserID** = String (max. 25 byte). Wird als "Source" übergeben bei Aufruf von Motionkommandos. Bei Fehlern kann so die Quelle des Befehls identifiziert werden.
  - Returnwert der Methode: "cmdID" des abgesetzten Motionkommandos. Bei Fehler wird 16#FFFFFFFFFFFFFFFF zurückgegeben.

Mit der Methode arKinCtrl_gb[uiKinIndex].Admin.mTriggerMoveCmd() ist es möglich mehrere gepufferte Befehle in einem Zyklus abzusetzen. Im Gegensatz zu Achsen sind bei Kinematiken alle Befehle gepuffert. Deshalb hat die Methode keinen Übergabeparameter "Buffered".

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

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

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

Beispiel:

- *Die Status-Quittung gibt nur dann `TRUE` zurück, wenn das Kommando zum Umschalten der Kinematik auf absolutem, linearem Positionierungsbetrieb ausgeführt wurde:*
  `arKinStatus_gb[].Admin.MODE_COORD_POS_LIN_ABS`
- *Die Status-Quittung wird sofort beim Absetzen eines neuen Kommandos zurückgesetzt:*
  `arKinStatus_gb[].Admin.CmdDone`
  Die Positionierungsbetriebsart wird aktiviert:
  `arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_POS_LIN_ABS`
  Der Ausgang wird auf `TRUE` gesetzt, wenn die Kinematik in den Positionierungsbetrieb schaltet und anfängt sich zu bewegen:
  `arKinStatus_gb[].Admin.MODE_COORD_POS_LIN_ABS`
  Hat die Kinematik die Zielkoordinaten erreicht, wird die Statusquittung gesetzt:
  `arKinStatus_gb[].Admin.CmdDone`

1.4.3.2 Kinematik Bereit

Die Aktivierung dieser Betriebsart schaltet die Kinematik in "GroupDisabled" und schaltet an allen zugeordneten Achsen das Drehmoment ab. Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode:= ModeCoordAB;
```

oder

```
arKinCtrl_gb[].Admin._OpModeBits.MODE_COORD_AB:=
TRUE;
```

*Das KinematikInterface benutzt intern die Funktion **ML_KinDisable** (Bibliothek CXA_Motion) und die Achs-Interface Betriebsart **ModeAb**, um die Umschaltung durchzuführen.*

Attribute Kinematik Bereit

Tab. 39: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinStatus_gb[]	Admin._OpModeAck-Bits.MODE_COORD_UNGROUPED	BOOL		entfällt

1.4.3.3 Kinematik Halt

Die Aktivierung dieser Betriebsart schaltet die Kinematik in "Group-Disabled" und schaltet alle zugeordneten Achsen in AH (Antrieb Halt) unter Aufrechterhaltung des Drehmoments. Folgendes Kommando aktiviert die Betriebsart:

Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode := ModeCoordAH;
```

oder

```
arKinCtrl_gb[].Admin._OpModeBits.MODE_COORD_AH :=
TRUE;
```

*Das KinematikInterface benutzt intern die Funktion **ML_KinDisable** (Bibliothek CXA_Motion) und die Achs-Interface Betriebsart **ModeAH**, um die Umschaltung durchzuführen.*

Attribute Kinematik Halt

Tab. 40: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinStatus_gb[]	Admin._OpModeAck-Bits.MODE_COORD_UNGROUPED	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

1.4.3.4 Absolutes lineares Positionieren

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

Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode := ModeCoordPosLi-
nAbs;
```

oder

```
arKinCtrl_gb[].Admin._OpMode-
Bits.MODE_COORD_POS_LIN_ABS := TRUE;
```

Die folgende Quittung muss dem Betriebsarten-Kommando folgen und gibt `TRUE` zurück, wenn die Kinematik die Zielkoordinaten erreicht hat:

```
arKinStatus_gb[].Admin.CmdDone
```

*Das KinematikInterface benutzt intern die Achs-Interface Betriebsart **ModeCoordinated** und die Funktionen **ML_KinEnable** und **ML_KinMoveLinAbs** (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.*

Attribute Absolutes lineares Positionieren

Tab. 41: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	CoordMode.Point[0..15]	LREAL	0.0	Ja
	CoordMode.DynValues.Velocity	LREAL	10.0	Nein
	CoordMode.DynValues.Acceleration	LREAL	10.0	Nein
	CoordMode.DynValues.Deceleration	LREAL	10.0	Nein
	CoordMode.DynValues.JerkAcc	LREAL	0.0	Nein
	CoordMode.DynValues.JerkDec	LREAL	0.0	Nein
arKinStatus_gb[ ]	Admin._OpModeAckBits.MODE_COORD_POS_LIN_ABS	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

1.4.3.5 Relatives lineares Positionieren

Die Aktivierung dieser Betriebsart führt eine relative lineare Bewegung auf vorher festgelegte Zielkoordinaten durch Addition der CoordMode.Point[] zu den aktuellen Istkoordinaten aus.

Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode := ModeCoordPos-
LinRel;
```

oder

```
arKinCtrl_gb[].Admin._OpMode-
Bits.MODE_COORD_POS_LIN_REL := TRUE;
```

Die folgende Quittung muss dem Betriebsarten-Kommando folgen und gibt `TRUE` zurück, wenn die Kinematik die Zielkoordinaten erreicht hat:

```
arKinStatus_gb[].Admin.CmdDone
```

*Das KinematikInterface benutzt intern die Achs-Interface Betriebsart **ModeCoordinated** und die Funktionen **ML_KinEnable** und **ML_KinMoveLinRel** (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.*

Attribute Relatives lineares Positionieren

Tab. 42: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	CoordMode.Point[0..15]	LREAL	0.0	Ja
	CoordMode.DynValues.Velocity	LREAL	10.0	Nein
	CoordMode.DynValues.Acceleration	LREAL	10.0	Nein
	CoordMode.DynValues.Deceleration	LREAL	10.0	Nein
	CoordMode.DynValues.JerkAcc	LREAL	0.0	Nein
	CoordMode.DynValues.JerkDec	LREAL	0.0	Nein
arKinStatus_gb[ ]	Admin._OpModeAckBits.MODE_COORD_POS_LIN_REL	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

1.4.3.6 Betriebsart "Standby"

Wird die Betriebsart von ModeCoordAb oder ModeCoordAH auf ModeCoordStandby geändert, schaltet das Kinematik-Interface alle zugeordneten Achsen in AF und gruppiert die Achsen zur Kinematik.

Bei Aktivierung dieser Betriebsart in "GroupMoving" werden alle laufenden Befehle abgebrochen und die Kinematik wird gestoppt mit dem Zielzustand "GroupStandby".

Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode := ModeCoordStandby;
```

oder

```
arKinCtrl_gb[].Admin._OpMode-
Bits.MODE_COORD_STANDBY:= TRUE;
```

*Das KinematikInterface benutzt intern die Achs-Interface Betriebsart **ModeCoordinated** und die Funktionen **ML_KinEnable** und **ML_KinAbort** (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.*

Attribute Betriebsart Standby

Tab. 43: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxis-Status_gb[ ]	Admin._OpModeAck-Bits.MODE_COORD_STANDBY	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

1.4.3.7 Betriebsart "Interrupt"

Bei Aktivierung dieser Betriebsart in "GroupMoving" werden alle laufenden Befehle unterbrochen und die Kinematik wird gestoppt. Die Kinematik bleibt aber im Zustand "GroupMoving" bis die Unterbrechung mit ModeCoordContinue aufgehoben wird.

Wird die Betriebsart von ModeCoordAb oder ModeCoordAH auf ModeCoordInterrupt geändert, dann schaltet das Kinematik-Interface alle zugeordneten Achsen in AF und gruppiert die Achsen zur Kinematik.

Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode := ModeCoordInter-
rupt;
```

oder

```
arKinCtrl_gb[].Admin._OpMode-
Bits.MODE_COORD_INTERRUPT:= TRUE;
```

*Das KinematikInterface benutzt intern die Funktion **ML_KinInterrupt** (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.*

Attribute Betriebsart "Interrupt"

Tab. 44: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxis-Status_gb[ ]	Admin._OpModeAck-Bits.MODE_COORD_INTER-RUPT	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

1.4.3.8 Betriebsart "Continue"

Bei Aktivierung dieser Betriebsart wird eine Unterbrechung durch ModeCoordInterrupt wieder aufgehoben.

Die Anwahl von ModeCoordContinue führt zu einem Fehler, wenn nicht vorher eine Unterbrechung ausgeführt wurde.

Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode := ModeCoordCon-
tinue;
```

oder

```
arKinCtrl_gb[].Admin._OpModeBits.MODE_COORD_CON-
TINUE:= TRUE;
```

*Das KinematikInterface benutzt intern die Funktion **ML_KinContinue** (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.*

Attribute Betriebsart "Continue"

Tab. 45: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arAxis-Status_gb[ ]	Admin._OpModeAck-Bits.MODE_COORD_CONTINUE	BOOL		entfällt
	Admin.CmdDone	BOOL		entfällt

1.4.3.9 Betriebsart "Externer Funktionsbaustein"

Wird die Betriebsart von ModeCoordAb oder ModeCoordAH auf ModeCoordExternalFB geändert, dann schaltet das Kinematik-Interface alle zugeordneten Achsen in AF, gruppiert die Achsen zur Kinematik und wartet auf einen externen Bewegungsbefehl, z. B. von einer Technologiefunktion.

Wenn sich die Kinematik bereits in einer Betriebsart befindet, dann wird bei der Aktivierung dieser Betriebsart kein Bewegungsbefehl vom Kinematik-Interface ausgeführt.

Folgendes Kommando aktiviert die Betriebsart:

```
arKinCtrl_gb[].Admin._OpMode := ModeCoordExter-
nalFB;
```

oder

```
arKinCtrl_gb[].Admin._OpMode-
Bits.MODE_COORD_EXTERNAL_FB:= TRUE;
```

*Das KinematikInterface benutzt intern die Achs-Interface Betriebsart **ModeCoordinated** und die Funktion **ML_KinEnable** (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.*

Attribute Betriebsart Externer Funktionsbaustein

Tab. 46: Attribute, die von dieser Betriebsart unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKin-Status_gb[ ]	Admin._OpModeAck-Bits.MODE_COORD_EXTERNAL_FB	BOOL		entfällt

1.4.3.10 Betriebsarten übergreifende Funktionen

Einige Funktionen wirken in allen Positionier Betriebsarten und werden in diesem Abschnitt beschrieben.

"VelocityOverride"

Mit dem Eingang "VelocityOverride" kann eine laufende Bewegung in der Geschwindigkeit reduziert werden. Erlaubt sind Werte zwischen **0.0%** (Kinematik steht) und **100.0%** (Bewegung wird mit vorgegebener Geschwindigkeit ausgeführt).

*Das KinematikInterface benutzt intern die Funktion **ML_SetOverride** (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.*

Tab. 47: Attribute, die von der Funktion "velocity override" unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	CoordMode.VelocityOverride	LREAL	100.0	ja

"PCS"

Mit den Eingängen "PCSSetName" und "ActivatePCS" kann ein Produktkoordinatensystem (PCS) für alle Bewegungskommandos aktiviert werden.

Eine steigende Flanke an "ActivatePCS" aktiviert die in "PCSSetName" als "Set" oder "Group" eingetragenen PCS Offsets und Orientierungen.

Mit einer fallenden Flanke an "ActivatePCS" wird das PCS wieder deaktiviert.

*Das KinematikInterface benutzt intern die Funktion **ML_KinPCSP** (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.*

Tab. 48: Attribute, die von der Funktion "PCS" unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	CoordMode.PCSSetName	STRING	''	nein
arKinCtrl_gb[ ]	CoordMode.ActivatePCS	BOOL	FALSE	ja

"PCSTool"

Mit den Eingängen "PCSToolSetName" und "ActivatePCSTool" kann ein Produktkoordinatensystem (PCS) bezogen auf ein Werkzeug für alle Bewegungskommandos aktiviert werden.

Eine steigende Flanke an "ActivatePCSTool" aktiviert die in PCSToolSetName" als "Set" oder "Group" eingetragenen PCS Offsets und Orientierungen.

Mit einer fallenden Flanke an "ActivatePCSTool" wird das PCS wieder deaktiviert.

*Das KinematikInterface benutzt intern die Funktion **ML_KinPCSToolP** (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.*

Tab. 49: Attribute, die von der Funktion "PCSTool" unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	CoordMode.PCSToolSetName	STRING	''	nein
arKinCtrl_gb[ ]	CoordMode.ActivatePCSTool	BOOL	FALSE	ja

"Blending"

Mit den Eingängen "BlendingStartD1", "BlendingEndD2" und "ActivateBlending" kann ein Überschleifen zwischen zwei Bewegungskommandos aktiviert werden. Ein grundlegender Anwendungsfall ist das Abrunden der Ecke zwischen zwei linearen Bahnen.

Eine steigende Flanke an "ActivateBlending" aktiviert die in "BlendingStartD1" und "BlendingEndD2" eingetragenen Überschleifgrenzen.

Mit einer fallenden Flanke an "ActivateBlending" wird das Überschleifen wieder deaktiviert.

*Das KinematikInterface benutzt intern die Funktion **ML_KinBlendP** (Bibliothek CXA_Motion), um die Umschaltung durchzuführen.*

Tab. 50: Attribute, die von der Funktion "Blending" unterstützt werden:

Element	Name	Typ	Standard	Zyklisch gescannt
arKinCtrl_gb[ ]	CoordMode.BlendingStartD1	LREAL	0.0	nein
arKinCtrl_gb[ ]	CoordMode.BlendingEndD2	LREAL	0.0	nein
arKinCtrl_gb[ ]	CoordMode.ActivateBlending	BOOL	FALSE	ja

1.4.4 Kinematik-Interface - Globale Variablen

Die Bibliothek CXA_MotionInterfaceUser enthält im Ordner "KinInterfaceUser/GlobalVariables" die globale Variablenliste "Global_Kinematics_Interface".

Diese Liste enthält die folgenden Strukturen/Variablen:

Tab. 51: VAR_GLOBAL Global_Kinematics_Interface

Name	Datentyp	Kommentar
arKinCtrl_gb[ ]	ARRAY [] OF TE_KINEMATICS_CONTROL_TYPE01	Kontrollstruktur des Kinematik-Interface
arKinStatus_gb[ ]	ARRAY [] OF TE_KINEMATICS_STATUS_TYPE01	Statusstruktur des Kinematik-Interface
arKinIdx_gb	ARRAY [] OF UINT	Nicht lückende Liste der Kinematiken
uiNofKinematics_gb	UINT	Anzahl der aktiven Kinematiken
VisuKinematicsNo	INT	Umschaltung des Kinematicsindex in den Visualisierungen
bClearErrorKin_gb	BOOL	Globales Fehler löschen

1.4.5 Kinematik-Interface - Strukturen

1.4.5.1 Überblick

Das Kinematik-Interface stellt eine Datenschnittstelle zur komfortablen Ansteuerung der Kinematiken zur Verfügung.

Informationen zu den Datenstrukturen siehe Online-Dokumentation in den Bibliotheken **CXA_MotionInterface** im Ordner "KinInterface/DUTs" bzw. in **CXA_MotionInterfaceUser** im Ordner "KinInterfaceUser/DUTs".

*Das Programm-Template "ctrlX CORE Axis/Kin-Interface" ist so vorbereitet, dass es durch den Anwender erweiterbar ist bzw. es sind einige Erweiterungen schon mit eingebaut. Um diese Erweiterungen zu ermöglichen, ist es nötig, eigene Strukturen im Anwenderprojekt zu definieren, die die Strukturen der Bibliothek erweitern. Die erweiterteten Strukturen sind durch den Präfix "TE_" gekennzeichnet. Wenn also imProgramm-Template "ctrlX CORE Axis/Kin-Interface" eine Struktur, z.B. **TE_**KINEMATICS_ADMIN_STATUS heisst, ist diese eine erweiterte Struktur der **MB_**KINEMATICS_ADMIN_STATUS.*

*Informationen zu der Struktur sind in der Online-Dokumentation unter dem Namen **MB_**KINEMATICS_ADMIN_STATUS in der Bibliothek **CXA_MotionInterface** und unter dem Namen **TE_**KINEMATICS_ADMIN_STATUS in der Bibliothek **CXA_MotionInterfaceUser** zu finden.*

1.4.6 Kinematik-Interface - Beispielprogramm

1.4.6.1 Überblick

In diesem Kapitel soll ein Überblick über die als offener Code verfügbaren Teile des Kinematik-Interface gegeben werden.

Die offenen Programmteile werden mit den folgenden Elementen geliefert:

- Das Programmiertemplate "ctrlX CORE Axis/Kin-Interface" dient als Beispielapplikation für das Kinematik-Interface. Siehe auch ↗ *Kapitel 1.2 „MotionInterface - Erstkonfiguration “ auf Seite 2*
- Die Bibliothek **CXA_MotionInterfaceUser.library** kann durch den Anwender verändert werden um das Kinematik-Interface an die jeweilige Applikation anzupassen. Siehe auch ↗ *Kapitel 1.4.7 „Kinematik-Interface Anwender-Erweiterung “ auf Seite 75*.

1.4.6.2 Programmiertemplate "ctrlX CORE Axis/Kin-Interface"

Im Programm "DemoKinematicsCommands" wird der Zugriff auf das Kinematik-Interface gezeigt.

Das Programmiertemplate "ctrlX CORE Axis/Kin-Interface" deckt die folgenden Punkte zum Kinematik-Interface ab:

- **PlcProg**:

  Aufruf des TE_KinInterfaceMainProg() - Initialisierung und Zyklischer Aufruf des Kinematik-Interface.
- **MotionProg**:

  Hier wird die Methode TE_KinInterfaceMainProg.mMotionTask() aufgerufen. In diesem Takt werden die Istwerte in arKinStatus_gb[].Data aufgefrischt. Falls die Methode nicht aufgerufen wird, werden die Istwerte im Takt des PlcProg aktualisiert.
- **GlobalKinematicsDefines**:

  Wird nur benötigt, wenn MOTIF_CONFIG.CONFIG_MODE auf GLOB_VAR eingestellt ist.

  Hier werden die Kinematiken als Konstanten vom Typ MB_AXESGROUPIF_REF definiert und in einer Liste an TE_KinInterfaceMainProg() übergeben. Die Konstanten müssen für das eigene Projekt entsprechend angepasst werden.
- **DemoKinematicsCommands**:

  Beispielcode mit einer Ablaufprogrammierung und dem Absetzen von Kinematikkommandos. Dieser Code muss für das eigene Projekt entsprechend angepasst werden.
- **OverviewKinematics**:

  Visualisierung zur Bedienung des Kinematik-Interface während der Inbetriebnahmephase. Durch Klicken auf Felder mit "<<" kann in weitere Bilder abgetaucht werden.
- **Version_AxisKinInterface**:

  Änderungshistorie und Disclaimer

1.4.6.3 Bibliothek CXA_MotionInterfaceUser.library"

Diese offene Bibliothek dient dazu die Funktionsbausteine und Strukturen der Basisbibliothek **CXA_MotionInterface** zu erweitern. Programme und Visualisierungen werden hier zur Verfügung gestellt. Hier sind auch die globalen Variablen der Interfaces instanziiert. Mit dieser Bibliothek sind Anpassungen / Erweiterungen der Interfaces durch den Anwender möglich.

Wie man die Anpassungen ausführen kann, ist hier ↗ *Kapitel 1.4.7 „Kinematik-Interface Anwender-Erweiterung “ auf Seite 75* beschrieben

In diesem Kapitel werden die POUs des KinematikInterface aus dem Ordner "KinInterfaceUser/POUs" beschrieben.

TE_KinInterfaceMainProg

Das Programm **TE_KinInterfaceMainProg** deckt die folgenden Punkte ab:

- **Initialisierung des Kinematik-Interface**:
  Bei Erreichen des Modus "Running" wird das Kinematik-Interface mit Hilfe des Funktionsbausteins **TE_KinematicsInitAllKinematics** initialisiert. Bei erfolgreicher Initialisierung wird der Ausgang "InitDone" gesetzt, bei Fehlern der Ausgang "Error".
- **Zyklischer Aufruf des Kinematik-Interface**:
  Nach erfolgreicher Initialisierung wird der Funktionsbaustein (FB) **TE_KinematicsInterface** zyklisch aufgerufen.
- **Methode TE_KinInterfaceMainProg.mMotionTask()**:
  Wird die Methode aus einer schnelleren MotionTask aufgerufen, werden die Elemente in "arKinStatus_gb[].Data" im schnelleren Takt aktualisiert. Der Aufruf des FB **TE_KinematicsInterface** kann mit Hilfe der Steuer-Variable "arKinCtrl_gb[].Admin.Config.MotionSync" in die schnellere Task verschoben werden.
  Wird die Methode nicht aufgerufen, erfolgen alle Aktualisierungen im Takt der PLC-Task.

Tab. 52: Schnittstellenvariablen TE_KinInterfaceMainProg

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	ClearError	BOOL	Fehler löschen wird durch eine positive Flanke an "ClearError" gestartet
	KinCfgIdx	POINTER TO ARRAY [] OF MB_AXES-GROUPIF_REF	Konfigurationsliste für die Indizes der Kinematiken (nur für Konfigurationsmodus "GLOB_VAR")
VAR_OUTPUT	InitDone	BOOL	Wird gesetzt, wenn das Programm die Initialisierung erfolgreich beendet hat
	Error	BOOL	Zeigt an, dass ein Fehler im Programm aufgetreten ist
	ErrorID	ERROR_CODE	Kurzer Hinweis zur Fehlerursache

I/O-Typ	Name	Datentyp	Kommentar
	ErrorIdent	ERROR_STRUCT	Detaillierte Information zum Fehler

Fehlerbehandlung: die Fehlercodes des intern benutzten Funktionsbausteines TE_KinematicsInitAllKinematics werden durchgereicht. Das Programm kann die folgenden Fehlercodes erzeugen:

Tab. 53: Fehlercodes des Programmes TE_KinInterfaceMainProg

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#0A0F0107	16#0C2301D0	Fehler im Ablauf des Programmes

Das Programm TE_KinInterfaceMainProg ist zum Integrieren des Kinematik-Interface in ein bestehendes Programm nützlich.

*Siehe auch **Example_KinIfApplicationPart** im Ordner "KinInterfaceUser/_Examples"*

TE_KinematicsInitAllKinematics

Der Funktionsbaustein **TE_KinematicsInitAllKinematics** initialisiert die Kinematik-Interface Strukturen.

Die Initialisierung kann gesteuert werden mit Hilfe der Parameterliste "MOTIF_CONFIG".

Es gibt die folgenden Möglichkeiten:

- AUTO = MOTIF_CONFIG.CONFIG_MODE: Es wird der Datalayer Knoten "motion/kin/" ausgelesen und die Kinematiken in der dort gefundenen Reihenfolge in die Kinematik-Interface Strukturen eingeordnet.
- GLOB_VAR = MOTIF_CONFIG.CONFIG_MODE: Die Kinematiken werden anhand des globalen Arrays "KINIF_CONFIG_INDEXES" in die Kinematik-Interface Strukturen eingeordnet.

Tab. 54: Schnittstellenvariablen TE_KinematicsInitAllKinematics

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT	Execute	BOOL	Die Initialisierung wird durch eine positive Flanke an "Execute" gestartet
	KinCfgIdx	POINTER TO ARRAY [] OF MB_AXES-GROUPIF_REF	Konfigurationsliste für die Indizes der Kinematiken (nur für Konfigurationsmodus "GLOB_VAR")
VAR_OUTPUT	Done	BOOL	Wird gesetzt, wenn der FB die Bearbeitung beendet hat
	Active	BOOL	Wird gesetzt, wenn der FB aktiv ist (nicht im Leerlaufbetrieb)
	Error	BOOL	Zeigt an, dass ein Fehler im Programm aufgetreten ist

I/O-Typ	Name	Datentyp	Kommentar
	ErrorID	ERROR_CODE	Kurzer Hinweis zur Fehlerursache
	ErrorIdent	ERROR_STRUCT	Detaillierte Information zum Fehler

Fehlerbehandlung: die Fehlercodes des intern benutzten Funktionsbausteines MB_KinematicsInit werden durchgereicht. Der Funktionsbaustein kann die folgenden Fehlercodes erzeugen:

Tab. 55: Fehlercodes des Funktionsbausteines TE_KinematicsInitAllKinematics

ErrorID	Additional1	Additional2	Beschreibung
INPUT_RANGE_ERROR	16#0A0F0107	16#0C2301C0	Eingang KinIndex ausserhalb des gültigen Bereiches [MB_KINIF_MIN_KIN_INDEX..MOTIF_CONFIG.MAX_KIN_INDEX]
INPUT_RANGE_ERROR	16#0A0F0107	16#0C2301C1	Unbekannter Konfigurationsmodus (MOTIF_CONFIG.CFG_MODE_KIN)
INPUT_RANGE_ERROR	16#0A0F0107	16#0C2301C2	Pointer KinCfgIdx zur globalen Variablen ist 0 (MOTIF_CONFIG.CFG_MODE_KIN = GLOB_VAR)
INPUT_RANGE_ERROR	16#0A0F0107	16#0C2301C3	Kinematikindex ausserhalb des gültigen Bereiches (MOTIF_CONFIG.CFG_MODE_KIN = GLOB_VAR)
INPUT_RANGE_ERROR	16#0A0F0107	16#0C2301C4	Kinematikindex doppelt (MOTIF_CONFIG.CFG_MODE_KIN = GLOB_VAR)
INPUT_RANGE_ERROR	16#0A0F0107	16#0C2301C5	Kinematikname doppelt (MOTIF_CONFIG.CFG_MODE_KIN = GLOB_VAR)
STATE_MACHINE_ERROR	16#0A0F0107	16#0C2301C8	Fehler im Ablauf des Funktionsbausteines

*Das Programm TE_KinInterfaceMainProg ruft diesen Funktionsbaustein **TE_KinematicsInitAllKinematics** bereits auf.*

TE_KinematicsInterface

Der Funktionsbaustein **TE_KinematicsInterface** erweitert den **MB_KinInterfaceBase** und bearbeitet im zyklischen Betrieb die Kinematik-Interface Strukturen.

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

Tab. 56: Schnittstellenvariablen TE_KinematicsInterface

I/O-Typ	Name	Datentyp	Kommentar
VAR_INPUT Steuer-Struktur	AdminCtrlExt	REFERENCE TO TE_KINEMATICS_ADMINISTRATION	Referenz zur Steuer-Struktur Admin
	CoordCtrlExt	REFERENCE TO TE_KINEMATICS_COORDINATED	Referenz zur Steuer-Struktur Coordinated
	SetupMode	REFERENCE TO TE_KINEMATICS_SETUP_MODE	Referenz zur Steuer-Struktur SetupMode
VAR_INPUT Status-Struktur	AdminStatusExt	REFERENCE TO TE_KINEMATICS_ADMIN_STATUS	Referenz zur Status-Struktur Admin
	DataStatusExt	REFERENCE TO TE_KINEMATICS_DATA	Referenz zur Status-Struktur Data
	DiagStatusExt	REFERENCE TO TE_KINEMATICS_DIAGNOSIS	Referenz zur Status-Struktur Diag
	SetupModeAck	REFERENCE TO TE_KINEMATICS_SETUP_MODE_STATUS	Referenz zur Status-Struktur SetupMode

Fehlerbehandlung: die Fehlercodes der intern benutzten Funktionsbausteine werden durchgereicht. Der Funktionsbaustein kann die folgenden Fehlercodes erzeugen:

Tab. 57: Fehlercodes des Funktionsbausteines TE_KinematicsInterface

ErrorID	Additional1	Additional2	Beschreibung
STATE_MACHINE_ERROR	16#0A0F0107	16#0C2301E0	Fehler im Ablauf des Funktionsbausteines

*Das Programm TE_KinInterfaceMainProg ruft diesen Funktionsbaustein **TE_KinematicsInterface** bereits auf.*

1.4.6.4 Kinematik-Interface Visualisierungen

1.4.6.4.1 Überblick

Zum Kinematik-Interface werden Visualisierungsmasken mitgeliefert, um ein vorgefertigtes und einfaches Interface zum Einstellen und Ansteuern der Kinematiken zur Verfügung zu stellen.

Folgende Visualisierungen sind Programmiertemplate "ctrlX CORE Axis/Kin-Interface" und in der Bibliothek **CXA_MotionInterfaceUser.library** enthalten:

Beispielprojekt Visualisierungen

Tab. 58: Beispielprojekt Visualisierungen

Visualisierung	Beschreibung
OverviewKinematics (im Template-Project)	Gesamtüberblick über alle definiertenKinematiken, einschließlich einfacher Diagnosen und Statusinformationen
Kinematics_Overview	Zeigt aktuelles Kommando und aktuelle Werte für Koordinaten und Geschwindigkeit, zusammen mit Navigation zur Positions- und Setup-Anzeige für die aktuelle Kinematik. Abschalten der Achsen ist ebenfalls möglich
KinPosition_mode	Überwachen des Positionierbetriebs
KinSetup_mode	Überwachen des Tippbetriebs

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

- arKinCtrl_gb[ ]
- arKinStatus_gb[ ]
- VisuKinematicsNo

1.4.6.4.2 Systemübersicht-Visualisierung

Die OverviewKinematics Visualisierung erlaubt es dem Anwender jede, im Projekt konfigurierte Kinematik, schnell anzukoppeln. Zusätzliche Kinematiken können im Offline-Betrieb zur Anzeige hinzugefügt werden.

Diese Anzeige liefert einen Gesamt-Systemstatus und ermöglicht das Löschen von Fehlern. Einzelne Kinematik-Bedienelemente liefern den Kinematiknamen, Diagnosen, aktuelle Koordinaten und Geschwindigkeit zusammen mit den Betriebsarten-Statusanzeigen.

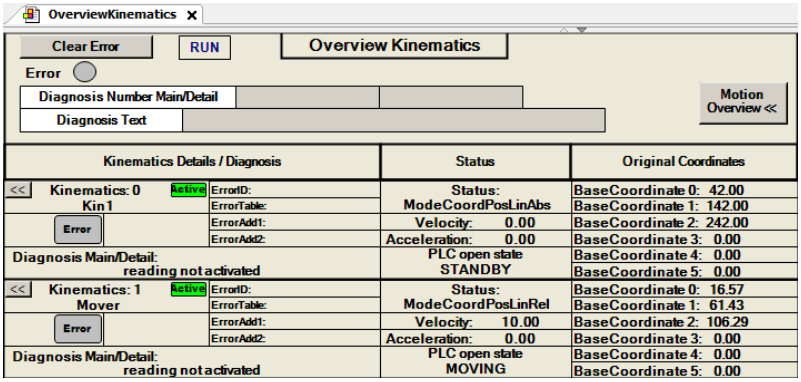


Abb. 14: OverviewKinematics

Hinzufügen einer Kinematik zur Systemübersicht

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

1. Mit ctrlX PLC Engineering im Offline-Betrieb Doppelklick auf die "OverviewAllKinematics" Visualisierung.
2. Wählen Sie im Fenster Visualisierungswerkzeuge die Schaltfläche *„Frame“* an und erzeugen Sie einen Grundriss unterhalb der letzten Kinematiktabellenzeile, der der Zeilenhöhe und Zeilenbreite entspricht.
3. Mit rechter Maustaste auf den Frame klicken und "Frameauswahl" aktivieren. Wählen Sie das **OverviewOneKinematics**-Element aus dem Visualisierungsauswahlfenster im Ordner **CXA_MotionInterfaceUser/KinInterfaceUser/Visualizations/SystemOverview**aus.

   ➥ Eine komplett neue Systemübersicht Kinematikzeile erscheint.
4. Nach einem Klick in der neuen Visualisierung tragen Sie im "Eigenschaften"-Fenster als Wert für **m_Input_KinIndex** den Index der definierten Kinematik ein.
5. Das neue Kinematik-Interface für die Systemübersicht kann nun in der Grösse angepasst und positioniert werden.
6. Übersetzen Sie das SPS-Projekt neu und gehen Sie Online.

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

Systemübersicht Navigation

Eine Einzelkinematik-Übersichtanzeige ist durch Klicken auf die Schaltfläche mit zwei Pfeilen "<<", die sich unter der Details-Spalte in der Kinematiktabelle befindet, erreichbar. Aus der Kinematikübersichtanzeige ist die Navigation zur Positions- und Setup-Betriebsartenanzeige möglich.

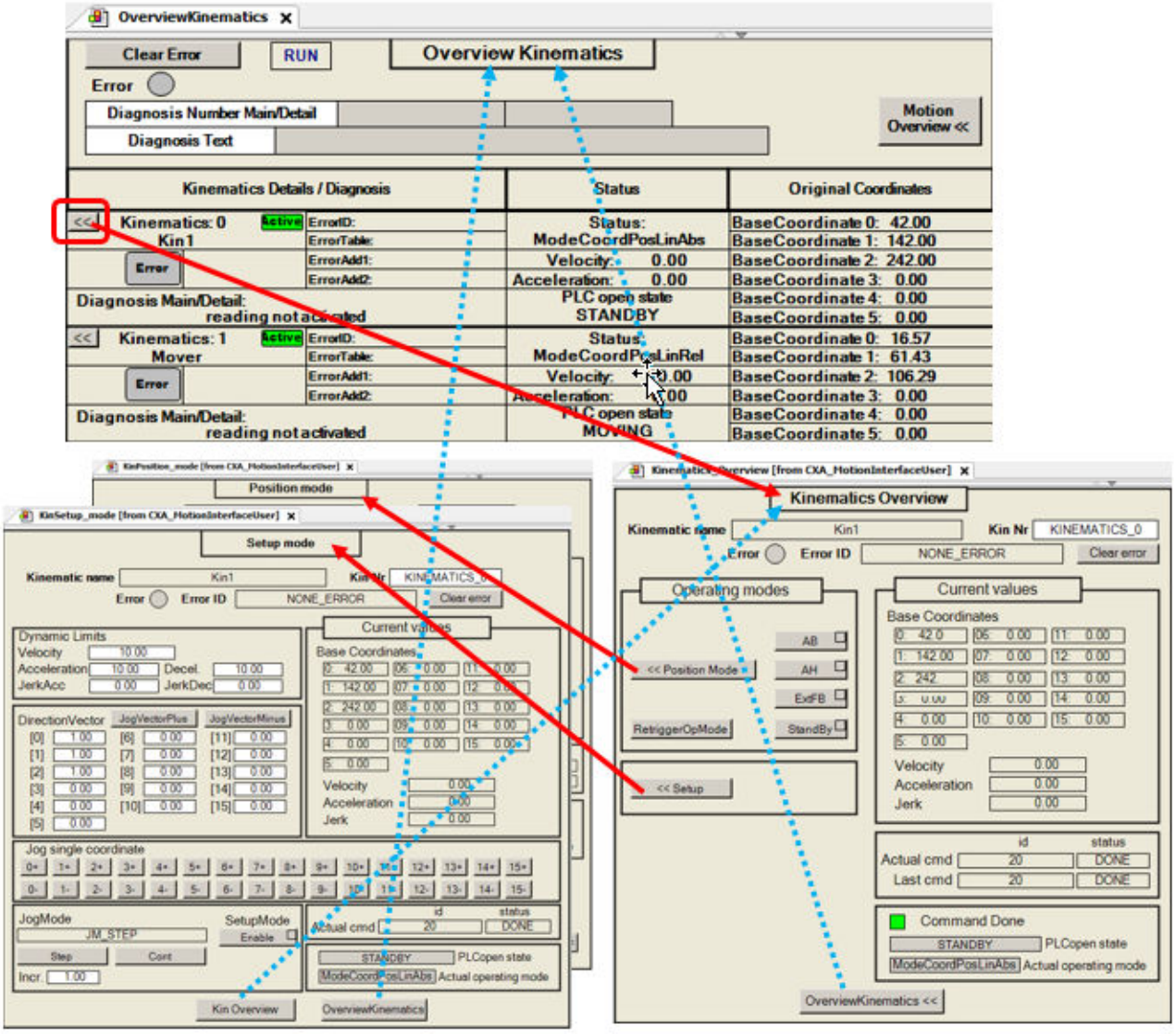


Abb. 15: Systemübersicht Navigation

1.4.7 Kinematik-Interface Anwender-Erweiterung

1.4.7.1 Überblick

Die **arKinCtrl_gb[ ]** und **arKinStatus_gb[ ]** Strukturen können durch den Anwender erweitert werden, um das Kinematik-Interface an spezielle Applikationen anzupassen.

*Die **arKinCtrl_gb[ ]**- und **arKinStatus_gb[ ]**-Strukturen des Anwender-Interface sind als Basistypen mit dem Präfix "MB_" in der geschlossenen Bibliothek **CXA_MotionInterface.compiled-library** definiert und daher für den Anwender nicht zugänglich.*

*Um Erweiterungen zu ermöglichen, ist es nötig, eigene Strukturen zu definieren, die die Strukturen "MB_" erweitern. Die erweiterteten Strukturen sind durch den Präfix "TE_" gekennzeichnet und befinden sich in der offenen Bibliothek **CXA_MotionInterfaceUser.library**.*

Empfohlene Vorgehensweise

Um die Anwendererweiterungen auszuführen, ist es notwendig die Bibliothek **CXA_MotionInterfaceUser.library** anzupassen. Zur Nachvollziehbarkeit ist es notwendig und dringend empfohlen der angepassten Bibliothek einen neuen Namen zu geben, z.B. CXA_MotionInterface**MyCompany**.library. Im Folgenden wird als Bibliotheksname **CXA_MotionInterfaceMyCompany.library** verwendet.

Arbeitsablauf

1. Im Bibliotheksverwalter die **CXA_MotionInterfaceUser.library** selektieren. Rechte Maustaste -> "Bibliothek exportieren" anwählen. Speicherort wählen und einen neuen Namen z.B. CXA_MotionInterface**MyCompany**.library vergeben. Nachträgliches Umbenennen ist ebenfalls möglich.
2. Mit einer zweiten Instanz von ctrlX PLC Engineering die Bibliothek **CXA_MotionInterfaceMyCompany.library** öffnen. In den Projektinformationen das Feld "freigegeben" abwählen, die weiteren Felder anpassen und in den Eigenschaften den Schlüssel "Placeholder" löschen.
3. Im Anwendungsprogramm (erste Instanz von ctrlX PLC Engineering) im Bibliotheksverwalter die **CXA_MotionInterfaceUser.library** entfernen und dafür **CXA_MotionInterfaceMyCompany.library** einbinden.
4. Anpassungen in der Bibliothek vornehmen. Am Ende ausführen: "Datei"->"Projekt speichern und ins Bibiliotheksrepository installieren".
5. Im Anwendungsprogramm (erste Instanz von ctrlX PLC Engineering) die Anpassungen testen. Debuggen im Code aus der Bibliothek ist auch möglich.
6. Schritte 4. und 5. wiederholen bis die Funktion fehlerfrei ist.

*Sobald ein Update der **CXA_MotionInterfaceUser.library** zur Verfügung steht, können Änderungen mit "Projekt"->"Vergleichen" in die **CXA_MotionInterfaceMyCompany.library** übernommen werden.*

*Die **CXA_MotionInterfaceUser.library** verwendet Fehlercodes mit "CXA_TABLE". Diese sind in der Produktdokumentation zu finden. Wenn in den Anwendererweiterungen weitere Fehlercodes benötigt werden, können diese frei definiert werden, müssen aber mit "USER1_TABLE..USER10_TABLE" gemeldet werden.*

ML_GetOverride

Dieser Abschnitt zeigt, wie das Kinematik-Interface durch Hinzufügen der **ML_GetOverride**-Funktionalität erweitert wird. Die Funktion ML_GetOverride ermöglicht es, den aktuellen VelocityOverride einer Kinematik abzufragen. Der Sollwert ist bereits in der Basis-Struktur als "arKinCtrl_gb[].CoordMode.VelocityOverride" vorhanden.

Die folgenden neuen Ein- und Ausgänge werden definiert:

- arKinCtrl_gb[ ].Admin.EnableReadVelocityOverride
- arKinStatus_gb[ ].Data.ActVelocityOverride

Im Programm-Template "ctrlX CORE Axis/Kin-Interface" sind folgende anwenderspezifische Erweiterungen implementiert:

- Jog-Funktionalität als "SetupMode"

Im Folgenden wird davon ausgegangen, dass die im Programm-Template "ctrlX CORE Axis/Kin-Interface" bereits vorbereitete Struktur verwendet wird. Es wird nur beschrieben, welche Änderungen in den dort vorgegebenen POUs notwendig sind.

Hinweise zur Implementation von Anwendererweiterungen

Die Anwendererweiterungen werden mit Hilfe der objektorientierten Erweiterungen von ctrlX PLC Engineering implementiert. Dabei sind einige Besonderheiten zu beachten:

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

  Die Eingänge AdminCtrl (ohne Ext) usw. gehören zum Basis-FB und sollten nicht genutzt werden
- Die Eingänge des FB TE_KinematicsInterfaceBase sind als "REFERENCE TO" definiert. In der Methode mInitExtension werden die Referenzen einmalig initialisiert und müssen dann beim zyklischen Aufruf des FB nicht mehr übergeben werden
- Wenn die Basisstrukturen TE_KINEMATICS_CONTROL_TYPE01 und TE_KINEMATICS_STATUS_TYPE01 mit zusätzlichen Unterstrukturen erweitert werden sollen, sind folgende zusätzliche Schritte zu der weiter unten beschriebenen Vorgehensweise notwendig (siehe SetupMode und SetupModeAck als Beispiele):
  - Neue Elemente in TE_KINEMATICS_CONTROL_TYPE01 bzw. TE_KINEMATICS_STATUS_TYPE01 eintragen
  - Am FB TE_KinematicsInterfaceBase die zusätzlichen Eingänge als REFERENCE TO hinzufügen
  - In der Methode mInitExtension des FB TE_KinematicsInterfaceBase die Referenzen initialisieren

1.4.7.2 Erweitern der arKinCtrl_gb[ ]-Struktur

Um die zusätzliche Funktionalität der arKinCtrl_gb[] Struktur hinzuzufügen, muss der Anwender eine neue Struktur anlegen, die Unterstrukturen von den bereits existierenden Struktur ableiten und dann die neuen Elemente hinzufügen. Im Programm-Template "ctrlX CORE Axis/Kin-Interface" ist eine Struktur TE_KINEMATICS_CONTROL_TYPE01 und die Unterstrukturen TE_KINEMATICS_ADMINISTRATION usw. bereits vorbereitet.

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

1. Mit ctrlX PLC Engineering die Bibliothek **CXA_MotionInterfaceMyCompany.library** öffnen
2. Öffnen Sie die Struktur TE_KINEMATICS_ADMINISTRATION, Ordner KinInterfaceUser/DUTs/Control.
3. Deklarieren Sie die folgende Variable
   - `EnableReadVelocityOverride: BOOL:=TRUE;`

1.4.7.3 Erweitern der arKinStatus_gb[ ] Struktur

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

1. Öffnen Sie die Struktur TE_KINEMATICS_DATA, Ordner KinInterfaceUser/DUTs/Status.
2. Deklarieren Sie die folgende Variable
   - `ActVelocityOverride: LREAL;`

1.4.7.4 Erweitern des Funktionsbausteines

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

1. Deklarieren Sie im FB TE_KinematicsInterface die folgende Variable:

   ```
   stML_GetOverrideData: ML_GetOverrideData;
   ```

2. Im FB TE_KinematicsInterface könnte die Funktion so ausprogrammiert werden:

   ```
   IF AdminCtrlExt.EnableReadVelocityOverride =
   TRUE THEN

   stML_GetOverrideData.In.ObjName := AdminCtr-
   lExt.Config.Group.KinName;

   ML_GetOverride(stML_GetOverrideData); // call
   motion function

   DataStatusExt.ActVelocityOverride :=
   stML_GetOverrideData.Out.Value;

   END_IF
   ```

3. Übersetzen Sie das Projekt neu und überprüfen Sie es auf Programmierfehler.
4. Laden Sie das Projekt in die Steuerung.

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

Watch 1

Expression	Type	Value
Global_Kinematics_Interface.arKinCtrl_gb[0]	TE_KINEMATICS_CONTROL_TYPE01	
Admin	TE_KINEMATICS_ADMINISTRATION	
Config	MB_KINEMATICS_ADMIN_CONFIG	
ClearError	BOOL	FALSE
RetriggerOpMode	BOOL	FALSE
_OpMode	MB_KINEMATICS_MODE	ModeCoordAB
_OpModeBits	TE_KINEMATICS_MODE_BITS	
EnableReadVelocityOverride	BOOL	TRUE
CoordMode	TE_KINEMATICS_COORDINATED	
SetupMode	TE_KINEMATICS_SETUP	
Global_Kinematics_Interface.arKinStatus_gb[0]	TE_KINEMATICS_STATUS_TYPE01	
Admin	TE_KINEMATICS_ADMIN_STATUS	
Diag	TE_KINEMATICS_DIAGNOSIS	
Data	TE_KINEMATICS_DATA	
Acceleration	LREAL	0
Disabled	BOOL	TRUE
ErrorStop	BOOL	FALSE
Jerk	LREAL	0
Moving	BOOL	FALSE
PLCopenState	STRING(50)	'DISABLED'
PositionBaseCoord	REFERENCE TO ARRAY [0..(MB_KINI...	
Standby	BOOL	FALSE
Stopping	BOOL	FALSE
Velocity	LREAL	0
ActVelocityOverride	LREAL	0.5
SetupMode	TE_KINEMATICS_SETUP_STATUS	

Abb. 16: arKinCtrl_gb und arKinStatus_gb Strukturen mit Anwender-Erweiterungen

1.4.8 HowTo: Typische Anwenderaktivitäten

1.4.8.1 Zugriff auf Kinematikdaten

Die folgenden Daten sind verfügbar (Name = jeweiliger Kinematikname):

- arKinCtrl_gb[Name.GroupNo] => Steuerstruktur des Kinematik-Interface
- arKinStatus_gb[Name.GroupNo] => Statusstruktur des Kinematik-Interface
- arKinStatus_gb[Name.GroupNo].Data => Istwerte und Statusinformationen
-

Azyklische Zugriffe auf Kinematikdaten sind über den ctrlX Data-Layer mit den Funktionsbausteinen DL_ReadNode und DL_Write-Node möglich.

1.4.8.2 Anpassung der maximalen Kinematikanzahl

Die Kinematikstrukturen können an die tatsächlich vorhandene Kinematikanzahl angepasst werden.

Die Bibliothek **CXA_MotionInterfaceUser** erlaubt Anpassungen über die Bibliotheksparameter "MOTIF_CONFIG".

Mit der Konstanten "MAX_KIN_INDEX" kann die Grösse der Strukturen passend zur Anwendung gewählt werden.

Beim Kinematik-Interface ist die untere Array-Grenze fest auf die Konstante "MB_KINIF_MIN_KIN_INDEX" mit dem Wert Null festgelegt.

1.4.8.3 Anpassung der Zuordnung Kinematikname<>KinematikIndex

Das KinematikInterface arbeitet mit einem KinematikIndex zur Adressierung in den Kinematikstrukturen. Die Motion-Firmware arbeitet mit dem Kinematiknamen. Die Zuordnung Kinematikname<>KinematikIndex kann mit verschiedenen Methoden erfolgen.

Die Bibliothek **CXA_MotionInterfaceUser** erlaubt eine Auswahl der Methode über die Bibliotheksparameter "MOTIF_CONFIG".

Zuordnung Kinematikname<>KinematikIndex in MOTIF_CONFIG.CFG_MODE_KIN

- AUTO: Auslesen des DataLayer Knotens "motion/kin" und Zuweisung des KinematikIndex in der hier vorgefundenen Reihenfolge
- GLOB_VAR: in der Application wird eine Liste von "MB_AXES-GROUPIF_REF" definiert und an das Programm "TE_KinInterfaceMainProg" übergeben. Siehe "GlobalKinematicsDefines" im Programm-Template "ctrlX CORE Axis/Kin-Interface".

1.4.8.4 Kinematik hinzufügen

Eine Kinematik kann in der Bedienoberfläche im Bereich Motion angelegt werden oder auch z.B. aus dem SPS-Programm erzeugt werden.

Für eine hinzugefügte Kinematik muss ggf. eine Initialisierung bestimmter Strukturelemente des Kinematik-Interface vorgenommen werden. Dies geschieht bei Verwendung des Programm-Template "ctrlX CORE Axis/Kin-Interface" automatisch beim Erreichen des Zustandes "Running". Die Zuordnung Kinematikname<>KinematikIndex muss ergänzt werden, wenn MOTIF_CONFIG.CFG_MODE_KIN = GLOB_VAR konfiguriert ist.
Siehe auch oben ↗ *weitere Informationen auf Seite 80*.

1.4.8.5 Kinematik entfernen/umbenennen

Eine vorhandene Kinematik kann in der Bedienoberfläche im Bereich Motion oder über diverse Schnittstellen gelöscht bzw. umbenannt werden.

Wird eine Kinematik umbenannt, muss der Zugriff über die Control- u. Statusstrukturen "arKinCtrl_gb[geänderterKinematikname.GroupNo]" und "arKinStatus_gb[geänderterKinematikname.GroupNo]" innerhalb des SPS-Programmes angepasst werden.

Die Zuordnung Kinematikname<>KinematikIndex muss angepasst werden, wenn MOTIF_CONFIG.CFG_MODE_KIN = GLOB_VAR konfiguriert ist. Siehe auch oben ➚ *weitere Informationen auf Seite 80*.

1.4.8.6 Kinematik-Interface Erweiterungen

Das Kinematik-Interface erlaubt fast beliebige Erweiterungen der Kinematik-Interface-Strukturen. Es können zusätzliche Unterstrukturen eingefügt und auch die vorhandenen Unterstrukturen erweitert werden.

Die folgenden Strukturelemente sind als Anwendererweiterungen in der KinCtrl-Struktur beispielhaft programmiert:

- SetupMode: zusätzliche Unterstruktur in der KinCtrl-Struktur
  - Enable: Freigabe Einrichtbetrieb
  - JogMode: Einstellung inkrementell oder kontinuierlich tippen
  - JogPlus[ ]: Tippen +, für jede Koordinate einzeln wählbar
  - JogMinus[ ]: Tippen -, für jede Koordinate einzeln wählbar
  - Increment: Schrittweite bei inkrementellem Tippen
  - DirectionVector[ ]: Bewegungsrichtung beim Vertippen auf der Bahn
  - JogVectorPlus: Tippen +, für die gesamte Kinematik
  - JogVectorMinus: Tippen -, für die gesamte Kinematik
  - DynValues: Tipp(brems)beschleunigung und Ruck.

Die folgenden Strukturelemente sind als Anwendererweiterungen in der KinStatus-Struktur beispielhaft programmiert:

- SetupMode: zusätzliche Unterstruktur in der KinStatus-Struktur
  - EnableAck: Einrichtbetrieb ist aktiv

Der Code zu diesen Erweiterungen ist in den Aktionen des Bausteins TE_KinematicsInterface() im Ordner "KinInterfaceUser/POUS" zu finden. Die dazugehörigen Strukturen sind in den Ordnern "KinInterfaceUser/DUTs/Control" und "KinInterfaceUser/DUTs/Status" zu finden.

Es können eigene Erweiterungen hinzugefügt werden (siehe dazu ↗ , *Seite*).

1.5 IMC-Interface

1.5.1 Einführung und Übersicht

Das IMC-Interface (**I**nterface-**M**otion-**C**ontrol) enthält in der Kontrollstruktur Steuersignale für die App **rexroth-motion** und in der Statusstruktur werden Statusbits und Diagnoseinformationen der App **rexroth-motion** zur Verfügung gestellt.

Funktionen des IMC-Interfaces

- Anwahl des Modus "Configuration" oder "Running"
- führt den Befehl "Clear Error" für die App **rexroth-motion** aus
- führt "RetriggerOpMode" aus, um es den Benutzern zu ermöglichen, den gewählten Betriebsmodus erneut auszuführen
- Bei einer Modusumschaltung über eine externe Instanz (z.B. Web-Interface) setzt das Imc-Interface den Ausgang "PassiveMode" und der angewählte Modus wird nicht angesteuert.

  Mit "RetriggerOpMode" oder durch Zurückschalten in den angewählten Modus wird der "PassiveMode" wieder verlassen

Die folgende Abbildung zeigt das Benutzerinterface mit den Datenstrukturen des IMC-Interface:

Watch 1

Expression	Type	Value	Prepared value
Global_ImcInterface	GLOBAL_IMCINTERFACE		
ImcCtrl	MB_IMC_CONTROL_TYPE01		
Admin	MB_IMC_ADMINISTRATION		
_OpMode	MB_IMC_MODE_CONTROL	IMC_MODE_CTRL_CONF	IMC_MODE_CTRL_RUN
RetriggerOpMode	BOOL	FALSE	
ClearError	BOOL	FALSE	
ImcStatus	MB_IMC_STATUS_TYPE01		
InOperation	BOOL	TRUE	
Admin	MB_IMC_ADMIN_STATUS		
_OpModeAck	MB_IMC_MODE_STATUS	IMC_MODE_STATUS_RUN	
ModeStatus_Boot	BOOL	FALSE	
ModeStatus_Config	BOOL	FALSE	
ModeStatus_Error	BOOL	FALSE	
ModeStatus_Run	BOOL	TRUE	
Diag	MB_IMC_DIAGNOSIS		
Error	BOOL	TRUE	
ErrorID	ERROR_CODE	ACCESS_ERROR	
ErrorIdent	ERROR_STRUCT		
Table	ERROR_TABLE	CXA_TABLE	
Additional1	DWORD	16#80000001	
Additional2	DWORD	16#00000000	
ClearErrorAck	BOOL	FALSE	
NumberMain	DWORD	16#090F2040	
NumberDetail	DWORD	16#0C560272	
Message	STRING	'vel: 10.000'	
ObjectName	STRING	'MoverX'	
PassiveMode	BOOL	TRUE	

Abb. 17: Datenstrukturen IMC-Interface

Weitere Informationen zu den Datenstrukturen siehe Online-Dokumentation in der Bibliothek **CXA_MotionInterface** im Ordner "ImcInterface/DUTs".

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

*In der Bibliothek **CXA_MotionInterface** ist im Ordner "ImcInterface/_Examples" ein Anwendungsbeispiel zum Imc-Interface enthalten.*

Dieses Beispiel zeigt, wie ImcStatus und ImcCtrl beim Erstellen von Achsen verwendet werden können.