Twincat Library: Accessing Drive Registers

Application Note

Version	Date	Editor	Comment
002	2012-1127	mvx	Convert to new document style
003	2014-01-16	mvx	Update for library 3.0.1

Document AN109_TwinCAT-AccessingDriveRegisters_EP
Version 003
Source Q:\doc\ApplicationNotes\
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 additional functions available in this library.

This note describes how to read and write to drive (dsp or fpga) registers through an edge trigger. If an application requires cyclic information at a high rate, consider AN105.

2 PLC Code

The code below sets the motor ***temperature sensor type*** and ***-limit*** and reads the ***temperature***.

The SET input of ***TL_MC_RegisterWrite*** depends on the input data type. Use

- setBin to specify an integer value
- setFloat to specify a float value
- The "setBin" version must be used, whenever 0<>setReg AND TL_REG_FLOAT32_MASK

The GET outputs of ***TL_MC_RegisterRead*** depend on the data type. Use

- GetBin to read an integer value (DWORD)
- GetFloat to read a float or integer value

The value is set or read after a positive edge on Execute.

Declaration in MAIN_SLOW

```
motorTemperature                    : TL_MC_RegisterRead;
motorTemperatureLimit               : TL_MC_RegisterWrite;
motorSensorType                     : TL_MC_RegisterWrite;
```

and the code

```
motorSensorType.Execute             := gAxis[4].ready;
motorSensorType.SetReg              := gAxis[4].MC_Axis.register.GenPar.MotorSensorType;
motorSensorType.SetBin              := TL_C.GenPar.MotorSensorType.KTY84;
motorSensorType(axis:=gAxis[4].MC_axis, Trialink:=Trialink);

motorTemperatureLimit.Execute       := motorSensorType.Done AND (Trialink.OccationalTimer <> 333);
motorTemperatureLimit.SetReg        := gAxis[4].MC_Axis.register.GenPar.MotorSensorUpperErrorLevel;
motorTemperatureLimit.SetFloat      := 80;
motorTemperatureLimit(axis:=gAxis[4].MC_axis, Trialink:=Trialink);

motorTemperature.Execute            := Triamec.Communication_Ready;
motorTemperature.SetReg             := gAxis[4].MC_Axis.register.GenSig.MotorTemperature;
motorTemperature(axis:=gAxis[4].MC_axis, Trialink:=Trialink);
```

Note, that the IN_OUT parameter ***axis*** uses the PLCopen-Axis "***.MC_axis***" of type ***TL_MC_AXIS_REF*** not the general axis module "***gAxis[4]***" of type ***TL_AxisSlow***.

Use intelliSense to find the possible registers and constants.