Axel Motion Control Library – Customizing Axis Driver

Sommario

Prerequisites

Axel Motion Control Library works with any recent PLC runtime, and a fieldbus enabled to motion control (currently EtherCAT).

It uses Object Oriented feature, so the runtime and compiler should support in and it should be enabled,

it uses references, and the compiler option "VAR_IN_OUT by reference" should be enabled.

Introduction

LogicLab Motion Control Library works on axis drivers, which have to be implemented for any drive and motor used with the library.

The driver model is based on CiA 402, or DS 402 specification model.

The MC Library mainly works on variables which are sent or received from fieldbus by means of PDO data exchange.

Some additional behavior can be customized in a driver function block running cyclically.

The customization of driver requires two steps:

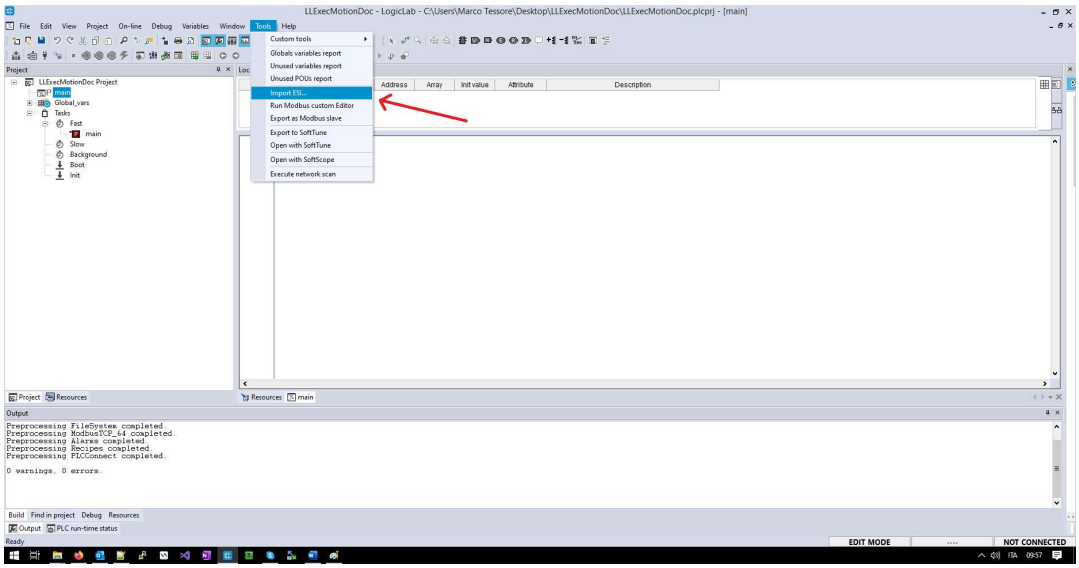
1) the customization of the fieldbus descriptor of the slave, actually only EtherCAT drives are supported.

2) the customization of a PLC function block representing the axis model, in PLC code.

Customization of the PCT descriptor

First step to enable an EtherCAT drive to the Motion library is to change its pct descriptor.

- Import the ESI file in the Catalog



- Find the related PCT in the folder C:\Program Files (x86)\Axel PC Tools\Catalog\EtherCATcustom

- Suppose your drive file is AKD_6A_00414B44_00000002.pct, you can copy it in AKD_6A_00414B44_00000002_Motion.pct, edit with a Text Editor and add the following sections:

```
	<?xml version="1.0" encoding="utf-8"?>
<devicetemplate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
	<deviceinfo caption="AKD_Motion_Motion" name="AKD_6A_00414B44"
version="2.0" icon="img\DRIVE.BMP"
deviceid="AKD_6A_00414B44_00000002_Motion" location="ECATVendor6A\Drive"
importedFromESI="true" ECATVendorID="106" ECATProductCode="4279108"
ECATRevisionNo="2">
		<description LcId="1033">AKD EtherCAT Drive (CoE) Motion
Enabled</description>
		<protocols>
			<protocol>EtherCAT_port_Y</protocol>
		</protocols>
		<groups>
			<group name="ECATVendor6A" icon="img\ECATVendor6A.BMP">
				<transl LcId="1033">Kollmorgen</transl>
			</group>
			<group name="Drive" icon="img\DRIVE.BMP"
SortOrder="520">
				<transl LcId="1033">Drives</transl>
			</group>
		</groups>
		<motionAxes>
			<axis name="axis1">
				<drivers>
					<driver>MC_Driver_KollmorgenAKD</driver>
				</drivers>
				<defaults roundSteps="1048576"/>
				<mappings ds402="false">
```

```
                        <mapping index="0x6040" subindex="0"
name="controlword"/>
                        <mapping index="0x6041" subindex="0"
name="statusword"/>
                        <mapping index="0x6060" subindex="0"
name="op_mode"/>
                        <mapping index="0x6061" subindex="0"
name="op_mode_display"/>
                        <mapping index="0x60C1" subindex="1"
name="target_position"/>
                        <mapping index="0x60FF" subindex="0"
name="target_velocity"/>
                        <mapping index="0x6071" subindex="0"
name="target_torque"/>
                        <mapping index="0x6063" subindex="0"
name="actual_position"/>
                        <mapping index="0x606C" subindex="0"
name="actual_velocity"/>
                        <mapping index="0x6077" subindex="0"
name="actual_torque"/>
                        <mapping index="0x603F" subindex="0"
name="errorcode"/>
                        <mapping index="0x60F4" subindex="0"
name="follow_err_actual"/>
                    </mappings>
                </axis>
            </motionAxes>
        </deviceinfo>
```

Note that in the previous example the following facts

- The whole section motionAxes is added to the imported PCT
- MC_Driver_KollmorgenAKD is the name of the driver function block to be written
- roundSteps field is the default value of encoder pulse for every round

The library will use the mapping information to map PDO input output variables directly into the fieldbus configuration.

Every mapping tag maps an object dictionary index/subindex with the default mnemonics in DS402 model

In the table the default mapping, if your drive have a standard DS402 mapping you can avoid inserting the mapping section, but you must specify `ds402="true"`

Type	Mnemonics	Dir	OD Index	OD subi	Description
UINT	controlword	Out	0x6040	0	Control word as described in CiA402
UINT	statusword	In	0x6041	0	Status word as described in CiA402
SINT	op_mode	Out	0x6060	0	Modes of operation
SINT	op_mode_display	In	0x6061	0	Modes of operation actual value
DINT	target_position	Out	0x607A	0	Target position, if position driven
DINT	target_velocity	Out	0x60FF	0	Target velocity, if velocity driven
INT	target_torque	Out	0x6071	0	Target torque, if torque driven
DINT	actual_position	In	0x6064	0	Actual position
DINT	actual_velocity	In	0x606C	0	Actual velocity
INT	actual_torque	In	0x6077	0	Actual torque

UINT	axis1_errorcode	In	0x603F	0	Error code
DINT	axis1_follow_err_actual	In	0x60F4	0	Position error

Another mapping example is:

```
<motionAxes>
    <axis name="axis1">
        <drivers>
            <driver>MC_Driver_Bonfiglioli</driver>
        </drivers>
        <mappings ds402="true"/>
        <defaults roundSteps="65536"/>
    </axis>
</motionAxes>
```

Note that the PCT reflects the ESI file of the axis drive,

NOTE: You may need to modify the ESI section in order to have PDO's already configured to map the required variables.

Multiple Axes devices

There exists EtherCAT slaves which control multiple axes,

they have mapped on PDOs the variable for each axis. The object, in the object dictionary are usually separated by a fixed offset, i.e. 0x800.

To manage that the field objectsIndexOffset have been introduced in the mappings tab.

An example of a multiple axes PCT is:

```
<motionAxes>
    <axis name="axis1">
        <drivers>
            <driver>MC_Driver_DS402</driver>
        </drivers>
        <mappings ds402="true" objectsIndexOffset="0x800"/>
        <defaults roundSteps="65536"/>
    </axis>
    <axis name="axis2">
        <drivers>
            <driver>MC_Driver_DS402</driver>
        </drivers>
        <mappings ds402="true" objectsIndexOffset="0x800"/>/>
        <defaults roundSteps="65536"/>
```

```
    </axis>
    <axis name="axis3">
        <drivers>
            <driver>MC_Driver_DS402</driver>
        </drivers>
        <mappings ds402="true" objectsIndexOffset="0x800"/>
        <defaults roundSteps="65536"/>
    </axis>
</motionAxes>
```

The index of the object to be mapped id calculated with the following formula:

effectiveIndex = normalIndex + objectIndexOffset * axisNumber

Customization of the driver function block

Every axis type needs a function block that represents its type of axis, the function block is responsible of:

- some measurement and state update
- execute long commands on guide of the library blocks (deferred commands)
- execute long commands in the background task on guide of the library blocks (background commands)
- react to movement errors with proper action, it could be an emergency halt ramp

Adopting an Object-Oriented approach, every driver function block must inherit from the base function block driver MC_Driver_DS402, in the library MC_Driver_DS402.

The library contains some driver examples.

The new driver could overwrite some methods to specialize or improve the behavior.

Method list of the MC_Driver_DS402 function block

Method	Task	Return value	Parameters	Should be overridden
Execute	cyclic	BOOL	AXIS_REF^	NO, entry point automatically called every cyclic execution
Background	bkg	BOOL	AXIS_REF^	NO, entry point automatically called every background execution
BackgroundCommandSM	bkg	BOOL	AXIS_REF^	NO, state machine of the background commands*

ComputeMotionStatus	cyclic	BOOL	AXIS_REF^	NO*, it computes motion status of the axis
DeferredCommandSM	cyclic	BOOL	AXIS_REF^	NO, state machine of the deferred commands*
SDOWrite	bkg	BOOL	Index, subindex, value, size	NO, utility command to write SDO to the slave
StartCmd_HomeSetParams	bkg	BOOL		
StartCmd_Homing	bkg	BOOL		
StartCmd_PowerOff	cyclic	BOOL		
StartCmd_PowerOn	cyclic	BOOL		
BackgroundProcessing_user	bkg	BOOL	AXIS_REF^	
CyclicProcessing_user	cyclic	BOOL	AXIS_REF^	
GetAxisReadyToMove	NA	BOOL		
OnMotionFailure	NA	BOOL		
OnMotionOutOfRange	NA	BOOL		

Fixed method list

Execute

It is the main method of the driver, it is automatically called every Fast cycle execution,

it initializes some global environment variables, calls the callback of the cyclic driver processing.

It computes motion status and calls DeferredCommandSM to perform the deferred commands.

Background

Similar to the Execute method, this is the background entry point to the services which are to be performed in the background.

BackgroundCommandSM

This manages the background state machine invoked by some blocks (actually the MC_Home block) *

ComputeMotionStatus

It computes the motion status of the axis.

It uses some filters, but it can be overwritten, although the default implementation should satisfy any situation.

DeferredCommandSM

This manages the timed state machine invoked by some blocks

SDOWrite

It is an utility function useful to write SDO on the axis for configuration purpose,

it cannot be called in a timed task and it is blocking.

Model of the SM callback - deferred or background services

When a block requires a deferred or background service, it signal the event on a field of the AXIS_REF of the axis.

The base implementation in the function block MC_Driver_DS402 invokes some callbacks, in the fast or background task, depending on the event type, to specify the Start of a service to be performed or periodic calls to subsequent evolution of the Action.

For example, when the engine requires the axis to be powered on, it first calls the callback method StartCmd_PowerOn and sequentially, every fast cycle, the method ActionCmd_PowerOn until the action is finished.

As an example let see the default implementation of the methods StartCmd_PowerOn and ActionCmd_PowerOn.

StartCmd_PowerOn:

```
          op_mode^ := TO_SINT(8); (* Sets the op_mode mapped variable to
the value 8 - Cyclic synchronous position mode *)
          controlword^ := 16#0; (* Sets zero to the control word *)
          driver^.di.command := MCD_CMD_BUSY; (* Signal the command is
busy - executing *)
          driver^.di.response := MCD_RESP_NONE; (* Signal no response -
command done or failed - is ready to be returned to blocks *)
          driver^.di.action := MCD_CMD_POWER_ON; (* Sets the next tyme
this action have to be called *)
          sm_step := 1; (* State machine step of the action *)
```

ActionCmd_PowerOn:

```
          // DS 402 transitions to power on
     CASE sm_step OF
     1:
          controlword^ := 16#80; // reset errors
          sm_step := 2;
     2:
          IF statusword^ MOD 16#100 = 16#50 THEN
               controlword^ := 6;
               sm_step := 3;
          END_IF;
     3:
          IF statusword^ MOD 16#100 = 16#31 THEN
               controlword^ := 7;
               target_position^ := actual_position^;
               sm_step := 4;
          END_IF;
     4:
          IF statusword^ MOD 16#100 = 16#33 THEN
               controlword^ := 16#F;
          END_IF;
          IF statusword^ MOD 16#100 = 16#37 THEN (* Power On procedure
done *)
               driver^.di.command := MCD_CMD_NONE;   (* Frees the
possibility for the engine to send another command *)
               driver^.di.response := MCD_RESP_POWER_ON; (* Send
response ok for the command to the engine *)
               driver^.di.action := MCD_CMD_NONE; (* No other action
callback should be called *)
               sm_step := 0;    (* Reset state machine step *)
          END_IF;
     END_CASE;
```

Lorem ipsum ecc…

Overwritable method list

StartCmd_HomeSetParams

Executed in background task -> starts the action of setting the homing parameters to drive.

StartCmd_Homing

Executed in background task -> starts the action of homing actual procedure.

StartCmd_PowerOff

Executed in fast task -> Starts the action of switching off the power of the axis.

StartCmd_PowerOn

Executed in fast task -> Starts the action of switching on the power of the axis.

ActionCmd_HomeSetParams

Executed in background task -> Set the DS402 Homing parameters by means of SDWrite method.

ActionCmd_Homing

Executed in background task -> Executes the DS402 Homing procedure

ActionCmd_PowerOff

Executed in fast task -> Switch the power off

ActionCmd_PowerOn

Executed in fast task -> Switch the power on

BackgroundProcessing_user

Empty callback method called every background cycle

CyclicProcessing_user

Empty callback method called every IO cycle

GetAxisReadyToMove

It could be overridden and is called from the library to specify conditions which inhibits the axis to move

OnMotionFailure

This callback is called every time there is an error on computation of motion blocks: it can be used to trigger an emergency stop of the axis.

OnMotionOutOfRange

This callback is called every time a movement violates the range constraints for an axis: it can be used to trigger an emergency stop of the axis.