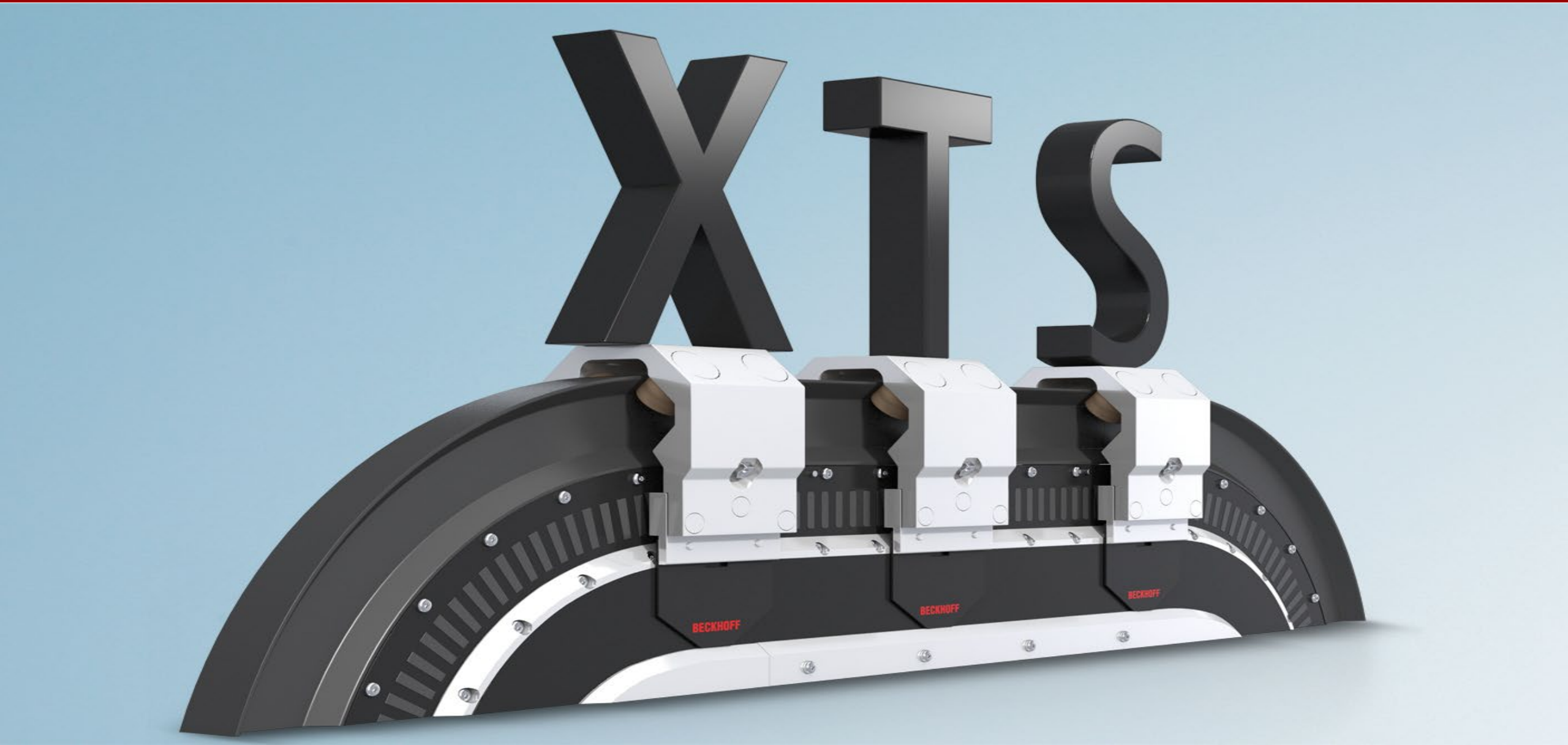
New Automation Technology
Beckhoff Automation

BECKHOFF

XTS Training

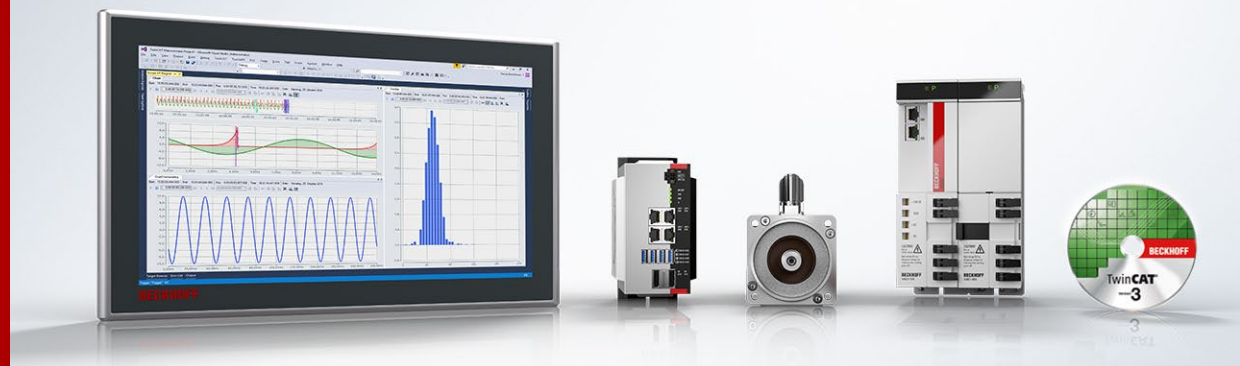
Beckhoff-Training | TR3056 : Q2/2023

BECKHOFF



BECKHOFF

XTS – PLC



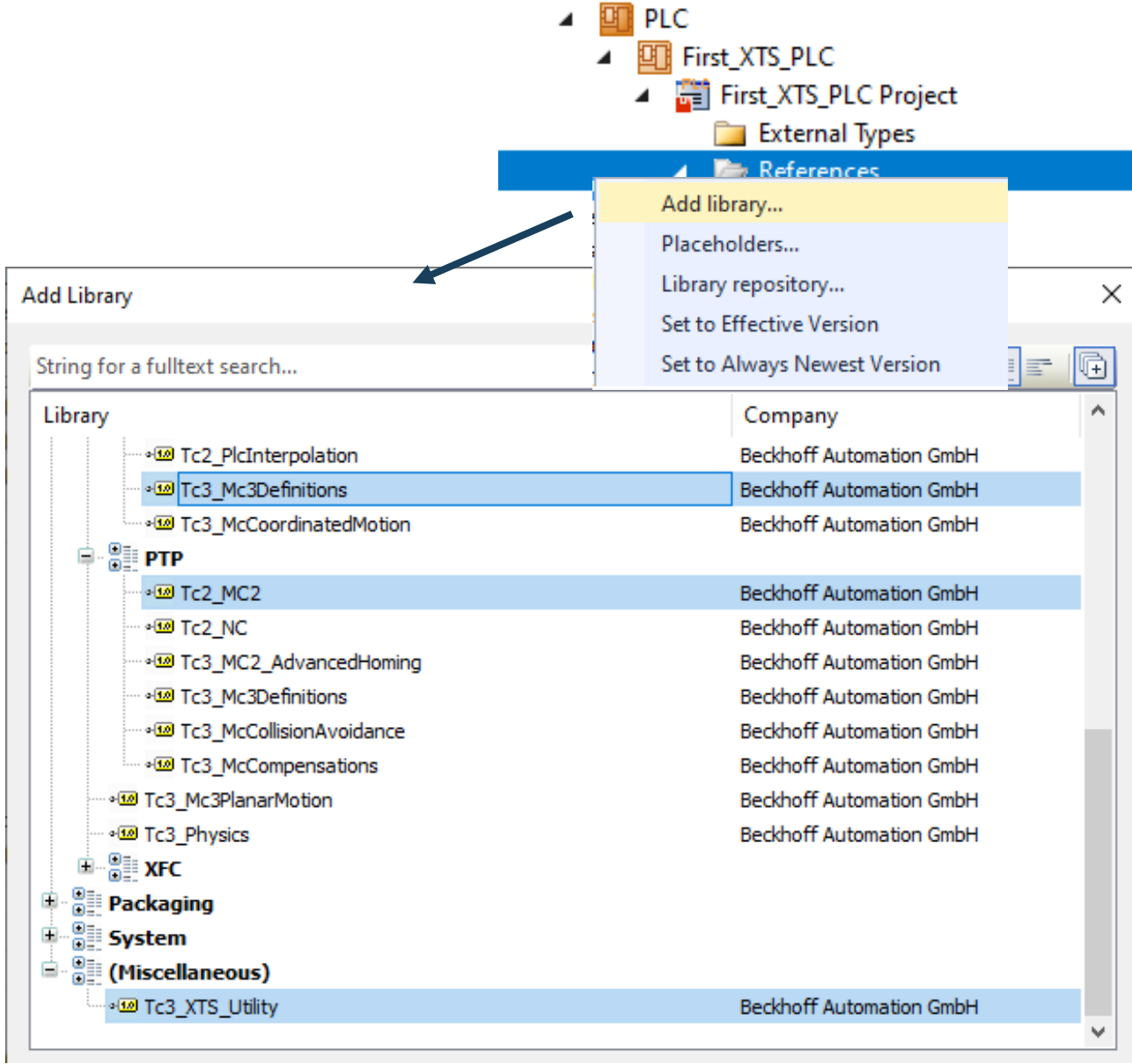
Agenda | XTS – PLC

BECKHOFF

1. **PLC-Library XTS**
2. AXIS_REF
3. XTS-Utility Library
4. PLC StartUp condition
5. Mover1 detection via PLC
6. Overview TF5400 Collision Avoidance (CA)
7. CA-Group Object
8. PLC-Library CA-Group
9. AXES_GROUP_REF
10. CA-Group handling
11. CA-Operation

Necessary PLC-Libraries

XTS



Necessary PLC-Libraries

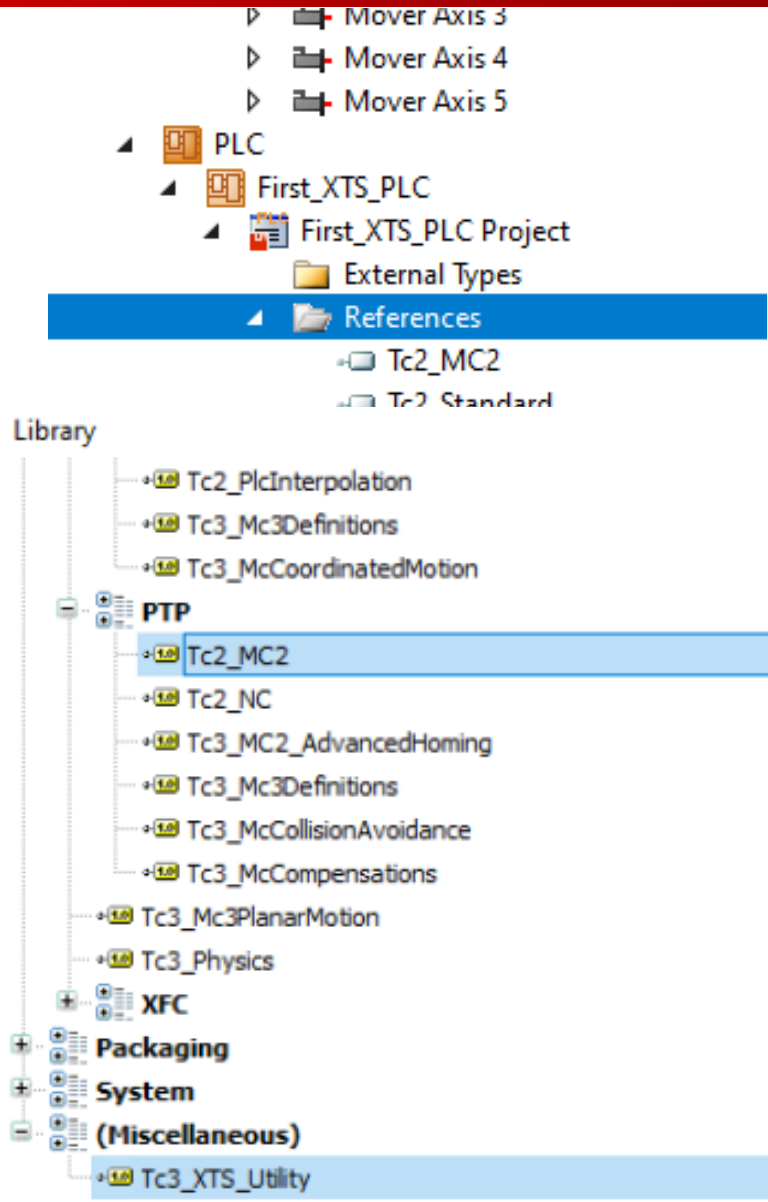
XTS

- Tc2_MC2

  Library containing PLCOpen standardized motion control function blocks

- Tc3_XTS_Utility

  Library containing Diagnosis and Visualization for the XTS-System



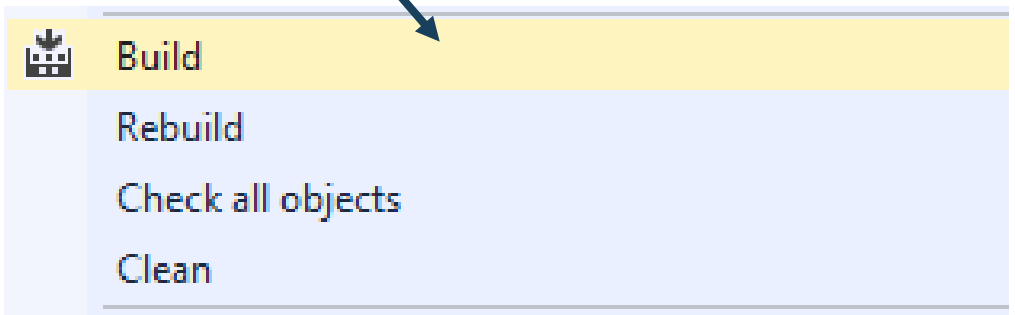
Agenda | XTS – PLC

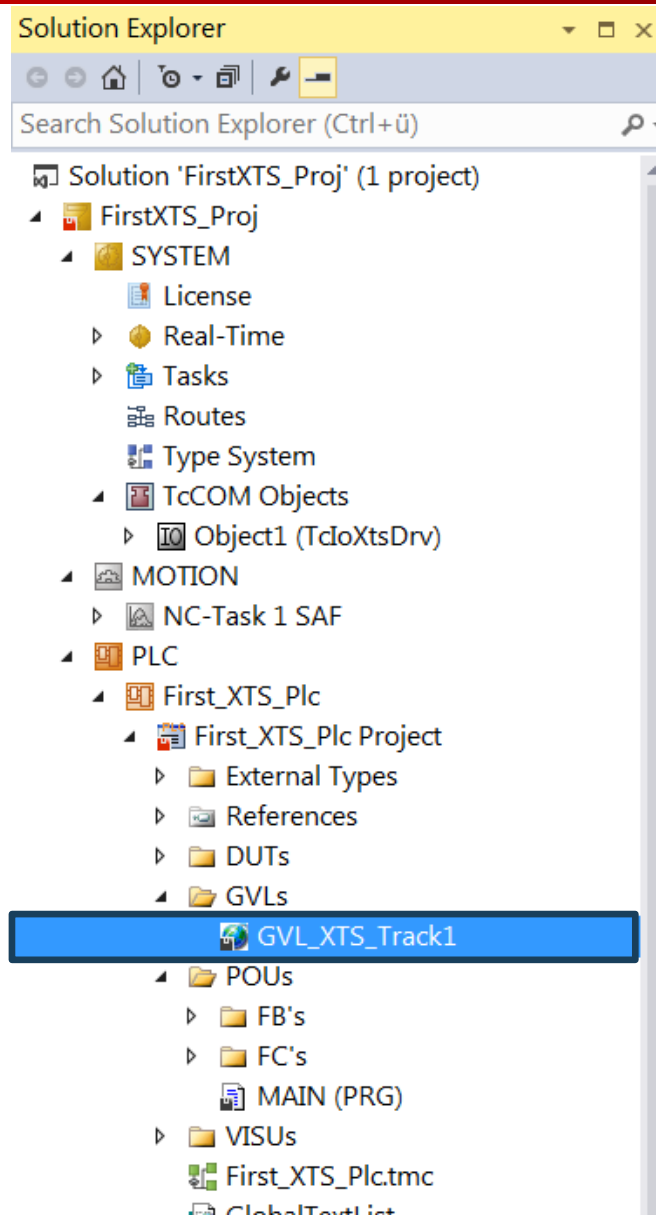
BECKHOFF

1. PLC-Library XTS
2. **AXIS_REF**
3. XTS-Utility Library
4. PLC StartUp condition
5. Mover1 detection via PLC
6. Overview TF5400 Collision Avoidance (CA)
7. CA-Group Object
8. PLC-Library CA-Group
9. AXES_GROUP_REF
10. CA-Group handling
11. CA-Operation

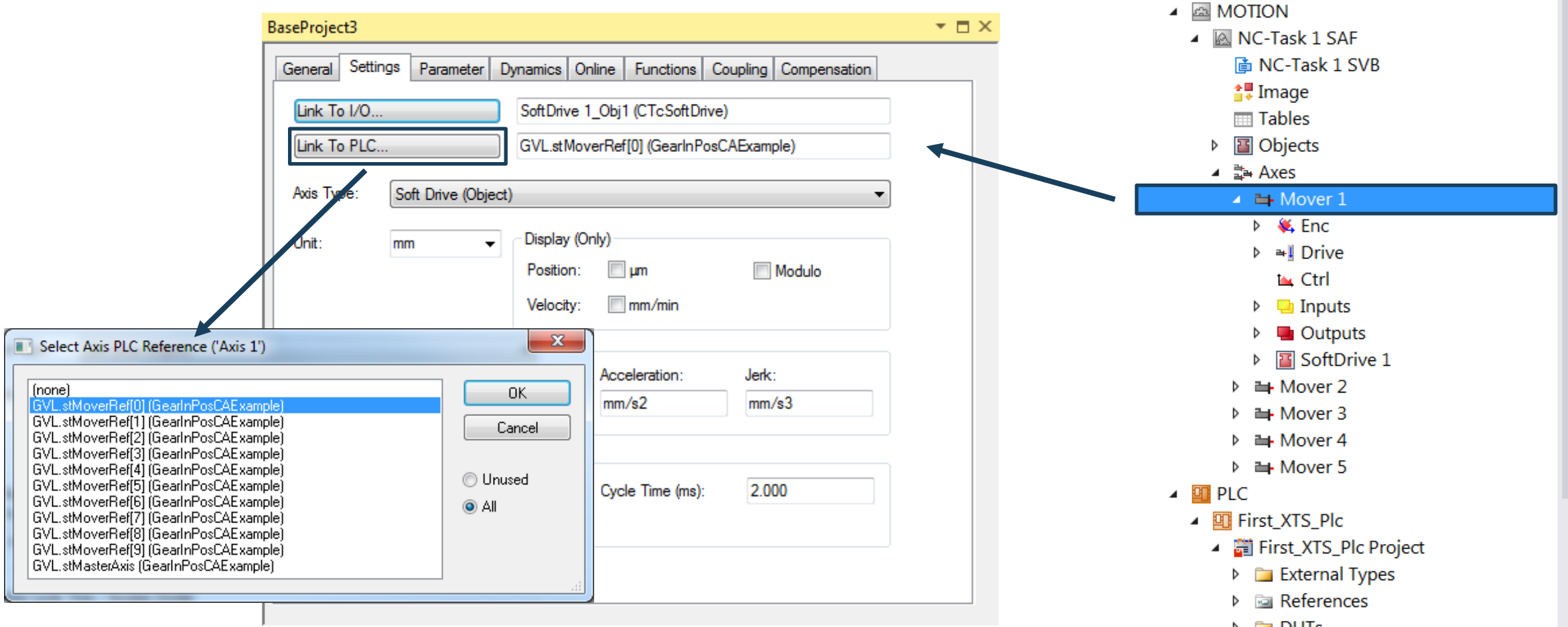
- **AXIS_REF for linking the PLC with the Axis**
  - definition of AXIS_REF
    - AXIS_REF is an interface between the PLC and the NC. It is added to MC function blocks as axis reference.

```
VAR_GLOBAL CONSTANT
    // Number of Movers (Starting by 1)
    gciNumMovers        : INT := 5 ;
END_VAR

VAR_GLOBAL
    // IO-Interface to MoverModule
    stMoverRef          : ARRAY [1..gciNumMovers] OF AXIS_REF;
END_VAR
```





- **AXIS_REF for linking the PLC with the Axis**
  - link the AXIS_REF to the Axis (Mover)

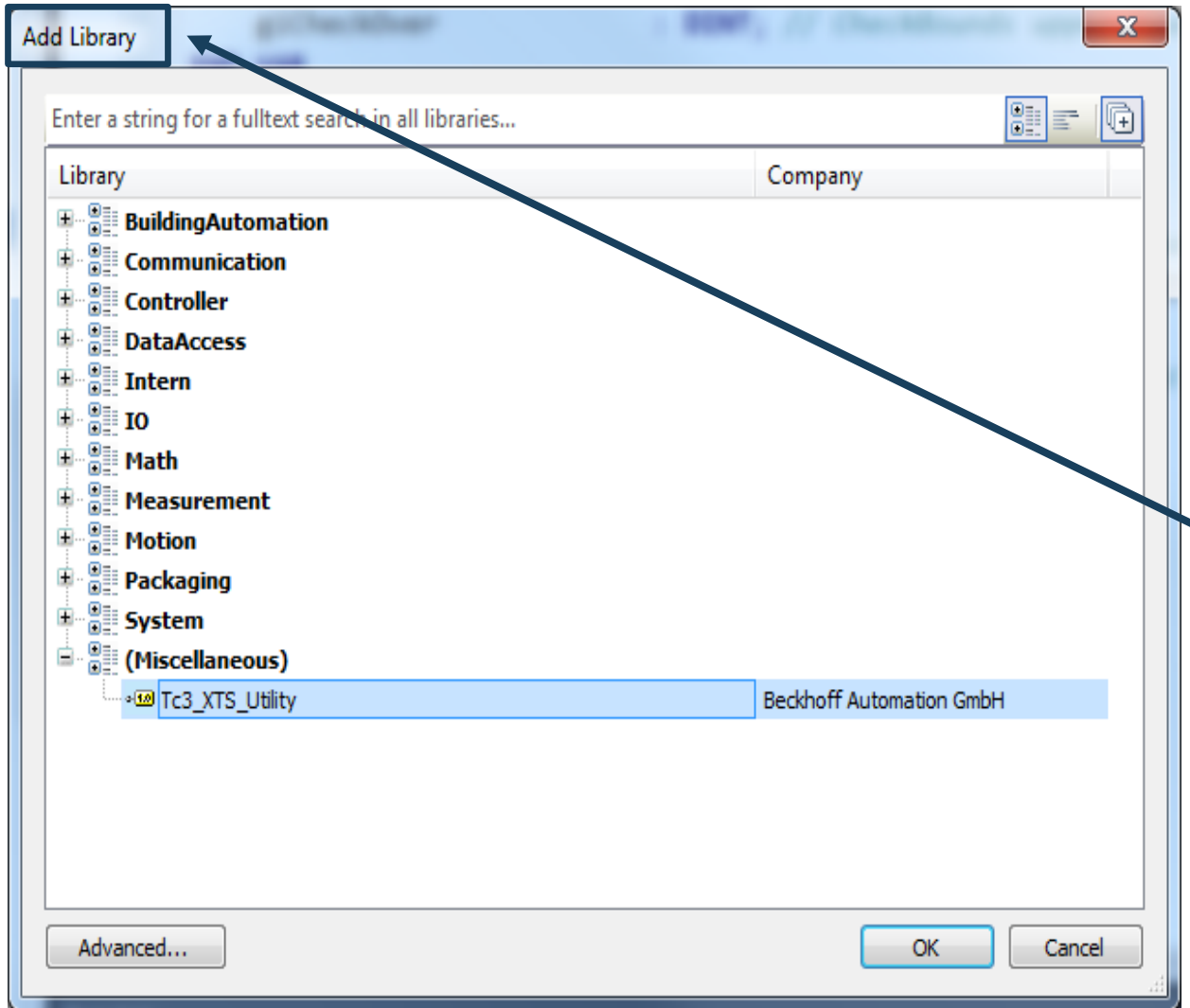


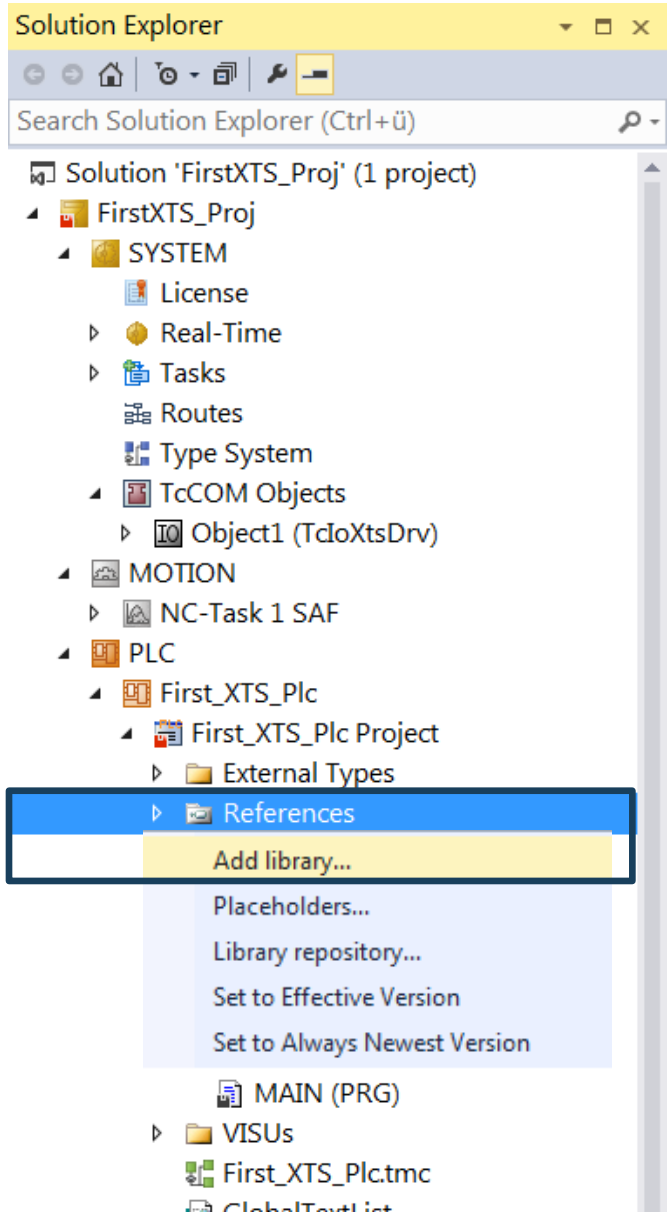
Agenda | XTS – PLC

BECKHOFF

1. PLC-Library XTS
2. AXIS_REF
3. **XTS-Utility Library**
4. PLC StartUp condition
5. Mover1 detection via PLC
6. Overview TF5400 Collision Avoidance (CA)
7. CA-Group Object
8. PLC-Library CA-Group
9. AXES_GROUP_REF
10. CA-Group handling
11. CA-Operation

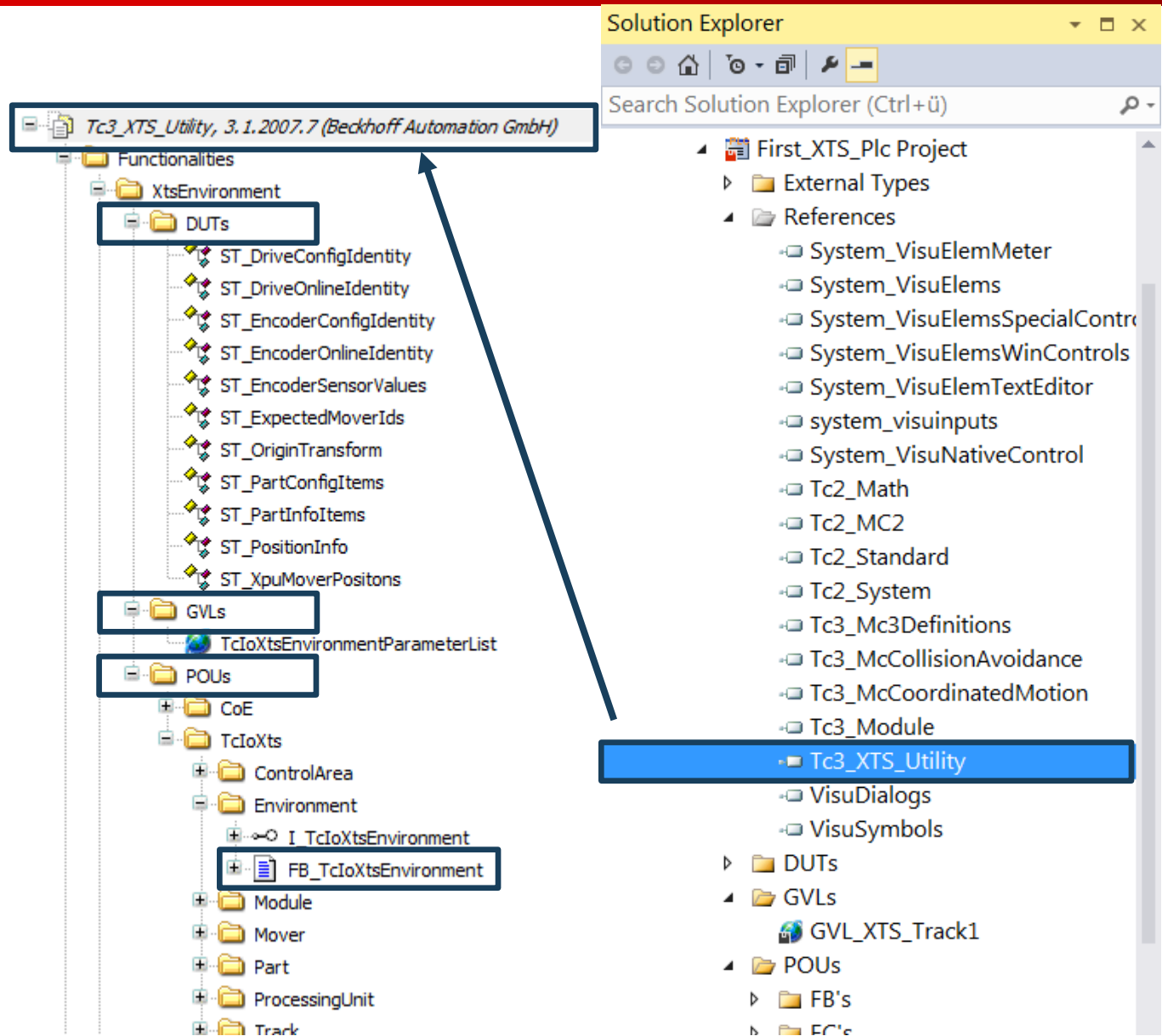
- **XTS-Utility PLC-Libraries**





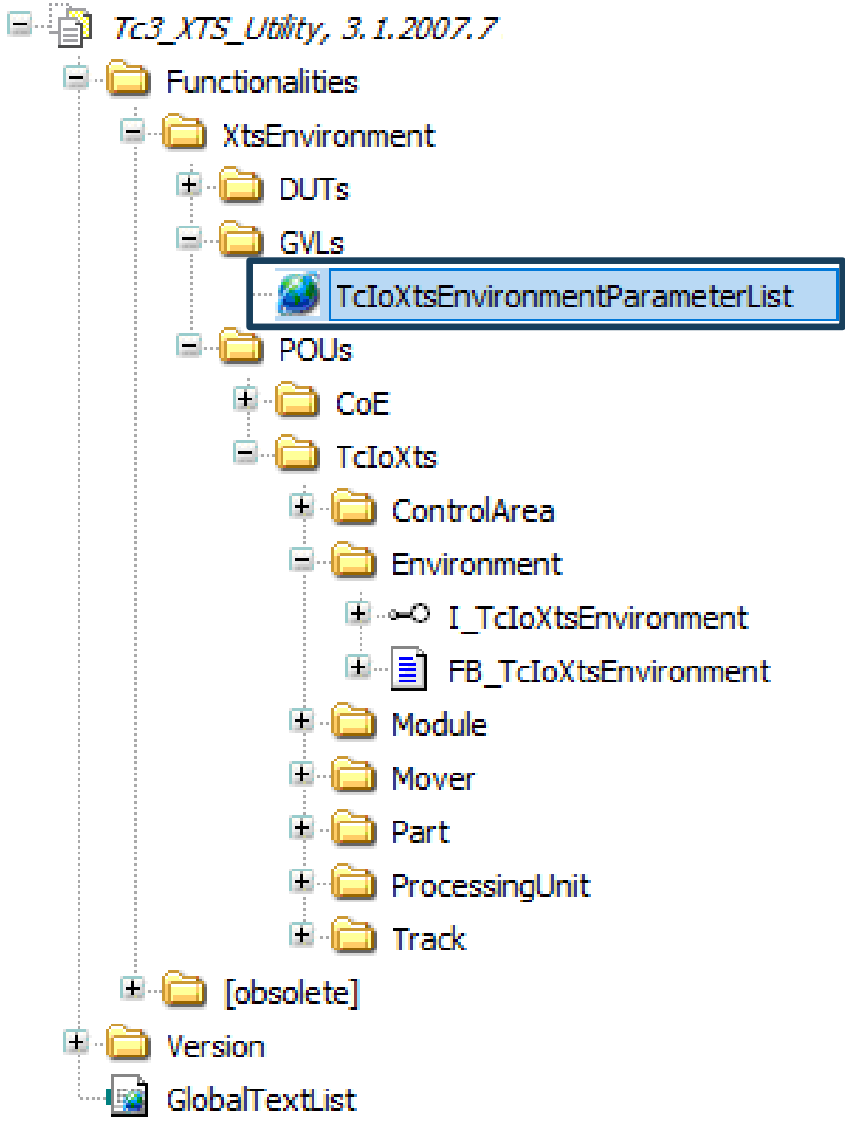
XTS-Utility PLC-Libraries contains

- DUTs
- GVLs
  - TcIoXtsEnvironmentParameterList
- POUs
  - FB_TcIoXtsEnvironment



TcIoXtsEnvironmentParameterList

The constant list can be changed in the library.



TcIoXtsEnvironmentParameterList (PARAMS)

InOut:

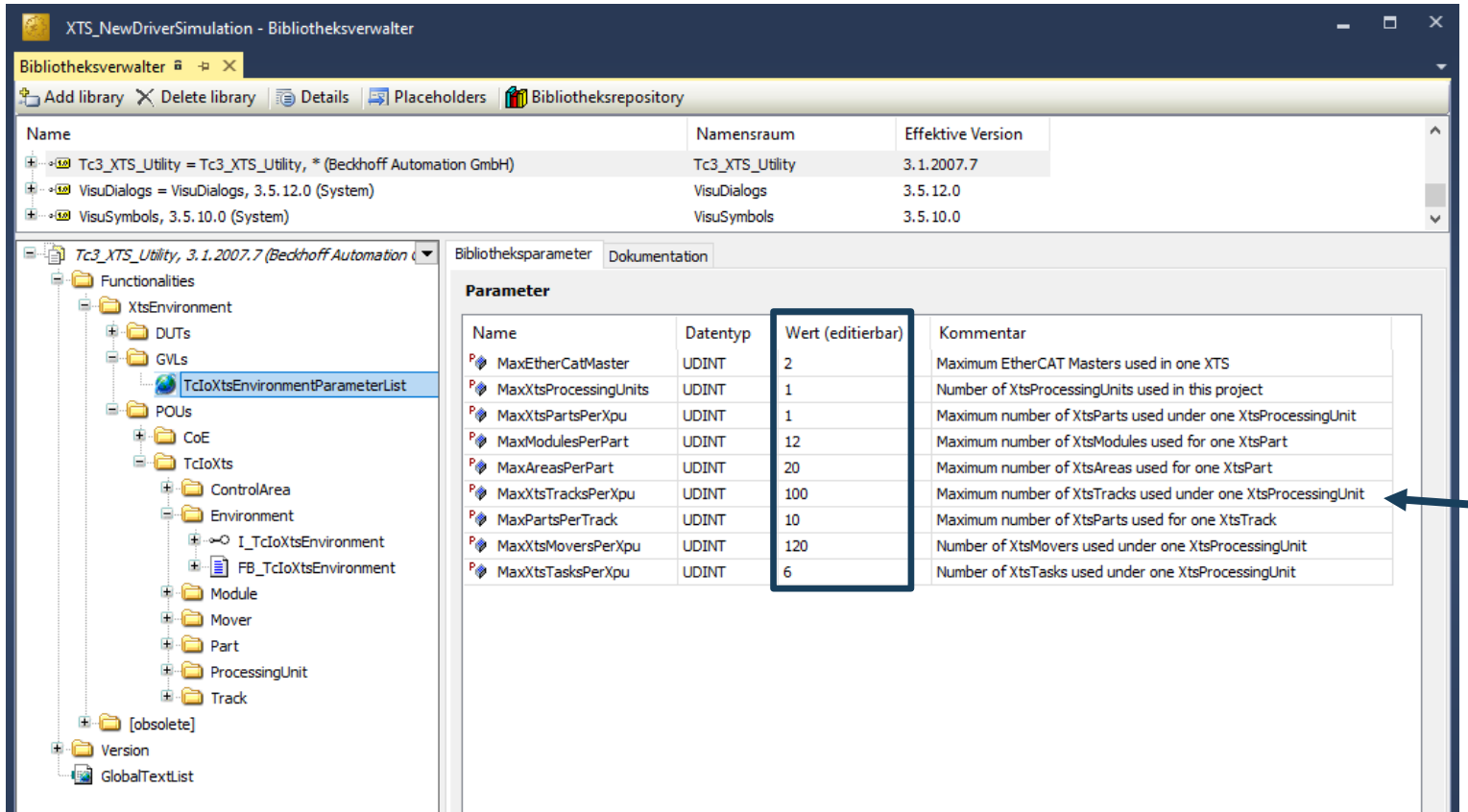
Scope	Name	Type	Initial	Comment
Constant	MaxEtherCatMaster	UDINT	32	Maximum EtherCAT Masters used in one XTS
	MaxXtsProcessingUnits	UDINT	5	Number of XtsProcessingUnits used in this project
	MaxXtsPartsPerXpu	UDINT	10	Maximum number of XtsParts used under one XtsProcessingUnit
	MaxModulesPerPart	UDINT	128	Maximum number of XtsModules used for one XtsPart
	MaxAreasPerPart	UDINT	20	Maximum number of XtsAreas used for one XtsPart
	MaxXtsTracksPerXpu	UDINT	100	Maximum number of XtsTracks used under one XtsProcessingUnit
	MaxPartsPerTrack	UDINT	10	Maximum number of XtsParts used for one XtsTrack
	MaxXtsMoversPerXpu	UDINT	120	Number of XtsMovers used under one XtsProcessingUnit
	MaxXtsTasksPerXpu	UDINT	6	Number of XtsTasks used under one XtsProcessingUnit

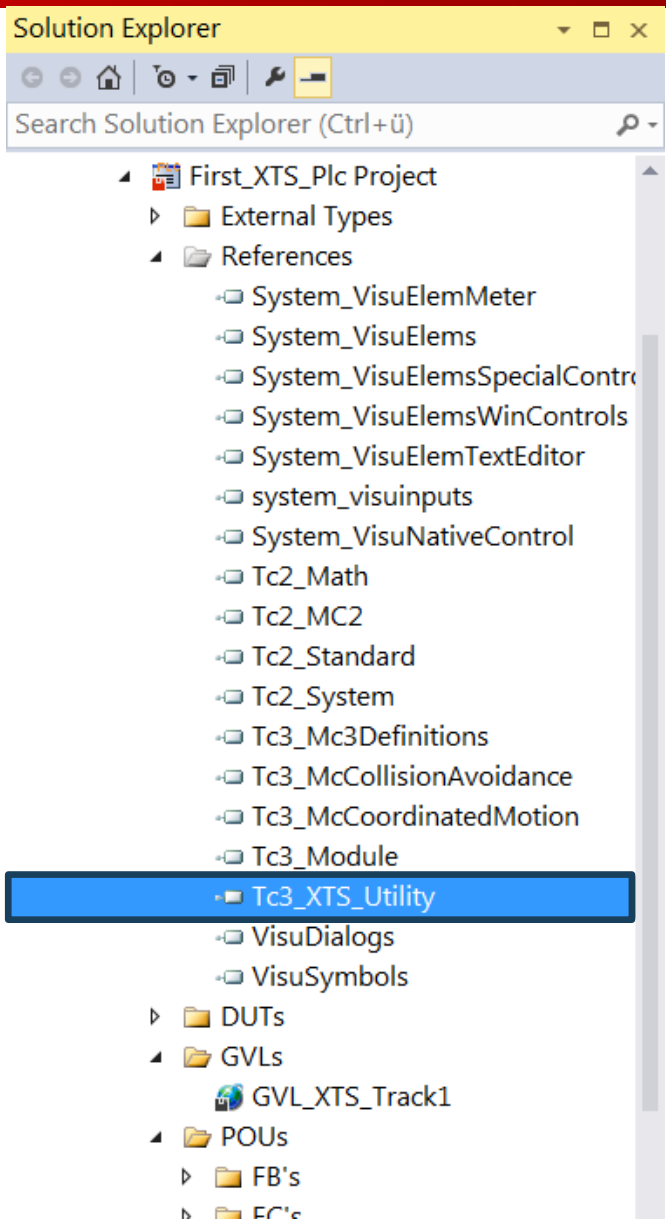
The parameters must be adapted for the respective system

TcIoXtsEnvironmentParameterList

To change this Parameters

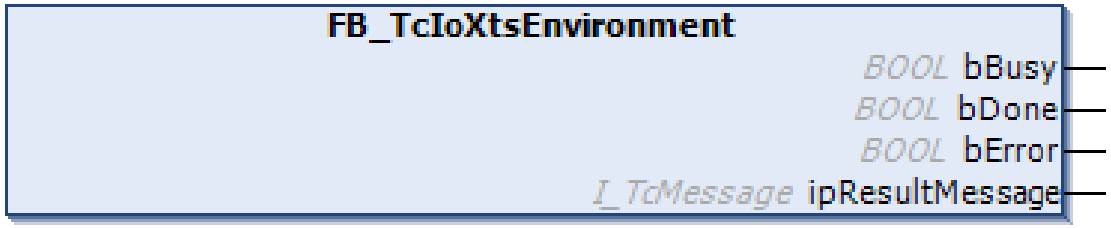
“Double-click” the entry within the library.

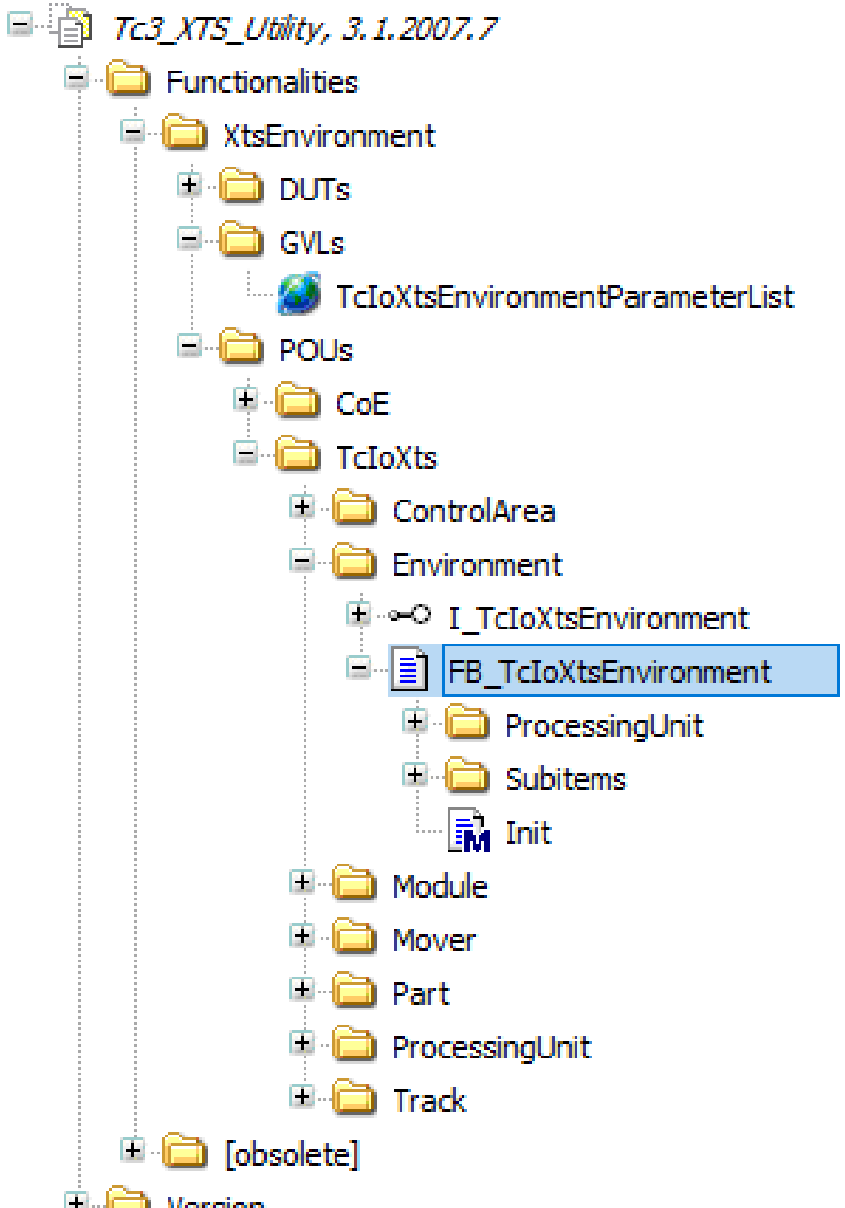




FB_TcIoXtsEnvironment

For easy gathering of XTS diagnostic data and accessing object parameters in the PLC.





FB_TcIoXtsEnvironment

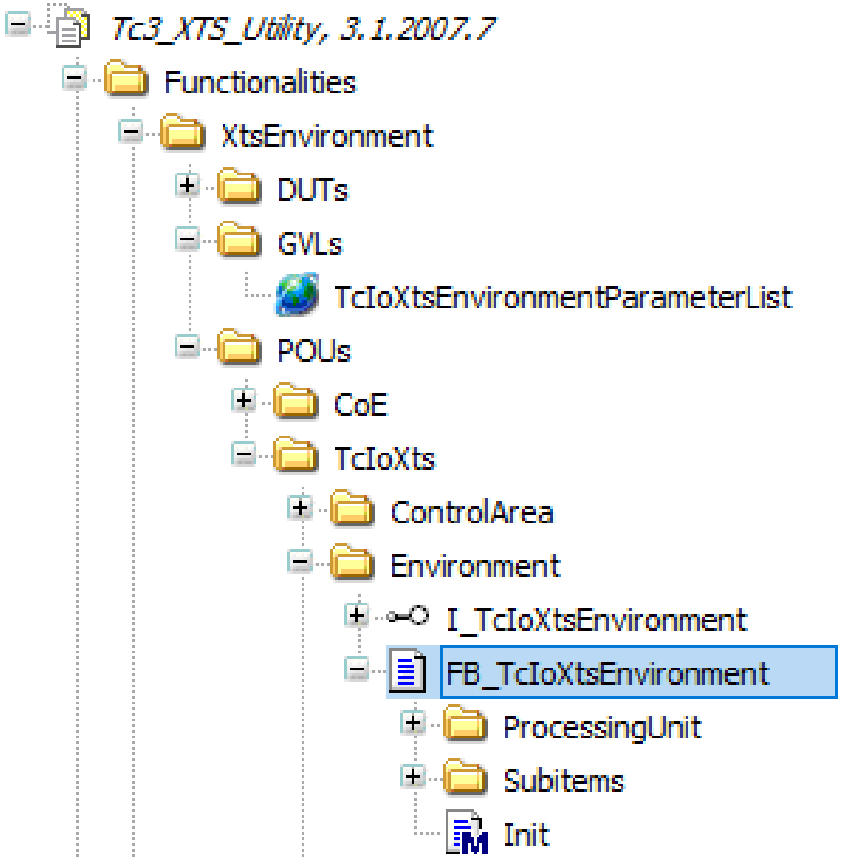
FB_TcIoXtsEnvironment (FB)

FUNCTION_BLOCK FB_TcIoXtsEnvironment IMPLEMENTS I_TcIoXtsEnvironment

InOut:

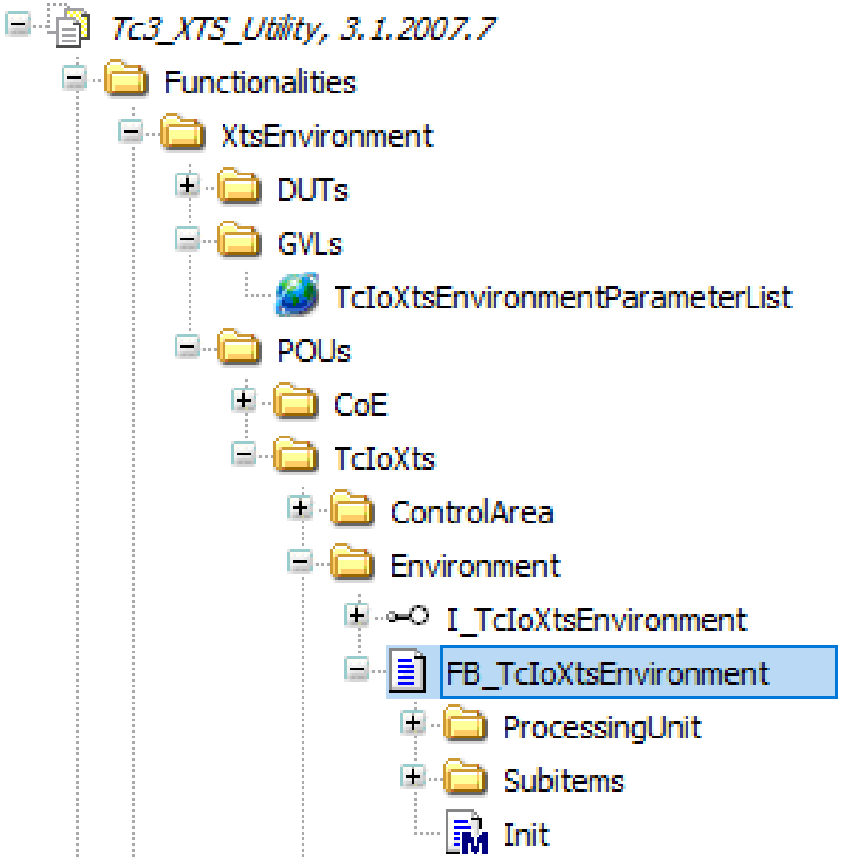
Scope	Name	Type	Initial
Output	bBusy	BOOL	
	bDone	BOOL	
	bError	BOOL	
	ipResultMessage	I_TcMessage	fbResult

- FB_TcIoXtsEnvironment.Init (METH)
- ProcessingUnit
  - FB_TcIoXtsEnvironment.GetXpuCount (METH)
  - FB_TcIoXtsEnvironment.GetXpuOids (METH)
  - FB_TcIoXtsEnvironment.P_XpuCount (PROP)
  - FB_TcIoXtsEnvironment.P_XpuOids (PROP)
- Subitems
  - FB_TcIoXtsEnvironment.XpuTcIo (METH)

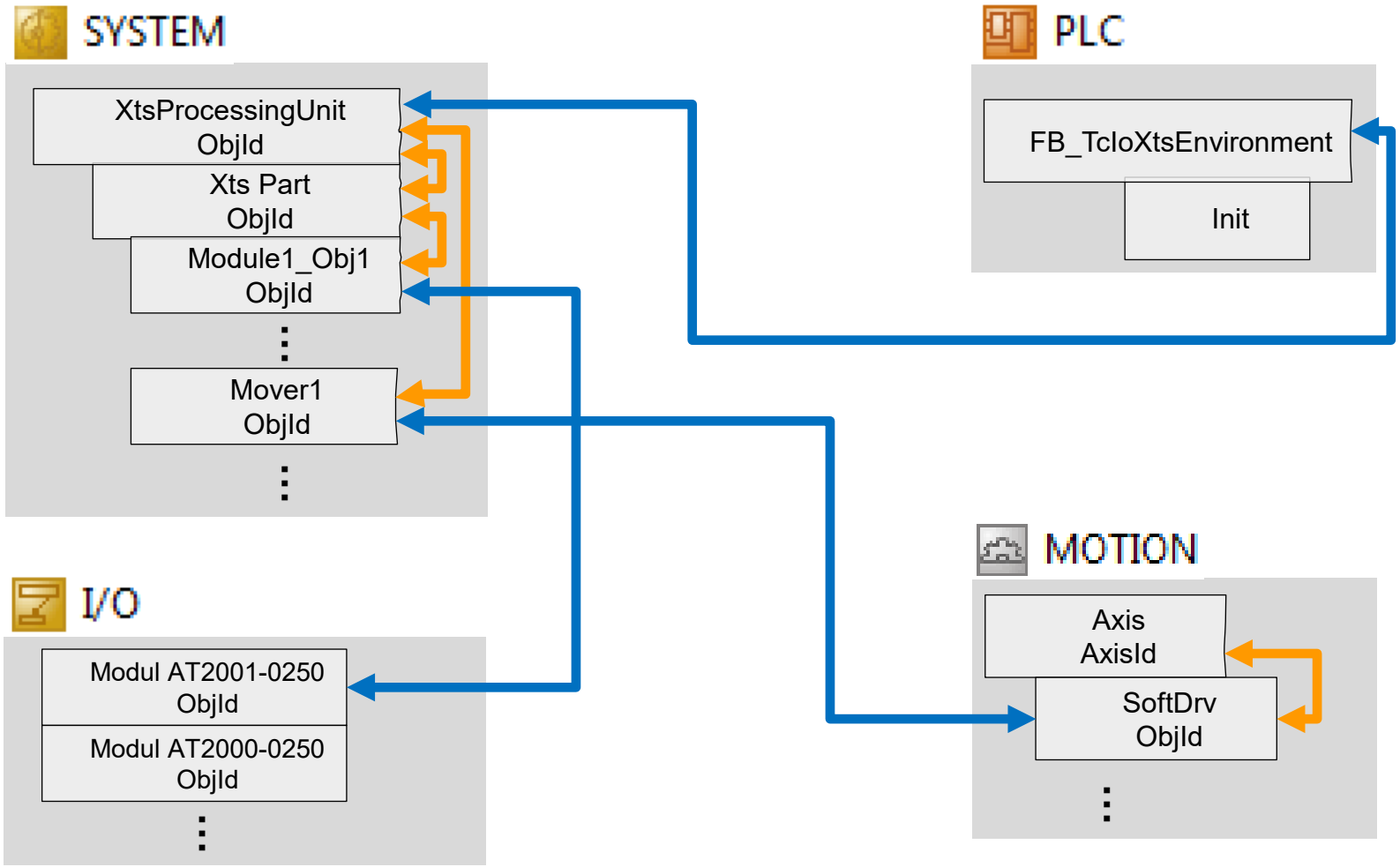


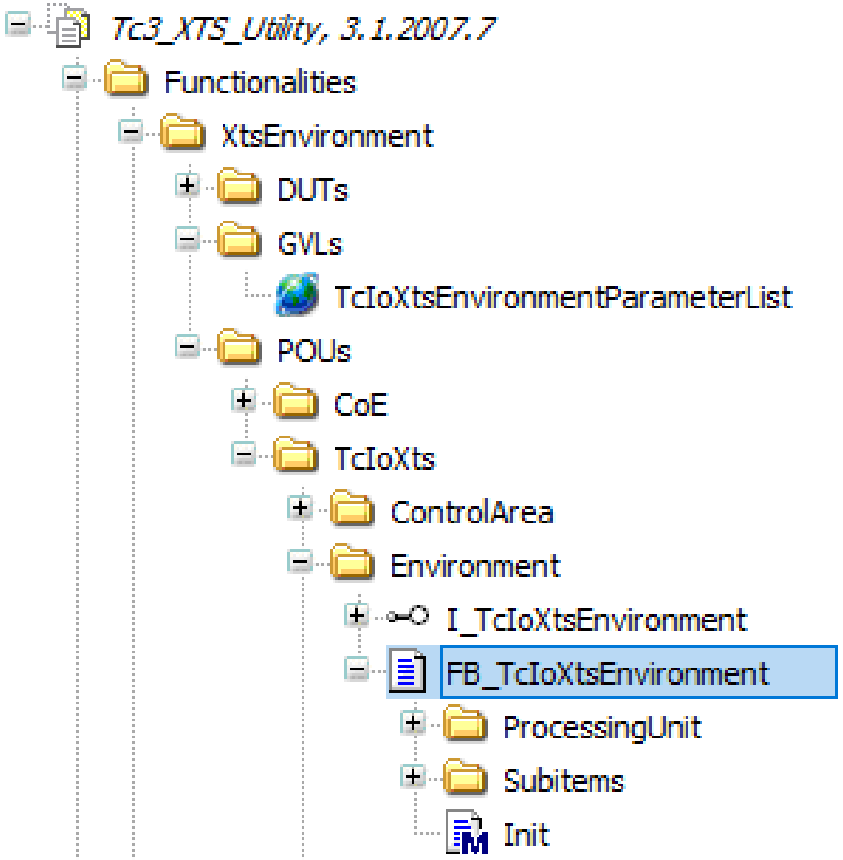
FB_TcIoXtsEnvironment

- only one FB_TcIoXtsEnvironment is needed the XTS project.
  From this FB, all the XtsProcessingUnits and other objects can be accessed
- Once initialized, the Function Block have access to all the connected objects of the XtsEnvironment structure and their methods.
- There is no automatic cyclic update for data!
- The user can use the methods to get the required information if necessary.



Communication via FB_TcIoXtsEnvironment





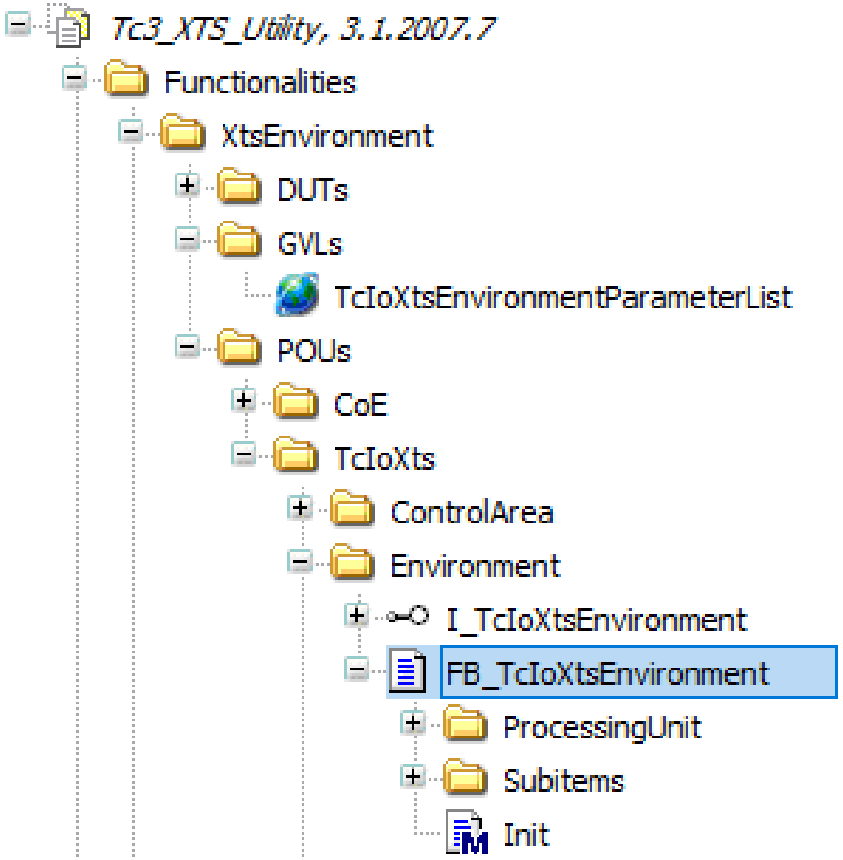
FB_TcIoXtsEnvironment

- declaration

```
// XTS Utility
VAR_GLOBAL
    fbXtsEnvironment          : FB_TcIoXtsEnvironment;

    {attribute 'init_on_onlchange'}
    bInit                     : BOOL := FALSE;
END_VAR
```

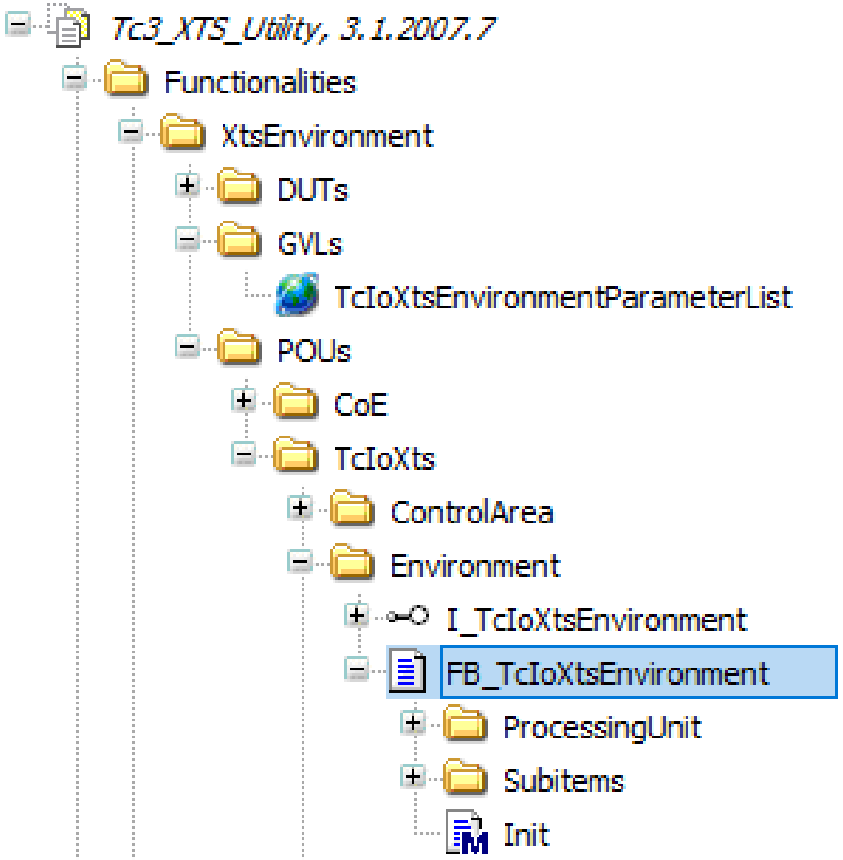
- cal in Main()

```
// Cal FB_XtsEnvironment
// Trigger only once in init
    IF NOT bInit THEN
       IF fbXtsEnvironment.Init(TRUE) THEN
             fbXtsEnvironment.Init(FALSE);
             bInit := TRUE;
       END_IF

       // If not yet initalized, dont‘t execute any of the XtsUtility code
       RETURN;
    END_IF
```



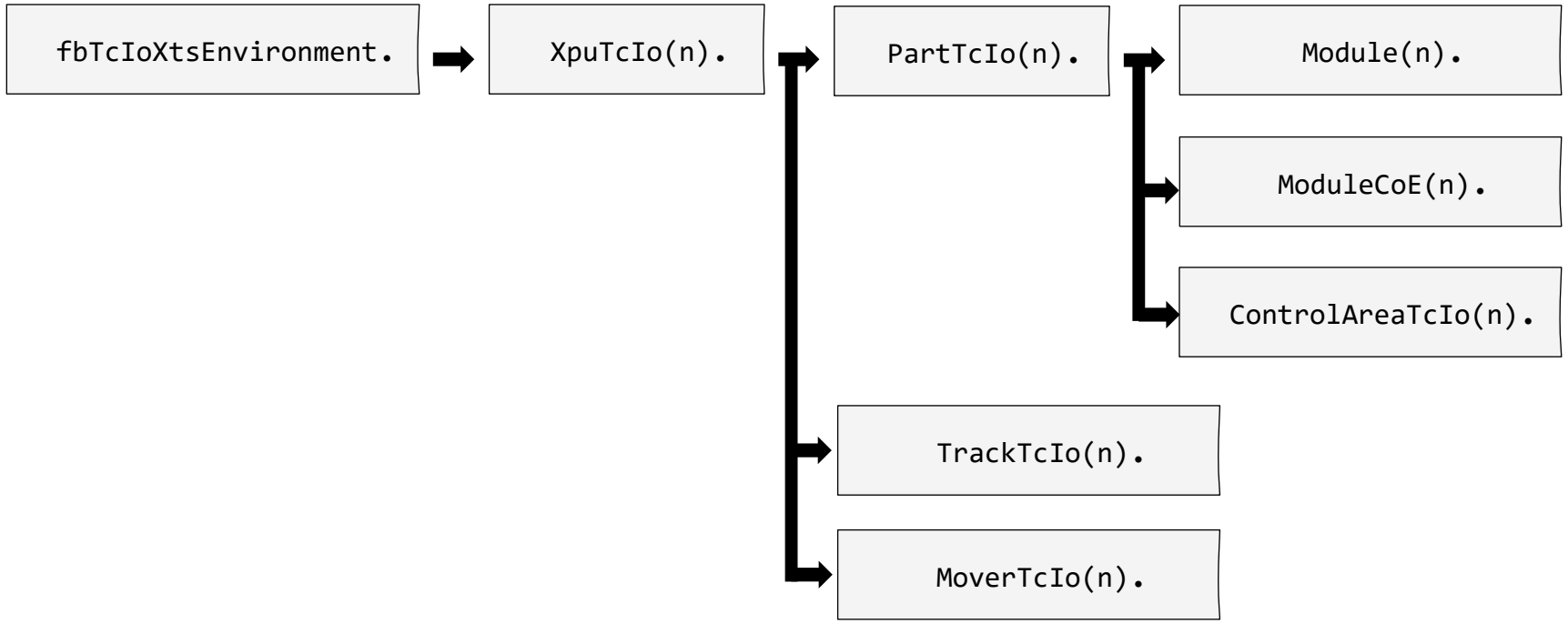
FB_TcIoXtsEnvironment

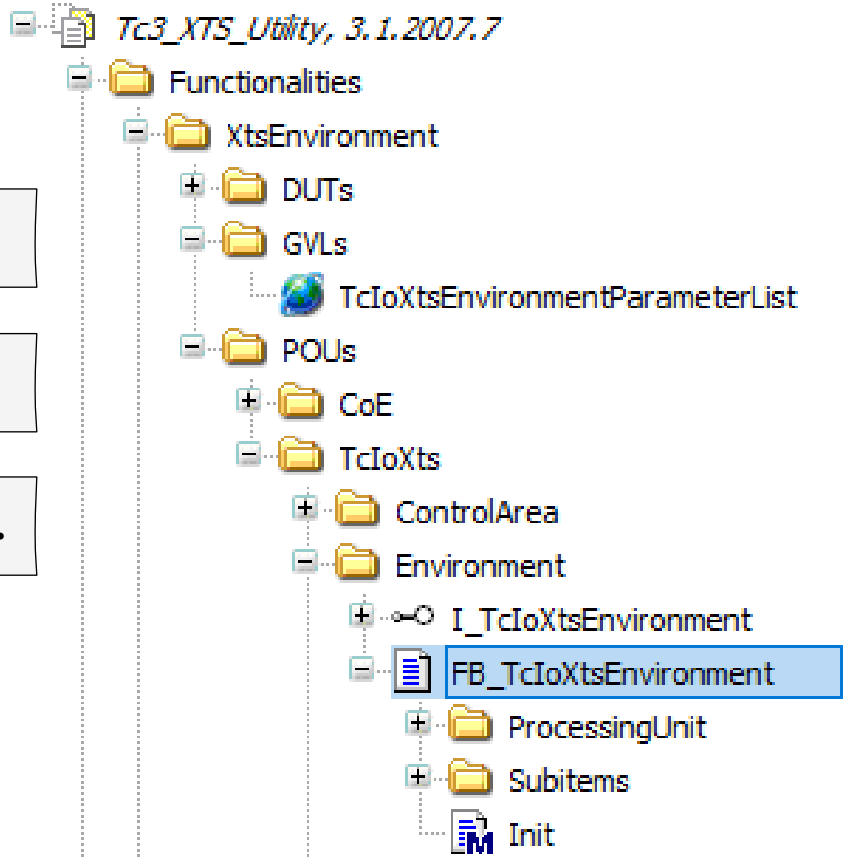
- The FB_TcIoXtsEnvironment is seen as a hub function block for all XTS related objects within the project.
- Each object has its respective function block (e.g.FB_TcIoXtsProcessingUnit) in the PLC. They can either be accessed via theFB_TcIoXtsEnvironment but also can be used standalone.



FB_TcIoXtsEnvironment

- To access the methods and gather the data easily, the below call chain can be followed which is very close to the actual object structure within the project:





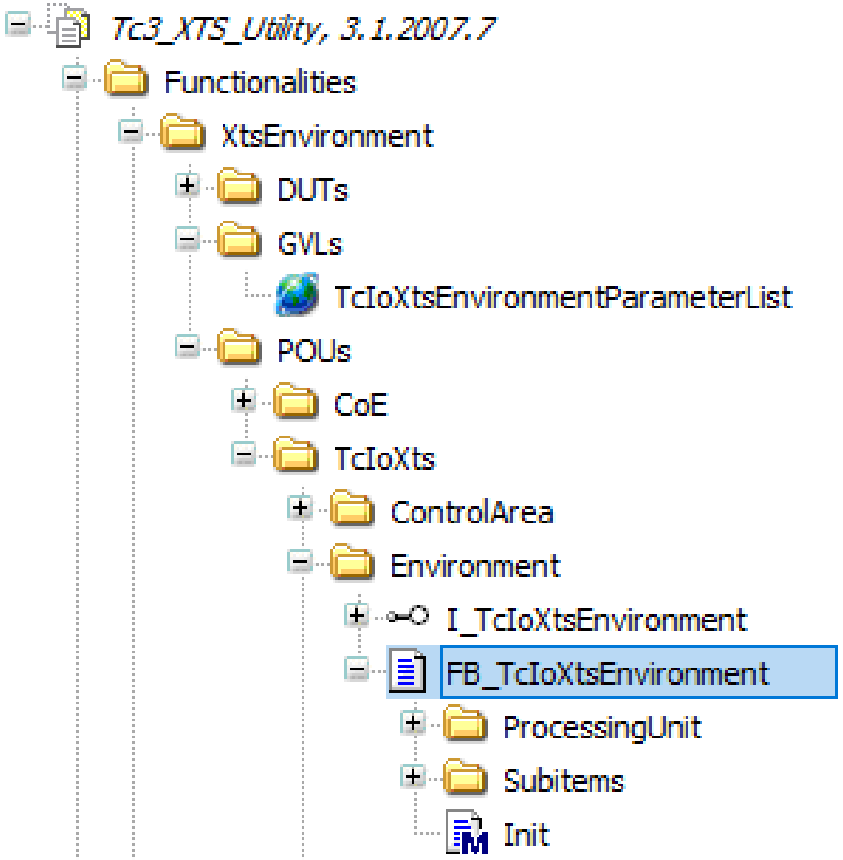
Read/Write Parameter via FB_TcIoXtsEnvironment

read parameter

```
// Position valid
    bAllPositionsValid :=
        fbXtsEnvironment.XpuTcIo(1).GetAreAllPositionsValid();

// Position valid
    bTeachingValid:=
        fbXtsEnvironment.XpuTcIo(1).GetIsTeachingValid();
```

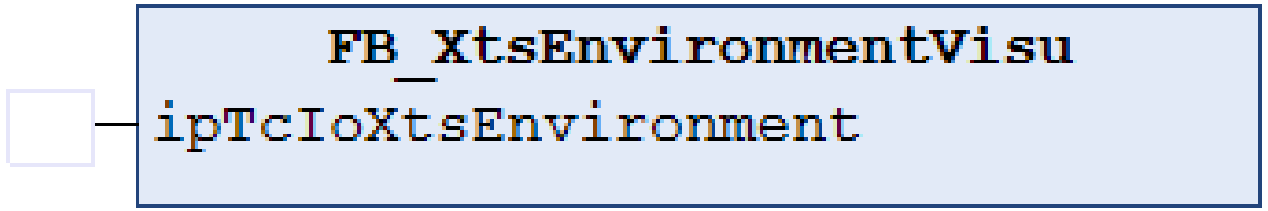
write parameter

```
// Set ModuleOffset
    fbXtsEnvironment.XpuTcIo(1).PartTcIo(1).ModuleTcIo(1).SetOffset(fNewModuleOffset);
```

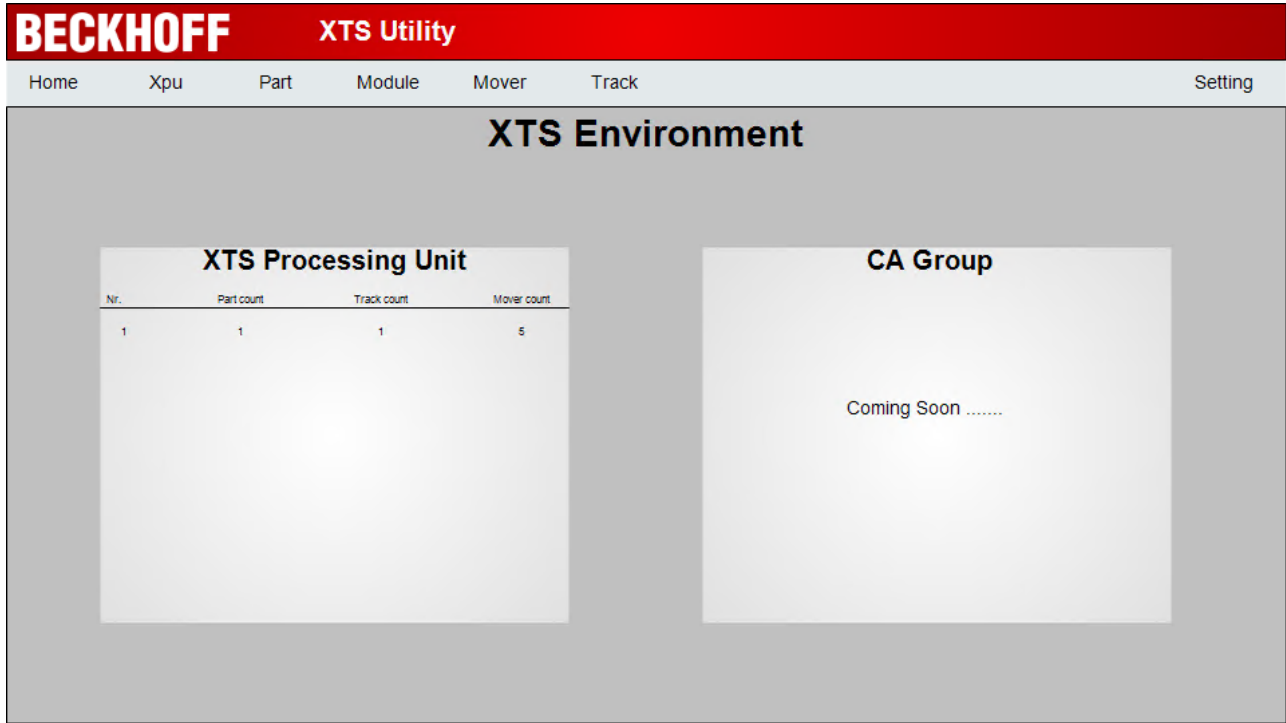


- **FB_XtsEnvironmentVisu & VI_XtsEnvironmentVisu**

  - FB_XtsEnvironmentVisu
    collect all necessary Information
    from the XTS-Track



  - VI_XtsEnvironmentVisu
    display all necessary Information
    from the XTS-system
    inside of a PLC-Visualization
    (as templet)



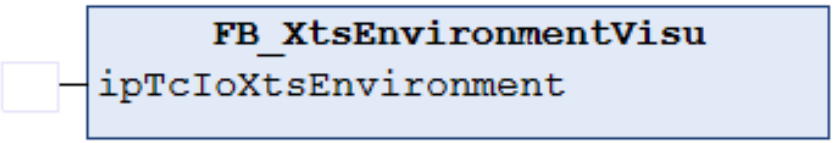
FB_VisuXtsDiag

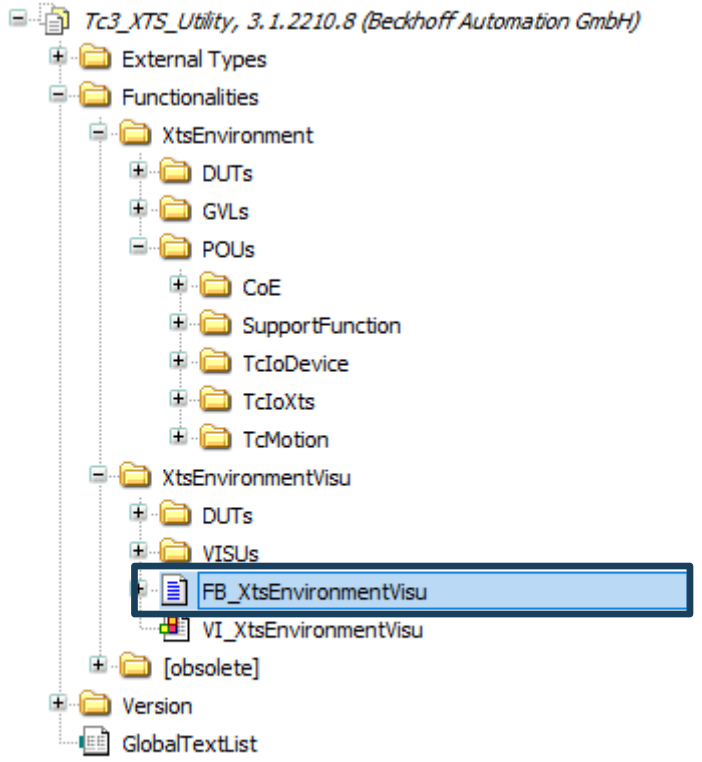
- declaration

```
// XTS Utility
VAR_GLOBAL

    // XTS Utility Visu
    fbXtsEnvironmentVisu                : FB_XtsEnvironmentVisu;

END_VAR
```

- cal in Main()

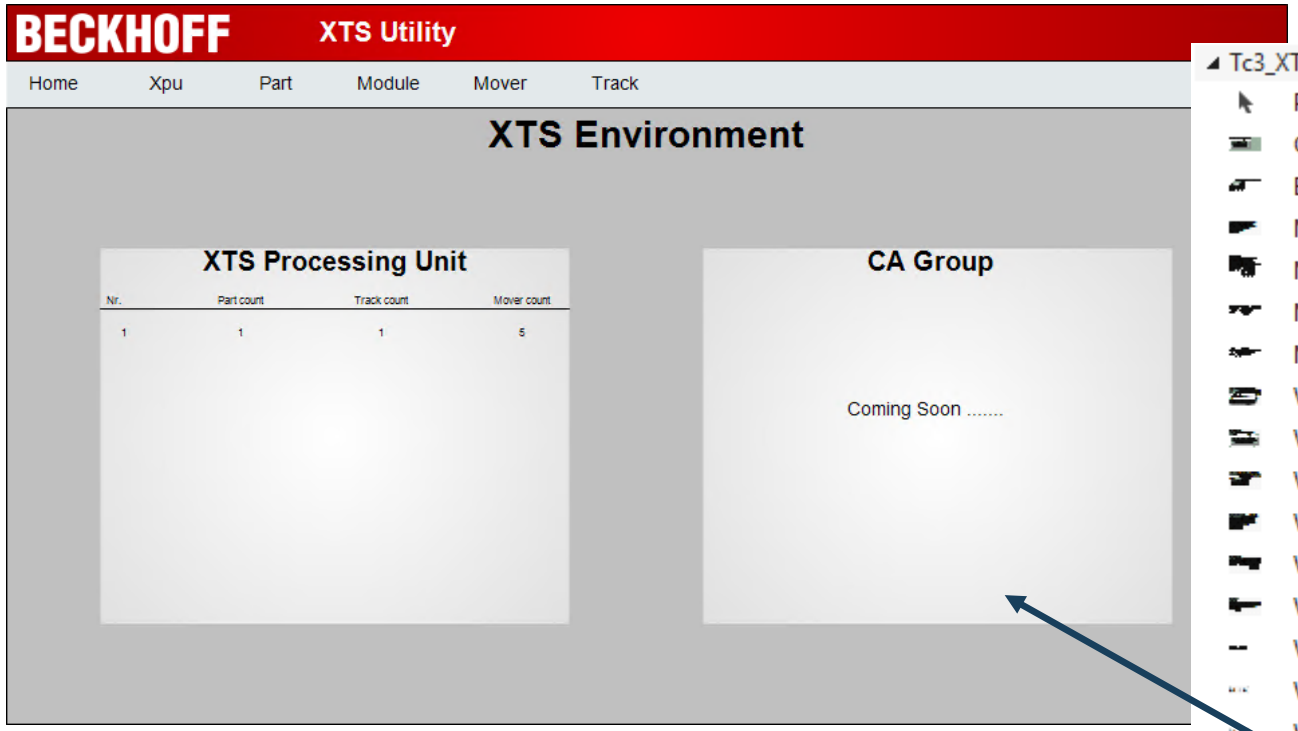
```
// Cal XTS Utility Visu
    // assignment of interface pointer
    fbXtsEnvironmentVisu.ipTcIoXtsEnvironment := fbXtsEnvironment;

    // cyclic update
    fbXtsEnvironmentVisu.Cycle();
```

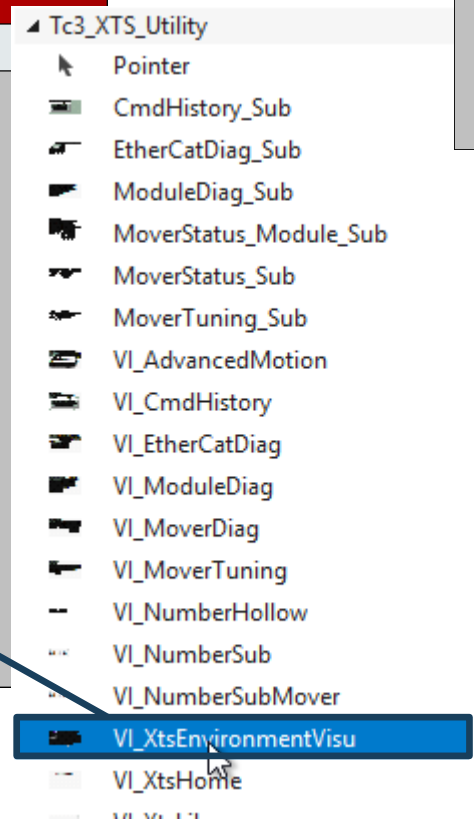


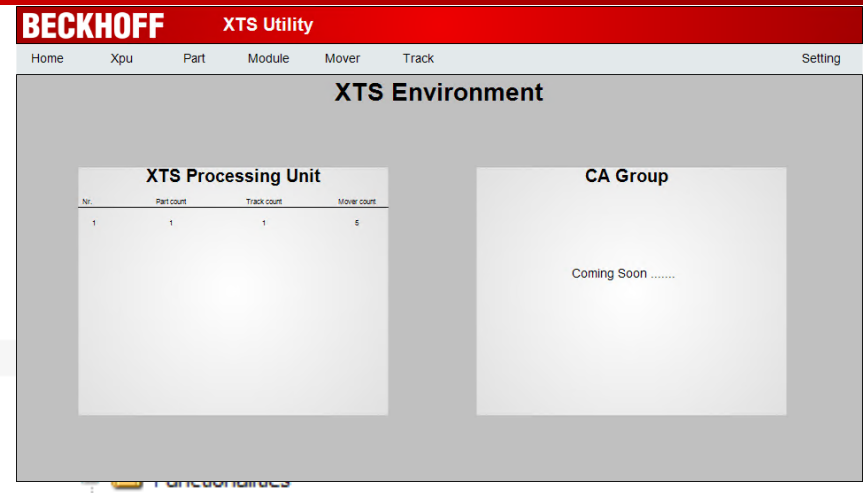


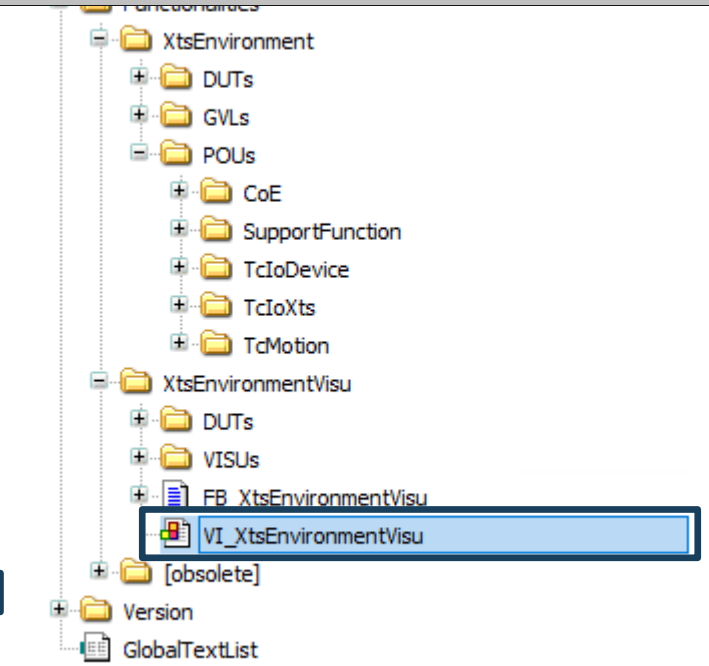
VI_XtsLib

- create in an PLC-Visualization









Agenda | XTS – PLC

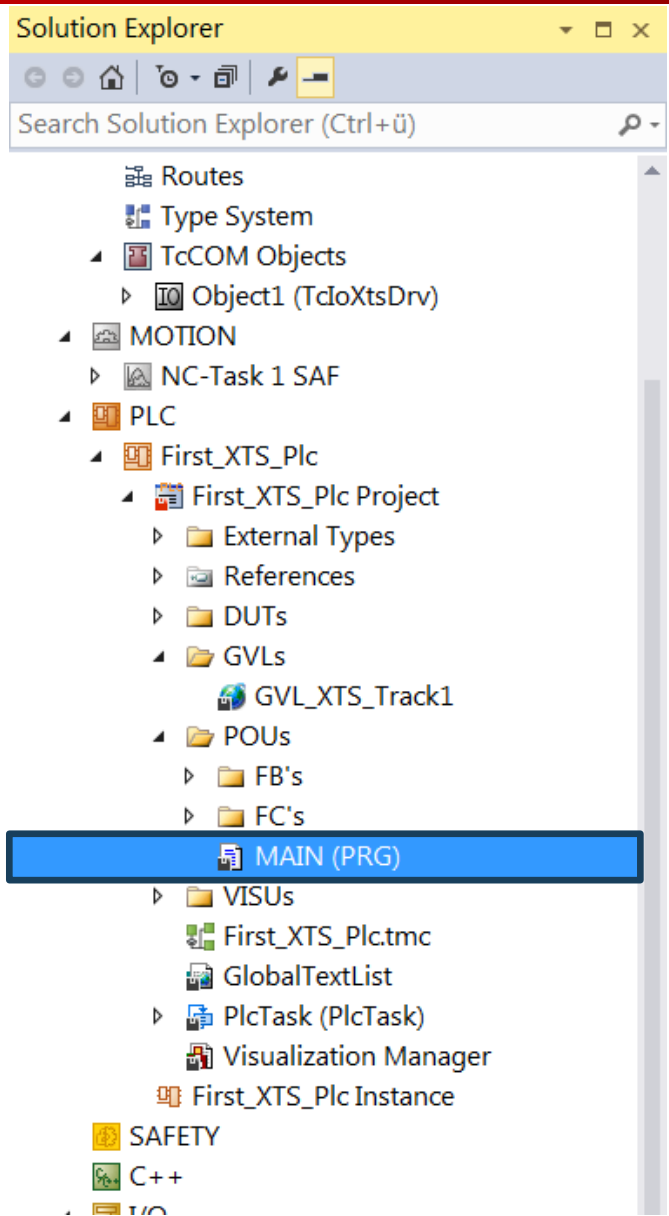
BECKHOFF

1. PLC-Library XTS
2. AXIS_REF
3. XTS-Utility Library
4. **PLC StartUp condition**
5. Mover1 detection via PLC
6. Overview TF5400 Collision Avoidance (CA)
7. CA-Group Object
8. PLC-Library CA-Group
9. AXES_GROUP_REF
10. CA-Group handling
11. CA-Operation

- **XTS- Start Up Condition**

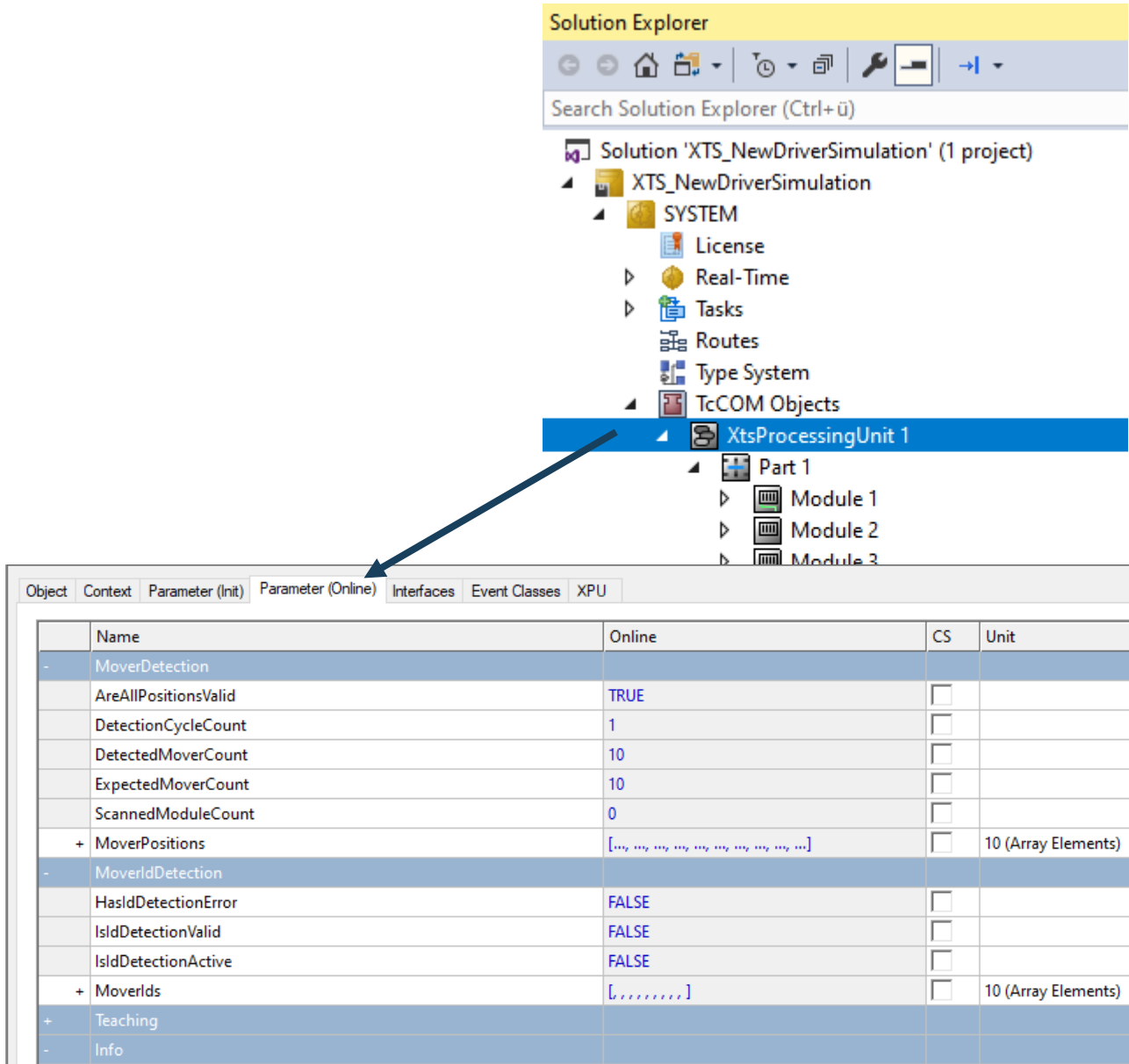
  - The XTS I/O Driver requires time to find all of the movers on the track. During this process the XTS Task will exceed its cycle time, preventing the PLC Task from running normally.
  - Therefore, the PLC program should not execute any logic until the XTS Task has successfully located all the configured movers

```
// Check if the position detection of the Movers has been completed.
    IF NOT fbXtsEnvironment.XpuTcIo(1).GetAreAllPositionsValid()
        OR NOT fbXtsEnvironment.XpuTcIo(1).GetIsTeachingValid()
        OR NOT ( fbXtsEnvironment.XpuTcIo(1).GetDetectedMoverCount() =
                    fbXtsEnvironment.XpuTcIo(1).GetExpectedMoverCount() )   THEN
        // If not all Mover positions are valid, do not process any further.
        RETURN;
    END_IF
```



XTS- StartUp Delay

- The actual state of the XTS-System are shown in “Parameter (Online)” for e.g.:
  - AreAllPositionsValid
  - DetectionCycleCount
  - DetectedMoverCount
  - ExpectedMoverCount
  - ScannedModuleCount



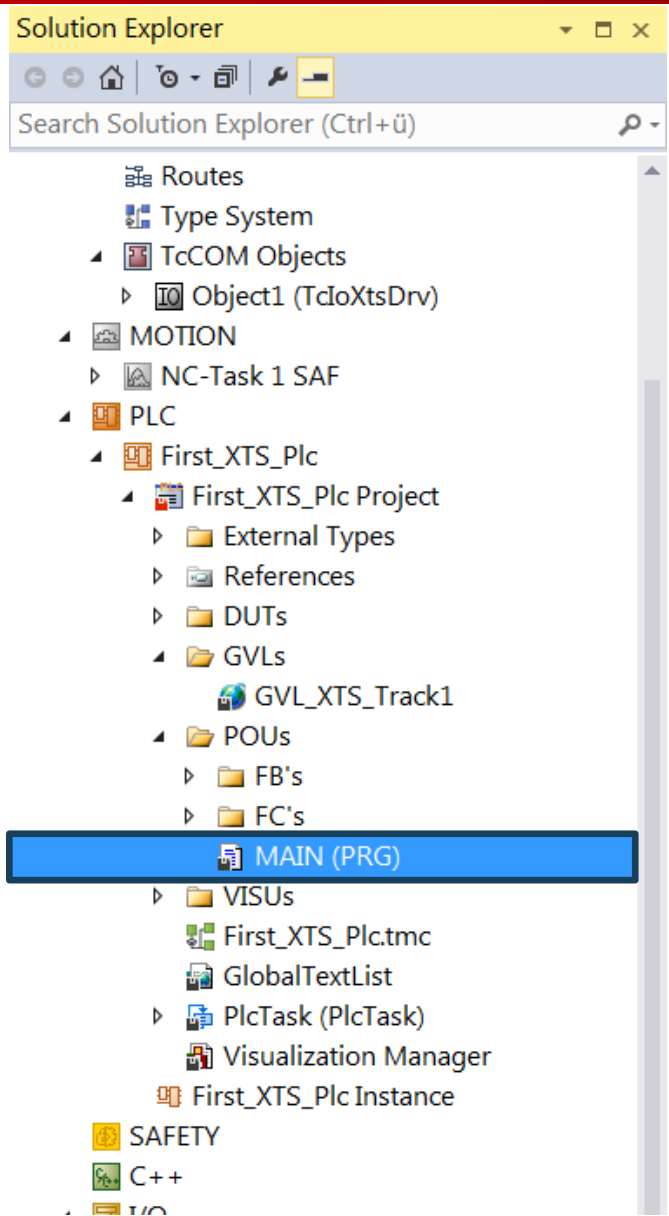
Agenda | XTS – PLC

BECKHOFF

1. PLC-Library XTS
2. AXIS_REF
3. XTS-Utility Library
4. PLC StartUp condition
5. **Mover1 detection via PLC**
6. Overview TF5400 Collision Avoidance (CA)
7. CA-Group Object
8. PLC-Library CA-Group
9. AXES_GROUP_REF
10. CA-Group handling
11. CA-Operation

Mover1 detection via PLC

```
// Mover1 detection
// Check if "MoverIdDetectionMode" has been activated under the XtsProcessingUnit object.
IF fbXtsEnvironment.XpuTcIo(1).GetMoverIdDetectionMode() = MoverIdDetectionMode.Mover1 THEN

    // Check if the "MoverIdDetection" has already been started or the Mover1 was found.
    IF bStartMoverIdDetection AND NOT (bMoverIdDetectionActive OR bMoverIdDetected) THEN
        bStartMoverIdDetection    := FALSE;
        // Trigger the "MoverIdDetection".
        fbXtsEnvironment.XpuTcIo(1).TriggerMoverIdDetection();
    END_IF

    // Check for error or success of "MoverIdDetection".
    // If the detection has an error, check for errors in output and try again.
    IF fbXtsEnvironment.XpuTcIo(1).GetHasMoverIdDetectionError() THEN
        RETURN;
    END_IF

    // Do not process any further until the "MoverId" was deteced.
    IF NOT fbXtsEnvironment.XpuTcIo(1).GetIsMoverIdDetectionValid() THEN
        RETURN;
    END_IF

// If the "MoverIdDetection" was not activated in the XtsProcessingUnit object,
// then the activation of the "MoverIdDetection" and the search of Mover1 is skipped.
ELSE
    ;
END_IF
```



Agenda | XTS – PLC

BECKHOFF

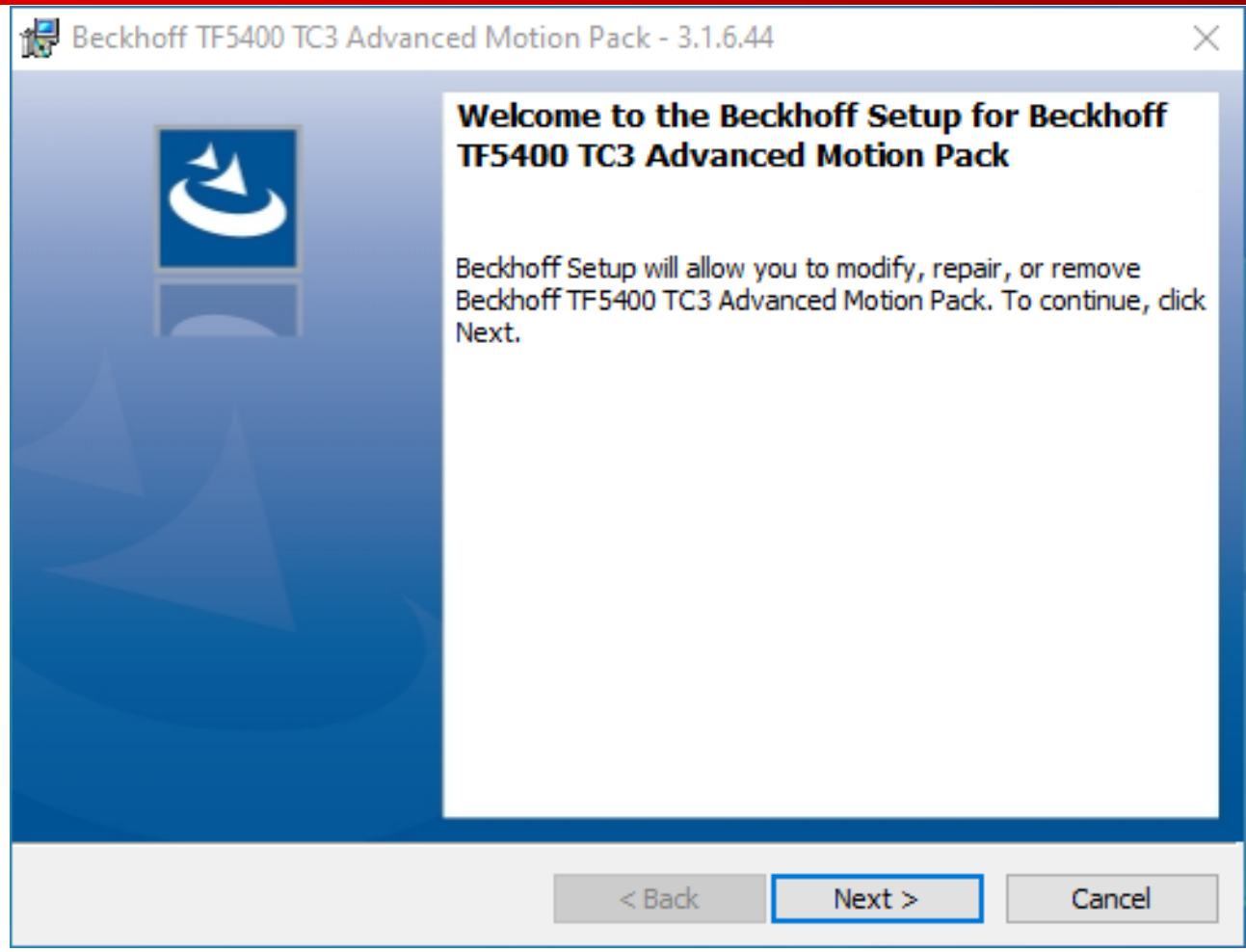
1. PLC-Library XTS
2. AXIS_REF
3. XTS-Utility Library
4. PLC StartUp condition
5. Mover1 detection via PLC
6. **Overview TF5400 Collision Avoidance (CA)**
7. CA-Group Object
8. PLC-Library CA-Group
9. AXES_GROUP_REF
10. CA-Group handling
11. CA-Operation

XTS – Collision Avoidance

BECKHOFF

- TC3 Motion Collision Avoidance as TcCOM Module (TwinCAT Component Object Models

  - TF5400 | TC3 Advanced Motion Pack
    - Included
      - TC3 Kinematic Transformation
      - TC3 Motion Collision Avoidance
      - TC3 Motion Pick-and-Place
      - TC3 Planar Motion



Collision Avoidance Overview

- The purpose of collision avoidance is to prevent collisions between movers, and also to maintain a parameterized “Gap” between any two movers.
- Therefor PTP axes are added to a CA (collision avoidance) group.
- Motion function blocks for collision avoidance are included in library Tc3_McCollisionAvoidance.
- Administrative function blocks are included in library Tc3_McCoordinatedMotion.
- For detailed collision avoidance information see the online Beckhoff Information System section on Advanced Motion

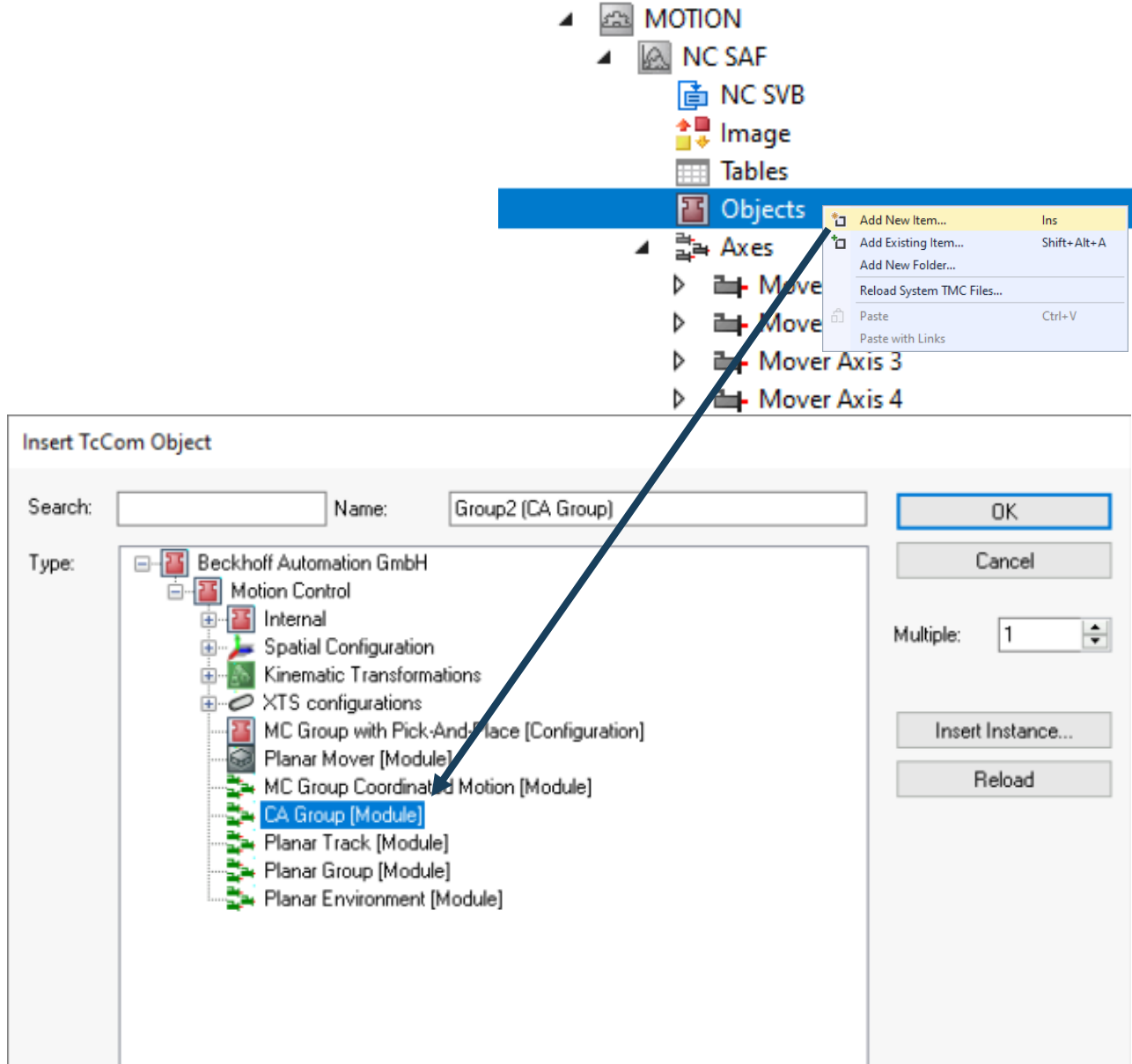
Link to Infosys

Agenda | XTS – PLC

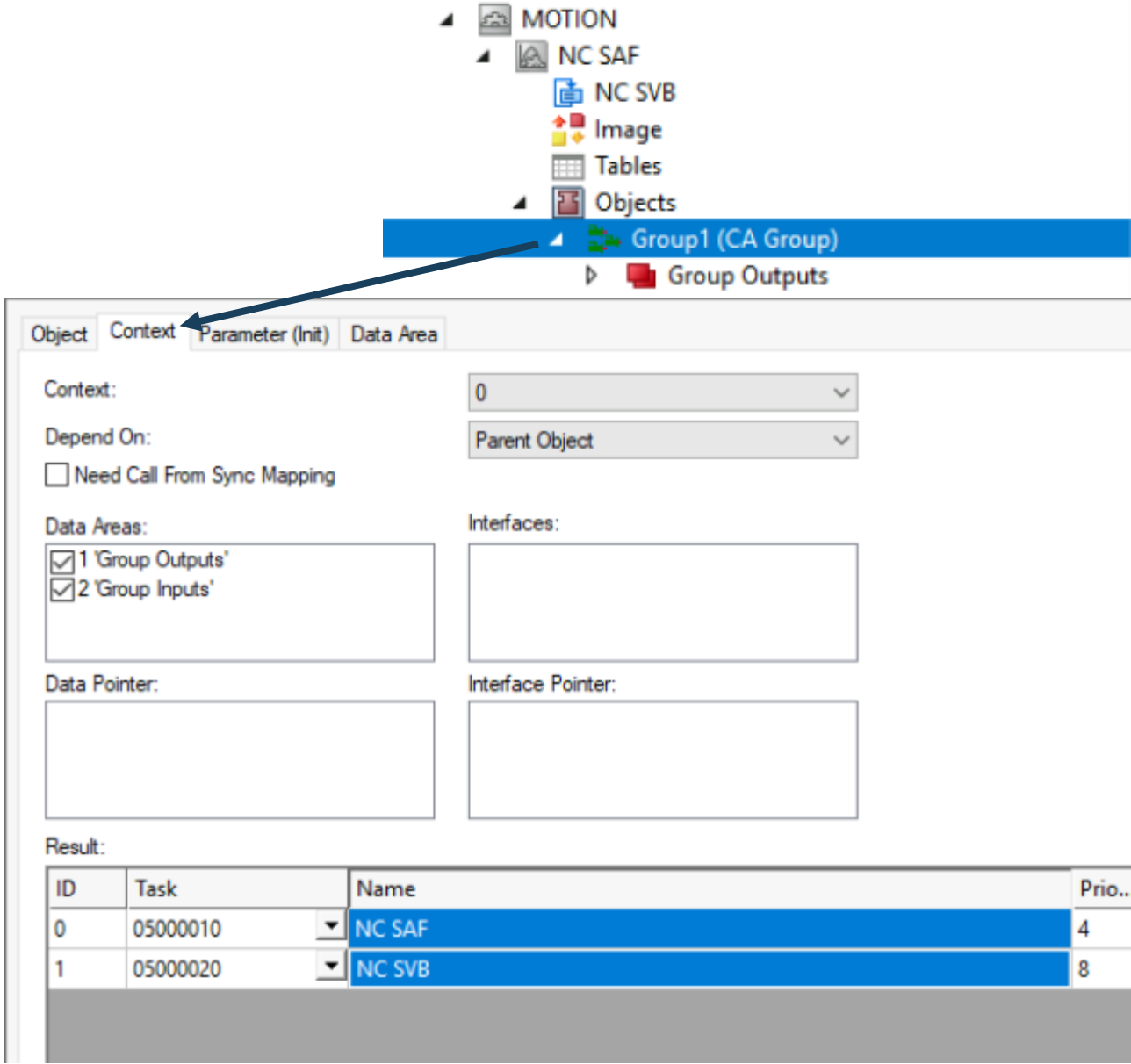
BECKHOFF

1. PLC-Library XTS
2. AXIS_REF
3. XTS-Utility Library
4. PLC StartUp condition
5. Mover1 detection via PLC
6. Overview TF5400 Collision Avoidance (CA)
7. **CA-Group Object**
8. PLC-Library CA-Group
9. AXES_GROUP_REF
10. CA-Group handling
11. CA-Operation

- Adding the CA Group Object



- Set the Context of the Collision Avoidance group to the NC-SAF Task

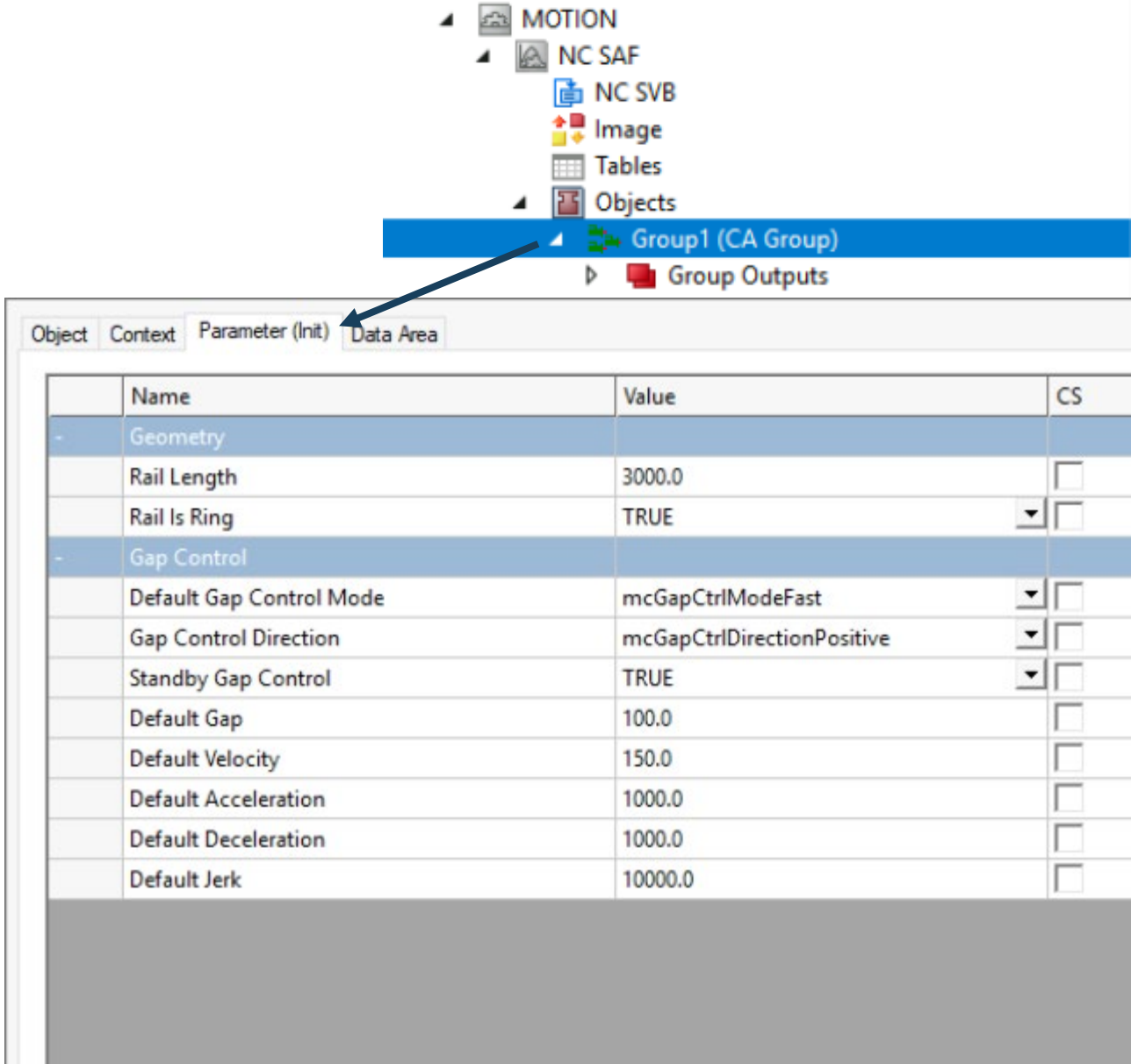


Collision Avoidance Group Settings

The CA Group has 4 key settings.

- Rail Length
  (Same as the HardwareModulo)
- Rail is a ring

- Gap Control Mode
  - mcGapCtrlModeStandard
  - mcGapCtrlModeFast
- Standby Gap Control

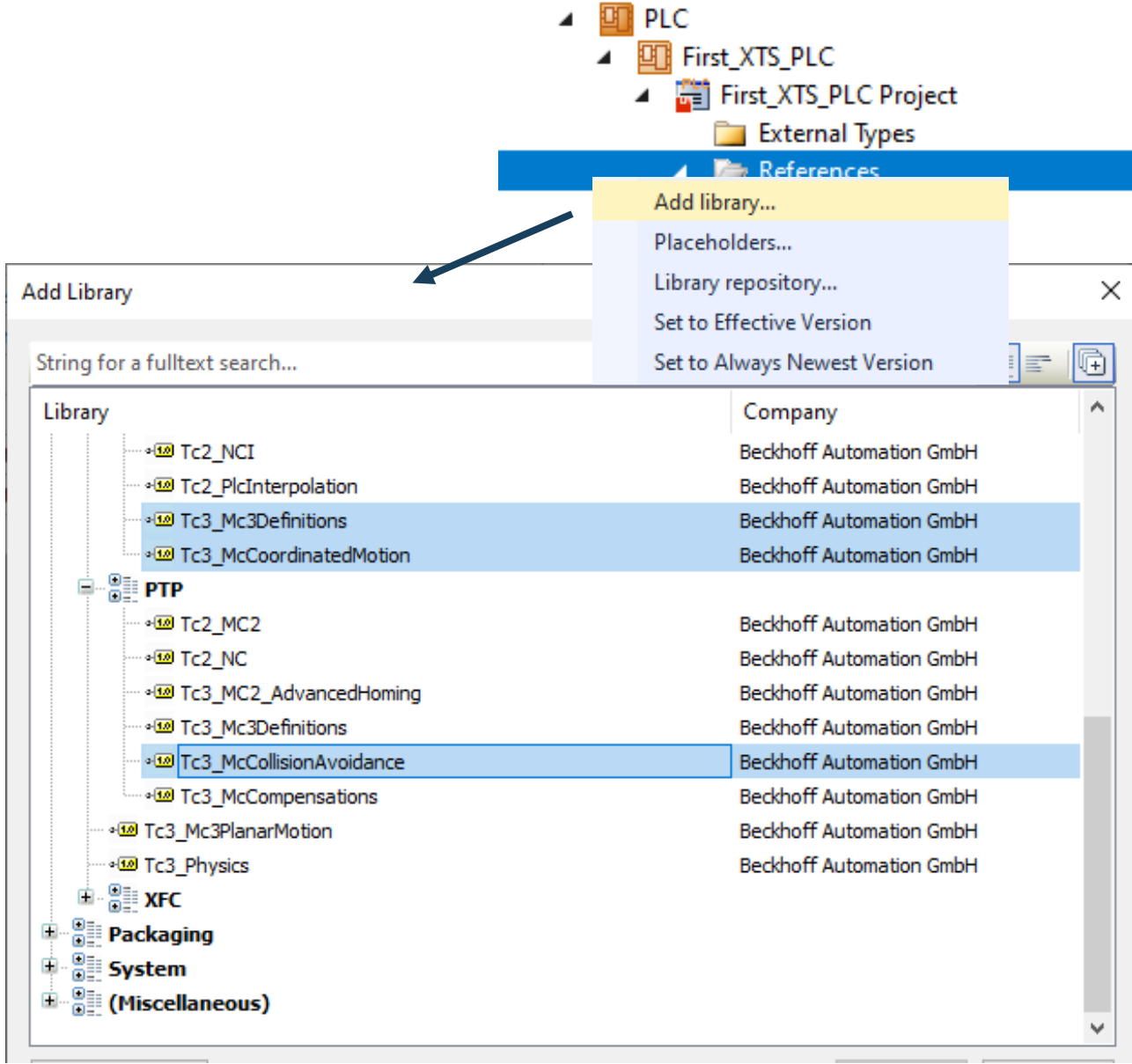


Agenda | XTS – PLC

BECKHOFF

1. PLC-Library XTS
2. AXIS_REF
3. XTS-Utility Library
4. PLC StartUp condition
5. Mover1 detection via PLC
6. Overview TF5400 Collision Avoidance (CA)
7. CA-Group Object
8. **PLC-Library CA-Group**
9. AXES_GROUP_REF
10. CA-Group handling
11. CA-Operation

Necessary PLC-Libraries
Collision Avoidance



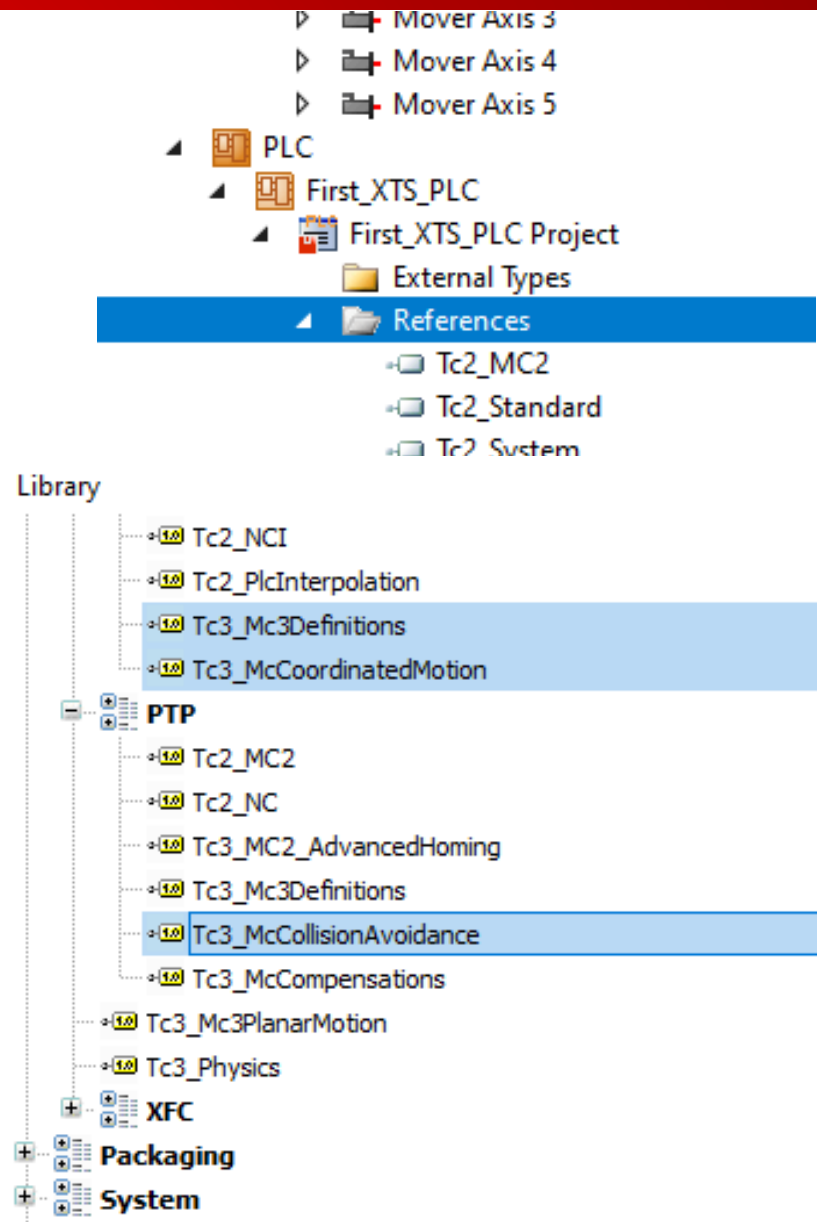
Necessary PLC-Libraries
Collision Avoidance

- Tc3_McCollisionAvoidance

  Library containing control function blocks for Collision Avoidance
- Tc3_McCoordinatedMotion

  Library containing control function blocks for Axis Group handling
- Tc3_Mc3Definitions

  Library with definition for the motion functions



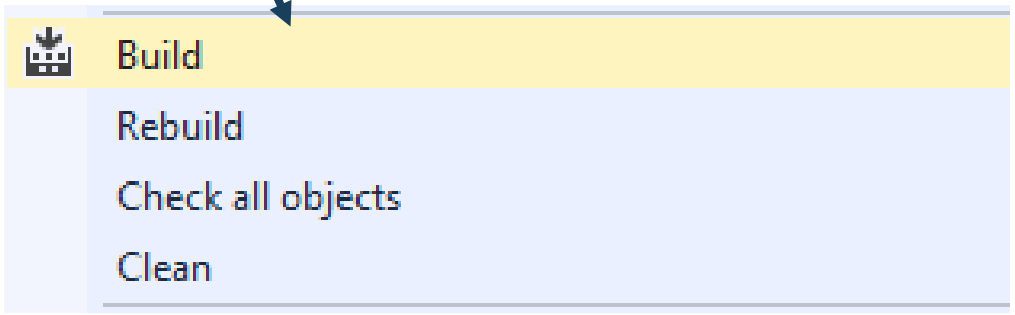
Agenda | XTS – PLC

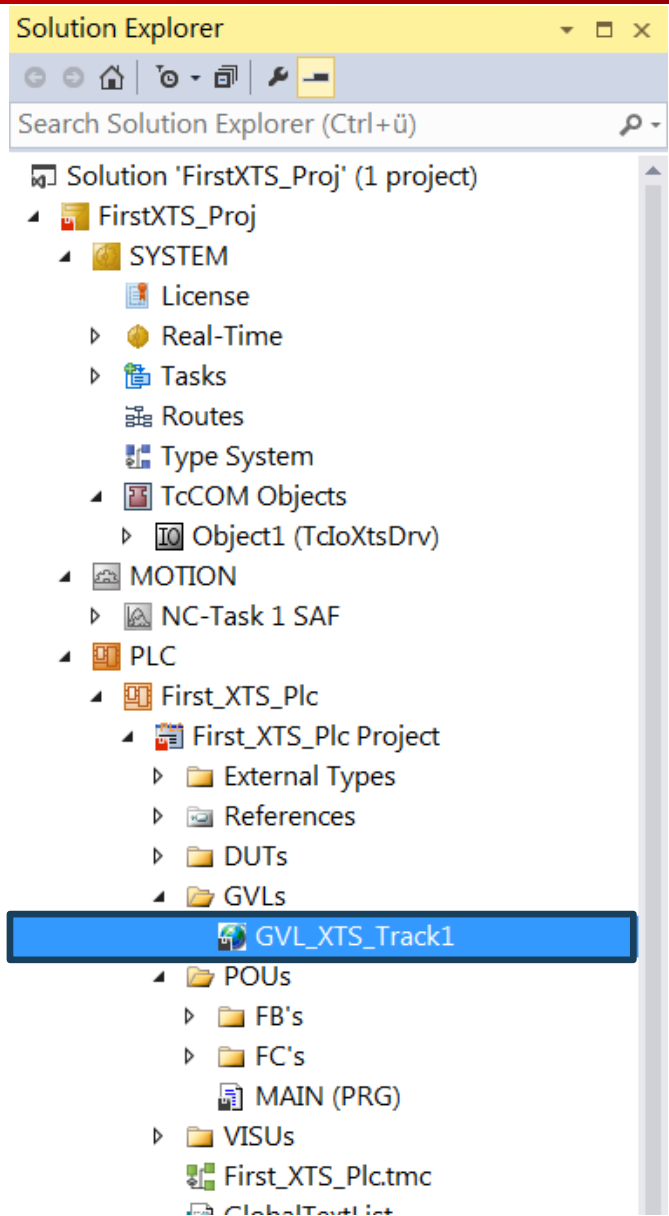
BECKHOFF

1. PLC-Library XTS
2. AXIS_REF
3. XTS-Utility Library
4. PLC StartUp condition
5. Mover1 detection via PLC
6. Overview TF5400 Collision Avoidance (CA)
7. CA-Group Object
8. PLC-Library CA-Group
9. **AXES_GROUP_REF**
10. CA-Group handling
11. CA-Operation

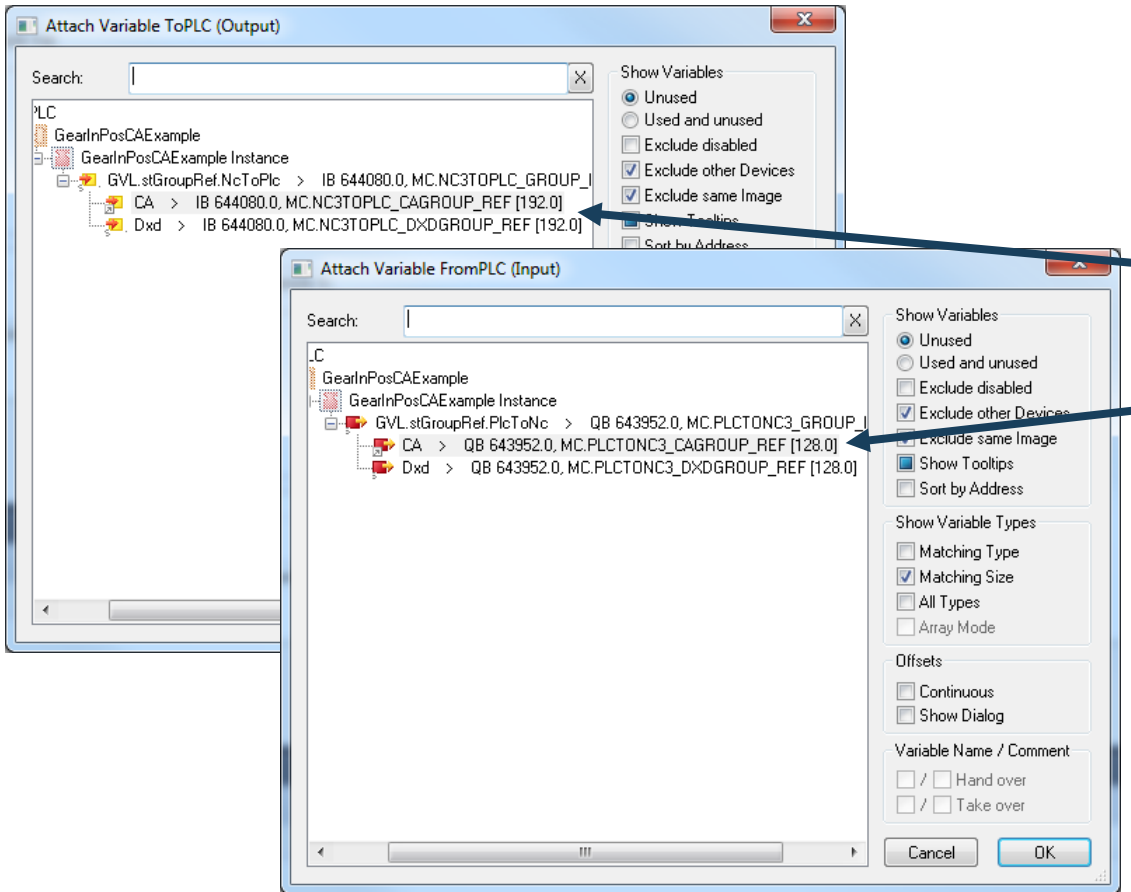
- **AXES_GROUP_REF for linking to the CA Group Object**
  - definition of AXES_GROUP_REF
    - AXES_GROUP_REF interface provides the cyclical data exchange between PLC and a NC group object.

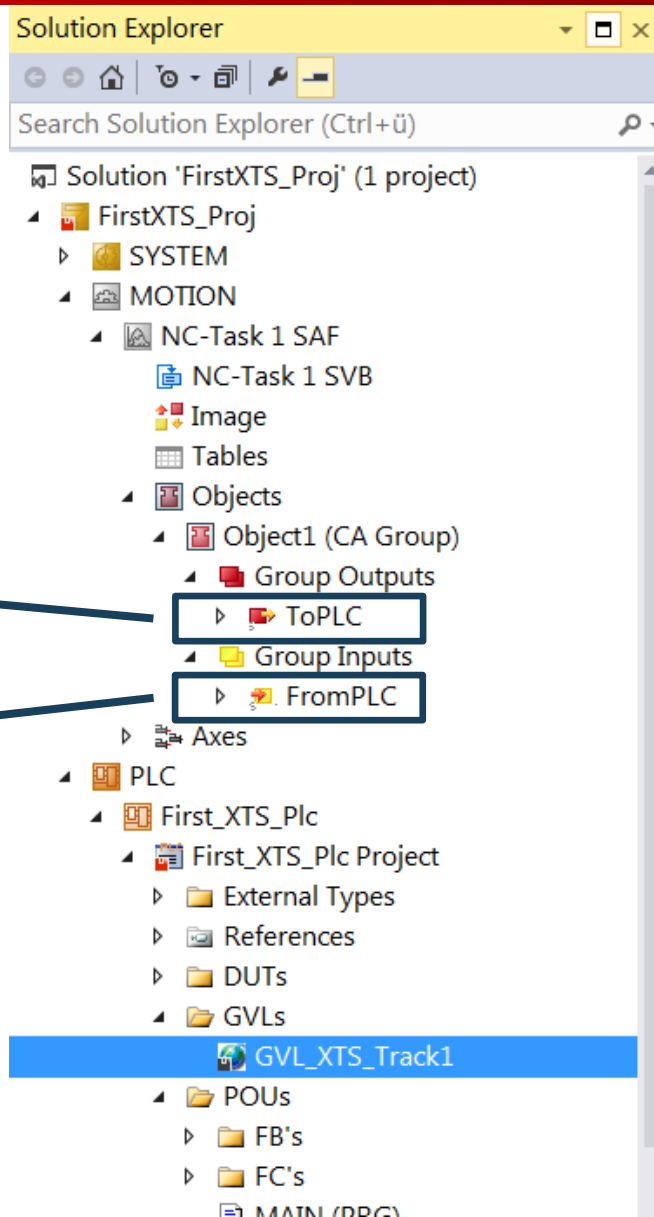
```
VAR_GLOBAL
    // IO-Interface to the CA-Group_object
    stCaGroupRef            : AXES_GROUP_REF;
END_VAR
```





- **AXES_GROUP_REF for linking then PLC with the CA Group Object**
  - link the AXES_GROUP_REF to the CA-Group



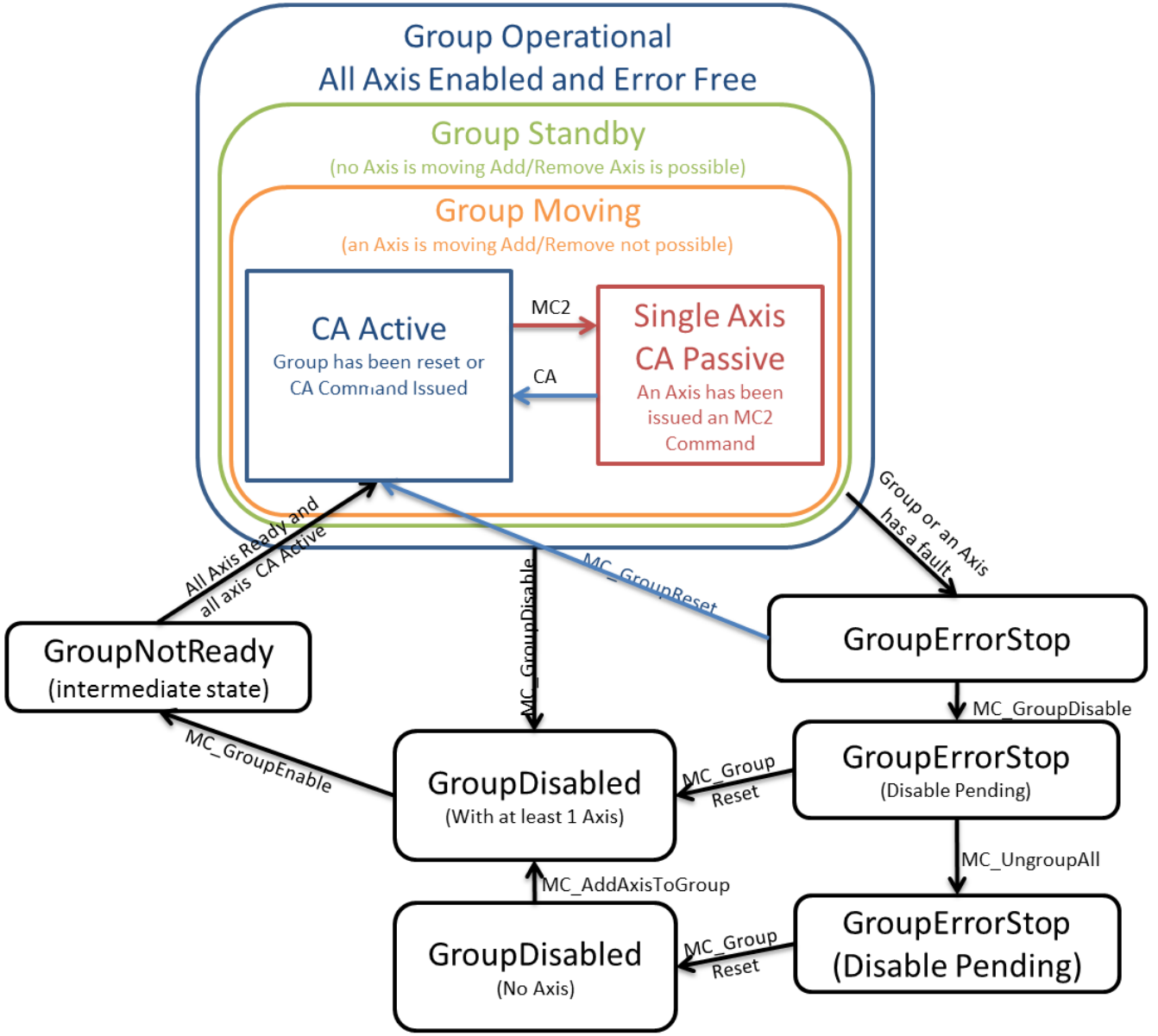


Agenda | XTS – PLC

BECKHOFF

1. PLC-Library XTS
2. AXIS_REF
3. XTS-Utility Library
4. PLC StartUp condition
5. Mover1 detection via PLC
6. Overview TF5400 Collision Avoidance (CA)
7. CA-Group Object
8. PLC-Library CA-Group
9. AXES_GROUP_REF
10. **CA-Group handling**
11. CA-Operation

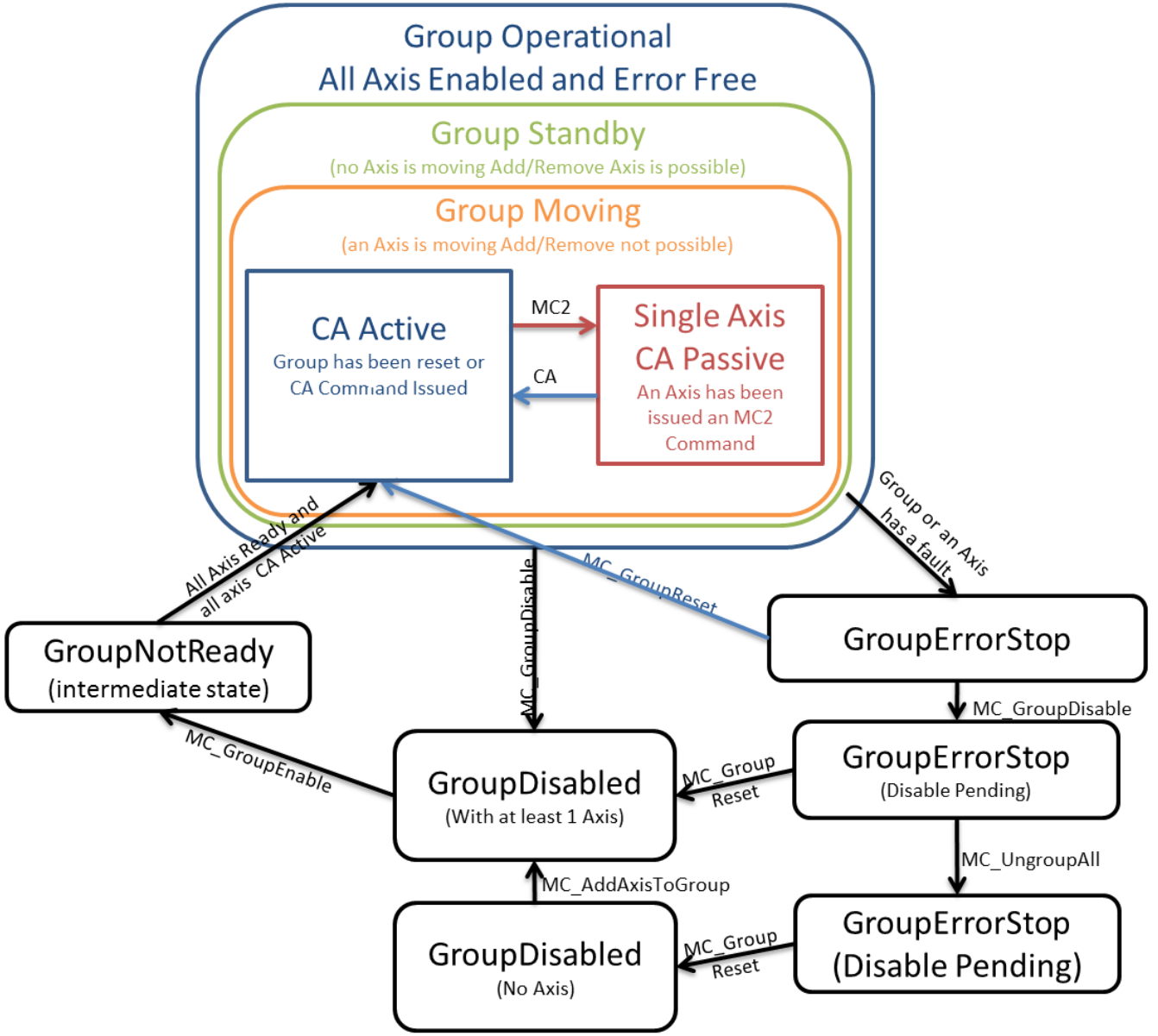
Collision avoidance State Model



Collision avoidance State Model

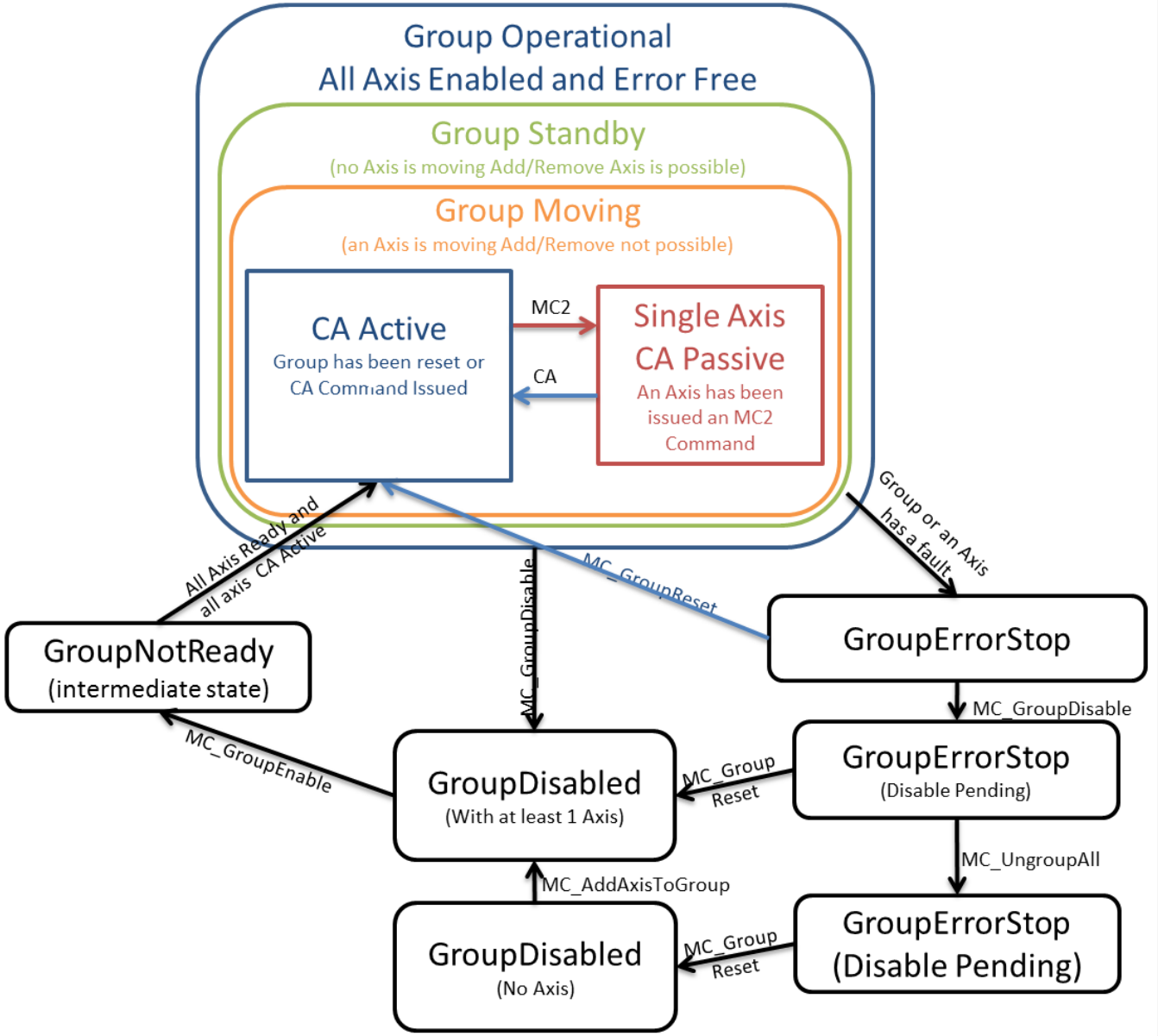
There are effectively 3 Group states that effect a Collision Avoidance axis.

1. Operational (Standby or Moving)
2. GroupErrorState (Faulted and Enabled, Faulted with Disable Pending, Faulted with Ungroup all pending)
3. Group Disabled (with at least 1 Axis or without any axis)



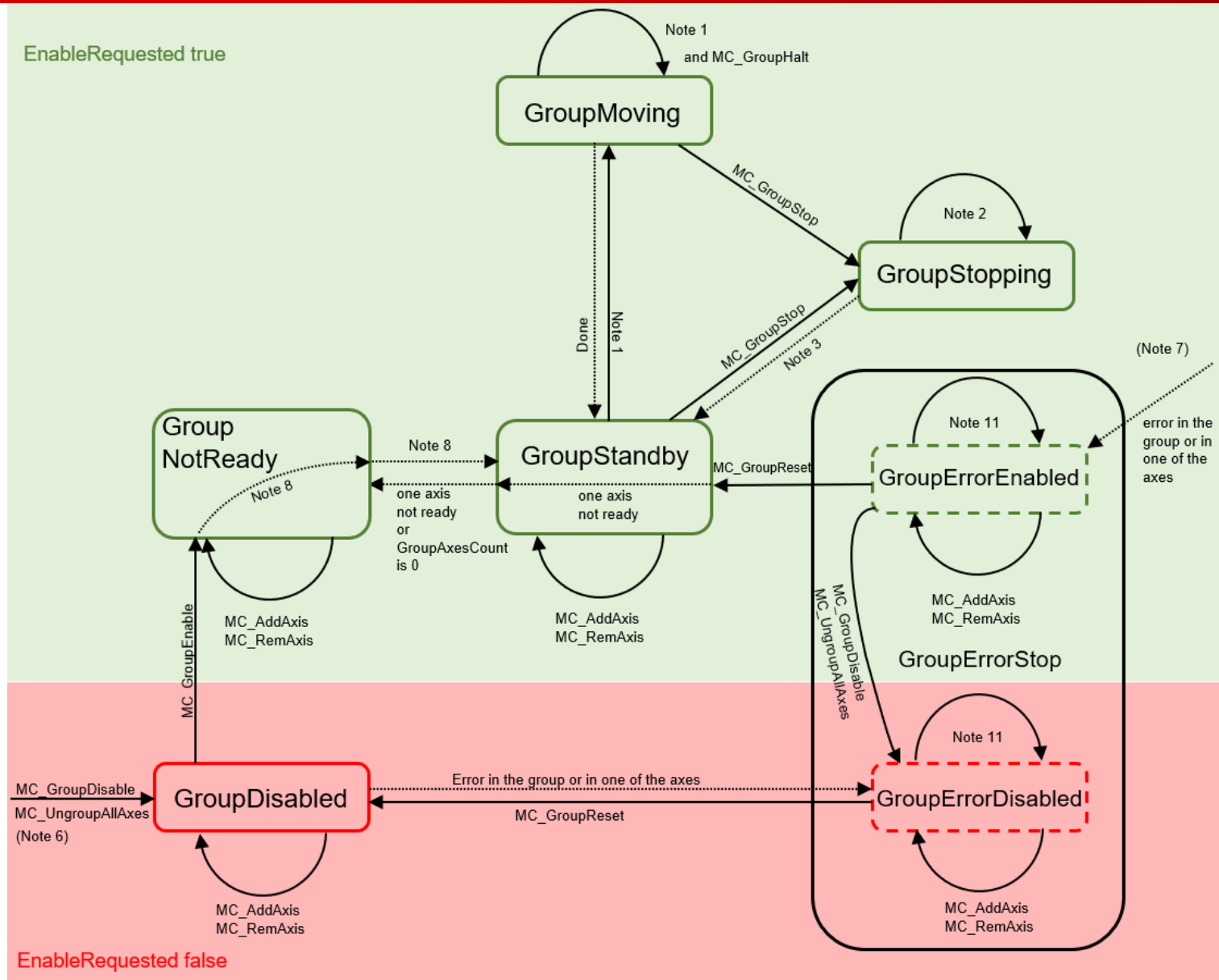
Collision Avoidance State Model

- A group without any axes is always disabled - it cannot be enabled.
- GroupNotReady is a short transitional state or waiting for axis to be enabled.
- GroupErrorState is a special state. The only method to leave this state is to issue a GroupReset.
- However, Group Reset will have different outcomes depending on the next requested or required state.

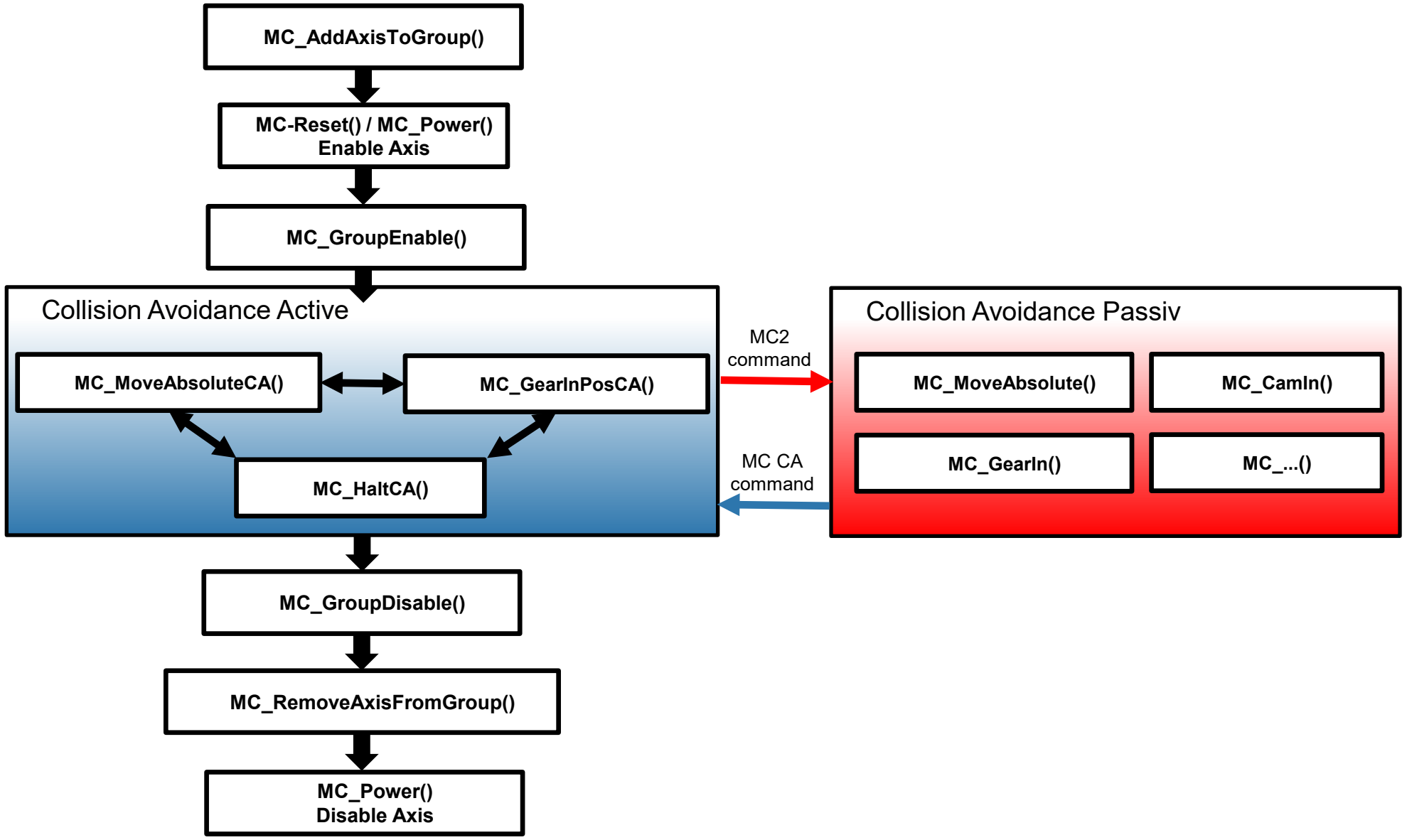


Collision Avoidance State Model

– simpler presentation
(State diagram valid for V3.1.10)



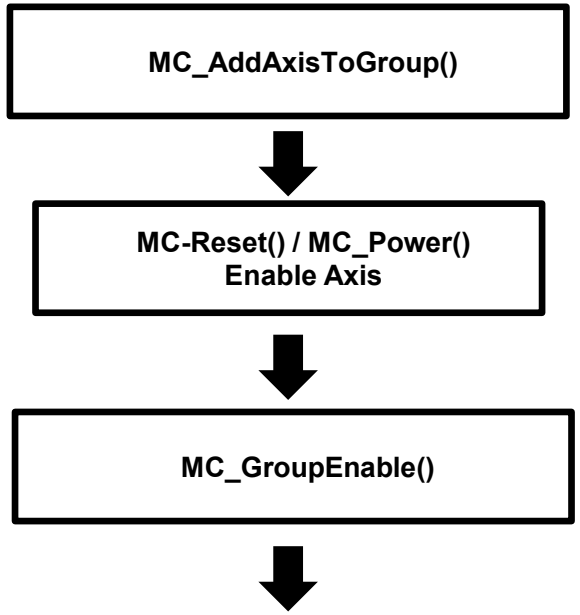
Full Sequence

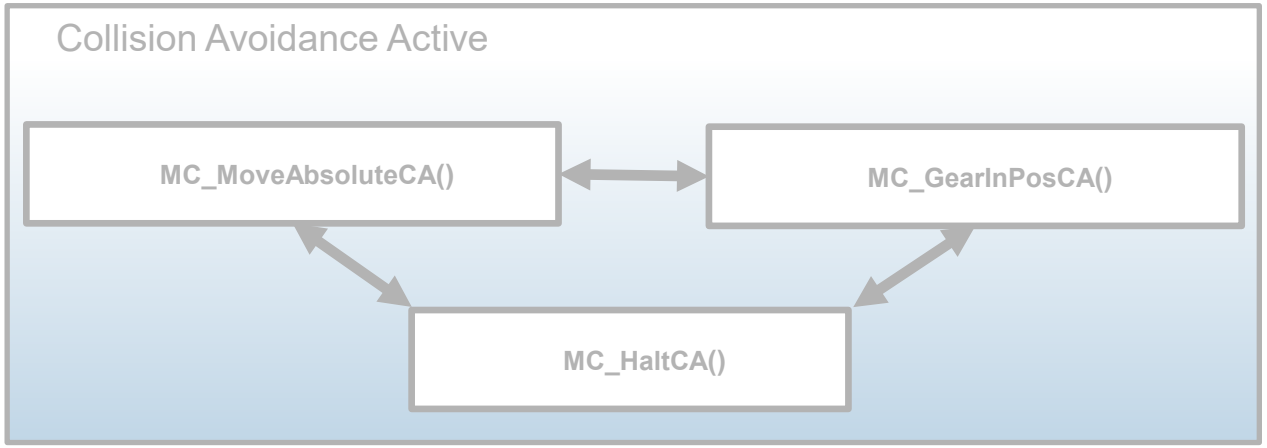


Full Sequence

Startup Sequence:

1. Add All Axes to the group
2. Reset/Enable All axes
   (ensure all axes are enabled
   and error free)
3. Enable the Group



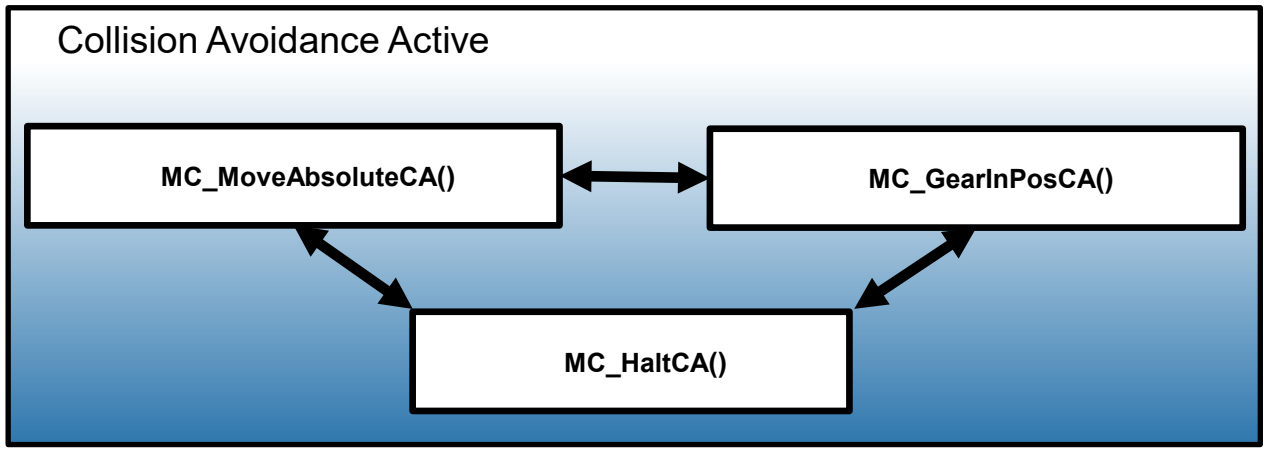


Full Sequence

Collision avoidance is active :

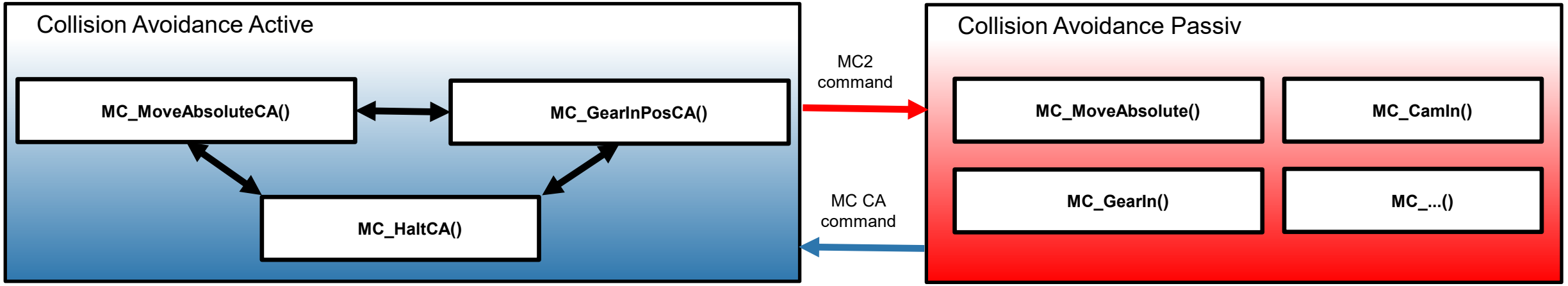
If Standby Gap is active the movers will separate

Should a failure occur, GroupReset will return the system to the running state



Full Sequence

– By using a MC2-Command the axis change in “CA-Passiv”



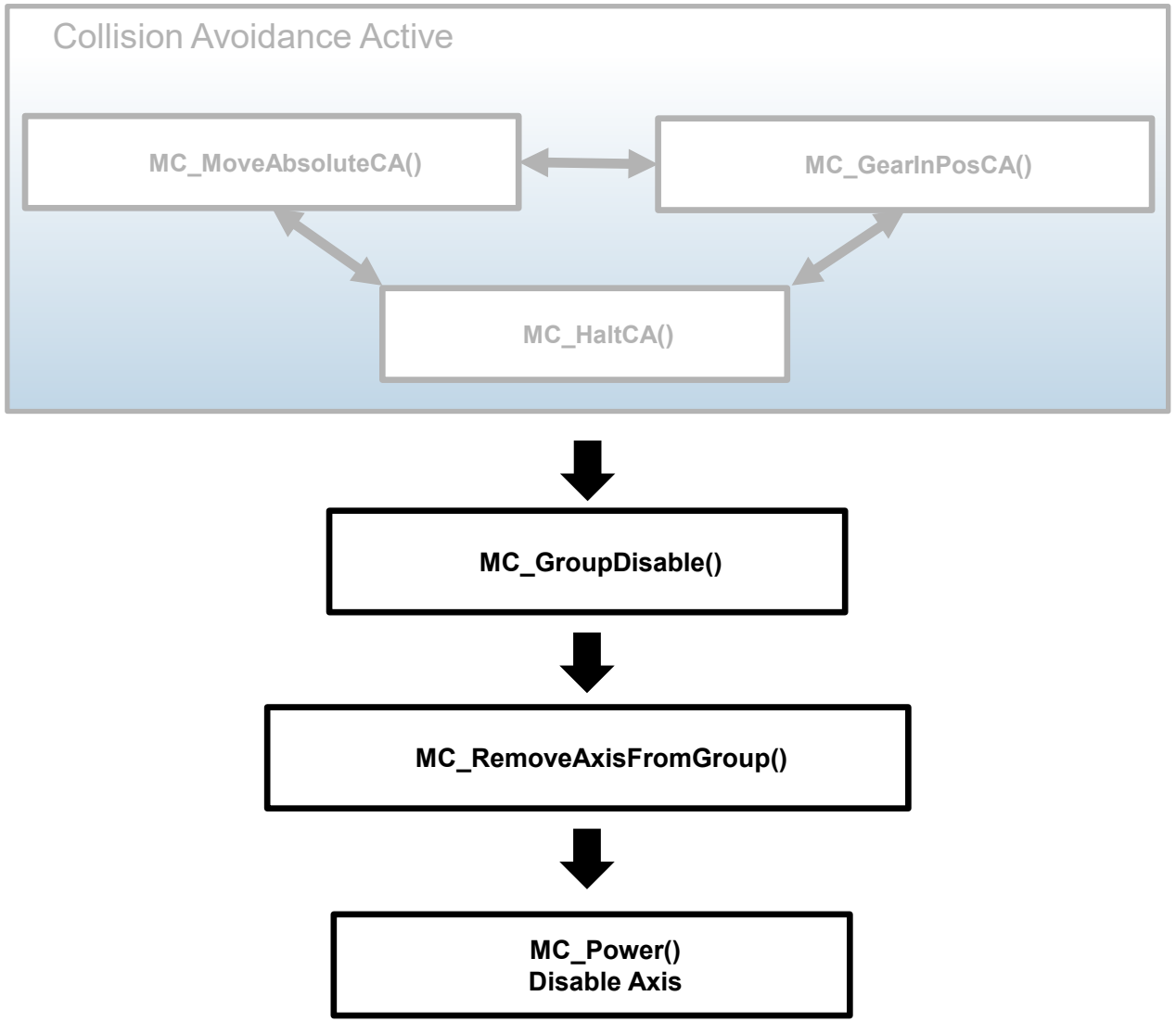
– With using a MC_CA-Command the Axis change in “CA-Active”

Full Sequence

Reset or Shutdown Sequence:

Halt-Command for all Movers

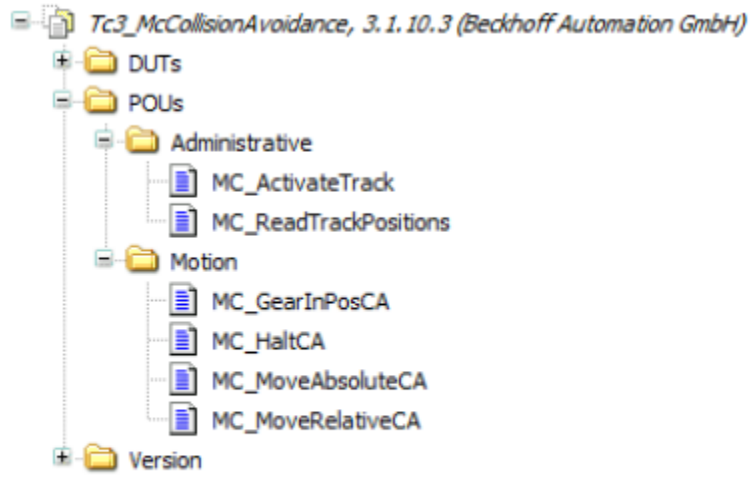
1. Disable Group
2. Remove all Axes from Group
3. Disable Axes



Agenda | XTS – PLC

BECKHOFF

1. PLC-Library XTS
2. AXIS_REF
3. XTS-Utility Library
4. PLC StartUp condition
5. Mover1 detection via PLC
6. Overview TF5400 Collision Avoidance (CA)
7. CA-Group Object
8. PLC-Library CA-Group
9. AXES_GROUP_REF
10. CA-Group handling
11. **CA-Operation**

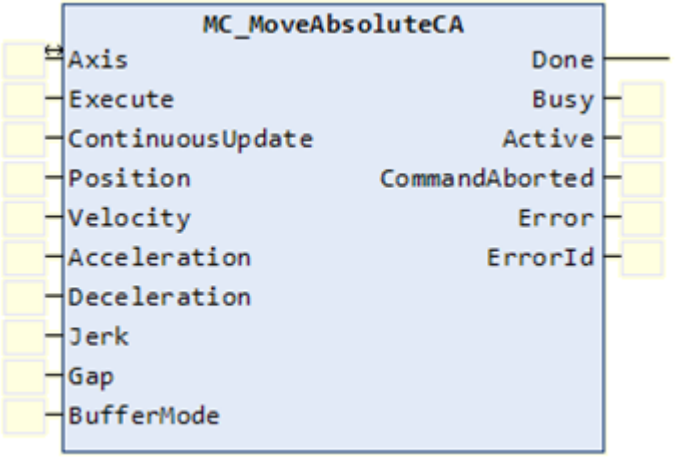


- MC_MoveAbsoluteCA
  Moves a single axis to an absolute position with collision avoidance.

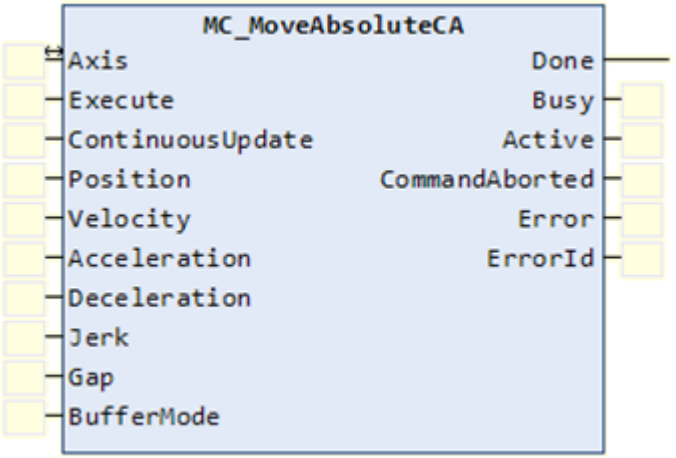
- MC_MoveRelativeCA
  Moves a single axis over a relative distance with collision avoidance.

- MC_HaltCA
  Stops a single axis with collision avoidance without locking it for further motion commands.

- MC_GearInPosCA
  Couples a slave axis with a gearing factor and collision avoidance to a master axis.



MC_MoveAbsoluteCA



This function block commands the specified mover to the specified absolute position according to the parameterized gap of the collision avoidance input.

The collision avoidance functionality has higher priority than the absolute motion command.

During execution of the move, the axis may slow down or stop and wait to avoid a collision. The axis will prioritize maintaining the gap overreaching the destination position.

When the path is clear the axis will complete the absolute position command.

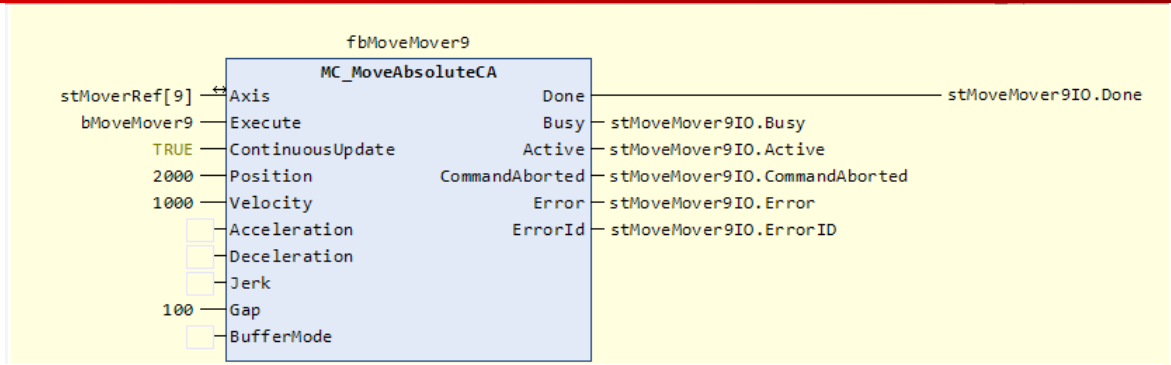
The Done output of the function block will become True when the target position has been reached.

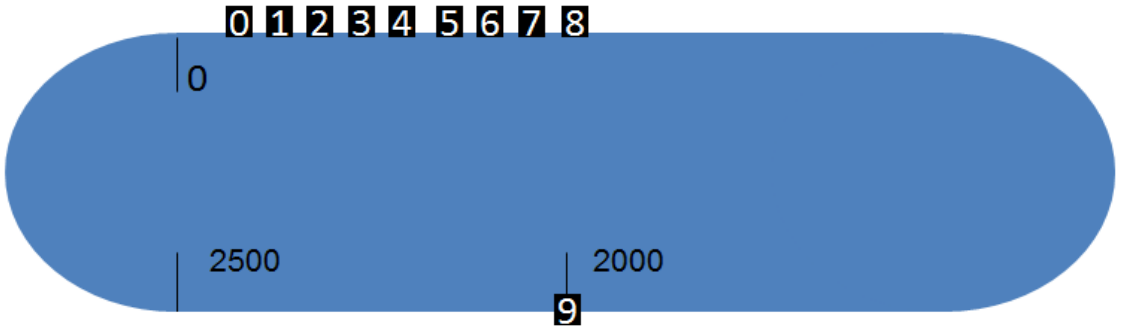
MC_MoveAbsoluteCA

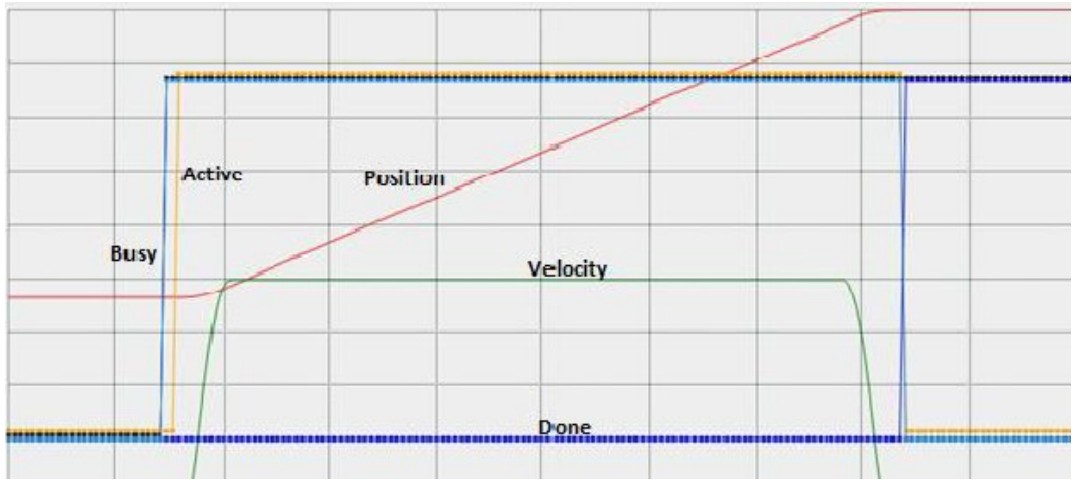
Mover 9 is free to move towards the target position of 2000 with a velocity of 1000.

Once the target position is reached, Done becomes True and Busy and Active become False.

The move has been completed successfully and mover 9 is now at position 2000.







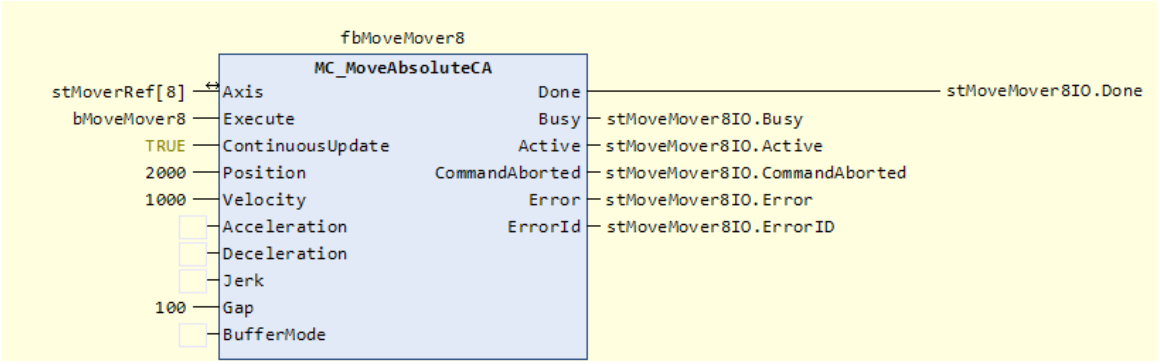
MC_MoveAbsoluteCA

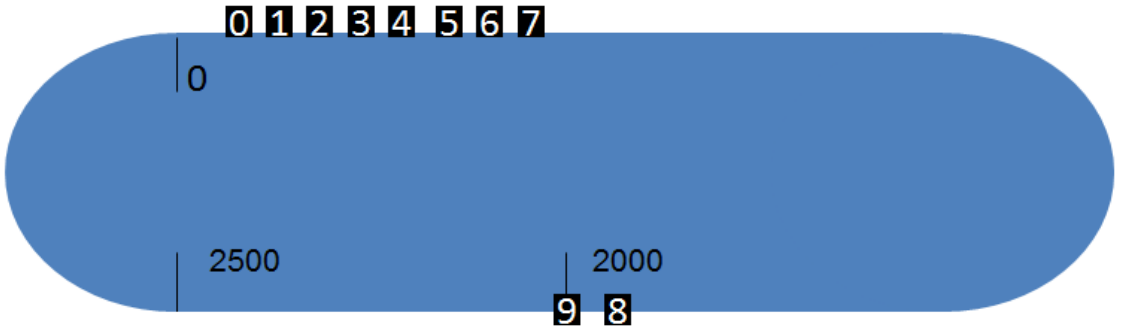
Mover 8 is commanded to move to the same position as Mover 9; however, Mover 9 is in the way. Collision Avoidance will prevent Mover 8 from reaching position 2000 but it will also prevent Mover 8 from colliding with Mover 9.

The profile of Mover 8 will look as follows.

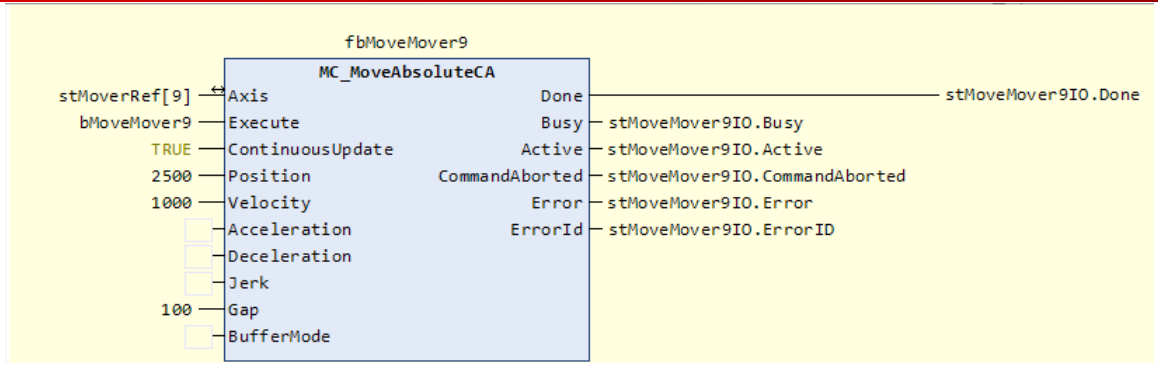
Mover 8 stops to maintain the gap spacing but the Active bit and the Busy bit remain true and Done remains false.

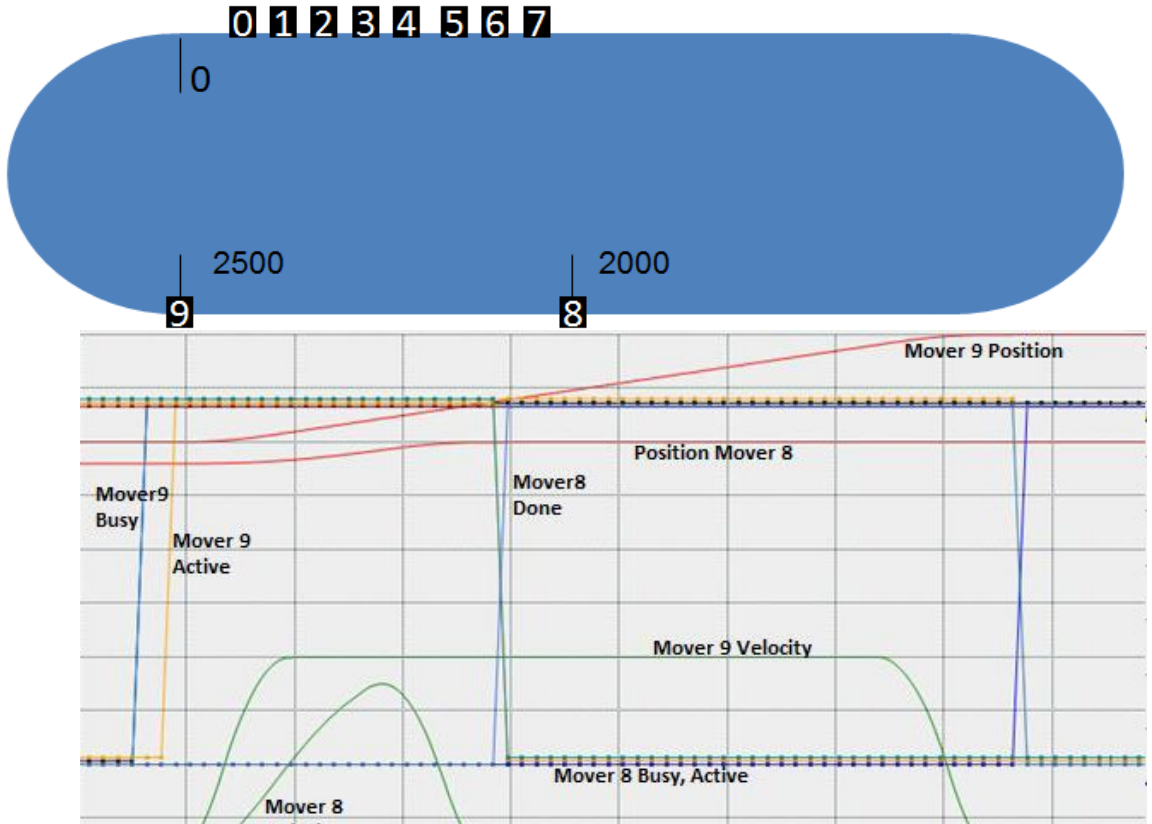
This move is not complete.





MC_MoveAbsoluteCA





Mover 9 is free to move towards the target position of 2500 with a velocity of 1000.

Mover 9 starts to move out of the way.

Mover 8 begins to move, ramps up its velocity and then stops in its final target position, “2000”.

Mover 8’s Busy and Active bits become false and the done bit for Mover 8 is set.

Mover 8’s mover is now complete.

Then finally mover 9 completes its commanded move.

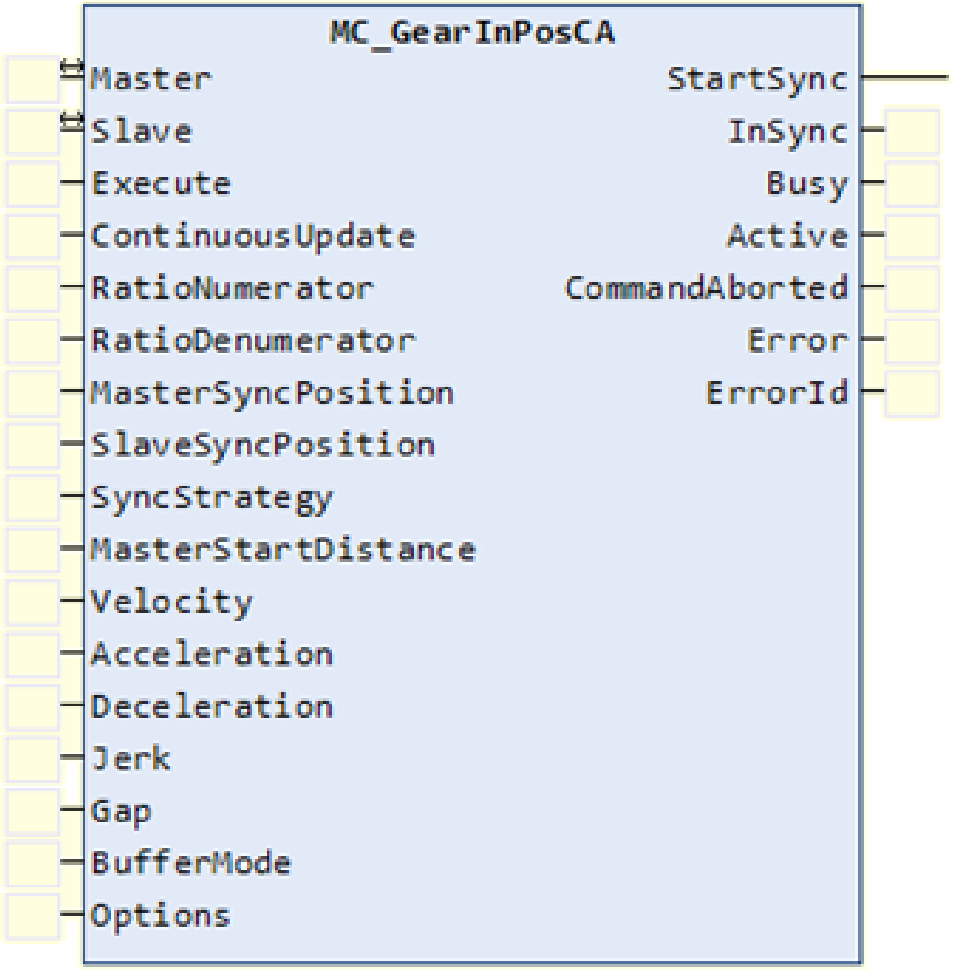
MC_GearInPosCA

This function block couples a slave axis to a master axis.

The set values for both axes are calculated based on the master axis set values.

Like all collision avoidance commands, priority is given to satisfying collision avoidance.

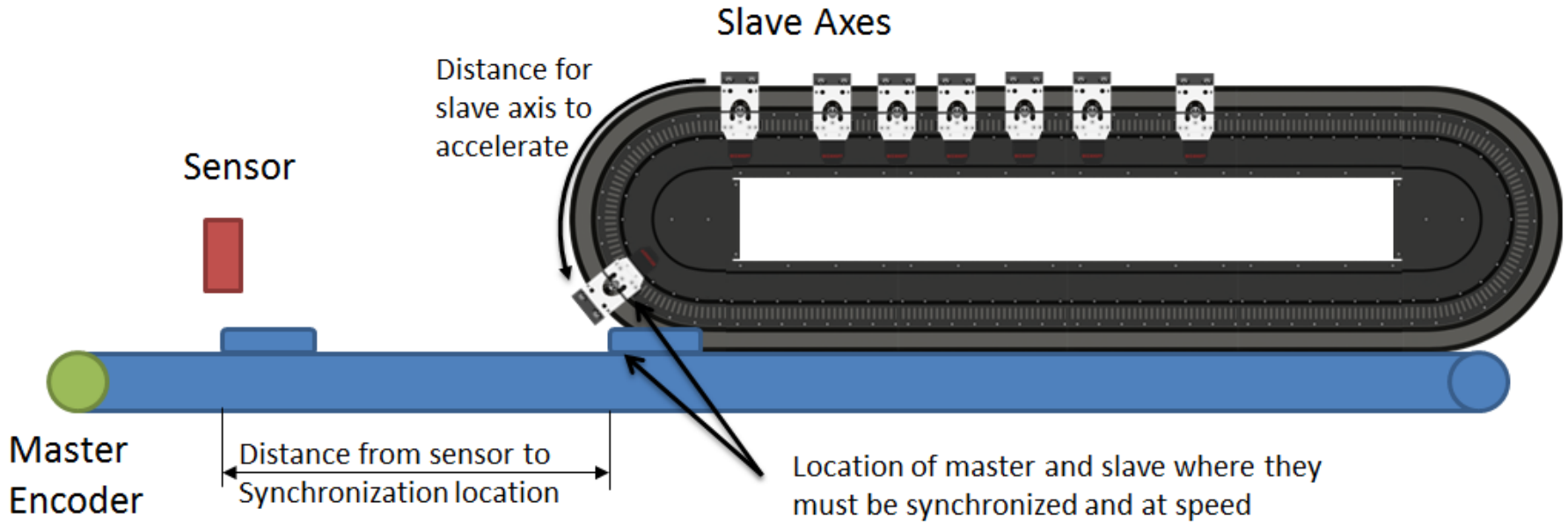
Completion of the commanded move occurs when there is no possibility of striking another axis or violating the collision avoidance gap.



MC_GearInPosCA

A typical use for the MC_GearInPosCA block is to synchronize a mover to a product on a conveyor. To accomplish this, the conveyor is equipped with a servo motor or encoder and a sensor is used to mark positions of products on the conveyor.

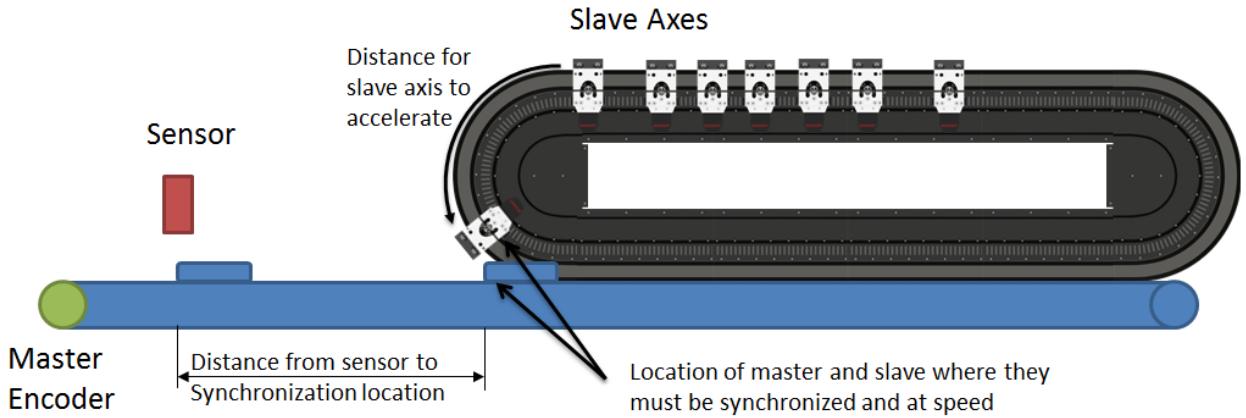
In all cases, Gap control overrides the move command, if the mover would approach another axis, it will slow down regardless of the command issued.



MC_GearInPosCA

Input „SyncStrategy“
from Type MC_SYNC_STRATEGY

This data type defines the synchronization profile of the slave



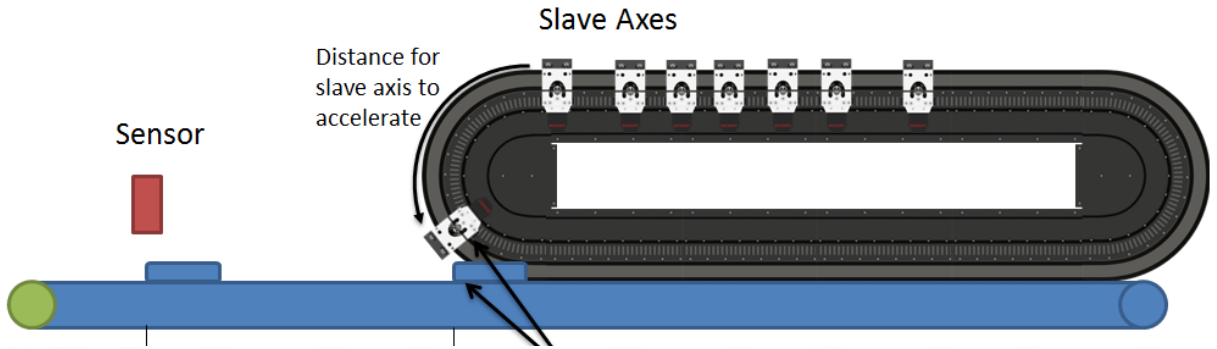
TYPE MC_SYNC_STRATEGY	
mcSyncStrategyLate	16#1
mcSyncStrategySlow	16#2
mcSyncStrategyEarly	16#3

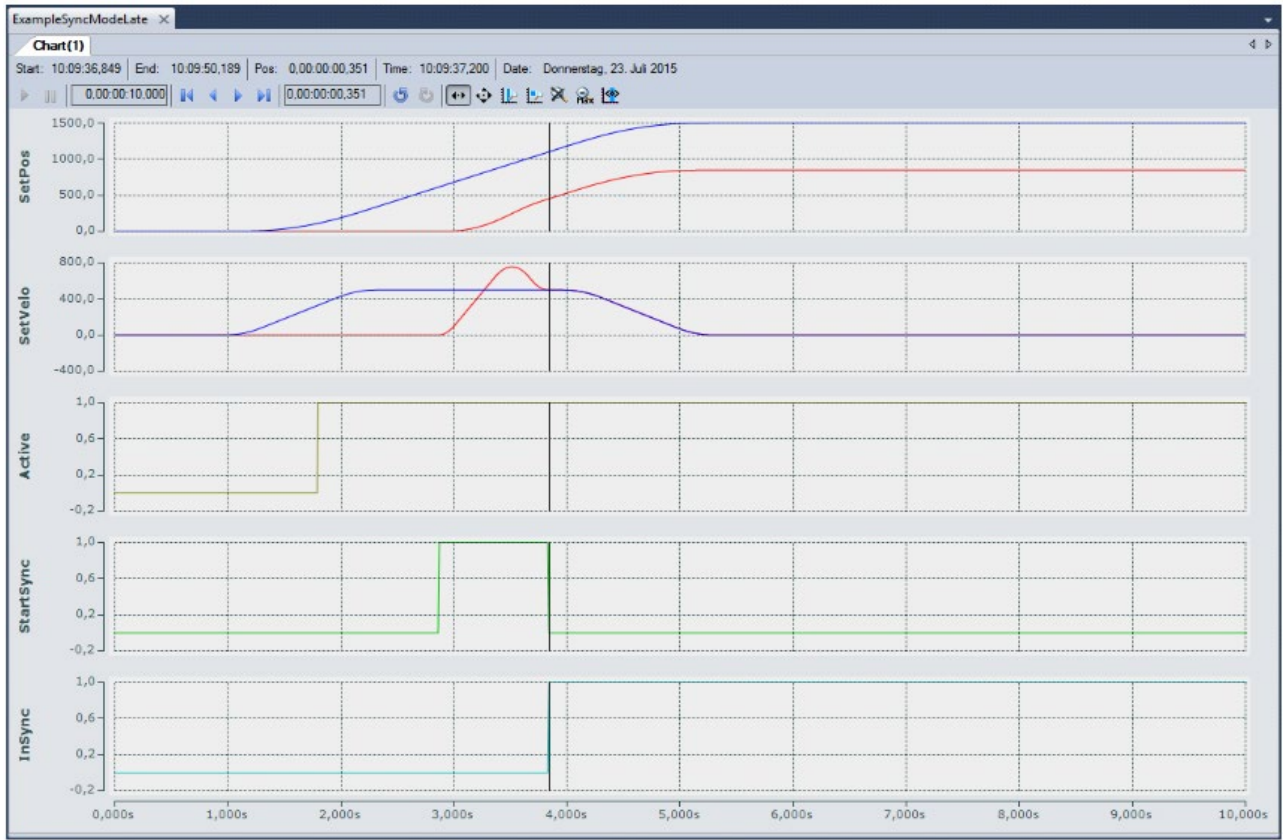
MC_GearInPosCA

mcSyncStrategyLate

The slave starts the synchronization as late as possible and with full dynamics (according to the input values velocity, acceleration, deceleration, jerk).

It reaches the SlaveSyncPosition just in time with the correct gear ratio.



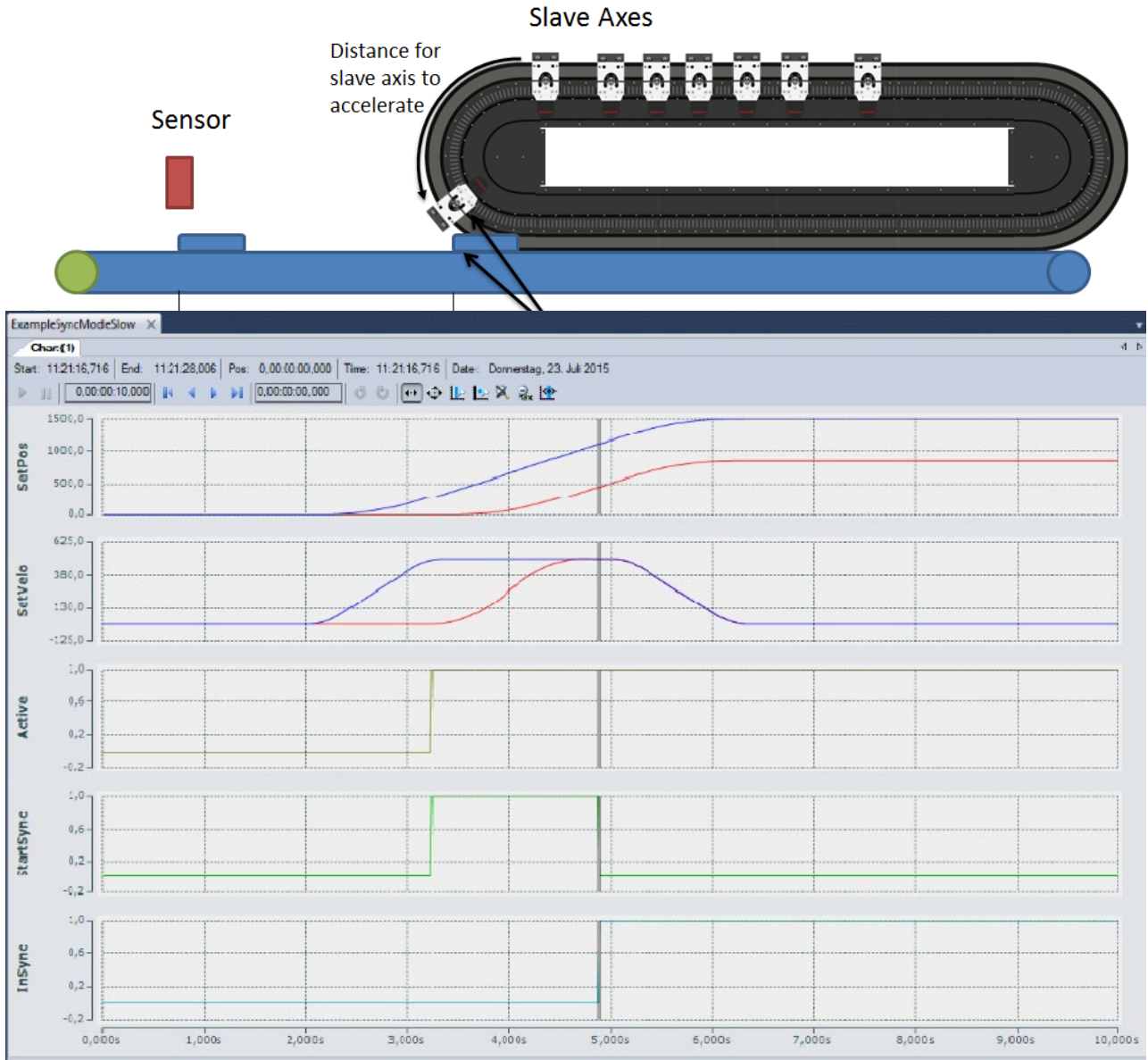


MC_GearInPosCA

mcSyncStrategySlow

The slave starts its sync in motion as soon as the master passes the MasterStartDist (MasterSyncPosition - MasterStartDistance). If the MasterStartDist is not set then the motion begins as soon as the FB is Active.

The dynamics of the slave are reduced such that the slave reaches the SlaveSyncPos with the correct gear ratio just in time when the master reaches the MasterSyncPos.



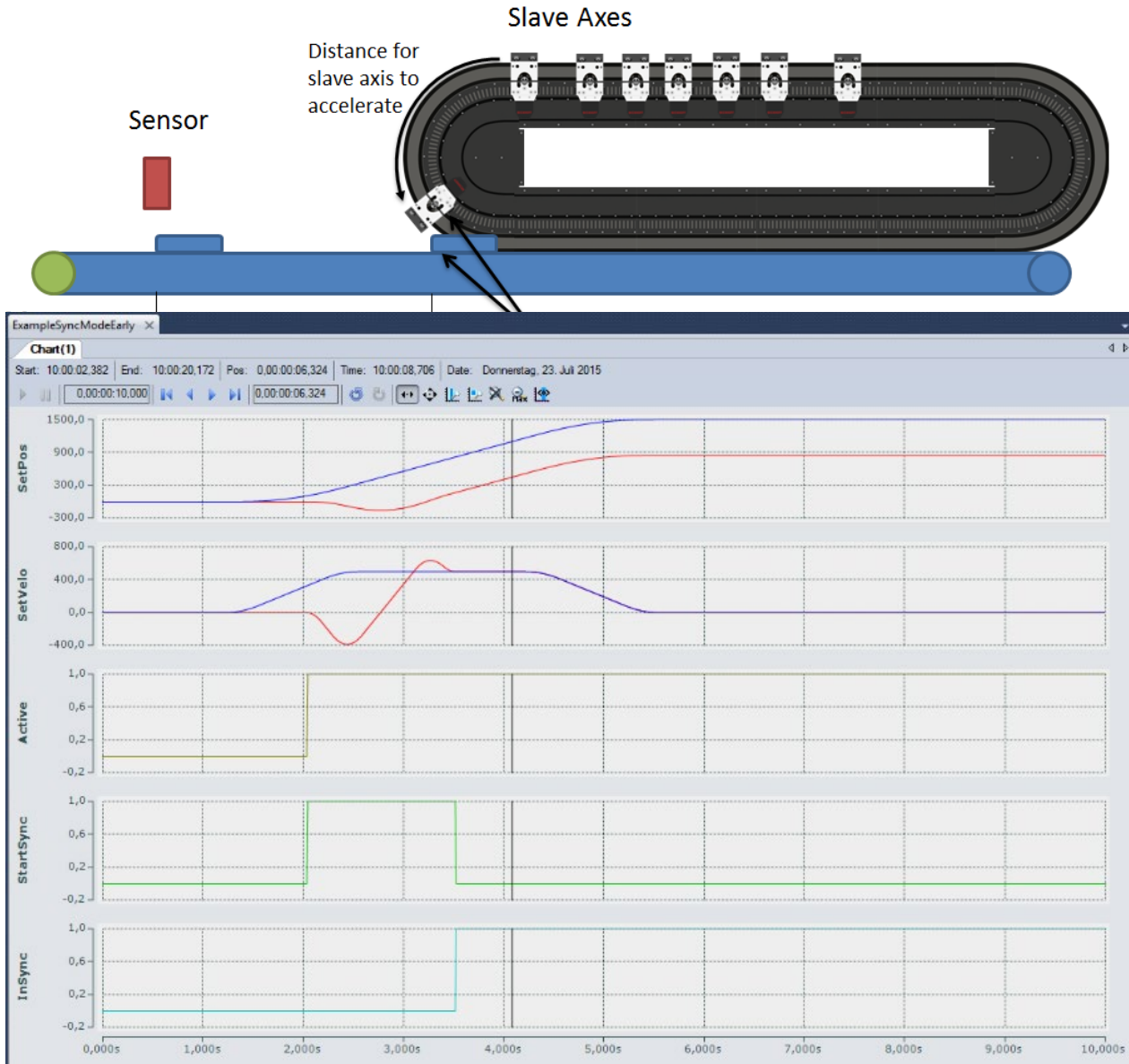
MC_GearInPosCA

mcSyncStrategyEarly

The slave starts its motion as soon as the master passes the MasterStartDist. If the MasterStartDist is not set then the motion begins as soon as the FB is Active.

The dynamics are not reduced.

The slave signals earlier InSync than demanded by the SlaveSyncPosition, but it is still guaranteed that the demanded offset between master and slave (MasterSyncPosition – SlaveSyncPosition) is reached with the correct gear ratio.



Complex sequences – simplified solution: XTS

Training material

BECKHOFF

This training material is provided to complement the presented training content. Outside the actual training the material may only be used for internal purposes at the company of the course participant. In addition, the material or extracts thereof may be used in end customer training for products containing Beckhoff products, or for presentations, provided the presentation refers to Beckhoff products. Extracts or copies of the training material must contain the following copyright acknowledgement: “© Beckhoff Automation GmbH & Co. KG”.

The same applies to extracts from presentation material. The user of the material is solely responsible for the completeness of extracts and copies. It is explicitly not permitted to offer commercial or free training for Beckhoff products. This applies to training with and or without the training material. The training material must not be edited, manipulated or modified.

Passing on of the aforementioned rights to third parties is not permitted.

Beckhoff Automation GmbH & Co. KG

Contact

BECKHOFF

Beckhoff Automation GmbH & Co. KG
Headquarters
Huelshorstweg 20
33415 Verl
Germany

Phone: +49 5246 963-0
E-mail: info@beckhoff.com
Web: www.beckhoff.com

© Beckhoff Automation GmbH & Co. KG 02/2021

All images are protected by copyright. The use and transfer to third parties is not permitted.

Beckhoff®, TwinCAT®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH. Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

The information provided in this presentation contains merely general descriptions or characteristics of performance which in case of actual application do not always apply as described or which may change as a result of further development of the products. An obligation to provide the respective characteristics shall only exist if expressively agreed in the terms of contract.