Motion Control Products

Application note
AC500 - EtherCAT® Fast Position Capture

AN00221

Rev D (EN)

When using an AC500 PLC as an EtherCAT® master the position of remote EtherCAT drives can be captured via easy-to-use function blocks provided with the PS552-MC-E motion library

Introduction

AC500 PLCs (PM59x) can be used to perform real-time motion control of ABB motion AC servo drives via EtherCAT. This application note follows on from AN00205 (AC500 – EtherCAT Getting Started Guide) and details how to use Automation Builder to define the hardware and software setup suitable for capturing (latching) the position of an axis. Typical uses for position capture include applications requiring registration (e.g. printing, labelling, indexing conveyors) as well as accurate homing (please refer to AN00220 for further details on the use of position capture and homing).

Pre-requisites

You will need to have the following to work through this application note:

- Mint Workbench build 5860 or later (see www.abbmotion.com for latest downloads and support information)
- A MicroFlex e190 or MotiFlex e180 drive with build 5868.7 firmware or later with digital inputs 1and 2 wired (i.e. the fast position capture inputs) – note that the text in this application note assumes the use of a MicroFlex e190 but the process is the same for a MotiFlex e180
- A PC or laptop running Automation Builder v2.1.1 or later
- An installed and licensed copy of the latest version of the ABB PLCopen motion control library (PS552-MC-E, version 3.2.0 or later)
- An AC500 PM59x-ETH PLC with CM579-ECAT communication module (CM579-ECAT module must be running firmware version 2.6.9 or later, version 4.3.0.2 or later is preferable – contact your local ABB PLC support team for details on how to check this and update if necessary)
- Ethernet cable to connect the CM579-ECAT module to the EtherCAT servo drive
- A working knowledge of the basic operation of the AC500 PLC and MicroFlex e190 / MotiFlex e180 drive via EtherCAT – refer to application note AN00205 for further details if necessary

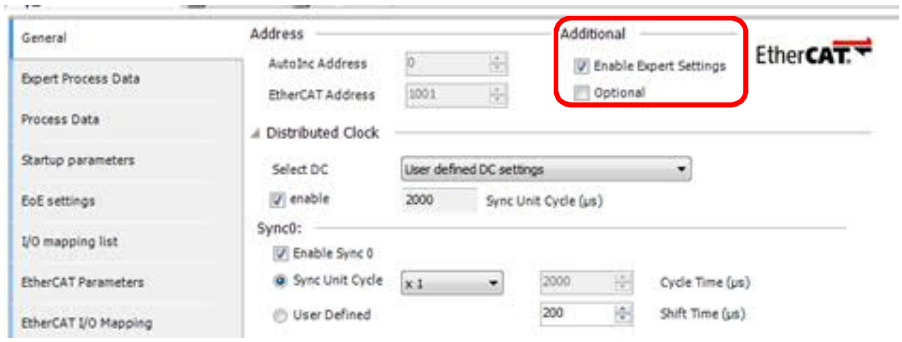
Drive set-up and ESI file

This application note assumes that you have already commissioned the ABB motion drive and that it is loaded with appropriate firmware (5868.7 or later). That is to say you have been through the commissioning wizard to define the motor and application settings and have then auto-tuned (and fine-tuned if necessary) the control loops for the drive. Details on commissioning the drive can be found in the relevant drive installation manual. It is also assumed that you have at least read and understood the content of application note AN00205. The text throughout this application note assumes that the project from AN00205 will be used as the starting point for new PLC code, however, for convenience a completed project archive is included with the download for this application note together with appropriate EtherCAT ESI/XML files should you need to install these into your Automation Builder device repository. The sample project provided with this application note includes a visualization to allow easier testing of the sample functions.

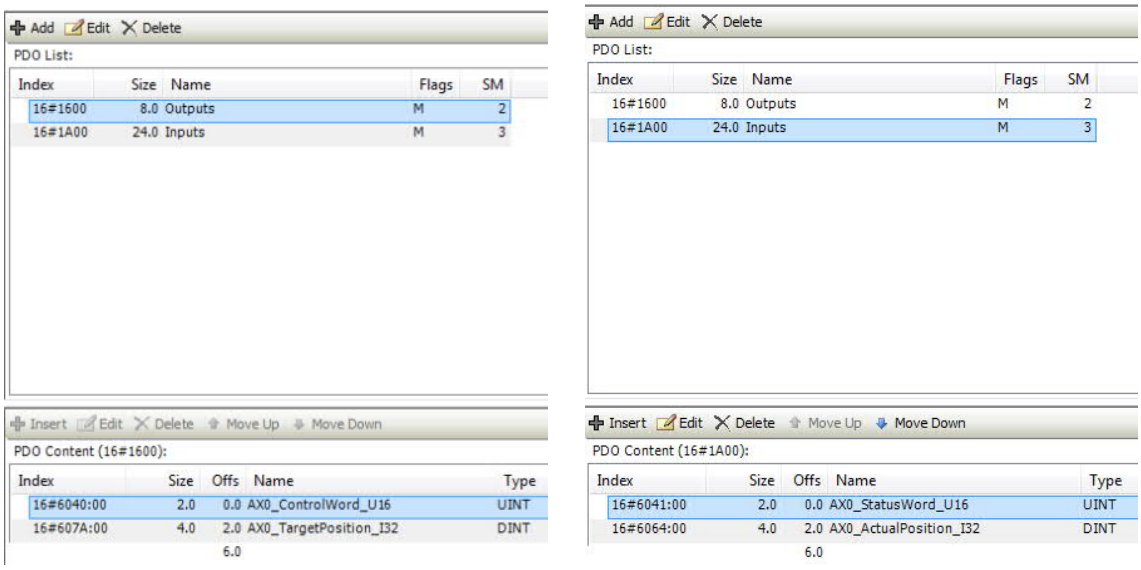
Adding Process Data Objects for Position Capture

Open the project from AN00220 if you wish to follow the text below to add the process data objects (PDOs) necessary to access the fast position capture data on the remote drive. Alternatively a pre-prepared project is available with the relevant code etc… already completed. Save the project with a new name, then double-click the drive icon and select the 'Slave' tab.

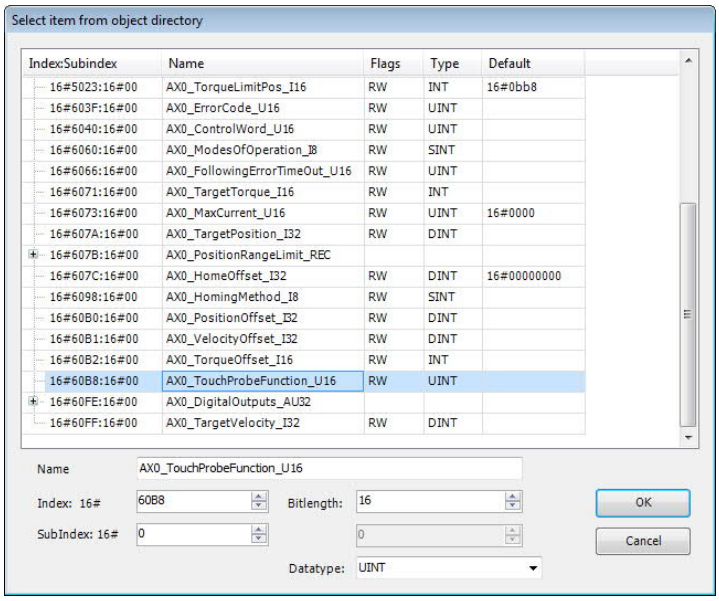
The fast position capture data is accessed via drive objects that are known as "Touchprobe" objects. We need to add some additional process data objects in order to use the touchprobe data so we must first select the 'Enable Expert Settings' checkbox as shown below…



Having selected this option Automation Builder provides an additional tab in the right pane named 'Expert Process Data' – select this tab. As you select between Outputs and Inputs in the right hand pane, Automation Builder shows the existing PDO mappings setup between the PLC and the drive…

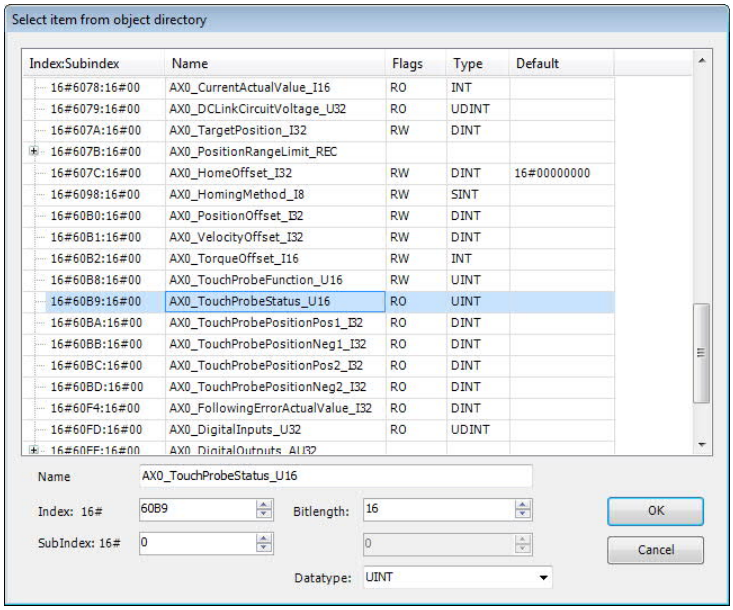


Note that these are from the PLC's perspective (i.e. Outputs are values that the PLC sends to the drive, Inputs are values that the PLC receives from the drive). Highlight Outputs and in the bottom right hand pane select '6.0' in the Offs column, right click and select 'Insert…'. Scroll through the list of available objects and select the DS402_TouchProbeFunction_U165 object (index 60B8)….



Click on OK and this object will be added to the list of Output PDO mappings. It is this object that allows the AC500 PLC to configure the operation of the drive so that either the fast inputs (1 and 2) or the Z pulse from the motor's encoder can be used to capture (latch) the axis position.

Now select Inputs in the top right pane and again select '6.0' in the Offs column, right click and select 'Insert…'. This time select the DS402_TouchProbeStatus_U16 object (index 60B9) and click on OK to add this to the Input PDO mappings. It is this object that allows the PLC to detect the status of the fast latches on the drive.



Now repeat the process to add another Input PDO mapping. This time however we need to select the "TouchProbePosition" objects. You will notice from the list that there are four possible objects…

- DS402_TouchProbePositionPos1_I32 (index 60BA)
- DS402_TouchProbePositionNeg1_I32 (index (60BB)
- DS402_TouchProbePositionPos2_I32 (index 60BC)
- DS402_TouchProbePositionNeg2_I32 (index 60BD)

Objects relating to POS1 and NEG1 are directly related to fast latch values captured by digital input 1 (or the encoder Z pulse) on the drive.
Objects relating to POS2 and NEG2 are directly related to fast latch values captured by digital input 2 (or the encoder Z pulse) on the drive.

(POS relates to rising edge latch data and NEG relates to falling edge latch data).

We would usually only select the objects we need for the application to keep the amount of mapped data to a minimum (e.g. for a registration application we might just want to use the rising edge of a registration sensor wired to digital input 1 on the drive so we'd select DS402_TouchProbePositionPos1_I32). But for this example we'll add all of the objects so repeat the 'Insert…' process until all four objects have been included. Once these are added the Input PDO mappings should look something like this (the order isn't important)…

PDO Content (16#1A00):

Index	Size	Offs	Name	Type
16#6041:00	2.0	0.0	AX0_StatusWord_U16	UINT
16#6064:00	4.0	2.0	AX0_ActualPosition_I32	DINT
16#60B9:00	2.0	6.0	AX0_TouchProbeStatus_U16	UINT
16#60BA:00	4.0	8.0	AX0_TouchProbePositionPos1_I32	DINT
16#60BB:00	4.0	12.0	AX0_TouchProbePositionNeg1_I32	DINT
16#60BC:00	4.0	16.0	AX0_TouchProbePositionPos2_I32	DINT
16#60BD:00	4.0	20.0	AX0_TouchProbePositionNeg2_I32	DINT
		24.0		

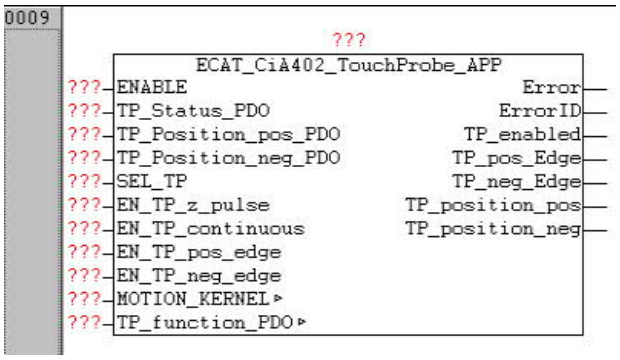
Now select the EtherCAT I/O Mapping tab in Automation Builder. We need to assign variable names to the new PDO mappings we've added.
The image below shows the variables we assigned to our new touchprobe objects…

Tool Bar

Object Name	Variable	Channel	Address	Type	Description
MicroFlex_e150	wAxis0TPFunction	AX0_TouchProbeFunction_U16	%QW1.4	UINT	AX0_TouchProbeFunction_U16
MicroFlex_e150	wAxis0TPStatus	AX0_TouchProbeStatus_U16	%IW1.4	UINT	AX0_TouchProbeStatus_U16
MicroFlex_e150	diAxis0TPPos1	AX0_TouchProbePositionPos1_I32	%ID1.3	DINT	AX0_TouchProbePositionPos1_I32
MicroFlex_e150	diAxis0TPNeg1	AX0_TouchProbePositionNeg1_I32	%ID1.4	DINT	AX0_TouchProbePositionNeg1_I32
MicroFlex_e150	diAxis0TPPos2	AX0_TouchProbePositionPos2_I32	%ID1.5	DINT	AX0_TouchProbePositionPos2_I32
MicroFlex_e150	diAxis0TPNeg2	AX0_TouchProbePositionNeg2_I32	%ID1.6	DINT	AX0_TouchProbePositionNeg2_I32
MicroFlex_e150	wAxis0ControlWord	AX0_ControlWord_U16	%QW1.0	UINT	AX0_ControlWord_U16
MicroFlex_e150	diAxis0TargetPos	AX0_TargetPosition_I32	%QD1.1	DINT	AX0_TargetPosition_I32
MicroFlex_e150	wAxis0StatusWord	AX0_StatusWord_U16	%IW1.0	UINT	AX0_StatusWord_U16
MicroFlex_e150	diAxis0ActualPos	AX0_ActualPosition_I32	%ID1.1	DINT	AX0_ActualPosition_I32

Save the project again and now launch Codesys by double-clicking the program icon in Automation Builder. Click on 'Update' to accept the changes to the hardware configuration.

Select the last rung in the program, right click and select 'Network (after)' to add a new rung to the program. Click the 'New box' toolbar button and enter the name ECAT_CiA402_TouchProbe_APP for the new block (or use the Input Assistant via F2 on the keyboard to find and select this function block)…



This function block allows the PLC to configure the operation of a fast interrupt (touchprobe) on the drive.
The function block inputs have the following functionality:

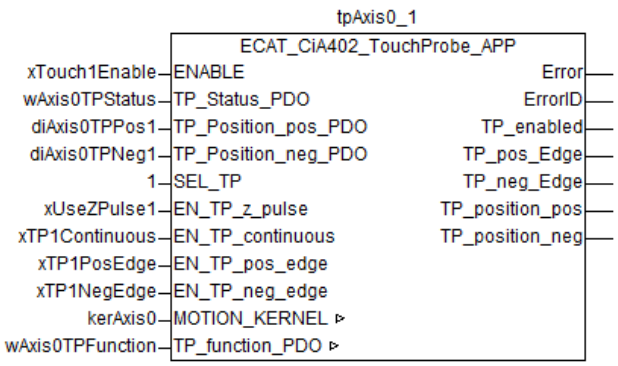
Enable	Set this to TRUE to enable the touchprobe (latching) on the remote drive.
TP_Status_PDO	This must be linked to the variable assigned to the touchprobe status object (wAxis0TPStatus in our case)
TP_Position_pos_PDO	This must be linked to the variable assigned to the rising edge latch for the touchprobe selected by the SEL_TP input (e.g. diAxis0TPPos1 if SEL_TP is set to 1)
TP_Position_neg_PDO	This must be linked to the variable assigned to the falling edge latch for the touchprobe selected by the SEL_TP input (e.g. diAxis0TPNeg1 if SEL_TP is set to 1)
SEL_TP	Selects whether the block is controlling touchprobe 1 or touchprobe 2 on the drive
EN_TP_z_pulse	Selects whether the latch is to be captured by the encoder's z pulse. If set to FALSE the drive will latch position according to the settings defined for the associated digital input (e.g. Touchprobe 1 will latch from digital input 1)
EN_TP_continuous	Selects whether the drive should continue to latch position every time the fast input (or z pulse, depending on configuration) occurs. If set to FALSE the drive will latch one value and will not latch again until the touchprobe is disabled and then re-enabled
EN_TP_pos_edge	Selects whether a rising edge of the relevant fast input (or z pulse, depending on configuration) is to be used to store a value in the TP_position_pos function block output
EN_TP_neg_edge	Selects whether a falling edge of the relevant fast input (or z pulse, depending on configuration) is to be used to store a value in the TP_position_neg function block output
MOTION_KERNEL	This must be linked to the instance of the CMC_MOTION_KERNEL_REAL function block used by the drive (e.g. kerAxis0)
TP_Function_PDO	This must be linked to the variable assigned to the touchprobe function PDO (e.g. wAxis0TPFunction)

Outputs from the Touchprobe function block indicate when valid latches have occurred (e.g. TP_pos_Edge) and the value of the fast latched position (e.g. TP_position_pos). These outputs would then be used as required by the PLC application code (e.g. to compare the captured position against an expected value to calculate a registration correction).

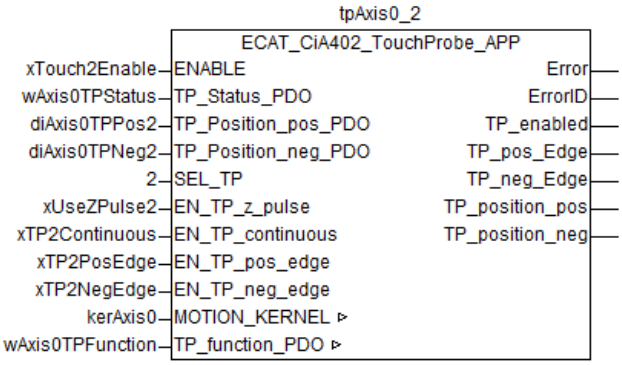
IMPORTANT
The touchprobe function block must be included in a program that is synchronised to the EtherCAT data cycle. Our example PLC project uses just a single program object, called by the task associated with the EtherCAT cycle so this requirement has been met. In most applications the main application logic is likely to be coded in a separate cyclic PLC task (and the motion kernel, parameter and DS402 function blocks will be in an EtherCAT related task/program), so it's important to remember to include the touchprobe function block in the same program element as the kernel function block.
The outputs 'TP_pos_edge' and 'TP_neg_edge' that indicate reception of a new value are only set true for one EtherCAT cycle when the touchprobe function block is operating in 'Continuous' mode and therefore these states will be missed if the function block is outside of the EtherCAT related processing.
Also note that the latched values themselves 'TP_position_pos' and 'TP_position_neg' will be modified if the axis position is adjusted (e.g. via MC_SetPosition) so if the application needs to actually latch/store these values then code should be added to transfer these values to other program variables (or an array if a queue of data needs to be stored).

For this application note we will assign variables to all of the configuration inputs which we can then force via CoDeSys to test the various latching options (the example project included with this application note includes a visualisation to assist with the test process).

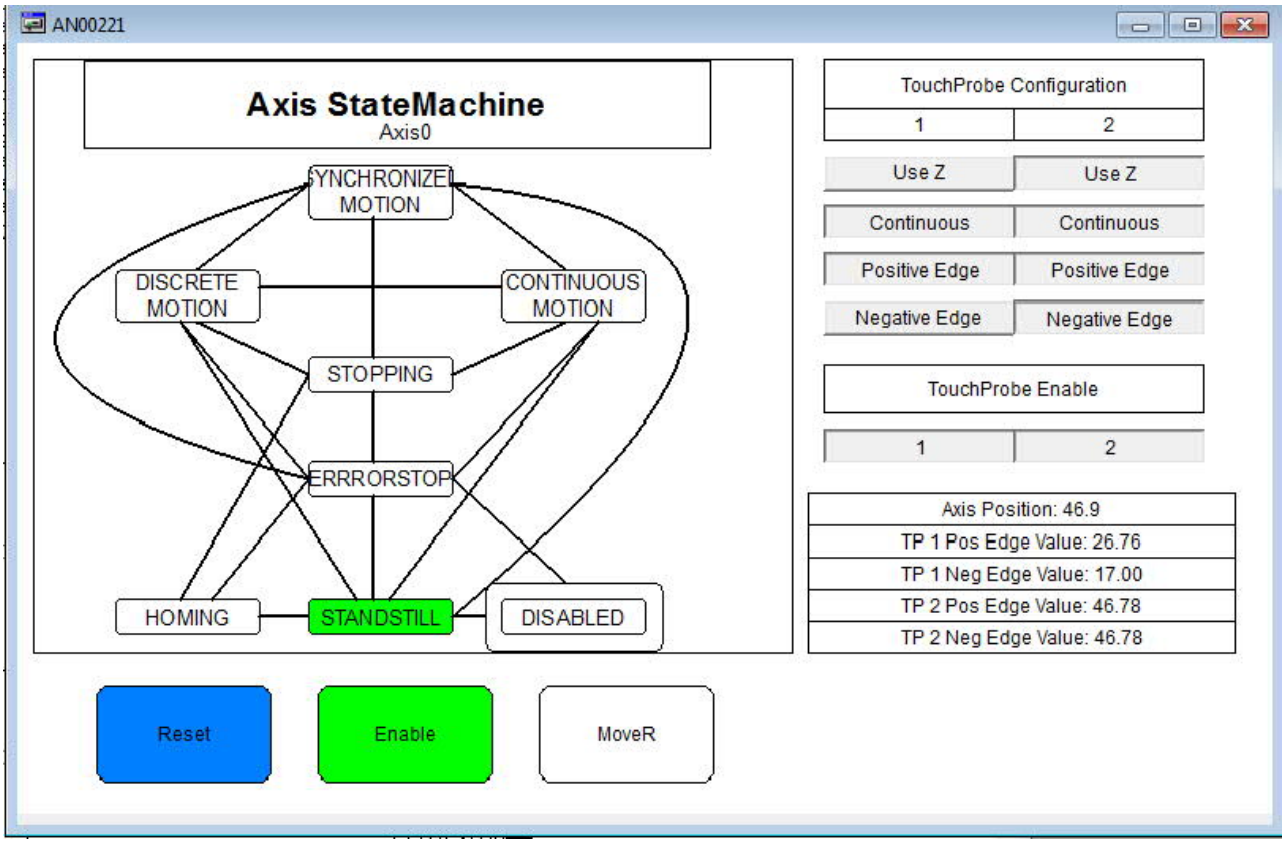
The image below shows our final settings for the Touchprobe configuration function block for the block we named tpAxis0_1…



Now right click this rung add a new network (after). Click on the 'New box' button and for the new block enter ECAT_CiA402_TouchProbe_APP again. Enter parameters/variables for tpAxis0_2 as shown below:



This is all the code required to test the operation of the drive's touchprobes. Login to the PLC and download this code to test it. Use CoDeSys to force the various configuration and enabling parameters and to view the outputs of the two touchprobe function blocks. Alternatively the project included with this application note includes a visualisation to allow the various touchprobe modes to be tested – remember that to test the modes associated with the fast inputs it will be necessary to wire these digital inputs on the ABB motion drive.



It is also possible to use the touchprobe objects to allow very accurate homing to be performed – please refer to application note AN00220 for further details.

Contact Us

For more information please contact your
local ABB representative or one of the following:

new.abb.com/motion
new.abb.com/drives
new.abb.com/drives/drivespartners
new.abb.com/PLC

EtherCAT® is a registered trademark and patented technology, licenced by Beckhoff Automation GmbH, Germany