Motion Control Products

Application note
AC500 - EtherCAT® homing methods

AN00220

Rev E (EN)

The ABB AC500 PLC provides the flexibility to profile homing on the PLC or the ABB motion AC servo drive itself depending on the application requirements

Introduction

AC500 PLCs (PM585 and PM59x) can be used to perform real-time motion control of ABB's EtherCAT enabled servo motion drives. 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 various EtherCAT axis homing methods.

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 drive with build 5868.7 firmware or later (or the files are easily adapted to suit a MotiFlex e180 with build 5868.7 or later too)
- A PC or laptop running Automation Builder 2.1.1 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 PM585 or PM59x-ETH PLC with CM579-ECAT communication module (CM579-ECAT module must be running firmware version 4.3.0.2 or later – contact your local ABB PLC support team for details on how to check this and update if necessary) or a PM595 PLC with integrated EtherCAT coupler
- Straight-through Ethernet cable (patch cable) to connect the CM579-ECAT module to the drive (as defined by the EtherCAT standard, although these products will also operate correctly with cross-over Ethernet cable)
- A working knowledge of the basic operation of the AC500 PLC and ABB motion drives via EtherCAT – refer to application note AN00205 for further details if necessary

Ideally you will also need:

- A digital input module of some type for the AC500 PLC (we used a DC523 module for our example project)

But this is not critical as it is possible to simulate the digital inputs via use of variables; this is described later in this document.

Drive set-up

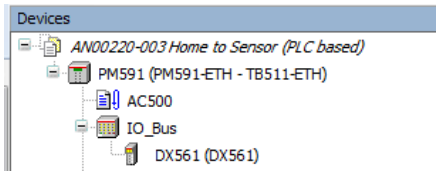
This application note assumes that you have already commissioned an ABB motion drive that it is loaded with appropriate firmware. That is to say you have been through the commissioning wizard to define the motor and application settings and have then auto-tuned (and fine-tuned if necessary) the control loops for the drive. Details on commissioning the drive can be found in the drive installation manual. It is also assumed that you have at least read and understood the content of application note AN00205.

For convenience example projects (all named in the format “AN00220 [homing type].project”) are included with this application note that utilize a MicroFlex e190 drive using the EtherCAT ESI/XML file suitable for 5868.7 firmware. ESI/XML files for 5868.7 firmware for all ABB motion drives are included with this application note and can be installed into Automation Builder’s device repository if needed. The sample projects provided with this application note include visualizations to allow easier testing of the sample functions.

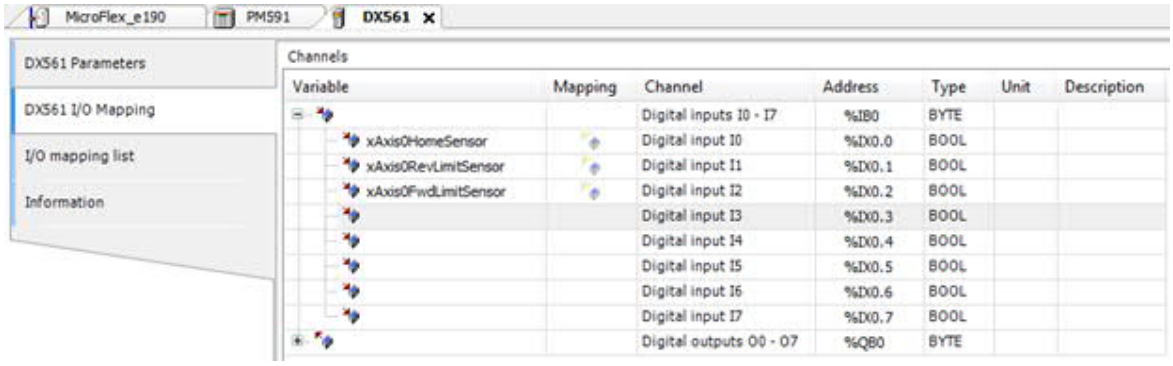
PLC profiled homing methods

In cases where the PLC is profiling the homing motion the home sensor status needs to be detected by the PLC itself. It is possible to wire the sensor to the drive and then read the state of this input either via remote object access (SDO) or by adding a PDO mapping to the drive inputs, but the simplest method is to wire the home sensor directly to a digital input in the PLC rack.

For our example ‘Home to Sensor’ project we added a DX561IO module to our AC500 PLC system to accept the home sensor input…



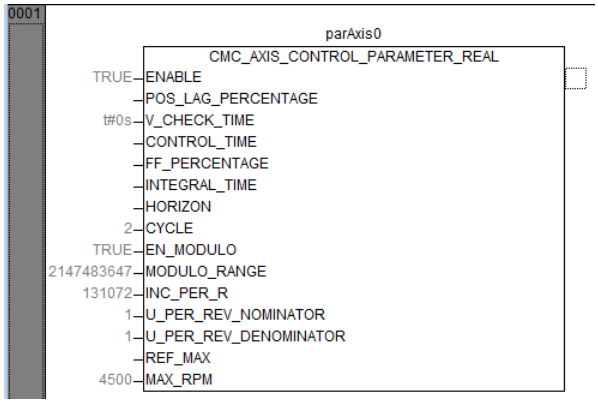
We then assigned a name to the input on our module to be used as the home input for our EtherCAT axis (Input 0). Because the PLCopen homing methods are able to automatically reverse off of limit sensors we also added variable names for forward and reverse limit sensors so we can test the functionality of this if needed…



To examine the example code, launch Codesys by double-clicking the program icon in the Automation Builder device tree (labelled AC500). Click on the Update button to acknowledge that the hardware configuration has changed and needs rebuilding if asked.

Once Codesys opens, navigate to the POU tab if necessary and open the PLC program (prgMotion). This is our simple example code and this single program is called by the EtherCAT related task. We will now detail some of the main elements of this example code…

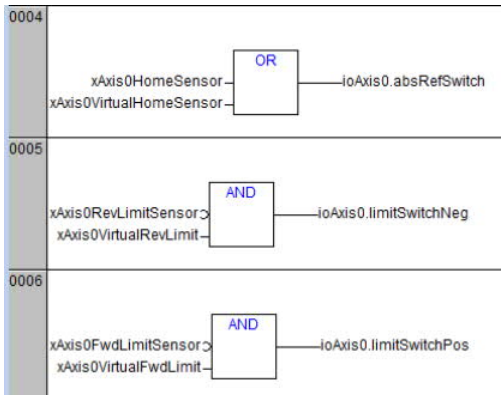
Scaling
For all our example projects the axis is scaled into units of “motor revolutions” via the CMC_AXIS_CONTROL_PARAMETER_REAL function block in our program…



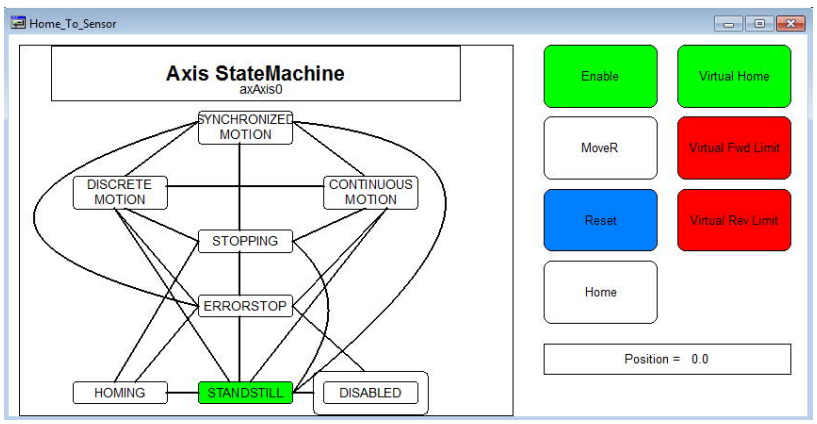
Our project was designed around a MicroFlex e190 demokit that uses a servo motor with a Smartabs encoder with 131072 counts per revolution. By setting the two U_PER_REV parameters to 1 this scales the axis into motor revolutions.

Home and Limit sensors
Code must be included within prgMotion to continually update the Axis IO structure with the current status of the Axis home and limit switches. The input states will be updated on every EtherCAT cycle so the accuracy of the final home position will depend on the configured EtherCAT cycle time and other factors such as the reaction time of the PLC inputs themselves, the speed reached during homing and the configured deceleration. To maximise the accuracy of homing to a sensor it is necessary to wire the sensor to the drive directly and use a drive based homing method as described later in this document, but for some applications using the PLC to profile the home may be advantageous (e.g. where there is a limited number of drive inputs and those that are available need to be used for other functions).

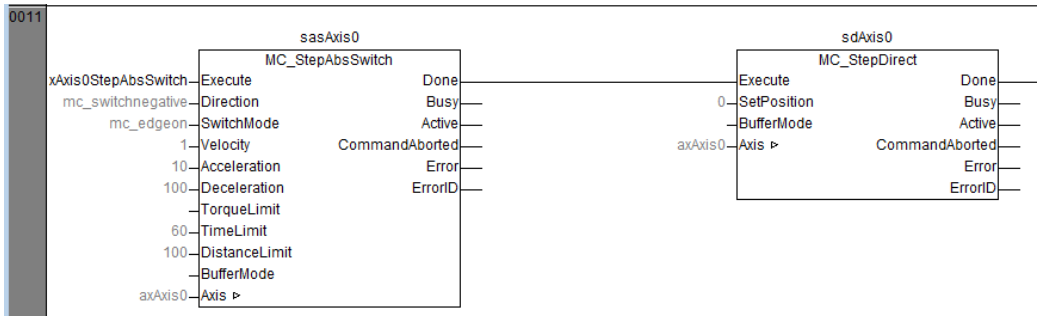
The example code comes complete with a visualisation to allow the home sensor and limit inputs to be activated ‘virtually’ in case it is not practical to wire inputs to the PLC being used for testing. These ‘virtual inputs’ are coded in parallel with the actual physical inputs in our example program…



The visualisations include buttons to activate and de-activate the virtual inputs where required (the buttons are red if the corresponding input is off or green if the corresponding input is on)…



Home to sensor
The 'Home to Sensor' project includes a MC_StepAbsSwitch function block that is used to execute a home to sensor operation. The 'Home' button in the visualisation is linked to the Execute input of our MC_StepAbsSwitch instance (sasAxis0)…



It is recommended that the reader downloads Part 5 (Homing Extensions) of the PLCopen motion control standard for further reference. This is available, together with the other parts of the PLCopen motion standard relating to motion control, at…

http://www.plcopen.org/pages/tc2_motion_control/.

The example code above gives a result that is similar to the Mint Home command HOME(axis) = _hmNEGATIVE_SWITCH but without a backoff operation for readers who are familiar with ABB's range of Mint based motion control products.

Note that currently the TorqueLimit input parameter is not supported and should be left blank.

If the user operates the forward and reverse limit switches during the homing process, then the PLC will automatically change direction if required as defined by the PLCopen motion standard.

On completion of the MC_StepAbsSwitch function block (and many of the other homing blocks) the axis remains in the homing state. Therefore further motion (e.g. MC_MoveRelative) is not possible until this state is left. This can be achieved in a number of ways…

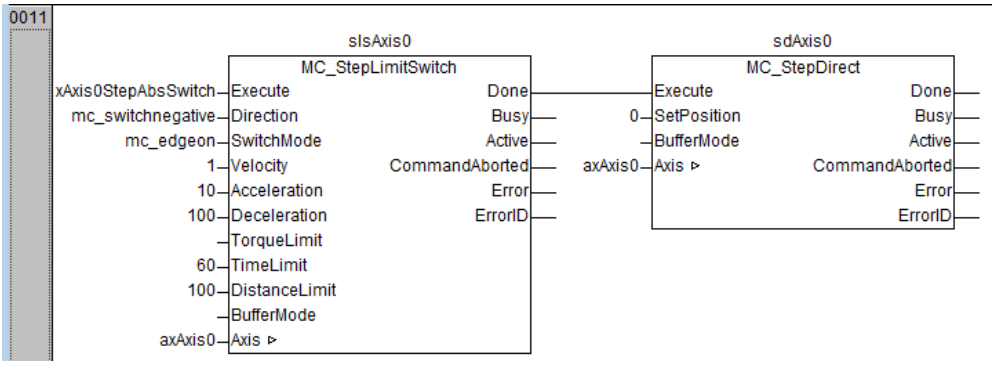
1. By disabling the axis by disabling the MC_Power block for the axis
2. By issuing the MC_Halt command
3. By issuing the MC_FinishHoming command (not currently supported by the ABB motion library)
4. By issuing the MC_StepDirect command

As it is typical to set an axis position at the end of the homing sequence (e.g. a position of zero) it is most common to use the MC_StepDirect block. As can be seen from the previous screenshot, this has been added to the same rung and triggered by the DONE output of the MC_StepAbsSwitch block.

MC_StepDirect can only be used if the axis is already in the homing state (as it would be if MC_StepAbsSwitch has been issued). If the user just wants to set the axis position normally (i.e. outside of a homing process) use MC_SetPosition if all that is required is the PLC's version of axis position needs setting. If the application requires that the position on the drive itself needs setting too (e.g. maybe there are some SENTINEL type operations being performed on the drive that are based on the drive's actual position) then a drive based 'Home on current position' should be performed instead of using one of the PLC based homing methods – this is described later in this document.

Note that a more detailed solution would require the program to examine the Error output of the blocks to detect homing failure (possibly to allow a subsequent call to the MC_Reset block to clear the fault), but this example at least shows the basic logic required to achieve a home to a sensor. Ensure that both limit switches are not simultaneously active at any point, this will result in the drive entering the error state – the error can be cleared by processing the MC_Reset block (the sample visualisations provide a Reset button to execute this if required).

Home to limit
Just as the axis can be homed to a dedicated home sensor it is also possible to home the axis to one of the defined limit switch inputs. The ABB PS552-MC library provides a PLCopen motion homing method for this…..MC_StepLimitSwitch. The 'Home to limit' project included with this application note duplicates many of the features of the 'home to sensor' example (e.g. the axis is still scaled in motor revolutions and the visualisation provides identical features), but the image below shows how the 'Home to limit' project code uses MC_StepLimitSwitch in place of MC_StepAbsSwitch…



The example is hard coded to run the axis in the negative direction (the Direction input of the function block is set to mc_switchnegative) and so the home completes when the negative limit switch is activated.

Again the reader is encouraged to refer to Part 5 (Homing Extensions) of the PLCopen motion control standard for further reference.

Homing to a drive touchprobe (fast input or z pulse)
The previously described methods may be adequate for some applications but the accuracy of the final home position depends to some extent on the EtherCAT cycle time, the reaction time of the PLC digital inputs, the defined deceleration rate and the speed of the axis at the time the sensor level or sensor edge is detected. To allow very accurate homing, methods are provided to allow the axis to be homed using the touchprobe objects (i.e. the fast interrupts) on the drive.

In this case the home sensor must be wired to one of the fast inputs on the ABB motion drive (either input 1 or input 2) or the z pulse from the motor's encoder may be used if it has one (in which case there is no need to wire either of the fast inputs – the drive will latch axis position automatically when the z pulse is detected).

Again an example project is provided with this application note to illustrate this homing method. Because this method makes use of a touchprobe object on the drive the Automation Builder configuration for the axis must now include PDO mappings for the touchprobe function, status and one of the touchprobe position 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 on the drive and are associated with touchprobe 1.
Objects relating to Pos2 and Neg2 are directly related to fast latch values captured by digital input 2 on the drive and are associated with touchprobe 2.
(Pos relates to rising edge latch data and Neg relates to falling edge latch data).

If using a Z pulse from the motor's encoder to latch axis position it doesn't matter which touchprobe is used, the Z channel is always set in the drive as channel 0.

Depending on the nature of the home sensor (i.e. whether it activates on a rising or a falling edge) and which input on the drive is used (i.e. input 1 or input 2) select the appropriate object (for our example we will use Input 1 on our drive and assume it is triggered on a rising edge so we will select DS402_TouchProbePositionPos1_I32.

In our example project for homing to a touchprobe you will see the required drive PDO mappings in Automation Builder…

PDO Content (16#1600):

Index	Size	Offs	Name	Type
16#6040:00	2.0	0.0	DS402_ControlWord_U16	UINT
16#607A:00	4.0	2.0	DS402_TargetPosition_I32	DINT
16#60B8:00	2.0	6.0	DS402_TouchProbeFunction_U16	UINT
		8.0		

PDO Content (16#1A00):

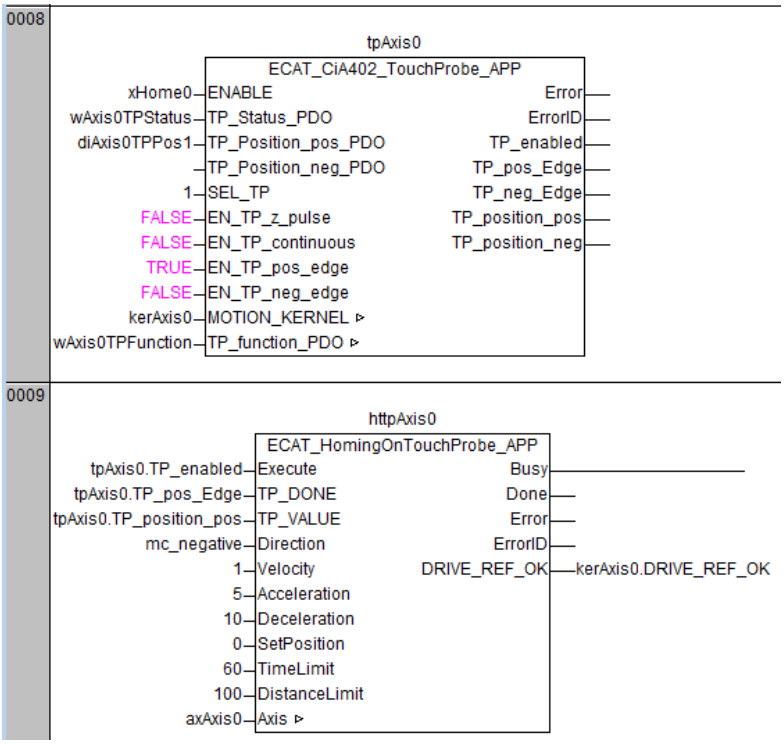
Index	Size	Offs	Name	Type
16#6041:00	2.0	0.0	DS402_StatusWord_U16	UINT
16#6064:00	4.0	2.0	DS402_ActualPosition_I32	DINT
16#60B9:00	2.0	6.0	DS402_TouchProbeStatus_U16	UINT
16#60BA:00	4.0	8.0	DS402_TouchProbePositionPos1_I32	DINT
		12.0		

The EtherCAT I/O mapping tab in the Automation Builder configuration page for the drive shows the variable names that have been assigned to these PDO mappings…

Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
wAxis0ControlWord		DS402_ControlWord_U16	%QW1.0	UINT		DS402_ControlWord_U16
diAxis0TargetPos		DS402_TargetPosition_I32	%QD1.1	DINT		DS402_TargetPosition_I32
wAxis0TPFunction		DS402_TouchProbeFunction_U16	%QW1.4	UINT		DS402_TouchProbeFunction_U16
wAxis0StatusWord		DS402_StatusWord_U16	%IW1.0	UINT		DS402_StatusWord_U16
diAxis0ActualPos		DS402_ActualPosition_I32	%ID1.1	DINT		DS402_ActualPosition_I32
wAxis0TPStatus		DS402_TouchProbeStatus_U16	%IW1.4	UINT		DS402_TouchProbeStatus_U16
diAxis0TPPos1		DS402_TouchProbePositionPos1_I32	%ID1.3	DINT		DS402_TouchProbePositionPos1_I32

The application code utilises an ABB specific motion function block called ECAT_HomingOnTouchProbe_APP that works in conjunction with the ABB specific touchprobe function block called ECAT_CiA402_TouchProbe_APP. The two PLC networks using these blocks are shown below…



IMPORTANT : The touchprobe and homing blocks should always be in a program element called by the EtherCAT related task

The touchprobe block is configured in our example to use touchprobe 1 (SEL_TP = 1). Because EN_TP_z_pulse is set to FALSE this configures the drive to use digital input 1 as the latching mechanism instead of the feedback encoder's z pulse (as we are using a BSM60R-240MT motor that has a Smartabs encoder there is no z pulse available anyway so we can only use a digital input to latch the axis position).

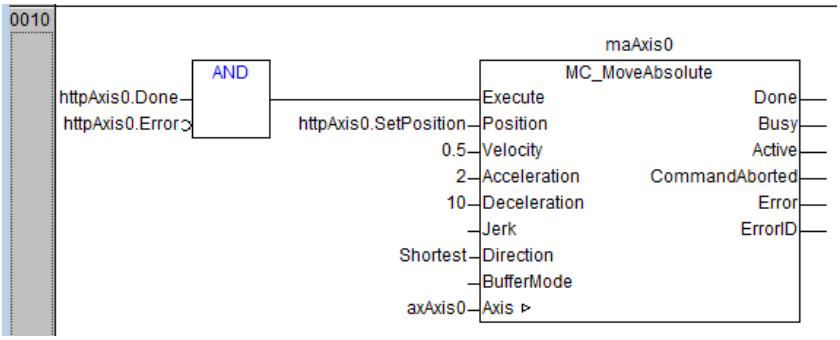
The homing block is triggered when the touchprobe block enables (which in turn is triggered by activation of the 'Home' button in our example visualisation). The velocity, acceleration, deceleration and direction parameters for the homing block determine how the PLC profiles the initial motion on the axis. This motion stops when the TP_DONE input on the homing block activates. This input is connected to the TP_pos_Edge output of the touchprobe block (we use TP_pos_Edge in this case because we configured the touchprobe block to operate from a rising edge via the EN_TP_pos_edge input parameter being set to TRUE).

The SetPosition input parameter on the homing block tells the PLC what position the axis should be considered to be at when the touchprobe occurred (so for example, if the SetPosition is zero, the touchprobe occurs and the axis has come to a halt two units later than the recorded TP_VALUE (linked to the TP_position_pos output of the touchprobe function block) then the new axis position is set to two units.

i.e. On completion of the homing function block the axis position is always how far past the latch position the axis has travelled.

In our example program we then use a MC_MoveAbsolute function block to move the axis back to our SetPosition value (zero in the case of this example). Effectively this moves the axis back to the exact point that the latch occurred at. This is very accurate as the only inaccuracy is caused by the latching latency in the servo drive (which is 1 microsecond at worst case).

The MC_MoveAbsolute is triggered by successful completion of the homing function block (the homing function block includes time and distance limit input parameters so the homing block will produce an error if either of these limits is exceeded)…



Note that in our example our SetPosition is zero. As we have used a modulo axis (because our axis could in theory run forever in one direction and eventually wrap at the 32 bit position boundary) then "jitter" on the enabled axis at position zero will result in the measured axis position fluctuating between zero and the maximum wrap position. We therefore use "Shortest" as the Direction input for the MC_MoveAbsolute to ensure the axis always travels the shortest distance to the required position. If we didn't specify this Direction type the PLC may profile motion from our maximum position all the way back to zero!

For applications requiring an initial home to a sensor, followed by a search for the index pulse from the motor's encoder that need to be profiled by the PLC, the PLC program should use a combination of the MC_StepAbsSwitch and the ECAT_HomingOnTouchProbe_APP blocks we have illustrated in this application note.

Homing methods profiled by the drive

In cases where the drive is profiling the homing motion the home sensor status needs to be detected by the drive itself. If forward and reverse limits are used in the application it is necessary to wire these to the drive also (e.g. so the homing routine can back off of these automatically as needed). The application may also require the status of the forward and reverse limit inputs to be detected by the PLC (e.g. to interrupt non-homing type motion). In these cases it is possible to wire the limit sensors to the drive and then read the state of these inputs either via remote object access (SDO) or by adding a PDO mapping to DIP_InState_AU32 (object 16#4020 subindex 01). The PLC code can then assign the values read from the drive to its local axis IO structure as was done previously with the earlier examples.

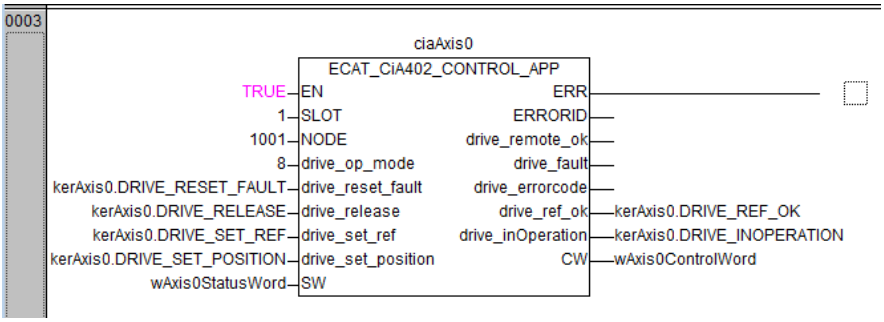
For our example project we configured input 3 as the home input on our drive (HOMEINPUT = 3), input 4 as the forward limit input (LIMITFORWARDINPUT = 4) and input 5 as the reverse input (LIMITREVERSEINPUT = 5) using Mint Workbench to configure these settings. It is possible to use SDO access from the PLC to configure these parameters too – please refer to application note AN00242 for further details.

To allow drive based homing to operate it is necessary to include an additional PDO mapping from the drive back to the PLC. The AX0_ModesOfOperationDisplay_I8 object (16#6061 subindex 00) is included as an additional output PDO as shown in our example project…

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#6061:00	1.0	6.0	AX0_ModesOfOperationDisplay_I8	SINT

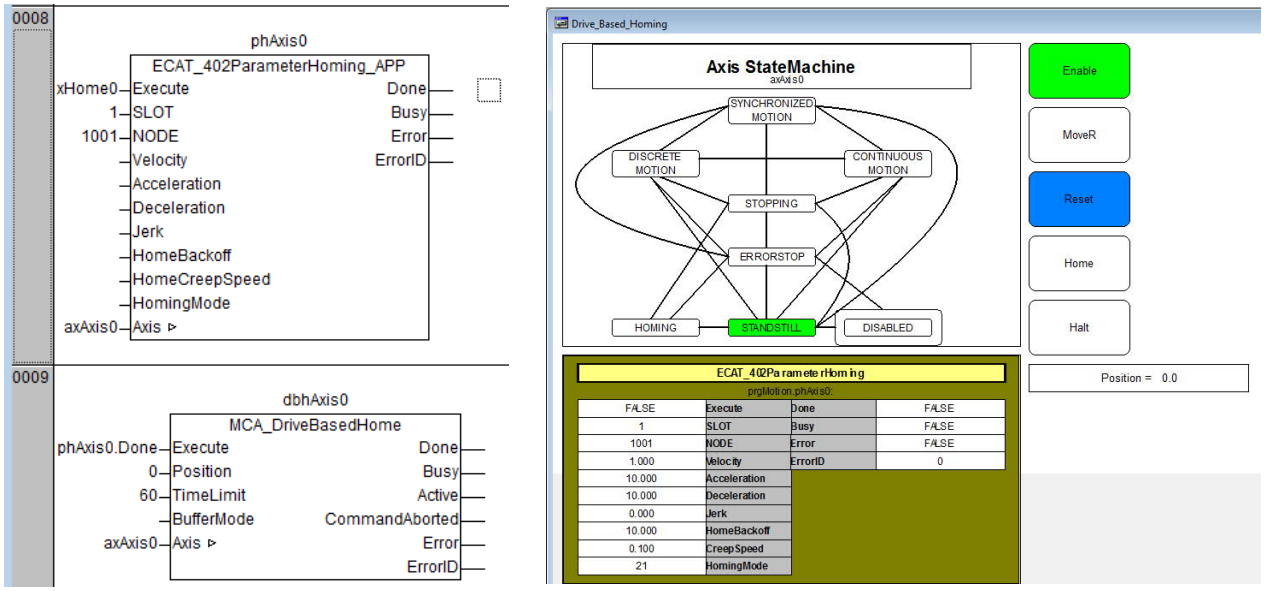
As the PLC needs to be able to switch the drive between its startup operating mode (e.g. Cyclic Synchronous Position, which is mode 8) and homing mode it becomes necessary to enter the startup mode as the drive_op_mode input parameter to the ECAT_CIA40_CONTROL_APP block as well as also linking the drive_set_ref and drive_set_position inputs to the associated Kernel outputs as shown below…



Two new function blocks are provided with version 3.2.0 onwards of the PS552-MC PLCopen motion library…

- ECAT_402ParameterHoming_APP : This function block allows the PLC to write to the drive's homing parameters (via SDO) allowing parameters such as home speed, home acceleration and deceleration etc… as well as the DS402 homing method to be used by the drive. Note that the drive's local PROFILEMODE parameter must be set for s-ramp in order for the Jerk parameter to take effect, otherwise the homing will use trapezoidal profiling
- MCA_DriveBasedHome : This function block initiates the home sequence on the drive and sets the required position the drive should set at the end of homing (which the PLC will then also consider to be the axis position). Provision is made to specify a time limit for the homing procedure to complete (in seconds). If the drive does not home within this period of time motion is aborted and the function block reports an error

The use of these two blocks is illustrated by the example project included with this application note…



The sample visualization allows the user to setup the homing parameters. There is no need to set the Execute input in the visualization, clicking the "Home" button will cause the homing parameters to be written to the drive and when this completes successfully the homing sequence will be triggered (the drive seven segment display will change to "h" to indicate homing is in progress and will change back to indicate the startup operating mode when homing is completed – e.g. "P" to indicate Cyclic Synchronous Position mode).

Notes:

1. For any of the drive based homing methods to work the drive/axis must be in the enabled state
2. All homing parameters are scaled by the PLC according to the axis position scale factor (even though the drive itself has separate scale factors for position, speed and acceleration). This must be taken into consideration when setting the homing parameters if the drive is configured to use different scale factors for these properties (e.g. mm for position but rpm for speed rather than mm/s)

Please refer to the following section for full details on the supported drive based homing methods.

Summary of available homing methods

PLC based homing methods		
Home type	*Description*	*Notes*
MC_StepAbsRefSwitch	Home to a home sensor	No support for torque limit parameter or switch modes MC_Off, MC_EdgeOff, MC_EdgeSwitchPositive or MC_EdgeSwitchNegative
MC_StepLimitSwitch	Home to a forward or reverse limit switch	As above
MC_StepDirect	Sets the PLCs version of the axis position and exits the homing state	Use in combination with MC_StepAbsRefSwitch or MC_StepLimitSwitch
ECAT_HomingOnTouchProbe_APP	Home to an axis position latched by a fast input or z pulse on the drive	Use in combination with ECAT_CiA402_TouchProbe_APP

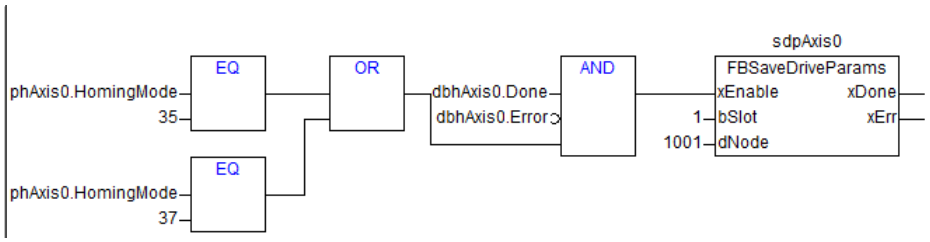
Drive based (DS402) homing methods		
Home type	*Description*	*Notes*
1	Negative limit switch and index pulse	Ensure feedback device supports index pulse
2	Positive limit switch and index pulse	Ensure feedback device supports index pulse
3	Positive home switch inactive and index pulse	Ensure feedback device supports index pulse
4	Positive home switch active and index pulse	Ensure feedback device supports index pulse
5	Negative home switch inactive and index pulse	Ensure feedback device supports index pulse
6	Negative home switch active and index pulse	Ensure feedback device supports index pulse
7	Home on home switch (inactive) and index pulse with positive initial direction. Reverse if positive limit switch hit	Ensure feedback device supports index pulse
8	Home on home switch (active) and index pulse with positive initial direction. Reverse if positive limit switch hit	Ensure feedback device supports index pulse
9	Home on opposite home switch (active) and index pulse with positive initial direction. Reverse if positive limit switch hit	Ensure feedback device supports index pulse
10	Home on opposite home switch (inactive) and index pulse with positive initial direction. Reverse if positive limit switch hit	Ensure feedback device supports index pulse
11	Home on home switch (inactive) and index pulse with negative initial direction. Reverse if negative limit switch hit	Ensure feedback device supports index pulse
12	Home on home switch (active) and index pulse with negative initial direction. Reverse if negative limit switch hit	Ensure feedback device supports index pulse
13	Home on opposite home switch (active) and index pulse with negative initial direction. Reverse if negative limit switch hit	Ensure feedback device supports index pulse
14	Home on opposite home switch (inactive) and index pulse with negative initial direction. Reverse if negative limit switch hit	Ensure feedback device supports index pulse
17	Home on negative limit switch	
18	Home on positive limit switch	
19	Home on positive home switch (inactive)	
20	Home on positive home switch (active)	
21	Home on negative home switch (inactive)	
22	Home on negative home switch (active)	
23	Home on home switch (inactive) with positive initial direction. Reverse if positive limit switch hit	
24	Home on home switch (active) with positive initial direction. Reverse if positive limit switch hit	
25	Home on opposite home switch (active) with positive initial direction. Reverse if positive limit switch hit	
26	Home on opposite home switch (inactive) with positive initial direction. Reverse if positive limit switch hit	
27	Home on home switch (inactive) with negative initial direction. Reverse if negative limit switch hit	
28	Home on home switch (active) with negative initial direction. Reverse if negative limit switch hit	
29	Home on opposite home switch (active) with negative initial direction. Reverse if negative limit switch hit	
30	Home on opposite home switch (inactive) with negative initial direction. Reverse if negative limit switch hit	
33	Home on index pulse with negative initial direction	
34	Home on index pulse with positive initial direction	
35	Home on current position	Supported but this is a legacy method that is now replaced by homing method 37
36	Home with touch probes	Not currently supported – use PLC based ECAT_HomingOnTouchProbe_APP function block instead
37	Home on current position	

Saving absolute position

Homing method 37 (or the legacy method 35) is used to set the current drive position to the defined home position value (the Position input parameter value for function block MCA_DriveBasedHome). If the axis uses an absolute encoder (e.g. Biss, Smartabs, SSI, Endat, resolver, Hiperface, Hiperface DSL) then there may be an application requirement to “teach” the axis position at some point (i.e. set a new reference position for the axis).

This can be achieved by performing a drive based home using method 37 (or 35) and then writing “evas” (“save” backwards) to object 16#1010 on the drive to force the drive to save the resulting position offsets and thereby store the new absolute position permanently.

The example project included with this application note has code to demonstrate this. The visualisation includes a pushbutton named “Set Home Pos” that when activated (appears pressed) and the home sequence is initated (via the Home button in the visualisation) will result in a pre-written function block (FBSaveDriveParameters) being called when the defined homing method is either 35 or 37 and the home sequence has been completed successfully…



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