Motion Control Products

Application note
Using the CD522 encoder module fast latch inputs

AN00240

Rev A (EN)

Use the fast latch/touch inputs on the CD522 2 channel encoder module to capture encoder position for use in applications such as those requiring registration corrections

Introduction

AC500 PLCs (PM585 and PM59x) can be used to perform real-time motion control of ABBs EtherCAT enabled servo drives. In some applications it is necessary to synchronize the motion of these drives to a master encoder (e.g. cut to length applications using a lay-on encoder to track the movement of the material to be cut to length). These applications may also require the master encoder/position value to be captured at high speed by a registration sensor for example. The CD522 encoder module is provided with two high speed inputs (I3 and I11), one for each of the two encoder channels, and in combination with the associated IEC61131 function blocks it is possible to utilize the captured values as part of a PLCopen motion application.

This application note details how to use Automation Builder to define the hardware setup suitable for use of the 2 channel encoder module (CD522) and how to then access the latched encoder value(s) provided by this module. For more general information on how to use the CD522 encoder module please refer to application note AN00239.

Pre-requisites

You will need to have the following to work through this application note and perform synchronized motion on an EtherCAT drive:

- Mint Workbench build 5812 or later (see new.abb.com/motion for latest downloads and support information)
- A MicroFlex e150 or MotiFlex e180 drive with build 5812 (e150) / 5819 (e180) or later firmware
- A PC or laptop running Automation Builder 1.2 or later
- An installed copy of the ABB PLCopen motion control library (PS552-MC-E v3.1.0 or later)
- One of the following AC500 PLC processors…..PM585, PM590, PM591, PM592 or PM595 (PLC processors should be running firmware version 2.5.1 or later). The PM595 is provided with an integrated EtherCAT coupler (this should be running firmware version 4.2.32.2 or later). All other processors require a CM579-ECAT communication module (which must be running firmware version 2.6.9 or later, but ideally version 4.3.0.2 or later). Contact your local ABB PLC support team for details on how to check these requirements and update if necessary or visit http://new.abb.com/plc/programmable-logic-controllers-plcs and select the link for ‘Software’. For the purposes of the text in this application note we have assumed the use of a PM591 PLC with CM579-ETHCAT coupler
- Straight-through Ethernet cable (patch cable) to connect the EtherCAT coupler to the drive (do not switch between patch and cross-over cabling as this can affect the order of devices on EtherCAT which is critical)
- Familiarity with application note AN00205 (AC500 and e150/e180 EtherCAT Getting Started Guide) and the Automation Builder PLC project that is included with it
- Familiarity with application note AN00239 (Using the CD522 encoder module for master encoder input) and the Automation Builder PLC project that is included with it
- A RS422 (5v differential line driver) incremental encoder

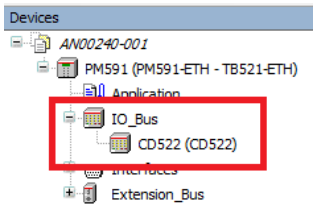
To follow the basic steps to create a hardware configuration and write some PLC code that uses the encoder values from the CD522 module only requires a PC or laptop running Automation Builder 1.2 or later and an installed copy of the PS552-MC-E motion control libraries. It is assumed the reader has a basic working knowledge of Automation Builder, CoDeSys and the

AC500 PLC and has commissioned an EtherCAT based servo drive (MicroFlex e150 or MotiFlex e180 for example) ready for use with the AC500 PLC.

Automation Builder – Configuring the operation of the fast inputs

Throughout this application note we will assume that the reader has opened the Automation Builder project supplied as part of AN00239 and we will just illustrate the additional steps needed to configure/use the CD522 module latch/touch inputs. Alternatively a ready-made example project is available as part of this application note.

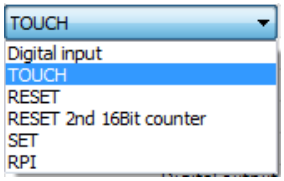
Open the PLC project provided as part of AN00239 from the Automation Builder 'File>Open Project…' menu and once opened select 'File>Save Project As…' to give your project a new name (to avoid damaging the original project). Once saved, double click the 'CD522' icon in the Devices tree…



…and in the right hand pane then scroll down to the 'Counter 0' and 'Counter 1' folders in the 'CD522 Parameters' section. The CD522 module is provided with 2 encoder channels (in application note AN00239 we showed how to configure one of these, Channel 0, to operate in conjunction with a RS422 5v differential line driver encoder). Each encoder channel can be latched/captured by a single dedicated fast input…

Channel 0 – Input I3
Channel 1 – Input I11

By default these inputs are configured as general purpose digital inputs, but by clicking on the existing value of the parameter we can instead select from a list of functions…



We only have a single encoder connected to encoder channel 0 on the CD522 module so we will only select 'TOUCH' for I3 (if there is a requirement to capture encoder channel 1 then repeat this process for input I11 – note that if the application requires a single master encoder/axis to be latched twice then it is necessary to wire the master encoder to both CD522 encoder channels, wire the registration sensor to both fast inputs in parallel and then configure both I3 and I11 as TOUCH inputs).

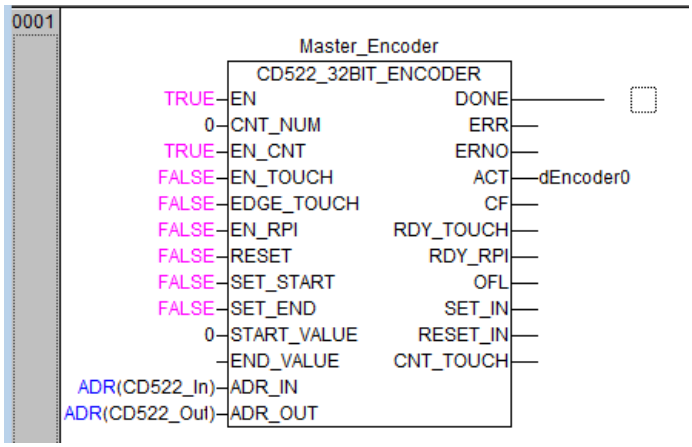
At the end of this process the parameters for Counter 0 should look like this…

Counter 0					
Input I3, channel configuration	Enumeration of BYTE	TOUCH	Digital input		Input I3 - Configuration of input channel
Input I4, channel configuration	Enumeration of BYTE	Digital input	Digital input		Input I4 - Configuration of input channel
Input I5, channel configuration	Enumeration of BYTE	Digital input	Digital input		Input I5 - Configuration of input channel
Input I6, channel configuration	Enumeration of BYTE	Digital input	Digital input		Input I6 - Configuration of input channel
Input I7, channel configuration	Enumeration of BYTE	Digital input	Digital input		Input I7 - Configuration of input channel
Output C4, channel configuration	Enumeration of BYTE	Digital output	Digital output		Output C4 - Configuration of output channel
Output C5, channel configuration	Enumeration of BYTE	Digital output	Digital output		Output C5 - Configuration of output channel
Output C6, channel configuration	Enumeration of BYTE	Digital output	Digital output		Output C6 - Configuration of output channel
Output C7, channel configuration	Enumeration of BYTE	Digital output	Digital output		Output C7 - Configuration of output channel

Save the project and launch the CoDeSys programming environment from Automation Builder.

PLC application – Configuring the CD522 function block

In application note AN00239 we showed how to include a CD522_32BIT_ENCODER function block in the application code. Open the EtherCAT_Control program and go to network 0001 to see the original setup for the CD522_32BIT_ENCODER function block…

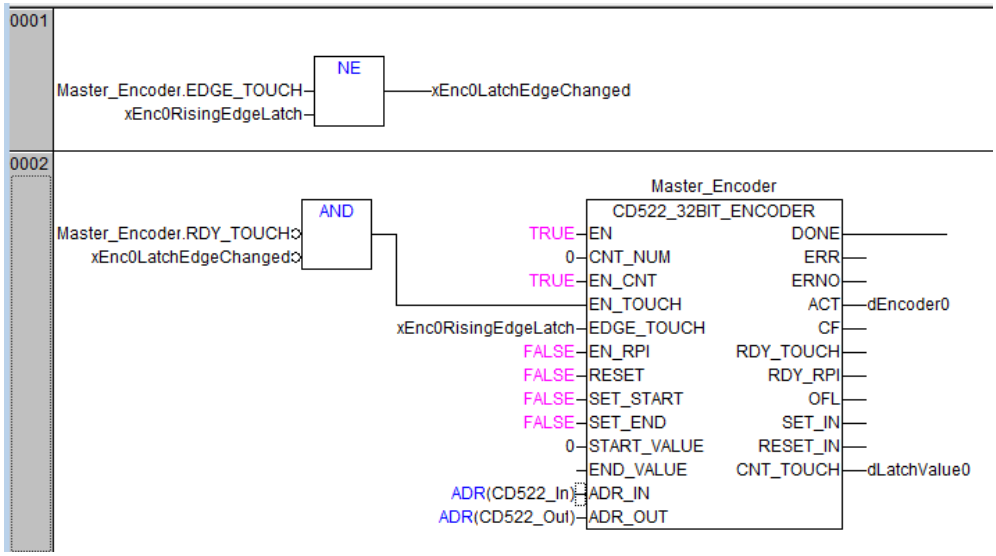


This function block was originally configured to not use the latch/touch inputs (EN_TOUCH = FALSE), so now we need to modify this network/function block so it does use the touch input (I3 as we are using encoder channel 0) by setting EN_TOUCH to TRUE.

However, we also “hard coded” TRUE as the EDGE_TOUCH input (for rising edge latching) when we created the code for AN00239. In many cases an application may only use one form of edge triggering so hard coding TRUE or FALSE may be acceptable, but in some applications it may be necessary for the code to handle switching between rising and falling edge latching of the encoder value. In these cases it is necessary to use an application variable (of type BOOL) to set the edge configuration.

New values for EDGE_TOUCH are only read in on a rising edge of EN_TOUCH so it becomes necessary to include some logic on the EN_TOUCH input to force a rising edge whenever the setting for EDGE_TOUCH changes…..this is quite easily achieved with a new rung comparing the current setting of EDGE_TOUCH with the (new) variable used to set this to create an “edge changed” flag.

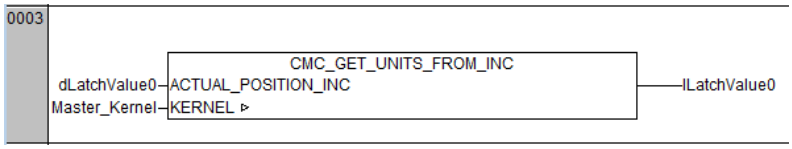
The screenshot below shows a typical program solution for handling changing the latch edge at runtime and assigning the latched encoder value to a new variable named dLatchValue0 (of type DINT)…



xEnc0LatchEdgeChanged and xEnc0RisingEdgeLatch are our two new (BOOL) variables used as part of detecting a change in the edge configuration.

PLC application – Accessing the latched axis position

CNT_TOUCH (assigned to our variable dLatchValue0 in this example) is an encoder value (i.e. in our application this will be a value in the range -2147483648 to +2147483647), but in our application we are most likely to require the fast/latched position of the master axis for registration purposes. We must therefore somehow "translate" the latched encoder value into an equivalent master axis position (taking into consideration that our master axis may also be a modulo axis). This is achieved via the CMC_GET_UNITS_FROM_INC function (provided as part of the PS552-MC-E motion libraries) which will convert the encoder value into a (LREAL) axis position as shown below…



lLatchValue0 is a new variable which we declared as type LREAL and which was assigned to the output of the CMC_GET_UNITS_FROM_INC function.

In this example we perform this conversion every cycle, but in a real application it is more typical to only take this action when RDY_TOUCH (the CD522_32BIT_ENCODER function block output indicating presence of a new latch value) is TRUE……remember that RDY_TOUCH is only TRUE for one scan of our EtherCAT_Control program.

Save the file so you don't lose any of these changes. If you like you can Login to the PLC and run the program to check the operation of the encoder block. As the connected master encoder rotates and the value of dEncoder0 changes try operating digital input I3 on the CD522 encoder module (you will need a switch or sensor of some kind wired to the input to do this). If all is well you should see the program store a latched axis position into the lLatchValue0 variable every time the switch is operated. Try using the variable force facility within CoDeSys to change the edge configuration and check to see if the encoder/axis position is now latched on the opposite switch edge.

Please consult the Automation Builder help system for further information about the CD522 module if required.

PLC application – Filtering the latched values

In many applications it will be necessary to reject/ignore the latched value unless either…

(a) the latched value is within a specific position range/window
Or
(b) the latch has occurred when the axis has travelled at least a certain distance since the previous latch (often known as a latch filter distance)

Detecting the latch is within a certain position range is relatively simple as the output of the CMC_GET_UNITS_FROM_INC function (assigned to lLatchValue0 in our example) can be compared with upper and lower position boundaries.

Example (in structured text):

```
IF (lLatchValue0 >= 100) AND (lLatchValue0 <= 400) THEN
   (* Valid latch position *)

END_IF;
```

Case (b) above, applying a latch filter distance, is a little harder to achieve. The application logic cannot use the output from CMC_GET_UNITS_FROM_INC in case of a modulo axis (e.g. the first latch may occur at position 10, the master axis then moves many cycles before another latch occurs at position 10.1……a comparison in position alone would conclude that the axis had only displaced by 0.1 units, whereas in reality the axis has moved much further than this having completed many cycles of operation).

For case (b) it is typical to include the filtering logic at the encoder level (i.e. by comparing the latched outputs from the CD522_32BIT_ENCODER function block, variable dLatchValue0 in our example). Care must be taken to ensure the wrap boundary of the encoder count (i.e. 2147483647 to -2147483648) is considered when calculating the distance travelled between a previously valid latch value and the next latch value (e.g. if one latch occurs at 2147483630 and the next occurs at -2147483640 this is a travel of 25 counts and not -4294967270 as would be obtained if the code simply subtracted the previous value from the new value).

A function named 'doWrap' is included in the sample project included with this application note to assist with detecting distance travelled. In structured text an example of its usage is shown below…

```
lDistanceTravelled := doWrap ( lNewValue – lLastValue , -2147483648, 2147483647);
```

Remember also that the application may require the latch filter distance to be adjustable and entered by the machine operator in user units for the master axis, so it may be necessary to convert the specified filter distance into encoder counts (by multiplying by the number of encoder counts in one user unit).

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/drivespartners
new.abb.com/PLC

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