Twincat Library: Using PLCopen

Application Note

Version	Date	Editor	Comment
002	2012-11-27	mvx	Convert to new document style, add description of referencing
003	2013-01-16	mvx	Add path planner description and signal outputs.
004	2013-02-11	mvx	Add state diagram and the description of the block TL_MC_MoveVelocity
005	2014-01-16	mvx	Update for library 3.0.0 valid for TwinCat2 and TC3
006	2014-02-18	mvx	Move Absolute allows fast task reaction time with library 3.0.1
007	2014-04-29	mvx	Correct reference method naming
008	2014-10-20	dg	Reference with Tama added (SVN410)
009	2016-01-18	dg	Reference with Encoder[1]
010	2017-01-11	dg	Reference with Encoder[] replaced by option module Move Stop added
011	2017-10-18	dg	Some correction in the homing section added.
012	2021-06-07	mvx	Update for Trialink2 and TL_Axis2 types

Document AN108_TwinCAT-MotionControl_EP
Version 012
Source Q:\doc\ApplicationNotes\
Destination T:\doc\ApplicationNotes
Owner mvx

Triamec Motion AG
Lindenstrasse 16
6340 Baar / Switzerland

Phone +41 41 747 4040
Email info@triamec.com
Web www.triamec.com

Disclaimer

This document is delivered subject to the following conditions and restrictions:

- This document contains proprietary information belonging to Triamec Motion AG. Such information is supplied solely for the purpose of assisting users of Triamec products.
- The text and graphics included in this manual are for the purpose of illustration and reference only. The specifications on which they are based are subject to change without notice.
- Information in this document is subject to change without notice.

Table of Contents

1 Target and Purpose

The Triamec TwinCat library comes with basic sample codes for NCI and CNC. This application note describes the axis module and additional motion functions available in this library.

The sample code uses a function block TL_Trialink2 that contains the access code for the PCI-board and ***TL_Axis2*** that contains all functions required to control a Triamec drives axis from NCI or CNC.

Normally, the path planner runs in the NCI/CNC and the PLC application just ensures, that the drives will follow the commanded position of the CNC/NCI. The path planner of the drive comes into action only, if a reference sequence is started. If an application requires other moves be controlled by the drive path planner instead of the CNC/NCI, this application note describes on how to do that. Such movements are special reference sequences not covered by the standard reference method or velocity moves for spindle control.

The concept of this library requires two threads of the same CPU. The first thread ***MAIN_SLOW*** handles state changes and may run at a relatively slow rate, for example 10ms. The second thread ***MAIN_FAST*** receives positions for the CNC and forwards them to the Trialink. This should run at a fast rate, for example 0.5ms. All inputs and outputs are updated in MAIN_SLOW if not explicitely specified for MAIN_FAST.

2 Function block TL_Trialink2

This blocks handles access to the PCI boards. In MAIN_SLOW set the following declarations, inputs and calls

- ***Trialink : Triamec.TL_Trialink2;***
- ***Trialink.Execute*** TRUE boots the Trialink ring. FALSE stops any access to the PCI and ring.
- ***Trialink.Config.nDevId*** DPRAM id, see TwinCatSystemManager.
- ***Trialink.Config.RootFolder*** Set a path if you desire creation of log files
- ***Trialink.Config.TcEventEnable*** Enables TwinCAT message creation.
- ***Trialink.Config.FastFilterFrequency*** All commanded position go through this filter.
- ***Trialink.CallSlow();***

In MAIN_FAST call

Trialink.CallFast();

3 Function block TL_Axis2

All axes of the sample code are defined and controlled by global axis modules ".gAxis[iAxis]" of type TL_Axis2, where iAxis is the logical axis number of an axis.

This function blocks contains all functions necessary for enabling, reset, stop, and coupling. For extended functions, use the library function blocks TL_MC_* that aim at following PLCopen standards, but are not certified as PLCopen.

3.1 Call in MAIN_SLOW

For each Triamec axis, use the following global declaration and call in MAIN_SLOW after setting up the configuration

gAxis : ARRAY [1..N_AXIS] OF Triamec.TL_Axis2;

gAxis[iAxis].CallSlow(Trialink:=Trialink);

3.2 Configuration

- ***Config.Simulate*** The state signals hide servo errors. Position signals are not simulated.
  This helps setup a new machine software and testing direction and scaling.
- ***Config.Station*** The station number of the servo given during persistence setup.
- ***Config.SubAxis*** TRUE for the second axis of a dual axis servo
- ***Config.GearFactor*** Use 180/pi if the servo is configured for radian and the plc uses degree.
  use 1000 if the servo is configured for meter and the plc uses mm.
  use 1.0 else.
- ***Config.ModuloWrap*** Modulo wrap value. Use 360° if the plc uses degrees as unit.

3.3 Inputs

The module contains the following inputs. See the function descriptions below for details.

- ***enable*** see chapter below
- ***stop*** TRUE stops any ongoing movement and leaves the coupled mode.
  Any further movement cannot be started as long as the input remains TRUE.
- ***couple*** Normally TRUE, see chapter pathPlanner
- ***referenceEnable***
  referenceStart
  referencePosition See chapter "Reference".
- ***reset*** A positive edge will reset any drive side errors.

3.4 Signals

The following signals are updated with the slow task.

- ***ready*** The communication to the axis has been established.
- ***ReadyToOperate*** The axis may be enabled, e.g., DcBus voltage and temperatures are fine.
  (after FW 1037).
  This corresponds to internal states "ReadyToSwitchOn" or "Operational".
- ***enabled*** The position controller is active.
- ***act_pos*** The actual position of the axis
- ***act_err*** The actual error of the axis position controller (cmd_pos-act_pos)
- ***digitalInputBits*** The state of the digital inputs with binary encoding.
  Use the global TL_C for encoding, e.g.,
  TL_C.GenSig.DigitalInput.AuxIn1 AND gAxis[1].din <> 0
- ***followMe*** The external cnc path planner should follow the servo.
  coupled The drive is following the external path planner
- ***error, errorId*** See below.

The errors of the sub-functions are merged into one boolean ***error*** and one number ***errorId*** with the following priorities

1. axis error or warning
2. homing error
3. power error

Any ***errorId***<>0 indicates an error or warning. The boolean error discriminates between

- errors (***error***=TRUE) or
- warnings (***error***=FALSE).
- Homing errors are treated like warnings and set referenceError TRUE.

The state diagram of the drive is shown in Figure 1 with the major state and one minor state per axis. The minor states are active if the major state is "Operational" and indicate that the motor PWMs are active. There are several classes of errors.

- major errors with internal state "Faultpending" or "FaultReactionActive", where the PWM is not operational and
- minor errors with internal state "Operational-Disabled" and error-pending where the PWM remains active.
- Errors not related to the drive, but to PLC-side configuration or communication errors.

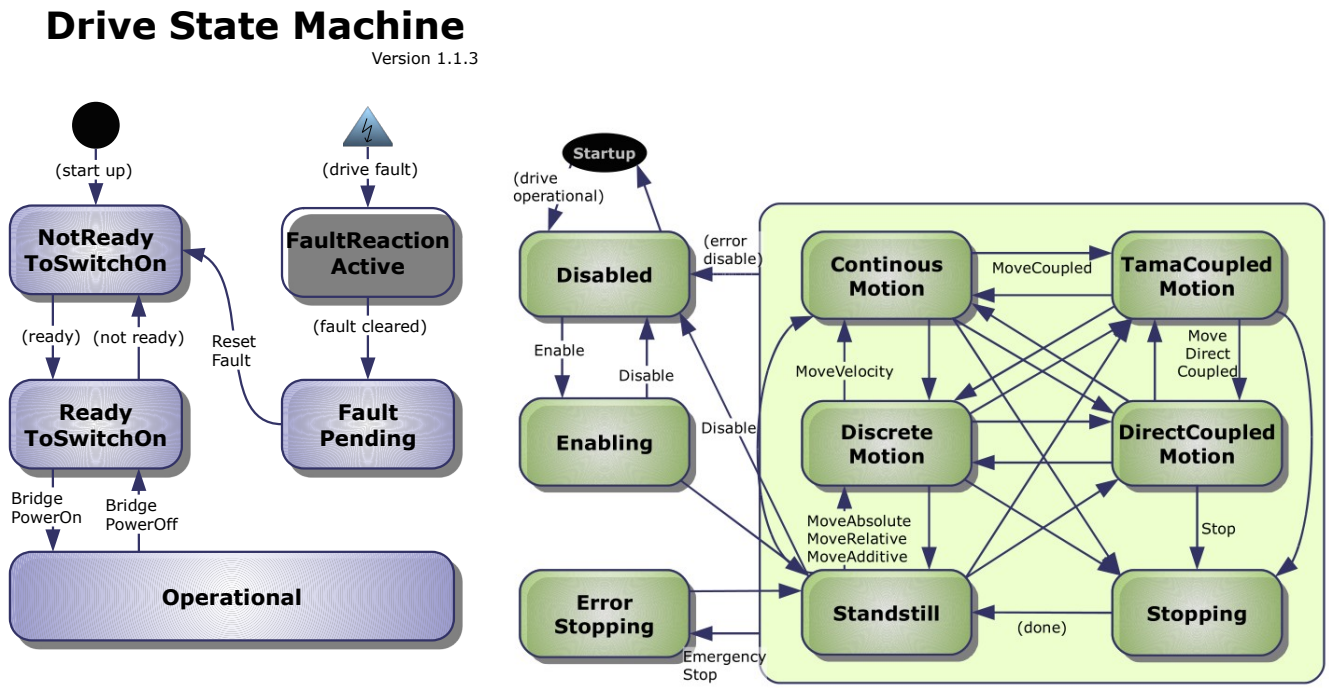


Figure 1: Statediagram of the Triamec Drives with the major state (left) and one minor state (right) per axis.

3.5 Enable

The input ***enable*** enables the axis. The input is not edge sensitive. If an axis is disabled due to an axis error and the input remained TRUE, it will enable again after clearing the error.

After a negative edge, an axis is first stopped. If stopped or the time ***intern.power.MaximumStopping-TimeBeforeBrake*** is passed, any configured axis brake will brake (See the TAM System Explorer Quick Start Guide on how to configure an axis brake). Set this to zero if an axis should not be stopped before disabling.

3.6 Call in MAIN_FAST

Call this in the MAIN_FAST thread, which must reside within the same CPU as the MAIN_SLOW but called at a higher rate

gAxis[iAxis].CallFast(Trialink:=Trialink);

Use the following for actual position and commanded position

gAxis[iAxis].fastPositionCmd := commandedPos;

gAxis[iAxis].CallFast(Trialink := Trialink);

actualPos :=***gAxis[iAxis].ActualPositionFast(Trialink:=Trialink);***

3.7 Path planner

While the axis *position controller* is always running at the drive side (if enabled), the axis *path planner* is running either on the drive or in TwinCat.

Drive pathplanner	The path planner runs on the drive. Moving to a new position or starting a velocity move is commanded asynchronously by function blocks "TL_MC_*". Such functions are used in the homing sequence or with extended motion functions.
TwinCat coupled pathplanner	The NC/CNC or a custom PLC code runs the path planner. An axis enters coupled mode using the function block TL_MC_MoveSynchronized, driven by the input `couple` of TL_Axis2.

After enabling, the axis starts up with the drive path planner.

The user must command coupling to give control to the cnc path planner. If the customer does not use extended motion items, the input couple may be set to TRUE always. The following sequence describes the behavior when using extended motion features or to aid debugging, if something went wrong:

Lets assume, the axis is not enabled yet. The output ***followMe*** indicates, that the axis is not following the commanded position of the plc, i.e., it is not coupled. This output should be used to start axis tracking on the NC side, e.g., the cnc should start to follow the axis actual position. Otherwise there will be a hard jump on starting to couple.

The couple sequence starts after a positive transition of the input ***couple*** or a steady TRUE couple input just after enabling/error recovery. The commanded positions from the PLC path planner are now streamed to the axis. Then the path planner of the drive starts to follow the commanded PLC positions (streaming). Feeding the axis with positions from a PLC path planner is done with the following input of TL_Axis2 in the task MAIN_FAST:

fastPositionCmd (LREAL)

Make sure, the positions streamed before this transition are equal to the actual position of the drive.

Any movement command, error, stop or drive based homing will stop this coupling and the path planner of the drive takes control. This is indicated by output ***couplingBroken*** as long as ***couple*** remaines TRUE. The axis will not re-couple until the next positive edge of "couple" or when changing from disabled to enabled state.

The following code of ***TL_Axis2*** ensures, that the coupling is repeated after a drive based homing or after clearing any axis error:

```
Marker couplingDelay.IN := couple AND (NOT ReferenceBusy) AND enabled AND
          NOT simulate) AND NOT
          (reset AND (MC_axis_Coupler.CommandAborted OR error));
   couplingDelay(PT := T#20ms); (* this delays coupling, to allow setpoint filter settling *)
   MC_axis_Coupler.execute := couplingDelay.Q;
```

Other cases of lost coupling, for example after a user defined absolute move, require a FALSE TRUE transition of couple for the external path planner to gain control again.

3.8 IndexReference move (Homing)

The standard reference function implements several homing procedures. To select the homing procedure, assign the required method form ***TL_HomingMethod*** to ***axis[axisId].Config.ReferenceMethod*** for example:

```
axes[1].Config.ReferenceMethod     := TL_HomingMethod.Immediate;
```

The following homing procedures are implemented (see also below for more detailed description):

Name	Short description
Disabled	Do nothing if homing is executed. The output ***referenced*** remains FALSE.
DriveControlled	The homing sequence is started by the host as ususal, but the sequence is carried out in the drive. This function is not available for first generation drives (TS381, etc)
Immediate	Consider homing as done and set output ***referenced*** = TRUE without doing SetPosition.
Index	Move to Index and SetPosition.
Marker	Move to digital IO marker and SetPosition.
MarkerIndex	First move to digital IO marker, then to Index then SetPosition.
SetPosition	Set position to the value of ***referencePosition*** of ***TL_Axis2.***
Tama	The asynchronous Tama VM is used to execute the homing sequence.
Index_Option	Same as 'Index' but encoder of option module is used for the index search and the definition of the homing position[1].
Marker_Option	Same as 'Marker' but encoder of option module is used for the definition of the homing position[1].
MarkerIndex_Option	Same as 'MarkerIndex' but but encoder of option module is used for the index search and the definition of the homing position[1].

Execution of the Homing

Setting ***ReferenceEnable*** to TRUE starts the homing sequence. Setting ReferenceEnable to FALSE clears the ***homing done*** flag. Alternatively, the sequence may be started by a positive edge of ***ReferenceStart***. In this case the ReferenceEnable must not be written cyclically, but it is still used to reset homing by writing FALSE. Consider AN107 chapter 8 on how to use homing together with the Beckhoff HMI.

If required by the configured ***ReferenceMethod***, the reference sequence is finished by setting the position to the configured ***referencePosition*** and the output ***referenced*** is set to TRUE.

All units are in PLC units u and u/s, which may be different from drive units if the GearFactor is not 1.

If a configured distance is reached before the corresponding event is detected, an error is thrown and the axis stopped.

1 Homing procedures which use the option module encoder are just supported if the drive is equipped with suitable option module. Using this procedure with a drive which supports just one encoder per axis causes an error.

Method: DriveControlled

This is the default homing method. The homing is carried out in the drive axis as specified in AN107 chapter 8.

Method: Index or Index_Option[1]

The homing ***index*** move is a move of the drive to the index marker of the encoder. The direction of the move is specified by the sign of the distance parameter. If ***Index_Option*** is configured, the encoder of the option-module is used for the homing sequence.

```
axes[1].Config.ReferenceMethod            := TL_HomingMethod.Index;
axes[1].Config.ReferenceIndexDistance     := 6.3;  (* direction and maximum distance of index move*)
axes[1].Config.ReferenceIndexVelocity     := 1.2;  (* velocity of index move, always positive     *)
```

Method: Marker or Marker_Option[1]

The homing ***marker*** move is a move of the drive until an endmarker or similar is detected. Use global structure ***TL_Config.ReferenceFirstInput*** to specify the digital input for the first move. Depending on the drive, not all inputs of the structure are supported. If an input is configured which is not supported, an error is thrown when the homing starts.

Supported Digital Inputs from structure ***TL_Config.ReferenceFirstInput***:

Drive Type	Supported digital inputs
TSD80, TSDXXX	IndexA, ExtIo0A, ExtIo1A, AuxIn1, AuxIn2, AuxIn3, AuxIn4, AuxIn5, AuxIn6
others	IndexA, ExtIo0A, ExtIo1A, AuxIn1, AuxIn2, AuxIn3, AuxIn4, AuxIn5, AuxIn6

The ***ReferenceFirstDistance*** specifies the maximum distance for the move and its sign specifies in which direction the move should start if the value of the ***ReferenceFirstInput*** is FALSE. If the input is TRUE from beginning, the move is done in opposite direction. The move stops, if the input sign changes.

If ***Marker_Option*** is configured and if the drive is equipped with a suitable option module, the encoder of the option module is used for the definition of the homing position.

```
axes[1].Config.ReferenceMethod            := TL_HomingMethod.Marker;
axes[1].Config.ReferenceFirstMask         := TL_Config.ReferenceFirstInput.AuxIn1;
axes[1].Config.ReferenceFirstDistance     := -0.2; (* direction and maximum distance of first move*)
axes[1].Config.ReferenceFirstVelocity     := 0.002;(* velocity in u/s, always positive           *)
```

Method: MarkerIndex or MarkerIndex_Option[1]

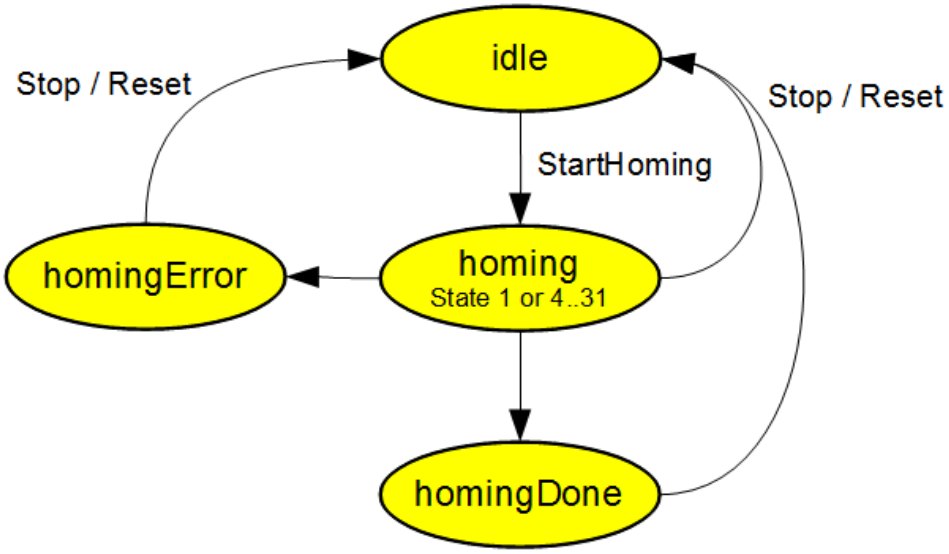
First a move to the marker is executed. When the marker is found, a move to the index marker of the encoder is done. To configure the homing method, same parameters are used as for the method ***Index*** and ***Marker***.
If ***MarkerIndex_Option*** is configured if the drive is equipped with a suitable option module, the encoder of the option module is used for the homing sequence.

```
axes[1].Config.ReferenceMethod            := TL_HomingMethod.MarkerIndex;
axes[1].Config.ReferenceFirstMask         := TL_Config.ReferenceFirstInput.AuxIn1;
axes[1].Config.ReferenceFirstDistance     := -0.2; (* direction and maximum distance of first move*)
axes[1].Config.ReferenceFirstVelocity     := 0.002;(* velocity in u/s, always positive           *)
axes[1].Config.ReferenceIndexDistance     := 6.3;  (* direction and maximum distance of index move*)
```

```
axes[1].Config.ReferenceIndexVelocity        := 1.2;  (* velocity of index move, always positive     *)
```

Method: Tama

If ***Tama*** is configured as reference method, it is possible to program the homing sequence as a Tama program. The Tama program has to implement a state machine with the following structure and states and commands. The Tama program has to reset the command to *NoCommand* as soon as the execution of the command starts.



AsynchronousMainState		AsynchronousMainCommand	
0	idle	**0**	NoCommand
1	homing	**1**	StartHoming
2	homingError	**2**	Stop / Reset
3	homingDone		
4...31	additional states		

4 Additional function blocks

The following PLCopen blocks are not part of the main module TL_Axis2 but can be used side-by-side with this module. Use the PLC-Open Reference ***MC_Axis*** of TL_Axis2 in all PLC-Open function blocks.

4.1 TL_MC_MoveAbsolute

Positive transition of the input "Execute" starts an absolute move to "Position" with "Velocity". The drive state (see chapter 3.4) will be "Operational-DiscreteMotion" during the move and "Operational-Standstill" after successfully reaching the desired position. The block output "Done" becomes TRUE for at least one cycle or until Execute is set to FALSE, when the final position was reached. As an example the following code might be used:

```
(* ---- Declaration ---- *)
VAR
   moveAbsolute  : TL_MC_MoveAbsolute;
END_VAR
(* ---- Code ---- *)
moveAbsolute.Position       := 3.0;
moveAbsolute.Direction      := TL_C.AxisPar.PathPlanner.Direction.Shortestway; (* default *)
moveAbsolute.DiscardVelocity:= TRUE;      (* ignore the Velocity input and use the *)
                                          (* drive pathplanner value of the velocity instead *)
moveAbsolute.Velocity       := 0.0;
moveAbsolute.Execute        := ***;
(* to be called in the slow task *)
moveAbsolute(axis:=axis.MC_axis, Trialink:=Trialink);
(* if Execute is set in the fast task, add the following line *)
(* in the fast task in addition to the slow task call above *)
moveAbsolute.CallFast(axis:=axis.MC_axis, Trialink:=Trialink);
```

Interrupting such a motion is done by the stop input of the module TL_Axis2 or by another PLC Open motion block. If another PLCopen block interrupts this motion, the output "CommandAborted" becomes true if and until the Execute input is set back to FALSE.

See the chapter *Pathplanner* on how to re-couple an axis to the PLC pathplanner after such a move.

If the input "DiscardVelocity" is TRUE or the Velocity is larger than the axis pathplanner velocity parameter, the velocity is set to the path planner value. Be aware that the drive-side path planner parameter "DynamicReductionFactor" may reduce the velocity you get.

TL_MC_MoveVelocity

Positive transition of the input "Execute" starts a Velocity move with parameter "Velocity". The drive state (see chapter 3.4) will be "Operational-ContinousMotion". The block output "Done" becomes TRUE for at least one cycle or until Execute is set to FALSE.

Interrupting such a motion is done by the function block TL_MC_Stop or any another motion block "TL_MC_*". If such a function block interrupts this motion, the output "CommandAborted" becomes true if and until the Execute input is set back to FALSE.

To re-couple an axis to the PLC pathplanner within such a move, stop the axis first and make sure the CNC commanded position is equal to the drive actual position. Then procede as discussed in the chapter *Pathplanner*.

If the Velocity is larger than the axis pathplanner velocity parameter, the velocity is set to the path planner value. Be aware that the drive-side path planner parameter "DynamicReductionFactor" may reduce the velocity you get.

5 Error messages and thread safety

The following TwinCAT warning information is important for error messages ([2]).

```
Stringfunktionen Bitte beachten: String-Funktionen sind nicht sicher bei Taskwechsel: Bei der
Verwendung von Tasks dürfen String-Funktionen nur in einer Task eingesetzt werden. Wird die gleiche
Funktion in verschiedenen Tasks benützt, besteht die Gefahr des Überschreibens.
```

2 http://bkinfosys.beckhoff.com/index.php?content=../content/1031/tcplccontrol/html/tcplcctrl_componentsoptions.htm&id=12819

TwinCAT mentions using the compiler parameter "Enable Inline String functions" which did not help in our case. We therefore recommend to call state handling (MAIN_SLOW) in the same thread as HMI handling (MAIN). Then all string functions are called in the same thread.