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
004	2015-10-13	dg	Update register names
005	2017-11-20	mvx	Add EtherCAT versions

Document AN109_TwinCAT-AccessingDriveRegisters_EP
Version 005
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 for TriaLink devices

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.MotorSensor.SensorType;
motorSensorType.SetBin              := TL_C.GenPar.MotorSensorType.KTY84;
motorSensorType(axis:=gAxis[4].MC_axis, Trialink:=Trialink);

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

motorTemperature.Execute            := 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.

For TSD80 drives, the two registers are:

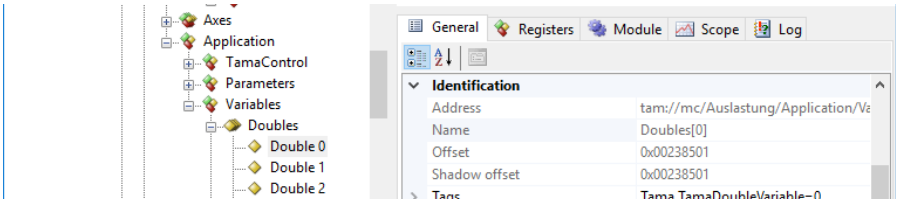
```
gAxis[4].MC_Axis.register2.Axes[x].Environment.MotorTemperatureSensorType
gAxis[4].MC_Axis.register2.Axes[x].Environment.MotorTemperatureUpperLimit
```

3 EtherCAT devices

Triamec EtherCAT drives support two register ranges. The first set is published to TwinCAT by the COE-Information method. As a consequence, these are shown in the TwinCAT tab “COE-Online”. A second range is not shown directly. This large set of registers is usually accessed using the TAM System Explorer when tuning the axis-motor-drive system. Nevertheless, the registers can be accessed from TwinCAT by the same method as those published as COE-Information.

3.1 Find the COE address

Open the TAM System Explorer and select the register in the tree view to your left. Use the tab “General” and find the entry “Offset”. The upper 16 bits are the index and the lower 8 bits are the subIndex.



Note: Most registers are 32 Bit wide. Some registers are 64Bit wide (mainly positions). The concept of arrays is

- REAL or INT array indices {0, 1, 2, …} use subIndices = {1, 2, 3, ...}
- LREAL array indices {0, 1, 2, …} use subIndices = {1, 3, 5, ...}

3.2 Using the Triamec library

The Triamec library is a wrapper of the TwinCAT COE access functions.

Declaration

```
axis                        : use TE_AxisDirect or TE_AxisNci (or TE_AxisCnc of the sample code);
readReg                     : TE_RegisterReadInt32;
writeReg                    : TE_RegisterWriteInt32;
```

Configuration

```
axis.config.driveId         := 1004;                    // see adapter netId
axis.config.netId           := '192.168.10.99.2.1';     // see drive EtherCAT Addr
```

code to read an integer register

```
State0:
   readReg(axis:=axis, execute := FALSE);
State1:
   readReg.address  := 16#238701;
   readReg(axis:=axis, execute := TRUE);
   IF readReg.done THEN
      // result is available in readReg.value
      readReg(axis:=axis, execute := FALSE);
   END_IF
```

code to write an integer register

```
state0:
```

```
   write(axis:=axis, execute := FALSE);
State1:
   write.value      := 137;
   write.address := 16#238701;
   write(axis:=axis, execute := TRUE);
   IF write.done THEN
       write(axis:=axis, execute := FALSE);
   END_IF
```

3.3 Using standard TwinCAT functions

Add the library Tc2_EtherCAT and declare functions for read and write

```
netId        : STRING(23) := '192.168.10.99.2.1'; // see adapter netId
driveId      : UINT := 1001;       // see drive EtherCAT Addr
valueRd      : REAL;
valueWr      : LREAL := 1.2345678;
readReg      : FB_EcCoeSdoRead;
writeReg     : FB_EcCoeSdoWrite;
```

to read Application.Variables.Floats[1]

```
readReg.sNetId := netId;
readReg.nSlaveAddr := driveId;
readReg( nIndex:=16#2386, nSubIndex :=2, pDstBuf:= ADR(valueRd), cbBufLen:=SIZEOF(valueRd));
```

to write Application.Variables.Doubles[1]

```
writeReg.sNetId := netId;
writeReg.nSlaveAddr := driveId;
writeReg(nIndex:=16#2385, nSubIndex :=3, pSrcBuf:= ADR(valueWr),cbBufLen:=SIZEOF(valueWr));
```

Then set bExecute to FALSE and TRUE to start the transaction.

Please note that the type of the variables readReg and writeReg must correspond to the type of the Drive register (DINT, REAL, LREAL).