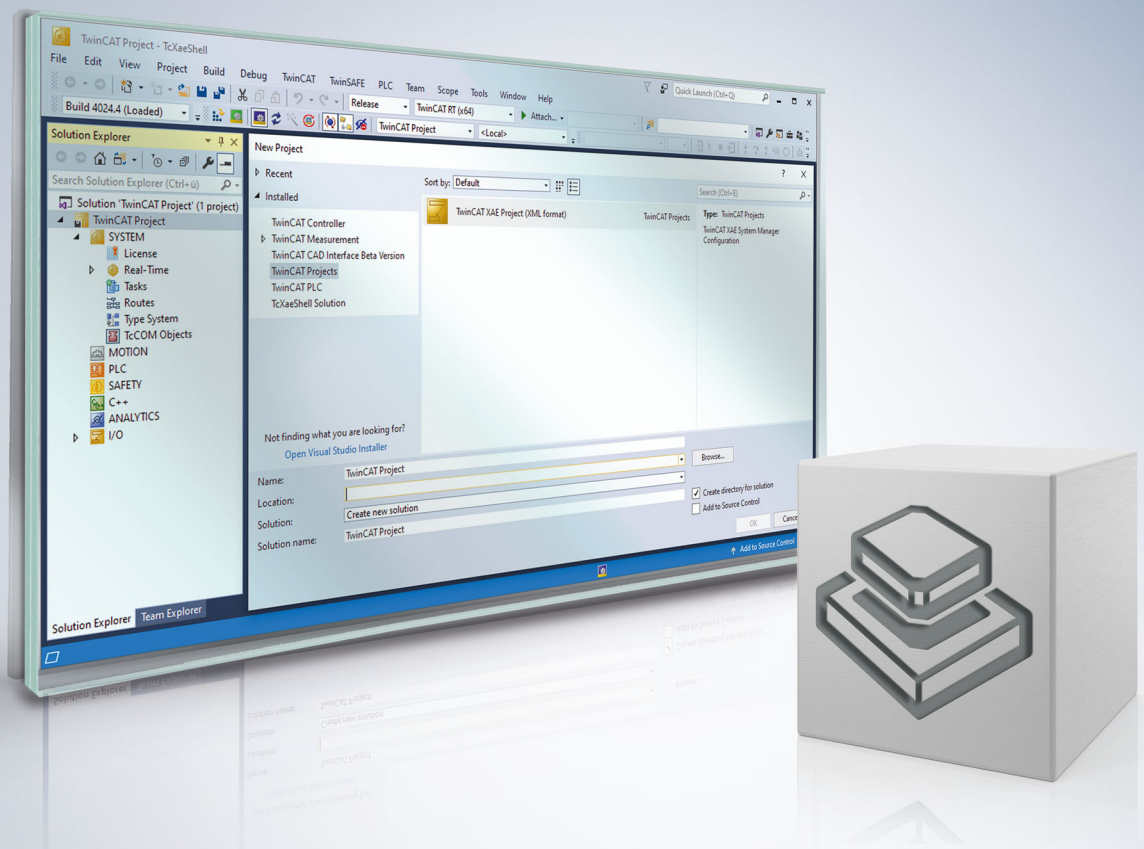
BECKHOFF New Automation Technology

Handbuch | DE

TF5430

TwinCAT 3 | Planar Motion



11.03.2024 | Version: 1.4.1

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, stets die aktuell gültige 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 weiterentwickelt.
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® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

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.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

⚠ GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

⚠ WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

⚠ VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt

Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

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.

Inhaltsverzeichnis

2 Übersicht der neuen Funktionen

Ab Version V3.2.60:

- Neu: Die Fläche, auf der Mover fahren, ist in Planar-Parts aufgeteilt, die zur Laufzeit bewegt werden können. Mover können Grenzen zwischen verbundenen bzw. angrenzenden Planar-Parts überqueren.
- Erfordert TwinCAT V3.1.4024.40 oder höher

Ab Version V3.1.10.63:

- Erfordert TwinCAT V3.1.4024.24 oder höher

Ab Version V3.1.10.51:

- Neu: AdoptTrackOrientation dreht den Mover auf dem Track zu der Track Orientierung. Dadurch ändert sich der C-Coordinate Modus von independent zu dependent.
- Erweitert: MoveC funktioniert ab jetzt immer für stehende Mover auf dem Track. Dadurch ändert sich ggf. der C-Coordinate Modus von dependent zu independent.

Ab Version V3.1.10.44:

- Neu: GearInPosOnTrack- und GearInPosOnTrackWithMasterMover-Befehle zur Kopplung eines Planar-Movers an eine Master-Achse bzw. einen Master-Mover
- Erweitert: Parameter des Planar-Track TcCOM-Moduls
- Erfordert TwinCAT V3.1.4024.17 oder höher

Ab Version V3.1.10.30:

- Neu: CRotation-Befehlsmodus (360° Drehung) mit Modulo-Positionierung
- Neu: Constraints als neue Variante, Dynamiken der Bewegungskommandos zu beschränken
- Erweitert: Parameter für Modulo-Positionierung auf „Closed Loop“-Tracks

Ab Version V3.1.10.11:

- Erste Version von Planar Motion veröffentlicht
- Erfordert TwinCAT V3.1.4024.12 oder höher

3 Einführung

Das TwinCAT 3 Planar Motion-Softwarepaket TF5430 wird zusammen mit dem Softwarepaket TF5400 installiert.

Zielsystem

Windows 7/8/10 (nur 64Bit)

TwinCAT 3 Planar Motion Base

Die Software TF5430 TwinCAT 3 Planar Motion kombiniert eine Vielzahl an Funktionalitäten zur Steuerung der XPlanar-Mover und ermöglicht eine effiziente und intelligente Umsetzung von individuellen XPlanar-Anwendungen. TF5430 TwinCAT 3 Planar Motion ist Bestandteil von TF5890 TwinCAT 3 XPlanar. Alle zugehörigen Funktionsbausteine sind in der Bibliothek Tc3_Mc3PlanarMotion enthalten, welche in Kombination mit der Bibliothek Tc3_Physics zu verwenden ist.

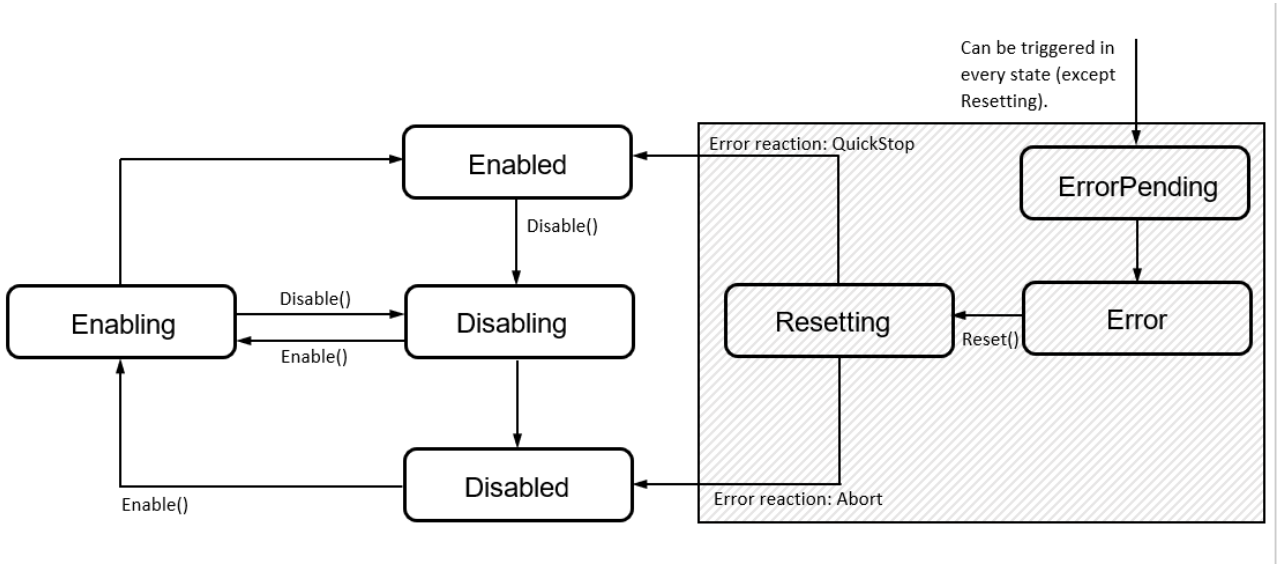
Zusätzliche Lizenzanforderungen

TF5430 TwinCAT 3 Planar Motion benötigt die Lizenz TC1250.

4 Zustände und Modi

4.1 Zustandsdiagramm Planar-Objekte

Die Planar State Machine wird vom Planar-Mover, vom Planar-Track und von der Planar-Group genutzt. Alle diese Komponenten können sich in den *sieben* Planar-Zuständen befinden: Enabling, Enabled, Disabling, Disabled, Resetting, ErrorPending, Error.



Die Fehlerreaktion hängt davon ab, wie schwerwiegend der Fehler ist. Für leichte Fehler ist die normale Fehlerreaktion ein QuickStop. Der Anwender kann durch Disable() auch in diesem Fall die Fehlerreaktion Abort erzwingen. Der Befehl muss in einem der drei Zustände abgesendet werden: ErrorPending, Error oder Resetting.

Enabling

Im Zustand Enabling wird der Enable-Befehl ausgeführt. Am Ende dieses Befehls befindet sich die Komponente im Zustand Enabled. Im Zustand Enabling kann ein Disable-Befehl gesendet werden, welcher den Enable-Befehl abbricht und dafür sorgt, dass der Zustand in Disabling wechselt.

Enabled

Im Zustand Enabled ist die Komponente voll funktionsfähig und kann vom Nutzer verwendet werden. In diesem Zustand kann ein Disable-Befehl gesendet werden. Der Zustand wechselt dann zu Disabling.

Disabling

Im Zustand Disabling wird der Disable-Befehl ausgeführt. Am Ende dieses Befehls befindet sich die Komponente im Zustand Disabled. Im Zustand Disabling kann ein Enable-Befehl gesendet werden, welcher den Disable-Befehl abbricht und dafür sorgt, dass der Zustand in Enabling wechselt.

Disabled

Nach dem Hochfahren des Systems befinden sich die Komponenten im Zustand Disabled. Sie können über einen Enable-Befehl in den Zustand Enabling versetzt werden. Im Zustand Disabled sind die Komponenten nicht funktionsfähig.

Resetting

Die Komponente ist dabei, den Fehler zu beheben. Abhängig von der Fehlerreaktion ist sie danach im Zustand Enabled oder Disabled.

ErrorPending

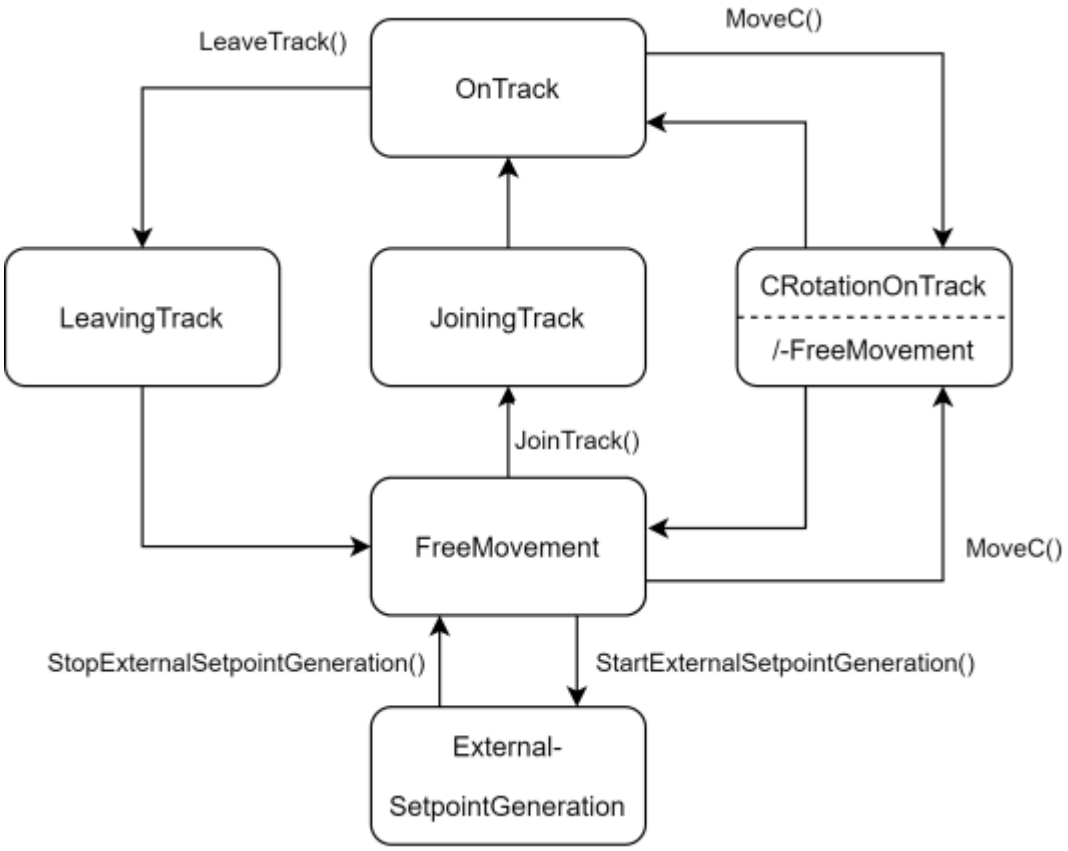
Den Zustand ErrorPending erreicht die Komponente aus allen anderen Zuständen, außer dem Zustand Resetting, wenn ein Fehler auftritt. Sobald der Fehler korrekt verarbeitet wurde, wechselt der Zustand zu Error.

Error

Der Zustand Error bedeutet, dass ein Fehler aufgetreten ist und die Komponente nun über den Reset-Befehl in den Zustand Resetting überführt werden kann, damit der Fehler behoben wird.

4.2 Befehlsdiagramm Planar-Mover

Der Planar-Mover hat sechs verschiedene Befehlsmodi, die anzeigen, welche Art von Befehl der Mover ausführt: OnTrack, LeavingTrack, JoiningTrack, ExternalSetpointGeneration und CRotationFreeMovement/-OnTrack (ab Version V3.1.10.51). In allen Modi, bis auf den Modus ExternalSetpointGeneration, ist die Kollisionsvermeidung für den Mover aktiv, wenn er in einer Gruppe ist.



OnTrack

Im Modus OnTrack ist der Mover auf einem Track eingekoppelt und kann auf diesem verfahren werden (MoveOnTrack). Der Mover kann den Track auch wieder verlassen (LeaveTrack), dieses ändert den Modus zu LeavingTrack. MoveC-Befehle bewirken ggf. einen Wechsel in den Modus CRotationOnTrack.

LeavingTrack

Im Modus LeavingTrack nimmt der Mover keine weiteren Befehle an. Der Modus wird automatisch verlassen, wenn der Mover den LeaveTrack-Befehl beendet hat. Dann befindet sich der Mover im Modus FreeMovement.

JoiningTrack

Im Modus JoiningTrack nimmt der Mover keine weiteren Befehle an. Der Modus wird automatisch verlassen, wenn der Mover den JoinTrack-Befehl beendet hat. Dann befindet sich der Mover im Modus OnTrack.

FreeMovement

Nach dem Enablen des Movers ist er automatisch in diesem Befehlsmodus, es sei denn, der Mover ist zu stark verdreht. Dann befindet er sich im Modus CRotationFreeMovement. Der Mover kann mit MoveToPosition-Befehlen frei verfahren werden. Startet der Nutzer per Befehl die externe Sollwertgenerierung, so wechselt der Modus zu ExternalSetpointGeneration. Auch JoinTrack-Befehle sind möglich, die den Modus in JoiningTrack ändern. MoveC Befehle bewirgen ggf. einen wechseln in den Modus CRotationFreeMovement.

CRotationFreeMovement/-OnTrack

Dieser Modus wird durch den Befehl MoveC gestartet, wenn die gestartete C-Bewegung nicht komplett innerhalb eines C-Positionsfensters liegt. Die Fenster werden durch die Positionslimits der C-Achse des Movers definiert und existieren 4 mal jeweils um 90° gedreht (die 90° Drehung ergibt sich aufgrund der Moversymmetrie: z. B. limits +-15° -> Fenster 1. [-15°,+15°], 2. [75°,105°], 3. [165°,195°], 4. [255°,285°]). Abhängig davon, ob man zuvor im Modus FreeMovement oder OnTrack war, ist der Modus entsprechend CRotationFreeMovement oder CRotationOnTrack. Der Modus ist beendet, wenn die C-Bewegung abgeschlossen ist und die Endposition innerhalb eines der vier Fenster liegt. Der Mover wechselt dann automatisch in den vorherigen Modus zurück. Der Mover wechselt also von CRotationFreeMovement zu *FreeMovement* bzw. von CRotationOnTrack zu *OnTrack*. Ansonsten verbleibt er im Modus CRotationFreeMovement/-OnTrack. In beiden Modi CRotation können die X- und die Y-Achse des Movers nicht bewegt werden. Hat der Mover beim Aufstarten bereits eine Orientierung außerhalb der 4 Fenster, so befindet er sich anstatt in FreeMovement sofort im Modus CRotationFreeMovement.

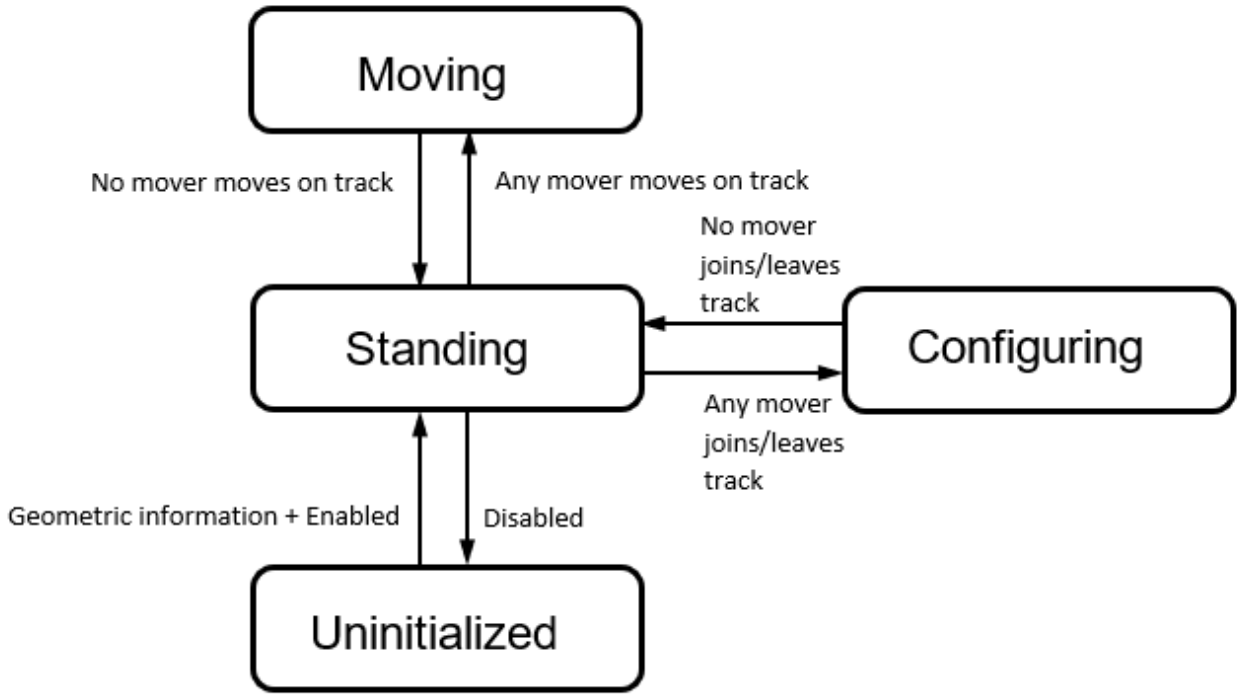
ExternalSetpointGeneration

Im Modus ExternalSetpointGeneration führt der Mover einen entsprechenden Befehl aus. Dieser Modus beginnt (bzw. endet) mit dem Beginn (bzw. Ende) des entsprechenden Befehls. Der Mover folgt im Modus ExternalSetpointGeneration den externen Sollwerten, die der Nutzer zyklisch bereitstellt.

Die Externe Sollwertgenerierung kann auch zusammen mit den anderen Modi genutzt werden. In dem Fall werden die externen Sollwerte einfach als relative Offsets zu den Sollwerten der anderen Modi addiert. Der Mover befindet sich dann allerdings nicht im Modus ExternalSetpointGeneration.

4.3 Operationsmodi Planar-Track

Der Planar-Track hat vier verschiedene Operationsmodi, die anzeigen, ob und in welcher Weise der Track seine Funktion als „Straße für Mover“ ausführt bzw. ausführen kann: Moving, Standing, Configuring und Uninitialized.



Moving

Im Modus Moving sind ein oder mehrere Mover dabei, auf dem Track eine Bewegung auszuführen (MoveOnTrack). Der erste Mover, der im Modus Standing eine Bewegung auf dem Track beginnt, ändert den Modus automatisch von Standing zu Moving. Dementsprechend ändert der letzte Mover, der seine Bewegung abschließt, den Modus zurück zu Standing. Während der Track im Modus Moving ist, darf kein Mover einen JoinTrack- oder LeaveTrack-Befehl ausführen. Wenn der Track in einer Planar-Group ist, blockiert er seine Fläche.

Standing

Im Modus Standing ist der Track von Movern benutzbar. Alle Mover auf dem Track stehen und warten auf Fahrbefehle. In diesem Modus sind JoinTrack-, LeaveTrack- und MoveOnTrack-Befehle für die Mover erlaubt. Jeder dieser Befehle beendet den Modus Standing des Tracks. Wenn der Track in einer Planar-Group ist, blockiert er seine Fläche nicht.

Configuring

Im Modus Configuring sind ein oder mehrere Mover dabei, den Track zu verlassen (LeaveTrack) oder sich auf den Track einzukoppeln (JoinTrack). Der erste Mover, der im Modus Standing den Track verlässt (oder sich einkoppelt), ändert den Modus automatisch von Standing zu Configuring. Dementsprechend ändert der letzte Mover, der das Verlassen oder Einkoppeln abschließt, den Modus zurück zu Standing. Während der Track im Modus Configuring ist, darf kein Mover einen MoveOnTrack-Befehl ausführen. Wenn der Track in einer Planar-Group ist, blockiert er seine Fläche nicht.

Uninitialized

Im Modus Uninitialized ist der Track von Movern nicht benutzbar. Er hat noch keine fertige geometrische Beschreibung. Wenn der Nutzer diese geometrische Beschreibung erstellt und enabled, dann wechselt der Track in den Modus Standing.

5 Parts

5.1 Parts und Koordinatensysteme

Ab Version V3.2.60: Das Part Feature, um welches es in diesem Abschnitt geht, steht zur Verfügung.

Die Fläche, auf der sich XPlanar-Mover bewegen, ist aus Statoren aufgebaut. Diese sind Quadrate mit einer Seitenlänge von 240 mm. Die gesamte Grundfläche unterteilt sich in ein oder mehrere Parts, die aus einer oder mehreren Statoren bestehen. Dabei kann ein Stator nur zu genau einem Part gehören, d. h., die Parts überlappen sich nicht. Die Menge aller Statoren eines Parts muss dabei eine zusammenhängende Fläche ergeben.

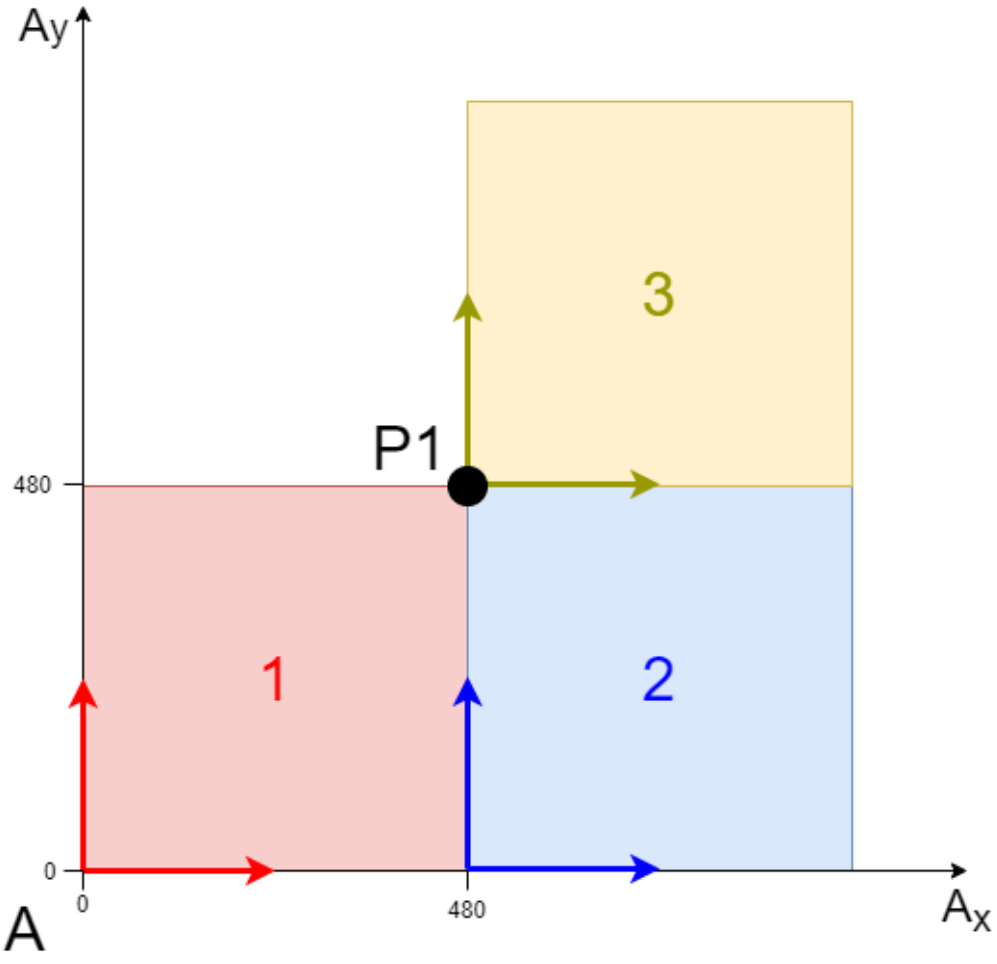
Diese geometrische Konfiguration des XPlanar-Systems ist meist statisch, sie ändert sich nicht zur Laufzeit des Systems. Um eine dynamische Konfiguration zu erzeugen, muss der Nutzer einzelne Parts bewegen bzw. versetzen, indem er ihnen mehr als eine Position zuweist und zur Laufzeit aktiviert.

Insgesamt ergibt sich die Schwierigkeit, dass Positionen nicht mehr eindeutig sind, bzw. es kein absolutes Koordinatensystem gibt. Der Nutzer kann mehrere (2D-) Koordinatensysteme definieren und seine Parts in diesen platzieren.

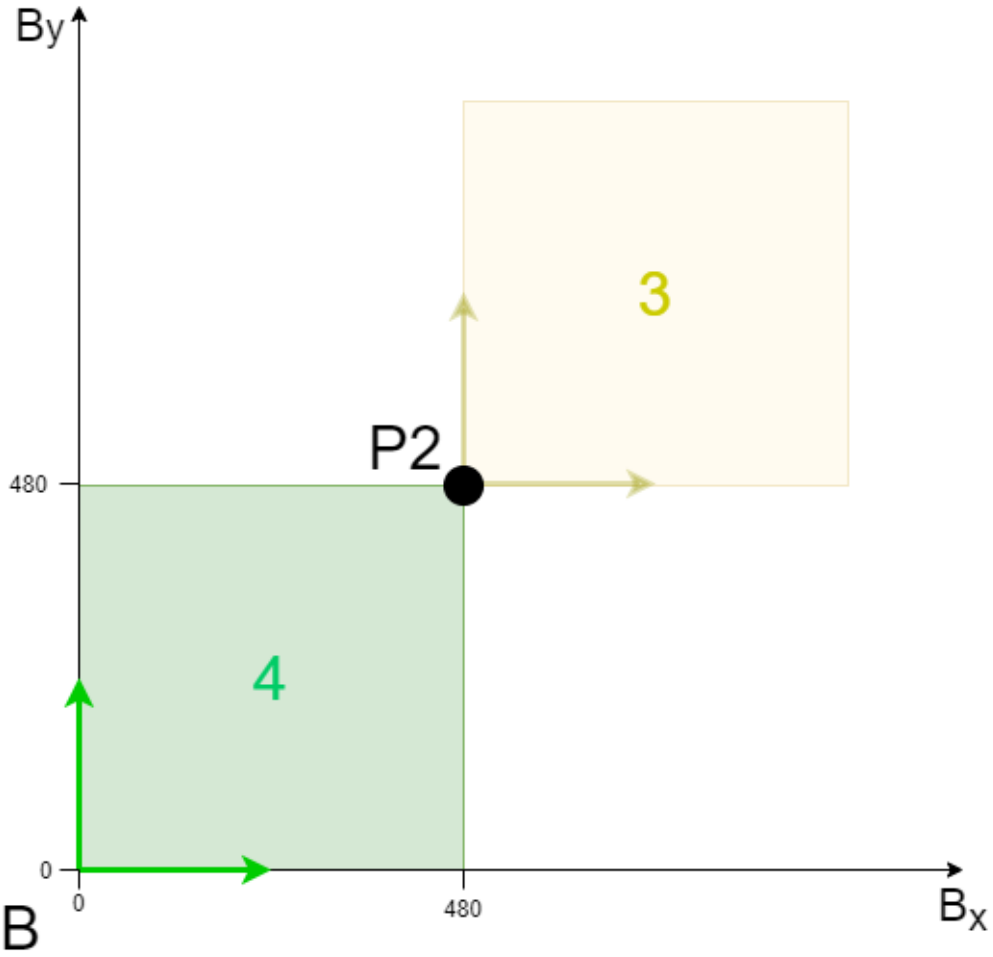
Eine 2D-Position (z. B. eines Movers oder Tracks) ist nun unvollständig, ohne die Angabe, in *welchem* Koordinatensystem sie sich befindet.

Beispiel

Es gibt vier Parts 1-4 und zwei Koordinatensystem A und B. Part 3 ist entweder in Koordinatensystem A oder B. Die anderen Parts sind fest einem Koordinatensystem zugeordnet. Die Position P1 liegt im Koordinatensystem A an der Position mit X=480 und y=480 (kurz P1=(480,480,A)).



Die Position P2 liegt im Koordinatensystem B an der Position mit X=480 und y=480 (kurz P2=(480,480,B)).



Ohne die Angabe des Koordinatensystems A wäre die Position P1 nicht von der Position P2 in Koordinatensystem B zu unterscheiden (480,480,B). Beide Positionen sind jedoch verschieden und liegen nicht einmal im selben Koordinatensystem, d. h., es existiert keine geometrische Verbindung zwischen ihnen (z. B. kann ein Mover nicht von P1 nach P2 fahren).

Zusätzlich zu den Koordinatensystemen A und B existieren noch lokale Part-Koordinatensysteme für jeden der Parts 1-4. Diese Koordinatensysteme haben ihren Ursprung jeweils in der unteren linken Ecke ihres Parts. Damit ist P1 = (0,0,3) = (0,480,2) = (480,480,1) in den Part-Koordinatensystemen der Parts 1-3.

Dies stimmt für Part 3 nur so lange, wie dieser in Koordinatensystem A liegt. Die transparente Darstellung von Part 3 in Koordinatensystem B zeigt, dass Part 3 seine Position aus Koordinatensystem A zu Koordinatensystem B wechseln kann. Die Position eines Parts ist identisch mit dem Ursprung seines Koordinatensystems. Part 3 würde also von Position P1 (in Koordinatensystem A) zu Position P2 bewegt, nun kann P1 nicht mehr im Koordinatensystem von Part 3 angegeben werden.

Stattdessen kann die Position P2 = (0,0,3) = (480,480,4) im Koordinatensystem von Part 3 (und 4) angegeben werden. Insgesamt sieht man, dass Positionen in Part-Koordinatensystemen *nicht statisch* sind, wenn der Part mit seinem Koordinatensystem bewegt wird.

5.2 Konfiguration

Ab Version V3.2.60: Das Part Feature, um welches es in diesem Abschnitt geht, steht zur Verfügung.

Die Konfiguration der Statoren eines Parts (d. h. der geometrischen Ausdehnung) und der Part Positionen in den verschiedenen Koordinatensystemen, findet im XPlanar-Treiber und nicht in der **MC Configuration** statt. Daher wird hier nicht beschrieben, wie diese Konfiguration erstellt wird.

Diese Informationen sind jedoch sowohl für die **MC Configuration** als auch für die PLC-Steuerung wichtig, daher werden sie von der Planar-Environment aus dem XPlanar-Treiber beim Aktivieren ausgelesen und von dort innerhalb der **MC Configuration** an alle Planar-Objekte verteilt. In der PLC wird über ein separates MC_PlanarPart [▶ 160]-Objekt die Steuerung der Part-Position und Zustände ermöglicht.

Um Positionen eindeutig anzugeben, werden die Positionsobjekte, PositionXYC und PositionXY, erweitert. Sie enthalten nun zusätzlich zu den Koordinatenwerten die ID des Koordinatensystems (ReferenceId). Da die Positionen nicht mehr eindeutig sind, kann die Position des Movers nun auf zwei Arten angegeben werden:

1. Als Position im übergeordneten Koordinatensystem (in dem sich der Part, auf dem der Mover ist, befindet).
2. Als Position im Part-Koordinatensystem. Im zyklischen Interface wird die Position im übergeordneten Koordinatensystem angegeben. Zusätzlich kann seine Position im Part-Koordinatensystem per Methode GetPositionOnCurrentPart [▶ 160] abgefragt werden.

Tracks können je nach Part-Position mit verschiedenen anderen Tracks verbunden sein. Diese Verbindungen können mit zwei Methoden, StartFromTrackAdvanced [▶ 169] und EndAtTrackAdvanced [▶ 169], geschlossen werden. Für die externe Sollwertgenerierung muss auch das Koordinatensystem, in dem sich der externe Sollwert befindet, angegeben werden. Dies geschieht mithilfe der Methode SetExternalSetpointReferenceId.

Diese Funktionalitäten der **MC Configuration** werden im Folgenden ausführlich beschrieben.

5.3 Positionen mit ReferenceId

Ab Version V3.2.60: Das Part Feature, um welches es in diesem Abschnitt geht, steht zur Verfügung.

Positionen ergeben mit der Einführung der Parts und der Koordinatensysteme keinen Sinn mehr ohne die Angabe des Referenzsystems, in dem sie sich befinden. Dafür gibt es für die Objekte PositionXYC und PositionXY die entsprechenden Erweiterungen um die Property „ReferenceId“. Sie speichern jetzt zusätzlich die ID des Referenzsystems.

Diese beiden Positionsobjekte, PositionXYC und PositionXY, können (und sollten) nun immer mit der expliziten ID des Referenzsystems genutzt werden. Die Angabe eines speziellen Referenzsystems ist dabei sicherer, da so explizit klar ist, welches System gemeint ist.

Die Angabe keines Referenzsystems, bzw. des Referenzsystems „Null“ ist nur noch in Ausnahmefällen erlaubt, wenn es nur ein statisches Koordinatensystem gibt in dem alle Parts eine feste („fixed“) Position haben. Die ID „Null“ wird dann intern in die ID des einzigen Koordinatensystems umgewandelt. In allen anderen Fällen wird das Referenzsystem „Null“ abgelehnt. Nicht gültige Referenzsystem (ungültige Objekt Id eines Parts/Koordinatensystems) außer der „Null“ werden immer abgelehnt.

6 Planar Motion-Komponenten

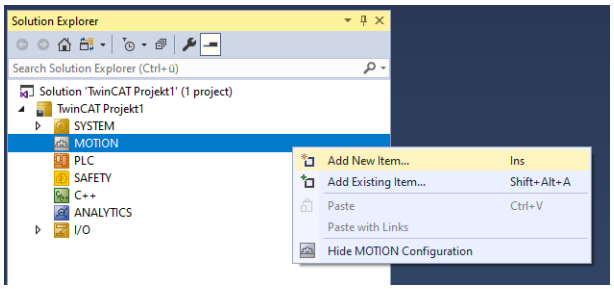
6.1 Planar-Mover

Der Planar-Mover ist ein Softwareobjekt, das einen XPlanar-Mover repräsentiert. Es fasst den Zustand des realen Movers (Position, Geschwindigkeit, usw.) für den Nutzer zusammen. Zusätzlich hat der Nutzer über den Planar-Mover die Möglichkeit, den Zustand des realen Movers zu beeinflussen bzw. zu steuern.

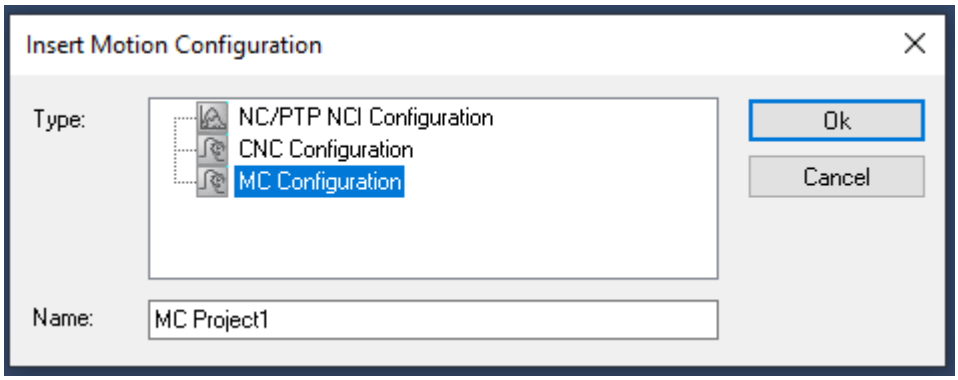
6.1.1 Konfiguration

✓ Um einen Planar-Mover anzulegen, muss zunächst eine **MC Configuration** angelegt werden.

1. Wählen Sie **MOTION** > **Add New Item…** aus.

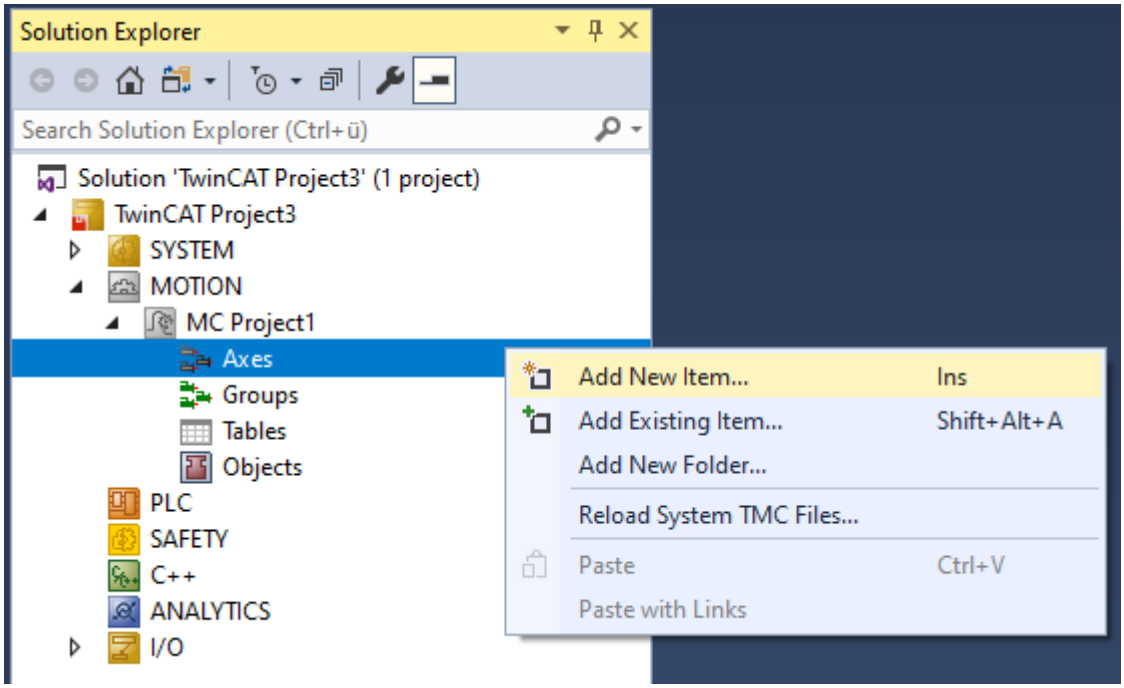


2. Wählen Sie im folgenden Dialogfenster **MC Configuration** aus und bestätigen Sie mit **OK**.

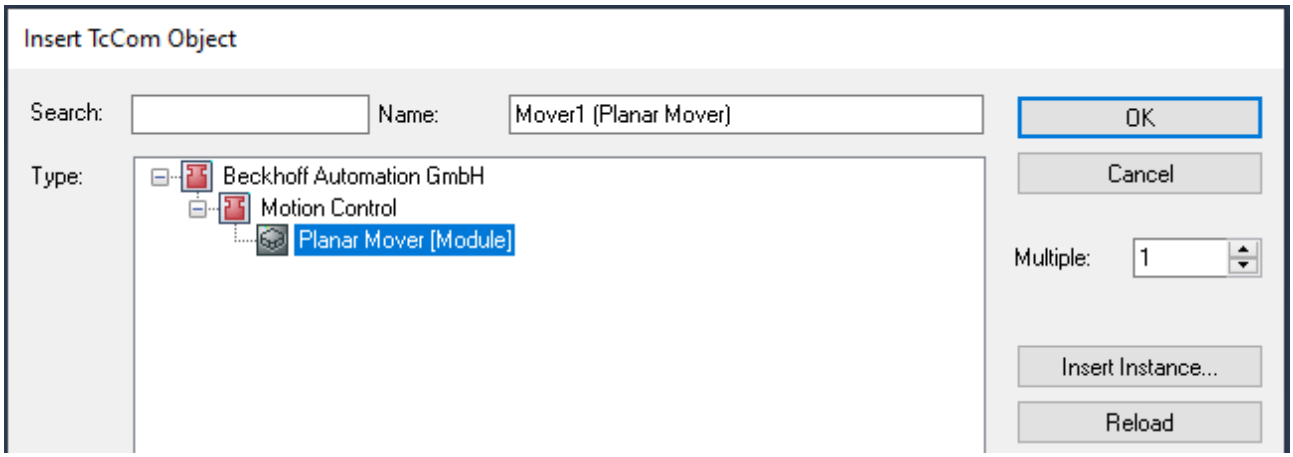


⇨ Sie haben ein MC Project angelegt.

3. Wählen Sie im erzeugten **MC Project** > **Axes** > **Add New Item…** aus.



4. Legen Sie im folgenden Dialogfenster einen (oder mehrere) Planar-Mover an und bestätigen Sie mit **OK**.



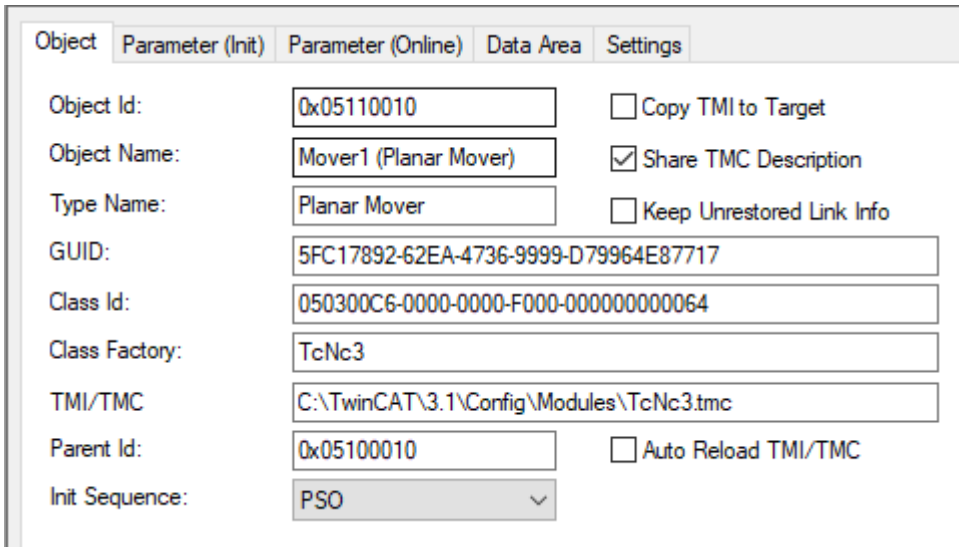
⇨ Der Planar-Mover ist nun angelegt und kann parametriert werden.

Detailbeschreibung öffnen

- Planar-Mover im Baum auswählen und doppelklicken.

Bedeutung der einzelnen Reiter

Object: Hier werden allgemeine Informationen (Name, Typ, Id, usw.) dargestellt.



Parameter (Init): Gibt Initialparameter an, die der Anwender verändern kann, um das Verhalten des Movers zu beeinflussen.

Parameter (Init) sollte vor dem Parametrieren in den Simulationsmodus (`TRUE`) versetzt werden, wenn kein Hardwaretreiber verlinkt wird. Der Parameter ist versteckt und wird nur sichbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.

Object | Parameter (Init) | Parameter (Online) | Data Area | Settings

	Name	Value	CS	Unit	Type	PTCID	Comment
-	General						
	Mover width	155.0	☐	mm	LREAL	0x0503...	Width that is used for internal collision checks.
	Mover height	155.0	☐	mm	LREAL	0x0503...	Height that is used for internal collision checks.
	C coordinate modulus	360.0	☐	°	LREAL	0x0503...	C coordinate modulus, can be set for disabled mover.
	C coordinate modulo tolerance wind...	0.0	☐	°	LREAL	0x0503...	C coordinate modulo tolerance window, can be set for disabled mover.
-	Maximum Dynamics						
+	Maximum dynamic XY	...	☐	mm per s, s^2, or s^3		0x0503...	Maximum dynamic of the mover in the XY plane.
+	Maximum dynamic C	...	☐	° per s, s^2, or s^3		0x0503...	Maximum dynamic of the movers C rotation.
+	Maximum dynamic Z	...	☐	mm per s, s^2, or s^3		0x0503...	Maximum dynamic of the movers Z coordinate.
+	Maximum dynamic A	...	☐	° per s, s^2, or s^3		0x0503...	Maximum dynamic of the movers A rotation.
+	Maximum dynamic B	...	☐	° per s, s^2, or s^3		0x0503...	Maximum dynamic of the movers B rotation.
-	Default Dynamics						
+	Default dynamic XY	...	☐	mm per s, s^2, or s^3		0x0503...	Default dynamic of the mover in the XY plane.
+	Default dynamic C	...	☐	° per s, s^2, or s^3		0x0503...	Default dynamic of the movers C rotation.
+	Default dynamic Z	...	☐	mm per s, s^2, or s^3		0x0503...	Default dynamic of the movers Z coordinate.
+	Default dynamic A	...	☐	° per s, s^2, or s^3		0x0503...	Default dynamic of the movers A rotation.
+	Default dynamic B	...	☐	° per s, s^2, or s^3		0x0503...	Default dynamic of the movers B rotation.
-	Monitoring						
	Position Lag Monitoring Enabled	TRUE	☐		BOOL	0x0503...	
+	Maximum Position Lag Value	...	☐	mm, °		0x0503...	A vector of six numeric values corresponding to the six coordinates of th
+	Maximum Position Lag Filter Time	...	☐	s		0x0503...	A vector of six numeric values corresponding to the six coordinates of th

☐ Show Online Values ☐ Show Hidden Parameter Expand All Collapse All

Die Initialparameter sind zunächst so eingestellt, dass der Planar-Mover (fertig verknüpft) mit der Hardware verfahren werden kann. Wenn der Anwender ohne Hardware verfahren möchte, muss der Parameter „Simulation Mode“ auf TRUE gestellt werden. Im Simulationsmodus sollten die Parameter „Initial Position“ und „PartOID“ (**ab Version V3.2.60**) gesetzt werden. Hat der reale Mover keine Standardmaße, so müssen die Parameter „Mover width“ und „Mover height“ angepasst werden.

Ab Version V3.1.10.30: Die versteckten Parameter „Minimal/Maximal Position“ dienen dazu, für die C-Achse festzulegen, wann der Mover in den Befehlsmodus CRotation wechselt. Für alle Zielpositionen von C-Bewegungen legen die Parameter „C coordinate modulus“ und „C coordinate modulo tolerance window“ im Fall von Modulo Positionierung die Umrechnung in die absolute Zielposition fest. Für Details siehe Modulo Positionierung.

Ab Version V3.1.10.51: Auch AdoptTrackOrientation ist eine C-Bewegung und wird entsprechend von „C coordinate modulus“ und „C coordinate modulo tolerance window“ beeinflust. Für Details siehe AdoptTrackOrientation [▶ 63].

Ab Version V3.2.60: Der Parameter „PartOID“ gibt für den Simulationsmodus an, auf welchem Part die „Initial Position“ liegt. Die Parameter „PartOID“, „Simulation Mode“ und „Initial Position“ sind alle versteckt und in einer eigenen Gruppierung „Simulation“.

Weitere Parameter sind die „Maximum Dynamic(s)“ und die „Default Dynamic(s)“. Außerdem gibt es noch „Monitoring“-Parameter, die eine Positionsüberwachung des realen Movers aktivieren bzw. parametrieren.

Parameter (Online): Zeigt den Zustand des Movers während der Laufzeit des Objektes. Die aktuell vorgegebene Position („SetPos“) und reale Position („ActPos“) sowie Zustandsinformationen werden angezeigt.

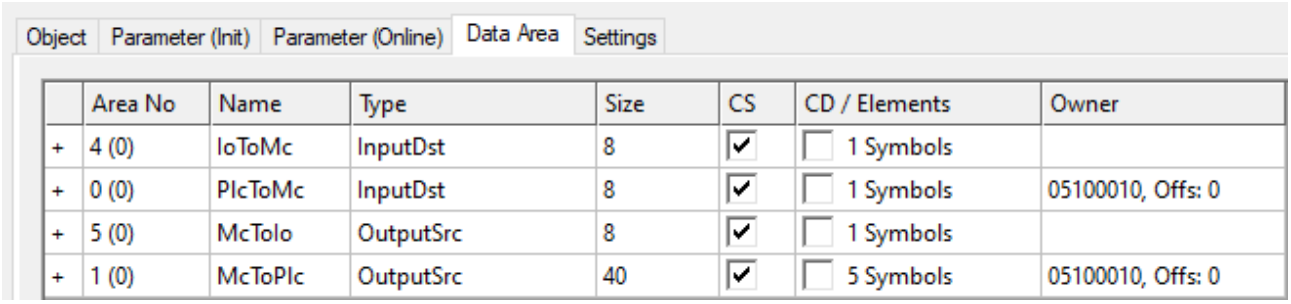
Object | Parameter (Init) | Parameter (Online) | Data Area | Settings

	Name	Online	CS	Unit	Type	PTCID	Comment
+	SetPos	...	☐	mm, °		0x050300D2	Mover position, read only.
	CoordinateSystemOID	00000000	☐		OTCID	0x05030126	The coordinate system the mover is on, set and act position are given in this system.
	GroupOID	00000000	☐		OTCID	0x050300C3	Object id of the PlanarGroup the mover is in, read only.
	State	Enabled	☐		MC.MC_PLANAR_STATE	0x050300B6	State, read only.
	Command mode	FreeMovement	☐		MC.MC_PLANAR_MOVER_COMMAND_MODE	0x050300B7	Command mode, read only.
+	ActPos	...	☐			0x050300C1	Mover hardware position, read only.
+	Mover SetPos on track	...	☐			0x050300C2	Mover state on track, read only.
	External setpoint generation mode	None	☐		MC.MC_EXTERNAL_SET_POSITION_MODE	0x050300DB	Indicates which external setpoint generation mode is active.

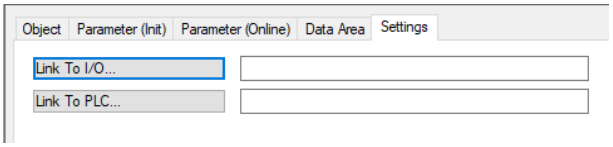
Ab Version V3.1.10.30: Der Parameter „External setpoint generation“ zeigt an, ob der Mover (absoluten oder relativen) externen Sollwerten des Nutzers folgt.

Ab Version V3.2.60: Der Parameter „CoordinateSystemOID“ gibt an, in welchem Koordinatensystem SetPos und ActPos angegeben sind, und in welchem Koordinatensystem sich der Mover befindet.

Data Area: Zeigt Speicherbereiche, über die der Mover mit anderen Objekten verknüpft ist und Informationen austauscht.



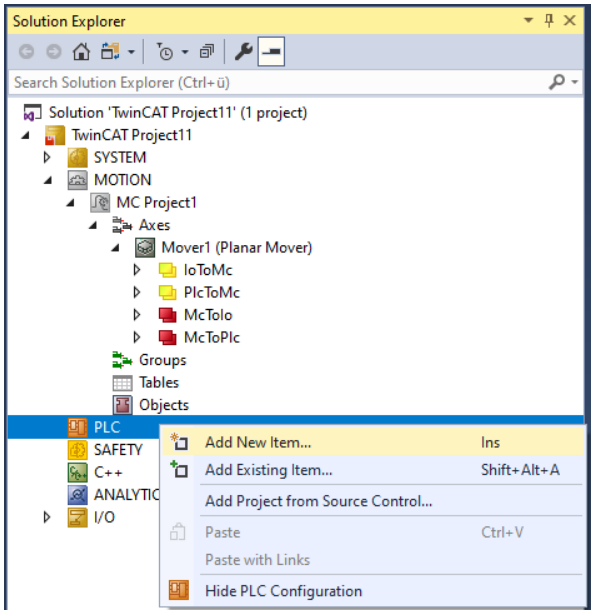
Settings: Hier kann der Anwender Verknüpfungen herstellen. Mit den beiden Buttons „Link To …“ kann der Planar-Mover mit den Movern in der PLC und im Xplanar-Treiber verknüpft werden.



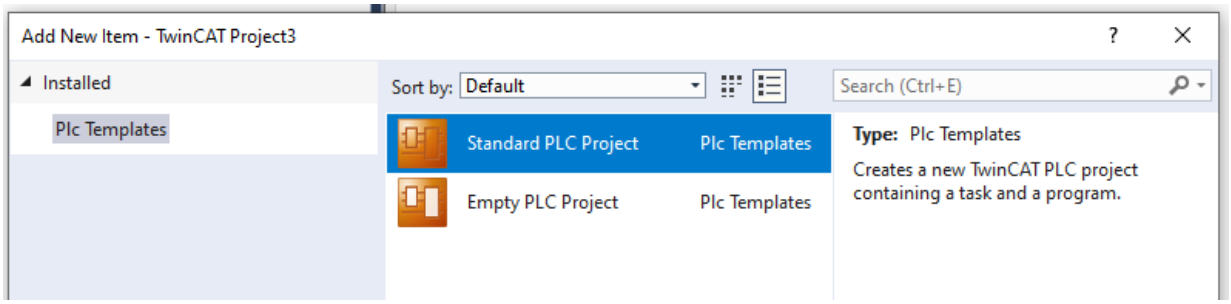
6.1.2 PLC anlegen

✓ Um den Mover, den Track oder die Group zu steuern, die Geometrie einer Environment zu erstellen oder ein Planar-Feedback zu nutzen, muss eine PLC angelegt werden.

1. Wählen Sie **PLC > Add New Item…** aus.



2. Wählen Sie im folgenden Dialogfenster **Standard PLC Project** aus und bestätigen Sie mit **OK**.



3. Fügen Sie dem PLC-Projekt die Bibliotheken Tc3_Physics und Tc3_Mc3PlanarMotion hinzu, siehe Bibliotheken einfügen [▶ 127].

⇨ Die PLC ist angelegt und Sie können, wie in den folgenden Beispielen beschrieben, Befehle an die entsprechenden Objekte geben.

6.1.3 Beispiel „Planar-Mover anlegen und verfahren“

Anhand dieser kurzen Anleitung werden Sie ein TwinCAT-Projekt anlegen, das einen Planar-Mover enthält und auf einfache Weise verfährt.

Planar-Mover anlegen

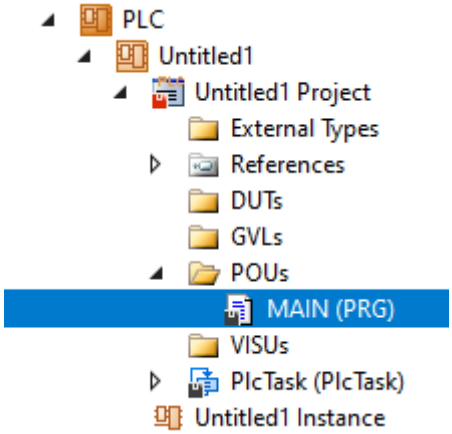
✓ Siehe Konfiguration [▶ 18].

1. Legen Sie für dieses Beispiel einen Planar-Mover an.
2. Versetzen Sie „Parameter (Init)“ in den Simulationsmodus (`TRUE`). Der Parameter ist versteckt und wird nur sichtbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** den oder die Mover („MC_PlanarMover [▶ 145]“) wie folgt an.



⇨ Diese/r repräsentiert den/die Mover in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, einen Planar-Mover, eine Zustandsvariable für eine Zustandsmaschine und eine Zielposition für einen Fahrbefehl des Movers an.

```
PROGRAM MAIN
VAR
    mover : MC_PlanarMover;
    state : UDINT;
    target_position : PositionXYC;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

⇨ Dieser Programmcode aktiviert den Mover und verfährt ihn zur Position x=100 und y=100.

```
CASE state OF
  0:
    mover.Enable(0);
    state := 1;
  1:
    IF mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 2;
    END_IF
  2:
    target_position.SetValuesXY(100, 100);
    mover.MoveToPosition(0, target_position, 0, 0);
    state := 3;
END_CASE
```

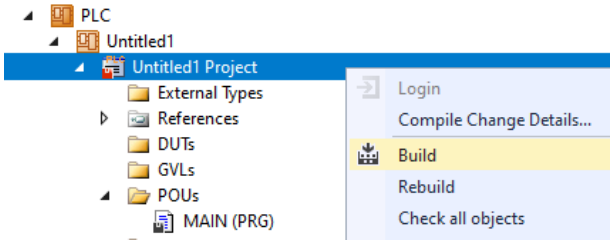
Befehl abschicken

4. Um den Befehl abzuschicken, müssen Sie den Mover nach dem END_CASE zyklisch mit seiner Update-Methode aufrufen:

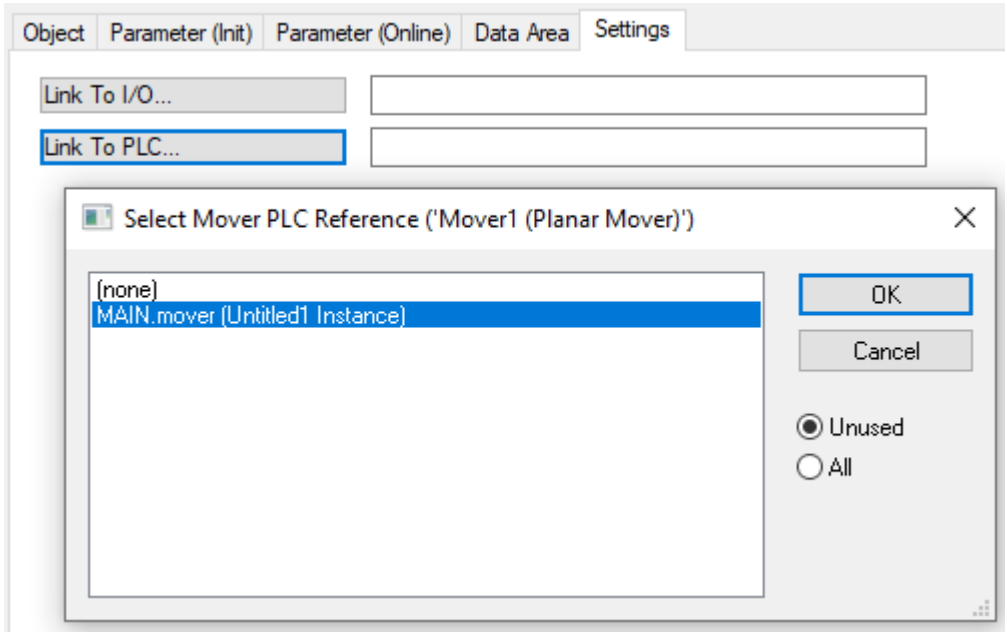
```
mover.Update();
```

Durch das Bauen der PLC wird ein Symbol des „PLC-Movers" erzeugt, welches anschließend mit der Mover-Instanz im MC-Projekt verknüpft werden kann.

1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**



⇨ Anschließend kann der Planar-Mover im „MC Project“ (Doppelklick) mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.



Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Der Mover steht am Ende des Zustandsautomaten (state=3) auf der gewünschten Position.

MAIN [Online] TwinCAT Project11

TwinCAT_Project11.Untitled1.MAIN

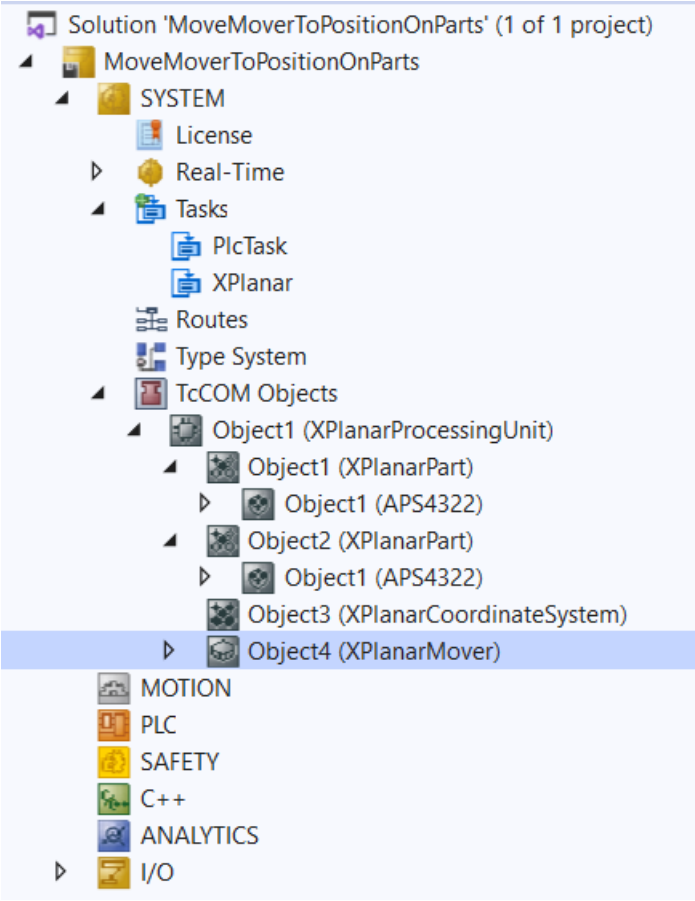
Expression	Type	Value	Prepared value	Address	Comment
mover	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC_PLANAR_M...			%Q*	Mover data that is tra...rred from the Plana...
MCTOPLC	CDT_MCTOPLC_PLANAR_M...			%I*	Mover data that is tra...rred from the Plana...
STD	REFERENCE TO CDT_MCTO...			%IB*	Mover standard data t...is transferred from ...
SET	REFERENCE TO CDT_MCTO...			%IB*	Mover setpoint data th...is transferred from t...
SetPos	MoverVector				Current position.
x	LREAL	100			X coordinate.
y	LREAL	100			Y coordinate.
z	LREAL	0			Z coordinate.
a	LREAL	0			A coordinate.
b	LREAL	0			B coordinate.
c	LREAL	0			C coordinate.
SetVelo	MoverVector				Current velocity.
SetAcc	MoverVector				Current acceleration.
DcTimeStamp	ULINT	66246161725...			Current time stamp.
PhysicalAreaID	UDINT	0			Current physical area id.
ACT	REFERENCE TO CDT_MCTO...			%IB*	Mover actpoint data th...is transferred from t...
COORDMODE	REFERENCE TO CDT_MCTO...			%IB*	Mover coordinate mod...ormation that is tra...
SETONTRACK	REFERENCE TO CDT_MCTO...			%IB*	Mover busy informatio...at is transferred fro...
Error	BOOL	FALSE			Flag indicating a PlanarMover error.
ErrorId	UDINT	0			Error id indicating the PlanarMover error type.
state	UDINT	3			
target_position	PositionXYC				

6.1.4 Beispiel „Planar-Mover auf Planar-Parts verfahren"

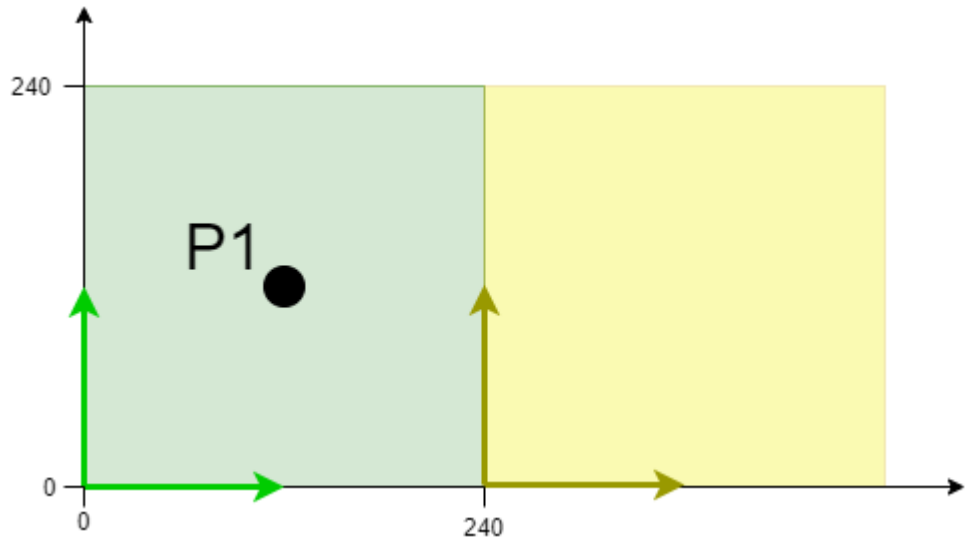
In diesem Beispiel wird ein Planar-Mover auf zwei Planar-Parts verfahren.

Ausgangspunkt

Sie starten mit einer Solution, die eine fertig konfigurierte XPlanar Processing Unit enthält. Unter der XPlanar Processing Unit sind zwei Parts, ein Koordinatensystem und ein Mover angelegt. Unter den beiden Parts ist jeweils eine Kachel angelegt.



Folgende geometrische Situation ist eingestellt: die beiden Parts liegen nebeneinander und der Mover startet in der Mitte des linken Parts (Position P1). Beide Parts sind nicht beweglich und die Konfiguration ist daher statisch.



Ausgehend von dieser Konfiguration wird das Beispiel entwickelt.

Die Erstellung der Ausgangssituation wird in der Dokumentation der XPlanar Processing Unit beschrieben.

Planar-Mover und Planar-Environment anlegen

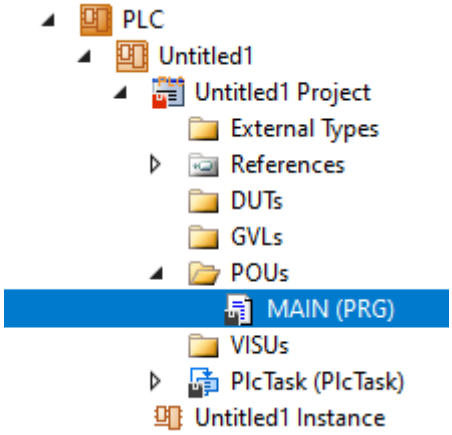
1. Legen Sie für dieses Beispiel einen Planar-Mover an, siehe Konfiguration [▶ 18].
2. Legen Sie eine Planar-Environment an, siehe Konfiguration [▶ 98].

3. Stellen Sie den Initialparameter XPlanar processing unit OID auf die Objekt Id der XPlanar Processing Unit. Damit ist das Part feature für alle **MC Configuration** Objekte aktiviert (besonders für den angelegten Planar-Mover).

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** den oder die Mover („MC_PlanarMover [▶ 145]“) wie folgt an.



⇨ Diese/r repräsentiert den/die Mover in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, einen Planar-Mover, eine Zustandsvariable für eine Zustandsmaschine und eine Zielposition für einen Fahrbefehl des Movers an.

```
PROGRAM MAIN
VAR
    mover : MC_PlanarMover;
    state : UDINT;
    target_position : PositionXYC;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

⇨ Dieser Programmcode aktiviert den Mover und verfährt ihn zur Position x=100 und y=100, diese ist im Part-Koordinatensystem des rechten Parts angegeben (dessen Objekt Id ist 16#01010030).

```
CASE state OF
  0:
    mover.Enable(0);
    state := 1;
  1:
    IF mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 2;
    END_IF
  2:
    target_position.SetValuesXYCReferenceId(100, 100, 0, 16#01010030);
    mover.MoveToPosition(0, target_position, 0, 0);
    state := 3;
END_CASE
```

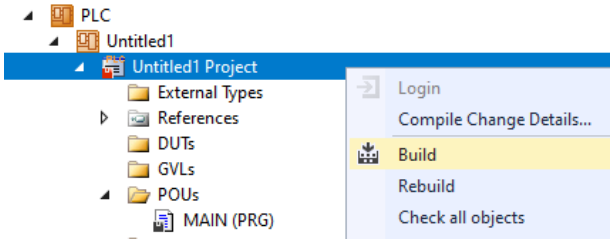
Befehl abschicken

4. Um den Befehl abzuschicken, müssen Sie den Mover nach dem END_CASE zyklisch mit seiner Update-Methode aufrufen:

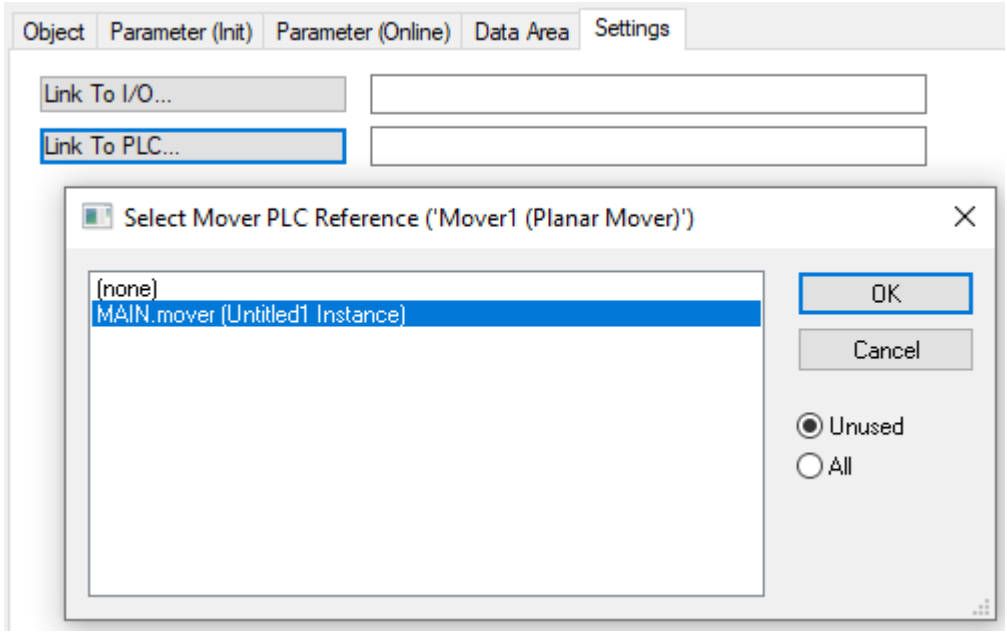
```
mover.Update();
```

Durch das Bauen der PLC wird ein Symbol des „PLC-Movers" erzeugt, welches anschließend mit der Mover-Instanz im MC-Projekt verknüpft werden kann.

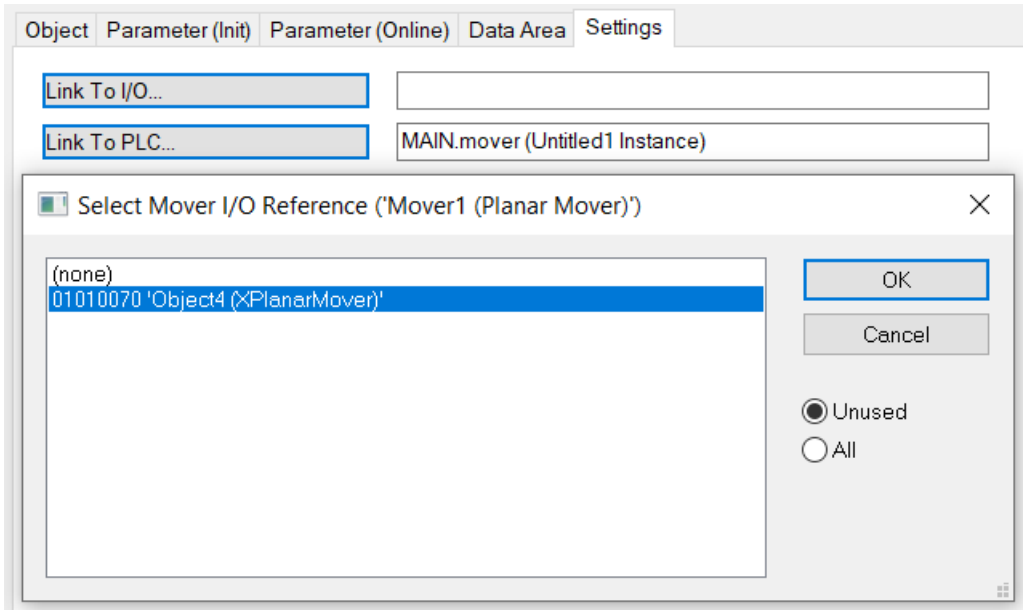
1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**



⇨ Anschließend kann der Planar-Mover im „MC Project“ (Doppelklick) mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.



⇨ Zusätzlich muss der Planar-Mover im „MC Project“ (Doppelklick) mit dem Button **Link To I/O…** im Reiter **Settings** verknüpft werden.



Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .

3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Der Mover steht am Ende des Zustandsautomaten (state=3) auf der gewünschten Position. Die Position ist im Koordinatensystem angegeben und nicht wie die target_position im Part-Koordinatensystem des rechten Parts (dessen Objekt Id ist 16#01010030). Beide Systeme sind zueinander um 240 mm in x-Richtung verschoben. Außerdem ist zu sehen, dass die Achsen a, b und z ein leichtes Rauschen aufweisen. Dieses wird durch die Simulation der XPlanar Processing Unit erzeugt und wirkt sich beim Aufstarten auf die initial übernommene Position aus.

Expression	Type	Value
mover	MC_PlanarMover	
PLCTOMC	CDT_PLCTOMC...	
MCTOPLC	CDT_MCTOPLC...	
STD	REFERENCE TO...	
SET	REFERENCE TO...	
SetPos	MoverVector	
x	LREAL	340
y	LREAL	100
z	LREAL	2.005329507...
a	LREAL	-0.00078074...
b	LREAL	-0.00097010...
c	LREAL	-3.05631426...
SetVelo	MoverVector	
SetAcc	MoverVector	
DcTimeStamp	ULINT	7562220860...
PhysicalAreaID	UDINT	16842848
ACT	REFERENCE TO...	
COORDMODE	REFERENCE TO...	
SETONTRACK	REFERENCE TO...	
Error	BOOL	FALSE
ErrorId	UDINT	0
state	UDINT	3
target_position	PositionXYC	

6.1.5 Beispiel „Planar-Mover mit Hilfsachsen anlegen und verfahren“

Anhand dieser kurzen Anleitung werden Sie ein TwinCAT-Projekt anlegen, das einen Planar-Mover enthält und auf einfache Weise verfährt.

Planar-Mover anlegen

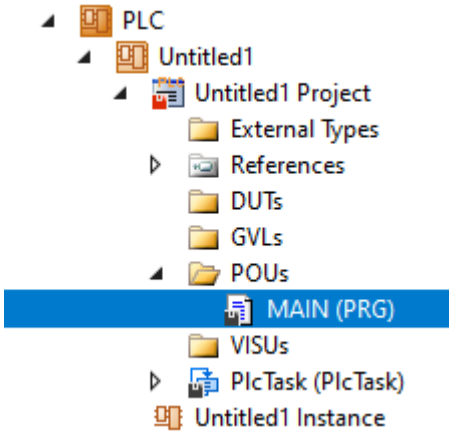
✓ Siehe Konfiguration [▶ 18].

1. Legen Sie einen Planar-Mover an.
2. Versetzen Sie „Parameter (Init)“ in den Simulationsmodus (`TRUE`). Der Parameter ist versteckt und wird nur sichbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** den oder die Mover („MC_PlanarMover [▶ 145]“) wie folgt an.



⇨ Diese/r repräsentiert den/die Mover in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, einen Planar-Mover, eine Zustandsvariable für eine Zustandsmaschine und eine Zielposition für einen Fahrbefehl des Movers an.

```
PROGRAM MAIN
VAR
    mover : MC_PlanarMover;
    state : UDINT;
    target_a : LREAL := 1.0;
    target_b : LREAL := -1.0;
    target_c : LREAL := 3.0;
    target_z : LREAL := 5.0;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

⇨ Dieser Programmcode aktiviert den Mover und verfährt die vier Hilfsachsen.

```
CASE state OF
  0:
    mover.Enable(0);
    state := 1;
  1:
    IF mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 2;
    END_IF
  2:
    mover.MoveA(0, target_a, 0);
    mover.MoveB(0, target_b, 0);
    // Since Version V3.1.10.11 MoveC has an options parameter,
    // details can be found in the CRotation example
    // and the options descriptions
    //mover.MoveC(0, target_c, 0);  // until version V3.1.10.11
    mover.MoveC(0, target_c, 0, 0); // since version V3.1.10.30
    mover.MoveZ(0, target_z, 0);
    state := 3;
END_CASE
```

Weiterführende Informationen:

- Beispiel „Planar-Mover in CRotation-Modus verfahren“ [▶ 39]
- Limits und Optionen der Bewegungsbefehle [▶ 41]

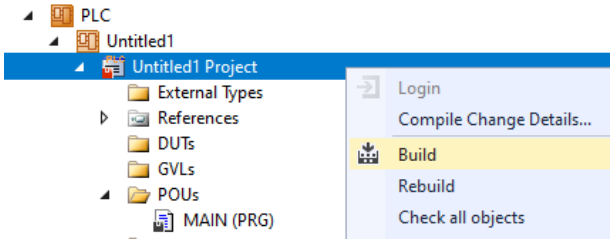
Befehl abschicken

4. Um den Befehl abzuschicken, müssen Sie den Mover nach dem END_CASE zyklisch mit seiner Update-Methode aufrufen:

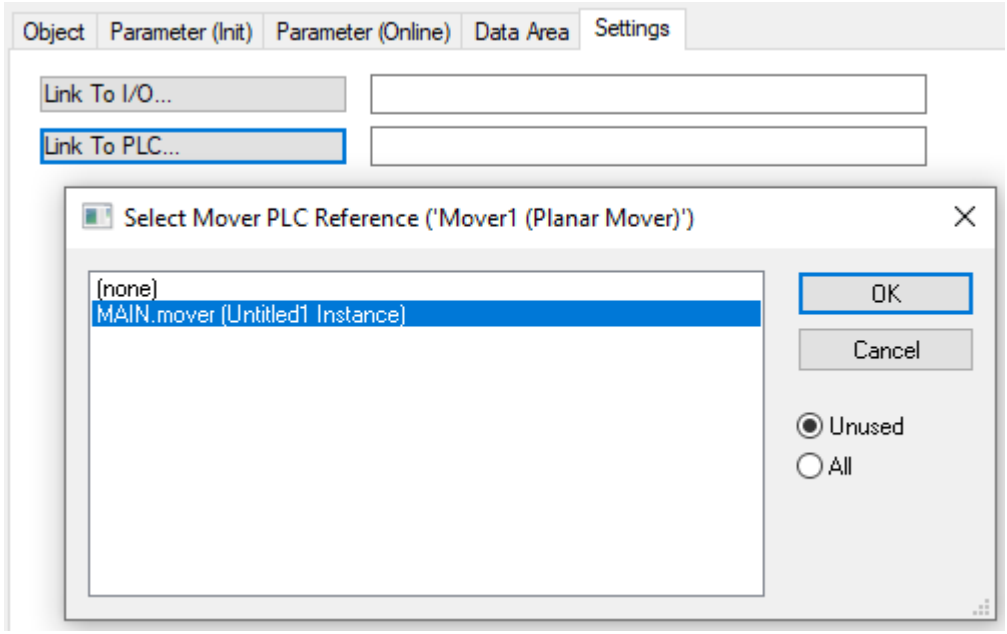
```
mover.Update();
```

Durch das Bauen der PLC wird ein Symbol des „PLC-Movers" erzeugt, welches anschließend mit der Mover-Instanz im MC-Projekt verknüpft werden kann.

1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**



⇨ Anschließend kann der Planar-Mover im „MC Project“ (Doppelklick) mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.



Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Der Mover steht am Ende des Zustandsautomaten (state=3) auf der gewünschten Position.

MAIN [Online] | TwinCAT Project14

TwinCAT_Project14.Untitled1.MAIN

Expression	Type	Value	Prepared value	Address	Comment
mover	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC_PLA...			%Q*	Mover data that is tra...rred from the Plana...
MCTOPLC	CDT_MCTOPLC_PLA...			%I*	Mover data that is tra...rred from the Plana...
STD	REFERENCE TO CDT...			%IB*	Mover standard data t...is transferred from ...
SET	REFERENCE TO CDT...			%IB*	Mover setpoint data th...is transferred from t...
SetPos	MoverVector				Current position.
x	LREAL	0			X coordinate.
y	LREAL	0			Y coordinate.
z	LREAL	5			Z coordinate.
a	LREAL	0.999999999999...			A coordinate.
b	LREAL	-0.99999999999...			B coordinate.
c	LREAL	2.999999999999...			C coordinate.
SetVelo	MoverVector				Current velocity.
SetAcc	MoverVector				Current acceleration.
DcTimeStamp	ULINT	6630642690730...			Current time stamp.
PhysicalAreaID	UDINT	0			Current physical area id.
ACT	REFERENCE TO CDT...			%IB*	Mover actpoint data th...is transferred from t...
COORDMODE	REFERENCE TO CDT...			%IB*	Mover coordinate mod...ormation that is tra...
SETONTRACK	REFERENCE TO CDT...			%IB*	Mover busy informatio...at is transferred fro...
Error	BOOL	FALSE			Flag indicating a PlanarMover error.
ErrorId	UDINT	0			Error id indicating the PlanarMover error type.
state	UDINT	3			
target_a	LREAL	1			
target_b	LREAL	-1			
target_c	LREAL	3			
target_z	LREAL	5			

6.1.6 Beispiel „Planar-Mover mit External Setpoint Generation anlegen und verfahren“

Anhand dieser kurzen Anleitung werden Sie ein TwinCAT-Projekt anlegen, das einen Planar-Mover enthält und auf einfache Weise mittels externer Sollwertgenerierung verfährt.

Planar-Mover anlegen

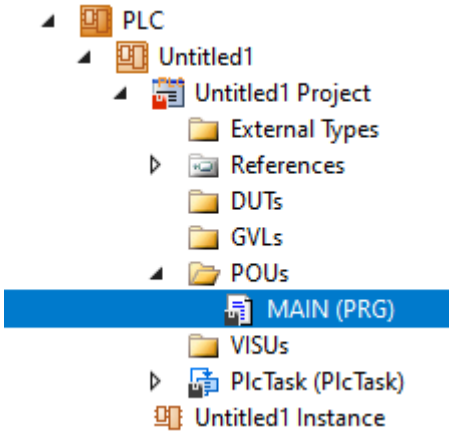
✓ Siehe Konfiguration [▶ 18].

1. Legen Sie einen Planar-Mover an.
2. Versetzen Sie „Parameter (Init)“ in den Simulationsmodus (`TRUE`). Der Parameter ist versteckt und wird nur sichbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** den oder die Mover („MC_PlanarMover [▶ 145]“) wie folgt an.



⇨ Diese/r repräsentiert den/die Mover in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, einen Planar-Mover, eine Zustandsvariable für eine Zustandsmaschine und Variablen für den externen Sollwert an.

```
PROGRAM MAIN
VAR
    mover : MC_PlanarMover;
    state : UDINT;
    p,v,a : MoverVector;
    deltat : LREAL := 0.001;
velo, acc, jerk : LREAL;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

   ⇨ Dieser Programmcode aktiviert den Mover und startet die externe Sollwertgenerierung. Dann wird ein Profil abgefahren, das mit positiver Geschwindigkeit endet. Das anschließende Stopp der externen Sollwertgenerierung sorgt dafür, dass der Mover seine Geschwindigkeit auf Null reduziert und nach dem Anhalten im Zustand FreeMovement ist (dies geschieht mit der Maximaldynamik des Movers).

```
CASE state OF
  0:
    mover.Enable(0);
    state := 1;
  1:
    IF mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 2;
    END_IF
  2:
    p.x := 0.0; v.x := 0.0; a.x := 0.0;
    mover.StartExternalSetpointGeneration(0,0);
    mover.SetExternalSetpoint(p,v,a);
    state := 3;
  3:
    velo := v.x;
    acc := a.x;
    p.x := p.x + deltat * velo + deltat * deltat / 2 * acc + deltat * deltat * deltat / 6 *
jerk;
    v.x := v.x + deltat * acc + deltat * deltat / 2 * jerk;
    a.x := a.x + deltat * jerk;
    mover.SetExternalSetpoint(p,v,a);
    IF a.x >= 10.0 THEN
      jerk := -1;
    END_IF;
    IF a.x <= 0.0 THEN
      state := 4;
    END_IF;
  5:
    mover.StopExternalSetpointGeneration(0);
    state := 6;
END_CASE
```

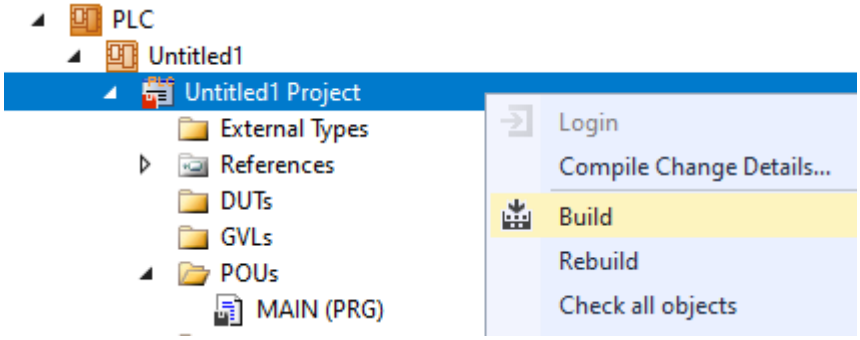
Befehl abschicken

4. Um die Kommandos zu senden, müssen Sie nach dem END_CASE die Update-Methode des Movers triggern:

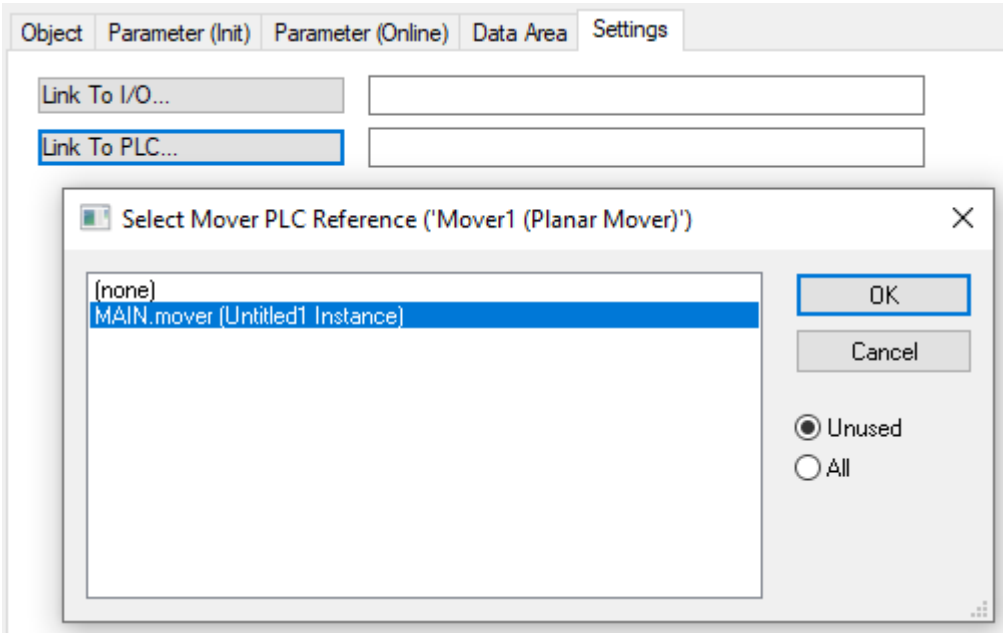
```
mover.Update();
```

Durch das Bauen der PLC wird ein Symbol des „PLC-Movers" erzeugt, welches anschließend mit der Mover-Instanz im MC-Projekt verknüpft werden kann.

1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**



⇨ Anschließend kann der Planar-Mover im „MC Project“ (Doppelklick) mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.



Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Der Mover steht am Ende des Zustandsautomaten (state=6) auf der gewünschten positiven x-Position.

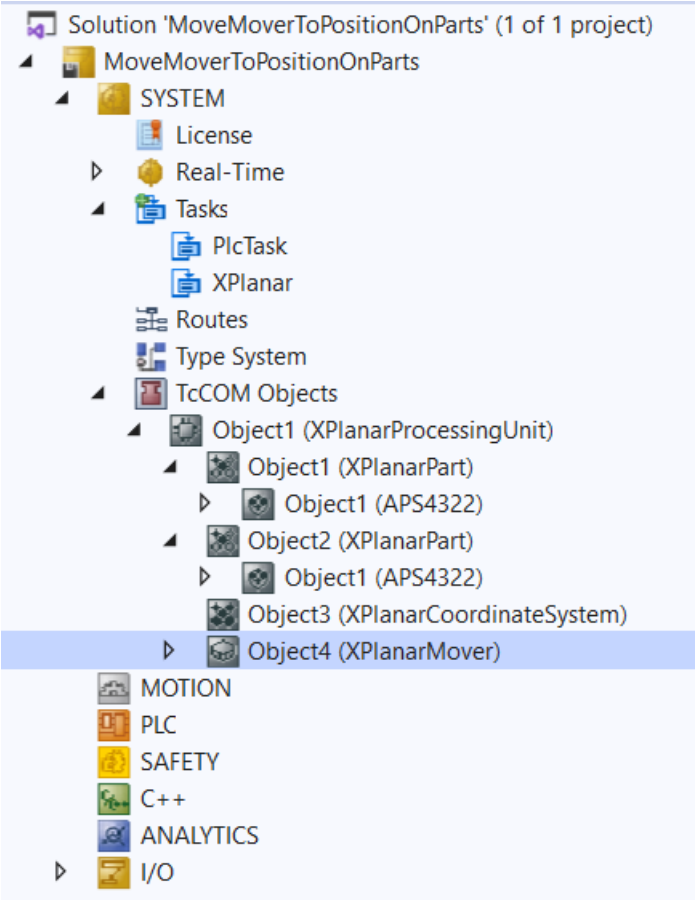
Expression	Type	Value	Prepared value	Address	Comment
mover	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC...			%Q*	Mover data that i...ansferred from ...
MCTOPLC	CDT_MCTOPLC...			%I*	Mover data that i...ansferred from ...
STD	REFERENCE TO...			%IB*	Mover standard d...that is transfer...
SET	REFERENCE TO...			%IB*	Mover setpoint d...hat is transferr...
SetPos	MoverVector				Current position.
x	LREAL	1007.249774...			X coordinate.
y	LREAL	0			Y coordinate.
z	LREAL	0			Z coordinate.
a	LREAL	0			A coordinate.
b	LREAL	0			B coordinate.
c	LREAL	0			C coordinate.
SetVelo	MoverVector				Current velocity.
SetAcc	MoverVector				Current acceleration.
DcTimeStamp	ULINT	7238174810...			Current time stamp.
PhysicalAreaID	UDINT	0			Current physical area id.
ACT	REFERENCE TO...			%IB*	Mover actpoint d...hat is transferr...
COORDMODE	REFERENCE TO...			%IB*	Mover coordinate...de information...
SETONTRACK	REFERENCE TO...			%IB*	Mover busy information that is tran...
Error	BOOL	FALSE			Flag indicating a Planar Mover error.
ErrorId	UDINT	0			Error id indicating...e Planar Mover ...
state	UDINT	6			
p	MoverVector				

6.1.7 Beispiel „Planar-Mover mit External Setpoint Generation auf Planar-Parts verfahren"

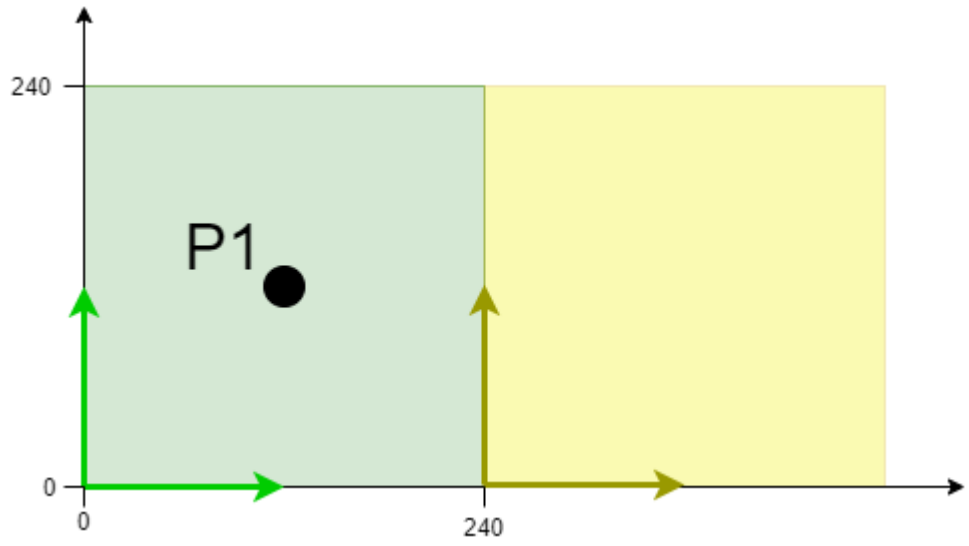
In diesem Beispiel wird ein Planar-Mover auf zwei Planar-Parts mit der externen Sollwertgenerierung verfahren.

Ausgangspunkt

Sie starten mit einer Solution, die eine fertig konfigurierte XPlanar Processing Unit enthält. Unter der XPlanar Processing Unit sind zwei Parts, ein Koordinatensystem und ein Mover angelegt. Unter den beiden Parts ist jeweils eine Kachel angelegt.



Folgende geometrische Situation ist eingestellt: die beiden Parts liegen nebeneinander und der Mover startet in der Mitte des linken Parts (Position P1). Beide Parts sind nicht beweglich; die Konfiguration ist daher statisch.



Ausgehend von dieser Konfiguration wird das Beispiel entwickelt.

Die Erstellung der Ausgangssituation wird in der Dokumentation der XPlanar Processing Unit beschrieben.

Planar-Mover und Planar-Environment anlegen

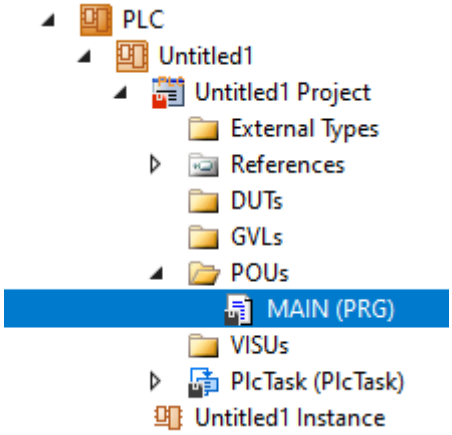
1. Legen Sie für dieses Beispiel einen Planar-Mover an, siehe Konfiguration [▶ 18].
2. Legen Sie eine Planar-Environment an, siehe Konfiguration [▶ 98].

3. Stellen Sie den Initialparameter XPlanar processing unit OID auf die Objekt Id der XPlanar Processing Unit. Damit ist das Part feature für alle **MC Configuration** Objekte aktiviert (besonders für den angelegten Planar-Mover).

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** den oder die Mover („MC_PlanarMover [▶ 145]“) wie folgt an.



⇨ Diese/r repräsentiert den/die Mover in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, einen Planar-Mover, eine Zustandsvariable für eine Zustandsmaschine und Variablen für den externen Sollwert an.

```
PROGRAM MAIN
VAR
    mover : MC_PlanarMover;
    state : UDINT;
    p,v,a : MoverVector;
    deltat : LREAL := 0.01;
    refsys : OTCID := 0;
    velo, acc, jerk : LREAL;
    feedback : MC_PlanarFeedback;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

⇨ Dieser Programmcode aktiviert den Mover und startet die externe Sollwertgenerierung. Dann wird ein Profil abgefahren, das mit positiver Geschwindigkeit endet. Das anschließende Stopp der externen Sollwertgenerierung sorgt dafür, dass der Mover seine Geschwindigkeit auf Null reduziert und nach dem Anhalten im Zustand FreeMovement ist (dies geschieht mit der Maximaldynamik des Movers).

```
CASE state OF
  0:
    mover.Enable(0);
    state := 1;
  1:
    IF mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 2;
    END_IF
  2:
    p := mover.MCTOPLC.SET.SetPos;
    v.x := 0.0; a.x := 0.0;
    mover.StartExternalSetpointGeneration(0,0);
    mover.SetExternalSetpointReferenceId(feedback,p,v,a,refsys);
    jerk := 10;
    state := 3;
  3:
    velo := v.x;
    acc := a.x;
    p.x := p.x + deltat * velo + deltat * deltat / 2 * acc + deltat * deltat * deltat / 6 *
jerk;
    v.x := velo + deltat * acc + deltat * deltat / 2 * jerk;
    a.x := acc + deltat * jerk;
    mover.SetExternalSetpointReferenceId(feedback,p,v,a,refsys);
    IF a.x >= 100.0 THEN
      jerk := -10;
    END_IF;
    IF a.x <= 0.0 THEN
```

```
      state := 4;
    END_IF;
  4:
    mover.StopExternalSetpointGeneration(0);
    state := 5;

END_CASE
```

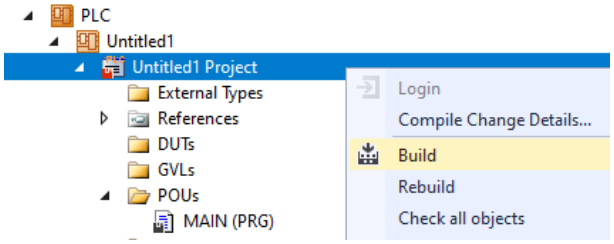
Befehl abschicken

4. Um die Kommandos zu senden, müssen Sie nach dem END_CASE die Update-Methode des Movers triggern:

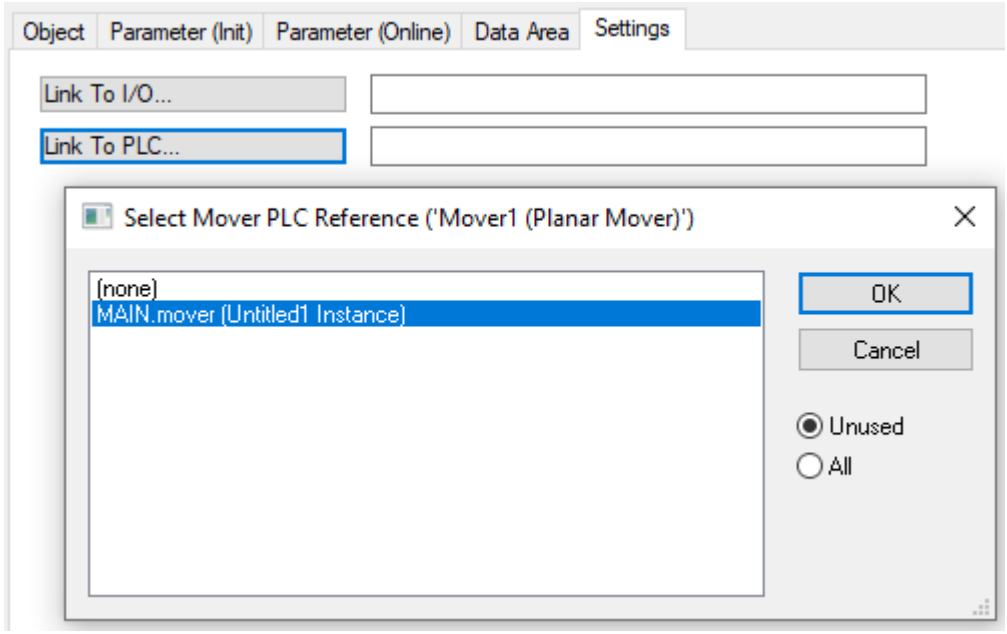
```
mover.Update();
```

Durch das Bauen der PLC wird ein Symbol des „PLC-Movers" erzeugt, welches anschließend mit der Mover-Instanz im MC-Projekt verknüpft werden kann.

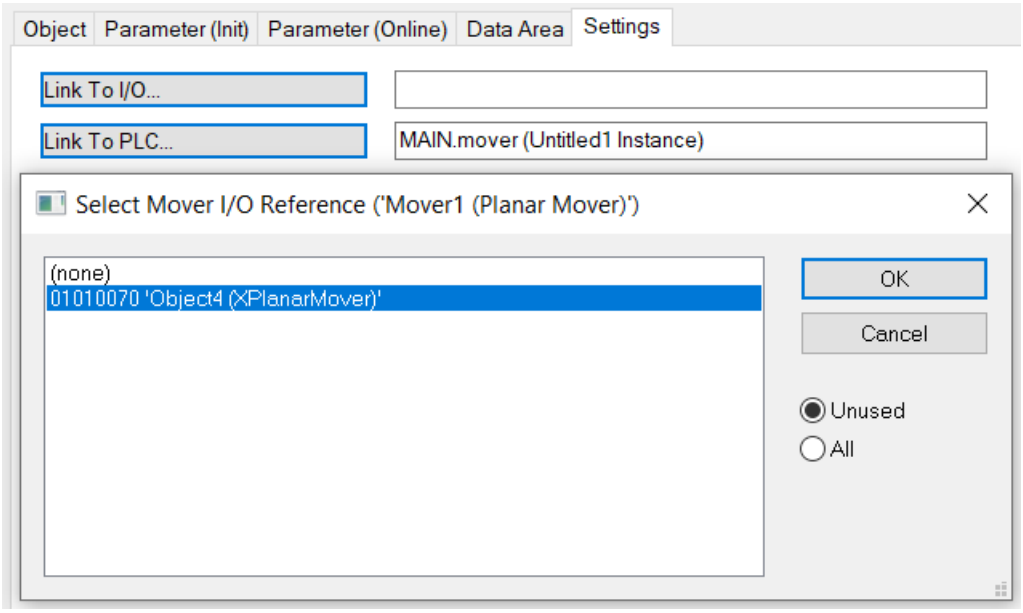
1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**



⇨ Anschließend kann der Planar-Mover im „MC Project“ (Doppelklick) mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.



⇨ Zusätzlich muss der Planar-Mover im „MC Project“ (Doppelklick) mit dem Button **Link To I/O…** im Reiter **Settings** verknüpft werden.



Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Der Mover steht am Ende des Zustandsautomaten (state=5) auf der gewünschten positiven x-Position. Er hat den ersten Part nicht verlassen, daher könnte man in diesem Fall als refsys auch die Objekt-ID des ersten Parts oder des Koordinatensystems angeben. Falls die Objekt-ID des ersten Parts angegeben wird und die Grenze zum zweiten Part überschritten wird, müsste die Objekt-ID des zweiten Parts genutzt werden und die x-Koordinate um 240 reduziert werden (gleichzeitig!). Die Objekt-ID des globalen Koordinatensystems funktioniert, egal auf welchem der Parts man sich befindet. Da die Konfiguration statisch ist, wird Null als Alternative für die globale Koordinatensystem-ID akzeptiert.

Expression	Type	Value	Prepared value	Address	Comm
mover	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC...			%Q*	Mover d
MCTOPLC	CDT_MCTOPLC...			%I*	Mover d
STD	REFERENCE TO...			%IB*	Mover s
SET	REFERENCE TO...			%IB*	Mover s
SetPos	MoverVector				Current
x	LREAL	223.2495384...			X coordi
y	LREAL	120.0356967...			Y coordi
z	LREAL	1.993983856...			Z coordi
a	LREAL	-0.00807865...			A coordi
b	LREAL	-0.00036782...			B coordi
c	LREAL	0.010318374...			C coordi
SetVelo	MoverVector				Current
SetAcc	MoverVector				Current
DcTimeStamp	ULINT	7238109826...			Current
PhysicalAreaID	UDINT	16842848			Current
ACT	REFERENCE TO...			%IB*	Mover a
COORDMODE	REFERENCE TO...			%IB*	Mover c
SETONTRACK	REFERENCE TO...			%IB*	Mover b
Error	BOOL	FALSE			Flag ind
ErrorId	UDINT	0			Error id
state	UDINT	5			
p	MoverVector				
v	MoverVector				
a	MoverVector				
deltat	LREAL	0.01			

Bemerkungen zum Feedback:

Das Feedback wurde in diesem Beispiel nur angelegt und in den SetExternalSetpointRefSys Aufruf gegeben, ohne damit etwas zu tun. Da in diesem Beispiel keine Fehler auftreten, ist das Feedback ab state 2 busy. Ansonsten sollte man IMMER Fehlerbehandlung mit diesem Feedback implementieren, um Fehler wie z.B. eine falsche RefSysId für die Position zu behandeln.

Die Besonderheit dieses Feedbacks ist, dass es zyklisch übergeben wird und man in jedem Aufruf dasselbe Feedback übergeben muss. Treten in der Externen Sollwertgenerierung Fehler auf, dann werden diese Fehler im Feedback angezeigt, nach dem nächsten Aufruf von SetExternalSetpointRefSys mit Feedback. Ein Update Aufruf für das Feedback ist hier nicht nötig. Außerdem wird das Feedback nicht done, wenn StopExternalSetpointGeneration aufgerufen wird, oder aborted, wenn Halt aufgerufen wird.

Auch wenn die Informationen im Feedback nach dem SetExternalSetpointRefSys Aufruf zur Verfügung stehen, müssen die Fehler nicht direkt aus diesem Aufruft stammen, sondern können auch aus vorherigen Aufrufen resultieren.

6.1.8 Beispiel „Planar-Mover in CRotationFreeMovement-Modus verfahren“

Anhand dieser kurzen Anleitung werden Sie ein TwinCAT-Projekt anlegen, das einen Planar-Mover enthält und auf einfache Weise verfährt.

Planar-Mover anlegen

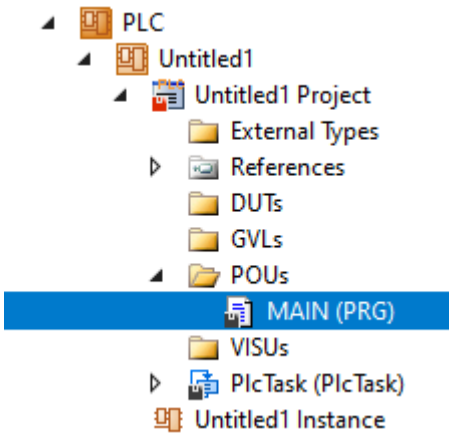
✓ Siehe Konfiguration [▶ 18].

1. Legen Sie einen Planar-Mover an.
2. Versetzen Sie „Parameter (Init)“ in den Simulationsmodus (`TRUE`). Der Parameter ist versteckt und wird nur sichbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** den oder die Mover („MC_PlanarMover [▶ 145]“) wie folgt an.



⇨ Diese/r repräsentiert den/die Mover in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, einen Planar-Mover, eine Zustandsvariable für eine Zustandsmaschine und eine Zielposition für einen Fahrbefehl des Movers an.

```
PROGRAM MAIN
VAR
    mover : MC_PlanarMover;
    state : UDINT;
    target_position_c : LREAL;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

⇨ Dieser Programmcode aktiviert den Mover und dreht ihn auf die Position c=20.

```
CASE state OF
  0:
    mover.Enable(0);
    state := 1;
  1:
    IF mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 2;
    END_IF
  2:
    target_position_c := 20.0;
    mover.MoveC(0, target_position_c, 0, 0);
    state := 3;
END_CASE
```

Befehl abschicken

4. Um den Befehl abzuschicken, müssen Sie den Mover nach dem END_CASE zyklisch mit seiner Update-Methode aufrufen:

```
mover.Update();
```

Durch das Bauen der PLC wird ein Symbol des „PLC-Movers" erzeugt, welches anschließend mit der Mover-Instanz im MC-Projekt verknüpft werden kann.

1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**

⇨ Anschließend kann der Planar-Mover im „MC Project“ (Doppelklick) mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.

Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Der Mover steht am Ende des Zustandsautomaten (state=3) auf der gewünschten (verdrehten) Position und ist im Befehlsmodus CRotationFreeMovement, da der Winkel > 15° ist. Ein weiteres Verfahren der C-Achse bis z.B. 90° würde den Befehlsmodus nach Beendigung des Kommandos auf Free Movement zurück ändern.

ExampleCRotationFreeMovement.Untitled1.MAIN

Expression	Type	Value	Prepared value	Address	Comment
mover	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC...			%Q*	Mover data that is transferred from the Planar Mover to this function block.
MCTOPLC	CDT_MCTOPLC...			%I*	Mover data that is transferred from the Planar Mover to this function block.
STD	REFERENCE TO...			%IB*	Mover standard data that is transferred from the Planar Mover to this function b
MoverOID	OTCID	16#05110010			Object id of the planar mover.
GroupOID	OTCID	16#00000000			Object id of the planar group the mover is in.
State	MC_PLANAR_S...	Enabled			State of the planar mover, e.g. enabled.
CommandMode	MC_PLANAR_M...	CRotationFre...			Command mode of the planar mover, e.g. onTrack.
Busy	MoverBusy				Busy state of the planar mover.
ErrorCode	HRESULT	16#00000000			Error code of the planar mover.
SET	REFERENCE TO...			%IB*	Mover setpoint data that is transferred from the Planar Mover to this function b
SetPos	MoverVector				Current position.
x	LREAL	0			X coordinate.
y	LREAL	0			Y coordinate.
z	LREAL	0			Z coordinate.
a	LREAL	0			A coordinate.
b	LREAL	0			B coordinate.
c	LREAL	19.99999999...			C coordinate.
SetVelo	MoverVector				Current velocity.
SetAcc	MoverVector				Current acceleration.
DcTimeStamp	ULINT	6915035679...			Current time stamp.
PhysicalAreaID	UDINT	0			Current physical area id.
ACT	REFERENCE TO...			%IB*	Mover actpoint data that is transferred from the Planar Mover to this function b
COORDMODE	REFERENCE TO...			%IB*	Mover coordinate mode information that is transferred from the Planar Mover to t
SETONTRACK	REFERENCE TO...			%IB*	Mover busy information that is transferred from the Planar Mover to this functio
Error	BOOL	FALSE			Flag indicating a Planar Mover error.
ErrorId	UDINT	0			Error id indicating the Planar Mover error type.
state	UDINT	3			
target_position_c	LREAL	20			

6.1.9 Limits und Optionen der Bewegungsbefehle

Der Planar-Mover kann verschiedene Arten von Bewegungsbefehlen ausführen. Bis auf den Sonderfall der externen Sollwertgenerierung sind diese einander ähnlich im Aufbau. Das Folgende gilt für die restlichen Bewegungsbefehle. Der erste Parameter des Methodenaufrufs ist immer das Feedback für das Kommando, das der Nutzer übergibt. Übergibt er eine „0“ so impliziert das, er möchte kein Feedback haben (bzw. nutzen). Die nächsten ein bis zwei Parameter beschreiben das Ziel der Bewegung, sie können nicht komplett weggelassen werden. Der nächste Parameter sind die Dynamiklimits, die während der Bewegung eingehalten werden sollen. Übergibt der Nutzer hier eine „0“, werden die Default Werte genutzt (TCOM Parameter des Movers im MC Project). Der letzte Parameter ist das Optionsobjekt, diese unterscheiden sich je nach Befehl.

Limits

Jeder Bewegungsbefehl wird zeitoptimal ausgeführt. Damit die resultierende Trajektorie stetig ist, müssen die Zeitableitungen der Position beschränkt werden. Die Limits enthalten Maximalwerte für die Geschwindigkeit, positive und negative Beschleunigung, sowie den Ruck. Übersteigen die hier angegeben Werte die maximalen Dynamiklimits des Movers (TCOM Parameter des Movers im MC Project), werden sie entsprechend reduziert, eine Warnung wird ausgegeben und der Befehl wird mit reduzierten Dynamikwerten ausgeführt. Es gibt nur ein Limit bzw. Constraint-Objekt. Dieses wird als Beschränkung der Dynamik tangential zur Bewegungsrichtung des Movers aufgefasst.

Bei der ExternenSollwertgenerierung gibt es nur die Parameter Feedback und Optionen.

Ab Version V3.1.10.30: Die Limits sollten durch Constraints ersetzt werden, siehe Dynamics.

Optionen

Die Optionen unterscheiden sich je nach Befehl:

MoveToPosition/JoinTrack/LeaveTrack: Die einzige Option dieser Befehle ist die „UseOrientation“ Flag. Diese Flag gibt an, ob von der XYC Zielposition auch die C-Koordinate genutzt werden soll oder ob diese nicht mit verfahren werden soll. Im letzten Fall kann die C-Koordinate über „MoveC“ separat verfahren werden.

MoveOnTrack: Die erste Option ist die „Gap“. Dieser Zahlenwert gibt den Abstand zum Vordermann während der Bewegung (und danach bis zum nächsten Bewegungsbefehl auf dem Track) an. Dieser Abstand wird entlang des Tracks gemessen (Differenz der Trackpositionen der beiden Mover). Daher müssen Krümmungen des Tracks berücksichtigt werden, da sie den realen 2D Abstand verringern. Die Gap wird von Mittelpunkt zu Mittelpunkt gerechnet, es muss also die Breite der Mover berücksichtigt werden. Die zweite Option ist „Direction“, die Fahrtrichtung auf dem Track Richtung Ziel. Diese kann die Werte „NonModulo“ (=Absolut), „Positiv“ (=Vorwärts), „ShortestWay“ (=KürzesterWeg) und „Negative“ (=Rückwärts) annehmen. Wenn in der entsprechenden Richtung das Ziel erreichbar ist, dann wird der Befehl ausgeführt.

Ab Version V3.1.10.30: Die dritte Option ist „AdditionalTurns“: die Anzahl der zusätzlichen gefahrenen Runden auf einem „Closed Loop“ Track mit „Direction“ „Positiv“ oder „Negative“. Für andere „Direction“-Fälle muss „AdditionalTurns“ Null sein. Die vierte Option ist die „ModuloTolerance“. Dieser Parameter wird verwendet, um unbeabsichtigte Umdrehungen zu vermeiden, wenn Start- und Zielposition ungefähr gleich sind. Falls die Entfernung zwischen Start- und Zielposition kleiner oder gleich der „ModuloTolerance“ ist, so wird die Zielposition auf kürzestem Weg (wie bei „Direction“ = „Shortest Way“), also auch entgegen der angegebenen „Direction“, angefahren. Für die „Direction“ „NonModulo“ muss die ModuloTolerance gleich Null sein. Für Details siehe Modulo-Positionierung.

Ab Version V3.1.10.30: MoveC: Die erste Option ist „AdditionalTurns“: die Anzahl der zusätzlichen ganzen C-Drehungen bezogen auf den „C coordinate modulus“-Parameter des Movers mit „Direction“ „Positiv“ oder „Negative“. Für andere „Direction“-Fälle muss „AdditionalTurns“ Null sein. Die zweite Option ist „Direction“: die Drehrichtung der C-Coordinate Richtung Ziel. Diese kann die Werte „NonModulo“ (=Absolut), „Positiv“ (=Vorwärts), „ShortestWay“ (=KürzesterWeg) und „Negative“ (=Rückwärts) annehmen. Für Details siehe Modulo-Positionierung.

Ab Version V3.1.10.51: AdoptTrackOrientation: Die erste Option ist „AdditionalTurns“: Die Anzahl der zusätzlichen ganzen C-Drehungen bezogen auf den „C coordinate modulus“-Parameter des Movers mit „Direction“, „Positiv“ oder „Negative“. Für andere „Direction“-Fälle muss „AdditionalTurns“ null sein. Die zweite Option ist „Direction“: Die Drehrichtung der C-Coordinate Richtung Ziel. Diese kann die Werte „NonModulo“ (= Absolut), „Positiv“ (= Vorwärts), „ShortestWay“ (= Kürzester Weg) und „Negative“ (= Rückwärts) annehmen. Für Details siehe Modulo-Positionierung.

Ab Version V3.1.10.44: GearInPosOnTrack: Die erste Option ist die „Gap“, die hier dieselbe Interpretation hat wie beim MoveOnTrack. Der zweite Parameter ist die „InSyncToleranceDistance“. Er gibt an, wie weit sich Master und Slave voneinander entfernen dürfen, bevor der ausführende Planar-Mover seine Synchronizität verliert. Die folgenden zwei Optionen sind „Direction“ und „ModuloTolerance“, die sich beide auf den Parameter „SlaveSyncPosition“ (als Eingang am Funktionsaufruf) beziehen. Diese Optionen stehen nur zur Verfügung, wenn der Planar-Track, auf welchem der Planar-Mover seine Synchronisationsbewegung durchführt, eine Closed Loop ist. In diesem Fall ist die Interpretation dieser Optionen analog derer beim MoveC, wobei hier der Modulus durch die Länge des Planar-Tracks gegeben ist. Für Details siehe Modulo-Positionierung. Der letzte Parameter „AllowedSlaveSyncDirections“ gibt an, in welche Richtung, also Positive (default), Negative oder Both, der Planar-Mover während der Phase des Aufsynchronisierens fahren darf. Dieser Parameter kann beispielsweise dazu verwendet werden, einen Rückschwinger zu unterbinden, welcher bei „Both“ auftreten würde, um eine schnellstmögliche Synchronisation zu erreichen. Wenn der Planar-Mover synchron ist, oder er es bereits einmal war, und versucht, aktuell die Synchronizität wiederherzustellen, so hat dieser Parameter keinen weiteren Einfluss.

Ab Version V3.1.10.30: GearInPosOnTrackWithMasterMover: Die ersten vier Optionen, „Gap“, „InSyncToleranceDistance“, sowie die beiden Modulo-Optionen für die SlaveSyncPosition, sind in ihrer Bedeutung identisch mit den ersten vier Optionen des GearInPosOnTrack-Befehls. Es folgen zwei Parameter „Direction“ und „ModuloTolerance“ für die MasterSyncPosition, die analog zu den Modulo-Parametern für die SlaveSyncPosition zur Verfügung stehen, wenn der Master-Planar-Mover sich auf einem Track befindet, der eine Closed Loop ist. Die darauf folgende Option „AllowedSlaveSyncDirections“ hat exakt dieselbe Funktion wie beim GearInPosOnTrack-Befehl. Die letzte Option „FollowMover“ sorgt für den Fall, dass sie gesetzt ist, dafür, dass der Slave-Planar-Mover nicht notwendigerweise einen Planar-TrackTrail [▶ 122] bekommen muss, um zu wissen, über welche Planar-Tracks er seine Bewegung durchführen wird. Der Slave-Planar-Mover wird einfach dem Master-Planar-Mover auf seinem Weg durch das Netzwerk folgen. Sollten Master-Planar-Mover und Slave-Planar-Mover auf unterschiedlichen Planar-Tracks stehen, wenn das

Bewegungskommando mit gesetzter „FollowMover“-Option empfangen wird, so wird der Slave versuchen, auf dem kürzesten Weg den Planar-Track zu erreichen, auf dem die MasterSyncPosition kommandiert ist. Es kann zusätzlich zur „FollowMover“-Option ein Planar-TrackTrail für den Slave-Planar-Mover spezifiziert werden. Er dient in diesem Fall dazu, den Pfad zum Planar-Track, auf dem die MasterSyncPosition liegt, zu kommandieren (z. B. falls er vom kürzesten Weg abweichen soll). Sollte er nicht vollständig bis zu diesem Planar-Track reichen, wird der verbleibende Pfad mit dem kürzesten Weg gefüllt. Für den Fall, dass ein Planar-TrackTrail bei gesetzter „FollowMover“-Option spezifiziert ist, ist es außerdem möglich, die SlaveSyncPosition auf einem anderem als dessen initialen Planar-Track anzugeben. Generell gilt: Ab dem Planar-Track, auf welchem sich die MasterSyncPosition befindet, folgt der Slave-Planar-Mover bei gesetzter „FollowMover“-Option dem Master, unabhängig davon, ob ein PlanarTrackTrail-Objekt angegeben wurde.

Ergänzung Version V3.1.10.30 - Option war vorher mit anderem Typ schon vorhanden:
StartExternalSetpointGeneration: Hier hat der Nutzer die Wahl zwischen dem Modus „Absolute“ und „Relative“. Im absoluten Modus folgt der Mover ausschließlich den externen Sollwerten des Nutzers und ist im Befehlsmodus ExternalSetpointGeneration, andernfalls ist der Mover in einem beliebigen anderen Befehlsmodus und addiert die externen Sollwerte des Nutzers als Offset auf seinen aktuellen Sollwert.

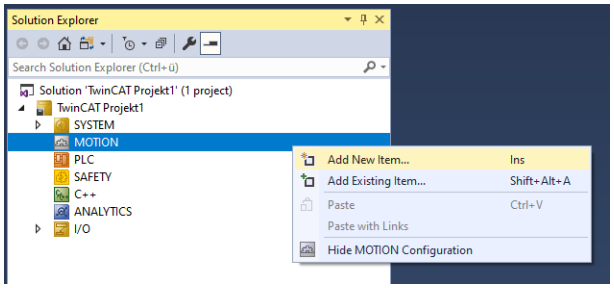
6.2 Planar-Track

Der Planar-Track ist ein Softwareobjekt, das einen (virtuellen) eindimensionalen Pfad auf der zweidimensionalen XPlanar Statorfläche repräsentiert. Auf diesem Pfad können mehrere Planar-Mover aufgereiht und verfahren werden. Kollisionen werden verhindert, indem ein vorgegebener Abstand zwischen den Movern eingehalten wird.

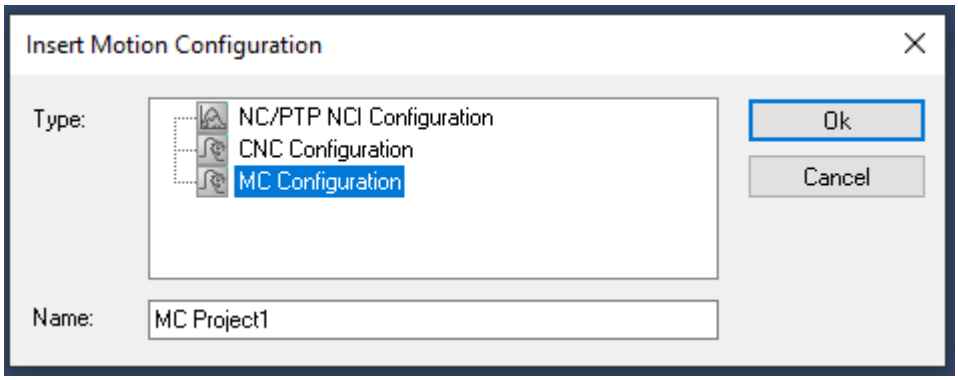
6.2.1 Konfiguration

✓ Um einen Planar-Track anzulegen, muss zunächst eine **MC Configuration** angelegt werden.

1. Wählen Sie **MOTION** > **Add New Item…** aus.

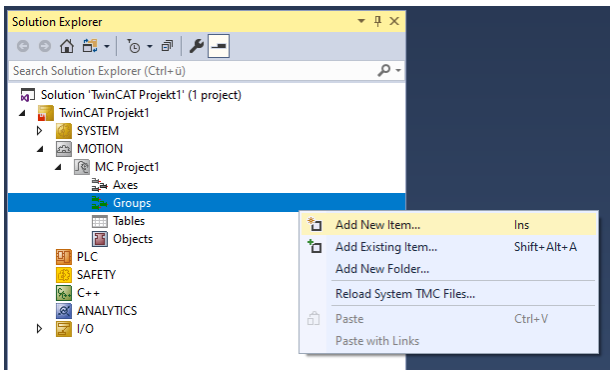


2. Wählen Sie im folgenden Dialogfenster **MC Configuration** aus und bestätigen Sie mit **OK**.

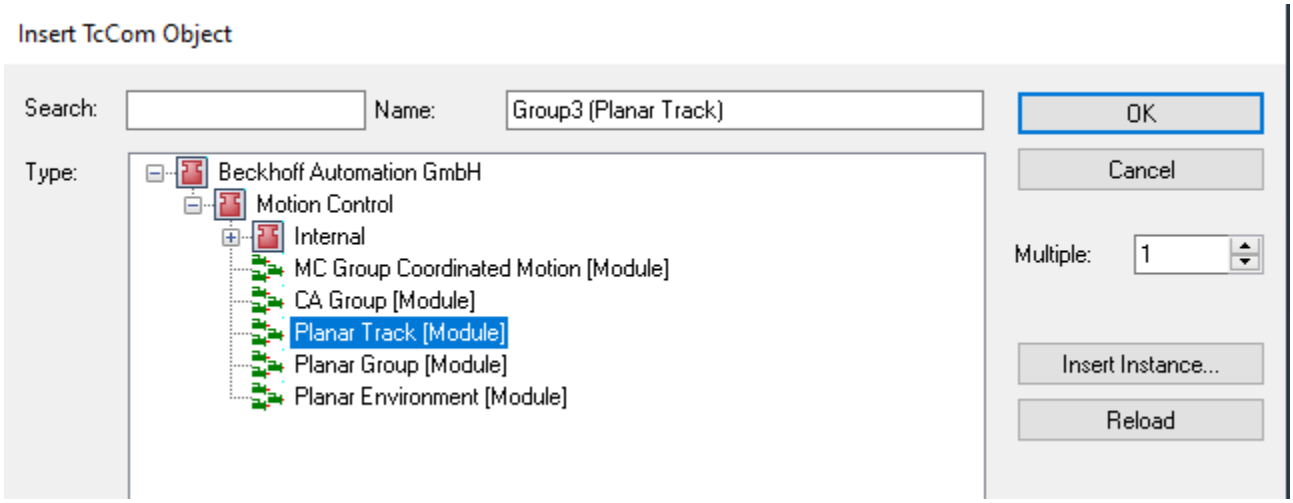


⇨ Sie haben ein MC Project angelegt.

3. Wählen Sie im erzeugten **MC Project** > **Groups** > **Add New Item…** aus.



4. Legen Sie im folgenden Dialogfenster einen (oder mehrere) Planar-Tracks an und bestätigen Sie mit **OK**.



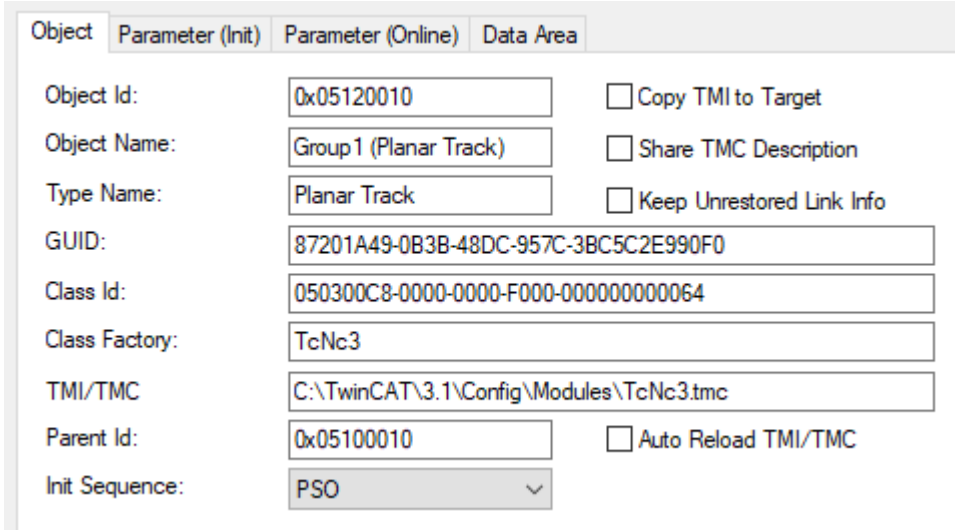
⇨ Der Planar-Track ist nun angelegt und kann parametriert werden.

Detailbeschreibung öffnen

- Planar-Track im Baum auswählen und doppelklicken.

Bedeutung der einzelnen Reiter

Object: Hier werden allgemeine Informationen (Name, Typ, Id, usw.) dargestellt.



Parameter (Init): Gibt Initialparameter an, die der Anwender verändern kann, um das Verhalten des Tracks zu beeinflussen.

Object | Parameter (Init) | Parameter (Online) | Data Area

	Name	Value	CS	Unit	Type	PTCID	Comment
	Maximal mover width	155.0	☐		LREAL	0x050300B2	Maximal width for movers on the track, used for internal collision checks.
	Maximal mover height	155.0	☐		LREAL	0x050300B3	Maximal height for movers on the track, used for internal collision checks.
	Check collisions against static objects	FALSE	☐		BOOL	0x050300B4	If TRUE, collisions are also checked against static objects, i.e. tracks and environment.
	Collision range at start	250.0	☐		LREAL	0x050300BC	Distance of a mover to the start of the track, where it does not interfere anymore with other tr
	Collision range at end	250.0	☐		LREAL	0x050300BD	Distance of a mover to the end of the track, where it does not interfere anymore with other tra
	Collision range mode	Automatic	☐		MC.MC_PLANAR_TRACK_COLLISION_RANGE_MODE	0x050300D1	Either the collision range at start and end is calculated automatically when the track is enable
	PartOID	00000000	☐		OTCID	0x050300E0	Object id of the PlanarPart the track is in.
+	Geometric information		☐	0 (Array Elements)		0x050300D7	Array containing the elements describing the geometric curve of the track.
	Closed loop	FALSE	☐		BOOL	0x05030066	Bool setting the tracks init configuration to closed loop.
+	Starts from tracks	[]	☐	0 (Array Elements)		0x050300D8	Connection data of tracks where this track starts.
+	Ends at tracks	[]	☐	0 (Array Elements)		0x050300D9	Connection data of tracks where this track ends.

Die Initialparameter sind zunächst so eingestellt, dass der Planar-Track (fertig verknüpft) mit Hardware verfahren werden kann. Wenn die Mover auf dem Track größer oder kleiner sind, sollten die beiden Parameter „Maximal mover width/height“ angepasst werden. Der Parameter „Check collision against static objects“ bestimmt, ob für einen Track in einer Planar-Group Kollisionen mit anderen statischen Objekten (Tracks/Rand der Statorfläche) geprüft werden. Der Parameter „Collision range mode“ bestimmt, ob die „Collision range at start/end“ vom Anwender über die entsprechenden Parameter angegeben wird oder vom Track intern automatisch berechnet wird. Die „Collision range“ ist die Entfernung vom Start/Ende des Tracks, ab der ein Planar-Mover für Planar-Mover auf anderen Tracks für die Kollisionsvermeidung berücksichtigt wird.

Ab Version V3.1.10.44: Die Parameter „Geometric information“, „Closed loop“, „Starts from tracks“ und „Ends at tracks” können genutzt werden, um die Geometrie des Tracks und seine Verbindung mit anderen Tracks zu definieren. Dabei wirken die Parameter exakt wie die entsprechenden PLC-Befehle.

Ab Version V3.2.60: Der Parameter „PartOID“ gibt an, welchem Part dieser Track fest zugeordnet ist. Wenn es einen eindeutigen Part gibt oder das Part-Feature nicht genutzt wird (kein Auslesen der Processing Unit durch die Environment), muss der Parameter nicht gesetzt werden. Andernfalls muss der Parameter gesetzt werden, damit der Track aufstartet. Alle Positionen in den Init- und Online-Parametern (Geometric [online] information) werden in diesem Part-System angegeben. Die Parameter „Starts from tracks“ und „Ends at tracks” sind erweitert worden, um die zusätzliche Funktionalität der entsprechenden PLC-Befehle für das Part-Feature abzubilden.

Parameter (Online): Zeigt den Zustand des Tracks zur Laufzeit, z. B. die Anzahl der Planar-Mover oder die -Länge.

Object | Parameter (Init) | Parameter (Online) | Data Area

	Name	Online	CS	Unit	Type	PTCID	Comment
	Track length		☐		LREAL	0x05030...	Length of the track (read only).
	GroupOID		☐		OTCID	0x05030...	Object id of the PlanarGroup the track is in, read only.
	State		☐		MC.MC_PLANAR_STATE	0x05030...	State, read only.
	Operation mode		☐		MC.MC_PLANAR_TRACK_OPERATION_MODE	0x05030...	Track state, read only.
	Mover count on track		☐		UDINT	0x05030...	Number of movers that are on this track, read only.
	Moving mover count		☐		UDINT	0x05030...	Number of movers that have requested a movement on the track, read only.
+	Geometric online information		☐	0 (Array Elements)		0x05030...	Array containing the elements describing the geometric curve of the track.
+	Previous tracks		☐	0 (Array Elements)		0x05030...	Array containing the object IDs of all tracks that end at this track's start vertex, read only.
+	Subsequent tracks		☐	0 (Array Elements)		0x05030...	Array containing the object IDs of all tracks that start at this track's end vertex, read only.
	Closed loop online information		☐		BOOL	0x05030...	Bool indicating if the tracks online configuration is a closed loop.

Ab Version V3.1.10.30: Die Parameter „Previous Tracks“ und „Subsequent Tracks” sind Arrays, die die OIDs aller direkt vor bzw. direkt hinter diesem Track liegenden Tracks enthalten.

Ab Version V3.1.10.44: Der Parameter „Geometric online information“ zeigt die zur Laufzeit vorhandene Geometrie des Tracks an. Diese resultiert aus dem entsprechenden Initialparameter und/oder den genutzten PLC-Befehlen.

Ab Version V3.2.1: Der Parameter „Closed loop online information“ gibt an, ob der Track eine geschlossene Schleife bildet (einen Kreis).

Data Area: Zeigt den Speicherbereich, über den der Track mit dem PLC-Track kommuniziert.

Object | Parameter (Init) | Parameter (Online) | Data Area

	Area No	Name	Type	Size	CS	CD / Elements
+	1 (0)	McToPlc	OutputSrc	20	☐	☐ 1 Symbols

6.2.2 Tracknetzwerke und Kollisionsvermeidung

Tracks und Tracknetzwerke

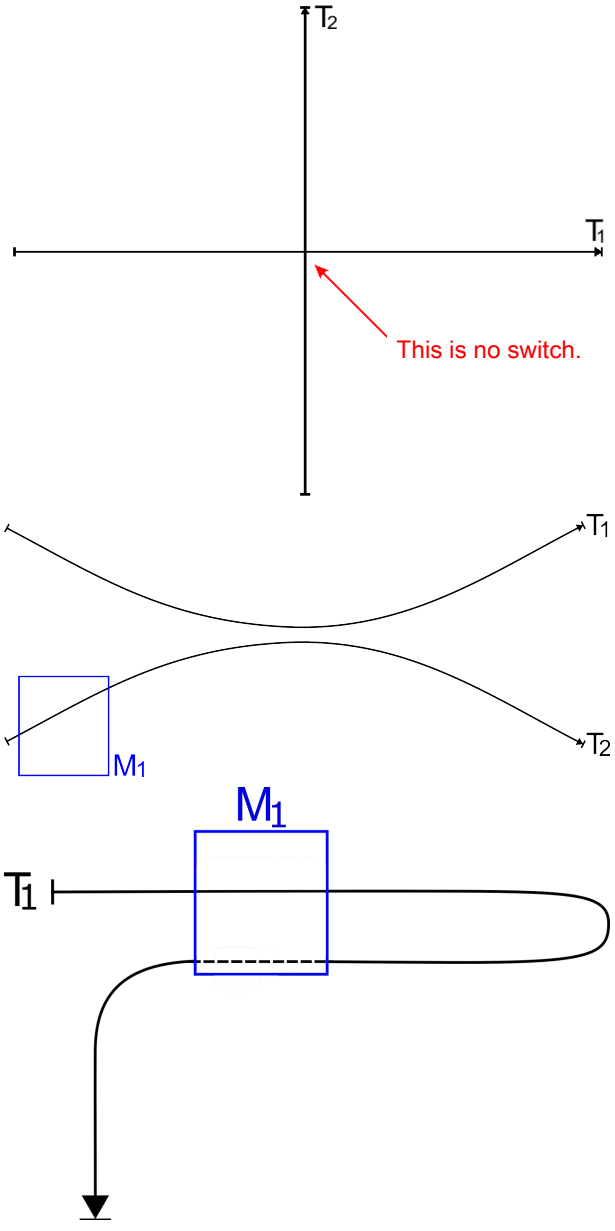
Tracks sind vom Nutzer vorgegebene statische Pfade auf der Statorfläche. Mehrere Tracks können stetig (inklusive Richtung und Krümmung) in einem Punkt verbunden werden, damit Mover von einem Track auf einen anderen wechseln können. Wenn mehr als zwei Tracks an einem Punkt derart verbunden sind, so entsteht dort eine Weiche. Dadurch kann ein Netzwerk von zusammenhängenden Tracks erstellt werden.

Auf einem einzelnen Track kann ein Mover sowohl vorwärts als auch rückwärts fahren. Ein Übergang zu einem anderen Track kann nur von einem Trackende zu einem Trackanfang vollzogen werden, nicht andersherum.

Kollisionsvermeidung in einem Tracknetzwerk

Mover, die sich in einem Tracknetzwerk bewegen, vermeiden Kollisionen mit anderen Movern im selben Tracknetzwerk. Ausgenommen hiervon sind Stellen, an denen sich Tracks ohne eine Weiche kreuzen oder zu nahe aneinander oder an sich selbst vorbeiführen (siehe Abbildungen). Derartige Konfigurationen sollten vermieden werden.

Negativbeispiele:



Jeder Mover hat einen Mindestabstand („Gap“) eingestellt, den er auf seinem Pfad zum Mover vor sich einhalten muss. Diese Gap wird gemessen zwischen den Positionen der Mover auf dem Track und kann mit jedem Verfahrbefehl neu gesetzt werden.

In der Nähe einer Weiche muss ein Mover gegebenfalls zusätzlich auf potentielle Kollisionen mit Movern achten, die sich auf anderen mit der Weiche verbundenen Tracks befinden, selbst wenn diese Tracks nicht Teil des geplanten Pfades des Movers sind. Ob diese zusätzliche Kollisionsvermeidung für einen Mover zu einem Zeitpunkt aktiv ist, hängt von vier Faktoren ab:

- der aktuellen Position des Movers,
- der frühest möglichen Ruheposition des Movers (ergibt sich aus der aktuellen Dynamik und den Dynamiklimits),
- der eingestellten Gap des Movers,
- dem entsprechenden Collision Range Parameter des aktuellen Tracks.

Wenn die Strecke zwischen aktueller Position und frühestmöglicher Ruheposition des Movers dabei an irgendeinem Punkt weniger als Gap oder Collision Range von der Weiche entfernt ist, ist die zusätzliche Kollisionsvermeidung für diesen Mover aktiv. Wenn dies der Fall ist, werden alle anderen Mover, für die diese Bedingung auch erfüllt ist, in der Dynamikplanung berücksichtigt.

Definition der Collision Ranges

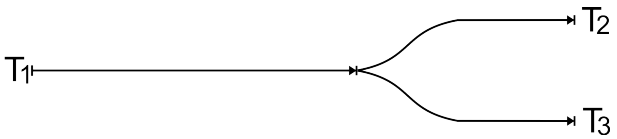
Die Bedeutung der Collision Range Parameter für die Kollisionsvermeidung wurde im vorigen Abschnitt beschrieben. Dabei bezieht sich „Collision Range at start“ auf den Abstand zur Weiche am Startpunkt des Tracks und „Collision Range at end“ auf den Abstand zur Weiche am Endpunkt des Tracks.

Ein intuitiveres Verständnis für die Collision Range Parameter ergibt sich aus folgender Empfehlung: Die Collision Range sollte so eingestellt sein, dass ein Mover, der sich in diesem Abstand zur zugehörigen Weiche (am Start oder Ende des Tracks) befindet, nicht mit Movern auf anderen Tracks, die an die Weiche anschließen, kollidieren kann.

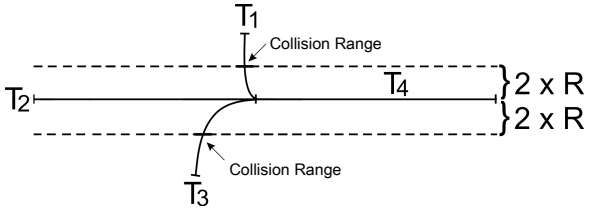
Um die Konfiguration zu vereinfachen, werden die entsprechenden Werte für die Collision Ranges automatisch berechnet und übernommen, wenn der Parameter „Collision range mode“ auf „Automatic“ eingestellt ist. Wird statt „Automatic“ „Manual“ gewählt, werden stattdessen die vom Nutzer eingetragenen Werte verwendet. Sollten diese zu kleine gewählt sein, kann dies unter Umständen Kollisionen zur Folge haben. Sind sie hingegen deutlich zu groß gewählt, blockieren sich möglicherweise Mover auf verschiedenen Tracks, die tatsächlich weit voneinander entfernt sind und gar nicht kollidieren können.

Wenn ein Track am Startpunkt (Endpunkt) entweder keine Weiche hat, oder an der Weiche keine anderen Tracks starten (enden), kann die entsprechende Collision Range auf 0 gesetzt werden.

Beispiele und Bilder:

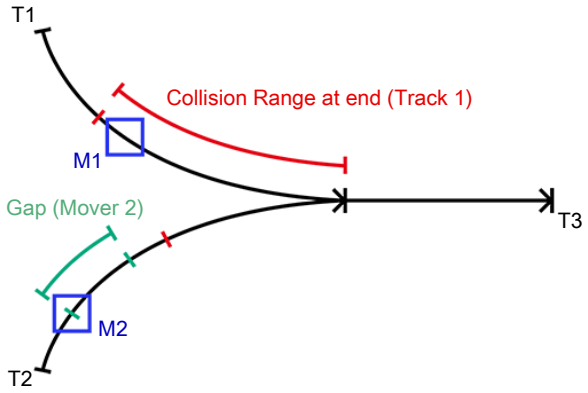


In diesem Beispiel kann die „Collision range at end“ für Track 1 auf Null gesetzt werden, da an der Weiche zwar zwei andere Tracks starten, jedoch keine anderen Tracks enden. Der Parameter „Collision range at start“ für die Tracks 2 und 3 sollte so gewählt sein, dass ein Mover mit diesem Abstand zur Weiche nicht mit Movern auf dem jeweils anderen Track kollidieren kann.



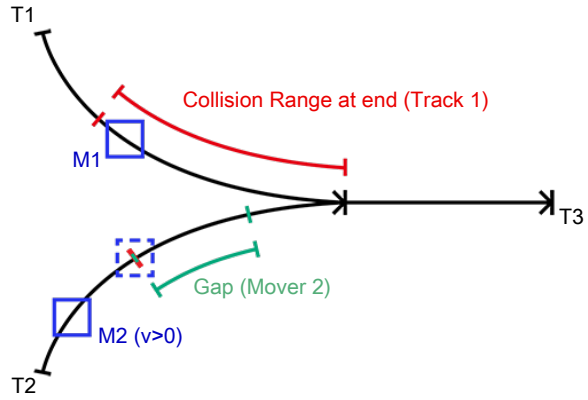
Beispiel für die Ermittlung sinnvoller Collision Range Parameter (T1, T2 und T3 enden am Start von T4): Wenn R der maximale Moverradius von Movern auf dem Track ist, kann ein „Schlauch“ mit Radius 2*R um einen Track gelegt werden (hier um Track 2), um ein Minimum für die Collision Ranges auf den anderen

Tracks zu bestimmen. In diesem Beispiel hat Track 1 eine kleinere „Collision range at end“, da er sich schnell von den anderen Tracks entfernt und Track 3 und Track 2 haben eine größere „Collision range at end“, da sie länger eng zusammen verlaufen.

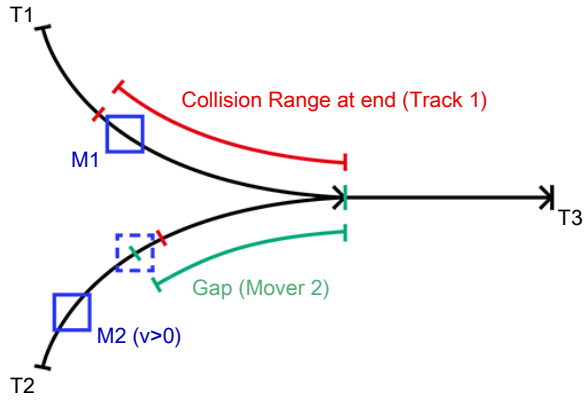


In diesem Beispiel ist für Mover 1 die zusätzliche Kollisionsvermeidung an der Weiche aktiv, da schon allein sein Abstand zur Weiche geringer ist als die eingestellte Collision range.

Mover 2 steht in diesem Beispiel still und ist weiter von der Weiche entfernt als Gap oder Collision Range. Die zusätzliche Kollisionsvermeidung ist daher nicht aktiv und die beiden Mover müssen zu diesem Zeitpunkt keine Rücksicht aufeinander nehmen.

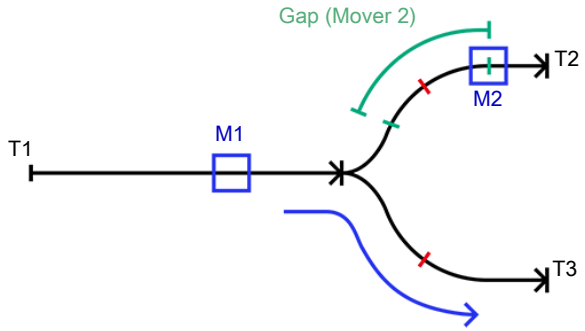


In diesem Beispiel steht Mover 1 auf Track 1 interhalb der Collision Range, daher blockiert Mover 1 die Bewegung von Mover 2 nach Track 3. Mover 2 stoppt genau am Anfang der Collision Range von Track 2, da dies der letzte sichere Haltepunk ist. Würde die Gap von Mover 1 und Mover 2 es erlauben, würde Mover 2 einen entsprechend späteren Halt anfahren.

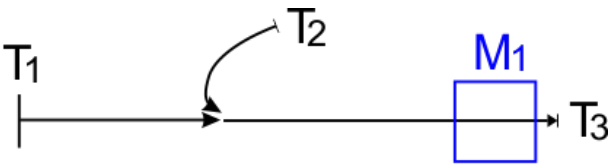


In diesem Beispiel steht Mover 1 auf Track 1 interhalb der Collision Range, daher blockiert Mover 1 die Bewegung von Mover 2 nach Track 3. Mover 2 stoppt genau mit Abstand seiner Gap zum Ende von Track 2. Würde die Gap von Mover 1 und Mover 2 es erlauben, würde Mover 2 einen entsprechend späteren Halt anfahren.

In den letzten beiden Beispielen fährt Mover 2 jeweils weiter, wenn Mover 1 sich so weit nach vorne bewegt hat, dass beide Mover einen Mindestabstand haben, der größer als die Gap beider Mover ist.



In diesem Beispiel ist Mover 2 weiter als Gap oder Collision Range von der Weiche entfernt, daher kann Mover 1 ungehindert auf Track 3 fahren. Sollte sich Mover 2 zurück bewegen, ergibt sich eventuell eine Blockierung, wenn der Abstand zur Weiche kleiner als Gap oder Collision Range ist.



Dies ist ein Beispiel für eine zu vermeidende Konstruktion, bei der das Ende eines Tracks (hier T2) die Collision Range at *start* eines anderen Tracks (T3) beeinflusst (und umgekehrt). Eine solche Situation wird im Falle vom *Automatic Collision Range Mode* nicht erkannt. Ist sie dennoch gewünscht, ist hier eine manuelle Einstellung der Collision Ranges nötig. Von Tracks mit derart engen Kurven, wie die des T2 in diesem Beispiel, ist jedoch auch aufgrund der starken Limitierung der Dynamik (enge Kurven erzeugen auch beim Durchfahren mit niedrigen Geschwindigkeiten große Fliehkräfte) dringend abzuraten.

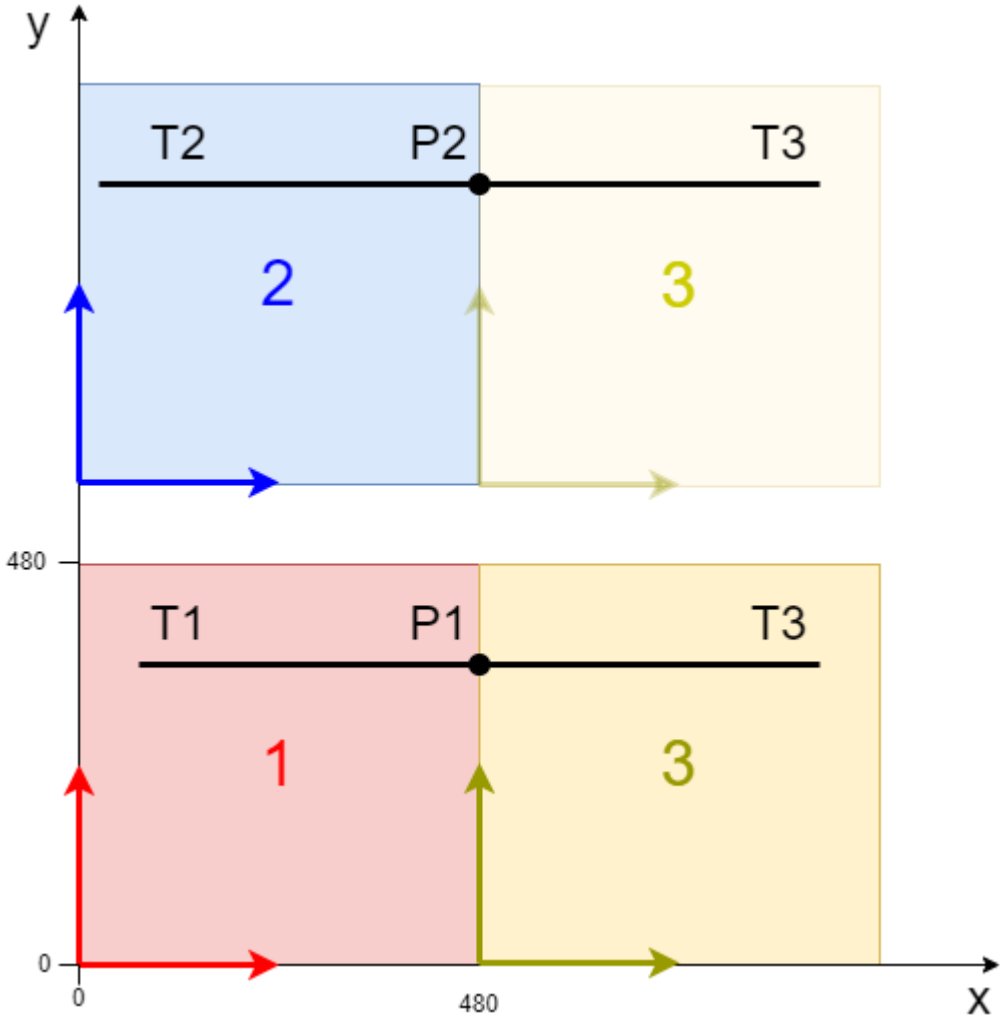
6.2.3 Tracks und Parts

Ab Version V3.2.60: Das Part Feature, um welches es in diesem Abschnitt geht, steht zur Verfügung.

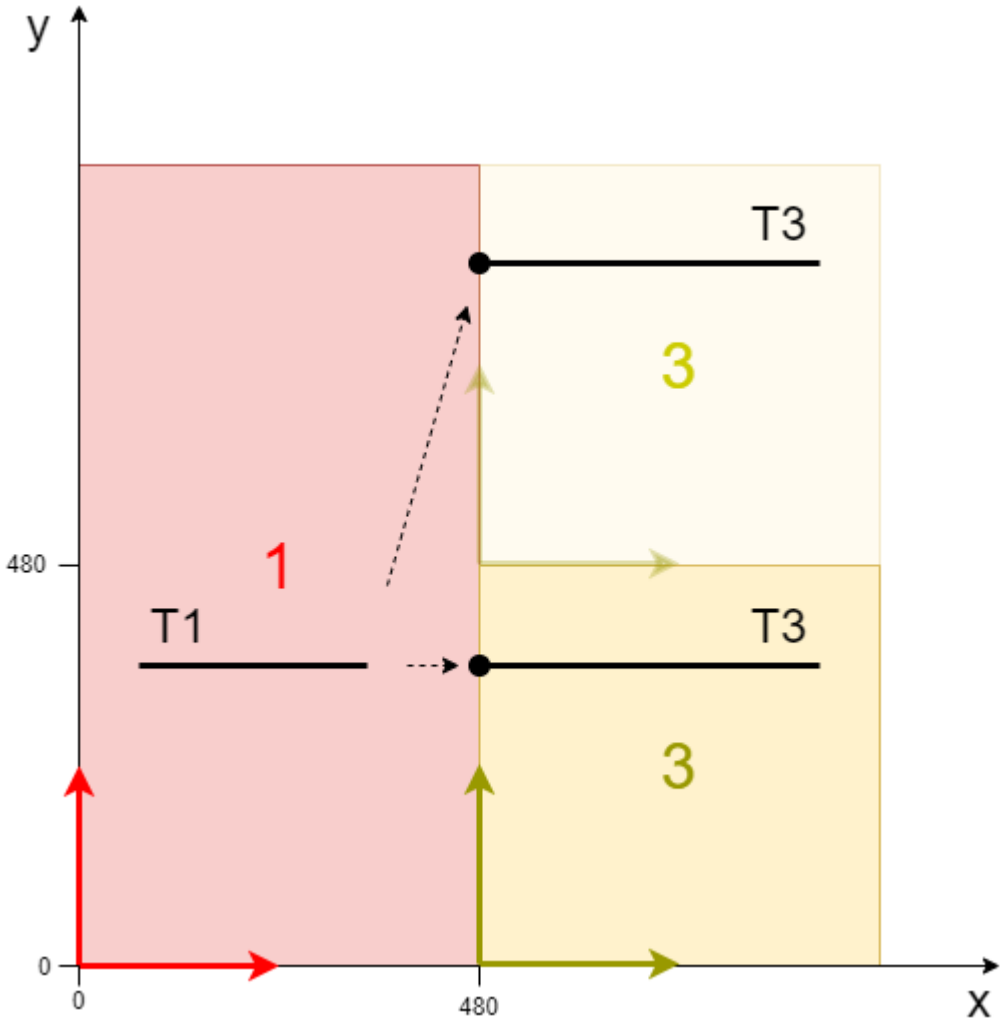
Tracks können zusammen mit Parts genutzt werden, allerdings sind einige Besonderheiten zu beachten:

- Ein Track ist immer fest einem einzigen Part zugeordnet. Das geschieht über den Initialparameter „PartOID“.
- Der Track muss sich geometrisch vollständig auf dem entsprechenden Part befinden.
- Der Track hat relativ zu seinem Part eine feste statische Geometrie. Ändert sich also die Position seines Parts, dann ändert sich entsprechend auch die Position des Tracks.
- Wenn mehr als ein Part in der Konfiguration existiert, muss der Initialparameter „PartOID“ angegeben werden, sonst wird (jeder) Track automatisch dem einzigen Part zugeordnet.
- Tracks auf dem gleichen Part können einfach verbunden werden. Jeder Track kann dabei mit seinem Start und seinem Ende nur jeweils einmal an einen anderen Tack angeschlossen werden. Das heißt, StartFromTrack [▶ 167] und EndAtTrack [▶ 168] können je Track maximal einmal erfolgreich aufgerufen werden.
- Tracks auf unterschiedlichen Parts können nur verbunden werden, wenn die Verbindung zwischen ihnen genau auf der Grenze beider Parts liegt. Das heißt, der Start oder das Ende einer der beiden Tracks muss bereits auf der Partgrenze liegen, damit der andere angeschlossen werden kann. Diese Verbindung kann nur überfahren werden, wenn beide Parts an dieser Stelle liegen. Wird einer der Parts in eine andere Position gebracht, kann die Verbindung nicht mehr überfahren werden (wie jedes offene Ende eines Tracks). Beide Tracks können jedoch in diesem Fall mit anderen Tracks auf anderen Parts in anderen Positionen verbunden sein.

Um all diese unterschiedlichen Verbindungen zwischen Tracks zu schließen, können die Methoden StartFromTrack [▶ 167] und EndAtTrack [▶ 168] je Track mehr als einmal aufgerufen werden.



In diesem Beispiel sind Part 1 und 2 statisch. Part 3 hat 2 Positionen, sodass er einmal mit Part 1 und einmal mit Part 2 in Verbindung steht. Nachdem die Tracks 1 und 2 auf Part 1 und 2 definiert sind, kann Track 3 auf Part 3 mit beiden verbunden werden. Je nach Richtung der Tracks muss entweder Track 3 mit zwei StartFromTrack [▶ 167]-Aufrufen oder mit zwei EndAtTrack [▶ 168]-Aufrufen mit den Positionen 1 und 2 (bzw. den Tracks 1 und 2) verbunden werden. Dabei legt der jeweils erste Aufruf die Geometrie und eine logische Verbindung des Startes oder Endes von Track 3 fest, während der zweite Aufruf nur noch eine logische Verbindung festlegt und lediglich voraussetzt (und überprüft), dass die geometrische Verbindung passend ist.



In diesem Beispiel ist Part 1 statisch und Part 3 hat 2 Positionen, sodass er Part 1 an verschiedenen Stellen berührt. Nachdem Track 3 definiert ist, ist es unklar, wie ein EndAtTrack [▶ 168] von Track 1 zu Track 3 zu interpretieren ist. Soll Track 1 geometrisch mit Track 3 in der oberen oder unteren Konfiguration verbunden werden? Das kann mit den neuen Methoden EndAtTrackAdvanced [▶ 169] und StartFromTrackAdvanced [▶ 169] realisiert werden. Dabei wird genau angegeben, in welcher Position die beiden Parts der beteiligten Tracks liegen, um die Tracks in dieser Konfiguration zu verbinden.

6.2.4 Beispiel „Planar-Mover auf Track einkoppeln und verfahren“

Anhand dieser Anleitung werden Sie ein TwinCAT-Projekt anlegen, das zwei Planar-Mover sowie einen Planar-Track enthält. Es werden beide Mover auf dem Track einkoppelt und verfahren.

Planar-Mover anlegen

✓ Siehe Konfiguration [▶ 18].

1. Legen Sie zwei Planar-Mover an.
2. Versetzen Sie „Parameter (Init)“ in den Simulationsmodus (`TRUE`). Der Parameter ist versteckt und wird nur sichbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.
3. Ändern Sie die Startposition des zweiten Movers auf x=240.

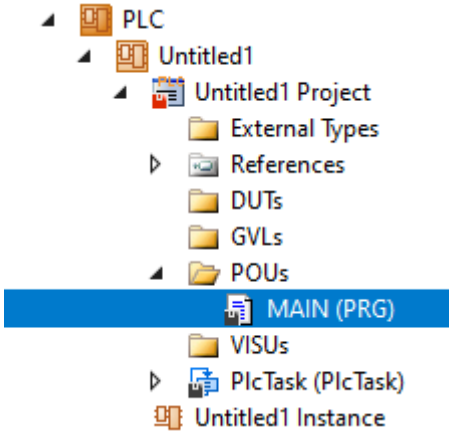
Planar-Track anlegen

4. Fügen Sie den Planar-Track über **Groups > Add New Item…** hinzu, siehe Konfiguration [▶ 43].

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** die gewünschte Anzahl an Movern („MC_PlanarMover“) und Tracks („MC_PlanarTrack“) an.



   ⇨ Diese repräsentieren Mover und Tracks in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, zwei Planar-Mover, einen Planar-Track, eine Zustandsvariable für eine Zustandsmaschine sowie zwei Hilfspositionen für den Track an.

```
PROGRAM MAIN
VAR
    mover_one, mover_two : MC_PlanarMover;
    track : MC_PlanarTrack;
    state : UDINT;
    pos1, pos2 : PositionXYC;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

   ⇨ Dieser Programmcode erstellt und aktiviert einen Track und beide Mover. Danach werden beide Mover auf dem Track eingekoppelt und verfahren.

```
CASE state OF
  0:
    pos1.SetValuesXY(0, 0);
    pos2.SetValuesXY(400, 0);
    track.AppendLine(0, pos1, pos2);
    track.Enable(0);
    state := 1;
  1:
    IF track.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 2;
    END_IF
  2:
    mover_one.Enable(0);
    mover_two.Enable(0);
    state := 3;
  3:
    IF mover_one.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled
    AND mover_two.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 4;
    END_IF
  4:
    mover_one.JoinTrack(0, track, 0, 0);
    mover_two.JoinTrack(0, track, 0, 0);
    state := 5;
  5:
    IF mover_one.MCTOPLC.STD.CommandMode = MC_PLANAR_MOVER_COMMAND_MODE.OnTrack
    AND mover_two.MCTOPLC.STD.CommandMode=MC_PLANAR_MOVER_COMMAND_MODE.OnTrack THEN
      state := 6;
    END_IF
  6:
    mover_one.MoveOnTrack(0, 0, 150.0, 0, 0);
    mover_two.MoveOnTrack(0, 0, 350.0, 0, 0);
    state := 7;
  7:
    IF mover_one.MCTOPLC.SETONTRACK.SetPos >= 149.9
    AND mover_two.MCTOPLC.SETONTRACK.SetPos >= 349.9 THEN
      state := 8;
    END_IF

END_CASE
```

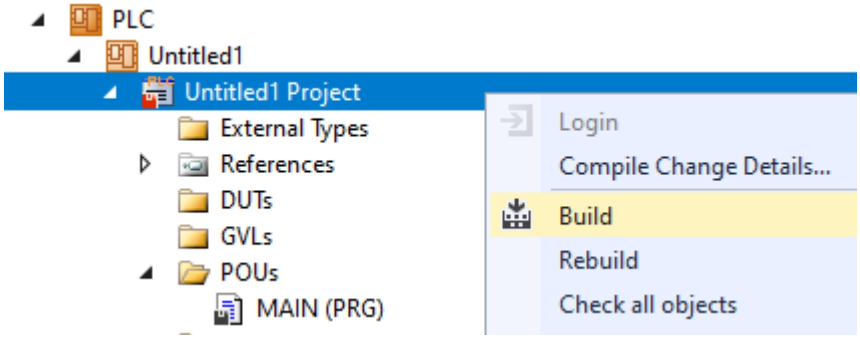
Befehl abschicken

4. Um den Befehl abzuschicken, müssen Sie die Mover und den Track nach dem END_CASE zyklisch mit ihrer Update-Methode aufrufen:

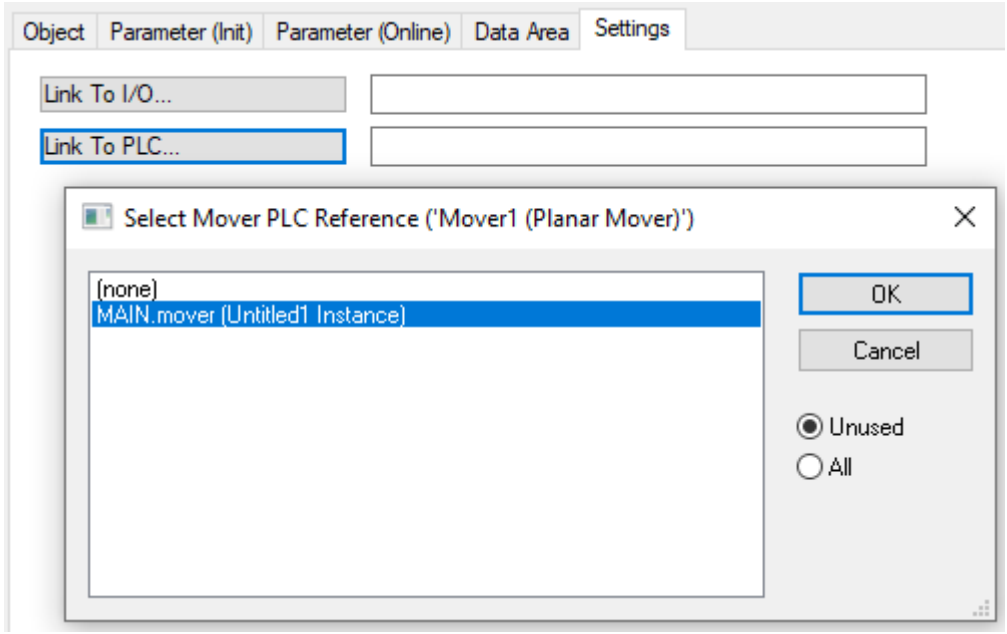
```
mover_one.Update();
mover_two.Update();
track.Update();
```

Durch das Bauen der PLC werden Symbole des „PLC-Movers" und „-Tracks“ erzeugt, welche anschließend mit der Mover- bzw. Track-Instanz im MC-Projekt verknüpft werden können.

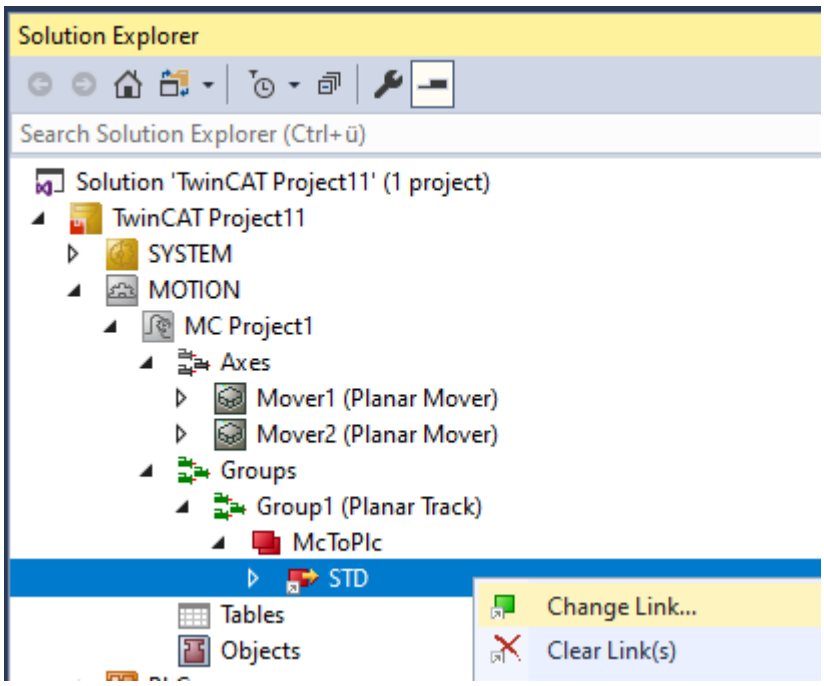
1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**

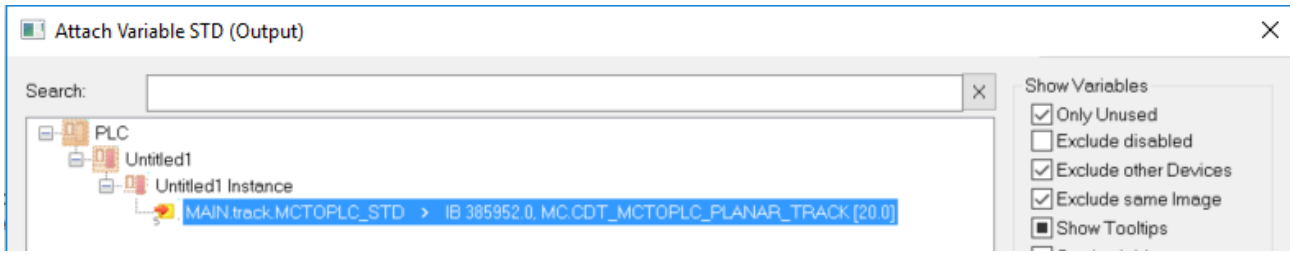


⇨ Anschließend können die Planar-Mover im „MC Project“ mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.



⇨ Der Track muss separat über die folgenden Dialogfenster verlinkt werden.





Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Die Mover stehen am Ende des Zustandsautomaten (state=8) auf den gewünschten Positionen.

MAIN [Online] TwinCAT Project11

TwinCAT_Project11.Untitled1.MAIN

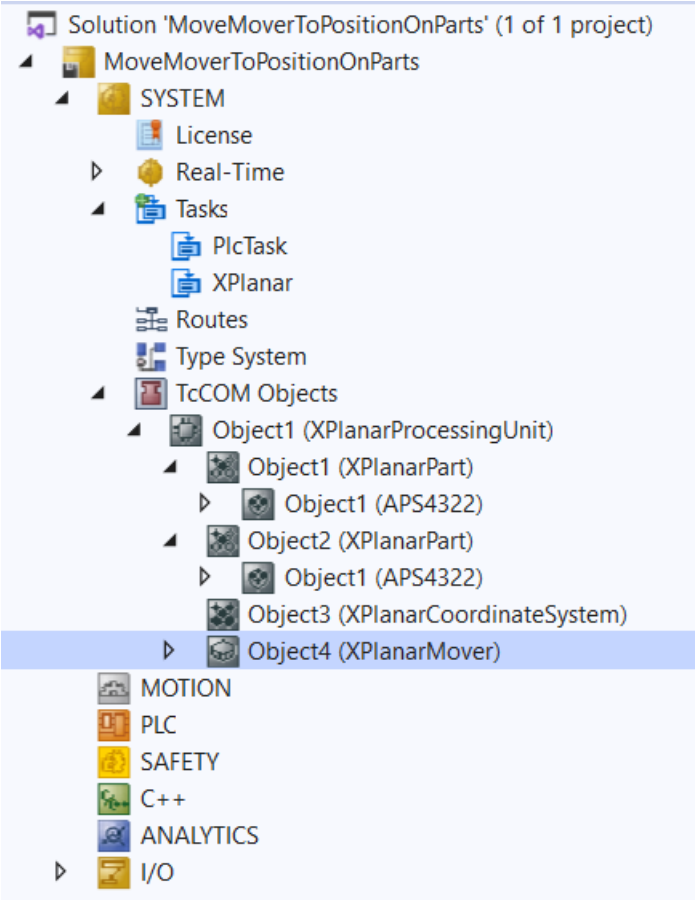
Expression	Type	Value	Prepared value	Address	Comment
mover_one	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC_PLANAR_M...			%Q*	Mover data that is tra...rred from the Plana...
MCTOPLC	CDT_MCTOPLC_PLANAR_M...			%I*	Mover data that is tra...rred from the Plana...
STD	REFERENCE TO CDT_MCTO...			%IB*	Mover standard data t...is transferred from ...
SET	REFERENCE TO CDT_MCTO...			%IB*	Mover setpoint data th...is transferred from t...
SetPos	MoverVector				Current position.
x	LREAL	149.9999999...			X coordinate.
y	LREAL	0			Y coordinate.
z	LREAL	0			Z coordinate.
a	LREAL	0			A coordinate.
b	LREAL	0			B coordinate.
c	LREAL	0			C coordinate.
SetVelo	MoverVector				Current velocity.
SetAcc	MoverVector				Current acceleration.
DcTimeStamp	ULINT	66246393761...			Current time stamp.
PhysicalAreaID	UDINT	0			Current physical area id.
ACT	REFERENCE TO CDT_MCTO...			%IB*	Mover actpoint data th...is transferred from t...
COORDMODE	REFERENCE TO CDT_MCTO...			%IB*	Mover coordinate mod...ormation that is tra...
SETONTRACK	REFERENCE TO CDT_MCTO...			%IB*	Mover busy informatio...at is transferred fro...
Error	BOOL	FALSE			Flag indicating a PlanarMover error.
ErrorId	UDINT	0			Error id indicating the PlanarMover error type.
mover_two	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC_PLANAR_M...			%Q*	Mover data that is tra...rred from the Plana...
MCTOPLC	CDT_MCTOPLC_PLANAR_M...			%I*	Mover data that is tra...rred from the Plana...
STD	REFERENCE TO CDT_MCTO...			%IB*	Mover standard data t...is transferred from ...
SET	REFERENCE TO CDT_MCTO...			%IB*	Mover setpoint data th...is transferred from t...
SetPos	MoverVector				Current position.
x	LREAL	349.9999999...			X coordinate.
y	LREAL	0			Y coordinate.
z	LREAL	0			Z coordinate.
a	LREAL	0			A coordinate.
b	LREAL	0			B coordinate.
c	LREAL	0			C coordinate.
SetVelo	MoverVector				Current velocity.
SetAcc	MoverVector				Current acceleration.
DcTimeStamp	ULINT	66246393761...			Current time stamp.
PhysicalAreaID	UDINT	0			Current physical area id.
ACT	REFERENCE TO CDT_MCTO...			%IB*	Mover actpoint data th...is transferred from t...
COORDMODE	REFERENCE TO CDT_MCTO...			%IB*	Mover coordinate mod...ormation that is tra...
SETONTRACK	REFERENCE TO CDT_MCTO...			%IB*	Mover busy informatio...at is transferred fro...
Error	BOOL	FALSE			Flag indicating a PlanarMover error.

6.2.5 Beispiel „Planar-Mover auf Tracks mit Planar-Parts verfahren"

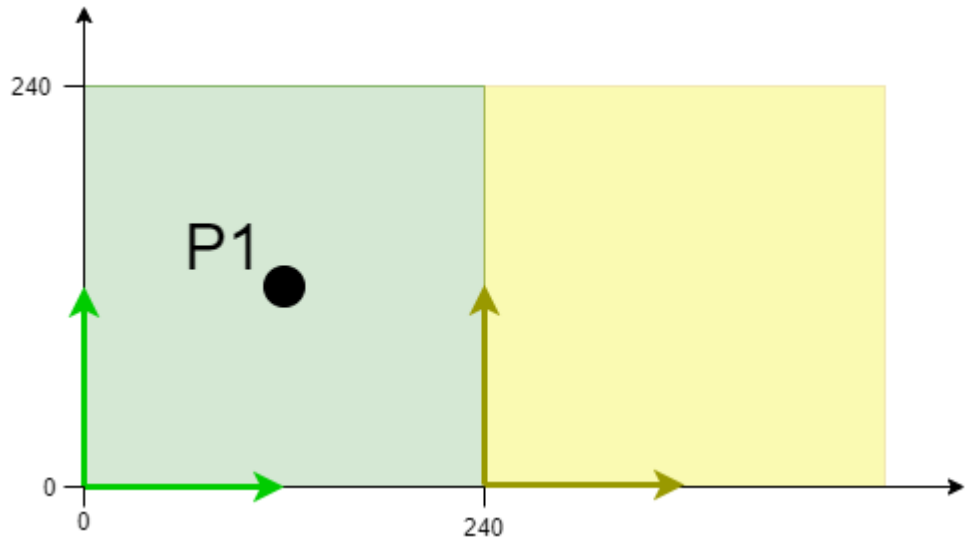
In diesem Beispiel wird ein Planar-Mover auf zwei Planar-Tracks über zwei Planar-Parts verfahren.

Ausgangspunkt

Sie starten mit einer Solution, die eine fertig konfigurierte XPlanar Processing Unit enthält. Unter der XPlanar Processing Unit sind zwei Parts, ein Koordinatensystem und ein Mover angelegt. Unter den beiden Parts ist jeweils eine Kachel angelegt.



Folgende geometrische Situation ist eingestellt: die beiden Parts liegen nebeneinander und der Mover startet in der Mitte des linken Parts (Position P1). Beide Parts sind nicht beweglich und die Konfiguration ist daher statisch.



Ausgehend von dieser Konfiguration wird das Beispiel entwickelt.

Die Erstellung der Ausgangssituation wird in der Dokumentation der XPlanar Processing Unit beschrieben.

Planar-Mover, Planar-Tracks und Planar-Environment anlegen

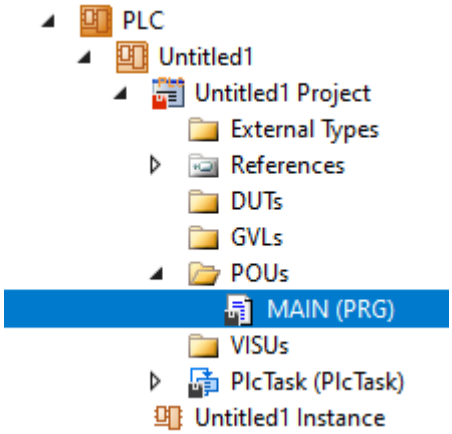
1. Legen Sie für dieses Beispiel einen Planar-Mover an, siehe Konfiguration [▶ 18].
2. Legen Sie eine Planar-Environment an, siehe Konfiguration [▶ 98].

3. Stellen Sie den Initialparameter XPlanar processing unit OID auf die Objekt Id der XPlanar Processing Unit. Damit ist das Part feature für alle **MC Configuration** Objekte aktiviert (besonders für den angelegten Planar-Mover).
4. Fügen Sie zwei Planar-Tracks über **Groups > Add New Item…** hinzu, siehe Konfiguration [▶ 43].
5. Stellen sie den Initialparameter „PartOID“ der beiden Tracks auf den entsprechenden Part, in diesem Beispiel ist der erste Track auf Part 1 und der zweite Track auf Part 2.

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** die gewünschte Anzahl an Movern („MC_PlanarMover“) und Tracks („MC_PlanarTrack“) an.



⇨ Diese repräsentieren Mover und Tracks in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, einen Planar-Mover, zwei Planar-Tracks, eine Zustandsvariable für eine Zustandsmaschine sowie zwei Hilfspositionen für die Tracks an.

```
PROGRAM MAIN
VAR
    mover : MC_PlanarMover;
    track_one, track_two : MC_PlanarTrack;
    state : UDINT;
    pos1, pos2 : PositionXYC;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

⇨ Dieser Programmcode erstellt und aktiviert zwei Tracks und den Mover. Danach wird der Mover auf den ersten Track eingekoppelt und auf den zweiten Track gefahren, wobei die Grenze zwischen Part 1 und Part 2 überquert wird.

```
CASE state OF
  0:
    pos1.SetValuesXYCReferenceId(40, 120, 0, 16#01010060);
    pos2.SetValuesXYCReferenceId(240, 120, 0, 16#01010060);
    track_one.AppendLine(0, pos1, pos2);
    track_two.StartFromTrack(0, track_one);
    pos1.SetValuesXYCReferenceId(260, 120, 0, 16#01010060);
    pos2.SetValuesXYCReferenceId(440, 120, 0, 16#01010060);
    track_two.AppendLine(0, pos1, pos2);
    track_one.Enable(0);
    track_two.Enable(0);
    state := 1;
  1:
    IF track_one.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled AND
    track_two.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 2;
    END_IF
  2:
    mover.Enable(0);
    state := 3;
  3:
    IF mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 4;
    END_IF
  4:
    mover.JoinTrack(0, track_one, 0, 0);
```

```
    state := 5;
  5:
    IF mover.MCTOPLC.STD.CommandMode = MC_PLANAR_MOVER_COMMAND_MODE.OnTrack THEN
      state := 6;
    END_IF
  6:
    mover.MoveOnTrack(0, track_two, 150.0, 0, 0);
    state := 7;

END_CASE
```

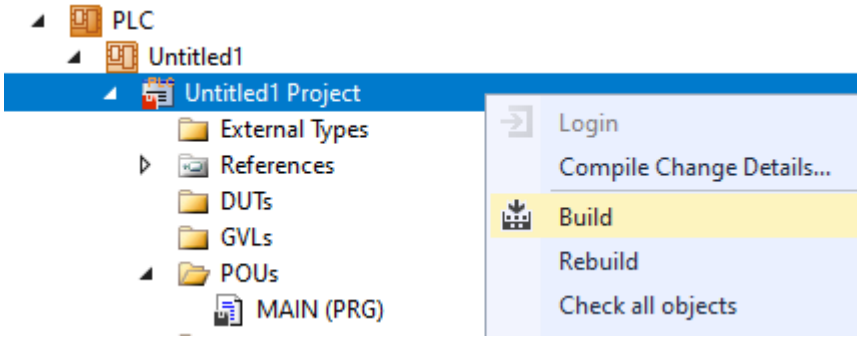
Befehl abschicken

4. Um den Befehl abzuschicken, müssen Sie die Mover und den Track nach dem END_CASE zyklisch mit ihrer Update-Methode aufrufen:

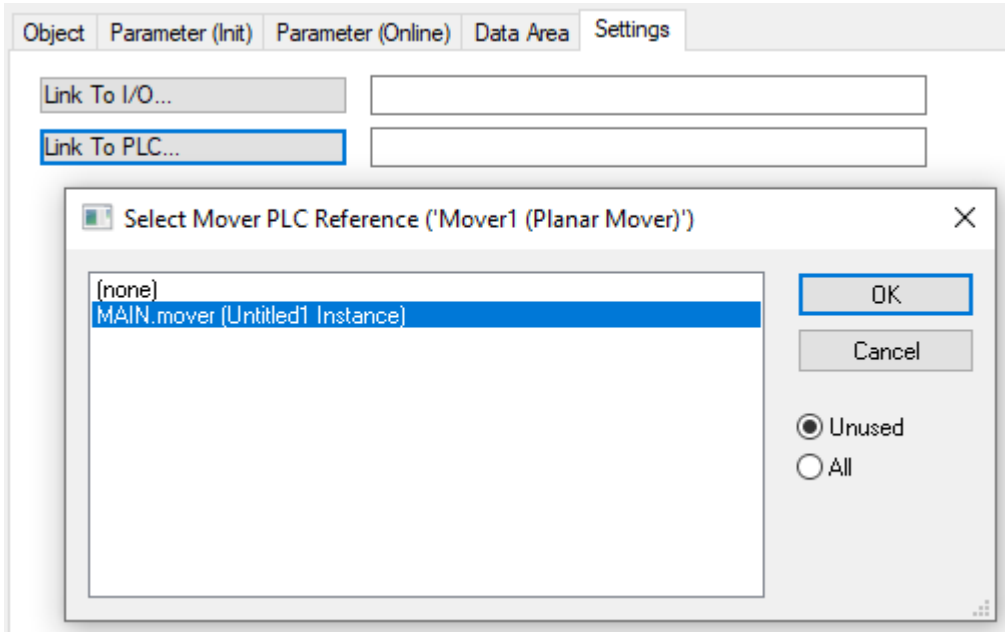
```
mover.Update();
track_one.Update();
track_two.Update();
```

Durch das Bauen der PLC werden Symbole des „PLC-Movers" und „-Tracks“ erzeugt, welche anschließend mit der Mover- bzw. Track-Instanz im MC-Projekt verknüpft werden können.

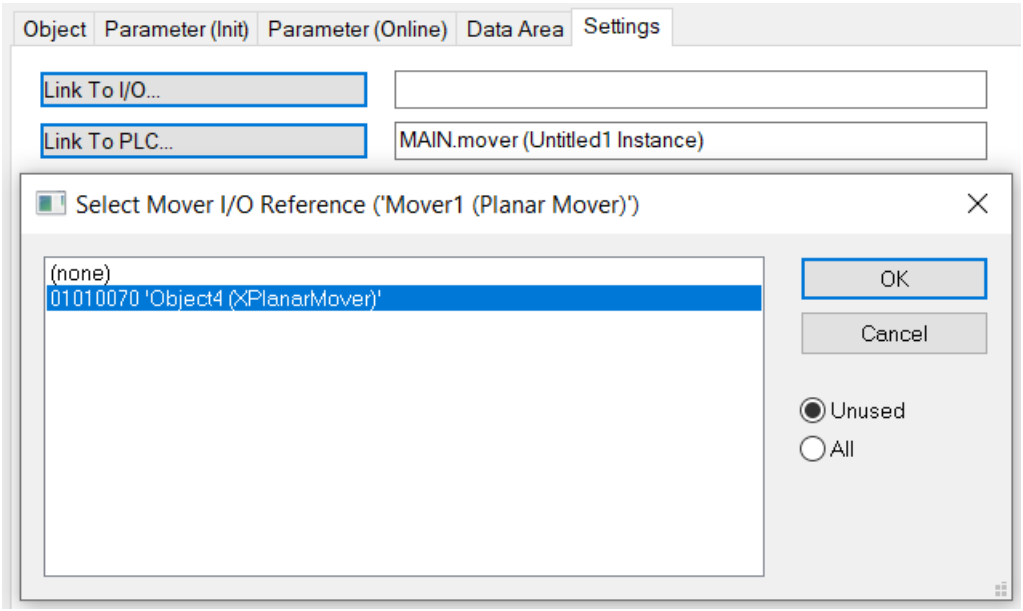
1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**



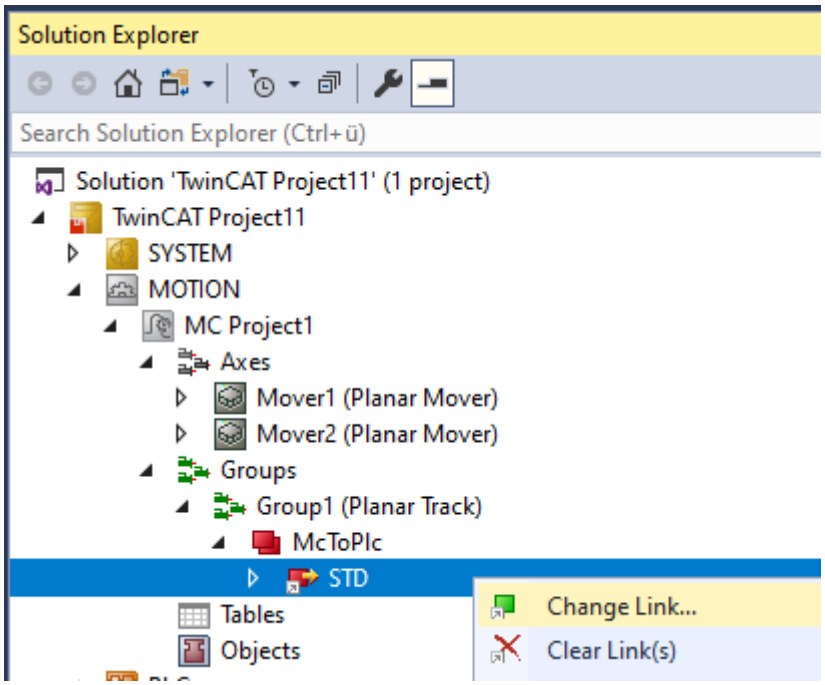
⇨ Anschließend können die Planar-Mover im „MC Project“ mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.

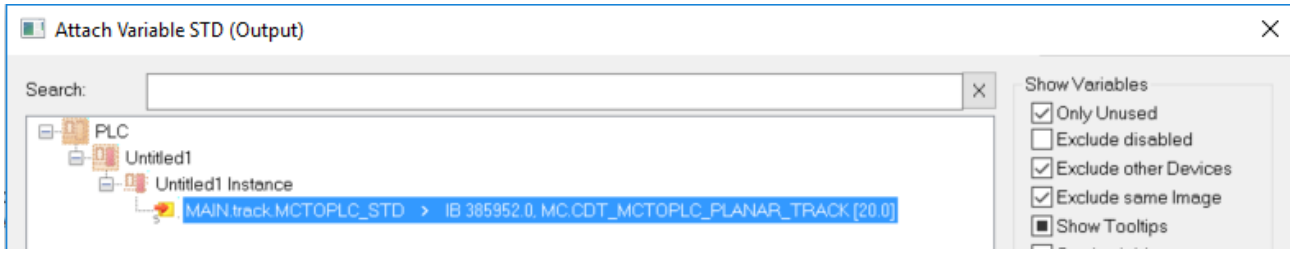


⇨ Zusätzlich muss der Planar-Mover im „MC Project“ (Doppelklick) mit dem Button **Link To I/O…** im Reiter **Settings** verknüpft werden.



⇨ Die Tracks müssen separat über die folgenden Dialogfenster verlinkt werden.





Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .

4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Der Mover steht am Ende des Zustandsautomaten (state=7) auf dem zweiten Track auf Part zwei. Die Positionen der AppendLine-Befehle wurden im globalen Koordinatensystem angegeben, genauso wie die Endposition des Movers.

Expression	Type	Value
mover	MC_PlanarMover	
PLCTOMC	CDT_PLCTOMC...	
MCTOPLC	CDT_MCTOPLC...	
STD	REFERENCE TO...	
SET	REFERENCE TO...	
SetPos	MoverVector	
x	LREAL	390.0000000...
y	LREAL	120
z	LREAL	2.013173942...
a	LREAL	-0.01372402...
b	LREAL	0.010024976...
c	LREAL	0
SetVelo	MoverVector	
SetAcc	MoverVector	
DcTimeStamp	ULINT	7562236335...
PhysicalAreaID	UDINT	16842848
ACT	REFERENCE TO...	
COORDMODE	REFERENCE TO...	
SETONTRACK	REFERENCE TO...	
Error	BOOL	FALSE
ErrorId	UDINT	0
track_one	MC_PlanarTrack	
track_two	MC_PlanarTrack	
state	UDINT	7
pos1	PositionXYC	
pos2	PositionXYC	

6.2.6 Beispiel „Planar-Mover auf Track einkoppeln und in CRotationOnTrack-Modus verfahren“

Anhand dieser Anleitung werden Sie ein TwinCAT-Projekt anlegen, das zwei Planar-Mover sowie einen Planar-Track enthält. Es werden beide Mover auf dem Track einkoppelt und verfahren.

Planar-Mover anlegen

✓ Siehe Konfiguration [▶ 18].

1. Legen Sie zwei Planar-Mover an.
2. Versetzen Sie „Parameter (Init)“ in den Simulationsmodus (`TRUE`). Der Parameter ist versteckt und wird nur sichbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.
3. Ändern Sie die Startposition des zweiten Movers auf x=240.

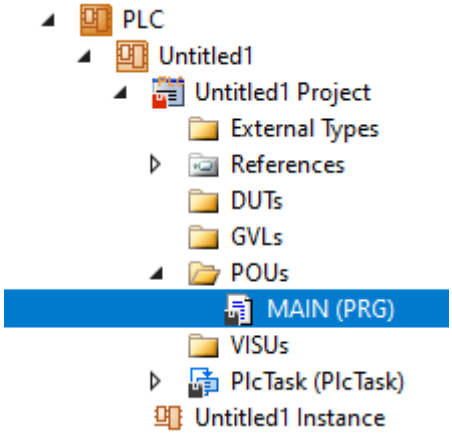
Planar-Track anlegen

4. Fügen Sie den Planar-Track über **Groups > Add New Item…** hinzu, siehe Konfiguration [▶ 43].

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** die gewünschte Anzahl an Movern („MC_PlanarMover“) und Tracks („MC_PlanarTrack“) an.



   ⇨ Diese repräsentieren Mover und Tracks in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, zwei Planar-Mover, einen Planar-Track, eine Zustandsvariable für eine Zustandsmaschine sowie zwei Hilfspositionen für den Track an.

```
PROGRAM MAIN
VAR
    mover_one, mover_two : MC_PlanarMover;
    track : MC_PlanarTrack;
    state : UDINT;
    pos1, pos2 : PositionXYC;
    join_track_options : ST_JoinTrackOptions;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

   ⇨ Dieser Programmcode erstellt und aktiviert einen Track und beide Mover. Danach werden beide Mover auf dem Track eingekoppelt und gedreht.

```
CASE state OF
  0:
    pos1.SetValuesXY(0, 0);
    pos2.SetValuesXY(400, 0);
    track.AppendLine(0, pos1, pos2);
    track.Enable(0);
    state := 1;
  1:
    IF track.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 2;
    END_IF
  2:
    mover_one.Enable(0);
    mover_two.Enable(0);
    state := 3;
  3:
    IF mover_one.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled
    AND mover_two.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 4;
    END_IF
  4:
    join_track_options.useOrientation := FALSE;
    mover_one.JoinTrack(0, track, 0, join_track_options);
    mover_two.JoinTrack(0, track, 0, join_track_options);
    state := 5;
  5:
    IF mover_one.MCTOPLC.STD.CommandMode = MC_PLANAR_MOVER_COMMAND_MODE.OnTrack
    AND mover_two.MCTOPLC.STD.CommandMode = MC_PLANAR_MOVER_COMMAND_MODE.OnTrack THEN
      state := 6;
    END_IF
  6:
    mover_one.MoveC(0, 20.0, 0, 0);
    mover_two.MoveC(0, 90.0, 0, 0);
    state := 7;
  7:
    IF mover_one.MCTOPLC.SET.SetPos.c >= 19.9
    AND mover_two.MCTOPLC.SET.SetPos.c >= 89.9 THEN
      state := 8;
```

```
    END_IF

END_CASE
```

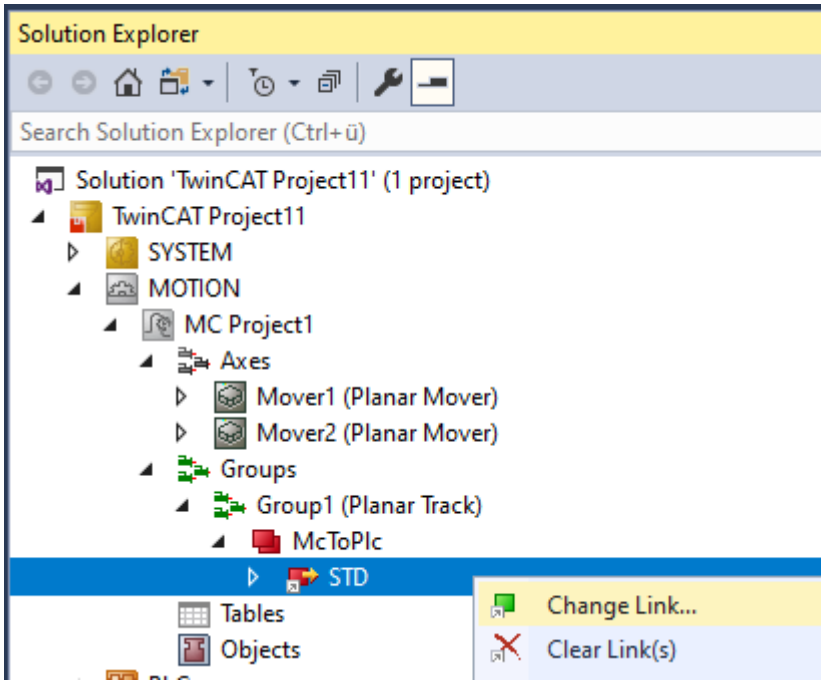
Befehl abschicken

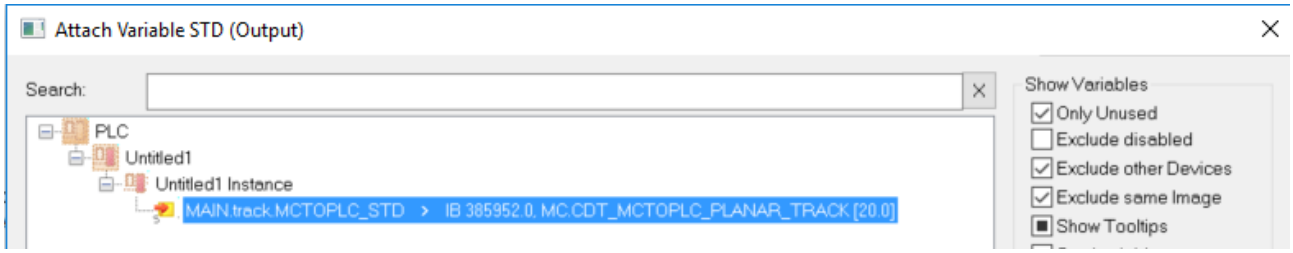
4. Um den Befehl abzuschicken, müssen Sie die Mover und den Track nach dem END_CASE zyklisch mit ihrer Update-Methode aufrufen:

```
mover_one.Update();
mover_two.Update();
track.Update();
```

Durch das Bauen der PLC werden Symbole des „PLC-Movers" und „-Tracks“ erzeugt, welche anschließend mit der Mover- bzw. Track-Instanz im MC-Projekt verknüpft werden können.

1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**
   ⇨ Anschließend können die Planar-Mover im „MC Project“ mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.
   ⇨ Der Track muss separat über die folgenden Dialogfenster verlinkt werden.





Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Die Mover stehen am Ende des Zustandsautomaten (state=8) auf den gewünschten Positionen. Mover zwei befindet sich (wieder) im Zustand OnTrack und Mover eins im Zustand CRotationOnTrack, nachdem beide während der Bewegung im Zustand CRotationOnTrack waren. Mover eins kann nun nur weiter gedreht werden, während Mover zwei auch weiter auf dem Track fahren oder den Track sogar verlassen kann.

ExampleCRotationOnTrack.Untitled1.MAIN

Expression	Type	Value	Prepared value	Address	Comment
mover_one	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC...			%Q*	Mover data that is transferred from the Planar Mover to this function block.
MCTOPLC	CDT_MCTOPLC...			%I*	Mover data that is transferred from the Planar Mover to this function block.
STD	REFERENCE TO...			%IB*	Mover standard data that is transferred from the Planar Mover to this function b
MoverOID	OTCID	16#05110010			Object id of the planar mover.
GroupOID	OTCID	16#00000000			Object id of the planar group the mover is in.
State	MC_PLANAR_S...	Enabled			State of the planar mover, e.g. enabled.
CommandMode	MC_PLANAR_M...	CRotationOn...			Command mode of the planar mover, e.g. onTrack.
Busy	MoverBusy				Busy state of the planar mover.
ErrorCode	HRESULT	16#00000000			Error code of the planar mover.
SET	REFERENCE TO...			%IB*	Mover setpoint data that is transferred from the Planar Mover to this function b
ACT	REFERENCE TO...			%IB*	Mover actpoint data that is transferred from the Planar Mover to this function b
COORDMODE	REFERENCE TO...			%IB*	Mover coordinate mode information that is transferred from the Planar Mover to t
SETONTRACK	REFERENCE TO...			%IB*	Mover busy information that is transferred from the Planar Mover to this functio
Error	BOOL	FALSE			Flag indicating a Planar Mover error.
ErrorId	UDINT	0			Error id indicating the Planar Mover error type.
mover_two	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC...			%Q*	Mover data that is transferred from the Planar Mover to this function block.
MCTOPLC	CDT_MCTOPLC...			%I*	Mover data that is transferred from the Planar Mover to this function block.
STD	REFERENCE TO...			%IB*	Mover standard data that is transferred from the Planar Mover to this function b
MoverOID	OTCID	16#05110020			Object id of the planar mover.
GroupOID	OTCID	16#00000000			Object id of the planar group the mover is in.
State	MC_PLANAR_S...	Enabled			State of the planar mover, e.g. enabled.
CommandMode	MC_PLANAR_M...	OnTrack			Command mode of the planar mover, e.g. onTrack.
Busy	MoverBusy				Busy state of the planar mover.
ErrorCode	HRESULT	16#00000000			Error code of the planar mover.
SET	REFERENCE TO...			%IB*	Mover setpoint data that is transferred from the Planar Mover to this function b
ACT	REFERENCE TO...			%IB*	Mover actpoint data that is transferred from the Planar Mover to this function b
COORDMODE	REFERENCE TO...			%IB*	Mover coordinate mode information that is transferred from the Planar Mover to t
SETONTRACK	REFERENCE TO...			%IB*	Mover busy information that is transferred from the Planar Mover to this functio
Error	BOOL	FALSE			Flag indicating a Planar Mover error.

6.2.7 Beispiel „Planar-Mover auf Track einkoppeln und mit AdoptTrackOrientation verfahren“

Anhand dieser Anleitung werden Sie ein TwinCAT-Projekt anlegen, das zwei Planar-Mover sowie einen Planar-Track enthält. Es werden beide Mover auf dem Track einkoppelt und verfahren.

Planar-Mover anlegen

✓ Siehe Konfiguration [▶ 18].

1. Legen Sie zwei Planar-Mover an.
2. Versetzen Sie „Parameter (Init)“ in den Simulationsmodus (`TRUE`). Der Parameter ist versteckt und wird nur sichbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.
3. Ändern Sie die Startposition des zweiten Movers auf x=240.

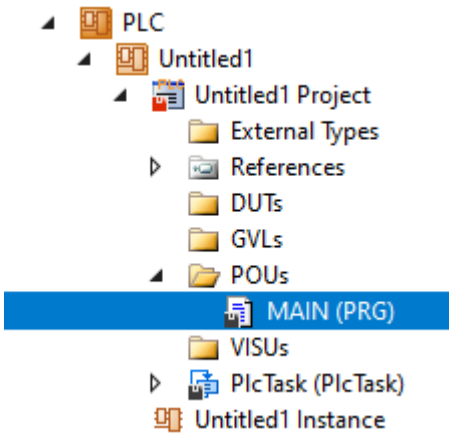
Planar-Track anlegen

4. Fügen Sie den Planar-Track über **Groups > Add New Item…** hinzu, siehe Konfiguration [▶ 43].

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** die gewünschte Anzahl an Movern („MC_PlanarMover“) und Tracks („MC_PlanarTrack“) an.



⇨ Diese repräsentieren Mover und Tracks in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, zwei Planar-Mover, einen Planar-Track, eine Zustandsvariable für eine Zustandsmaschine sowie zwei Hilfspositionen für den Track an.

```
PROGRAM MAIN
VAR
    mover_one, mover_two : MC_PlanarMover;
    track : MC_PlanarTrack;
    state : UDINT;
    pos1, pos2 : PositionXYC;
    join_track_options : ST_JoinTrackOptions;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

⇨ Dieser Programmcode erstellt und aktiviert einen Track und beide Mover. Danach werden beide Mover auf dem Track eingekoppelt und gedreht.

```
CASE state OF
  0:
    pos1.SetValuesXY(0, 0);
    pos2.SetValuesXY(400, 0);
    track.AppendLine(0, pos1, pos2);
    track.Enable(0);
    state := 1;
  1:
    IF track.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 2;
    END_IF
  2:
    mover_one.Enable(0);
    mover_two.Enable(0);
    state := 3;
  3:
    IF mover_one.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled
    AND mover_two.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 4;
    END_IF
  4:
    join_track_options.useOrientation := TRUE;
    mover_one.JoinTrack(0, track, 0, join_track_options);
    mover_two.JoinTrack(0, track, 0, join_track_options);
    state := 5;
  5:
    IF mover_one.MCTOPLC.STD.CommandMode = MC_PLANAR_MOVER_COMMAND_MODE.OnTrack
    AND mover_two.MCTOPLC.STD.CommandMode = MC_PLANAR_MOVER_COMMAND_MODE.OnTrack THEN
      state := 6;
    END_IF
  6:
    mover_one.MoveC(0, 20.0, 0, 0);
    mover_two.MoveC(0, 190.0, 0, 0);
    state := 7;
  7:
    IF mover_one.MCTOPLC.SET.SetPos.c >= 19.9
    AND NOT mover_one.MCTOPLC.STD.Busy.busyMover
    AND mover_two.MCTOPLC.SET.SetPos.c >= 189.9
    AND NOT mover_two.MCTOPLC.STD.Busy.busyMover THEN
      state := 8;
    END_IF
  8:
    mover_one.AdoptTrackOrientation(0, 0, 0);
```

```
    mover_two.AdoptTrackOrientation (0, 0, 0);
    state := 9;

END_CASE
```

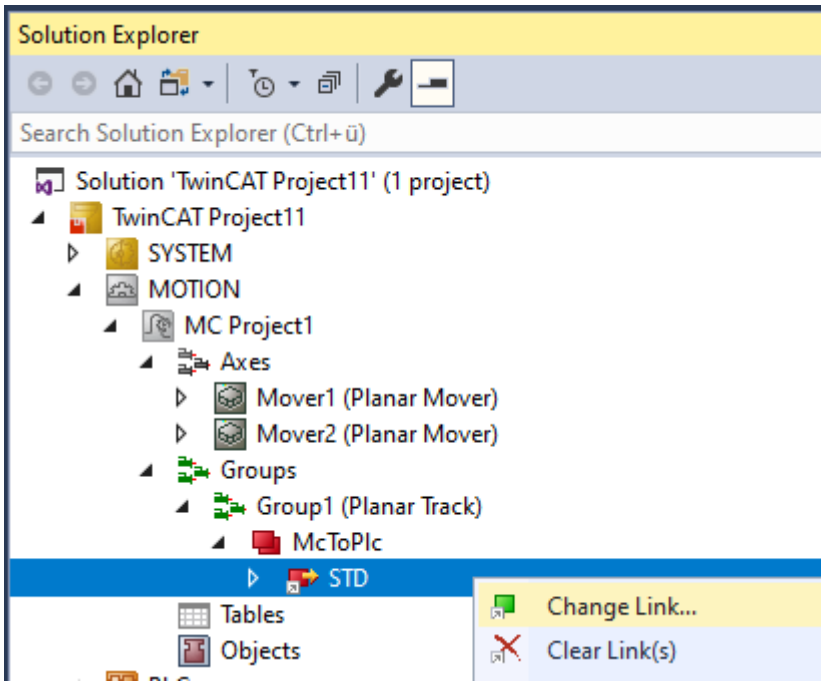
Befehl abschicken

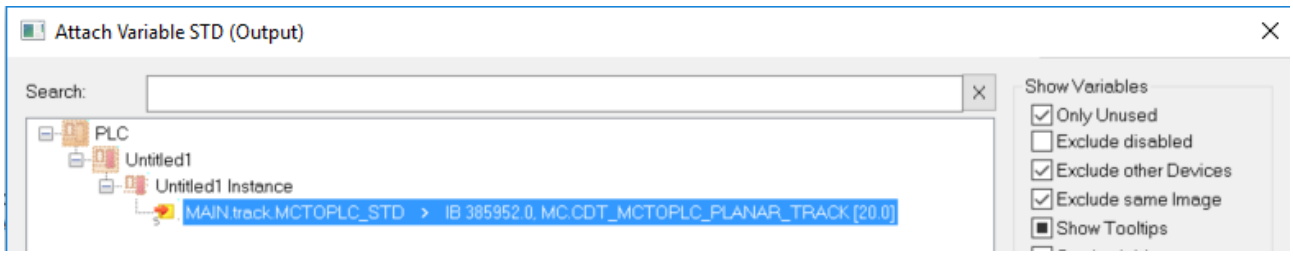
4. Um den Befehl abzuschicken, müssen Sie die Mover und den Track nach dem END_CASE zyklisch mit ihrer Update-Methode aufrufen:

```
mover_one.Update();
mover_two.Update();
track.Update();
```

Durch das Bauen der PLC werden Symbole des „PLC-Movers" und „-Tracks“ erzeugt, welche anschließend mit der Mover- bzw. Track-Instanz im MC-Projekt verknüpft werden können.

1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**
   ⇨ Anschließend können die Planar-Mover im „MC Project“ mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.
   ⇨ Der Track muss separat über die folgenden Dialogfenster verlinkt werden.





Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Beide Mover werden mit an den Track gekoppelter Orientierung zum Track hinzugefügt. Danach wird jeweils durch ein MoveC die Orientierung vom Track entkoppelt. Die Mover stehen am Ende des Zustandsautomaten (state=9) auf den gewünschten Positionen. Beide Mover befinden sich (wieder) im Zustand OnTrack und haben durch den Befehl AdoptTrackOrientation wieder die an den Track gekoppelte Orientierung. Der Befehl hat 3 Parameter: Erstens ein optionales Feedback-Objekt, zweitens ein optionales Constraints-Objekt und drittens ein optionales Options-Objekt.

ExampleAdoptTrackOrientation.Untitled1.MAIN

Expression	Type	Value	Prepared value	Address	Comment
mover_one	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC...			%Q*	Mover data that is transferred from the Planar Mover to this function block.
MCTOPLC	CDT_MCTOPLC...			%I*	Mover data that is transferred from the Planar Mover to this function block.
STD	REFERENCE TO...			%IB*	Mover standard data that is transferred from the Planar Mover to this function b
MoverOID	OTCID	16#05110010			Object id of the planar mover.
GroupOID	OTCID	16#00000000			Object id of the planar group the mover is in.
State	MC_PLANAR_S...	Enabled			State of the planar mover, e.g. enabled.
CommandMode	MC_PLANAR_M...	OnTrack			Command mode of the planar mover, e.g. onTrack.
Busy	MoverBusy				Busy state of the planar mover.
ErrorCode	HRESULT	16#00000000			Error code of the planar mover.
SET	REFERENCE TO...			%IB*	Mover setpoint data that is transferred from the Planar Mover to this function b
ACT	REFERENCE TO...			%IB*	Mover actpoint data that is transferred from the Planar Mover to this function b
COORDMODE	REFERENCE TO...			%IB*	Mover coordinate mode information that is transferred from the Planar Mover to t
XYCoordinateMode	MC_PLANAR_C...	Dependent			X and Y coordinate.
ZCoordinateMode	MC_PLANAR_C...	Independent			Z coordinate.
ACoordinateMode	MC_PLANAR_C...	Independent			A coordinate.
BCoordinateMode	MC_PLANAR_C...	Independent			B coordinate.
CCoordinateMode	MC_PLANAR_C...	Dependent			C coordinate.
SETONTRACK	REFERENCE TO...			%IB*	Mover busy information that is transferred from the Planar Mover to this functio
Error	BOOL	FALSE			Flag indicating a Planar Mover error.
ErrorId	UDINT	0			Error id indicating the Planar Mover error type.
mover_two	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC...			%Q*	Mover data that is transferred from the Planar Mover to this function block.
MCTOPLC	CDT_MCTOPLC...			%I*	Mover data that is transferred from the Planar Mover to this function block.
STD	REFERENCE TO...			%IB*	Mover standard data that is transferred from the Planar Mover to this function b
MoverOID	OTCID	16#05110020			Object id of the planar mover.
GroupOID	OTCID	16#00000000			Object id of the planar group the mover is in.
State	MC_PLANAR_S...	Enabled			State of the planar mover, e.g. enabled.
CommandMode	MC_PLANAR_M...	OnTrack			Command mode of the planar mover, e.g. onTrack.
Busy	MoverBusy				Busy state of the planar mover.
ErrorCode	HRESULT	16#00000000			Error code of the planar mover.
SET	REFERENCE TO...			%IB*	Mover setpoint data that is transferred from the Planar Mover to this function b
ACT	REFERENCE TO...			%IB*	Mover actpoint data that is transferred from the Planar Mover to this function b
COORDMODE	REFERENCE TO...			%IB*	Mover coordinate mode information that is transferred from the Planar Mover to t
XYCoordinateMode	MC_PLANAR_C...	Dependent			X and Y coordinate.
ZCoordinateMode	MC_PLANAR_C...	Independent			Z coordinate.
ACoordinateMode	MC_PLANAR_C...	Independent			A coordinate.
BCoordinateMode	MC_PLANAR_C...	Independent			B coordinate.
CCoordinateMode	MC_PLANAR_C...	Dependent			C coordinate.
SETONTRACK	REFERENCE TO...			%IB*	Mover busy information that is transferred from the Planar Mover to this functio
Error	BOOL	FALSE			Flag indicating a Planar Mover error.
ErrorId	UDINT	0			Error id indicating the Planar Mover error type.
track	MC_PlanarTrack				
state	UDINT	9			
pos1	PositionXYC				
pos2	PositionXYC				
join_track_options	ST_JoinTrackO...				

6.2.8 Beispiel „Planar-Mover auf einem Track mit einer Achse synchronisieren“

Anhand dieser Anleitung werden Sie ein TwinCAT-Projekt anlegen, in welchem ein Planar-Mover, der sich auf einem Track befindet, an eine Achse gekoppelt wird, deren Sollwerten er daraufhin folgt.

Hier wird der Planar-Mover nicht direkt über einen MoveOnTrack-Befehl gesteuert, bei welchem eine vorgegebene Zielposition mit anschließendem Stillstand angefahren wird, siehe Beispiel „Planar-Mover auf Track einkoppeln und verfahren“ [▶ 51]. Stattdessen bleibt der Planar-Mover so lange an eine Achse gekoppelt, bis ein Folgebefehl diese Kopplung beendet, oder ein Fehler auftritt.

Nach dem Abschicken des GearInPosOnTrack [▶ 150]-Befehls, der die Kopplung an eine Achse initiiert, wird der Planar-Mover versuchen, an der angegebenen slaveSyncPosition zu sein, wenn die Achse, an welche er gekoppelt ist, sich an der masterSyncPosition befindet und gleichzeitig die Dynamik der Masterachse zu übernehmen. Sollte es möglich sein, schon eher synchron zu sein (d. h. der Planar-Mover hat schon bei slaveSyncPosition – x dieselbe Dynamik wie die Master-Achse, die sich zu diesem Zeitpunkt bei masterSyncPosition – x befindet), dann wird der Planar-Mover diese Konfiguration ansteuern und schon eher synchron werden. Sollte es nicht möglich sein, zum angegebenen Punkt synchron zu werden, so wird der Planar-Mover so lange versuchen mit der Master-Achse synchron zu werden, bis ein Folgebefehl empfangen wird oder ein Fehler auftritt.

Sollte der Planar-Mover seinen Synchronisationsstatus verlieren, z. B. aufgrund sich schnell ändernder Dynamik der Master-Achse, wird er versuchen, möglichst schnell wieder synchron zu werden. Der Synchronisationsstatus ist jederzeit aus der PLC über das entsprechende Feedback-Objekt zugänglich. Auch kann die Synchronisation verloren werden, wenn das Einhalten des vorgegebenen Abstands zum voranfahrenden Planar-Mover ein Abbremsen des synchronen Planar-Movers nötig macht. Auch hier gilt, dass schnellstmöglich versucht wird, die Synchronisation wiederzuerlangen, wenn das Hindernis sich entfernt hat.

Ein Beispiel für einen Fehler, der zum Abbruch des Befehls führt, ist ein Verhalten der Master-Achse, welches den Planar-Mover dazu zwingen würde, sich mit negativer Geschwindigkeit über den Start eines Planar-Tracks hinaus zu bewegen - eine solche Bewegung ist auch mit einem MoveOnTrack [▶ 149]-Befehl nicht zulässig. In einem solchen Fall wird der Planar-Mover so lange synchron bleiben (oder versuchen es zu werden, falls er es noch nicht ist), bis er gezwungen wird, anzuhalten, sodass er am Beginn des Planar-Tracks zum Stehen kommt. Außerdem wird ein Fehler zurückgemeldet. Die genaue Position, an welcher der Planar-Mover seinen Halt einleitet, ist abhängig von den gegenwärtigen Dynamiklimits.

Wenn dem GearInPosOnTrack [▶ 150]-Befehl Dynamiklimits mitgegeben werden, deren Geschwindigkeitslimit unterhalb der momentanen Geschwindigkeit der Master-Achse liegt, so wird der Planar-Mover nichtsdestotrotz versuchen, aufzusynchronisieren, da es nicht ausgeschlossen ist, dass die Master-Achse zu einem späteren Zeitpunkt derart abbremst, dass sie wieder erreichbar wird. Insbesondere wird in einem solchen Fall kein Fehler zurückgegeben.

Planar-Mover anlegen

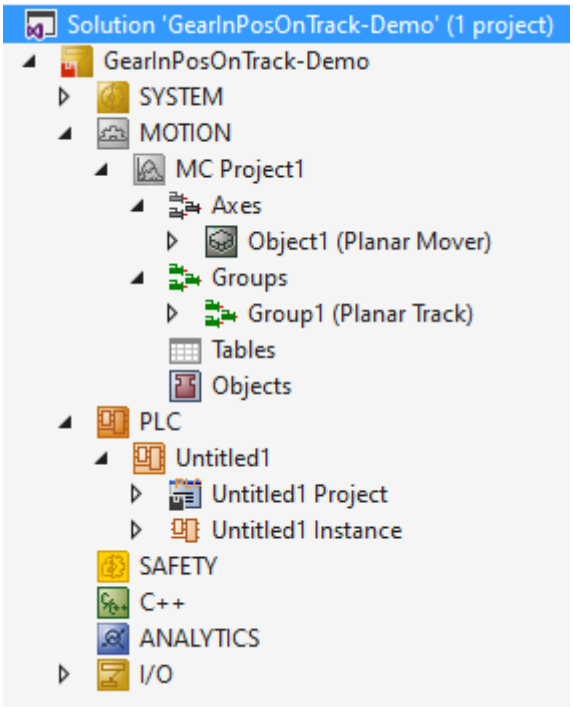
✓ Siehe Konfiguration [▶ 18].

1. Legen Sie einen Planar-Mover an.
2. Versetzen Sie „Parameter (Init)“ in den Simulationsmodus (`TRUE`). Der Parameter ist versteckt und wird nur sichbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.

Planar-Track anlegen

3. Fügen Sie den Planar-Track über **Groups > Add New Item…** hinzu, siehe Konfiguration [▶ 43].

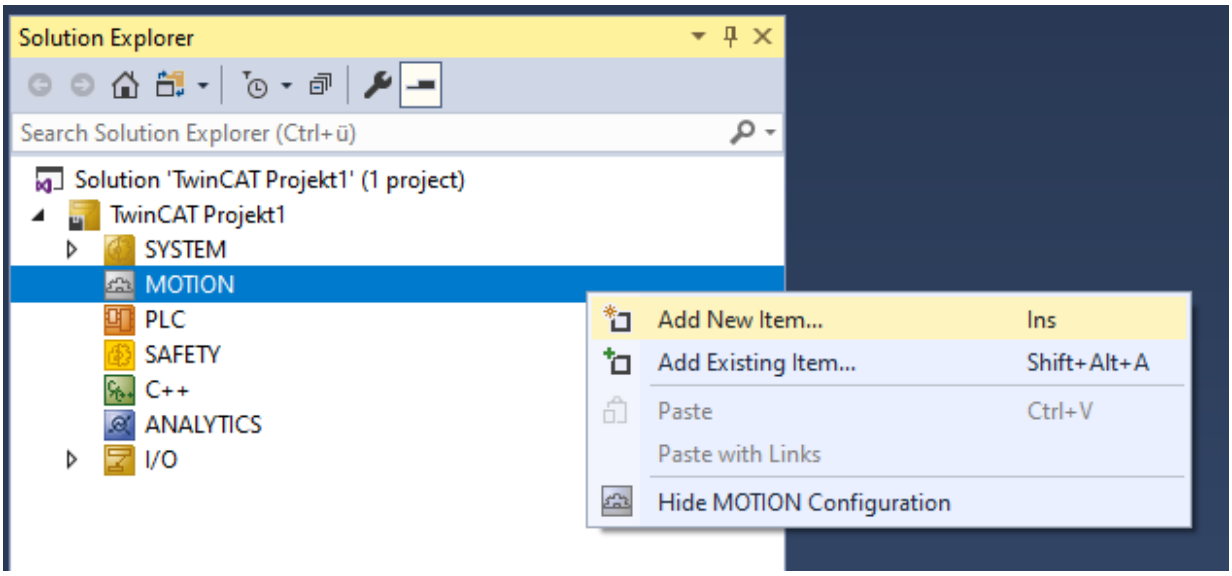
⇨ Der Solution Explorer weist die folgenden Einträge auf:



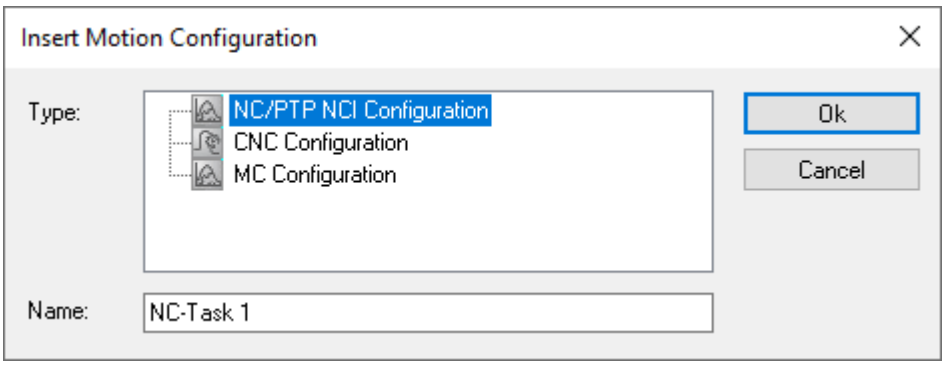
Master-Achse anlegen

✓ Um eine Master-Achse anzulegen, muss zunächst eine **NC/PTP NCI Configuration** angelegt werden.

1. Wählen Sie **MOTION** > **Add New Item…** aus.

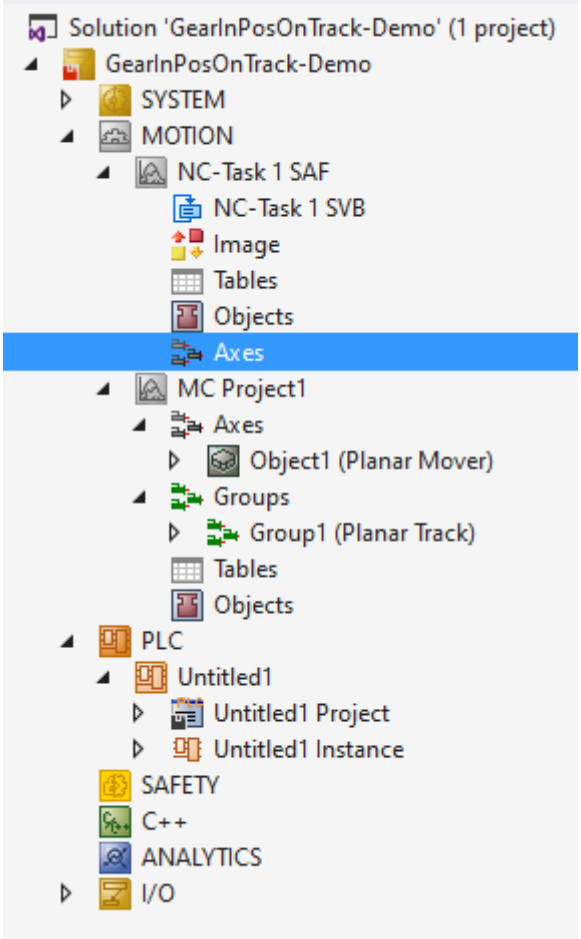


2. Wählen Sie im folgenden Dialogfenster **NC/PTP NCI Configuration** aus und bestätigen Sie mit **OK**.

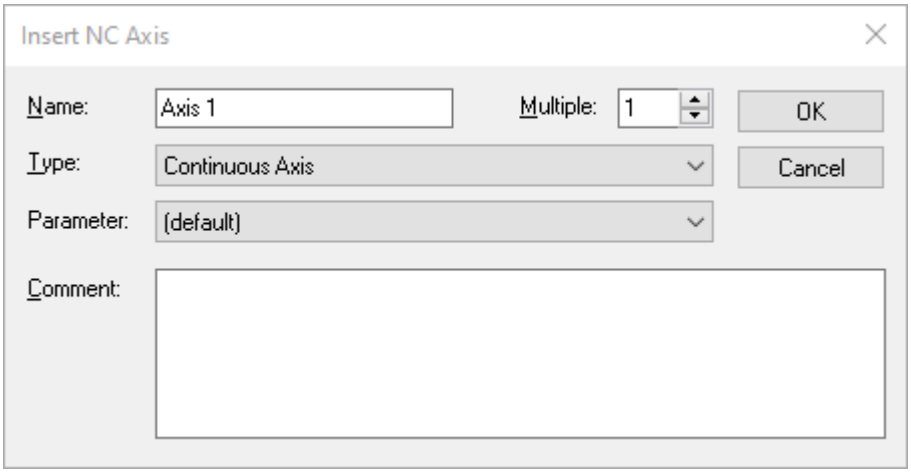


⇨ Sie haben ein NC/PTP NCI Project angelegt.

3. Wählen Sie per rechter Maustaste im erzeugten NC-Project **Axes** > **Add New Item…** aus.



4. Legen Sie im folgenden Dialogfenster eine (oder mehrere) Achsen an und bestätigen Sie mit **OK**

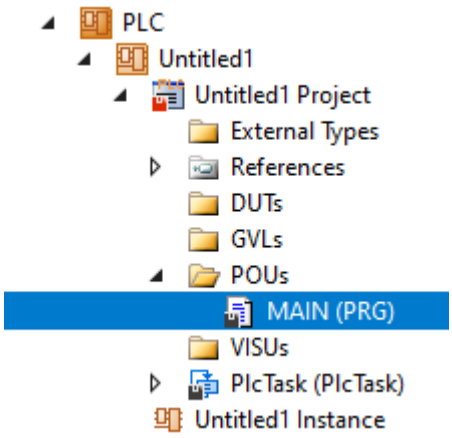


PLC anlegen

Für dieses PLC-Projekt müssen Sie zum Steuern der Master-Achse zusätzlich „Tc2_MC2“ hinzufügen, siehe Bibliotheken einfügen [▶ 127].

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** die gewünschte Anzahl an Movern („MC_PlanarMover“) und Tracks („MC_PlanarTrack“) an.

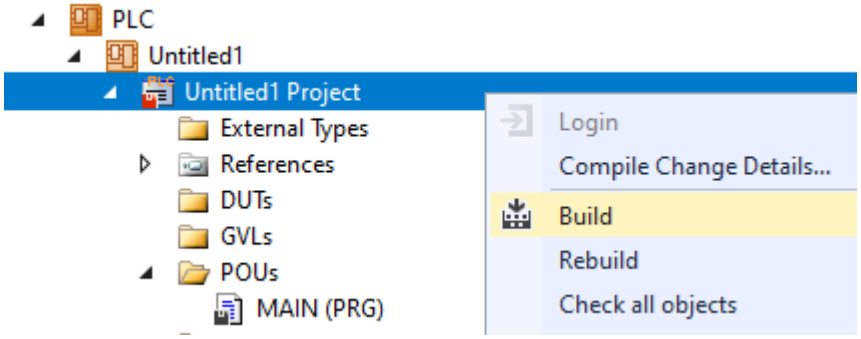


⇨ Diese repräsentieren Mover und Tracks in der MC Configuration.

2. Legen Sie die folgenden Variablen an.

```
PROGRAM MAIN
VAR
    mover      : MC_PlanarMover;
    track      : MC_PlanarTrack;
    axis       : AXIS_REF;
    power_axis : MC_Power;
    move_axis  : MC_MoveAbsolute;
    state      : UDINT;
    pos1, pos2 : PositionXYC;
END_VAR
```

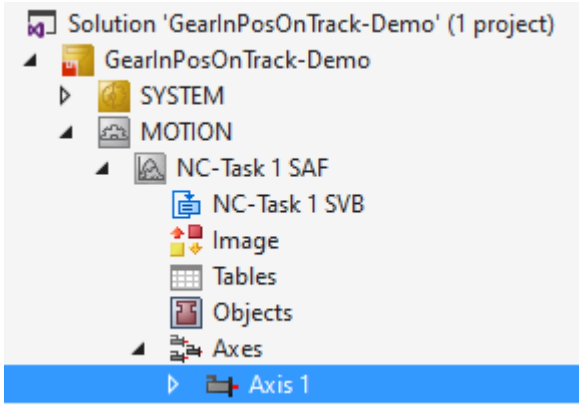
3. Bauen Sie die PLC, um Symbole des „PLC-Mover“, des „PLC-Track“ und der „PLC-Achse“ zu erzeugen.



4. Verlinken Sie Planar-Mover, Planar-Track (siehe Beispiel „Planar-Mover auf Track einkoppeln und verfahren“ [▶ 51]) und, wie im nächsten Abschnitt beschrieben, die Achse.

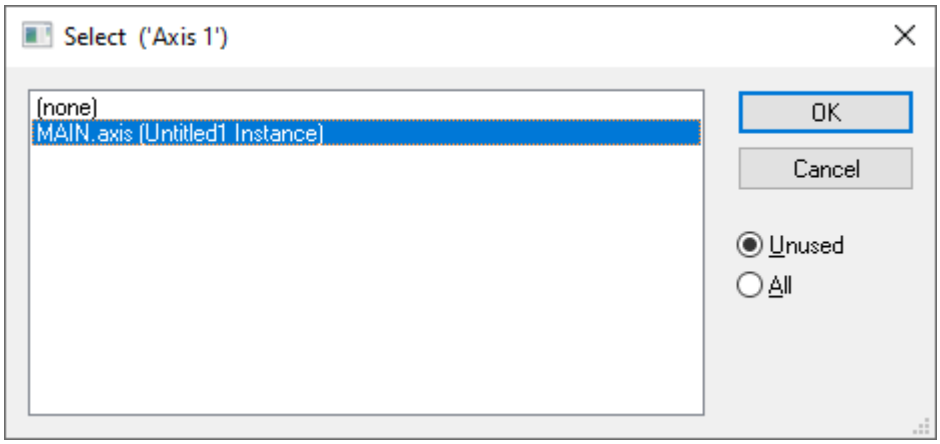
Achse verlinken

5. Doppelklicken Sie im Solution Explorer **Axis 1.**



6. Wechseln Sie in den Reiter **Settings**.

7. Klicken Sie **Link to PLC…** und wählen Sie im folgenden Dialog den Eintrag **MAIN.axis** aus und bestätigen Sie mit **OK**.



Zustandsautomaten programmieren

Mit der folgenden Zustandsmaschine, die im MAIN programmiert wird, werden der Planar-Track geometrisch definiert und aktiviert (State 0), der Planar-Mover aktiviert und auf den Planar-Track eingekoppelt (State 2 bzw. 4), die Master-Achse freigegeben (State 6) und verfahren (State 7).

Schlussendlich wird der Befehl zum Starten der Synchronisation mit der Master-Achse („GearInPosOnTrack [▶ 150]“) an den Planar-Mover geschickt (State 8). Auch hier werden zyklisch die Planar-Objekte geupdated bzw. die Achs-FBs aufgerufen (nach `END_CASE`-Anweisung):

```
CASE state OF
  0:
    pos1.SetValuesXYC(100, 100, 0);
    pos2.SetValuesXYC(860, 100, 0);
    track.AppendLine(0, pos1, pos2);
    track.Enable(0);
    state := state + 1;
  1:
    IF track.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
      state := state + 1;
    END_IF
  2:
    mover.Enable(0);
    state := state + 1;
  3:
    IF mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := state + 1;
    END_IF
  4:
    mover.JoinTrack(0, track, 0, 0);
    state := state + 1;
  5:
    IF mover.MCTOPLC.STD.CommandMode = MC_PLANAR_MOVER_COMMAND_MODE.OnTrack THEN
      state := state + 1;
    END_IF
  6:
    power_axis(Axis := axis,
    Enable := TRUE,
    Enable_Positive := TRUE);
    IF power_axis.Status THEN
      move_axis(Axis := axis, Execute := FALSE);
      state := state + 1;
    END_IF
  7:
    move_axis(Axis := axis,
    Position := 600,
    Velocity := 30,
    Acceleration := 100,
    Deceleration := 100,
    Jerk := 100,
    Execute := TRUE);
    state := state + 1;
  8:
    mover.GearInPosOnTrack(0, axis.DriveAddress.TcAxisObjectId, 0, 100, 100, track, 0, 0);
    state := state + 1;
```

```
END_CASE

mover.Update();
track.Update();
power_axis(Axis := axis);
move_axis(Axis := axis);
axis.ReadStatus();
```

Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Die Master-Achse wird zur gegebenen Zielposition (hier 600) verfahren und der Planar-Mover wird ihrer Bewegung folgen. Die Position des Planar-Movers kann im Online-View (über Klicken des Buttons) verfolgt werden.

Expression	Type	Value
mover	MC_PlanarMover	
PLCTOMC	CDT_PLCTOMC_PLA...	
MCTOPLC	CDT_MCTOPLC_PLA...	
STD	REFERENCE TO CDT...	
SET	REFERENCE TO CDT...	
ACT	REFERENCE TO CDT...	
COORDMODE	REFERENCE TO CDT...	
SETONTRACK	REFERENCE TO CDT...	
SetPos	LREAL	274.1483232748...
SetVelo	LREAL	29.99999999999...
SetAcc	LREAL	0
SetJerk	LREAL	0
TrackOID	OTCID	16#05120010
Error	BOOL	FALSE
ErrorId	UDINT	0

Der Mover kommt bei Position 600 zum Stillstand, da hier auch die Master-Achse Nulldynamik erreicht. Falls für die Zielposition der Master-Achse ein Wert größer als die Länge des Tracks (hier 760) in State 7 programmiert wird, so kommt der Planar-Mover am Ende des Planar-Tracks zum Stehen um nicht zu entgleisen und folgt der Master-Achse nicht weiter. Der Fehler in einem solchen Szenario wird zwar potentiell von der MC an die PLC zurückgegeben, von obigem PLC-Code in diesem Fall jedoch nicht angenommen. Hierfür und zur Überwachung des Synchronisationsstatus ist ein Feedback [▶ 121]-Objekt nötig.

Im Funktionsaufruf in State 8 werden dem Planar-Mover die Syncpositionen der Master-Achse (als drittes Argument) bzw. des Slave-Planar-Movers (viertes Argument) übergeben. Dies sind die jeweiligen Positionen, an welchen der Slave synchron mit dem Master wird, also seine Dynamikwerte erreicht. Das fünfte Argument im Funktionsaufruf gibt an, auf welchen Planar-Track sich die Position im vorigen Argument bezieht. Tatsächlich ist es dem Slave hier möglich bedeutend eher mit seinem Master synchron zu werden.

Eine Synchronisationsbewegung über eine Folge von aneinander anschließenden Tracks ist durch die Verwendung eines Planar-TrackTrail [▶ 122]-Objektes möglich. In einem solchen Fall ist ein Übergang von einem zum nächsten Planar-Track während der Phase des Aufsynchronisierens oder bei bereits

bestehender Sychronizität möglich. Das Abbremsen des Planar-Movers analog zum obigen Beispiel mit nur einem Planar-Track würde erst am Ende des letzten Planar-Tracks eintreten, falls die Bewegung der Master-Achse ein Überschreiten desselben verlangen würde.

6.2.9 Beispiel „Planar-Mover auf einem Track mit einem anderen Planar-Mover synchronisieren“

Anhand dieser Anleitung werden Sie ein TwinCAT-Projekt anlegen, in welchem ein Planar-Mover, der sich auf einem Planar-Track befindet, an einen anderen Planar-Mover auf einem parallelen Planar-Track gekoppelt wird und daraufhin dessen Sollwerten folgt.

Die Kopplung eines Planar-Movers an einen anderen Planar-Mover ist weitgehend analog zur Kopplung eines Planar-Movers an eine Achse, siehe Beispiel „Planar-Mover auf einem Track mit einer Achse synchronisieren“ [▶ 66]. Das aktuelle Beispiel ist kurz gefasst und baut auf das obengenannte Beispiel auf.

Planar-Mover anlegen

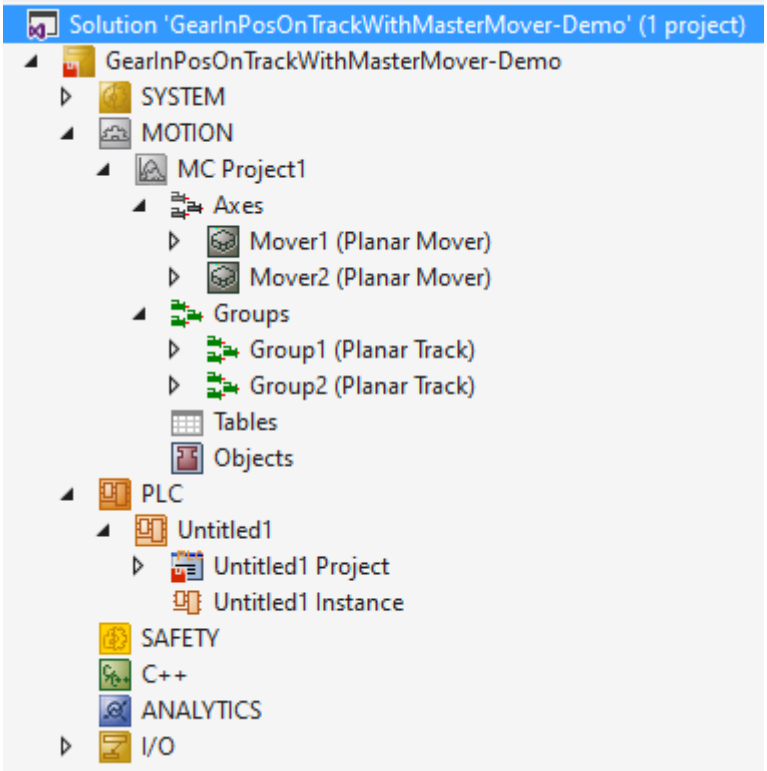
✓ Siehe Konfiguration [▶ 18].

1. Legen Sie zwei Planar-Mover an.
2. Versetzen Sie „Parameter (Init)“ in den Simulationsmodus (`TRUE`). Der Parameter ist versteckt und wird nur sichbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.

Planar-Track anlegen

3. Fügen Sie zwei Planar-Tracks über **Groups > Add New Item…** hinzu, siehe Konfiguration [▶ 43].

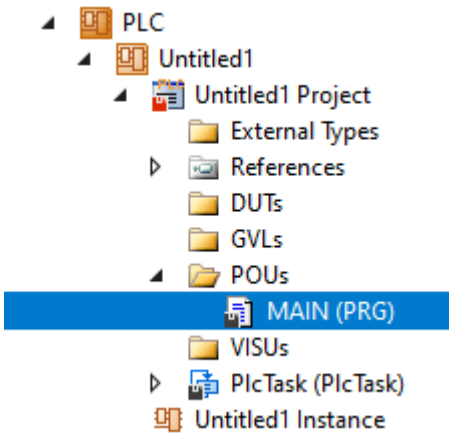
⇨ Der Solution Explorer weist die folgenden Einträge auf:



PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** die gewünschte Anzahl an Movern („MC_PlanarMover“) und Tracks („MC_PlanarTrack“) an.

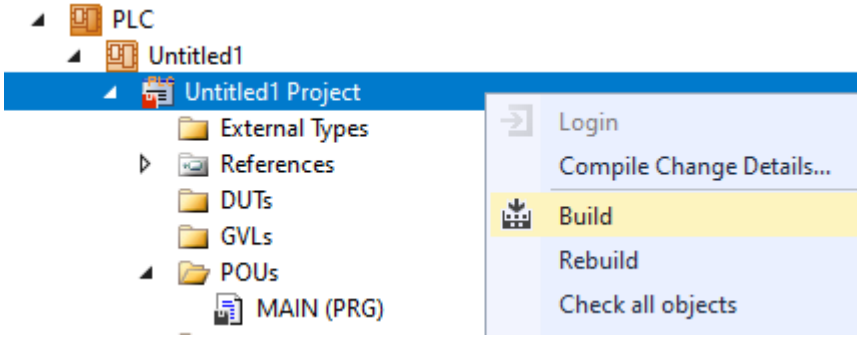


⇨ Diese repräsentieren Mover und Tracks in der MC Configuration.

2. Legen Sie die folgenden Variablen an.

```
PROGRAM MAIN
VAR
    master_mover : MC_PlanarMover;
    slave_mover  : MC_PlanarMover;
    master_track : MC_PlanarTrack;
    slave_track  : MC_PlanarTrack;
    state        : UDINT;
    pos1, pos2   : PositionXYC;
END_VAR
```

3. Bauen Sie die PLC um Symbole der „PLC-Mover“ und „-Tracks“ zu erzeugen.



4. Verlinken Sie die Planar-Mover und die Planar-Tracks (siehe Beispiel „Planar-Mover auf Track einkoppeln und verfahren“ [▶ 51]).

Zustandsautomaten programmieren

Mit der folgenden Zustandsmaschine, die im MAIN programmiert wird, werden die Planar-Tracks geometrisch definiert und aktiviert (States 0 bis 3), die Planar-Mover aktiviert und auf den jeweiligen Planar-Track eingekoppelt (States 4 bis 11) und der als Master fungierende Planar-Mover auf seinem Track verfahren (State 12).

Schlussendlich wird der Befehl zum Starten der Synchronisation mit dem Master-Planar-Mover (GearInPosOnTrackWithMasterMover [▶ 151]) an den Slave-Planar-Mover geschickt (State 13). Nach der END_CASE-Anweisung werden zyklisch die Planar-Objekte aktualisiert.

```
CASE state OF
  0:
    pos1.SetValuesXYC(100, 620, 0);
    pos2.SetValuesXYC(860, 620, 0);
    master_track.AppendLine(0, pos1, pos2);
    master_track.Enable(0);
    state := state + 1;
  1:
    IF master_track.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
      state := state + 1;
    END_IF
  2:
    pos1.SetValuesXYC(100, 100, 0);
    pos2.SetValuesXYC(860, 100, 0);
    slave_track.AppendLine(0, pos1, pos2);
```

```
    slave_track.Enable(0);
    state := state + 1;
  3:
    IF slave_track.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
      state := state + 1;
    END_IF
  4:
    master_mover.Enable(0);
    state := state + 1;
  5:
    IF master_mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := state + 1;
    END_IF
  6:
    master_mover.JoinTrack(0, master_track, 0, 0);
    state := state + 1;
  7:
    IF master_mover.MCTOPLC.STD.CommandMode = MC_PLANAR_MOVER_COMMAND_MODE.OnTrack THEN
      state := state + 1;
    END_IF
  8:
    slave_mover.Enable(0);
    state := state + 1;
  9:
    IF slave_mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := state + 1;
    END_IF
  10:
    slave_mover.JoinTrack(0, slave_track, 0, 0);
    state := state + 1;
  11:
    IF slave_mover.MCTOPLC.STD.CommandMode = MC_PLANAR_MOVER_COMMAND_MODE.OnTrack THEN
      state := state + 1;
    END_IF
  12:
    master_mover.MoveOnTrack(0, 0, 500.0, 0, 0);
    state := state + 1;
  13:
    slave_mover.GearInPosOnTrackWithMasterMover(0, master_mover, 0, 100.0, master_track, 100.0, slav
e_track, 0, 0);
    state := state + 1;
END_CASE

master_mover.Update();
slave_mover.Update();
master_track.Update();
slave_track.Update();
```

Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Der Master-Planar-Mover wird zur gegebenen Zielposition (hier 500) auf dem angegebenen Planar-Track verfahren und der Slave-Planar-Mover wird seiner Bewegung folgen. Die Positionen der Planar-Mover können im Online-View (über Klicken des Buttons) verfolgt werden.

Der Slave-Planar-Mover kommt bei Position 500 zum Stillstand, da hier auch der Master-Planar-Mover Nulldynamik erreicht.

Im Funktionsaufruf in State 13 werden dem Slave-Planar-Mover die Syncpositionen des Masters (Argumente 4 und 5) bzw. Slaves (Argumente 6 und 7) übergeben. Dies sind die jeweiligen Positionen, an welchen der Slave synchron mit dem Master wird, also seine Dynamikwerte erreicht. Tatsächlich ist es dem Slave hier wie auch im Beispiel „Planar-Mover auf einem Track mit einer Achse synchronisieren“ [▶ 66] möglich

bedeutend eher mit seinem Master synchron zu werden. Wie auch bei der Synchronisation mit einer Achse ist für die Überwachung von Synchronizitätsstatus und möglichen Fehlern ein spezielles Spezialisierte Feedback-Typen [▶ 121]-Objekt nötig.

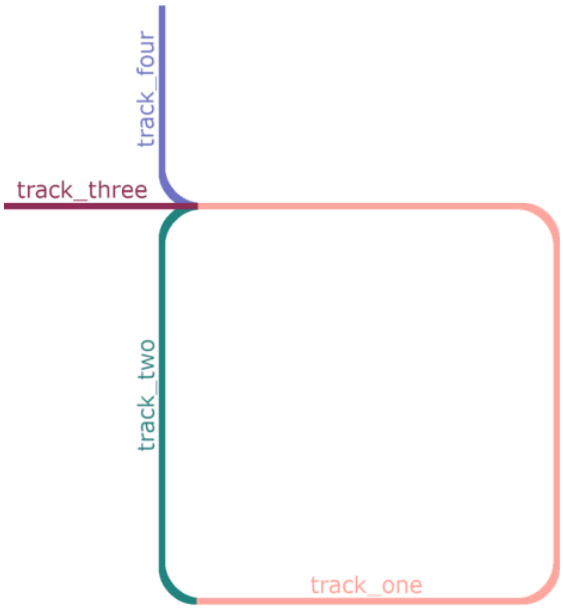
Wie auch beim Synchronisieren mit einer Master-Achse kann durch das Spezifizieren eines Planar-TrackTrail [▶ 122]-Objektes die Synchronisationsbewegung des Slaves über mehrere Tracks programmiert werden.

Fährt der Master-Planar-Mover während eines aktiven Synchronisationskommandos über eine Trackgrenze hinweg, so wird die Position, die er an seinen Slave weitergibt, einfach über die Trackgrenze hinweg summiert.

Falls eine Master-Syncposition auf einem in der Zukunft vom Master-Planar-Mover passierten Planar-Track angegeben werden soll, so ist darauf zu achten, dass der Master-Planar-Mover zum Zeitpunkt des Abschickens des GearInPosOnTrackWithMasterMover [▶ 151]-Befehls bereits eine Bewegung kommandiert hat, die diesen Planar-Track beinhaltet.

6.2.10 Beispiel „Planar-Tracks zu Netzwerk verbinden“

Anhand dieser Anleitung werden Sie ein TwinCAT-Projekt anlegen, das vier Planar-Tracks zu einem Netzwerk verbindet.



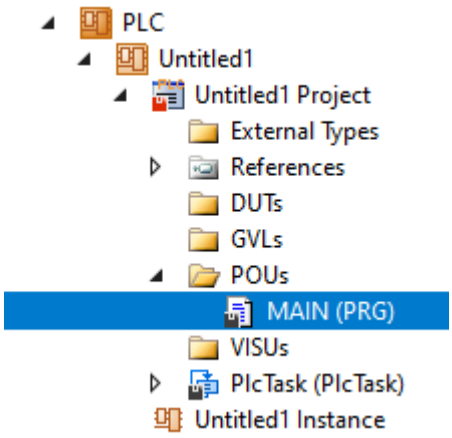
Planar-Track anlegen

1. Fügen Sie vier Planar-Tracks über **Groups > Add New Item…** hinzu, siehe Konfiguration [▶ 43].

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** die gewünschte Anzahl an Movern („MC_PlanarMover“) und Tracks („MC_PlanarTrack“) an.



⇨ Diese repräsentieren Mover und Tracks in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, vier Tracks sowie eine Zustandsvariable für eine Zustandsmaschine und zwei Hilfspositionen für die Tracks an.

```
PROGRAM MAIN
VAR
    track_one, track_two, track_three, track_four : MC_PlanarTrack;
    state : UDINT;
    pos1, pos2 : PositionXYC;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

⇨ Dieser Programmcode erstellt und aktiviert vier Tracks, die zu einem Netzwerk verbunden sind, wie in obiger Abbildung gezeigt. Die sogenannten „Verschleifungen“, also alle nicht-linearen Stücke der Tracks in diesem Beispiel, werden hier automatisch generiert. Sie geben nur die geraden Teilstücke vor.

```
CASE state OF
  0:
    pos1.SetValuesXY(250, 120);
    pos2.SetValuesXY(650, 120);
    track_one.AppendLine(0, pos1, pos2);
    pos1.SetValuesXY(700, 170);
    pos2.SetValuesXY(800, 450);
    track_one.AppendLine(0, pos1, pos2);
    pos1.SetValuesXY(650, 500);
    pos2.SetValuesXY(250, 500);
    track_one.AppendLine(0, pos1, pos2);
    state := 1;
  1:
    pos1.SetValuesXY(200, 450);
    pos2.SetValuesXY(200, 170);
    track_two.StartFromTrack(0,track_one);
    track_two.AppendLine(0, pos1, pos2);
    track_two.EndAtTrack(0,track_one);
    state := 2;
  2:
    pos1.SetValuesXY(200, 500);
    pos2.SetValuesXY(120, 500);
    track_three.StartFromTrack(0,track_one);
    track_three.AppendLine(0, pos1, pos2);
    state := 3;
  3:
    pos1.SetValuesXY(200, 550);
    pos2.SetValuesXY(200, 750);
    track_four.StartFromTrack(0,track_one);
    track_four.AppendLine(0, pos1, pos2);
    state := 4;
  4:
    track_one.Enable(0);
    track_two.Enable(0);
    track_three.Enable(0);
    track_four.Enable(0);
    state := 5;
  5:
    IF track_one.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled AND
    track_two.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled AND
    track_three.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled AND
    track_four.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
```

```
      state := 6;
    END_IF
END_CASE
```

Tracks müssen an allen Punkten C^2-stetig sein. Das heißt, dass ihre Positionen, Richtungen, sowie Krümmungen nahtlos ineinander übergehen müssen. Die automatisch generierten Verschleifungen berücksichtigen diese Vorgabe. Auch wenn die Eckstücke wie Viertelkreise aussehen, sind sie keine, da Kreise an jedem Punkt eine positive (konstante) Krümmung haben und Geraden die Krümmung Null aufweisen.

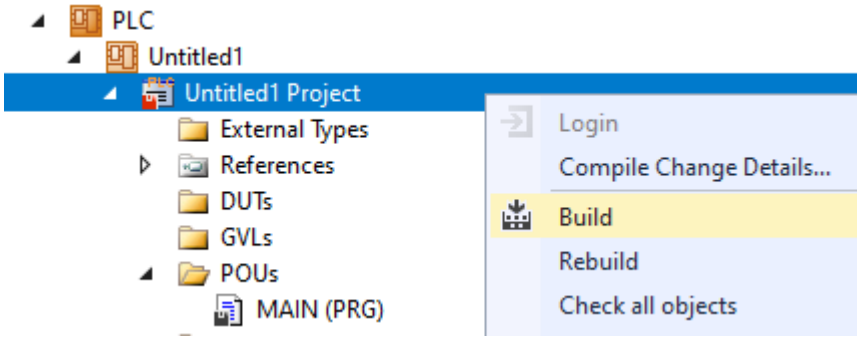
Befehl abschicken

4. Um den Befehl auch abzuschicken, müssen Sie die Tracks nach dem END_CASE zyklisch mit ihrer Update-Methode triggern:

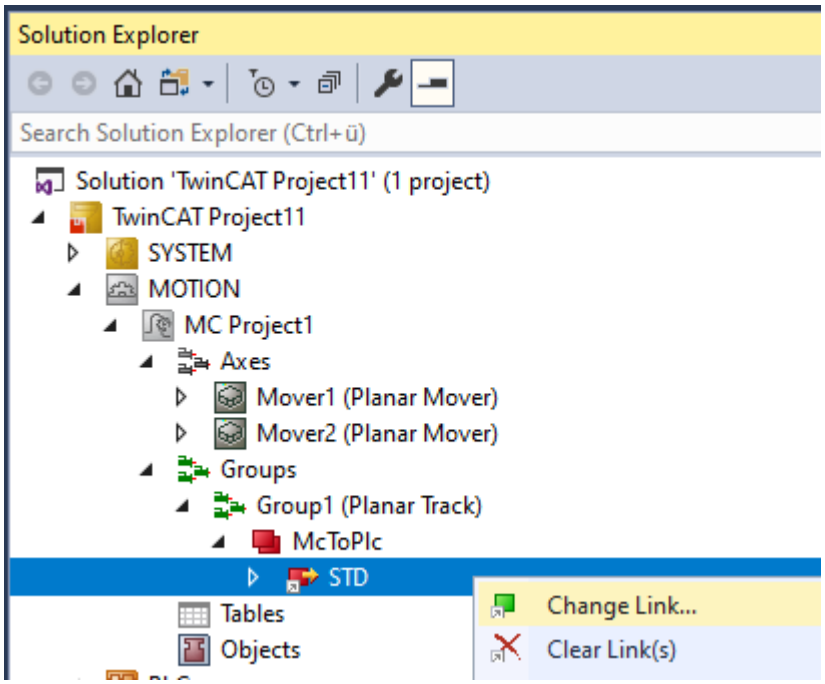
```
track_one.Update();
track_two.Update();
track_three.Update();
track_four.Update();
```

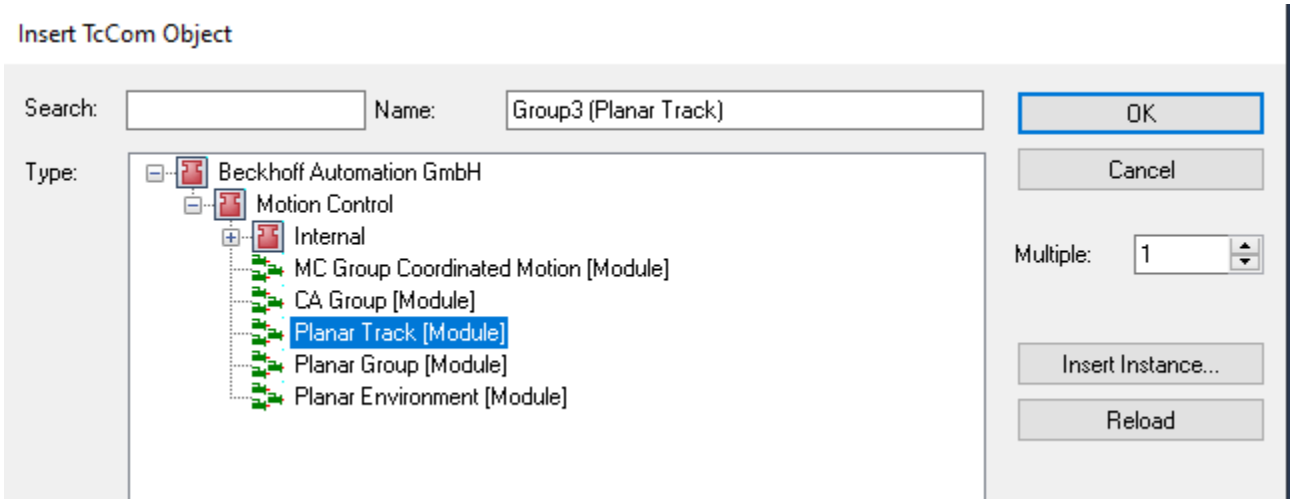
Durch das Bauen der PLC werden Symbole des „PLC-Movers" und „-Tracks“ erzeugt, welche anschließend mit der Mover- bzw. Track-Instanz im MC-Projekt verknüpft werden können.

1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**



⇨ Die Tracks müssen jeweils separat über die folgenden Dialogfenster verlinkt werden.





Projekt aktivieren und starten

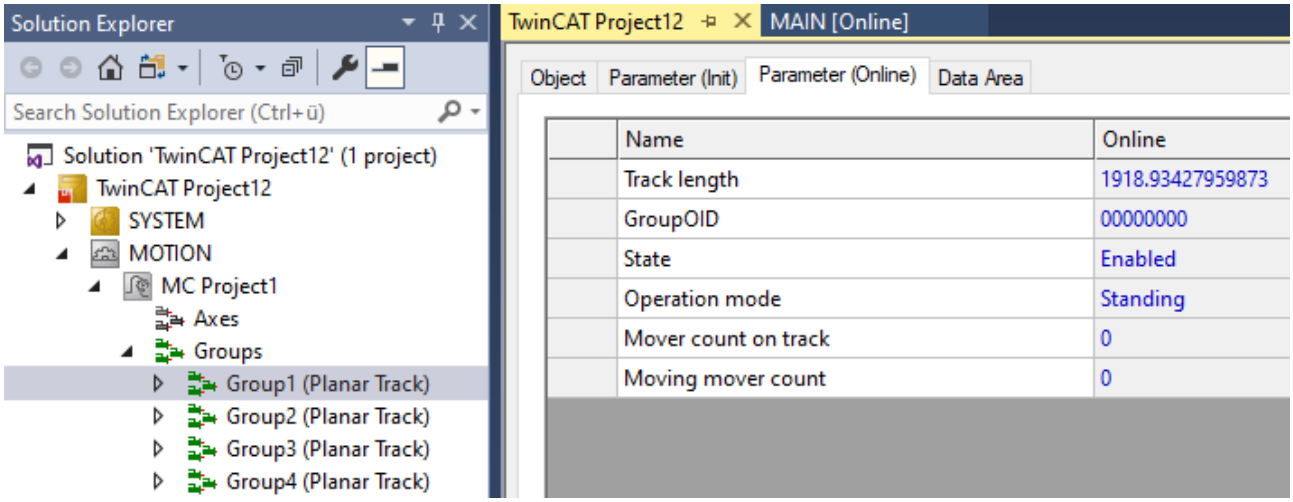
1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Das Track-Netzwerk ist am Ende des Zustandsautomaten (state = 6) fertig erstellt.

TwinCAT_Project12.Untitled1.MAIN

Expression	Type	Value	Prepared value	Address	Comment
track_one	MC_PlanarTrack				
MCTOPLC_STD	CDT_MCTOPLC_PLA...			%I*	Track data that is tran...rred from the Planar...
TrackOID	OTCID	16#05120010		%IB*	Object id of the planar track.
GroupOID	OTCID	16#00000000		%IB*	Object id of the planar group the track is in.
State	MC_PLANAR_STATE	Enabled		%IB*	State of the planar track, e.g. disabled.
OperationMode	MC_PLANAR_TRACK...	Standing		%IB*	Operation mode of the...nar track, e.g. movi...
MoverCountOnTrack	UDINT	0		%IB*	Number of movers coupled to this track.
MovingMoverCount	UDINT	0		%IB*	Number of movers ha... a movement planne...
Error	BOOL	FALSE			Flag indicating a PlanarTrack error.
ErrorId	UDINT	0			Error id indicating the PlanarTrack error type.
track_two	MC_PlanarTrack				
MCTOPLC_STD	CDT_MCTOPLC_PLA...			%I*	Track data that is tran...rred from the Planar...
TrackOID	OTCID	16#05120010		%IB*	Object id of the planar track.
GroupOID	OTCID	16#00000000		%IB*	Object id of the planar group the track is in.
State	MC_PLANAR_STATE	Enabled		%IB*	State of the planar track, e.g. disabled.
OperationMode	MC_PLANAR_TRACK...	Standing		%IB*	Operation mode of the...nar track, e.g. movi...
MoverCountOnTrack	UDINT	0		%IB*	Number of movers coupled to this track.
MovingMoverCount	UDINT	0		%IB*	Number of movers ha... a movement planne...
Error	BOOL	FALSE			Flag indicating a PlanarTrack error.
ErrorId	UDINT	0			Error id indicating the PlanarTrack error type.
track_three	MC_PlanarTrack				
MCTOPLC_STD	CDT_MCTOPLC_PLA...			%I*	Track data that is tran...rred from the Planar...
TrackOID	OTCID	16#05120010		%IB*	Object id of the planar track.
GroupOID	OTCID	16#00000000		%IB*	Object id of the planar group the track is in.
State	MC_PLANAR_STATE	Enabled		%IB*	State of the planar track, e.g. disabled.
OperationMode	MC_PLANAR_TRACK...	Standing		%IB*	Operation mode of the...nar track, e.g. movi...
MoverCountOnTrack	UDINT	0		%IB*	Number of movers coupled to this track.
MovingMoverCount	UDINT	0		%IB*	Number of movers ha... a movement planne...
Error	BOOL	FALSE			Flag indicating a PlanarTrack error.
ErrorId	UDINT	0			Error id indicating the PlanarTrack error type.
track_four	MC_PlanarTrack				
MCTOPLC_STD	CDT_MCTOPLC_PLA...			%I*	Track data that is tran...rred from the Planar...
TrackOID	OTCID	16#05120010		%IB*	Object id of the planar track.
GroupOID	OTCID	16#00000000		%IB*	Object id of the planar group the track is in.
State	MC_PLANAR_STATE	Enabled		%IB*	State of the planar track, e.g. disabled.
OperationMode	MC_PLANAR_TRACK...	Standing		%IB*	Operation mode of the...nar track, e.g. movi...
MoverCountOnTrack	UDINT	0		%IB*	Number of movers coupled to this track.
MovingMoverCount	UDINT	0		%IB*	Number of movers ha... a movement planne...
Error	BOOL	FALSE			Flag indicating a PlanarTrack error.
ErrorId	UDINT	0			Error id indicating the PlanarTrack error type.
state	UDINT	6			

Die Länge der Tracks steht jeweils in den Online-Parametern der TCOM-Objekte im MC Project.

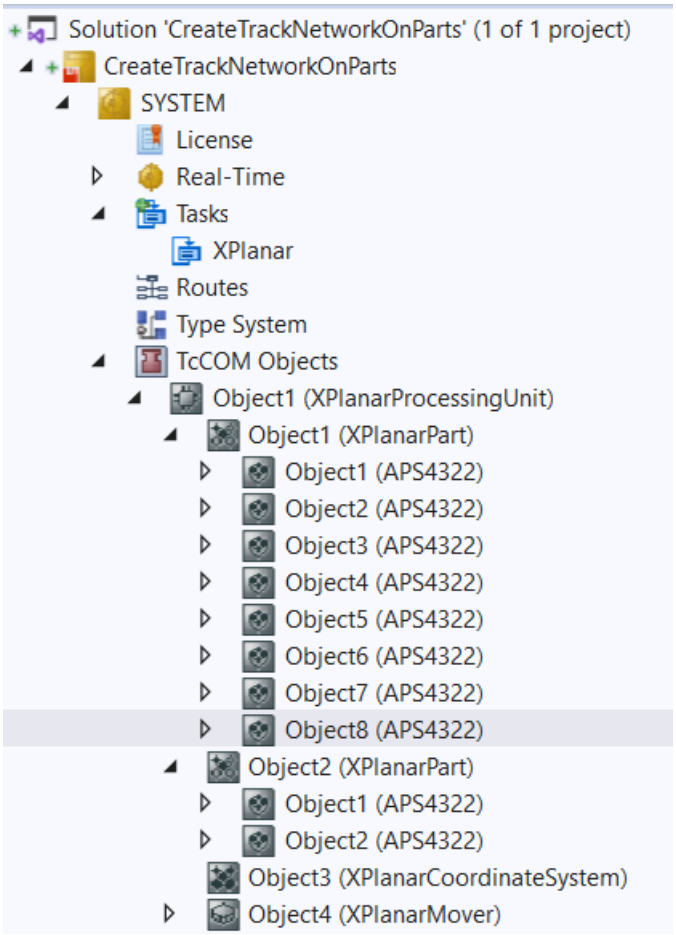


6.2.11 Beispiel „Planar-Tracks zu Netzwerk auf Planar-Parts verbinden"

In diesem Beispiel wird ein Netzwerk aus Planar-Tracks auf beweglichen Planar-Parts erstellt.

Ausgangspunkt

Wir starten mit einer Solution, die eine fertige konfigurierte XPlanar Processing Unit enthält. Unter der XPlanar Processing Unit sind zwei Parts, ein Koordinatensystem und ein Mover angelegt. Unter dem ersten Part sind 8 Kacheln angelegt und unter dem zweiten Part 2 Kacheln.



Folgende geometrische Situation ist eingestellt: die beiden Parts liegen nebeneinander und Part 2 kann zwei Positionen einnehmen. Damit existieren zwei verschiedene Verbindungen zwischen den Parts, abhängig von den Part Positionen.

Ausgehend von dieser Konfiguration wird das Beispiel entwickelt. Die Erstellung der Ausgangssituation wird in der Dokumentation der XPlanar Processing Unit beschrieben.

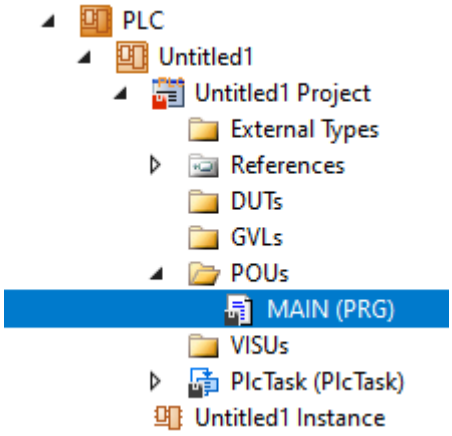
Planar-Tracks und Planar-Environment anlegen

1. Legen Sie eine Planar-Environment an, siehe Konfiguration [▶ 98].
2. Stellen Sie den Initialparameter XPlanar processing unit OID auf die Objekt Id der XPlanar Processing Unit. Damit ist das Part feature für alle **MC Configuration** Objekte aktiviert (besonders für den angelegten Planar-Mover).
3. Fügen Sie drei Planar-Tracks über **Groups > Add New Item…** hinzu, siehe Konfiguration [▶ 43].
4. Stellen sie den Initialparameter „PartOID“ der drei Tracks auf die entsprechenden Parts. In diesem Beispiel ist der erste und zweite Track auf Part 1 und der dritte Track auf Part 2.

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** die gewünschte Anzahl an Movern („MC_PlanarMover“) und Tracks („MC_PlanarTrack“) an.



⇨ Diese repräsentieren Mover und Tracks in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, drei Planar-Tracks, eine Zustandsvariable für eine Zustandsmaschine sowie zwei Hilfspositionen für die Tracks an.

```
PROGRAM MAIN
VAR
    track_one, track_two, track_three : MC_PlanarTrack;
    state : UDINT;
    pos1, pos2 : PositionXYC;
    part_one_oid : OTCID := 16#01010020;
    part_two_oid : OTCID := 16#01010030;
    start_options : ST_StartFromTrackAdvancedOptions;
    end_options : ST_EndAtTrackAdvancedOptions;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

⇨ Dieser Programmcode erstellt drei Tracks auf den Parts. Je nach aktiver Partposition sind die Tracks an verschiedenen Positionen verbunden. Zuerst wird Track 3 auf Part 2 erstellt (state=0). Danach wird Track 2 an Track 3 gestartet, wobei Part 2 in der oberen Position (index=2) ist. Track 2 endet auch an Track 3, allerdings ist Part 2 dafür in der unteren Position (index=1). Zuletzt wird Track 1 an Track 3 gestartet, Part 2 ist in der unteren Position (index=1), und beendet, Part 2 ist in der oberen Position (index=2).

```
CASE state OF
  0:
    pos1.SetValuesXYCReferenceId(0, 360, 0, part_two_oid);
    pos2.SetValuesXYCReferenceId(80, 360, 0, part_two_oid);
    track_three.AppendLine(0, pos1, pos2);
    pos1.SetValuesXYCReferenceId(120, 320, 0, part_two_oid);
    pos2.SetValuesXYCReferenceId(120, 160, 0, part_two_oid);
    track_three.AppendLine(0, pos1, pos2);
    pos1.SetValuesXYCReferenceId(80, 120, 0, part_two_oid);
    pos2.SetValuesXYCReferenceId(0, 120, 0, part_two_oid);
    track_three.AppendLine(0, pos1, pos2);
    state := 1;
  1:
    start_options.thisTrackPartPositionIndex := 1;
    start_options.otherTrackPartPositionIndex := 2;
    track_two.StartFromTrackAdvanced(0, track_three, start_options);
    pos1.SetValuesXYCReferenceId(440, 600, 0, part_one_oid);
    pos2.SetValuesXYCReferenceId(400, 600, 0, part_one_oid);
    track_two.AppendLine(0, pos1, pos2);
    pos1.SetValuesXYCReferenceId(360, 560, 0, part_one_oid);
    pos2.SetValuesXYCReferenceId(360, 400, 0, part_one_oid);
    track_two.AppendLine(0, pos1, pos2);
    pos1.SetValuesXYCReferenceId(400, 360, 0, part_one_oid);
    pos2.SetValuesXYCReferenceId(440, 360, 0, part_one_oid);
    track_two.AppendLine(0, pos1, pos2);
    end_options.thisTrackPartPositionIndex := 1;
    end_options.otherTrackPartPositionIndex := 1;
    track_two.EndAtTrackAdvanced(0, track_three, end_options);
    state := 2;
  2:
    start_options.thisTrackPartPositionIndex := 1;
    start_options.otherTrackPartPositionIndex := 1;
    track_one.StartFromTrackAdvanced(0, track_three, start_options);
    pos1.SetValuesXYCReferenceId(440, 120, 0, part_one_oid);
    pos2.SetValuesXYCReferenceId(160, 120, 0, part_one_oid);
    track_one.AppendLine(0, pos1, pos2);
```

```
    pos1.SetValuesXYCReferenceId(120, 160, 0, part_one_oid);
    pos2.SetValuesXYCReferenceId(120, 800, 0, part_one_oid);
    track_one.AppendLine(0, pos1, pos2);
    pos1.SetValuesXYCReferenceId(160, 840, 0, part_one_oid);
    pos2.SetValuesXYCReferenceId(440, 840, 0, part_one_oid);
    track_one.AppendLine(0, pos1, pos2);
    end_options.thisTrackPartPositionIndex := 1;
    end_options.otherTrackPartPositionIndex := 2;
    track_one.EndAtTrackAdvanced(0, track_three, end_options);
    state := 3;
END_CASE
```

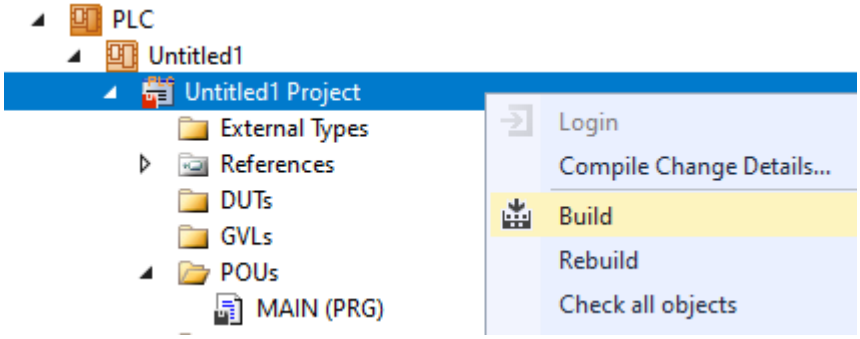
Befehl abschicken

4. Um die Befehle abzuschicken, müssen Sie die Tracks nach dem END_CASE zyklisch mit ihrer Update-Methode aufrufen:

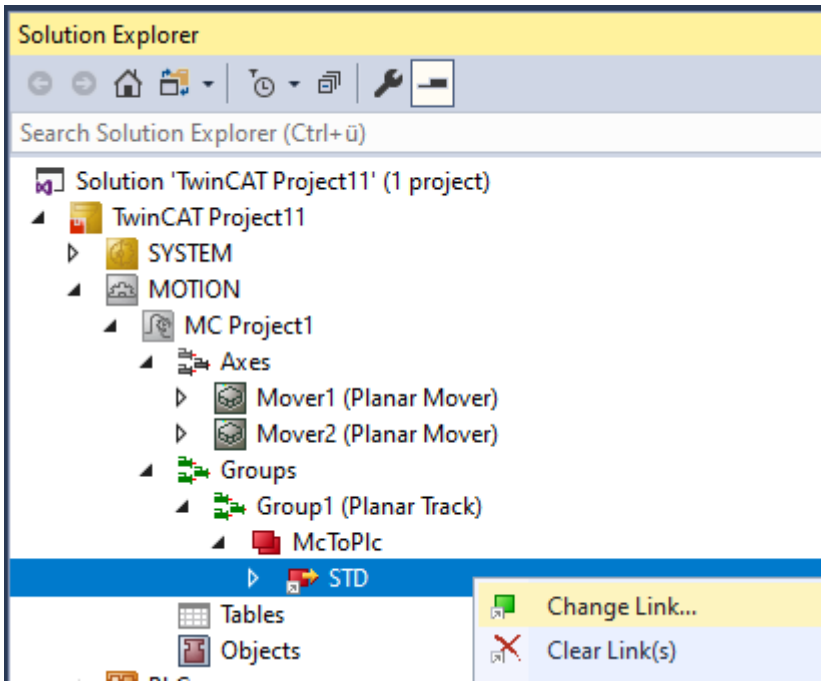
```
track_one.Update();
track_two.Update();
track_three.Update();
```

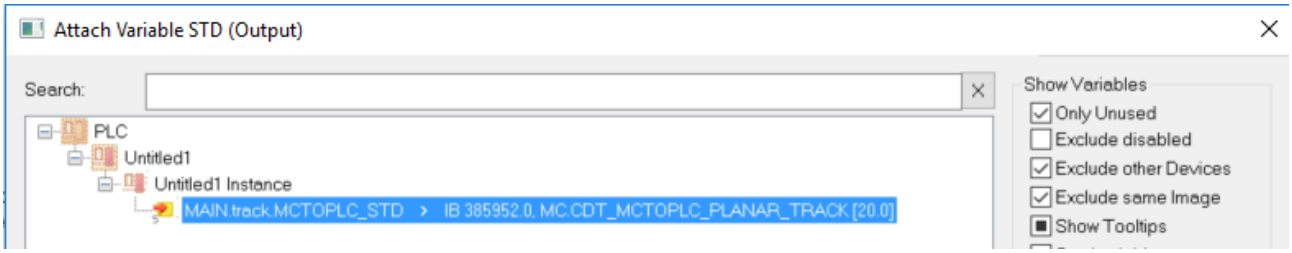
Durch das Bauen der PLC werden Symbole des „PLC-Movers" und „-Tracks“ erzeugt, welche anschließend mit der Mover- bzw. Track-Instanz im MC-Projekt verknüpft werden können.

1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**



⇨ Die Tracks können nun über die folgenden Dialogfenster verlinkt werden.

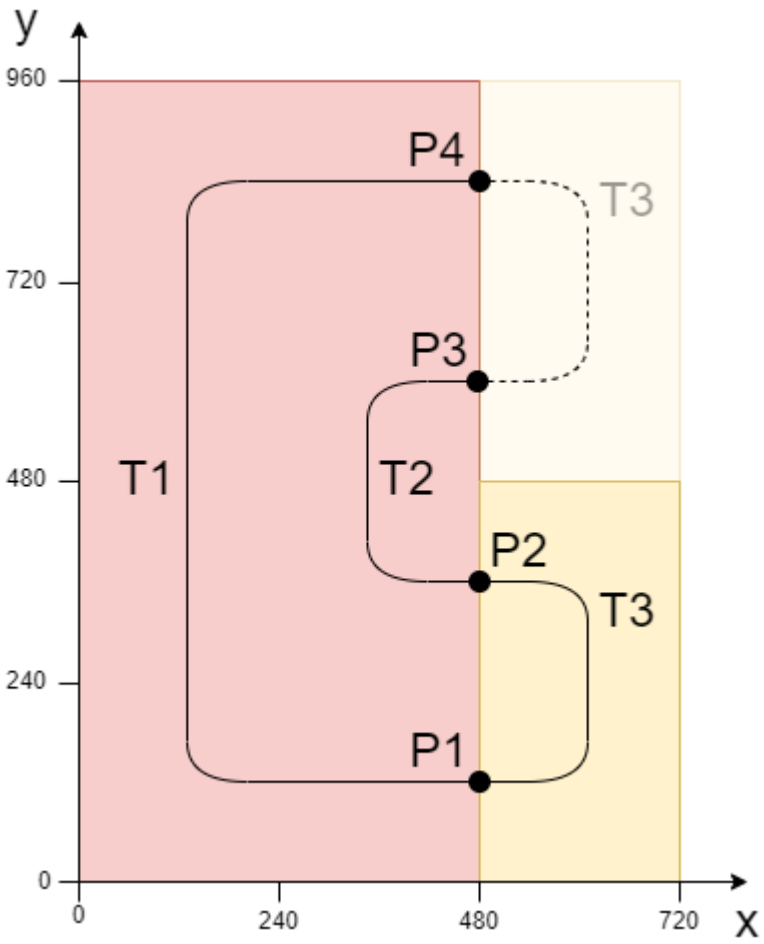




Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Am Ende des Zustandsautomaten (state=3) sind die drei Tracks wie abgebildet konfiguriert. Der Start von Track 3 ist mit dem Ende von Track 2 an Position 2 verbunden und das Ende von Track 3 mit dem Start von Track 1 and Position 2 verbunden, wenn Part 2 in der unteren Position ist. In der oberen Position ist der Start von Track 3 mit dem Ende von Track 1 an Position 4 verbunden und das Ende von Track 3 mit dem Start von Track 2 and Position 1 verbunden. Track 2 und Track 3 sind gleich lang.



6.2.12 Beispiel „Planar-Mover durch ein Track-Netzwerk folgen“

Anhand dieser Anleitung werden Sie ein TwinCAT-Projekt anlegen, in welchem ein Planar-Mover, der sich auf einem Planar-Track befindet, einem vorausfahrenden Planar-Mover auf demselben Planar-Track auf dessen Weg durch ein Tracknetzwerk folgt.

Das Folgen durch ein Tracknetzwerk wird realisiert durch das Kommando GearInPosOnTrackWithMasterMover [▶ 151], welches im Beispiel „Planar-Mover auf einem Track mit einem anderen Planar-Mover synchronisieren“ [▶ 73] genauer beschrieben wird. Das Erstellen und Aufbauen eines Netzwerks aus Planar-Tracks wird im Beispiel „Planar-Tracks zu Netzwerk verbinden“ [▶ 76] genauer erläutert. Das aktuelle Beispiel ist kurz gefasst und baut auf die o.g. Beispiele auf.

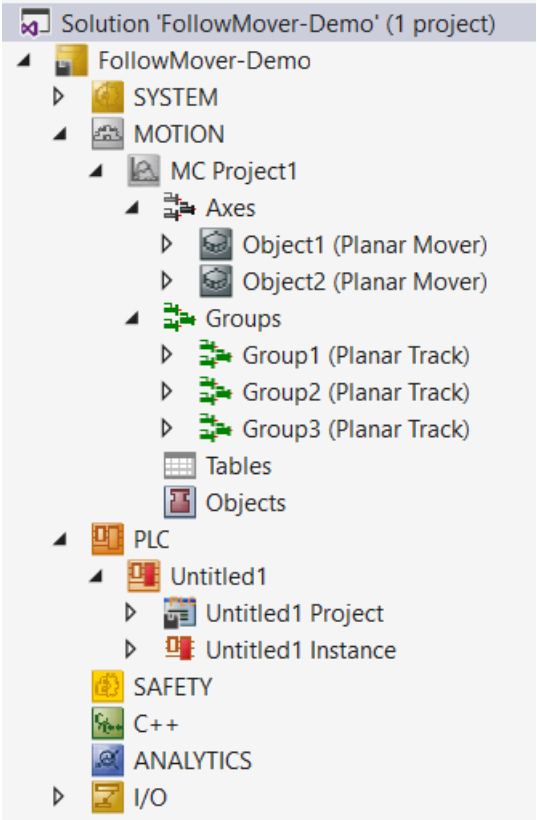
Planar-Mover anlegen

✓ Siehe Konfiguration [▶ 18].

1. Legen Sie zwei Planar-Mover an.
2. Versetzen Sie „Parameter (Init)“ in den Simulationsmodus (`TRUE`). Der Parameter ist versteckt und wird nur sichbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.

Planar-Track anlegen

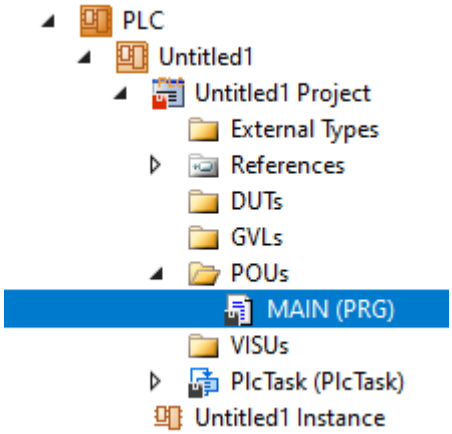
3. Fügen Sie drei Planar-Tracks über **Groups > Add New Item…** hinzu, siehe Konfiguration [▶ 43].
   ⇨ Der Solution Explorer weist die folgenden Einträge auf:



PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** die gewünschte Anzahl an Movern („MC_PlanarMover“) und Tracks („MC_PlanarTrack“) an.

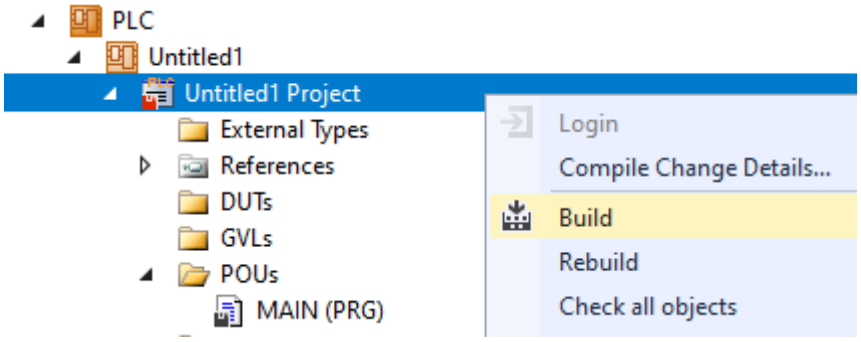


⇨ Diese repräsentieren Mover und Tracks in der MC Configuration.

2. Legen Sie die folgenden Variablen an.

```
PROGRAM MAIN
VAR
    master_mover  : MC_PlanarMover;
    slave_mover   : MC_PlanarMover;
    track_in      : MC_PlanarTrack;
    track_out1    : MC_PlanarTrack;
    track_out2    : MC_PlanarTrack;
    move_feedback : MC_PlanarFeedback;
    options       : ST_GearInPosOnTrackWithMasterMoverOptions;
    state         : UDINT;
    pos1, pos2    : PositionXYC;
END_VAR
```

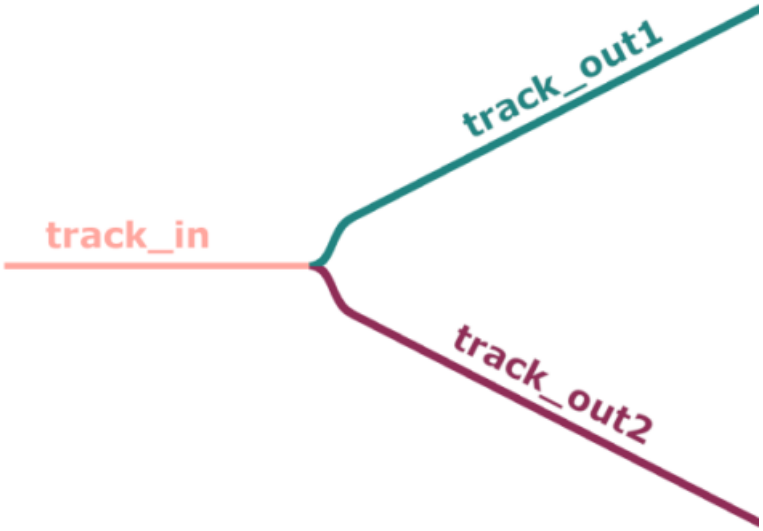
3. Bauen Sie die PLC um Symbole der „PLC-Mover“ und „-Tracks“ zu erzeugen.



4. Verlinken Sie die Planar-Mover und die Planar-Tracks (siehe Beispiel „Planar-Mover auf Track einkoppeln und verfahren“ [▶ 51]).

Zustandsautomaten programmieren

Mit der folgenden Zustandsmaschine, die im MAIN programmiert wird, werden zunächst die Planar-Tracks geometrisch definiert und aktiviert (States 0 bis 7), sodass sie die folgende Weichenkonfiguration abbilden:



In States 8 bis 19 werden die beiden Planar-Mover aktiviert, auf den Planar-Track vor der Weiche (`track_in`) eingekoppelt und an Position 200 (`master_mover`) bzw. 0 (`slave_mover`) verfahren. Daraufhin wird der Master-Planar-Mover auf Position 500 auf dem oberen der beiden abzweigenden Planar-Tracks (`track_out1`) kommandiert (State 20). Schlussendlich wird in State 21 der GearInPosOnTrackWithMasterMover [▶ 151]-Befehl an den Slave-Planar-Mover geschickt. Wie üblich werden nach der `END_CASE`-Anweisung zyklisch die Planar-Objekte aktualisiert.

```
CASE state OF
  0:
    pos1.SetValuesXYC(100, 360, 0);
    pos2.SetValuesXYC(400, 360, 0);
    track_in.AppendLine(0, pos1, pos2);
    track_in.Enable(0);
    state := state + 1;
  1:
    IF track_in.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
      state := state + 1;
    END_IF
  2:
    track_out1.StartFromTrack(0, track_in);
    state := state + 1;
  3:
    pos1.SetValuesXYC(450, 410, 0);
    pos2.SetValuesXYC(860, 620, 0);
    track_out1.AppendLine(0, pos1, pos2);
    track_out1.Enable(0);
    state := state + 1;
  4:
    IF track_out1.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
      state := state + 1;
    END_IF
  5:
    track_out2.StartFromTrack(0, track_in);
    state := state + 1;
  6:
    pos1.SetValuesXYC(450, 310, 0);
    pos2.SetValuesXYC(860, 100, 0);
    track_out2.AppendLine(0, pos1, pos2);
    track_out2.Enable(0);
    state := state + 1;
  7:
    IF track_out2.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
      state := state + 1;
    END_IF
  8:
    master_mover.Enable(0);
    state := state + 1;
  9:
    IF master_mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := state + 1;
    END_IF
  10:
    master_mover.JoinTrack(0, track_in, 0, 0);
    state := state + 1;
  11:
```

```
    IF master_mover.MCTOPLC.STD.CommandMode = MC_PLANAR_MOVER_COMMAND_MODE.OnTrack THEN
      state := state + 1;
    END_IF
  12:
    master_mover.MoveOnTrack(move_feedback, track_in, 200, 0, 0);
    state := state + 1;
  13:
    IF move_feedback.Done THEN
      state := state + 1;
    END_IF
  14:
    slave_mover.Enable(0);
    state := state + 1;
  15:
    IF slave_mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := state + 1;
    END_IF
  16:
    slave_mover.JoinTrack(0, track_in, 0, 0);
    state := state + 1;
  17:
    IF slave_mover.MCTOPLC.STD.CommandMode = MC_PLANAR_MOVER_COMMAND_MODE.OnTrack THEN
      state := state + 1;
    END_IF
  18:
    slave_mover.MoveOnTrack(move_feedback, track_in, 0, 0, 0);
    state := state + 1;
  19:
    IF move_feedback.Done THEN
      state := state + 1;
    END_IF
  20:
    master_mover.MoveOnTrack(0, track_out1, 500, 0, 0);
    state := state + 1;
  21:
    options.followMover := TRUE;
    slave_mover.GearInPosOnTrackWithMasterMover(0, master_mover, 0, 210, track_in, 10, track_in, 0,
options);
    state := state + 1;
END_CASE

master_mover.Update();
slave_mover.Update();
track_in.Update();
track_out1.Update();
track_out2.Update();
move_feedback.Update();
```

Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Der Master-Planar-Mover wird zur gegebenen Zielposition (hier 500) auf dem angegebenen Planar-Track verfahren und der Slave-Planar-Mover wird seiner Bewegung folgen. Die Positionen der Planar-Mover können im Online-View (über Klicken des Buttons ) verfolgt werden.

Da als SyncPositionen der beiden Planar-Mover im Funktionsaufruf in State 21 die Positionen 210 für den Master bzw. 10 für den Slave angegeben wurden, wird der Slave-Planar-Mover seinem Master in einem Abstand von 200 durch das Netzwerk folgen. Er kommt also bei Position 300 auf dem oberen der beiden abzweigenden Planar-Tracks (auf dem sich auch der Master-Planar-Mover befindet) zum Stehen, was im Online-View überprüft werden kann:

Expression	Type	Value
master_mover	MC_PlanarMover	
PLCTOMC	CDT_PLCTOMC...	
MCTOPLC	CDT_MCTOPLC...	
STD	REFERENCE TO...	
SET	REFERENCE TO...	
ACT	REFERENCE TO...	
COORDMODE	REFERENCE TO...	
SETONTRACK	REFERENCE TO...	
SetPos	LREAL	499.9999999...
SetVelo	LREAL	0
SetAcc	LREAL	0
SetJerk	LREAL	0
TrackOID	OTCID	16#05120020
Error	BOOL	FALSE
ErrorId	UDINT	0
slave_mover	MC_PlanarMover	
PLCTOMC	CDT_PLCTOMC...	
MCTOPLC	CDT_MCTOPLC...	
STD	REFERENCE TO...	
SET	REFERENCE TO...	
ACT	REFERENCE TO...	
COORDMODE	REFERENCE TO...	
SETONTRACK	REFERENCE TO...	
SetPos	LREAL	299.9999999...
SetVelo	LREAL	0
SetAcc	LREAL	0
SetJerk	LREAL	0
TrackOID	OTCID	16#05120020
Error	BOOL	FALSE
ErrorId	UDINT	0

Beachten Sie, dass durch das Setzen der Option „FollowMover“ im Options-Objekt und übergeben desselben im Funktionsaufruf in State 21 kein Planar-TrackTrail [▶ 122]-Objekt spezifiziert werden muss. Der Pfad durchs Netzwerk, den der Slave-Planar-Mover nehmen soll, muss also nicht explizit bestimmt werden, da er automatisch dem Master-Planar-Mover folgt und an der Weiche auf den korrekten Planar-Track abbiegt. Dieses Verhalten wird mit der gesetzten Option auch in einem größeren Netzwerk, in welchem sich der Master-Planar-Mover über mehrere Trackgrenzen hinweg bewegt, reproduziert.

6.2.13 Optionen für die „StartFromTrackAdvanced" und „EndAtTrackAdvanced" Befehle

Wie in der Einleitung beschrieben und in den Beispielen gezeigt, ist es nicht immer eindeutig, was ein StartFromTrack [▶ 167]- oder ein EndAtTrack [▶ 168]-Befehl bedeuten. Daher werden beide Befehle in „Advanced“ Varianten (StartFromTrackAdvanced [▶ 169], EndAtTrackAdvanced [▶ 169]) angeboten und durch ein Optionsargument erweitert. Diese Erweiterung ist auch in den Init-Parametern abgebildet.

Das Optionsargument hat drei Komponenten: „thisTrackPartPositionIndex“, „otherTrackPartPositionIndex“ and „linkOnlyInSpecifiedPartPositions“. Die erste Komponente „thisTrackPartPositionIndex“ gibt an, welche Position der Part einnimmt, auf dem der aufgerufene Track liegt. Dabei muss der eindeutige Index der Position angegeben werden. Die zweite Komponente hat dieselbe Bedeutung für den Track, der als Argument übergeben wird. Die dritte Komponente gibt an, ob beide Tracks nur in der angegeben Position verbunden werden, oder auch in allen anderen geometrisch kompatiblen Positionen.

Sind beide Indices null und die Flag `false`, so ist das Verhalten identisch zu den bisherigen Befehlen. Dies kann auch erreicht werden, indem einfach „0“ als Optionsargument der „Advanced“ Befehle übergeben wird.

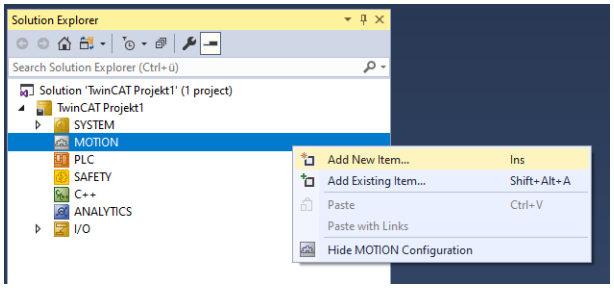
6.3 Planar-Group

Die Planar-Group ist ein Softwareobjekt, das auf der zweidimensionalen XPlanar Statorfläche Kollisionen zwischen Planar-Movern und von Planar-Movern mit dem Rand der Statorfläche verhindert. Dazu werden die 2D-Flächen aller Objekte in der Gruppe blockiert. Wenn ein Fahrauftrag an einen Mover übergeben wird, wird die dafür benötigte Fläche bei der Planar-Group angefragt und der Fahrauftrag abgelehnt, falls diese Fläche mit bereits reservierten Flächen kollidieren würde. Wenn der Fahrauftrag ausgeführt werden kann, wird die Fläche der Menge reservierter Flächen hinzugefügt und entsprechend blockiert.

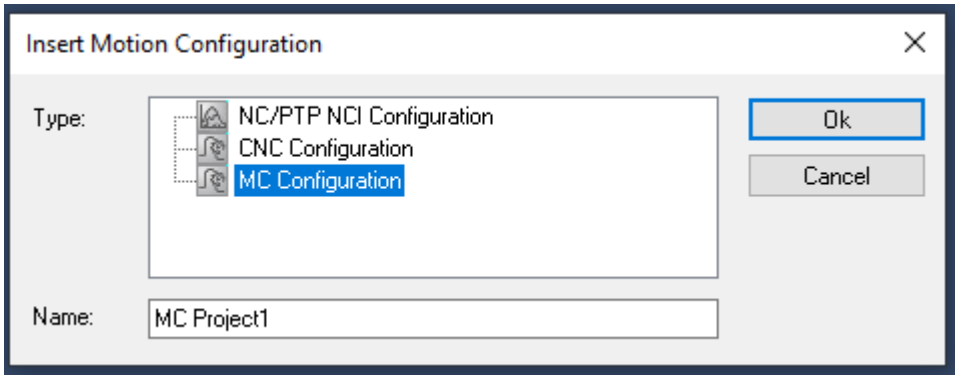
6.3.1 Konfiguration

✓ Um eine Planar-Group anzulegen, muss zunächst eine **MC Configuration** angelegt werden.

1. Wählen Sie **MOTION** > **Add New Item…** aus.

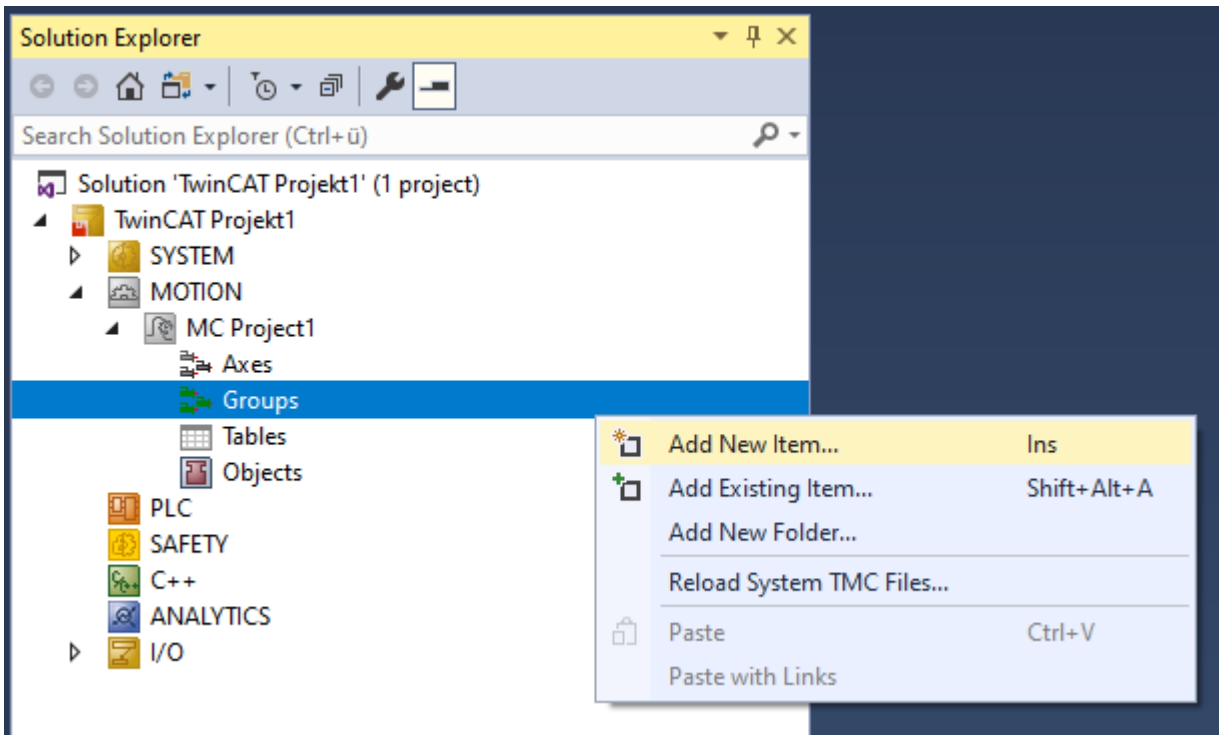


2. Wählen Sie im folgenden Dialogfenster **MC Configuration** aus und bestätigen Sie mit **OK**.

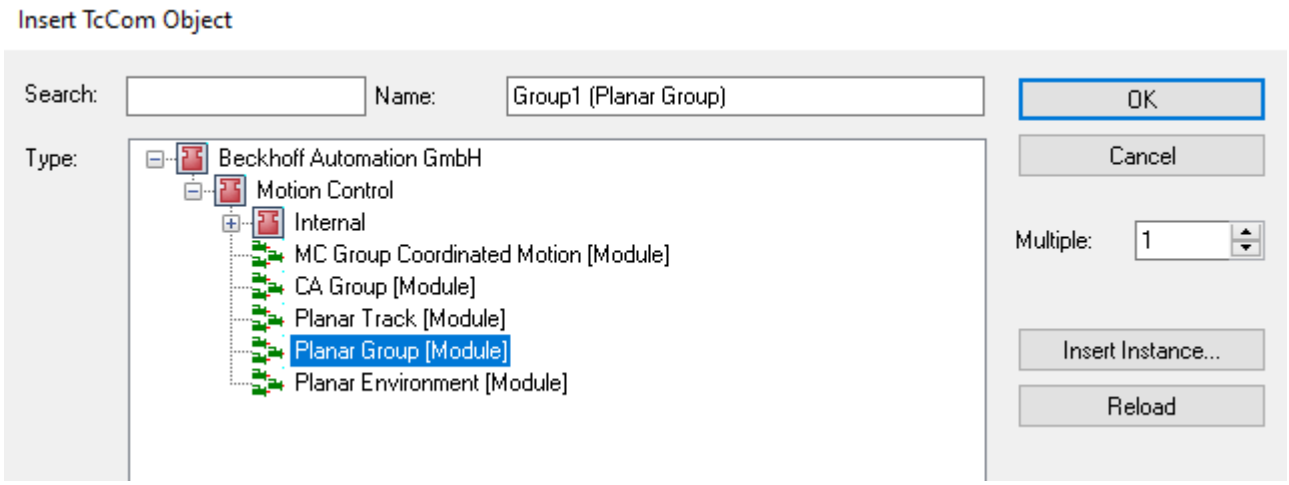


⇨ Sie haben ein MC Project angelegt.

3. Wählen Sie im erzeugten **MC Project** > **Groups** > **Add New Item…** aus.



4. Legen Sie im folgenden Dialogfenster eine (oder mehrere) Planar-Groups an und bestätigen Sie mit **OK**.



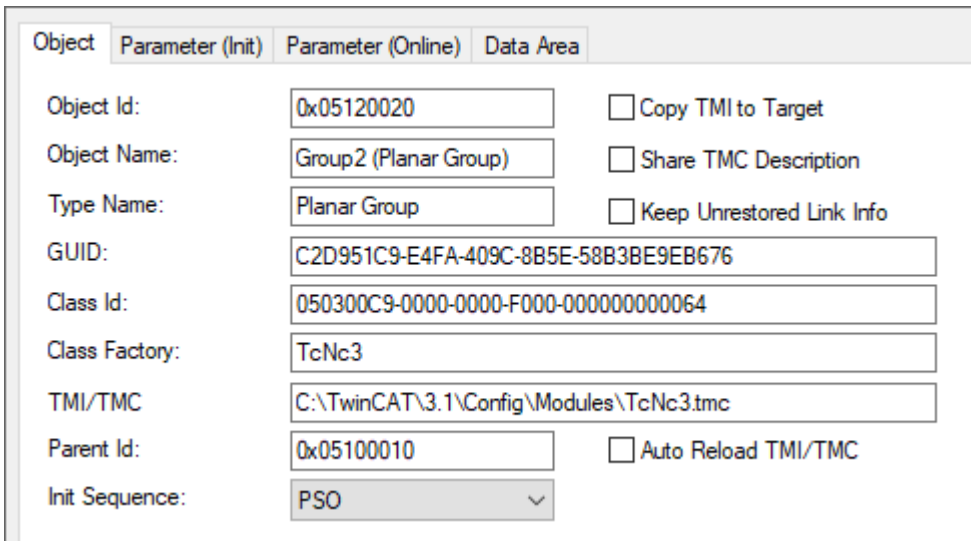
⇨ Die Planar-Group ist nun angelegt und kann parametriert werden.

Detailbeschreibung öffnen

- Planar-Group im Baum auswählen und doppelklicken.

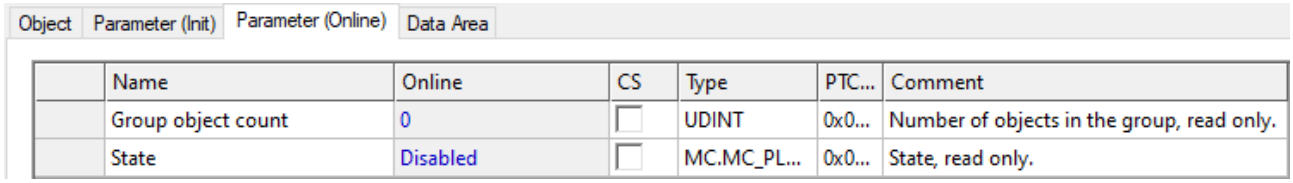
Bedeutung der einzelnen Reiter

Object: Hier werden allgemeine Informationen (Name, Typ, Id, usw.) dargestellt.



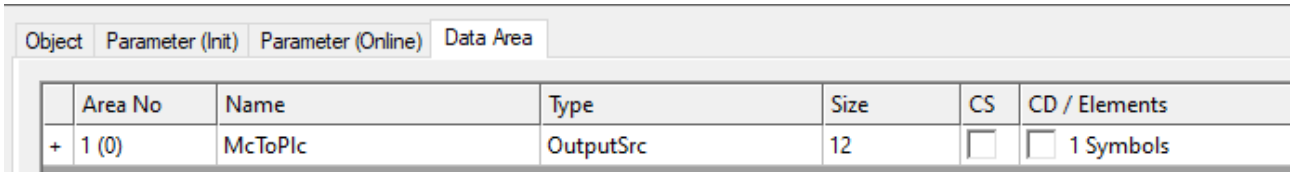
Parameter (Init): Die Gruppe hat keine Initialparameter.

Parameter (Online): Hier wird die Anzahl der in der Gruppe verwalteten Objekte (Mover, Tracks, Environment) angezeigt. Auch der Zustand der Gruppe wird dargestellt.



	Name	Online	CS	Type	PTC...	Comment
	Group object count	0	☐	UDINT	0x0...	Number of objects in the group, read only.
	State	Disabled	☐	MC.MC_PL...	0x0...	State, read only.

Data Area: Zeigt den Speicherbereich, über den die Gruppe mit dem PLC-Track kommuniziert.



	Area No	Name	Type	Size	CS	CD / Elements
+	1 (0)	McToPlc	OutputSrc	12	☐	☐ 1 Symbols

6.3.2 Beispiel "Planar-Mover mit Group anlegen und verfahren"

Anhand dieser Anleitung werden Sie ein TwinCAT-Projekt anlegen, das zwei Planar-Mover sowie eine Planar-Group enthält. Es werden beide Mover zur Gruppe hinzugefügt und verfahren.

Planar-Mover anlegen

✓ Siehe Konfiguration [▶ 18].

1. Legen Sie zwei Planar-Mover an.
2. Versetzen Sie „Parameter (Init)“ in den Simulationsmodus (`TRUE`). Der Parameter ist versteckt und wird nur sichbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.
3. Ändern Sie die Startposition des zweiten Movers auf x=240.

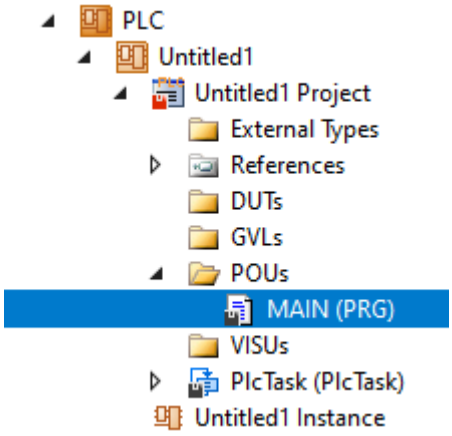
Planar-Group anlegen

4. Fügen Sie die Planar-Group über **Groups > Add New Item…** hinzu, siehe Konfiguration [▶ 91].

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** zwei Mover („MC_PlanarMover“) und eine Planar-Group „MC_PlanarGroup“ an.



⇨ Diese repräsentieren die Mover und die Gruppe in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, eine Zustandsvariable für eine Zustandsmaschine an sowie zwei Hilfspositionen für die MoveToPosition [▶ 147]-Befehle der Mover.

```
PROGRAM MAIN
VAR
    mover_one, mover_two : MC_PlanarMover;
    group : MC_PlanarGroup;
    state : UDINT;
    pos1, pos2 : PositionXYC;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

⇨ Dieser Programmcode aktiviert die Gruppe und beide Mover. Danach werden beide Mover zur Gruppe hinzugefügt.

```
CASE state OF
  0:
    mover_one.Enable(0);
    mover_two.Enable(0);
    state := 1;
  1:
    IF mover_one.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled
    AND mover_two.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 2;
    END_IF
  2:
     group.Enable(0);
     state := 3;
  3:
    IF group.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 4;
    END_IF
  4:
    mover_one.AddToGroup(0, group);
    mover_two.AddToGroup(0, group);
    state := 5;
  5:
    IF mover_one.MCTOPLC.STD.GroupOID = group.MCTOPLC_STD.GroupOID
    AND mover_two.MCTOPLC.STD.GroupOID = group.MCTOPLC_STD.GroupOID THEN
      state := 6;
    END_IF
  6:
    pos1.SetValuesXY(0, 240);
    pos2.SetValuesXY(0, 0);
    mover_one.MoveToPosition(0, pos1, 0, 0);
    mover_two.MoveToPosition(0, pos2, 0, 0);
    state := 7;
END_CASE
```

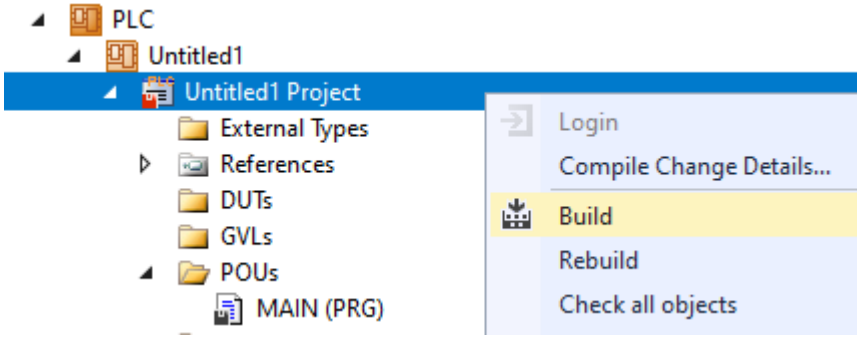
Befehl abschicken

4. Um den Befehl auch abzuschicken, müssen Sie die Mover und die Gruppe zyklisch mit den Update-Methoden triggern:

```
mover_one.Update();
mover_two.Update();
group.Update();
```

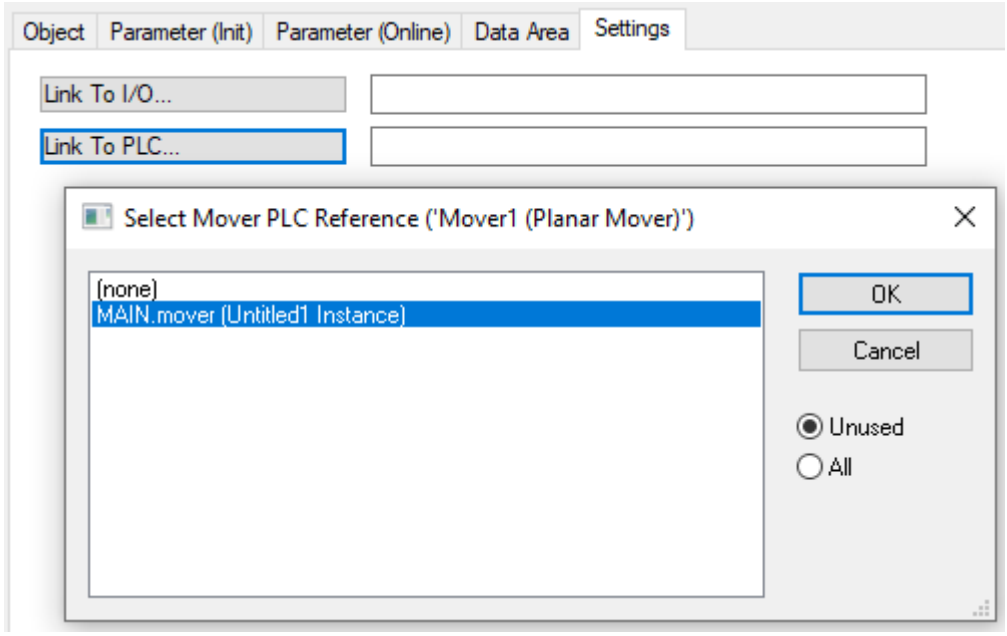
Durch das Bauen der PLC werden Symbole des „PLC-Movers" und der „-Gruppe“ erzeugt, welche anschließend mit der Mover- bzw. Group-Instanz im MC-Projekt verknüpft werden können.

5. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**

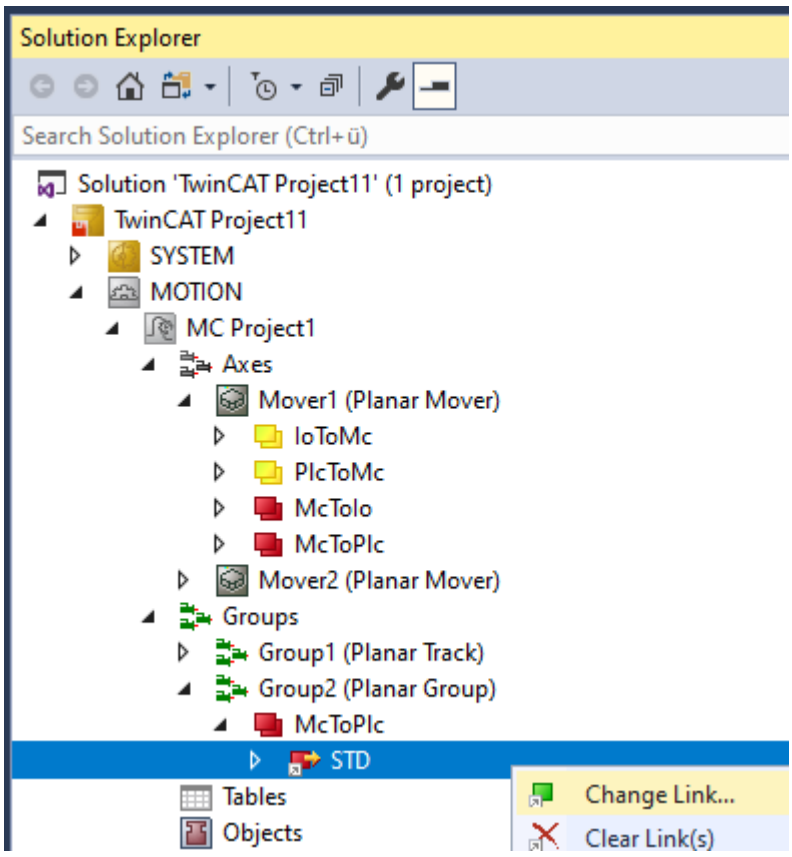


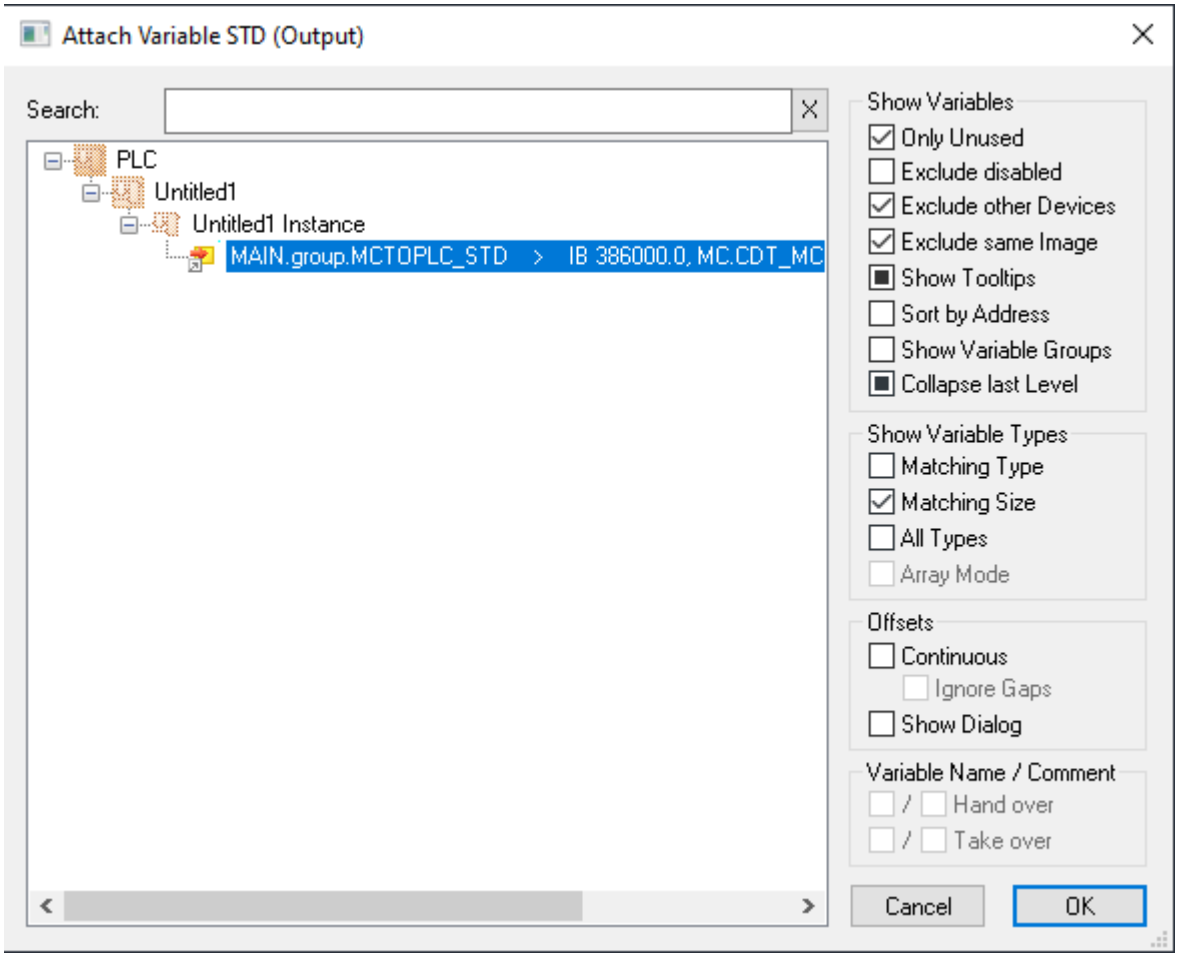
⇨ Anschließend können die Planar-Mover im „MC Project“ mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.

6. Doppelklicken Sie erst Mover eins, danach Mover zwei.



⇨ Die Gruppe muss über die folgenden Dialogfenster separat verlinkt werden.


Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'TwinCAT Project11' (1 project)
TwinCAT Project11
SYSTEM
MOTION
MC Project1
Axes
Mover1 (Planar Mover)
IoToMc
PlcToMc
McTolo
McToPlc
Mover2 (Planar Mover)
Groups
Group1 (Planar Track)
Group2 (Planar Group)
McToPlc
STD
Tables
Objects
Change Link...
Clear Link(s)


Attach Variable STD (Output)
Search:
PLC
Untitled1
Untitled1 Instance
MAIN.group.MCTOPLC_STD > IB 386000.0, MC.CDT_MC
Show Variables
Only Unused
Exclude disabled
Exclude other Devices
Exclude same Image
Show Tooltips
Sort by Address
Show Variable Groups
Collapse last Level
Show Variable Types
Matching Type
Matching Size
All Types
Array Mode
Offsets
Continuous
Ignore Gaps
Show Dialog
Variable Name / Comment
/ Hand over
/ Take over
Cancel
OK

Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Nach Einloggen in die PLC und Starten sehen Sie, dass die Mover am Ende des Zustandsautomaten (state=7) nicht beide auf den Zielpositionen stehen. Mover eins ist nach x=0 und y=240 gefahren. Mover zwei ist nicht zum Ursprung gefahren, da dort noch Mover eins stand und das Kommando daher abgelehnt wurde, weil beide in einer gemeinsamen Gruppe sind.

Da die Dynamik-Limits der Mover per Default recht hoch sind, kann unter Umständen die Änderung der Positionen nach Einloggen mit dem Auge schlecht nachzuverfolgen sein. Zu den Dynamik-Limits siehe Planar-Mover [▶ 18].

MAIN [Online] | TwinCAT Project14

TwinCAT_Project14.Untitled1.MAIN

Expression	Type	Value	Prepared value	Address	Comment
mover_one	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC_PLA...			%Q*	Mover data that is tra...rred from the Plana...
MCTOPLC	CDT_MCTOPLC_PLA...			%I*	Mover data that is tra...rred from the Plana...
STD	REFERENCE TO CDT...			%IB*	Mover standard data t...is transferred from ...
SET	REFERENCE TO CDT...			%IB*	Mover setpoint data th...is transferred from t...
SetPos	MoverVector				Current position.
x	LREAL	0			X coordinate.
y	LREAL	240			Y coordinate.
z	LREAL	0			Z coordinate.
a	LREAL	0			A coordinate.
b	LREAL	0			B coordinate.
c	LREAL	0			C coordinate.
SetVelo	MoverVector				Current velocity.
SetAcc	MoverVector				Current acceleration.
DcTimeStamp	ULINT	6630674363340...			Current time stamp.
PhysicalAreaID	UDINT	0			Current physical area id.
ACT	REFERENCE TO CDT...			%IB*	Mover actpoint data th...is transferred from t...
COORDMODE	REFERENCE TO CDT...			%IB*	Mover coordinate mod...ormation that is tra...
SETONTRACK	REFERENCE TO CDT...			%IB*	Mover busy informatio...at is transferred fro...
Error	BOOL	FALSE			Flag indicating a PlanarMover error.
ErrorId	UDINT	0			Error id indicating the PlanarMover error type.
mover_two	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC_PLA...			%Q*	Mover data that is tra...rred from the Plana...
MCTOPLC	CDT_MCTOPLC_PLA...			%I*	Mover data that is tra...rred from the Plana...
STD	REFERENCE TO CDT...			%IB*	Mover standard data t...is transferred from ...
SET	REFERENCE TO CDT...			%IB*	Mover setpoint data th...is transferred from t...
SetPos	MoverVector				Current position.
x	LREAL	240			X coordinate.
y	LREAL	0			Y coordinate.
z	LREAL	0			Z coordinate.
a	LREAL	0			A coordinate.
b	LREAL	0			B coordinate.
c	LREAL	0			C coordinate.
SetVelo	MoverVector				Current velocity.
SetAcc	MoverVector				Current acceleration.
DcTimeStamp	ULINT	6630674363340...			Current time stamp.
PhysicalAreaID	UDINT	0			Current physical area id.
ACT	REFERENCE TO CDT...			%IB*	Mover actpoint data th...is transferred from t...
COORDMODE	REFERENCE TO CDT...			%IB*	Mover coordinate mod...ormation that is tra...
SETONTRACK	REFERENCE TO CDT...			%IB*	Mover busy informatio...at is transferred fro...

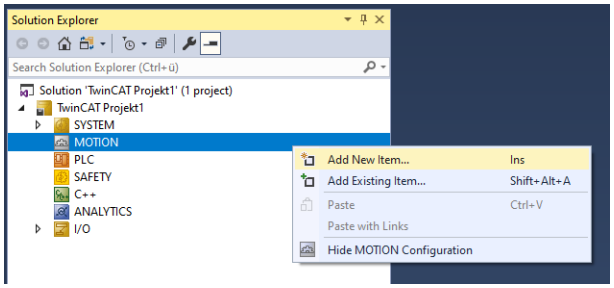
6.4 Planar-Environment

Die Planar-Environment ist ein Softwareobjekt, das die zweidimensionalen XPlanar Statorfläche repräsentiert. Zusammen mit Planar-Movern in einer Gruppe verhindert es Kollisionen der Mover mit dem Rand der Fläche.

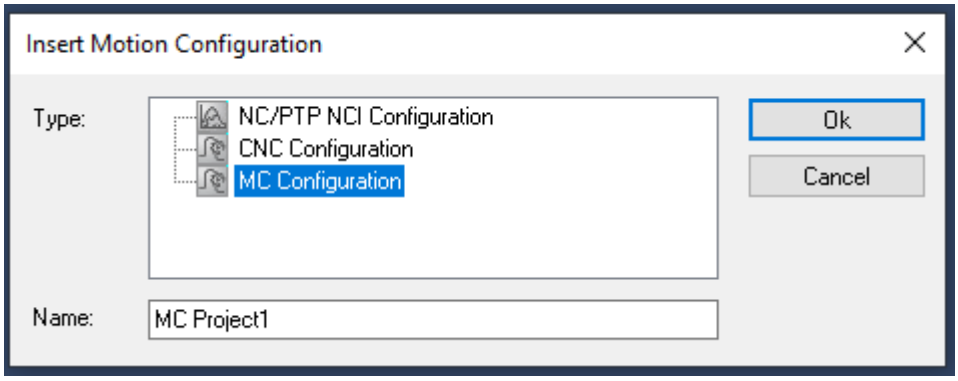
6.4.1 Konfiguration

✓ Um eine Planar-Environment anzulegen, muss zunächst eine **MC Configuration** angelegt werden.

1. Wählen Sie **MOTION** > **Add New Item…** aus.

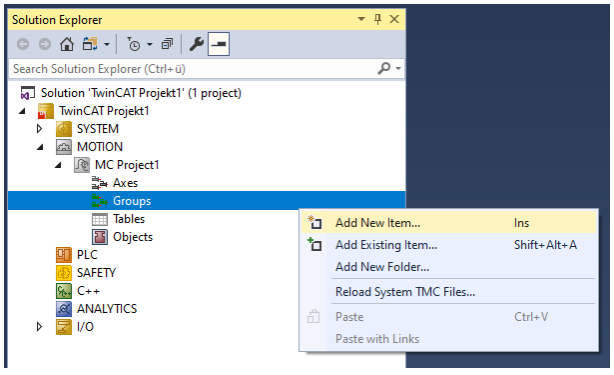


2. Wählen Sie im folgenden Dialogfenster **MC Configuration** aus und bestätigen Sie mit **OK.**

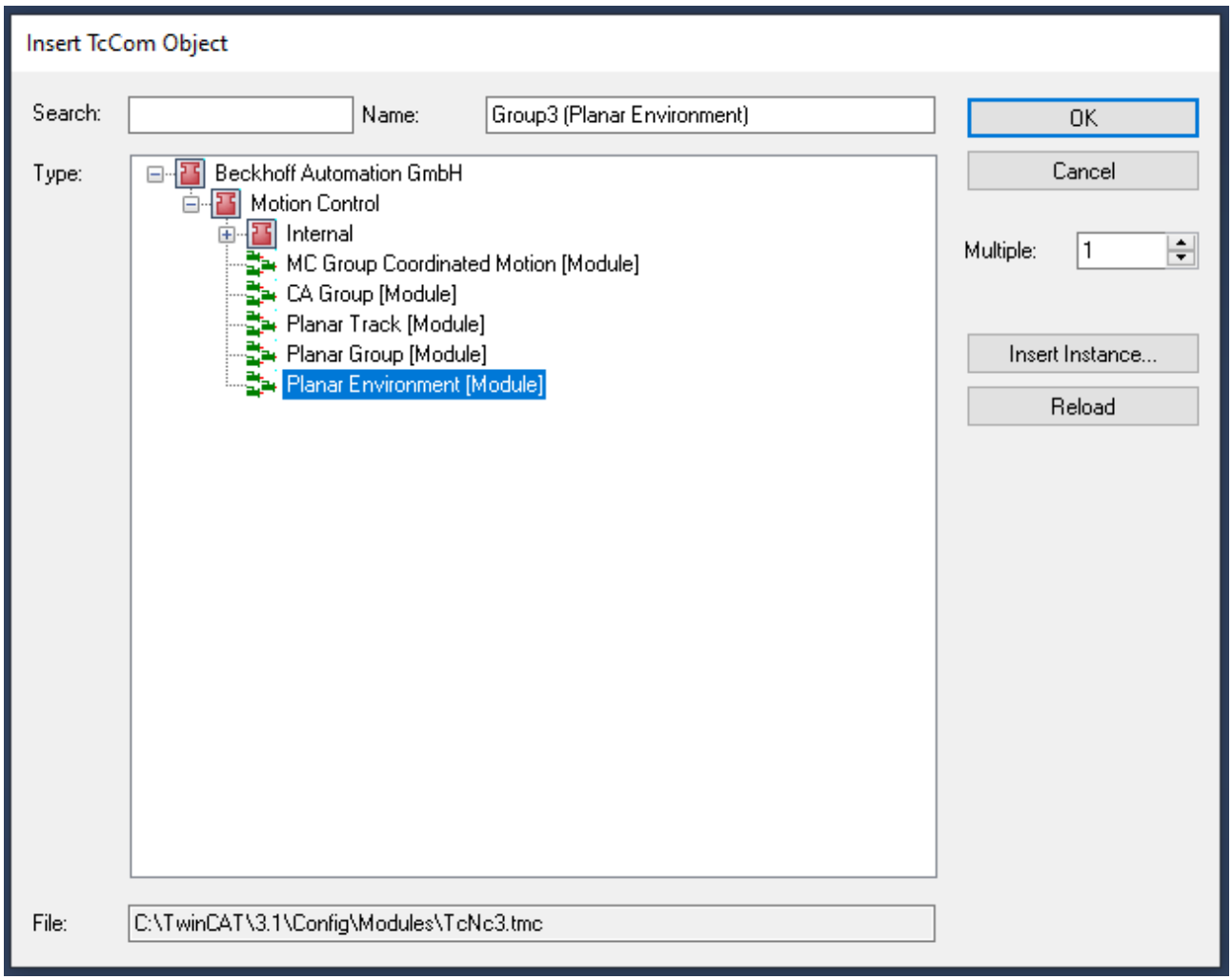


⇨ Sie haben ein MC Project angelegt.

3. Wählen Sie im erzeugten **MC Project** > **Groups** > **Add New Item…** aus.



4. Legen Sie im folgenden Dialogfenster eine (oder mehrere) Planar-Environments an und bestätigen Sie mit **OK**.



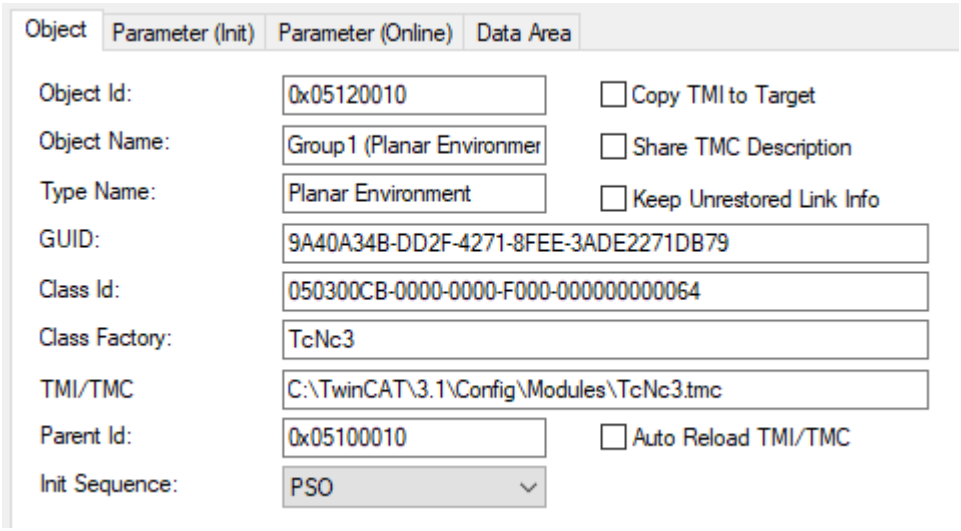
⇨ Die Planar-Environment ist nun angelegt und kann parametriert werden.

Detailbeschreibung öffnen

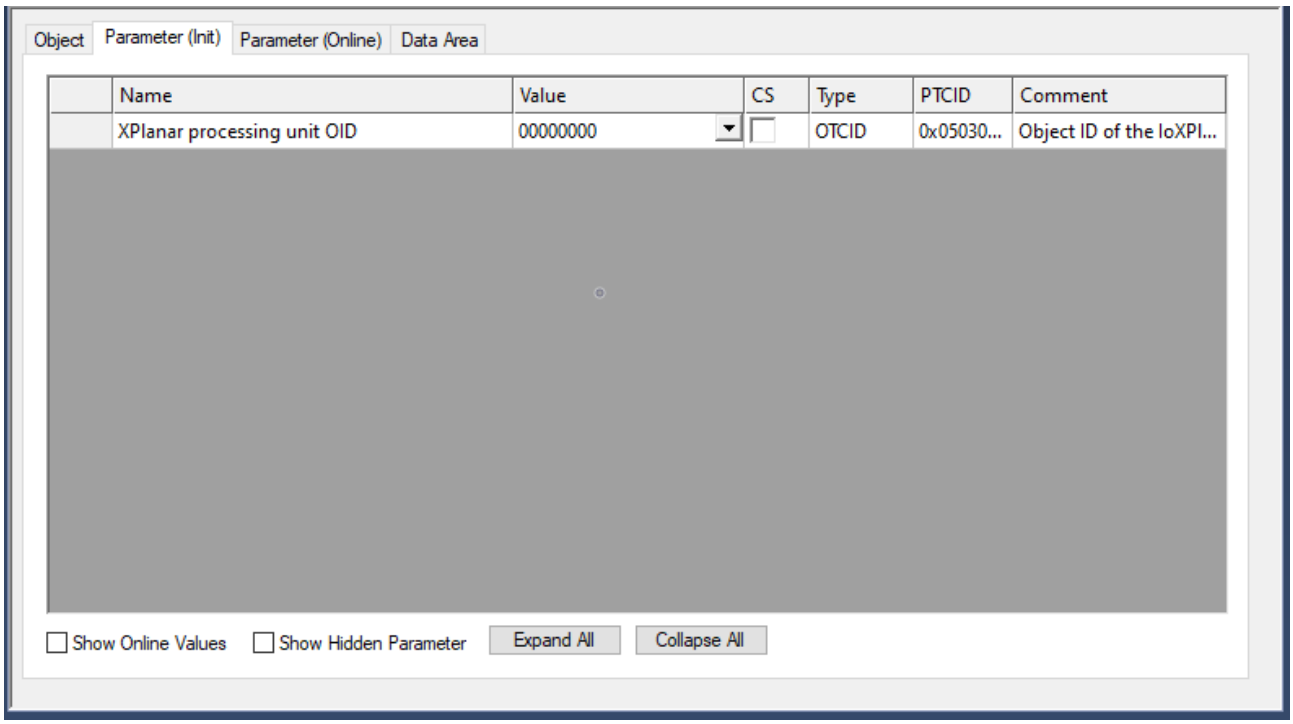
- Planar-Environment im Baum auswählen und doppelklicken.

Bedeutung der einzelnen Reiter

Object: Hier werden allgemeine Informationen (Name, Typ, Id, usw.) dargestellt.



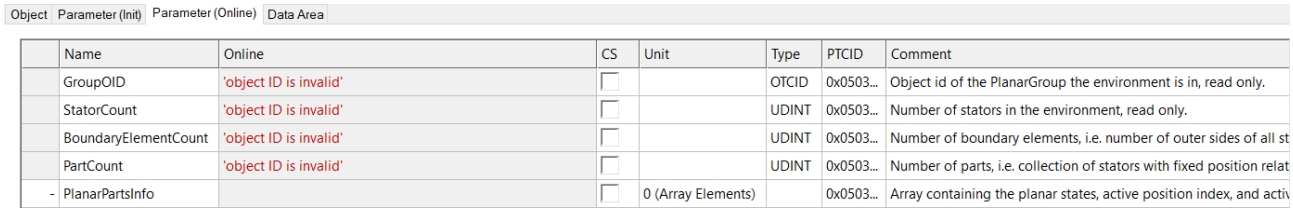
Parameter (Init): Gibt Initialparameter an, die der Nutzer verändern kann, um das Verhalten der Environment zu beeinflussen.



Die Environment hat den Initialparameter „XPlanar processing unit OID". Wenn dieser (>0) auf die Objekt Id der XPlanar Processing unit gesetzt wird, liest die Environment automatisch die Stator Konfiguration aus der XPlanar processing unit aus und generiert aus dieser Information die Randelemente für die Kollisionserkennung. Dies geschieht sobald der Nutzer den CreateBoundary() Befehl in der PLC aufruft.

Ab Version V3.2.60: Wenn der Parameter „XPlanar processing unit OID" auf die Objekt Id der XPlanar Processing unit gesetzt wird, liest die Environment zusätzlich die Partkonfiguration aus der XPlanar processing unit aus und generiert daraus eine interne Repräsentation aller Parts. Diese wird sowohl genutzt, um Kollisionschecks mit dem Rand der Parts durchzuführen, wenn die Environment in der Planar-Group ist, als auch um allen Komponenten (Mover, Tracks, Gruppe) eine vollständige Systembeschreibung bereitzustellen.

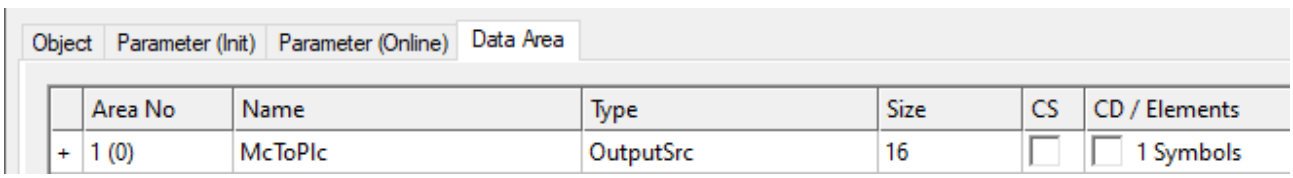
Parameter (Online): Zeigt den Zustand der Environment während der Laufzeit des Objektes.



Hier wird die Anzahl der in die Environment eingefügten Statoren und der daraus berechneten Randelemente angezeigt.

Ab Version V3.2.60: Der Parameter „PartCount" gibt die Anzahl der ausgelesenen und intern erstellten Parts an. Der Parameter „PlanarPartsInfo" zeigt für alle Parts Informationen an. Diese Informationen bestehen aus der Objekt Id des Parts, dem Planar State des Parts, dem aktiven Positionsindex, der Position des Parts bestehend aus Objekt Id des Koordinatensystems und x/y Koordinaten, und dem „disableForced" Flag des Parts.

Data Area: Zeigt den Speicherbereich, über den die Gruppe mit der PLC-Environment kommuniziert.



6.4.2 Beispiel „Statorfläche und Begrenzung konfigurieren“

Anhand dieser Anleitung werden Sie ein TwinCAT-Projekt anlegen, das eine Planar-Environment enthält und Sie werden deren Statorfläche und Begrenzung konfigurieren.

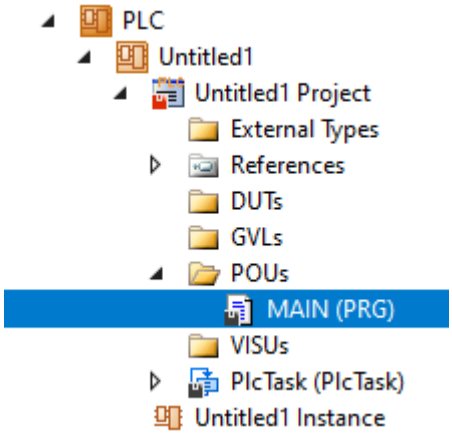
Planar-Environment anlegen

1. Legen Sie eine Planar-Environment an, siehe Konfiguration [▶ 98].

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** eine „MC_PlanarEnvironment“ an.



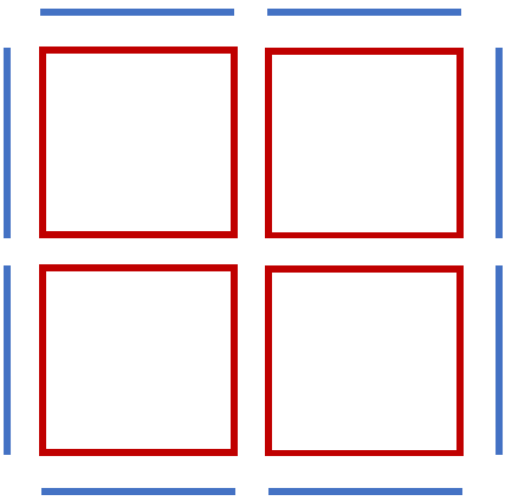
⇨ Diese repräsentiert die Environment in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, eine Zustandsvariable für eine Zustandsmaschine an.

```
PROGRAM MAIN
VAR
    environment : MC_PlanarEnvironment;
    state : UDINT;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

⇨ Dieser Programmcode fügt der Environment vier Statoren hinzu. Angegeben wird jeweils die untere linke Ecke der quadratischen Statoren (Seitenlänge 240 mm). Danach wird durch den CreateBoundary() die äußere Begrenzung der Statorfläche berechnet.
Die Statoren (rot) und die Begrenzungselemente (blau) sind schematisch im folgenden Bild dargestellt.



```
CASE state OF
  0:
    environment.AddStator(0,0.0,0.0);
    environment.AddStator(0,240.0,0.0);
    environment.AddStator(0,0.0,240.0);
    environment.AddStator(0,240.0,240.0);
    environment.CreateBoundary(0);
    state := 1;
END_CASE
```

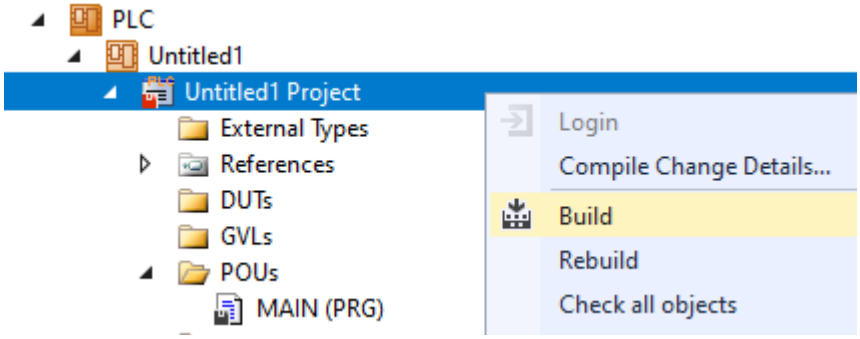
Befehl abschicken

4. Um den Befehl abzuschicken, müssen Sie die Environment nach dem END_CASE zyklisch mit ihrer Update-Methode aufrufen:

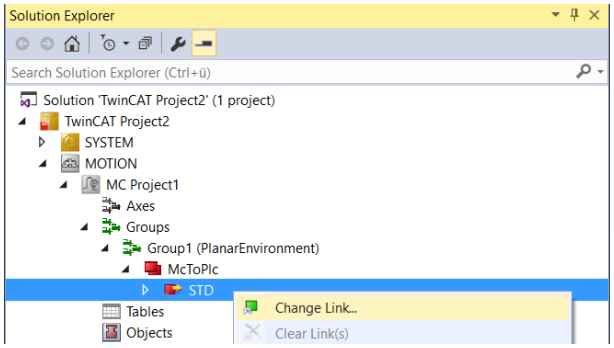
```
environment.Update();
```

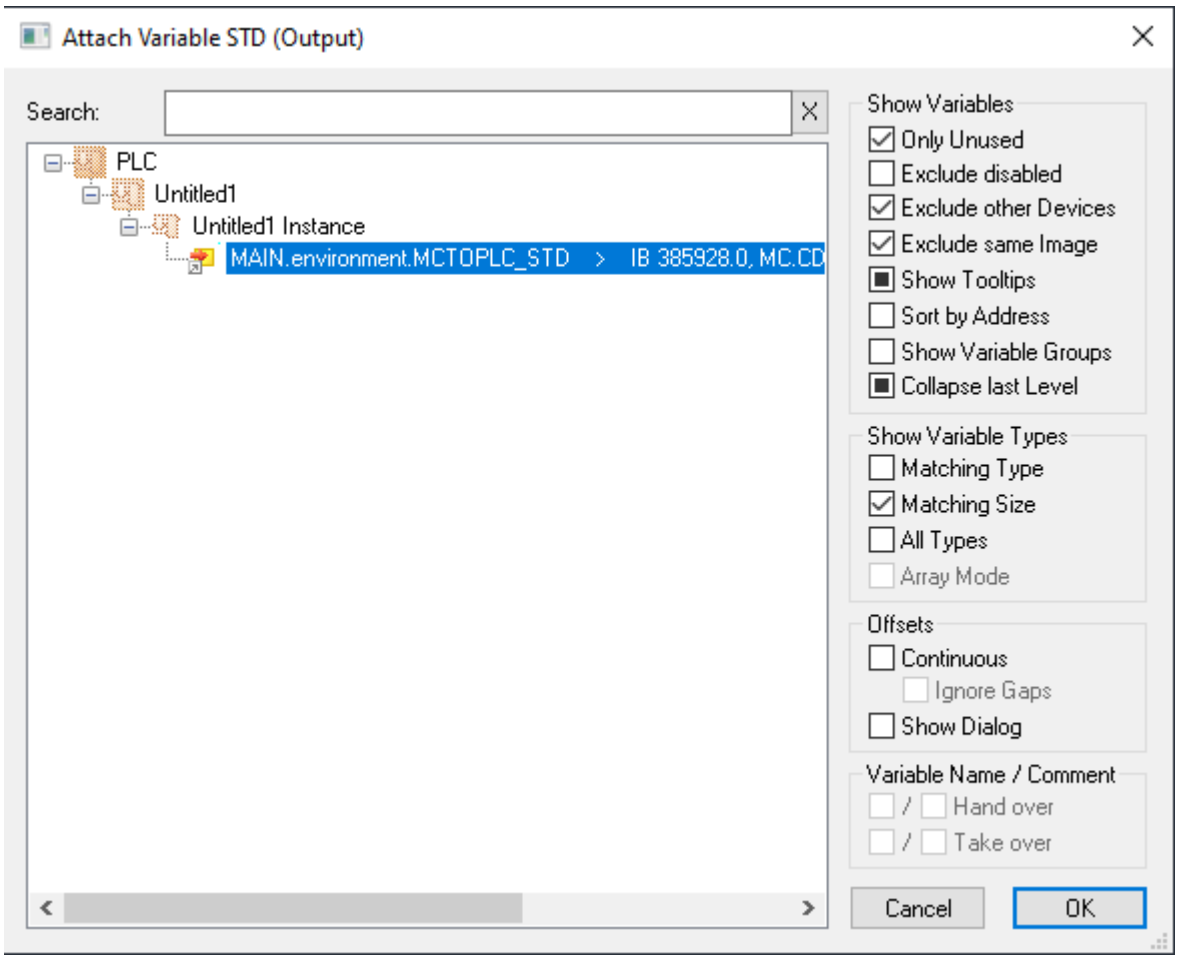
Durch das Bauen der PLC wird ein Symbol der „PLC-Environment“ erzeugt, welches anschließend mit der Planar-Environment im MC-Projekt verknüpft werden kann.

5. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**



⇨ Anschließend kann die Planar-Environment im „MC Project“ verlinkt werden.





Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Die Environment ist am Ende des Zustandsautomaten (state=1) im gewünschten Zustand.

MAIN [Online] | TwinCAT Project14

TwinCAT_Project14.Untitled1.MAIN

Expression	Type	Value	Prepared value	Address	Comment
environment	MC_PlanarEnvironment				
MCTOPLC_STD	CDT_MCTOPLC_PLA...			%I*	Environment data that...transferred from the...
EnvironmentOID	OTCID	16#05120010		%IB*	Object id of the planar environment.
GroupOID	OTCID	16#00000000		%IB*	Object id of the planar...up the environment...
StatorCount	UDINT	4		%IB*	Number of stators the environment is holding.
BoundaryElementCount	UDINT	8		%IB*	Number of outer sides of the stators that for...
Error	BOOL	FALSE			Flag indicating a PlanarEnvironment error.
ErrorId	UDINT	0			Error id indicating the...narEnvironment erro...
state	UDINT	1			

```
1  CASE state 1 OF
2      0:
3          environment.AddStator(0,0.0,0.0);
4          environment.AddStator(0,240.0,0.0);
5          environment.AddStator(0,0.0,240.0);
6          environment.AddStator(0,240.0,240.0);
7          environment.CreateBoundary(0);
8          state 1 := 1;
9  END_CASE
10
11 environment.Update();
12 RETURN
```

6.5 Beispiel „Planar-Mover mit Track und Gruppe anlegen und verfahren“

Anhand dieser Anleitung wird ein TwinCAT-Projekt angelegt, das zwei Planar-Mover, einen Planar-Track und eine Planar-Group enthält und die Mover sowohl auf als auch neben dem Track verfährt.

Planar-Mover anlegen

✓ Siehe Konfiguration [▶ 18].

1. Legen Sie zwei Planar-Mover an.
2. Versetzen Sie „Parameter (Init)“ in den Simulationsmodus (`TRUE`). Der Parameter ist versteckt und wird nur sichbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.
3. Ändern Sie die Startposition des zweiten Movers auf x=240.

Object | Parameter (Init) | Parameter (Online) | Data Area | Settings

Name	Value	CS	Unit	Type	PTCID	Comment
General						
Simulation mode	TRUE			BOOL	0x050...	Specifies if mo...
Mover width	155.0		mm	LREAL	0x050...	Width that is u...
Mover height	155.0		mm	LREAL	0x050...	Height that is u...
Initial position	...		mm, °		0x050...	Mover position...
.x	240.0			LREAL		X coordinate.
.y	0.0			LREAL		Y coordinate.
.z	0.0			LREAL		Z coordinate.
.a	0.0			LREAL		A coordinate.
.b	0.0			LREAL		B coordinate.
.c	0.0			LREAL		C coordinate.

Planar-Track und Planar-Group anlegen

4. Fügen Sie den Planar-Track über **Groups > Add New Item…** hinzu, siehe Konfiguration [▶ 43].
5. Gehen Sie für die Planar-Group analog vor.

PLC anlegen

✓ Um die Mover, den Track und die Gruppe zu steuern, muss eine PLC angelegt werden, aus der der Nutzer Befehle an den Mover geben kann, siehe PLC anlegen [▶ 21].

6. Legen Sie über MAIN zwei Mover (MC_PlanarMover [▶ 145]), einen MC_PlanarTrack [▶ 164] und eine MC_PlanarGroup [▶ 143] an.
   ⇨ Diese repräsentieren die Mover, den Track und die Gruppe in der MC Configuration.

7. Legen Sie, wie nachfolgend gezeigt, eine Zustandsvariable für eine Zustandsmaschine an sowie zwei Hilfspositionen für den Track.
8. Legen Sie außerdem ein Feedback an.

   ⇨ Dem Feedback können beliebige Kommandos mitgegeben werden. Es liefert Detailinformationen zur Kommandoausführung und über die Ausführungszeit.

```
PROGRAM MAIN
VAR
    mover_one, mover_two : MC_PlanarMover;
    track : MC_PlanarTrack;
    group : MC_PlanarGroup;
    state : UDINT;
    pos1, pos2 : PositionXYC;
    feedback : MC_PlanarFeedback;
END_VAR
```

9. Programmieren Sie anschließend in der MAIN einen Ablauf.

   ⇨ Dieser Programmcode erstellt und aktiviert einen Track, eine Gruppe und beide Mover. Sowohl die Mover als auch der Track werden der Gruppe hinzugefügt. Danach wird Mover eins auf dem Track eingekoppelt und verfahren. Beim Verfahren wird ein Feedback mitgegeben, über das wir die Ablehnung des Kommandos als Fehler zurückbekommen. Das Kommando wird abgelehnt, da Mover zwei den Track blockiert (Kollisionsfehler).

```
CASE state OF
  0:
    pos1.SetValuesXY(0, 0);
    pos2.SetValuesXY(400, 0);
    track.AppendLine(0, pos1, pos2);
    track.Enable(0);
    group.Enable(0);
    state := 1;
  1:
    IF track.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled
    AND group.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
         state := 2;
    END_IF
  2:
    mover_one.Enable(0);
    mover_two.Enable(0);
    state := 3;
  3:
    IF mover_one.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled
    AND mover_two.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
         state := 4;
    END_IF
  4:
    mover_one.AddToGroup(0,group);
    mover_two.AddToGroup(0,group);
    track.AddToGroup(0,group);
    state := 5;
  5:
    IF mover_one.MCTOPLC.STD.GroupOID > 0
    AND mover_two.MCTOPLC.STD.GroupOID > 0
    AND track.MCTOPLC_STD.GroupOID > 0 THEN
         state := 6;
    END_IF
  6:
    mover_one.JoinTrack(0, track, 0, 0);
    state := 7;
  7:
    IF mover_one.MCTOPLC.STD.CommandMode = MC_PLANAR_MOVER_COMMAND_MODE.OnTrack THEN
         state := 8;
    END_IF
  8:
    mover_one.MoveOnTrack(feedback, 0, 150.0, 0, 0);
    pos2.SetValuesXY(240, 320);
    mover_two.MoveToPosition(0, pos2, 0, 0);
    state := 9;
  9:
    IF NOT mover_two.MCTOPLC.STD.Busy.busyXYC THEN
         state := 10;
    END_IF
  10:
    mover_one.MoveOnTrack(0, 0, 150.0, 0, 0);
    state := 11;
  11:
    IF NOT mover_one.MCTOPLC.STD.Busy.busyXYC THEN
```

```
            state := 12;
        END_IF

END_CASE
```

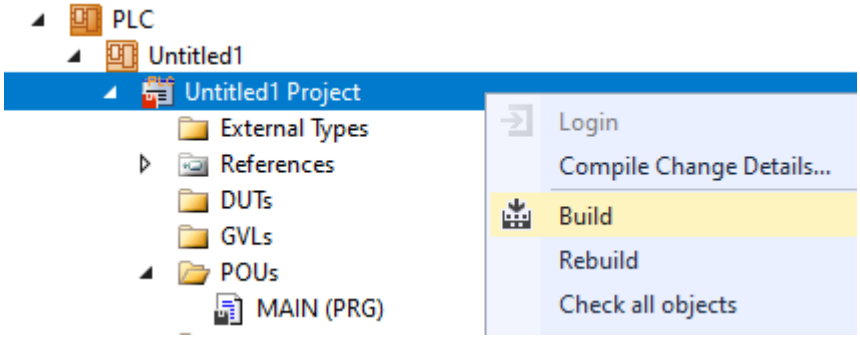
Befehl abschicken

10. Um den Befehl abzuschicken, müssen Sie die Mover, den Track und die Group nach dem END_CASE zyklisch mit seiner Update-Methode aufrufen:

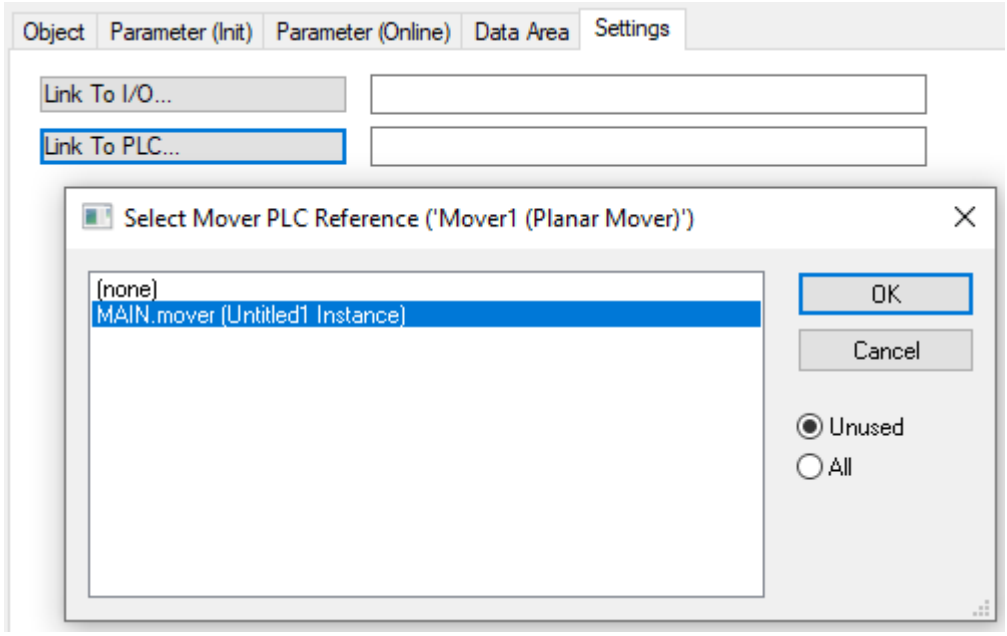
```
mover_one.Update();
mover_two.Update();
track.Update();
group.Update();
feedback.Update();
```

Durch das Bauen der PLC wird ein Symbol des „PLC-Movers" erzeugt, welches anschließend mit der Mover-Instanz im MC-Projekt verknüpft werden kann.

1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**



⇨ Anschließend kann der Planar-Mover im „MC Project“ (Doppelklick) mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.



Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Die Mover stehen am Ende des Zustandsautomaten (state=12) auf der gewünschten Position.

Das Feedback zeigt den Kollisionsfehler an. Zusätzlich wird bei Kollisionsfehlern im Feedback das blockierende Objekt mit seiner OID angezeigt. Es wäre nun möglich, nachdem Mover zwei aus dem Weg gefahren wurde, Mover eins auf dem Track zu verfahren.

MAIN [Online] | TwinCAT Project14

TwinCAT_Project14.Untitled1.MAIN

Expression	Type	Value	Prepared value	Address	Comment
mover_one	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC_PLA...			%Q*	Mover data that is tra...rred from the Plana...
MCTOPLC	CDT_MCTOPLC_PLA...			%I*	Mover data that is tra...rred from the Plana...
STD	REFERENCE TO CDT...			%IB*	Mover standard data t...is transferred from ...
SET	REFERENCE TO CDT...			%IB*	Mover setpoint data th...is transferred from t...
SetPos	MoverVector				Current position.
x	LREAL	0			X coordinate.
y	LREAL	240			Y coordinate.
z	LREAL	0			Z coordinate.
a	LREAL	0			A coordinate.
b	LREAL	0			B coordinate.
c	LREAL	0			C coordinate.
SetVelo	MoverVector				Current velocity.
SetAcc	MoverVector				Current acceleration.
DcTimeStamp	ULINT	6630674363340...			Current time stamp.
PhysicalAreaID	UDINT	0			Current physical area id.
ACT	REFERENCE TO CDT...			%IB*	Mover actpoint data th...is transferred from t...
COORDMODE	REFERENCE TO CDT...			%IB*	Mover coordinate mod...ormation that is tra...
SETONTRACK	REFERENCE TO CDT...			%IB*	Mover busy informatio...at is transferred fro...
Error	BOOL	FALSE			Flag indicating a PlanarMover error.
ErrorId	UDINT	0			Error id indicating the PlanarMover error type.
mover_two	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC_PLA...			%Q*	Mover data that is tra...rred from the Plana...
MCTOPLC	CDT_MCTOPLC_PLA...			%I*	Mover data that is tra...rred from the Plana...
STD	REFERENCE TO CDT...			%IB*	Mover standard data t...is transferred from ...
SET	REFERENCE TO CDT...			%IB*	Mover setpoint data th...is transferred from t...
SetPos	MoverVector				Current position.
x	LREAL	240			X coordinate.
y	LREAL	0			Y coordinate.
z	LREAL	0			Z coordinate.
a	LREAL	0			A coordinate.
b	LREAL	0			B coordinate.
c	LREAL	0			C coordinate.
SetVelo	MoverVector				Current velocity.
SetAcc	MoverVector				Current acceleration.
DcTimeStamp	ULINT	6630674363340...			Current time stamp.
PhysicalAreaID	UDINT	0			Current physical area id.
ACT	REFERENCE TO CDT...			%IB*	Mover actpoint data th...is transferred from t...
COORDMODE	REFERENCE TO CDT...			%IB*	Mover coordinate mod...ormation that is tra...
SETONTRACK	REFERENCE TO CDT...			%IB*	Mover busy informatio...at is transferred fro...

6.6 Planar-Part

Ab Version V3.2.60: Das Part Feature, um welches es in diesem Abschnitt geht, steht zur Verfügung.

Der MC_PlanarPart [▶ 160] ist ein PLC-Softwareobjekt, das den Part in der PLC repräsentiert. Es zeigt den Zustand des Parts an und bietet Methoden, um den Zustand zu ändern.

Die Verbindung des MC_PlanarPart [▶ 160]s wird indirekt über die MC_PlanarEnvironment [▶ 136] hergestellt. Dazu muss nach dem Starten der PLC die Initialize-Methode des MC_PlanarPart [▶ 160]s mit der korrekten Objekt-ID des Parts aufgerufen werden. Danach kann der Zustand des MC_PlanarPart [▶ 160]s sowohl gelesen als auch durch Methodenaufrufe verändert werden.

Mögliche Befehle für den MC_PlanarPart [▶ 160] sind die ActivatePosition [▶ 162]-Methode und die AllowEnable [▶ 162]-, ForceDisable [▶ 163]- sowie Reset [▶ 163]-Methoden. Die ActivatePosition [▶ 162]-Methode verschiebt den Part zu einer der möglichen Positionen. Die Methoden AllowEnable [▶ 162], ForceDisable [▶ 163] und Reset [▶ 163] verändern den PlanarState des MC_PlanarPart [▶ 160]s.

Nach dem Aufruf der AllowEnable [▶ 162]-Methode ist es dem PlanarPart erlaubt, die PlanarState Machine hochzufahren (bis zum CoE Zustand OperationEnabled). Sobald der erste Mover in einem Koordinatensystem mit einem Enable() Kommando aktiviert wird, werden alle Parts in diesem System aktiviert. Das Aktivieren des Parts, auf dem der Mover sich befindet, ist notwendig für das Aktivieren des Movers und erfolgt entsprechend zeitlich zuerst. Das Aktivieren der restlichen Parts in einem Koordinatensystem ist optional und darf fehlschlagen, ohne das Aktivieren zu verhindern bzw. abzubrechen. Umgekehrt führt das Aufrufen des Disable() Kommandos auf dem letzten aktiven Mover in einem Koordinatensystem dazu, dass alle Parts ausgeschaltet werden. Das Ausschalten der Parts ist nicht notwendig für das Ausschalten des Movers und erfolgt zeitlich nach dem Ausschalten des Movers.

Analog werden Parts aktiviert oder abgeschaltet, wenn der erste/letzte aktivierte Mover ein Koordinatensystem durch einen Aufruf der ActivatePosition [▶ 162]-Methode seines Parts betritt oder verlässt. Die Parts werden auch deaktiviert, wenn der letzte aktive Mover einen Fehler mit Fehlerreaktion „abort“ hat. Hat ein Mover einen Fehler mit der Fehlerreaktion „quick stop“, dann bleiben die Parts aktiviert. Dies gilt während des Fehlers und des anschließenden resets bis der Part wieder aktiviert ist. Sollte während des Fehlers oder des Reset [▶ 163]-Kommando durch ein Disable() Kommando des Movers die Fehlerreaktion auf „abort“ geändert werden, schalten sich die Parts aus.

Wird ein Part durch den Aufruf der ForceDisable [▶ 163]-Methode in den Disabled Zustand gezwungen so werden alle Mover auf diesem Part in den Fehlerzustand gebracht und andere Parts im Koordinatensystem eventuell ausgeschaltet, sollten sich keine weiteren aktiven Mover im Koordinatensystem befinden. Der Part bleibt im Disabled Zustand bis mindestens zum nächsten AllowEnable [▶ 162]-Aufruf.

Fehlerzustände des Parts erzwingen Fehler aller Mover auf diesem Part. Die Fehler der Mover werden zeitlich vor dem Part Fehler erreicht. Andere Mover oder Parts im Koordinatensystem werden durch diese Fehler nicht beeinflusst.

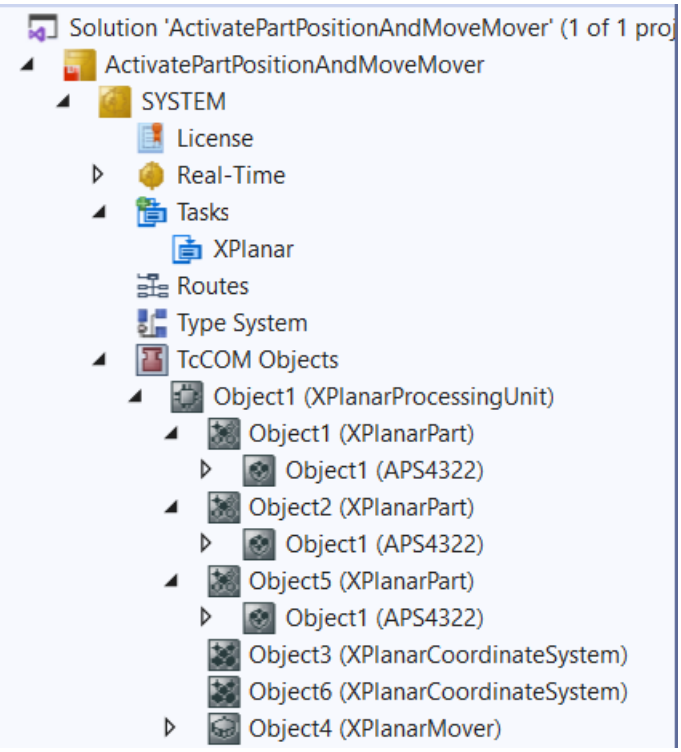
Im Fehlerfall eines Parts kann ein Reset [▶ 163]-Kommando angestoßen werden, um den Fehler zu beheben. Eventuelle Mover Fehler werden hiervon nicht berührt. Umgekehrt bewirkt ein Reset [▶ 163]-Kommando eines Movers ein Reset [▶ 163]-Kommando für alle Parts im Koordinatensystem mit einem Fehler. Das Reset [▶ 163]-Kommando des Movers hat als notwendige Bedingung die Fehlerfreiheit des eigenen Parts, entsprechend wird der Part zeitlich vor dem Mover resettet.

6.6.1 Beispiel „Aktiviere Planar-Part Position und verfahre Planar-Mover"

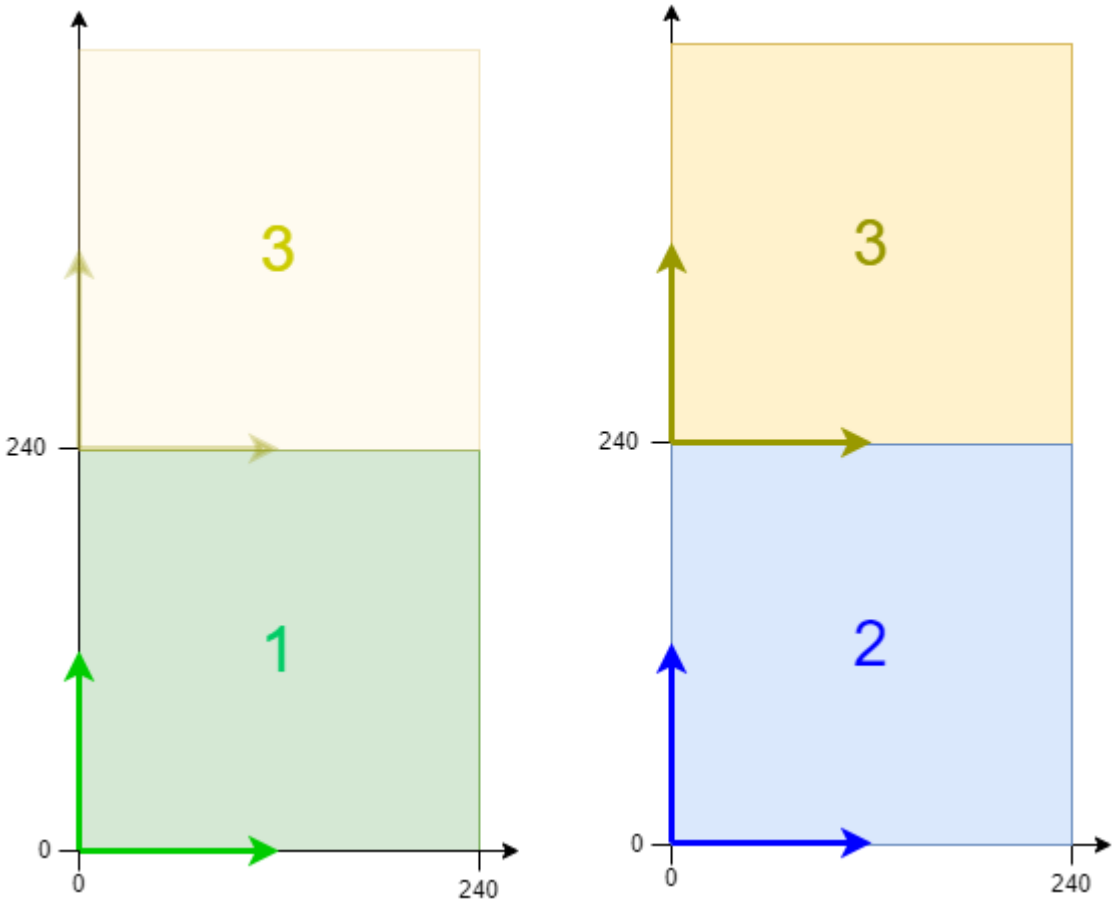
In diesem Beispiel wird ein Planar-Mover auf drei Planar-Parts verfahren, während der bewegliche der Parts verschoben wird.

Ausgangspunkt

Sie starten mit einer Solution, die eine fertig konfigurierte XPlanar Processing Unit enthält. Unter der XPlanar Processing Unit sind drei Parts, zwei Koordinatensysteme und ein Mover angelegt. Unter den drei Parts ist jeweils eine Kachel angelegt.



Folgende geometrische Situation ist eingestellt: die ersten beiden Parts liegen fest in den beiden Koordinatensystem im Ursprung und der dritte Part kann die Position zwischen beiden Koordinatensystemen wechseln. Es ist zum Beispiel denkbar, dass die beiden Koordinatensysteme übereinander in zwei Ebenen angeordnet sind und Part 3 ein Aufzug zwischen beiden Systemen ist. Der Mover startet in der Mitte des ersten Parts auf Koordinatensystem 1, während der dritte Part in Koordinatensystem 2 startet.



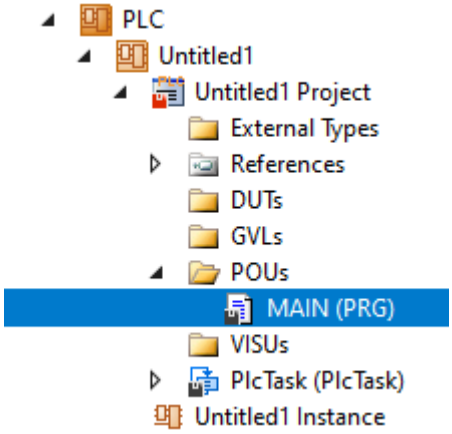
Planar-Mover und Planar-Environment anlegen

1. Legen Sie für dieses Beispiel einen Planar-Mover an, siehe Konfiguration [▶ 18].
2. Legen Sie eine Planar-Environment an, siehe Konfiguration [▶ 98].
3. Stellen Sie den Initialparameter XPlanar processing unit OID auf die Objekt Id der XPlanar Processing Unit. Damit ist das Part feature für alle **MC Configuration** Objekte aktiviert (besonders für den angelegten Planar-Mover).

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über **MAIN** den oder die Mover („MC_PlanarMover [▶ 145]“) wie folgt an.



⇨ Diese/r repräsentiert den/die Mover in der MC Configuration.

2. Legen Sie, wie nachfolgend gezeigt, einen Planar-Mover, eine Planar-Environment, einen Planar-Part, eine Zustandsvariable für eine Zustandsmaschine und eine Zielposition für die Fahrbefehle des Movers an.

```
PROGRAM MAIN
VAR
    mover : MC_PlanarMover;
    environment : MC_PlanarEnvironment;
    part_three : MC_PlanarPart;
    state : UDINT;
    target_position : PositionXYC;
END_VAR
```

3. Programmieren Sie anschließend in der MAIN einen Ablauf.

⇨ Dieser Programmcode initialisiert Part 3, aktiviert den Mover, verschiebt Part 3 in Koordinatensystem 1, fährt den Mover auf Part 3, verschiebt den Part 3 wieder in Koordinatensystem 2 und verfährt abschließend den Mover auf Part 2 in Koordinatensystem 2.

```
CASE state OF
  0:
    part_three.Initialize(0, 16#01010080, environment);
    state := 1;
  1:
    IF part_three.IsInitialized THEN
      state := 2;
    END_IF
  2:
    mover.Enable(0);
    state := 3;
  3:
    IF mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 4;
    END_IF
  4:
    part_three.ActivatePosition(0,1);
    state := 5;
  5:
    IF part_three.PositionIndex = 1 THEN
      state := 6;
    END_IF
  6:
    target_position.SetValuesXYCReferenceId(120, 120, 0, part_three.PartOID);
```

```
    mover.MoveToPosition(0, target_position, 0, 0);
    state := 7;
  7:
    IF mover.MCTOPLC.SET.SetPos.y > 300 AND NOT mover.MCTOPLC.STD.Busy.busyXYC THEN
      state := 8;
    END_IF
  8:
    part_three.ActivatePosition(0,2);
    state := 9;
  9:
    IF part_three.PositionIndex = 2 THEN
      state := 10;
    END_IF
  10:
    target_position.SetValuesXYCReferenceId(120, 120, 0, 16#01010030); // Position on part two
    mover.MoveToPosition(0, target_position, 0, 0);
    state := 11;

END_CASE
```

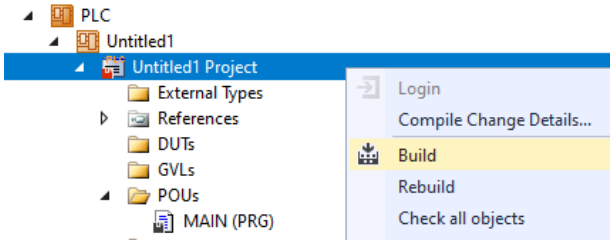
Befehl abschicken

4. Um den Bewegungsbefehl abzuschicken, müssen Sie den Mover nach dem END_CASE zyklisch mit seiner Update-Methode aufrufen; um die Befehle des Planar-Parts zu versenden muss die Environment zyklisch mit ihrer Update-Methode aufgerufen werden:

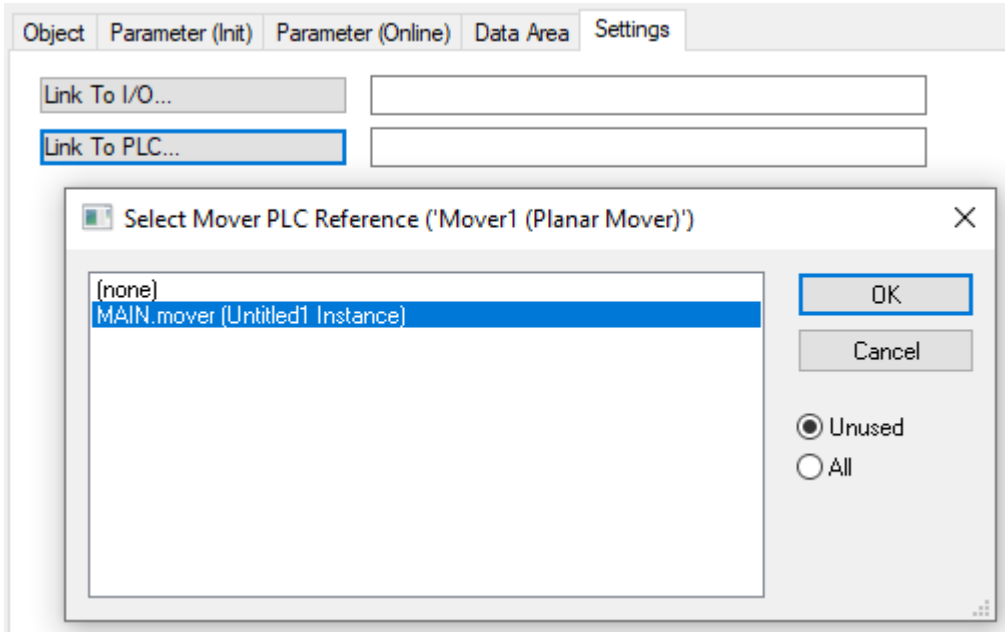
```
mover.Update();
environment.Update();
```

Durch das Bauen der PLC wird ein Symbol des „PLC-Movers" erzeugt, welches anschließend mit der Mover-Instanz im MC-Projekt verknüpft werden kann.

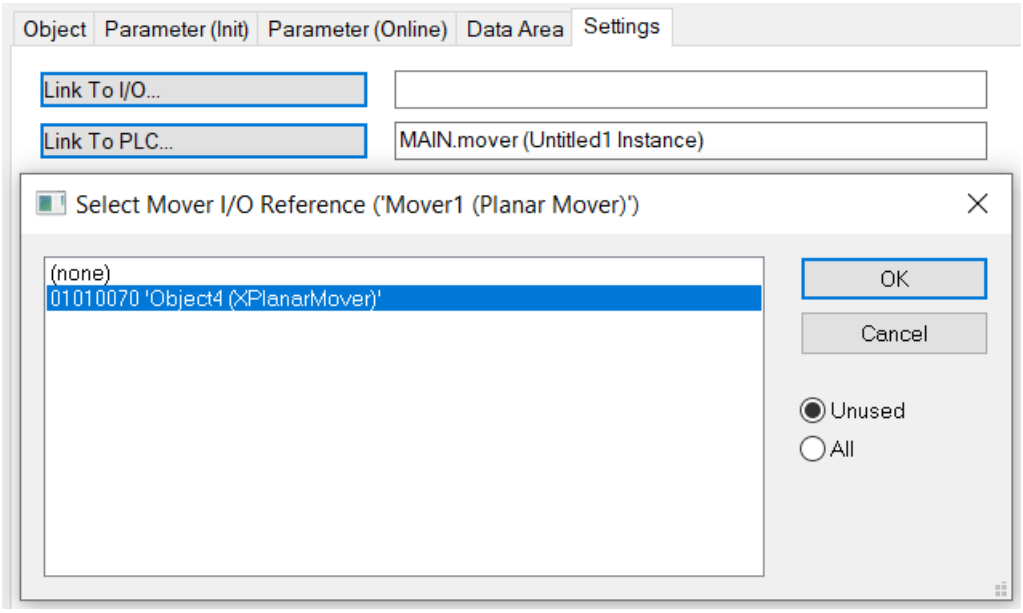
1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**



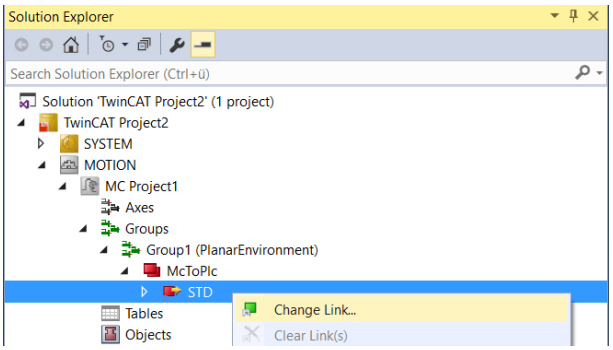
⇨ Anschließend kann der Planar-Mover im „MC Project“ (Doppelklick) mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.

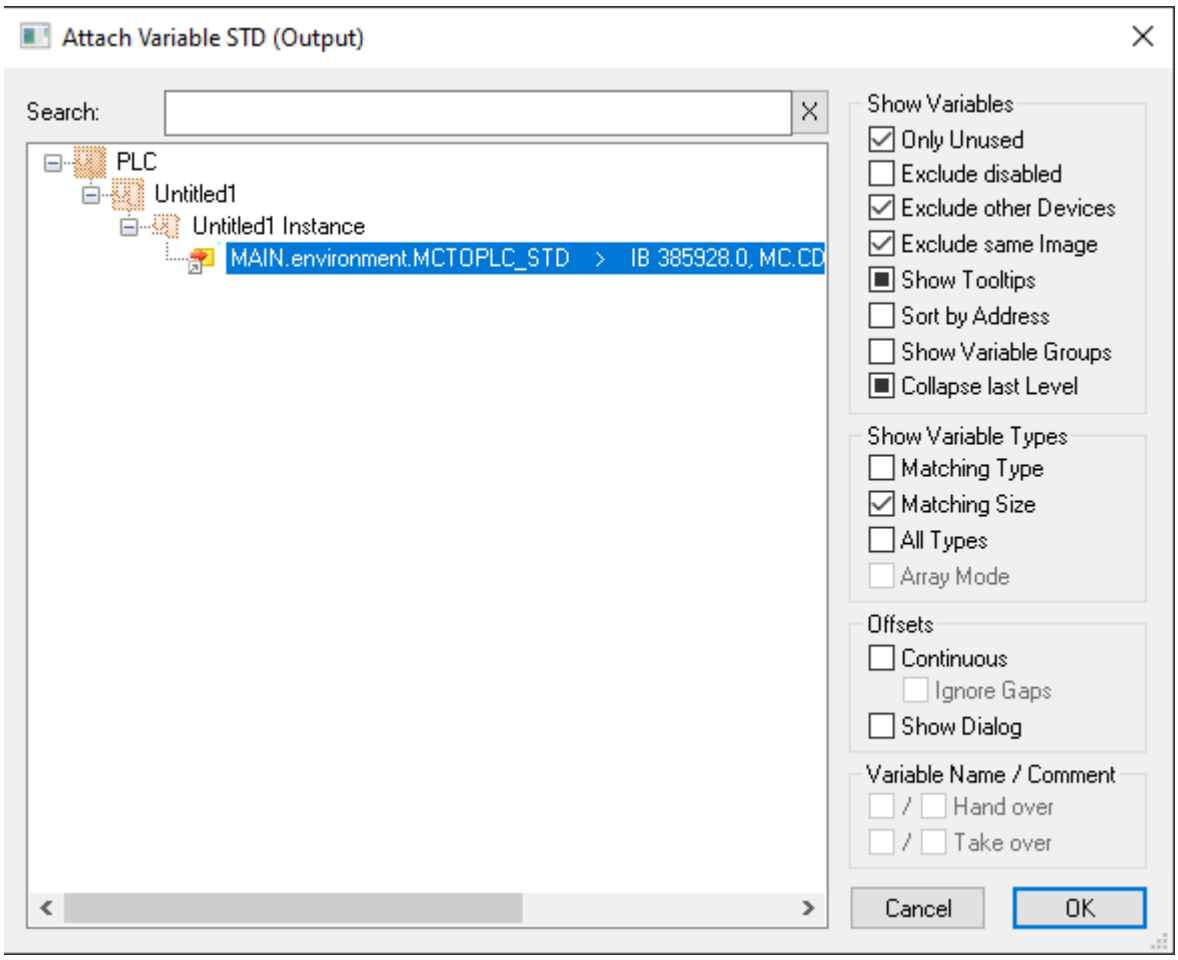


⇨ Zusätzlich muss der Planar-Mover im „MC Project“ (Doppelklick) mit dem Button **Link To I/O…** im Reiter **Settings** verknüpft werden.



⇨ Anschließend kann die Planar-Environment im „MC Project“ verlinkt werden.





Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Der Mover steht am Ende des Zustandsautomaten (state=11, im Bild unten hexadezimal dargestellt als B) auf der gewünschten Position. Die Position ist im Koordinatensystem zwei (Objekt Id 16#010100A0) angegeben. Der Mover hat zusammen mit Part 3 das Koordinatensystem gewechselt, bzw. ist mit dem Aufzug auf eine andere Ebene gefahren. Insgesamt kann man im Beispiel gut sehen, dass die Planar-Part PLC Objekte nur leichte Environment Wrapper sind, die mit der Environment initialisiert werden müssen und ihre Befehle über die Environment verschicken.

Expression	Type	Value
mover	MC_PlanarMover	
PLCTOMC	CDT_PLCTOMC_PLANAR_MOVER	
MCTOPLC	CDT_MCTOPLC_PLANAR_MOVER	
STD	REFERENCE TO CDT_MCTOPLC_PLANAR_MOVER_STD	
SET	REFERENCE TO CDT_MCTOPLC_PLANAR_MOVER_SET	
SetPos	MoverVector	
x	LREAL	120
y	LREAL	120
z	LREAL	1.9944148439562
a	LREAL	-0.00494750319896669...
b	LREAL	-0.00251608611460904...
c	LREAL	0
SetVelo	MoverVector	
SetAcc	MoverVector	
DcTimeStamp	ULINT	16#0A7EA6CA5EFC0000
PhysicalAreaID	UDINT	16#010100A0
ACT	REFERENCE TO CDT_MCTOPLC_PLANAR_MOVER_ACT	
COORDMODE	REFERENCE TO CDT_MCTOPLC_PLANAR_MOVER_COORD...	
SETONTRACK	REFERENCE TO CDT_MCTOPLC_PLANAR_MOVER_TRACK	
Error	BOOL	FALSE
ErrorId	UDINT	16#00000000
environment	MC_PlanarEnvironment	
part_three	MC_PlanarPart	
state	UDINT	16#0000000B
target_position	PositionXYC	

6.7 Planar-Feedback

Das MC_PlanarFeedback [▶ 139] ist ein PLC-Softwareobjekt, das sämtliche Statusinformationen für einen Befehl bündelt, der vom Nutzer an einen Mover, einen Track, eine Gruppe, oder andere Planar Komponenten, abgegeben wird.

Dies reicht vom Senden des Befehls durch den Nutzer über die Verarbeitung des Befehls durch die Komponenten und die anschließende Annahme (evtl. Ablehnung) bis zur Ausführung und Beendigung des Befehls. All dies kann der Nutzer anhand eines Feedback-Objekts nachverfolgen, falls er dies möchte.

Dazu muss er ein Feedback-Objekt in der PLC beim Aufruf der Befehlsmethode als erstes Argument übergeben. Immer, wenn der Nutzer danach das Feedback-Objekt triggert (oder seine Update-Methode aufruft), kann er den aktuellen Befehlsstatus abrufen.

Um ein Planar-Feedback zu benutzen, muss dieses in der PLC deklariert werden. Das Planar-Feedback hat keine feste Entsprechung in einem TCOM-Objekt auf Motion Control Seite. Von dort erhält es die Informationen direkt vom entsprechenden TCOM-Objekt (z. B. Planar-Mover), das den entsprechenden Befehl ausführt. Daher muss ein Feedback nicht separat im MC-Projekt angelegt, parametriert oder verknüpft werden.

6.7.1 Beispiel „Planar-Mover und Planar-Feedback anlegen“

Anhand dieser kurzen Anleitung werden Sie ein TwinCAT-Projekt anlegen, das einen Planar-Mover und ein Planar-Feedback enthält.

Planar-Mover anlegen

✓ Siehe Konfiguration [▶ 18].

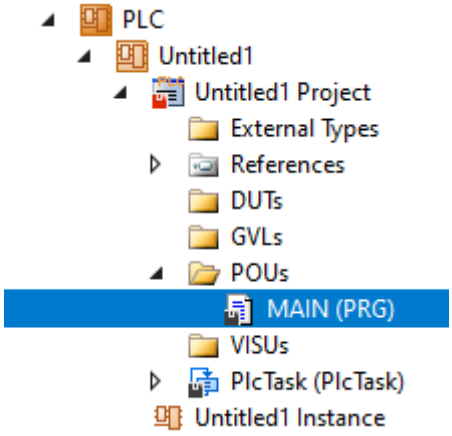
1. Legen Sie einen Planar-Mover an.

2. Versetzen Sie „Parameter (Init)“ in den Simulationsmodus (`TRUE`). Der Parameter ist versteckt und wird nur sichtbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.

PLC anlegen

✓ Siehe Vorabschritte PLC anlegen [▶ 21].

1. Legen Sie über MAIN einen Mover („MC_PlanarMover“) und ein Planar-Feedback („MC_PlanarFeedback“) wie folgt an.



⇨ Diese repräsentieren den Mover und das Planar-Feedback in der MC Configuration.

```
PROGRAM MAIN
VAR
    mover : MC_PlanarMover;
    feedback : MC_PlanarFeedback;
    state : UDINT;
    target_position : PositionXYC;
END_VAR
```

Sie haben in diesem einfachen Beispiel eine Zustandsvariable für eine Zustandsmaschine und eine Zielposition für einen Fahrbefehl des Movers angelegt. Auch ein Feedback ist deklariert, um den Befehlsverlauf überwachen zu können. Damit kann anschließend in der MAIN ein Ablauf programmiert werden:

```
CASE state OF
  0:
    mover.Enable(feedback);
    state := 1;
  1:
    IF feedback.Done THEN
      state := 2;
    END_IF
  2:
    target_position.SetValuesXY(1000, 1000);
    mover.MoveToPosition(feedback, target_position, 0, 0);
    state := 3;
END_CASE
```

Dieser Programmcode aktiviert den Mover und verfährt ihn zur Position x=1000 und y=1000.

Es ist zu beachten, dass der Zustandsautomat nur weitergeschaltet wird, wenn das Feedback über seine „Done“ Flag die erfolgreiche Beendigung des Befehls signalisiert.

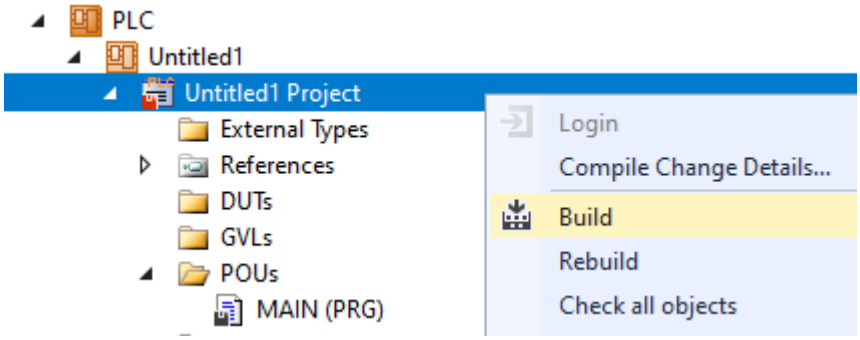
Befehl abschicken

2. Um den Befehl abzuschicken und das Feedback zu überwachen, müssen Sie den Mover und das Feedback nach dem END_CASE zyklisch mit ihren Update-Methoden aufrufen:

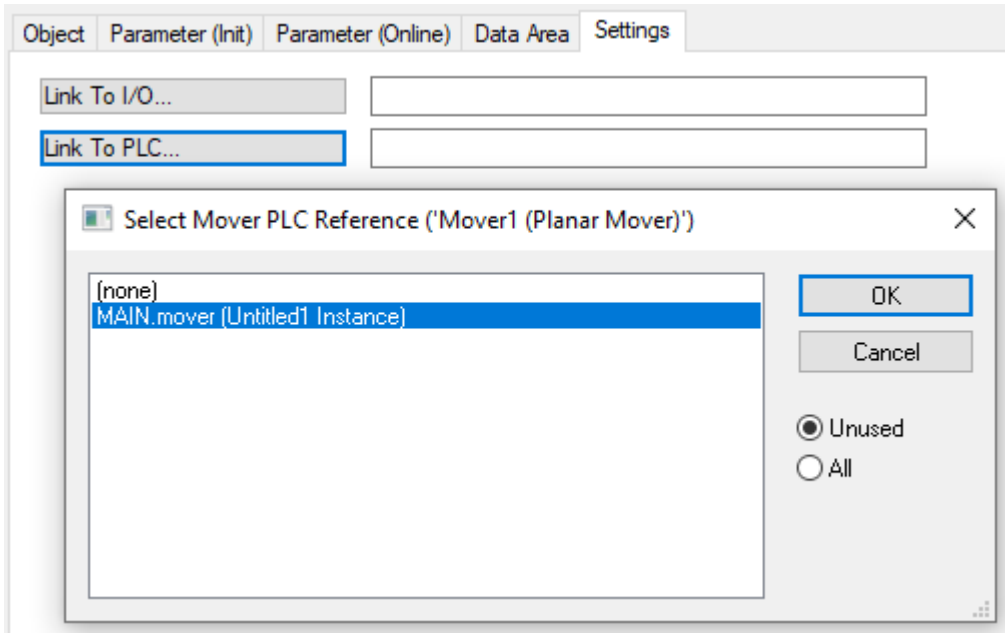
```
mover.Update();
feedback.Update();
```

Durch das Bauen der PLC wird ein Symbol des „PLC-Movers" erzeugt, welches anschließend mit der Mover-Instanz im MC-Projekt verknüpft werden kann.

1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**



⇨ Anschließend kann der Planar-Mover im „MC Project“ (Doppelklick) mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.



Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Der Mover steht am Ende des Zustandsautomaten (state=3) auf der gewünschten Position und das Feedback signalisiert die Beendigung des Befehls mit der Flag „Done“.

MAIN [Online] TwinCAT Project14

TwinCAT_Project14.Untitled1.MAIN

Expression	Type	Value	Prepared value	Address	Comment
mover	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC_PLA...			%Q*	Mover data that is tra...rred from the Plana...
MCTOPLC	CDT_MCTOPLC_PLA...			%I*	Mover data that is tra...rred from the Plana...
STD	REFERENCE TO CDT...			%IB*	Mover standard data t...is transferred from ...
SET	REFERENCE TO CDT...			%IB*	Mover setpoint data th...is transferred from t...
SetPos	MoverVector				Current position.
x	LREAL	1000			X coordinate.
y	LREAL	1000			Y coordinate.
z	LREAL	0			Z coordinate.
a	LREAL	0			A coordinate.
b	LREAL	0			B coordinate.
c	LREAL	0			C coordinate.
SetVelo	MoverVector				Current velocity.
SetAcc	MoverVector				Current acceleration.
DcTimeStamp	ULINT	6630698779040...			Current time stamp.
PhysicalAreaID	UDINT	0			Current physical area id.
ACT	REFERENCE TO CDT...			%IB*	Mover actpoint data th...is transferred from t...
COORDMODE	REFERENCE TO CDT...			%IB*	Mover coordinate mod...ormation that is tra...
SETONTRACK	REFERENCE TO CDT...			%IB*	Mover busy informatio...at is transferred fro...
Error	BOOL	FALSE			Flag indicating a PlanarMover error.
ErrorId	UDINT	0			Error id indicating the PlanarMover error type.
feedback	MC_PlanarFeedback				
objectInfo	PlanarObjectInfo				Indicates which object one would collide with.
Active	BOOL	FALSE			Indicates an active co...nd, i.e. command w...
Busy	BOOL	FALSE			Indicates a busy com..., i.e. command is b...
Done	BOOL	TRUE			Indicates the comman...done, i.e. execution...
Aborted	BOOL	FALSE			Indicates the comman...aborted, i.e. execut...
Error	BOOL	FALSE			Indicates the command has an error.
ErrorId	UDINT	0			Indicates the error id of the command error.
state	UDINT	3			
target_position	PositionXYC				

6.7.2 Beispiel „Planar Motion Komponenten: Kollision abwenden“

Anhand dieser kurzen Anleitung werden Sie ein TwinCAT-Projekt anlegen, das einen Planar-Mover enthält, dessen Fahrbefehl wegen einer Kollision mit der Planar-Environment abgelehnt wird.

Planar-Mover anlegen

✓ Siehe Konfiguration [▶ 18].

1. Legen Sie einen Planar-Mover an.
2. Versetzen Sie „Parameter (Init)“ in den Simulationsmodus (`TRUE`). Der Parameter ist versteckt und wird nur sichbar, wenn man die Checkbox „Show Hidden Parameters“ aktiviert.

Planar-Environment anlegen

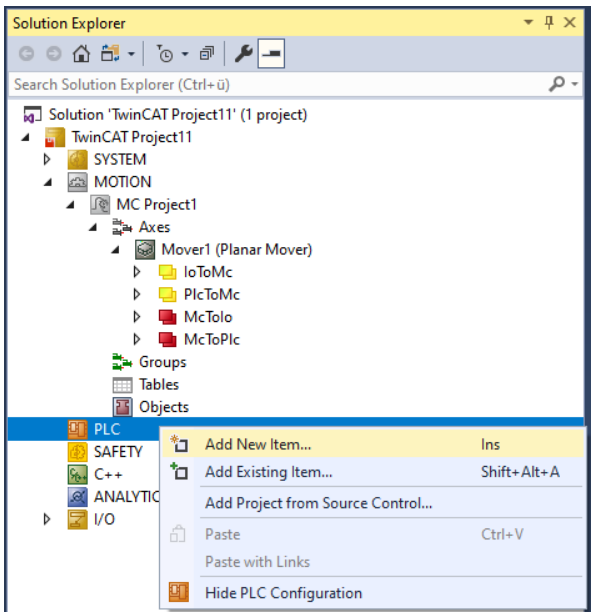
3. Legen Sie eine Planar-Environment an, siehe Konfiguration [▶ 98].

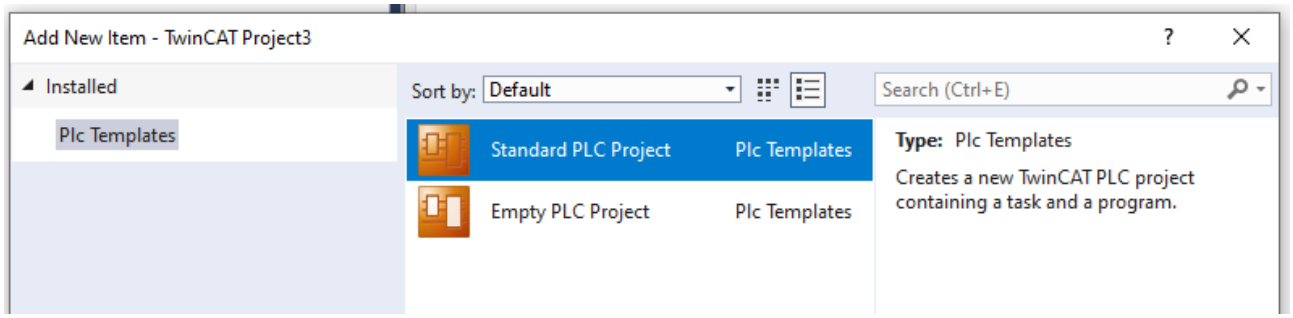
Planar-Group anlegen

4. Legen Sie eine Planar-Group an, siehe Konfiguration [▶ 91].

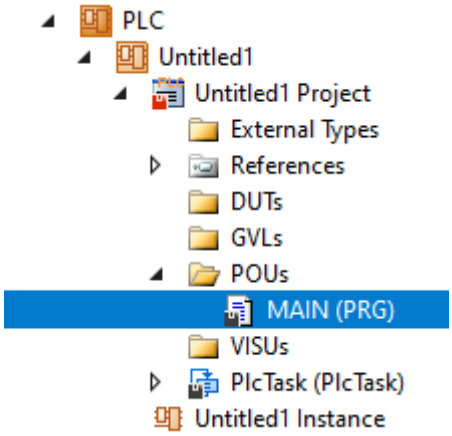
PLC anlegen

✓ Um die Geometrie der Environment zu erstellen und den Mover zu steuern, muss eine PLC angelegt werden, aus der der Nutzer Befehle an beide geben kann.





5. Fügen Sie dem PLC-Projekt die Bibliotheken Tc3_Physics und Tc3_Mc3PlanarMotion hinzu, siehe Bibliotheken einfügen [▶ 127].
6. Legen Sie über **MAIN** einen MC_PlanarMover [▶ 145] und eine MC_PlanarEnvironment [▶ 136] an.



⇨ Diese repräsentieren den Mover und die Environment in der MC Configuration.

```
PROGRAM MAIN
VAR
    mover           : MC_PlanarMover;
    environment     : MC_PlanarEnvironment;
    group           : MC_PlanarGroup;
    feedback        : MC_PlanarFeedback;
    state           : UDINT;
    target_position : PositionXYC;
END_VAR
```

Sie haben in diesem Beispiel eine Zustandsvariable für eine einfache Zustandsmaschine und eine Zielposition für einen einfachen Fahrbefehl des Movers angelegt. Damit kann anschließend in der MAIN ein Ablauf programmiert werden:

```
CASE state OF
  0:
    environment.AddStator(0,-120.0,-120.0);
    environment.CreateBoundary(0);
    state := 1;
  1:
    mover.Enable(0);
    group.Enable(0);
    state := 2;
  2:
    IF mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled AND
    group.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
      state := 3;
    ENDIF
  3:
    mover.AddToGroup(0,group);
    environment.AddToGroup(0,group);
    state := 4;
  4:
    IF mover.MCTOPLC.STD.GroupOID > 0 AND
    environment.MCTOPLC_STD. GroupOID > 0 THEN
      state := 5;
    ENDIF
  5:
    target_position.SetValuesXY(100, 100);
    mover.MoveToPosition(feedback, target_position, 0, 0);
    state := 6;
END_CASE
```

Dieser Programmcode aktiviert den Mover und erstellt eine Environment von einer Kachel, auf der der Mover steht. Dann wird versucht, den Mover zur Position x=100 und y=100 zu fahren.

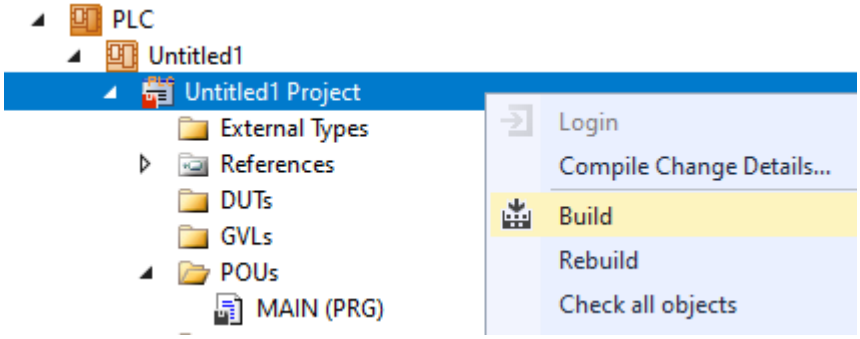
Befehl abschicken

7. Um den Befehl abzuschicken und das Feedback zu überwachen, müssen Sie die Objekte nach dem END_CASE zyklisch mit ihren Update-Methoden aufrufen:

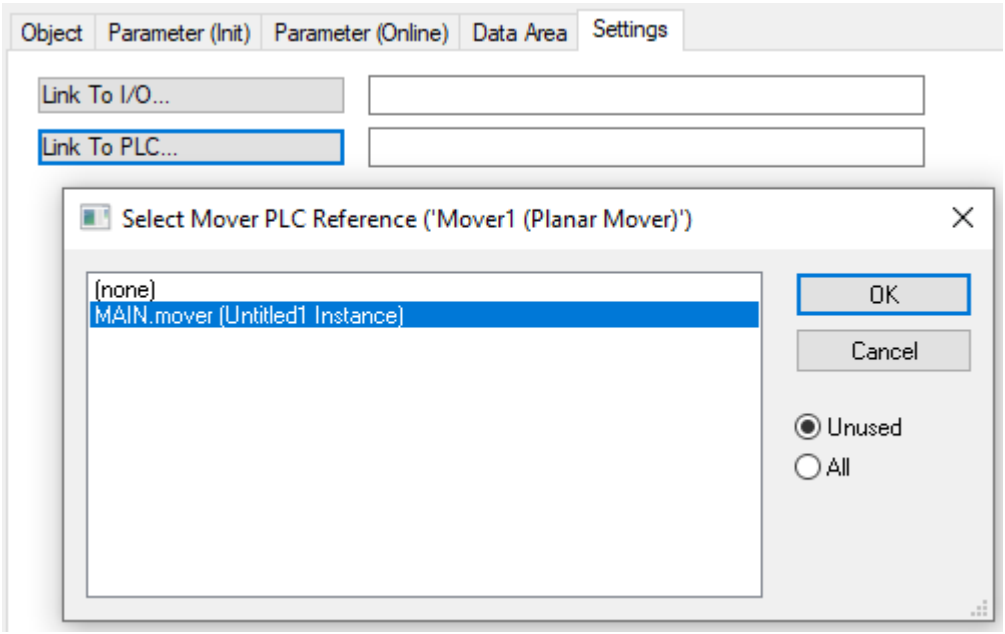
```
mover.Update();
environment.Update();
group.Update();
feedback.Update();
```

Durch das Bauen der PLC wird ein Symbol des „PLC-Movers" erzeugt, welches anschließend mit der Mover-Instanz im MC-Projekt verknüpft werden kann.

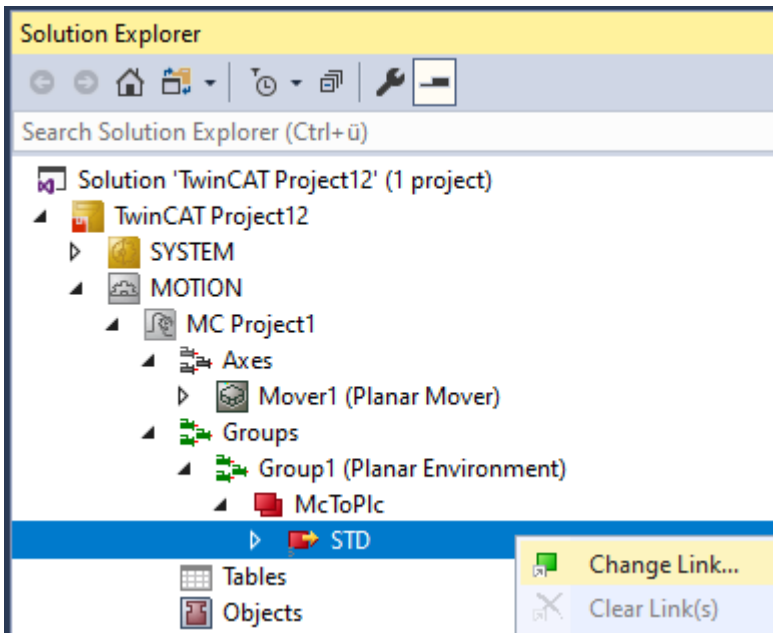
1. Verwenden Sie zum Bauen den Pfad **PLC > Untitled1 > Untitled1 Project > Build.**

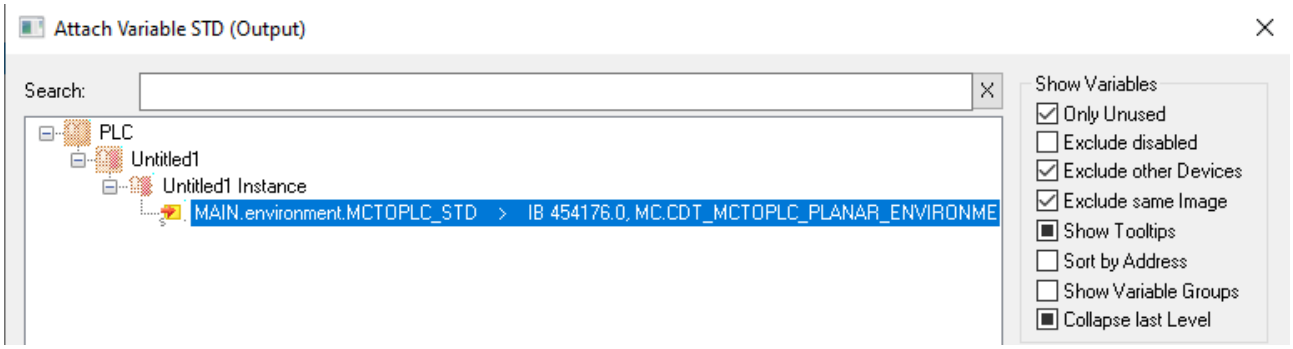


⇨ Anschließend kann der Planar-Mover im „MC Project“ (Doppelklick) mit dem Button **Link To PLC…** im Reiter **Settings** verknüpft werden.



⇨ Danach kann die Planar-Environment über die folgenden Dialogfenster im „MC Project“ verlinkt werden.





⇨ Die Gruppe wird analog verlinkt.

Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .

2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Der Mover steht am Ende des Zustandsautomaten (state=6) auf der gewünschten Position. Der Mover hat sich nicht bewegt, weil der Befehl abgelehnt wurde. Das Feedback zeigt einen Kollisionsfehler und in der ObjectInfo ist die Environment als Kollisionspartner angegeben.

TwinCAT Project12 MAIN [Online]

TwinCAT_Project12.Untitled1.MAIN

Expression	Type	Value	Prepared value	Address	Comment
mover	MC_PlanarMover				
PLCTOMC	CDT_PLCTOMC_PLA...			%Q*	Mover data that is tra...rred from the Plana...
MCTOPLC	CDT_MCTOPLC_PLA...			%I*	Mover data that is tra...rred from the Plana...
STD	REFERENCE TO CDT...			%IB*	Mover standard data t...is transferred from ...
SET	REFERENCE TO CDT...			%IB*	Mover setpoint data th...is transferred from t...
SetPos	MoverVector				Current position.
x	LREAL	0			X coordinate.
y	LREAL	0			Y coordinate.
z	LREAL	0			Z coordinate.
a	LREAL	0			A coordinate.
b	LREAL	0			B coordinate.
c	LREAL	0			C coordinate.
SetVelo	MoverVector				Current velocity.
SetAcc	MoverVector				Current acceleration.
DcTimeStamp	ULINT	6631508341750...			Current time stamp.
PhysicalAreaID	UDINT	0			Current physical area id.
ACT	REFERENCE TO CDT...			%IB*	Mover actpoint data th...is transferred from t...
COORDMODE	REFERENCE TO CDT...			%IB*	Mover coordinate mod...ormation that is tra...
SETONTRACK	REFERENCE TO CDT...			%IB*	Mover busy informatio...at is transferred fro...
Error	BOOL	FALSE			Flag indicating a PlanarMover error.
ErrorId	UDINT	0			Error id indicating the PlanarMover error type.
environment	MC_PlanarEnvironment				
group	MC_PlanarGroup				
feedback	MC_PlanarFeedback				
objectInfo	PlanarObjectInfo				Indicates which object one would collide with.
ObjectType	EPLANAROBJECTTYPE	Environment			Object type.
Id	UDINT	85065744			Object id.
Active	BOOL	FALSE			Indicates an active co...nd, i.e. command w...
Busy	BOOL	FALSE			Indicates a busy com..., i.e. command is b...
Done	BOOL	FALSE			Indicates the comman...done, i.e. execution...
Aborted	BOOL	FALSE			Indicates the comman...aborted, i.e. execut...
Error	BOOL	TRUE			Indicates the command has an error.
ErrorId	UDINT	33158			Indicates the error id of the command error.
state	UDINT	6			
target_position	PositionXYC				

6.7.3 Spezialisierte Feedback-Typen

Neben dem generellen Typ MC_PlanarFeedback, der von den meisten Befehlen angenommen wird, können bestimmte Befehle einen spezialisierten Feedback-Typ verlangen. Für diese Typen gelten alle Aussagen [▶ 114], die auch auf das allgemeine Feedback zutreffen.

Spezialisierte Feedbacks können je nach Typ eine Untermenge der Outputs, die das allgemeine Feedback aufweist, besitzen; also Informationen darüber, ob ein Kommando aktiv ist, es einen Fehler hervorgerufen hat usw. Zusätzlich können spezialisierte Feedback-Typen noch weitere Outputs oder Funktionen aufweisen, die ihrem Anwendungsbereich entsprechen.

MC_PlanarFeedbackGearInPosOnTrack

Dieser Feedback-Typ wird von einem GearInPosOnTrack-Befehl [▶ 66] angenommen. Er hat einen zusätzlichen Output `inSync`, an welchem angezeigt wird, ob der ausführende Mover synchron mit der Master-Achse ist.

MC_PlanarFeedbackGearInPosOnTrackWithMasterMover

Dieser Feedback-Typ wird von einem GearInPosOnTrackWithMasterMover-Befehl [▶ 73] angenommen. Er hat einen zusätzlichen Output `inSync`, an welchem angezeigt wird, ob der ausführende Mover synchron mit dem Master-Planer-Mover ist.

6.8 Planar-TrackTrail

Der MC_PlanarTrackTrail [▶ 174] ist ein Objekt, welches einen Pfad von aneinander anschließenden Planar-Tracks in einem Netzwerk definiert. Im Gegensatz zu den individuellen Planar-Tracks, aus denen der Planar-TrackTrail gebaut wird, hat der Planar-TrackTrail keine feste Entsprechung in einem TCOM-Objekt auf der Seite der MC, sondern wird allein in der PLC deklariert, ähnlich einem Planar-Feedback [▶ 114].

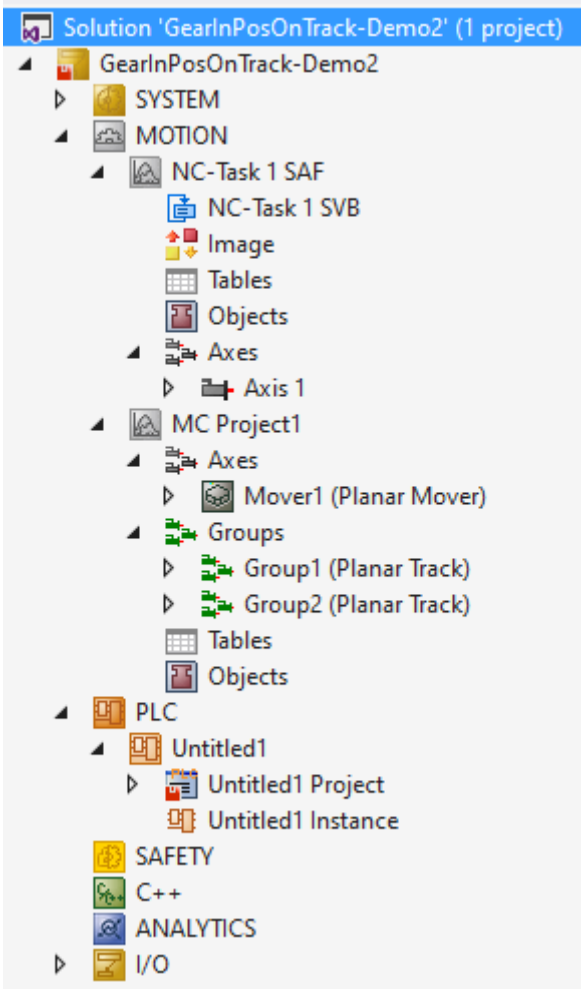
Ein Planar-TrackTrail kann genutzt werden, um einen Pfad von Planar-Tracks zu definieren, über welchen eine Synchronisationsbewegung eines Slave-Planar-Movers mit einer Master-Achse [▶ 66] oder mit einem Master-Planar-Mover [▶ 73] stattfinden soll (falls dieser Pfad aus mehreren als dem aktuellen Planar-Track des Slave-Planar-Movers besteht).

Der Planar-TrackTrail bietet Methoden zum Hinzufügen eines Planar-Tracks an sein Ende und zum Leeren seiner Konfiguration an. Diese Methoden modifizieren ausschließlich den Planar-TrackTrail und lassen insbesondere die zugrundeliegenden Planar-Tracks sowie das Netzwerk unberührt.

Beim Hinzufügen eines Planar-Tracks ist darauf zu achten, dass dieser an das Ende des aktuell letzten Planar-Tracks im Planar-TrackTrail anschließt. Außerdem ist ausgeschlossen, dass ein Planar-Track mehr als einmal hinzugefügt wird.

6.8.1 Beispiel „Synchronisationsbewegung über zwei Planar-Tracks“

Dieses Beispiel stellt eine Erweiterung des Beispiels „Planar-Mover auf einem Track mit einer Achse synchronisieren“ [▶ 66] dar, in welchem die Synchronisationsbewegung des Planar-Movers über zwei Planar-Tracks erfolgt. Das genannte Beispiel wird so abgewandelt, dass *zwei* Planar-Tracks in der MC Configuration angelegt werden, sodass der Solution Explorer die folgenden Einträge aufweist:

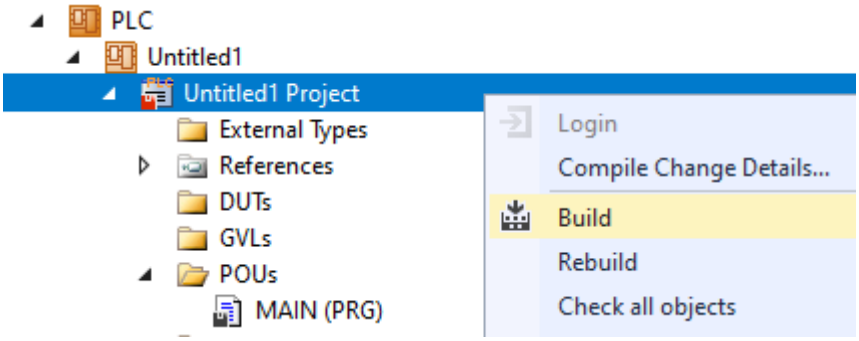


PLC Projekt anpassen

1. Fügen Sie dem PLC-Projekt die Bibliotheken Tc3_Mc3PlanarMotion, Tc3_Physics, und Tc2_MC2 hinzu, siehe Bibliotheken einfügen [▶ 127].
2. Deklarieren Sie in der MAIN die folgenden Variablen:

```
PROGRAM MAIN
VAR
    mover      : MC_PlanarMover;
    track1     : MC_PlanarTrack;
    track2     : MC_PlanarTrack;
    trail      : MC_PlanarTrackTrail;
    axis       : AXIS_REF;
    power_axis : MC_Power;
    move_axis  : MC_MoveAbsolute;
    state      : UDINT;
    pos1, pos2 : PositionXYC;
END_VAR
```

3. Bauen Sie die PLC, um Symbole des „PLC-Mover“, der „PLC-Tracks“ und der „PLC-Achse“ zu erzeugen.



4. Verlinken Sie Planar-Mover, Planar-Tracks (siehe Beispiel „Planar-Mover auf Track einkoppeln und verfahren [▶ 22]“) und die Achse (siehe Beispiel „Planar-Mover auf einem Track mit einer Achse synchronisieren [▶ 66]“).

Alle bisherigen Schritte sind, bis auf die Verdopplung der Anzahl der Planar-Tracks und den leicht modifizierten Code, identisch mit denen im Beispiel „Planar-Mover auf einem Track mit einer Achse synchronisieren“ [▶ 66].

Zustandsautomaten programmieren

Im nächsten Schritt wird der Programmcode so abgewandelt, dass dem `GearInPosOnTrack`-Befehl der Planar-TrackTrail übergeben wird. Letzterer wird vorher mit beiden Planar-Tracks befüllt, die in diesem Beispiel ein einfaches Netzwerk [▶ 76], bestehend aus einer L-Konfiguration mit Verschleifungsstück, bilden:

```
CASE state OF
  0:
    pos1.SetValuesXYC(100, 100, 0);
    pos2.SetValuesXYC(400, 100, 0);
    track1.AppendLine(0, pos1, pos2);
    track1.Enable(0);
    state := state + 1;
  1:
    IF track1.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
      state := state + 1;
    END_IF
  2:
    track2.StartFromTrack(0, track1);
    state := state + 1;
  3:
    pos1.SetValuesXYC(500, 100, 0);
    pos2.SetValuesXYC(860, 100, 0);
    track2.AppendLine(0, pos1, pos2);
    track2.Enable(0);
    state := state + 1;
  4:
    IF track2.MCTOPLC_STD.State = MC_PLANAR_STATE.Enabled THEN
      state := state + 1;
    END_IF
  5:
    mover.Enable(0);
    state := state + 1;
  6:
    IF mover.MCTOPLC.STD.State = MC_PLANAR_STATE.Enabled THEN
      state := state + 1;
    END_IF
  7:
    mover.JoinTrack(0, track1, 0, 0);
    state := state + 1;
  8:
    IF mover.MCTOPLC.STD.CommandMode =
    MC_PLANAR_MOVER_COMMAND_MODE.OnTrack THEN
      state := state + 1;
    END_IF
  9:
    power_axis(Axis := axis,
    Enable := TRUE,
    Enable_Positive := TRUE);
    IF power_axis.Status THEN
      move_axis(Axis := axis, Execute := FALSE);
```

```
      state := state + 1;
    END_IF
  10:
    move_axis(Axis := axis,
              Position := 700,
              Velocity := 30,
              Acceleration := 100,
              Deceleration := 100,
              Jerk := 100,
              Execute := TRUE);
    state := state + 1;
  11:
    trail.AddTrack(track1);
    trail.AddTrack(track2);
    mover.GearInPosOnTrack(0, axis.DriveAddress.TcAxisObjectId, trail, 100, 100, track1, 0, 0);
    state := state + 1;
END_CASE

mover.Update();
track1.Update();
track2.Update();
power_axis(Axis := axis);
move_axis(Axis := axis);
axis.ReadStatus();
```

Die beiden Planar-Tracks werden dem Planar-TrackTrail in State 11 hinzugefügt. Die Reihenfolge ist hier entscheidend, da `track2` an `track1` anschließt und nicht umgekehrt. Der Planar-TrackTrail wird dem `GearInPosOnTrack`-Befehl als drittes Argument übergeben.

Projekt aktivieren und starten

1. Aktivieren Sie die Konfiguration über den Button in der Menüleiste  .
2. Versetzen Sie das TwinCAT-System in den Zustand „Run“ über den Button  .
3. Loggen Sie die PLC über den Button in der Menüleiste ein  .
4. Starten Sie die PLC über den Play-Button in der Menüleiste.

Ablauf im Online-View beobachten

5. Beobachten Sie im Online-View, wie der Planar-Mover zunächst auf dem ersten Planar-Track in Richtung dessen Ende verfährt:

Expression	Type	Value
mover	MC_PlanarMover	
PLCTOMC	CDT_PLCTOMC_PLA...	
MCTOPLC	CDT_MCTOPLC_PLA...	
STD	REFERENCE TO CDT...	
SET	REFERENCE TO CDT...	
ACT	REFERENCE TO CDT...	
COORDMODE	REFERENCE TO CDT...	
SETONTRACK	REFERENCE TO CDT...	
SetPos	LREAL	190.8083232748...
SetVelo	LREAL	29.99999999999...
SetAcc	LREAL	0
SetJerk	LREAL	0
TrackOID	OTCID	16#05120010

6. Danach sehen Sie, wie er auf den anschließenden Planar-Track wechselt (vgl. die TrackOIDs):

Expression	Type	Value
mover	MC_PlanarMover	
PLCTOMC	CDT_PLCTOMC_PLA...	
MCTOPLC	CDT_MCTOPLC_PLA...	
STD	REFERENCE TO CDT...	
SET	REFERENCE TO CDT...	
ACT	REFERENCE TO CDT...	
COORDMODE	REFERENCE TO CDT...	
SETONTRACK	REFERENCE TO CDT...	
SetPos	LREAL	33.90832327486...
SetVelo	LREAL	29.99999999999...
SetAcc	LREAL	0
SetJerk	LREAL	0
TrackOID	OTCID	16#05120020

7. Abschließend sehen Sie, wie er auf dem zweiten Planar-Track zum Stillstand kommt:

Expression	Type	Value
mover	MC_PlanarMover	
PLCTOMC	CDT_PLCTOMC_PLA...	
MCTOPLC	CDT_MCTOPLC_PLA...	
STD	REFERENCE TO CDT...	
SET	REFERENCE TO CDT...	
ACT	REFERENCE TO CDT...	
COORDMODE	REFERENCE TO CDT...	
SETONTRACK	REFERENCE TO CDT...	
SetPos	LREAL	400.0000000000...
SetVelo	LREAL	0
SetAcc	LREAL	0
SetJerk	LREAL	0
TrackOID	OTCID	16#05120020

⇨ Auch in diesem Beispiel bricht der Planar-Mover seine Synchronisationsbewegung ab, falls das Verhalten der Master-Achse ein Überfahren des Endes des zweiten Planar-Tracks erfordern sollte (z. B. indem die Zielposition der Master-Achse größer als die Summe der Längen der beiden Planar-Tracks gewählt wird). In diesem Fall kommt der Planar-Mover am Ende des zweiten Tracks zum Stillstand, verliert seinen potentiellen Synchronisationsstatus und meldet einen Fehler.

In dem Fall, dass ein weiterer Planar-Track ans Ende des *ersten* angehängt wird, sodass an dessen Ende eine Weiche entsteht, „weiß“ der Planar-Mover durch den Planar-TrackTrail eindeutig, auf welchen der beiden Planar-Tracks er abbiegen und somit seine Synchronisationsbewegung fortsetzen soll (die Master-Achse produziert ihre Sollwerte schließlich unabhängig von Planar-Tracks). Auf diese Art und Weise kann mithilfe eines Planar-TrackTrails eine Synchronisationsbewegung durch beliebig komplizierte Tracknetzwerke auf einem eindeutigen, beliebig langen Pfad durchgeführt werden.

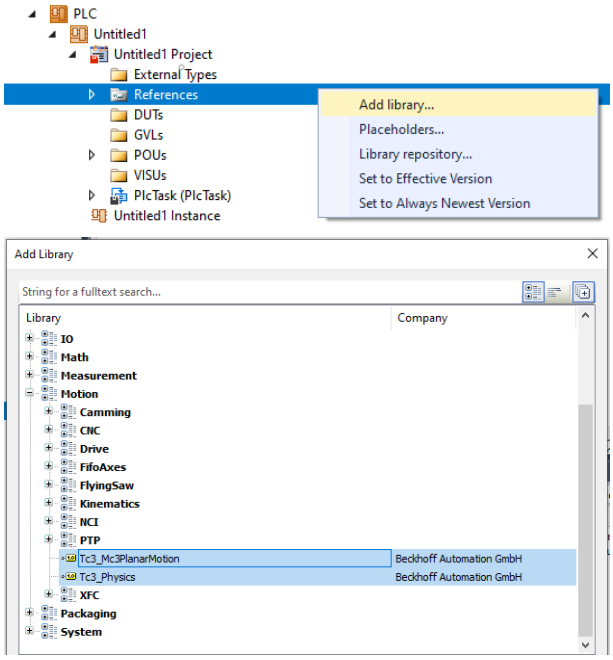
Da der Planar-TrackTrail ein reines PLC-Objekt ist, welches nicht über TCOM kommuniziert, sondern nur als Container fungiert, ist kein zyklisches Update, wie beispielsweise für den Planar-Mover, die Planar-Tracks, oder auch Planar-Feedbacks [▶ 114] (die in diesem Beispiel nicht vorkommen), nötig, und eine entsprechende Methode ist nicht vorhanden.

7 PLC Bibliotheken

7.1 Bibliotheken einfügen

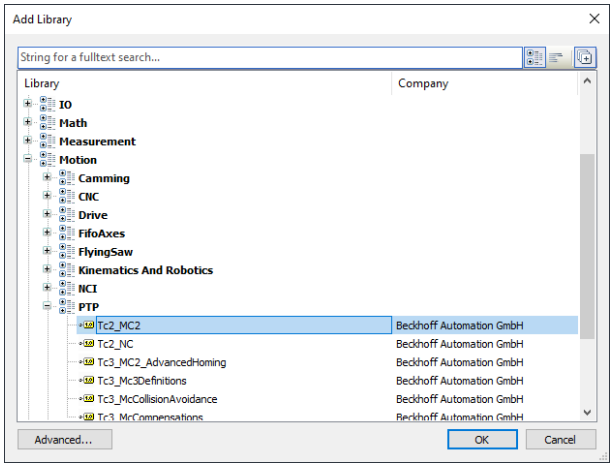
✓ Um XPlanar-Komponenten anzusteuern, müssen die Bibliotheken Tc3_Physics und Tc3_Mc3PlanarMotion eingebunden werden.

1. Fügen Sie Ihrem Projekt die gewünschten Bibliotheken jeweils nacheinander über **References > Add library…** hinzu.



⇨ Wenn die Bibliotheken eingebunden sind, können Sie den weiteren Ablauf in der PLC programmieren.

Zum Steuern einer Master-Achse muss zusätzlich die Bibliothek „Tc2_MC2“ eingebunden werden.



7.2 Tc3_Mc3PlanarMotion API

7.2.1 Data Types

7.2.1.1 Enums

7.2.1.1.1 EPlanarObjectType

Identifies a planar object type.

Syntax

Definition:

```
TYPE EPlanarObjectType :
(
    Invalid     := 0,
    None        := 301,
    Mover       := 302,
    Track       := 303,
    Environment := 304
)UINT;
END_TYPE
```

Values

Name	Description
Invalid	Indicates invalid information, e.g. no connection or component not yet ready.
None	No planar object.
Mover	Planar Mover.
Track	Planar Track.
Environment	Planar Environment.

7.2.1.1.2 MC_DIRECTION

Indicates the movement direction of the Planar Mover on a Planar Track.

Syntax

Definition:

```
TYPE MC_DIRECTION :
(
    mcDirectionNonModulo   := 0,
    mcDirectionPositive    := 1,
    mcDirectionShortestWay := 2,
    mcDirectionNegative    := 3
)UINT;
END_TYPE
```

Values

Name	Description
mcDirectionNonModulo	The Planar Mover moves to the absolute value of the target position. Depending on the current position, this may induce forward or backward movement. On looped tracks, multiple passes are possible.
mcDirectionPositive	The Planar Mover moves to the target position in a forward direction. No backward movement is allowed.

Name	Description
mcDirectionShortestWay	The Planar Mover takes the shortest way to the target position. May induce forward or backward movement.
mcDirectionNegative	The Planar Mover moves to the target position in a backward direction. No forward movement is allowed.

In Kombination mit der Tc2_MC2 oder Tc3_Mc3Definitions Bibliothek kann es sein, dass der Datentyp nicht eindeutig aufgelöst werden kann (ambiguous use of name 'MC_Direction'). Dann muss jeweils der Namespace bei Verwendung des Datentyps mit angegeben werden (Tc3_Mc3PlanarMotion.MC_DIRECTION bzw. Tc3_Mc3Definitions.MC_DIRECTION bzw. Tc2_MC2.MC_DIRECTION).

7.2.1.1.3 MC_SYNC_DIRECTIONS

Directions in which a slave is allowed to move during synchronizing phase.

Syntax

Definition:

```
TYPE MC_SYNC_DIRECTIONS :
(
    Positive := 1,
    Negative := 2,
    Both     := 3
)UINT;
END_TYPE
```

Values

Name	Description
Positive	Movement is allowed only in positive direction while synchronizing.
Negative	Movement is allowed only in negative direction while synchronizing.
Both	Movement is allowed in any direction while synchronizing.

7.2.1.2 Structs

7.2.1.2.1 CDT_MCTOPLC_PLANAR_MOVER

Contains the information of the Planar Mover passed from MC to PLC.

Syntax

Definition:

```
TYPE CDT_MCTOPLC_PLANAR_MOVER :
STRUCT
    STD        : Reference To CDT_MCTOPLC_PLANAR_MOVER_STD;
    SET        : Reference To CDT_MCTOPLC_PLANAR_MOVER_SET;
    ACT        : Reference To CDT_MCTOPLC_PLANAR_MOVER_ACT;
    COORDMODE  : Reference To CDT_MCTOPLC_PLANAR_MOVER_COORDMODE;
    SETONTRACK : Reference To CDT_MCTOPLC_PLANAR_MOVER_TRACK;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
STD	Reference To CDT_MCTOPLC_PLANAR_MOVER_STD	Mover standard data that is transferred from the Planar Mover to this function block.
SET	Reference To CDT_MCTOPLC_PLANAR_MOVER_SET	Mover setpoint data that is transferred from the Planar Mover to this function block.
ACT	Reference To CDT_MCTOPLC_PLANAR_MOVER_ACT	Mover actpoint data that is transferred from the Planar Mover to this function block.
COORDMODE	Reference To CDT_MCTOPLC_PLANAR_MOVER_COORDMODE	Mover coordinate mode information that is transferred from the Planar Mover to this function block.
SETONTRACK	Reference To CDT_MCTOPLC_PLANAR_MOVER_TRACK	Mover busy information that is transferred from the Planar Mover to this function block.

7.2.1.2.2 CDT_PLCTOMC_PLANAR_MOVER

Contains the information of the Planar Mover passed from PLC to MC.

Syntax

Definition:

```
TYPE CDT_PLCTOMC_PLANAR_MOVER :
STRUCT
    STD : Reference To CDT_PLCTOMC_PLANAR_MOVER_STD;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
STD	Reference To CDT_PLCTOMC_PLANAR_MOVER_STD	Mover standard data that is transferred from this function block to the Planar Mover.

7.2.1.2.3 PlanarObjectInfo

Identifies a planar object uniquely by object id and type.

Syntax

Definition:

```
TYPE PlanarObjectInfo :
STRUCT
    ObjectType : EPlanarObjectType;
    Id         : UDINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
ObjectType	EPlanarObjectType [▶ 128]	Object type.
Id	UDINT	Object id.

7.2.1.2.4 ST_AdoptTrackOrientationOptions

Options for the "AdoptTrackOrientation" command of the Planar Mover.

Syntax

Definition:

```
TYPE ST_AdoptTrackOrientationOptions :
STRUCT
    additionalTurns : UDINT;
    direction       : MC_DIRECTION;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
additionalTurns	UDINT	0	Addition turns to move in modulo movement (positive or negative).
direction	MC_DIRECTION [▶ 128]	MC_DIRECTION.mcDirectionShortestWay	Direction in which the target is approached.

7.2.1.2.5 ST_EndAtTrackAdvancedOptions

Options for the "EndAtTrackAdvanced" command of the Planar Track.

Syntax

Definition:

```
TYPE ST_EndAtTrackAdvancedOptions :
STRUCT
    thisTrackPartPositionIndex       : UDINT;
    otherTrackPartPositionIndex      : UDINT;
    linkOnlyInSpecifiedPartPositions : BOOL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
thisTrackPartPositionIndex	UDINT	0	The index of the position in which the part of this track is for track connection.
otherTrackPartPositionIndex	UDINT	0	The index of the position in which the part of the other track is for track connection.
linkOnlyInSpecifiedPartPositions	BOOL	FALSE	If false the tracks are connected not only in the given positions configuration of their parts but also in all other (geometrically compatible) locations, otherwise only the specified location is connected.

7.2.1.2.6 ST_ExternalSetpointGenerationOptions

Options for the "ExternalSetpointGeneration" command of the Planar Mover.

Syntax

Definition:

```
TYPE ST_ExternalSetpointGenerationOptions :
STRUCT
    mode : MC_EXTERNAL_SET_POSITION_MODE;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
mode	MC_EXTERNAL_SET_POSITION_MODE	MC_EXTERNAL_SET_POSITION_MODE.Absolute	Mode can be relative or absolute, relative can be used parallel to all other commands, absolute only alone.

7.2.1.2.7 ST_GearInPosOnTrackOptions

Options for the "GearInPosOnTrack" command of the Planar Mover.

Syntax

Definition:

```
TYPE ST_GearInPosOnTrackOptions :
STRUCT
    gap                              : LREAL;
    inSyncToleranceDistance          : LREAL;
    directionSlaveSyncPosition       : MC_DIRECTION;
    moduloToleranceSlaveSyncPosition : LREAL;
    allowedSlaveSyncDirections       : MC_SYNC_DIRECTIONS;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
gap	LREAL	200.0	Minimal distance to next Planar Mover on track.
inSyncToleranceDistance	LREAL	0.0	Tolerance in absolute value of position difference to master axis for inSync flag.
directionSlaveSyncPosition	MC_DIRECTION [▶ 128]	MC_DIRECTION.mcDirectionNonModulo	Direction in which the slave sync position is approached.
moduloToleranceSlaveSyncPosition	LREAL	0.0	Tolerance "window" for slave sync position.
allowedSlaveSyncDirections	MC_SYNC_DIRECTIONS [▶ 129]	MC_SYNC_DIRECTIONS.Positive	Directions in which the slave is allowed to move while in synchronizing phase.

7.2.1.2.8 ST_GearInPosOnTrackWithMasterMoverOptions

Options for the "GearInPosOnTrackWithMasterMover" command of the Planar Mover.

Syntax

Definition:

```
TYPE ST_GearInPosOnTrackWithMasterMoverOptions :
STRUCT
    gap                               : LREAL;
    inSyncToleranceDistance           : LREAL;
    directionSlaveSyncPosition        : MC_DIRECTION;
    moduloToleranceSlaveSyncPosition  : LREAL;
    directionMasterSyncPosition       : MC_DIRECTION;
    moduloToleranceMasterSyncPosition : LREAL;
    allowedSlaveSyncDirections        : MC_SYNC_DIRECTIONS;
    followMover                       : BOOL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
gap	LREAL	200.0	Minimal distance to next Planar Mover on track.
inSyncToleranceDistance	LREAL	0.0	Tolerance in absolute value of position difference to master axis for inSync flag.
directionSlaveSyncPosition	MC_DIRECTION [▶ 128]	MC_DIRECTION.mcDirectionNonModulo	Direction in which the slave sync position is approached.
moduloToleranceSlaveSyncPosition	LREAL	0.0	Tolerance "window" for slave sync position.
directionMasterSyncPosition	MC_DIRECTION [▶ 128]	MC_DIRECTION.mcDirectionNonModulo	Direction in which the master sync position is approached.
moduloToleranceMasterSyncPosition	LREAL	0.0	Tolerance "window" for master sync position.
allowedSlaveSyncDirections	MC_SYNC_DIRECTIONS [▶ 129]	MC_SYNC_DIRECTIONS.Positive	Directions in which the slave is allowed to move while in synchronizing phase.
followMover	BOOL	FALSE	If true, the slave PlanarMover will proceed to follow the master PlanarMover after the latter has traversed the masterSyncPosition. In this case the PlanarTrackTrail describes the slave's path towards the masterSyncPosition. If false, the slave moves exclusively on the PlanarTrackTrail specified.

7.2.1.2.9 ST_JoinTrackOptions

Options for the "JoinTrack" command of the Planar Mover.

Syntax

Definition:

```
TYPE ST_JoinTrackOptions :
STRUCT
    useOrientation : BOOL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
useOrientation	BOOL	TRUE	If true, the target orientation is also reached at the end of the movement.

7.2.1.2.10 ST_LeaveTrackOptions

Options for the "LeaveTrack" command of the Planar Mover.

Syntax

Definition:

```
TYPE ST_LeaveTrackOptions :
STRUCT
    useOrientation : BOOL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
useOrientation	BOOL	TRUE	If true, the target orientation is also reached at the end of the movement.

7.2.1.2.11 ST_MoveCOptions

Options for the "MoveC" command of the Planar Mover.

Syntax

Definition:

```
TYPE ST_MoveCOptions :
STRUCT
    additionalTurns : UDINT;
    direction       : MC_DIRECTION;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
additionalTurns	UDINT	0	Addition turns to move in modulo movement (positive or negative).
direction	MC_DIRECTION [▶ 128]	MC_DIRECTION.mcDirectionNonModulo	Direction in which the target is approached.

7.2.1.2.12 ST_MoveOnTrackOptions

Options for the "MoveOnTrack" command of the Planar Mover.

Syntax

Definition:

```
TYPE ST_MoveOnTrackOptions :
STRUCT
    gap             : LREAL;
    direction       : MC_DIRECTION;
    additionalTurns : UDINT;
    moduloTolerance : LREAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
gap	LREAL	200.0	Minimal distance to next Planar Mover on track.
direction	MC_DIRECTION [▶ 128]	MC_DIRECTION.mcDirectionNonModulo	Direction in which the target is approached.
additionalTurns	UDINT	0	Addition turns to move in modulo movement (positive or negative).
moduloTolerance	LREAL	0.0	Tolerance "window" in modulo movement.

7.2.1.2.13 ST_MoveToPositionOptions

Options for the "MoveToPosition" command of the Planar Mover.

Syntax

Definition:

```
TYPE ST_MoveToPositionOptions :
STRUCT
    useOrientation : BOOL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
useOrientation	BOOL	TRUE	If true, the target orientation is also reached at the end of the movement.

7.2.1.2.14 ST_StartFromTrackAdvancedOptions

Options for the "StartFromTrackAdvanced" command of the Planar Track.

Syntax

Definition:

```
TYPE ST_StartFromTrackAdvancedOptions :
STRUCT
    thisTrackPartPositionIndex       : UDINT;
    otherTrackPartPositionIndex      : UDINT;
    linkOnlyInSpecifiedPartPositions : BOOL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
thisTrackPartPositionIndex	UDINT	0	The index of the position in which the part of this track is for track connection.
otherTrackPartPositionIndex	UDINT	0	The index of the position in which the part of the other track is for track connection.
linkOnlyInSpecifiedPartPositions	BOOL	FALSE	If false the tracks are connected not only in the given positions configuration of their parts but also in all other (geometrically compatible) locations, otherwise only the specified location is connected.

7.2.2 Function Blocks

7.2.2.1 MC_PlanarEnvironment

A Planar Environment object specifies the environment that Planar Movers can move in. It contains information about the stator objects and boundaries of the movement area.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
Clear [▶ 137]	Clears the Planar Environment (stators and boundary).
AddStator [▶ 137]	Adds a stator to the Planar Environment.
CreateBoundary [▶ 137]	Creates a boundary for the Planar Environment based on the previously added stator information or hardware information.
Update [▶ 138]	Updates internal state of the object, must be triggered each cycle.
AddToGroup [▶ 138]	Adds the Planar Environment to the given Planar Group.
RemoveFromGroup [▶ 139]	Removes the Planar Environment from its current Planar Group, i.e. disables collision checks.
GetPlanarObjectInfo [▶ 139]	Returns environment object info (type: environment, id: OID of nc environment).

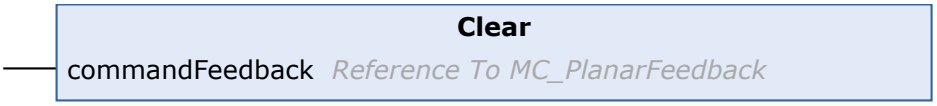
Required License

TC3 Planar Motion Base

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.11	PC or CX (x64)	Tc3_Mc3PlanarMotion, Tc3_Physics

7.2.2.1.1 Clear



Clears the Planar Environment (stators and boundary).

Syntax

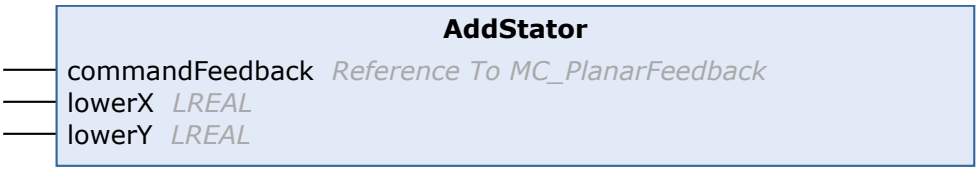
Definition:

```
METHOD Clear
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.1.2 AddStator



Adds a stator to the Planar Environment.

Syntax

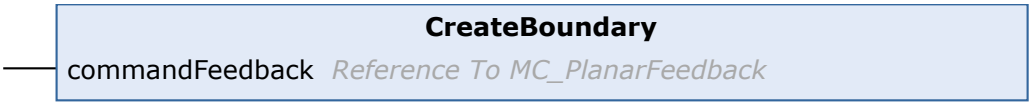
Definition:

```
METHOD AddStator
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
    lowerX          : LREAL;
    lowerY          : LREAL;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
lowerX	LREAL	The lower x position of the stator.
lowerY	LREAL	The lower y position of the stator.

7.2.2.1.3 CreateBoundary



Creates a boundary for the Planar Environment based on the previously added stator information or hardware information.

Syntax

Definition:

```
METHOD CreateBoundary
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.1.4 Update

Update

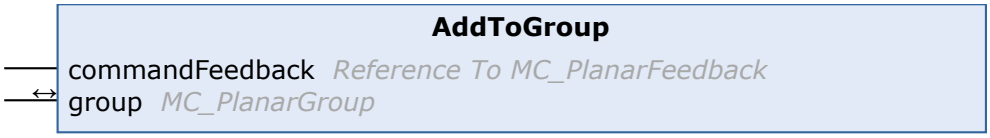
Updates internal state of the object, must be triggered each cycle.

Syntax

Definition:

```
METHOD Update
```

7.2.2.1.5 AddToGroup



Adds the Planar Environment to the given Planar Group.

Syntax

Definition:

```
METHOD AddToGroup
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
VAR_IN_OUT
    group           : MC_PlanarGroup;
END_VAR
```

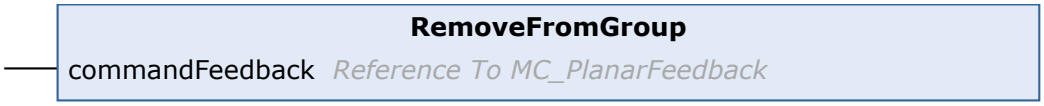
Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

In/Outputs

Name	Type	Description
group	MC_PlanarGroup [▶ 143]	The Planar Group that the mover joins.

7.2.2.1.6 RemoveFromGroup



Removes the Planar Environment from its current Planar Group, i.e. disables collision checks.

Syntax

Definition:

```
METHOD RemoveFromGroup
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.1.7 GetPlanarObjectInfo

Returns environment object info (type: environment, id: OID of nc environment).

Syntax

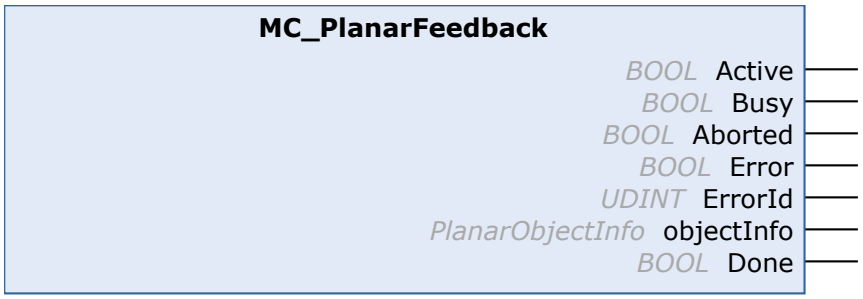
Definition:

```
METHOD GetPlanarObjectInfo : PlanarObjectInfo
```

Return value

PlanarObjectInfo [▶ 130]

7.2.2.2 MC_PlanarFeedback



Displays specific command status information for an associated command, given back by the MC Project.

Syntax

Definition:

```
FUNCTION_BLOCK MC_PlanarFeedback
VAR_OUTPUT
    Active      : BOOL;
```

```
    Busy       : BOOL;
    Aborted    : BOOL;
    Error      : BOOL;
    ErrorId    : UDINT;
    objectInfo : PlanarObjectInfo;
    Done       : BOOL;
END_VAR
```

Outputs

Name	Type	Description
Active	BOOL	Indicates an active command, i.e. command was accepted and is being executed.
Busy	BOOL	Indicates a busy command, i.e. command is being processed, waiting for execution, or already executing (= also active).
Aborted	BOOL	Indicates the command is aborted, i.e. execution of the command finished due the start of other commands.
Error	BOOL	Indicates the command has an error.
ErrorId	UDINT	Indicates the error id of the command error.
objectInfo	PlanarObjectInfo [▶ 130]	Indicates which object one would collide with.
Done	BOOL	Indicates the command is done, i.e. execution of the command finished successfully.

Methods

Name	Description
Update [▶ 140]	Updates internal state of the object.

Required License

TC3 Planar Motion Base

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.11	PC or CX (x64)	Tc3_Mc3PlanarMotion, Tc3_Physics

7.2.2.2.1 Update

Update

Updates internal state of the object.

Syntax

Definition:

```
METHOD Update
```

7.2.2.3 MC_PlanarFeedbackBase

Displays general command status information for an associated command, given back by the MC Project.

Do not call the main FB directly. Only use the available methods.

Required License

TC3 Planar Motion Base

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.11	PC or CX (x64)	Tc3_Mc3PlanarMotion, Tc3_Physics

7.2.2.4 MC_PlanarFeedbackGearInPosOnTrack

Displays specific command status information for an associated GearInPosOnTrack command, given back by the MC Project.

Syntax

Definition:

```
FUNCTION_BLOCK MC_PlanarFeedbackGearInPosOnTrack
VAR_OUTPUT
    Active     : BOOL;
    Busy       : BOOL;
    Aborted    : BOOL;
    Error      : BOOL;
    ErrorId    : UDINT;
    objectInfo : PlanarObjectInfo;
    inSync     : BOOL;
END_VAR
```

Outputs

Name	Type	Description
Active	BOOL	Indicates an active command, i.e. command was accepted and is being executed.
Busy	BOOL	Indicates a busy command, i.e. command is being processed, waiting for execution, or already executing (= also active).
Aborted	BOOL	Indicates the command is aborted, i.e. execution of the command finished due the start of other commands.
Error	BOOL	Indicates the command has an error.
ErrorId	UDINT	Indicates the error id of the command error.
objectInfo	PlanarObjectInfo [▶ 130]	Indicates which object one would collide with.
inSync	BOOL	Indicates whether the mover is currently in sync with the master (within tolerance specified in the command options).

Methods

Name	Description
Update [▶ 142]	Updates internal state of the object.

Required License

TC3 Planar Motion Base

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.11	PC or CX (x64)	Tc3_Mc3PlanarMotion, Tc3_Physics

7.2.2.4.1 Update

Update

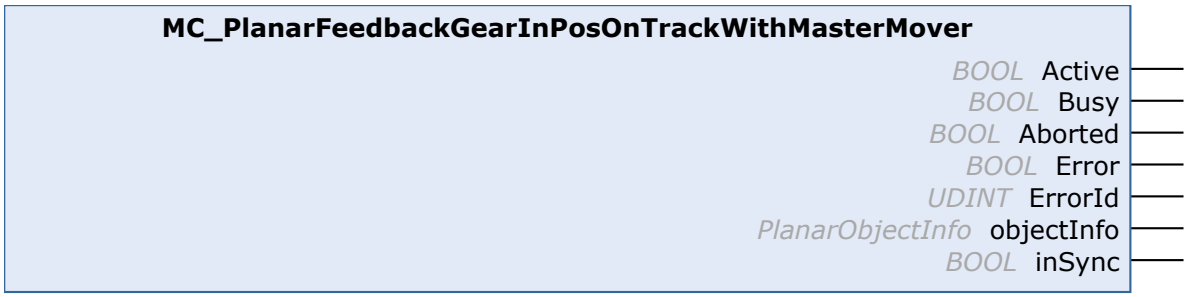
Updates internal state of the object.

Syntax

Definition:

```
METHOD Update
```

7.2.2.5 MC_PlanarFeedbackGearInPosOnTrackWithMasterMover



Displays specific command status information for an associated GearInPosOnTrack command, given back by the MC Project.

Syntax

Definition:

```
FUNCTION_BLOCK MC_PlanarFeedbackGearInPosOnTrackWithMasterMover
VAR_OUTPUT
    Active     : BOOL;
    Busy       : BOOL;
    Aborted    : BOOL;
    Error      : BOOL;
    ErrorId    : UDINT;
    objectInfo : PlanarObjectInfo;
    inSync     : BOOL;
END_VAR
```

Outputs

Name	Type	Description
Active	BOOL	Indicates an active command, i.e. command was accepted and is being executed.
Busy	BOOL	Indicates a busy command, i.e. command is being processed, waiting for execution, or already executing (= also active).
Aborted	BOOL	Indicates the command is aborted, i.e. execution of the command finished due the start of other commands.
Error	BOOL	Indicates the command has an error.

Name	Type	Description
ErrorId	UDINT	Indicates the error id of the command error.
objectInfo	PlanarObjectInfo [▶ 130]	Indicates which object one would collide with.
inSync	BOOL	Indicates whether the mover is currently in sync with the master (within tolerance specified in the command options).

Methods

Name	Description
Update [▶ 143]	Updates internal state of the object.

Required License

TC3 Planar Motion Base

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.11	PC or CX (x64)	Tc3_Mc3PlanarMotion, Tc3_Physics

7.2.2.5.1 Update

Update

Updates internal state of the object.

Syntax

Definition:

```
METHOD Update
```

7.2.2.6 MC_PlanarFeedbackInSync

Base class for all specialized feedbacks featuring an inSync output.

Do not call the main FB directly. Only use the available methods.

Required License

TC3 Planar Motion Base

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.11	PC or CX (x64)	Tc3_Mc3PlanarMotion, Tc3_Physics

7.2.2.7 MC_PlanarGroup

A Planar Group object. Planar Movers and other objects added to the group perform collision checks against each other.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
Enable [▶ 144]	Starts enabling the Planar Group.
Disable [▶ 144]	Starts disabling the Planar Group.
Reset [▶ 145]	Starts resetting the Planar Group.
Update [▶ 145]	Updates internal state of the object, must be triggered each cycle.

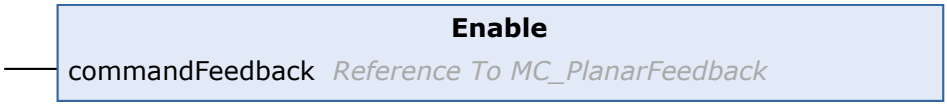
Required License

TC3 Planar Motion Base

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.11	PC or CX (x64)	Tc3_Mc3PlanarMotion, Tc3_Physics

7.2.2.7.1 Enable



Starts enabling the Planar Group.

Syntax

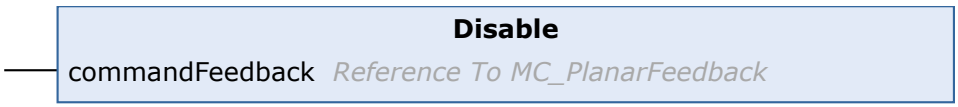
Definition:

```
METHOD Enable
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.7.2 Disable



Starts disabling the Planar Group.

Syntax

Definition:

```
METHOD Disable
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.7.3 Reset

Starts resetting the Planar Group.

Syntax

Definition:

```
METHOD Reset
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.7.4 Update

Updates internal state of the object, must be triggered each cycle.

Syntax

Definition:

```
METHOD Update
```

7.2.2.8 MC_PlanarMover

A Planar Mover object capable of moving within a plane. Limited movement vertical to the plane is available.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
MoveToPosition [▶ 147]	Initiates a direct movement to the specified position.
JoinTrack [▶ 147]	Initiates a direct movement to the specified track. At the end of the movement the mover joins the track.
LeaveTrack [▶ 148]	Initiates a direct movement to the specified position. At the beginning of the movement the track is left.

Name	Description
MoveOnTrack [▶ 149]	Initiates a movement on the track to the specified position and returns command ID.
GearInPosOnTrack [▶ 150]	Initiates a GearInPos movement along a specified trail.
GearInPosOnTrackWithMasterMover [▶ 151]	Initiates a GearInPos movement along a specified trail, in which the master setpoints are provided by another PlanarMover.
MoveZ [▶ 152]	Initiates a movement for the z component.
MoveA [▶ 152]	Initiates a movement for the a component.
MoveB [▶ 153]	Initiates a movement for the b component.
MoveC [▶ 153]	Initiates a movement for the c component.
AdoptTrackOrientation [▶ 154]	Initiates a movement for the c component.
Halt [▶ 155]	Initiates a halt.
Enable [▶ 155]	Starts enabling the Planar Mover.
Disable [▶ 155]	Starts disabling the Planar Mover.
Reset [▶ 156]	Starts resetting the Planar Mover.
Update [▶ 156]	Updates internal state of the object, must be triggered each cycle.
SetPosition [▶ 156]	Sets the position of the Planar Mover. Only possible if the Planar Mover is disabled.
StartExternalSetpointGeneration [▶ 157]	Starts the external setpoint generation, the user must supply setpoints from this PLC cycle on in every PLC cycle.
StopExternalSetpointGeneration [▶ 157]	Ends the external setpoint generation, called after last SetExternalSetpoint (in the same PLC cycle).
SetExternalSetpoint [▶ 158]	Sets the external setpoint for the Planar Mover without ref sys id (for relative mode), must be called each PLC cycle during external setpoint generation.
SetExternalSetpointReferenceId [▶ 158]	Sets the external setpoint for the Planar Mover with corresponing reference system id, must be called each PLC cycle during external setpoint generation.
AddToGroup [▶ 159]	Adds the Planar Mover to the given Planar Group.
RemoveFromGroup [▶ 159]	Removes the Planar Mover from its current Planar Group, i.e. disables collision checks.
GetPositionOnCurrentPart [▶ 160]	Sets the values of the given position to the movers position values on the current part.
GetPlanarObjectInfo [▶ 160]	Returns mover object info (type: mover, id: OID of nc mover).

Required License

TC3 Planar Motion Base

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.11	PC or CX (x64)	Tc3_Mc3PlanarMotion, Tc3_Physics

7.2.2.8.1 MoveToPosition

Initiates a direct movement to the specified position.

Syntax

Definition:

```
METHOD MoveToPosition
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
VAR_IN_OUT
    targetPosition  : PositionXYC;
END_VAR
VAR_INPUT
    constraint      : Reference To IPlcDynamicConstraint;
    options         : Reference To ST_MoveToPositionOptions;
END_VAR
```

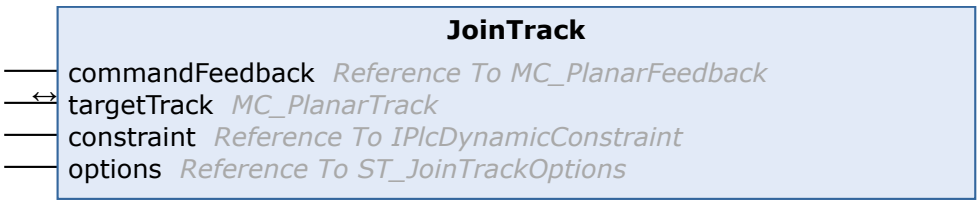
Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
constraint	Reference To IPlcDynamicConstraint	Dynamic constraints for this movement.
options	Reference To ST_MoveToPositionOptions [▶ 135]	Options for the movement.

In/Outputs

Name	Type	Description
targetPosition	PositionXYC	Target position for the movement.

7.2.2.8.2 JoinTrack



Initiates a direct movement to the specified track. At the end of the movement the mover joins the track.

Syntax

Definition:

```
METHOD JoinTrack
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

```
VAR_IN_OUT
    targetTrack     : MC_PlanarTrack;
END_VAR
VAR_INPUT
    constraint      : Reference To IPlcDynamicConstraint;
    options         : Reference To ST_JoinTrackOptions;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
constraint	Reference To IPlcDynamicConstraint	Dynamic constraints for this movement.
options	Reference To ST_JoinTrackOptions [▶ 133]	Options for the movement.

In/Outputs

Name	Type	Description
targetTrack	MC_PlanarTrack [▶ 164]	Target track for the movement.

7.2.2.8.3 LeaveTrack

Initiates a direct movement to the specified position. At the beginning of the movement the track is left.

Syntax

Definition:

```
METHOD LeaveTrack
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
VAR_IN_OUT
    targetPosition  : PositionXYC;
END_VAR
VAR_INPUT
    constraint      : Reference To IPlcDynamicConstraint;
    options         : Reference To ST_LeaveTrackOptions;
END_VAR
```

Inputs

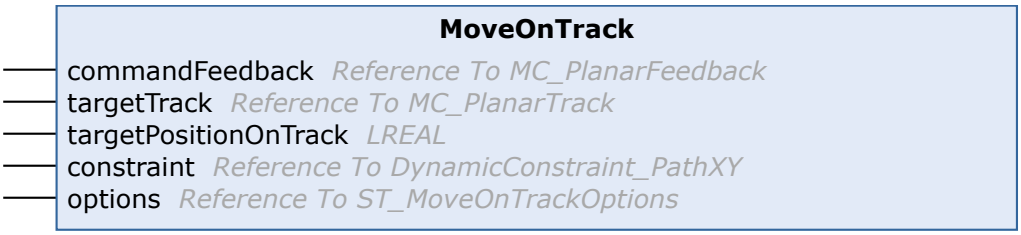
Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
constraint	Reference To IPlcDynamicConstraint	Dynamic constraints for this movement.

Name	Type	Description
options	Reference To ST_LeaveTrackOptions [▶ 134]	Options for the movement.

In/Outputs

Name	Type	Description
targetPosition	PositionXYC	Target position for the movement.

7.2.2.8.4 MoveOnTrack



Initiates a movement on the track to the specified position and returns command ID.

Syntax

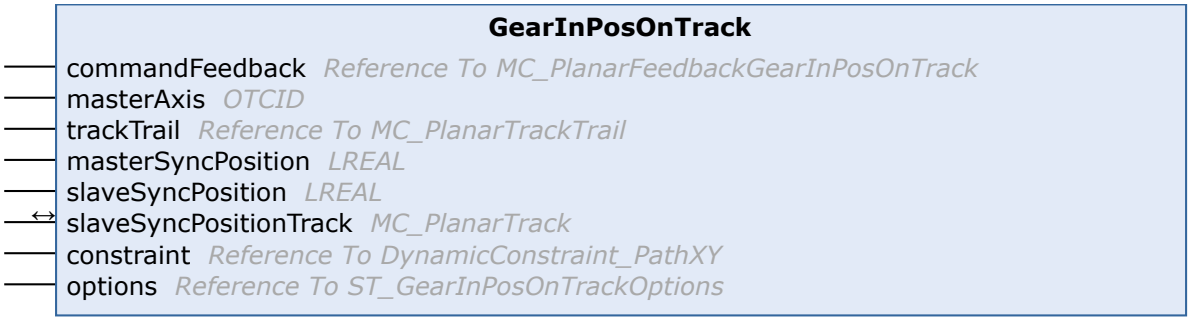
Definition:

```
METHOD MoveOnTrack
VAR_INPUT
    commandFeedback       : Reference To MC_PlanarFeedback;
    targetTrack           : Reference To MC_PlanarTrack;
    targetPositionOnTrack : LREAL;
    constraint            : Reference To DynamicConstraint_PathXY;
    options               : Reference To ST_MoveOnTrackOptions;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
targetTrack	Reference To MC_PlanarTrack [▶ 164]	Target track for the movement. If none is specified, this defaults to the current track.
targetPositionOnTrack	LREAL	Target position on the target track.
constraint	Reference To DynamicConstraint_PathXY	Constraint on maximal dynamics during the movement (V,A,D,J).
options	Reference To ST_MoveOnTrackOptions [▶ 134]	Options for the movement.

7.2.2.8.5 GearInPosOnTrack



Initiates a GearInPos movement along a specified trail.

Syntax

Definition:

```
METHOD GearInPosOnTrack
VAR_INPUT
    commandFeedback        : Reference To MC_PlanarFeedbackGearInPosOnTrack;
    masterAxis             : OTCID;
    trackTrail             : Reference To MC_PlanarTrackTrail;
    masterSyncPosition     : LREAL;
    slaveSyncPosition      : LREAL;
END_VAR
VAR_IN_OUT
    slaveSyncPositionTrack : MC_PlanarTrack;
END_VAR
VAR_INPUT
    constraint             : Reference To DynamicConstraint_PathXY;
    options                : Reference To ST_GearInPosOnTrackOptions;
END_VAR
```

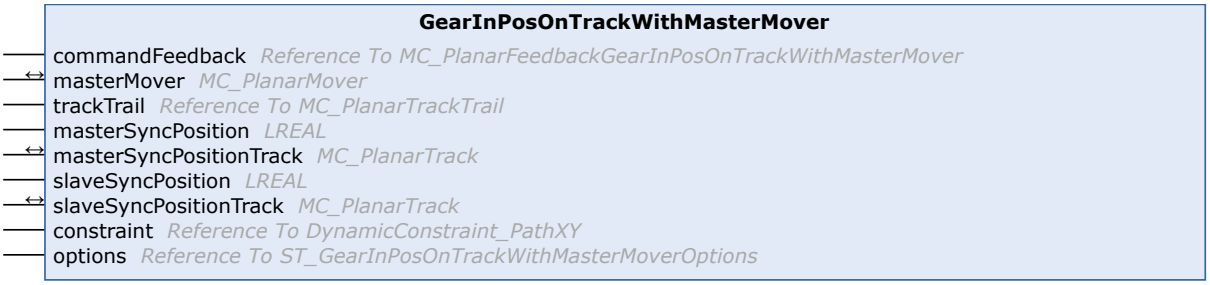
Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedbackGearInPosOnTrack [▶ 141]	The command specific feedback object for the command.
masterAxis	OTCID	Master axis being followed.
trackTrail	Reference To MC_PlanarTrackTrail [▶ 174]	Track trail determining along which tracks the GearInPos movement is allowed to proceed.
masterSyncPosition	LREAL	Position of the master axis at which the slave is inSync.
slaveSyncPosition	LREAL	Arc length on track given by slaveSyncPositionTrackOID at which the slave is inSync. Possibly interpreted in modulo fashion, depending on options.
constraint	Reference To DynamicConstraint_PathXY	Constraint on maximal dynamics during the movement (V,A,D,J).
options	Reference To ST_GearInPosOnTrackOptions [▶ 132]	Options for the movement.

In/Outputs

Name	Type	Description
slaveSyncPositionTrack	MC_PlanarTrack [▶ 164]	Track on which the slave is inSync.

7.2.2.8.6 GearInPosOnTrackWithMasterMover



Initiates a GearInPos movement along a specified trail, in which the master setpoints are provided by another PlanarMover.

Syntax

Definition:

```
METHOD GearInPosOnTrackWithMasterMover
VAR_INPUT
    commandFeedback         : Reference To MC_PlanarFeedbackGearInPosOnTrackWithMasterMover;
END_VAR
VAR_IN_OUT
    masterMover             : MC_PlanarMover;
END_VAR
VAR_INPUT
    trackTrail              : Reference To MC_PlanarTrackTrail;
    masterSyncPosition      : LREAL;
END_VAR
VAR_IN_OUT
    masterSyncPositionTrack : MC_PlanarTrack;
END_VAR
VAR_INPUT
    slaveSyncPosition       : LREAL;
END_VAR
VAR_IN_OUT
    slaveSyncPositionTrack  : MC_PlanarTrack;
END_VAR
VAR_INPUT
    constraint              : Reference To DynamicConstraint_PathXY;
    options                 : Reference To ST_GearInPosOnTrackWithMasterMoverOptions;
END_VAR
```

Inputs

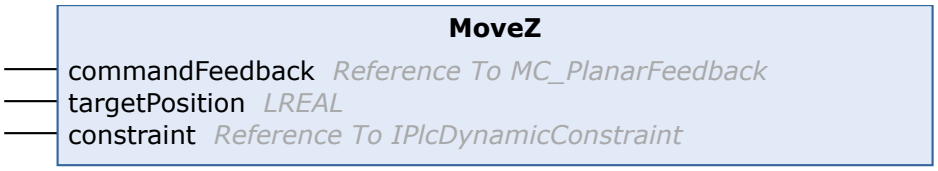
Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedbackGearInPosOnTrackWithMasterMover [▶ 142]	The command specific feedback object for the command.
trackTrail	Reference To MC_PlanarTrackTrail [▶ 174]	Track trail determining along which tracks the GearInPos movement is allowed to proceed.
masterSyncPosition	LREAL	Position of the master axis at which the slave is inSync.
slaveSyncPosition	LREAL	Arc length on track given by slaveSyncPositionTrackOID at which the slave is inSync. Possibly interpreted in modulo fashion, depending on options.
constraint	Reference To DynamicConstraint_PathXY	Constraint on maximal dynamics during the movement (V,A,D,J).

Name	Type	Description
options	Reference To ST_GearInPosOnTrackWithMasterMoverOptions [▶ 132]	Options for the movement.

In/Outputs

Name	Type	Description
masterMover	MC_PlanarMover [▶ 145]	Master mover being followed.
masterSyncPositionTrack	MC_PlanarTrack [▶ 164]	Track on which the master is inSync.
slaveSyncPositionTrack	MC_PlanarTrack [▶ 164]	Track on which the slave is inSync.

7.2.2.8.7 MoveZ



Initiates a movement for the z component.

Syntax

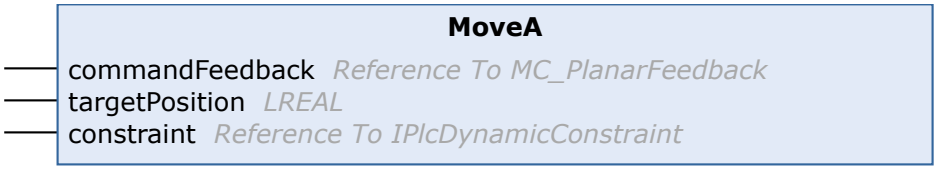
Definition:

```
METHOD MoveZ
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
    targetPosition  : LREAL;
    constraint      : Reference To IPlcDynamicConstraint;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
targetPosition	LREAL	Target position for the movement.
constraint	Reference To IPlcDynamicConstraint	Dynamic constraints for this movement.

7.2.2.8.8 MoveA



Initiates a movement for the a component.

Syntax

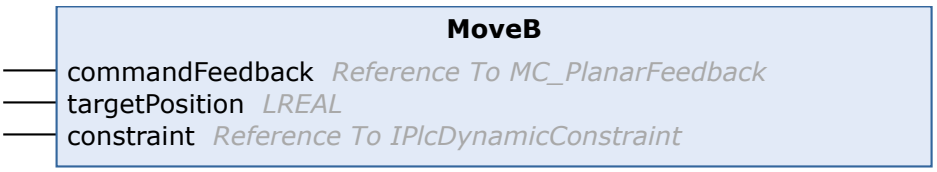
Definition:

```
METHOD MoveA
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
    targetPosition  : LREAL;
    constraint      : Reference To IPlcDynamicConstraint;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
targetPosition	LREAL	Target position for the movement.
constraint	Reference To IPlcDynamicConstraint	Dynamic constraints for this movement.

7.2.2.8.9 MoveB



Initiates a movement for the b component.

Syntax

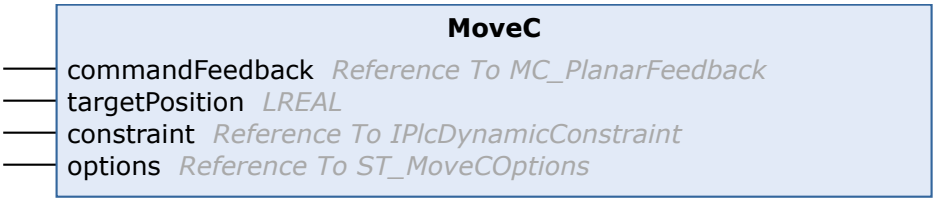
Definition:

```
METHOD MoveB
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
    targetPosition  : LREAL;
    constraint      : Reference To IPlcDynamicConstraint;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
targetPosition	LREAL	Target position for the movement.
constraint	Reference To IPlcDynamicConstraint	Dynamic constraints for this movement.

7.2.2.8.10 MoveC



Initiates a movement for the c component.

Syntax

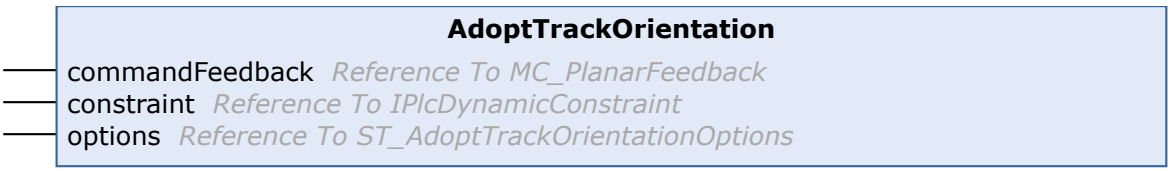
Definition:

```
METHOD MoveC
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
    targetPosition  : LREAL;
    constraint      : Reference To IPlcDynamicConstraint;
    options         : Reference To ST_MoveCOptions;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
targetPosition	LREAL	Target position for the movement.
constraint	Reference To IPlcDynamicConstraint	Dynamic constraints for this movement.
options	Reference To ST_MoveCOptions [▶ 134]	Options for the rotation.

7.2.2.8.11 AdoptTrackOrientation



Initiates a movement for the c component.

Syntax

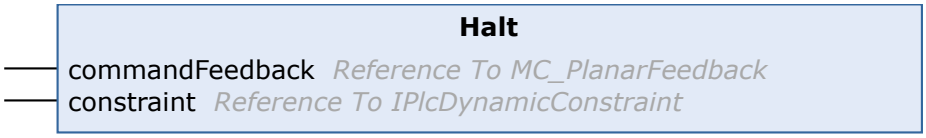
Definition:

```
METHOD AdoptTrackOrientation
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
    constraint      : Reference To IPlcDynamicConstraint;
    options         : Reference To ST_AdoptTrackOrientationOptions;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
constraint	Reference To IPlcDynamicConstraint	Dynamic constraints for this movement.
options	Reference To ST_AdoptTrackOrientationOptions [▶ 131]	Options for the rotation.

7.2.2.8.12 Halt



Initiates a halt.

Syntax

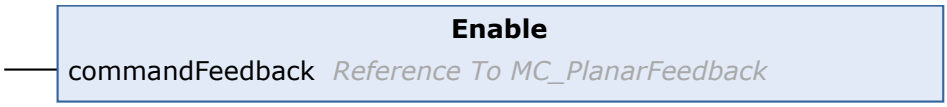
Definition:

```
METHOD Halt
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
    constraint      : Reference To IPlcDynamicConstraint;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
constraint	Reference To IPlcDynamicConstraint	Dynamic constraints for this movement.

7.2.2.8.13 Enable



Starts enabling the Planar Mover.

Syntax

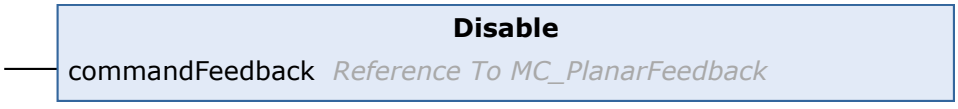
Definition:

```
METHOD Enable
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.8.14 Disable



Starts disabling the Planar Mover.

Syntax

Definition:

```
METHOD Disable
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.8.15 Reset

Starts resetting the Planar Mover.

Syntax

Definition:

```
METHOD Reset
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.8.16 Update

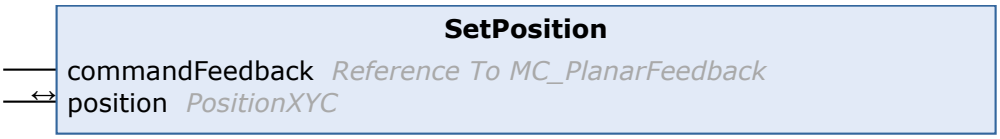
Updates internal state of the object, must be triggered each cycle.

Syntax

Definition:

```
METHOD Update
```

7.2.2.8.17 SetPosition



Sets the position of the Planar Mover. Only possible if the Planar Mover is disabled.

Syntax

Definition:

```
METHOD SetPosition
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
VAR_IN_OUT
    position        : PositionXYC;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

In/Outputs

Name	Type	Description
position	PositionXYC	New position of the Planar Mover.

7.2.2.8.18 StartExternalSetpointGeneration

StartExternalSetpointGeneration
commandFeedback *Reference To MC_PlanarFeedback*
options *Reference To ST_ExternalSetpointGenerationOptions*

Starts the external setpoint generation, the user must supply setpoints from this PLC cycle on in every PLC cycle.

Syntax

Definition:

```
METHOD StartExternalSetpointGeneration
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
    options         : Reference To ST_ExternalSetpointGenerationOptions;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
options	Reference To ST_ExternalSetpointGenerationOptions [▶ 131]	Options for the movement.

7.2.2.8.19 StopExternalSetpointGeneration

StopExternalSetpointGeneration
commandFeedback *Reference To MC_PlanarFeedback*

Ends the external setpoint generation, called after last SetExternalSetpoint (in the same PLC cycle).

Syntax

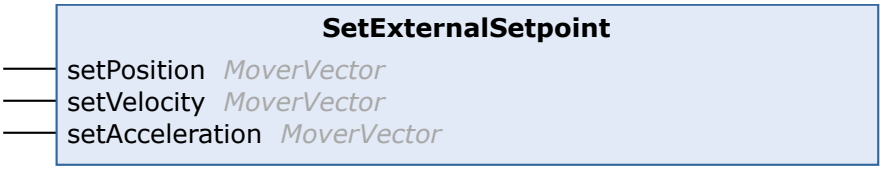
Definition:

```
METHOD StopExternalSetpointGeneration
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.8.20 SetExternalSetpoint



Sets the external setpoint for the Planar Mover without ref sys id (for relative mode), must be called each PLC cycle during external setpoint generation.

Syntax

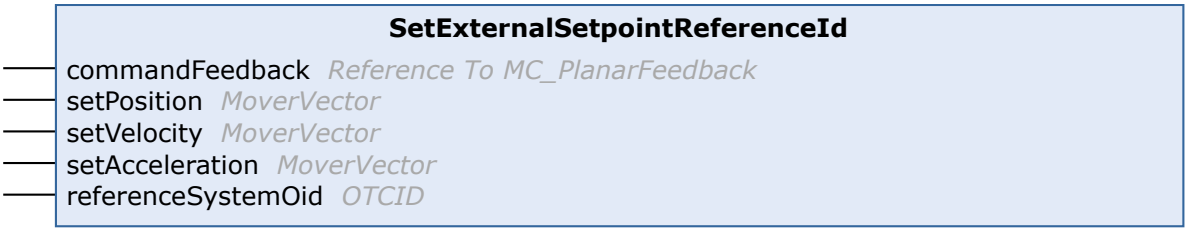
Definition:

```
METHOD SetExternalSetpoint
VAR_INPUT
    setPosition     : MoverVector;
    setVelocity     : MoverVector;
    setAcceleration : MoverVector;
END_VAR
```

Inputs

Name	Type	Description
setPosition	MoverVector	Position that is send to the Planar Mover.
setVelocity	MoverVector	Velocity that is send to the Planar Mover.
setAcceleration	MoverVector	Acceleration that is send to the Planar Mover.

7.2.2.8.21 SetExternalSetpointReferenceId



Sets the external setpoint for the Planar Mover with corresponing reference system id, must be called each PLC cycle during external setpoint generation.

Syntax

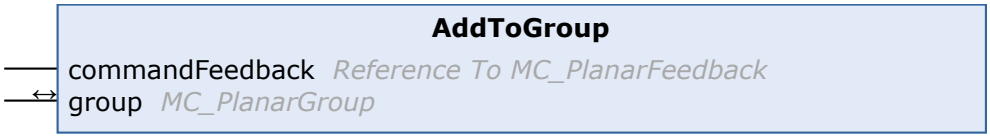
Definition:

```
METHOD SetExternalSetpointReferenceId
VAR_INPUT
    commandFeedback    : Reference To MC_PlanarFeedback;
    setPosition        : MoverVector;
    setVelocity        : MoverVector;
    setAcceleration    : MoverVector;
    referenceSystemOid : OTCID;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
setPosition	MoverVector	Position that is send to the Planar Mover.
setVelocity	MoverVector	Velocity that is send to the Planar Mover.
setAcceleration	MoverVector	Acceleration that is send to the Planar Mover.
referenceSystemOid	OTCID	Part or coordinate system id.

7.2.2.8.22 AddToGroup



Adds the Planar Mover to the given Planar Group.

Syntax

Definition:

```
METHOD AddToGroup
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
VAR_IN_OUT
    group           : MC_PlanarGroup;
END_VAR
```

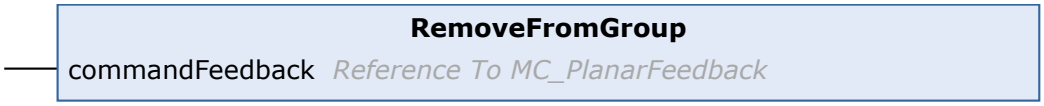
Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

In/Outputs

Name	Type	Description
group	MC_PlanarGroup [▶ 143]	The Planar Group that the Planar Mover joins.

7.2.2.8.23 RemoveFromGroup



Removes the Planar Mover from its current Planar Group, i.e. disables collision checks.

Syntax

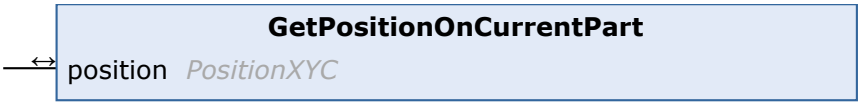
Definition:

```
METHOD RemoveFromGroup
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.8.24 GetPositionOnCurrentPart



Sets the values of the given position to the movers position values on the current part.

Syntax

Definition:

```
METHOD GetPositionOnCurrentPart
VAR_IN_OUT
    position : PositionXYC;
END_VAR
```

In/Outputs

Name	Type	Description
position	PositionXYC	The position on the planar part.

7.2.2.8.25 GetPlanarObjectInfo

Returns mover object info (type: mover, id: OID of nc mover).

Syntax

Definition:

```
METHOD GetPlanarObjectInfo : PlanarObjectInfo
```

Return value

PlanarObjectInfo [▶ 130]

7.2.2.9 MC_PlanarPart

A Planar Part object represents the area that Planar Movers can move on. It contains information about the stator objects.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
Initialize [▶ 161]	Initialize the Planar Part, i.e. connecting it to the MC via its OID.
ActivatePosition [▶ 162]	Activates the position given by part position index.
AllowEnable [▶ 162]	From now on the part can be enabled until ForceDisablePart is called.
ForceDisable [▶ 163]	Disables the part and keeps it disabled until AllowEnabledPart is called.
Reset [▶ 163]	Resets the part.
GetPosition [▶ 163]	Sets the values of the given position to the parts position values.

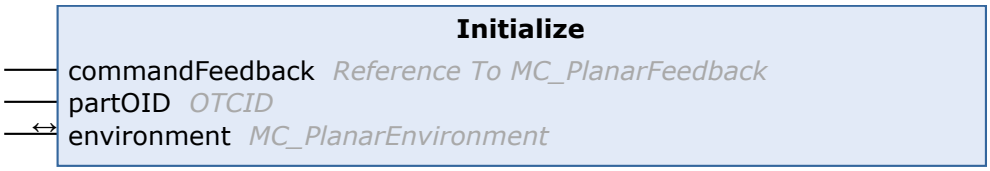
Required License

TC3 Planar Motion Base

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.40 Advanced Motion Pack V3.2.60	PC or CX (x64)	Tc3_Mc3PlanarMotion, Tc3_Physics

7.2.2.9.1 Initialize



Initialize the Planar Part, i.e. connecting it to the MC via its OID.

Syntax

Definition:

```
METHOD Initialize
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
    partOID         : OTCID;
END_VAR
VAR_IN_OUT
    environment     : MC_PlanarEnvironment;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
partOID	OTCID	OID of the part.

In/Outputs

Name	Type	Description
environment	MC_PlanarEnvironment [▶ 136]	Environment the part is in.

7.2.2.9.2 ActivatePosition

Activates the position given by part position index.

Syntax

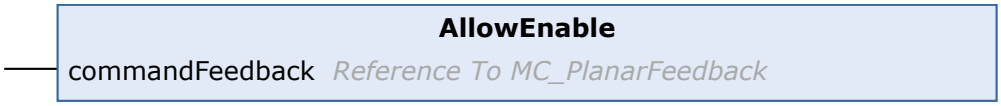
Definition:

```
METHOD ActivatePosition
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
    positionIndex   : UDINT;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
positionIndex	UDINT	Index of the position.

7.2.2.9.3 AllowEnable



From now on the part can be enabled until ForceDisablePart is called.

Syntax

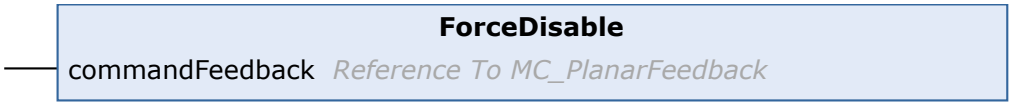
Definition:

```
METHOD AllowEnable
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.9.4 ForceDisable



Disables the part and keeps it disabled until AllowEnabledPart is called.

Syntax

Definition:

```
METHOD ForceDisable
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.9.5 Reset

Resets the part.

Syntax

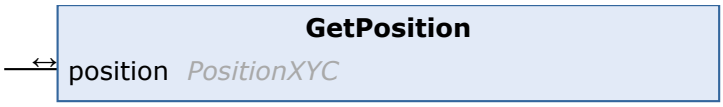
Definition:

```
METHOD Reset
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.9.6 GetPosition



Sets the values of the given position to the parts position values.

Syntax

Definition:

```
METHOD GetPosition
VAR_IN_OUT
    position : PositionXYC;
END_VAR
```

In/Outputs

Name	Type	Description
position	PositionXYC	The position of the Planar Part.

7.2.2.10 MC_PlanarTrack

A track within a plane which Planar Movers can follow. Planar Movers on the track automatically avoid collisions with each other. The Planar Track can consist of several consecutive segments and be joined with other Planar Tracks at its start/end.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
Clear [▶ 165]	Clears the geometric information of the Planar Track.
AppendPosition [▶ 165]	Appends a position to the Planar Track.
AppendLine [▶ 166]	Appends a line to the Planar Track.
AppendCircle [▶ 166]	Appends a circular arc to the Planar Track.
CloseLoop [▶ 167]	Closes the loop of the Planar Track, no further part can be appended.
StartFromTrack [▶ 167]	Sets the other Planar Track's endpoint as start point of this Planar Track, transition is smooth. The other Planar Track is blocked for further changes (until it is cleared).
EndAtTrack [▶ 168]	Appends a smooth transition from the end of this Planar Track to the other Planar Track's start point. The Planar Track is blocked for further changes (until it is cleared).
StartFromTrackAdvanced [▶ 169]	Sets the other Planar Track's endpoint as start point of this Planar Track, transition is smooth. The other Planar Track is blocked for further geometrical changes (until it is cleared).
EndAtTrackAdvanced [▶ 169]	Appends a smooth transition from the end of this Planar Track to the other Planar Track's start point. The Planar Track is blocked for further geometrical changes (until it is cleared).
Enable [▶ 170]	Starts enabling the Planar Track.
Disable [▶ 170]	Starts disabling the Planar Track.
Reset [▶ 171]	Starts resetting the Planar Track.
GetArcLengthClosestTo [▶ 171]	Calculate the arc length value where the Planar Track is closest to a geometry's center point.
GetPositionAt [▶ 172]	Get a position on the Planar Track at a specific arc length value.
GetLength [▶ 172]	Returns the Planar Track's length, -1 return value indicates no connection to Nc Track.
GetPlanarObjectInfo [▶ 172]	Returns track object info (type: track, id: OID of nc track).
Update [▶ 173]	Updates internal state of the object, must be triggered each cycle.
AddToGroup [▶ 173]	Adds the Planar Track to the given Planar Group.
RemoveFromGroup [▶ 173]	Removes the Planar Track from its current Planar Group, i.e. disables collision checks.

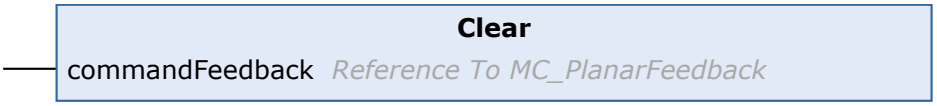
Required License

TC3 Planar Motion Base

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.11	PC or CX (x64)	Tc3_Mc3PlanarMotion, Tc3_Physics

7.2.2.10.1 Clear



Clears the geometric information of the Planar Track.

Syntax

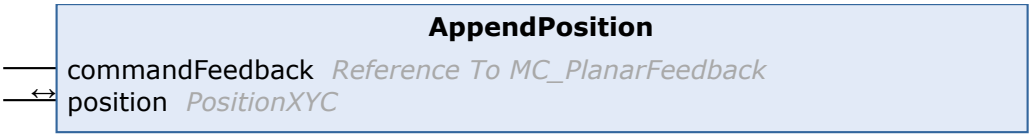
Definition:

```
METHOD Clear
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.10.2 AppendPosition



Appends a position to the Planar Track.

Syntax

Definition:

```
METHOD AppendPosition
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
VAR_IN_OUT
    position        : PositionXYC;
END_VAR
```

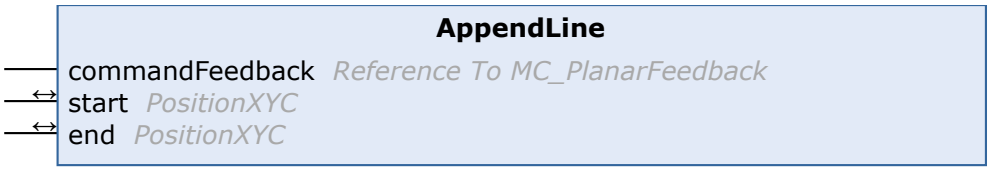
Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

In/Outputs

Name	Type	Description
position	PositionXYC	Position that is the new endpoint of the Planar Track.

7.2.2.10.3 AppendLine



Appends a line to the Planar Track.

Syntax

Definition:

```
METHOD AppendLine
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
VAR_IN_OUT
    start           : PositionXYC;
    end             : PositionXYC;
END_VAR
```

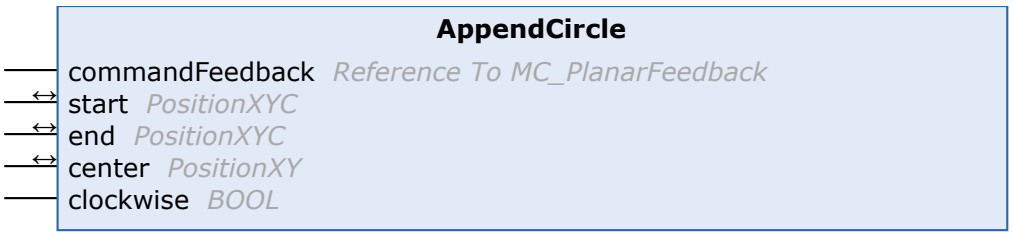
Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

In/Outputs

Name	Type	Description
start	PositionXYC	Start position of the line.
end	PositionXYC	End position of the line, this position is the new endpoint of the Planar Track.

7.2.2.10.4 AppendCircle



Appends a circular arc to the Planar Track.

Syntax

Definition:

```
METHOD AppendCircle
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

```
VAR_IN_OUT
    start           : PositionXYC;
    end             : PositionXYC;
    center          : PositionXY;
END_VAR
VAR_INPUT
    clockwise       : BOOL;
END_VAR
```

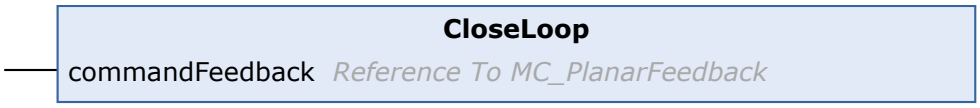
Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
clockwise	BOOL	Indicates if the clockwise circle is appended.

In/Outputs

Name	Type	Description
start	PositionXYC	Start position of the circular arc.
end	PositionXYC	End position of the circular arc, this position is the new endpoint of the Planar Track.
center	PositionXY	Center of the circular arc.

7.2.2.10.5 CloseLoop



Closes the loop of the Planar Track, no further part can be appended.

Syntax

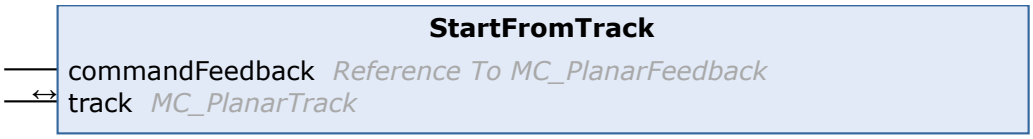
Definition:

```
METHOD CloseLoop
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.10.6 StartFromTrack



Sets the other Planar Track's endpoint as start point of this Planar Track, transition is smooth. The other Planar Track is blocked for further changes (until it is cleared).

Syntax

Definition:

```
METHOD StartFromTrack
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
VAR_IN_OUT
    track           : MC_PlanarTrack;
END_VAR
```

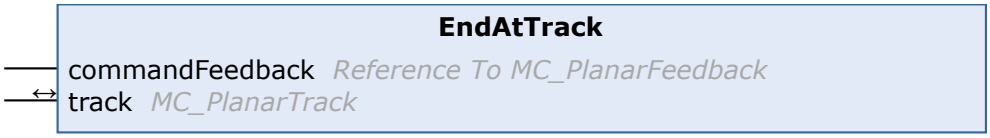
Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

In/Outputs

Name	Type	Description
track	MC_PlanarTrack [▶ 164]	The other Planar Track.

7.2.2.10.7 EndAtTrack



Appends a smooth transition from the end of this Planar Track to the other Planar Track's start point. The Planar Track is blocked for further changes (until it is cleared).

Syntax

Definition:

```
METHOD EndAtTrack
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
VAR_IN_OUT
    track           : MC_PlanarTrack;
END_VAR
```

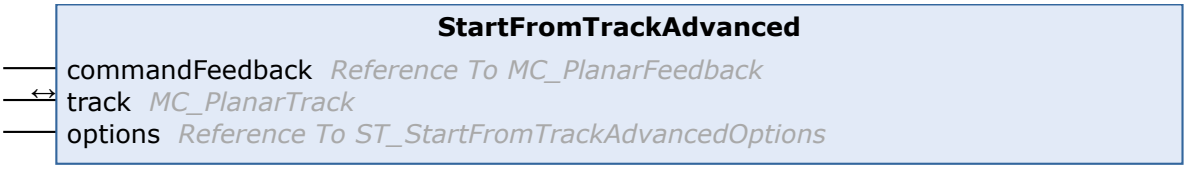
Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

In/Outputs

Name	Type	Description
track	MC_PlanarTrack [▶ 164]	The other Planar Track.

7.2.2.10.8 StartFromTrackAdvanced



Sets the other Planar Track's endpoint as start point of this Planar Track, transition is smooth. The other Planar Track is blocked for further geometrical changes (until it is cleared).

Syntax

Definition:

```
METHOD StartFromTrackAdvanced
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
VAR_IN_OUT
    track           : MC_PlanarTrack;
END_VAR
VAR_INPUT
    options         : Reference To ST_StartFromTrackAdvancedOptions;
END_VAR
```

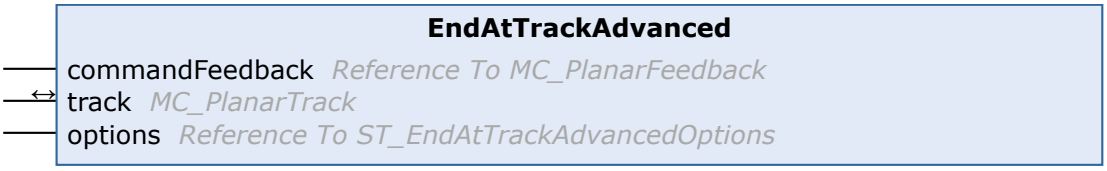
Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
options	Reference To ST_StartFromTrackAdvancedOptions [▶ 135]	Options for the connection, i.e. which connections are closed.

In/Outputs

Name	Type	Description
track	MC_PlanarTrack [▶ 164]	The other Planar Track.

7.2.2.10.9 EndAtTrackAdvanced



Appends a smooth transition from the end of this Planar Track to the other Planar Track's start point. The Planar Track is blocked for further geometrical changes (until it is cleared).

Syntax

Definition:

```
METHOD EndAtTrackAdvanced
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
VAR_IN_OUT
    track           : MC_PlanarTrack;
END_VAR
```

```
VAR_INPUT
    options         : Reference To ST_EndAtTrackAdvancedOptions;
END_VAR
```

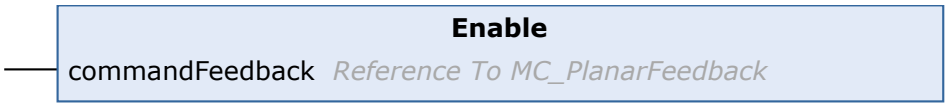
Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.
options	Reference To ST_EndAtTrackAdvancedOptions [▶ 131]	Options for the connection, i.e. which connections are closed.

In/Outputs

Name	Type	Description
track	MC_PlanarTrack [▶ 164]	The other Planar Track.

7.2.2.10.10 Enable



Starts enabling the Planar Track.

Syntax

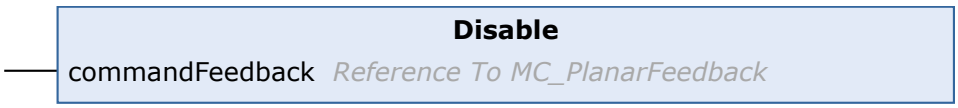
Definition:

```
METHOD Enable
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.10.11 Disable



Starts disabling the Planar Track.

Syntax

Definition:

```
METHOD Disable
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.10.12 Reset

Starts resetting the Planar Track.

Syntax

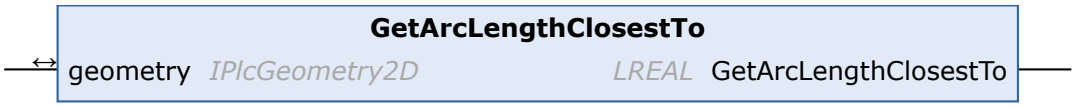
Definition:

```
METHOD Reset
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.10.13 GetArcLengthClosestTo



Calculate the arc length value where the Planar Track is closest to a geometry's center point.

Syntax

Definition:

```
METHOD GetArcLengthClosestTo : LREAL
VAR_IN_OUT
    geometry : IPlcGeometry2D;
END_VAR
```

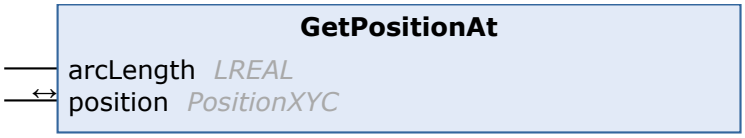
In/Outputs

Name	Type	Description
geometry	IPlcGeometry2D	The geometry to check the arc length for.

Return value

LREAL

7.2.2.10.14 GetPositionAt



Get a position on the Planar Track at a specific arc length value.

Syntax

Definition:

```
METHOD GetPositionAt
VAR_INPUT
    arcLength : LREAL;
END_VAR
VAR_IN_OUT
    position  : PositionXYC;
END_VAR
```

Inputs

Name	Type	Description
arcLength	LREAL	Arc length value where the position is evaluated.

In/Outputs

Name	Type	Description
position	PositionXYC	The position at the specified arc parameter.

7.2.2.10.15 GetLength

Returns the Planar Track's length, -1 return value indicates no connection to Nc Track.

Syntax

Definition:

```
METHOD GetLength : LREAL
```

Return value

LREAL

7.2.2.10.16 GetPlanarObjectInfo

Returns track object info (type: track, id: OID of nc track).

Syntax

Definition:

```
METHOD GetPlanarObjectInfo : PlanarObjectInfo
```

Return value

PlanarObjectInfo [▶ 130]

7.2.2.10.17 Update

Update

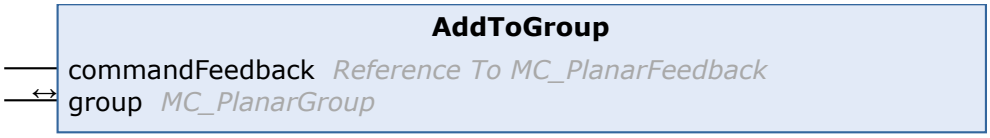
Updates internal state of the object, must be triggered each cycle.

Syntax

Definition:

```
METHOD Update
```

7.2.2.10.18 AddToGroup



Adds the Planar Track to the given Planar Group.

Syntax

Definition:

```
METHOD AddToGroup
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
VAR_IN_OUT
    group           : MC_PlanarGroup;
END_VAR
```

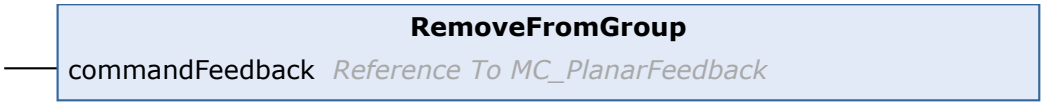
Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

In/Outputs

Name	Type	Description
group	MC_PlanarGroup [▶ 143]	The Planar Group that the mover joins.

7.2.2.10.19 RemoveFromGroup



Removes the Planar Track from its current Planar Group, i.e. disables collision checks.

Syntax

Definition:

```
METHOD RemoveFromGroup
VAR_INPUT
    commandFeedback : Reference To MC_PlanarFeedback;
END_VAR
```

Inputs

Name	Type	Description
commandFeedback	Reference To MC_PlanarFeedback [▶ 139]	The feedback object for the command.

7.2.2.11 MC_PlanarTrackTrail

A list of distinct tracks each starting at the ending vertex of its predecessor.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
Clear [▶ 174]	Clears the TrackTrail.
AddTrack [▶ 174]	Adds a track to the TrackTrail. The track should start at the end vertex of the currently last track.

Required License

TC3 Planar Motion Base

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.11	PC or CX (x64)	Tc3_Mc3PlanarMotion, Tc3_Physics

7.2.2.11.1 Clear

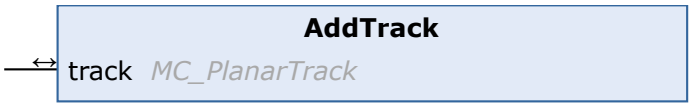
Clears the TrackTrail.

Syntax

Definition:

```
METHOD Clear
```

7.2.2.11.2 AddTrack



Adds a track to the TrackTrail. The track should start at the end vertex of the currently last track.

Syntax

Definition:

```
METHOD AddTrack
VAR_IN_OUT
    track : MC_PlanarTrack;
END_VAR
```

In/Outputs

Name	Type	Description
track	MC_PlanarTrack [▶ 164]	The track to be added to the end of the TrackTrail.

8 Support und Service

Beckhoff und seine weltweiten Partnerfirmen bieten einen umfassenden Support und Service, der eine schnelle und kompetente Unterstützung bei allen Fragen zu Beckhoff Produkten und Systemlösungen zur Verfügung stellt.

Downloadfinder

Unser Downloadfinder beinhaltet alle Dateien, die wir Ihnen zum Herunterladen anbieten. Sie finden dort Applikationsberichte, technische Dokumentationen, technische Zeichnungen, Konfigurationsdateien und vieles mehr.

Die Downloads sind in verschiedenen Formaten erhältlich.

Beckhoff Niederlassungen und Vertretungen

Wenden Sie sich bitte an Ihre Beckhoff Niederlassung oder Ihre Vertretung für den lokalen Support und Service zu Beckhoff Produkten!

Die Adressen der weltweiten Beckhoff Niederlassungen und Vertretungen entnehmen Sie bitte unserer Internetseite: www.beckhoff.com

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

Beckhoff Support

Der Support bietet Ihnen einen umfangreichen technischen Support, der Sie nicht nur bei dem Einsatz einzelner Beckhoff Produkte, sondern auch bei weiteren umfassenden Dienstleistungen unterstützt:

- Support
- Planung, Programmierung und Inbetriebnahme komplexer Automatisierungssysteme
- umfangreiches Schulungsprogramm für Beckhoff Systemkomponenten

Hotline:	+49 5246 963-157
E-Mail:	support@beckhoff.com

Beckhoff Service

Das Beckhoff Service-Center unterstützt Sie rund um den After-Sales-Service:

- Vor-Ort-Service
- Reparaturservice
- Ersatzteilservice
- Hotline-Service

Hotline:	+49 5246 963-460
E-Mail:	service@beckhoff.com

Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

Telefon:	+49 5246 963-0
E-Mail:	info@beckhoff.com
Internet:	www.beckhoff.com

Mehr Informationen:
www.beckhoff.com/TF5430

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