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

Document	AN108_TwinCAT-MotionControl_EP
Version	006
Source	Q:\doc\ApplicationNotes\AN108_TwinCAT-MotionControl\
Destination	T:\doc\ApplicationNotes
Owner	mvx

Triamec Motion AG
Industriestrasse 49
6300 Zug / 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_AxisSlow*** that contains all functions required to control Triamec drives 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.

Also, this note describes the function blocks inside "TL_AxisSlow" and its code is attached.

2 Function block TL_AxisSlow

All axes of the sample code are defined and controlled by axis modules ".gAxis[iAxis]" of type TL_AxisSlow, where iAxis is the logical axis number of an axis. The code of this module is attached to this application note for the interested reader.

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.

2.1 Configuration

- ***Config.Simulate*** The state signals hide servo errors. This helps setup a new machine software. Position signals are not simulated. This helps 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.

2.2 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***
- ***referenceEnable***
  referenceStart
  referencePosition See chapter "Reference".
- ***reset*** A positive edge will reset any drive side errors.

2.3 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)
- ***din*** 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.

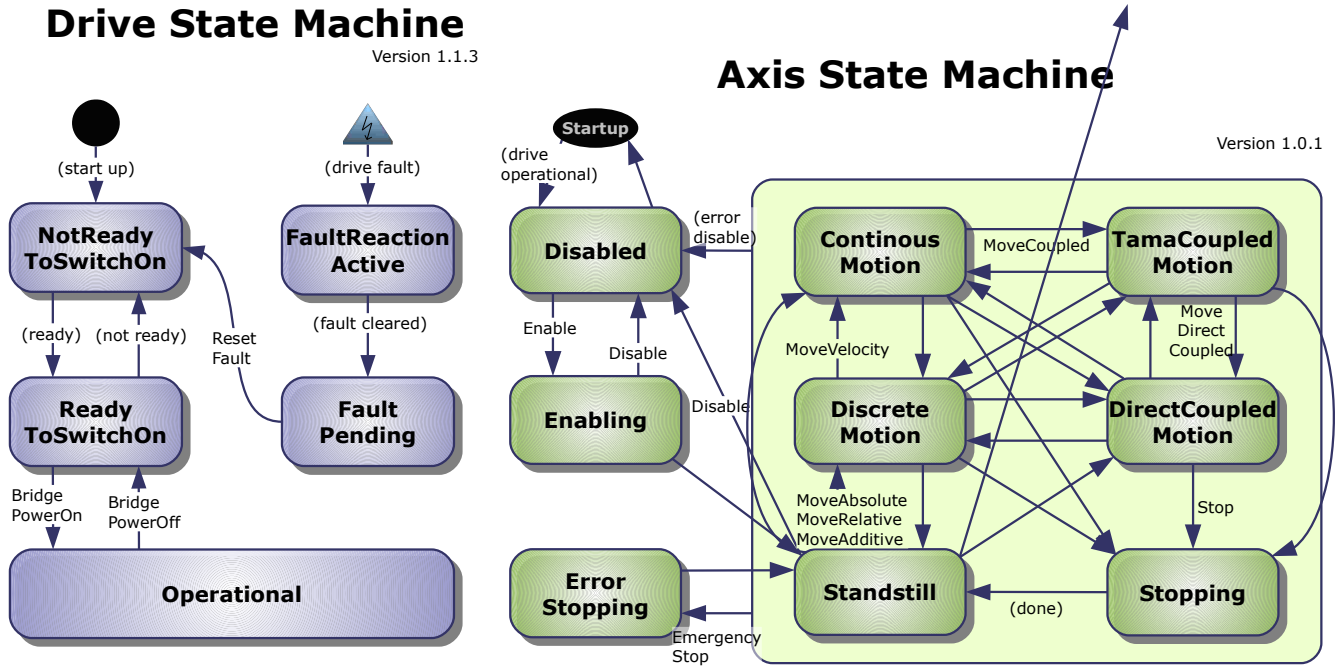

Figure 1: Statediagram of the Triamec Drives with the major state (left) and one minor state (right) per axis.

2.4 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 ***MC_axis_Power.MaximumStoppingTimeBeforeBrake*** 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.

After the time ***MC_axis_Power.BrakeTimeBeforeDisable*** the axis is disabled.

2.5 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.
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_AxisSlow.

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

The user must command coupling to give control to the cnc path planner. A positive transition of the input ***couple*** will couple the axis to the PLC path planner, i.e., the path planner of the drive will start to follow the commanded PLC positions (streaming). Feeding the axis with positions from a PLC path planner is done with objects of type ***TL_AxisFast*** in the task MAIN_FAST.

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

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.

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 ***MC_axis_Coupler.CommandAborted*** as long as ***couple*** remaines TRUE. The axis will not re-couple until the next positive edge of "couple".

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

```
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.

2.6 Reference move (Homing)

The standard reference function implements several homing procedures. This is configured using

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

The following sequences are implemented

- Disabled Do not do anything
- Immediate Consider homing as finished immediately without doing SetPosition
- Marker Move to digital IO marker and SetPosition (refer to parameters keyword "First")
- Index Move to Index and SetPosition
- IndexMarker First move to digital IO marker, then to Index then SetPosition

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.

The homing ***marker*** move is a move of the drive until an endmarker or similar is detected. Use global struct ***TL_Config.ReferenceFirstInput*** to specify the digital input for the first move. The distance specifies the maximum distance for the move and its sign specifies in which direction the move should start if the input is FALSE. If the input is TRUE from beginning, the move is done in opposite direction. The move stops, if the input sign changed.

```
axes[1].Config.ReferenceFirstInput        := 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     *)
```

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.

```
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     *)
```

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

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

If specified by the *ReferenceMethod* input, the reference sequence is finished by setting the position to ***referencePosition*** and setting the output ***referenced*** TRUE.

3 Additional function blocks

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

3.1 TL_MC_MoveAbsolute

Positive transition of the input "Execute" starts an absolute move to "Position" with "Velocity". The drive state (see chapter 2.3) 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_AxisSlow 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.

3.2 TL_MC_MoveVelocity

Positive transition of the input "Execute" starts a Velocity move with parameter "Velocity". The drive state (see chapter 2.3) 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.

4 Error messages and thread safety

The following TwinCAT warning information is important for error messages ([1]).

```
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.
```

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.

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