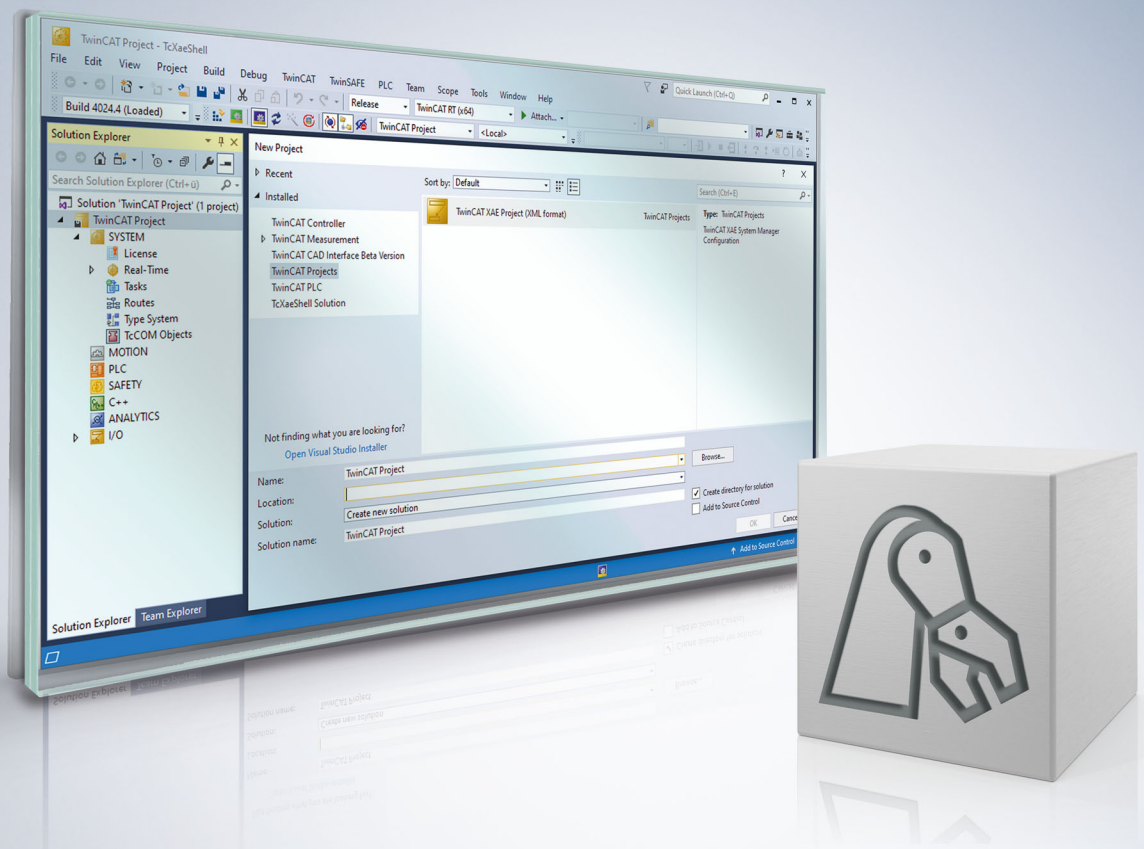
BECKHOFF New Automation Technology

Handbuch | DE

TF5120

TwinCAT 3 | Robotics mxAutomation



06.04.2023 | Version: 2.0.0

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, für jede Installation und Inbetriebnahme die zu dem betreffenden Zeitpunkt veröffentliche Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiter entwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT®

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

1.2 Sicherheitshinweise

Sicherheitsbestimmungen

Beachten Sie die folgenden Sicherheitshinweise und Erklärungen!
Produktspezifische Sicherheitshinweise finden Sie auf den folgenden Seiten oder in den Bereichen Montage, Verdrahtung, Inbetriebnahme usw.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Erklärung der Symbole

In der vorliegenden Dokumentation werden die folgenden Symbole mit einem nebenstehenden Sicherheitshinweis oder Hinweistext verwendet. Die Sicherheitshinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

⚠ GEFAHR

Akute Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!

⚠ WARNUNG

Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht Gefahr für Leben und Gesundheit von Personen!

⚠ VORSICHT

Schädigung von Personen!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen geschädigt werden!

HINWEIS

Schädigung von Umwelt oder Geräten

Wenn der Hinweis neben diesem Symbol nicht beachtet wird, können Umwelt oder Geräte geschädigt werden.

Tipp oder Fingerzeig

Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 Einleitung

2.1 Zielgruppe

Diese Dokumentation richtet sich an Benutzer mit folgenden Kenntnissen:

- Systemkenntnisse der Robotersteuerung
- Fortgeschrittene SPS-Programmierkenntnisse
- Fortgeschrittene Kenntnisse über Feldbus-Anbindungen

Für den optimalen Einsatz von KUKA-Produkten empfiehlt KUKA eine Schulung im KUKA College. Informationen zum Schulungsprogramm sind unter www.kuka.com oder direkt bei den KUKA-Niederlassungen zu finden.

2.2 Dokumentation des Industrieroboters

Ist der KUKA Dokumentation zu entnehmen.

2.3 Verwendete Begriffe

Begriff	Beschreibung
Achsgruppe	Gruppe von Roboterachsen und Zusatzachsen, die zusammen ein Robotersystem ergeben (z. B. ein Roboter mit 6 Achsen und 1 Zusatzachse)
FIFO	First In First Out Verfahren zum Abarbeiten eines Datenspeichers Elemente, die zuerst gespeichert wurden, werden zuerst wieder aus dem Speicher entnommen.
KR C	KUKA Robot Control Robotersteuerung
KRL	KUKA Robot Language KUKA Roboter Programmiersprache
KUKA smartHMI	Siehe "smartHMI"
KUKA smartPAD	Siehe "smartPAD"
mxA-Schnittstelle	Optionspaket KUKA.PLC mxAutomation auf der Robotersteuerung mit einer Kommunikationsschnittstelle zur SPS
NULLFRAME	Kartesisches Koordinatensystem, bei dem alle Koordinaten den Wert Null haben
EtherCAT	Ethernet-basierter Feldbus (Ethernet-Schnittstelle)
Roboter-Interpreter	Synchron arbeitender Prozess, in dem das aktuelle Roboterprogramm abgearbeitet wird
SAK-Fahrt	Der Roboter wird auf die Koordinaten des Bewegungssatzes verfahren, an der sich der Satzzeiger befindet. Auf diese Weise wird die Roboterstellung mit den Koordinaten des aktuellen Punkts in Übereinstimmung gebracht.
smartHMI	smart Human-Machine Interface Bedienoberfläche auf dem smartPAD
smartPAD	Programmierhandgerät für die Robotersteuerung Das smartPAD hat alle Bedien- und Anzeigemöglichkeiten, die für die Bedienung und Programmierung des Industrieroboters benötigt werden. Es existieren 2 Modelle: • smartPAD • smartPAD-2

Begriff	Beschreibung
	Zu jedem Modell existieren wiederum Varianten, z. B mit unterschiedlichen Längen der Anschlusskabel. Die Bezeichnung "KUKA smartPAD" oder "smartPAD" bezieht sich auf beide Modelle, sofern diese nicht explizit unterschieden werden.
SPS (PLC)	Speicherprogrammierbare Steuerung (Programmable Logic Controller) Wird in Anlagen als übergeordnetes Master-Modul im Bussystem eingesetzt.
Submit-Interpreter	Zyklisch arbeitendes Logikprogramm, das parallel zum Bewegungsprogramm auf der Robotersteuerung läuft
WorkVisual	Engineering-Umgebung für KR C-gesteuerte Roboterzellen

3 Produktbeschreibung

3.1 Übersicht

Die TwinCAT Bibliothek enthält Funktionsbausteine zur Programmierung von Automatisierungsaufgaben mit TwinCAT 3.

Kommunikation

Für den Datenaustausch zwischen SPS und Robotersteuerung ist die Klemme EL6695-1001 von KUKA vorgesehen.

WorkVisual

Zur Konfiguration der Feldbusse und zur Verschaltung der Feldbussignale wird folgende Software benötigt:

- WorkVisual

3.2 Bestimmungsgemäße Verwendung und Fehlanwendung

Verwendung

TF5120 TwinCAT 3 mxAutomation darf ausschließlich unter den dafür spezifizierten Systemvoraussetzungen [▶ 14] betrieben werden:

Eine andere oder darüber hinausgehende Verwendung gilt als Fehlanwendung und ist unzulässig. Für hieraus resultierende Schäden haftet der Hersteller nicht. Das Risiko trägt allein der Betreiber.

Zur bestimmungsgemäßen Verwendung gehört auch die Beachtung der Inbetriebnahme- und Konfigurationsanweisung in dieser Dokumentation.

Fehlanwendung

Alle von der bestimmungsgemäßen Verwendung abweichenden Anwendungen gelten als Fehlanwendung und sind unzulässig. Dazu zählen z. B.:

- Fehlkonfiguration (abweichend von dieser Dokumentation). Die Folge kann sein, dass der Roboter andere Aktionen ausführt, als vom Programmierer der SPS geplant.
- Verwendung in einer anderen Programmierumgebung als TwinCAT 3.1.4020.14 oder höher.

4 Sicherheit

Diese Dokumentation enthält Sicherheitshinweise, die sich spezifisch auf die hier beschriebene Software beziehen.

⚠ GEFAHR

Sicherheitsrelevante Informationen beachten

- Die sichere Nutzung dieses Produkts erfordert die Kenntnis und Einhaltung grundlegender Sicherheitsmaßnahmen.
  Tod, schwere Verletzungen oder Sachschäden können sonst die Folge sein.
- Machen Sie sich auch beim Lesen anderer Dokumentationen mit den darin verwendeten Sicherheits-Zeichen und mit der Bedeutung dieser Sicherheits-Zeichen vertraut.
  Beachten Sie Sicherheits-Zeichen und Sicherheits-Hinweise auch innerhalb anderer Dokumentationen sorgfältig.

⚠ GEFAHR

Das Kapitel "Sicherheit" in der Bedien- und Programmieranleitung der KUKA System Software (KSS) muss beachtet werden. Tod von Personen, schwere Verletzungen oder erheblicher Sachschaden können sonst die Folge sein.

5 Installation

5.1 Systemvoraussetzungen

Hardware

Robotersteuerung:

- KR C4
- KR C4 compact
- KR C5
- KR C5 micro

EtherCAT

- EL6695-**1001** (Vertrieb, Support und Service erfolgt von KUKA)

Externe SPS:

- Beckhoff TwinCAT 3 Steuerung

Software

Robotersteuerung:

- KUKA System Software 8.3, KUKA System Software 8.5, KUKA System Software 8.6 oder KUKA System Software 8.7

Folgende KRL-Ressourcen müssen frei sein:

KRL-Ressource	Nummer
E/As	2049 … 4 080
Interrupts	Index[90 … 97]

Robotersteuerung KR C4:

Bibliothek Tc3_mxAutomation für KSS 8.3	Bibliothek Tc3_mxAutoma-tionV3_0 für KSS 8.5	Bibliothek Tc3_mxAutoma-tionV3_1 für KSS 8.6
KUKA.PLC ProConOS 4-1 4.1 Option mit ConveyorTech: • KUKA.ConveyorTech 6.0 Option mit VectorMove: • KUKA.VectorMove 1.0 Standard-Laptop/-PC: • WorkVisual 4.0	KUKA.PLC ProConOS 4-1 5.0 Option mit ConveyorTech: • KUKA.ConveyorTech 7.0 Option mit VectorMove: • KUKA.VectorMove 2.0 Standard-Laptop/-PC: • WorkVisual 5.0	KUKA.PLC ProConOS 4-1 5.0 Option mit ConveyorTech: • KUKA.ConveyorTech 8.0 Option mit VectorMove: • KUKA.VectorMove 2.0 Standard-Laptop/-PC: • WorkVisual 6.0
TwinCAT 4020.14 oder höher	**TwinCAT 4022.27 oder höher**	**TwinCAT 4024.11 oder höher**

Robotersteuerung KR C5:

Bibliothek Tc3_mxAutomationV3_2 für KSS8.7
KUKA.PLC ProConOS 4-1 6.0 Option mit ConveyorTech: • KUKA.ConveyorTech 8.1 Option mit VectorMove: • KUKA.VectorMove 2.0

Bibliothek Tc3_mxAutomationV3_2 für KSS8.7
Standard-Laptop/-PC: • WorkVisual 6.0
TwinCAT 4024.17 oder höher

Robotersteuerung KR C4 oder KR C5:

Bibliothek Tc3_mxAutomationV3_3	
für KR C5 mit KSS8.7	**für KR C4 mit KSS8.6**
KUKA.PLC ProConOS 4-1 6.0 Option mit ConveyorTech: • KUKA.ConveyorTech 8.1	KUKA.PLC ProConOS 4-1 5.0 Option mit ConveyorTech: • KUKA.ConveyorTech 8.0
Option mit VectorMove: • KUKA.VectorMove 2.0 Standard-Laptop/-PC: • WorkVisual 6.0 Option mit LoadDataDetermination: • KUKA.LoadDataDetermination 7.2	
TwinCAT 3.1.4024.40 oder höher	

Industrie-PC/Embedded-PC:

- TwinCAT 3.1.4020.14 oder höher, abhängig von der benötigten mxAutomation-Version

6 Programmierung

6.1 Häufig verwendete Ein-/Ausgangsignale in den MC-Funktionsbausteinen

Die MC-Funktionsbausteine unterscheiden sich von den KRC-Funktionsbausteinen darin, dass sie der Norm PLC OPEN entsprechen oder näher kommen. Insbesondere das Verhalten des Signalausgangs Busy ist bei den MC-Funktionsbausteinen unterschiedlich. Für die Verkettung von Funktionsbausteinen muss hier der Signalausgang ComAcpt verwendet werden.

6.1.1 Eingangssignale

AxisGroupIdx

Über diesen Signaleingang wird die Nummer des Roboters gesetzt, der von einem Funktionsbaustein angesprochen wird.

Execute

Wenn dieser Signaleingang gesetzt wird, überträgt mxAutomation den zugehörigen Funktionsbaustein an den Roboter. Der Funktionsbaustein wird vom Roboter in einem Anweisungspuffer gespeichert, vorausgesetzt im Puffer ist noch Platz frei. Wird der Execute-Eingang zurückgesetzt, löscht mxAutomation den Funktionsbaustein wieder aus dem Puffer, es sei denn mit der Ausführung der Anweisung wurde bereits begonnen.

QueueMode

Modus, in dem eine Anweisung auf der Robotersteuerung ausgeführt wird

Wert	Name	Beschreibung
0	DIRECT	Anweisung wird direkt vom Submit-Interpreter (Submit-Programm) ausgeführt. **Dieser Modus steht bei einigen Funktionsbausteinen nicht zur Verfügung.**
1	ABORTING	Anweisung wird sofort vom Roboter-Interpreter (Hauptprogramm) ausgeführt. Zuvor werden alle aktiven Bewegungen und gepufferten Anweisungen abgebrochen und der Roboter wird vollständig abgebremst.
2	BUFFERED	Anweisung wird gepuffert. Gepufferte Anweisungen werden vom Roboter-Interpreter (Hauptprogramm) nach dem FIFO-Prinzip abgearbeitet.

CircType

Orientierungsführung für die Kreisbewegung

Wert	Name	Beschreibung
0	BASE	Basisbezogene Orientierungsführung während einer Kreisbewegung
1	PATH	Bahnbezogene Orientierungsführung während einer Kreisbewegung

OriType

Orientierungsführung für den TCP

Wert	Name	Beschreibung
0	VAR	Die Orientierung des TCP ändert sich während der Bewegung kontinuierlich.
1	CONSTANT	Die Orientierung des TCP bleibt während der Bewegung konstant.
2	JOINT	Die Orientierung des TCP ändert sich während der Bewegung kontinuierlich, jedoch nicht ganz gleichmäßig. Dies geschieht durch lineare Überführung (achsspezifisches Verfahren) der Handachswinkel. **Dieser Orientierungstyp ist nicht geeignet, wenn ein bestimmter Verlauf der Orientierung exakt gehalten werden muss.**

6.1.2 Ausgangssignale

ComBusy

Dieser Signalausgang zeigt an, dass der zugehörige Funktionsbaustein von der SPS in den Anweisungspuffer des Roboters gesendet und korrekt übertragen wurde.

ComAcpt

Dieser Signalausgang zeigt an, dass der zugehörige Funktionsbaustein von der SPS in den Anweisungspuffer des Roboters gesendet und korrekt übertragen wurde. Dieser Signalausgang ist identisch zum Signalausgang Done der KRC-Funktionsbausteine. Es wird empfohlen, diesen Signalausgang zum Überschleifen von Bewegungen zu verwenden.

Busy

Dieser Signalausgang zeigt an, dass mit der Ausführung des zugehörigen Funktionsbausteins begonnen wurde. Es ist jedoch möglich, dass der Funktionsbaustein noch nicht in den Anweisungspuffer des Roboters übertragen wurde. Dies unterscheidet diesen Signalausgang vom Signalausgang Busy der KRC-Funktionsbausteine.

Active

Dieser Signalausgang zeigt an, dass der zugehörige Funktionsbaustein aktuell auf dem Roboter ausgeführt wird. Er wird zurückgesetzt, wenn der Execute-Eingang zurückgesetzt wird.

Error

Dieser Signalausgang zeigt an, dass bei der Ausführung des zugehörigen Funktionsbausteins auf dem Roboter ein Fehler aufgetreten ist. In diesem Fall enthält der Signalausgang ErrorID eine Fehlernummer. Er wird zurückgesetzt, wenn der Execute-Eingang zurückgesetzt wird.

ErrorID

Dieser Signalausgang enthält eine Fehlernummer.

Die zur Fehlernummer gehörenden Fehler und Fehlerursachen sind hier beschrieben: (>>> Meldungen [▶ 145])

CommandAborted

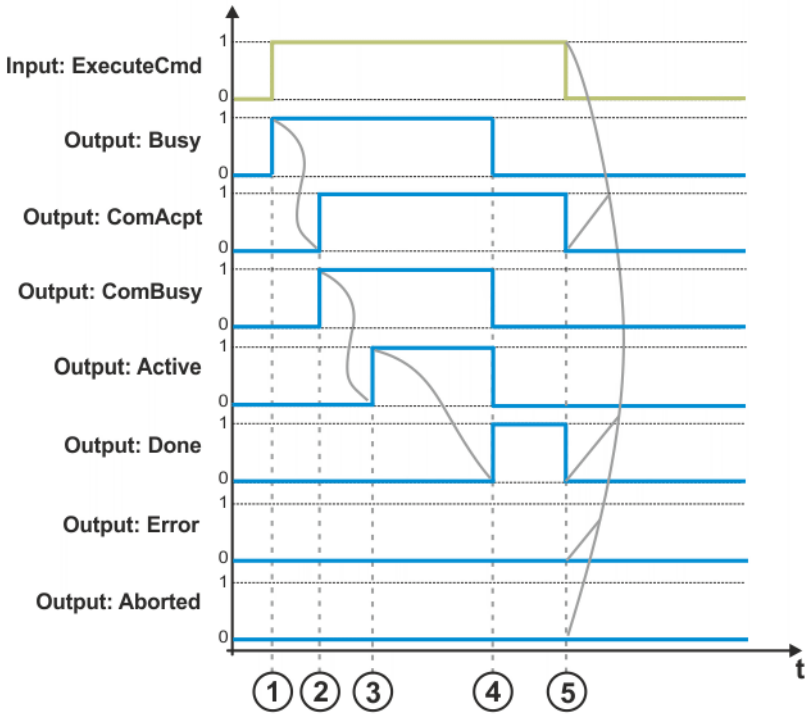
Dieser Signalausgang zeigt an, dass die Ausführung einer Anweisung oder Bewegung abgebrochen wurde.

6.1.3 Signalverlauf beim Ausführen von Execute

Beispiel

Der Signalverlauf wird für folgenden Fall dargestellt:

- Eine Anweisung wurde über Execute übertragen und erfolgreich ausgeführt.



Signalverlauf – Execute erfolgreich

Pos.	Beschreibung
1	Der Funktionsbaustein wird an den Roboter übertragen (= Aufforderung, Anweisung auszuführen).
2	Die Anweisung wurde übertragen (= befindet sich im Anweisungspuffer des Roboters). Die Ausgänge ComAcpt und ComBusy werden gesetzt.
3	Die Anweisung wird aktuell ausgeführt.
4	Die Anweisung wurde erfolgreich beendet. Weder ist ein Fehler aufgetreten noch wurde die Anweisung abgebrochen, z. B. durch KRC_Abort. Bei einem Fehler würde statt dem Done-Signal das Error-Signal und bei einem Abbruch statt dem Done-Signal das Aborted-Signal gesetzt.
5	Wenn der Execute-Eingang zurückgesetzt wird, werden auch die Ausgänge zurückgesetzt.

6.2 Strukturen für die Bewegungsprogrammierung (STRUCT)

In der SPS-Bibliothek können vordefinierte Datenstrukturen (STRUCT) verwendet werden.

Übersicht

Struktur	Beschreibung
APO	Überschleifparameter für einen Move-Bewegungsbefehl (>>> APO [▶ 19])

Struktur	Beschreibung
COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Zielposition bei einem Move- oder Jog-Bewegungsbefehl beziehen. (>>> COORDSYS [▶ 20])
E6AXIS	Winkel- oder Translationswerte der Achsen einer Achsgruppe für einen MoveAxis-Bewegungsbefehl (>>> E6AXIS [▶ 20])
E6POS	Kartesische Koordinaten der Zielposition für einen Move- oder Jog-Bewegungsbefehl (>>> E6POS [▶ 20])
FRAME	Raumkoordinaten und Orientierung für das TOOL- oder BASE-Koordinatensystem (>>> FRAME [▶ 21])
POSITION	Enthält alle Positionen sowie das Koordinatensystem, auf das sich ein Punkt bezieht. (>>> POSITION [▶ 21])
POSITION_ARRAY	Feld aus 100 Positionen vom Strukturtyp POSITION (>>> POSITION_ARRAY [▶ 21])

APO

Element	Typ	Beschreibung
PTP_MODE	INT	Gibt an, ob und wie der Zielpunkt einer PTP-Bewegung überschliffen wird. • **0**: Ohne Überschleifen (Default) • **1**: Bewirkt, dass der Zielpunkt überschliffen wird. Beim PTP-PTP-Überschleifen genügt die Angabe C_PTP. Beim PTP-CP-Überschleifen, d. h. wenn nach dem überschliffenen PTP-Satz ein LIN- oder CIRC-Satz folgt, muss ein weiterer Überschleifparameter angegeben werden. • **2**: PTP-CP-Überschleifen mit Distanzparameter • **3**: PTP-CP-Überschleifen mit Orientierungsparameter • **4**: PTP-CP-Überschleifen mit Geschwindigkeitsparameter
CP_MODE	INT	Gibt an, ob und wie der Zielpunkt einer CP-Bewegung (LIN, CIRC) überschliffen wird. • **0**: Ohne Überschleifen (Default) • **1**: Überschleifen mit Distanzparameter • **2**: Überschleifen mit Orientierungsparameter • **3**: Überschleifen mit Geschwindigkeitsparameter
CPTP	INT	Überschleifdistanz für PTP-Bewegungen (= Distanz vor dem Zielpunkt, bei der das Überschleifen frühestens beginnt). • **1 … 100 %** Maximaldistanz 100 %: die halbe Entfernung zwischen Startpunkt und Zielpunkt, bezogen auf die Kontur der PTP-Bewegung ohne Überschleifen.
CDIS	REAL	Distanzparameter (Einheit: mm) Das Überschleifen beginnt frühestens, wenn die Entfernung zum Zielpunkt den hier angegebenen Wert unterschreitet.
CORI	REAL	Orientierungsparameter (Einheit: °) Das Überschleifen beginnt frühestens, wenn der dominierende Orientierungswinkel (Drehen oder Schwenken der Längsachse des Werkzeugs) die hier angegebene Winkeldistanz zum Zielpunkt unterschreitet.

Element	Typ	Beschreibung
CVEL	INT	Geschwindigkeitsparameter • **1 … 100 %** Der Überschleifparameter gibt an, bei wieviel Prozent der programmierten Geschwindigkeit das Überschleifen in der Abbremsphase zum Zielpunkt hin frühestens beginnt.

COORDSYS

Element	Typ	Beschreibung
Tool	INT	Nummer des TOOL-Koordinatensystems • **-1**: Koordinatensystem wird nicht verändert • **0**: NULLFRAME • **1 … 16**: TOOL_DATA[1 … 16] Default: **-1**
Base	INT	Nummer des BASE-Koordinatensystems • **-1**: Koordinatensystem wird nicht verändert • **0**: NULLFRAME • **1 … 32**: BASE_DATA[1 … 32] Default: **-1**
IPO_MODE	INT	Interpolationsmodus • **0**: Das Werkzeug ist am Anbauflansch montiert. • **1**: Das Werkzeug ist ein feststehendes Werkzeug. Default: **0**

E6AXIS

Element	Typ	Beschreibung
A1	REAL	Position der Roboterachse A1 (Einheit: mm oder °)
A2	REAL	Position der Roboterachse A2 (Einheit: mm oder °)
A3	REAL	Position der Roboterachse A3 (Einheit: mm oder °)
A4	REAL	Position der Roboterachse A4 (Einheit: mm oder °)
A5	REAL	Position der Roboterachse A5 (Einheit: mm oder °)
A6	REAL	Position der Roboterachse A6 (Einheit: mm oder °)
E1	REAL	Position der Zusatzachse E1 (optional), (Einheit: mm oder °)
E2	REAL	Position der Zusatzachse E2 (optional), (Einheit: mm oder °)
E3	REAL	Position der Zusatzachse E3 (optional), (Einheit: mm oder °)
E4	REAL	Position der Zusatzachse E4 (optional), (Einheit: mm oder °)
E5	REAL	Position der Zusatzachse E5 (optional), (Einheit: mm oder °)
E6	REAL	Position der Zusatzachse E6 (optional), (Einheit: mm oder °)

E6POS

Element	Typ	Beschreibung
X	REAL	Position auf der X-Achse (Einheit: mm)
Y	REAL	Position auf der Y-Achse (Einheit: mm)
Z	REAL	Position auf der Z-Achse (Einheit: mm)
A	REAL	Rotation um die Z-Achse • **-180° … +180°**
B	REAL	Rotation um die Y-Achse

Element	Typ	Beschreibung	
		• **-180° … +180°**	
C	REAL	Rotation um die X-Achse • **-180° … +180°**	
S	INT	Status	Die Werte von Position (X, Y, Z) und Orientierung (A, B, C) des TCP reichen nicht aus, um die Position eines Roboters eindeutig festzulegen, da bei gleichem TCP dennoch mehrere Achsstellungen möglich sind. Status und Turn dienen dazu, aus mehreren möglichen Achsstellungen eine eindeutige Stellung festzulegen. **Weitere Informationen zu Status und Turn sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.**
T	INT	Turn	
E1	REAL	Position der Zusatzachse E1 (optional), (Einheit: mm oder °)	
E2	REAL	Position der Zusatzachse E2 (optional), (Einheit: mm oder °)	
E3	REAL	Position der Zusatzachse E3 (optional), (Einheit: mm oder °)	
E4	REAL	Position der Zusatzachse E4 (optional), (Einheit: mm oder °)	
E5	REAL	Position der Zusatzachse E5 (optional), (Einheit: mm oder °)	
E6	REAL	Position der Zusatzachse E6 (optional), (Einheit: mm oder °)	

FRAME

Element	Typ	Beschreibung
X	REAL	Position auf der X-Achse (Einheit: mm)
Y	REAL	Position auf der Y-Achse (Einheit: mm)
Z	REAL	Position auf der Z-Achse (Einheit: mm)
A	REAL	Orientierung der Z-Achse • **-180° … +180°**
B	REAL	Orientierung der Y-Achse • **-180° … +180°**
C	REAL	Orientierung der X-Achse • **-180° … +180°**

POSITION

Element	Beschreibung
COORDSYS	(>>> COORDSYS [▶ 20])
E6POS	(>>> E6POS [▶ 20])
E6AXIS	(>>> E6AXIS [▶ 20])

POSITION_ARRAY

Element	Beschreibung
POSITION	• **1 … 100** (>>> POSITION [▶ 21])

6.3 Daten eines kartesischen Arbeitsraums

Hier sind die Daten eines kartesischen Arbeitsraums, die in einigen Funktionsbausteinen verwendet werden, vorab beschrieben.

Ursprung und Orientierung

Mit den folgenden Elementen werden der Ursprung und die Orientierung eines kartesischen Arbeitraums angegeben. Diese Elemente beziehen sich auf das WORLD-Koordinatensystem und werden in der Variable BOX definiert.

Element	Datentyp	Einheit	Minimum	Maximum
X	REAL	mm	-	-
Y	REAL	mm	-	-
Z	REAL	mm	-	-
A	REAL	°	-180	180
B	REAL	°	-180	180
C	REAL	°	-180	180

Abmessungen

Mit den folgenden Elementen werden die Abmessungen eines kartesischen Arbeitsraums angegeben.

Element	Datentyp	Einheit
X1	REAL	mm
X2	REAL	mm
Y1	REAL	mm
Y2	REAL	mm
Z1	REAL	mm
Z2	REAL	mm

6.4 Daten eines achsspezifischen Arbeitsraums

Hier sind die Daten eines achsspezifischen Arbeitsraums, die in einigen Funktionsbausteinen verwendet werden, vorab beschrieben.

Diese Daten werden mit den folgenden Elementen in der Variable AXBOX definiert.

Roboterachsen

Element	Datentyp	Einheit	Beschreibung
A1_N	REAL	mm/°	Untergrenze für Achswinkel
A2_N	REAL	mm/°	
A3_N	REAL	mm/°	
A4_N	REAL	mm/°	
A5_N	REAL	mm/°	
A6_N	REAL	mm/°	
A1_P	REAL	mm/°	Obergrenze für Achswinkel
A2_P	REAL	mm/°	
A3_P	REAL	mm/°	
A4_P	REAL	mm/°	
A5_P	REAL	mm/°	
A6_P	REAL	mm/°	

Zusatzachsen

Element	Datentyp	Einheit	Beschreibung
E1_N	REAL	mm/°	Untergrenze für Achswinkel
E2_N	REAL	mm/°	

Element	Datentyp	Einheit	Beschreibung
E3_N	REAL	mm/°	
E4_N	REAL	mm/°	
E5_N	REAL	mm/°	
E6_N	REAL	mm/°	
E1_P	REAL	mm/°	Obergrenze für Achswinkel
E2_P	REAL	mm/°	
E3_P	REAL	mm/°	
E4_P	REAL	mm/°	
E5_P	REAL	mm/°	
E6_P	REAL	mm/°	

6.5 Programmiertipps für Tc3_mxAutomation

Instanziierung

Folgende Funktionsbausteine dürfen pro Roboter nur einfach instanziiert werden. Bei einer mehrfachen Instanziierung werden die Signale des zuletzt aufgerufenen Funktionsbausteins ausgegeben.

- KRC_ReadAxisGroup
- KRC_Initialize
- KRC_SetOverride
- KRC_AutomaticExternal
- KRC_AutoStart
- KRC_Diag
- KRC_WriteAxisGroup

Alle weiteren Funktionsbausteine, die im mxAutomation-Roboterprogramm verwendet werden, können als Multiinstanz-Aufruf angelegt werden.

ExecuteCmd

- Einen ExecuteCmd-Eingang möglichst immer nur für einen Funktionsbaustein desselben Roboters gleichzeitig setzen.
- Einen ExecuteCmd-Eingang nach einer Aktivierung erst wieder zurücksetzen, wenn der Funktionsbaustein die Ausführung der Anweisung durch das Done-Signal bestätigt oder durch das Error- oder Aborted-Signal anzeigt, dass die Anweisung nicht ausgeführt wurde. Wird der ExecuteCmd-Eingang vorher zurückgesetzt, wird nicht zurückgemeldet, ob die Anweisung ausgeführt wurde.
- Wenn der Busy- oder ComAcpt-Ausgang eines Funktionsbausteins mit dem ExecuteCmd-Eingang des folgenden Bausteins verbunden wird, kann eine Sequenz aufeinanderfolgender Funktionen in den Anweisungspuffer übertragen und ausgeführt werden.

Programm-Override

Wenn der Funktionsbaustein KRC_AutomaticExternal verwendet wird, muss der Programm-Override auf einen Wert größer Null gesetzt sein. Nur dann kann ein SPS-Programm abgearbeitet werden.

Überschleifen

Überschleifen bedeutet: Der programmierte Punkt wird nicht genau angefahren. Überschleifen ist eine Option, welche bei der Bewegungsprogrammierung ausgewählt werden kann.

- Überschleifen ist nur möglich, wenn 2 Bewegungsanweisungen aufeinander folgen.
- Überschleifen ist nur möglich, wenn nach der Bewegungsanweisung eine Bewegungsanweisung folgt, die im Modus BUFFERED übertragen wird.

6.5.1 Aufbau eines SPS-Programms

Jedes SPS-Programm muss folgende Funktionsbausteine enthalten:

1. KRC_ReadAxisGroup
2. KRC_Initialize
3. KRC_WriteAxisGroup

Zwischen den Funktionsbausteinen KRC_Initialize und KRC_WriteAxisGroup können beliebig viele Funktionsbausteine eingefügt werden, z. B. KRC_AutomaticExternal, KRC_MoveDirectAbsolute usw.

6.5.2 Roboter stoppen

Es gibt 4 Arten, den Roboter zu stoppen. Diese unterscheiden sich darin, wie die Bewegung fortgesetzt werden soll:

- Stoppen und die Programmbearbeitung abbrechen (Eingang RESET am Funktionsbaustein KRC_AutomaticExternal)
- Stoppen und die gepufferten Anweisungen löschen (KRC_Abort)
- Stoppen und auf eine Bedingung warten, danach weiterfahren und die gepufferten Anweisungen durchführen (KRC_DeclareInterrupt)
- Stoppen und auf den Funktionsbaustein KRC_Continue warten, danach weiterfahren und die gepufferten Anweisungen durchführen (KRC_Interrupt)

6.6 Häufig verwendete Ein-/Ausgangsignale in den KRC-Funktionsbausteinen

6.6.1 Eingangssignale

AxisGroupIdx

Über diesen Signaleingang wird das Robotersystem ausgewählt. Es können bis zu 5 Robotersysteme verwendet werden.

Ein Robotersystem ist eine Gruppierung von Achsen zu einer Achsgruppe.

ExecuteCmd

Wenn dieser Signaleingang gesetzt wird, überträgt mxAutomation den zugehörigen Funktionsbaustein an den Roboter. Der Funktionsbaustein wird vom Roboter in einem Anweisungspuffer gespeichert, vorausgesetzt im Puffer ist noch Platz frei. Wird der ExecuteCmd-Eingang zurückgesetzt, löscht mxAutomation den Funktionsbaustein wieder aus dem Puffer, es sei denn mit der Ausführung der Anweisung wurde bereits begonnen.

BufferMode

Modus, in dem eine Anweisung auf der Robotersteuerung ausgeführt wird

Wert	Name	Beschreibung
0	DIRECT	Anweisung wird direkt vom Submit-Interpreter (Submit-Programm) ausgeführt. **Dieser Modus steht bei einigen Funktionsbausteinen nicht zur Verfügung.**

Wert	Name	Beschreibung
1	ABORTING	Anweisung wird sofort vom Roboter-Interpreter (Hauptprogramm) ausgeführt. Zuvor werden alle aktiven Bewegungen und gepufferten Anweisungen abgebrochen und der Roboter wird vollständig abgebremst.
2	BUFFERED	Anweisung wird gepuffert. Gepufferte Anweisungen werden vom Roboter-Interpreter (Hauptprogramm) nach dem FIFO-Prinzip abgearbeitet.

6.6.2 Ausgangssignale

Busy

Dieser Signalausgang zeigt an, dass der zugehörige Funktionsbaustein aktuell in den Anweisungspuffer des Roboters übertragen wird oder bereits übertragen wurde. Er wird zurückgesetzt, wenn der ExecuteCmd-Eingang zurückgesetzt wird.

Active

Dieser Signalausgang zeigt an, dass der zugehörige Funktionsbaustein aktuell auf dem Roboter ausgeführt wird. Er wird zurückgesetzt, wenn der ExecuteCmd-Eingang zurückgesetzt wird.

Done

Dieser Signalausgang zeigt an, dass der zugehörige Funktionsbaustein erfolgreich vom Roboter ausgeführt wurde. Er wird zurückgesetzt, wenn der ExecuteCmd-Eingang zurückgesetzt wird.

Error

Dieser Signalausgang zeigt an, dass bei der Ausführung des zugehörigen Funktionsbausteins ein Fehler aufgetreten ist. In diesem Fall enthält der Signalausgang ErrorID eine Fehlernummer. Er wird zurückgesetzt, wenn der ExecuteCmd-Eingang zurückgesetzt wird.

ErrorID

Dieser Signalausgang enthält eine Fehlernummer.

Die zur Fehlernummer gehörenden Fehler und Fehlerursachen sind hier beschrieben: (>>> Meldungen [▶ 145])

Aborted

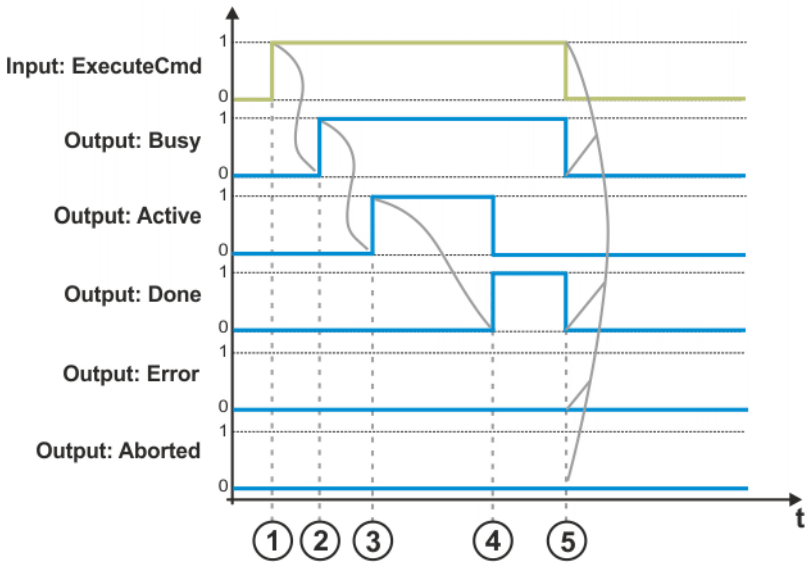
Dieser Signalausgang wird entweder gesetzt, wenn der Funktionsbaustein KRC_Abort ausgeführt wird, oder wenn eine Anweisung im Modus ABORTING ausgeführt wird. Er wird zurückgesetzt, wenn der ExecuteCmd-Eingang zurückgesetzt wird.

6.6.3 Signalverlauf beim Ausführen von ExecuteCmd

Beispiel

Der Signalverlauf wird für folgenden Fall dargestellt:

- Eine Anweisung wurde über ExecuteCmd übertragen und erfolgreich ausgeführt.



Signalverlauf – ExecuteCmd erfolgreich

Pos.	Beschreibung
1	Der Funktionsbaustein wird an den Roboter übertragen (= Aufforderung, Anweisung auszuführen).
2	Die Anweisung wird übertragen.
3	Die Anweisung wird aktuell ausgeführt.
4	Die Anweisung wurde erfolgreich beendet. Weder ist ein Fehler aufgetreten noch wurde die Anweisung abgebrochen, z. B. durch KRC_Abort. Bei einem Fehler würde statt dem Done-Signal das Error-Signal und bei einem Abbruch statt dem Done-Signal das Aborted-Signal gesetzt.
5	Wenn der ExecuteCmd-Eingang zurückgesetzt wird, werden auch die Ausgänge zurückgesetzt.

Variationen

- ExecuteCmd wird zurückgesetzt, bevor Done gesetzt wird. In diesem Fall wird die Anweisung zwar ausgeführt, jedoch das Done-Signal nicht gesetzt. D. h., es wird nicht zurückgemeldet, dass die Anweisung ausgeführt wurde.
- ExecuteCmd wird zurückgesetzt, bevor Error oder Aborted gesetzt wird. In diesem Fall wird die Anweisung abgebrochen, jedoch das Error- oder das Aborted-Signal nicht gesetzt. D. h., es wird nicht zurückgemeldet, dass die Anweisung abgebrochen wurde.
- ExecuteCmd wird zurückgesetzt, bevor Active gesetzt wird. In diesem Fall wird der Funktionsbaustein aus dem Anweisungspuffer des Roboters gelöscht.
- ExecuteCmd wird zurückgesetzt, bevor Busy gesetzt wird. In diesem Fall wird der Funktionsbaustein nicht an den Roboter übertragen und die Anweisung folglich nicht ausgeführt.

7 Funktionsbausteine

7.1 Übersicht Funktionsbausteine

Administrative Funktionen	
KRC_ReadAxisGroup	(>>> Ausgänge des Robotersystems lesen [▶ 117])
KRC_WriteAxisGroup	(>>> Eingänge des Robotersystems schreiben [▶ 117])
KRC_Initialize	(>>> mxA-Schnittstelle initialisieren [▶ 118])
KRC_SetOverride	(>>> Programm-Override (POV) einstellen [▶ 119])
KRC_AutomaticExternal	(>>> Automatik Extern-Signale der Robotersteuerung ansteuern und lesen [▶ 120])
KRC_AutoStart	(>>> Eingänge von KRC_AutomaticExternal automatisch setzen [▶ 122])
KRC_ReadActualPosition	(>>> Aktuelle Roboterposition lesen [▶ 123])
KRC_ReadActualAxisPosition	(>>> Aktuelle Achsposition lesen [▶ 124])
KRC_ReadActualVelocity	(>>> Aktuelle Bahngeschwindigkeit lesen [▶ 125])
KRC_ReadActualAxisVelocity	(>>> Aktuelle Achsgeschwindigkeit lesen [▶ 126])
KRC_ReadActualAcceleration	(>>> Aktuelle Roboterbeschleunigung lesen [▶ 127])
KRC_ReadDigitalInput	(>>> Digitalen Eingang lesen [▶ 128])
KRC_ReadDigitalInput1To8	(>>> Digitalen Eingang 1 bis 8 lesen [▶ 128])
KRC_ReadDigitalInputArray	(>>> Mehrere digitale Eingänge lesen [▶ 129])
KRC_ReadDigitalOutput	(>>> Digitalen Ausgang lesen [▶ 130])
KRC_WriteDigitalOutput	(>>> Digitalen Ausgang schreiben [▶ 131])
KRC_WriteDigitalOutput1To8	(>>> Digitalen Ausgang 1 bis 8 schreiben [▶ 132])
KRC_ReadAnalogInput	(>>> Analogen Eingang lesen [▶ 132])
KRC_ReadAnalogOutput	(>>> Analogen Ausgang lesen [▶ 133])
KRC_WriteAnalogOutput	(>>> Analogen Ausgang schreiben [▶ 134])
KRC_SetCoordSys	(>>> Werkzeug, Basis und Interpolationsmodus auswählen [▶ 135])
KRC_ReadToolData	(>>> TOOL-Daten lesen [▶ 135])
KRC_WriteToolData	(>>> TOOL-Daten schreiben [▶ 136])
KRC_ReadBaseData	(>>> BASE-Daten lesen [▶ 137])
KRC_WriteBaseData	(>>> BASE-Daten schreiben [▶ 138])
KRC_ReadLoadData	(>>> Lastdaten lesen [▶ 139])
KRC_WriteLoadData	(>>> Lastdaten schreiben [▶ 140])
KRC_ReadSoftEnd	(>>> Software-Endschalter der Roboterachsen lesen [▶ 141])
KRC_ReadSoftEndExt	(>>> Software-Endschalter der Zusatzachsen lesen [▶ 142])
KRC_WriteSoftEnd	(>>> Software-Endschalter der Roboterachsen schreiben [▶ 143])
KRC_WriteSoftEndExt	(>>> Software-Endschalter der Zusatzachsen schreiben [▶ 144])

Funktionen für die Bewegungsprogrammierung	
KRC_MoveLinearAbsolute	(>>> Absolute kartesische Position linear anfahren [▶ 30])
KRC_MoveLinearRelative	(>>> Relative kartesische Position mit einer Linearbewegung anfahren [▶ 32])
KRC_MoveDirectAbsolute	(>>> Absolute kartesische Position schnellstmöglich anfahren [▶ 33])
KRC_MoveDirectRelative	(>>> Relative kartesische Position schnellstmöglich anfahren [▶ 35])

Funktionen für die Bewegungsprogrammierung	
KRC_MoveAxisAbsolute	(>>> Achsspezifische Position schnellstmöglich anfahren [▶ 37])
KRC_MoveCircAbsolute	(>>> Absolute kartesische Position mit Kreisbewegung anfahren [▶ 38])
KRC_MoveCircRelative	(>>> Relative kartesische Position mit Kreisbewegung anfahren [▶ 40])
KRC_Jog	(>>> Zielposition manuell anfahren [▶ 43])
KRC_JogToolRelative	(>>> Verfahren per Tippbetrieb auf eine relative Endposition im TOOL-Koordinatensystem mit einer Linearbewegung [▶ 45])
KRC_JogLinearRelative	(>>> Verfahren per Tippbetrieb auf eine relative kartesische Position mit einer Linearbewegung [▶ 46])
KRC_JogAdvanced	(>>> Zielposition manuell anfahren (erweitert) [▶ 48])

Funktionen für die Bewegungsprogrammierung (Konform zu PLC OPEN)	
MC_MoveLinearAbsolute	(>>> Absolute kartesische Position linear anfahren [▶ 51])
MC_MoveLinearRelative	(>>> Anfahren einer relativen kartesischen Position mit einer Linearbewegung [▶ 53])
MC_MoveDirectAbsolute	(>>> Absolute kartesische Position schnellstmöglich anfahren [▶ 54])
MC_MoveDirectRelative	(>>> Relative kartesische Position schnellstmöglich anfahren [▶ 56])
MC_MoveAxisAbsolute	(>>> Achsspezifische Position schnellstmöglich anfahren [▶ 57])
MC_MoveCircularAbsolute	(>>> Absolute kartesische Position mit Kreisbewegung anfahren [▶ 58])
MC_MoveCircularRelative	(>>> Anfahren einer relativen kartesischen Position mit einer Kreisbewegung [▶ 60])

Funktionen zur Programmablaufkontrolle	
KRC_Abort	(>>> Programm abbrechen [▶ 63])
KRC_AbortAdvanced	(>>> Programm abbrechen (erweitert) [▶ 63])
KRC_Interrupt	(>>> Roboterbewegung pausieren [▶ 64])
KRC_Continue	(>>> Programm fortsetzen [▶ 65])
KRC_WaitForInput	(>>> Auf digitalen Eingang warten [▶ 66])

Funktionen zur Interrupt-Programmierung	
KRC_DeclareInterrupt	(>>> Interrupt deklarieren [▶ 67])
KRC_ActivateInterrupt	(>>> Interrupt aktivieren [▶ 68])
KRC_DeactivateInterrupt	(>>> Interrupt deaktivieren [▶ 69])
KRC_ReadInterruptState	(>>> Status eines Interrupts lesen [▶ 70])

Funktionen für bahnbezogene Schaltaktionen	
KRC_SetDistanceTrigger	(>>> Schaltaktion zu Bahnpunkten aktivieren [▶ 71])
KRC_SetPathTrigger	(>>> Bahnbezogene Schaltaktion aktivieren [▶ 72])

Diagnose-Funktionen	
KRC_Error	(>>> Fehlerzustände lesen und quittieren [▶ 74])
KRC_ReadMXAStatus	(>>> Aktuellen Status der mxA-Schnittstelle lesen [▶ 75])
KRC_ReadMXAError	(>>> Fehlermeldungen der mxA-Schnittstelle lesen [▶ 76])
KRC_MessageReset	(>>> Fehlermeldungen der mxA-Schnittstelle quittieren [▶ 77])
KRC_ReadKRCError	(>>> Fehlermeldungen der Robotersteuerung lesen [▶ 77])
KRC_Diag	(>>> Diagnosesignale lesen [▶ 78])

Allgemeine Sonderfunktionen	
KRC_ReadSysVar	(>>> Systemvariablen lesen [▶ 80])
KRC_WriteSysVar	(>>> Systemvariablen schreiben [▶ 81])
KRC_BrakeTest	(>>> Bremsentest aufrufen [▶ 83])
KRC_MasRef	(>>> Justagereferenzierung aufrufen [▶ 84])
KRC_ReadSafeOPStatus	(>>> Signale der Sicherheitssteuerung lesen [▶ 86])
KRC_ReadTouchUPState	(>>> Zustand der TouchUp-Statustasten lesen [▶ 87])
KRC_TouchUP	(>>> Punkte teachen [▶ 87])
KRC_SetAdvance	(>>> Einstellungen für den Vorlauf ändern [▶ 88])
KRC_GetAdvance	(>>> Einstellungen für den Vorlauf auslesen [▶ 89])
KRC_Forward	(>>> Kartesische Roboterposition aus Achswinkeln berechnen [▶ 90])
KRC_ForwardAdvanced	(>>> Kartesische Roboterposition aus Achswinkeln berechnen (erweitert) [▶ 91])
KRC_Inverse	(>>> Achswinkel aus kartesischer Roboterposition berechnen [▶ 92])
KRC_InverseAdvanced	(>>> Achswinkel aus kartesischer Roboterposition berechnen (erweitert) [▶ 94])
KRC_TechFunction	(>>> KRL-Programme ausführen [▶ 95])
KRC_TechFunctionAdvanced	(>>> KRL-Programme ausführen (erweitert) [▶ 96])
KRC_ActivatePosConversion	(>>> Aktuelle Roboterposition in anderem Koordinatensystem anzeigen [▶ 97])

Sonderfunktionen für Conveyor	
KRC_ConvIniOff	(>>> Conveyor initialisieren [▶ 98])
KRC_ConvOn	(>>> Conveyor aktivieren [▶ 99])
KRC_ConvFollow	(>>> Bauteil auf Conveyor verfolgen [▶ 99])
KRC_ConvSkip	(>>> Bauteil von Conveyor aufnehmen [▶ 101])
KRC_ConvDelWPS	(>>> Bauteil auf Conveyor löschen [▶ 102])
KRC_ActivateConvInterrupt	(>>> Interrupts für Überwachung aktivieren [▶ 103])
KRC_DeactivateConvInterrupt	(>>> Interrupts für Überwachung deaktivieren [▶ 104])

Um diese Funktionsbausteine verwenden zu können, muss KUKA.ConveyorTech auf der Robotersteuerung installiert sein.

Sonderfunktionen für VectorMove	
KRC_VectorMoveOn	(>>> Bewegung entlang eines Vektors aktivieren [▶ 105])
KRC_VectorMoveOff	(>>> Bewegung entlang eines Vektors deaktivieren [▶ 106])

Um diese Funktionsbausteine verwenden zu können, muss KUKA.VectorMove auf der Robotersteuerung installiert sein.

Sonderfunktionen für LoadDataDetermination	
KRC_LDDconfig	(>>> Lastdatenermittlung konfigurieren [▶ 107])
KRC_LDDcheckPos	(>>> Startposition der Lastdatenermittlung prüfen [▶ 108])
KRC_LDDtestRun	(>>> Testfahrt vor Lastdatenermittlung durchführen [▶ 109])
KRC_LDDstart	(>>> Lastdatenermittlung durchführen [▶ 110])
KRC_LDDwriteLoad	(>>> Lastdaten zuweisen [▶ 111])

Um diese Funktionsbausteine verwenden zu können, muss KUKA.LoadDataDetermination auf der Robotersteuerung installiert sein.

Sonderfunktionen für Arbeitsräume	
KRC_WriteWorkspace	(>>> Kartesische Arbeitsräume konfigurieren [▶ 111])
KRC_ReadWorkspace	(>>> Konfiguration der kartesischen Arbeitsräume lesen [▶ 113])
KRC_WriteAxWorkspace	(>>> Achsspezifische Arbeitsräume konfigurieren [▶ 113])
KRC_ReadAxWorkspace	(>>> Konfiguration der achsspezifischen Arbeitsräume lesen [▶ 115])
KRC_ReadWorkstates	(>>> Status der Arbeitsräume lesen [▶ 115])

7.2 Funktionen für die Bewegungsprogrammierung

Bewegungsanweisungen können grundsätzlich nur im Modus ABORTING oder BUFFERED ausgeführt werden. Wenn eine Bewegung überschliffen werden soll, muss die nachfolgende Bewegung im Modus BUFFERED übertragen werden.

Mit dem Active-Ausgang ist Überschleifen nicht möglich, da die nächste Bewegungsanweisung erst gesendet wird, wenn die vorherige ausgeführt wird. Überschleifen ist nur möglich, wenn der Busy-Ausgang des vorherigen Funktionsbausteins mit dem ExecuteCmd-Eingang des folgenden Bausteins verbunden wird.

Weitere Informationen zu den Grundlagen der Bewegungsprogrammierung - Bewegungsarten, Orientierungsführung, Überschleifen - sind in der Bedien- und Programmieranleitung der KUKA System Software zu finden.

7.2.1 Absolute kartesische Position linear anfahren

Beschreibung

Mit dem Funktionsbaustein KRC_MoveLinearAbsolute wird eine Linearbewegung zu einer kartesischen Zielposition ausgeführt. Die Koordinaten der Zielposition sind absolut.

Wenn die Bewegung als Spline-Bewegung ausgeführt wird, muss folgendes beachtet werden: Wenn die Bewegung überschliffen wird und sich keine weitere Bewegungsanweisung im Puffer befindet, wird der Ausgang Done für die Spline-Bewegung nicht gesetzt. Die Bewegung wird in diesem Fall nicht am Zielpunkt beendet, sondern am Überschleifpunkt.

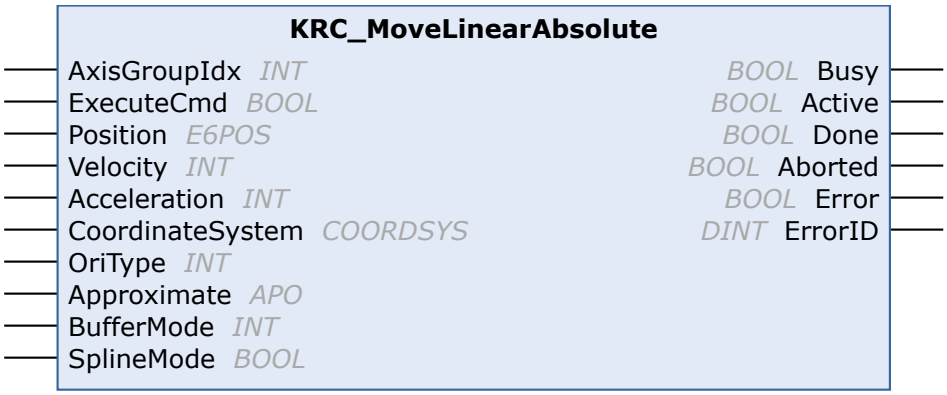


Abb. 1: Funktionsbaustein KRC_MoveLinearAbsolute

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
Position	E6POS	Koordinaten der kartesischen Zielposition (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Zielposition (= Position des TCP bezogen auf den Ursprung des BASE-Koordinatensystems).
Velocity	INT	Geschwindigkeit • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_VEL_CP des Robotersystems. Default: **0 %** (= Geschwindigkeit wird nicht verändert)
Acceleration	INT	Beschleunigung • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_ACC_CP des Robotersystems. Default: **0 %** (= Beschleunigung wird nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Zielposition beziehen (>>> COORDSYS [▶ 20])
OriType	INT	Orientierungsführung des TCP • **0**: VAR • **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 16])
Approximate	APO	Überschleifparameter (>>> APO [▶ 19])
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])
SplineMode	BOOL	TRUE = Bewegung wird als Spline-Bewegung ausgeführt. FALSE = Bewegung wird als herkömmliche Linearbewegung ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet

Parameter	Typ	Beschreibung
Aborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.2.2 Relative kartesische Position mit einer Linearbewegung anfahren

Beschreibung

Mit dem Funktionsbaustein KRC_MoveLinearRelative wird eine Linearbewegung zu einer relativen kartesischen Zielposition ausgeführt. Der Parameter Position enthält die Strecke von der aktuellen Position zur Zielposition.

Diese Anweisung bezieht sich immer auf die aktuelle Roboterposition. Wenn die Bewegung abgebrochen wurde und wieder ausgeführt wird, fährt der Roboter von der Abbruch-Position aus noch einmal die komplette Strecke.

Wenn die Bewegung als Spline-Bewegung ausgeführt wird, muss folgendes beachtet werden: Wenn die Bewegung überschliffen wird und sich keine weitere Bewegungsanweisung im Puffer befindet, wird der Ausgang Done für die Spline-Bewegung nicht gesetzt. Die Bewegung wird in diesem Fall nicht am Zielpunkt beendet, sondern am Überschleifpunkt.

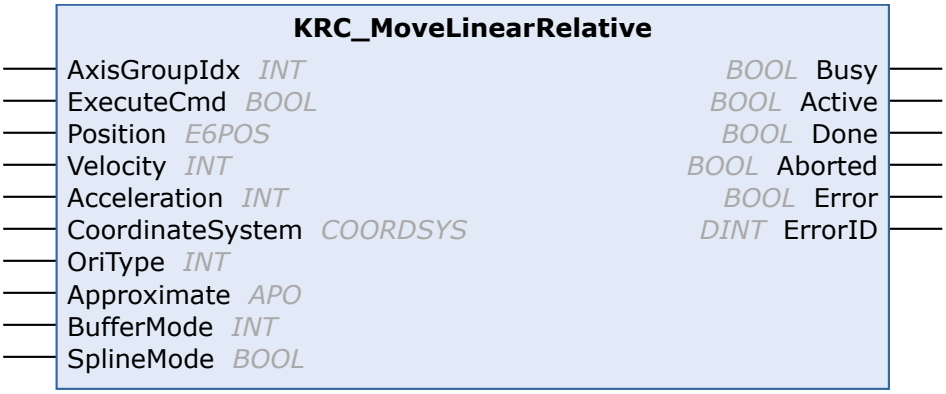


Abb. 2: Funktionsbaustein KRC_MoveLinearRelative

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
Position	E6POS	Abstand zwischen Zielposition und aktueller Position. Die Zielposition basiert auf der aktuellen Position. (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Zielposition (= Position des TCP relativ zum Ursprung des ausgewählten Koordinatensystems).
Velocity	INT	Geschwindigkeit • **0 … 100 %**

Parameter	Typ	Beschreibung
		Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_VEL_CP des Robotersystems. Default: **0 %** (= Geschwindigkeit wird nicht verändert)
Acceleration	INT	Beschleunigung • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_ACC_CP des Robotersystems. Default: **0 %** (= Beschleunigung wird nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Zielposition beziehen (>>> COORDSYS [▶ 20])
OriType	INT	Orientierungsführung des TCP • **0**: VAR • **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 16])
Approximate	APO	Überschleifparameter (>>> APO [▶ 19])
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])
SplineMode	BOOL	TRUE = Bewegung wird als Spline-Bewegung ausgeführt. FALSE = Bewegung wird als herkömmliche Linearbewegung ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
Aborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.2.3 Absolute kartesische Position schnellstmöglich anfahren

Beschreibung

Mit dem Funktionsbaustein KRC_MoveDirectAbsolute wird eine Punkt-zu-Punkt-Bewegung zu einer kartesischen Zielposition ausgeführt. Die Koordinaten der Zielposition sind absolut.

Hierbei bewegt sich der Roboter schnellstmöglich zur Zielposition. Die schnellste Bahn ist in der Regel nicht die kürzeste Bahn und somit keine Gerade. Auf dem Robotersystem entspricht dies einer PTP-Bewegung.

Wenn die Bewegung als Spline-Bewegung ausgeführt wird, muss folgendes beachtet werden: Wenn die Bewegung überschliffen wird und sich keine weitere Bewegungsanweisung im Puffer befindet, wird der Ausgang Done für die Spline-Bewegung nicht gesetzt. Die Bewegung wird in diesem Fall nicht am Zielpunkt beendet, sondern am Überschleifpunkt.

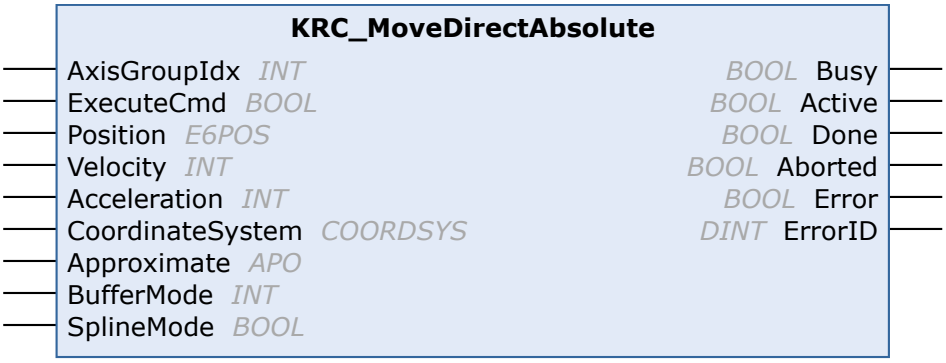


Abb. 3: Funktionsbaustein KRC_MoveDirectAbsolute

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
Position	E6POS	Koordinaten der kartesischen Zielposition (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Zielposition (= Position des TCP bezogen auf den Ursprung des BASE-Koordinatensystems). **Hinweis**: Wird für Status und Turn der Wert -1 übergeben, so wird die Zielposition auf dem kürzesten Weg angefahren.
Velocity	INT	Geschwindigkeit • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_VEL_CP des Robotersystems. Default: **0 %** (= Geschwindigkeit wird nicht verändert)
Acceleration	INT	Beschleunigung • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_ACC_CP des Robotersystems. Default: **0 %** (= Beschleunigung wird nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Zielposition beziehen (>>> COORDSYS [▶ 20])
Approximate	APO	Überschleifparameter (>>> APO [▶ 19])
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING

Parameter	Typ	Beschreibung
		• **2**: BUFFERED (>>> BufferMode [▶ 24])
SplineMode	BOOL	TRUE = Bewegung wird als Spline-Bewegung ausgeführt. FALSE = Bewegung wird als herkömmliche Punkt-zu-Punkt-Bewegung ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
Aborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.2.4 Relative kartesische Position schnellstmöglich anfahren

Beschreibung

Mit dem Funktionsbaustein KRC_MoveDirectRelative wird eine Punkt-zu-Punkt-Bewegung zu einer relativen kartesischen Zielposition ausgeführt. Der Parameter Position enthält die Strecke von der aktuellen Position zur Zielposition. Auf dem Robotersystem entspricht dies einer PTP_REL-Bewegung.

Diese Anweisung bezieht sich immer auf die aktuelle Roboterposition. Wenn die Bewegung abgebrochen wurde und wieder ausgeführt wird, fährt der Roboter von der Abbruch-Position aus noch einmal die komplette Strecke.

Wenn die Bewegung als Spline-Bewegung ausgeführt wird, muss folgendes beachtet werden: Wenn die Bewegung überschliffen wird und sich keine weitere Bewegungsanweisung im Puffer befindet, wird der Ausgang Done für die Spline-Bewegung nicht gesetzt. Die Bewegung wird in diesem Fall nicht am Zielpunkt beendet, sondern am Überschleifpunkt.

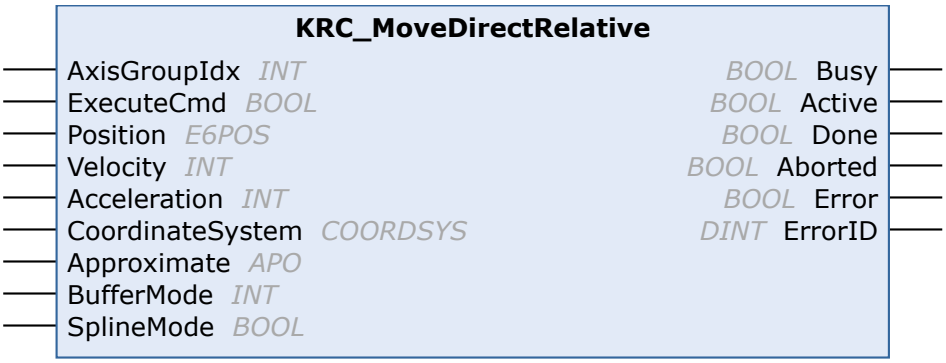


Abb. 4: Funktionsbaustein KRC_MoveDirectRelative

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.

Parameter	Typ	Beschreibung
Position	E6POS	Abstand zwischen Zielposition und aktueller Position. Die Zielposition basiert auf der aktuellen Position. (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Zielposition (= Position des TCP bezogen auf den Ursprung des ausgewählten Koordinatensystems). **Hinweis**: Wird für Status und Turn der Wert -1 übergeben, so wird die Zielposition auf dem kürzesten Weg angefahren.
Velocity	INT	Geschwindigkeit • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_VEL_CP des Robotersystems. Default: **0 %** (= Geschwindigkeit wird nicht verändert)
Acceleration	INT	Beschleunigung • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_ACC_CP des Robotersystems. Default: **0 %** (= Beschleunigung wird nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Zielposition beziehen (>>> COORDSYS [▶ 20])
Approximate	APO	Überschleifparameter (>>> APO [▶ 19])
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])
SplineMode	BOOL	TRUE = Bewegung wird als Spline-Bewegung ausgeführt. FALSE = Bewegung wird als herkömmliche Punkt-zu-Punkt-Bewegung ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
Aborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.2.5 Achsspezifische Position schnellstmöglich anfahren

Beschreibung

Mit dem Funktionsbaustein KRC_MoveAxisAbsolute wird eine Punkt-zu-Punkt-Bewegung zu einer achsspezifischen Zielposition ausgeführt. Die Achspositionen sind absolut.

> Wenn die Bewegung als Spline-Bewegung ausgeführt wird, muss folgendes beachtet werden: Wenn die Bewegung überschliffen wird und sich keine weitere Bewegungsanweisung im Puffer befindet, wird der Ausgang Done für die Spline-Bewegung nicht gesetzt. Die Bewegung wird in diesem Fall nicht am Zielpunkt beendet, sondern am Überschleifpunkt.

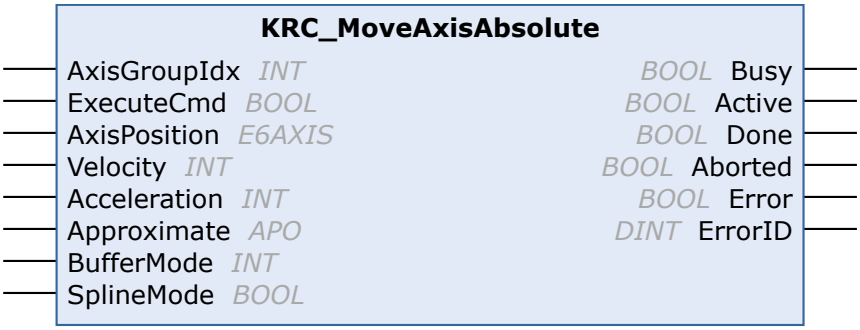


Abb. 5: Funktionsbaustein KRC_MoveAxisAbsolute

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
AxisPosition	E6AXIS	Achsspezifische Zielposition (>>> E6AXIS [▶ 20]) Die Datenstruktur E6Axis enthält die Winkel- oder Translationswerte für alle Achsen der Achsgruppe in der Zielposition.
Velocity	INT	Geschwindigkeit • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Default: **0 %** (= Geschwindigkeit wird nicht verändert)
Acceleration	INT	Beschleunigung • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Default: **0 %** (= Beschleunigung wird nicht verändert)
Approximate	APO	Überschleifparameter (>>> APO [▶ 19])
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])
SplineMode	BOOL	TRUE = Bewegung wird als Spline-Bewegung ausgeführt.

Parameter	Typ	Beschreibung
		FALSE = Bewegung wird als herkömmliche Punkt-zu-Punkt-Bewegung ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
Aborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.2.6 Absolute kartesische Position mit Kreisbewegung anfahren

Beschreibung

Mit dem Funktionsbaustein KRC_MoveCircAbsolute wird eine Kreisbewegung zu einer kartesischen Zielposition ausgeführt. Damit die Robotersteuerung die Kreisbewegung berechnen kann, muss neben der Zielposition eine Hilfsposition angegeben werden.

Die Koordinaten von Hilfs- und Zielposition sind absolut. Die Hilfsposition kann nicht überschliffen werden. Sie wird immer genau angefahren.

Wenn die Bewegung als Spline-Bewegung ausgeführt wird, muss folgendes beachtet werden: Wenn die Bewegung überschliffen wird und sich keine weitere Bewegungsanweisung im Puffer befindet, wird der Ausgang Done für die Spline-Bewegung nicht gesetzt. Die Bewegung wird in diesem Fall nicht am Zielpunkt beendet, sondern am Überschleifpunkt.

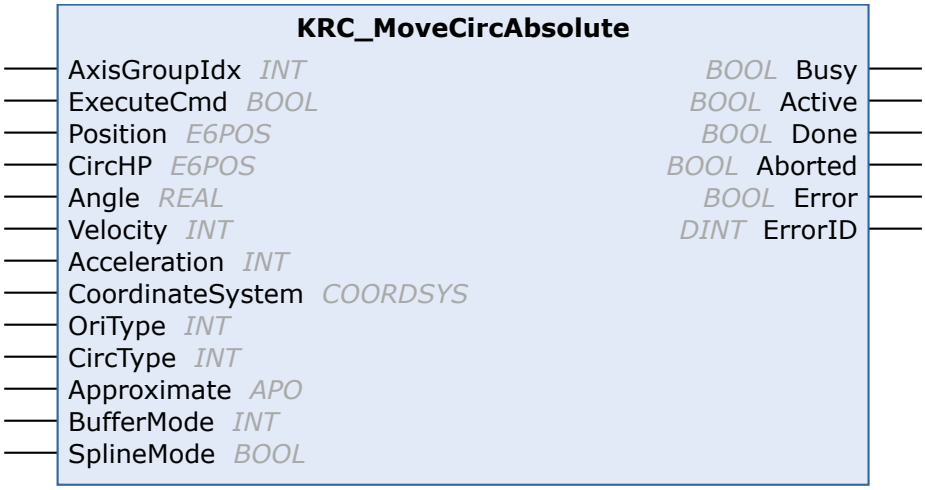


Abb. 6: Funktionsbaustein KRC_MoveCircAbsolute

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
Position	E6POS	Koordinaten der kartesischen Zielposition

Parameter	Typ	Beschreibung
		(>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Zielposition (= Position des TCP bezogen auf den Ursprung des BASE-Koordinatensystems).
CircHP	E6POS	Koordinaten der kartesischen Hilfsposition (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Hilfsposition (= Position des TCP bezogen auf den Ursprung des BASE-Koordinatensystems).
Angle	REAL	Kreiswinkel (= Gesamtwinkel der Kreisbewegung) Der Kreiswinkel ermöglicht eine Verlängerung der Bewegung über den programmierten Zielpunkt hinaus oder auch eine Verkürzung. Der tatsächliche Zielpunkt entspricht dadurch nicht mehr dem programmierten Zielpunkt. Der Kreiswinkel ist nicht begrenzt, d. h. es kann ein Kreiswinkel größer ±360° angegeben werden: • **> 0.0°**: Bei einem positiven Winkel wird vom Startpunkt aus über **CircHP** in Richtung **Position** gefahren. • **< 0.0°**: Bei einem negativen Winkel wird vom Startpunkt aus über **Position** in Richtung **CircHP** gefahren. • **= 0.0°**: Der Kreiswinkel wird ignoriert. Zielposition ist **Position**. Der Kreisradius wird anhand der Startposition, **CircHP** und **Position** berechnet. Default: **0.0°**
Velocity	INT	Geschwindigkeit • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_VEL_CP des Robotersystems. Default: **0 %** (= Geschwindigkeit wird nicht verändert)
Acceleration	INT	Beschleunigung • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_ACC_CP des Robotersystems. Default: **0 %** (= Beschleunigung wird nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Hilfs- oder Zielposition beziehen (>>> COORDSYS [▶ 20])
OriType	INT	Orientierungsführung des TCP • **0**: VAR • **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 16])
CircType	INT	Orientierungsführung während der Kreisbewegung • **0**: BASE • **1**: PATH (>>> CircType [▶ 16])

Parameter	Typ	Beschreibung
Approximate	APO	Überschleifparameter (>>> APO [▶ 19])
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])
SplineMode	BOOL	TRUE = Bewegung wird als Spline-Bewegung ausgeführt. FALSE = Bewegung wird als herkömmliche Kreisbewegung ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
Aborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.2.7 Relative kartesische Position mit Kreisbewegung anfahren

Beschreibung

Mit dem Funktionsbaustein KRC_MoveCircRelative wird eine Kreisbewegung zu einer kartesischen Zielposition ausgeführt. Damit die Robotersteuerung die Kreisbewegung berechnen kann, muss neben der Zielposition eine Hilfsposition angegeben werden.

Die Koordinaten von Hilfs- und Zielposition sind relativ zur aktuellen Position (= Startposition der Kreisbewegung). Die Hilfsposition kann nicht überschliffen werden. Sie wird immer genau angefahren.

Diese Anweisung bezieht sich immer auf die aktuelle Roboterposition. Wenn die Bewegung abgebrochen wurde und wieder ausgeführt wird, fährt der Roboter von der Abbruch-Position aus noch einmal die komplette Strecke.

Wenn die Bewegung als Spline-Bewegung ausgeführt wird, muss folgendes beachtet werden: Wenn die Bewegung überschliffen wird und sich keine weitere Bewegungsanweisung im Puffer befindet, wird der Ausgang Done für die Spline-Bewegung nicht gesetzt. Die Bewegung wird in diesem Fall nicht am Zielpunkt beendet, sondern am Überschleifpunkt.

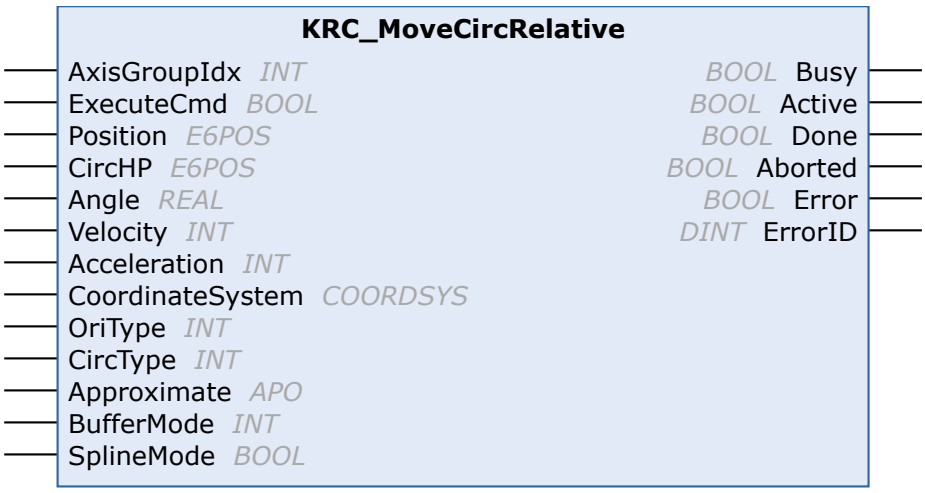


Abb. 7: Funktionsbaustein KRC_MoveCircRelative

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
Position	E6POS	Abstand zwischen Zielposition und aktueller Position. Die Zielposition basiert auf der aktuellen Position. (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Zielposition (= Position des TCP relativ zum Ursprung des BASE-Koordinatensystems).
CircHP	E6POS	Koordinaten der kartesischen Hilfsposition (relativ zur aktuellen Position) (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Hilfsposition (= Position des TCP relativ zum Ursprung des BASE-Koordinatensystems).
Angle	REAL	Kreiswinkel (= Gesamtwinkel der Kreisbewegung) Der Kreiswinkel ermöglicht eine Verlängerung der Bewegung über den programmierten Zielpunkt hinaus oder auch eine Verkürzung. Der tatsächliche Zielpunkt entspricht dadurch nicht mehr dem programmierten Zielpunkt. Der Kreiswinkel ist nicht begrenzt, d. h. es kann ein Kreiswinkel größer ±360° angegeben werden: • **> 0,0°**: Bei einem positiven Winkel wird die Bewegung vom Startpunkt über **CircHP** in Richtung **Position** ausgeführt. • **< 0,0°**: Bei einem negativen Winkel wird die Bewegung vom Startpunkt aus über **Position** in Richtung **CircHP** ausgeführt. • **= 0.0°**: Der Kreiswinkel wird ignoriert. Zielposition ist **Position**. Der Kreisradius wird anhand der Startposition, **CircHP** und **Position** berechnet. Default: **0.0°**
Velocity	INT	Geschwindigkeit • **0 … 100 %**

Parameter	Typ	Beschreibung
		Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_VEL_CP des Robotersystems. Default: **0 %** (= Geschwindigkeit wird nicht verändert)
Acceleration	INT	Beschleunigung • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_ACC_CP des Robotersystems. Default: **0 %** (= Beschleunigung wird nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Hilfs- oder Zielposition beziehen (>>> COORDSYS [▶ 20])
OriType	INT	Orientierungsführung des TCP • **0**: VAR • **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 16])
CircType	INT	Orientierungsführung während der Kreisbewegung • **0**: BASE • **1**: PATH (>>> CircType [▶ 16])
Approximate	APO	Überschleifparameter (>>> APO [▶ 19])
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])
SplineMode	BOOL	TRUE = Bewegung wird als Spline-Bewegung ausgeführt. FALSE = Bewegung wird als herkömmliche Kreisbewegung ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
Aborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.2.8 Zielposition manuell anfahren

Beschreibung

Mit dem Funktionsbaustein KRC_Jog kann eine Zielposition mit einer Linearbewegung oder einer Punkt-zu-Punkt-Bewegung angefahren werden.

Die Funktion wird immer im Modus ABORTING ausgeführt, d. h. alle aktiven Bewegungen und gepufferten Anweisungen werden abgebrochen, der Roboter abgebremst und dann die Bewegung ausgeführt.

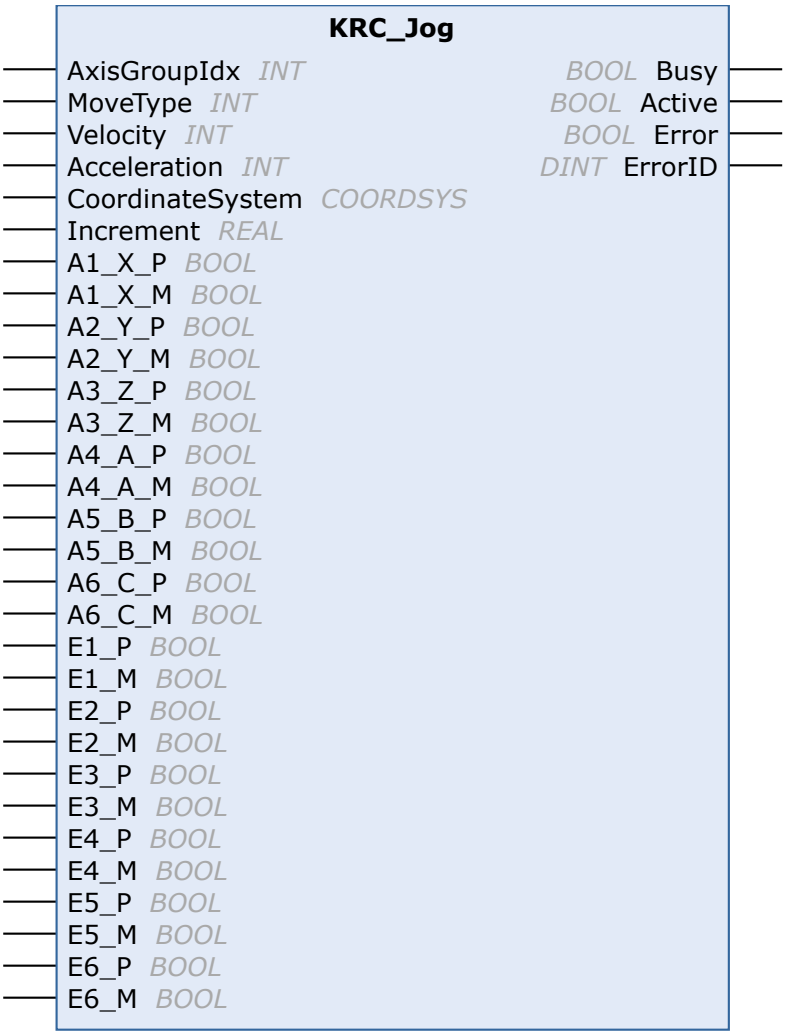


Abb. 8: Funktionsbaustein KRC_Jog

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
MoveType	INT	Bewegungsart für das kartesische oder achsspezifische Verfahren • **0**: Achsspezifisch • **1**: Kartesisch • **2**: Achsspezifisch, als Spline-Bewegung • **3**: Kartesisch, als Spline-Bewegung
Velocity	INT	Geschwindigkeit • **0 … 100 %**

Parameter	Typ	Beschreibung
		Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp. Bei MoveType 1 und 3 bezieht sich der Maximalwert auf den Wert von DEF_VEL_CP des Robotersystems. Default-Wert: **0 %** (= Geschwindigkeit wird nicht verändert)
Acceleration	INT	Beschleunigung • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp. Bei MoveType 1 und 3 bezieht sich der Maximalwert auf den Wert von DEF_ACC_CP des Robotersystems. Default-Wert: **0 %** (= Beschleunigung wird nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die Koordinaten der Zielposition beziehen. (>>> COORDSYS [▶ 20])
Increment	REAL	Inkrementelles Handverfahren Mit diesem Parameter kann die maximale Distanz des inkrementellen Verfahrens begrenzt werden. • **> 0.0**: Roboter fährt maximal die angegebene Distanz. Beim achsspezifischen Verfahren ist die maximale Distanz automatisch auf die Software-Endschalter begrenzt. Beim kartesischen Verfahren in A-, B- oder C-Richtung ist die maximale Distanz auf 90° begrenzt. Wenn das Eingangssignal zurückgesetzt wird, bevor der Roboter die Zielposition erreicht hat, stoppt der Roboter sofort. • **≤ 0.0**: Beim kartesischen Verfahren in X-, Y- oder Z-Richtung ist die maximale Distanz auf 100000 mm begrenzt. Bei jeder Änderung der Eingangssignale wird die Roboterbewegung neu begonnen.
A1_X_P	BOOL	Bewegungsanweisung Die Bewegung wird bei einer steigenden Flanke des Signals gestartet und bei einer fallenden Flanke des Signals gestoppt. Bei MoveType 0 und 2 können die Roboterachsen bis auf 0,1 mm/° vor den Software-Endschaltern verfahren werden. Dazu werden die Werte der Software-Endschalter beim Start der SPS einmalig gelesen. Es können gleichzeitig mehrere Roboterachsen verfahren werden. Der TCP kann entlang der Achsen mehrerer Koordinatensysteme verfahren werden. Bei einer Änderung der Eingangssignale wird die Bewegung gestoppt und anschließend mit der geänderten Konfiguration fortgesetzt. Wenn die positive und negative Verfahrrichtung gleichzeitig aktiviert wird, wird eine Fehlernummer ausgegeben. Die Eingänge für die Achsen A1 … A6 sind je nach MoveType mit den Koordinaten doppelt belegt, z. B. A1 mit X, A2 mit Y usw. Die Eingänge mit der Endung "P" (z. B. **A2_Y_P**) verfahren in die positive Richtung. Die Eingänge mit der Endung "M" (z. B. **A3_Z_M**) verfahren in die negative Richtung.
A1_X_M	BOOL	
A2_Y_P	BOOL	
A2_Y_M	BOOL	
A3_Z_P	BOOL	
A3_Z_M	BOOL	
A4_A_P	BOOL	
A4_A_M	BOOL	
A5_B_P	BOOL	
A5_B_M	BOOL	
A6_C_P	BOOL	
A6_C_M	BOOL	
E1_P	BOOL	
E1_M	BOOL	
E2_P	BOOL	
E2_M	BOOL	
E3_P	BOOL	
E3_M	BOOL	
E4_P	BOOL	
E4_M	BOOL	
E5_P	BOOL	
E5_M	BOOL	

Parameter	Typ	Beschreibung
E6_P	BOOL	
E6_M	BOOL	

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
Aborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.2.9 Verfahren per Tippbetrieb auf eine relative Endposition im TOOL-Koordinatensystem mit einer Linearbewegung

Beschreibung

Mit dem Funktionsbaustein KRC_JogToolRelative kann eine kartesische Zielposition im TOOL-Koordinatensystem mit einer Linearbewegung angefahren werden. Die Koordinaten der Zielposition sind relativ zur aktuellen Position. Status und Turn der Zielposition werden ignoriert, d. h. die Achstellungen sind in der Zielposition nicht eindeutig festgelegt.

Die Funktion wird immer im Modus ABORTING ausgeführt, d. h. alle aktiven Bewegungen und gepufferten Anweisungen werden abgebrochen, der Roboter abgebremst und dann die Linearbewegung ausgeführt.

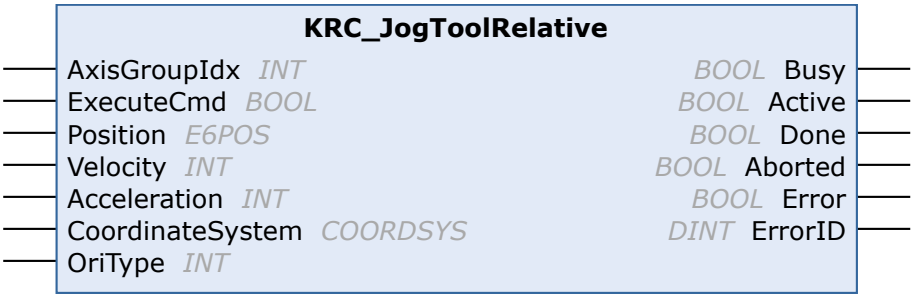


Abb. 9: Funktionsbaustein KRC_JogToolRelative

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/puffert die Bewegung bei steigender Flanke des Signals.
Position	E6POS	Abstand zwischen Zielposition und aktueller Position. Die Zielposition basiert auf der aktuellen Position. (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Endposition (= Position des TCP relativ zum Ursprung des ausgewählten Koordinatensystems).
Geschwindigkeit	INT	Geschwindigkeit • **0 … 100%**

Parameter	Typ	Beschreibung
		Bezieht sich auf den in den Maschinendaten vorgegebenen Maximalwert. Der Maximalwert hängt vom Robotertyp ab und bezieht sich auf den Wert von DEF_VEL_CP im Robotersystem. Standard: **0%** (= Geschwindigkeit nicht verändert)
Acceleration	INT	Acceleration • **0 … 100%** Bezieht sich auf den in den Maschinendaten vorgegebenen Maximalwert. Der Maximalwert hängt vom Robotertyp ab und bezieht sich auf den Wert von DEF_ACC_CP im Robotersystem. Standard: **0%** (= Beschleunigung nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Endposition beziehen (>>> COORDSYS [▶ 20])
OriType	INT	Orientierungssteuerung des TCP • **0**: VAR • **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 16])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
Aborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.2.10 Verfahren per Tippbetrieb auf eine relative kartesische Position mit einer Linearbewegung

Beschreibung

Der Funktionsbaustein KRC_JogLinearRelative kann verwendet werden, um mit einer Linearbewegung in eine kartesische Endposition zu fahren. Die Endposition liegt relativ zur aktuellen Position.

Die Funktion wird immer in der Betriebsart ABORTING ausgeführt, d. h., alle aktiven Bewegungen und gepufferten Anweisungen werden abgebrochen, der Roboter wird abgebremst und anschließend wird die Linearbewegung ausgeführt.

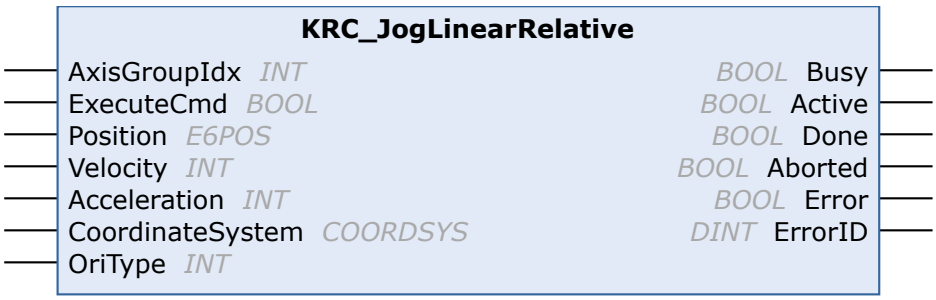


Abb. 10: Funktionsbaustein KRC_JogLinearRelative

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/puffert die Bewegung bei steigender Flanke des Signals.
Position	E6POS	Abstand zwischen Zielposition und aktueller Position. Die Zielposition basiert auf der aktuellen Position. (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Endposition (= Position des TCP relativ zum Ursprung des ausgewählten Koordinatensystems).
Geschwindigkeit	INT	Geschwindigkeit • **0 … 100%** Bezieht sich auf den in den Maschinendaten vorgegebenen Maximalwert. Der Maximalwert hängt vom Robotertyp ab und bezieht sich auf den Wert von DEF_VEL_CP im Robotersystem. Standard: **0%** (= Geschwindigkeit nicht verändert)
Acceleration	INT	Acceleration • **0 … 100%** Bezieht sich auf den in den Maschinendaten vorgegebenen Maximalwert. Der Maximalwert hängt vom Robotertyp ab und bezieht sich auf den Wert von DEF_ACC_CP im Robotersystem. Standard: **0%** (= Beschleunigung nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Endposition beziehen (>>> COORDSYS [▶ 20])
OriType	INT	Orientierungssteuerung des TCP • **0**: VAR • **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 16])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet

Parameter	Typ	Beschreibung
Aborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.2.11 Zielposition manuell anfahren (erweitert)

Beschreibung

Mit dem Funktionsbaustein KRC_JogAdvanced kann eine Zielposition mit einer Linearbewegung oder einer Punkt-zu-Punkt-Bewegung angefahren werden. Die Funktion wird durch den Parameter JogAdvanced aktiviert.

Folgende geänderte Werte werden permanent an die Robotersteuerung übertragen, solange der Parameter JogAdvanced aktiviert ist:

- Werkzeug
- Basis
- Interpolationsmodus
- Bewegungsart

Der Roboter-Interpreter kann keine anderen Berechnungen durchführen, wenn der Ausgang Busy den Wert TRUE hat.

Dieser Funktionsbaustein kann mit Conveyor-Anlagen nicht verwendet werden.

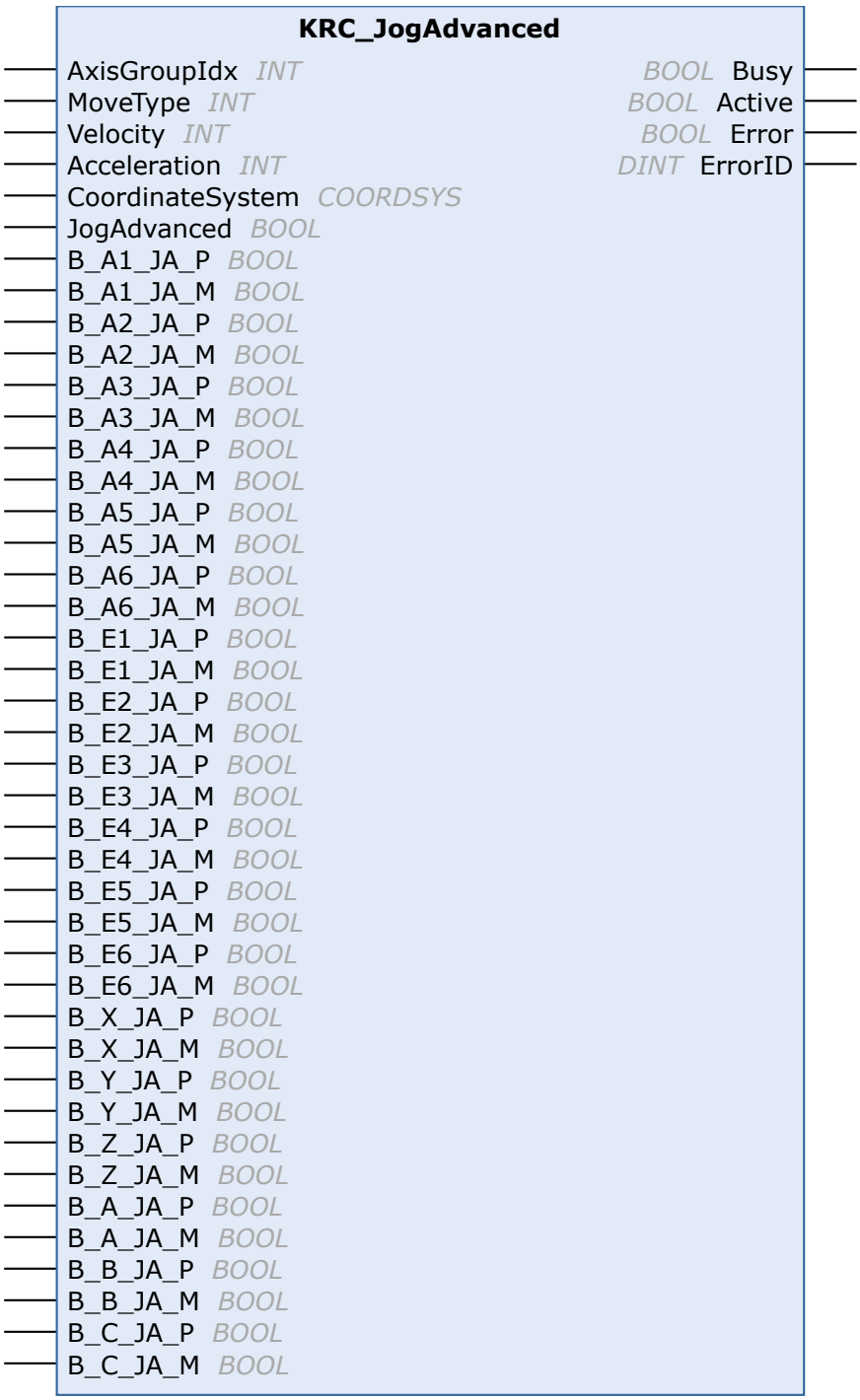


Abb. 11: Funktionsbaustein KRC_JogAdvanced

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
MoveType	INT	Bewegungsart für das kartesische oder achsspezifische Verfahren • **1**: Achsspezifisch als Spline-Bewegung und kartesisch als Spline-Bewegung im BASE-Koordinatensystem • **2**: Achsspezifisch und kartesisch als Spline-Bewegung im TOOL-Koordinatensystem
Velocity	INT	Geschwindigkeit

Parameter	Typ	Beschreibung
		• **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp. Default-Wert: **0 %** (= Geschwindigkeit wird nicht verändert)
Acceleration	INT	Beschleunigung • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp. Default-Wert: **0 %** (= Beschleunigung wird nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die Koordinaten der Zielposition beziehen. (>>> COORDSYS [▶ 20])
JogAdvanced	BOOL	Die Funktion JogAdvanced wird bei einer steigenden Flanke des Signals aktiviert und bei einer fallenden Flanke des Signals deaktiviert.
B_A1_JA_P	BOOL	Bewegungsanweisung Die Bewegungsanweisung wird nur ausgeführt, wenn der Parameter JogAdvanced aktiviert ist. Die Bewegung wird bei einer steigenden Flanke des Signals gestartet und bei einer fallenden Flanke des Signals gestoppt. Bei der achsspezifischen Bewegung können die Roboterachsen bis auf 0,1 mm/° vor den Software-Endschaltern verfahren werden. Dazu werden die Werte der Software-Endschalter beim Start der SPS einmalig gelesen. Der TCP kann entlang der Achsen mehrerer Koordinatensysteme verfahren werden. Bei einer Änderung der Eingangssignale wird die Bewegung gestoppt und anschließend mit der geänderten Konfiguration fortgesetzt. Wenn die positive und negative Verfahrrichtung gleichzeitig aktiviert wird, wird eine Fehlernummer ausgegeben. Die Eingänge mit der Endung "P" (z. B. **B_A2_JA_P**) verfahren in die positive Richtung. Die Eingänge mit der Endung "M" (z. B. **B_A3_JA_M**) verfahren in die negative Richtung.
B_A1_JA_M	BOOL	
B_A2_JA_P	BOOL	
B_A2_JA_M	BOOL	
B_A3_JA_P	BOOL	
B_A3_JA_M	BOOL	
B_A4_JA_P	BOOL	
B_A4_JA_M	BOOL	
B_A5_JA_P	BOOL	
B_A5_JA_M	BOOL	
B_A6_JA_P	BOOL	
B_A6_JA_M	BOOL	
B_E1_JA_P	BOOL	
B_E1_JA_M	BOOL	
B_E2_JA_P	BOOL	
B_E2_JA_M	BOOL	
B_E3_JA_P	BOOL	
B_E3_JA_M	BOOL	
B_E4_JA_P	BOOL	
B_E4_JA_M	BOOL	
B_E5_JA_P	BOOL	
B_E5_JA_M	BOOL	
B_E6_JA_P	BOOL	
B_E6_JA_M	BOOL	
B_X_JA_P	BOOL	
B_X_JA_M	BOOL	
B_Y_JA_P	BOOL	
B_Y_JA_M	BOOL	
B_Z_JA_P	BOOL	
B_Z_JA_M	BOOL	
B_A_JA_P	BOOL	
B_A_JA_M	BOOL	
B_B_JA_P	BOOL	
B_B_JA_M	BOOL	

Parameter	Typ	Beschreibung
B_C_JA_P	BOOL	
B_C_JA_M	BOOL	

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Funktionsbaustein ist aktiv und wartet auf eine Bewegungsanweisung
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.3 Funktionen für die Bewegungsprogrammierung (Konform zu PLC OPEN)

Die nachfolgend beschriebenen MC-Funktionsbausteine unterscheiden sich von den KRC-Funktionsbausteinen darin, dass sie der Norm PLC OPEN entsprechen oder näher kommen.

Mit dem Active-Ausgang ist Überschleifen nicht möglich, da die nächste Bewegungsanweisung erst gesendet wird, wenn die vorherige ausgeführt wird. Überschleifen ist nur möglich, wenn der ComAcpt-Ausgang des vorherigen Funktionsbausteins mit dem Execute-Eingang des folgenden Bausteins verbunden wird.

Informationen zu den häufig verwendeten Signalen in den MC-Funktionsbausteinen siehe (>>> Häufig verwendete Ein-/Ausgangsignale in den MC-Funktionsbausteinen).

7.3.1 Absolute kartesische Position linear anfahren

Beschreibung

Mit dem Funktionsbaustein MC_MoveLinearAbsolute wird eine Linearbewegung zu einer kartesischen Zielposition ausgeführt. Die Koordinaten der Zielposition sind absolut.

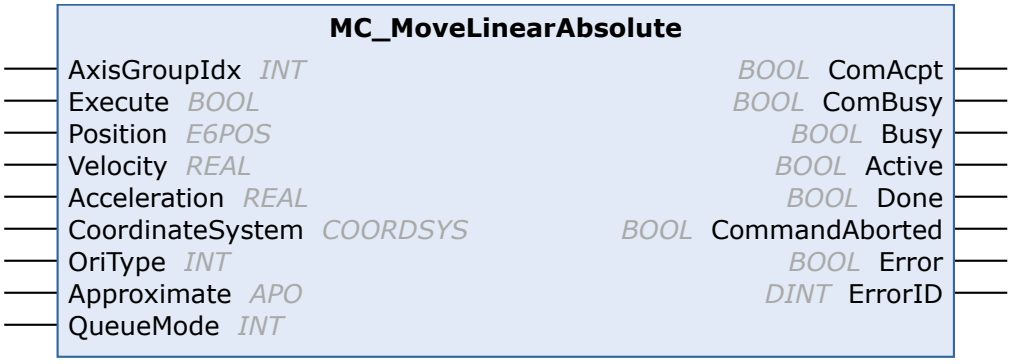


Abb. 12: Funktionsbaustein MC_MoveLinearAbsolute

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**

Parameter	Typ	Beschreibung
Execute	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
Position	E6POS	Koordinaten der kartesischen Zielposition (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Zielposition (= Position des TCP bezogen auf den Ursprung des BASE-Koordinatensystems).
Velocity	REAL	Geschwindigkeit für die Bahnbewegung • **0 … 2 m/s** Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_VEL_CP des Robotersystems. Default: **0 m/s** (= Geschwindigkeit wird nicht verändert)
Acceleration	REAL	Beschleunigung für die Bahnbewegung • 0 … 2.3 m/s^2 Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_ACC_CP des Robotersystems. Default: 0 m/s^2 (= Beschleunigung wird nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Zielposition beziehen (>>> COORDSYS [▶ 20]) **Hinweis**: Bei einer Linearbewegung beziehen sich die kartesischen Koordinaten immer auf das BASE-Koordinatensystem.
OriType	INT	Orientierungsführung des TCP • **0**: VAR • **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 16])
Approximate	APO	Überschleifparameter (>>> APO [▶ 19])
QueueMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> QueueMode [▶ 16])

Ausgänge

Parameter	Typ	Beschreibung
ComAcpt	BOOL	TRUE = Anweisung wurde vollständig übertragen und von der Robotersteuerung bestätigt.
ComBusy	BOOL	TRUE = Anweisung wurde übertragen und von der Robotersteuerung bestätigt, ist jedoch noch nicht vollständig ausgeführt.
Busy	BOOL	TRUE = Funktionsbaustein wurde noch nicht vollständig ausgeführt
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
CommandAborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.3.2 Anfahren einer relativen kartesischen Position mit einer Linearbewegung

Beschreibung

Mit dem Funktionsbaustein MC_MoveLinearRelative wird eine Linearbewegung zu einer relativen kartesischen Zielposition ausgeführt. Der Parameter Position enthält die Strecke von der aktuellen Position zur Zielposition.

> Diese Anweisung bezieht sich immer auf die aktuelle Roboterposition. Wenn die Bewegung abgebrochen wurde und wieder ausgeführt wird, fährt der Roboter von der Abbruch-Position aus noch einmal die komplette Strecke.

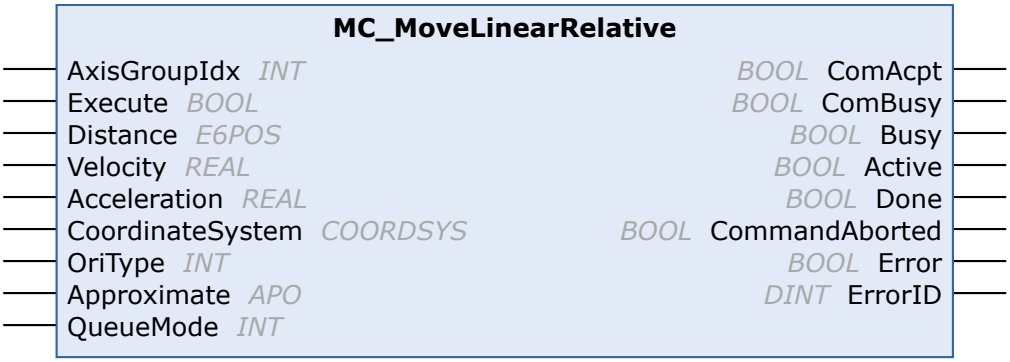


Abb. 13: Funktionsbaustein MC_MoveLinearRelative

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
Execute	BOOL	Startet/puffert die Bewegung bei steigender Flanke des Signals.
Position	E6POS	Abstand zwischen Zielposition und aktueller Position. Die Zielposition basiert auf der aktuellen Position. (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Endposition (= Position des TCP relativ zum Ursprung des ausgewählten Koordinatensystems).
Geschwindigkeit	REAL	Geschwindigkeit der Bahnbewegung • **0 … 2 m/s** Der Maximalwert hängt vom Robotertyp ab und bezieht sich auf den Wert von DEF_VEL_CP im Robotersystem. Standard: **0 m/s** (= Geschwindigkeit nicht verändert)
Acceleration	REAL	Beschleunigung der Bahnbewegung • 0 … 2,3 m/s^2 Der Maximalwert hängt vom Robotertyp ab und bezieht sich auf den Wert von DEF_ACC_CP im Robotersystem. Standard: 0 m/s^2 (= Beschleunigung nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Endposition beziehen (>>> COORDSYS [▶ 20])
OriType	INT	Orientierungssteuerung des TCP • **0**: VAR

Parameter	Typ	Beschreibung
		• **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 16])
Approximate	APO	Näherungsparameter (>>> APO [▶ 19])
QueueMode	INT	Betriebsart, in der die Anweisung ausgeführt wird • **1**: WIRD ABGEBROCHEN • **2**: GEPUFFERT (>>> QueueMode [▶ 16])

Ausgänge

Parameter	Typ	Beschreibung
ComAcpt	BOOL	TRUE = Anweisung wurde vollständig übertragen und von der Robotersteuerung bestätigt.
ComBusy	BOOL	TRUE = Anweisung wurde übertragen und von der Robotersteuerung bestätigt, ist jedoch noch nicht vollständig ausgeführt.
Busy	BOOL	TRUE = Funktionsbaustein wurde noch nicht vollständig ausgeführt
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
CommandAborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.3.3 Absolute kartesische Position schnellstmöglich anfahren

Beschreibung

Mit dem Funktionsbaustein MC_MoveDirectAbsolute wird eine Punkt-zu-Punkt-Bewegung zu einer kartesischen Zielposition ausgeführt. Die Koordinaten der Zielposition sind absolut.

Hierbei bewegt sich der Roboter schnellstmöglich zur Zielposition. Die schnellste Bahn ist in der Regel nicht die kürzeste Bahn und somit keine Gerade. Auf dem Robotersystem entspricht dies einer PTP-Bewegung.

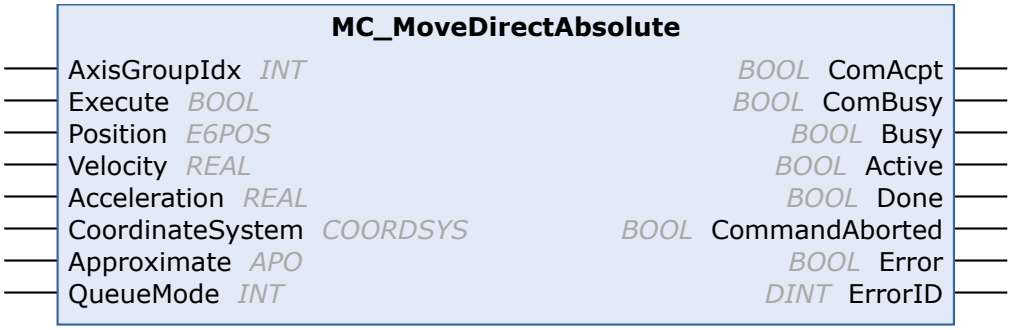


Abb. 14: Funktionsbaustein MC_MoveDirectAbsolute

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**

Parameter	Typ	Beschreibung
Execute	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
Position	E6POS	Koordinaten der kartesischen Zielposition (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Zielposition (= Position des TCP bezogen auf den Ursprung des BASE-Koordinatensystems). **Hinweis**: Wird für Status und Turn der Wert -1 übergeben, so wird die Zielposition auf dem kürzesten Weg angefahren.
Velocity	REAL	Geschwindigkeit für die Bahnbewegung • **0 … 100 %** Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_VEL_CP des Robotersystems. Default: **0 %**
Acceleration	REAL	Beschleunigung für die Bahnbewegung • **0 … 100 %** Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_ACC_CP des Robotersystems. Default: **0 %** (= Geschwindigkeit wird nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Zielposition beziehen (>>> COORDSYS [▶ 20]) **Hinweis**: Bei einer Punkt-zu-Punkt-Bewegung beziehen sich die kartesischen Koordinaten immer auf das BASE-Koordinatensystem.
Approximate	APO	Überschleifparameter (>>> APO [▶ 19])
QueueMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> QueueMode [▶ 16])

Ausgänge

Parameter	Typ	Beschreibung
ComAcpt	BOOL	TRUE = Anweisung wurde vollständig übertragen und von der Robotersteuerung bestätigt.
ComBusy	BOOL	TRUE = Anweisung wurde übertragen und von der Robotersteuerung bestätigt, ist jedoch noch nicht vollständig ausgeführt.
Busy	BOOL	TRUE = Funktionsbaustein wurde noch nicht vollständig ausgeführt
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
CommandAborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.3.4 Relative kartesische Position schnellstmöglich anfahren

Beschreibung

Mit dem Funktionsbaustein MC_MoveDirectRelative wird eine Punkt-zu-Punkt-Bewegung zu einer relativen kartesischen Zielposition ausgeführt. Der Parameter Position enthält die Strecke von der aktuellen Position zur Zielposition. Auf dem Robotersystem entspricht dies einer PTP_REL-Bewegung.

Diese Anweisung bezieht sich immer auf die aktuelle Roboterposition. Wenn die Bewegung abgebrochen wurde und wieder ausgeführt wird, fährt der Roboter von der Abbruch-Position aus noch einmal die komplette Strecke.

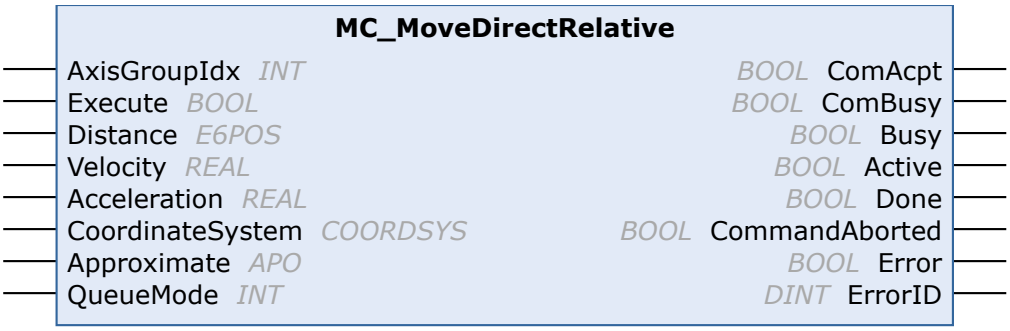


Abb. 15: Funktionsbaustein MC_MoveDirectRelative

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
Execute	BOOL	Startet/Puffert die Bewegung bei einer steigenden Flanke des Signals.
Position	E6POS	Abstand zwischen Zielposition und aktueller Position. Die Zielposition basiert auf der aktuellen Position. (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Zielposition (= Position des TCP bezogen auf den Ursprung des ausgewählten Koordinatensystems). **Hinweis**: Wird für Status und Turn der Wert -1 übergeben, so wird die Zielposition auf dem kürzesten Weg angefahren.
Velocity	REAL	Geschwindigkeit für die Bahnbewegung • **0 … 100%** Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_VEL_CP des Robotersystems. Standard: **0%** (= Geschwindigkeit nicht verändert)
Acceleration	REAL	Beschleunigung für die Bahnbewegung • **0 … 100%** Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_ACC_CP des Robotersystems. Standard: **0%** (= Beschleunigung nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Zielposition beziehen (>>> COORDSYS [▶ 20])
Approximate	APO	Überschleifparameter

Parameter	Typ	Beschreibung
		(>>> APO [▶ 19])
QueueMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> QueueMode [▶ 16])

Ausgänge

Parameter	Typ	Beschreibung
ComAcpt	BOOL	TRUE = Anweisung wurde vollständig übertragen und von der Robotersteuerung bestätigt.
ComBusy	BOOL	TRUE = Anweisung wurde übertragen und von der Robotersteuerung bestätigt, ist jedoch noch nicht vollständig ausgeführt.
Busy	BOOL	TRUE = Funktionsbaustein wurde noch nicht vollständig ausgeführt
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
CommandAborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.3.5 Achsspezifische Position schnellstmöglich anfahren

Beschreibung

Mit dem Funktionsbaustein MC_MoveAxisAbsolute wird eine Punkt-zu-Punkt-Bewegung zu einer achsspezifischen Zielposition ausgeführt. Die Achspositionen sind absolut.

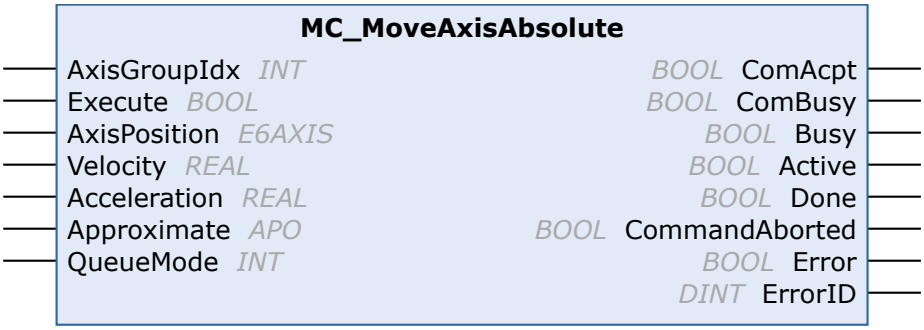


Abb. 16: Funktionsbaustein MC_MoveAxisAbsolute

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
Execute	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
AxisPosition	E6AXIS	Achsspezifische Zielposition (>>> E6AXIS [▶ 20]) Die Datenstruktur E6Axis enthält die Winkel- oder Translationswerte für alle Achsen der Achsgruppe in der Zielposition.
Velocity	REAL	Geschwindigkeit für die Bahnbewegung

Parameter	Typ	Beschreibung
		• **0 … 100 %** Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_VEL_CP des Robotersystems. Default: **0 %** (= Geschwindigkeit wird nicht verändert)
Acceleration	REAL	Beschleunigung für die Bahnbewegung • **0 … 100 %** Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_ACC_CP des Robotersystems. Default: **0 %** (= Beschleunigung wird nicht verändert)
Approximate	APO	Überschleifparameter (>>> APO [▶ 19])
QueueMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> QueueMode [▶ 16])

Ausgänge

Parameter	Typ	Beschreibung
ComAcpt	BOOL	TRUE = Anweisung wurde vollständig übertragen und von der Robotersteuerung bestätigt.
ComBusy	BOOL	TRUE = Anweisung wurde übertragen und von der Robotersteuerung bestätigt, ist jedoch noch nicht vollständig ausgeführt.
Busy	BOOL	TRUE = Funktionsbaustein wurde noch nicht vollständig ausgeführt
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
CommandAborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.3.6 Absolute kartesische Position mit Kreisbewegung anfahren

Beschreibung

Mit dem Funktionsbaustein MC_MoveCircularAbsolute wird eine Kreisbewegung zu einer kartesischen Zielposition ausgeführt. Damit die Robotersteuerung die Kreisbewegung berechnen kann, muss neben der Zielposition eine Hilfsposition angegeben werden.

Die Koordinaten von Hilfs- und Zielposition sind absolut. Die Hilfsposition kann nicht überschliffen werden. Sie wird immer genau angefahren.

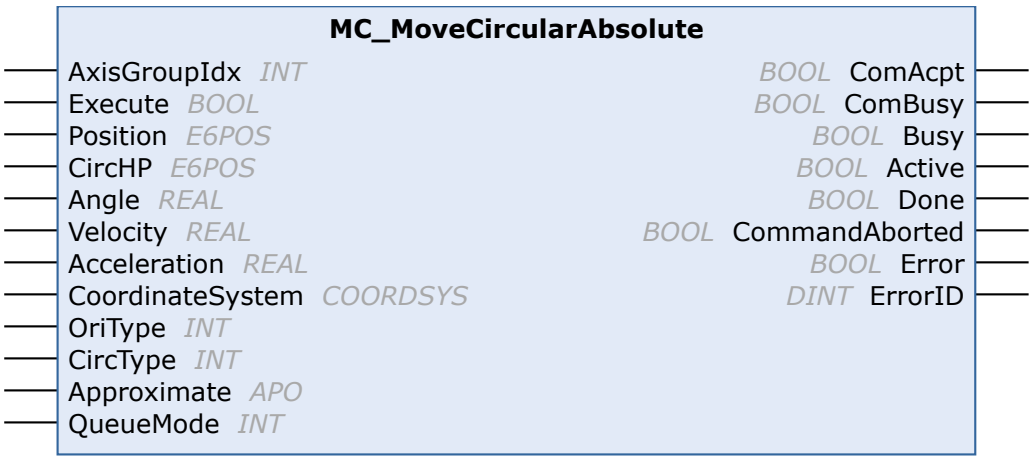


Abb. 17: Funktionsbaustein MC_MoveCircularAbsolute

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
Execute	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
Position	E6POS	Koordinaten der kartesischen Zielposition (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Zielposition (= Position des TCP bezogen auf den Ursprung des BASE-Koordinatensystems).
CircHP	E6POS	Koordinaten der kartesischen Hilfsposition (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Hilfsposition (= Position des TCP bezogen auf den Ursprung des BASE-Koordinatensystems).
Angle	REAL	Kreiswinkel (= Gesamtwinkel der Kreisbewegung) Der Kreiswinkel ermöglicht eine Verlängerung der Bewegung über den programmierten Zielpunkt hinaus oder auch eine Verkürzung. Der tatsächliche Zielpunkt entspricht dadurch nicht mehr dem programmierten Zielpunkt. Der Kreiswinkel ist nicht begrenzt, d. h. es kann ein Kreiswinkel größer ±360° angegeben werden: • **> 0.0°**: Bei einem positiven Winkel wird vom Startpunkt aus über **CircHP** in Richtung **Position** gefahren. • **< 0.0°**: Bei einem negativen Winkel wird vom Startpunkt aus über **Position** in Richtung **CircHP** gefahren. • **= 0.0°**: Der Kreiswinkel wird ignoriert. Zielposition ist **Position**. Der Kreisradius wird anhand der Startposition, **CircHP** und **Position** berechnet. Default: **0.0°**
Velocity	REAL	Geschwindigkeit für die Bahnbewegung • **0 … 2 m/s** Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_VEL_CP des Robotersystems. Default: **0 m/s** (= Geschwindigkeit wird nicht verändert)

Parameter	Typ	Beschreibung
Acceleration	REAL	Beschleunigung für die Bahnbewegung • 0 … 2.3 m/s² Der Maximalwert ist abhängig vom Robotertyp und bezieht sich auf den Wert von DEF_ACC_CP des Robotersystems. Default: 0 m/s² (= Beschleunigung wird nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Hilfs- oder Zielposition beziehen (>>> COORDSYS [▶ 20]) **Hinweis**: Bei einer Kreisbewegung beziehen sich die kartesischen Koordinaten immer auf das BASE-Koordinatensystem.
OriType	INT	Orientierungsführung des TCP • **0**: VAR • **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 16])
CircType	INT	Orientierungsführung während der Kreisbewegung • **0**: BASE • **1**: PATH (>>> CircType [▶ 16])
Approximate	APO	Überschleifparameter (>>> APO [▶ 19])
QueueMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> QueueMode [▶ 16])

Ausgänge

Parameter	Typ	Beschreibung
ComAcpt	BOOL	TRUE = Anweisung wurde vollständig übertragen und von der Robotersteuerung bestätigt.
ComBusy	BOOL	TRUE = Anweisung wurde übertragen und von der Robotersteuerung bestätigt, ist jedoch noch nicht vollständig ausgeführt.
Busy	BOOL	TRUE = Funktionsbaustein wurde noch nicht vollständig ausgeführt
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
CommandAborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.3.7 Anfahren einer relativen kartesischen Position mit einer Kreisbewegung

Beschreibung

Mit dem Funktionsbaustein MC_MoveCircularRelative wird eine Kreisbewegung zu einer kartesischen Zielposition ausgeführt. Damit die Robotersteuerung die Kreisbewegung berechnen kann, muss neben der Zielposition eine Hilfsposition angegeben werden.

Die Koordinaten von Hilfs- und Zielposition sind relativ zur aktuellen Position (= Startposition der Kreisbewegung). Die Hilfsposition kann nicht überschliffen werden. Sie wird immer genau angefahren.

Diese Anweisung bezieht sich immer auf die aktuelle Roboterposition. Wenn die Bewegung abgebrochen wurde und wieder ausgeführt wird, fährt der Roboter von der Abbruch-Position aus noch einmal die komplette Strecke.

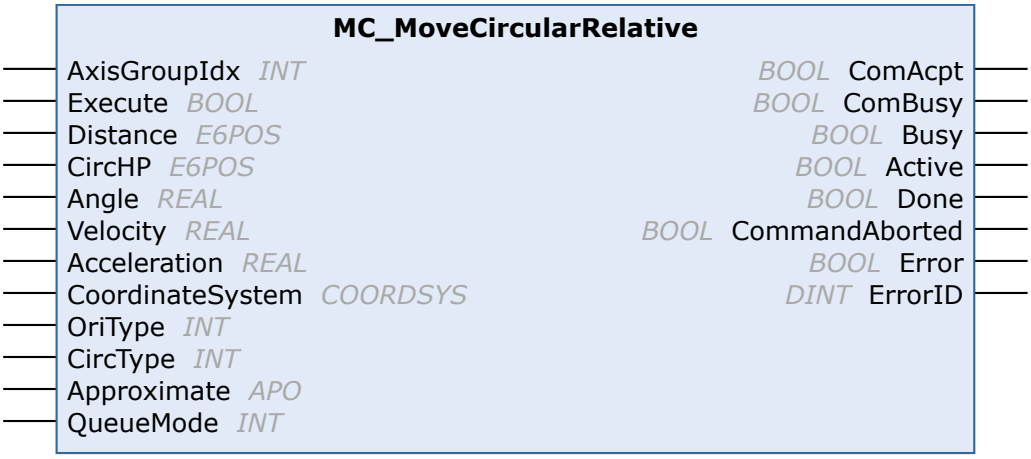


Abb. 18: Funktionsbaustein MC_MoveCircularRelative

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
Execute	BOOL	Startet/puffert die Bewegung bei steigender Flanke des Signals.
Position	E6POS	Abstand zwischen Zielposition und aktueller Position. Die Zielposition basiert auf der aktuellen Position. (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Endposition (= Position des TCP relativ zum Ursprung des BASE-Koordinatensystems).
CircHP	E6POS	Koordinaten der kartesischen Hilfsposition (relativ zur aktuellen Position) (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Hilfsposition (= Position des TCP relativ zum Ursprung des BASE-Koordinatensystems).
Winkel	REAL	Kreiswinkel (= Gesamtwinkel der Kreisbewegung) Der Kreiswinkel ermöglicht es, die Bewegung über den programmierten Endpunkt hinaus zu verlängern oder zu verkürzen. Der tatsächliche Endpunkt entspricht nicht mehr dem programmierten Endpunkt. Der Kreiswinkel ist nicht begrenzt, d. h., ein Kreiswinkel größer als ±360° kann angegeben werden: • **> 0,0°**: Bei einem positiven Winkel wird die Bewegung vom Startpunkt über **CircHP** in Richtung **Position** ausgeführt. • **< 0,0°**: Bei einem negativen Winkel wird die Bewegung vom Startpunkt aus über **Position** in Richtung **CircHP** ausgeführt. • **= 0,0°**: Der Kreiswinkel wird ignoriert. Die Endposition ist **Position**. Der Radius des Kreises wird auf der Basis der Startposition, **CircHP** und **Position** berechnet.

Parameter	Typ	Beschreibung
		Standard: **0,0°**
Geschwindigkeit	REAL	Geschwindigkeit der Bahnbewegung • **0 … 2 m/s** Der Maximalwert hängt vom Robotertyp ab und bezieht sich auf den Wert von DEF_VEL_CP im Robotersystem. Standard: **0 m/s** (= Geschwindigkeit nicht verändert)
Acceleration	REAL	Beschleunigung der Bahnbewegung • 0 … 2,3 m/s^2 Der Maximalwert hängt vom Robotertyp ab und bezieht sich auf den Wert von DEF_ACC_CP im Robotersystem. Standard: 0 m/s^2 (= Beschleunigung nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Hilfs- oder Endposition beziehen (>>> COORDSYS [▶ 20])
OriType	INT	Orientierungssteuerung des TCP • **0**: VAR • **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 16])
CircType	INT	Orientierungssteuerung während der Kreisbewegung • **0**: BASE • **1**: PATH (>>> CircType [▶ 16])
Approximate	APO	Näherungsparameter (>>> APO [▶ 19])
QueueMode	INT	Betriebsart, in der die Anweisung ausgeführt wird • **1**: WIRD ABGEBROCHEN • **2**: GEPUFFERT (>>> QueueMode [▶ 16])

Ausgänge

Parameter	Typ	Beschreibung
ComAcpt	BOOL	TRUE = Anweisung wurde vollständig übertragen und von der Robotersteuerung bestätigt.
ComBusy	BOOL	TRUE = Anweisung wurde übertragen und von der Robotersteuerung bestätigt, ist jedoch noch nicht vollständig ausgeführt.
Busy	BOOL	TRUE = Funktionsbaustein wurde noch nicht vollständig ausgeführt
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
CommandAborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

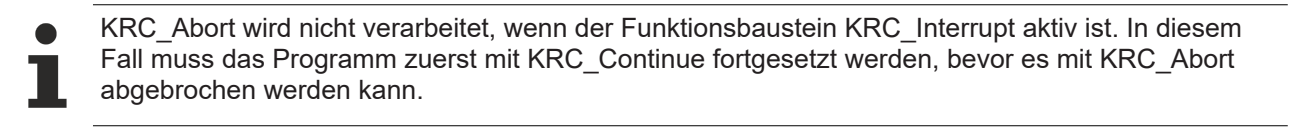
7.4 Funktionen zur Programmablaufkontrolle

7.4.1 Programm abbrechen

Beschreibung

Mit dem Funktionsbaustein KRC_Abort werden aktive und gepufferte Anweisungen und Bewegungen abgebrochen. Der Parameter Active gibt an, ob die Anweisung aktuell noch ausgeführt wird oder nicht. Falls der Roboter-Interpreter nicht mehr aktiv ist, kann dies dazu führen, dass die Anweisung zwar übertragen, aber noch nicht vollständig ausgeführt wurde.

Nicht abgebrochen werden die Anweisungen und Bewegungen von Funktionsbausteinen ohne BufferMode oder QueueMode und die zyklisch ausgeführt werden.



KRC_Abort wird nicht verarbeitet, wenn der Funktionsbaustein KRC_Interrupt aktiv ist. In diesem Fall muss das Programm zuerst mit KRC_Continue fortgesetzt werden, bevor es mit KRC_Abort abgebrochen werden kann.

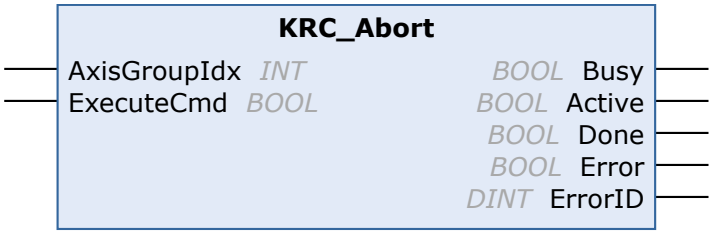


Abb. 19: Funktionsbaustein KRC_Abort

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Anweisung wird aktuell ausgeführt
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.4.2 Programm abbrechen (erweitert)

Beschreibung

Mit dem Funktionsbaustein KRC_AbortAdvanced werden aktive und gepufferte Anweisungen und Bewegungen abgebrochen. Der Parameter Active gibt an, ob die Anweisung aktuell noch ausgeführt wird oder nicht. Falls der Roboter-Interpreter nicht mehr aktiv ist, kann dies dazu führen, dass die Anweisung zwar übertragen, aber noch nicht vollständig ausgeführt wurde.

Nicht abgebrochen werden die Anweisungen und Bewegungen von Funktionsbausteinen ohne BufferMode oder QueueMode und die zyklisch ausgeführt werden.

Mit dem Parameter BrakeReaction kann die Bremsreaktion des Roboters festgelegt werden.

KRC_AbortAdvanced wird nicht verarbeitet, wenn der Funktionsbaustein KRC_Interrupt aktiv ist. In diesem Fall muss das Programm zuerst mit KRC_Continue fortgesetzt werden, bevor es mit KRC_AbortAdvanced abgebrochen werden kann.

Abb. 20: Funktionsbaustein KRC_AbortAdvanced

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
BrakeReaction	INT	Bremsreaktion des Roboters • **1**: BRAKE • **2**: BRAKE F (Default) • **3**: BRAKE FF **Hinweis**: Weitere Informationen zu den BRAKE-Anweisungen sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Anweisung wird aktuell ausgeführt
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.4.3 Roboterbewegung pausieren

Beschreibung

Mit dem Funktionsbaustein KRC_Interrupt wird der Roboter angehalten. Dabei bremst er schonend (BRAKE) oder schnellstmöglich (BRAKE F) aus hohen Geschwindigkeiten ab.

Wenn eine BRAKE-Anweisung aktiv ist, werden keine Anweisungen mehr über die mxA-Schnittstelle verarbeitet. Auch der Funktionsbaustein KRC_Abort wird nicht mehr verarbeitet. KRC_Abort kann das Programm erst dann abbrechen, wenn es mit KRC_Continue fortgesetzt wurde, also die BRAKE-Anweisung nicht mehr aktiv ist. Während die BRAKE-Anweisung aktiv ist, kann das Programm nur durch einen RESET am Funktionsbaustein KRC_AutomaticExternal abgebrochen werden.

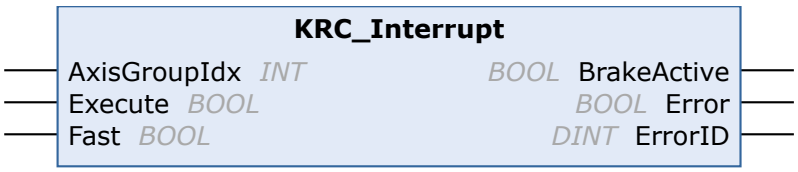


Abb. 21: Funktionsbaustein KRC_Interrupt

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
Execute	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt. Das Roboterprogramm ist unterbrochen, solange der Eingang **Execute** TRUE ist.
Fast	BOOL	TRUE = Roboter stoppt schnellstmöglich FALSE = Roboter stoppt schonend

Ausgänge

Parameter	Typ	Beschreibung
BrakeActive	BOOL	TRUE = Anweisung ist aktiv und Roboter wartet auf Freigabe
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.4.4 Programm fortsetzen

Beschreibung

Mit dem Funktionsbaustein KRC_Continue kann ein Programm, das durch einen Interrupt unterbrochen wurde, fortgesetzt werden.

Wenn KRC_Continue zusammen mit KRC_Interrupt verwendet wird, muss der Eingang Execute bei KRC_Interrupt den Wert FALSE haben, bevor KRC_Continue ausgeführt werden kann.

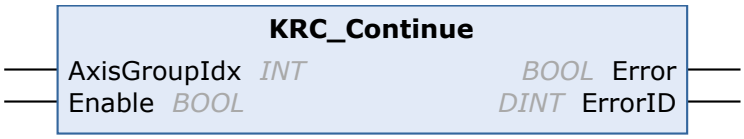


Abb. 22: Funktionsbaustein KRC_Continue

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
Enable	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Error	BOOL	TRUE = Fehler im Funktionsbaustein

Parameter	Typ	Beschreibung
ErrorID	DINT	Fehlernummer

7.4.5 Auf digitalen Eingang warten

Beschreibung

Mit dem Funktionsbaustein KRC_WaitForInput wird das Programm angehalten bis ein digitaler Eingang einen definierten Wert annimmt. Danach wird das Programm fortgesetzt.

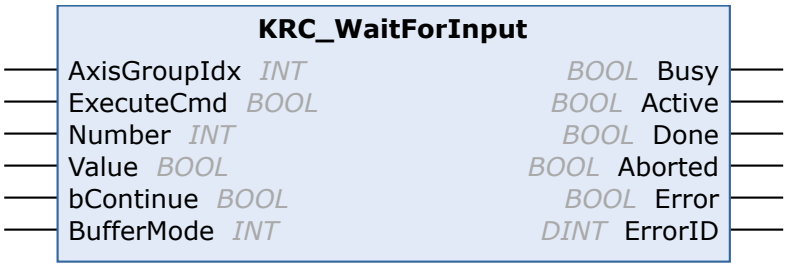


Abb. 23: Funktionsbaustein KRC_WaitForInput

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
Number	INT	Nummer des digitalen Eingangs (entspricht $IN[1 … 2048] auf der Robotersteuerung) • **1 … 2 048**
Value	BOOL	Wert des digitalen Eingangs
bContinue	BOOL	TRUE = Eingang im Vorlauf abfragen **Hinweis**: Die Robotersteuerung arbeitet Programme mit Vor- und Hauptlauf ab. Weitere Informationen zum Vor- und Hauptlauf sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Anweisung wird aktuell ausgeführt (Roboter wartet auf Eingang)
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.5 Funktionen zur Interrupt-Programmierung

7.5.1 Interrupt deklarieren

Beschreibung

Mit dem Funktionsbaustein KRC_DeclareInterrupt wird ein Interrupt auf einen digitalen Eingang deklariert. Hierfür stehen 8 vordefinierte Interrupts zur Verfügung.

Mit einem Interrupt kann der Roboter während der Bewegung angehalten werden. Je nachdem, wie der Parameter Reaction konfiguriert ist, bremst der Roboter schonend von hohen Geschwindigkeiten (BRAKE) oder schnellstens (BRAKE F). Der weitere Programmverlauf kann mit einem Eingang des Roboters oder mit einem Funktionsbaustein der SPS festgelegt werden.

Wenn eine BRAKE-Anweisung aktiv ist, werden keine Anweisungen mehr über die mxA-Schnittstelle verarbeitet. Auch der Funktionsbaustein KRC_Abort wird nicht mehr verarbeitet. KRC_Abort kann das Programm erst dann abbrechen, wenn es mit KRC_Continue fortgesetzt wurde, also die BRAKE-Anweisung nicht mehr aktiv ist. Während die BRAKE-Anweisung aktiv ist, kann das Programm nur durch einen RESET am Funktionsbaustein KRC_AutomaticExternal abgebrochen werden.

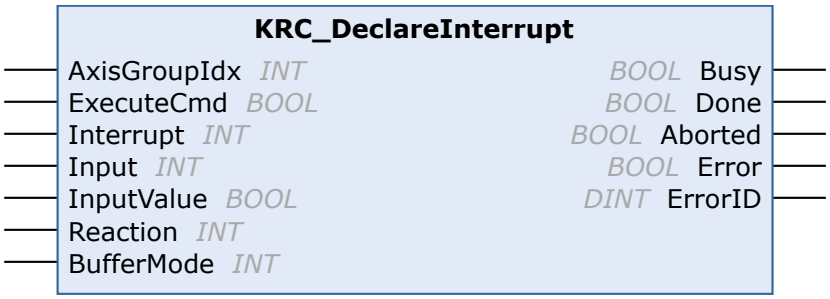


Abb. 24: Funktionsbaustein KRC_DeclareInterrupt

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals gepuffert.
Interrupt	INT	Nummer des Interrupts • **1 … 8** **Hinweis**: Nummer 1 ist der Interrupt mit der höchsten Priorität.
Input	INT	Nummer des digitalen Eingangs, auf den der Interrupt deklariert wird • **1 … 2 048** **Hinweis**: Es ist darauf zu achten, dass keine Eingänge verwendet werden, die bereits vom System belegt sind. Beispiel: $IN[1025] ist immer TRUE.
InputValue	BOOL	TRUE = Anweisung wird bei einer steigenden Flanke des Signals ausgeführt FALSE = Anweisung wird bei einer fallenden Flanke des Signals ausgeführt
Reaction	INT	Reaktion auf den Interrupt • **0**: Schnell bremsen und warten auf den Parameter EXT_START des Funktionsbausteins KRC_AutomaticExternal

Parameter	Typ	Beschreibung
		• **1**: Normal bremsen und warten auf den Parameter EXT_START des Funktionsbausteins KRC_AutomaticExternal • **2**: Schnell bremsen und warten auf den Parameter Input • **3**: Normal bremsen und warten auf den Parameter Input • **4**: Schnell bremsen und warten auf die Flanke des Funktionsbausteins KRC_Continue • **5**: Normal bremsen und warten auf die Flanke des Funktionsbausteins KRC_Continue • **6**: Schnell bremsen und warten auf die Flanke des Funktionsbausteins KRC_Continue und auf den Parameter Input • **7**: Normal bremsen und warten auf die Flanke des Funktionsbausteins KRC_Continue und auf den Parameter Input
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde bearbeitet **Hinweis**: Die Anweisung kann nicht mehr abgebrochen werden. Ausnahme: Programm wird abgewählt oder zurückgesetzt.
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.5.2 Interrupt aktivieren

Beschreibung

Mit dem Funktionsbaustein KRC_ActivateInterrupt wird ein zuvor deklarierter Interrupt aktiviert. Hierfür stehen 8 vordefinierte Interrupts zur Verfügung.

Über den Funktionsbaustein KRC_ReadInterruptState kann abgefragt und überprüft werden, ob ein Interrupt aktiv ist.

Abb. 25: Funktionsbaustein KRC_ActivateInterrupt

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe

Parameter	Typ	Beschreibung
		• **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals gepuffert.
Interrupt	INT	Nummer des Interrupts • **1 … 8**
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde bearbeitet
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.5.3 Interrupt deaktivieren

Beschreibung

Mit dem Funktionsbaustein KRC_DeactivateInterrupt wird ein zuvor deklarierter Interrupt deaktiviert. Hierfür stehen 8 vordefinierte Interrupts zur Verfügung.

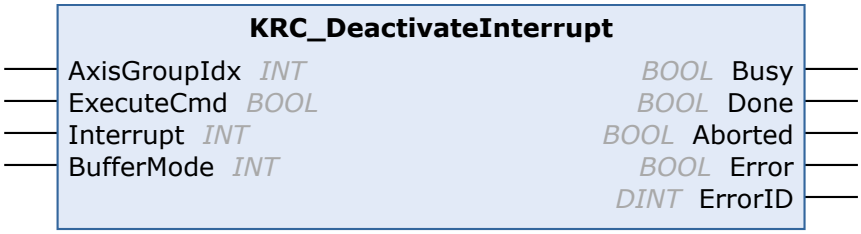


Abb. 26: Funktionsbaustein KRC_DeactivateInterrupt

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals gepuffert.
Interrupt	INT	Nummer des Interrupts • **1 … 8**
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.5.4 Status eines Interrupts lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadInterruptState wird der Status eines Interrupts gelesen. Dieser wird zyklisch aktualisiert.

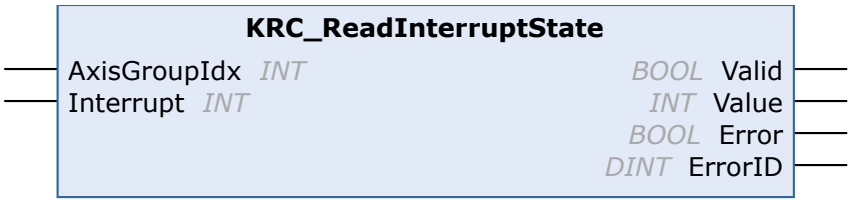


Abb. 27: Funktionsbaustein KRC_ReadInterruptState

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
Interrupt	INT	Nummer des Interrupts • **1 … 8**

Ausgänge

Parameter	Typ	Beschreibung
Valid	BOOL	TRUE = Daten sind gültig
Value	INT	Status des angegebenen Interrupts • **0**: Interrupt wurde nicht deklariert. • **1**: Interrupt wurde deklariert. • **2**: Interrupt wurde deklariert und aktiviert. • **3**: Interrupt wurde deklariert, aktiviert und ist jetzt wieder deaktiviert (siehe Status 1). • **4**: Interrupt wurde ausgelöst und ist aktiv. • **5**: Interrupt wurde ausgelöst und das Hauptprogramm mit dem Funktionsbaustein KRC_Continue bereits wieder fortgesetzt.
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.6 Funktionen für bahnbezogene Schaltaktionen

7.6.1 Schaltaktion zu Bahnpunkten aktivieren

Beschreibung

Mit dem Funktionsbaustein KRC_SetDistanceTrigger wird eine bahnbezogene Schaltaktion bei PTP- oder LIN-Bewegungen ausgelöst.

Der Trigger löst eine definierte Anweisung aus. Die Anweisung bezieht sich auf den Start- oder auf den Zielpunkt des Bewegungssatzes. Die Anweisung wird parallel zur Roboterbewegung ausgeführt.

Es ist möglich, die Anweisung zeitlich zu verschieben. Sie wird dann nicht genau am Start- oder Zielpunkt ausgelöst, sondern früher oder verzögert.

Weiterführende Informationen zu Triggern, zur Verschiebung des Schaltpunkts und zu den Grenzen für die Verschiebung sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.

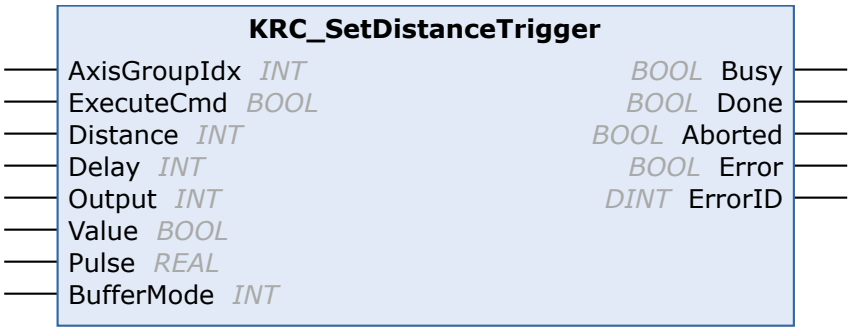


Abb. 28: Funktionsbaustein KRC_SetDistanceTrigger

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals gepuffert.
Distance	INT	Schaltpunkt des Triggers • **0**: Schaltaktion im Startpunkt • **1**: Schaltaktion im Zielpunkt
Delay	INT	Zeitliche Verschiebung der Anweisung • **Delay = 0 ms**: Keine zeitliche Verschiebung Die Anweisung kann nicht beliebig zeitlich verschoben werden. Welche Verschiebungen möglich sind, ist abhängig davon, welcher Wert für **Distance** gewählt wurde. Weitere Informationen hierzu sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.
Output	INT	Nummer des digitalen Ausgangs, der bei der Schaltaktion gesetzt werden kann • **1 … 2 048** **Hinweis**: Es ist darauf zu achten, dass keine Ausgänge verwendet werden, die bereits vom System belegt sind. Beispiel: $OUT[1025] ist immer TRUE.
Value	BOOL	TRUE = Ausgang einschalten

Parameter	Typ	Beschreibung
		FALSE = Ausgang ausschalten
Pulse	REAL	Länge des Impulses • **0.0 s** Kein Puls aktiv • **0.1 … 3.0 s** Pulsraster = 0.1 s
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde bearbeitet **Hinweis**: Die Anweisung kann nicht mehr abgebrochen werden. Ausnahme: Programm wird abgewählt oder zurückgesetzt. Das Signal gibt keinen Aufschluss darüber, ob die Schaltaktion tatsächlich ausgelöst wurde.
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.6.2 Bahnbezogene Schaltaktion aktivieren

Beschreibung

Mit dem Funktionsbaustein KRC_SetPathTrigger wird eine bahnbezogene Schaltaktion bei CP-Bewegungen ausgelöst.

Der Trigger löst eine definierte Anweisung aus. Die Anweisung bezieht sich auf den Zielpunkt des Bewegungssatzes. Die Anweisung wird parallel zur Roboterbewegung ausgeführt.

Es ist möglich, die Anweisung örtlich und/oder zeitlich zu verschieben. Sie wird dann nicht genau am Zielpunkt ausgelöst, sondern vorher oder nachher.

Path-Trigger können nur vor CP-Bewegungen aktiviert werden. Ist die nachfolgende Bewegung keine CP-Bewegung, gibt die Robotersteuerung eine Fehlermeldung aus.

Weiterführende Informationen zu Triggern, zur Verschiebung des Schaltpunkts und zu den Grenzen für die Verschiebung sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.

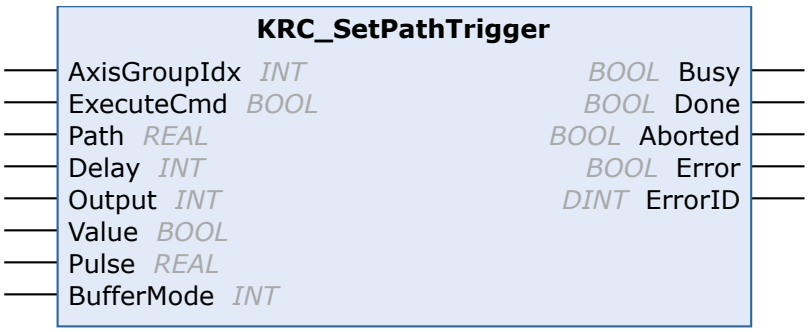


Abb. 29: Funktionsbaustein KRC_SetPathTrigger

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals gepuffert.
Path	REAL	Örtliche Verschiebung der Anweisung Wenn die Anweisung örtlich verschoben werden soll, muss hier die gewünschte Entfernung zum Zielpunkt angegeben werden. Wenn der Zielpunkt überschliffen ist, dann ist **Path** die Entfernung zu derjenigen Position auf dem Überschleifbogen, die dem Zielpunkt am nächsten liegt. • **Path = 0.0 mm**: Keine örtliche Verschiebung • **Path > 0.0 mm**: Verschiebt die Anweisung in Richtung Bewegungsende • **Path < 0.0 mm**: Verschiebt die Anweisung in Richtung Bewegungsanfang
Delay	INT	Zeitliche Verschiebung der Anweisung • **Delay = 0 ms**: Keine zeitliche Verschiebung Die Anweisung kann nicht beliebig zeitlich verschoben werden. Welche Verschiebungen möglich sind, ist abhängig davon, welcher Wert für **Path** gewählt wurde. Weitere Informationen hierzu sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.
Output	INT	Nummer des digitalen Ausgangs, der bei der Schaltaktion gesetzt werden kann • **1 … 2 048** **Hinweis**: Es ist darauf zu achten, dass keine Ausgänge verwendet werden, die bereits vom System belegt sind. Beispiel: $OUT[1025] ist immer TRUE.
Value	BOOL	TRUE = Ausgang einschalten FALSE = Ausgang ausschalten
Pulse	REAL	Länge des Impulses • **0.0 s** Kein Puls aktiv • **0.1 … 3.0 s** Pulsraster = 0.1 s
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED

Parameter	Typ	Beschreibung
		(>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde bearbeitet **Hinweis**: Die Anweisung kann nicht mehr abgebrochen werden. Ausnahme: Programm wird abgewählt oder zurückgesetzt. Das Signal gibt keinen Aufschluss darüber, ob die Schaltaktion tatsächlich ausgelöst wurde.
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.7 Diagnose-Funktionen

7.7.1 Fehlerzustände lesen und quittieren

Beschreibung

Mit dem Funktionsbaustein KRC_Error wird der aktuelle Fehlerzustand der mxA-Schnittstelle, der Fehlerzustand der Robotersteuerung und der Fehlerzustand der Funktionsbausteine gesammelt gelesen und quittiert.

Wenn mehrere Fehler gleichzeitig im Funktionsbaustein aufgetreten sind, wird nur die Fehlernummer angezeigt, die zuletzt aufgetreten ist. Fehler in einem Funktionsbaustein führen zum Entzug der Fahrfreigabe.

Wenn mehrere Fehler gleichzeitig aufgetreten sind, werden diese mit folgender Priorität angezeigt:

1. Fehler der mxA-Schnittstelle im Roboter-Interpreter
2. Fehler der mxA-Schnittstelle im Submit-Interpreter
3. ProConOS-Fehler
4. Fehler in der SPS
5. Fehler in einem Funktionsbaustein der lokalen SPS
6. Fehler der Robotersteuerung

Im Funktionsbaustein KRC_Error sind alle Diagnose-Funktionsbausteine enthalten, dadurch zeigt dieser Baustein alle wichtigen Diagnosedaten an.

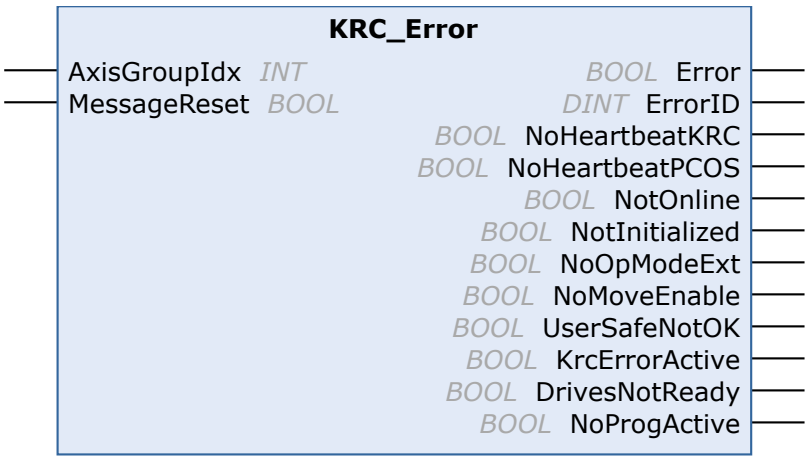


Abb. 30: Funktionsbaustein KRC_Error

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
MessageReset	BOOL	Quittiert Fehlermeldungen der mxA-Schnittstelle und der Robotersteuerung TRUE = Meldung quittieren **Hinweis**: Die Meldungen können nur quittiert werden, wenn der Roboter stillsteht.

Ausgänge

Parameter	Typ	Beschreibung
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	INT	Fehlernummer
NoHeartbeatKRC	BOOL	Submit-Interpreter sendet kein Lebenszeichen
NoHeartbeatPCOS	BOOL	ProConOS sendet kein Lebenszeichen
NotOnline	BOOL	Keine Verbindung zur Robotersteuerung
NotInitialized	BOOL	Es können keine Anweisungen ausgeführt werden, da die Verbindung nicht initialisiert wurde.
NoOpModeExt	BOOL	Roboter steht nicht in der Betriebsart Automatik Extern
NoMoveEnable	BOOL	Keine Fahrfreigabe vorhanden.
UserSafeNotOK	BOOL	Der Bedienerschutz ist verletzt. Das Signal $USER_SAF der Schnittstelle Automatik Extern ist nicht aktiv.
KrcErrorActive	BOOL	Fehlermeldungen der Robotersteuerung sind aktiv. Das Signal $STOPMESS der Schnittstelle Automatik Extern ist aktiv.
DrivesNotReady	BOOL	Die Antriebe sind nicht bereit. Das Signal $PERI_RDY der Schnittstelle Automatik Extern ist nicht aktiv.
NoProgActive	BOOL	Das Roboterprogramm ist nicht aktiv. Das Signal $PRO_ACT der Schnittstelle Automatik Extern ist nicht aktiv.

7.7.2 Aktuellen Status der mxA-Schnittstelle lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadMXAStatus wird der aktuelle Status der mxA-Schnittstelle gelesen.

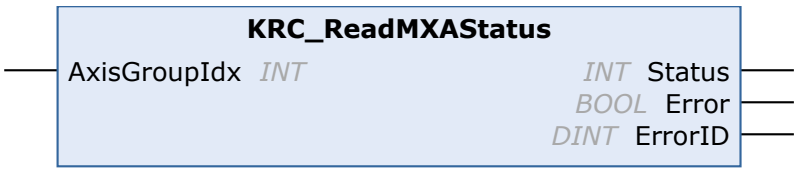


Abb. 31: Funktionsbaustein KRC_ReadMXAStatus

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**

Ausgänge

Parameter	Typ	Beschreibung
Status	INT	Aktueller Status der mxA-Schnittstelle (>>> Status [▶ 76])
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

Status

Aktueller Status der mxA-Schnittstelle (Funktionsbaustein KRC_ReadMXAStatus)

Wert	Name	Beschreibung
0	Invalid	Es können keine Funktionsbausteine verarbeitet werden. Häufige Ursachen: • Submit-Interpreter gestoppt oder abgewählt • E/A-Fehler wegen fehlerhafter Buskonfiguration • Robotersteuerung nicht gestartet
1	Error	Eine mxA-Fehlermeldung ist aktiv. Die Fehlermeldung muss mit dem Funktionsbaustein KRC_MessageReset quittiert werden.
2	ProgramStopped	Roboter-Interpreter ist nicht aktiv (Hauptprogramm wurde gestoppt oder abgewählt).
3	StandBy	Roboter-Interpreter ist aktiv und wartet auf Anweisungen, z. B. Warten auf einen Eingang.
4	Executing	Roboter-Interpreter ist aktiv (Hauptprogramm wird abgearbeitet).
5	Aborting	Roboter wurde angehalten und alle Anweisungen abgebrochen.

7.7.3 Fehlermeldungen der mxA-Schnittstelle lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadMXAError wird der aktuelle Fehlerzustand einer Achsgruppe gelesen. Es werden nur Fehlermeldungen angezeigt, die von der mxA-Schnittstelle generiert wurden.

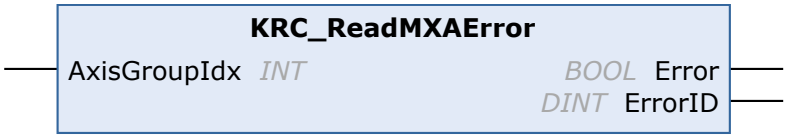


Abb. 32: Funktionsbaustein KRC_ReadMXAError

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**

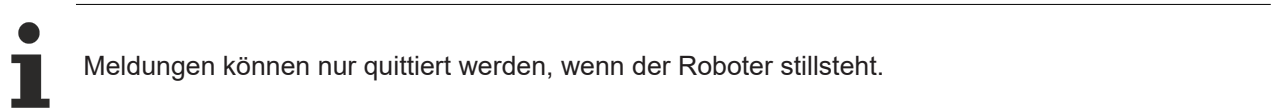
Ausgänge

Parameter	Typ	Beschreibung
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.7.4 Fehlermeldungen der mxA-Schnittstelle quittieren

Beschreibung

Mit dem Funktionsbaustein KRC_MessageReset wird der aktuelle Fehlerzustand einer Achsgruppe quittiert. Es werden nur Fehlermeldungen quittiert, die von der mxA-Schnittstelle generiert wurden.



Meldungen können nur quittiert werden, wenn der Roboter stillsteht.

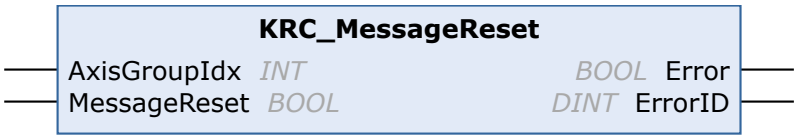


Abb. 33: Funktionsbaustein KRC_MessageReset

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
MessageReset	BOOL	TRUE = Meldung quittieren

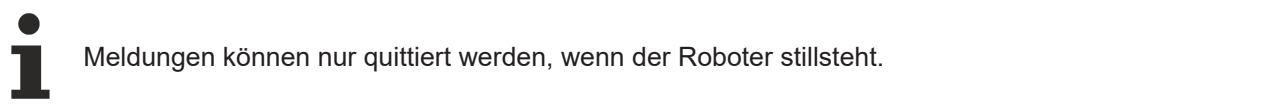
Ausgänge

Parameter	Typ	Beschreibung
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.7.5 Fehlermeldungen der Robotersteuerung lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadKRCError wird der aktuelle Fehlerzustand der Robotersteuerung gelesen. Es werden nur Fehlermeldungen angezeigt, die von der Robotersteuerung generiert wurden.



Meldungen können nur quittiert werden, wenn der Roboter stillsteht.

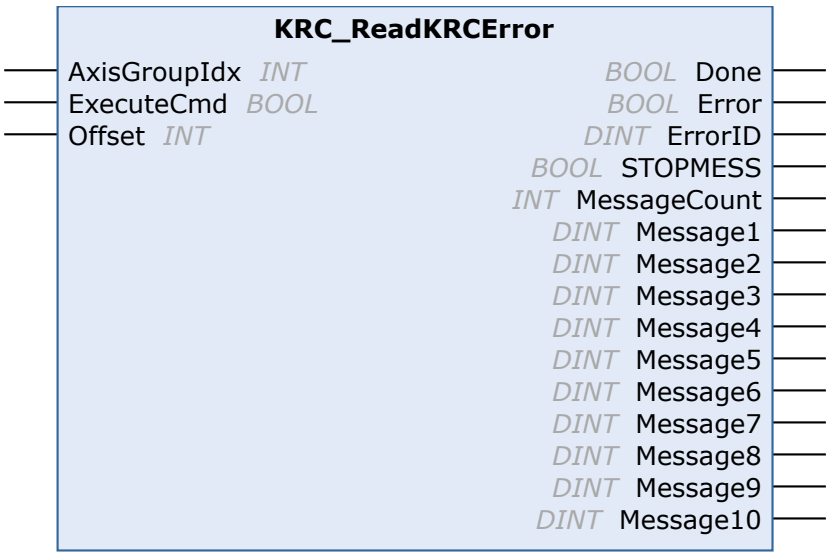


Abb. 34: Funktionsbaustein KRC_ReadKRCError

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
Offset	INT	Wenn mehr als 10 Meldungen im Meldungspuffer stehen, kann über den Offset der gewünschte Startindex des Meldungspuffers ausgewählt werden. **Beispiel**: Wenn 15 Meldungen im Meldungspuffer stehen, muss der Offset 6 sein, damit die Meldungen 6 bis 15 gelesen werden.

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Daten sind gültig
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer
STOPMESS	BOOL	TRUE = Sicherheitskreis ist unterbrochen (Roboterfehler)
MessageCount	INT	Anzahl der Meldungen im Meldungspuffer
Message1 … Message10	DINT	Die Nummern von bis zu 10 Meldungen, die im Meldungspuffer stehen, können ausgegeben werden.

7.7.6 Diagnosesignale lesen

Beschreibung

Mit dem Funktionsbaustein KRC_Diag werden Diagnosesignale der Robotersteuerung gelesen.

Der Funktionsbaustein darf pro Achsgruppe nur einfach instanziiert werden. Bei einer mehrfachen Instanziierung werden die Signale des zuletzt aufgerufenen Funktionsbausteins ausgegeben.

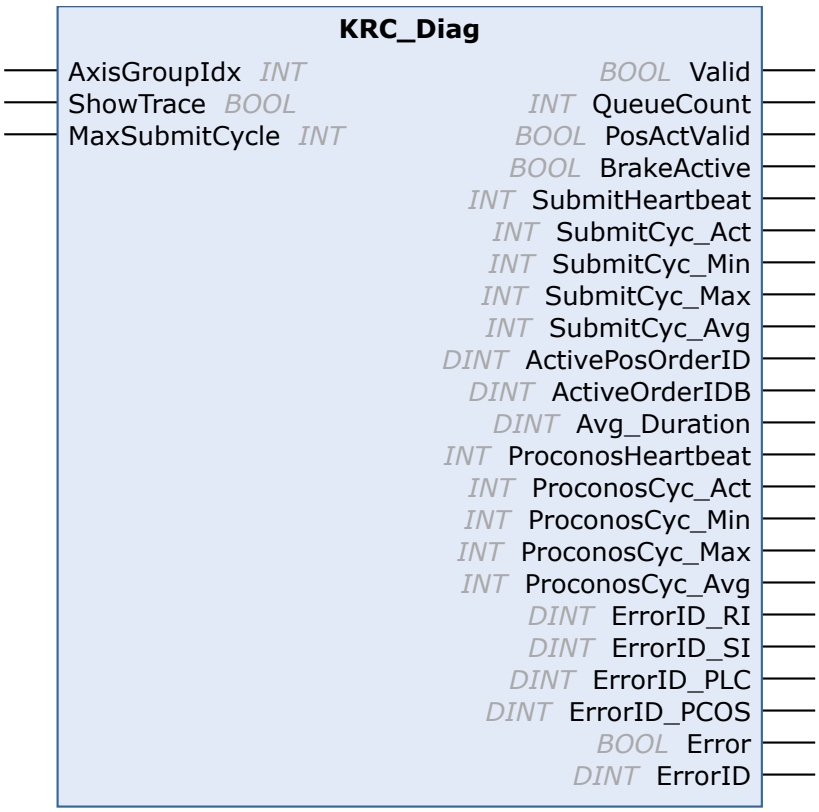


Abb. 35: Funktionsbaustein KRC_Diag

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ShowTrace	BOOL	TRUE = Anzeige der aktiven Funktionsbausteine im Meldungsfenster der KUKA smartHMI aktivieren FALSE = Anzeige der aktiven Funktionsbausteine im Meldungsfenster der KUKA smartHMI deaktivieren. **Hinweis**: Die Anzeige nur für Test- und Diagnosezwecke aktivieren. Wenn die Anzeige aktiv ist, ist kein Überschleifen mehr möglich und die Zykluszeit des Submit-Interpreters wird negativ beeinflusst.
MaxSubmitCycle	INT	Maximale Zykluszeit des Submit-Interpreters Default: **1 000 ms** **Hinweis**: Wird die maximale Zykluszeit überschritten, wird das Signal $MOVE_ENABLE für die Fahrfreigabe zurückgesetzt.

Ausgänge

Parameter	Typ	Beschreibung
Valid	BOOL	TRUE = Daten sind gültig
QueueCount	INT	Anzahl der gepufferten Anweisungen • **1 … 90**
PosActValid	BOOL	TRUE = Positionsdaten sind gültig (SAK)
BrakeActive	BOOL	TRUE = Roboter wird durch eine BRAKE-Anweisung angehalten
SubmitHeartbeat	INT	Heartbeat-Signal des Submit-Interpreters (Zähler wird in jedem Submit-Zyklus um 1 erhöht) • **1 … 245**

Parameter	Typ	Beschreibung
SubmitCyc_Act	REAL	Aktuelle Zykluszeit des Submit-Interpreters; Einheit: ms Mittelwert über 1 000 ms = 1/Anzahl der Zyklen pro Sekunde
SubmitCyc_Min	REAL	Kürzeste Zykluszeit des Submit-Interpreters seit der letzten Verbindungsunterbrechung; Einheit: ms
SubmitCyc_Max	REAL	Längste Zykluszeit des Submit-Interpreters seit der letzten Verbindungsunterbrechung; Einheit: ms
SubmitCyc_Avg	INT	Mittelwert der Zykluszeit des Submit-Interpreters im Ermittlungszeitraum **Avg_Duration**; Einheit: ms
ActivePosOrderID	DINT	Order ID des KRC_Move-Bewegungsbefehls, der aktuell ausgeführt wird
ActiveOrderIDB	DINT	Order ID des aktuellen KRC_Move-Bewegungsbefehls im Vorlauf
Avg_Duration	DINT	Dauer des aktuellen Ermittlungszeitraums für den Mittelwert der Zykluszeit; Einheit: ms Der Ermittlungszeitraum beginnt nach einer Unterbrechung der Verbindung zum Submit-Interpreter oder nach spätestens 60 Minuten neu.
ProconosHeartbeat	INT	Lebenszeichen von ProConOS (Zähler wird in jedem ProConOS-Zyklus um 1 erhöht)
ProconosCyc_Act	INT	Aktuelle Zykluszeit von ProConOS; Einheit: ms Mittelwert über 1 000 ms = 1/Anzahl der Zyklen pro Sekunde
ProconosCyc_Min	INT	Kürzeste Zykluszeit von ProConOS seit der letzten Verbindungsunterbrechung; Einheit: ms
ProconosCyc_Max	INT	Längste Zykluszeit von ProConOS seit der letzten Verbindungsunterbrechung; Einheit: ms
ProconosCyc_Avg	INT	Mittelwert der Zykluszeit von ProConOS im Ermittlungszeitraum **Avg_Duration**; Einheit: ms
ErrorID_RI	DINT	Fehlernummer Roboter-Interpreter
ErrorID_SI	DINT	Fehlernummer Submit-Interpreter
ErrorID_PLC	DINT	Fehlernummer SPS
ErrorID_PCOS	DINT	Fehlernummer ProConOS
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.8 Allgemeine Sonderfunktionen

7.8.1 Systemvariablen lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadSysVar können Systemvariablen gelesen werden.

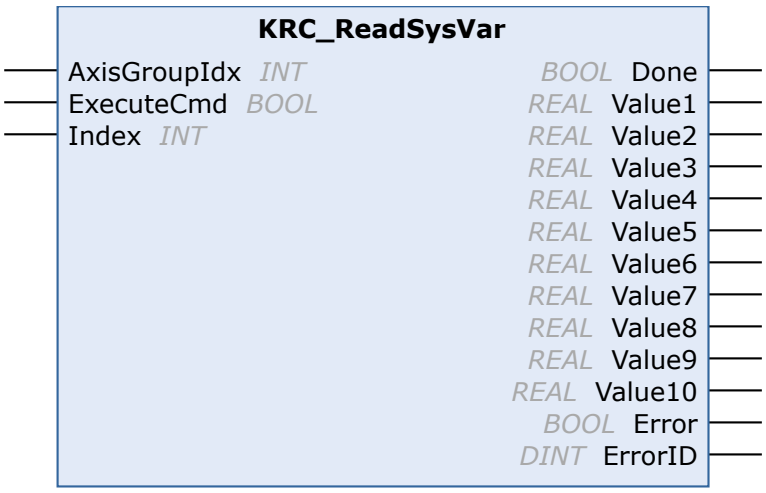


Abb. 36: Funktionsbaustein KRC_ReadSysVar

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
Index	INT	Index der Systemvariablen • **1**: $ADVANCE

Bislang kann nur die Systemvariable $ADVANCE gelesen werden. Wenn es die kundenspezifische Anwendung erfordert, kann die Liste der lesbaren Systemvariablen durch KUKA erweitert werden.

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer
Value1 … Value10	REAL	Wert der Systemvariablen Wenn die Systemvariable ein Strukturtyp ist, können bis zu 10 Komponenten der Struktur gelesen werden.

7.8.2 Systemvariablen schreiben

Beschreibung

Mit dem Funktionsbaustein KRC_WriteSysVar können Systemvariablen geschrieben werden.

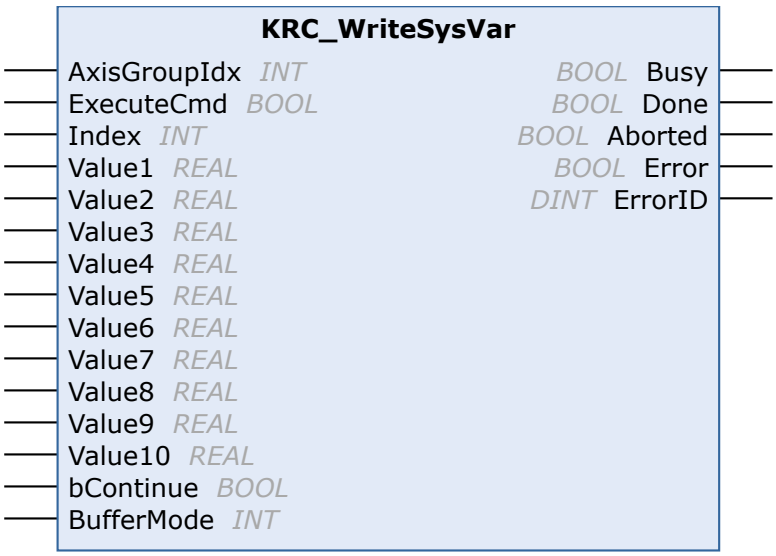


Abb. 37: Funktionsbaustein KRC_WriteSysVar

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
Index	INT	Index der Systemvariablen • **1**: $ADVANCE
Value1 … Value10	REAL	Wert der Systemvariablen Wenn die Systemvariable ein Strukturtyp ist, können bis zu 10 Komponenten der Struktur geschrieben werden.
bContinue	BOOL	TRUE = Systemvariable ohne Vorlaufstopp beschreiben **Hinweis**: Nur bei bestimmten Systemvariablen möglich.
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Bislang kann nur die Systemvariable $ADVANCE geschrieben werden. Wenn es die kundenspezifische Anwendung erfordert, kann die Liste der schreibbaren Systemvariablen durch KUKA erweitert werden.

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.8.3 Bremsentest aufrufen

Beschreibung

Mit dem Funktionsbaustein KRC_BrakeTest wird das Programm für den Bremsentest aufgerufen. Der Bremsentest wird an der Position gestartet, an der sich der Roboter bei Programmaufruf befindet.

Der Bremsentest muss mit einem Programm-Override von 100 % ausgeführt werden (Funktionsbaustein KRC_SetOverride).

Beim Bremsentest wird für alle Bremsen geprüft, ob eine Bremse die Verschleißgrenze erreicht hat. Dazu beschleunigt der Roboter auf eine definierte Geschwindigkeitsgrenze. Wenn der Roboter die Geschwindigkeit erreicht hat, fällt die Bremse ein und das Ergebnis für diesen Bremsvorgang wird angezeigt.

Bei einem erfolgreichen Bremsentest steht der Roboter am Ende der Messung wieder in der Startposition.

Ist der Bremsentest fehlgeschlagen, d. h. eine Bremse wurde als defekt erkannt, fährt der Roboter direkt eine Parkposition an. Die Koordinaten der Parkposition müssen im Funktionsbaustein angegeben werden.

Parkposition

Die Parkposition muss so gewählt werden, dass keine Personen gefährdet werden, falls der Roboter aufgrund der defekten Bremse zusammensackt. Als Parkposition kann z. B. die Transportstellung gewählt werden.

Weitere Informationen zur Transportstellung sind in der Betriebsanleitung oder der Montageanleitung für den Roboter zu finden.

Detaillierte Informationen zum Bremsentest sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.

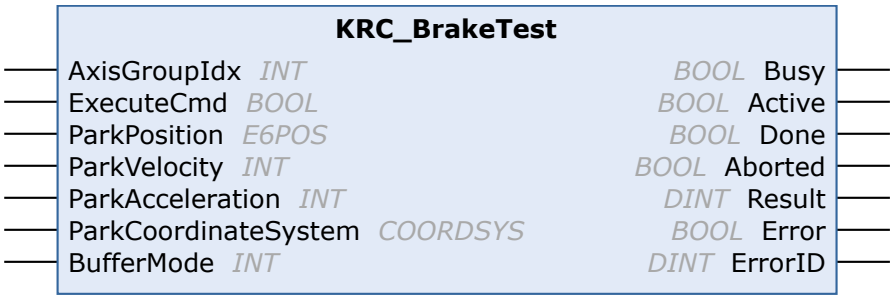


Abb. 38: Funktionsbaustein KRC_BrakeTest

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
ParkPosition	E6POS	Koordinaten der kartesischen Parkposition (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Parkposition (= Position des TCP bezogen auf den Ursprung des ausgewählten Koordinatensystems).
ParkVelocity	INT	Geschwindigkeit

Parameter	Typ	Beschreibung
		• **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp. Default: **0 %** (= Geschwindigkeit wird nicht verändert)
ParkAcceleration	INT	Beschleunigung • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp. Default: **0 %** (= Beschleunigung wird nicht verändert)
ParkCoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Parkposition beziehen (>>> COORDSYS [▶ 20])
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet
Aborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Result	DINT	Ergebnis des Bremsentests • **0**: Bremsentest fehlgeschlagen (Bremse als defekt erkannt oder keine Verbindung zur Robotersteuerung) • **1**: Bremsentest erfolgreich (keine Bremse defekt, aber mindestens eine Bremse hat Verschleißgrenze erreicht) • **2**: Bremsentest erfolgreich (keine Bremse defekt oder Verschleißgrenze erreicht)
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.8.4 Justagereferenzierung aufrufen

Beschreibung

Mit dem Funktionsbaustein KRC_MasRef wird die Justagereferenzierung durchgeführt.

Nach dem Aufruf des Funktionsbausteins fährt der Roboter von der aktuellen Position linear zur Referenzposition. Wenn der Roboter die Referenzposition erreicht hat, werden die aktuellen Achswerte mit den Achswerten verglichen, die in KUKA.SafeOperation gespeichert wurden. Anschließend fährt der Roboter zur Startposition (= Position vor Aufruf des Funktionsbausteins) zurück.

Die Referenzposition wird im Funktionsbaustein mit dem Eingangsparameter Position definiert und entspricht der mit KUKA.SafeOperation definierten Referenzposition.

Wenn die Abweichung zwischen aktueller Position und Referenzposition zu groß ist, ist die Justagereferenzierung fehlgeschlagen.

Detaillierte Informationen zur Justagereferenzierung sind in der Dokumentation KUKA.SafeOperation zu finden.

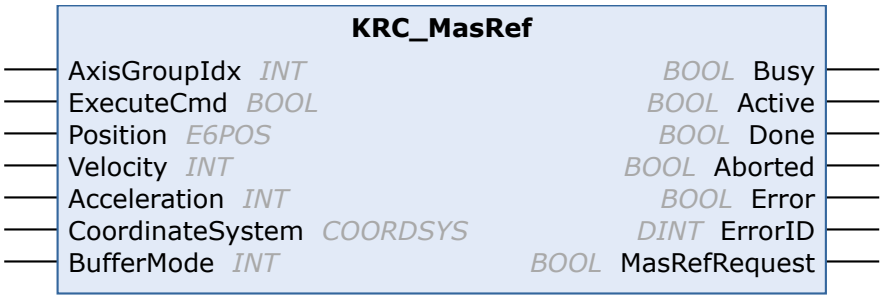


Abb. 39: Funktionsbaustein KRC_MasRef

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
Position	E6POS	Koordinaten der kartesischen Referenzposition (>>> E6POS [▶ 20]) Die Datenstruktur E6POS enthält alle Komponenten der Referenzposition (= Position des TCP bezogen auf den Ursprung des ausgewählten Koordinatensystems).
Velocity	INT	Geschwindigkeit • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp. Default: **0 %** (= Geschwindigkeit wird nicht verändert)
Acceleration	INT	Beschleunigung • **0 … 100 %** Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp. Default: **0 %** (= Beschleunigung wird nicht verändert)
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die kartesischen Koordinaten der Referenzposition beziehen (>>> COORDSYS [▶ 20])
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Bewegung wird aktuell ausgeführt
Done	BOOL	TRUE = Bewegung ist beendet

Parameter	Typ	Beschreibung
Aborted	BOOL	TRUE = Anweisung/Bewegung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer
MasRefRequest	BOOL	TRUE = Justagereferenzierung wurde von Robotersteuerung intern angefordert

7.8.5 Signale der Sicherheitssteuerung lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadSafeOPStatus werden Signale der Sicherheitssteuerung gelesen. (Nur relevant, wenn KUKA.SafeOperation installiert ist.)

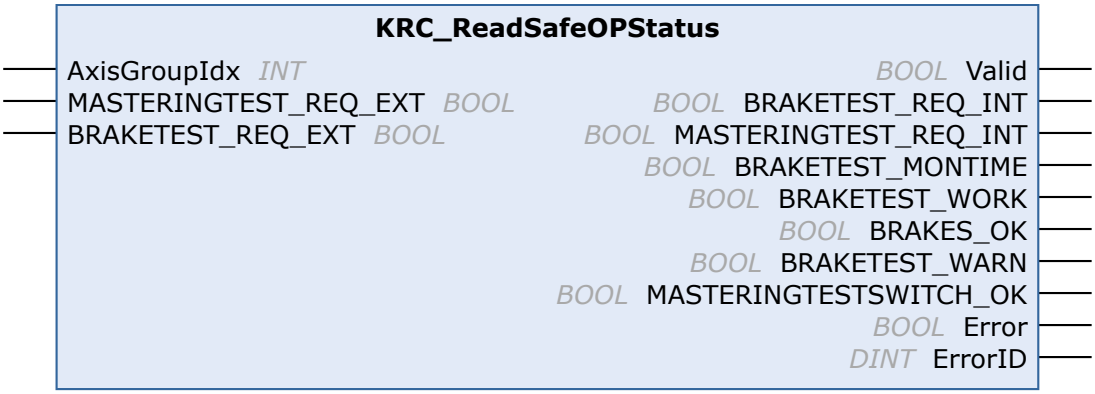


Abb. 40: Funktionsbaustein KRC_ReadSafeOPStatus

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
MASTERINGTEST_REQ_EXT	BOOL	TRUE = Justagereferenzierung wird von der SPS angefordert
BRAKETEST_REQ_EXT	BOOL	TRUE = Bremsentest wird von der SPS angefordert

Ausgänge

Parameter	Typ	Beschreibung
Valid	BOOL	TRUE = Daten sind gültig
BRAKETEST_REQ_INT	BOOL	TRUE = Bremsentest von der Sicherheitssteuerung angefordert
MASTERINGTEST_REQ_INT	BOOL	TRUE = Justagereferenzierung von der Sicherheitssteuerung angefordert
BRAKETEST_MONTIME	BOOL	TRUE = Roboter wurde gestoppt, da Monitoring-Zeit für Bremsentest abgelaufen
BRAKETEST_WORK	BOOL	TRUE = Bremsentest wird aktuell durchgeführt
BRAKES_OK	BOOL	Flanke **TRUE** --> **FALSE**: Eine Bremse wurde als defekt erkannt
BRAKETEST_WARN	BOOL	Flanke **FALSE** --> **TRUE**: Für mindestens 1 Bremse wurde ermittelt, dass die Verschleißgrenze erreicht ist.
MASTERINGTESTSWITCH_OK	BOOL	TRUE = Referenztaster i. O.

Parameter	Typ	Beschreibung
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.8.6 Zustand der TouchUp-Statustasten lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadTouchUPState wird der aktuelle Zustand der TouchUp-Statustasten am smartPAD gelesen. Um Punkte über die Statustasten am smartPAD zu teachen, muss der Funktionsbaustein mit dem Funktionsbaustein KRC_TouchUP verknüpft werden.

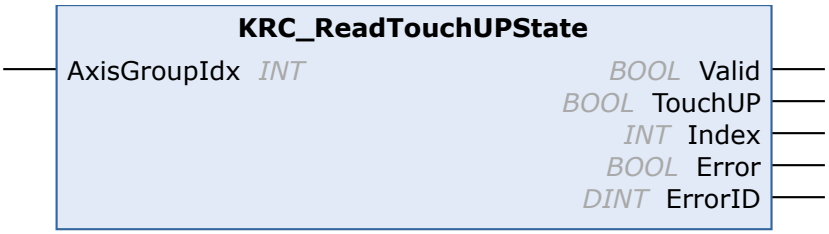


Abb. 41: Funktionsbaustein KRC_ReadTouchUPState

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**

Ausgänge

Parameter	Typ	Beschreibung
Valid	BOOL	TRUE = Daten sind gültig
TouchUP	BOOL	Zustand der TouchUp-Statustaste am smartPAD TRUE = TouchUp-Statustaste wurde gedrückt
Index	INT	Nummer, die mit der Statustaste am smartPAD ausgewählt wurde, um eine Position zu teachen • **1 … 100**
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.8.7 Punkte teachen

Beschreibung

Mit dem Funktionsbaustein KRC_TouchUP kann ein Punkt direkt in der SPS geteacht werden. Werkzeug, Basis und Interpolationsmodus, die zu diesem Punkt gehören, werden vom Funktionsbaustein automatisch gespeichert.

Abb. 42: Funktionsbaustein KRC_TouchUP

Eingänge

Parameter	Typ	Beschreibung
PositionArray	POSITION_ ARRAY	Feld mit den geteachten Punkten und den zugehörigen Koordinatensystemen **Hinweis**: Das Feld kann in der SPS direkt verwendet werden. (>>> POSITION_ ARRAY [▶ 21])
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	TRUE = Punkt wird geteacht
Index	INT	Nummer, mit der der geteachte Punkt in der SPS gespeichert wird • **1 … 100**

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.8.8 Einstellungen für den Vorlauf ändern

Beschreibung

Mit dem Funktionsbaustein KRC_SetAdvance werden die Einstellungen für den Vorlauf geändert.

Der Vorlauf ist die maximale Anzahl der Bewegungssätze, die die Robotersteuerung beim Programmlauf im Voraus berechnet und plant. Die tatsächliche Anzahl ist abhängig von der Rechnerauslastung. Der Vorlauf ist unter anderem notwendig, um Überschleifbewegungen berechnen zu können.

Wenn die Programmbearbeitung zurückgesetzt wird, werden die eingestellten Werte auf die Default-Werte zurückgesetzt.

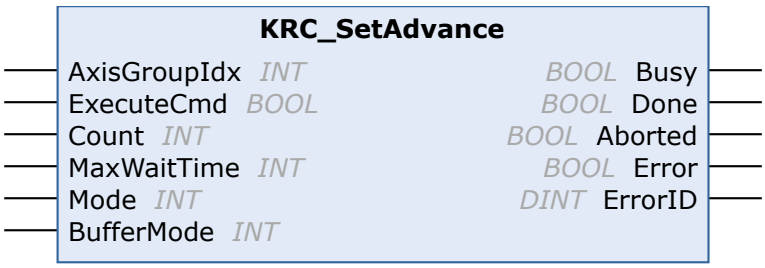


Abb. 43: Funktionsbaustein KRC_SetAdvance

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals gepuffert.
Count	INT	Anzahl der Funktionen, die vor der 1. Roboterbewegung übertragen werden sollen • **1 … 50** Default-Wert: 2
MaxWaitTime	INT	Maximale Wartezeit vor dem Beginn der Programmbearbeitung, wenn die eingestellte Anzahl der Funktionen im Parameter Count nicht erreicht wird. • **1 … 32 767 ms** Default-Wert: 300 ms
Mode	INT	Warte-Modus • **0**: Der aktuell eingestellte Modus wird nicht verändert. • **1**: Wenn die 1. Anweisung eine überschliffene Bewegungsanweisung ist, wird auf weitere Anweisungen gewartet. • **2**: Es wird immer auf die Anzahl der eingestellten Funktionen oder den Ablauf der maximalen Wartezeit gewartet. Default-Wert: 1
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wurde übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.8.9 Einstellungen für den Vorlauf auslesen

Beschreibung

Mit dem Funktionsbaustein KRC_GetAdvance werden die Werte ausgelesen, die im Funktionsbaustein KRC_SetAdvance eingestellt wurden.

Abb. 44: Funktionsbaustein KRC_GetAdvance

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Count	INT	Anzahl der Funktionen, die vor der 1. Roboterbewegung von der SPS übertragen werden sollen • **1 … 50** Default-Wert: 2
MaxWaitTime	INT	Maximale Wartezeit vor dem Beginn der Programmbearbeitung, wenn die eingestellte Anzahl der Funktionen im Parameter Count nicht erreicht wird. • **1 … 32 767 ms** Default-Wert: 300 ms
Mode	INT	Warte-Modus • **0**: Der aktuell eingestellte Modus wird nicht verändert. • **1**: Wenn die 1. Anweisung eine überschliffene Bewegungsanweisung ist, wird auf weitere Anweisungen gewartet. • **2**: Es wird immer auf die Anzahl der eingestellten Funktionen oder den Ablauf der maximalen Wartezeit gewartet. Default-Wert: 1
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.8.10 Kartesische Roboterposition aus Achswinkeln berechnen

Beschreibung

Mit dem Funktionsbaustein KRC_Forward wird aus vorgegebenen Achswinkeln die kartesische Roboterposition berechnet.

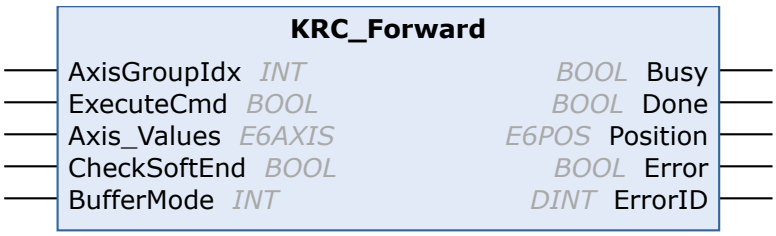


Abb. 45: Funktionsbaustein KRC_Forward

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Anweisung mit einer steigenden Flanke des Signals.
Axis_Values	E6AXIS	Achsspezifische Werte, die in kartesische Koordinaten umgerechnet werden sollen (>>> E6AXIS [▶ 20]) Die Datenstruktur E6AXIS enthält die Winkel- oder Translationswerte für alle Achsen der Achsgruppe in dieser Position.
CheckSoftEnd	BOOL	Prüft, ob die vorgegebenen Achswinkel innerhalb der Software-Endschalter liegen. Wenn nicht, wird eine Fehlernummer ausgegeben.
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Position	E6POS	Kartesische Roboterposition, die aus den vorgegebenen Achswinkeln berechnet wurde
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.8.11 Kartesische Roboterposition aus Achswinkeln berechnen (erweitert)

Beschreibung

Mit dem Funktionsbaustein KRC_ForwardAdvanced wird aus vorgegebenen Achswinkeln die kartesische Roboterposition berechnet. Bei der Berechnung werden das TOOL- und BASE-Koordinatensystem sowie der Interpolationsmodus miteinbezogen.

Die Funktion wird automatisch im BufferMode 0 ausgeführt und kann deshalb unabhängig von der Bewegungsausführung genutzt werden.

Der vorgegebene Wert für den Interpolationsmodus muss mit dem aktuellen Wert auf der Robotersteuerung übereinstimmen.

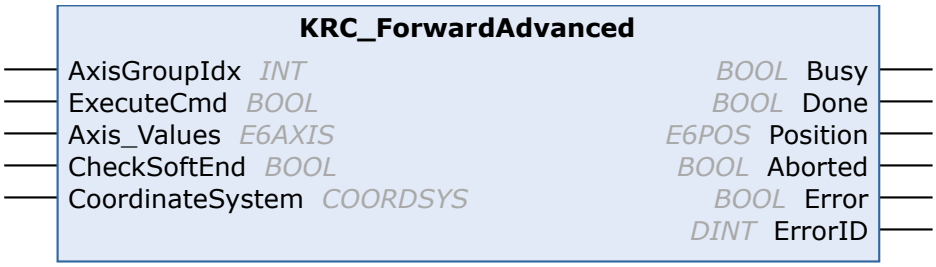


Abb. 46: Funktionsbaustein KRC_ForwardAdvanced

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Anweisung mit einer steigenden Flanke des Signals.
Axis_Values	E6AXIS	Achsspezifische Werte, die in kartesische Koordinaten umgerechnet werden sollen (>>> E6AXIS [▶ 20]) Die Datenstruktur E6AXIS enthält die Winkel- oder Translationswerte für alle Achsen der Achsgruppe in dieser Position.
CheckSoftEnd	BOOL	Prüft, ob die vorgegebenen Achswinkel innerhalb der Software-Endschalter liegen. Wenn nicht, wird eine Fehlernummer ausgegeben.
CoordinateSystem	COORDSYS	Angabe des TOOL- und BASE-Koordinatensystems sowie des Interpolationsmodus (>>> COORDSYS [▶ 20])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Position	E6POS	Kartesische Roboterposition, die aus den vorgegebenen Achswinkeln berechnet wurde **Hinweis**: Die Zusatzachsen E4 bis E6 werden bei der Berechnung nicht berücksichtigt.
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.8.12 Achswinkel aus kartesischer Roboterposition berechnen

Beschreibung

Mit dem Funktionsbaustein KRC_Inverse werden aus einer vorgegebenen kartesischen Roboterposition die Achswinkel berechnet.

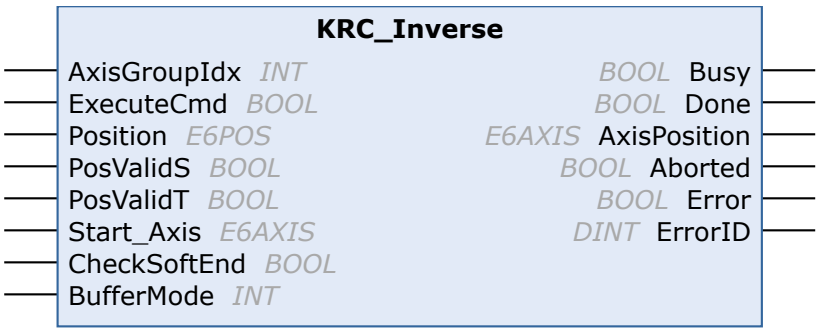


Abb. 47: Funktionsbaustein KRC_Inverse

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Anweisung mit einer steigenden Flanke des Signals.
Position	E6POS	Kartesische Roboterposition
PosValidS	BOOL	TRUE = Der Status-Wert, der im Parameter Position enthalten ist, ist gültig. FALSE = Der Status-Wert ist nicht bekannt.
PosValidT	BOOL	TRUE = Der Turn-Wert, der im Parameter Position enthalten ist, ist gültig. FALSE = Der Turn-Wert ist nicht bekannt.
Start_Axis	E6Axis	Achsspezifische Werte am Startpunkt der Bewegung Der Startpunkt ist die achsspezifische Position, von der aus der Roboter zu der Position fährt, die berechnet werden soll.
CheckSoftEnd	BOOL	Prüft, ob die Werte aus dem Parameter Start_Axis innerhalb der Software-Endschalter liegen. Wenn nicht, wird eine Fehlernummer ausgegeben.
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
AxisPosition	E6AXIS	Achswinkel, die aus der vorgegebenen kartesischen Roboterposition berechnet wurden (>>> E6AXIS [▶ 20]) Die Datenstruktur E6AXIS enthält alle Achspositionen der Achsgruppe.
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen, bevor sie im Vorlauf bearbeitet wurde
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.8.13 Achswinkel aus kartesischer Roboterposition berechnen (erweitert)

Beschreibung

Mit dem Funktionsbaustein KRC_InverseAdvanced werden aus einer vorgegebenen kartesischen Roboterposition die Achswinkel berechnet. Bei der Berechnung werden das TOOL- und BASE-Koordinatensystem sowie der Interpolationsmodus miteinbezogen.

Die Funktion wird automatisch im BufferMode 0 ausgeführt und kann deshalb unabhängig von der Bewegungsausführung genutzt werden.

Der vorgegebene Wert für den Interpolationsmodus muss mit dem aktuellen Wert auf der Robotersteuerung übereinstimmen.

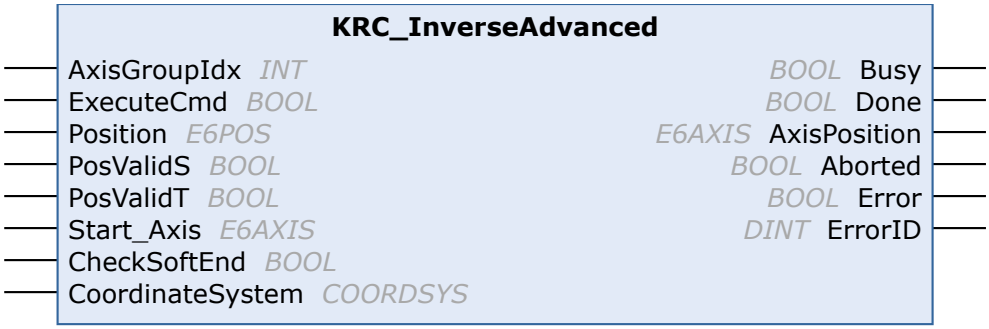


Abb. 48: Funktionsbaustein KRC_InverseAdvanced

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Anweisung mit einer steigenden Flanke des Signals.
Position	E6POS	Kartesische Roboterposition **Hinweis**: Die Zusatzachsen E4 bis E6 werden bei der Berechnung nicht berücksichtigt.
PosValidS	BOOL	TRUE = Der Status-Wert, der im Parameter Position enthalten ist, ist gültig. FALSE = Der Status-Wert ist nicht bekannt.
PosValidT	BOOL	TRUE = Der Turn-Wert, der im Parameter Position enthalten ist, ist gültig. FALSE = Der Turn-Wert ist nicht bekannt.
Start_Axis	E6Axis	Achsspezifische Werte am Startpunkt der Bewegung Der Startpunkt ist die achsspezifische Position, von der aus der Roboter zu der Position fährt, die berechnet werden soll.
CheckSoftEnd	BOOL	Prüft, ob die Werte aus dem Parameter Start_Axis innerhalb der Software-Endschalter liegen. Wenn nicht, wird eine Fehlernummer ausgegeben.
CoordinateSystem	COORDSYS	Angabe des TOOL- und BASE-Koordinatensystems sowie des Interpolationsmodus (>>> COORDSYS [▶ 20])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
AxisPosition	E6AXIS	Achswinkel, die aus der vorgegebenen kartesischen Roboterposition berechnet wurden (>>> E6AXIS [▶ 20]) Die Datenstruktur E6AXIS enthält alle Achspositionen der Achsgruppe.
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen, bevor sie im Vorlauf bearbeitet wurde
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.8.14 KRL-Programme ausführen

Beschreibung

Mit dem Funktionsbaustein KRC_TechFunction werden KRL-Programme im Roboter- oder Submit-Interpreter ausgeführt. Mit dem Parameter BufferMode kann ausgewählt werden, in welchem Modus das Programm ausgeführt wird.

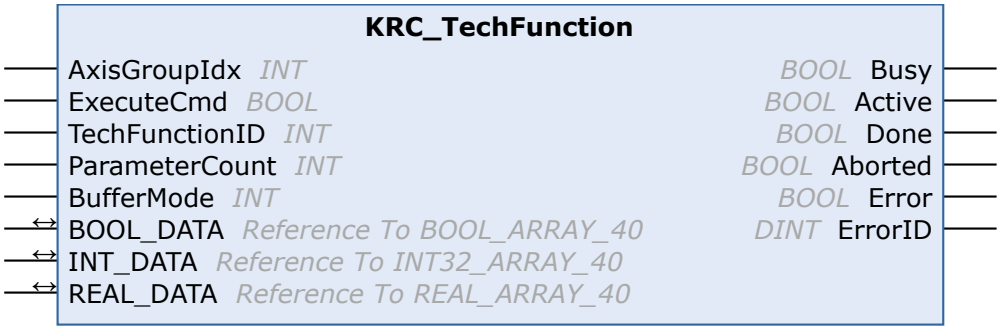


Abb. 49: Funktionsbaustein KRC_TechFunction

Eingänge

Parameter	Typ	Beschreibung
BOOL_DATA	ARRAY [1 … 40] OF BOOL	Feld vom Typ BOOL als Übergabeparameter
INT_DATA	ARRAY [1 … 40] OF DINT	Feld vom Typ DINT als Übergabeparameter
REAL_DATA	ARRAY [1 … 40] OF REAL	Feld vom Typ REAL als Übergabeparameter
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
TechFunctionID	INT	ID zur Auswahl der Funktion auf KRL-Ebene (SWITCH CASE ID)
ParameterCount	INT	Maximaler Variablenindex, der für die Parameter BOOL_DATA, INT_DATA und REAL_DATA verwendet werden kann

Parameter	Typ	Beschreibung
		• **1 … 40**
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Anweisung wird aktuell ausgeführt
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.8.15 KRL-Programme ausführen (erweitert)

Beschreibung

Mit dem Funktionsbaustein KRC_TechFunctionAdvanced werden KRL-Programme im Submit-Interpreter ausgeführt. Die Rückgabewerte der KRL-Funktion werden im Parameter ReturnValue ausgegeben.

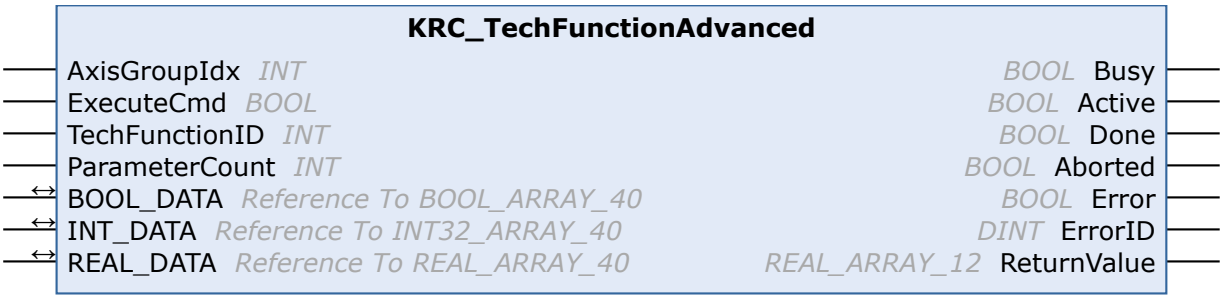


Abb. 50: Funktionsbaustein KRC_TechFunctionAdvanced

Eingänge

Parameter	Typ	Beschreibung
BOOL_DATA	ARRAY [1 … 40] OF BOOL	Feld vom Typ BOOL als Übergabeparameter
INT_DATA	ARRAY [1 … 40] OF DINT	Feld vom Typ DINT als Übergabeparameter
REAL_DATA	ARRAY [1 … 40] OF REAL	Feld vom Typ REAL als Übergabeparameter
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
TechFunctionID	INT	ID zur Auswahl der Funktion auf KRL-Ebene (SWITCH CASE ID)

Parameter	Typ	Beschreibung
ParameterCount	INT	Maximaler Variablenindex, der für die Parameter BOOL_DATA, INT_DATA und REAL_DATA verwendet werden kann • **1 … 40**

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Anweisung wird aktuell ausgeführt
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer
ReturnValue	ARRAY [1 … 12] OF REAL	Feld vom Typ REAL als Rückgabewerte aus der KRL-Funktion

7.8.16 Aktuelle Roboterposition in anderem Koordinatensystem anzeigen

Beschreibung

Mit dem Funktionsbaustein KRC_ActivatePosConversion kann die aktuelle Roboterposition bezüglich dem gewählten TOOL- oder BASE-Koordinatensystem oder Interpolationsmodus angezeigt werden.

Abb. 51: Funktionsbaustein KRC_ActivatePosConversion

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
ActivateConversion	BOOL	TRUE = Aktuelle Roboterposition wird im gewählten Koordinatensystem angezeigt. FALSE = Aktuelle Roboterposition wird im aktuellen Koordinatensystem der Robotersteuerung angezeigt.
CoordSysToDisplay	COORDSYS	Koordinatensystem, in dem die aktuelle Roboterposition angezeigt werden soll. (>>> COORDSYS [▶ 20])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.9 Sonderfunktionen für Conveyor

7.9.1 Conveyor initialisieren

Beschreibung

Mit dem Funktionsbaustein KRC_ConvIniOff wird ein Conveyor initialisiert. Das AMI wird dafür auf den Status #INITIALIZED gesetzt und die Conveyor-Distanz auf 0.

Abb. 52: Funktionsbaustein KRC_ConvIniOff

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
ConveyorNumber	INT	Nummer des Conveyors • **1 … 3**
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.9.2 Conveyor aktivieren

Beschreibung

Mit dem Funktionsbaustein KRC_ConvOn wird das AMI aktiviert, d. h. auf den Status #ACTIVE gesetzt. Wenn das AMI aktiviert ist, werden die Synchronisierungssignale am Eingang der Schnittstelle X33 (Schnelles Messen) ausgewertet.

Die Erkennung des Conveyor-Versatzes kann im Hintergrund erfolgen, wobei die Robotersteuerung andere Tasks ausführen kann. Dies ermöglicht dem Roboter die fliegende Verfolgung eines Teils auf dem Conveyor.

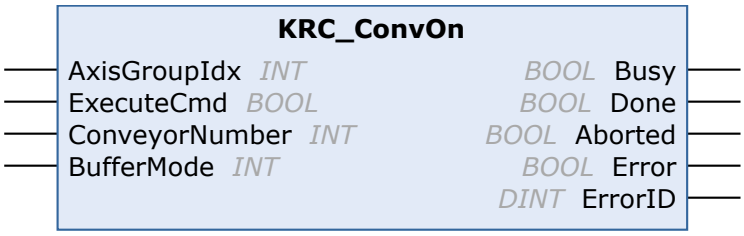


Abb. 53: Funktionsbaustein KRC_ConvOn

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
ConveyorNumber	INT	Nummer des Conveyors • **1 … 3**
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.9.3 Bauteil auf Conveyor verfolgen

Beschreibung

Mit dem Funktionsbaustein KRC_ConvFollow wird ein Bauteil auf dem Conveyor durch den Roboter verfolgt. Mit KRC_ConvFollow kann ein Bereich auf dem Conveyor festgelegt werden, in dem der Roboter damit beginnt, das Bauteil zu verfolgen.

Wenn das Bauteil zum Zeitpunkt des Aufrufs die maximale Conveyor-Distanz (Eingang **MaxDistance**) bereits überschritten hat, wird der Ausgang **MaxDistanceReached** gesetzt.

Dieser Funktionsbaustein kann nur ausgeführt werden, wenn das AMI mit KRC_ConvOn aktiviert wurde.

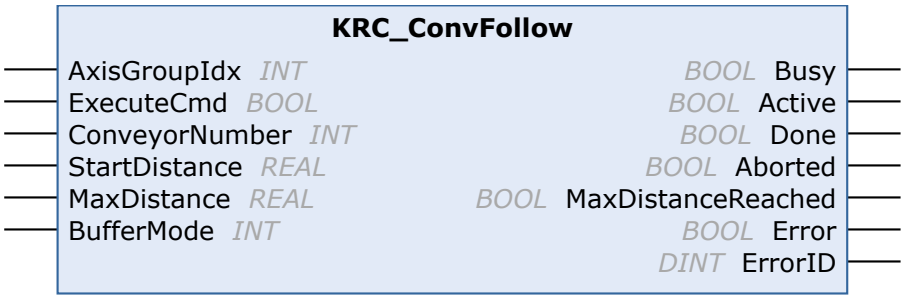


Abb. 54: Funktionsbaustein KRC_ConvFollow

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
ConveyorNumber	INT	Nummer des Conveyors • **1 … 3**
StartDistance	REAL	Verfahrstrecke des Bauteils, die der Roboter abwartet, bevor er mit der Verfolgung des Bauteils auf dem Conveyor beginnt. • Bei einem Linear-Conveyor: Angabe in Millimeter • Bei einem Zirkular-Conveyor: Angabe in Grad
MaxDistance	REAL	Maximale Verfahrstrecke des Bauteils, innerhalb der der Roboter damit beginnt, sich mit dem Bauteil zu synchronisieren. • Bei einem Linear-Conveyor: Angabe in Millimeter • Bei einem Zirkular-Conveyor: Angabe in Grad **Hinweis**: Dieser Eingang wird während synchronisierten Bewegungen des Conveyors nicht überwacht. Die Distanz, die das Bauteil zurückgelegt hat, wird von einem Interrupt überwacht. Die Einstellungen dazu werden in WorkVisual durchgeführt.
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Anweisung wird aktuell ausgeführt
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen

Parameter	Typ	Beschreibung
MaxDistanceReached	BOOL	TRUE = Die maximale Verfahrstrecke des Bauteils (Eingang **MaxDistance**) wurde zum Zeitpunkt der Ausführung bereits überschritten. Die Anweisung wurde nicht ausgeführt. Die Abarbeitung des Programms wird angehalten (WAIT FOR FALSE) und wartet auf einen Abbruch des Programms (Abort).
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.9.4 Bauteil von Conveyor aufnehmen

Beschreibung

Mit dem Funktionsbaustein KRC_ConvSkip wird festgelegt, welche Bauteile aufgenommen werden sollen, z. B. jedes 2. Bauteil, jedes 3. Bauteil usw. Insgesamt können bis zu 1024 Bauteile im Hintergrund überwacht werden.

Wenn das Bauteil zum Zeitpunkt des Aufrufs die maximale Conveyor-Distanz (Eingang **MaxDistance**) bereits überschritten hat, wird der Ausgang **MaxDistanceReached** gesetzt.

Dieser Funktionsbaustein kann nur ausgeführt werden, wenn das AMI mit KRC_ConvOn aktiviert wurde.

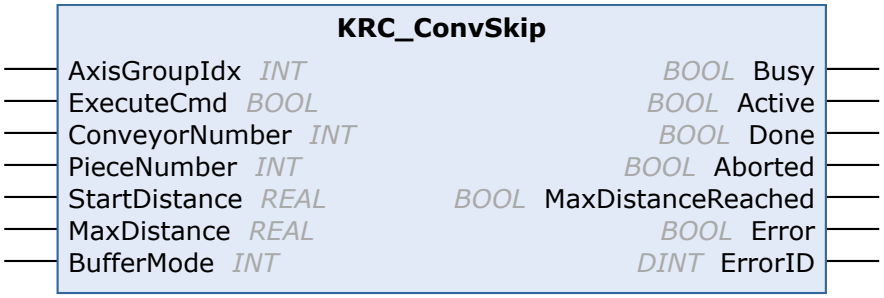


Abb. 55: Funktionsbaustein KRC_ConvSkip

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
ConveyorNumber	INT	Nummer des Conveyors • **1 … 3**
PieceNumber	INT	Die eingegebene Zahl gibt an, welche Bauteile aufgenommen werden sollen. Beispiele: • **1**: Jedes Bauteil wird aufgenommen. • **3**: Jedes 3. Bauteil wird aufgenommen. • **5**: Jedes 5. Bauteil wird aufgenommen.
StartDistance	REAL	Verfahrstrecke des Bauteils, die der Roboter abwartet, bevor er mit der Verfolgung des Bauteils auf dem Conveyor beginnt. • Bei einem Linear-Conveyor: Angabe in Millimeter • Bei einem Zirkular-Conveyor: Angabe in Grad

Parameter	Typ	Beschreibung
MaxDistance	REAL	Maximale Verfahrstrecke des Bauteils, innerhalb der der Roboter damit beginnt, sich mit dem Bauteil zu synchronisieren. • Bei einem Linear-Conveyor: Angabe in Millimeter • Bei einem Zirkular-Conveyor: Angabe in Grad **Hinweis**: Dieser Eingang wird während synchronisierten Bewegungen des Conveyors nicht überwacht. Die Distanz, die das Bauteil zurückgelegt hat, wird von einem Interrupt überwacht. Die Einstellungen dazu werden in WorkVisual vorgenommen.
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Anweisung wird aktuell ausgeführt
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
MaxDistanceReached	BOOL	TRUE = Die maximale Verfahrstrecke des Bauteils (Eingang **MaxDistance**) wurde zum Zeitpunkt der Ausführung bereits überschritten. Die Anweisung wurde nicht ausgeführt. Die Abarbeitung des Programms wird angehalten (WAIT FOR FALSE) und wartet auf einen Abbruch des Programms (Abort).
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.9.5 Bauteil auf Conveyor löschen

Beschreibung

Mit dem Funktionsbaustein KRC_ConvDelWPS wird die angegebene Anzahl an Bauteilen aus der Bauteilliste des Conveyor-Treibers gelöscht. Die Löschung beginnt beim ältesten Bauteil in der Liste.

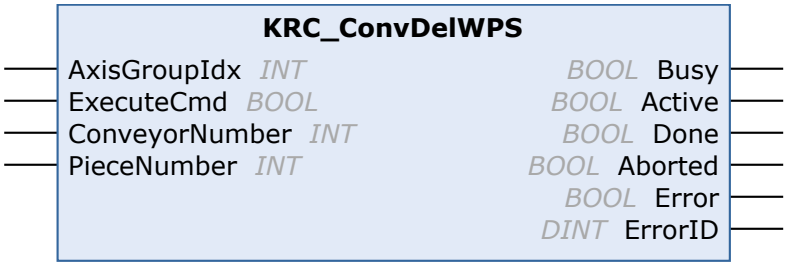


Abb. 56: Funktionsbaustein KRC_ConvDelWPS

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**

Parameter	Typ	Beschreibung
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
ConveyorNumber	INT	Nummer des Conveyors • **1 … 3**
PieceNumber	INT	Anzahl der Bauteile, die gelöscht werden sollen

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Anweisung wird aktuell ausgeführt
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.9.6 Interrupts für Überwachung aktivieren

Beschreibung

Mit dem Funktionsbaustein KRC_ActivateConvInterrupt werden folgende Interrupts aktiviert:

- Überwachung Alarm-Abstand
- Überwachung maximaler Abstand
- Überwachung $STOPMESS-Fehler

Ein Interrupt kann erst dann erfasst werden, wenn der Interrupt vom Hauptlauf des Roboter-Interpreters aktiviert wurde.

Die Überwachungen werden durch die Funktionsbausteine KRC_ConvFollow und KRC_ConvSkip aktiviert, sofern diese erfolgreich mit einem Bauteil synchronisiert wurden. Der Aufruf dieses Funktionsbausteins ist nur notwendig, wenn die Überwachung beendet und wieder aktiviert werden soll.

Abb. 57: Funktionsbaustein KRC_ActivateConvInterrupt

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
ConveyorNumber	INT	Nummer des Conveyors • **1 … 3**
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird

Parameter	Typ	Beschreibung
		• **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Anweisung wird aktuell ausgeführt
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.9.7 Interrupts für Überwachung deaktivieren

Beschreibung

Mit dem Funktionsbaustein KRC_DeactivateConvInterrupt werden folgende Interrupts deaktiviert:

- Überwachung Alarm-Abstand
- Überwachung maximaler Abstand
- Überwachung $STOPMESS-Fehler

Es wird empfohlen, diesen Funktionsbaustein aufzurufen, wenn der Conveyor-Bereich verlassen wird oder keine Überwachung gewünscht ist.

Abb. 58: Funktionsbaustein KRC_DeactivateConvInterrupt

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
ConveyorNumber	INT	Nummer des Conveyors • **1 … 3**
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Active	BOOL	TRUE = Anweisung wird aktuell ausgeführt
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.10 Sonderfunktionen für VectorMove

7.10.1 Bewegung entlang eines Vektors aktivieren

Beschreibung

Mit dem Funktionsbaustein KRC_VectorMoveOn kann ein Roboter durch eine extern einwirkende Kraft entlang eines definierten Vektors im kartesischen Raum bewegt werden.

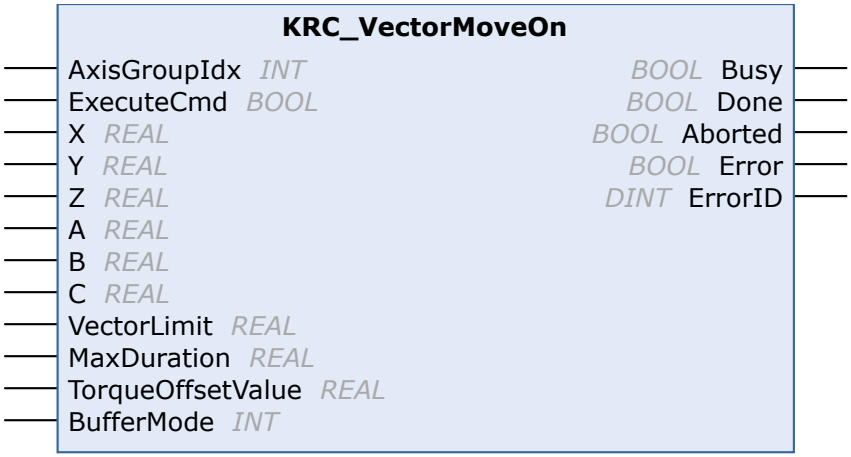


Abb. 59: Funktionsbaustein KRC_VectorMoveOn

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
X	REAL	Definiert die Richtung des Vektors Der Vektor muss in Bezug auf den TCP des TOOL-Koordinatensystems angegeben werden. Der letzte geteachte Punkt vor dem Aufruf des Funktionsbausteins ist der Fußpunkt des Vektors. Begrenzungen: • Translatorische Bewegung (X, Y, Z): max. 200 mm in positiver und negativer Richtung • Rotatorische Bewegung (A, B, C): max. 30° in positiver und negativer Richtung
Y	REAL	
Z	REAL	
A	REAL	
B	REAL	
C	REAL	
VectorLimit	REAL	Zulässige Überschreitung der Länge des Vektors

Parameter	Typ	Beschreibung
		Wenn dieser Wert überschritten wird, wird der Roboter mit einem Rampenstopp angehalten. • **0 … 30 %** Default: **10 %**
MaxDuration	REAL	Zeit, nach der VectorMove deaktiviert wird, wenn ein Fehler aufgetreten ist • **0 … 10000 s** Default: **100 s**
TorqueOffsetValue	REAL	Definiert das Widerstandsmoment des Roboters (Einheit: Nm) Das Widerstandsmoment ist das Moment, mit dem der Roboter der externen Kraft entgegenwirkt. Es dient dazu, dass der Roboter erst ab einer bestimmten Krafteinwirkung beginnt, sich zu bewegen. Das Widerstandsmoment kann zusätzlich zum Haltemoment definiert werden. Das Haltemoment ist abhängig von der Stellung, dem Typ und der Größe des Roboters sowie von der zusätzlich angebrachten Last. • **1 … 200 Nm** Default: **1 Nm**
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.10.2 Bewegung entlang eines Vektors deaktivieren

Beschreibung

Mit dem Funktionsbaustein KRC_VectorMoveOff wird die Bewegung entlang eines Vektors deaktiviert.

Abb. 60: Funktionsbaustein KRC_VectorMoveOff

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.11 Sonderfunktionen für LoadDataDetermination

7.11.1 Lastdatenermittlung konfigurieren

Beschreibung

Mit dem Funktionsbaustein KRC_LDDconfig wird die Lastdatenermittlung konfiguriert.

Voraussetzung für die Ausführung des Funktionsbausteins ist, dass der Funktionsbaustein KRC_LDDcheckPos zuvor erfolgreich ausgeführt wurde.

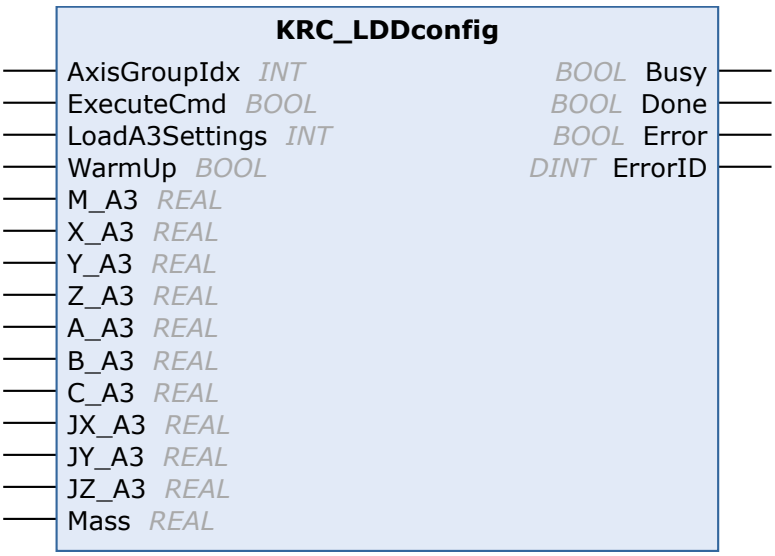


Abb. 61: Funktionsbaustein KRC_LDDconfig

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
LoadA3Settings	INT	• **0**: Keine Zusatzlast an Achse 3 • **1**: Für den Roboter maximal zulässige Last an Achse 3 ($DEF_LA3_M, $DEF_LA3_CM, $DEF_LA3_J) • **2**: Bei Inbetriebnahme definierte Werte aus LOAD_A3_DATA werden verwendet (Default) • **3**: Übergebene Daten vom Funktionsbaustein werden übernommen und in der Datei $config.dat gespeichert
WarmUp	BOOL	TRUE = Vor der Lastdatenermittlung wird eine Warmfahrt durchgeführt. FALSE = Vor der Lastdatenermittlung wird keine Warmfahrt durchgeführt.
M_A3	REAL	Masse der Zusatzlast an Achse 3
X_A3	REAL	Lage des Massenschwerpunkts der Zusatzlast an Achse 3 bezogen auf das FLANGE-Koordinatensystem
Y_A3		
Z_A3		
A_A3	REAL	Orientierung des Massenschwerpunkts der Zusatzlast an Achse 3 bezogen auf das FLANGE-Koordinatensystem
B_A3		
C_A3		
JX_A3	REAL	Massenträgheitsmomente der Zusatzlast an Achse 3 bezogen auf das FLANGE-Koordinatensystem
JY_A3		
JZ_A3		
Mass	REAL	• **≤ 0**: Traglast wird automatisch ermittelt • **> 0**: Traglast ist aus CAD-Daten oder einer Messung bereits bekannt

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.11.2 Startposition der Lastdatenermittlung prüfen

Beschreibung

Mit dem Funktionsbaustein KRC_LDDcheckPos wird die Startposition für die Lastdatenermittlung geprüft. Falls die Startposition nicht geeignet ist, wird eine Fehlernummer ausgegeben.

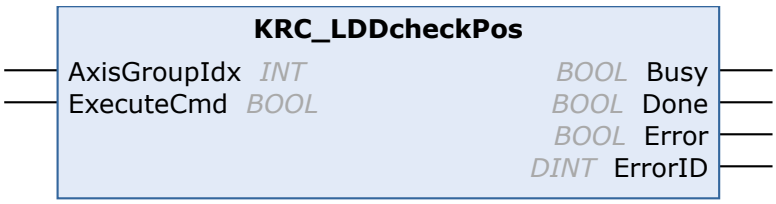


Abb. 62: Funktionsbaustein KRC_LDDcheckPos

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.11.3 Testfahrt vor Lastdatenermittlung durchführen

Beschreibung

Mit dem Funktionsbaustein KRC_LDDtestRun wird eine Testfahrt ohne Ermittlung der Lastdaten durchgeführt. Hierbei wird der Bewegungsablauf der Lastdatenermittlung mit niedriger Geschwindigkeit durchgeführt. Die Testfahrt dient dazu, mögliche Kollisionen bei der Lastdatenermittlung zu vermeiden.

Die Testfahrt vor der Lastdatenermittlung muss mit einem Programm-Override von ≤ 10 % durchgeführt werden (Funktionsbaustein KRC_SetOverride).

Abb. 63: Funktionsbaustein KRC_LDDtestRun

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.11.4 Lastdatenermittlung durchführen

Beschreibung

Mit dem Funktionsbaustein KRC_LDDstart wird eine Lastdatenermittlung durchgeführt.

Die Lastdatenermittlung muss mit einem Programm-Override von 100 % durchgeführt werden (Funktionsbaustein KRC_SetOverride).

Abb. 64: Funktionsbaustein KRC_LDDstart

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
Tool	INT	Nummer des Werkzeugs, für das die Lastdatenermittlung durchgeführt werden soll • **1 … maximale Anzahl der Werkzeuge**

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.11.5 Lastdaten zuweisen

Beschreibung

Mit dem Funktionsbaustein KRC_LDDwriteLoad werden die ermittelten Lastdaten dem angegebenen Werkzeug zugewiesen. Hierbei werden immer die zuletzt ermittelten Lastdaten verwendet. Somit ist es auch möglich, die zuletzt ermittelten Lastdaten mehreren Werkzeugen zuzuweisen.

Abb. 65: Funktionsbaustein KRC_LDDwriteLoad

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])
Tool	INT	Nummer des Werkzeugs, dem die Lastdaten zugewiesen werden sollen • **1 … maximale Anzahl der Werkzeuge**

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.12 Sonderfunktionen für Arbeitsräume

7.12.1 Kartesische Arbeitsräume konfigurieren

Beschreibung

Mit dem Funktionsbaustein KRC_WriteWorkspace werden kartesische (= kubische) Arbeitsräume für den Roboter konfiguriert. Arbeitsräume dienen dem Anlagenschutz. Maximal 8 kartesische Arbeitsräume können gleichzeitig konfiguriert werden. Die Arbeitsräume dürfen sich überlappen.

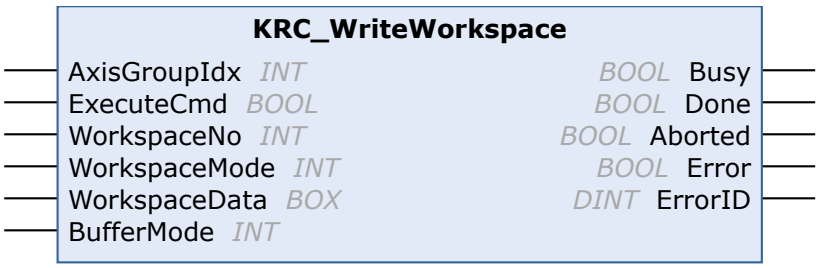


Abb. 66: Funktionsbaustein KRC_WriteWorkspace

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
WorkspaceNo	INT	Nummer des Arbeitsraums • **1 … 8**
WorkspaceMode	INT	Modus für Arbeitsräume • **0**: #OFF • **1**: #INSIDE • **2**: #OUTSIDE • **3**: #INSIDE_STOP • **4**: #OUTSIDE_STOP **Hinweis**: Weitere Informationen zum Modus für Arbeitsräume sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.
WorkspaceData	BOX	Daten des Arbeitsraums (>>> Daten eines kartesischen Arbeitsraums [▶ 21])
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.12.2 Konfiguration der kartesischen Arbeitsräume lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadWorkspace wird die Konfiguration der kartesischen Arbeitsräume für den Roboter gelesen.

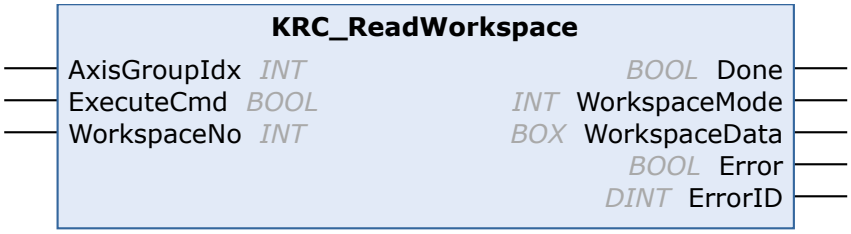


Abb. 67: Funktionsbaustein KRC_ReadWorkspace

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
WorkspaceNo	INT	Nummer des Arbeitsraums • **1 … 8**

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
WorkspaceMode	INT	Modus für Arbeitsräume • **0**: #OFF • **1**: #INSIDE • **2**: #OUTSIDE • **3**: #INSIDE_STOP • **4**: #OUTSIDE_STOP **Hinweis**: Weitere Informationen zum Modus für Arbeitsräume sind sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.
WorkspaceData	BOX	Daten des Arbeitsraums (>>> Daten eines kartesischen Arbeitsraums [▶ 21])
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.12.3 Achsspezifische Arbeitsräume konfigurieren

Beschreibung

Mit dem Funktionsbaustein KRC_WriteAxWorkspace werden achsspezifische Arbeitsräume für den Roboter konfiguriert. Diese dienen dem Anlagenschutz. Maximal 8 achsspezifische Arbeitsräume können gleichzeitig konfiguriert werden. Die Arbeitsräume dürfen sich überlappen.

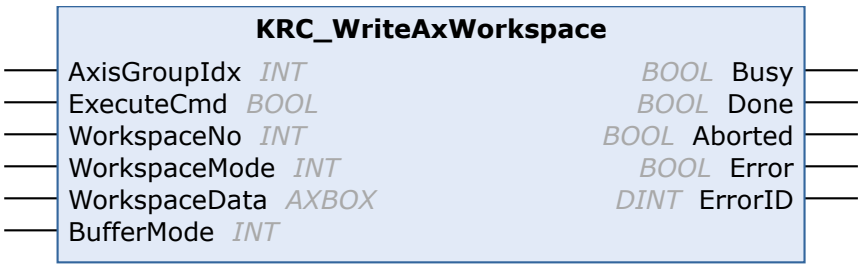


Abb. 68: Funktionsbaustein KRC_WriteAxWorkspace

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
WorkspaceNo	INT	Nummer des Arbeitsraums • **1 … 8**
WorkspaceMode	INT	Modus für Arbeitsräume • **0**: #OFF • **1**: #INSIDE • **2**: #OUTSIDE • **3**: #INSIDE_STOP • **4**: #OUTSIDE_STOP **Hinweis**: Weitere Informationen zum Modus für Arbeitsräume sind sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.
WorkspaceData	AXBOX	Daten des Arbeitsraums (>>> Daten eines achsspezifischen Arbeitsraums [▶ 22])
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.12.4 Konfiguration der achsspezifischen Arbeitsräume lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadAxWorkspace wird die Konfiguration der achsspezifischen Arbeitsräume für den Roboter gelesen.

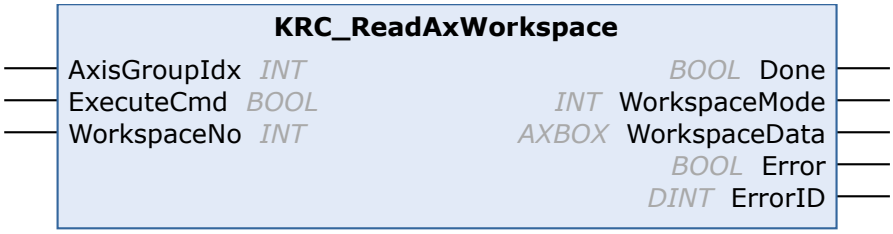


Abb. 69: Funktionsbaustein KRC_ReadAxWorkspace

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Startet/Puffert die Bewegung mit einer steigenden Flanke des Signals.
WorkspaceNo	INT	Nummer des Arbeitsraums • **1 … 8**

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
WorkspaceMode	INT	Modus für Arbeitsräume • **0**: #OFF • **1**: #INSIDE • **2**: #OUTSIDE • **3**: #INSIDE_STOP • **4**: #OUTSIDE_STOP **Hinweis**: Weitere Informationen zum Modus für Arbeitsräume sind sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.
WorkspaceData	AXBOX	Daten des Arbeitsraums (>>> Daten eines achsspezifischen Arbeitsraums [▶ 22])
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.12.5 Status der Arbeitsräume lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadWorkstates wird der aktuelle Status der Arbeitsräume gelesen. Der Status der Arbeitsräume wird zyklisch aktualisiert.

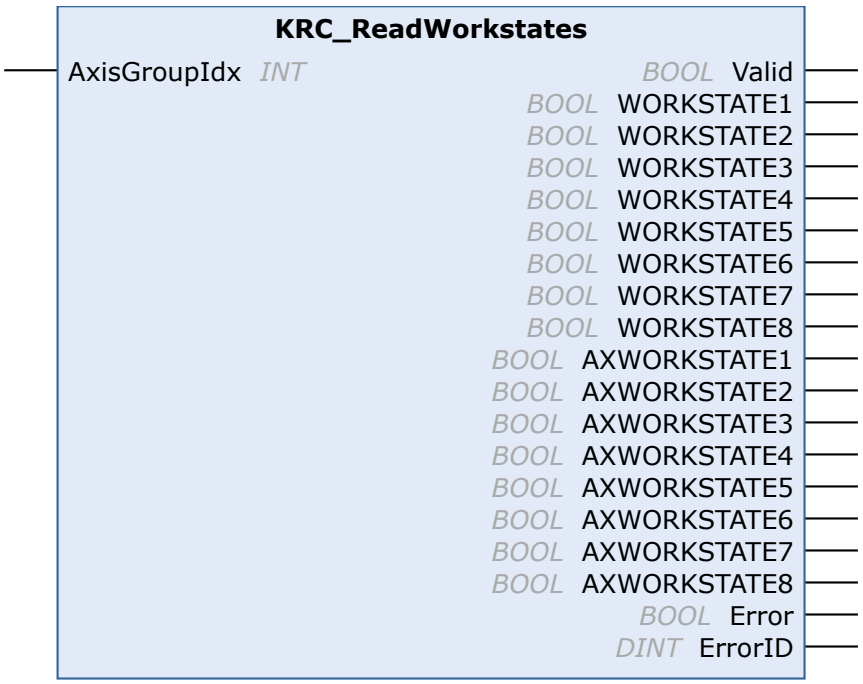


Abb. 70: Funktionsbaustein KRC_ReadWorkstates

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**

Ausgänge

Parameter	Typ	Beschreibung
Valid	BOOL	TRUE = Daten sind gültig
WORKSTATE1	BOOL	Status der Arbeitsräume
WORKSTATE2	BOOL	
WORKSTATE3	BOOL	
WORKSTATE4	BOOL	
WORKSTATE5	BOOL	
WORKSTATE6	BOOL	
WORKSTATE7	BOOL	
WORKSTATE8	BOOL	
AXWORKSTATE1	BOOL	
AXWORKSTATE2	BOOL	
AXWORKSTATE3	BOOL	
AXWORKSTATE4	BOOL	
AXWORKSTATE5	BOOL	
AXWORKSTATE6	BOOL	
AXWORKSTATE7	BOOL	
AXWORKSTATE8	BOOL	
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13 Administrative Funktionen

7.13.1 Ausgänge des Robotersystems lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadAxisGroup wird der Parameter KRC4_Input dem Parameter AxisGroupIdx zugewiesen. Nachfolgende Funktionsbausteine referenzieren anhand des Parameters AxisGroupIdx immer auf das gleiche Robotersystem.

Der Funktionsbaustein KRC_ReadAxisGroup muss immer am Anfang des Programms aufgerufen werden. Ein Zugriff auf den Parameter AxisGroupIdx ist nur zwischen den Funktionsbausteinen KRC_ReadAxisGroup und KRC_WriteAxisGroup zulässig.

Der Funktionsbaustein darf pro Achsgruppe nur einfach instanziiert werden. Bei einer mehrfachen Instanziierung werden die Signale des zuletzt aufgerufenen Funktionsbausteins ausgegeben.

Abb. 71: Funktionsbaustein KRC_ReadAxisGroup

Eingänge

Parameter	Typ	Beschreibung
KRC4_Input	POINTER TO BYTE	Legt den Startindex des Ausgangsbereichs der Robotersteuerung fest.
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**

Ausgänge

Parameter	Typ	Beschreibung
ErrorID	DINT	Fehlernummer
Error	BOOL	TRUE = Fehler im Funktionsbaustein

7.13.2 Eingänge des Robotersystems schreiben

Beschreibung

Mit dem Funktionsbaustein KRC_WriteAxisGroup werden anhand des Parameters AxisGroupIdx die Eingänge in den Bereich geschrieben, der mit dem Parameter KRC4_Output festgelegt wird.

Der Funktionsbaustein KRC_WriteAxisGroup muss immer am Ende des Programms aufgerufen werden. Ein Zugriff auf die AxisGroupIdx ist nur zwischen den Funktionsbausteinen KRC_ReadAxisGroup und KRC_WriteAxisGroup zulässig.

Der Funktionsbaustein darf pro Achsgruppe nur einfach instanziiert werden. Bei einer mehrfachen Instanziierung werden die Signale des zuletzt aufgerufenen Funktionsbausteins ausgegeben.

Abb. 72: Funktionsbaustein KRC_WriteAxisGroup

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
KRC4_Output	POINTER TO BYTE	Legt den Startindex des Eingangsbereichs der Robotersteuerung fest

Ausgänge

Parameter	Typ	Beschreibung
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.3 mxA-Schnittstelle initialisieren

Beschreibung

Mit dem Funktionsbaustein KRC_Initialize wird die mxA-Schnittstelle auf der Robotersteuerung initialisiert. Anweisungen können erst nach der Initialisierung der Schnittstelle übertragen werden.

Der Funktionsbaustein darf pro Achsgruppe nur einfach instanziiert werden. Bei einer mehrfachen Instanziierung werden die Signale des zuletzt aufgerufenen Funktionsbausteins ausgegeben.

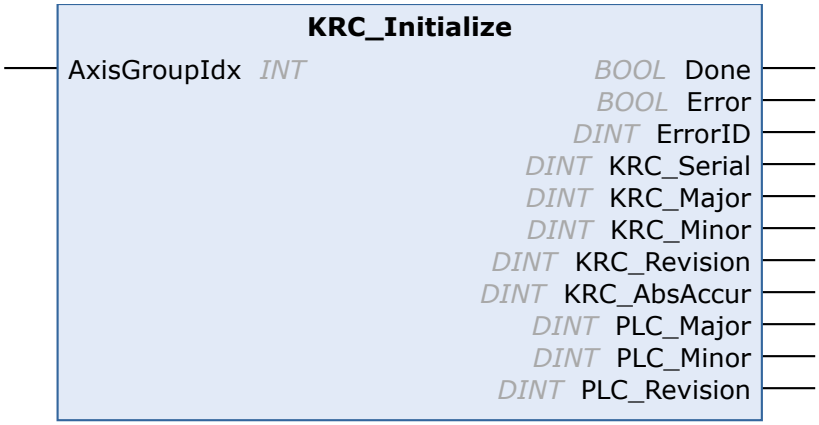


Abb. 73: Funktionsbaustein KRC_Initialize

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Initialisierung erfolgreich abgeschlossen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer
KRC_Serial	DINT	Seriennummer der Robotersteuerung
KRC_Major	DINT	Versionskennung der mxA-Schnittstelle (1. Stelle)
KRC_Minor	DINT	Versionskennung der mxA-Schnittstelle (2. Stelle)
KRC_Revision	DINT	Versionskennung der mxA-Schnittstelle (3. Stelle)
KRC_AbsAccur	DINT	Versionskennung des positioniergenauen Robotermodells • **1**: ACTIVE (= Positioniergenaues Robotermodell ist geladen und wird verwendet. Auf der RDC ist ein positioniergenaues Robotermodell gesichert.) • **2**: INACTIVE (= Positioniergenaues Robotermodell ist geladen, aber es wird nicht verwendet. Auf der RDC ist ein positioniergenaues Robotermodell gesichert, aber es ist aktuell deaktiviert (bis zum nächsten Kaltstart der Robotersteuerung).) • **3**: NONE (= Standard-Robotermodell. Es wird kein positioniergenaues Robotermodell verwendet. Auf der RDC ist kein positioniergenaues Robotermodell gesichert.)
PLC_Major	DINT	Versionskennung der SPS-Bibliothek (1. Stelle)
PLC_Minor	DINT	Versionskennung der SPS-Bibliothek (2. Stelle)
PLC_Revision	DINT	Versionskennung der SPS-Bibliothek (3. Stelle)

7.13.4 Programm-Override (POV) einstellen

Beschreibung

Mit dem Funktionsbaustein KRC_SetOverride wird der Programm-Override eingestellt.

Der Programm-Override ist die Geschwindigkeit des Roboters beim Programmablauf. Er wird in Prozent angegeben und bezieht sich auf die programmierte Geschwindigkeit. Der eingestellte Override wird mit jedem SPS-Zyklus an den Roboter übertragen. Bei einer Override-Änderung wird diese vom Roboter erkannt und übernommen.

Der Override wird nur in der Betriebsart Automatik Extern übernommen, damit in den Test-Betriebsarten T1 und T2 der Override über das smartPAD einstellt werden kann, z. B. zum Teachen.

Der Programm-Override bezieht sich auf alle Bewegungen des Robotersystems.

Der Funktionsbaustein darf pro Achsgruppe nur einfach instanziiert werden. Bei einer mehrfachen Instanziierung werden die Signale des zuletzt aufgerufenen Funktionsbausteins ausgegeben.

Abb. 74: Funktionsbaustein KRC_SetOverride

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
Override	INT	Programm-Override • **0 … 100 %**

Ausgänge

Parameter	Typ	Beschreibung
Valid	BOOL	TRUE = Daten sind gültig
ActualOverride	INT	Aktuell eingestellter Override • **0 … 100 %**
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.5 Automatik Extern-Signale der Robotersteuerung ansteuern und lesen

Beschreibung

Mit dem Funktionsbaustein KRC_AutomaticExternal wird die Schnittstelle Automatik Extern aktiviert und die Signale der Schnittstelle werden gelesen.

Zur vereinfachten Anwendung von KRC_AutomaticExternal kann der Funktionsbaustein KRC_AutoStart verwendet werden.

(>>> Eingänge von KRC_AutomaticExternal automatisch setzen [▶ 122])

Der Funktionsbaustein erfordert die Betriebsart Automatik Extern des Robotersystems. Weitere Informationen zu dieser Funktionalität sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.

Der Funktionsbaustein darf pro Achsgruppe nur einfach instanziiert werden. Bei einer mehrfachen Instanziierung werden die Signale des zuletzt aufgerufenen Funktionsbausteins ausgegeben.

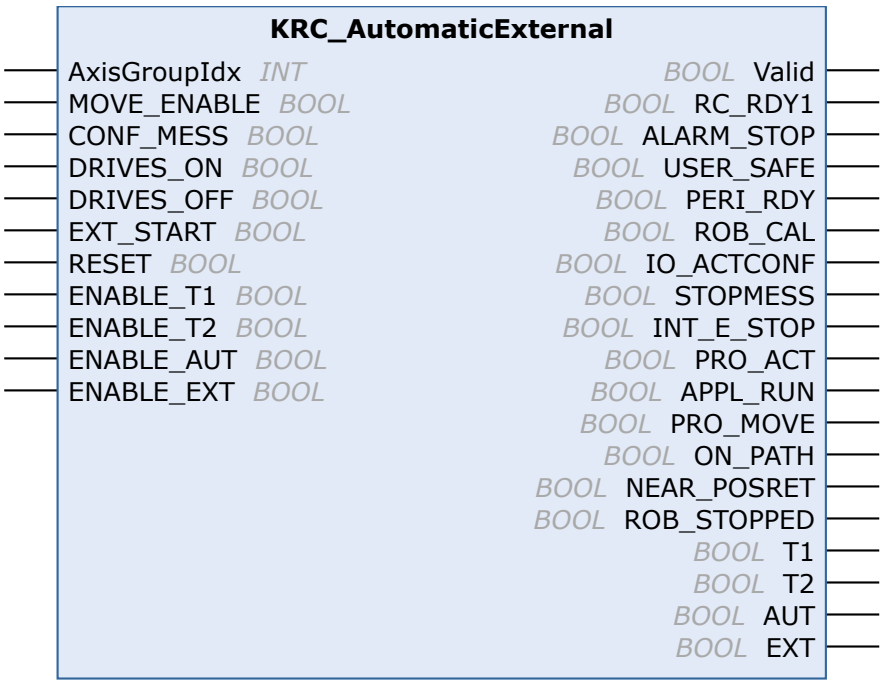


Abb. 75: Funktionsbaustein KRC_AutomaticExternal

Eingänge

Parameter	Typ	Signalname (Robotersteuerung)	Beschreibung
AxisGroupIdx	INT	—	Index der Achsgruppe • **1 … 5**
MOVE_ENABLE	BOOL	$MOVE_ENABLE	TRUE = Fahrfreigabe für den Roboter **Hinweis**: Diese Systemvariable wird von der Robotersteuerung in allen Betriebsarten überwacht.
CONF_MESS	BOOL	$CONF_MESS	TRUE = Quittieren von Fehlermeldungen
DRIVES_ON	BOOL	$DRIVES_ON	TRUE = Einschalten der Roboterantriebe
DRIVES_OFF	BOOL	$DRIVES_OFF	TRUE = Ausschalten der Roboterantriebe
EXT_START	BOOL	$EXT_START	TRUE = Starten oder Fortsetzen des Roboterprogramms
RESET	BOOL	—	Wählt das mxAutomation-Roboterprogramm bei einer steigenden Flanke des Signals an und startet es. Zuvor werden alle gepufferten Anweisungen abgebrochen.
ENABLE_T1	BOOL	—	TRUE = Freigabe der Betriebsart T1 Das Signal $MOVE_ENABLE wird bei fehlender Freigabe unterdrückt. Der Roboter kann nicht verfahren werden.
ENABLE_T2	BOOL	—	TRUE = Freigabe der Betriebsart T2 Das Signal $MOVE_ENABLE wird bei fehlender Freigabe unterdrückt. Der Roboter kann nicht verfahren werden.
ENABLE_AUT	BOOL	—	TRUE = Freigabe der Betriebsart Automatik Das Signal $MOVE_ENABLE wird bei fehlender Freigabe unterdrückt. Der Roboter kann nicht verfahren werden.
ENABLE_EXT	BOOL	—	TRUE = Freigabe der Betriebsart Automatik Extern

Parameter	Typ	Signalname (Robotersteuerung)	Beschreibung
			Das Signal $MOVE_ENABLE wird bei fehlender Freigabe unterdrückt. Der Roboter kann nicht verfahren werden.

Ausgänge

Parameter	Typ	Signalname (Robotersteuerung)	Beschreibung
Valid	BOOL	—	TRUE = Daten gültig
RC_RDY1	BOOL	$RC_RDY1	TRUE = Robotersteuerung bereit für Programmstart
ALARM_STOP	BOOL	$ALARM_STOP	FALSE = Roboterstopp durch NOT-HALT
USER_SAFE	BOOL	$USER_SAF	FALSE = Bedienerschutz verletzt
PERI_RDY	BOOL	$PERI_RDY	TRUE = Roboterantriebe eingeschaltet
ROB_CAL	BOOL	$ROB_CAL	TRUE = Roboterachsen justiert
IO_ACTCONF	BOOL	$IO_ACTCONF	TRUE = Schnittstelle Automatik Extern aktiv
STOPMESS	BOOL	$STOPMESS	TRUE = Sicherheitskreis unterbrochen (Roboterfehler)
INT_E_STOP	BOOL	Int. NotAus	TRUE = Externer NOT-HALT FALSE = NOT-HALT-Gerät am smartPAD gedrückt
PRO_ACT	BOOL	$PRO_ACT	TRUE = Prozess auf Roboterebene aktiv
APPL_RUN	BOOL	APPL_RUN	TRUE = Roboterprogramm läuft
PRO_MOVE	BOOL	$PRO_MOVE	TRUE = Synchrone Roboterbewegung aktiv
ON_PATH	BOOL	$ON_PATH	TRUE = Roboter auf programmierter Bahn
NEAR_POSRET	BOOL	$NEAR_POSRET	TRUE = Roboter nahe der zuletzt gespeicherten Position auf der programmierten Bahn (nach Verlassen der Bahn)
ROB_STOPPED	BOOL	$ROB_STOPPED	TRUE = Roboter steht
T1	BOOL	$T1	TRUE = Betriebsart T1 angewählt
T2	BOOL	$T2	TRUE = Betriebsart T2 angewählt
AUT	BOOL	$AUT	TRUE = Betriebsart Automatik angewählt
EXT	BOOL	$EXT	TRUE = Betriebsart Automatik Extern angewählt

7.13.6 Eingänge von KRC_AutomaticExternal automatisch setzen

Beschreibung

Mit dem Funktionsbaustein KRC_AutoStart werden die bereits vorhandenen Signale des Funktionsbausteins KRC_AutomaticExternal in der richtigen Reihenfolge automatisch gesetzt. Mit diesem Funktionsbaustein kann die Schnittstelle Automatik Extern des Robotersystems aktiviert werden, ohne dass tiefergehende Kenntnisse über die einzelnen Schritte der Aktivierung notwendig sind.

Die Signale zur Aktivierung der Schnittstelle Automatik Extern werden vor dem Start geprüft. Wenn eines oder mehrere Signale fehlen, werden entsprechende Fehlernummern ausgegeben.

Am Funktionsbaustein KRC_AutomaticExternal müssen zusätzlich folgende Eingänge gesetzt werden:

- MOVE_ENABLE
- ENABLE_T1
- ENABLE_T2
- ENABLE_AUT

- ENABLE_EXT
- DRIVES_OFF

Alle anderen Eingänge werden vom Funktionsbaustein KRC_AutoStart gesetzt.

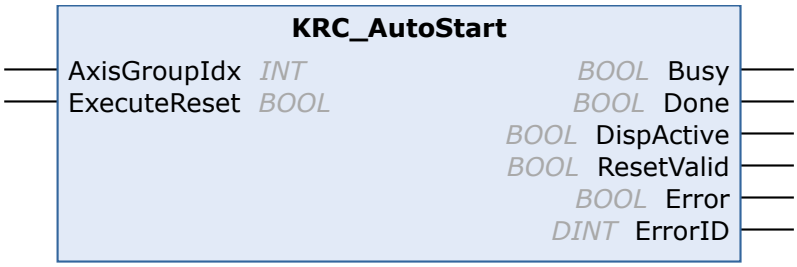


Abb. 76: Funktionsbaustein KRC_AutoStart

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteReset	BOOL	Setzt das Programm zurück und bricht alle gepufferten Anweisungen ab. mxAutomation ist nun bereit für neue Anweisungen. ExecuteReset muss den Wert TRUE haben, bis der Funktionsbaustein erfolgreich vom Roboter ausgeführt wurde.

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	Die Sequenz ist aktiv, aber noch nicht beendet.
Done	BOOL	Die Sequenz ist beendet.
DispActive	BOOL	TRUE = Roboterprogramm ist aktiv
ResetValid	BOOL	TRUE = Bedingungen für einen RESET am Funktionsbaustein KRC_AutomaticExternal sind erfüllt
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.7 Aktuelle Roboterposition lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadActualPosition wird die aktuelle kartesische Istposition des Roboters gelesen. Diese wird zyklisch aktualisiert.

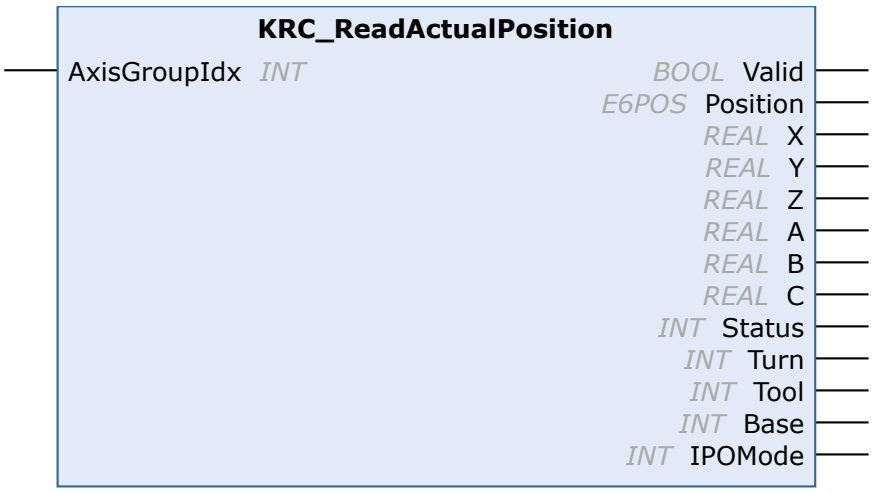


Abb. 77: Funktionsbaustein KRC_ReadActualPosition

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**

Ausgänge

Parameter	Typ	Beschreibung
Valid	BOOL	TRUE = Daten sind gültig
Position	E6POS	Aktuelle kartesische Istposition ($POS_ACT auf der Robotersteuerung) Die Datenstruktur E6POS enthält alle Komponenten der kartesischen Istposition (= Position des TCP bezogen auf den Ursprung des BASE-Koordinatensystems).
X, Y, Z	REAL	Aktuelle Istposition in X-, Y-, Z-Richtung
A, B, C	REAL	Orientierung A, B, C in der aktuellen Istposition
Status	INT	Status der aktuellen Istposition
Turn	INT	Turn der aktuellen Istposition
Tool	INT	Nummer des aktuell verwendeten TOOL-Koordinatensystems ($ACT_TOOL_C auf der Robotersteuerung)
Base	INT	Nummer des aktuell verwendeten BASE-Koordinatensystems ($ACT_BASE_C auf der Robotersteuerung)
IPOMode	INT	Aktueller Interpolationsmodus ($IPO_MODE_C auf der Robotersteuerung)

7.13.8 Aktuelle Achsposition lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadActualAxisPosition wird die aktuelle achsspezifische Roboterposition gelesen. Diese wird zyklisch aktualisiert.

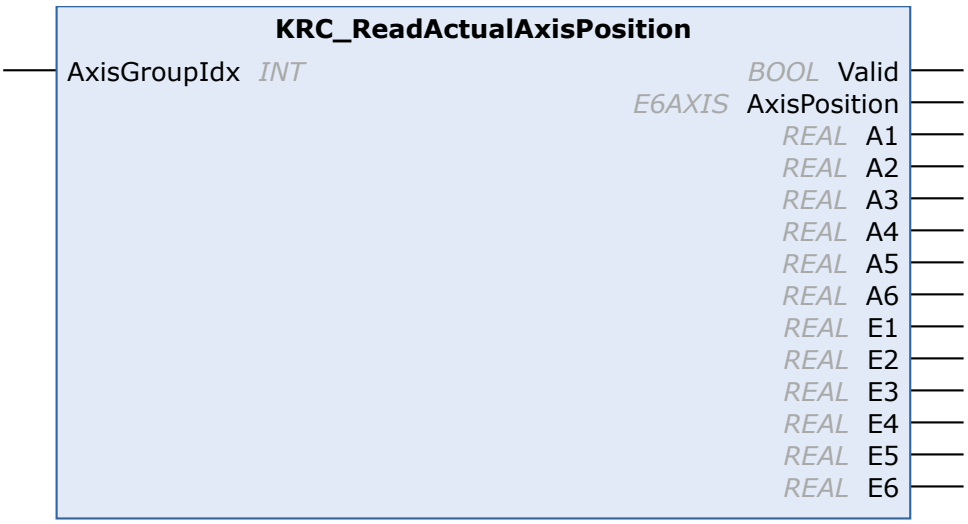


Abb. 78: Funktionsbaustein KRC_ReadActualAxisPosition

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**

Ausgänge

Parameter	Typ	Beschreibung
Valid	BOOL	TRUE = Daten sind gültig
AxisPosition	E6AXIS	Aktuelle achsspezifische Roboterposition ($AXIS_ACT auf der Robotersteuerung) Die Datenstruktur E6AXIS enthält alle Achspositionen der Achsgruppe.
A1 … A6	REAL	Aktuelle Position der Roboterachsen A1 … A6
E1 … E6	REAL	Aktuelle Position der Zusatzachsen E1 … E6

7.13.9 Aktuelle Bahngeschwindigkeit lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadActualVelocity wird die aktuelle Istgeschwindigkeit am TCP des Roboters gelesen. Diese wird zyklisch aktualisiert.

Die aktuelle Bahngeschwindigkeit kann nur bei CP-Bewegungen im Programmbetrieb gelesen werden.

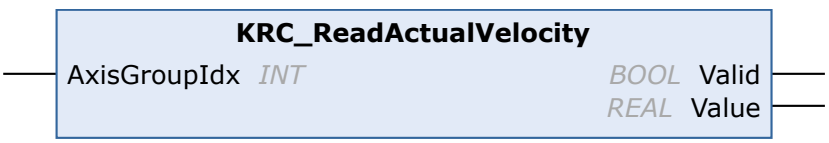


Abb. 79: Funktionsbaustein KRC_ReadActualVelocity

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**

Ausgänge

Parameter	Typ	Beschreibung
Valid	BOOL	TRUE = Daten sind gültig
Value	REAL	Aktuelle Bahngeschwindigkeit ($VEL_ACT auf der Robotersteuerung) Einheit: m/s

7.13.10 Aktuelle Achsgeschwindigkeit lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadActualAxisVelocity wird die aktuelle achsspezifische Geschwindigkeit des Roboters gelesen.

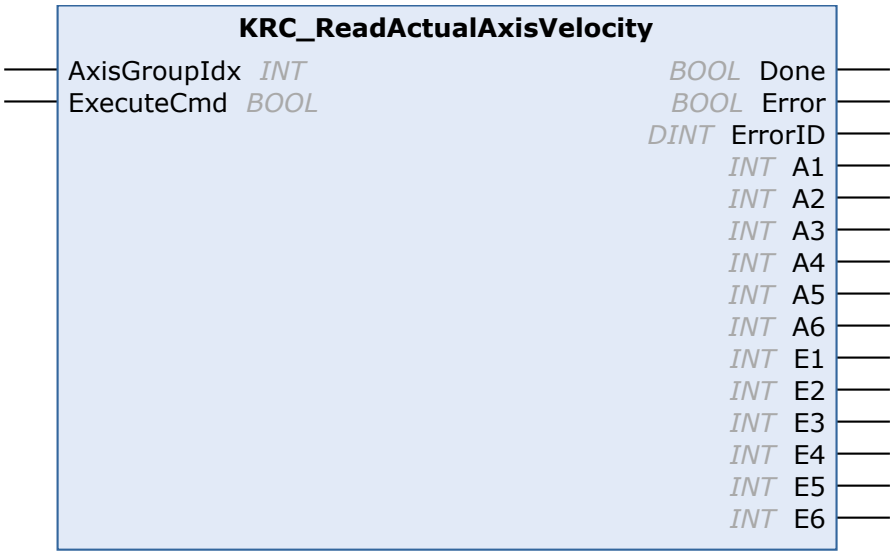


Abb. 80: Funktionsbaustein KRC_ReadActualAxisVelocity

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

Parameter	Typ	Beschreibung
A1 … A6	INT	Aktuelle Drehzahl des Motors (-100 % … +100 %) von A1 … A6, bezogen auf die maximale Drehzahl des Motors ($VEL_AXIS_MA auf der Robotersteuerung) **Hinweis**: Die tatsächlich resultierende Geschwindigkeit der Roboterachse ($VEL_AXIS_ACT auf der Robotersteuerung) ist abhängig von der Getriebeübersetzung.
E1 … E6	INT	Aktuelle Drehzahl des Motors (-100 % … +100 %) von E1 … E6, bezogen auf die maximale Drehzahl des Motors ($VEL_AXIS_MA auf der Robotersteuerung) **Hinweis**: Die tatsächlich resultierende Geschwindigkeit der Zusatzachse ist abhängig von der Getriebeübersetzung.

7.13.11 Aktuelle Roboterbeschleunigung lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadActualAcceleration wird die aktuelle kartesische Beschleunigung am TCP des Roboters gelesen.

Die aktuelle kartesische Beschleunigung um die Winkel A, B, C wird nicht ausgewertet.

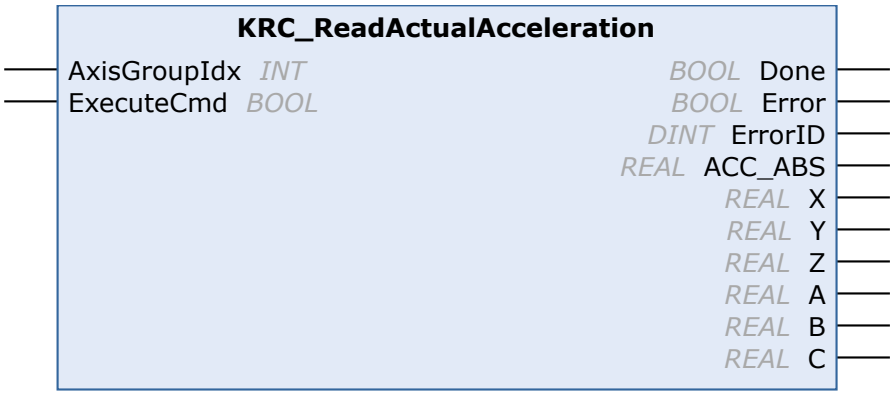


Abb. 81: Funktionsbaustein KRC_ReadActualAcceleration

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer
ACC_ABS	REAL	Aktuelle kartesische Beschleunigung bezogen auf den Betrag der Gesamtbeschleunigung ($ACC_CAR_ACT auf der Robotersteuerung)

Parameter	Typ	Beschreibung
		Einheit: m/s²
X, Y, Z	REAL	Aktuelle kartesische Beschleunigung in X-, Y-, Z-Richtung Einheit: m/s²
A, B, C	REAL	Aktuelle kartesische Beschleunigung um die Winkel A, B, C 0 m/s² (wird nicht berechnet)

7.13.12 Digitalen Eingang lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadDigitalInput wird ein digitaler Eingang der Robotersteuerung abgefragt und gelesen.

Abb. 82: Funktionsbaustein KRC_ReadDigitalInput

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
Number	INT	Nummer des digitalen Eingangs (entspricht $IN[1 … 2048] auf der Robotersteuerung) • **1 … 2 048**

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Value	BOOL	Wert des digitalen Eingangs
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.13 Digitalen Eingang 1 bis 8 lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadDigitalInput1To8 werden die digitalen Eingänge 1 bis 8 der Robotersteuerung abgefragt und gelesen.

Abb. 83: Funktionsbaustein KRC_ReadDigitalInput1To8

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**

Ausgänge

Parameter	Typ	Beschreibung
Valid	BOOL	TRUE = Daten sind gültig
IN1 … IN8	BOOL	Ist-Wert des digitalen Eingangs $IN[1] … $IN[8]
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.14 Mehrere digitale Eingänge lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadDigitalInputArray werden mehrere digitale Eingänge der Robotersteuerung abgefragt und gelesen.

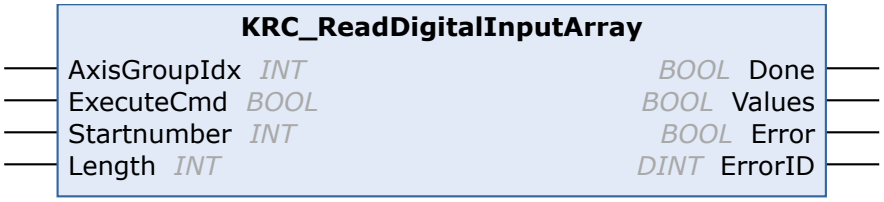


Abb. 84: Funktionsbaustein KRC_ReadDigitalInputArray

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
Startnumber	INT	Nummer des 1. digitalen Eingangs, der abgefragt wird (entspricht $IN[1 … 2048] auf der Robotersteuerung) • **1 … 2 048**

Parameter	Typ	Beschreibung
Length	INT	Anzahl der Eingänge, die abgefragt werden • **1 … 2 00** **Hinweis**: Wenn die Anzahl der Eingänge, die gelesen werden sollen, den Bereich 1 … 2048 überschreitet, wird keine Fehlermeldung ausgegeben. Eingänge außerhalb dieses Bereichs werden nicht eingelesen.

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Values	BOOL[200]	Werte der digitalen Eingänge
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.15 Digitalen Ausgang lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadDigitalOutput wird ein digitaler Ausgang der Robotersteuerung abgefragt und gelesen.

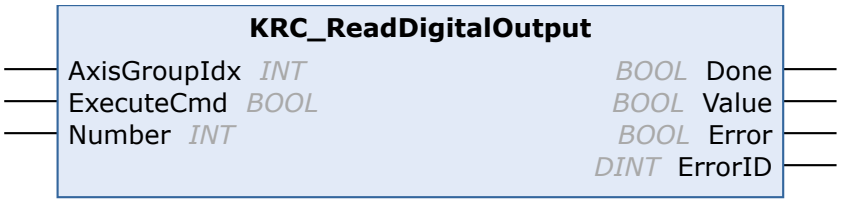


Abb. 85: Funktionsbaustein KRC_ReadDigitalOutput

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
Number	INT	Nummer des digitalen Ausgangs (entspricht $OUT[1 … 2048] auf der Robotersteuerung) • **1 … 2 048**

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Value	BOOL	Wert des digitalen Ausgangs
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.16 Digitalen Ausgang schreiben

Beschreibung

Mit dem Funktionsbaustein KRC_WriteDigitalOutput wird ein digitaler Ausgang oder ein Impulsausgang auf der Robotersteuerung geschrieben.

Abb. 86: Funktionsbaustein KRC_WriteDigitalOutput

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
Number	INT	Nummer des digitalen Ausgangs (entspricht $OUT[1 … 2048] auf der Robotersteuerung) • **1 … 2 048** **Hinweis**: Es ist darauf zu achten, dass keine Ausgänge verwendet werden, die bereits vom Robotersystem belegt sind. Beispiel: $OUT[1025] ist immer TRUE.
Value	BOOL	Wert des digitalen Ausgangs
Pulse	REAL	Länge des Impulses • **0.0 s** Kein Puls aktiv • **0.1 … 3.0 s** Pulsraster = 0.1 s
bContinue	BOOL	TRUE = Ausgang im Vorlauf beschreiben **Hinweis**: Die Robotersteuerung arbeitet Programme mit Vor- und Hauptlauf ab. Weitere Informationen zum Vor- und Hauptlauf sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.17 Digitalen Ausgang 1 bis 8 schreiben

Beschreibung

Mit dem Funktionsbaustein KRC_WriteDigitalOutput1To8 werden die digitalen Ausgänge 1 bis 8 auf der Robotersteuerung geschrieben. Die Ausgänge werden zyklisch geschrieben.

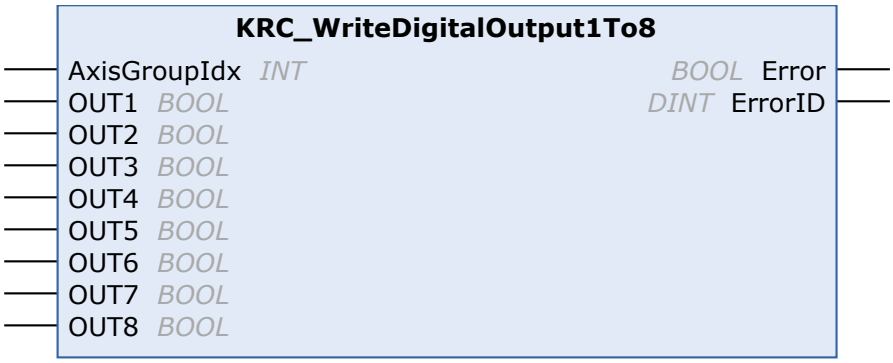


Abb. 87: Funktionsbaustein KRC_WriteDigitalOutput1To8

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
OUT1 … OUT8	BOOL	Soll-Wert des Ausgangs $OUT[1] … $OUT[8]

Ausgänge

Parameter	Typ	Beschreibung
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.18 Analogen Eingang lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadAnalogInput wird ein analoger Eingang der Robotersteuerung abgefragt und gelesen.

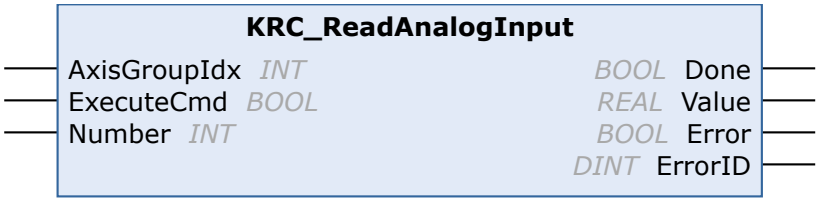


Abb. 88: Funktionsbaustein KRC_ReadAnalogInput

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
Number	INT	Nummer des analogen Eingangs ($ANIN[1 … 32] auf der Robotersteuerung) • **1 … 32**

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Value	REAL	Wert des analogen Eingangs
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.19 Analogen Ausgang lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadAnalogOutput wird ein analoger Ausgang der Robotersteuerung abgefragt und gelesen.

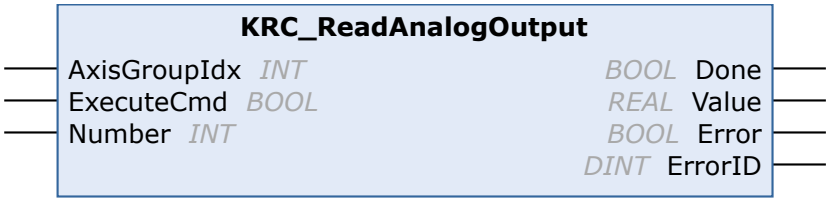


Abb. 89: Funktionsbaustein KRC_ReadAnalogOutput

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
Number	INT	Nummer des analogen Ausgangs ($ANOUT[1 … 32] auf der Robotersteuerung) • **1 … 32**

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Value	REAL	Wert des analogen Ausgangs
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.20 Analogen Ausgang schreiben

Beschreibung

Mit dem Funktionsbaustein KRC_WriteAnalogOutput wird ein analoger Ausgang der Robotersteuerung abgefragt und gelesen.

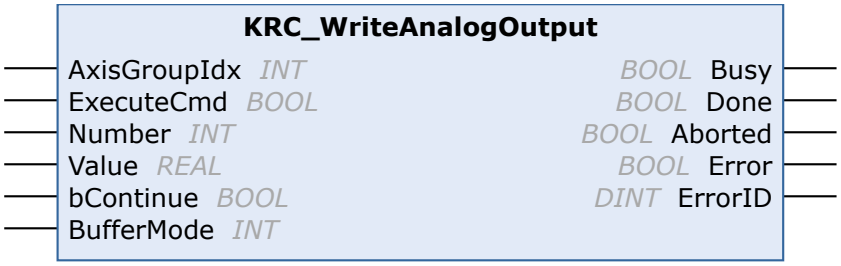


Abb. 90: Funktionsbaustein KRC_WriteAnalogOutput

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
Number	INT	Nummer des analogen Ausgangs ($ANOUT[1 … 32] auf der Robotersteuerung) • **1 … 32**
Value	REAL	Wert des analogen Ausgangs
bContinue	BOOL	TRUE = Ausgang im Vorlauf beschreiben **Hinweis**: Die Robotersteuerung arbeitet Programme mit Vor- und Hauptlauf ab. Weitere Informationen zum Vor- und Hauptlauf sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.21 Werkzeug, Basis und Interpolationsmodus auswählen

Beschreibung

Mit dem Funktionsbaustein KRC_SetCoordSys können Werkzeug, Basis und Interpolationsmodus gesetzt werden, ohne gleichzeitig eine Verfahrbewegung auszuführen. Diese Funktion wird beispielsweise benötigt, um die aktuelle Position in verschiedenen Koordinatensystemen auszulesen.

Weitere Informationen zu Werkzeug und Basis im Robotersystem sind in der Bedien- und Programmieranleitung der KUKA System Software (KSS) zu finden.

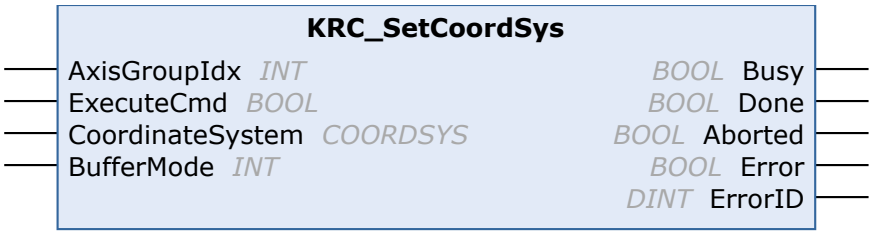


Abb. 91: Funktionsbaustein KRC_SetCoordSys

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
CoordinateSystem	COORDSYS	Koordinatensystem, auf das sich die Angaben beziehen (>>> COORDSYS [▶ 20])
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.22 TOOL-Daten lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadToolData werden die TOOL-Daten des Roboters gelesen.

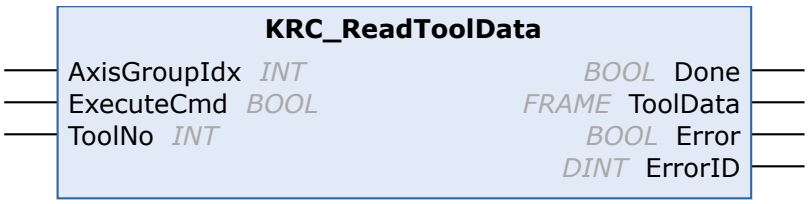


Abb. 92: Funktionsbaustein KRC_ReadToolData

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
ToolNo	INT	Nummer des TOOL-Koordinatensystems • **1 … 16**: TOOL_DATA[1 … 16]

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
ToolData	FRAME	Die Datenstruktur FRAME enthält folgende TOOL-Daten: • X, Y, Z: Ursprung des TOOL-Koordinatensystems, bezogen auf das FLANGE-Koordinatensystem • A, B, C: Orientierung des TOOL-Koordinatensystems, bezogen auf das FLANGE-Koordinatensystem (>>> FRAME [▶ 21])
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.23 TOOL-Daten schreiben

Beschreibung

Mit dem Funktionsbaustein KRC_WriteToolData werden die TOOL-Daten des Roboters geschrieben.

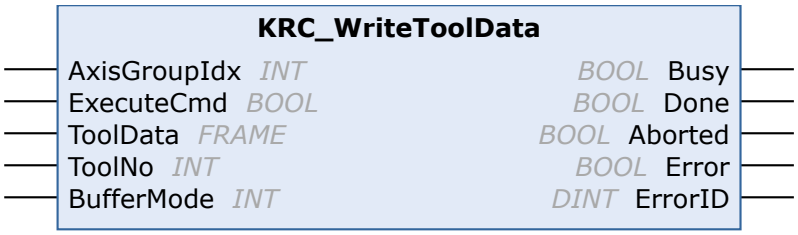


Abb. 93: Funktionsbaustein KRC_WriteToolData

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.

Parameter	Typ	Beschreibung
ToolData	FRAME	Die Datenstruktur FRAME enthält folgende TOOL-Daten: • X, Y, Z: Ursprung des TOOL-Koordinatensystems, bezogen auf das FLANGE-Koordinatensystem • A, B, C: Orientierung des TOOL-Koordinatensystems, bezogen auf das FLANGE-Koordinatensystem (>>> FRAME [▶ 21])
ToolNo	INT	Nummer des TOOL-Koordinatensystems • **1 … 16**: TOOL_DATA[1 … 16]
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.24 BASE-Daten lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadBaseData werden die BASE-Daten des Roboters gelesen.

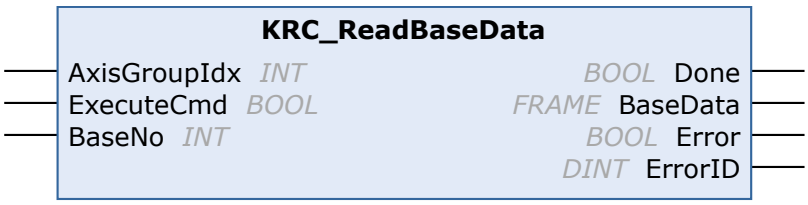


Abb. 94: Funktionsbaustein KRC_ReadBaseData

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
BaseNo	INT	Nummer des BASE-Koordinatensystems • **1 … 32**: BASE_DATA[1 … 32]

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
BaseData	FRAME	Die Datenstruktur FRAME enthält folgende BASE-Daten: • X, Y, Z: Ursprung des BASE-Koordinatensystems, bezogen auf das WORLD-Koordinatensystem • A, B, C: Orientierung des BASE-Koordinatensystems, bezogen auf das WORLD-Koordinatensystem (>>> FRAME [▶ 21])
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.25 BASE-Daten schreiben

Beschreibung

Mit dem Funktionsbaustein KRC_WriteBaseData werden die BASE-Daten des Roboters geschrieben.

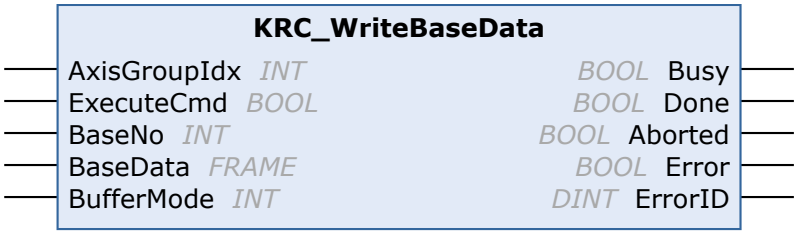


Abb. 95: Funktionsbaustein KRC_WriteBaseData

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
BaseNo	INT	Nummer des BASE-Koordinatensystems • **1 … 32**: BASE_DATA[1 … 32]
BaseData	FRAME	Die Datenstruktur FRAME enthält folgende BASE-Daten: • X, Y, Z: Ursprung des BASE-Koordinatensystems, bezogen auf das WORLD-Koordinatensystem • A, B, C: Orientierung des BASE-Koordinatensystems, bezogen auf das WORLD-Koordinatensystem (>>> FRAME [▶ 21])
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.26 Lastdaten lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadLoadData werden die Lastdaten des Roboters gelesen (Traglastdaten oder Zusatzlastdaten). Jedes Werkzeug der Robotersteuerung hat eigene Lastdaten. Die Zusatzlastdaten gelten für den gesamten Roboter.

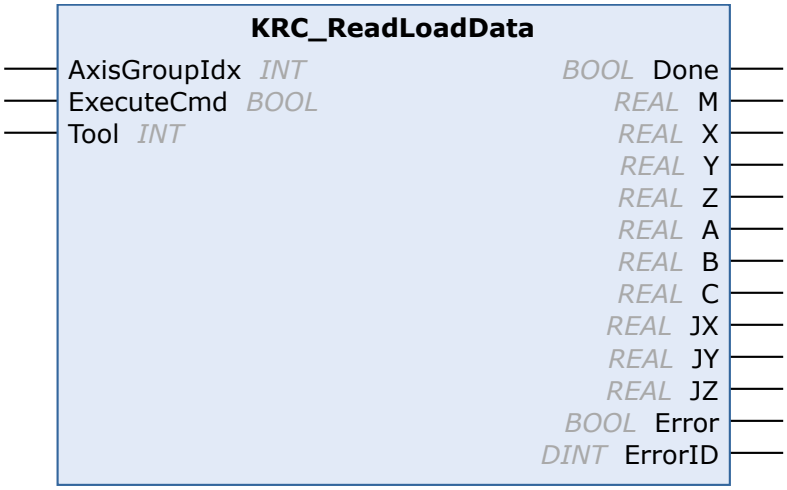


Abb. 96: Funktionsbaustein KRC_ReadLoadData

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
Tool	INT	Nummer des TOOL-Koordinatensystems zum Lesen der Traglastdaten oder Nummer zum Lesen der Zusatzlastdaten • **1 … 16**: TOOL_DATA[1 … 16] • **-1**: Zusatzlast A1 • **-2**: Zusatzlast A2 • **-3**: Zusatzlast A3

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
M	REAL	Masse

Parameter	Typ	Beschreibung
X, Y, Z	REAL	Lage des Schwerpunkts relativ zum Flansch
A, B, C	REAL	Orientierung der Hauptträgheitsachsen relativ zum Flansch
JX, JY, JZ	REAL	Massenträgheitsmomente
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.27 Lastdaten schreiben

Beschreibung

Mit dem Funktionsbaustein KRC_WriteLoadData werden die Lastdaten des Roboters geschrieben (Traglastdaten oder Zusatzlastdaten).

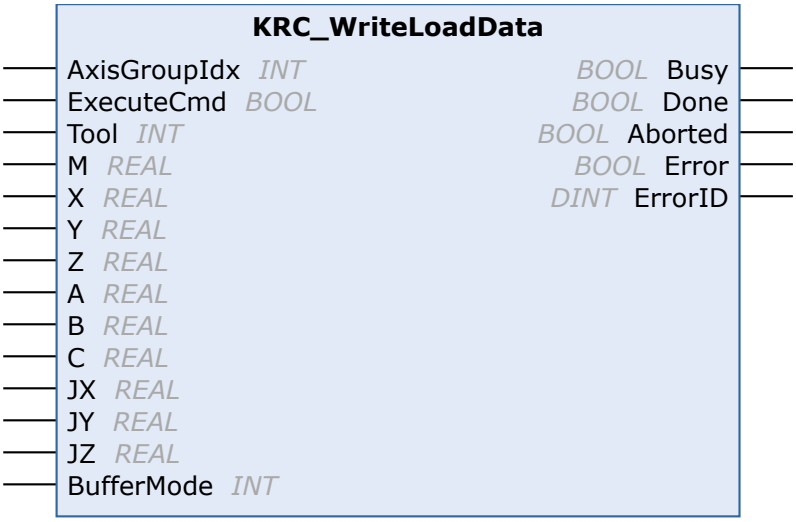


Abb. 97: Funktionsbaustein KRC_WriteLoadData

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
Tool	INT	Nummer des TOOL-Koordinatensystems zum Schreiben der Traglastdaten oder Nummer zum Schreiben der Zusatzlastdaten • **1 … 16**: TOOL_DATA[1 … 16] • **-1**: Zusatzlast A1 • **-2**: Zusatzlast A2 • **-3**: Zusatzlast A3
M	REAL	Masse
X, Y, Z	REAL	Lage des Schwerpunkts relativ zum Flansch
A, B, C	REAL	Orientierung der Hauptträgheitsachsen relativ zum Flansch
JX, JY, JZ	REAL	Massenträgheitsmomente
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED

Parameter	Typ	Beschreibung
		(>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.28 Software-Endschalter der Roboterachsen lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadSoftEnd werden die Software-Endschalter der Roboterachsen gelesen.

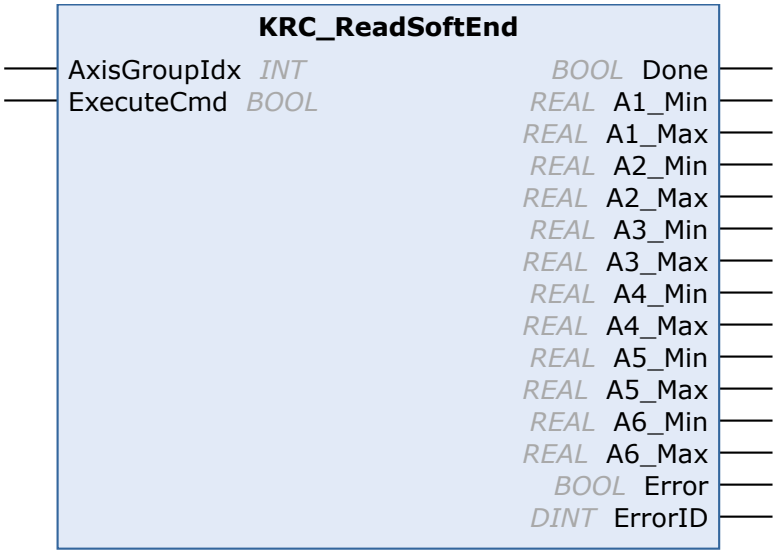


Abb. 98: Funktionsbaustein KRC_ReadSoftEnd

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
A1_Min … A6_Min	REAL	Negativer Software-Endschalter der Achse A1 … A6 Einheit: mm oder °
A1_Max … A6_Max	REAL	Positiver Software-Endschalter der Achse A1 … A6

Parameter	Typ	Beschreibung
		Einheit: mm oder °
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.29 Software-Endschalter der Zusatzachsen lesen

Beschreibung

Mit dem Funktionsbaustein KRC_ReadSoftEndExt werden die Software-Endschalter der Zusatzachsen gelesen.

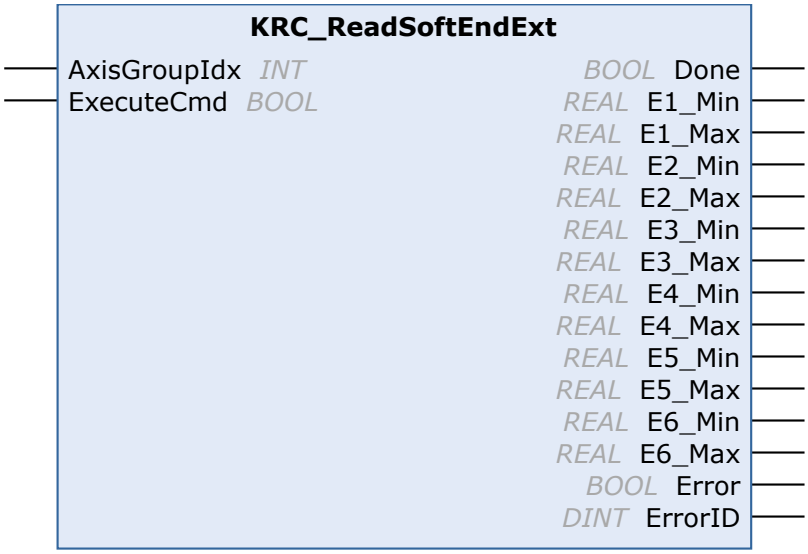


Abb. 99: Funktionsbaustein KRC_ReadSoftEndExt

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.

Ausgänge

Parameter	Typ	Beschreibung
Done	BOOL	TRUE = Anweisung wurde ausgeführt
E1_Min … E6_Min	REAL	Negativer Software-Endschalter der Achse E1 … E6 Einheit: mm oder °
E1_Max … E6_Max	REAL	Positiver Software-Endschalter der Achse E1 … E6 Einheit: mm oder °
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.30 Software-Endschalter der Roboterachsen schreiben

Beschreibung

Mit dem Funktionsbaustein KRC_WriteSoftEnd werden die Software-Endschalter der Roboterachsen geschrieben.

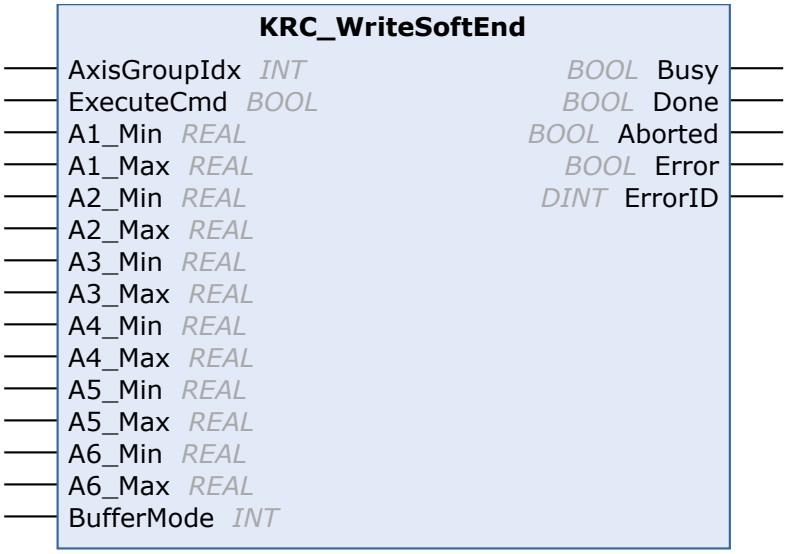


Abb. 100: Funktionsbaustein KRC_WriteSoftEnd

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
A1_Min … A6_Min	REAL	Negativer Software-Endschalter der Achse A1 … A6 Einheit: mm oder °
A1_Max … A6_Max	REAL	Positiver Software-Endschalter der Achse A1 … A6 Einheit: mm oder °
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

7.13.31 Software-Endschalter der Zusatzachsen schreiben

Beschreibung

Mit dem Funktionsbaustein KRC_WriteSoftEndExt werden die Software-Endschalter der Zusatzachsen geschrieben.

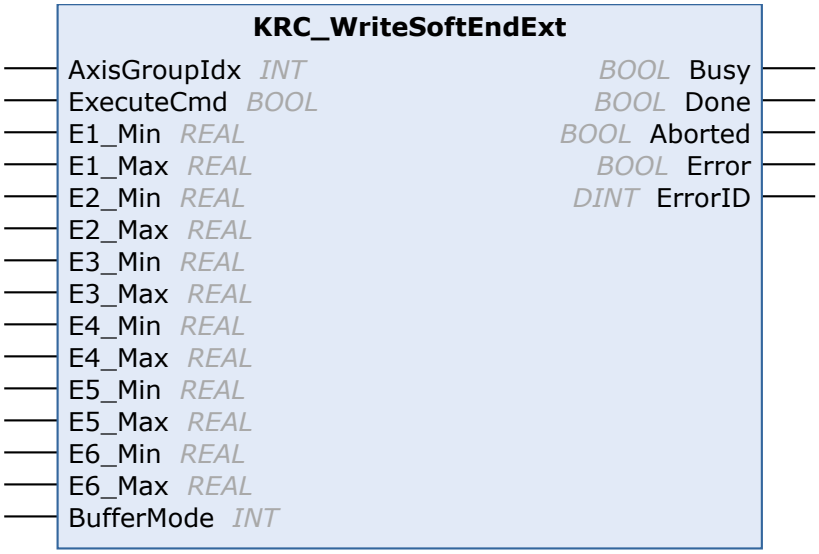


Abb. 101: Funktionsbaustein KRC_WriteSoftEndExt

Eingänge

Parameter	Typ	Beschreibung
AxisGroupIdx	INT	Index der Achsgruppe • **1 … 5**
ExecuteCmd	BOOL	Die Anweisung wird bei einer steigenden Flanke des Signals ausgeführt.
E1_Min … E6_Min	REAL	Negativer Software-Endschalter der Achse E1 … E6 Einheit: mm oder °
E1_Max … E6_Max	REAL	Positiver Software-Endschalter der Achse E1 … E6 Einheit: mm oder °
BufferMode	INT	Modus, in dem die Anweisung ausgeführt wird • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])

Ausgänge

Parameter	Typ	Beschreibung
Busy	BOOL	TRUE = Anweisung wird aktuell übertragen oder wurde bereits übertragen
Done	BOOL	TRUE = Anweisung wurde ausgeführt
Aborted	BOOL	TRUE = Anweisung wurde abgebrochen
Error	BOOL	TRUE = Fehler im Funktionsbaustein
ErrorID	DINT	Fehlernummer

8 Meldungen

8.1 Fehlermeldungen der mxA-Schnittstelle im Roboter-Interpreter

Nr.	Meldungstext	Ursache	Abhilfe
0	—	—	—
1	INTERNAL ERROR	Interner Ausnahmefehler	Service kontaktieren.
2	ASSERT FAILED	Interner Ausnahmefehler	
3	OVERFLOW STATUS RETURN QUEUE (MAIN)	Es sind mehr als 100 Rückmeldungen von Statusänderungen von der Robotersteuerung an die SPS zu übertragen. Die Übertragungsgeschwindigkeit ist wesentlich kleiner als die Verarbeitungsgeschwindigkeit.	Anzahl der Anweisungen, die gleichzeitig gepuffert werden, reduzieren. Service kontaktieren, wenn dies nicht möglich ist.
4	OVERFLOW STATUS RETURN QUEUE (TRIGGER)		
5	INVALID COMMAND QUEUE INDEX	Interner Ausnahmefehler	Service kontaktieren.
6	INVALID COMMAND STATE	Interner Ausnahmefehler	
7	INVALID COMMAND ID	Interner Ausnahmefehler	
8	INVALID MOVE TYPE	Interner Ausnahmefehler	
9	OVERFLOW TRIGGER FIFO	Interner Ausnahmefehler	
10	UNDERFLOW TRIGGER FIFO	Interner Ausnahmefehler	
11	INVALID TRIGGER FIFO INDEX	Interner Ausnahmefehler	
12	EXECUTION OF T_AFTER MISSING	Interner Ausnahmefehler	
13	EXECUTION OF T_START MISSING	Interner Ausnahmefehler	
14	INVALID ADVANCE_ACT	Interner Ausnahmefehler	
16	TIMEOUT HEARTBEAT FROM PLC	Verbindung zur SPS unterbrochen	Verbindung wiederherstellen, dann Fehler quittieren.
		SPS-Programm gestoppt	SPS-Programm neu starten.
		Verbindungsleitung defekt oder nicht korrekt angeschlossen	Verbindungsleitung austauschen oder korrekt anschließen.
17	INVALID ORDERID (INVERSE)	Interner Ausnahmefehler	Service kontaktieren.
30	INVALID PTP APO	Für eine PTP-Bewegung wurde ein ungültiger Überschleifparameter übergeben.	Gültigen Wert programmieren (Parameter **Approximate**). (>>> APO [▶ 19])
31	INVALID CP APO	Für eine CP-Bewegung (LIN, CIRC) wurde ein ein ungültiger Überschleifparameter übergeben.	
32	INVALID BASE NUMBER	Im Funktionsbaustein KRC_ReadBaseData oder KRC_WriteBaseData wurde eine ungültige Nummer für das BASE-Koordinatensystem programmiert.	Die Nummer des BASE-Koordinatensystems angeben, das aktuell in der Robotersteuerung verwendet wird (Parameter **BaseNo**).

Nr.	Meldungstext	Ursache	Abhilfe
			• **1 … 32**
		In einem KRC_Move- oder KRC_Jog-Funktionsbaustein wurde eine ungültige Nummer für das BASE-Koordinatensystem programmiert.	Die Nummer des BASE-Koordinatensystems angeben, das aktuell in der Robotersteuerung verwendet wird (Parameter **CoordinateSystem** - **COORDSYS.Base**). (>>> COORDSYS [▶ 20])
33	INVALID TOOL NUMBER	Im Funktionsbaustein KRC_ReadToolData oder KRC_WriteToolData wurde eine ungültige Nummer für das TOOL-Koordinatensystem programmiert.	Die Nummer des TOOL-Koordinatensystems angeben, das aktuell in der Robotersteuerung verwendet wird (Parameter **ToolNo**). • **1 … 16**
		In einem KRC_Move- oder KRC_Jog-Funktionsbaustein wurde eine ungültige Nummer für das TOOL-Koordinatensystem programmiert.	Die Nummer des TOOL-Koordinatensystems angeben, das aktuell in der Robotersteuerung verwendet wird (Parameter **CoordinateSystem** - **COORDSYS.Tool**). (>>> COORDSYS [▶ 20])
34	INVALID VELOCITY	In einem Funktionsbaustein wurde ein ungültiger Wert für die Geschwindigkeit programmiert.	Gültigen Wert programmieren (Parameter **Velocity**): • **0 … 100 %**
35	INVALID ACCELERATION	In einem Funktionsbaustein wurde ein ungültiger Wert für die Beschleunigung programmiert.	Gültigen Wert programmieren (Parameter **Acceleration**): • **0 … 100 %**
36	INVALID C_PTP	Für eine PTP-Bewegung wurde eine ungültige Überschleifdistanz übergeben.	Gültigen Wert programmieren (Parameter **Approximate**). (>>> APO [▶ 19])
37	INVALID C_DIS	Für eine überschliffene Bewegung wurde ein ungültiger Distanzparameter übergeben.	
38	INVALID C_VEL	Für eine überschliffene Bewegung wurde ein ungültiger Geschwindigkeitsparameter übergeben.	
39	INVALID C_ORI	Für eine überschliffene Bewegung wurde ein ungültiger Orientierungsparameter übergeben.	
40	INVALID ORI_TYPE	In einem KRC_Move- oder KRC_Jog-Funktionsbaustein wurde ein ungültiger Wert für die Orientierungsführung des TCP programmiert.	Gültigen Wert programmieren (Parameter **OriType**). (>>> OriType [▶ 16])
41	POSITION DATA NOT INITIALIZED	Beim Aufruf eines KRC_Move-Funktionsbausteins wurde keine Zielposition übergeben.	Mindestens 1 Element der Zielposition definieren (Parameter **Position**). (>>> E6POS [▶ 20])
42	AXISPOSITION DATA NOT INITIALIZED	Beim Aufruf eines KRC_MoveAxis-Funktionsbausteins wurde keine Achsposition übergeben.	Mindestens 1 Achsposition definieren (Parameter **AxisPosition**). (>>> E6AXIS [▶ 20])

Nr.	Meldungstext	Ursache	Abhilfe
43	INVALID TRIGGER DISTANCE	In einem Funktionsbaustein KRC_SetDistanceTrigger wurde ein ungültiger Wert für den Schaltpunkt des Triggers programmiert.	Gültigen Wert programmieren (Parameter **Distance**): • **0**: Schaltaktion im Startpunkt • **1**: Schaltaktion im Zielpunkt
44	INVALID TRIGGER IO	In einem Funktionsbaustein KRC_SetDistanceTrigger oder KRC_SetPathTrigger wurde ein ungültiger Ausgang programmiert.	Gültigen Wert programmieren (Parameter **Output**): • **1 … 2 048**
45	INVALID TRIGGER PULSE	In einem Funktionsbaustein KRC_SetDistanceTrigger oder KRC_SetPathTrigger wurde ein ungültiger Wert für die Länge des Impulses programmiert.	Gültigen Wert programmieren (Parameter **Pulse**): • **0.1 … 3.0 s** • **0.0 s** (Kein Puls aktiv)
46	INVALID CIRC_HP	Beim Aufruf eines KRC_MoveCirc-Funktionsbausteins wurde keine Hilfsposition übergeben.	Mindestens 1 Element der Hilfsposition definieren (Parameter **CircHP**). (>>> E6AXIS [▶ 20])
47	INVALID INTERRUPT IO	Die Nummer des digitalen Eingangs, auf den ein Interrupt deklariert ist, ist ungültig (Funktionsbaustein KRC_DeclareInterrupt).	Gültigen Wert programmieren (Parameter **Input**): • **1 … 2 048**
48	INVALID INTERRUPT PRIORITY	Beim Aufruf eines KRC_…Interrupt-Funktionsbausteins wurde eine ungültige Nummer übergeben.	Gültigen Wert programmieren (Parameter **Interrupt**): • **1 … 8**
49	INTERRUPT NOT DECLARED	Interrupt wurde nicht deklariert.	Interrupt deklarieren. (>>> Interrupt deklarieren [▶ 67])
50	INVALID INTERRUPT ACTION	Die Reaktion auf den Interrupt, die bei der Deklaration eines Interrupts programmiert wurde, ist ungültig.	Gültige Reaktion programmieren (Parameter **Reaction**) (>>> Interrupt deklarieren [▶ 67])
51	INVALID IO NUMBER	Die Nummer des digitalen Eingangs, auf den ein Interrupt deklariert ist, ist ungültig (Funktionsbaustein KRC_DeclareInterrupt).	Gültigen Wert programmieren (Parameter **Input**): • **1 … 2 048**
52	INVALID PULSE DURATION	Im Funktionsbaustein KRC_ WriteDigitalOutput wurde ein ungültiger Wert für die Länge des Impulses programmiert.	Gültigen Wert programmieren (Parameter **Pulse**): • **0.1 … 3.0 s** • **0.0 s** (Kein Puls aktiv)
53	INVALID BUFFER_MODE	In einem Funktionsbaustein wurde ein ungültiger **BufferMode** programmiert. Z. B. steht bei einigen Funktionsbausteinen der Modus DIRECT nicht Verfügung.	Gültigen **BufferMode** programmieren. (>>> BufferMode [▶ 24])
54	INVALID TOOL NUMBER FOR LOAD_DATA	Im Funktionsbaustein KRC_ReadLoadData oder KRC_WriteLoadData wurde eine ungültige Nummer zum Lesen oder Schreiben der Lastdaten oder der Zusatzlastdaten programmiert.	Gültigen Wert programmieren (Parameter **Tool**). (>>> Lastdaten lesen [▶ 139]) (>>> Lastdaten schreiben [▶ 140])
55	INVALID ANALOG IO NUMBER	In einem Funktionsbaustein wurde eine ungültige Nummer für den analogen Ein- oder Ausgang programmiert.	Gültigen Wert programmieren (Parameter **Number**): • **1 … 32**

Nr.	Meldungstext	Ursache	Abhilfe
56	INVALID IPO_MODE	In einem Funktionsbaustein wurde ein ungültiger Wert für den Interpolationsmodus programmiert, z. B. in einem KRC_Move-Funktionsbaustein.	Gültigen Wert programmieren (Parameter **CoordinateSystem** - **COORDSYS.IPO_MODE**). (>>> COORDSYS [▶ 20])
57	INVALID CIRC_TYPE	In einem KRC_MoveCirc-Funktionsbaustein wurde ein ungültiger Wert für die Orientierungsführung während der Kreisbewegung programmiert.	Gültigen Wert programmieren (Parameter **CircType**). (>>> CircType [▶ 16])
58	INVALID FRAME DATA	In einem KRC_WriteToolData- oder KRC_WriteBaseData-Funktionsbaustein wurden ungültige TOOL- oder BASE-Daten programmiert.	Gültige Daten programmieren (Parameter **ToolData** oder **BaseData**). (>>> TOOL-Daten schreiben [▶ 136]) (>>> BASE-Daten schreiben [▶ 138])
59	INVALID LOAD DATA	In einem KRC_WriteLoadData-Funktionsbaustein wurden ungültige Lastdaten programmiert.	Gültige Daten programmieren. (>>> Lastdaten schreiben [▶ 140])
60	INVALID SOFT_END (REVERSED)	Fehler beim Schreiben der Software-Endschalter: Positiver Software-Endschalter < negativer Software-Endschalter (Funktionsbaustein KRC_WriteSoftEnd oder KRC_WriteSoftEndEx)	Für die negativen Software-Endschalter kleinere Wert programmieren als für die positiven Software-Endschalter.
61	INVALID INTERRUPT STATE	Interner Ausnahmefehler	Service kontaktieren.
62	INVALID SYS VAR INDEX	In einem KRC_ReadSysVar- oder KRC_WriteSysVar-Funktionsbaustein wurde ein Index übergeben, für den keine Systemvariable hinterlegt ist.	Gültigen Wert programmieren (Parameter **Index**). (>>> Systemvariablen lesen [▶ 80]) (>>> Systemvariablen schreiben [▶ 81])
63	INVALID SYS VAR VALUE	In einem KRC_WriteSysVar-Funktionsbaustein wurde ein ungültiger Wert für die Systemvariable programmiert.	Gültigen Wert programmieren (Parameter **Value1 … Value10**). (>>> Systemvariablen schreiben [▶ 81])
64	SYS VAR NOT WRITEABLE	Beim Schreiben einer Systemvariablen ist ein Fehler aufgetreten. Die angegebene Systemvariable existiert nicht oder darf im aktuellen Betriebszustand nicht beschrieben werden.	
65	INVALID REAL VALUE	Der programmierte Real-Wert ist ungültig.	Gültigen Wert programmieren: • **-2.147.483.500 … +2.147.483.500**
66	ERROR SETTING OUTPUT	Fehler beim Schreiben eines digitalen Ausgangs. Möglicherweise ist der Ausgang bereits vom System belegt.	Einen anderen digitalen Ausgang verwenden (Parameter **Number**): • **1 … 2 048**
67	ERROR SETTING SOFTEND	Fehler beim Schreiben der Software-Endschalter: Ein möglicher Fehler ist z. B, dass eine rotatorische Achse mit einem Wert außerhalb +/-360° beschrieben wird.	Gültige Werte für die Software-Endschalter programmieren (siehe Maschinendaten).

Nr.	Meldungstext	Ursache	Abhilfe
68	INVALID TECH FUNCTION INDEX	In einem KRC_TechFunction-Funktionsbaustein wurde eine TechFunctionID übergeben, für die keine Technologiefunktion hinterlegt ist.	Service kontaktieren.)
69	INVALID TECH FUNCTION PARAMETER	In einem KRC_TechFunction-Funktionsbaustein wurde ein ungültiger Wert für einen Parameter programmiert.	Service kontaktieren.
70	INVALID PARAMETER VALUE	Im aufgerufenen Funktionsbaustein wurde ein ungültiger Wert für einen oder mehrere Parameter programmiert.	Gültige Werte für die Parameter programmieren.

8.2 Fehlermeldungen der mxA-Schnittstelle im Submit-Interpreter

Nr.	Meldungstext	Ursache	Abhilfe
401	INTERNAL ERROR	Interner Ausnahmefehler	Service kontaktieren.
402	ASSERT FAILED	Interner Ausnahmefehler	
403	INVALID COMMAND ID	Interner Ausnahmefehler	
404	INVALID COMMAND STATE	Interner Ausnahmefehler	
405	OVERFLOW COMMAND QUEUE	Interner Ausnahmefehler	
406	INVALID COMMAND QUEUE INDEX	Interner Ausnahmefehler	
407	INVALID COMMAND (PRE) QUEUE INDEX	Interner Ausnahmefehler	
408	INVALID WRITE_Q_IDX AND WRITE_PRE_Q_IDX	Interner Ausnahmefehler	
409	OVERFLOW STATUS RETURN QUEUE (SUBMIT)	Es sind mehr als 100 Rückmeldungen von Statusänderungen von der Robotersteuerung an die SPS zu übertragen. Die Übertragungsgeschwindigkeit ist wesentlich kleiner als die Verarbeitungsgeschwindigkeit.	Anzahl der Anweisungen, die gleichzeitig gepuffert werden, reduzieren. Service kontaktieren, wenn dies nicht möglich ist.
410	INVALID FIELDBUS TELEGRAMM LENGTH	Interner Ausnahmefehler	Service kontaktieren.
411	TIMEOUT ABORT_REQUEST	Interner Ausnahmefehler	
412	INVALID CHECKSUM PLC -> KRC	Die Prüfsumme bei der Datenübertragung von der SPS an die Robotersteuerung ist ungültig:	
		Fehler bei der Inbetriebnahme: • EtherCAT-Konfiguration in WorkVisual oder TwinCAT fehlerhaft	Konfiguration in WorkVisual und TwinCAT überprüfen und EtherCAT korrekt konfigurieren.
		Fehler im laufenden Betrieb: • Bitfehler bei der Datenübertragung	Service kontaktieren.
413	INVALID MOVE TYPE	Interner Ausnahmefehler	Service kontaktieren.

Nr.	Meldungstext	Ursache	Abhilfe
414	TIMEOUT HEARTBEAT FROM PLC	Verbindung zur SPS unterbrochen	Verbindung wiederherstellen, dann Fehler quittieren.
		SPS-Programm gestoppt	SPS-Programm neu starten.
		Submit-Interpeter abgewählt oder gestoppt	Submit-Interpeter neu starten.
		Verbindungsleitung defekt oder nicht korrekt angeschlossen	Verbindungsleitung austauschen oder korrekt anschließen.
416	SYS VAR NOT INITIALIZED	Beim Lesen einer Systemvariablen ist ein Fehler aufgetreten. Die angegebene Systemvariable existiert nicht oder darf im aktuellen Betriebszustand nicht gelesen werden. **Beispiel**: Auf $POS_ACT kann erst nach einer SAK-Fahrt zugegriffen werden.	
417	UNDERFLOW OF NIBBLE	Interner Ausnahmefehler	Service kontaktieren.
418	OVERFLOW OF NIBBLE	Interner Ausnahmefehler	
419	UNDERFLOW OF BYTE	Interner Ausnahmefehler	
420	OVERFLOW OF BYTE	Interner Ausnahmefehler	
421	UNDERFLOW OF INT16	Interner Ausnahmefehler	
422	OVERFLOW OF INT16	Interner Ausnahmefehler	
423	UNDERFLOW OF INT32	Interner Ausnahmefehler	
424	OVERFLOW OF INT32	Interner Ausnahmefehler	
425	UNDERFLOW OF REAL	Interner Ausnahmefehler	
426	OVERFLOW OF REAL	Interner Ausnahmefehler	
430	INVALID PTP APO	Für eine PTP-Bewegung wurde ein ungültiger Überschleifparameter übergeben.	Gültigen Wert programmieren (Parameter **Approximate**). (>>> APO [▶ 19])
431	INVALID CP APO	Für eine CP-Bewegung (LIN, CIRC) wurde ein ein ungültiger Überschleifparameter übergeben.	
432	INVALID BASE NUMBER	Im Funktionsbaustein KRC_ReadBaseData oder KRC_WriteBaseData wurde eine ungültige Nummer für das BASE-Koordinatensystem programmiert.	Die Nummer des BASE-Koordinatensystems angeben, das aktuell in der Robotersteuerung verwendet wird (Parameter **BaseNo**). • **1 … 32**
		In einem KRC_Move- oder KRC_Jog-Funktionsbaustein wurde eine ungültige Nummer für das BASE-Koordinatensystem programmiert.	Die Nummer des BASE-Koordinatensystems angeben, das aktuell in der Robotersteuerung verwendet wird (Parameter **CoordinateSystem** - **COORDSYS.Base**). (>>> COORDSYS [▶ 20])
433	INVALID TOOL NUMBER	Im Funktionsbaustein KRC_ReadToolData oder KRC_WriteToolData wurde eine ungültige Nummer für das TOOL-Koordinatensystem programmiert.	Die Nummer des TOOL-Koordinatensystems angeben, das aktuell in der Robotersteuerung verwendet wird (Parameter **ToolNo**). • **1 … 16**

Nr.	Meldungstext	Ursache	Abhilfe
		In einem KRC_Move- oder KRC_Jog-Funktionsbaustein wurde eine ungültige Nummer für das TOOL-Koordinatensystem programmiert.	Die Nummer des TOOL-Koordinatensystems angeben, das aktuell in der Robotersteuerung verwendet wird (Parameter **CoordinateSystem** - **COORDSYS.Tool**). (>>> COORDSYS [▶ 20])
434	INVALID VELOCITY	In einem Funktionsbaustein wurde ein ungültiger Wert für die Geschwindigkeit programmiert.	Gültigen Wert programmieren (Parameter **Velocity**): • **0 … 100 %**
435	INVALID ACCELERATION	In einem Funktionsbaustein wurde ein ungültiger Wert für die Beschleunigung programmiert.	Gültigen Wert programmieren (Parameter **Acceleration**): • **0 … 100 %**
436	INVALID C_PTP	Für eine PTP-Bewegung wurde eine ungültige Überschleifdistanz übergeben.	Gültigen Wert programmieren (Parameter **Approximate**). (>>> APO [▶ 19])
437	INVALID C_DIS	Für eine überschliffene Bewegung wurde ein ungültiger Distanzparameter übergeben.	
438	INVALID C_VEL	Für eine überschliffene Bewegung wurde ein ungültiger Geschwindigkeitsparameter übergeben.	
439	INVALID C_ORI	Für eine überschliffene Bewegung wurde ein ungültiger Orientierungsparameter übergeben.	
440	INVALID ORI_TYPE	In einem KRC_Move- oder KRC_Jog-Funktionsbaustein wurde ein ungültiger Wert für die Orientierungsführung des TCP programmiert.	Gültigen Wert programmieren (Parameter **OriType**). (>>> OriType [▶ 16])
441	POSITION DATA NOT INITIALIZED	Beim Aufruf eines KRC_Move-Funktionsbausteins wurde keine Zielposition übergeben.	Mindestens 1 Element der Zielposition definieren (Parameter **Position**). (>>> E6POS [▶ 20])
442	AXISPOSITION DATA NOT INITIALIZED	Beim Aufruf eines KRC_MoveAxis-Funktionsbausteins wurde keine Achsposition übergeben.	Mindestens 1 Achsposition definieren (Parameter **AxisPosition**). (>>> E6AXIS [▶ 20])
443	INVALID TRIGGER DISTANCE	Im Funktionsbaustein KRC_SetDistanceTrigger wurde ein ungültiger Wert für den Schaltpunkt des Triggers programmiert.	Gültigen Wert programmieren (Parameter **Distance**): • **0**: Schaltaktion im Startpunkt • **1**: Schaltaktion im Zielpunkt
444	INVALID TRIGGER IO	In einem Funktionsbaustein KRC_SetDistanceTrigger oder KRC_SetPathTrigger wurde ein ungültiger Ausgang programmiert.	Gültigen Wert programmieren (Parameter **Output**): • **1 … 2 048**
445	INVALID TRIGGER PULSE	In einem Funktionsbaustein KRC_SetDistanceTrigger oder KRC_SetPathTrigger wurde ein ungültiger Wert für die Länge des Impulses programmiert.	Gültigen Wert programmieren (Parameter **Pulse**): • **0.1 … 3.0 s** • **0.0 s** (Kein Puls aktiv)
446	INVALID CIRC_HP	Beim Aufruf eines KRC_MoveCirc-Funktionsbausteins wurde keine Hilfsposition übergeben.	Mindestens 1 Element der Hilfsposition definieren (Parameter **CircHP**).

Nr.	Meldungstext	Ursache	Abhilfe
			(>>> E6AXIS [▶ 20])
447	INVALID INTERRUPT IO	Die Nummer des digitalen Eingangs, auf den ein Interrupt deklariert ist, ist ungültig (Funktionsbaustein KRC_DeclareInterrupt).	Gültigen Wert programmieren (Parameter **Input**): • **1 … 2 048**
448	INVALID INTERRUPT NUMBER/ PRIORITY	Beim Aufruf eines KRC_…Interrupt-Funktionsbausteins wurde eine ungültige Nummer übergeben.	Gültigen Wert programmieren (Parameter **Interrupt**): • **1 … 8**
449	INTERRUPT NOT DECLARED	Interrupt wurde nicht deklariert.	Interrupt deklarieren. (>>> Interrupt deklarieren [▶ 67])
450	INVALID INTERRUPT ACTION	Die Reaktion auf den Interrupt, die bei der Deklaration eines Interrupts programmiert wurde, ist ungültig.	Gültige Reaktion programmieren (Parameter **Reaction**) (>>> Interrupt deklarieren [▶ 67])
451	INVALID IO NUMBER	Die Nummer des digitalen Eingangs, auf den ein Interrupt deklariert ist, ist ungültig (Funktionsbaustein KRC_DeclareInterrupt).	Gültigen Wert programmieren (Parameter **Input**): • **1 … 2 048**
452	INVALID PULSE DURATION	Im Funktionsbaustein KRC_ WriteDigitalOutput wurde ein ungültiger Wert für die Länge des Impulses programmiert.	Gültigen Wert programmieren (Parameter **Pulse**): • **0.1 … 3.0 s** • **0.0 s** (Kein Puls aktiv) (>>> Digitalen Ausgang schreiben [▶ 131])
453	INVALID BUFFER_MODE	In einem Funktionsbaustein wurde ein ungültiger **BufferMode** programmiert. Z. B. steht bei einigen Funktionsbausteinen der Modus DIRECT nicht Verfügung.	Gültigen **BufferMode** programmieren. (>>> BufferMode [▶ 24])
454	INVALID TOOL NUMBER FOR LOAD_DATA	Im Funktionsbaustein KRC_ReadLoadData oder KRC_WriteLoadData wurde eine ungültige Nummer zum Lesen oder Schreiben der Lastdaten oder der Zusatzlastdaten programmiert.	Gültigen Wert programmieren (Parameter **Tool**). (>>> Lastdaten lesen [▶ 139]) (>>> Lastdaten schreiben [▶ 140])
455	INVALID ANALOG IO NUMBER	In einem Funktionsbaustein wurde eine ungültige Nummer für den Analogeingang oder -ausgang programmiert.	Gültige Nummer programmieren (Parameter **Number**): • **1 … 32**
456	INVALID IPO_MODE	In einem Funktionsbaustein wurde ein ungültiger Wert für den Interpolationsmodus programmiert, z. B. in einem KRC_Move-Funktionsbaustein.	Gültigen Wert programmieren (Parameter **CoordinateSystem** - **COORDSYS.IPO_MODE**). (>>> COORDSYS [▶ 20])
457	INVALID CIRC_TYPE	In einem KRC_MoveCirc-Funktionsbaustein wurde ein ungültiger Wert für die Orientierungsführung während der Kreisbewegung programmiert.	Gültigen Wert programmieren (Parameter **CircType**). (>>> CircType [▶ 16])
458	INVALID FRAME DATA	In einem KRC_WriteToolData- oder KRC_WriteBaseData-Funktionsbaustein wurden ungültige TOOL- oder BASE-Daten programmiert.	Gültige Daten programmieren (Parameter **ToolData** oder **BaseData**). (>>> TOOL-Daten schreiben [▶ 136]) (>>> BASE-Daten schreiben [▶ 138])

Nr.	Meldungstext	Ursache	Abhilfe
459	INVALID LOAD DATA	In einem KRC_WriteLoadData-Funktionsbaustein wurden ungültige Lastdaten programmiert.	Gültige Daten programmieren. (>>> Lastdaten schreiben [▶ 140])
460	INVALID SOFT_END (REVERSED)	Fehler beim Schreiben der Software-Endschalter: Positiver Software-Endschalter < negativer Software-Endschalter	Für die negativen Software-Endschalter kleinere Wert programmieren als für die positiven Software-Endschalter.
461	INVALID INTERRUPT STATE	Interner Ausnahmefehler	Service kontaktieren.
462	INVALID SYS VAR INDEX	In einem KRC_ReadSysVar- oder KRC_WriteSysVar-Funktionsbaustein wurde ein Index übergeben, für den keine Systemvariable hinterlegt ist.	Gültigen Wert programmieren (Parameter **Index**). (>>> Systemvariablen lesen [▶ 80]) (>>> Systemvariablen schreiben [▶ 81])
463	INVALID SYS VAR VALUE	In einem KRC_WriteSysVar-Funktionsbaustein wurde ein ungültiger Wert für die Systemvariable programmiert.	Gültigen Wert programmieren (Parameter **Value1 … Value10**). (>>> Systemvariablen schreiben [▶ 81])
464	SYS VAR NOT WRITEABLE	Beim Schreiben einer Systemvariablen ist ein Fehler aufgetreten. Die angegebene Systemvariable existiert nicht oder darf im aktuellen Betriebszustand nicht beschrieben werden.	
465	INVALID REAL VALUE	Der programmierte Real-Wert ist ungültig.	Gültigen Wert programmieren: • **-2.147.483.500 … +2.147.483.500**
466	ERROR SETTING OUTPUT	Fehler beim Schreiben eines Ausgangs. Möglicherweise ist der Ausgang bereits vom System belegt.	Einen anderen digitalen Ausgang verwenden (Parameter **Number**): • **1 … 2 048**
467	ERROR SETTING SOFTEND	Beim Schreiben eines Software-Endschalters ist ein Fehler aufgetreten. Ein möglicher Fehler ist z. B, dass eine rotatorische Achse mit einem Wert außerhalb +/-360° beschrieben wird.	Gültige Werte für die Software-Endschalter programmieren (siehe Maschinendaten).
468	INVALID TECH FUNCTION INDEX	In einem KRC_TechFunction-Funktionsbaustein wurde eine TechFunctionID übergeben, für die keine Technologiefunktion hinterlegt ist.	Service kontaktieren.
469	INVALID TECH FUNCTION PARAMETER	In einem KRC_TechFunction-Funktionsbaustein wurde ein ungültiger Wert für einen Parameter programmiert.	Service kontaktieren.
470	INVALID PARAMETER VALUE	Im aufgerufenen Funktionsbaustein wurde ein ungültiger Wert für einen oder mehrere Parameter programmiert.	Gültige Werte für die Parameter programmieren.
471	PDAT NOT INITIALIZED	Beim Aufruf eines KRC_ForwardAdvanced oder KRC_InverseAdvanced Funktionsbausteins wurden keine PDAT-Daten übergeben.	Mindestens 1 Element der PDAT_ACT definieren.

Nr.	Meldungstext	Ursache	Abhilfe
472	FDAT NOT INITIALIZED	Beim Aufruf eines KRC_ForwardAdvanced oder KRC_InverseAdvanced Funktionsbausteins wurden keine FDAT-Daten übergeben.	Mindestens 1 Element der FDAT_ACT definieren.
473	LDAT NOT INITIALIZED	Beim Aufruf eines KRC_ForwardAdvanced oder KRC_InverseAdvanced Funktionsbausteins wurden keine LDAT-Daten übergeben.	Mindestens 1 Element der LDAT_ACT definieren.
474	INVALID TOOL OR BASE NUMBER	Im Funktionsbaustein KRC_ForwardAdvanced oder KRC_InverseAdvanced wurde eine ungültige Nummer für das TOOL-Koordinatensystem oder für das BASE-Koordinatensystem programmiert.	Die Nummer des TOOL-Koordinatensystems angeben, das aktuell in der Robotersteuerung verwendet wird (Parameter **Tool**). • **1 … 16** Die Nummer des BASE-Koordinatensystems angeben, das aktuell in der Robotersteuerung verwendet wird (Parameter **Base**). • **1 … 32**

8.3 Fehler im Funktionsbaustein

Nr.	Meldungstext	Ursache	Abhilfe
501	INTERNAL ERROR	Interner Ausnahmefehler	Service kontaktieren.
502	INVALID BUFFER_MODE	BufferMode **0**: DIRECT ist für diesen Funktionsblock nicht zulässig.	Richtigen Modus programmieren: • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 24])
503	INVALID MXA VERSION	Die Softwareversionen der mxA-Schnittstelle und der SPS-Bibliothek sind nicht kompatibel.	Kompatible Softwareversionen auf Robotersteuerung und SPS installieren. (>>> mxA-Schnittstelle initialisieren [▶ 118])
504	INVALID OVERRIDE	Ungültiger Override-Wert im Funktionsbaustein KRC_SetOverride	Einen gültigen Wert programmieren (Parameter **Override**): • **0 … 100%**
505	MAX GROUP REF IDX REACHED	Der im Funktionsbaustein KRC_ReadAxisGroup angegebene Index der Achsgruppe ist bereits belegt.	Den Funktionsbaustein KRC_ReadAxisGroup in einem Programm nur einfach instanziieren.
506	INVALID GROUPREFIDX	Der im Funktionsbaustein angegebene Index der Achsgruppe ist ungültig.	Einen gültigen Index für die Achsgruppe angeben (Parameter **AxisGroupIdx**).
507	INVALID FB ORDER	Die Reihenfolge, in der die Funktionsbausteine aufgerufen werden, ist ungültig.	Die Funktionsblöcke in der richtigen Reihenfolge programmieren.
508	CONNECTION NOT INITIALIZED	Es können keine Anweisungen übertragen werden, da die mxA-Schnittstelle nicht initialisiert wurde.	Die mxA-Schnittstelle initialisieren. (>>> mxA-Schnittstelle initialisieren [▶ 118])
509 510	NO CONNECTION TO KRC	Verbindung zur Robotersteuerung unterbrochen	Verbindung wiederherstellen, dann Fehler quittieren.

Nr.	Meldungstext	Ursache	Abhilfe
	TIMEOUT HEARTBEAT FROM KRC	Die Robotersteuerung ist ausgeschaltet.	Robotersteuerung neu starten.
		Submit-Interpreter abgewählt oder gestoppt	Submit-Interpreter neu starten.
		Busfehler oder E/A-Konfiguration fehlerhaft	E/A-Konfiguration überprüfen.
		Verbindungskabel defekt oder nicht richtig angeschlossen.	Verbindungskabel austauschen oder korrekt anschließen.
		Maximale Zykluszeit des Submit-Interpreters ist zu kurz (nur bei Meldung Nr. 510)	Den Wert für **MaxSubmitCycle** im Funktionsbaustein KRC_Diag erhöhen.
		Eine Funktion wurde vom Eingang **ExecuteCmd** aufgerufen, bevor die Schnittstelle initialisiert wurde.	Vor dem Aufruf einer Funktion mit **ExecuteCmd** auf den Ausgang **KRC_Initialize.Done** warten. Zum Zurücksetzen des Fehlers den Eingang **ExecuteCmd** auf den Wert FALSE setzen.
511	TIMEOUT CMD INTERFACE BLOCKED	Der Eingang **ExecuteCmd** wurde zurückgesetzt, bevor das Busy-Signal gesetzt wurde.	Die Meldung quittieren und zukünftig den Eingang **ExecuteCmd** erst zurücksetzen, nachdem das Done-, Error- oder Aborted-Signal gesetzt wurde.
512	INVALID CHECKSUM KRC -> PLC	Die Prüfsumme für die Datenübertragung von der Robotersteuerung zur SPS ist ungültig.	
		Fehler beim Hochlaufen: • EtherCAT-Konfiguration in WorkVisual oder TwinCAT fehlerhaft	Konfiguration in WorkVisual und TwinCAT überprüfen und EtherCAT korrekt konfigurieren.
		Fehler im Betrieb: • Bitfehler während der Datenübertragung	Service kontaktieren.
513	INVALID POSITION INDEX	Im Funktionsbaustein KRC_TouchUP wurde eine ungültige Nummer für die zu teachende Position übergeben.	Einen gültigen Wert programmieren (Parameter **Index**): • **1 … 100**
514	POS_ACT INVALID	Die aktuelle Position kann nicht geteacht werden, da die Positionsdaten ungültig sind (kein SAK).	SAK mit einem RESET beim Funktionsbaustein KRC_AutomaticExternal herstellen.
517	INVALID COMMAND SIZE	Interner Ausnahmefehler	Service kontaktieren.
518	KRC STOPMESS ACTIVE	Sammelfehler, der die Fahrfreigabe verhindert	Überprüfen, wie der Fehler ausgelöst wurde und den Fehler beheben. • Die Meldungen im Meldungsfenster der KUKA smartHMI analysieren. • Den aktuellen Fehlerzustand der Robotersteuerung mit dem Funktionsbaustein KRC_ReadKRCError auslesen.
519	INVALID ABSOLUTE VELOCITY	In einem Funktionsbaustein KRC_Move wurde ein ungültiger Wert für den Parameter **AbsoluteVelocity** programmiert.	Service kontaktieren.

Nr.	Meldungstext	Ursache	Abhilfe
520	VELOCITY CONFLICT	In einem Funktionsbaustein KRC_Move wurden mehrere Werte für die Geschwindigkeit programmiert.	Service kontaktieren.
521	INVALID PARAMETER COUNT	In einem Funktionsbaustein KRC_TechFunction wurde ein ungültiger Wert für den Parameter **ParameterCount** programmiert.	Service kontaktieren.
522	INVALID PARAMETER USAGE	Im Funktionsbaustein Parameter KRC_TechFunction wurde der Parameter **ParameterCount** falsch konfiguriert.	Service kontaktieren.
523	INVALID OPERATION MODE	Der Roboter befindet sich in der falschen Betriebsart.	Betriebsart Automatik Extern anwählen.
524	USER_SAF SIGNAL NOT ACTIVE	Der Bedienerschutz ist verletzt.	Schutzeinrichtung schließen und die Schließung quittieren.
525	ALARM_STOP SIGNAL NOT ACTIVE	Die Sicherheitskonfiguration ist fehlerhaft, dadurch wurde ein NOT-HALT ausgelöst.	Sicherheitskonfiguration des Systems (Robotersteuerung und SPS) prüfen und ändern.
		Keine Verbindung zum NOT-HALT der Anlage	NOT-HALT der Anlage prüfen und Verbindung wiederherstellen.
		Ein- und Ausgänge der Schnittstelle Automatik Extern sind falsch konfiguriert.	1. Im Hauptmenü Anzeige > Ein-/Ausgänge > Automatik Extern wählen. 2. Die Konfiguration der Ein- und Ausgänge prüfen und ändern.
526	APPL_RUN SIGNAL ACTIVE	RESET kann nicht durchgeführt werden, weil ein Roboterprogramm ausgeführt wird.	1. Warten, bis das Roboterprogramm abgearbeitet ist. 2. Die Anweisung erneut ausführen.
527	TIMEOUT MESSAGE CONFIRM	Die Meldung kann von der SPS nicht quittiert werden.	Die Meldung an der Robotersteuerung quittieren.
528	TIMEOUT MXA MESSAGE CONFIRM	Im Funktionsbaustein KRC_AutoStart kann ein Fehler nicht quittiert werden.	Service kontaktieren.
529	TIMEOUT SWITCHING DRIVES ON	Interner Ausnahmefehler	Service kontaktieren.
530	TIMEOUT PROGRAM SELECTION	Interner Ausnahmefehler	
531	TIMEOUT PROGRAM START	Interner Ausnahmefehler	
532	MOVE_ENABLE SIGNAL NOT ACTIVE	Roboter hat keine Fahrfreigabe	Fahrfreigabe mit dem Parameter **MOVE_ENABLE** erteilen.
533	INVALID AXIS_VALUES	Im Funktionsbaustein KRC_Forward sind nicht alle für die Ausführung erforderlichen Achswinkel definiert.	Die fehlenden Achswinkel im Funktionsbaustein KRC_Forward definieren.
534	INVALID $BASE	Interner Ausnahmefehler	Service kontaktieren.
535	INVALID $TOOL	Interner Ausnahmefehler	
536	INVALID SOFTEND	Fehler im Funktionsbaustein KRC_Forward: Die angegebenen Achswinkel liegen außerhalb der Softwareendschalter.	Achswinkel eingeben, die innerhalb der Softwareendschalter liegen (Parameter **Axis_Values**). oder: Die Softwareendschalter ändern.

Nr.	Meldungstext	Ursache	Abhilfe
537	ERR MATH TRAFO	Fehler im Funktionsbaustein KRC_Forward: Der Roboter kann die vorgegebenen Achswinkel nicht erreichen.	Achswinkel eingeben, die der Roboter erreichen kann (Parameter **Axis_Values**).
538	INVALID AXIS_VALUES	Fehler im Funktionsbaustein KRC_Inverse: • Die kartesische Roboterposition wurde nicht vollständig angegeben. • Die achsspezifischen Werte am Startpunkt der Bewegung wurden unvollständig angegeben.	• Die kartesische Roboterposition vollständig angeben (Parameter **Position**). • Die achsspezifischen Werte am Startpunkt der Bewegung vollständig angeben (Parameter **Start_Axis**).
539	INVALID $BASE	Interner Ausnahmefehler	Service kontaktieren.
540	INVALID $TOOL	Interner Ausnahmefehler	
541	INVALID SOFTEND	Fehler im Funktionsbaustein KRC_Inverse: Die achsspezifischen Werte am Startpunkt der Bewegung liegen außerhalb der Softwareendschalter.	Werte eingeben, die innerhalb der Softwareendschalter liegen (Parameter **Start_Axis**). Oder: Die Softwareendschalter ändern.
542	ERR MATH TRAFO	Fehler im Funktionsbaustein KRC_Inverse: Der Roboter kann die vorgegebenen achsspezifischen Werte am Startpunkt der Bewegung nicht erreichen.	Werte eingeben, die der Roboter erreichen kann (Parameter **Start_Axis**).
543	INVALID EXECUTE	Bei einer überschleiffähigen verketteten Bewegung wurde der Execute-Eingang zurückgesetzt, bevor das ComAcpt-Signal vom Funktionsbaustein gesetzt wurde.	Die Meldung quittieren und zukünftig den Execute-Eingang erst zurücksetzen, wenn das ComAcpt-Signal gesetzt wurde.
544	INVALID DEV_VEL_CP	Die Initialisierung der mxA-Schnittstelle auf der Robotersteuerung ist noch nicht abgeschlossen oder fehlerhaft.	Überprüfen, ob der Ausgang **Done** im Funktionsbaustein KRC_Initialize aktiv ist.
547	INVALID TURN	Fehler im Funktionsbaustein KRC_Inverse: Mit dem vorgegebenen Turn liegen die achsspezifischen Werte an der vorgegebenen Position außerhalb der Softwareendschalter.	Werte eingeben, die innerhalb der Softwareendschalter liegen (Parameter **Position**). Oder: Die Softwareendschalter ändern.
550	Check of LDD Installation (LDD in EXT) failed (Err=-1)!	Fehler im Funktionsbaustein KRC_LDDconfig: Die Prüfung der Installation und Aktivierung des Funktionsbausteins ist fehlgeschlagen.	Weitere Informationen zur Fehlerquelle sind im Windows-Event-Log zu finden.
551	Check of LDD Installation (LDD in EXT) failed (Err=-2)!	Fehler im Funktionsbaustein KRC_LDDconfig: Die Initialisierung der internen Datenschnittstelle ist nicht korrekt.	
552	Check of LDD Installation (LDD in EXT) failed (Err=-3)!	Fehler im Funktionsbaustein KRC_LDDconfig: Die Konfiguration der Benutzerdaten ist nicht gültig.	Die Fehlermeldungen beachten.

Nr.	Meldungstext	Ursache	Abhilfe
553	Default settings not correctly finished (Err=-40)!	Interner Fehler	Weitere Informationen zur Fehlerquelle sind im Windows-Event-Log zu finden.
554	Default settings not correctly finished (Err=-41)!	Interner Fehler	
555	Default settings not correctly finished (Err=-42)!	Interner Fehler	
556	A4 < 0.0 … Move A4 to 0.0	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 4 ist kleiner als 0°.	Achse 4 in positiver Richtung auf 0° bewegen.
557	A4 > 0.0 … Move A4 to 0.0	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 4 ist größer als 0°.	Achse 4 in negativer Richtung auf 0° bewegen.
558	Axis ranges: A2 / A3 not valid!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Die Stellung der Achsen 2 und 3 ist nicht gültig.	Achse 2 und 3 so positionieren, dass der Roboterarm waagerecht steht.
559	Axis ranges: A3 range not valid!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 3: Achsbereich ist nicht gültig.	Den Achsbereich verkleinern oder vergrößern. Ideale Werte: • Achsbereich: 4° • Startposition: ±2°
560	Axis position: A4 position not valid!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 4: Achsbereich ist nicht gültig.	Achse 4 auf 0° bewegen.
561	Axis ranges: A5 range not valid!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 5: Achsbereich ist nicht gültig.	Den Achsbereich verkleinern oder vergrößern. Ideale Werte: • Achsbereich: 80° • Startposition: ±40°
562	Axis ranges: A6 range not valid!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 6: Achsbereich ist nicht gültig.	Den Achsbereich verkleinern oder vergrößern. Ideale Werte: • Achsbereich: 120° • Startposition: ±60°
563	Axis ranges: A1 too close to the lower limit!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 1 verletzt den zulässigen unteren Achsbereich.	Achse 1 in positiver Bewegungsrichtung verfahren.
564	Axis ranges: A2 too close to the lower limit!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 2 verletzt den zulässigen unteren Achsbereich.	Achse 2 in positiver Bewegungsrichtung verfahren.

Nr.	Meldungstext	Ursache	Abhilfe
565	Axis ranges: A3 too close to the lower limit!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 3 verletzt den zulässigen unteren Achsbereich.	Achse 3 in positiver Bewegungsrichtung verfahren.
566	Axis ranges: A5 too close to the lower limit!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 5 verletzt den zulässigen unteren Achsbereich.	Achse 5 in positiver Bewegungsrichtung verfahren.
567	Axis ranges: A6 too close to the lower limit!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 6 verletzt den zulässigen unteren Achsbereich.	Achse 6 in positiver Bewegungsrichtung verfahren.
568	Axis ranges: A1 too close to the upper limit!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 1 verletzt den zulässigen oberen Achsbereich.	Achse 1 in negativer Bewegungsrichtung verfahren.
569	Axis ranges: A2 too close to the upper limit!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 2 verletzt den zulässigen oberen Achsbereich.	Achse 2 in negativer Bewegungsrichtung verfahren.
570	Axis ranges: A3 too close to the upper limit!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 3 verletzt den zulässigen oberen Achsbereich.	Achse 3 in negativer Bewegungsrichtung verfahren.
571	Axis ranges: A5 too close to the upper limit!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 5 verletzt den zulässigen oberen Achsbereich.	Achse 5 in negativer Bewegungsrichtung verfahren.
572	Axis ranges: A6 range not valid!	Fehler im Funktionsbaustein KRC_LDDconfig oder KRC_LDDcheckPos: Achse 6 verletzt den zulässigen oberen Achsbereich.	Achse 6 in negativer Bewegungsrichtung verfahren.
574	LDD configuration not valid!	Fehler im Funktionsbaustein KRC_LDDtestRun: Die Lastdatenermittlung wurde nicht korrekt konfiguriert.	Weitere Informationen zur Fehlerquelle sind im Windows-Event-Log zu finden.
575	LDD KRL Program not correctly finished	Fehler im Funktionsbaustein KRC_LDDstart: Die Lastdatenermittlung wurde nicht korrekt beendet.	Weitere Informationen zur Fehlerquelle sind im Windows-Event-Log zu finden.
576	Kuka.Load 5 result unknown	Fehler im Funktionsbaustein KRC_LDDstart: Die Belastungsanalyse hat kein Ergebnis geliefert.	Weitere Informationen zur Fehlerquelle sind im Logbuch von KUKA.Load zu finden.
577	LDD_LOAD_RESULT = #STATOVL	Fehler im Funktionsbaustein KRC_LDDstart:	Service kontaktieren.

Nr.	Meldungstext	Ursache	Abhilfe
		Die Belastungsanalyse hat ergeben, dass eine statische Überlastung vorliegt. Der ausgewählte Robotertyp ist für den angegebenen Lastfall nicht zulässig.	
578	LDD_LOAD_RESULT = #DYNOVL	Fehler im Funktionsbaustein KRC_LDDstart: Die Belastungsanalyse hat ergeben, dass eine dynamische Überlastung vorliegt. Der ausgewählte Robotertyp ist für den angegebenen Lastfall nicht zulässig.	
579	LDD_LOAD_RESULT = #OVL	Fehler im Funktionsbaustein KRC_LDDstart: Die Belastungsanalyse hat ergeben, dass eine statische und dynamische Überlastung vorliegt. Der ausgewählte Robotertyp ist für den angegebenen Lastfall nicht zulässig.	
580	LDD_LOAD_RESULT = #SLOOR	Fehler im Funktionsbaustein KRC_LDDstart: Die Belastungsanalyse hat ergeben, dass die Zusatzlast außerhalb des spezifizierten Bereichs liegt. Der ausgewählte Robotertyp ist für den angegebenen Lastfall nicht zulässig.	
581	Tool Number=-1: $config.dat unchanged!	Fehler im Funktionsbaustein KRC_LDDstart oder KRC_LDDwriteLoad: Parameter **Tool** = -1 Die ermittelten Lastdaten werden keinem Werkzeug zugewiesen.	Um die Lastdaten einem Werkzeug zuzuweisen, den Parameter **Tool** auf einen gültigen Wert setzen und den Funktionsbaustein KRC_LDDwriteLoad aufrufen. • **1 … maximale Anzahl der Werkzeuge**
582	LDD_start not correctly finished!	Fehler im Funktionsbaustein KRC_LDDstart oder KRC_LDDwriteLoad: Die Lastdatenermittlung wurde nicht korrekt beendet.	Die Fehlermeldungen beachten.
583	Kuka.Load 5 not correctly finished!	Fehler im Funktionsbaustein KRC_LDDstart oder KRC_LDDwriteLoad: Die Auswertung der Lastdaten durch KUKA.Load wurde nicht korrekt beendet.	Die Fehlermeldungen beachten. Ggf. weitere Informationen dem Windows-Event-Log oder dem Logbuch von KUKA.Load entnehmen.
584	Tool number not correct!	Fehler im Funktionsbaustein KRC_LDDstart oder KRC_LDDwriteLoad: Die Werkzeugnummer ist nicht gültig.	Werkzeugnummer (Parameter **Tool**) auf einen gültigen Wert setzen: • **1 … maximale Anzahl der Werkzeuge**
585	LDD KRL Test run Program not correctly finished!	Fehler im Funktionsbaustein KRC_LDDtestRun: Die Testfahrt wurde nicht korrekt beendet.	Das Programm für die Testfahrt komplett durchfahren.
586	program override check, should be less or equal 10	Fehler im Funktionsbaustein KRC_LDDtestRun:	1. Den Programm-Override auf ≤ 10 % setzen (Funktionsbaustein KRC_SetOverride).

Nr.	Meldungstext	Ursache	Abhilfe
		Funktionsbaustein wurde mit einem Programm-Override ≥ 10 % ausgeführt.	2. Funktionsbaustein KRC_LDDtestRun erneut ausführen.
587	Internal error	Fehler im Funktionsbaustein KRC_LDDstart oder KRC_LDDwriteLoad: Interner Fehler	Weitere Informationen zur Fehlerquelle sind im Windows-Event-Log zu finden.
588	Internal error	Fehler im Funktionsbaustein KRC_LDDstart oder KRC_LDDwriteLoad: Interner Fehler	Weitere Informationen zur Fehlerquelle sind im Windows-Event-Log zu finden.

8.4 ProConOS-Fehler

Nr.	Meldungstext	Ursache	Abhilfe
701	INTERNAL ERROR	Interner Ausnahmefehler	Service kontaktieren.
702	ASSERT FAILED	Interner Ausnahmefehler	
703	INVALID COMMAND ID	Interner Ausnahmefehler	
704	INVALID HEADER DATA	Interner Ausnahmefehler	
709	ERROR READING SOFTPLC	Interner Ausnahmefehler	
710	ERROR FROM KRC SUBMIT	Interner Ausnahmefehler	
712	INVALID CHECKSUM PLC -> KRC	Die Prüfsumme für die Datenübertragung von der SPS zur Robotersteuerung ist ungültig.	Service kontaktieren.
713	INVALID MOVE TYPE	In einem KRC_Move-Funktionsbaustein wurde ein ungültiger Wert für den Parameter MoveType programmiert.	Service kontaktieren.
730	INVALID PTP APO	Für eine PTP-Bewegung wurde ein ungültiger Überschleifparameter übergeben.	Einen gültigen Wert programmieren (Parameter **Approximate**). (>>> APO [▶ 19])
731	INVALID CP APO	Für eine CP-Bewegung (LIN, CIRC) wurde ein ungültiger Überschleifparameter übergeben.	
732	INVALID BASE NUMBER	Im Funktionsbaustein KRC_ReadBaseData oder KRC_WriteBaseData wurde für das BASE-Koordinatensystem eine ungültige Nummer programmiert.	Die Nummer des BASE-Koordinatensystems angeben, die derzeit in der Robotersteuerung verwendet wird (Parameter **BaseNo**). • **1 … 32**
		In einem KRC_Move- oder KRC_Jog-Funktionsbaustein wurde für das BASE-Koordinatensystem eine ungültige Nummer programmiert.	Die Nummer des BASE-Koordinatensystems angeben, die derzeit in der Robotersteuerung verwendet wird (Parameter **CoordinateSystem** – **COORDSYS.Base**). (>>> COORDSYS [▶ 20])

Nr.	Meldungstext	Ursache	Abhilfe
733	INVALID TOOL NUMBER	Im Funktionsbaustein KRC_ReadToolData oder KRC_WriteToolData wurde für das TOOL-Koordinatensystem eine ungültige Nummer programmiert.	Die Nummer des TOOL-Koordinatensystems angeben, die derzeit in der Robotersteuerung verwendet wird (Parameter **ToolNo**). • **1 … 16**
		In einem KRC_Move- oder KRC_Jog-Funktionsbaustein wurde für das TOOL-Koordinatensystem eine ungültige Nummer programmiert.	Die Nummer des TOOL-Koordinatensystems angeben, die derzeit in der Robotersteuerung verwendet wird (Parameter **CoordinateSystem** – **COORDSYS.Tool**). (>>> COORDSYS [▶ 20])
734	INVALID VELOCITY	In einem Funktionsbaustein wurde für die Geschwindigkeit ein ungültiger Wert programmiert.	Einen gültigen Wert programmieren (Parameter **Velocity**): • **0 … 100%** • **0 … 2 m/s** (nur bei den Bausteinen MC_MoveLinear und MC_MoveCircular)
735	INVALID ACCELERATION	In einem Funktionsbaustein wurde für die Beschleunigung ein ungültiger Wert programmiert.	Einen gültigen Wert programmieren (Parameter **Acceleration**): • **0 … 100%** • 0 … 2,3 m/s^2 (nur bei den Bausteinen MC_MoveLinear und MC_MoveCircular)
736	INVALID C_PTP	Für eine PTP-Bewegung wurde ein ungültiger Überschleifabstand übergeben.	Einen gültigen Wert programmieren (Parameter **Approximate**). (>>> APO [▶ 19])
737	INVALID C_DIS	Für eine Überschleifbewegung wurde ein ungültiger Abstandsparameter übergeben.	
738	INVALID C_VEL	Für eine Überschleifbewegung wurde ein ungültiger Geschwindigkeitsparameter übergeben.	
739	INVALID C_ORI	Für eine Überschleifbewegung wurde ein ungültiger Ausrichtungsparameter übergeben.	
740	INVALID ORI_TYPE	In einem KRC_Move- oder KRC_Jog-Funktionsbaustein wurde für die Ausrichtungssteuerung des TCP ein ungültiger Wert programmiert.	Einen gültigen Wert programmieren (Parameter **OriType**). (>>> OriType [▶ 16])
741	POSITION DATA NOT INITIALIZED	Beim Aufruf eines KRC_Move-Funktionsbausteins wurde keine Zielposition übergeben.	Mindestens 1 Element der Zielposition festlegen (Parameter **Position**). (>>> E6POS [▶ 20])
742	AXISPOSITION DATA NOT INITIALIZED	Beim Aufruf eines KRC_MoveAxis-Funktionsbausteins wurde keine Achsposition übergeben.	Mindestens 1 Achsposition festlegen (Parameter **AxisPosition**). (>>> E6AXIS [▶ 20])
743	INVALID TRIGGER DISTANCE	In einem KRC_SetDistanceTrigger-Funktionsbaustein wurde für den Schaltpunkt des Schalters ein ungültiger Wert programmiert.	Einen gültigen Wert programmieren (Parameter **Distance**): • **0**: Schaltaktion am Startpunkt

Nr.	Meldungstext	Ursache	Abhilfe
			• **1**: Schaltaktion am Zielpunkt
744	INVALID TRIGGER IO	In einem KRC_SetDistanceTrigger- oder KRC_SetPathTrigger-Funktionsbaustein wurde ein ungültiger Ausgang programmiert.	Einen gültigen Wert programmieren (Parameter **Output**): • **1 … 2048**
745	INVALID TRIGGER PULSE	In einem KRC_SetDistanceTrigger- oder KRC_SetPathTrigger-Funktionsbaustein für die Impulslänge wurde ein ungültiger Wert programmiert.	Einen gültigen Wert programmieren (Parameter **Pulse**): • **0.1 … 3.0 s** • **0.0 s** (kein Impuls aktiv)
746	INVALID CIRC_HP	Beim Aufruf eines KRC_MoveCirc-Funktionsbausteins wurde keine Hilfsposition übergeben.	Mindestens 1 Element der Hilfsposition festlegen (Parameter **CircHP**). (>>> E6AXIS [▶ 20])
747	INVALID INTERRUPT IO	Die Nummer des digitalen Eingangs, für den der Interrupt deklariert wird, ist ungültig (Funktionsbaustein KRC_DeclareInterrupt).	Einen gültigen Wert programmieren (Parameter **Input**): • **1 … 2048**
748	INVALID INTERRUPT PRIORITY	Beim Aufruf eines KRC_…Interrupt-Funktionsbausteins wurde eine ungültige Nummer übergeben.	Einen gültigen Wert programmieren (Parameter **Interrupt**): • **1 … 8**
750	INVALID INTERRUPT ACTION	Die bei der Deklaration des Interrupts programmierte Interrupt-Reaktion ist ungültig.	Eine gültige Reaktion programmieren (Parameter **Reaction**). (>>> Interrupt deklarieren [▶ 67])
751	INVALID IO NUMBER	Die Nummer des digitalen Eingangs, für den der Interrupt deklariert wird, ist ungültig (Funktionsbaustein KRC_DeclareInterrupt).	Einen gültigen Wert programmieren (Parameter **Input**): • **1 … 2048**
752	INVALID PULSE DURATION	Im Funktionsbaustein KRC_WriteDigitalOutput für die Impulslänge wurde ein ungültiger Wert programmiert.	Einen gültigen Wert programmieren (Parameter **Pulse**): • **0.1 … 3.0 s** • **0.0 s** (kein Impuls aktiv)
753	INVALID BUFFER_MODE	In einem Funktionsbaustein wurde ein ungültiger **BufferMode** programmiert, z. B, ist die Betriebsart DIRECT bei bestimmten Funktionsbausteinen nicht verfügbar.	Einen gültigen **BufferMode** programmieren. (>>> BufferMode [▶ 24])
754	INVALID TOOL NUMBER FOR LOAD_DATA	Im Funktionsbaustein KRC_ReadLoadData oder KRC_WriteLoadData für das Lesen bzw. Schreiben der Lastdaten oder ergänzenden Lastdaten wurde eine ungültige Nummer programmiert.	Einen gültigen Wert programmieren (Parameter **Tool**). (>>> Lastdaten lesen [▶ 139]) (>>> Lastdaten schreiben [▶ 140])
755	INVALID ANALOG IO NUMBER	In einem Funktionsbaustein für den analogen Eingang oder Ausgang wurde ein ungültiger Wert programmiert.	Einen gültigen Wert programmieren (Parameter **Number**): • **1 … 32**
756	INVALID IPO_MODE	In einem Funktionsbaustein für die Interpolationsart, z. B. KRC_Move, wurde ein ungültiger Wert programmiert.	Einen gültigen Wert programmieren (Parameter **CoordinateSystem** – **COORDSYS.IPO_MODE**). (>>> COORDSYS [▶ 20])

Nr.	Meldungstext	Ursache	Abhilfe
757	INVALID CIRC_TYPE	In einem KRC_MoveCirc-Funktionsbaustein wurde für die Ausrichtungssteuerung während der Kreisbewegung ein ungültiger Wert programmiert.	Einen gültigen Wert programmieren (Parameter **CircType**). (>>> CircType [▶ 16])
758	INVALID FRAME DATA	In einem KRC_WriteToolData- oder KRC_WriteBaseData-Funktionsbaustein wurden ungültige TOOL- oder BASE-Daten programmiert.	Gültige Daten programmieren (Parameter **ToolData** oder **BaseData**). (>>> TOOL-Daten schreiben [▶ 136]) (>>> BASE-Daten schreiben [▶ 138])
759	INVALID LOAD DATA	In einem KRC_WriteLoadData-Funktionsbaustein wurden ungültige Lastdaten programmiert.	Gültige Daten programmieren. (>>> Lastdaten schreiben [▶ 140])
760	INVALID SOFT_END (REVERSED)	Fehler beim Schreiben der Software-Endschalter: positiver Software-Endschalter < negativer Software-Endschalter (Funktionsbaustein KRC_WriteSoftEnd oder KRC_WriteSoftEndEx)	Für den negativen Software-Endschalter niedrigere Werte programmieren als für den positiven Software-Endschalter.
765	INVALID REAL VALUE	Der programmierte Real-Wert ist ungültig.	• **-2.147.483.500 … +2.147.483.500**
770	INVALID PARAMETER VALUE	Im aufgerufenen Funktionsbaustein wurde ein ungültiger Wert für einen oder mehrere Parameter programmiert.	Gültige Werte für die Parameter programmieren.
771	INVALID ADVANCE COUNT	Im Funktionsbaustein KRC_SetAdvance wurde ein ungültiger Wert für die Anzahl der Funktionen programmiert, die vor der ersten Roboterbewegung übertragen werden sollen.	Einen gültigen Wert programmieren (Parameter **Count**): • **1 … 50**
772	INVALID MAXWAITTIME	Im Funktionsbaustein KRC_SetAdvance wurde ein ungültiger Wert für die maximale Wartezeit vor dem Start der Programmausführung für den Fall programmiert, dass die vorgegebene Anzahl der Funktionen im Parameter **Count** nicht erreicht wird.	Einen gültigen Wert programmieren (Parameter **MaxWaitTime**): • **1 … 32 767 ms**
773	INVALID ADVANCE MODE	Im Funktionsbaustein KRC_SetAdvance wurde ein ungültiger Wert für den Wartemodus programmiert.	Einen gültigen Wert programmieren (Parameter **Mode**): • **0 … 2**
774	INVALID DI STARTNUMBER	Im Funktionsbaustein KRC_ReadDigitalInputArray wurde ein ungültiger Wert für die Nummer des ersten digitalen Eingangs programmiert, der aufgerufen wird.	Einen gültigen Wert programmieren (Parameter **Startnumber**): • **1 … 2048**
775	INVALID DI LENGTH	Im Funktionsbaustein KRC_ReadDigitalInputArray wurde ein ungültiger Wert für die Anzahl der Eingänge programmiert, die abgefragt werden.	Einen gültigen Wert programmieren (Parameter **Length**): • **1 … 200**

Nr.	Meldungstext	Ursache	Abhilfe
776	INVALID CONVEYOR NUMBER	Im aufgerufenen Funktionsbaustein wurde ein ungültiger Wert für die Nummer des Förderers programmiert.	Einen gültigen Wert programmieren (Parameter **ConveyorNumber**): • **1 … 3**
777	INVALID CONVEYOR STARTDISTANCE	Im Funktionsbaustein KRC_ConvFollow oder KRC_ConvSkip wurde ein ungültiger Wert für die Strecke programmiert, die das Werkstück während der Wartezeit des Roboters vor dem Start der Verfolgung des Werkstücks auf dem Förderer zurücklegt.	Einen gültigen Wert programmieren (Parameter **StartDistance**): • Bei einem Linearförderer: Angabe in Millimetern • Bei einem Kreisförderer: Angabe in Grad
778	INVALID CONVEYOR MAXDISTANCE	Im Funktionsbaustein KRC_ConvFollow oder KRC_ConvSkip wurde ein ungültiger Wert für die maximale Strecke programmiert, die das Werkstück zurücklegt, bevor der Roboter beginnt, mit dem Werkstück zu synchronisieren.	Einen gültigen Wert programmieren (Parameter **MaxDistance**): • Bei einem Linearförderer: Angabe in Millimetern • Bei einem Kreisförderer: Angabe in Grad
779	INVALID CONVEYOR PIECENUMBER	Im Funktionsbaustein KRC_ConvSkip wurde ein ungültiger Wert für die Zahl programmiert, die angibt, welche Werkstücke aufgenommen werden sollen.	Einen gültigen Wert programmieren (Parameter **PieceNumber**). Beispiele: • **1**: Jedes Werkstück wird aufgenommen. • **3**: Jedes dritte Werkstück wird aufgenommen.
780	INVALID WORKSPACENO	Im aufgerufenen Funktionsbaustein wurde ein ungültiger Wert für die Nummer des Arbeitsbereichs programmiert.	Einen gültigen Wert programmieren (Parameter **WorkspaceNo**): • **1 … 8**
781	INVALID WORKSPACEMODE	Im aufgerufenen Funktionsbaustein wurde ein ungültiger Wert für die Betriebsart für Arbeitsbereiche programmiert.	Einen gültigen Wert programmieren (Parameter **WorkspaceMode**): • **0 … 4**
782	INVALID WORKSPACEPART	Interner Ausnahmefehler	Service kontaktieren.
783	UDP CONNECTION TIMEOUT	Zeitüberschreitung der Verbindung bei UDP-Kommunikation	Die IP-Adressen der beiden Anschlussstellen und das Kabel dazwischen überprüfen. In der Datei mxa_config.dat einen längeren Wert für die Zeitüberschreitung festlegen.
801	STOPMESS ACTIVE	Gruppenfehler, der die Bewegungsfreigabe verhindert	Überprüfen, wie der Fehler ausgelöst wurde, und den Fehler beheben. • Die Meldungen im Meldungsfenster der KUKA smartHMI analysieren. • Den aktuellen Fehlerzustand der Robotersteuerung mit dem Funktionsbaustein KRC_ReadKRCError auslesen.

Mehr Informationen:
www.beckhoff.de/tf5120

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.com
www.beckhoff.com