Motion Control Products

Application note
Using an AC500 to control ACS drives over EtherCAT

AN00255

Rev A (EN)

ABB make a wide array of industrial drives that can be used in conjunction with an AC500 motion control system. Use of a FECA-01 option card allows for EtherCAT communication between these devices.

Introduction

This application note provides an insight into some of the techniques which can be used to easily implement control of one or more ABB industrial (ACS) drives over EtherCAT using an AC500 PLC as the EtherCAT manager.

The code that accompanies this application note is targeted at the hardware illustrated in the images above, namely a motion capable AC500 PLC with a CM579-ETHCAT coupler and an ACS380 Machinery drive with a FECA-01 EtherCAT adapter fitted. The drive parameter backup as well as an optional CP600 HMI control program are embedded in the example project should you care to expand the scope of the program operation.

Though this example discusses the ACS380, any drive that is compatible with the FECA-01 can use the methods and control techniques described. These drives include ACS380, ACS580, and ACS880. These drives are typically used in speed or torque control modes with asynchronous or permanent magnet motors which can be run either open loop (sensor less) or closed loop using optional (FEN-xx, MTAC-xx or BTAC-xx) feedback modules.

A lot of the general steps for configuring and adding objects to an Automation builder project are covered in the following application note: AN00205 - AC500 and ABB Motion Drives - EtherCAT Quick Start Guide Rev C (EN). This can be downloaded from the website: new.abb.com/motion

Additional support and .xml files can be sourced from the FECA-01 home page; http://new.abb.com/drives/connectivity/fieldbus-connectivity/EtherCAT/EtherCAT-feca-01

The ACS drives support both the CiA402 drive profile and a manufacturer (ABB) specific drive profile and the AC500 PLC provides function blocks for both of these. However, it is more typical to use the ABB profile and the dedicated ACS function blocks when using AC500 as these are easier to use than the more generic CiA402 features. If a third-party EtherCAT master device is used to control the ACS drives then this is a situation where the CiA402 profile would be selected instead – please refer to application note AN00256 for further details.

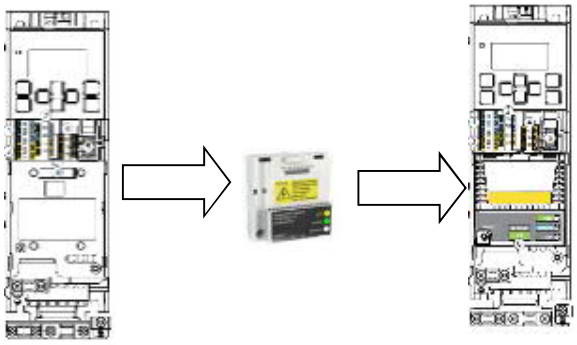
Pre-requisites

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

- An ACS380 with firmware AMCK6 v1.73.8.0 or later (see parameter 07.05) and FECA-01 EtherCAT adaptor with firmware version 1.30 or later (or the included files are easily adapted to suit other ACS series drives).
- A PC or laptop running Automation Builder 1.2 with at least a standard license (DM100-TOOL).
- A copy of Drive Composer Pro 1.12 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) or a PM595 PLC with integrated EtherCAT coupler
- A working knowledge of the basic operation of the AC500 PLC and ACS380 drive

Control hardware

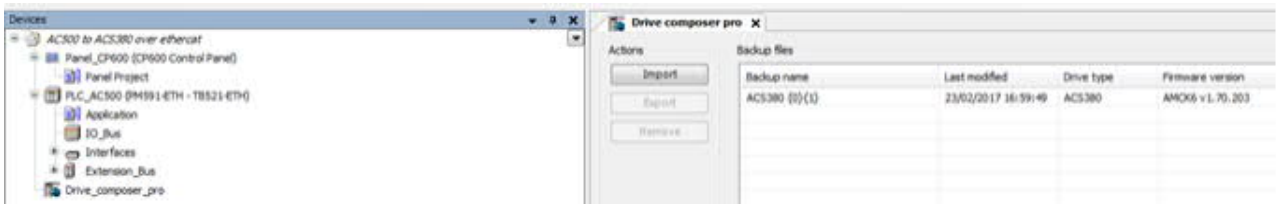
Not all ABB ACS drives support EtherCAT as standard and as such a FECA-01 needs to be added to the 'base variant'. This can be done either by adding a '+ code' at point of order (+K469 FECA-01-M Preconfigured EtherCAT protocol) or retro-fitting a FECA-01 after the drive is delivered. If you do the latter the FECA-01 must be hardware revision J or later.



If the drive is ordered with the + K469 option then all the relevant settings should already be made within the module. If it is retro fitted after the drive has been ordered the parameters must be set up as required.

Drive parameters

In the Drive Composer Pro section of Automation Builder you can see there is a saved backup of ACS380 settings that will configure the drive to operate via EtherCAT using the ABB drive profile;



If we wanted to set up a blank drive manually we would carry out the steps below;

- Start by adding the FECA-01 module
- Configure the drive to accept it by setting **50.01 FBA A enable** to **'Enable' (1)**
- This adapter in the ACS380 will always be called FBA A (though in ACS880 drives the FECA-01 could be installed in 'Slot A' or 'Slot B' and so would be FBA-A or FBA-B). The adapter type should then read the module identifier and **51.01 FBA A** type should change automatically to **EtherCAT (135)**
- Ensure the drive profile parameter **51.02** is set to **ABB Drives (1)** (the alternative setting is **CiA402 / 0** which is discussed in application note AN00256). Once set then **51.27** should be set to **Refresh** while online and the drive will henceforth operate with the ABB drive profile

Now the drive is ready to communicate with the PLC it must be configured to accept the signals to control it;

- Firstly the drive must be configured to ensure it is being controlled from the correct 'External Control Location'. There is a way to configure whether a drive can be controlled from single or multiple external locations (Ext 1 and Ext 2). By setting **19.11 Ext1/Ext2 Sel = EXT1 (0)** the drive will only use EXT1 as a control source (and in turn we will configure EXT1 to be a fieldbus)
- The Ext1 control command location is defined by parameter **20.01 Ext1 commands,** this can be set to accept commands from the control panel, digital inputs or fieldbus. To use the fieldbus adapter we must set this to **Fieldbus A (12)**
- When the start and stop is controlled via a fieldbus we must also set **20.02 Ext1 start trigger type** to **Level (1)**
- The profile being used by EXT1 can control the drive in either Speed or Torque mode. Setting **19.12 Ext1 control mode** to **Speed (2)** sets the drive in **velocity** control mode

- The source for the EXT1 demand/reference is defined by **22.11**. To use Ref 1 as sent by the fieldbus adapter A then we need to set **22.11 Ext1 speed ref 1** to **FBA Ref 1 (4)**
- For the drive to interpret and route the data sent to it from the bus master in the correct way we must configure how the data is routed inside the drive. As you will see from the PDO mappings on the following page, with RxPDO set 21 the first two words we use are Control Word and Reference 1 so we must mimic these settings in the drive. Set **53.01 FBA data Out 1** = **CW (1)** and **53.02 FBA data Out 2** = **Ref 1 (2)**
- With TxPDO set 21 the first two words we use are Status Word and Actual 1 so we must also mimic these settings in the drive. Set **52.01 FBA data In 1** = **SW 16 bit (4)** and **52.02 FBA data In 2** = **Act 1 16 bit (5)**

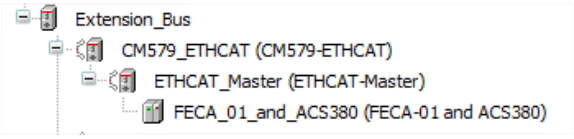
Once all of the relevant settings have been made the drive is configured and can now be used with the AC500 PLC.

Configuring Automation builder hardware

In case you don't have the target files as part of your Automation builder installation, or if you want to ensure you have the most up to date versions you can download the zip file from the link below (or from the FECA-01 home page);
https://search.abb.com/library/Download.aspx?DocumentID=9AKK105152A4454&LanguageCode=en&DocumentPartId=&Action=Launch

Then extract the zip file and in Automation Builder (via Tools > Device repository > Install) add the downloaded file**; FECA-01_1.30_ACS380_v2.00.0.4.xml** to the repository.

Add a motion capable AC500 CPU to the project, then add a CM579_ETHCAT to Slot 1 of your configuration and add to the ETHCAT Master section a 'FECA-01 and ACS380'. Your hardware tree will look something like this;



Note that only the ACSM1 and the motion variant of ACS880 (ACS880-M04) drives support synchronisation over EtherCAT (i.e. operation in either SM [Synchronous Mode] or DC [Distributed Clock] modes), so for the other drives in the ACS series which operate in Asynchronous / Free run mode (such as the ACS380 we are using in this example) there is no need to configure any further network settings relating to the FECA module other than adding Process Data Object mappings for the data to be exchanged with the drive (the EtherCAT master cycle time would be set to suit any other slave devices that do require synchronisation, such as a MicroFlex e190 servo drive for example).

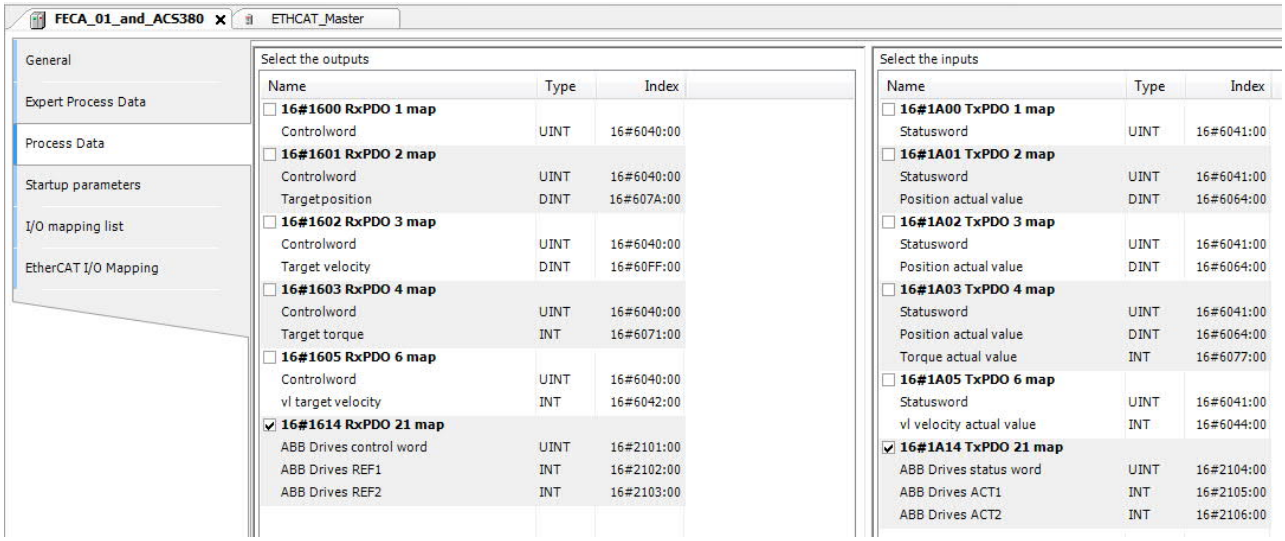
Understanding mapping objects and Process Data Objects

To communicate with any EtherCAT node on a cyclic basis we can configure data to be exchanged in both directions. This data is known as Process Data Objects (or PDOs). Within the FECA-01 .xml file there are default sets of PDO's available called data objects maps. These can be found in the **Automation Builder > Extension Bus > FECA-01 configuration > Process Data** tab. There are data mapping objects for both PLC outputs (drive RxPDO) and PLC inputs (drive TxPDO) which contain certain default PDO data elements. These can also be edited by enabling 'Expert settings' and adding or removing other PDO's in the **Expert Process Data** tab if needed.

These PDO sets have been created to simplify configuration of the drive to suit pre-defined operation modes and to suit whether the drive is being used in conjunction with the ABB drive profile or the CiA402 drive profile.

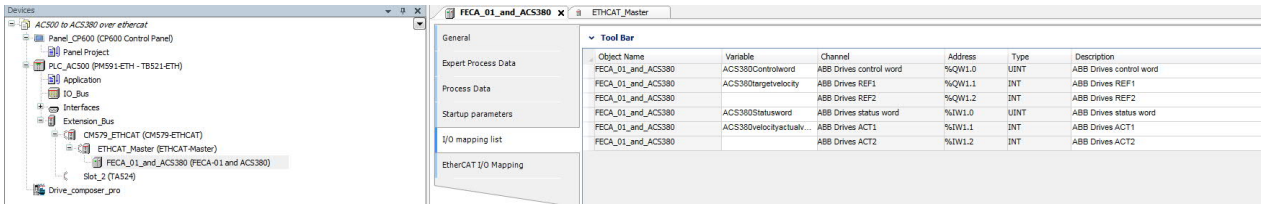
Each of the output PDO sets contains a Controlword mapping and each of the input PDO sets contains a Statusword mapping. The other PDOs included with each PDO set depend on which drive profile is used and how the drive is to operate (e.g. RxPDO 3 contains a Target Velocity PDO mapping which might be used if an ACSM1 motion variant drive was operating with the CiA402 drive profile in cyclic synchronous velocity mode)…

In this application note we are only considering operation with the ABB drive profile (please refer to AN00256 for further information about the available PDO sets and how these are used in conjunction with the CiA402 drive profile). When using the ABB drive profile the ACS drives access data from manufacturer specific objects for the control word, status word and references (objects 0x2101, 0x2102 and 0x2103 respectively) and these can be mapped by selecting PDO data set 21 as shown below…



In our example we will only use REF1 and ACT1…we configured the drive earlier (via parameter 19.12) to treat REF1 as a Speed reference and ACT1 as an actual velocity value (when using the drive with the ABB drive profile it is only possible to use velocity or torque references that the drive profiles/ramps itself).

After the PDO set has been selected we can see that variables appear in the EtherCAT I/O mapping tab. In here we can enter variable names for all the mapped variables that we need to use in the IEC61131-3 programming environment;



Controlling the drive using the ABB Drives profile and associated function blocks

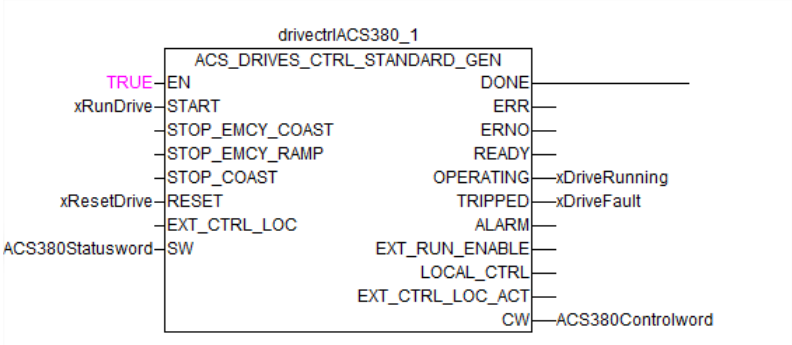
Now we have configured our PDO mappings we can use the variables we defined for these in our application code to control and interact with the ACS drive. As we mentioned previously, the easiest method to control the ABB ACS drives from an AC500 PLC is by using the ABB drives profile and AC500 function blocks from a dedicated library (PS553-DRIVES) provided specifically for control of these (we saw earlier how we selected the ABB Drives profile by setting parameter **51.02 = ABB Drives** and then setting **51.27 = Refresh**).

PS553-Drives-Library

This library is included in all versions of Automation Builder. To add this library to your project you will need to go the IEC61131-3 programming environment and select Resources > Library manager > Right Click in the lib list window > Additional Library > Look for PS553-DRIVES Folder > Add ACSDrivesBase_AC500_V20.lib. Once installed a number of pre-written function blocks to control and monitor the ACS drives become available.

Control Word and Status Word

The PS552-DRIVES library function block we will use to control and monitor the ACS drive is the 'ACS_DRIVES_CTRL_STANDARD_GEN' block which requires that control word (CW) and status word (SW) variables be connected to it. The drive can then be set into run by setting the START input to TRUE as long as no faults are present.



Further information about this function block can be found in the Automation Builder Help system.

Reference / Demand

50.04 FBA A ref1 type selects the type and scaling of reference 1 received from fieldbus adapter A. In this instance this needs to be set to ***50.04* = Speed**. The ABB Drives profile reference is a 16-bit word containing a sign bit and a 15-bit integer. The reference values are scaled as defined by;

+/- 20000 = +/- full speed or +/- full frequency as set by parameter ***46.01*** (speed scaling) or ***46.02*** (frequency scaling)
+/- 10000 = +/- motor nominal torque as set by parameter ***46.03*** *(torque scaling)*

In our example with **50.04** set to **Speed** and ***46.01 = 1500rpm*** a reference of 20000 will issue a command velocity of 1500rpm to the drive

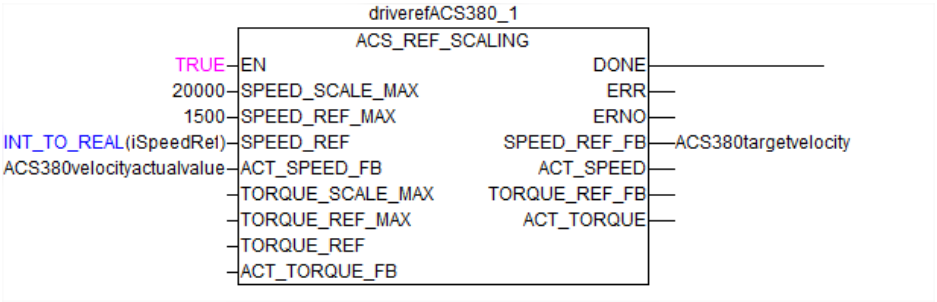
Actual values

50.07 FBA A actual 1 type selects the type and scaling of reference 1 received from fieldbus adapter A. In this instance this needs to be set to ***50.07* = Speed**. The ABB Drives profile actual signal used is a 16-bit word each containing a sign bit and a 15-bit integer. The actual values are scaled as defined by;

+/- 20000 = +/- full speed or +/- full frequency as set by parameter ***46.01*** (speed scaling) or ***46.02*** (frequency scaling)
+/- 10000 = +/- motor nominal torque as set by parameter ***46.03*** *(torque scaling)*

In our example we have set **50.07** to **Speed** and ***46.01 = 1500rpm*** so an actual speed of 20000 read back over EtherCAT will show us the drive is running at 1500rpm.

In the example program we also make use of a standard PS553-DRIVES library function block associated with the ABB Drives profile called 'ACS_REF_SCALING' which, as you can see from the screenshot below, takes the raw values (0-20000) and scales them into rpm (0 – Motor nominal speed) for use inside the code.



Notice how our PDO mapping for actual speed (ACS380velocityactualvalue) is connected to the ACT_SPEED_FB input and our PDO mapping for demand velocity (ACS380targetvelocity) is connected to the SPEED_REF_FB output.

The ACS_REF_SCALING block can also handle the scaling of a torque reference. This could either be used to provide a scaled value for REF2 on the drive (allowing the drive to possibly switch between speed and torque references for example) or to provide the scaled torque reference to REF1 if the drive was configured for torque mode instead of speed mode.

Just through the use of these two function blocks (ACS_DRIVES_CTRL_STANDARD_GEN and ACS_REF_SCALING) we are now able to start, stop, set demand velocities, monitor drive status etc… via EtherCAT.

Control from CP600 HMI

The example project included with this application note includes a simple CP600 HMI project that can be used to interact with the example PLC program. You can use it to run and reset the drive as well as setting a speed reference by clicking on the set speed box and entering a new value. The HMI will also report back on the drive state via the Running and Fault lights as well as displaying the actual speed.

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