iPOS
CoE
Programming

TECHNOSOFT

MOTION TECHNOLOGY

User Manual

© Technosoft 2024

P091.064.UM.1224

Read This First

Whilst Technosoft believes that the information and guidance given in this manual is correct, all parties must rely upon their own skill and judgment when making use of it. Technosoft does not assume any liability to anyone for any loss or damage caused by any error or omission in the work, whether such error or omission is the result of negligence or any other cause. Any and all such liability is disclaimed.

All rights reserved. No part or parts of this document may be reproduced or transmitted in any form or by any means, electrical or mechanical including photocopying, recording or by any information-retrieval system without permission in writing from Technosoft S.A.

The information in this document is subject to change without notice.

About This Manual

This manual describes how to program the Technosoft intelligent drives equipped with **EtherCAT®** communication interface. These drives support **CAN application protocol over EtherCAT® (CoE)** in conformance with **CiA 402** device profile. The manual presents the object dictionary associated with this profile. The manual also explains how to combine the Technosoft Motion Language and the **CoE** commands in order to distribute the application between the **EtherCAT®** master and the Technosoft drives.

In order to operate the Technosoft drives with EtherCAT® communication, you need to pass through 3 steps:

- ❑ **Step 1 Hardware installation**
- ❑ **Step 2 Drive commissioning** using initially the Technosoft **EasySetUp** or **EasyMotion Studio** software platforms and later via an EtherCAT® master

  A.

  B. developed using Technosoft **EasyMotion Studio** software

  C. A **distributed control** approach which combines the above options, like for example a master calling motion functions programmed on the drives in TML

This manual covers steps 2 and 3. For step 1, refer to the Technical Reference manual specific for each drive.

Scope of This Manual

This manual applies to the iPOS family of Technosoft intelligent drives having an EtherCAT® communication interface.

Notational Conventions

This document uses the following conventions:

AL – Application Layer
CoE - CAN application protocol over EtherCAT®
TML – Technosoft Motion Language
iPOS – a Technosoft drive family, the code is usually iPOSxx0x xx-CAN
GUI – Graphical User Interface
IU – drive/motor internal units
IP – Interpolated Position
RegisterY.x- bit x or register Y; **Example: Controlword.5** – bit 5 of Controlword data
cs – command specifier
CSP – Cyclic Synchronous Position
CSV – Cyclic Synchronous Velocity
CST – Cyclic Synchronous Torque
RO – read only
RW – read and write
SW – software
H/W or HW – hardware
***X1** – Subindex 1 of object 60C1h - Interpolation data record*
***X2** – Subindex 2 of object 60C1h - Interpolation data record*

Trademarks

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

Related Documentation

Help of the EasySetup software – describes how to use **EasySetup** to quickly setup any Technosoft drive for your application using only 2 dialogues. The output of EasySetup is a set of setup data that can be downloaded into the drive EEPROM or saved on a PC file. At power-on, the drive is initialized with the setup data read from its EEPROM. With EasySetup it is also possible to retrieve the complete setup information from a previously programmed drive. **EasySetup can be downloaded free of charge from Technosoft web page**

Technical Reference Manual of each iPOS drive version – describes the hardware including the technical data, the connectors, the wiring diagrams needed for installation and detailed setup information.

Motion Programming using EasyMotion Studio (part no. P091.034.ESM.UM.xxxx) – describes how to use the EasyMotion Studio to create motion programs using in Technosoft Motion Language (TML). EasyMotion Studio platform includes **EasySetup** for the drive/motor setup, and a **Motion Wizard** for the motion programming. The Motion Wizard provides a simple, graphical way of creating motion programs and automatically generates all the TML instructions. *With EasyMotion Studio you can fully benefit from a key advantage of Technosoft drives – their capability to execute complex motions without requiring an external motion controller, thanks to their built-in motion controller.* **A demo version of EasyMotion Studio (with EasySetup part fully functional) can be downloaded free of charge from Technosoft web page**

If you Need Assistance …

If you want to …	Contact Technosoft at …
Visit Technosoft online	World Wide Web: http://www.technosoftmotion.com/
Receive general information or assistance (see Note)	World Wide Web: http://www.technosoftmotion.com/ Email: sales@technosoftmotion.com
Ask questions about product operation or report suspected problems (see Note)	Tel: +41 (0)32 732 5500 Email: support@technosoftmotion.com
Make suggestions about, or report errors in documentation.	Mail: Technosoft SA Avenue des Alpes 20 Ch-2000 Neuchatel, NE Switzerland

1 Getting Started

1.1 Setting up the drive using EasySetup or EasyMotion Studio

1.1.1 What are EasySetup and EasyMotion Studio?

EasySetup is a PC software platform for the setup of the Technosoft drives. Via EasySetup you can quickly commission any Technosoft drive for your application using only 2 dialogues.

The output of EasySetup is the *setup data* that can be stored into the drive EEPROM or saved on a PC file. The *setup data* contains all the information needed to configure and parameterize a Technosoft drive. At power-on, the drive is initialized with the *setup data* read from its EEPROM. EasySetup may also be used to retrieve the *setup data* previously stored in a drive EEPROM.

EasySetup also includes evaluation tools like: Data Logger, Control Panel and Command Interpreter which help you to quickly measure, check and analyze your drive commissioning.

EasyMotion Studio is an advanced PC software platform that can be used both for the drives setup and for their motion programming. With EasyMotion Studio you can fully benefit from a key advantage of the Technosoft drives – their capability to execute stand-alone complex motion programs thanks to their built-in motion controller.

EasyMotion Studio includes **EasySetup** for the drive setup, and a **Motion Wizard** for the motion programming. The Motion Wizard provides a simple, graphical way of creating motion programs written in Technosoft Motion Language (TML). It automatically generates all the TML instructions, hence you do not need to learn or write any TML code. Via TML you can:

- ❑ Set various motion modes
- ❑ Change the motion modes and/or the motion parameters
- ❑ Execute homing sequences
- ❑ Control the program flow through:
    - o Conditional jumps and calls of TML functions
    - o Interrupts generated on pre-defined or programmable conditions (protections triggered, transitions of limit switch or capture inputs, etc.)
    - o Waits for programmed events to occur
- ❑ Handle digital I/O and analogue input signals
- ❑ Execute arithmetic and logic operations

The output of EasyMotion Studio is the *application data* that can be loaded into the drive EEPROM or saved on a file. The *application data* includes both the *setup data* and the *TML motion program*.

Using TML, you can really simplify complex applications, by distributing the intelligence between the master and the drives. Thus, instead of trying to command each step of an axis movement from the master, you can program the drives using TML to execute complex tasks, and inform the master when these tasks have been completed.

Important: *You need* ***EasyMotion Studio full version****, only if you use TML programming. For electronic camming applications, you need the free of charge* ***EasyMotion Studio demo version*** *to format the cam data. For all the other cases, you can use the free of charge* ***EasySetup****.*

1.1.2 Installing EasySetup or EasyMotion Studio

EasySetup and **EasyMotion Studio demo version** can be downloaded ***free of charge*** from Technosoft web page. Both include an ***Update via Internet*** tool through which you can check if your software version is up-to-date, and when necessary download and install the latest updates.

EasyMotion Studio demo version includes a fully functional version of **EasySetup**, hence you do not need to install both of them.

You can install the EasyMotion Studio full version in 2 ways:

a) Using the CD provided by Technosoft. In this case, after installation, use the ***Update via Internet*** tool to check for the latest updates;
b) Transforming EasyMotion Studio demo into a full version, by introducing in the application menu command **Help | Registration Info** the serial number provided by Technosoft.

The 2nd option is especially convenient if the EasyMotion Studio demo version is already installed.

Remark*: The next paragraphs present only the drive commissioning with EasySetup. Par.* ***18.1.1****. shows how to perform the same steps with EasyMotion Studio demo or full version.*

1.1.3 Establishing serial communication with the drive

EasySetup communicates with the drive via an RS-232 serial link. If your PC has no serial port, use an USB to RS232 adapter. For the serial connections, refer to the drive Technical Reference manual. If the drive or the Starter Kit board accompanying the drive has a 9-pin serial port, use a standard 9-wire, non-inverting (one to one) serial cable.

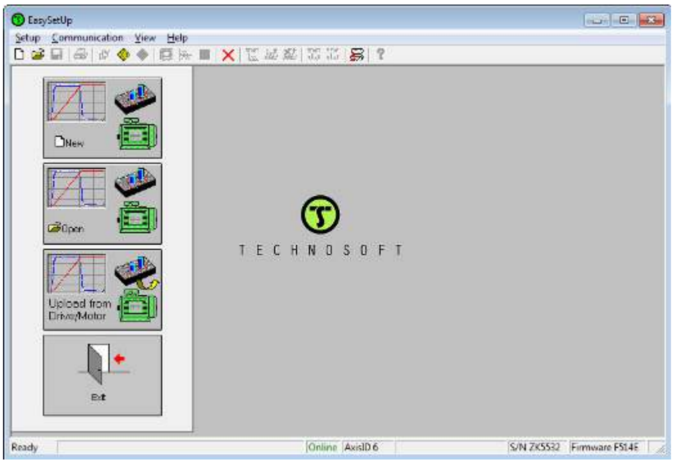


Figure 1.1.1. *EasySetup - Opening window*

All Technosoft drives with EtherCAT® interface have a unique AxisID (address) for serial communication. The AxisID value is by default 255 or it is set by the AxisID selection inputs, when these exist.

Remark: *When first started, EasySetUp tries to communicate via RS-232 and COM1 with a drive having axis ID=255 (default communication settings). When it is connected to your PC port COM1 via an RS-232 cable, the communication shall establish automatically.*

If the communication is established, EasySetUp displays in the status bar (the bottom line) the text "**Online**" plus the axis ID of your drive/motor and its firmware version. Otherwise the text displayed is "**Offline**" and a communication error message tells you the error type. In this case, use menu command **Communication | Setup** to check/change your PC communication settings. Check the following:

Channel Type: RS232
CAN Protocol: can be any selection. The CANbus is not used
Port: Select the COM port where you have connected the drive
Baud rate: can be any value. The baud rate is automatically detected. For best performance, we recommend to use the highest value: 115200.
Remark: *Once the communication is established, you can reopen the* ***Communication*** **| Setup** dialogue and change the baud rate

Axis ID of drive/motor connected to PC is: autodetected or 255

Close the **Communication | Setup** dialogue with OK and check the status bar. If the communication is established, the text "**Online**" shall occur in the status bar. If the communication is still not established, check the serial cable connections and the drive power. Refer to the Technical reference manual of the drive for details.

Remark: *Reopen the* ***Communication | Setup*** *dialogue and press the* ***Help*** *button. Here you can find detailed information about communication setup and troubleshooting.*

1.1.4 Choosing the drive, motor and feedback configuration

Press **New** button  and select your drive category.

Continue the selection tree with the motor technology: brushless, brushed or stepper.

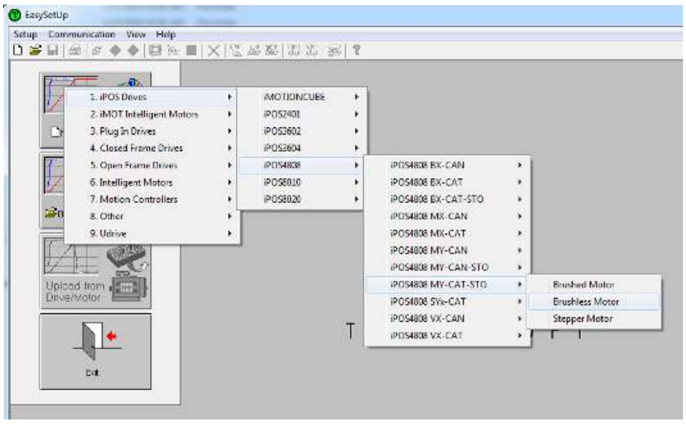


Figure 1.1.2. *EasySetup – Selecting the drive, motor and feedback*

The selection opens 2 setup dialogues: for **Motor Setup** and for **Drive setup** through which you can introduce your motor data and commission the drive, plus several predefined control panels customized for the drive selected.

1.1.5 Introducing motor data

Figure 1.1.3 shows the **Motor setup** dialogue where you can introduce the data of your motor and the associated sensors. Use the **Guideline Assistant**, and follow the steps described. This will guide you through the whole process of introducing and/or checking the motor and sensors data. Use the **Next** button to see the next guideline step and the **Previous** button to return to the previous step. Data introduction is accompanied by a series of tests having as goal to check the connections to the drive and/or to determine or validate a part of the motor and sensors parameters.

When finished, click on **Drive Setup** button to move to the 2nd dialogue.

Remark*: Press the **Help** button from the Motor setup dialogue for detailed information*

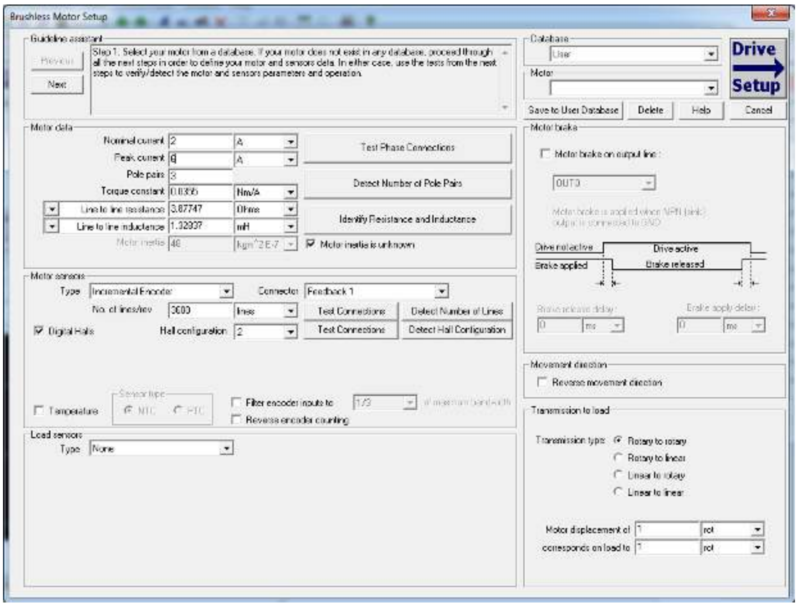


Figure 1.1.3. *EasySetup – Introducing motor data*

1.1.6 Commissioning the drive; configuring motor tuning and protections

Figure 1.1.4 shows the **Drive setup** dialogue where you can configure and parameterize the drive for your application.

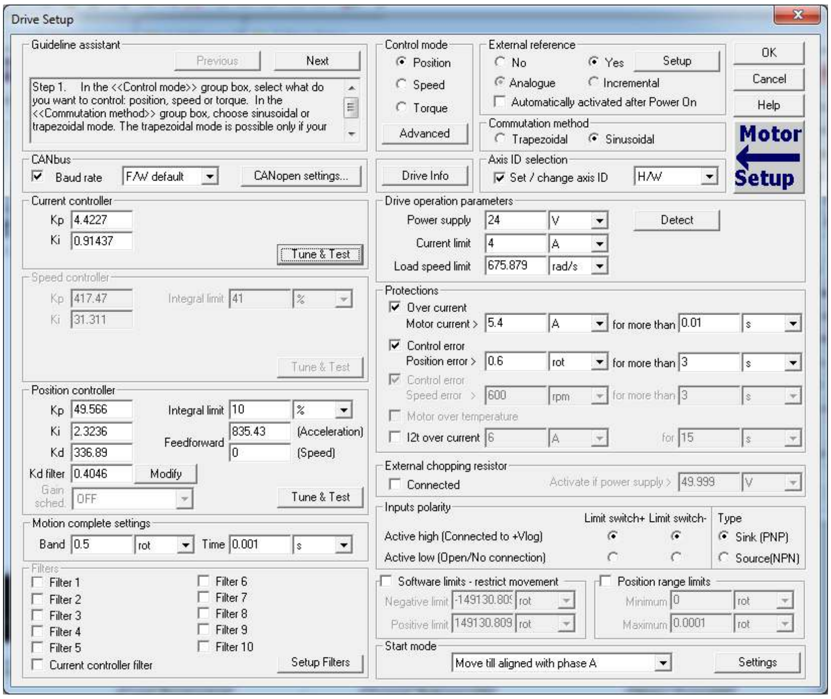


Figure 1.1.4. *EasySetup – Commissioning the drive*

iPOS firmwares have an auto tuning feature. Assuming the motor data was entered or identified correctly, just click on any “Tune & Test” button and a new window will appear.

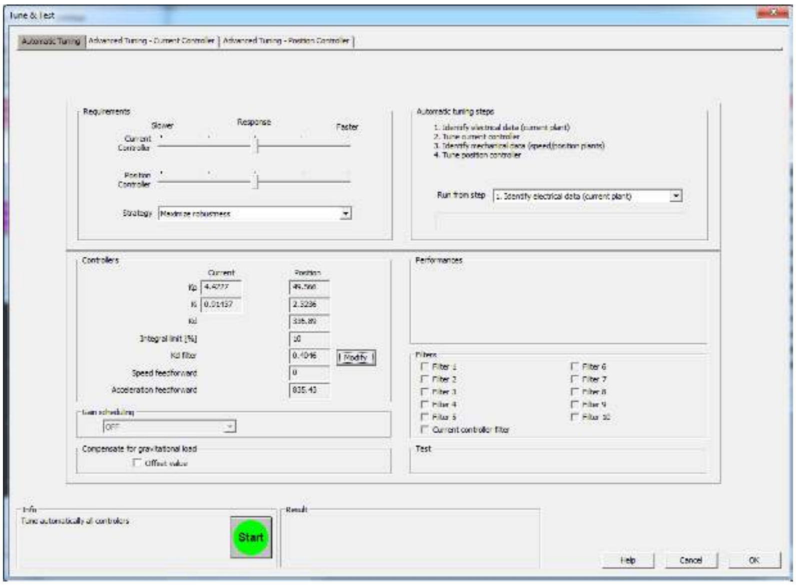
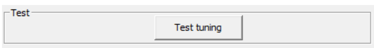


Figure 1.1.5. *EasySetup – Auto tuning interface*

Click the Start button and wait for the procedure to finish.

Once the procedure is finished, the tuning can be tested by pressing the newly appeared “Test tuning button”.



Click start and observe the motor move. If the Load position follows the Target position without error, then the tuning is OK.

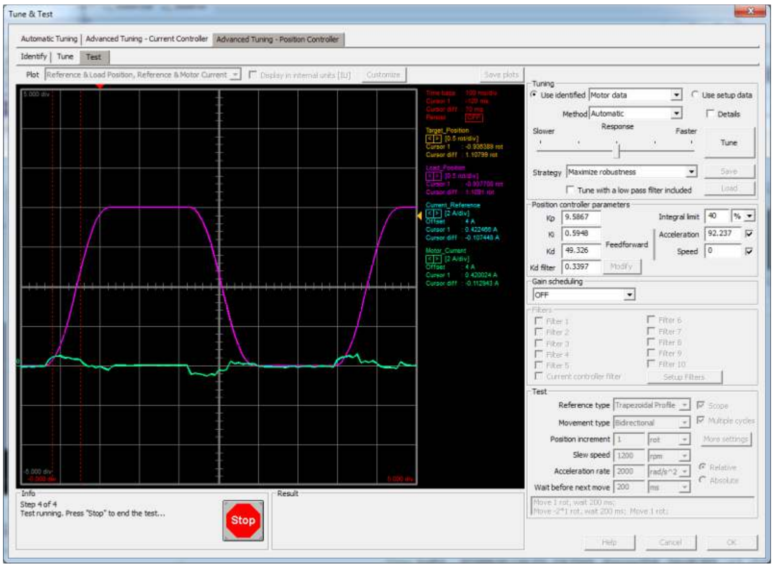


Figure 1.1.6. *EasySetup – Testing the motor tuning*

Eventually, if the motor vibrates or a softer tuning is needed, manually decrease the Kp, Ki and Kd gains.

Click Stop and wait for the test to stop. Click Ok to exit the window and keep the newly found tuning values. Click OK once again to exit the Drive Setup window and proceed to the next chapter to download the setup to the drive.

Remark: the drive will not move the motor unless a valid setup is downloaded to the drive.

1.1.7 Downloading setup data to drive/motor

Closing the Drive setup dialogue with **OK**, keeps the new settings only in the EasySetup project. In order to store the new settings into the drive you need to press the **Download to Drive/Motor** button 

. This downloads the entire setup data in the drive EEPROM memory. The new settings become effective after the next power-on, when the setup data is copied into the active RAM memory used at runtime.

1.1.8 Saving setup data in a file

It is also possible to **Save** 

the setup data on your PC and use it later.

To summarize, you can define or change the setup data in the following ways:

- create a new setup data by going through the motor and drive dialogues
- use setup data previously saved in the PC
- upload setup data from a drive/motor EEPROM memory

1.1.9 Creating a .sw file with the setup data

Once the setup was validated, you can create with the menu command **Setup | Create EEPROM Programmer File** a software file (with extension **.sw**) which contains all the setup data to write in the EEPROM of your drive.

A software file is a text file that can be read with any text editor. It contains blocks of data separated by an empty line. Each block of data starts with the *block start address*, followed by the block *data values* ordered in ascending order at consecutive addresses: first *data value* – what to write in drive EEPROM memory at *block start address*, second data – what to write at *block start address + 1*, third data – what to write at *block start address* +2 etc. All data are hexadecimal 16- bit values (maximum 4 hexadecimal digits). Each line contains a single data value. When less than 4 hexadecimal digits are shown, the value must be right justified. For example 92 is 0x0092.

The **.sw** file can be programmed into a drive:

- from an EtherCAT® master, using the communication objects for writing data into the drive EEPROM (see Par *17.3* for a detailed example)
- from an EtherCAT® master, using the FoE (File over EtherCAT) protocol available only with F515F or newer firmware. Check Par. *17.4.3*.
- using the EEPROM Programmer tool, which comes with EasySetUp but may also be installed separately. The EEPROM Programmer was specifically designed for repetitive fast and easy programming of **.sw** files into the Technosoft drives during production

1.1.10 Checking and updating setup data via .sw files with an EtherCAT® master

You can program an EtherCAT® master to automatically check after power on if all the Technosoft drives connected to the EtherCAT® network have the right setup data stored in their EEPROM. The comparison shall be done with the reference .sw files of each axis. These need to be loaded into the EtherCAT® master. There fastest way to compare a .sw file with a drive EEPROM contents is by comparing the checksums computed on the .sw file data with those computed by the drive on the same address range. In case of mismatch, the reference .sw file has to be reloaded into the drive by the EtherCAT® master. Par *17.4* presents examples how to program a .sw file in a drive and how to check its consistency versus a .sw reference file.

1.1.11 Testing and monitoring the drive behavior

The **Data Logger** or the **Control Panel** can be used for evaluation tools to quickly measure and analyze the application behavior. In case of errors like protections triggered, check the Drive Status control panel to find the cause.

1.2 Setting the current limit

In Easy Setup if a feedback device is used, the user can choose a current limit. It is advised to use a lower value than the one set in current protection.

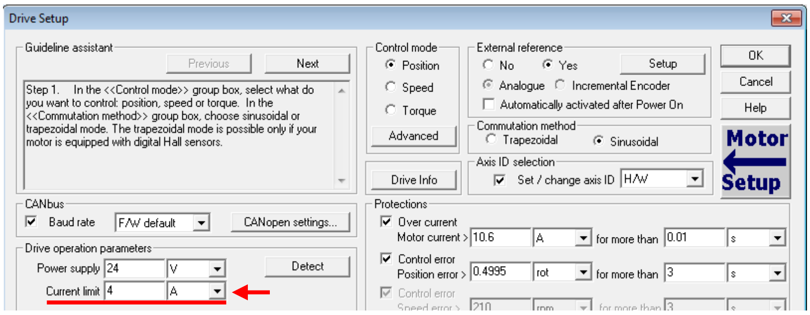


Figure 1.2.1. *EasySetup – Setting the current limit*

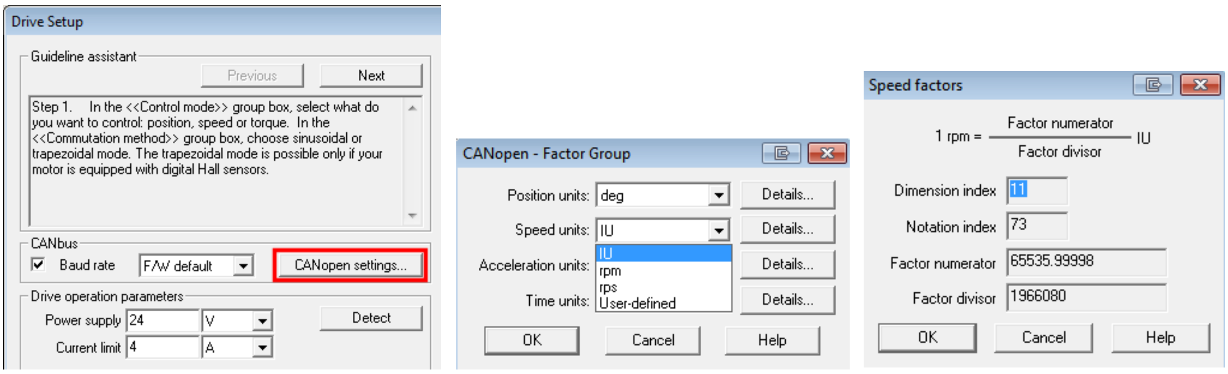
The current limit can also be set using Object 207Fh: Current limit.

1.3 Factor group setting

The factor group settings currently implemented are complying with:
- CiA-402-2 and later versions – starting with F515K / FA0xx firmware versions
- CiA-402 – for other firmware versions

1.3.1 Factor group setting - CiA-402 (obsolete)

The CANopen Settings button opens an interface that allows access to the scaling factors for position, speed, acceleration and time objects. These settings are linked directly to the objects 6089_h, $608A_h$, $608B_h$, $608C_h$, $608D_h$, $608E_h$, $206F_h$, 2070_h, 6093_h, 6094_h, 6097_h and 2071_h. This means that these settings can be chosen either from Setup or by later setting the objects themselves. The factor group dialogue can select the units to be used when writing or reading the Position, Velocity or Acceleration objects. These settings already have a list standard units defined in the standard CiA402 and there is the option of customization.



In the last case, the user can set the factor numerator and divisor in order to obtain the needed scaling. The dimension and notation index (and their linked objects) have no influence over any scaling. Their purpose is only to define an [SI] unit name like rpm, rad, deg, etc. The factor group settings are stored in the setup table. By default, the drive uses its internal units. The correspondence between the drive internal units and the [SI] units is presented in the drives user manual.

For the [SI] dimension and notation index list, see ***Dimension/Notation Index Table.***

Remarks:

- the dimension and notation index objects (6089$_h$, 608A$_h$, 608B$_h$, 608C$_h$, 608D$_h$, 608E$_h$, 206F$_h$ and 2070$_h$) have been classified as obsolete by the CiA 402 standard. They are now used only for legacy purposes, on EtherCAT masters which still need them.
- because the iPOS drives work with Fixed 32 bit numbers (not floating point), some calculation round off errors might occur when using objects 6093$_h$, 6094$_h$, 6097$_h$ and 2071$_h$. If the EtherCAT® master supports handling the scaling calculations on its side, it is recommended to use them instead of using the "*Factor*" scaling objects.

1.3.2 Factor group setting - CiA-402-2

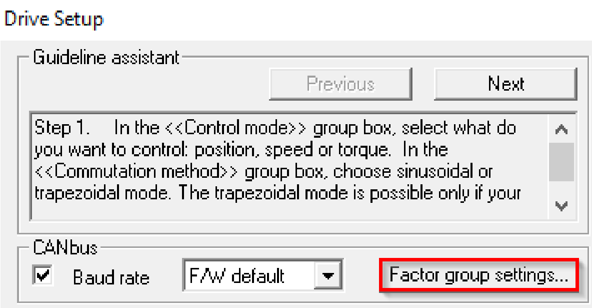
The Factor Group Settings button opens an interface which allows to select the desired physical unit for position, speed, acceleration and jerk values. These settings are linked directly to the objects presented in chapter Factor group objects - CiA-402-2.

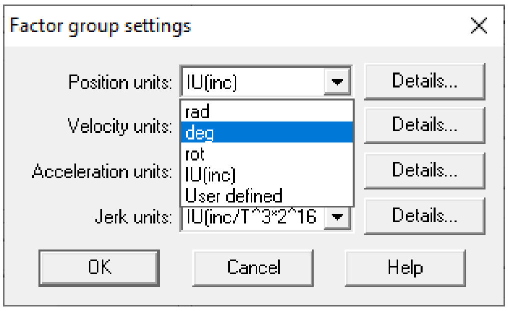
The factor group that complies with CiA-402-2 is available starting with firmware version F515K / FA0xx. EasyMotion Studio / EasySetUp must be updated to the latest version. Otherwise, the obsolete version will be displayed.

The factor group settings can be modified either in the Setup part of the project, or by changing the factor group objects directly using EtherCAT protocol.

If the settings are changed in the Setup part, once the desired unit is selected, EasyMotion Studio automatically computes the scaling factors according to each mechanical setup.In this case, the settings are stored in the non volatile memory and remain active regardless of the drive state (reset, power lost, etc.).

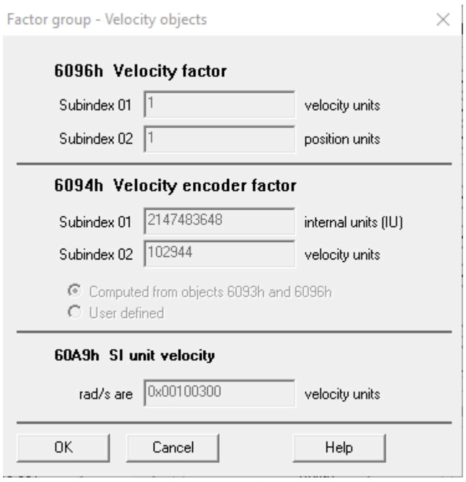
The Factor Group should be adjusted once before any type of movement is realised and not changed during the movement.





In the "Details…" part can be found all the objects that corresponds to the specific unit and the scalling factors computed by EasyMotion Studio according to the feedback, transmission and slow loop period.

If other units than the standardized option are needed, the scaling can be obtained also in the setup part if "User defined" option is selected.



By default, the drive uses its internal units (IU). The correspondence between the drive internal units and the [SI] units is presented in the Help menu of EasySetup/EasyMotion Studio.

1.4 Using the built-in Motion Controller and TML

One of the key advantages of the Technosoft drives is their capability to execute complex motions without requiring an external motion controller. This is possible because Technosoft drives offer in a single compact package both a state of art digital drive and a powerful motion controller.

1.4.1 Technosoft Motion Language Overview

Programming motion directly on a Technosoft drive requires to create and download a TML (Technosoft Motion Language) program into the drive memory. The TML allows you to:

- Set various motion modes (profiles, PVT, PT, electronic gearing or camming, etc.)
- Change the motion modes and/or the motion parameters
- Execute homing sequences
- Control the program flow through:
  - Conditional jumps and calls of TML functions
  - Interrupts generated on pre-defined or programmable conditions (protections triggered, transitions of limit switch or capture inputs, etc.)
  - Waits for programmed events to occur
- Handle digital I/O and analogue input signals
- Execute arithmetic and logic operations

In order to program a motion using TML you need EasyMotion Studio software platform.

Chapter *18* describes in detail how the TML features can be combined with the CoE programming.

1.5 Setting up EtherCAT® communication. Example with TwinCAT3

This paragraph shows how to set up the EtherCAT® communication using Beckhoff TwinCAT3 software, running on a PC and provides several examples how to program the drive for different modes of operation: position profile and cyclic synchronous position. Another example shows how to map objects in TxPDOs and RxPDOs.

The TwinCAT version used in all examples presented is the 7 day free version v3.1.4022.29 downloaded from www.beckhoff.de.

1.5.1 Adding the XML file

The .xml file can be found on the Technosoft web page of each EtherCAT drive. After TwinCAT installation is complete, copy the appropriate .xml file for your product in:

TwinCAT2 installation folder \Io\EtherCAT. The default location is "C:\TwinCAT\Io\EtherCAT"

TwinCAT3 installation folder \3.1\Config\Io\EtherCAT. The default location is "C:\TwinCAT\3.1\Config\Io\EtherCAT"

TwinCAT will need a restart in order to load the new file.

1.5.2 Understanding EtherCAT® addressing modes supported

Three device addressing modes are available: auto increment addressing, configured station address, and broadcast. EtherCAT® devices can have up to two configured station addresses, one is assigned by the master (Configured Station Address) and the other one can be changed by the iPOS drive (Configured Station Alias address). The Configured Station Alias address is loaded from the drive only after power-on or reset.

In case of device addressing mode based on node address, the iPOS-CAT drive sets the *configured station alias* address with its AxisID value. The drive AxisID value is set after power on in one of the following ways:

a) By hardware[1], function of the axis ID inputs. The AxisID value is computed in the same way as in the case of iPOS-CAN drives using the CANopen® protocol (see the user manual of the drive)

   Remark: *any other combination of voltage levels not included in the CANopen addressing table, like for example the levels from TMLCAN addresses will set the axis ID equal with 255.*

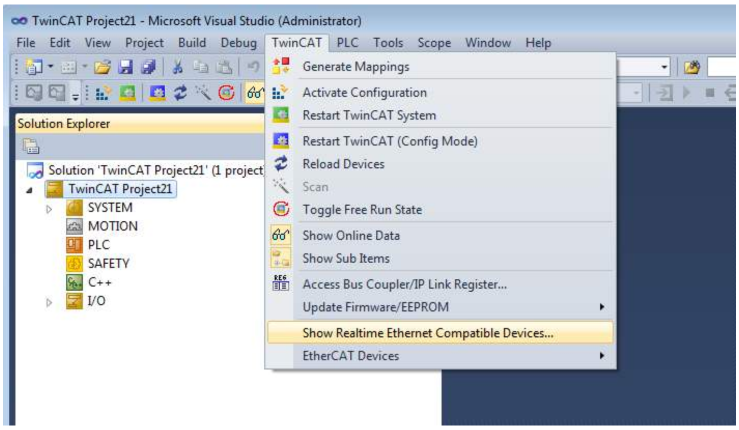
By software, imposed via EasySetUp a specific AxisID value in the range 1-255.

1.5.3 Detecting the drive with TwinCAT3

If everything is connected and turned on, the link and activity LEDs on the EtherCAT® drive should be ON.

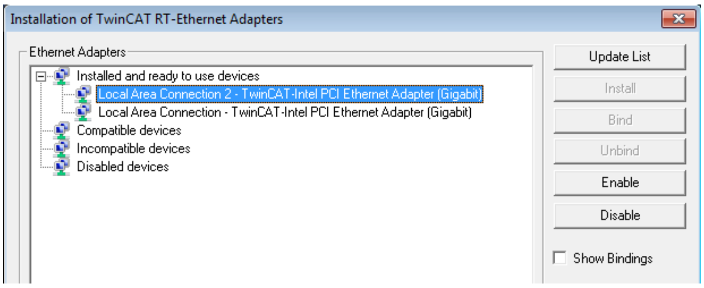
[1] Some drives do not have hardware Axis ID input pins

Start TwinCAT XAE and create a new project. Select the menu command *TwinCAT->Show Real Time Ethernet Compatible Devices…*

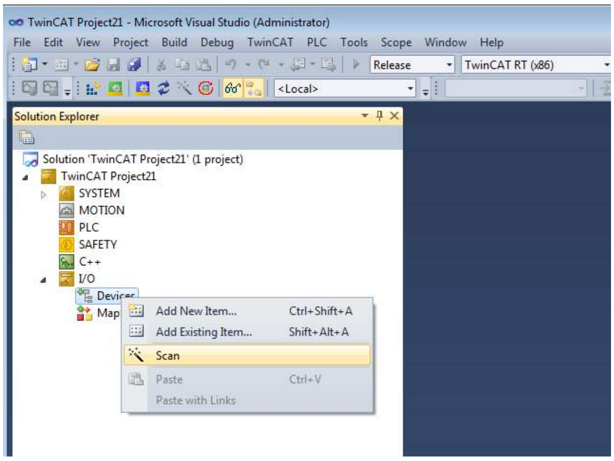


If in *Installed and ready to use devices* no network card is displayed, you must first install one with an EtherCAT® driver. Select one Network Ethernet Card (NIC) from those listed at *Compatible devices* and click on the button *Install.*

***Remark:** In most cases, this operation is done automatically when TwinCAT is installed*

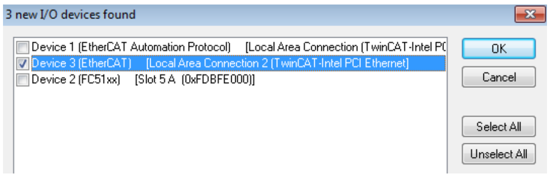


On the left tree of the Solution Explorer, expand the *I/O Configuration,* then click the right mouse button over the *Devices* and choose *Scan.* Confirm the next dialogue with OK.



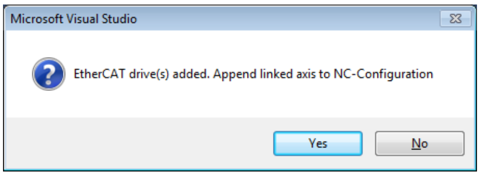
In the list of I/O interfaces found, select only the ones you want to add to the list and click OK.

Remark: if the scan is commanded a few seconds after the drive powers up, it might not be detected. After powering up the drive, wait about 10-20 seconds before scanning for new devices.

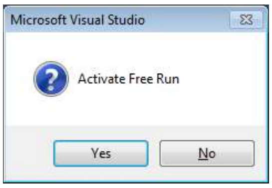


The following dialog appears. Click Yes to confirm:

Click Yes, to add drives to NC-Configuration.



Click Yes to activate Free run in the next dialogue.



Remark: *The Free Run mode was used in the following examples to keep them as simple as possible, without adding the PLC extra layer.*

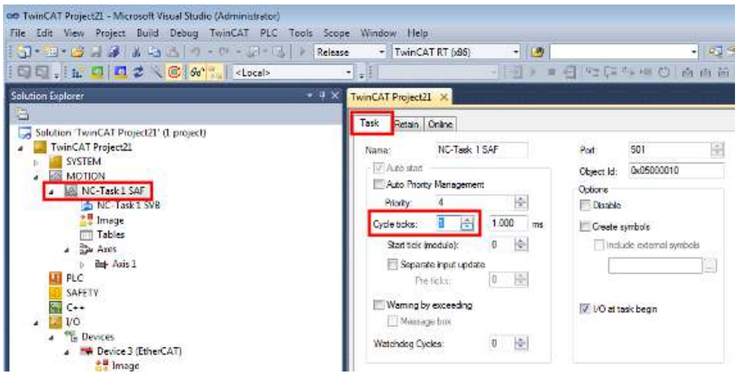
1.5.4 Configuring Technosoft EtherCAT drives for NC PTP compatibility (CSP example)

After the instructions from chapter **1.5.3** are complete, do the following:

1.5.4.1 Setting the communication cycle time for RUN mode

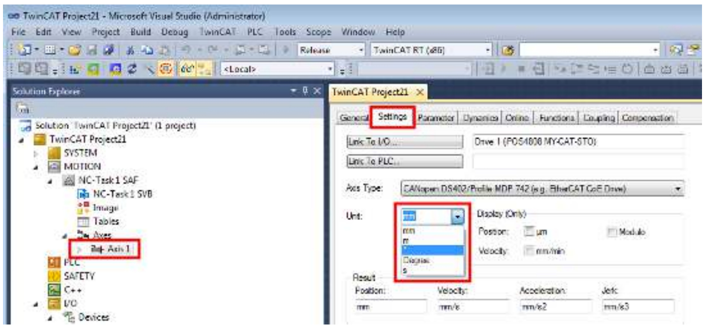
Double-Click the NC-Task to select the communication cycle time. Under the Tab Task, select 1 for cycle ticks to set the communication cycle time of 1 ms (same as the drive slow loop) to obtain the best performance.

Note: If the drive slow loop time is set different than 1 ms, the communication cycle time must be equal or a multiple of that time.



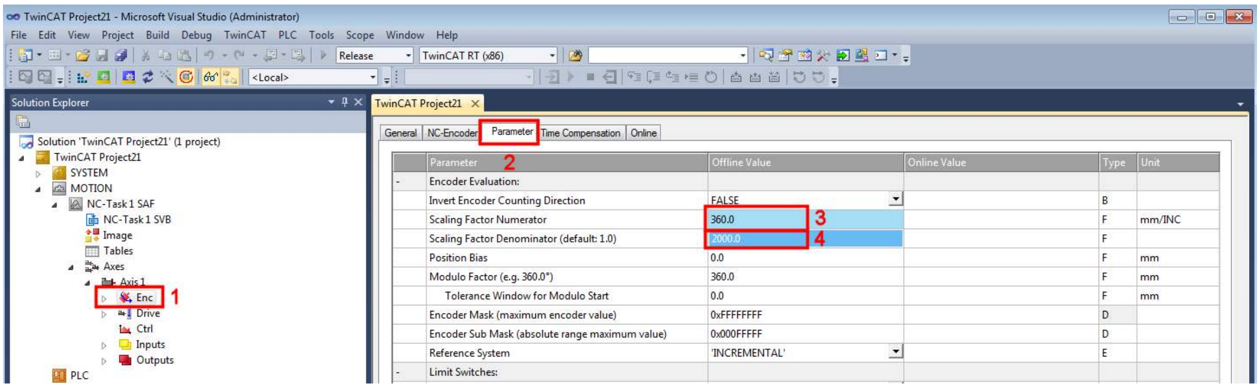
1.5.4.2 Setting the interface factor group settings

Click Axis 1, select the Settings tab and select ◦ if your motor is rotative and mm if it is linear.



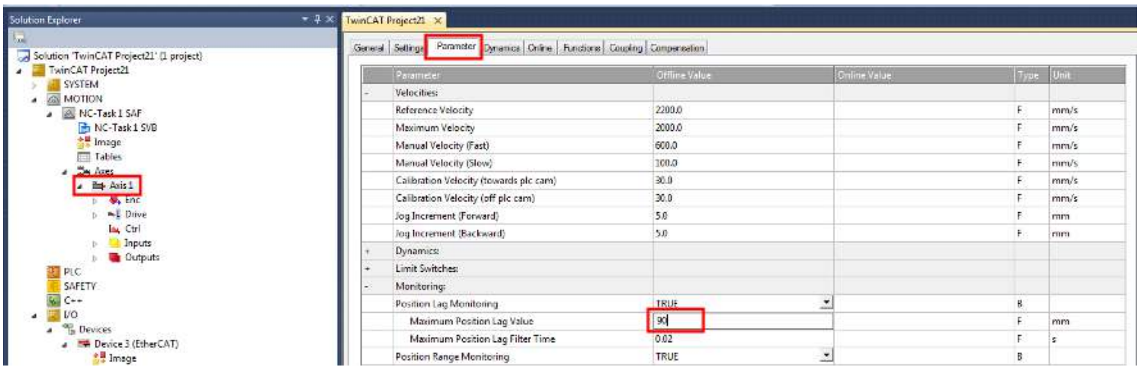
Setting the scaling factor:

Click Axis 1_Enc(1) and select the Parameter(2) tab. Write the scaling factor for your encoder (3). The formula is [360º/ Number of encoder counts of one full motor rotation]. If the encoder is 500 lines quadrature, it will have 2000 encoder counts per rotation. So the scaling factor in this case would be 360/2000 = 0.18.



1.5.4.3 Choosing a position lag value

Double-Click Axis 1 and choose the Parameter tab. Choose a larger number for the Maximum Position Lag Value like 90. This setting is just for demonstration purposes and for the drive not to enter in position control error too fast in case of bad motor tuning.

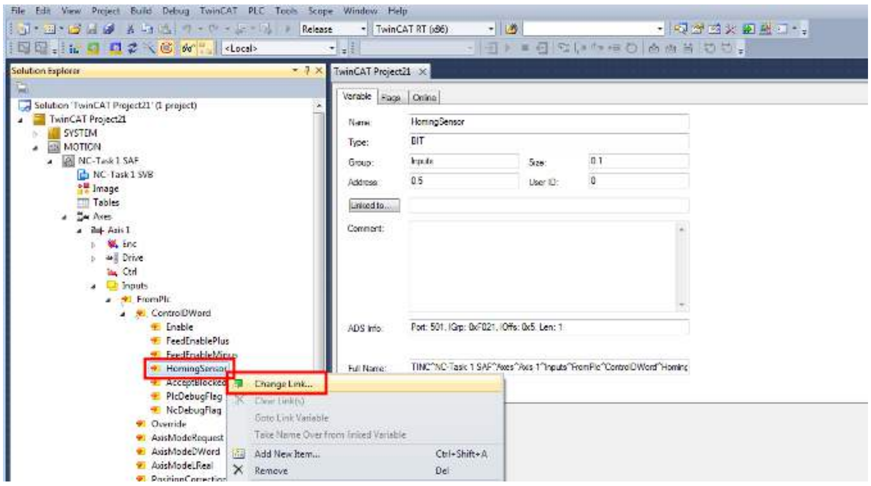


Remark: the position lag protection also exists on all Technosoft drives and can be set with Easy Setup under the "Control error/ Position error" fields.

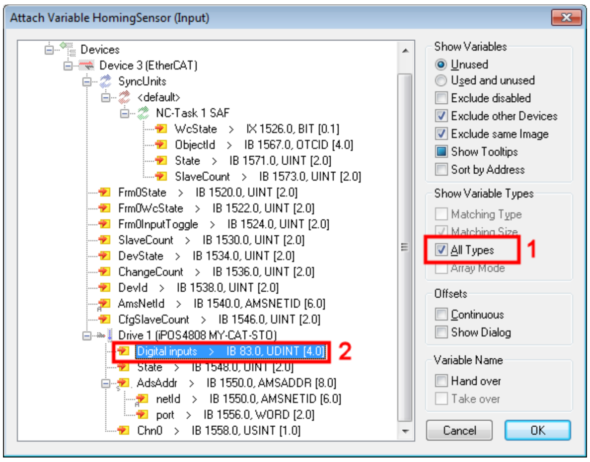
1.5.4.4 Mapping a digital input as the home switch for the NC-PTP interface

In the left tree, under NC – Configuration/Axes/Axis1/Inputs, expand the Axis1_FromPLC, expand ControlDWord and select the variable HomingSensor.

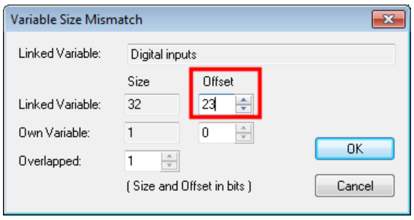
Right click the variable, and select change link… .



Click the checkbox "All Types" (1) and then choose the variable Digital inputs (2). Click OK.

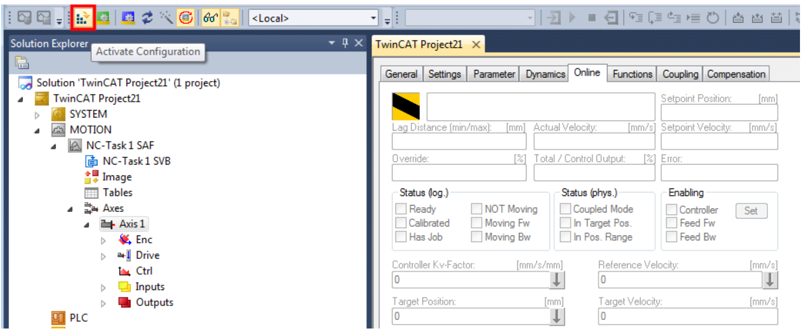


A new menu will appear where the offset will have to be chosen. Digital inputs status is a 32 bit variable and the Homing Sensor is a BOOLEAN (1 bit) variable. Choose an offset of 23 to select the IN0 input and click OK.

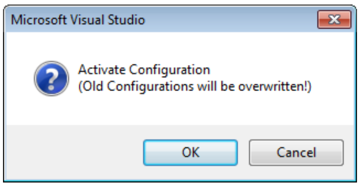


1.5.4.5 Running the NC-PTP interface

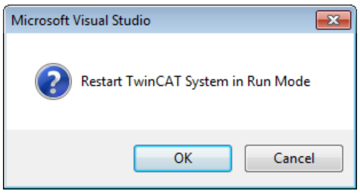
Click the Activate configuration button and to enter PLC Run mode.



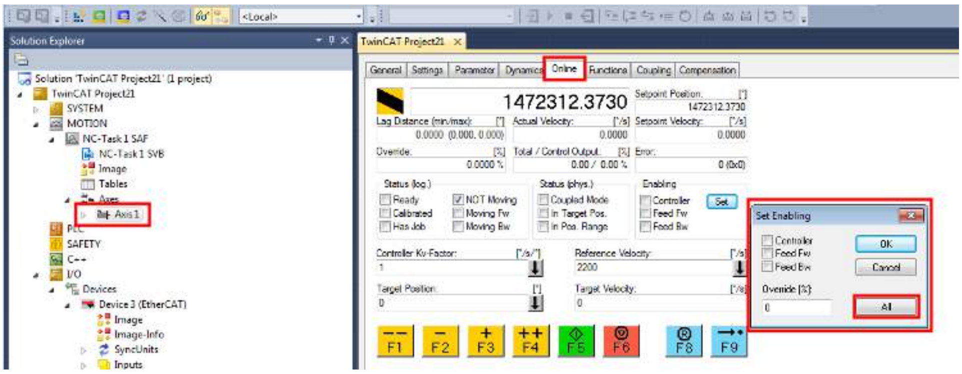
Click OK.



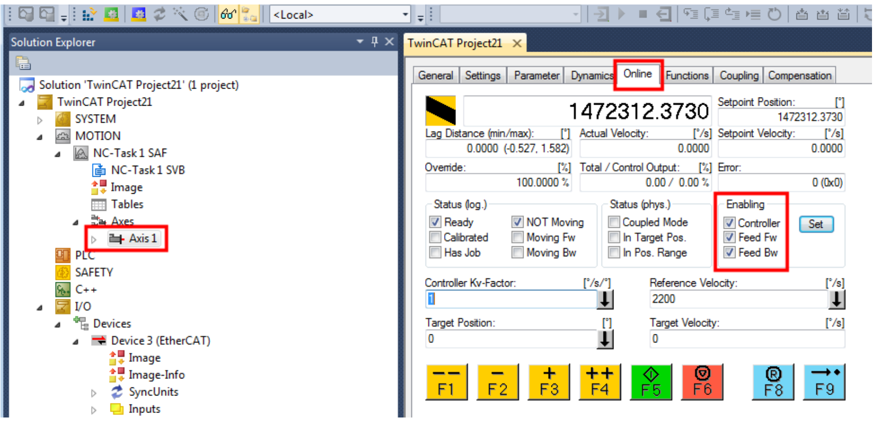
Click OK.



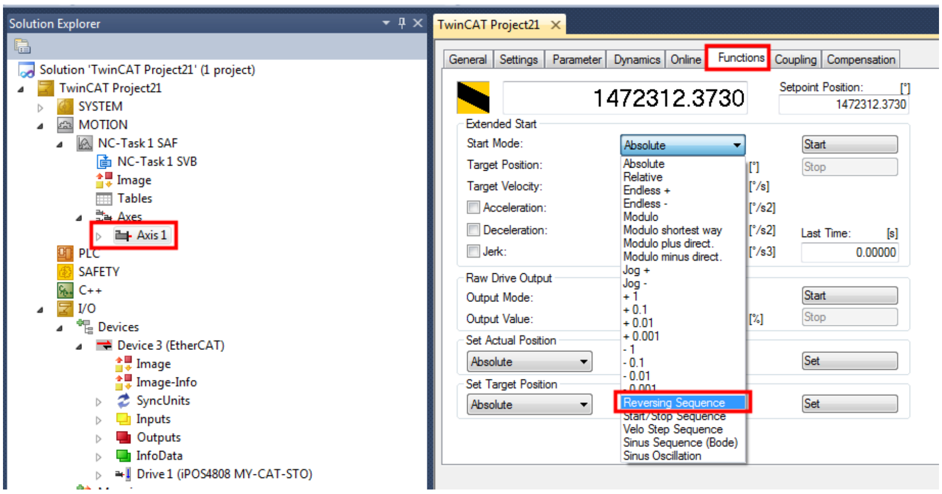
Click Axis 1 and choose the Online tab. Click the Set button and a new window will appear. Click the All button to activate the power to the motor.



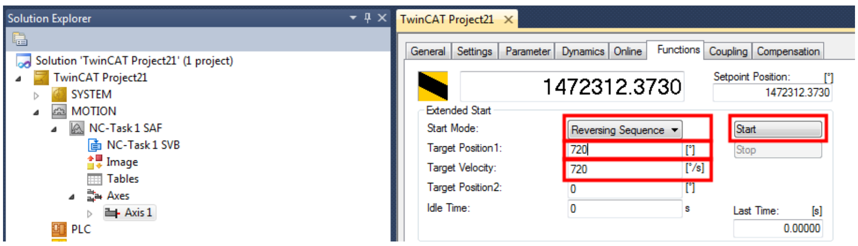
If everything is OK, all the checkboxes under Enabling will be ON. The motor will start keeping its position. For debugging purposes, you can click the yellow F1-F4 buttons to move the motor.



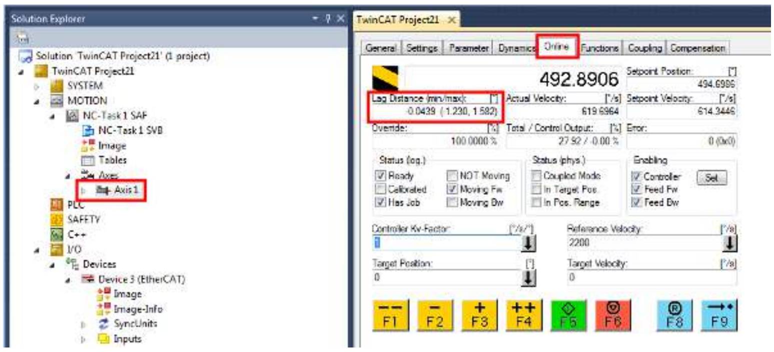
Click the Functions Tab and choose a Reversing Sequence.



Write 720 for target position 1 for the motor to rotate twice. Write 720 ◦/s velocity (2 rps). Click the start button. The motor should start rotating smoothly, without shaking, back and forth 2 rotations.



Click on the Online tab (2) and observe the Lag Distance (position error according to TwinCAT).



After these settings are done, the setup will be compatible with PLC open motion blocks in the PLC Control program to run automated scripts.

1.5.4.6 Checking and updating the XML file stored in the drive

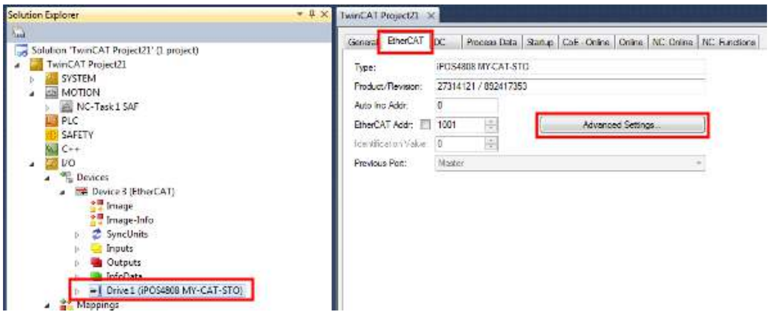
The XML file data is stored in the EtherCAT ASIC EEPROM, not of the drive. The EtherCAT EEPROM can be written only via the Ethernet port. The XML data also contains supported CoE object data information. With firmware updates, new objects might be implemented and an XML update is also needed.

To correctly identify your product, read the label on the drive or read Object 1018_h sub index 2 which shows the correct Product code.

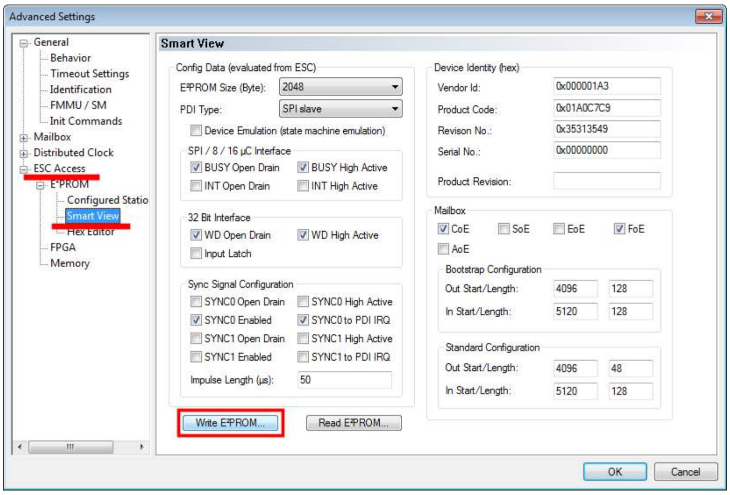
Technosoft Drive name	Product code in 1018_h, sub 02_h
iPOS360x VX-CAT	28002021
iPOS4808 MY-CAT-STO	27314121
iPOS4808 BX-CAT-STO	27314221

To write a new XML file, first make sure that this is present in the right folder i.e. in TwinCAT installation folder \Io\EtherCAT.

The default location is "C:\TwinCAT\Io\EtherCAT" for TwinCAT2 and "C:\TwinCAT\3.1\Config\Io\EtherCAT" for TwinCAT3. In TwinCAT System Manager, if you know the product ID of the drive, select the target drive from the left side menu. Select the *EtherCAT* tab and click *Advanced Settings…*button. See figure below.

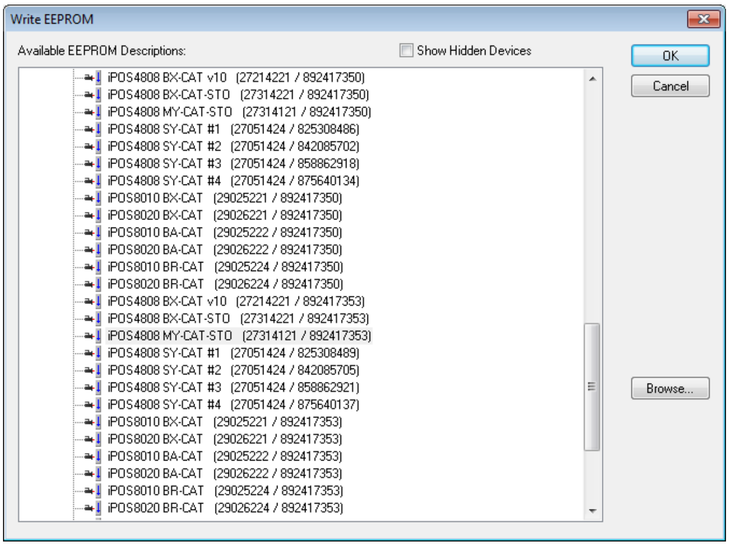


In the Advanced Settings window, select from the left side tree option *ESC access/E²PROM/ Smart View*. Click the *Write E²PROM…* button.



If the XML file is present (see 1.5.1 Adding the XML file), a list of available product descriptions will be available.

The current XML on the drive will be already selected. Choose your correct drive name with the highest revision number (number on the right) and click OK.



The first number represents the drive Product number and the second one represents the revision number. The revision number represents the current firmware version. It can be converted into hex and later into ASCII characters. So, if the current firmware F515I, the revision number will be 515I (ASCII), 0x35313149 (hex) and 892417353 (decimal).

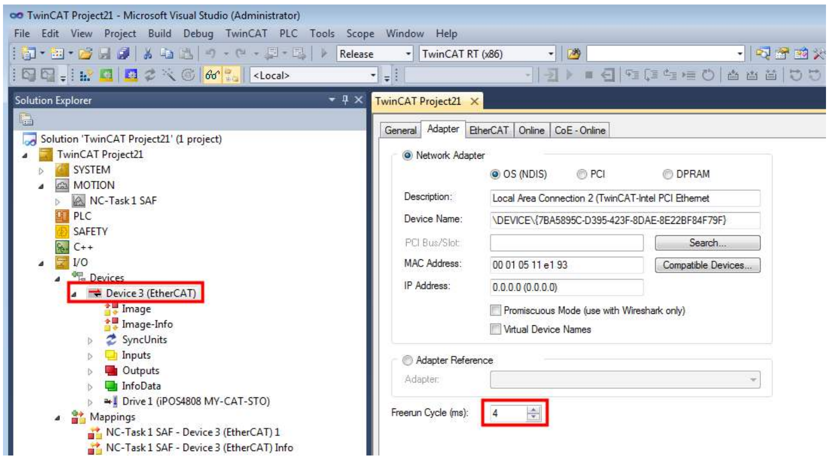
Remark: Newer firmwares work even with older XML file information. There is no need to update the XML info in the ECAT adapter with every firmware update. Sometimes, newer firmwares do not have an associated XML. Example: F515G firmware has no new CoE objects or functionalities, so it works with the XML made for F515F.

After the writing process is complete, reset the drive, and rescan the devices in the network. Now the correct device description will be found automatically.

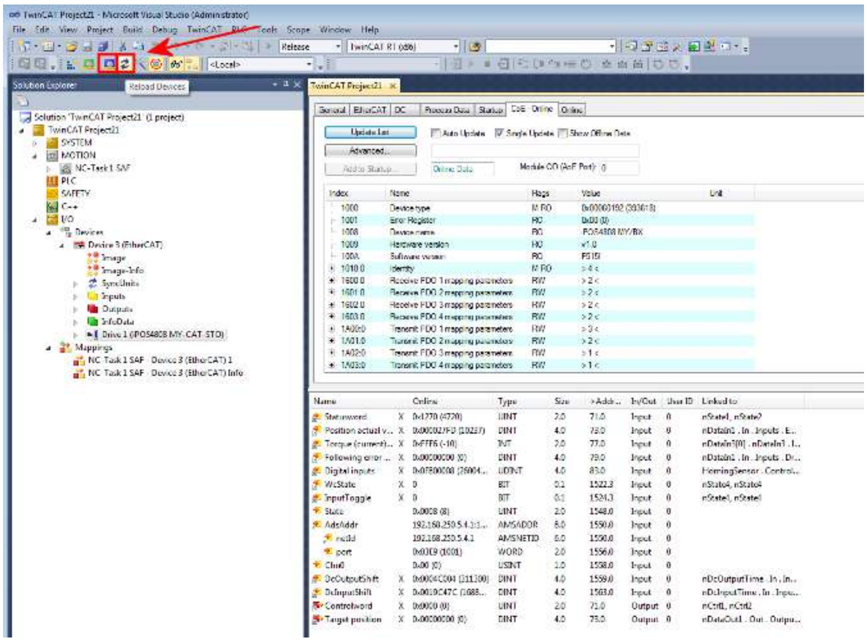
1.5.5 Setting the free run communication cycle

In the left tree, at *I/O devices*, select *Device 3 (EtherCAT)*. On the right side, select *Adapter* tab. Set the *Freerun Cycle* (ms) for example at 1ms.

> ***Remark:** When TwinCAT is operated in Freerun mode, this is the only setting needed for the communication cycle time*



To activate all settings done so far (including PDO mapping changes) click the *Reload I/O Devices (F4)* button shown below. This will also activate SYNC 0 signal if it was enabled.

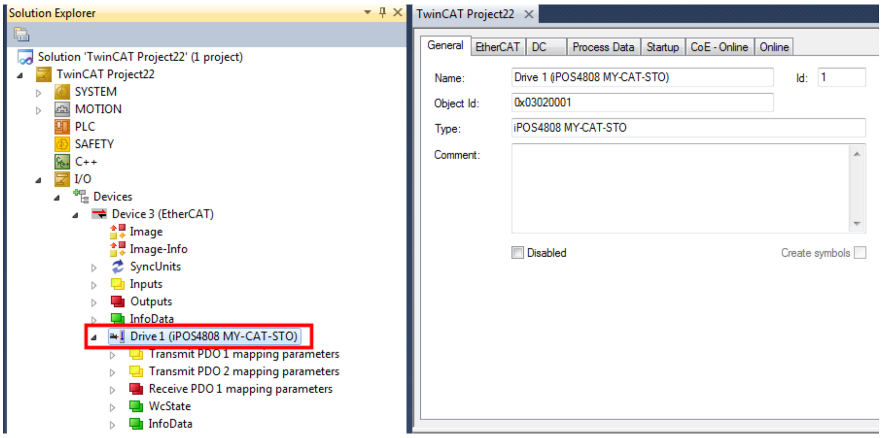


1.6 Controlling the drive using CoE commands. Examples

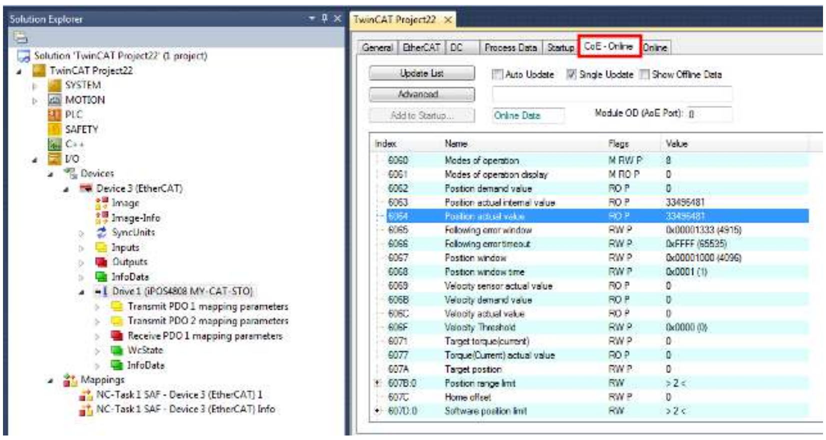
1.6.1 Starting a position profile with CoE commands in TwinCAT

Assuming the motor has been connected to the drive, it has a valid setup downloaded (see **1.1**) and it has been identified in TwinCAT (see **1.5.3)**, the next steps describe a positive trapezoidal motion in position profile mode:

1. In the left tree, at *I/O Devices*, click on Drive 1.



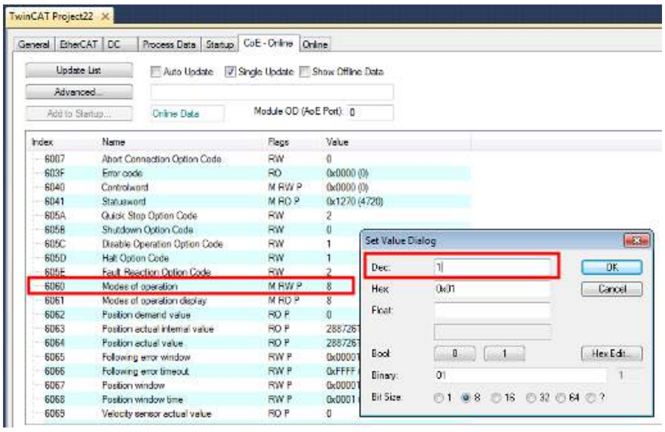
2. Click on *CoE – Online* tab on the right side. If the provided .xml file matches the characteristics of the drive, the CoE objects will have an associated name, like object 0x6064 Position Actual value. Else, all CoE objects will be read directly from the drive without anything in the name column.



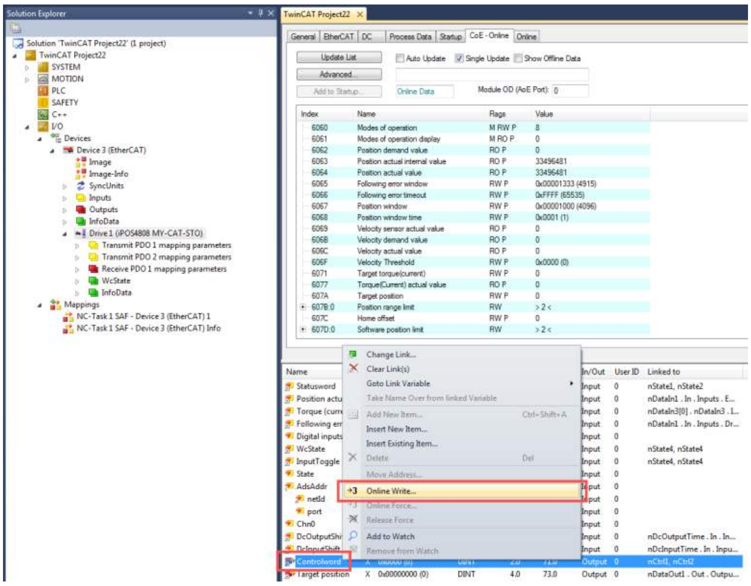
3. In the CoE online list, search for object 6060h and double click it. In the new window, write the value 1.

This step writes with an SDO command into object 6060h the value 1 to set the modes of operation object into position profile.

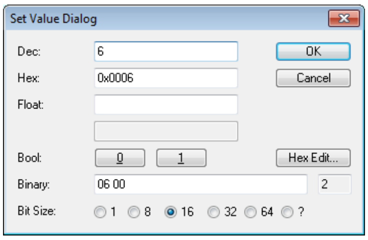
Remark: if the drive is reset, object 6060h will be re-initialized with 8 (CSP mode). This happens because this command is defined in the Startup tab (the one beside the CoE-Online tab) and is added if the Technosoft XML is present.



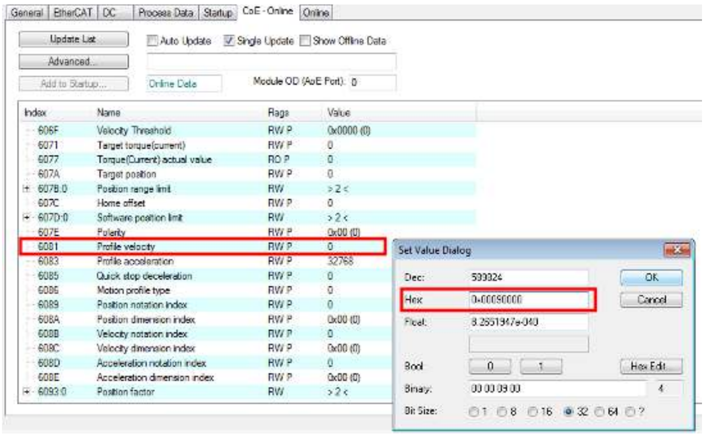
4. Right click on *Control word* variable, and choose *Online Write..* .



5. To enter in **Ready to switch** on state, write in the hex field 0x0006 and click OK.

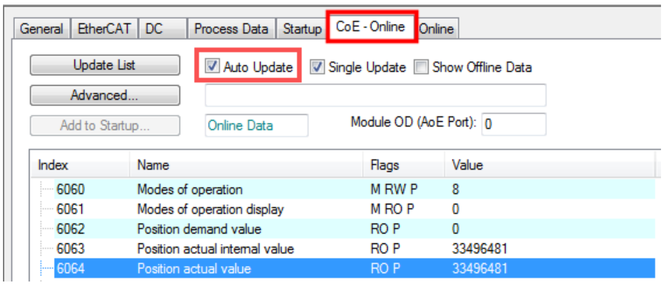


6. Repeat Step 4 and send **Switch** On (0x0007) into *Control word*.
7. Do the same as Step 4 for the PDO *Target position* and write 80000 into Dec field. The drive will finally execute 80000 internal position units (encoder counts) after the start motion command is given.
8. In the CoE object list choose the object with index *0x6081 Profile velocity* and double click on it just like step 3. Write the hex value 0x00090000 in the Hex field and click *OK*. The profile will be executed with a velocity of 9.0 IU (encoder counts)/ms.

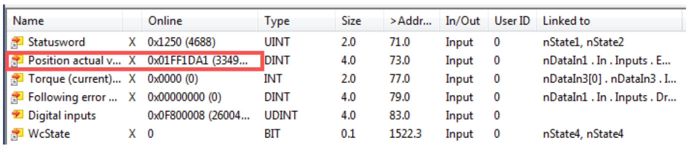


Remark: if the object you need to write is also mapped as a RxPDO, writing in it via SDO protocol is useless because the PDO data comes each communication cycle and overwrites the data in the object. For example, object 607Ah is mapped by default to an RxPDO. Writing in it via the CoE Online list will be overwritten by the PDO value. That is why in Step 7, the PDO data is changed for Target position and not written with SDO protocol.

9. Repeat Step 4 and send **Operation Enable** (0F 00) into *Control word*. After this command, the drive will apply voltage to the motor and will keep its current position.
10. To start the motion, write in *Control word* variable (0x001F) as in Step 4.
11. To follow the actual position of the motor, scroll to the object index *0x6064 Position actual value* and click the check box *Auto Update*. The motor actual value shall update automatically by reading continuously through SDO protocol.



An alternative is to watch the TxPDO value where this object is mapped.



Name		Online	Type	Size	>Addr...	In/Out	User ID	Linked to
Statusword	X	0x1250 (4688)	UINT	2.0	71.0	Input	0	nState1, nState2
Position actual v...	X	0x01FF1DA1 (3349...	DINT	4.0	73.0	Input	0	nDataIn1 . In . Inputs . E...
Torque (current)...	X	0x0000 (0)	INT	2.0	77.0	Input	0	nDataIn3[0] . nDataIn3 . I...
Following error ...	X	0x00000000 (0)	DINT	4.0	79.0	Input	0	nDataIn1 . In . Inputs . Dr...
Digital inputs		0x0F800008 (26004...	UDINT	4.0	83.0	Input	0	
WcState	X	0	BIT	0.1	1522.3	Input	0	nState4, nState4

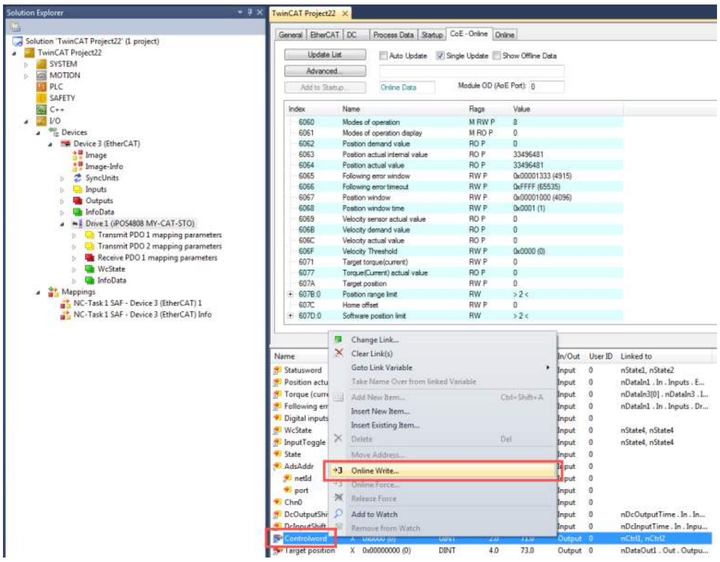
12. After reaching 80000 counts, the motor will stop and hold its position until it receives new motion commands.
13. To issue a new motion command, reset bit 4 of control word to 0. Like in step 4, write the value 0x000F into control word. It was previously set to 0x001F.
14. As in step 7, right click the Target position PDO value and change it from 80000 to 40000.
15. Right click the control word and set it to 0x001F again to start a new motion.

1.6.2 Starting a Cyclic Synchronous Position mode (CSP) (manual commands)

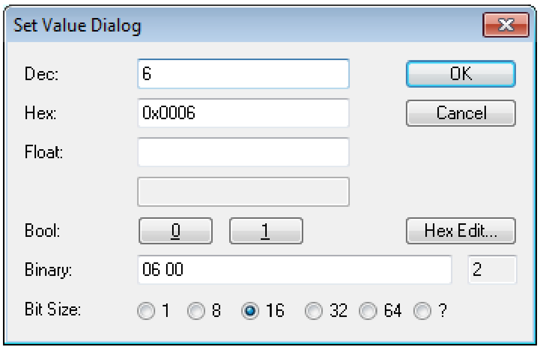
Remark: The cyclic synchronous position mode is the one used by the TwinCAT software in chapter *1.5.4.5 Running the NC-PTP interface*. This example shows the manual steps that are behind this operation mode.

To write in a mapped RPDO variable, right click on the variable and choose *Online Write…*

First Edit the Control Word (which is the variable for object 6040h Controlword)



Write inside the Controlword 06 and then click *OK*.

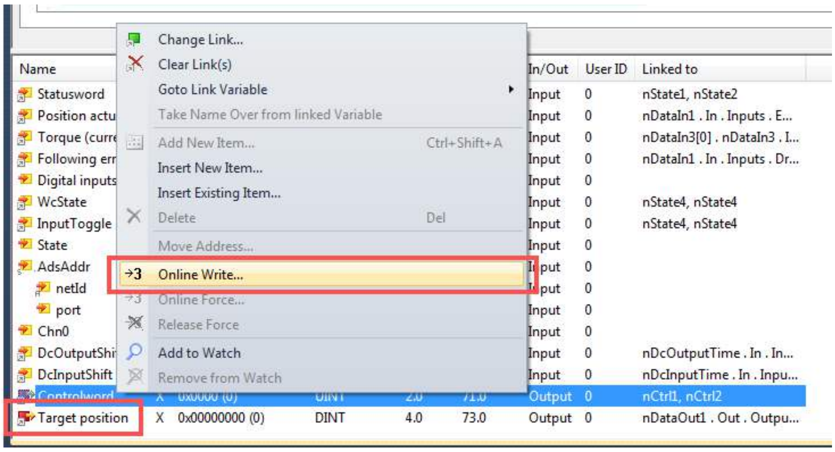


Using the same method, write 07 and then 0x000F. After 0x000F, the drive should be in Operation Enable.

6060h should already be set to the value 08 if the XML file is loaded. If it is not 08, then modify Modes of operation object 6060h value 08 via SDO write. This value sets the drive in Cyclic synchronous position mode.

Finally write a small value into RPDO variable Target position (which is object 607Ah). Maximum 20-200 encoder counts.

The data in the target position will be updated every free run cycle (which was previously set to 1ms). Every time the data inside the Target position changes, the motor will move to that destination within that communication cycle time value.



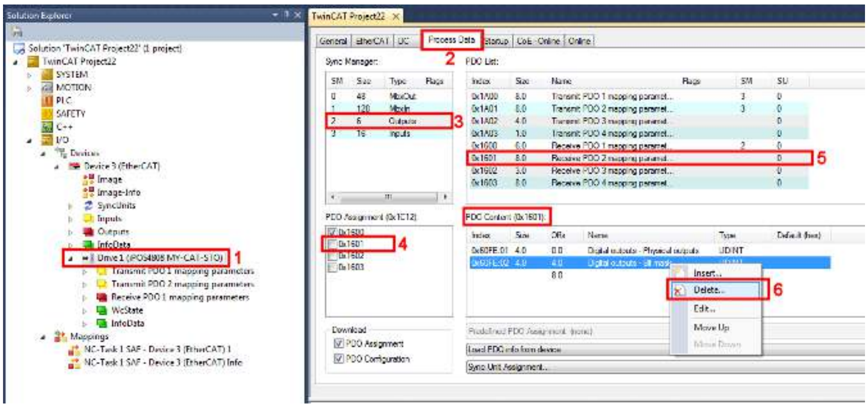
1.6.3 Mapping objects to TxPDOs and RxPDOs in TwinCAT System Manager

1.6.3.1 Mapping objects to RxPDO2

- Set the drive in Config mode. The drive can be with free run mode activated or not.

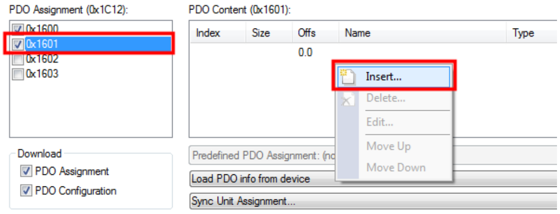
- To map new RxPDO data follow the steps:
  1. Select or double-click Drive 1 on the left.
  2. Select Process Data tab to the upper right of the screen
  3. In Sync Manager, select the Outputs to enable or disable RxPDO1 to 4.
  4. Under PDO Assignment (0x1C12), select which RxPDOs objects should be active. In this case, check 0x1601 to activate RxPDO2.
  5. Under PDO List, select object 0x1601 (Receive PDO 2) by clicking on it.

     Below, in the PDO content window, the default mapped objects are shown.
  6. First clear the existing objects by right click and delete on them.

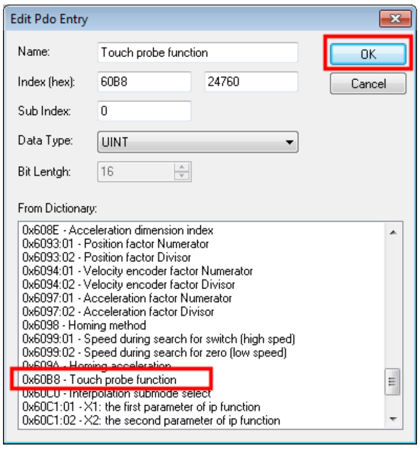


Do this step until all objects are deleted.

- After current objects are deleted from PDO 0x1601, right click on the white space and choose Insert...



- A new list of mappable objects will open. Choose for this example, object 60B8$_h$ – Touch probe function and click Ok.

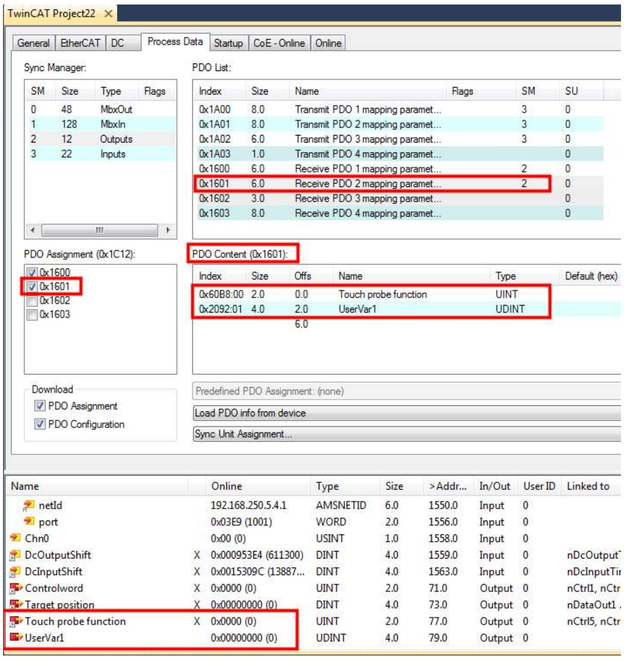


- Repeat the same step as before and add/insert object 2092h UserVar1

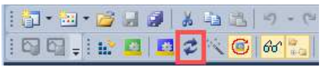
Remark: each RxPDO or TxPDO can support up to 64 bits of data. This means that the sum of the size of all objects mapped into a PDO must not exceed 64 bits. TwinCAT actually allows in the GUI to map more objects, exceeding 64 bits. When the drive will be re-initialized with the new settings, it will send an emergency message that it ignored the PDO mapping.

Remark: the more data is mapped into the PDOs, the more data is transferred to and from the drive each communication cycle. The data transfer time is increased and leads to decreased performance like bad synchronization during CSP mode. It is recommended to map only the data that is used most often and leave one-time configuration objects like 6060$_h$ Modes of operation to be written into only once using SDO protocol.

- Because 0x1601 is checked under PDO Assignment, the newly mapped objects names will be visible in the PDO list below



- To activate the new PDO setup, click Reload I/O devices button.



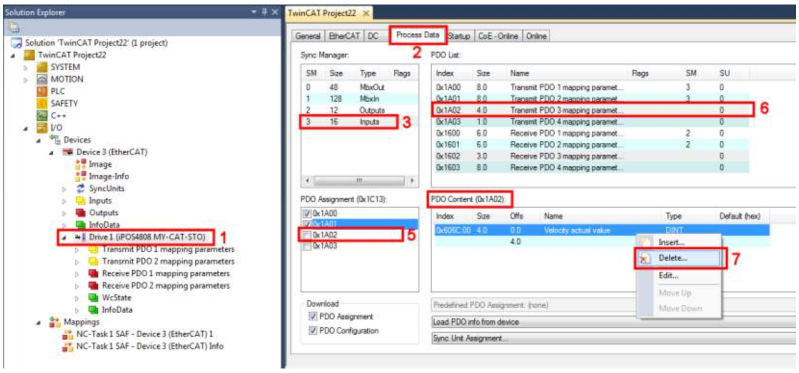
1.6.3.2 Mapping objects to TxPDO3

- Set the drive in Config mode if not already.

- To map new RxPDO data follow the steps:

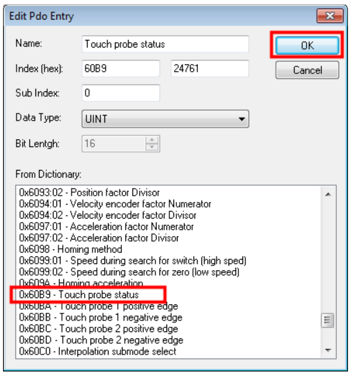
1. Select or double-click Drive 1 in the left.
2. Select Process Data tab to the upper right of the screen
3. In Sync Manager, select the Inputs to enable or disable TxPDO1 to 4.
4. Under PDO Assignment (0x1C13), select which TxPDOs objects should be active. In this case, check 0x1A02 to activate TxPDO3.
5. Under PDO List, select object 0x1A02 (Transmit PDO 3) by clicking on it.

   Below, in the PDO content window, the default mapped objects are shown.
6. First clear the existing objects by right click and delete on them.



Do this step until all objects are deleted.

- After current objects are deleted from PDO 0x1A02, right click on the white space and choose Insert...
- A new list of mappable objects will open. Choose for this example, object 60B9h – Touch probe status and click Ok.

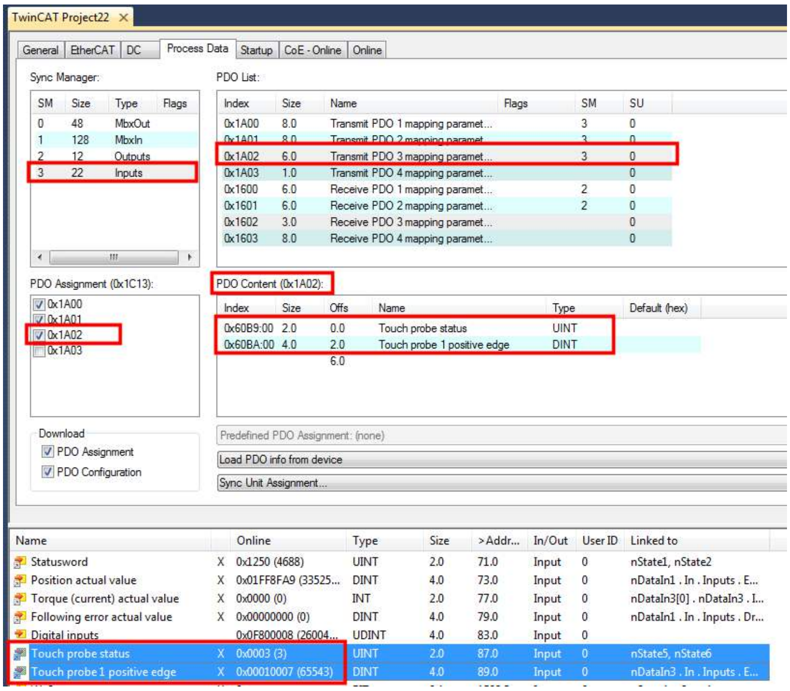


- Repeat the same step as before and add/insert object 0x60BA – Touch probe 1 positive edge

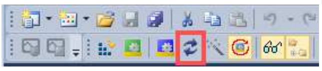
Remark: each RxPDO or TxPDO can support up to 64 bits of data.

Remark: the more data is mapped into the PDOs, the data transfer time is increased and leads to decreased performance.

- Because 0x1A02 is checked under PDO Assignment, the newly mapped objects names will be visible in the PDO list below



- To activate the new PDO setup, click Reload I/O devices button.



2 CAN application protocol over EtherCAT® (CoE protocol)

EtherCAT® (Ethernet for Control Automation Technology) is an open high performance Ethernet-based fieldbus system used in automation control systems. The CAN application protocol over EtherCAT® (CoE) enables the complete CANopen profile family to be used via EtherCAT® and it specifies how various types of devices can use the EtherCAT® network.

2.1 EtherCAT® Architecture

EtherCAT® fieldbus accepts different network topologies, the most commonly used being presented in

Figure 2.1.1.

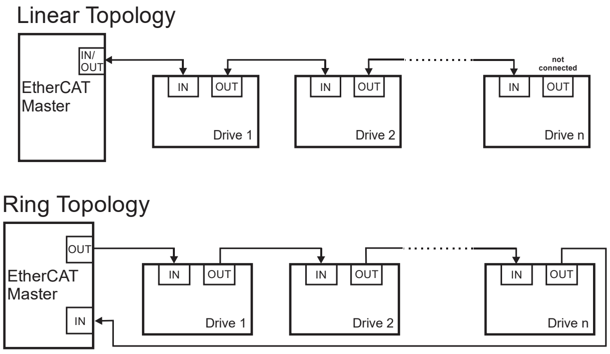


Figure 2.1.1. *EtherCAT® Architecture*

Technosoft has extended the concept of distributed motion application allowing splitting the motion application between the Technosoft drives and the EtherCAT® master. Using TML the user can build complex motion applications locally, on each drive, leaving on the EtherCAT® master only a high level motion application and thus reducing the network master complexity. The master has the vision of the motion application, specific tasks being executed on the Technosoft drives.

2.2 Accessing EtherCAT® devices

An EtherCAT® device is controlled through read/write operations to/from objects performed by an EtherCAT® master.

2.2.1 CoE elements

Table 2.2.1 describes the Mailbox Header and CoE Header.

Table 2.2.1 *– CoE elements*

Frame part	Data Field	Data Type	Value/Description
Mailbox Header	Length	WORD	Length of the Mailbox Service Data
	Address	WORD(32 Bit)	Station Address of the source, if a master is client, Station Address of the destination, if a slave is client
	Channel	Unsigned6	0x00 (Reserved for future)
	Priority	Unsigned2	0x00: lowest priority … 0x03: highest priority
	Type	Unsigned4	0x03: CoE
	Cnt	Unsigned3	Counter of the mailbox services (0 is the start value, next value after 7 is 1)
	Reserved	Unsigned1	0x00
CANopen Header	Number	Unsigned9	Depending on the CANopen service
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x01: Emergency 0x02: SDO Request 0x03: SDO Response 0x04: TxPDO 0x05: RxPDO 0x06: TxPDO remote request 0x07: RxPDO remote request 0x08: SDO Information

2.2.2 Object dictionary

The Object Dictionary is a group of objects that describe the complete functionality of a device by way of communication objects and is the link between the communication interface and the application. All communication objects of a device (application data and configuration parameters) are described in the Object Dictionary in a standardized way.

2.2.3 Object access using index and sub-index

The objects defined for a device are accessed using a 16-bit index and an 8-bit sub-index. In case of arrays and records there is an additional sub-index for each element of the array or record.

2.2.4 Service Data Objects (SDO)

Service Data Objects are used by the EtherCAT® master to access any object from the drive's Object Dictionary.

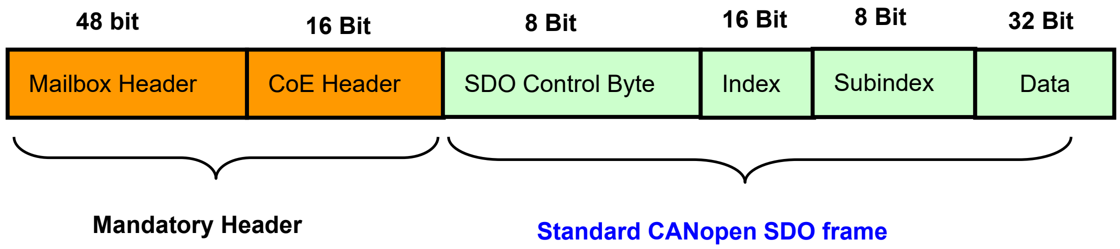


Figure 2.2.1. *EtherCAT®* message SDO structure

Standard CANopen SDO frames can be used:

– Initiate SDO Download

– Download SDO Segment

– Initiate SDO Upload

– Upload SDO Segment

– Abort SDO Transfer

The SDOs are typically used for drive configuration after power-on, for PDO mapping and for infrequent low priority communication.

SDO transfers are confirmed services. In case of an error, an Abort SDO message is transmitted with one of the codes listed below.

Table 2.2.2 *– SDO Abort Codes*

Abort code	Description
0503 0000h	Toggle bit not changed
0504 0000h	SDO protocol timeout
0504 0001h	Client/server command specifier not valid or unknown
0504 0005h	Out of memory
0601 0000h	Unsupported access to an object.
0601 0001h	Attempt to read to a write only object
0601 0002h	Attempt to write to a read only object
0602 0000h	Object does not exist in the object dictionary.
0604 0041h	Object cannot be mapped to the PDO.
0604 0042h	The number and length of the objects to be mapped would exceed PDO length.
0604 0043h	General parameter incompatibility reason.
0604 0047h	General internal incompatibility error in the device.
0606 0000h	Access failed due to a hardware error
0607 0010h	Data type does not match, length of service parameter does not match
0607 0012h	Data type does not match, length of service parameter too high
0607 0013h	Data type does not match, length of service parameter too low
0609 0011h	Sub-index does not exist.
0609 0030h	Value range of parameter exceeded (only for write access).
0609 0031h	Value of parameter written too high.
0609 0032h	Value of parameter written too low.
0609 0036h	Maximum value is less than minimum value
0800 0000h	General error
0800 0020h	Data cannot be transferred or stored to the application.
0800 0021h	Data cannot be transferred or stored to the application because of local control.
0800 0022h	Data cannot be transferred or stored to the application because of the present device state.
0800 0023h	Object dictionary dynamic generation fails or no object dictionary is present

2.2.5 Process Data Objects (PDO)

Process Data Objects are used for high priority, real-time data transfers between the EtherCAT® master and the drives. The PDOs are unconfirmed services and are performed with no protocol overhead. Transmit PDOs are used to send data from the drive, and receive PDOs are used to receive data. The Technosoft drives accept 4 transmit PDOs and 4 receive PDOs. The contents of the PDOs can be set according with the application needs through the dynamic PDO-mapping. This operation can be done during the drive configuration phase using SDOs.

The mapping PDO object contains the descriptions of the objects mapped into the PDO, i.e. the index, sub-index and size of the mapped objects.

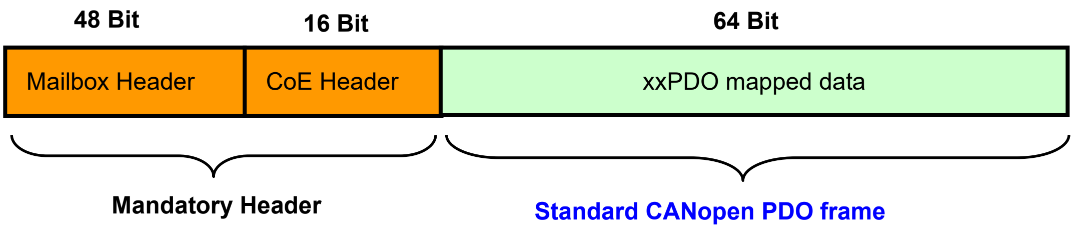


Figure 2.2.2. *EtherCAT®* message PDO structure

2.3 Objects that define SDOs and PDOs

2.3.1 Object 1600_h: Receive PDO1 Mapping Parameters

This object contains the mapping parameters of the receive PDO1. The sub-index 00_h contains the number of valid entries within the mapping record. This number of entries is also the number of the objects that shall be transmitted/received with the corresponding PDO. The sub-indices from 01_h to the number of entries contain the information about the mapped objects. These entries describe the PDO contents by their index, sub-index and length. The length entry contains the length of the mapped object in bits and is used to verify the overall mapping length.

The structure of the entries from sub-index 01_h to the number of entries is as follows:

MSB		LSB
Index (16 bits)	Sub-index (8 bits)	Object length (8 bits)

In order to change the PDO mapping, first the PDO has to be disabled - the object $160x_h$ sub-index 00_h has to be set to 0. Now the objects can be remapped. If a wrong mapping parameter is introduced (object does not exist, the object cannot be mapped or wrong mapping length is detected) the SDO transfer will be aborted with an appropriate error code ($0602\ 0000_h$ or $0604\ 0041_h$). After all objects are mapped, sub-index 00_h has to be set to the valid number of mapped objects thus enabling the PDO.

If data types (index 01_h - 07_h) are mapped, they serve as “dummy entries”. The corresponding data is not evaluated by the drive. This feature can be used to transmit data to several drives using only one PDO, each drive using only a part of the PDO. This feature is only valid for receive PDOs.

Object description:

Index	1600_h
Name	RPDO1 Mapping Parameters
Object code	RECORD
Data type	PDO Mapping

Entry description:

Sub-index	00_h
Description	Number of mapped objects
Access	RW
PDO mapping	No
Value range	0: Mapping disabled 1 – 64: Sub-index 1 to x is valid
Default value	2

Sub-index	01_h
Description	1st mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60400010_h – Controlword

© Technosoft 2024

Sub-index	02h
Description	2nd mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	607A0020h – Target position

2.3.2 Object 1601h: Receive PDO2 Mapping Parameters

This object contains the mapping parameters of the receive PDO2. The sub-index 00h contains the number of valid entries within the mapping record. This number of entries is also the number of the objects that shall be transmitted/received with the corresponding PDO.

Object description:

Index	1601h
Name	RPDO2 Mapping Parameter
Object code	RECORD
Data type	PDO Mapping

Entry description:

Sub-index	00h
Description	Number of mapped objects
Access	RW
PDO mapping	No
Value range	0: Mapping disabled 1 – 64: Sub-index 1 to x is valid
Default value	2

Sub-index	01h
Description	1st mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60FE0120h – Digital outputs - Physical outputs

Sub-index	02h
Description	2nd mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60FE0220h – Digital outputs - Bit mask

2.3.3 Object 1602h: Receive PDO3 Mapping Parameters

This object contains the mapping parameters of the receive PDO3. The sub-index 00h contains the number of valid entries within the mapping record. This number of entries is also the number of the objects that shall be transmitted/received with the corresponding PDO.

Object description:

Index	1602h
Name	RPDO3 Mapping Parameter
Object code	RECORD
Data type	PDO Mapping

Entry description:

Sub-index	00h
Description	Number of mapped objects
Access	RW
PDO mapping	No
Value range	0: Mapping disabled 1 – 64: Sub-index 1 to x is valid
Default value	2

Sub-index	01h
Description	1st mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60600008h – Modes of operation

Sub-index	02h
Description	2nd mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60710010h – Target torque

2.3.4 Object 1603h: Receive PDO4 Mapping Parameters

This object contains the mapping parameters of the receive PDO4. The sub-index 00h contains the number of valid entries within the mapping record. This number of entries is also the number of the objects that shall be transmitted/received with the corresponding PDO.

Object description:

Index	1603h
Name	RPDO4 Mapping Parameters
Object code	RECORD
Data type	PDO Mapping

Entry description:

Sub-index	00h
Description	Number of mapped objects
Access	RW
PDO mapping	No
Value range	0: Mapping disabled 1 – 64: Sub-index 1 to x is valid
Default value	2

Sub-index	01h
Description	1st mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60810020h – Profile velocity

Sub-index	02h
Description	2nd mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60830020h – Profile acceleration

2.3.5 Object 1A00h: Transmit PDO1 Mapping Parameters

This object contains the mapping parameters of the transmit PDO1. For detailed description see object 1600h (Receive PDO1 mapping parameters)

Object description:

Index	1A00h
Name	TPDO1 Mapping Parameters
Object code	RECORD
Data type	PDO Mapping

Entry description:

Sub-index	00h
Description	Number of mapped objects
Access	RW
PDO mapping	No
Value range	0: Mapping disabled 1 – 64: Sub-index 1 to x is valid
Default value	3

Sub-index	01h
Description	1st mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60410010h – Statusword

Sub-index	02h
Description	2nd mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60640020h – Position actual value

Sub-index	03h
Description	3rd mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60770010h – Torque actual value

2.3.6 Object 1A01h: Transmit PDO2 Mapping Parameters

This object contains the mapping parameters of the transmit PDO2. For detailed description see object 1600h (Receive PDO1 mapping parameters)

Object description:

Index	1A01h
Name	TPDO2 Mapping Parameter
Object code	RECORD
Data type	PDO Mapping

Entry description:

Sub-index	00h
Description	Number of mapped objects
Access	RW
PDO mapping	No
Value range	0: Mapping disabled 1 – 64: Sub-index 1 to x is valid
Default value	2

Sub-index	01h
Description	1st mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60F40020h – Following error actual value

Sub-index	02h
Description	2nd mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60FD0020h – Digital inputs

2.3.7 Object 1A02h: Transmit PDO3 Mapping Parameters

This object contains the mapping parameters of the transmit PDO3. For detailed description see object 1600h (Receive PDO1 mapping parameters). By default, this PDO is disabled with object 1802h Sub-index 01 by setting Bit31 to 1.

Object description:

Index	1A02h
Name	TPDO3 Mapping Parameter
Object code	RECORD
Data type	PDO Mapping

Entry description:

Sub-index	00h
Description	Number of entries
Access	RW
PDO mapping	No
Value range	0: Mapping disabled 1 – 64: Sub-index 1 to x is valid
Default value	1

Sub-index	01h
Description	1st mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	606C0020h – Velocity actual value

2.3.8 Object 1A03h: Transmit PDO4 Mapping Parameters

This object contains the mapping parameters of the transmit PDO4. For detailed description see object 1600h (Receive PDO1 mapping parameters). By default, this PDO is disabled with object 1803h Sub-index 01 by setting Bit31 to 1.

Object description:

Index	1A03h
Name	TPDO4 Mapping Parameter
Object code	RECORD
Data type	PDO Mapping

Entry description:

Sub-index	00h
Description	Number of entries
Access	RW
PDO mapping	No
Value range	0: Mapping disabled 1 – 64: Sub-index 1 to x is valid
Default value	1

Sub-index	01h
Description	1st mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60610008h – Modes of operation display

2.3.9 Object 1C00h: Sync Manager Communication type

Object description:

Index	1C00h
Name	Sync Manager Com. type
Object code	ARRAY
Data type	UNSIGNED 8

Entry description:

Sub-index	00h
Description	Number of entries
Access	RO
PDO mapping	No
Default value	4

Sub-index	01h
Description	Communication Type Sync Manager 0
Access	RO
PDO mapping	No
Value range	UNSIGNED8 1 : mailbox receive (master to slave)
Default value	1

Sub-index	02h
Description	Communication Type Sync Manager 1
Access	RO
PDO mapping	No
Value range	UNSIGNED8 2 : mailbox send (slave to master)
Default value	2

Sub-index	03h
Description	Communication Type Sync Manager 2
Access	RO
PDO mapping	No
Value range	UNSIGNED8 0: unused 3 : process data output (master to slave)
Default value	3

Sub-index	04h
Description	Communication Type Sync Manager 3
Access	RO
PDO mapping	No
Value range	UNSIGNED8 0: unused 1 : mailbox receive (master to slave) 2 : mailbox send (slave to master) 3 : process data output (master to slave) 4: process data input (slave to master)
Default value	3

2.3.10 Object 1C12h: Sync Manager Channel 2 (Process Data Output)

Assigns the RxPDO. This object can be modified only when State Machine is in Pre Operational and Subindex 0 = 0. After configuration is done, set in Subindex 0 the number of configured RxPDOs.

Object description:

Index	1C12h
Name	Sync Manager Channel 2
Object code	ARRAY
Data type	UNSIGNED8

Entry description:

Sub-index	00h
Description	Number of entries
Access	RO
PDO mapping	No
Value Range	UNSIGNED8 1-4
Default value	1

Sub-index	01h
Description	PDO Mapping object index of assigned RxPDO : 1st mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED16 0x1600 (RxPDO1) 0x1601 (RxPDO2) 0x1602 (RxPDO3) 0x1603 (RxPDO4)
Default value	0x1600 (RxPDO1)

2.3.11 Object 1C13h: Sync Manager Channel 3 (Process Data Input)

Assigns the TxPDO. This object can be modified only when State Machine is in Pre Operational and Subindex 0 = 0. After configuration is done, set in Subindex 0 the number of configured TxPDOs.

Object description:

Index	1C13h
Name	Sync Manager Channel 3
Object code	ARRAY
Data type	UNSIGNED8

Entry description:

Sub-index	00h
Description	Number of entries
Access	RO
PDO mapping	No
Value Range	UNSIGNED8 0-4
Default value	2

Sub-index	01h
Description	PDO Mapping object index of assigned TxPDO : 1st mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED16 0x1A00 – 0x1A03 (TxPDO1 - TxPDO4)
Default value	0x1A00 (TxPDO1)

Sub-index	02h
Description	PDO Mapping object index of assigned TxPDO : 2nd mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED16 0x1A00 – 0x1A03 (TxPDO1 - TxPDO4)
Default value	0x1A01 (TxPDO2)

2.3.12 Object 1C32h: Output Sync Manager Parameter

This object gives the specifications of the EtherCAT communications mode for Sync Manager for output process data.

Object description:

Index	1C32h
Name	Output Sync Manager Parameter
Object code	RECORD
Data type	UNSIGNED8

Entry description:

Sub-index	00h
Description	Number of entries
Access	RO
PDO mapping	No
Value Range	UNSIGNED8 0-32
Default value	32

Sub-index	01h	
Description	Synchronization Type	
Access	RO	
PDO mapping	No	
Value range	UNSIGNED16	
Default value	0x0002	
Value description	Value	Current synchronization mode:
	0	Free Run
	2	DC-Mode - Synchronization with SYNC0 Event

Sub-index	02h	
Description	Cycle Time	
Access	RO	
PDO mapping	No	
Value range	UNSIGNED32	
Default value	0x000F4240	
Value description	Free Run	Cycle time of the local timer (ns)
	DC-Mode	SYNC0 Cycle Time (ns)

Sub-index	04h		
Description	Synchronization Types supported		
Access	RO		
PDO mapping	No		
Value range	UNSIGNED16		
Default value	0x0005		
Value description	Bit	Value	Supported synchronization modes:
	0	1	Free run is supported
	2:4	001	DC SYNC0 mode is supported
	5:6	00	No Output Shift supported

Sub-index	05h
Description	Minimum Cycle Time
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	0x00000000
Value description	Gives the minimum Cycle Time that is supported by the slave (ns).

Sub-index	06h
Description	Calc and Copy Time
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	0x00000000
Value description	Minimum time between SYNC0 and SYNC1 event (ns, DC mode only)

Sub-index	09h
Description	Delay Time
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	0x00000000
Value description	Time between SYNC1 event and output of the outputs (ns, DC mode only)

Sub-index	0Bh
Description	SM-Event Missed Counter
Access	RO
PDO mapping	No
Value range	UNSIGNED16
Default value	0x00000000
Value description	The error counter increases whenever a SM event is expected but does not arrive within the expected time frame.

Sub-index	0Ch
Description	Cycle Time Too Small Counter
Access	RO
PDO mapping	No
Value range	UNSIGNED16
Default value	0x00000000
Value description	The error counter increases when the cycle time is too small so the local cycle was not completed in time.

Sub-index	20h	
Description	Sync Error	
Access	RO	
PDO mapping	No	
Value range	BOOL	
Default value	FALSE	
Value description	The synchronization was not correct in the last cycle	
	FALSE	No Synchronization error in the last cycle
	TRUE	Synchronization error in the last cycle

If the communication cycle time is configured to be less than the "Minimum Cycle Time" (specified in sub-index 5h), the system will detect an error. Specifically:

1. The Sync Error flag (sub-index 20h) will be set to TRUE, indicating a synchronization issue.
2. The Cycle Time Too Small Counter (sub-index Ch) will increment, tracking the number of occurrences of this issue.
3. As a result of this error, the EtherCAT state machine will not transition to the Safe-Operational state, preventing the system from operating.

2.3.13 Object 1C33h: Input Sync Manager Parameter

This object gives the specifications of the EtherCAT communications mode for Sync Manager for input process data.

Object description:

Index	1C33h
Name	Input Sync Manager Parameter
Object code	RECORD
Data type	UNSIGNED8

Entry description:

Sub-index	00_h
Description	Number of entries
Access	RO
PDO mapping	No
Value Range	UNSIGNED8 0-32
Default value	32

Sub-index	01_h		
Description	Synchronization Type		
Access	RO		
PDO mapping	No		
Value range	UNSIGNED16		
Default value	0x0002		
Value description	Value	Current synchronization mode:	
	0	Free Run	
	2	DC-Mode - Synchronized with SYNC0 Event	

Sub-index	02_h	
Description	Cycle Time	
Access	RO	
PDO mapping	No	
Value range	UNSIGNED32	
Default value	0x000F4240	
Value description	Free Run	Cycle time of the local timer (ns)
	DC-Mode	SYNC0 Cycle Time (ns)

Sub-index	04_h		
Description	Synchronization Types supported		
Access	RO		
PDO mapping	No		
Value range	UNSIGNED16		
Default value	0x0005		
Value description	Bit	Value	Supported synchronization modes:
	0	1	Free run is supported
	2:4	001	DC SYNC0 mode is supported
	5:6	00	No Output Shift supported

Sub-index	05_h
Description	Minimum Cycle Time
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	0x00000000
Value description	Gives the minimum Cycle Time that is supported by the slave (ns).

Sub-index	06_h
Description	Calc and Copy Time
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	0x00000000
Value description	Minimum time between SYNC0 and SYNC1 event (ns, DC mode only)

Sub-index	09_h
Description	Delay Time
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	0x00000000
Value description	Time between SYNC1 event and output of the outputs (ns, DC mode only)

Sub-index	$0B_h$
Description	SM-Event Missed Counter
Access	RO
PDO mapping	No
Value range	UNSIGNED16
Default value	0x00000000
Value description	The error counter increases whenever a SM event is expected but does not arrive within the expected time frame.

Sub-index	$0C_h$
Description	Cycle Time Too Small Counter
Access	RO
PDO mapping	No
Value range	UNSIGNED16
Default value	0x00000000
Value description	The error counter increases when the cycle time is too small so the local cycle was not completed in time.

Sub-index	20_h	
Description	Sync Error	
Access	RO	
PDO mapping	No	
Value range	BOOL	
Default value	FALSE	
Value description	The synchronization was not correct in the last cycle	
	FALSE	No Synchronization error in the last cycle
	TRUE	Synchronization error in the last cycle

If the communication cycle time is configured to be less than the "Minimum Cycle Time" (specified in sub-index 5_h), the system will detect an error. Specifically:

1. The Sync Error flag (sub-index 20_h) will be set to TRUE, indicating a synchronization issue.
2. The Cycle Time Too Small Counter (sub-index C_h) will increment, tracking the number of occurrences of this issue.
3. As a result of this error, the EtherCAT state machine will not transition to the Safe-Operational state, preventing the system from operating.

2.3.14 Object $207D_h$: Dummy

This object may be used to fill a RPDO up to a length matching the EtherCAT® master requirements.

Object description:

Index	$207D_h$
Name	Dummy
Object code	VAR
Data type	UNSIGNED8

Entry description:

Access	RW
PDO mapping	Possible
Value range	0 … 255
Default value	0

2.4 PDOs mapping general example

Follow the next steps to change the default mapping of a PDO:

1. **Set the drive into Pre-Operational.**
2. **Disable the Sync Manager Channels 2 and 3** for the RxPDOs and the TxPDOs: write 0 in sub-index 0 of objects $1C12_h$ and $1C13_h$
3. **Disable the PDO**. In the PDO's mapping object (index 1600_h-1603_h for RxPDOs and $1A00_h$-$1A03_h$ for TxPDOs) set the first sub-index (the number of mapped objects) to 0. This will disable the PDO.
4. **Map the new objects**. Write in the PDO's mapping object (index 1600_h-1603_h for RxPDOs and $1A00_h$-$1A03_h$ for TxPDOs) sub-indexes (1-8) the description of the objects that will be mapped.
5. **Enable the PDO**. In sub-index 0 of the PDO's associated mapping object (index 1600_h-1603_h for RxPDOs and $1A00_h$-$1A03_h$ for TxPDOs) write the number of mapped objects.
6. **Map the enabled PDOs intro Sync Manager Channels 2 and 3**. Set in sub-index 1-4 the hex value of the enabled PDOs ($1A00_h$-$1A03_h$ for TxPDOs in $1C13_h$ and 1600_h-1603_h for RxPDOs in $1C12_h$).
7. **Enable the Sync Manager channels.** Set in their sub-index 0 the number of enabled Tx/Rx PDOs.
8. **Set the drive into Operational** state.

2.5 PDOs size

Our drives support the configuration of up to 8 PDOs in total. The size of these PDOs depends on the firmware version of the drive:

- Firmware Version **F515K**
  - RPDO1/TPDO1: 32 bytes (256 bits)
  - RPDO2/TPDO2, RPDO3/TPDO3, RPDO4/TPDO4: 8 bytes (64 bits each)
- Firmware Version **F515J or Earlier**
  - RPDO1/TPDO1, RPDO2/TPDO2, RPDO3/TPDO3, RPDO4/TPDO4: 8 bytes (64 bits each)
- Firmware Versions **FA00x or FA02x**
  - RPDO1/TPDO1: 64 bytes (512 bits)
  - RPDO2/TPDO2, RPDO3/TPDO3, RPDO4/TPDO4: 8 bytes (64 bits each)

2.6 RxPDOs mapping example

Remark: for a TwinCAT example about mapping RxPDOs see paragraph *1.6.3.1.*

Map the receive PDO3 with **ControlWord** (index 6040_h) and **Modes of Operation** (index 6060_h).

1. **Set the drive to Pre-Operational.**
2. **Disable the Sync manager with RxPDOs**. Write zero in object $1C12_h$ sub-index 0, this will disable the PDO.
3. **Disable RxPDO3 mapping PDO**. Write 0 in object 1602_h sub-index 0, this will disable the PDO's mapping.
   Send the following message: SDO access to object 1602_h sub-index 0, 8-bit value 0.
4. **Map the new objects**.
   a. Write in object 1602_h sub-index 1 the description of the Controlword:

Index	Sub-index	Length	Resulting data
6040_h	**00_h**	**10_h**	**60400010_h**

   Send the following message: SDO access to object 1602_h sub-index 1, 32-bit value 60400010_h.
   b. Write in object 1602_h sub-index 2 the description of the Modes of Operation:

Index	Sub-index	Length	Resulting data
6060_h	**00_h**	**08_h**	**60600008_h**

   Send the following message: SDO access to object 1602_h sub-index 2, 32-bit value 60600008_h.
5. **Enable the RxPDO3 mapped objects**. Set the object 1602_h sub-index 0 with the value 2 to enable both mapped objects.
   Send the following message: SDO access to object 1602_h sub-index 0, 8-bit value 2.
6. **Add the new TPDO to the Sync Manager**.
   -Write in object $1C12_h$ sub-index 3 the RPDO3 mapping parameter object number:
   Send the following message: SDO access to object $1C12_h$ sub-index 3, 16-bit value 1602_h.
   -Write 03_h in object $1C12_h$ sub-index 0, this will enable the Sync. Manager.
   Send the following message: SDO access to object $1C12_h$ sub-index 0, 8-bit value 03_h.
   Note: if using TwinCAT System Manager, enter in Configuration Mode, select the drive, select Process Data tab, uncheck the PDO Assignment and PDO Configuration boxes. Click Load PDO info from device button to load the new PDO configuration. Reload the IO devices and enter in Operation state.
7. **Set the drive to Safe-Operational**.
8. **Set the drive to Operational.**

2.7 TxPDOs mapping example

Remark: for a TwinCAT example about mapping TxPDOs see paragraph *1.6.3.2.*

Map the transmit PDO4 with **Position actual value** (index 6064_h) and **Digital inputs** (index $60FD_h$).

1. **Set the drive to Pre-Operational.**
2. **Disable the Sync manager with TxPDOs**. Write zero in object $1C13_h$ sub-index 0, this will disable the PDO.
3. **Disable RxPDO4 mapping PDO**. Write 0 in object $1A03_h$ sub-index 0, this will disable the PDO's mapping.
   Send the following message: SDO access to object $1A03_h$ sub-index 0, 8-bit value 0.
4. **Map the new objects**.
   a. Write in object $1A03_h$ sub-index 1 the description of the Position actual value:

Index	Sub-index	Length	Resulting data
6064h	00h	20h	60640020h

Send the following message (SDO access to object 1A03h sub-index 1, 32-bit value 60640020h).

b. Write in object 1A03h sub-index 2 the description of the Digital inputs:

Index	Sub-index	Length	Resulting data
60FDh	00h	20h	60FD0020h

Send the following message (SDO access to object 1A03h sub-index 2, 32-bit value 60FD0020h).

5. **Enable the RxPDO3 mapped objects**. Set the object 1A03h sub-index 0 with the value 2 to enable both mapped objects.

   Send the following message: SDO access to object 1A03h sub-index 0, 8-bit value 2.

6. **Add the new TPDO to the Sync Manager**.

   -Write in object 1C13h sub-index 4 the TPDO4 mapping parameter object number:

   Send the following message: SDO access to object 1C13h sub-index 4, 16-bit value 1A03h.

   -Write 04 h in object 1C13h sub-index 0, this will enable the Sync. Manager.

   Send the following message: SDO access to object 1C13h sub-index 0, 8-bit value 04h.

   Note: if using TwinCAT System Manager, enter in Configuration Mode, select the drive, select Process Data tab, uncheck the PDO Assignment and PDO Configuration boxes. Click Load PDO info from device button to load the new PDO configuration. Reload the IO devices and enter in Operation state.

7. **Set the drive to Safe-Operational**.
8. **Set the drive to Operational.**

3 EtherCAT® State Machine (ESM)

3.1 Overview

The EtherCAT® State Machine (ESM) is responsible for the coordination of master and slave at start up and during operation. State changes are mainly caused by interactions between master and slave. They are primarily related to writes to Application Layer Controlword.

After Initialization of Data Layer and Application Layer the machine enters the **INIT** State. The 'Init' state defines the root of the communication relationship between the master and the slave in application layer. No direct communication between the master and the slave on application layer is possible. The master uses the 'Init' state to initialize a set of configuration register. The corresponding sync manager configurations are also done in the 'Init' state.

The '**Pre-Operational**' state can be entered if the settings of the mailbox have been set. Both the master and the slave can use the mailbox and the appropriate protocols to exchange application specific initializations and parameters.

No process data communication is possible in this state.

The '**Safe-Operational**' state can be entered if the settings of the input buffer have been updated. The application of the slave shall deliver actual input data without processing the output data. The real outputs of the slave will be set to their "safe state".

The '**Operational**' state can be entered if the settings of the output buffer have been done and actual outputs have been delivered to the slave (provides outputs of the slave will be used).

The application of the slave shall deliver actual input data and the application of the master will provide output data.

The '**Bootstrap**' state can be entered only to update the firmware via FoE protocol. Entering this state will disable all other communication channels (including RS232).

The ESM defines four states, which are supported:

- Init
- Pre-Operational
- Safe-Operational
- Operational
- Bootstrap (only with F515F or newer)

3.1.1 Device control

All state changes are possible except for the 'Init' state, where only the transition to the 'Pre Operational' state is possible and for the 'Pre-Operational' state, where no direct state change to 'Operational' exists.

State changes are normally requested by the master. The master requests a write to the Application Layer Control register which results in a Register Event 'Application Layer Control' indication in the slave. The slave will respond to the change in Application Layer Control through a local Application Layer Status write service after a successful or a failed state change. If the requested state change failed, the slave will respond with the error flag set.

ESM is specified in *Figure 3.1.1*.

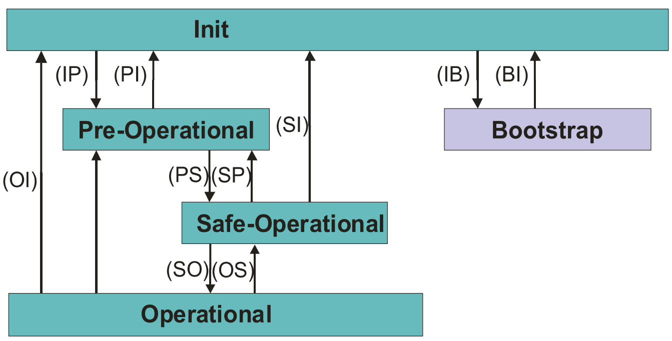


Figure 3.1.1. *EtherCAT®* State Machine Diagram

The local management services are related to the transitions in the ESM, as specified in *Table 3.1.1*. If there is more than one service related to the transition, the slave's application will process all of the related services.

Table 3.1.1 – State transitions and local management services

State transition	Local management services
(IP)	Start Mailbox Communication
(PI)	Stop Mailbox Communication
(PS)	Start Input Update
(SP)	Stop Input Update
(SO)	Start Output Update
(OS)	Stop Output Update
(OP)	Stop Output Update, Stop Input Update
(SI)	Stop Input Update, Stop Mailbox Communication
(OI)	Stop Output Update, Stop Input Update, Stop Mailbox Communication
(IB)	Init to Bootstrap
(BI)	Bootstrap to Init

Table 3.1.2 – AL Control Description

Parameter	Data Type	Value
State	Unsigned4	1: Init 3: Bootstrap 2: Pre-Operational 4: Safe-Operational 8: Operational
Acknowledge	Unsigned1	0: Parameter Change of the **AL** Status Register will be unchanged 1: Parameter Change of the **AL** Status Register will be reset
Reserved	Unsigned3	Shall be zero
Application Specific	Unsigned8	

3.1.2 EtherCAT® State Machine and CANopen State Machine

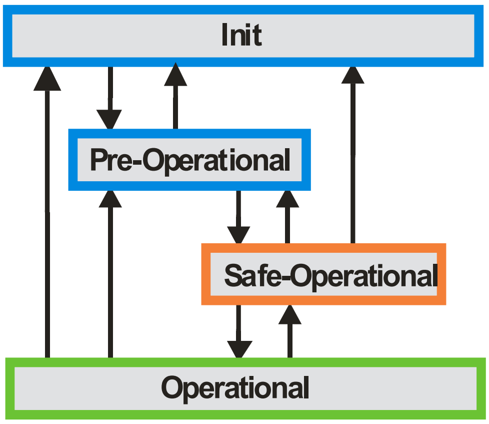


Figure 3.1.2. *EtherCAT®* State Machine

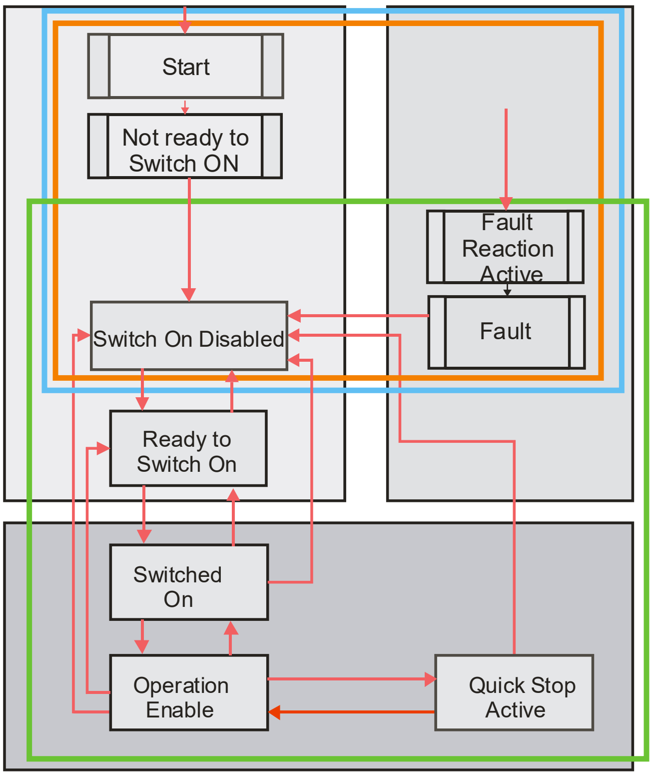


Figure 3.1.3. *Drive State-machine based on CANopen (DS402).*

3.1.3 Emergency messages

A drive sends an emergency message (EMCY) when a drive internal error occurs. An emergency message is transmitted only once per 'error event'. As long as no new errors occur, the drive will not transmit further emergency messages.

The emergency error codes supported by the Technosoft drives are listed in **Table 3.1.3**. Details regarding the conditions that may generate emergency messages are presented at object Motion Error Register index 2000_h.

Table 3.1.3 – Emergency Error Codes

Error code (hex)	Description
00xx	Error Reset or No Error
1000	Generic Error; sent when a communication error occurs on CAN (object 2000_h bit0=1; usually followed by EMCY code 0x7500
2310	Continuous over-current
2340	Short-circuit
30xx	Voltage
3210	DC-link over-voltage
3220	DC-link under-voltage
33xx	Output Voltage
4280	Over temperature motor
4310	Over temperature drive
5441	Drive disabled due to enable or STO input
5442	Negative limit switch active
5443	Positive limit switch active
6100	Invalid setup data
7300	Sensor error; this emergency message also contains other data; see its description at the end of this table
7500	Communication error
8100	EtherCAT® communication error
8210	PDO not processed due to length error
8220	PDO length exceeded
8331	I2t protection triggered
8580	Position wraparound / or hall sensor error
8611	Control error / Following error
9000	Command error
A0xx	ESM Transition Error
F0xx	Additional Functions
FF01	Generic interpolated position mode error (PVT / PT error)
FF02	Change set acknowledge bit wrong value
FF03	Specified homing method not available
FF04	A wrong mode is set in object 6060_h, modes_of_operation
FF05	Specified digital I/O line not available
FF06	Positive software position limit triggered
FF07	Negative software position limit triggered
FF08	Enable/STO circuit hardware error

3.1.3.1 Emergency message structures

The Emergency message contains 8 data bytes having the following contents:

Most EMCY messages:

0 1	2	3 7
Emergency Error Code	Error Register (Object 1001_h)	Manufacturer specific error field

0x7300 Sensor error:

0 1	2	3 4	5 7
Emergency Error Code	Error Register (Object 1001_h)	Detail Error Register 2 (Object 2009_h)	Manufacturer specific error field

0xFF01 Generic interpolated position mode error (PVT / PT error):

0 1	2	3	4	5 7
Emergency Error Code (0xFF01)	Error Register (Object 1001_h)	Interpolated position (Object 2072_h)	status	Manufacturer specific error field

To disable the sending of PVT emergency message with ID 0xFF01, the setup variable PVTSENDOFF must be set to 1.

To disable the automatic sending of some of the emergency messages, use *Object 2001h: Motion Error Register Mark*.

3.2 EtherCAT® Synchronization

3.2.1 Overview

The drive uses the SYNC 0 signal in order to synchronize with the EtherCAT® master and the network. The SYNC 0 signal must have the period equal or multiple than the drive slow control loop which is by default at 1ms.

The synchronization assures the good functioning of modes like Cyclic Synchronous Position, Cyclic Synchronous Velocity and Cyclic Synchronous Torque.

3.2.2 Synchronization signals

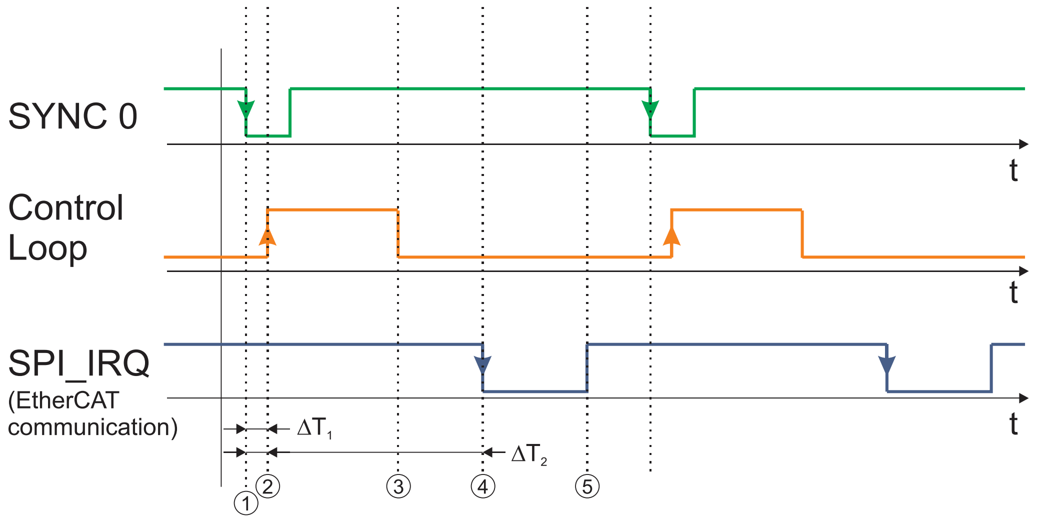


Figure 3.2.1. *Synchronization signal and control loop timing*

Time moments description:

1 – SYNC0 descending edge. Everything is synchronized with this event

2 – Control loop start. Actual position Pi is read, immediately after entering in the control loop

3 – Control loop end. At this moment the actual position Pi is stored in the EtherCAT slave ASIC as TxPDO, so it will be transmitted on next EtherCAT communication cycle

4 – A new EtherCAT frame is received. It puts in the ASIC in RxPDO the reference Ri+1 for next control loop and reads from the ASIC from TxPDO the actual position Pi

5 – The EtherCAT data processing is finalized. The new reference Ri+1 is stored in the drive internal variables which are used by the control loops

In order to work correctly the synchronization, the following conditions shall be respected:

a) communication cycle shall be a multiple of the SYNC0 cycle
b) SYNC0 shall be a multiple of control loop
c) time moment 3 shall be before time moment 4
d) time moment 5 shall be before time moment 1 of the next cycle

Remarks:

the minimum jitter between SYNC0 and the control loop is obtained when both have the cycle tine

it is possible to have different values for control loop, SYNC0 and communication cycle as long as conditions a) and b) are respected. Example: control loop at 1ms; SYNC0 at 2 ms and communication cycle at 4ms

On Beckhoff EtherCAT masters, the delay ΔT_2 between time moments 1 and 5 can be adjusted. By default, the Beckhoff master tries to place the communication cycle to minimize the time interval between time moment 5 and time moment 1 of next cycle. In many cases the “default” settings are not correct and moment 5 is superposed with 1. So, the “default” “shift time” in the Beckhoff master needs to be changed in order to reduce ΔT_2 (increase the time interval between time moment 5 and next SYNC0).

Object 2109h: Sync offset can also adjust the ∆T1 time. Although it is recommended to modify only the sync0 shift time form the EtherCAT master.

Important:

Technosoft drives can be configured to generate on 3 outputs the above mentioned key signals:

- SYNC0 on OUT0; Object 2089_h bit0=1;
- Control loop on Ready/OUT3; Object 2089_h bit1=1;
- EtherCAT Communication processing on Error/OUT2; Object 2089_h bit2=1;

This feature can be activated by writing via SDO value 7 in object 2089_h. It can be deactivated via reset or by writing 0 in object 2089_h.

In order to preserve the synchronization when changing ∆T1 and ∆T2, conditions c) and d) shall be checked with an oscilloscope to make sure that a safe margin exists considering the worst case jitter of the control loop and the EtherCAT communication processing time

3.2.3 Object 2089_h: Synchronization test config

This object enables the visualization of SYNC0, Control Loop and EtherCAT communication signals over the drive digital outputs.

Object description:

Index	2089_h
Name	Synchronization test config
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	No
Value range	0-7
Default value	No

***Table 3.2.1** –Bit Assignment in Synchronization test config*

Bit	Value	Description
3-15	-	Reserved
2	1	Trigger EtherCAT communication on Error/OUT2
1	1	Trigger Control Loop on Ready/OUT3
0	1	View SYNC0 on OUT0

Remarks:

Before activating this feature, disconnect any other device connected to the 3 outputs;

Ready and Error outputs are also connected to the green and red LEDs. These will flicker when this feature is activated. This case shall not be treated as an error condition!

SYNC0 signal on OUT0 is generated by the drive starting from SYNC0 signal generated by the EtherCAT ASIC. OUT0 goes down almost simultaneously with the SYNC0 generated by the ASIC, but it rises much faster. So SYNC0 from OUT0 will be shorter than the real signal.

3.2.4 Object 2109h: Sync offset

This object adjusts the ΔT_1 time from *3.2.2 Synchronization* signals. This time represents the difference between the sync0 signal start time and the Control loop start time.

Remark:

Use this object only if sync0 offset cannot be adjusted from the EtherCAT master.

Some EtherCAT masters start the communication loop immediately after the sync0 signal. If the communication loop is active while the control loop is active, the synchronization will not work and vibrations in the motor will be noticed while moving in CSP mode.

If the sync0 offset cannot be adjusted, this object will offset the control loop start.

Object description:

Index	2109_h
Name	Sync offset
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Value range	0 … 55000 (not higher)
Default value	1000

The default value for this object can be changed by editing the parameter "SLSyncOffset" found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

3.2.5 Object 210Ah: Sync rate

This object adjusts the timing corrections applied to the internal clock in order to sync faster.

Remark:

Use this object only for fine tuning the synchronization signals. It should be adjusted only by viewing the sync signals with the oscilloscope and observing the results while modifying it.

Object description:

Index	$210A_h$
Name	Sync rate
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	No
Value range	0..32767
Default value	3

3.2.6 Object 2112_h: TML priority[1]

By changing the default value of the object, the TML language will have an increased priority, therefore changes of Modes of operations (6060_h) and Controlword (6040_h) objects are not allowed. Any TML commands that the drive receives will be executed.

Object description:

Index	2112_h
Name	TML priority
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Default value	0

Table 3.2.2 *– TML priority description*

Value	Description
0	Default value.
1…65535	Writing or Changing Modes of operations (6060_h) and Controlword (6040_h) objects is not allowed

[1] Available only for FA0xx firmware version.

4 Drive control and status

4.1 CiA402 State machine and command coding

The state machine from **Drives and motion control device profile** (CiA 402) describes the drive status and the possible control sequences of the drive. The drive has to pass through the described states in order to control the motor. The drive states can be changed by the object 6040h (Controlword) and/or by internal events. The drive current state is reflected in the object **6041h (Statusword)**. **Figure 4.1.1** describes the state machine of the drive along with Controlword and Statusword values for each transition. **Table 4.1.1** describes each transition present in the state machine.

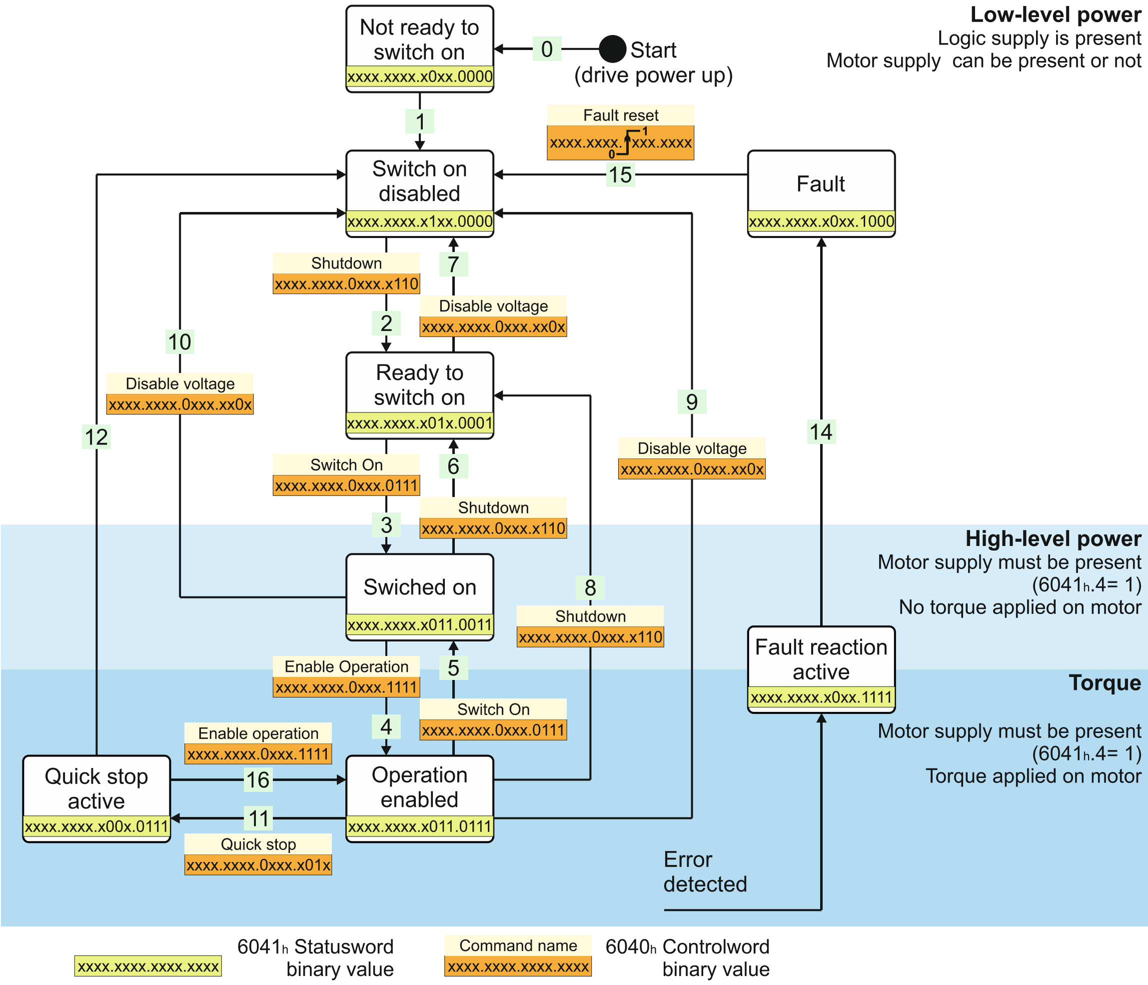


Figure 4.1.1. *Drive's status machine. States and transitions*

Table 4.1.1 *– Drive State Transitions*

Transition	Event	Action
0	Automatic transition after power-on or reset application	Hardware Initialization
1	Automatic transition.	Initialization completed successfully. Communication is active
2	Bits 1 and 2, are set in Controlword (*Shutdown* command). Motor voltage may be present.	None
3	Bits 0,1 and 2 are set in Controlword (*Switch On* command)	Motor supply voltage must be present (6041h bit 4=1). The undervoltage protection is active. The motor will not be powered and have no torque.
4	Bits 0,1,2 and 3 are set in Controlword (*Enable Operation* command)	Motion function and power stage are enabled, assuming the enable or STO input is also enabled. Depending on the mode of operation that is set, the motor will apply torque and keep its current position or velocity to 0. Depending on the motor start mode, this transition may take more than a few ms to finish. Example: When using the start mode "Move till aligned with phase A" which is the default method, the first executed Enable operation transition takes 2 seconds.

5	Bit 3 is cancelled in Controlword (*Disable Operation* command)	Motion function is inhibited. The drive will execute the instructions from Object 605Ch: Disable operation option code and finally transition into *Switched On* state. The motor has no torque.
6	Bit 0 is cancelled in Controlword (*Shutdown* command)	Motor supply may be disabled. Motor has no torque.
7	Bit 1 or 2 is cancelled in Controlword (*Quick Stop* or *Disable Voltage* command)	None
8	Bit 0 is cancelled in Controlword (*Shutdown* command)	The drive will execute the instructions from Object 605Bh: Shutdown option code and finally transition into *Ready to switch on* state. The motor has no torque.
9	Bit 1 is cancelled in Controlword (*Disable Voltage* command)	The drive will execute the instructions from Object 605Ch: Disable operation option code and finally transition into *Switch on disabled* state. The motor has no torque.
10	Bit 1 or 2 is cancelled in Controlword (*Quick Stop* or *Disable Voltage* command)	Motor supply may be disabled. Drive has no torque.
11	Bit 2 is cancelled in Controlword (*Quick Stop* command)	The drive will execute the instructions from Object 605Ah: Quick stop option code.
12	*Quick Stop* is completed or bit 1 is cancelled in Controlword (*Disable Voltage* command)	Output stage is disabled. Motor has no torque.
13	Fault signal	Execute specific fault treatment routine from Object 605Eh: Fault reaction option code
14	The fault treatment is complete	The drive function is disabled
15	Bit 7 is set in Controlword (*Reset Fault* command)	Some of the bits from *4.4.2 Object 2000h: Motion Error Register* are reset. If all the error conditions are reset, the drive returns to Switch On Disabled status. After leaving the state *Fault* bit 7, *Fault Reset* of the Controlword has to be cleared by the host.
16	Bit 2 is set in Controlword (*Enable Operation* command). This transition is possible if *Quick-Stop-Option-Code* is 5, 6, 7 or 8	Drive exits from Quick Stop state. Drive function is enabled.

Table 4.1.2 *– Drive States*

State	**Description**
Not Ready to switch on	The drive performs basic initializations after power-on. The drive function is disabled The transition to this state is automatic.
Switch On Disabled	The drive basic initializations are done and the green led must turn-on if no error is detected. The drive is not Ready to switch on; any drive parameters can be modified, including a complete update of the whole EEPROM data (setup table, TML program, cam files, etc.) The motor supply can be switched on, but the motion functions cannot be carried out yet. The transition to this state is automatic.
Ready to switch on	The motor supply voltage may be switched on, most of the drive parameter settings can still be modified, and motion functions cannot be carried out yet.
Switched On (Operation Disabled)	The motor supply voltage must be applied. The power stage is switched off. The motion functions cannot be carried out yet.
Operation Enabled	No fault present, power stage is switched on, motion functions are enabled. If the operation mode set performs position control, the motor is held in position. If the operation mode set performs speed control, the motor is kept at zero speed. If the operation mode is torque external, the motor is kept with zero torque. From this state, the motor can execute motion commands.
Quick Stop Active	Drive has been stopped with the quick stop deceleration. The power stage is enabled. If the drive was operating in position control when quick stop command was issued, the motor is held in position. If the drive was operating in speed control, the motor is kept at zero speed. If the drive was operating in torque control, the motor is kept at zero torque.
Fault Reaction Active	The drive performs a default reaction to the occurrence of an error condition
Fault	The motor power is turned off. The drive remains in fault condition, until it receives a Reset Fault command. If following this command, all the bits from the Motion Error Register are reset, the drive exits the fault state

4.2 Drive control and status objects

4.2.1 Object 6040h: Controlword

The object controls the status of the drive. It is used to enable/disable the power stage of the drive, start/halt the motions and to clear the fault status. The status machine is controlled through the Controlword.

Object description:

Index	6040h
Name	Controlword
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Yes
Units	-
Value range	0 … 65535
Default value	No

***Table 4.2.1** – Bit Assignment in Controlword*

Bit	Value	Meaning
15	0	Registration mode inactive
	1	Activate registration mode
14	0	When an update is performed, keep unchanged the demand values for speed and position (TML command TUM1;)
	1	When an update is performed, update the demand values for speed and position with the actual values of speed and position (TML command TUM0;)
13		When it is set, it cancels the execution of the TML function called through object 2006h. The bit is automatically reset by the drive when the command is executed.
12	0	No action
	1	If bit 14 = 1 – Force *position demand value* to 0 If bit 14 = 0 – Force *position actual value* to 0 This bit is valid regardless of the status of the drive or other bits in Controlword
11		Manufacturer Specific - Operation Mode Specific. The meaning of this bit is detailed further in this manual for each operation mode
10-9		Reserved. Writes have no effect. Read as 0
8	0	No action
	1	Halt command – the motor will slow down on slow down ramp
7	0	No action
	1	Reset Fault. The faults are reset on 0 to 1 transition of this bit. After a Reset Fault command, the master has to reset this bit.
4-6		Operation Mode Specific. The meaning of these bits is detailed further in this manual for each operation mode
3		Enable Operation
2		Quick Stop
1		Enable Voltage
0		Switch On

The following table lists the bit combinations for the Controlword that lead to the corresponding state transitions. An X corresponds to a bit state that can be ignored. The single exception is the fault reset: The transition is only started by a bit transition from 0 to 1.

***Table 4.2.2** – Command coding in Controlword*

Command	Bit in object 6040h					Transition
	Bit 7	Bit 3	Bit 2	Bit 1	Bit 0	
Shutdown	0	X	1	1	0	2,6,8
Switch on	0	0	1	1	1	3
Disable voltage	0	X	X	0	X	7,9,10,12
Quick stop	0	X	0	1	X	7,10,11
Disable operation	0	0	1	1	1	5
Enable operation	0	1	1	1	1	4,16
Fault reset	0→1	X	X	X	X	13

For the command coding values see also *Figure 4.1.1. Drive's status machine. States and transitions.*

4.2.2 Object 6041$_h$: Statusword

Object description:

Index	6041$_h$
Name	Statusword
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Yes
Units	-
Value range	0 … 65535
Default value	No

The Statusword has the following bit assignment:

***Table 4.2.3** – Bit Assignment in Statusword*

Bit	Value	Description	
15	0	Axis off. Power stage is disabled. Motor control is not performed	
	1	Axis on. Power stage is enabled. Motor control is performed	
14	0	No event set or the programmed event has not occurred yet	
	1	Last event set has occurred	
13..12		Operation Mode Specific. The meaning of these bits is detailed further in this manual for each operation mode	
11		Internal Limit Active – see ***Remark 1*** below	
10		Target reached	
9	0	Remote – drive is in local mode and will not execute the command message.	
	1	Remote – drive parameters may be modified via ECAT messages and the drive will execute the command message.	
8	0	No TML function or homing is executed. The execution of the last called TML function or homing is completed.	
	1	A TML function or homing is executed. Until the function or homing execution ends or is aborted, no other TML function / homing may be called	
7	0	No Warning	
	1	Warning. A TML function / homing was called, while another TML function / homing is still in execution. The last call is ignored.	
6		Switch On Disabled.	
5		Quick Stop. When this bit is zero, the drive is performing a quick stop	
4	0	Motor supply voltage is absent	See ***Remark 2*** below
	1	Motor supply voltage is present	
3		Fault. If set, a fault condition is or was present in the drive.	
2		Operation Enabled	
1		Switched On	
0		Ready to switch on	

The drive state can be identified when Statusword coding is the following:

***Table 4.2.4** – State coding in Statusword*

Statusword	Drive state
xxxx xxxx x0xx 0000$_b$	Not Ready to switch on
xxxx xxxx x1xx 0000$_b$	Switch on disabled
xxxx xxxx x01x 0001$_b$	Ready to switch on
xxxx xxxx x01x 0011$_b$	Switched on
xxxx xxxx x01x 0111$_b$	Operation enabled
xxxx xxxx x00x 0111$_b$	Quick stop active
xxxx xxxx x0xx 1111$_b$	Fault reaction active
xxxx xxxx x0xx 1000$_b$	Fault

For the state coding values see also *Figure 4.1.1. Drive's status machine. States and transitions.*

Remark 1: Bit11 internal limit active is set when either the Positive or Negative limit switches is active. If the internal register LSACTIVE = 1 or object 60B8$_h$ bit 6 = 1, this bit will not be set and the emergency messages for the active limit switches will be disabled.

Remark 2: Bit 4 shows whether the +Vmot Input is supplied. The state machine cannot transition to states Switched On and Operation enabled without this bit being set first. If this bit transitions to 0 while in Operation enabled or Switched On states (+Vmot input is not present), the drive will enter fault state due to undervoltage error. If in a lower state than switch On, the absence of +Vmot in will not trigger an undervoltage error.

4.2.3 Object 6060h: Modes of Operation

The object selects the mode of operation of the drive.

Object description:

Index	6060h
Name	Modes of Operation
Object code	VAR
Data type	INTEGER8

Entry description:

Access	RW
PDO mapping	Yes
Units	-
Value range	-128 … 127
Default value	No

Data description:

Value	Description
-128…-1	Reserved
0	No mode change/no mode assigned
1	Profile Position Mode
2	Reserved
3	Profile Velocity Mode
4	Profile Torque Mode[1]
5	Reserved
6	Homing Mode
7	Interpolated Position Mode
8	Cyclic sync Position Mode (CSP)
9	Cyclic sync Velocity Mode (CSV)
10	Cyclic sync Torque Mode (CST)
11…127	Reserved

Remark: *The actual mode is reflected in object 6061h (Modes of Operation Display).*

4.2.4 Object 6061h: Modes of Operation Display

The object reflects the actual mode of operation set with object Modes of Operation (index 6060h).

If the drive is in an inferior state than Operation enabled and object 6060h Modes of operation is changed, object 6061h will take the value of 6060h only after the drive reached Operation enabled state.

Object description:

Index	6061h
Name	Modes of Operation Display
Object code	VAR
Data type	INTEGER8

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	-128 … 127
Default value	-

Data description: Same as for object 6060h Modes of Operation.

4.3 Limit Switch functionality explained

4.3.1 Hardware limit switches LSP and LSN functionality

All iPOS drives have two limit switch inputs:

- LSP – positive limit switch
- LSN – negative limit switch

Triggering a limit switch during a motion causes the drive to automatically stop using the deceleration value defined in Object 6085h: Quick stop deceleration. After the motor stops, it will continue to hold its position and wait until a new motion command is received in the opposite direction of the active limit switch. A new motion in the opposite direction will be accepted only after the motor ends its deceleration, signaled by Statusword 6041h bit 10.

[1] This mode is available starting with firmware versions F515K / FA00x.

While the motor stops due to an activated limit switch, the Statusword will still report the Operation enabled state and NOT actually enter Quick stop state (where Statusword = xxxx xxxx x00x 0111_b). *Object 605Ah: Quick stop option code* will have no effect if a limit switch is activated.

If during a positive motion LSP is activated, the motor will stop.

If during a negative motion LSN is activated, the motor will stop.

If during a positive motion LSN is activated, nothing will happen.

If during a negative motion LSP is activated, nothing will happen.

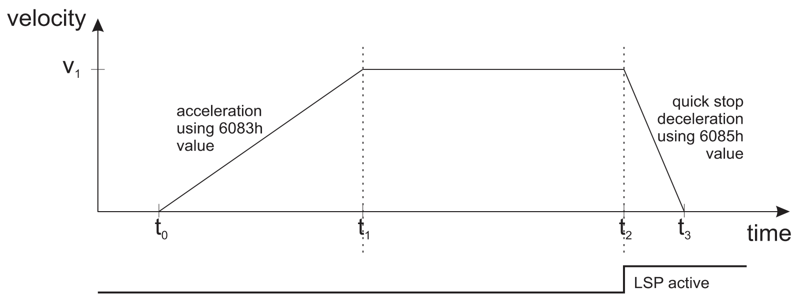


Figure 4.3.1. *Stopping a motion on the positive limit switch*

Figure 4.3.1 depicts a positive motion where the speed increases from t0 until t1 using the acceleration value defined in Object 6081h: Profile velocity. At moment t2, the positive limit switch is activated and the drive automatically stops and it decelerates using the value defined in Object 6085h: Quick stop deceleration.

While the positive limit switch is active, no new positive motion will be accepted by the drive. Only a negative motion is accepted while LSP is active.

While the negative limit switch is active, no new negative motion will be accepted by the drive. Only a positive motion is accepted while LSN is active.

A limit switch can be defined as active while the input is in the low or high state in Drive setup:

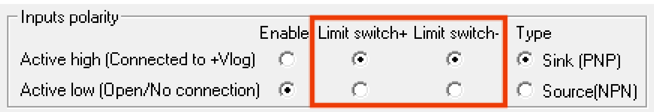


Figure 4.3.2. *Configuring the limit witch active state in Drive setup.*

Status word Bit11 (internal limit active) is set when either the Positive or Negative limit switch is active. If the internal parameter LSACTIVE = 1 or object $60B8_h$ bit 6 = 1, status word bit11 will not be set and the emergency messages for the active limit switches will be disabled. If the limit switches inputs are disabled, they can be used as regular digital inputs.

If the positive limit switch is activated, the emergency error code 0x5443 will be sent automatically and object 2000_h bit 6 will be 1.

If the negative limit switch is activated, the emergency error code 0x5442 will be sent automatically and object 2000_h bit 7 will be 1.

When a limit switch becomes inactive, the emergency error code 0x0000 will be sent automatically and object 2000_h bit 6 or 7 will return to 0.

All iPOS drives can also use the limit switch inputs in order to capture the motor or load position. This function is configurable through Object 60B8h: Touch probe function and Object 2104h: Auxiliary encoder function. If the feedback type is incremental encoder, the position is captured within several µs. If the feedback type is SSI/BiSS/Resolver/Linear halls or Sin/Cos, the captured position is the latest one computed in the position loop, so by default it may be up to 1 ms old.

4.3.2 Software limit switches functionality

The software limit switches work just like the hardware limit switches (LSP, LSN) in terms of functionality. An individual position value is chosen for the negative and positive limits and when those values are reached, the motor will decelerate until it stops. A new motion will be accepted only if the motion is opposite the active software or hardware limit switch.

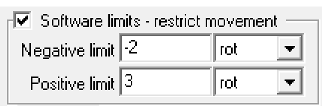


Figure 4.3.3. *Configuring the software limit switches position values in Drive setup.*

The software limit switches can also be configured through Object 607Dh: Software position limit.

If the positive software limit switch is activated, the emergency error code 0xFF06 will be sent automatically and object 2002h bit 6 will be 1.

If the negative software limit switch is activated, the emergency error code 0xFF07 will be sent automatically and object 2002h bit 7 will be 1.

When a limit switch becomes inactive, the emergency error code 0x0000 will be sent automatically and object 2002h bit 6 or 7 will return to 0.

4.4 Error monitoring

4.4.1 Object 1001h: Error Register

This object is an error register for the device. The device can map internal errors in this byte. This entry is mandatory for all devices. It is a part of an Emergency object.

Object description:

Index	1001h
Name	Error register
Object code	VAR
Data type	UNSIGNED8

Entry description:

Access	RO
PDO mapping	No
Value range	UNSIGNED8
Default value	No

Table 4.4.1 *– Bit description of object **1001h***

Bit	Description
0	Generic error
1	Current
2	Voltage
3	Temperature
4	Communication error
5	Device profile specific
6	Reserved (always 0)
7	Manufacturer specific.

Valid bits while an error occurs – bit 0 and bit 4. The other bits will remain 0.

4.4.2 Object 2000h: Motion Error Register

The Motion Error Register displays all the drive possible errors. A bit set to 1 signals that a specific error has occurred. When the error condition disappears or the error is reset using a Fault Reset command, the corresponding bit is reset to 0.

The Motion Error Register is continuously checked for changes of the bits status. When a bit is set (e.g. an error has occurred), if the corresponding bit from Motion Error Register Mask (2001h) is set to 1, an emergency message with the specific error code is sent. When a bit is reset, if the corresponding bit from Motion Error Register Mask (2001h) is set to 1, an emergency message for error reset is sent.

Object description:

Index	2000h
Name	Motion Error Register
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	0 … 65535
Default value	0

***Table 4.4.2** – Bit Assignment in Motion Error Register*

Bit	Description
15	Drive disabled due to enable or STO input. Set when enable or STO input is on disable state. Reset when enable or STO input is on enable state
14	Command error. This bit is set in several situations and acts as a warning. They can be distinguished either by the associated emergency code, or in conjunction with other bits from the DER (2002h) register.
13	Under-voltage. Set when protection is triggered. Reset by a Reset Fault command. Can be triggered only in Switch On or Operation enabled states. If in a lower state, the drive will not fault if no motor voltage input is present.
12	Over-voltage. Set when protection is triggered. Reset by a Reset Fault command
11	Over temperature drive. Set when protection is triggered. Reset by a Reset Fault command.
10	Over temperature motor. Set when protection is triggered. Reset by a Reset Fault command. This protection may be activated if the motor has a PTC or NTC temperature contact.
9	I^2T protection. Set when protection is triggered. Reset by a Reset Fault command
8	Over current. Set when protection is triggered. Reset by a Reset Fault command
7	Negative limit switch active. Set when LSN input is in active state. Reset when LSN input is inactive state
6	Positive limit switch active. Set when LSP input is in active state. Reset when LSP input is inactive state
5	For F515F and newer: Feedback error. Details found in DER2 (2009h) bits. Set when protection is triggered. Reset by a Reset Fault command. For F510x/511x; it represents either digital Hall sensor missing or position wraparound.
4	Communication error. Set when protection is triggered. Reset by a Reset Fault command
3	Control error (position/speed error too big). Set when protection is triggered. Reset by a Reset Fault command
2	Invalid setup data. Set when the EEPROM stored setup data is not valid or not present.
1	Short-circuit. Set when protection is triggered. Reset by a Reset Fault command
0	EtherCAT® communication error. Reset by a Reset Fault command or by Clear Error in the EtherCAT® State Machine.

4.4.3 Object 2001h: Motion Error Register Mark

The Motion Error Register Mask offers the possibility to choose which of the errors set or reset in the Motion Error Register to be signaled via emergency messages. The Motion Error Register Mask has the same bit codification as the Motion Error Register (see Table above) and the following meaning:

1 – Send an emergency message when the corresponding bit from the Motion Error Register is set

0 – Don't send an emergency message when the corresponding bit from the Motion Error Register is set.

Object description:

Index	2001h
Name	Motion Error Register Mask
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	0 … 65535
Default value	0

4.4.4 Object 2002h: Detailed Error Register (DER)

The Detailed Error Register displays detailed information about the errors signaled with command Error bit from Motion Error Register. Not all bits represent errors. This register also displays the status of software limit switches and lock EEPROM status. A bit set to 1 signals that a specific error has occurred. When the error condition disappears or the error is reset using a Fault Reset command, the corresponding bit is reset to 0.

Object description:

Index	2002h
Name	Detailed Error Register
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	0 … 65535
Default value	0

Table 4.4.3 – Bit Assignment in Detailed Error Register

Bit	Description
15	EEPROM is Locked. The EEPROM can be locked via object 2091h or by Easy Motion Studio – Select Communication – Lock EEPROM.
14	STO or Enable circuit hardware error
13	Self-check error. The ECAT adapter EEPROM memory is not programmed with the XML/ESI file data or has errors.
12	reserved
11	Start mode failed; Motionless start or pole lock minimum movement failed
10	Encoder broken wire; On a brushless motor, either the digital halls or the incremental encoder signal was interrupted
9	Update ignored for S-curve
8	S-curve parameters caused an invalid profile. UPD instruction was ignored.
7	Negative software limit switch is active.
6	Positive software limit switch is active.
5	Cancelable call instruction received while another cancelable function was active.
4	UPD instruction received while AXISON was executed. The UPD instruction was ignored and it must be sent again when AXISON is completed.
3	A call to an inexistent function was received.
2	A call to an inexistent homing routine was received.
1	A RET/RETI instruction was executed while no function/ISR was active.
0	The number of nested function calls exceeded the length of TML stack. Last function call was ignored.

4.4.5 Object 2009$_h$: Detailed Error Register 2 (DER2)[1]

The Detailed Error Register 2 mostly displays detailed information about the errors signaled with command Feedback error bit 5 from Motion Error Register (2000$_h$). A bit set to 1 signals that a specific error has occurred. When the error condition disappears or the error is reset using a Fault Reset command, the corresponding bit is reset to 0.

Object description:

Index	2009$_h$
Name	Detailed Error Register 2
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	0 … 65535
Default value	0

Table 4.4.4 – Bit Assignment in Detailed Error Register 2

Bit	Description
15	Output frequency. The imposed speed exceeds the DUAL USE European regulation limit.
14..6	reserved
6	Position wraparound. The position 2^{31} was exceeded. It does not represent a Fault condition.
5	Hall sensor missing; can be either Digital or Linear analogue hall error.
4	Absolute Encoder Interface (AEI) interface error; applies only to iPOS80x0 BA drives
3	BiSS sensor missing; No BiSS sensor communication detected.
2	BiSS data error bit is set. The BiSS protocol includes an error bit in its data.
1	BiSS data warning bit is set. If ASR2.10 = 1, this error will represent a Fault condition.
0	BiSS data CRC error. BiSS data stream CRC does not match computed CRC.

4.4.6 Object 603Fh: Error code

This object provides the error code of the last error which occurred in the drive device. These error codes are always transmitted as Emergency messages.

The error codes are described in *3.1.3 Emergency messages*.

Object description:

Index	603F$_h$
Name	Error Code
Object code	VAR
Data type	UNSIGNED16

[1] Available only with F515x firmwares

Entry description:

Access	RO
PDO mapping	Yes
Units	-
Value range	0 … 65535
Default value	0

4.4.7 Object 605Ah: Quick stop option code

This object determines what action should be taken if the quick stop function is executed. The slow down ramp is a deceleration value set by the Profile acceleration object, index 6083h. The quick stop ramp is a deceleration value set by the Quick stop deceleration object, index 6085h.

Object description:

Index	605Ah
Name	Quick stop option code
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	No
Value range	-32768 … 32767
Default value	2

Data description:

Value	Description
-32768…-1	Manufacturer specific
0	Disable drive function
1	Slow down on slow down ramp and transit into Switch On Disabled
2	Slow down on quick stop ramp and transit into Switch On Disabled
3	Reserved
4	Reserved
5	Slow down on slow down ramp and stay in Quick Stop Active
6	Slow down on quick stop ramp and stay in Quick Stop Active
7…32767	Reserved

4.4.8 Object 605Bh: Shutdown option code

This object determines what action is taken if when there is a transition from Operation Enabled state to Ready to Switch On state. The slowdown ramp is a deceleration value set by the Profile acceleration object, index 6083h.

Object description:

Index	605Bh
Name	Shutdown option code
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	No
Value range	-32768 … 32767
Default value	0

Data description:

Value	Description
-32768…-1	Manufacturer specific
0	Disable drive function (switch-off the drive power stage)
1	Slow down on slowdown ramp and disable the drive function
2…32767	Reserved

4.4.9 Object 605Ch: Disable operation option code

This object determines what action is taken if when there is a transition from Operation Enabled state Switched On state. The slowdown ramp is a deceleration value set by the Profile acceleration object, index 6083h.

Object description:

Index	605Ch
Name	Disable operation option code
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	No
Value range	-32768 … 32767
Default value	1

Data description:

Value	Description
-32768…-1	Manufacturer specific
0	Disable drive function (switch-off the drive power stage)
1	Slow down on slow down ramp and disable the drive function
2…32767	Reserved

4.4.10 Object 605Dh: Halt option code

This object determines what action is taken if when the halt command is executed. The slowdown ramp is a deceleration value set by Object 6083h: Profile acceleration. The quick stop ramp is a deceleration value set by Object 6085h: Quick stop deceleration.

Object description:

Index	605Dh
Name	Halt option code
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	No
Value range	-32768 … 32767
Default value	1

Data description:

Value	Description
-32768…-1	Manufacturer specific
0	Reserved
1	Slow down on slow down ramp and stay in Operation Enabled
2	Slow down on quick stop ramp and stay in Operation Enabled
3…32767	Reserved

4.4.11 Object 605Eh: Fault reaction option code

This object determines what action should be taken if a non-fatal error occurs in the drive. The non-fatal errors are by default the following:

- Under-voltage
- Over-voltage
- I^2t error[1] –when the internal register ASR bit1 is 0 in setup.
- Drive over-temperature
- Motor over-temperature
- Communication error (when object 6007h option 1 is set)

Remark: the under-voltage protection is monitored while in Switched On and Operation enabled states. If in a lower state, the drive will not fault if no motor voltage input is present.

[1] Starting with firmware version FA00G / FA02G, I^2t is no longer a "non-fatal error" that can be configured through object 605Eh.

Object description:

Index	605Eh
Name	Fault reaction option code
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	No
Value range	-32768 … 32767
Default value	2

Data description:

Value	Description
-32768…-2	Manufacturer specific
-1	No action
0	Disable drive, motor is free to rotate
1	Reserved
2	Slow down with quick stop ramp
3…32767	Reserved

4.4.12 Object 6007h: Abort connection option code

The object sets the action performed by the drive when a communication error occurs.

Object description:

Index	6007h
Name	Abort connection option code
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	Yes
Value range	-32768…32767
Default value	1

***Table 4.4.5** – Abort connection option codes values*

Option code	Description
-32768…-1	Manufacturer specific (reserved)
0	No action
+1	Fault signal - Execute specific fault routine set in Object 605Eh: Fault reaction option code
+2	Disable voltage command
+3	Quick stop command
+4…+32767	Reserved

The default value for this object can be changed by editing the parameter "x6007" found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

4.4.13 Object 2114h: Fault Override Option Code[1]

This object serves as a mean to define a custom action routine when specific errors are triggered. Once activated, the custom routine has a higher priority in comparison to the actions defined in objects 6007h: Abort connection option code and 605Eh: Fault reaction option code.

Each bit within this object corresponds to an error found in Object 2000h: Motion Error Register, and by setting the corresponding bit to 1, the fault routine can be customized using the options described in object 2113h: Detailed Option Code.

Object description:

Index	2114Eh
Name	Override Option Code
Object code	VAR
Data type	UNSIGNED16

Entry description:

[1] Available starting with FA00G / FA02G firmware versions or newer

Access	RW
PDO mapping	No
Value range	0 … 65535
Default value	32768

Data description:

Bit	Description
0	Communication error
1	Short-Circuit
2	Reserved
3	Control error
4…7	Reserved
8	Over current
9	Reserved
10	Over temperature - Motor
11	Over temperature - Drive
12	Over voltage
13	Under voltage
14	Reserved
15	Enable / STO inactive

4.4.14 Object 2113h: Detailed Option Code[1]

This object establishes the available actions for customizing a fault routine associated with each error described in Object 2114h: Fault Override Option Code. These designated options will be implemented only when the corresponding bit in Object 2114h is activated; otherwise, the settings will remain inactive.

Object description:

Index	2113Eh
Name	Detailed Option Code
Object code	VAR
Data type	UNISGNED16

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	1…15
Default value	15

Sub-index	1
Description	Short-Circuit option code*
Access	RW
PDO mapping	NO
Value range	UNSIGNED16
Default value	0

Sub-index	2
Description	Reserved

Sub-index	3
Description	Control error option code
Access	RW
PDO mapping	NO
Value range	UNSIGNED16
Default value	0

Sub-index	4
Description	Communication error option code
Access	RW
PDO mapping	NO
Value range	UNSIGNED16
Default value	0

Sub-index	5, 6, 7
Description	Reserved

[1] Available starting with FA00G / FA02G firmware versions or newer

Sub-index	8
Description	Over current option code
Access	RW
PDO mapping	NO
Value range	UNSIGNED16
Default value	0

Sub-index	9
Description	Reserved

Sub-index	10
Description	Over temperature – Motor option code
Access	RW
PDO mapping	NO
Value range	UNSIGNED16
Default value	0

Sub-index	11
Description	Over temperature – Drive option code
Access	RW
PDO mapping	NO
Value range	UNSIGNED16
Default value	0

Sub-index	12
Description	Over voltage option code
Access	RW
PDO mapping	NO
Value range	UNSIGNED16
Default value	0

Sub-index	13
Description	Under voltage option code
Access	RW
PDO mapping	NO
Value range	UNSIGNED16
Default value	0

Sub-index	14
Description	Reserved

Sub-index	15
Description	Enable / STO inactive* option code
Access	RW
PDO mapping	NO
Value range	UNSIGNED16
Default value	32768

Table 4.6 *– Sub-index bit description*

Bit	Value	Description
15	0	Do not generate a TML interrupt
	1	Generate a TML interrupt
8…14	0	Reserved
0…7	0	Disable drive
	2	Quick stop
	-1	No action

* For the Short circuit and Enable/STO inactive option codes, only the customization of bit 15 is possible.

4.5 Digital I/O control and status objects

4.5.1 Object 60FD$_h$: Digital inputs

The object contains the actual value of the digital inputs available on the drive. Each bit from the object corresponds to a digital input (manufacturer specific or device profile defined). If a bit is SET, then the status of the corresponding input is logical ‘1’ (high). If the bit is RESET, then the corresponding drive input status is logical ‘0’ (low).
Remarks:

- *The device profile defined inputs (limit switches, home input and interlock) are mapped also on the manufacturer specific inputs. Hence, when one of these inputs changes the status, then both bits change, from the manufacturer specific list and from the device profile list.*
- *The number of available digital inputs is product dependent. Check the drive user manual for the available digital inputs.*

Object description:

Index	60FD$_h$
Name	Digital inputs
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RO
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0

	Bit	Value	Description
Manufacturer specific	31		IN15 status
	30		IN14 status
	29		IN13 status
	28		IN12 status
	27		IN11 status
	26		IN10 status
	25		IN9 status
	24		IN8 status
	23		IN7 status
	22		IN6 status
	21		IN5 status
	20		IN4 status
	19		IN3 status
	18		IN2 status
	17		IN1 status
	16		IN0 status
Device profile defined	15..4		Reserved
	3	0	Interlock (Drive enable/ STO input) deactivated; drive may not apply power to motor. Enter *Switch on disabled* state.
		1	Interlock (Drive enable/ STO input) activated; drive may apply power to motor.
	2	0	Home switch input status is low
		1	Home switch input status is high
	1	0	Positive limit switch is inactive
		1	Positive limit switch is active
	0	0	Negative limit switch is inactive
		1	Negative limit switch is active

4.5.2 Object 208F$_h$: Digital inputs 8bit

This object has 2x8 bit sub-indexes that show the same data as object 60FD$_h$ Digital inputs. Mapping shorter data to a PDO decreases the total communication bus load and processing time.

Remark:

The number of available digital inputs is product dependent. Check the drive user manual for the available digital inputs.

Object description:

Index	208F$_h$
Name	Digital inputs 8bit
Object code	ARRAY
Data type	UNSIGNED8

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	1…2
Default value	2

Sub-index	1
Description	Device profile defined inputs
Access	RO
PDO mapping	Possible
Value range	UNSIGNED8
Default value	no

Sub-index	2
Description	Manufacturer specific inputs
Access	RO
PDO mapping	Possible
Value range	UNSIGNED8
Default value	no

***Table 4.5.1** – Sub-index 1 bit description*

	Bit	Value	Description
208Fh:01 Device profile defined input	4..7		Reserved
	3	0	Interlock (Drive enable/STO input) activated; drive may apply power to motor
		1	Interlock (Drive enable/STO input) deactivated; drive may not apply power to motor. Enter *Switch on disabled* state.
	2	0	Home switch input status is low
		1	Home switch input status is high
	1	0	Positive limit switch is inactive
		1	Positive limit switch is active
	0	0	Negative limit switch is inactive
		1	Negative limit switch is active

***Table 4.5.2** – Sub-index 2 bit description*

	Bit	Value	Description
208Fh:02 Manufacturer specific inputs	7		IN7 status
	6		IN6 status
	5		IN5 status
	4		IN4 status
	3		IN3 status
	2		IN2 status
	1		IN1 status
	0		IN0 status

4.5.3 Object 60FEh: Digital outputs

The object controls the digital outputs of the drive. The first sub-index sets the outputs state to high or low if the mask allows it in the second sub-index, which defines the outputs that can be controlled.

All iPOS drives have NPN type outputs. If an output bit is **SET (1)**, then the corresponding drive output will be switched to logical '1' (high). The output will disconnect the load from the GND. If the bit is **RESET(0)**, then the corresponding drive output will be switched to logical '0' (low). The output will connect the load to the GND.

Remarks:

The actual number of available digital outputs is product dependent. Check the drive user manual for the available digital outputs.

If an unavailable digital output is selected in sub-index 2, the drive will issue an emergency message with ID 0xFF05.

Object description:

Index	60FEh
Name	Digital outputs
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	1…2
Default value	2

Sub-index	1
Description	Physical outputs
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0

Sub-index	2
Description	Bit mask
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0

***Table 4.5.3** – Bit mask description*

	Bit	Description
Manufacturer Specific	31	OUT15 command
	30	OUT14 command
	29	OUT13 command
	28	OUT12 command
	27	OUT11 command
	26	OUT10 command
	25	OUT9 command
	24	OUT8 command
	23	OUT7 command
	22	OUT6 command
	21	OUT5 command
	20	OUT4 command
	19	OUT3 command
	18	OUT2 command
	17	OUT1 command
	16	OUT0 command
Device profile Defined	15…0	Reserved

4.5.3.1 Example for setting the digital outputs

The example will Set OUT0 to 0(connect to GND) and OUT1 to 1 (disconnect from GND).

1. **Set sub-index 1 with the needed outputs states.** Set bit 16 (OUT0) to 0 and bit17 (OUT1) to 1.

   Set in **60FEh** **sub-index1** to 0x00020000.

2. **Set sub-index 2 bit mask only with the output values that need to be changed.** Set bit 16 and 17 to 1 to allow the change of OUT0 and OUT1 states.

   Set in **60FEh** **sub-index2** to 0x00030000.

After the second sub-index is set, the selected outputs will switch their state to the values defined in sub-index 1.

4.5.4 Object 2090h: Digital outputs 8bit

Has the same functionality as object 60FEh digital outputs, only that its two sub-indexes are 8 bit instead of 32bit. Mapping shorter data to a PDO decreases the total communication bus load and processing time.

Object description:

Index	2090h
Name	Digital outputs 8bit
Object code	ARRAY
Data type	UNSIGNED8

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	1…2
Default value	2

Sub-index	1
Description	Physical outputs 8bit
Access	RW
PDO mapping	Possible
Value range	UNSIGNED8
Default value	0

Sub-index	2
Description	Bit mask 8bit
Access	RW
PDO mapping	Possible
Value range	UNSIGNED8
Default value	0

***Table 4.5.4** – Sub-index 1&2 Bit description*

	Bit	Description
Manufacturer Specific Outputs	7	OUT7 command
	6	OUT6 command
	5	OUT5 command
	4	OUT4 command
	3	OUT3 command
	2	OUT2 command
	1	OUT1 command
	0	OUT0 command

4.5.5 Object 2045h: Digital outputs status

The actual status of the drive outputs can be monitored using this object.

Object description:

Index	2045h
Name	Digital outputs status
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	UNSIGNED16
Default value	No

Data description:

Bit	Meaning	Bit	Meaning
15	OUT15 status	7	OUT7 status
14	OUT14 status	6	OUT6 status
13	OUT13 status	5	OUT5 status
12	OUT12 status	4	OUT4 status
11	OUT11 status	3	OUT3 status
10	OUT10 status	2	OUT2 status
9	OUT9 status	1	OUT1 status
8	OUT8 status	0	OUT0 status

If the any of the bits is **SET**, then the corresponding drive output status is logical '1' (high). If the bit is **RESET**, then the corresponding drive output status is logical '0' (low).

4.5.6 Object 2102h: Brake status

The object shows the status for the digital output assigned to operate a mechanical brake on the motor. When bit1 is SET (=1), the brake output is active. This object will show an inactive brake depending on the brake release delay parameter set in the Motor Setup. The brake will start to deactivate when the command Switch On is received in Control Word and it may still be active even when the drive reaches the Operation Enabled state is Status Word. In case a mechanical brake is used, the CoE master should not send a motion command until this object is 0.

Object description:

Index	2102h
Name	Brake status
Object code	VAR
Data type	USINT8

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	0 or 1
Default value	No

4.5.7 Object 2046h: Analogue input: Reference

The object contains the actual value of the analog reference applied to the drive. Through this object, one can supervise the analogue input dedicated to receive the analogue reference in the external control modes.

Object description:

Index	2046h
Name	Analogue input: Reference
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	0 … 65520
Default value	No

4.5.8 Object 2047h: Analogue input: Feedback

The object contains the actual value of the analogue feedback applied to the drive.

Object description:

Index	2047h
Name	Analogue input: Feedback
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	0 … 65520
Default value	No

4.5.9 Object 2055h: DC-link voltage

The object contains the actual value of the DC-link voltage. The object is expressed in internal voltage units.

Object description:

Index	2055h
Name	Analogue input: DC-link voltage
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Possible
Units	DC-VU
Value range	0 … 65520
Default value	No

The computation formula for the voltage [IU] in [V] is:

$$Voltage\_measured[V] = \frac{VDCMaxMeasurable[V]}{65520} \cdot Voltage\_measured[IU]$$

where *VDCMaxMeasurable* is the maximum measurable DC voltage expressed in [V]. This value can be read in the “Drive Info” dialogue, which can be opened from the “Drive Setup”.

4.5.10 Object 2058h: Drive Temperature

The object contains the actual drive temperature. The object is expressed in temperature internal units.

Object description:

Index	2058h
Name	Analogue input for drive temperature
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	0 … 65535
Default value	No

Note: if the drive does not have a temperature sensor, this object should not be used.

The computation formula for the temperature [IU] in [°C] is:

$$\mathrm{Temp}[^\circ\mathrm{C}] = \frac{3.3}{\mathrm{DriveTempSensorGain}*65520} * \left(\mathrm{Temp}[\mathrm{IU}] - \frac{\mathrm{DriveTempOutAt0oC}*65520}{3.3}\right)$$

where *DriveTempSensorGain* and *DriveTempOutAt0oC* can be found as *Sensor gain* and *Output at 0 °C* in the "Drive Info" dialogue, which can be opened from the "Drive Setup".

4.5.11 Object $208B_h$[1]: Sin AD signal from Sin/Cos encoder

The object contains the actual value of the analogue sine signal of a Sin/Cos encoder.

Object description:

Index	$208B_h$
Name	Sin AD signal from Sin/Cos encoder
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	-32768 … 32767
Default value	No

4.5.12 Object $208C_h$[2]: Cos AD signal from Sin/Cos encoder

The object contains the actual value of the analogue cosine signal of a Sin/Cos encoder.

Object description:

Index	$208C_h$
Name	Cos AD signal from Sin/Cos encoder
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	-32768 … 32767
Default value	No

4.6 Protections Setting Objects

4.6.1 Object $607D_h$: Software position limit

The object sets the maximal and minimal software position limits. If the actual position is lower than the negative position limit or higher than the positive one, a software position limit emergency message will be launched. If either of these limits is passed, the motor will start decelerating using the value set in *Object 6085h: Quick stop deceleration*. Once it has decelerated, the motor will stand still until a new command is given to travel within the space defined by the limits.

Remarks:

A value of -2147483648 for Minimal position limit and 2147483647 for Maximal position limit disables the position limit check.

[1] Object $208B_h$ is available only on firmware F515x.

[2] Object $208C_h$ is available only on firmware F515x.

Object description:

Index	$607D_h$
Name	Software position limit
Object code	ARRAY
Data type	INTEGER32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	2
Default value	2

Sub-index	1
Description	Minimal position limit
Access	RW
PDO mapping	Possible
Value range	INTEGER32
Default value	0x80000000

Sub-index	2
Description	Maximal position limit
Access	RW
PDO mapping	Possible
Value range	INTEGER32
Default value	0x7FFFFFFF

4.6.2 Object 2050_h: Over-current protection level

The Over-Current Protection Level object together with object Over-Current Time Out (2051_h) defines the drive over-current protection limits. The object defines the value of current in the drive, over which the over-current protection will be activated, if lasting more than a time interval that is specified in object 2051_h. It is set in current internal units.

Object description:

Index	2050_h
Name	Over-current protection level
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Units	CU
Value range	0 … 32767
Default value	No

The computation formula for the current [IU] in [A] is:

$$current[A] = \frac{2 \cdot Ipeak}{65520} \cdot current[IU]$$

where *Ipeak* is the peak current supported by the drive and *current[IU]* is the command value for object 2050_h.

4.6.3 Object 2051_h: Over-current time out

The Over-Current time out object together with object Over-Current Protection Limit (2050_h) defines the drive over-current protection limits. The object sets the time interval after which the over-current protection is triggered if the drive current exceeds the value set through object 2050_h. It is set in time internal units.

Object description:

Index	2051_h
Name	Over-current time out
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Possible
Units	TU
Value range	0 … 65535
Default value	No

4.6.4 Object 2052h: Motor nominal current

The object sets the maximum motor current RMS value for continuous operation. This value is used by the I2t motor protection and one of the start methods. It is set in current internal units. See object 2053h for more details about the I2t motor protection.

Object description:

Index	2052h
Name	Motor nominal current
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Units	CU
Value range	0 … 32767
Default value	No

The computation formula for the current [IU] in [A] is:

$$current[A] = \frac{2 \cdot Ipeak}{65520} \cdot current[IU]$$

where *Ipeak* is the peak current supported by the drive and *current[IU]* is the read value from object 2052h.

4.6.5 Object 2053h: I2t protection integrator limit

Objects 2053h and 2054h contain the parameters of the I^2t protection (against long-term motor over-currents). Their setting must be coordinated with the setting of the object 2052h, motor nominal current. Select a point on the I^2t motor thermal protection curve, which is characterized by the points I_I2t (current, [A]) and t_I2t: (time, [s]) (see **Figure 4.6.1**)

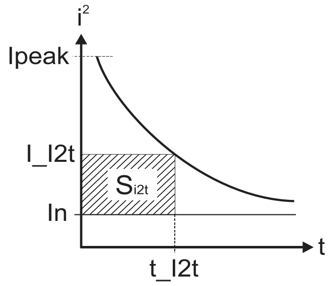


Figure 4.6.1.*I2t motor thermal protection curve*

The points I_I2t and t_I2t on the motor thermal protection curve together with the nominal motor current **In** define the surface S_{I2t}. If the motor instantaneous current is greater than the nominal current In and the I2t protection is activated, the difference between the square of the instantaneous current and the square of the nominal current is integrated and compared with the SI2t value (see **Figure 4.6.2**). When the integral equals the SI2t surface, the I2t protection is triggered.

Object description:

Index	2053h
Name	I2t protection integrator limit
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	No
Units	-
Value range	0 … 2^{31}-1
Default value	No

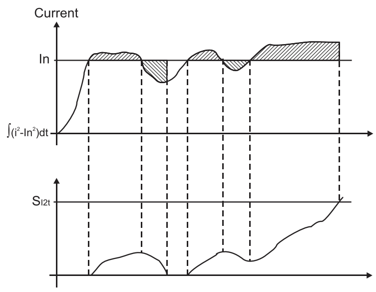


Figure 4.6.2. *I2t protection implementation*

The computation formula for the i2t protection integrator limit (I2TINTLIM) is

$$I2TINTLIM = \frac{(I\_I2t)^2 - (In)^2}{32767^2} \cdot 2^{26}$$

where I_I2t and In are represented in current units (CU).

4.6.5.1 I2t protection explained

The I2t protection can behave in two ways, depending on bit 1 of the ASR register found in parameters.cfg. The default setting for all iPOS templates is ASR.1=1.

If ASR.1=0 – the drive will enter fault state and motor power will be OFF. The Software Protections Interrupt will be executed when I2T protection triggered.

If ASR.1=1 – the motion will continue running with 90% of Nominal Current set as the current limit until the I2T integral drops to 0. The Software Protections Interrupt will not be executed when I2T protection triggered.

4.6.6 Object 2054h: I2t protection scaling factor

Object description:

Index	2054h
Name	I2t protection scaling factor
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Units	-
Value range	0 … 65535
Default value	No

The computation formula for the i2t protection scaling factor (SFI2T) is

$$SFI2T = 2^{26} \cdot \frac{Ts\_S}{t\_I2t}$$

where Ts_S is the sampling time of the speed control loop [s], and t_I2t is the I2t protection time corresponding to the point on the graphic in **Figure 4.6.1**.

4.6.7 Object 207Fh: Current limit

The object defines the maximum current that will pass through the motor. This object is valid only for the configurations using: brushless, DC brushed and stepper closed loop motor. The value is set in current internal units.

Object description:

Index	207Fh
Name	Current limit
Object code	VAR
Data type	Unsigned16

Entry description:

Access	RW
PDO mapping	YES
Units	-
Value range	0 … 65535
Default value	No

The computation formula for the current_limit [A] to [IU] is:

$$Current\_Limit[IU] = 32767 - \frac{Current\_Limit[A] \cdot 65520}{2 \cdot Ipeak}$$

where *Ipeak* is the peak current supported by the drive, C*urrent_Limit[A]* is the targ
et current in [A] and C*urrent_Limit[IU]* is the target value to be written in object 207Fh.

4.7 Step Loss Detection for Stepper Open Loop configuration

By using a stepper open loop configuration, the command resolution can be greater than the one used for a normal closed loop configuration. For example if a motor has 200 steps/ revolution and 256 microsteps / step, results in 51200 Internal Units/ revolution position command. If a 1000 lines quadrature encoder is used, it means it will report 4000 Internal Units/ revolution.

By using the step loss detection, means using a stepper in open loop configuration and an encoder to detect lost steps. When the protection triggers, the drive enters Fault state signaling a Control error. To enable the protection, a stepper open loop + encoder on motor must be selected along with a correct Control error protection value.

4.7.1 Object 2083_h: Encoder Resolution for step loss protection

Sets the number of encoder increments for one full motor rotation. For example, if an encoder has 2000 increments/revolution, then 2000 must be written into the object.

Remark: The value for this object is automatically calculated in the setup when choosing a Stepper Open Loop with feedback on motor configuration.

Object description:

Index	2083_h
Name	Encoder resolution for step loss protection
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Yes
Value range	UNSIGNED32
Default value	-

The value for this object can be changed by editing the parameter “ENCRESLONG” found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

4.7.2 Object 2084_h: Stepper Resolution for step loss protection

Sets the number of microsteps the step motor does for one full rotation. For example, if the motor has 100 steps / revolution (see **Figure 4.7.1**) and is controlled with 256 microsteps / step (see **Figure 4.7.2**), the value 100x256=25600 should be found into this object.

Remark: The value for this object is automatically calculated in the setup when choosing a Stepper Open Loop with feedback on motor configuration.

Object description:

Index	2084_h
Name	Stepper resolution for step loss protection
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Yes
Value range	UNSIGNED32
Default value	-

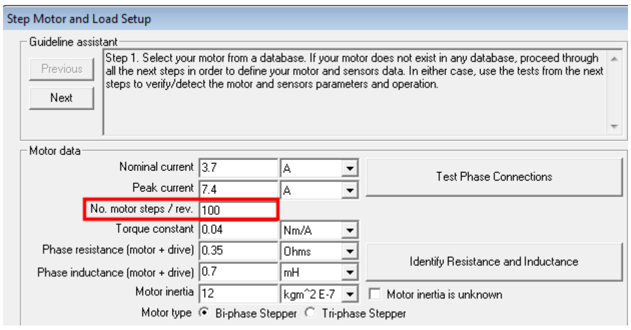


Figure 4.7.1. *Motor steps / revolution*

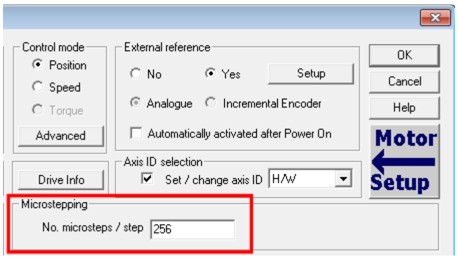


Figure 4.7.2. *Motor microsteps / step*

The value for this object can be changed by editing the parameter “STEPRESLONG” found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

4.7.3 Enabling step loss detection protection

Before enabling the step loss detection protection, the *Encoder resolution* in object 2083h and the *Stepper resolution* in object 2084h must be set correctly. These two objects should already be set automatically if the correct setup parameters were introduced. In addition, the feedback sensor must be set *on motor* in Motor Setup:

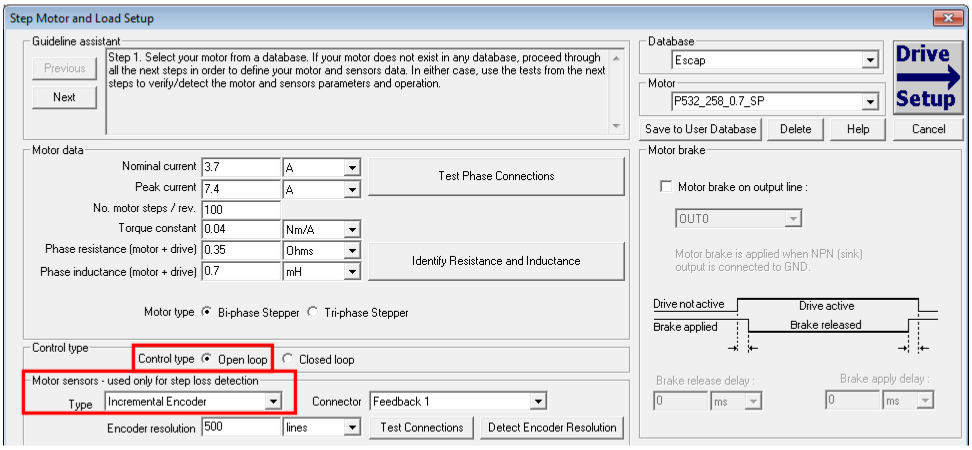


Figure 4.7.3. *Configuring the feedback sensor for step loss detection*

The step loss detection protection parameters are actually the control error parameters: object *6066h - Following error time* and object *6065h - Following error window*. The protection is triggered if the error between the commanded position and the position measured via the encoder is greater than the value set in object 6065h for a time interval greater than the value set in object 6066h.

The following error window is expressed in microsteps. The Following error time is expressed in multiples of position/speed control loops (1ms by default for stepper configurations).

To enable the step loss detection protection, set first the *Following error window* in object 6056h, then set the *Following error time* in object 6066h to a value different from 65535 (0xFFFF). To disable this protection, set a 65535 value in object 6066h.

Example: Following error window is set to 1000 and *Following error time* is set to 20. The step motor has 100 steps/rev and is controlled with 256 microsteps/step. The step loss protection will be triggered if the difference between the commanded position and the measured position is bigger than 1000 microsteps (i.e. 1000/(100*256) rev = 14,06 degrees) for a time interval bigger or equal than 20 control loops of 1ms each i.e. 20ms.

Remark: the actual value of the error between the commanded position and the measured position can be read from object 60F4h. It is expressed in microsteps.

4.7.4 Step loss protection setup

The following steps are recommended for optimal setup of the step loss protection parameters:

Move your motor with the highest velocity and load planned to be used in your application

During the movement at maximal speed, read object 60F4h - *Following error actual value* as often as possible to determine its highest value.

> **Remark:** *Following error actual value* can be read at every control loop using EasyMotion Studio or Easy Setup by logging the TML variable POSERR.

Add a margin of about 25% to the highest error value determined at previous step and set the new obtained value into object 6065h - *Following error window*.

Activate the step loss detection by writing a non-zero value in object 6066h - *Following error time out.* Recommended values are between 1 and 10.

4.7.5 Recovering from step loss detection fault

When the step loss detection protection is triggered, the drive enters in Fault state. The EtherCAT® master will receive an emergency message from the drive with control error/following error code. In order to exit from Fault state and restart a motion, the following steps must be performed:

- Send fault reset command to the drive. The drive will enter in Switch On Disabled state;
- Send Disable voltage command into Controlword.
- Send Switch On command into Controlword.
- Send Enable operation into Controlword. At this moment, voltage is applied to the motor and it will execute the phase alignment procedure again. The position error will be reset automatically.
- Start a homing procedure to find again the motor zero position.

4.7.6 Remarks about Factor Group settings when using step the loss detection

When the drive controls stepper motors in open loop, if the factor group settings are activated they are automatically configured for correspondence between motor position in user units and microsteps as internal units. Because the motor position is read in encoder counts, it leads to incorrect values reported in objects 6064_h Position actual value and 6062_h Position demand value.

Only object 6063_h *Position actual internal value* will always show the motor position correctly in encoder counts.

If the factor group settings are not used, i.e. all values reported are in internal units (default), both 6064_h *Position actual value* and 6062_h *Position demand value* will provide correct values.

4.8 Drive info objects

4.8.1 Object 1000_h: Device Type

The object contains information about drive type and its functionality. The 32-bit value contains 2 components of 16-bits: the 16 LSB describe the CiA standard that is followed.

Object description:

Index	1000_h
Name	Device type
Object code	VAR
Data type	UNSIGNED32

Value description:

Access	RO
PDO mapping	NO
Value range	UNSIGNED32
Default value	60192_h for iPOS family

4.8.2 Object 6502_h: Supported drive modes

This object gives an overview of the operating modes supported on the Technosoft drives. Each bit from the object has assigned an operating mode. If the bit is set then the drive supports the associated operating mode.

Object description:

Index	6502_h
Name	Supported drive modes
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RO
PDO mapping	Possible
Value range	UNSIGNED32
Default value	$001C03E5_h$ - iPOS-CAT drives

The modes of operation supported by the Technosoft drives, and their corresponding bits, are the following:

Data description:

MSB **LSB**

0	0	x	…	x	0		0	1	1	0	0	1	0	1
Manufacturer specific					rsvd			ip	hm	rsvd	tq	pv	vl	pp
31	21	20	…	16	15	…	7	6	5	4	3	2	1	0

Data description – manufacturer specific:

Bit	Description
31 … 21	Reserved
20	External Reference Torque Mode
19	External Reference Speed Mode
18	External Reference Position Mode
17	Electronic Gearing Position Mode
16	Electronic Camming Position Mode

4.8.3 Object 1008_h: Manufacturer Device Name

The object contains the manufacturer device name in ASCII form, maximum 15 characters.

Object description:

Index	1008_h
Name	Manufacturer device name
Object code	VAR
Data type	Visible String

Entry description:

Access	Const
PDO mapping	No
Value range	No
Default value	iPOS

4.8.4 Object $100A_h$: Manufacturer Software Version

The object contains the firmware version programmed on the drive in ASCII form with the maximum length of 15 characters.

Object description:

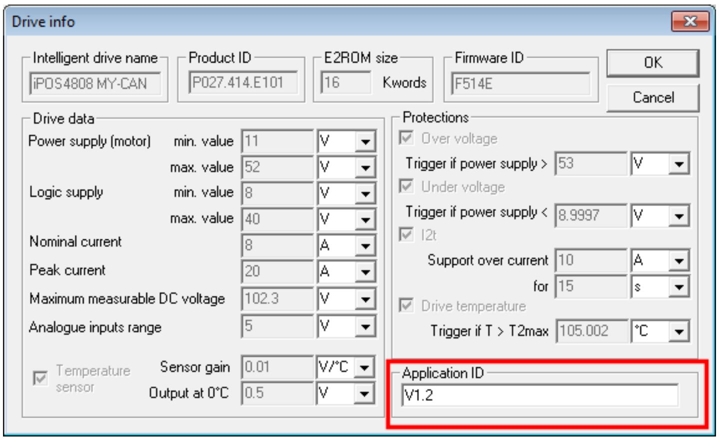
Index	$100A_h$
Name	Manufacturer software version
Object code	VAR
Data type	Visible String

Entry description:

Access	Const
PDO mapping	No
Value range	No
Default value	Product dependent

4.8.5 Object 2060_h: Software version of a TML application

By inspecting this object, the user can find out the software version of the TML application (drive setup plus motion setup and eventually cam tables) that is stored in the EEPROM memory of the drive. The object shows a string of the first 4 elements written in the TML application field, grouped in a 32-bit variable. If more character are written, only the first 4 will be displayed. Each byte represents an ASCII character.



Object description:

Index	2060_h
Name	Software version of TML application
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RO
PDO mapping	No
Units	-
Value range	No
Default value	No

Example:

If object 2060$_h$ contains the value 0x322E3156, then the software version of the TML application is read as:

0x56 – ASCII code of letter **V**

0x31 – ASCII code of number **1**

0x2E – ASCII code of character . (point)

0x32 – ASCII code of number **2**

Therefore, the version is **V1.2**.

4.8.6 Object 1018$_h$: Identity Object

This object provides general information about the device.

Sub-index 01$_h$ shows the unique Vendor ID allocated to Technosoft (1A3$_h$).

Sub-index 02$_h$ contains the Technosoft drive product ID. It can be found physically on the drive label or in Drive Setup/ Drive info button under the field product ID. If the Technosoft product ID is P027.214.E121, sub-index 02$_h$ will be read as the number 27214121 in decimal.

Sub-index 03$_h$ shows the Revision number.

Sub-index 04$_h$ shows the drives Serial number. For example the number 0x4C451158 will be 0x4C (ASCII L); 0x45 (ASCII E); 0x1158 --> the serial number will be LE1158.

Object description:

Index	1018$_h$
Name	Identity Object
Object code	RECORD
Data type	Identity

Entry description:

Sub-index	00$_h$
Description	Number of entries
Access	RO
PDO mapping	No
Value range	1..4
Default value	1

Sub-index	01$_h$
Description	Vendor ID
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	000001A3$_h$

Sub-index	02$_h$
Description	Product Code
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	Product dependent

Sub-index	03$_h$
Description	Revision number
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	0x30313030 (ASCII 0100)

Sub-index	04$_h$
Description	Serial number
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	Unique number

4.9 Miscellaneous Objects

4.9.1 Object 2025h: Stepper current in open-loop operation

In this object, one can set the level of the current to be applied when controlling a stepper motor in open loop operation at runtime.

Object description:

Index	2025h
Name	Stepper current in open-loop operation
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	Possible
Units	IU
Value range	-32768 … 32767
Default value	No

The computation formula for the current [IU] in [A] is:

$$current[A] = \frac{2 \cdot Ipeak}{65520} \cdot curent[IU]$$

where *Ipeak* is the peak current supported by the drive and *current[IU]* is the commanded value in object 2025h.

4.9.2 Object 2026h: Stand-by current for stepper in open-loop operation

In this object, one can set the level of the current to be applied when controlling a stepper motor in open loop operation in stand-by.

Object description:

Index	2026h
Name	Stand-by current for stepper in open-loop operation
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	Possible
Units	CU
Value range	-32768 … 32767
Default value	No

4.9.3 Object 2027h: Timeout for stepper stand-by current

In this object, one can set the amount of time after the value set in object 2026h, *stand-by current for stepper in open-loop operation* will activate as the reference for the current applied to the motor after the reference has reached the target value.

Object description:

Index	2027h
Name	Timeout for stepper stand-by current
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Possible
Units	TU
Value range	0 … 65535
Default value	No

4.9.4 Object 2075h: Position triggers

This object is used in order to define a set of four position values. If the *position actual value* is the value set as a position trigger, then the corresponding bit in *SRH – Status Register High (bits 1…4)* will be set.

Object description:

Index	2075_h
Name	Position triggers
Object code	ARRAY
Data type	INTEGER32

Entry description:

Sub-index	00_h
Description	Number of sub-indexes
Access	RO
PDO mapping	No
Default value	4

Sub-index	01_h – 04_h
Description	Position trigger 1 - 4
Access	RW
PDO mapping	Possible
Value range	INTEGER32
Default value	No

4.9.5 Object 2085$_h$: Position triggered outputs

The object controls the digital outputs 0, 1 and 5 in concordance with the position triggers 1, 2 and 4 status from the *SRH – Status Register High (bits 1, 2 and 4)*.

Object description:

Index	2085_h
Name	Position triggered outputs
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Units	-
Value range	0 … 65535
Default value	No

The *Position triggered outputs* object has the following bit assignment:

***Table 4.9.1** – Bit Assignment in Position triggered outputs*

Bit	Value	Meaning
12-15	0	Reserved.
11	0	OUT5 = 1 when Position trigger 4 = 0 OUT5 = 0 when Position trigger 4 = 1
	1	OUT5 = 0 when Position trigger 4 = 0 OUT5 = 1 when Position trigger 4 = 1
10	0	Reserved.
9	0	OUT1 = 1 when Position trigger 2 = 0 OUT1 = 0 when Position trigger 2 = 1
	1	OUT1 = 0 when Position trigger 2 = 0 OUT1 = 1 when Position trigger 2 = 1
8	0	OUT0 = 1 when Position trigger 1 = 0 OUT0 = 0 when Position trigger 1 = 1
	1	OUT0 = 0 when Position trigger 1 = 0 OUT0 = 1 when Position trigger 1 = 1
4-7	0	Reserved
3[1]	1	Enable position trigger 4 control of OUT5
	0	Disable position trigger 4 control of OUT5
2	0	Reserved
1	1	Enable position trigger 2 control of OUT1
	0	Disable position trigger 2 control of OUT1
0	1	Enable position trigger 1 control of OUT0
	0	Disable position trigger 1 control of OUT0

Note: Some drives may not have some outputs available. The object will control only the ones that exist.

[1] Some outputs may not be available on all drives.

4.9.6 Object 2076h: Save current configuration

This object is used in order to enable saving the current configuration of the operating parameters of the drive. These parameters are the ones that are set when doing the setup of the drive. The purpose of this object is to be able to save the new values of these parameters in order to be re-initialized at subsequent system re-starts.

Writing any value in this object will trigger the save in the non-volatile EEPROM memory of the current drive operating parameters.

Object description:

Index	2076h
Name	Save current configuration
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	WO
PDO mapping	No
Value range	UNSIGNED16
Default value	-

4.9.7 Object 208Ah: Save setup status

This object is used in order to monitor the parameters saving process. Bit 0 will be set to 1 when the 2076h object can be activated or the save function has been completed. It will stay 0 while the save function is ongoing.

Object description:

Index	208Ah
Name	Save setup status
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Yes
Value range	INTEGER16
Default value	0

4.9.8 Object 2080h: Reset drive

This object is used to reset the drive by writing any non-zero value in it.

Remark: it resets only the drive; it does not reset the ECAT interface.

Object description:

Index	2080h
Name	Reset drive
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	WO
PDO mapping	No
Value range	USIGNED16
Default value	0

4.9.9 Object 2082h: Sync on fast loop

This object is used to synchronize the drive on the fast or slow loop sample period. The Distributed Clock time (SYNC 0) must be set accordingly with the time of the chosen sample loop in this object.

By default, the fast loop period for all configurations is set to 0.1 ms, the slow loop period 1ms.

Object description:

Index	2082h
Name	Sync on fast loop
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Value range	0: synchronize on slow loop 1: synchronize on fast loop
Default value	0

4.9.10 Object 2108h: Filter variable 16bit

This object applies a first order low pass filer on a 16 bit variable value. It does not affect the motor control when applied. It can be used only for sampling filtered values of one variable like the motor current.

Object description:

Index	2108h
Name	Filter variable 16bit
Object code	Record
Data type	Filter variable record

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	3
Default value	3

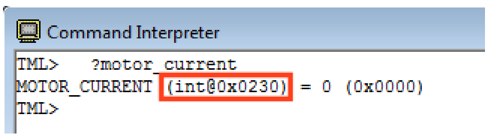
Sub-index	1
Description	16 bit variable address
Access	RW
PDO mapping	Possible
Value range	UNSIGNED16
Default value	0x0230 (address. or motor current)

Sub-index	2
Description	Filter strength
Access	RW
PDO mapping	Possible
Value range	UNSIGNED16
Default value	50

Sub-index	3
Description	Filtered variable 16bit
Access	RO
PDO mapping	Possible
Value range	0 -32767
Default value	-

How it works:

Sub-index 1 sets the filtered variable address. To find a variable address, in EasySetup or Easy Motion Studio, click View/ Command Interpreter. The communication must be online with the drive. Write the desired variable name with a ? in front and press Enter.



The variable address can be found between the parenthesis.

Sub-index 2 sets the filter strength. The filter is strongest when Sub-index 2 = 0 and weakest when it is 32767. A strong filter increases the time lag between the unfiltered variable change and the filtered value reaching that value.

Sub-index 3 shows the filtered value of the 16 bit variable whose address is declared in Sub-index 1.

4.9.11 Object 208Eh: Auxiliary Settings Register

This object is used as a configuration register that enables various advanced control options.

Object description:

Index	208Eh
Name	Auxiliary Settings Register
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	0x0100

***Table 4.9.2** – Bit Assignment in Auxiliary Settings Register*

Bit	Value	Description
9-15	0	Reserved.
8	0	Set interpolation mode compatible with PT and PVT (legacy)
	1	Set interpolation mode (when 6060=7) as described in the CiA402 standard
4-7	0	Reserved
3	0	When 6040 bit 14 = 1, at the next *update*[1], the Target Speed Starting Value is the Actual Speed
	1	When 6040 bit 14 = 1, at the next *update*, the Target Speed Starting Value is zero.
0-2	0	Reserved.

4.9.12 Object 210Bh: Auxiliary Settings Register2

This object is used as a configuration register that enables various advanced control options. The bits in this object are linked to the internal register ASR2.

Object description:

Index	210Bh
Name	Auxiliary Settings Register2
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	0x0000

***Table 4.9.3** – Bit Assignment in Auxiliary Settings Register2*

Bit	Value	Description
13-15	0	Reserved.
12	0	Set actual position to the value of the homing offset 607Ch at the end of the homing procedure
	1	After finishing a homing procedure, do not reset the actual position. Homing ends keeping position on home switch.
0-11	0	Reserved

4.9.13 Object 20A0h: Load Position and Speed monitoring[2]

This object shows the position and speed of the load sensor, when its functionality is set as only monitoring (not used in position control). The load sensor functionality can be selected using the Motor Setup dialogue. The object is not affected by Factor Group settings – it will always return values in IU.

Object description:

Index	20A0h
Name	Load Position and Speed monitoring
Object code	VAR
Data type	INTEGER32

Entry description:

[1] *update* can mean a 0 to 1 transition of bit4 in Controlword or setting a new value into object 60FFh while in velocity mode

[2] Object 20A0h is available only in F515K / FA00C firmware or newer.

Sub-index	00h
Description	Number of sub-indexes
Access	RO
PDO mapping	Yes
Default value	3

Sub-index	01h
Description	Reserved
Access	RO
PDO mapping	-
Default value	-

Sub-index	02h
Description	Load Position Monitor
Access	RO
PDO mapping	Yes
Default value	-

Sub-index	03h
Description	Load Speed Monitor
Access	RO
PDO mapping	Yes
Default value	-

4.9.14 Object 2100h: Number of steps per revolution

This object shows the number of motor steps per revolution in case a stepper motor is used. This number is defined automatically in Motor Setup when configuring the motor data.

Object description:

Index	2100h
Name	Number of steps per revolution
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	Yes
Value range	INTEGER16
Default value	-

4.9.15 Object 2101h: Number of microsteps per step

This object shows the number of motor microsteps per step in case a stepper open loop configuration is used. This number is defined automatically when configuring Drive Setup.

Object description:

Index	2101h
Name	Number of microsteps per step
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	Yes
Value range	INTEGER16
Default value	-

4.9.16 Object 2103h: Number of encoder counts per revolution

This object shows the number of encoder counts for one full motor rotation.

For example, if this object indicates 4000 and a 4000IU position command is given, the motor will rotate 1 full mechanical rotation.

Remark: this object will not indicate a correct number in case a Brushed DC motor is used.

Object description:

Index	2103h
Name	Number of encoder counts per revolution
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Yes
Value range	INTEGER32
Default value	-

4.9.17 Object 2091$_h$: Lock EEPROM[1]

This object can be used to lock/unlock the EEPROM data from being written. By reading it, it also acts as a status. Once TML or Setup data is written into the drive memory, it can be protected from being overwritten by using this object. If the EEPROM memory is already locked, it can be unlocked using this object in order to write new setup data.

Object description:

Index	2091$_h$
Name	Lock EEPROM
Object code	VAR
Data type	UNSIGNED8

Entry description:

Access	RW
PDO mapping	NO
Value range	UNSIGNED8
Default value	0

Table 4.9.4 *– Bit Assignment in Lock EEPROM*

Bit	Value	Meaning
2-7	0	Reserved.
0	0	EEPROM is unlocked.
	1	EEPROM is locked.

4.9.18 Object 2092$_h$: User Variables[2]

This object contains 4x sub-indexes, each a 32bit User Variable. These variables are directly linked to parameters present in the template and their values can be saved using object 2076$_h$ *Save current configuration*.

The variables are named: *UserVar1, UserVar2, UserVar3* and *UserVar4*. They are linked to sub-index 1 to 4 of this object.

Object description:

Index	2092$_h$
Name	User Variables
Object code	ARRAY
Data type	ULONG32

Entry description:

Sub-index	00$_h$
Description	Number of sub-indexes
Access	RO
PDO mapping	No
Default value	4

Sub-index	01$_h$ – 04$_h$
Description	UserVar1 - 4
Access	RW
PDO mapping	Possible
Value range	ULONG32
Default value	No

[1] Object 2091$_h$ is available only on firmware F515x

[2] Object 2092$_h$ is available only on firmware F515x

5 Factor group

The iPOS drives family offers the possibility to interchange physical dimensions and sizes into the device internal units. This chapter describes the factors that are necessary to do the interchanges.

The factors defined in Factor Group set up a relationship between device internal units and physical units.

The factor group settings currently implemented are complying with:

- Factor group objects - CiA 402-2 and later versions – starting with firmware version F515K / FA00x
- **Factor group objects - CiA-402 (obsolete)** – for other firmware versions

5.1 Factor group objects - CiA-402 (obsolete)

The actual factors used for scaling are the *position factor* (object 6093_h), the *velocity encoder factor* (object 6094_h), the *acceleration factor* (object 6097_h) and the *time encoder factor* (object 2071_h). Writing a non-zero value into the respective dimension index objects validates these factors. The notation index objects are used for status only and can be set by the user depending on each user-defined value for the factors.

Because the iPOS drives work with Fixed 32 bit numbers (not floating point), some calculation round off errors might occur when using objects 6093_h, 6094_h, 6097_h and 2071_h. If the ECAT master supports handling the scaling calculations on its side, it is recommended to use them instead of using the "*Factor*" scaling objects.

5.1.1 Object 607Eh: Polarity

This object is used to multiply by 1 or -1 position and velocity objects. The object applies only to position profile, velocity profile, CSP and CSV in modes of operation.

Object description:

Index	$607E_h$
Name	Polarity
Object code	VAR
Data type	UNSIGNED8

Entry description:

Access	RW
PDO mapping	Possible
Value range	0..256
Default value	0

The *Polarity* object has the following bit assignment:

***Table 5.1.1** – Bit Assignment in Polarity object*

Bit	Bit name	Value	Meaning
7	Position polarity	0	Multiply by 1 the values of objects $607A_h$, 6062_h and 6064_h
		1	Multiply by -1 the values of objects $607A_h$, 6062_h and 6064_h
6	Velocity polarity	0	Multiply by 1 the values of objects $60FF_h$, $606B_h$ and $606C_h$
		1	Multiply by -1 the values of objects $60FF_h$, $606B_h$ and $606C_h$
5-0	reserved	0	Reserved

The default value for this object can be changed by editing the parameter "POLARITY" found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

5.1.2 Object 6089h: Position notation index

The *position notation index* is used to define the position into [SI] units. Its purpose is purely informative for EtherCAT masters which still use it and has no influence over the actual unit scaling. In the CiA 402 standard, the dimension and notion index objects have been declared as obsolete. In case a custom position scaling is used, set it to 1 instead of 0.
For position scaling, use Object 6093h: Position factor.
A list of predefined values can be found in the Dimension/Notation Index Table.

Object description:

Index	6089h
Name	Position notation index
Object code	VAR
Data type	INTEGER8

Entry description:

Access	RW
PDO mapping	Possible
Value range	-128 … 127
Default value	0

5.1.3 Object 608Ah: Position dimension index

The *position dimension index* is used to define the position into [SI] units. Its purpose is purely informative for EtherCAT masters which still use it and has no influence over the actual unit scaling. In the CiA 402 standard, the dimension and notion index objects have been declared as obsolete. In case a custom position scaling is used, set it to 1 instead of 0. For position scaling, use Object 6093h: Position factor.

A list of predefined values can be found in the Dimension/Notation Index Table.

Object description:

Index	608Ah
Name	Position dimension index
Object code	VAR
Data type	UNSIGNED8

Entry description:

Access	RW
PDO mapping	Possible
Value range	0 … 255
Default value	0

5.1.4 Object 608Bh: Velocity notation index

The *velocity notation index* is used to define the velocity into [SI] units. Its purpose is purely informative for EtherCAT masters which still use it and has no influence over the actual unit scaling. In the CiA 402 standard, the dimension and notion index objects have been declared as obsolete. In case a custom velocity scaling is used, set it to 1 instead of 0. For velocity scaling, use Object 6094h: Velocity encoder factor.

A list of predefined values can be found in the Dimension/Notation Index Table.

Object description:

Index	608Bh
Name	Velocity notation index
Object code	VAR
Data type	INTEGER8

Entry description:

Access	RW
PDO mapping	Possible
Value range	-128 … 127
Default value	0

5.1.5 Object 608Ch: Velocity dimension index

The *velocity dimension index* is used to define the velocity into [SI] units. Its purpose is purely informative for EtherCAT masters which still use it and has no influence over the actual unit scaling. In the CiA 402 standard, the dimension and notion index objects have been declared as obsolete. In case a custom velocity scaling is used, set it to 1 instead of 0. For velocity scaling, use Object 6094h: Velocity encoder factor.

A list of predefined values can be found in the Dimension/Notation Index Table.

Object description:

Index	608Ch
Name	Velocity dimension index
Object code	VAR
Data type	UNSIGNED8

Entry description:

Access	RW
PDO mapping	Possible
Value range	0 … 255
Default value	0

5.1.6 Object 608D$_h$: Acceleration notation index

The *acceleration notation index i*s used to define the acceleration into [SI] units. Its purpose is purely informative for EtherCAT masters which still use it and has no influence over the actual unit scaling. In the CiA 402 standard, the dimension and notion index objects have been declared as obsolete. In case a custom acceleration scaling is used, set it to 1 instead of 0. For acceleration scaling, use Object 6097h: Acceleration factor.

A list of predefined values can be found in the Dimension/Notation Index Table.

Object description:

Index	608D$_h$
Name	Acceleration notation index
Object code	VAR
Data type	INTEGER8

Entry description:

Access	RW
PDO mapping	Possible
Value range	-128 … 127
Default value	0

5.1.7 Object 608E$_h$: Acceleration dimension index

The *acceleration dimension index i*s used to define the acceleration into [SI] units. Its purpose is purely informative for EtherCAT masters which still use it and has no influence over the actual unit scaling. In the CiA 402 standard, the dimension and notion index objects have been declared as obsolete. In case a custom acceleration scaling is used, set it to 1 instead of 0. For acceleration scaling, use Object 6097h: Acceleration factor.

A list of predefined values can be found in the Dimension/Notation Index Table.

Object description:

Index	608E$_h$
Name	Acceleration dimension index
Object code	VAR
Data type	UNSIGNED8

Entry description:

Access	RW
PDO mapping	Possible
Value range	0 … 255
Default value	0

5.1.8 Object 206F$_h$: Time notation index

The *time dimension index i*s used to define the time into [SI] units. Its purpose is purely informative for EtherCAT masters which still use it and has no influence over the actual unit scaling. In the CiA 402 standard, the dimension and notion index objects have been declared as obsolete. In case a custom time scaling is used, set it to 1 instead of 0. For time scaling, use Object 2071h: Time factor.

Object description:

Index	206F$_h$
Name	Time notation index
Object code	VAR
Data type	INTEGER8

Entry description:

Access	RW
PDO mapping	Possible
Value range	-128 … 127
Default value	0

5.1.9 Object 2070h: Time dimension index

The *time dimension index is* used to define the time into [SI] units. Its purpose is purely informative for EtherCAT masters which still use it and has no influence over the actual unit scaling. In the CiA 402 standard, the dimension and notion index objects have been declared as obsolete. In case a custom time scaling is used, set it to 1 instead of 0. For time scaling, use Object 2071h: Time factor.

Object description:

Index	2070h
Name	Time dimension index
Object code	VAR
Data type	UNSIGNED8

Entry description:

Access	RW
PDO mapping	Possible
Value range	0 … 255
Default value	0

5.1.10 Object 6093h: Position factor

The *position factor* converts the drive internal position units (increments) to the desired position (in position units) into the internal format (in increments) for the drive to use.

Writing any non-zero value into the respective dimension and notation index objects activates this object.

$$Position[IU] = Position[UserUnits] \times \frac{PositionFactor.Numerator}{PositionFactor.Divisor}$$

It scales the following objects:

6064h Position actual value; 6062h Position demand value; 607Ah Target position; 6067h Position window; 6068h Following error window; 60F4h Following error actual value

Object description:

Index	6093h
Name	Position factor
Object code	ARRAY
Number of elements	2
Data type	UNSIGNED32

Entry description:

Sub-index	01h
Description	Numerator
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	1

Sub-index	02h
Description	Divisor
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	1

5.1.10.1 Setting the numerator and divisor in a factor group object. Example

Important: when small values are used, errors may occur due to the internal calculation round off errors. In order to avoid this, use larger values giving the same desired ratio Example = 6093.1 = 0x20000 and 6093.2 = 0x10000. This will mean a factor of 2:1. In case 6093.1 = 0x2 and 0x6093.2 = 0x1, the position would not be computed correctly. As a general rule, the bigger the numerator and denominator values are, the more precise is the fraction calculation.

Example

The desired user position units are radians. The drive internal position units are encoder counts. The load is connected directly to the motor shaft and the motor has a 500-lines incremental encoder.

The conversion between user and internal units is:

$$Position[rad] \times \frac{(4 \times 500)}{(2 \times \pi)} = Position[UserUnits]$$

Hence (6093.2/6093.1) = 2 * pi / (4 x 500) = 0.0031415926535897932384626433832795…

How to set the 2 numbers? Being a number less than 1, the denominator (6093.1) is bigger than the numerator (6093.2). Hence set the denominator to the largest integer value for 32 bits i.e. 0xFFFF FFFF = 4294967295 and the numerator to

0.0031415926535897932384626433832795 x 4294967295 = 13493037.701380426305009189410434, rounded to integer i.e. = 13493038.

In conclusion: 6093.1 = 4294967295 (0xFFFF FFFF) and 6093.2 = 13493038 i.e. user position [rad] * 4294967295 / 13493038 = internal position [counts]

5.1.11 Object 6094$_h$: Velocity encoder factor

The *velocity encoder factor* converts the desired velocity (in velocity units) into the internal format (in increments) for the drive to use.

Writing any non-zero value into the respective dimension and notation index objects activates this object.

$$Velocity[IU] = Velocity[UserUnits] \times \frac{VelocityEncoderFactor.Numerator}{VelocityEncoderFactor.Divisor}$$

It scales the following objects:

606C$_h$ Velocity actual value; 606B$_h$ Velocity demand value; 606F$_h$ Velocity threshold; 60FF$_h$ Target velocity; 60F8$_h$ Max slippage; 6081$_h$ Profile velocity

To configure the object with optimal values, see *Setting the numerator and divisor in a factor group object. Example*.

Object description:

Index	6094$_h$
Name	Velocity encoder factor
Object code	ARRAY
Number of elements	2
Data type	UNSIGNED32

Entry description:

Sub-index	01$_h$
Description	Numerator
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	1

Sub-index	02$_h$
Description	Divisor
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	1

5.1.12 Object 6097$_h$: Acceleration factor

The *acceleration factor* converts the velocity (in acceleration units/sec^2) into the internal format (in increments/sampling^2) for the drive to use.

Writing any non-zero value into the respective dimension and notation index objects activates this object.

$$Acceleration[IU] = Acceleration[UserUnits] \times \frac{AccelerationFactor.Numerator}{AccelerationFactor.Divisor}$$

It scales the following objects:

6083$_h$ Profile acceleration; 6085$_h$ Quick stop deceleration

To configure the object with optimal values, see *Setting the numerator and divisor in a factor group object. Example*.

Object description:

Index	6097$_h$
Name	Acceleration factor
Object code	ARRAY
Number of elements	2
Data type	UNSIGNED32

Entry description:

Sub-index	01_h
Description	Numerator
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	1

Sub-index	02_h
Description	Divisor
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	1

5.1.13 Object 2071h: Time factor

The *time factor* converts the desired time values (in time units) into the internal format (in speed / position loop samplings) for the drive to use.

Writing any non-zero value into the respective dimension and notation index objects activates this object.

$$Time[IU] = Time[UserUnits] \times \frac{TimeFactor.Numerator}{TimeFactor.Divisor}$$

It scales the following objects:

6066h Following error time out; 6068h Position window time; 2023h Jerk time; 2005h Max slippage time out; 2051h Over-current time out

To configure the object with optimal values, see *Setting the numerator and divisor in a factor group object. Example*.

Object description:

Index	2071_h
Name	Time factor
Object code	ARRAY
Number of elements	2
Data type	UNSIGNED32

Entry description:

Sub-index	01_h
Description	Numerator
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	1

Sub-index	02_h
Description	Divisor
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	1

5.2 Factor group objects - CiA-402-2

The user-defined units are translated to internal units (IU) by the factor / scaling objects: 6093h (Position factor), 6094h (Velocity encoder factor), 210Fh (Acceleration encoder factor) and 2110h (Jerk encoder factor). For the calculation of the respective values (and their physical units) specific formulas presented in the chapter are used.

Remark: *This feature is available starting with firmware versions F515K / FA0xx.*

All units are specified using a 32-bit notation index[1] that have no influence over any scaling. Their purpose is only to define an [SI] unit name (rpm, rad, deg, etc) and their exponent (prefix). The SI unit objects are: 60A8h (SI unit position), 60A9h (SI unit velocity), 60AAh (SI unit acceleration) and 60ABh (SI unit jerk).

Table 5.2 – SI Objects Structure

MSB **LSB**

Prefix			SI numerator			SI denominator			Profile-specific		
31	…	24	23	…	16	15	...	8	7	...	0

If the SI base unit is used, the bit field SI numerator contains the notation index of the base unit. The SI denominator is not used and its bit field is equal to 1. If SI derived units are used, the SI numerator bit field contains the notation index corresponding to the numerator of the unit and the SI denominator contains the notation index corresponding to the denominator of the unit. Additionally, the parameter definition may contain notation index for profile specific units.

Listed in the following table are the possible exponents (prefixes) and their values:

Table 5.3 – Prefix Representation[1]

Prefix	Factor	Symbol	Notation Index
kilo	10^3	k	03
-	10^0	-	00
milli	10^{-3}	m	FD
micro	10^{-6}	µ	FA

Listed in the following table all default units for the SI numerator field:

Table 5.4 – Notation Index for SI Numerator [1]

Name	Symbol	Notation Index	Description
Internal Unit	IU(inc)	B5	Encoder counts. Dependent on the used sensor configuration. It's value can be found also in object 2103h: Number of encoder counts per revolution.
Step	IU(step)	AC	Available only for step motors. The value can be computed as object 2100h: Number of steps per revolution multiplied by object 2101h: Number of microsteps per step.
Radian	rad	10	Radian
Degree	deg	41	Degrees
Mechanical Revolution	rot	B4	Revolution
Meter	m	01	Available only if transmission type is rotary to linear or linear to linear.
Dimensionless	-	00	Dimensionless length unit

Listed in the following table are all default units for the SI denominator:

Table 5.5 – Notation Index for SI Denominator[1]

Name	Symbol	Notation Index
Second	s	03
Minute	min	47
Square Second	s^2	57
Cubic second	s^3	A0

[1] Specified in CiA-303-2 v.1.5.0/27.04.2015 Recommendation – "Part 2: Representation of SI units and prefixes"

If needed, the full list of notation indexes is specified in CiA-303-2 v.1.5.0/27.04.2015 Recommendation – “Part 2: Representation of SI units and prefixes”.

5.2.1 Object 60A8$_h$: SI unit position

This object indicates the user-defined position units. The object structure is defined in table Table 5.2 – *SI Objects Structure*. The profile specific field (bit 0 to bit 7) of this object is reserved (00$_h$).

Object description:

Index	60A8$_h$
Name	SI unit position
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	0x00B50100

Example:

- If the object is configured in deg:

MSB **LSB**

Prefix	SI numerator	SI denominator	Profile-specific
00$_h$ (means 10^0)	41$_h$(means deg)	01$_h$(default)	00$_h$(default)
31 … 24	23 … 16	15 ... 8	7 ... 0

- If the object is configured in mm:

MSB **LSB**

Prefix	SI numerator	SI denominator	Profile-specific
FD$_h$ (means 10^{-3})	01$_h$(means m)	01$_h$(default)	00$_h$(default)
31 … 24	23 … 16	15 ... 8	7 ... 0

5.2.2 Object 6093$_h$: Position Factor / Position Scaling

The object converts all values of length of the application from position internal units (IU) to position units (PU). Its value takes into consideration three objects: Position Encoder Resolution - 608F$_h$, Gear Ratio - 6091$_h$ and Feed Constant - 6092$_h$.

The calculation of the position factor is done using the following equation:

$$Position\ Internal\ Units\ (IU) = \frac{Position\ Units\ (PU) \times Position\ Encoder\ Resolution\ \times Gear\ Ratio}{Feed\ Constant}$$

The Position Units are computed automatically by EasyMotion Studio for each mechanical setup (rot-rot / rot-lin / lin-lin transmission) and each position sensor configuration (type, on motor or on load).

Object description:

Index	6093$_h$
Name	Position Factor / Position Scaling
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	2
Default value	2

Sub-index	1
Description	Position internal units (IU)
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

Sub-index	2
Description	Position units (PU)
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

5.2.3 Object 608F$_h$: Position Encoder Resolution

The object indicates the configured encoder increments and the number of motor revolutions. The position encoder resolution is calculated as follows:

$$Position\ Encoder\ Resolution = \frac{Encoder\ Increments}{Motor\ Rotation}$$

Object description:

Index	608F$_h$
Name	Position encoder resolution
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	2
Default value	2

Sub-index	1
Description	Encoder increments
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x000007D0 (2000 IU)

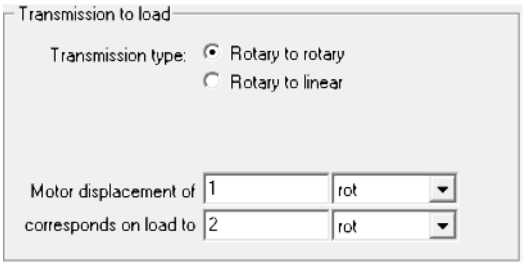
Sub-index	2
Description	Motor rotation
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

5.2.4 Object 6091$_h$: Gear Ratio

The object indicates the configured number of load rotations corresponding to the number of motor rotations. The gear ratio is calculated as follows:

$$Gear\ Ratio = \frac{Motor\ Rotation}{Load\ Rotation}$$

In EasyMotion Studio, this object is automatically configured in the Motor Setup window – “Transmission to load” dialogue:



Object description:

Index	6091$_h$
Name	Gear Ratio
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	2
Default value	2

Sub-index	1
Description	Motor rotation
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

Sub-index	2
Description	Load rotation
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

5.2.5 Object 6092$_h$: Feed Constant

The object indicates the measurement distance per one rotation of the driving shaft of the gearbox. The feed constant is calculated as follows:

$$Feed\ Constant = \frac{Feed}{Driving\ Shaft\ Rotation}$$

The feed is given in user-defined position units, and the driving shaft revolutions value is dimensionless.

Object description:

Index	6092$_h$
Name	Feed Constant
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	2
Default value	2

Sub-index	1
Description	Feed
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

Sub-index	2
Description	Shaft Rotations
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

5.2.6 Object 60A9$_h$: SI unit velocity

This object indicates the user-defined velocity units. The object structure is defined in Table 5.2 – *SI Objects Structure*. The profile specific field (bit 0 to bit 7) of this object is reserved (00$_h$).

Object description:

Index	60A9h
Name	SI unit velocity
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	0x00000000

Example:

- If the object is configured in rpm:

MSB **LSB**

Prefix			SI numerator			SI denominator			Profile-specific		
00h (means 10^0)			B4h(means rot)			47h(means min)			00h(default)		
31	…	24	23	…	16	15	...	8	7	...	0

- If the object is configured in mm/s:

MSB **LSB**

Prefix			SI numerator			SI denominator			Profile-specific		
FDh (means 10^{-3})			01h(means m)			03h(means s)			00h(default)		
31	…	24	23	…	16	15	...	8	7	...	0

5.2.7 Object 6094h: Velocity encoder factor

The object converts all values of speed of the application from velocity internal units (IU) to velocity units (VU). Its value takes into consideration two objects: Position Factor / Position Scaling - 6093h and Velocity Factor - 6096h.

The calculation of the position factor is done using the following equation:

$$Velocity\ Internal\ Units\ (IU) = \frac{Velocity\ Units\ (VU) \times Position\ Factor}{Velocity\ Factor} \times T \times 2^{16}\ \ where\ T = slow\ loop\ period$$

The Velocity Units are computed automatically by EasyMotion Studio for each mechanical setup (rot-rot / rot-lin / lin-lin transmission), position sensor configuration (type, on motor or on load) and slow loop period.

Object description:

Index	6094h
Name	Velocity encoder factor
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	2
Default value	2

Sub-index	1
Description	Velocity internal units (IU)
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001
Sub-index	2
Description	Velocity units (VU)
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

5.2.8 Object 6096h: Velocity Factor

The object converts PU (position units) per second into VU (velocity units). The calculation of the velocity factor is done using the following equation:

$$Velocity\ Factor = \frac{Velocity\ Units\ (VU)}{\frac{Position\ Units\ (PU)}{s}}$$

For example, if the user defined position unit is radian (rad) and the user defined velocity unit is rpm, the velocity factor will be 60/2/π. If the user defined position unit is radian (rad) and the user defined velocity unit is rad/s, the velocity factor will be 1.

Object description:

Index	6096h
Name	Velocity Factor
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	2
Default value	2

Sub-index	1
Description	Velocity units (VU)
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

Sub-index	2
Description	Position units (PU)
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

5.2.9 Object 60AAh: SI unit acceleration

This object indicates the user-defined acceleration units. The object structure is defined in Table 5.2 – *SI Objects Structure*. The profile specific field (bit 0 to bit 7) of this object is reserved (00h).

Object description:

Index	60AAh
Name	SI unit acceleration
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	0x00000000

Example:

- If the object is configured in deg/s^2:

MSB **LSB**

Prefix	SI numerator	SI denominator	Profile-specific
00h (means 10^0)	41h(means rad)	57h(means s^2)	00h(default)
31 … 24	23 … 16	15 ... 8	7 ... 0

- If the object is configured in krad/s^2:

MSB **LSB**

Prefix			SI numerator			SI denominator			Profile-specific		
03_h (means 10^3)			10_h(means rad)			57_h(means s^2)			00_h(default)		
31	…	24	23	…	16	15	...	8	7	...	0

5.2.10 Object $210F_h$: Acceleration encoder factor

The object converts all values of acceleration of the application from acceleration internal units (IU) to acceleration units (AU). Its value takes into consideration two objects: Velocity Encoder Factor - 6094_h and Acceleration Factor - 6097_h.

The calculation of the position factor is done using the following equation:

$$Acceleration\ Internal\ Units\ (IU) = \frac{Acceleration\ \ Units\ (AU) \times Velocity\ Encoder\ Factor}{Acceleration\ Factor} \times\ T\ \ where\ T = slow\ loop\ period$$

The Acceleration Units are computed automatically by EasyMotion Studio for each mechanical setup (rot-rot / rot-lin / lin-lin transmission), position sensor configuration (type, on motor or on load) and slow loop period.

Object description:

Index	$210F_h$
Name	Acceleration encoder factor
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	2
Default value	2

Sub-index	1
Description	Acceleration internal units (IU)
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

Sub-index	2
Description	Acceleration units (AU)
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

5.2.11 Object 6097_h: Acceleration Factor

The object converts VU (velocity units) per second into AU (acceleration units). The calculation of the acceleration factor is done using the following equation:

$$Acceleration\ Factor = \frac{Acceleration\ Units\ (AU)}{\frac{Velocity\ Units\ (VU)}{s}}$$

For example, if the user defined velocity unit is rad/s and the user defined acceleration unit is krad/s^2, the acceleration factor will be 0.001. If the user defined velocity unit is rad/s and the user defined acceleration unit is rad/s^2, the acceleration factor will be 1.

Object description:

Index	6097_h
Name	Acceleration Factor
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	2
Default value	2

Sub-index	1
Description	Acceleration units (AU)
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

Sub-index	2
Description	Velocity units (VU)
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

5.2.12 Object 60AB$_h$: SI unit jerk

This object indicates the user-defined jerk units. The object structure is defined in Table 5.2 – *SI Objects Structure*. The profile specific field (bit 0 to bit 7) of this object is reserved (00$_h$).

Object description:

Index	60AB$_h$
Name	SI unit jerk
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	0x00000000

Example:

- If the object is configured in deg/s^3:

MSB **LSB**

Prefix	SI numerator	SI denominator	Profile-specific
00$_h$ (means 10^0)	41$_h$(means rad)	A0$_h$(means s^3)	00$_h$(default)
31 … 24	23 … 16	15 ... 8	7 ... 0

- If the object is configured in krad/s^3:

MSB **LSB**

Prefix	SI numerator	SI denominator	Profile-specific
03$_h$ (means 10^3)	10$_h$(means rad)	A0$_h$(means s^3)	00$_h$(default)
31 … 24	23 … 16	15 ... 8	7 ... 0

5.2.13 Object 2110$_h$: Jerk encoder factor

The object converts all values of jerk of the application from jerk internal units (IU) to jerk units (JU). Its value takes into consideration two objects: Acceleration Encoder Factor – 210F$_h$ and Jerk Factor – 60A2$_h$.

The calculation of the position factor is done using the following equation:

$$Jerk\ Internal\ Units\ (IU) = \frac{Jerk\ Units\ (JU) \times Acceleration\ Encoder\ Factor}{Jerk\ Factor} \times T \quad where\ T = slow\ loop\ period$$

The Jerk Units are computed automatically by EasyMotion Studio for each mechanical setup (rot-rot / rot-lin / lin-lin transmission), position sensor configuration (type, on motor or on load) and slow loop period.

Object description:

Index	2110$_h$
Name	Jerk encoder factor
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	2
Default value	2

Sub-index	1
Description	Jerk internal units (IU)
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

Sub-index	2
Description	Jerk units (JU)
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

5.2.14 Object 60A2$_h$: Jerk Factor

The object converts AU (acceleration units) per second into JU (jerk units). The calculation of the jerk factor is done using the following equation:

$$Acceleration\ Factor = \frac{Jerk\ Units\ (JU)}{\frac{Acceleration\ Units\ (AU)}{s}}$$

For example, if the user defined acceleration unit is rad/s^2 and the user defined jerk unit is krad/s^3, the jerk factor will be 0.001. If the user defined acceleration unit is rad/s^2 and the user defined jerk unit is rad/s^3, the jerk factor will be 1.

Object description:

Index	60A2$_h$
Name	Jerk Factor
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	2
Default value	2

Sub-index	1
Description	Jerk Units (JU)
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

Sub-index	2
Description	Acceleration Units (AU)
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

6 Homing Mode

6.1 Overview

Homing is the method by which a drive seeks the home position. There are various methods to achieve this position using the four available sources for the homing signal: limit switches (negative and positive), home switch (IN0) and encoder index pulse.

Remark: on an iPOS drive or iMOT intelligent motor, the "home switch" is always the digital input IN0.

A homing move is started by setting bit 4 of the *Controlword* object 6040h. The results of a homing operation can be accessed in the *Statusword* (index 0x6041).

After the physical home position is found, the drive actual position (object 6064h or internal variable APOS) will be set with the value of

Object 607Ch: Home offset.

A homing mode is chosen by writing a value to homing method (object 6098h) which will clearly establish:

1. the used homing signal (positive limit switch, negative limit switch, home switch or index pulse)
2. the initial direction of motion
3. the position of the index pulse (if used).

The user can specify the home method, the home offset, two homing speeds and the acceleration.

The **home offset** (object 607Ch) is the difference between the zero position for the application and the machine home position. During homing, the home position is found. Once the homing is completed, the zero position is offset from the home position by adding the home offset to the home position. This is illustrated in the following diagram.

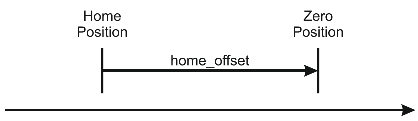


Figure 6.1.1. *Home Offset*

In other words, after the home position has been found, the drive will set the actual position (object 6064h) with the value found in object 607Ch.

There are two **homing speeds:** a fast speed (which is used for the initial motion to find the home switch), and a slow speed (which is used after the home switch transition, when the motion is reversed).

The **homing acceleration** establishes the acceleration to be used for all accelerations and decelerations with the standard homing modes.

The homing method descriptions in this document are based on those in the Profile for Drives and Motion Control (CiA402 or IEC61800 Standard).

The figure below explains the homing method 1 diagram in detail. The other homing method diagrams are similar and the explanation below applied to all of them.

The colors **black** and **grey** represent the original homing diagram as explained in the CiA 402 standard.

The **green** color represents the explanation for the various elements in the diagram.

The **purple** color represents the motion explanation for the current homing method diagram.

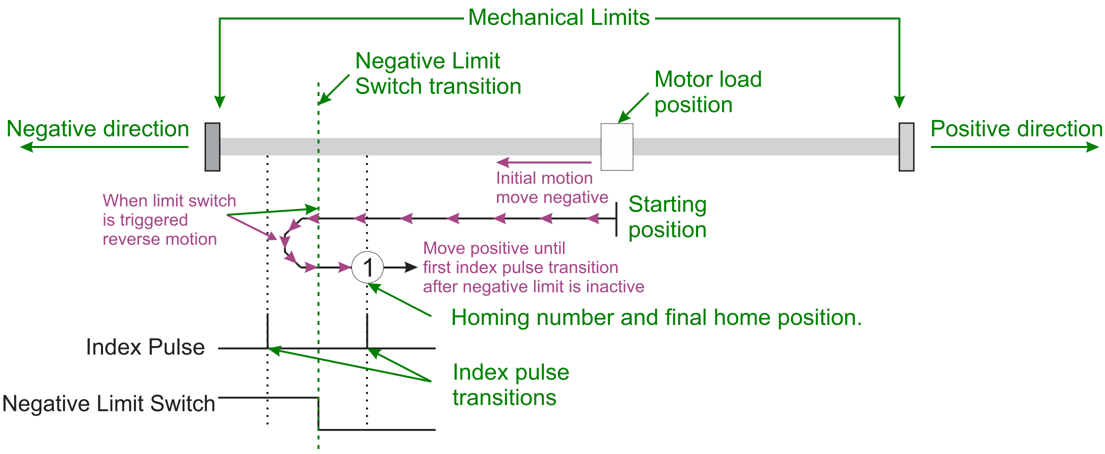


Figure 6.1.2. Homing method 1 diagram explained

6.2 Homing methods

6.2.1 Method 1: Homing on the Negative Limit Switch and Index Pulse

If the negative limit switch is inactive (low) the initial direction of movement is leftward (negative sense). After negative limit switch is reached the motor will reverse the motion, moving in the positive sense with slow speed. The home position is at the first index pulse to the right of the position where the negative limit switch becomes inactive.

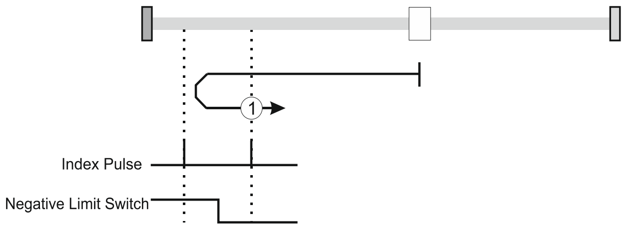


Figure 6.2.1. *Homing on the Negative Limit Switch and Index Pulse*

6.2.2 Method 2: Homing on the Positive Limit Switch and Index Pulse

If the positive limit switch is inactive (low) the initial direction of movement is rightward (negative sense). After positive limit switch is reached the motor will reverse the motion, moving in the negative sense with slow speed. The home position is at the first index pulse to the left of the position where the positive limit switch becomes inactive.

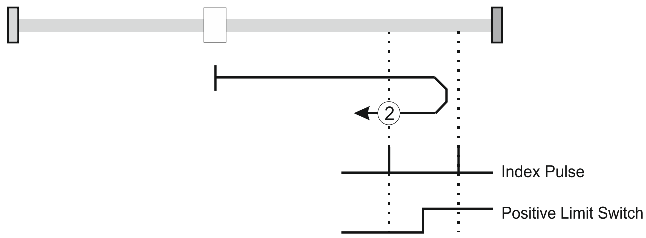


Figure 6.2.2. *Homing on the Positive Limit Switch and Index Pulse*

6.2.3 Methods 3 and 4: Homing on the Positive Home Switch and Index Pulse.

The home position is at the index pulse either after home switch high-low transition (method 3) or after home switch low-high transition (method 4).

The diagram shows two initial movements for each type of method. This is because the initial direction of movement is dependent on the state of the home switch (if low - move positive, if high - move negative).

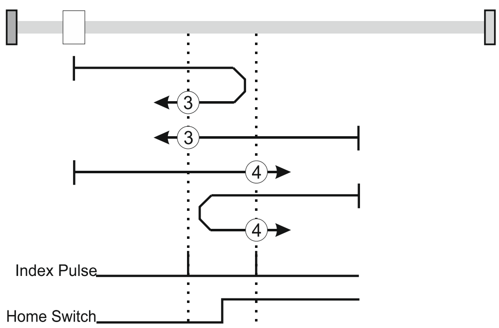


Figure 6.2.3. *Homing on the Positive Home Switch and Index Pulse*

For **method 3**, if home input is high the initial direction of movement will be negative, or positive if home input is low, and reverse (with slow speed) after home input low-high transition. The motor will stop at first index pulse after home switch high-low transition.

For **method 4**, if home input is low the initial direction of movement will be positive, or negative if home input is high, and reverse (with slow speed) after home input high-low transition. The motor will stop at first index pulse after home switch low-high transition.

In all cases after home switch transition, the speed of the movement is slow.

6.2.4 Methods 5 and 6: Homing on the Negative Home Switch and Index Pulse.

The home position is at the index pulse either after home switch high-low transition (method 5) or after home switch low-high transition (method 6).

The initial direction of movement is dependent on the state of the home switch (if high - move positive, if low - move negative).

In all cases after home switch transition, the speed of the movement is slow.

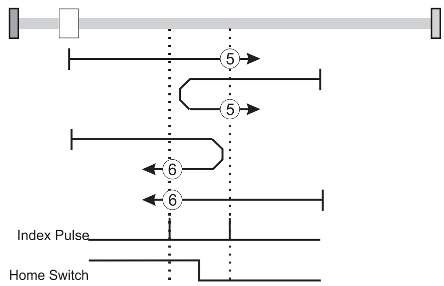


Figure 6.2.4. *Homing on the Negative Home Switch and Index Pulse*

For **method 5**, if home input is high the initial direction of movement will be positive, or negative if home input is low, and reverse (with slow speed) after home input low-high transition. The motor will stop at first index pulse after home switch high-low transition.

For **method 6**, if home input is low the initial direction of movement will be negative, or positive if home input is high, and reverse (with slow speed) after home input high-low transition. The motor will stop at first index pulse after home switch low-high transition.

6.2.5 Methods 7 to14: Homing on the Home Switch using limit switches and Index Pulse.

These methods use a home switch that is active over only a portion of the travel distance; in effect the switch has a 'momentary' action as the axle's position sweeps past the switch.

Using methods 7 to 10 the initial direction of movement is to the right (positive), and using methods 11 to 14 the initial direction of movement is to the left (negative), except the case when the home switch is active at the start of the motion (initial direction of motion is dependent on the edge being sought – the rising edge or the falling edge).

The home position is at the index pulse on either side of the rising or falling edges of the home switch, as shown in the following two diagrams.

If the initial direction of movement leads away from the home switch, the drive will reverse on encountering the relevant limit switch (negative limit switch for methods 7 to 10, or positive limit switch for methods 11 to 14).

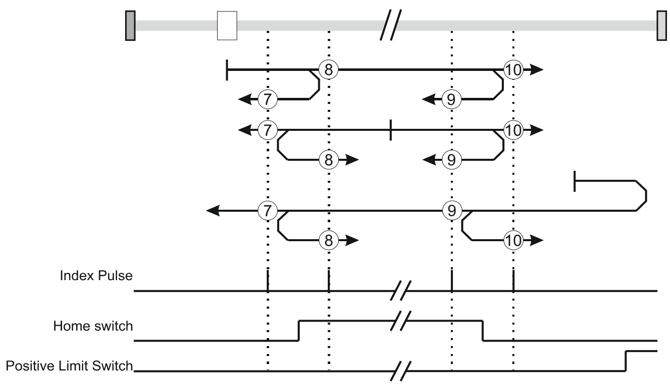


Figure 6.2.5. *Homing on the Home Switch using limit switches and Index Pulse – Positive Initial Move*

Using **method 7** the initial move will be positive if home input is low and reverse after home input low-high transition, or move negative if home input is high. Reverse also if the positive limit switch is reached. Stop at first index pulse after home switch active region ends (high-low transition). In all cases after high-low home switch transition the motor speed will be slow.

Using **method 8** the initial move will be positive if home input is low, or negative if home input is high and reverse after home input high-low transition. Reverse also if the positive limit switch is reached. Stop at first index pulse after home switch active region starts (low-high transition). In all cases after low-high home switch transition the motor speed will be slow.

Using **method 9** the initial move will be positive and reverse (slow speed) after home input high-low transition. Reverse also if the positive limit switch is reached. Stop at first index pulse after home switch active region starts (low-high transition).

Using **method 10** the initial move will be positive. Reverse if the positive limit switch is reached, then reverse once again after home input low-high transition. Stop at first index pulse after home switch active region ends (high-low transition). In all cases after high-low home switch transition the motor speed will be slow.

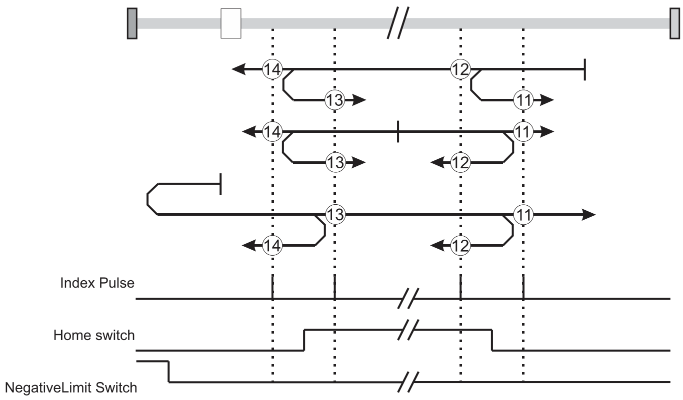


Figure 6.2.6. *Homing on the Home Switch using limit switches and Index Pulse – Negative Initial Move*

Using **method 11** the initial move will be negative if home input is low and reverse after home input low-high transition. Reverse also if the negative limit switch is reached. If home input is high move positive. Stop at first index pulse after home switch active region ends (high-low transition). In all cases after high-low home switch transition the motor speed will be slow.

Using **method 12** the initial move will be negative if home input is low. If home input is high move positive and reverse after home input high-low transition. Reverse also if the negative limit switch is reached. Stop at first index pulse after home switch active region starts (low-high transition). In all cases after low-high home switch transition the motor speed will be slow.

Using **method 13** the initial move will be negative and reverse after home input high-low transition. Reverse also if the negative limit switch is reached. Stop at first index pulse after home switch active region starts (low-high transition). In all cases after high-low home switch transition the motor speed will be slow.

Using **method 14** the initial move will be negative. Reverse if the negative limit switch is reached, then reverse once again after home input low-high transition. Stop at first index pulse after home switch active region ends (high-low transition). In all cases after high-low home switch transition the motor speed will be slow.

Methods 15 and 16: Reserved

6.2.6 Methods 17 to 30: Homing without an Index Pulse

These methods are similar to methods 1 to 14 except that the home position is not dependent on the index pulse but only on the relevant home or limit switch transitions.

6.2.7 Method 17: Homing on the Negative Limit Switch

Using **method 17** if the negative limit switch is inactive (low) the initial direction of movement is leftward (negative sense). After negative limit switch reached the motor will reverse the motion, moving in the positive sense with slow speed. The home position is at the right of the position where the negative limit switch becomes inactive.

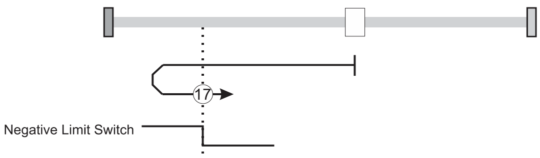


Figure 6.2.7. *Homing on the Negative Limit Switch*

6.2.8 Method 18: Homing on the Positive Limit Switch

Using **method 18** if the positive limit switch is inactive (low) the initial direction of movement is rightward (negative sense). After positive limit switch reached the motor will reverse the motion, moving in the negative sense with slow speed. The home position is at the left of the position where the positive limit switch becomes inactive.

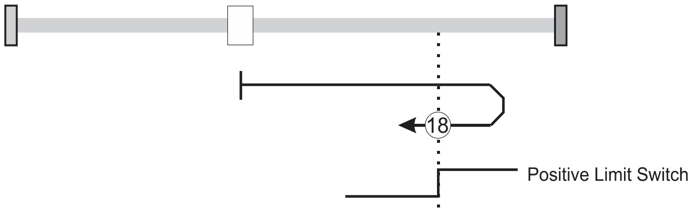


Figure 6.2.8. *Homing on the Positive Limit Switch*

6.2.9 Methods 19 and 20: Homing on the Positive Home Switch

The home position is at the home switch high-low transition (method 19) or low-high transition (method 20).

The diagram shows two initial movements for each type of method. This is because the initial direction of movement is dependent on the state of the home switch (if low - move positive, if high - move negative).

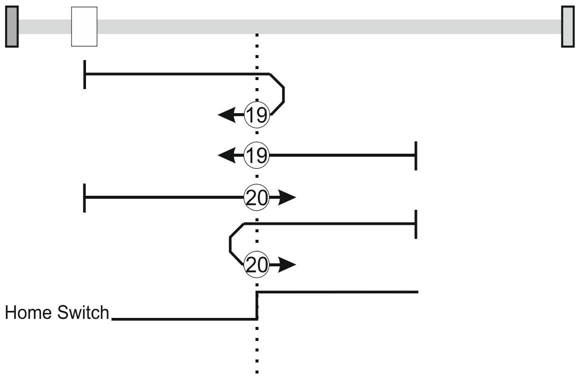


Figure 6.2.9. *Homing on the Positive Home Switch*

Using **method 19**, if home input is high, the initial direction of movement will be negative, or positive if home input is low, and reverse (with slow speed) after home input low-high transition. The motor will stop right after home switch high-low transition.

Using **method 20**, if home input is low, the initial direction of movement will be positive, or negative if home input is high, and reverse (with slow speed) after home input high-low transition. The motor will stop after right home switch low-high transition.

6.2.10 Methods 21 and 22: Homing on the Negative Home Switch

The home position is at the home switch high-low transition (method 21) or after home switch low-high transition (method 22).

The initial direction of movement is dependent on the state of the home switch (if high - move positive, if low - move negative).

In all cases after home switch transition, the speed of the movement is slow.

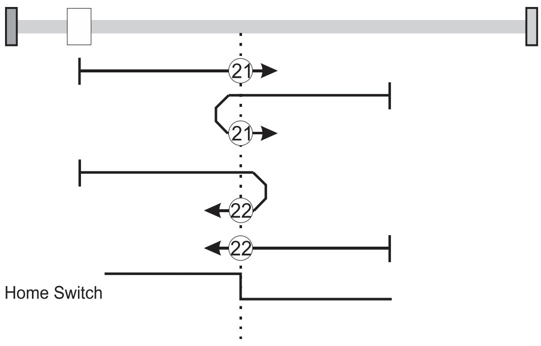


Figure 6.2.10. *Homing on the Negative Home Switch*

Using **method 21**, if home input is high, the initial direction of movement will be positive, or negative if home input is low, and reverse (with slow speed) after home input low-high transition. The motor will stop right after home switch high-low transition.

Using **method 22**, if home input is low, the initial direction of movement will be negative, or positive if home input is high, and reverse (with slow speed) after home input high-low transition. The motor will stop right after home switch low-high transition.

6.2.11 Methods 23 to30: Homing on the Home Switch using limit switches

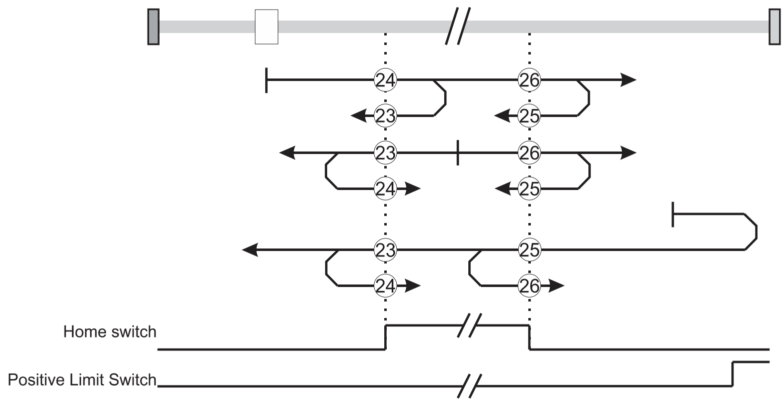


Figure 6.2.11. *Homing on the Home Switch using limit switches – Positive Initial Move*

Using **method 23** the initial move will be positive if home input is low and reverse after home input low-high transition, or move negative if home input is high. Reverse also if the positive limit switch is reached. Stop right after home switch active region ends (high-low transition).

Using **method 24** the initial move will be positive if home input is low, or negative if home input is high and reverse after home input high-low transition. Reverse also if the positive limit switch is reached. Stop right after home switch active region starts (low-high transition).

Using **method 25** the initial move will be positive and reverse after home input high-low transition. Reverse also if the positive limit switch is reached. Stop right after home switch active region starts (low-high transition).

Using **method 26** the initial move will be positive. Reverse if the positive limit switch is reached, then reverse once again after home input low-high transition. Stop right after home switch active region ends (high-low transition).

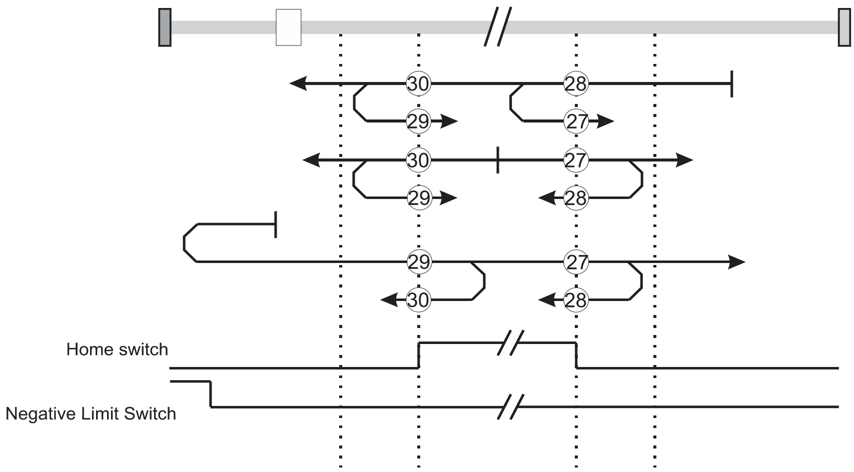


Figure 6.2.12. *Homing on the Home Switch using limit switches – Negative Initial Move*

Using **method 27** the initial move will be negative if home input is low and reverse after home input low-high transition. Reverse also if the negative limit switch is reached. If home input is high move positive. Stop right after home switch active region ends (high-low transition).

Using **method 28** the initial move will be negative if home input is low. If home input is high move positive and reverse after home input high-low transition. Reverse also if the negative limit switch is reached. Stop right after home switch active region starts (low-high transition).

Using **method 29** the initial move will be negative and reverse after home input high-low transition. Reverse also if the negative limit switch is reached. Stop right after home switch active region starts (low-high transition).

Using **method 30** the initial move will be negative. Reverse if the negative limit switch is reached, then reverse once again after home input low-high transition. Stop right after home switch active region ends (high-low transition).

Methods 31 and 32: Reserved

6.2.12 Methods 33 and 34: Homing on the Index Pulse

Using **methods 33** or **34** the direction of homing is negative or positive respectively. During these procedures, the motor will move only at slow speed. The home position is at the index pulse found in the selected direction.

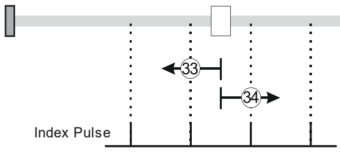


Figure 6.2.13. *Homing on the Index Pulse*

6.2.13 Method 35: Homing on the Current Position

In **method 35** the current position set with the value of home position (object 607Cₕ).

Remark: see also *Object 2081h: Set/Change the actual motor position* which can be used to obtain the same outcome as in Method 35.

6.2.14 Method -1: Homing on the Negative Mechanical Limit and Index Pulse

6.2.14.1 Method -1 based on motor current increase

This method applies to all closed loop motor configurations. It does not apply to Stepper Open Loop configurations.

Move negative until the "Current threshold" is reached for a specified amount of time, then reverse and stop at the first index pulse. When the motor current is greater than the *Homing Current Threshold* (index 0x207B) for a specified amount of time in the *Homing Current Threshold Time* object (index 0x207C), the motor will reverse direction. The home position is at the first index pulse to the right of the negative mechanical limit. At the end of the procedure, the reported motor position will be the one set in *Home offset* (index 0x607C).

Warning!	The value of *Homing Current Threshold* must be lower than the drive current limit. Otherwise, the homing will not complete successfully (no homing error will be issued). The current limit is set during setup. See Paragraph 1.2. Setting the current limit. *Current Threshold < current limit*

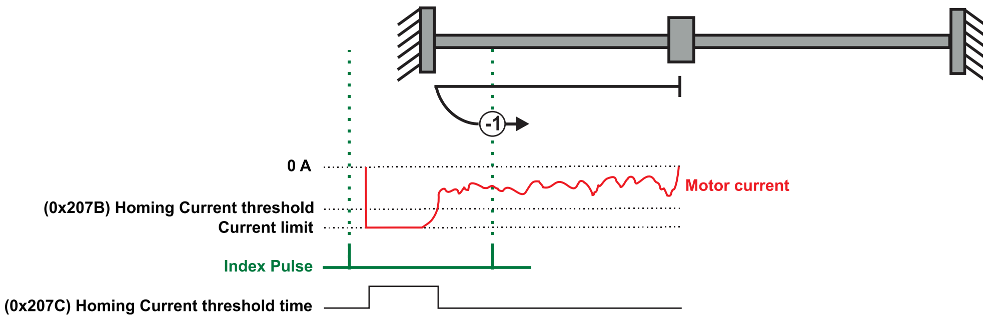


Figure 6.2.14. *Homing on the Negative Mechanical Limit and Index Pulse detecting the motor current increase*

6.2.14.2 Method -1 based on step loss detection

This method applies only to Stepper Open Loop with Encoder on motor (step loss detection) or Encoder on Load. It does not apply to Closed loop configurations or Stepper Open Loop without an incremental encoder present.

If a Stepper Open Loop with Encoder on motor (step loss detection) or Encoder on Load configuration is selected, this homing method will detect a Position control error when reaching the mechanical limit. The homing Position Control Error parameters are set in the objects 6065ₕ *Following error window* and 207Cₕ *Homing current threshold time.*

Move negative until a control error is detected, then reverse and stop at the first index pulse. The home position is at the first index pulse to the right of the negative mechanical limit. At the end of the procedure, the reported motor position will be the one set in *Home offset* (index 0x607C).

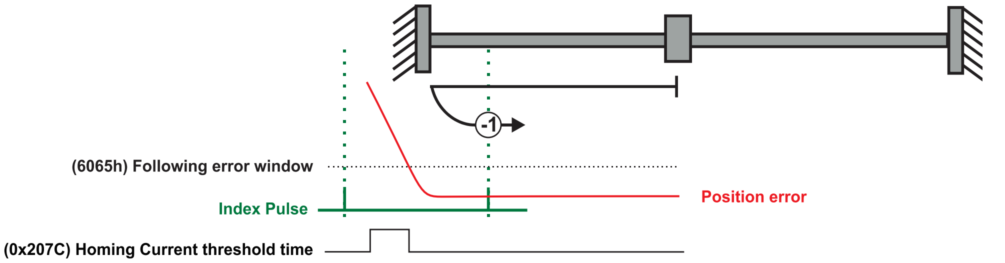


Figure 6.2.15. *Homing on the Negative Mechanical Limit and Index Pulse detecting a control error*

6.2.15 Method -2: Homing on the Positive Mechanical Limit and Index Pulse

6.2.15.1 Method -2 based on motor current increase

This method applies to all closed loop motor configurations. It does not apply to Stepper Open Loop configurations.

Move positive until the "Current threshold" is reached for a specified amount of time, then reverse and stop at the first index pulse. When the motor current is greater than the *Homing Current Threshold* (index 0x207B) for a specified amount of time in the *Homing Current Threshold Time* object (index 0x207C), the motor will reverse direction. The home position is at the first index pulse to the left of the positive mechanical limit. At the end of the procedure, the reported motor position will be the one set in *Home offset* (index 0x607C).

Warning! The value of *Homing Current Threshold* must be lower than the drive current limit. Otherwise, the homing will not complete successfully (no homing error will be issued). The current limit is set during setup. See Paragraph 1.2. Setting the current limit. *Current Threshold < current limit*

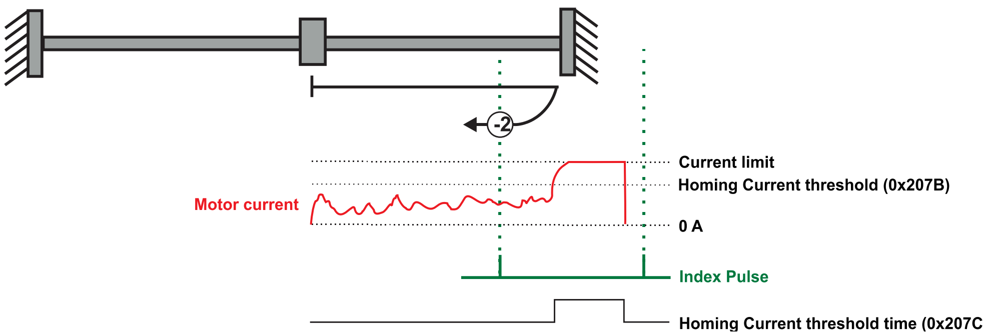


Figure 6.2.16. *Homing on the Positive Mechanical Limit and Index Pulse detecting the motor current increase*

6.2.15.2 Method -2 based on step loss detection

This method applies only to Stepper Open Loop with Encoder on motor (step loss detection) or Encoder on Load. It does not apply to Closed loop configurations or Stepper Open Loop without an incremental encoder present.

If a Stepper Open Loop with Encoder on motor (step loss detection) or Encoder on Load configuration is selected, this homing method will detect a Position control error when reaching the mechanical limit. The homing Position Control Error parameters are set in the objects 6065_h *Following error window* and $207C_h$ *Homing current threshold time.*

Move positive until a control error is detected, then reverse and stop at the first index pulse. The home position is at the first index pulse to the left of the positive mechanical limit. At the end of the procedure, the reported motor position will be the one set in *Home offset* (index 0x607C).

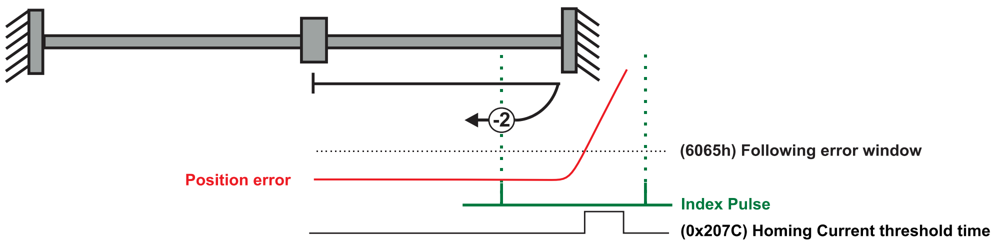


Figure 6.2.17. *Homing on the Positive Mechanical Limit and Index Pulse detecting a control error*

6.2.16 Method -3: Homing on the Negative Mechanical Limit without an Index Pulse.

6.2.16.1 Method -3 based on motor current increase

This method applies to all closed loop motor configurations. It does not apply to Stepper Open Loop configurations.

Move negative until the "Current threshold" is reached for a specified amount of time, then reverse and stop at the position set in "Home position". When the motor current is greater than the *Homing Current Threshold* (index 0x207B) for specified amount of time set in the *Homing Current Threshold Time* object (index 0x207C), the motor will reverse direction and stop after it has travelled the value set in *Home offset* (index 0x607C). At the end of the procedure, the reported motor position will be the one set in *Home offset* (index 0x607C).

Warning!	The value of *Homing Current Threshold* must be lower than the drive current limit. Otherwise, the homing will not complete successfully (no homing error will be issued). The current limit is set during setup. See Paragraph 1.2. Setting the current limit. *Current Threshold < current limit*

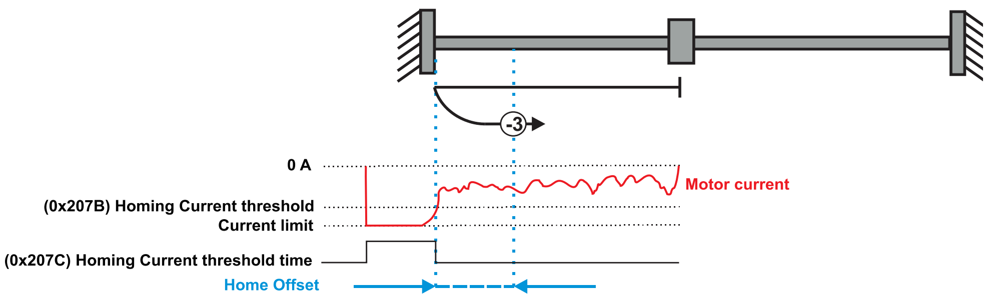


Figure 6.2.18. *Homing on the Positive Mechanical Limit without an Index Pulse detecting the motor current increase*

6.2.16.2 Method -3 based on step loss detection

This method applies only to Stepper Open Loop with Encoder on motor (step loss detection) or Encoder on Load. It does not apply to Closed loop configurations or Stepper Open Loop without an incremental encoder present.

If a Stepper Open Loop with Encoder on motor (step loss detection) or Encoder on Load configuration is selected, this homing method will detect a Position control error when reaching the mechanical limit. The homing Position Control Error parameters are set in the objects 6065h *Following error window* and 207Ch *Homing current threshold time.*

Move negative until a control error is detected, then reverse and stop at the position set in "Home position". The motor will reverse direction and stop after it has travelled the value set in *Home offset* (index 0x607C). At the end of the procedure, the reported motor position will be the one set in *Home offset* (index 0x607C).

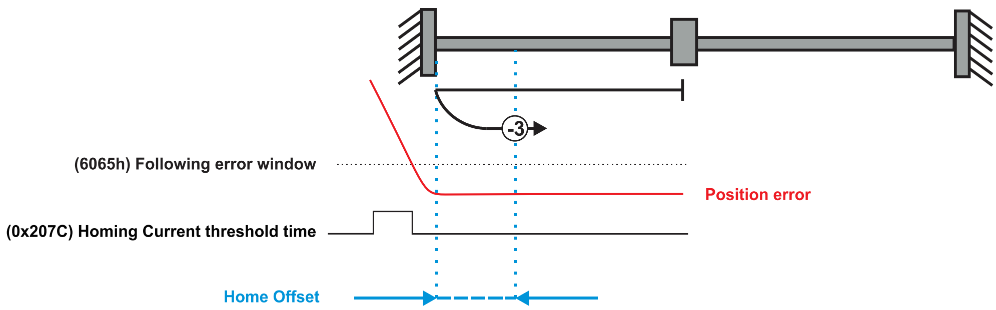


Figure 6.2.19. *Homing on the Positive Mechanical Limit without an Index Pulse detecting a control error*

6.2.17 Method -4: Homing on the Positive Mechanical Limit without an Index Pulse.

6.2.17.1 Method -4 based on motor current increase

This method applies to all closed loop motor configurations. It does not apply to Stepper Open Loop configurations.

Move positive until the "Current threshold" is reached for a specified amount of time, then reverse and stop at the position set in "Home position". When the motor current is greater than the *Homing Current Threshold* (index 0x207B) for specified amount of time set in the *Homing Current Threshold Time* object (index 0x207C), the motor will reverse direction and stop after it has travelled the absolute value set in *Home offset* (index 0x607C). At the end of the procedure, the reported motor position will be the one set in *Home offset* (index 0x607C).

	Warning!	The value of *Homing Current Threshold* must be lower than the drive current limit. Otherwise, the homing will not complete successfully (no homing error will be issued). The current limit is set during setup. See Paragraph 1.2. Setting the current limit. *Current Threshold < current limit*

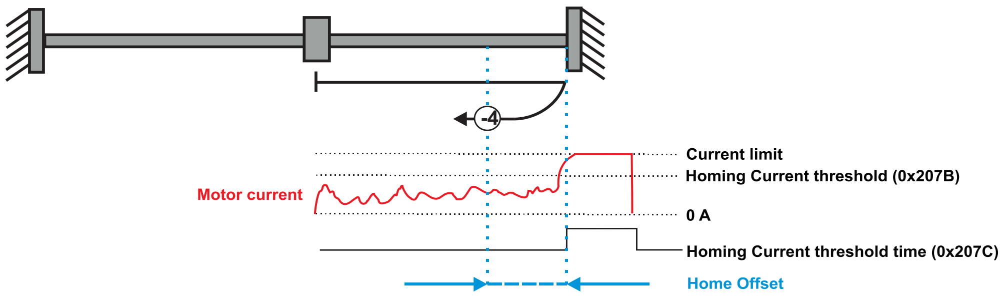


Figure 6.2.20. *Homing on the Positive Mechanical Limit without an Index Pulse detecting the motor current increase*

6.2.17.2 Method -4 based on step loss detection

This method applies only to Stepper Open Loop with Encoder on motor (step loss detection) or Encoder on Load. It does not apply to Closed loop configurations or Stepper Open Loop without an incremental encoder present.

If a Stepper Open Loop with Encoder on motor (step loss detection) or Encoder on Load configuration is selected, this homing method will detect a Position control error when reaching the mechanical limit. The homing Position Control Error parameters are set in the objects 6065_h *Following error window* and $207C_h$ *Homing current threshold time.*

Move positive until a control error is detected, then reverse and stop at the position set in "Home position". The motor will reverse direction and stop after it has travelled the value set in *Home offset* (index 0x607C). At the end of the procedure, the reported motor position will be the one set in *Home offset* (index 0x607C).

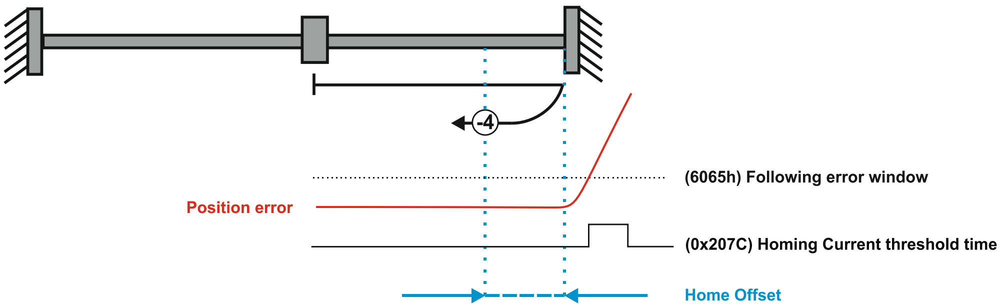


Figure 6.2.21. *Homing on the Positive Mechanical Limit without an Index Pulse detecting the motor current increase*

6.3 Homing Mode Objects

This chapter describes the method by which the drive seeks the home position. There are 35 built-in homing methods, as described in **paragraph 6.1**. Using the EasyMotion Studio software, one can alter each of these homing methods to create a custom homing method.

You can select which homing method to be used by writing the appropriate number in the object 6098_h *homing method*.

The user can specify the speeds and acceleration to be used during the homing. There is a further object *homing offset* that allows the user to displace zero in the user's coordinate system from the home position.

In the homing mode, the bits in *Controlword* and *Statusword* have the following meaning:

6.3.1 Controlword in homing mode

MSB **LSB**

See 6040_h	Halt	See 6040_h	Reserved	Homing operation start	See 6040_h
15 9	8	7	6 5	4	3 0

Table 6.3.1 – *Controlword bits description for Homing Mode*

Bit	Name	Value	Description
4	Homing operation start	0 -> 1	Only a 0 to 1 transition will start homing mode
8	Halt	0	Execute the instruction of bit 4
		1	Stop drive with *homing acceleration*

6.3.2 Statusword in homing mode

MSB LSB

See 6041h		Homing error	Homing attained	See 6041h	Target reached		See 6041h
15	14	13	12	11	10	9	0

Table 6.3.2 – *Statusword bits description for Homing Mode*

Bit	Name	Value	Description
13	Homing error	0	No homing error
		1	Homing error occurred; homing mode not carried out successfully.
12	Homing attained	0	Homing mode not yet completed
		1	Homing mode carried out successfully
10	Target reached	0	Halt = 0: Home position not reached Halt = 1: Drive decelerates
		1	Halt = 0: Home position reached Halt = 1: Velocity of drive is 0

Table 6.3.3 – *Definition of Statusword bit 10,bit 12 and bit 13 in homing mode*

Bit 13	Bit 12	Bit 10	Definition
0	0	0	Homing procedure is in progress
0	0	1	Homing procedure is interrupted or not started
0	1	0	Homing is attained, but target is not reached
0	1	1	Homing procedure is completed successfully
1	0	0	Homing error occurred, velocity is not 0
1	0	1	Homing error occurred, velocity is 0
1	1	X	reserved

6.3.3 Object 607Ch: Home offset

The *home offset* will be set as the new drive position (reported in object 6064h) after a homing procedure is finished. An exception applies only to the homing motions -3 and -4. See their description for more details.

If Object 210Bh: Auxiliary Settings Register2 bit 12 is set to 1, then after the homing ends, the actual position (6064h) will not be reset to the value of 607Ch. This option is useful when using an absolute encoder, and only the absolute position of the home sensor is needed. The homing will end the positioning right on the home sensor.

Object description:

Index	607Ch
Name	Home offset
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	Possible
Units	PU
Value range	INTEGER32
Default value	0

The default value for this object can be changed by editing the parameter “HOME_OFFSET_607C” found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

6.3.4 Object 6098h: Homing method

The *homing method* determines the method that will be used during homing.

Object description:

Index	6098_h
Name	Homing method
Object code	VAR
Data type	INTEGER8

Entry description:

Access	RW
PDO mapping	Possible
Value range	INTEGER8
Default value	0

Data description:

Value	Description
-128 … -5	Reserved
-4..-1	Methods -4 to -1
0	No homing operation will be executed
1 … 14	Methods 1 to 14
15,16	reserved
17..30	Methods 17 to 30
31,32	reserved
33..35	Methods 33 to 35
36 … 127	reserved

There are 35 built-in homing methods, conforming to DSP402 device profile. Using the EasyMotion Studio software, one can customize each of these homing methods.

The default value for this object can be changed by editing the parameter “HOME_NR_6098” found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

6.3.5 Object 6099_h: Homing speeds

This object defines the speeds used during homing. It is given in velocity units. There are 2 homing speeds; in a typical cycle the faster speed is used to find the home switch and the slower speed is used to find the index pulse.

Object description:

Index	6099_h
Name	Homing speeds
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	2
Default value	2

Sub-index	1
Description	Speed during search for switch
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00010000 (1.0 IU)

Sub-index	2
Description	Speed during search for zero
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00010000 (1.0 IU)

The default value for sub-index 1 can be changed by editing the parameter “HOME_HSPD_6099_01” found in parameters.cfg of the project file.

The default value for sub-index 2 can be changed by editing the parameter “HOME_LSPD_6099_02” found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

6.3.6 Object $609A_h$: Homing acceleration

The *homing acceleration* establishes the acceleration to be used for all the accelerations and decelerations with the standard homing modes and is given in acceleration units.

Object description:

Index	$609A_h$
Name	Homing acceleration
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Possible
Units	AU
Value range	UNSIGNED32
Default value	0x0000199A (0.1 IU)

The default value for this object can be changed by editing the parameter "HOME_ACC_609A" found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

6.3.7 Object $207B_h$: Homing current threshold

The Homing Current Threshold Level object together with object Homing current threshold time ($207C_h$) defines the protection limits when reaching a mechanical stop during homing methods -1,-2,-3 and -4. The object defines the value of current in the drive, over which the homing procedure determines that the mechanical limit has been reached when it lasts more than the time interval specified in object $207C_h$. The current is set in internal units.

Warning!	The value of *Homing Current Threshold* must be lower than the drive current limit. Otherwise, the homing will not complete successfully (no homing error will be issued). The current limit is set during setup. See Paragraph *1.2. Setting the current limit*. *Current Threshold < current limit*

Object description:

Index	$207B_h$
Name	Homing current threshold
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	Possible
Units	IU
Value range	-32768 … 32767
Default value	0

The default value for this object can be changed by editing the parameter "HOME_CRT_207B" found in parameters.cfg of the project file. Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

6.3.8 Object $207C_h$: Homing current threshold time

The Homing current threshold time object together with object Homing current threshold ($207B_h$) defines the protection limits when reaching a mechanical stop during homing methods -1,-2,-3 and -4. The object sets the time interval after the homing current threshold is exceeded. After this time is completed without the current dropping below the threshold, the next step in the homing shall be executed. It is set in time internal units.

In case a Stepper Open Loop with Step loss detection is used, this object will set the control error time detection when methods -1 to -4 are used.

Object description:

Index	$207C_h$
Name	Homing current threshold time
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Possible
Units	TU
Value range	0 … 65535
Default value	0

The default value for this object can be changed by editing the parameter “HOME_TIME_207C” found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

6.4 Homing example

Execute homing method number 18.

1. **Start remote node**.
   Enter **Pre-Operational** state.
   Enter **Safe-Operational** state.
   Enter **Operational** state.
2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.
   Set in **Control Word** mapped in RPDO1 the value 06_h.
3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.
   Set in **Control Word** mapped in RPDO1 the value 07_h.
4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.
   Set in **Control Word** mapped in RPDO1 the value $0F_h$.
5. **Homing speed during search for zero.** Set the speed during search for zero to 150 rpm. By using a 500 lines incremental encoder and 1ms sample rate for position/speed control the corresponding value of object 6099_h sub-index 2 expressed in encoder counts per sample is 50000_h.
   Send the following message: SDO access to object 6099_h sub-index 2, 32-bit value 00050000_h.
6. **Homing speed during search for switch.** Set the speed during search for switch to 600 rpm. By using a 500 lines incremental encoder and 1ms sample rate for position/speed control the corresponding value of object 6099_h sub-index 1 expressed in encoder counts per sample is 140000_h.
   Send the following message: SDO access to object 6099_h sub-index 1, 32-bit value 00140000_h.
7. **Homing acceleration.** The homing acceleration establishes the acceleration to be used with the standard homing moves. Set this value at 5 rot/s^2. By using a 500 lines incremental encoder and 1ms sample rate for position/speed control the corresponding value of object $609A_h$ expressed in encoder counts per square sample is $28F_h$.
   Send the following message: SDO access to object $609A_h$, 32-bit value $0000028F_h$.
8. **Home offset.** Set the home offset to 1 rotation. By using a 500 lines incremental encoder the corresponding value of object $607C_h$ expressed in encoder counts is $7D0_h$.
   Send the following message: SDO access to object $607C_h$, 32-bit value $000007D0_h$.
9. **Homing method.** Select homing method number 18.
   Send the following message: SDO access to object 6098_h, 8-bit value 12_h.
10. **Modes of operation.** Select homing mode.
    Set in **Modes of Operation** mapped in RPDO1 the value 06_h.
11. **Start the homing.**
    Set in **Control Word** mapped in RPDO1 the value $001F_h$.
12. **Press for 5s the LSP button.**
13. **Wait for homing to end.**
    When Status Word (object 6040_h) bit13=0, bit12=1 and bit10=1, means homing procedure is completed successfully.
14. **Check the value of motor actual position.**
    Read by SDO protocol the value of object 6064_h.

The node will return the value of motor actual position that should be the same with the value of home offset (plus or minus few encoder counts depending on your position tuning).

7 Position Profile Mode

7.1 Overview

In Position Profile Mode, the drive controls the position.

The Position Profile Mode supports 2 motion modes:

- **Trapezoidal profile**. The built-in reference generator computes the position profile with a trapezoidal shape of the speed, due to a limited acceleration. The EtherCAT® master specifies the absolute or relative **Target Position** (index 607Ah), the **Profile Velocity** (index 6081h) and the **Profile Acceleration** (6083h)

  In relative mode, the position to reach can be computed in 2 ways: standard (default) or additive. In standard relative mode, the position to reach is computed by adding the position increment to the instantaneous position in the moment when the command is executed. In the additive relative mode, the position to reach is computed by adding the position increment to the previous position to reach, independently of the moment when the command was issued. Bit 11 of *Controlword* activates the additive relative mode.

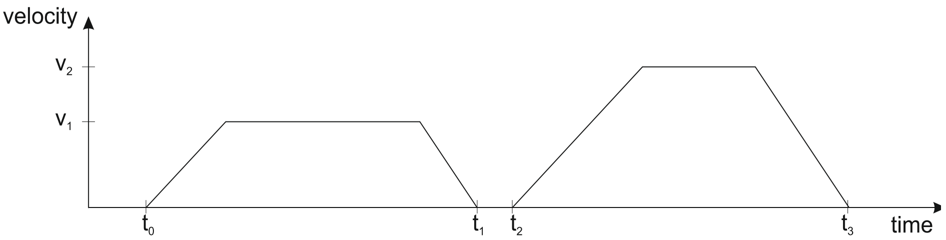
- **S-curve profile** the built-in reference generator computes a position profile with an S-curve shape of the speed. This shape is due to the jerk limitation, leading to a trapezoidal or triangular profile for the acceleration and an S-curve profile for the speed. The EtherCAT® master specifies the absolute or relative **Target Position** (index 607Ah), the **Profile Velocity** (index 6081h), the **Profile Acceleration** (6083h) and the jerk rate. The jerk rate is set indirectly via the **Jerk time** (index 2023h), which represents the time needed to reach the maximum acceleration starting from zero.

There are two different ways to apply *target positions* to a drive, controlled by the *change set immediately* bit in Controlword:

7.1.1 Discrete motion profile (*change set immediately* = 0)

After reaching the *target position* the drive unit signals this status to a EtherCAT® master and then receives a new set-point. After reaching a *target position* the velocity normally is reduced to zero before starting a move to the next set-point.

After the *target position* is sent to the drive, the EtherCAT® master has to set the *new set-point* bit in *Controlword*. The drive responds with bit *set-point acknowledge* set in *Statusword*. After that, the master has to reset bit *new set-point* to 0. Following this action, the drive will signalize that it can accept a new set-point by resetting *set-point acknowledge* bit in *Statusword* after the reference generator has reached the designated demand position.



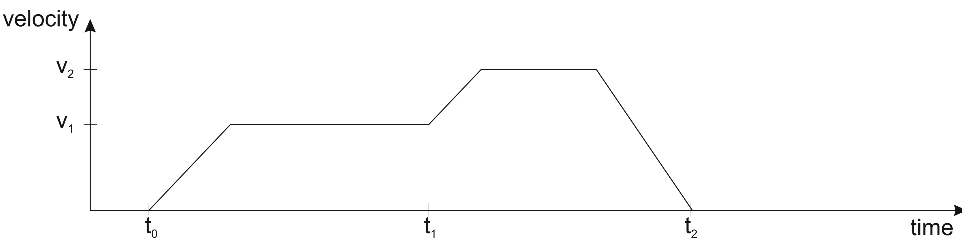
7.1.2 Continuous motion profile (*change set immediately = 1*)

The drive unit immediately processes the next *target position*, even if the actual movement is not completed. The drive readapts the actual move to the new target position.

In this case, the handshake presented for *change set immediately* = 0 is not necessary. By setting the *new set-point* bit, the master will trigger the immediate update of the target position. In this case, if the *target position* is set as relative, also bit 11 is taken into consideration (with or without additive movement).

Remark:

In case object 6086h (Motion Profile Type) is set to 3 (jerk-limited ramp = S-curve profile), then *change set immediately* bit must be 0, else a command error is issued.



7.1.3 Controlword in profile position mode

MSB **LSB**

See 6040h	Operation Mode	See 6040h	Halt	See 6040h	Abs/rel	Change set immediately	New set-point	See 6040h
15 12	11	10 9	8	7	6	5	4	3 0

***Table 7.1.1** – Controlword bits description for Position Profile Mode*

Name	Value	Description
Operation Mode	0	Trapezoidal profile - In case the movement is relative, do not add the new target position to the old demand position S-curve profile – Stop the motion with S-curve profile (jerk limited ramp)
	1	Trapezoidal profile - In case the movement is relative, add the new target position to the old demand position to obtain the new target position S-curve profile – Stop the motion with trapezoidal profile (linear ramp)
New set-point	0 -> 1	Only a 0 to 1 transition will start a new motion
Change set immediately	0	Finish the actual positioning and then start the next positioning
	1	Interrupt the actual positioning and start the next positioning. Valid only for linear ramp profile.
Abs / rel	0	*Target position* is an absolute value
	1	*Target position* is a relative value
Halt	0	Execute positioning
	1	Stop drive with *profile acceleration*

7.1.4 Statusword in profile position mode

MSB **LSB**

See 6041h	Following error	Set-point acknowledge	See 6041h	Target reached	See 6041h
15 14	13	12	11	10	9 0

***Table 7.1.2** – Statusword bits description for Position Profile Mode*

Name	Value	Description
Target reached	0	Halt = 0: *Target position* not reached Halt = 1: Drive decelerates
	1	Halt = 0: *Target position* reached Halt = 1: Velocity of drive is 0
Set-point acknowledge	0	Trajectory generator will accept a new set-point
	1	Trajectory generator will not accept a new set-point.
Following error	0	No following error
	1	Following error

7.2 Position Profile Mode Objects

7.2.1 Object 607Ah: Target position

The *target position* is the position that the drive should move to in position profile mode using the current settings of motion control parameters such as velocity, acceleration, and *motion profile type* etc. It is given in position units.

The position units are user defined. The value can be converted into position increments using the *position factor* (see Chapter Factor group).

If Controlword bit 6 = 0 (e.g. absolute positioning), represents the position to reach.

If Controlword bit 6 = 1 (e.g. relative positioning), represents the position displacement to do. When Controlword bit 14 = 0, the new position to reach is computed as: motor actual position (6064h) + displacement. When Controlword bit 14 = 1, the new position to reach is computed as: actual demand position (6062h) + displacement.

Object description:

Index	607Ah
Name	Target position
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	Yes
Value range	$-2^{31} \ldots 2^{31}-1$
Default value	No

7.2.2 Object 6081h: Profile velocity

In a position profile, it represents the maximum speed to reach at the end of the acceleration ramp. The *profile velocity* is given in speed units.

The speed units are user defined. The value can be converted into internal units using the *velocity encoder factor* (see Chapter Factor group).

By default, the velocity value is given in internal units. They are encoder increments/Sample loop. The default Sample loop is 1ms. The velocity variable is 32 bits long and it receives 16.16 data. The MSB takes the integer part and the LSB takes the factionary part.

Example: for a target speed of 50.00 IU, 0x00320000 must be set in 6081h if no factor group is chosen.

Object description:

Index	6081h
Name	Profile velocity
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	-

7.2.3 Object 6083h: Profile acceleration

In position or speed profiles, represents the acceleration and deceleration rates used to change the speed between 2 levels. The same rate is used when *Quick Stop* or *Disable Operation* commands are received. The *profile acceleration* is given in acceleration units.

The acceleration units are user defined. The value can be converted into internal units using the *acceleration factor* (see Chapter Factor group).

If no factor is applied, the same description as object 6081h applies. So 65536 IU = 1 encoder increment / $sample^2$.

Object description:

Index	6083h
Name	Profile acceleration
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Possible
Value range	$0..(2^{32}-1)$
Default value	-

7.2.4 Object 6085h: Quick stop deceleration

The *quick stop deceleration* is the deceleration used to stop the motor if the *Quick Stop* command is received and the *quick stop option code* object (index 605Ah) is set to 2 or 6.

It is also used when:

- the *fault reaction option code* object (index 605Eh) and the *halt option code* object (index 605Dh) is 2.

- a limit switch is active. See also *4.3 Limit Switch functionality explained*.

The quick stop deceleration is given in user-defined acceleration units.

Object description:

Index	6085h
Name	Quick stop deceleration
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Possible
Value range	$0..(2^{32}-1)$
Default value	-

7.2.5 Object 2023h: Jerk time

In this object, you can set the time to use for S-curve profile (jerk-limited ramp set in Object 6086h – Motion Profile Type). The time units are given in ms.

Object description:

Index	2023h
Name	Jerk time
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Possible
Value range	0 … 65535
Default value	-

7.2.6 Object 6086h: Motion profile type

Object description:

Index	6086h
Name	Motion profile type
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	Possible
Value range	INTEGER16
Default value	0

Data description:

Profile code	Profile type
-32768 … -1	Manufacturer specific (reserved)
0	Linear ramp (trapezoidal profile)
1,2	Reserved
3	Jerk-limited ramp (S-curve)
4 … 32767	Reserved

7.2.7 Object 6062h: Position demand value

This object represents the output of the trajectory generation. The *position demand value* is given in user-defined position units.

Object description:

Index	6062h
Name	Position demand value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Value range	-2^{31} … 2^{31}-1
Default value	-

7.2.8 Object 6063h: Position actual internal value

This object represents the actual value of the position measurement device in increments.

Object description:

Index	6063h
Name	Position actual value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Units	increments
Value range	-2^{31} … 2^{31}-1
Default value	-

7.2.9 Object 6064h: Position actual value

This object represents the actual value of the position measurement device. The *position actual value* is given in user-defined position units.

Remarks:

1. When using a stepper open loop motor with no encoder this object reports the value of object 6062h Position demand value. In this case, object 6063h will report the value 0, as there is no feedback device.
2. When using a stepper open loop motor with no encoder with encoder on motor configuration (for step loss detection), based on the internal register ASR bit 11, this object reports:
   - ASR.11=0 (default) - the value of object 6062h Position demand value. In this case, object 6063h will show the actual encoder value in increments.
   - [1]ASR.11=1 – the value of the feedback device scaled into microsteps which are the same value that is used for position commands in 607Ah

Object description:

Index	6064h
Name	Position actual value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Yes
Value range	$-2^{31} \ldots 2^{31}-1$
Default value	-

7.2.10 Object 6065h: Following error window

This object defines a range of tolerated position values symmetrically to the *position demand value*, expressed in position units. If the *position actual value* is above the *following error window* for a period larger than the one defined in *following error time out*, a following error occurs. If the value of the *following error window* is 2^{32}-1, the following control is switched off.

The maximum value allowed for the *following error window* parameter, expressed in increments, is:

- 2^{32}-1 for F515x or newer firmware
- 32767 for F510x/511x firmware

Object description:

Index	6065h
Name	Following error window
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	-

This object is automatically set in Drive Setup by modifying the Position control error.

The value for this object can be changed by editing the parameter:

- “ERRMAXL” for F515x
- “ERRMAX” for F510x/511x

found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

7.2.11 Object 6066h: Following error time out

See 6065h, *following error window*. The value is given in control loop time which is by default 1ms.

Remark: By default, without the Factor Group set, the time units are equal to the drive Slow/Control Loop time value.

Object description:

Index	6066h
Name	Following error time out
Object code	VAR
Data type	UNSIGNED16

[1] ASR.11=1 implementation is available only of F515x firmwares.

Entry description:

Access	RW
PDO mapping	Possible
Units	TU
Value range	0 … 65535
Default value	-

The value for this object can be changed by editing the parameter “TERRMAX” found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

7.2.12 Object 6067h: Position window

The *position window* defines a symmetrical range of accepted positions relative to the *target position*. If the *position actual value* is within the *position window* for a time period defined inside the *position window time* object, this *target position* is regarded as reached. The *position window* is given in position units. If the value of the *position window* is 2^{32}-1, the position window control is switched off and the target position will be regarded as reached when the position reference is reached.

The maximum value allowed for the *position window* parameter, expressed in increments, is 32767.

Object description:

Index	6067h
Name	Position window
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	-

This object is automatically set in Drive Setup by modifying the Band in Motion complete settings in Drive setup.

The value for this object can be changed by editing the parameter “POSOKLIM” found in parameters.cfg of the project file.

Activating Object 2076h: Save current configuration, will set its current values as the a new default.

7.2.13 Object 6068h: Position window time

See description of Object 6067h: Position window.

Remark: By default, without the Factor Group set, the time units are equal to the drive Slow/Control Loop time value.

Object description:

Index	6068h
Name	Position window time
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Possible
Units	TU
Value range	0 … 65535
Default value	-

This object is automatically set in Drive Setup by modifying the Time in Motion complete settings in Drive setup.

The value for this object can be changed by editing the parameter “TONPOSOK” found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

7.2.14 Object 607B$_h$: Position range limit

This object indicates the configured maximal and minimal position range limits. It limits the numerical range of the input value. On reaching or exceeding these limits, the input value shall wrap automatically to the other end of the range. Wrap-around of the input value may be prevented by setting software position limits as defined in software position limit object (607D$_h$). To disable the position range limits, the min position range limit (sub-index 01h) and max position range limit (sub-index 02h) must be set to 0. The values are given in user-defined position units.

Object description:

Index	607B$_h$
Name	Position range limit
Object code	ARRAY
Data type	INTEGER32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Default value	2

Sub-index	1
Description	Min position range limit
Access	RW
PDO mapping	Possible
Value range	INTEGER32
Default value	No

Sub-index	2
Description	Max position range limit
Access	RW
PDO mapping	Possible
Value range	INTEGER32
Default value	No

This object and its values can be defined directly in Drive Setup under the “Position range limits” area.
Also, activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

7.2.15 Object 60F2$_h$: Positioning option code

This object configures the positioning behavior as for the profile positioning mode or the interpolated positioning mode.

Object description:

Index	60F2$_h$
Name	Positioning option code
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Possible
Value range	UNSIGNED16
Default value	0000h

MSB **LSB**

Reserved		rado		Reserved	
15	8	7	6	5	0

***Table 7.2.1** – Positioning option code bits description*

Name	bit 7	bit 6	Description
rado	0	0	Normal positioning similar to linear axis; If reaching or exceeding the Position range limits (607Bh) the input value shall wrap automatically to the other end of the range. Positioning can be relative or absolute. Only with this bit combination, the movement greater than a modulo value is possible.
	0	1	Positioning only in negative direction; if target position is higher than actual position, axis moves over the min position limit (607D$_h$, sub-index 01h) to the target position.

	1	0	Positioning only in positive direction; if target position is lower than actual position, axis moves over the max position limit (607Bh, sub-index 02h) to the target position.
	1	1	Positioning with the shortest way to the target position. NOTE: If the difference between actual value and target position in a 360° system is 180°, the axis moves in positive direction.

The figure below shows movement examples depending on settings of the bits 6 and 7. Here the min position range limit (607Bh, sub-index 01h) is 0° and the max position range limit (607Bh, sub-index 02h) is 360°.

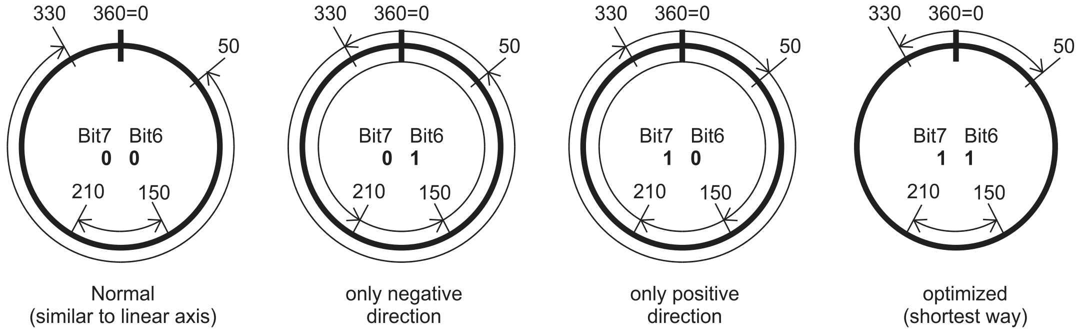


Figure 7.2.1. *Rotary axis positioning example*

A movement greater than a modulo value with more than 360° (bit 6 and 7 in this object are set to 0) on a rotary axis can be done with relative and absolute values depending on the bit 6 in the controlword. There are positive and negative values possible.

The figure below shows an example for absolute positioning in a 360° system. The actual position is 90° and absolute target position is 630°. The axis will move in positive direction one time via the max position limit to 270°. To move in negative direction, the negative sign for target position shall be used.

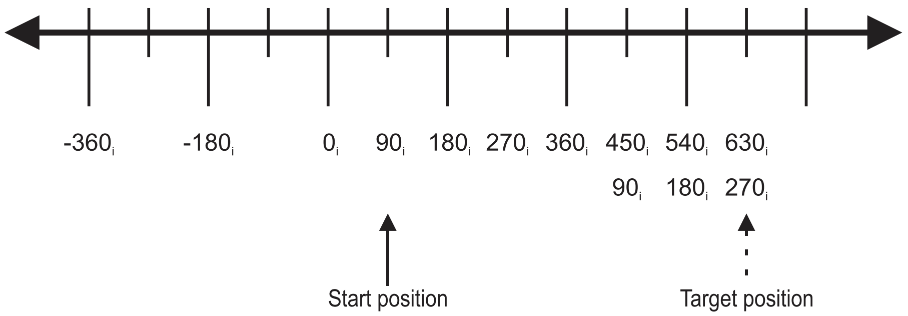


Figure 7.2.2. *Example for absolute movement greater than modulo value*

The figure below shows an example for relative positioning in a 360° system. The actual position is 300° and relative target position is 500°. The axis will move in positive direction two times via the max position limit to 80°. To move in negative direction, the negative sign for target position is used. The difference between min and max position range limits (see object 607Bh) are representable in multiples of encoder increments.

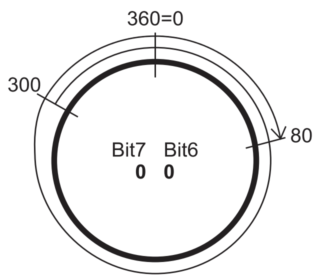


Figure 7.2.3. *Example for relative movement greater than modulo value*

The default value for this object can be changed by editing the parameter “POSOPTCODE” found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

7.2.16 Object 60F4$_h$: Following error actual value

This object represents the actual value of the following error, given in user-defined position units.

Object description:

Index	60F4$_h$
Name	Following error actual value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Value range	INTEGER32
Default value	-

7.2.17 Object 60FC$_h$: Position demand internal value

This output of the trajectory generator in profile position mode is an internal value using position increments as units. It can be used as an alternative to *position demand value* (6062$_h$).

Object description:

Index	60FC$_h$
Name	Position demand internal value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Units	Increments
Value range	$-2^{31} \ldots 2^{31}-1$
Default value	-

7.2.18 Object 2022$_h$: Control effort

This object can be used to visualize the control effort of the drive (the reference for the current controller). It is available in internal units.

Object description:

Index	2022$_h$
Name	Control effort
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	Yes
Value range	INTEGER16
Default value	-

7.2.19 Object 2081$_h$: Set/Change the actual motor position

This object sets the motor position to the value written in it. It affects object 6064$_h$, 6063$_h$ and 6062$_h$.

The object is not affected by the Factor Group and it receives its value in Internal Units.

Object description:

Index	2081$_h$
Name	Set actual position
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	No
Value range	$-2^{31}\ldots2^{31}-1$
Default value	-

7.2.20 Object 2088h[1]: Actual internal position from sensor on motor

This object shows the position value read from the encoder on the motor in increments, in case a dual loop control method is used.

The factor group objects have no effect on it.

Object description:

Index	2088h
Name	Actual internal position from sensor on motor
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Units	increments
Value range	$-2^{31} \dots 2^{31}-1$
Default value	-

7.2.21 Object 208Dh[2]: Auxiliary encoder position

This object represents the actual value of the auxiliary position measurement device, expressed in internal units, when operating in the digital external reference mode (signal type set to Encoder). The factor group objects have no effect on it.

Object description:

Index	208Dh
Name	Auxiliary encoder value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Units	increments
Value range	$-2^{31} \dots 2^{31}-1$
Default value	-

7.3 Position Profile Examples

7.3.1 Relative trapezoidal example

Execute an absolute trapezoidal profile with limited speed. First perform 4 rotations, wait motion complete and then set the target position of 16 rotations.

1. **Start remote node**.
   Enter **Pre-Operational** state.
   Enter **Safe-Operational** state.
   Enter **Operational** state.
2. **Modes of operation.** Select position mode.
   Set in **Modes of Operation** mapped in RPDO1 the value 01h.
3. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.
   Set in **Control Word** mapped in RPDO1 the value 06h.
4. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.
   Set in **Control Word** mapped in RPDO1 the value 07h.
5. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.
   Set in **Control Word** mapped in RPDO1 the value 0Fh.
6. **Target position.** Set the target position to 20 rotations. By using a 500 lines incremental encoder the corresponding value of object 607Ah expressed in encoder counts is 9C40h.
   Set in **Target position** mapped in RPDO2 the value 00009C40h.
7. **Target speed.** Set the target speed normally attained at the end of acceleration ramp to 500 rpm. By using a 500 lines incremental encoder and 1ms sample rate for position/speed control the corresponding value of object 6081h expressed in encoder counts per sample is 10AAACh.
   Send the following message: SDO access to object 6081h, 32-bit value 0010AAACh.
8. **Start the profile.**

---

[1] Object 2088h applies only to drives which have a secondary feedback

[2] Object 208Dh is available only drives which have a secondary feedback

Set in **Control Word** mapped in RPDO1 the value 005F$_h$. If Controlword bit 6 is set (Controlword.6 = 1), a relative positioning will start.

9. **Wait movement to finish.**
   Wait for Bit10 to become 1 in Status Word.
10. **Reset the set point.**
    Set in **Control Word** mapped in RPDO1 the value 000F$_h$.
11. **Target position.** Set the target position to 200 rotations. By using a 500 lines incremental encoder the corresponding value of object 607A$_h$ expressed in encoder counts is 61A80$_h$.
    Send the following message: SDO access to object 607A$_h$ 32-bit value 00061A80$_h$.
12. **Start the profile.**
    Set in **Control Word** mapped in RPDO1 the value 005F$_h$.
13. **Wait movement to finish.**
    Wait for Bit10 to become 1 in Status Word.
14. **Check the value of motor actual position.**
    Read by SDO protocol the value of object 6064$_h$.
15. **Check the value of position demand value.**
    Read by SDO protocol the value of object 6062$_h$.

At the end of movement the motor position actual value should be equal with position demand value (plus or minus few encoder counts depending on your position tuning) and the motor should rotate 220 times.

7.3.2 Absolute trapezoidal example

Execute an absolute trapezoidal profile with limited speed. First perform 4 rotations, wait motion complete and then set the target position of 16 rotations.

16. **Start remote node**.
    Enter **Pre-Operational** state.
    Enter **Safe-Operational** state.
    Enter **Operational** state.
17. **Modes of operation.** Select position mode.
    Set in **Modes of Operation** mapped in RPDO1 the value 01$_h$.
18. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.
    Set in **Control Word** mapped in RPDO1 the value 06$_h$.
19. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.
    Set in **Control Word** mapped in RPDO1 the value 07$_h$.
20. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.
    Set in **Control Word** mapped in RPDO1 the value 0F$_h$.
21. **Target position.** Set the target position to 4 rotations. By using a 500 lines incremental encoder the corresponding value of object 607A$_h$ expressed in encoder counts is 1F40$_h$.
    Set in **Target position** mapped in RPDO2 the value 00001F40$_h$.
22. **Target speed.** Set the target speed normally attained at the end of acceleration ramp to 500 rpm. By using a 500 lines incremental encoder and 1ms sample rate for position/speed control the corresponding value of object 6081$_h$ expressed in encoder counts per sample is 10AAAC$_h$.
    Send the following message: SDO access to object 6081$_h$, 32-bit value 0010AAAC$_h$.
23. **Start the profile.**
    Set in **Control Word** mapped in RPDO1 the value 001F$_h$.
24. **Wait movement to finish.**
    Wait for Bit10 to become 1 in Status Word.
25. **Reset the set point.**
    Set in **Control Word** mapped in RPDO1 the value 000F$_h$.
26. **Target position.** Set the target position to 16 rotations. By using a 500 lines incremental encoder the corresponding value of object 607A$_h$ expressed in encoder counts is 7D00$_h$.
    Send the following message: SDO access to object 607A$_h$ 32-bit value 00007D00$_h$.
27. **Start the profile.**
    Set in **Control Word** mapped in RPDO1 the value 001F$_h$.
28. **Wait movement to finish.**
    Wait for Bit10 to become 1 in Status Word.
29. **Check the value of motor actual position.**
    Read by SDO protocol the value of object 6064$_h$.
30. **Check the value of position demand value.**
    Read by SDO protocol the value of object 6062$_h$.

At the end of movement the motor position actual value should be equal with position demand value (plus or minus few encoder counts depending on your position tuning) and the motor should rotate 16 times.

7.3.3 Relative Jerk-limited ramp profile example

Execute an absolute Jerk-limited ramp profile.

1. **Start remote node**.
   Enter **Pre-Operational** state.
   Enter **Safe-Operational** state.
   Enter **Operational** state.

2. **Modes of operation.** Select position mode.
   Set in **Modes of Operation** mapped in RPDO1 the value 01_h.
3. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.
   Set in **Control Word** mapped in RPDO1 the value 06_h.
4. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.
   Set in **Control Word** mapped in RPDO1 the value 07_h.
5. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.
   Set in **Control Word** mapped in RPDO1 the value $0F_h$.
6. **Motion profile type.** Select Jerk-limited ramp.
   Send the following message: SDO access to object 6086_h, 16-bit value 0003_h.
7. **Target position.** Set the target position to 10 rotations. By using a 500 lines incremental encoder the corresponding value of object $607A_h$ expressed in encoder counts is $4E20_h$.
   Set in **Target position** mapped in RPDO2 the value $00004E20_h$.
8. **Target speed.** Set the target speed to 450 rpm. By using a 500 lines incremental encoder and 1ms sample rate for position/speed control the corresponding value of object 6081_h expressed in encoder counts per sample is $000F0000_h$.
   Send the following message: SDO access to object 6081_h, 32-bit value $000F0000_h$.
9. **Jerk time.** Set the time to use for Jerk-limited ramp. For more information related to this parameter, see the EMS help
   Send the following message: SDO access to object 2023_h, 16-bit value $1F4B_h$.
10. **Start the profile.** If Controlword bit 6 is set (Controlword.6 = 1), a relative positioning will start.
    Set in **Control Word** mapped in RPDO1 the value $005F_h$.
11. **Wait movement to finish.**
    Wait for Bit10 to become 1 in Status Word.
12. **Check the value of motor actual position.**
    Read by SDO protocol the value of object 6064_h.
13. **Check the value of position demand value.**
    Read by SDO protocol the value of object 6062_h.

At the end of movement the motor position actual value should be equal with position demand value (plus or minus few encoder counts depending on your position tuning).

7.3.4 Absolute Jerk-limited ramp profile example

Execute an absolute Jerk-limited ramp profile.

14. **Start remote node**.
    Enter **Pre-Operational** state.
    Enter **Safe-Operational** state.
    Enter **Operational** state.
15. **Modes of operation.** Select position mode.
    Set in **Modes of Operation** mapped in RPDO1 the value 01_h.
16. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.
    Set in **Control Word** mapped in RPDO1 the value 06_h.
17. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.
    Set in **Control Word** mapped in RPDO1 the value 07_h.
18. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.
    Set in **Control Word** mapped in RPDO1 the value $0F_h$.
19. **Motion profile type.** Select Jerk-limited ramp.
    Send the following message: SDO access to object 6086_h, 16-bit value 0003_h.
20. **Target position.** Set the target position to 5 rotations. By using a 500 lines incremental encoder the corresponding value of object $607A_h$ expressed in encoder counts is 2710_h.
    Set in **Target position** mapped in RPDO2 the value 00002710_h.
21. **Target speed.** Set the target speed to 150 rpm. By using a 500 lines incremental encoder and 1ms sample rate for position/speed control the corresponding value of object 6081_h expressed in encoder counts per sample is 00050000_h.
    Send the following message: SDO access to object 6081_h, 32-bit value 00050000_h.
22. **Jerk time.** Set the time to use for Jerk-limited ramp. For more information related to this parameter, see the EMS help
    Send the following message: SDO access to object 2023_h, 16-bit value $13B_h$.
23. **Start the profile.**
    Set in **Control Word** mapped in RPDO1 the value $001F_h$.
24. **Wait movement to finish.**
    Wait for Bit10 to become 1 in Status Word.
25. **Check the value of motor actual position.**
    Read by SDO protocol the value of object 6064_h.
26. **Check the value of position demand value.**
    Read by SDO protocol the value of object 6062_h.

At the end of movement the motor position actual value should be equal with position demand value (plus or minus few encoder counts depending on your position tuning).

8 Torque Profile Mode

8.1 Overview

The profile torque mode allows to control the motor in torque mode by transmitting the target torque and torque slope values, which are processed via the trajectory generator.

Remark: *This mode is available starting with firmware versions F515K / FA00x.*

8.1.1 Controlword in profile torque mode

MSB **LSB**

See 6040h		Halt	See 6040h	Reserved		See 6040h	
15	9	8	7	6	4	3	0

Table 8.1 *– Controlword bits description for Torque Profile Mode*

Name	Value	Description
Halt	0	Execute torque profile
	1	Stop drive according to the halt option code (605Dh)

8.1.2 Statusword in profile torque mode

MSB **LSB**

See 6041h		Reserved		See 6041h	Target reached	See 6041h	
15	14	13	12	11	10	9	0

Table 8.2 *– Statusword bits description for Torque Profile Mode*

Name	Value	Description
Target reached	0	Halt = 0: *Target torque* not reached Halt = 1: Drive decelerates
	1	Halt = 0: *Target torque* reached Halt = 1: Velocity of drive is 0

8.2 Torque Profile Mode Objects

8.2.1 Object 6071h: Target torque

This parameter specifies the input value configured for the torque controller when operating in Torque Profile mode. The unit for this object is given in IU, except for FA00x and FA02x firmware versions, where Object 2115h: ASR4 bit 0 controls the unit in which the object is given:

- If ASR4.0 = 0, the unit for this object is given in IU
- If ASR4.0 = 1, the unit is in thousandths (‰) of the motor's rated current specified in object 6075h. Example:
- If the target torque is set to 500, it represents 50.0% (500 ‰) of the motor's rated current.
- If the target torque is set to 255, it represents 25.5% (255 ‰) of the motor's rated current.

Remarks:
1. When object 2115h is set to 1, the target torque can exceed 100% (equivalent to 1000 ‰) of the motor's rated current, as defined by object 6075h.
2. The current limit is set through Object 207Fh: Current limit. This value acts as a safety threshold and will restrict the maximum current, regardless of the value specified in object 6071h.

Object description:

Index	6071h
Name	Target torque
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	Yes
Value range	INTEGER16
Default value	0000h

The computation formula for the current [IU] in [A] is:

$$curent[IU] = \frac{65520 \cdot current[A]}{2 \cdot Ipeak}$$

where I_{peak} is the peak current supported by the drive and *current[IU]* is the command value for object 6071_h.

8.2.2 Object 6075_h: Motor rated current

The motor rated current is the motor's nominal current which needs to be expressed in mA.

Object description:

Index	6075_h
Name	Motor rated current
Object code	VAR
Data type	UNSIGNED32

Entry description:

Sub-index	00_h
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	Motor nominal current specified in the setup part.

8.2.3 Object 6087_h: Torque slope

The torque slope indicate the rate of change of current. The value needs to be given in in units of per thousand of rated current specified in object 6075_h per second.

The rate of change of current is calculated as follows:

$$\frac{Rated\ Current\ (6075_h)}{1000} \times Torque\ Slope\ (6087_h)/s$$

Example: If the Rated Current specified in object 6075_h is set to 2000mA and the Torque Slope specified in object 6087_h is set to 1000, the rate of change of current is 2A/s.

Object description:

Index	6087_h
Name	Torque slope
Object code	VAR
Data type	UNSIGNED32

Entry description:

Sub-index	00_h
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0

8.3 Torque Profile Example

Execute a torque profile.

1. **Start remote node**.

   Enter **Pre-Operational** state.

   Enter **Safe-Operational** state.

   Enter **Operational** state.

2. **Modes of operation.** Select position mode.

   Set in **Modes of Operation** mapped in RPDO1 the value 04_h.

3. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.

   Set in **Control Word** mapped in RPDO1 the value 06_h.

4. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.

   Set in **Control Word** mapped in RPDO1 the value 07_h.

5. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.

Set in **Control Word** mapped in RPDO1 the value $0F_h$.

6. **Motor rated current.** Define the motor nominal(rated) current of 1000mA using object 6075_h. Send the following message: SDO access to object 6075_h, 32-bit value 03E8h
7. **Target slope.** Define a target slope of 1000mA using object 6087_h. Send the following message: SDO access to object 6087_h, 32-bit value 03E8h
8. **Configure the formatting and representation of object 6071h**[1]**.** To define the unit as thousandths (‰) of the motor's rated current (specified in object 6075_h), set bit 0 of object 2115_h to 1. Send the following message: SDO access to object 2115_h, 32-bit value 1_h.
9. **Target torque.** Define a target torque of 50.0% of the motor rated current. Send the following message: SDO access to object 6071_h, 16-bit value $01F4_h$. The motor will move positive and reach a current of 50.0% of the motor rated current (500mA).
10. **Set a new Target torque value.** Define a target torque of -120.0% of the motor rated current. Send the following message: SDO access to object 6071_h, 16-bit value $FB50_h$. The motor will move negative and reach a current of -120.0% of the motor rated current (-1200mA).

[1] Only for FA00x and FA02x firmware versions.

9 Interpolated Position Mode

9.1 Overview

The interpolated Position Mode is used to control multiple coordinated axis or a single on with the need for time-interpolation of set-point data. The Interpolated Position Mode can use the time synchronization mechanism for a time coordination of the related drive units.
The Interpolated Position Mode allows a host controller to transmit a stream of interpolation data to a drive unit. The interpolation data is better sent in bursts because the drive supports an input buffer. The buffer size is the number of *interpolation data records* that may be sent to the drive to fill the input buffer.
The interpolation algorithm can be defined in the *interpolation sub mode select*. Linear (PT – Position Time) interpolation is the default interpolation method.

9.1.1 Internal States

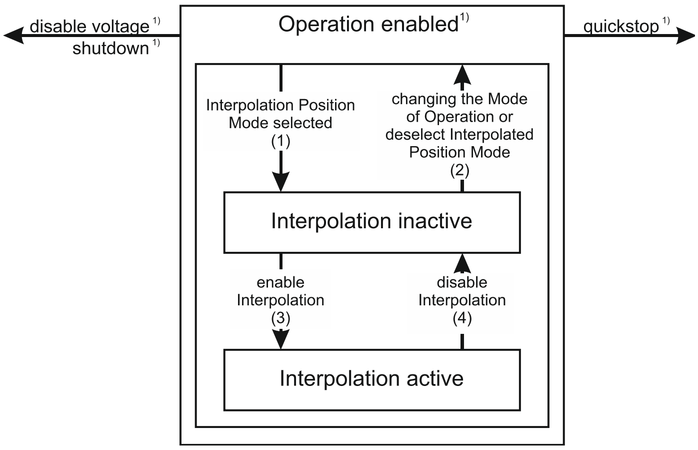


Figure 9.1.1. *Internal States for the Interpolated Position Mode*

1) See state machine Operation enabled1)

***Interpolation inactive*:** This state is entered when the device is in state Operation enabled and the Interpolated Position Mode is selected. The drive will accept input data and will buffer it for interpolation calculations, but it does not move the motor.
***Interpolation active*:** This state is entered when a device is in state Operation enabled and the Interpolation Position Mode is selected and enabled. The drive will accept input data and will move the motor.

State Transitions of the Internal States
State Transition 1: NO IP-MODE SELECTED => IP-MODE INACTIVE
Event: Select ip-mode with *modes of operations* while inside Operation enable
State Transition 2: IP-MODE INACTIVE => NO IP-MODE SELECTED
Event: Select any other mode while inside Operation enable
State Transition 3: IP-MODE INACTIVE => IP-MODE ACTIVE
Event: Set bit *enable ip mode* (bit4) of the *Controlword* while in ip-mode and Operation enable
State Transition 4: IP-MODE ACTIVE => IP-MODE INACTIVE
Event: Reset bit *enable ip mode* (bit4) of the *Controlword* while in ip-mode and Operation enable

9.1.2 Controlword in interpolated position mode

MSB LSB

See 6040h	Stop option	See 6040h	Halt	See 6040h	Abs / rel	Reserved	Enable ip mode	See 6040h
15 12	11	10 9	8	7	6	5	4	3 0

Table 9.1.1 *– Controlword bits description for Interpolated Position Mode*

Name	6040h bit	Value	Description
Enable ip mode	4	0	Interpolated position mode inactive
		1	Interpolated position mode active
Abs / rel	6	0	Set position is an absolute value
		1	Set position is a relative value (similar to Cyclic Synchronous Velocity)
Halt	8	0	Execute the instruction of bit 4
		1	Stop drive with (*profile acceleration*)
Stop option	11	0	On transition to inactive mode, stop drive immediately using *profile acceleration*

1	On transition to inactive mode, stop drive after finishing the current segment.

9.1.3 Statusword in interpolated position mode

MSB **LSB**

See 6041h		Reserved	ip mode active	See 6041h	Target reached	See 6041h	
15	14	13	12	11	10	9	0

Table 9.1.2 *– Statusword bits description for Interpolated Position Mode*

Name	Value	Description
Target reached	0	Halt = 0: Final position not reached Halt = 1: Drive decelerates
	1	Halt = 0: Final position reached Halt = 1: Velocity of drive is 0
ip mode active	0	Interpolated position mode inactive
	1	Interpolated position mode active

9.2 Interpolated Position Objects

9.2.1 Object 60C0h: Interpolation sub mode select

In the Interpolated Position Mode the drive supports three interpolation modes:

1. **Linear interpolation** as described in the CiA 402 standard (when object 208Eh bit8=1); This mode is almost identical with Cyclic Synchronous Position mode, only that it receives its position data into 60C1h sub-index 01 instead of object 607Ah. No interpolation point buffer will be used.
2. **PT (Position – Time)** linear interpolation (legacy) (when object 208Eh bit8=0)
3. **PVT (Position – Velocity – Time)** cubic interpolation (legacy) (when object 208Eh bit8=0).

The interpolation mode is selected with Interpolation sub-mode select object. The sub-mode can be changed only when the drive is in Interpolation inactive state.

Each change of the interpolation mode will trigger the reset of the buffer associated with the interpolated position mode (because the physical memory available is the same for both the sub-modes, size of each data record is different).

Object description:

Index	60C0h
Name	Interpolation sub mode select
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	Possible
Value range	$-2^{15} \ldots 2^{15}-1$
Default value	0

Data description:

Profile code	Profile type
-32768 … -2	Manufacturer specific (reserved)
-1	PVT (Position – Velocity – Time) cubic interpolation
0	Linear Interpolation or PT (Position – Time)
+1…+32767	Reserved

9.2.2 Object 60C1h: Interpolation data record

The **Interpolation Data Record** contains the data words that are necessary to perform the interpolation algorithm. The number of data words in the record is defined by the *interpolation data configuration.*

Object description:

Index	60C1h
Name	Interpolation data record
Object code	ARRAY
Number of elements	2
Data Type	Interpolated Mode dependent

Entry description

Sub-index	01h
Description	X1: the first parameter of ip function
Access	RW

PDO mapping	Possible
Value range	Interpolated Mode dependent
Default value	-
Sub-index	02_h
Description	X2: the second parameter of ip function
Access	RW
PDO mapping	Possible
Value range	Interpolated Mode dependent
Default value	-

Description of the sub-indexes:

X1 and X2 form a 64-bit data structure as defined below:

9.2.2.1 a) For linear interpolation (standard DS402 implementation)

To work with this mode, object $208E_h$ bit8 must be 1. The default value of this bit is 1 with the current iPOS templates.

There are 2 parameters in this mode:

Position – a 32-bit long integer value representing the target position (relative or absolute). Unit - position increments.

– the **Linear interpolation** position command is received in object $60C1_h$ sub-index1; sub-index2 is not used

Time – the time is defined in object $60C2_h$.

The position points should be sent in a synchronous RxPDO at fixed time intervals defined in object $60C2_h$.

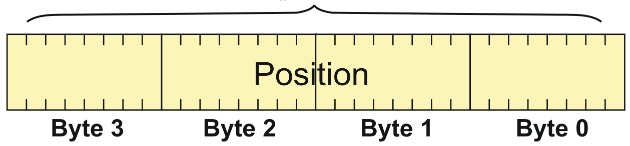


Figure 9.2.1. *Linear interpolation point 32-bit data structure*

9.2.2.2 b) For PT (Position –Time) linear interpolation (legacy).

To work with this mode, object $208E_h$ bit8 must be 0. The default value of this bit is 1 with the current iPOS templates.

There are 3 parameters in this mode:

Position – a 32-bit long integer value representing the target position (relative or absolute). Unit - position increments.

Time – a 16-bit unsigned integer value representing the time of a PT segment. Unit - position / speed loop samplings.

Counter – a 7-bit unsigned integer value representing an integrity counter. It can be used in order to have a feedback of the last point sent to the drive and detect errors in transmission.

In the example below Position[7…0] represents bits 0..7 of the position value.

Byte 0	Position [7...0]	
Byte 1	Position [15...8]	
Byte 2	Position [23...16]	
Byte 3	Position [31...24]	
Byte 4	Time [7...0][1]	
Byte 5	Time [15...8][1]	
Byte 6	Reserved	
Byte 7	Counter[6…0]	Reserved

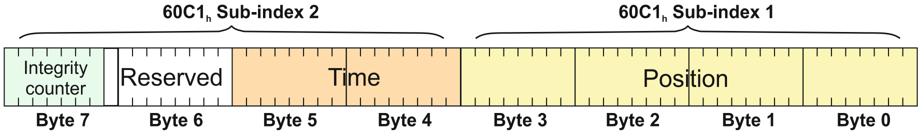


Figure 9.2.2. *PT interpolation point 64-bit data structure*

Remarks:

- The integrity counter is written in byte 3 of $60C1_h$ Sub-index 2, on the most significant 7 bits (bit 1 to bit 7).

[1] If object $207A_h$ Interpolated position 1st order time is used, these bits will we overwritten with the value defined in it

- The integrity counter is 7 bits long, so it can have a value up to 127. When the integrity counter reaches 127, the next value is 0

9.2.2.3 c) For PVT (Position – Velocity – Time) cubic interpolation

To work with this mode, object 208E$_h$ bit8 must be 0. The default value of this bit is 1 with the current iPOS templates.
There are 4 parameters in this mode:
Position – a 24-bit long integer value representing the target position (relative or absolute). Unit - position increments.
Velocity – a 24-bit fixed value representing the end point velocity (16 MSB integer part and 8 LSB fractional part). Unit - increments / sampling
Time – a 9-bit unsigned integer value representing the time of a PVT segment. Unit - position / speed loop samplings.
Counter – a 7-bit unsigned integer value representing an integrity counter. It can be used in order to have a feedback of the last point sent to the drive and detect errors in transmission.
In the example below Position 0 [7…0] represents bits 0..7 of the position value.

Byte 0	Position 0 [7...0]			
Byte 1	Position 1 [15...8]			
Byte 2	Velocity 0 [15...8]			
Byte 3	Position 2 [23...16]			
Byte 4	Velocity 1 [23...16]			
Byte 5	Velocity 2 [31...24]			
Byte 6	Time [7...0]			
Byte 7	Counter[6…0]			Time[8]
	bit7	- - - -	bit1	bit0

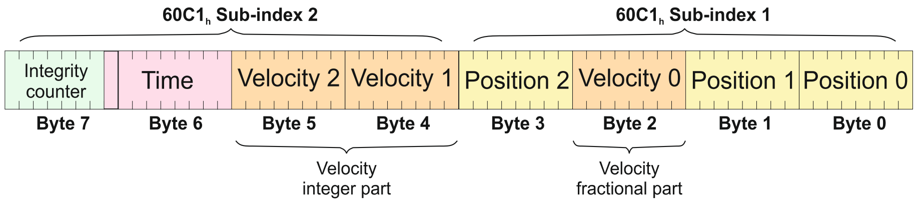


Figure 9.2.3. *PVT interpolation point 64-bit data structure*

Remarks:
- The integrity counter is written in byte 3 of 60C1$_h$ Sub-index 2, on the most significant 7 bits (bit 1 to bit 7).
- The integrity counter is 7 bits long, so it can have a value up to 127. When the integrity counter reaches 127, the next value is 0.

9.2.3 Object 2072$_h$: Interpolated position mode status

The object provides additional status information for the interpolated position mode.

Object description:

Index	2072$_h$
Name	Interpolated position mode status
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Possible
Value range	UNSIGNED16
Default value	-

Table 9.2.1 *– Interpolated position mode status bit description*

Bit	Value	Description
15	0	Buffer is not empty
	1	Buffer is empty – there is no point in the buffer.
14	0	Buffer is not low
	1	Buffer is low – the number of points from the buffer is equal or less than the low limit set using object 2074$_h$.
13	0	Buffer is not full
	1	Buffer is full – the number of points in the buffer is equal with the buffer dimension.
12	0	No integrity counter error
	1	Integrity counter error. If integrity counter error checking is enabled and the integrity counter sent by the master does not match the integrity counter of the drive.
11	0	Valid only for PVT (cubic interpolation): Drive has maintained interpolated position mode after a buffer empty condition (the velocity of the last point was 0).

	1	Valid only for PVT (cubic interpolation): Drive has performed a quick stop after a buffer empty condition because the velocity of the last point was different from 0
10 … 7		Reserved
6 … 0		Current integrity counter value

Remark: *when a status bit changes from this object, an emergency message with the code 0xFF01 will be generated. This emergency message will have mapped object 2072h data onto bytes 3 and 4.*

The Emergency message contains of 8 data bytes having the following contents:

0-1	2	3-4	5-7
Emergency Error Code (0xFF01)	Error Register (Object 1001h)	Interpolated position status (Object 2072h)	Manufacturer specific error field

To disable the sending of PVT emergency message with ID 0xFF01, the setup variable PVTSENDOFF must be set to 1.

9.2.4 Object 2073h: Interpolated position buffer length

Through **Interpolated position buffer length** object you can change the default buffer length. When writing in this object, the buffer will automatically reset its contents and then re-initialize with the new length. The length of the buffer is the maximum number of interpolation data that can be queued, and does not mean the number of data locations physically available.

Remark: *It is NOT allowed to write a "0" into this object.*

Object description:

Index	2073h
Name	Interpolated position buffer length
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	WO
PDO mapping	No
Value range	UNSIGNED16
Default value	7

9.2.5 Object 2074h: Interpolated position buffer configuration

Through this object you can control more in detail the behavior of the buffer.

Object description:

Index	2074h
Name	Interpolated position buffer configuration
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	WO
PDO mapping	No
Value range	UNSIGNED16
Default value	-

Table 9.2.2 *– Interpolated position buffer configuration*

Bit	Value	Description
15	0	Nothing
	1	Clear buffer and reinitialize buffer internal variables
14	0	Enable the integrity counter error checking
	1	Disable the integrity counter error checking
13	0	No change in the integral integrity counter
	1	Change internal integrity counter with the value specified in bits 0 to 6
12	0	If absolute positioning is set (bit 6 of *Controlword* is 0), the initial position is read from object 2079h. It is used to compute the distance to move up to the first PVT point.
	1	If absolute positioning is set (bit 6 of *Controlword* is 0), the initial position is the current *position demand value*. It is used to compute the distance to move up to the first PVT point.
11 ... 8		New parameter for buffer low signaling. When the number of entries in the buffer is equal or less than buffer low value, bit 14 of object 2072h will set.

7	0	No change in the buffer low parameter
	1	Change the buffer low parameter with the value specified in bits 8 to 11
6 … 0		New integrity counter value

9.2.6 Object 2079h: Interpolated position initial position

Through this object, you can set an initial position for absolute positioning in order to be used to compute the distance to move up to the first point. It is given in position units.

Object description:

Index	2079h
Name	Interpolated position initial position
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	Possible
Value range	INTEGER32
Default value	0

9.2.7 Object 207Ah: Interpolated position 1st order time

Through this object, you can set the time in a PT (Position – Time) Linear Interpolation mode. By setting a value in this object, there is no need to send the time together with the position and integrity counter in **Object** 60C1h: Interpolation data record. This object is disabled when it is set with 0. It is given in IU which is by default 1ms.

Remark: By default, without the Factor Group set, the time units are equal to the drive Slow/Control Loop time value.

Object description:

Index	207Ah
Name	Interpolated position 1st order time
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Yes
Value range	UNSIGNED16
Default value	0

9.2.8 Loading the interpolated points

The points can be loaded only in Legacy interpolation mode (object 208Eh bit8 must be 0 and its default is 1).

If the integrity counter is enabled, the drive considers and loads a valid IP point when it receives a new valid integrity counter number. If the drive receives interpolation data with the same integrity number, it will ignore the point and send an emergency message with the code 0xFF01. If it receives a lower or a +2 higher integrity number, it will ignore the data and send an emergency message with code 0xFF01 and

Object 207Ah: Interpolated position 1st order time mapped on bytes 4 and 5 showing and integrity counter error. This error will be automatically reset when the data with correct integrity number will be received. The 7 bit integrity counter can have values between 0 and 127. Therefore, when the counter reaches the value 127, the next logical value is 0.

After receiving each point, the drive calculates the trajectory it has to execute. Because of this, the points must be loaded after the absolute/relative bit is set in Controlword.

A correct interpolated PT/PVT motion would be like this:

- Enter mode 07 in Modes of Operation
- set the IP (Interpolated Position) buffer size
- Clear the buffer and reinitialize the integrity counter
- Set in Controlword the bit for absolute or relative motion
- If the motion is absolute, set in 2079h the actual position of the drive (read from object 6063h)
- If the motion is PT, set in object 207Ah a fixed time interval if not supplied in 60C1h sub-index2
- Load the first IP points
- Start the motion by toggling from 0 to 1 bit4 in Controlword
- Monitor the interpolated status for buffer low warning (an emergency message will be sent automatically containing the interpolated status when one of the status bits changes)
- Load more points until buffer full bit is active
- Return to monitoring the buffer status and load points until the profile is finished

9.3 Linear interpolation example

To work with this mode, object 208Eh bit8 must be 1. The default value of this bit is 1, so there is no need to change it. This example is identical with the *Cyclic Synchronous Position Mode basic* example with the following changes:

- the modes of operation 6060h must be set = 7 instead of 8
- object 60C1h sub-index 1 must be used instead of object 607Ah.

All the other commands and behavior is the same.

9.4 PT absolute movement example

Execute an absolute PT movement.

Remarks: *Because this is a demo for a single axis, the synchronization mechanism is not used here.*

To work with this mode, object 208Eh bit8 must be 0. The default value of this bit is 1

1. **Start remote node**.

Enter **Pre-Operational** state.

2. **Disable the RPDO3**. Write zero in object 1602h sub-index 0, this will disable the PDO.

Send the following message: SDO access to object 1602h sub-index 0, 8-bit value 0.

3. **Map the new objects**:
   a. Write in object 1602h sub-index 1 the description of the interpolated data record sub-index 1:
   Send the following message: SDO access to object 1602h sub-index 1, 32-bit value 60C10120h.
   b. Write in object 1602h sub-index 2 the description of the interpolated data record sub-index 2:
   Send the following message: SDO access to object 1602h sub-index 2, 32-bit value 60C10220h.
4. **Enable the RPDO3**. Set the object 1602h sub-index 0 with the value 2.

Send the following message: SDO access to object 1602h sub-index 0, 8-bit value 2.

5. **Add the new TPDO to the Sync Manager**:
   a. Write zero in object 1C12h sub-index 0, this will disable the Sync. Manager.
   Send the following message: SDO access to object 1C12h sub-index 0, 8-bit value 00h.
   b. Write in object 1C12h sub-index 3 the RPDO3 mapping parameter object number:
   Send the following message: SDO access to object 1C12h sub-index 3, 16-bit value 1602h.
   c. Write 03 h in object 1C12h sub-index 0, this will enable the Sync. Manager.
   Send the following message: SDO access to object 1C12h sub-index 0, 8-bit value 03h.

   Note: if using TwinCAT System Manager, enter in Configuration Mode, select the drive, select Process Data tab, uncheck the PDO Assignment and PDO Configuration boxes. Click Load PDO info from device button to load the new DO configuration. Press F4 to reload the IO devices and enter in Operation state.
6. Enter **Safe-Operational** state.
7. Enter **Operational** state.
8. **Ready to switch on.** Set in **Control Word** mapped in RPDO1 the value 06h.
9. **Switch on.** Set in **Control Word** mapped in RPDO1 the value 07h.
10. **Enable Operation.** Set in **Control Word** mapped in RPDO1 the value 0Fh. For relative motion, set 4Fh.
11. Set in **Modes of operation** mapped in RPDO1 the value 7 to enable Interpolated mode.
12. **Interpolation sub mode select**. Select PT interpolation position mode.
    Send the following message: SDO access to object 60C0h, 16-bit value 0000h.
13. **Interpolated position buffer length**.
    Send the following message: SDO access to object 2073h, 16-bit value 000Ch. The maximum is 000Fh.
14. **Interpolated position buffer configuration**. By setting the value A001h, the buffer is cleared and the integrity counter will be set to 1.
    Send the following message: SDO access to object 2074h, 16-bit value A001h.
15. **Interpolated position initial position**. Set the initial position to 0.5 rotations. By using a 500 lines incremental encoder the corresponding value of object 2079h expressed in encoder counts is (1000d) 3E8h. By using the settings done so far, if the final position command were to be 0, the drive would travel to (Actual position – 1000).
    Send the following message: SDO access to object 2079h, 32-bit value 3E8h.
16. **Loading the PT points.** Assuming X1 and X2 are object 60C1h's sub index 01 and 02, which were recently mapped, send the following data:
17. **Send the 1st PT point**.
    Position= 20000 IU (0x00004E20) 1IU = 1 encoder pulse
    Time = 1000 IU (0x03E8) 1IU = 1 control loop = 1ms by default
    IC = 1 (0x01) IC=Integrity Counter

The drive motor will do 10 rotations (20000 counts) in 1000 milliseconds.

Set X1=00004E20h; X2=020003E8 h;

18. **Send the 2nd PT point**.

    Position= 30000 IU (0x00007530)

    Time = 2000 IU (0x07D0)

    IC = 2 (0x02)

Set X1=00007530h; X2=040007D0 h;

19. **Send the 3rd PT point**.

    Position= 2000 IU (0x000007D0)

    Time = 1000 IU (0x03E8)

    IC = 3 (0x03)

Set X1=000007D0h; X2=060003E8 h;

20. **Send the last PT point**.

Set X1=00000000 h (0 counts); X2=080001F4 (IC=4 (0x08), time =500 (0x01F4))

    Position= 0 IU (0x00000000)

    Time = 500 IU (0x01F4)

    IC = 4 (0x04)

Set X1=00000000h; X2=080001F4h;

21. **Start an absolute motion**.

    Set in **Control Word** mapped in RPDO1 the value 1Fh.

After the sequences are executed, if the drive actual position before starting the motion was 0, now it should be -1000 counts because of Step 15.

9.5 PVT absolute movement example

Execute an absolute PVT movement. The PVT position points will be given as absolute positions.

Remarks: *Because this is a demo for a single axis the synchronization mechanism is not used here.*

To work with this mode, object 208Eh bit8 must be 0. The default value of this bit is 1.

1. **Start remote node**.

   Enter **Pre-Operational** state.

2. **Disable the RPDO3**. Write zero in object 1602h sub-index 0, this will disable the PDO.

   Send the following message: SDO access to object 1602h sub-index 0, 8-bit value 0.

3. **Map the new objects**:

   - Write in object 1602h sub-index 1 the description of the interpolated data record sub-index 1:

     Send the following message: SDO access to object 1602h sub-index 1, 32-bit value 60C10120h.

   - Write in object 1602h sub-index 2 the description of the interpolated data record sub-index 2:

     Send the following message: SDO access to object 1602h sub-index 2, 32-bit value 60C10220h.

4. **Enable the RPDO3**. Set the object 1602h sub-index 0 with the value 2.

   Send the following message: SDO access to object 1602h sub-index 0, 8-bit value 2.

5. **Add the new TPDO to the Sync Manager**:

   - Write zero in object 1C12h sub-index 0, this will disable the Sync. Manager.

     Send the following message: SDO access to object 1C12h sub-index 0, 8-bit value 00h.

   - Write in object 1C12h sub-index 3 the RPDO3 mapping parameter object number:

     Send the following message: SDO access to object 1C12h sub-index 3, 16-bit value 1602h.

   - Write 03 h in object 1C12h sub-index 0, this will enable the Sync. Manager.

     Send the following message: SDO access to object 1C12h sub-index 0, 8-bit value 03h.

   Note: if using TwinCAT System Manager, enter in Configuration Mode, select the drive, select Process Data tab, uncheck the PDO Assignment and PDO Configuration boxes. Click Load PDO info from device button to load the new PDO configuration. Press F4 to reload the IO devices and enter in Operation state.

6. Enter **Safe-Operational** state.
7. Enter **Operational** state.
8. **Ready to switch on.** Set in **Control Word** mapped in RPDO1 the value 06h.
9. **Switch on.** Set in **Control Word** mapped in RPDO1 the value 07h.
10. **Enable Operation and set an absolute motion.** Set in **Control Word** mapped in RPDO1 the value 0Fh.
11. Set in **Modes of operation** mapped in RPDO1 the value 7 to enable Interpolated mode.
12. **Interpolation sub mode select**. Select PVT interpolation position mode.

    Send the following message: SDO access to object 60C0h, 16-bit value FFFFh.

13. **Interpolated position buffer length**.

    Send the following message: SDO access to object 2073h, 16-bit value 000Fh. The maximum is 000Fh.

14. **Interpolated position buffer configuration**. By setting the value B001h, the buffer is cleared and the integrity counter will be set to 1.

    Send the following message: SDO access to object 2074h, 16-bit value B001h.

15. **Loading the PVT points.** Assuming X1 and X2 are 60C1 sub index 01 and 02 which were recently mapped, send the following data:

16. **Send the 1st PVT point**.

    Position = 88 IU (0x000058) 1IU = 1 encoder pulse

    Velocity = 3.33 IU (0x000354) 1IU = 1 encoder pulse/ 1 control loop

    Time = 55 IU (0x37) 1IU = 1 control loop = 1ms by default

    IC = 1 (0x01) IC=Integrity Counter

Set X1=00540058h; X2=02370003h;

17. **Send the 2nd PVT point**.

    Position = 370 IU (0x000172)

    Velocity = 6.66 IU (0x0006A8)

    Time = 55 IU (0x37)

    IC = 2 (0x02)

Set X1=00A80172h; X2=04370006h;

18. **Send the 3rd PVT point**.

    Position = 2982 IU (0x000BA6)

    Velocity = 6.66 IU (0x0006A8)

    Time = 390 IU (0x186)

    IC = 3 (0x03)

Set X1=00A80BA6h; X2=07860006h;

19. **Send the 4th PVT point**.

    Position = 5631 IU (0x0015FF)

    Velocity = 6.66 IU (0x0006A8)

    Time = 400 IU (0x190)

    IC = 4 (0x04)

Set X1=00A815FFh; X2=09900006h;

20. **Send the 5th PVT point**.

    Position = 5925 IU (0x001725)

    Velocity = 3.00 IU (0x000300)

    Time = 60 IU (0x3C)

    IC = 5 (0x05)

Set X1=00001725h; X2=0A3C0003h;

21. **Send the 6th PVT point**.

    Position = 6000 IU (0x001770)

    Velocity = 0.00 IU (0x000000)

    Time = 50 IU (0x32)

    IC = 6 (0x06)

Set X1=00001770h; X2=0C320000h;

22. **Send the 7th PVT point**.

    Position = 5127 IU (0x001407)

    Velocity = -7.5 IU (0xFFF880)

    Time = 240 IU (0xF0)

    IC = 7 (0x07)

Set X1=00801407h; X2=0EF0 FFF8h;

23. **Send the 8th PVT point**.

    Position = 3115 IU (0x000C2B)

    Velocity = -13.33 IU (0xFFF2AB)

    Time = 190 IU (0xBE)

    IC = 8 (0x08)

Set X1=00AB0C2B h; X2=10BEFFF2h;

24. **Send the 9th PVT point**.

Position = -1686 IU (0xFFF96A)

Velocity = -13.33 IU (0xFFF2AB)

Time = 360 IU (0x168)

IC = 9 (0x09)

Set X1=FFABF96A h; X2=1368FFF2h;

25. Send the 10nth PVT point.

Position = -7145 IU (0xFFE417)

Velocity = -13.33 IU (0xFFF2AB)

Time = 410 IU (0x19A)

IC = 10 (0x0A)

Set X1=FFABE417 h; X2=159AFFF2h;

26. Send the 11th PVT point.

Position = -9135 IU (0xFFDC51)

Velocity = -7.4 IU (0xFFF899)

Time = 190 IU (0xBE)

IC = 11 (0x0B)

Set X1=FF990C2B h; X2=16BEFFF8h;

27. Send the 12th PVT point. The last.

Position = -10000 IU (0xFFD8F0)

Velocity = -7.4 IU (0x000000)

Time = 240 IU (0xF0)

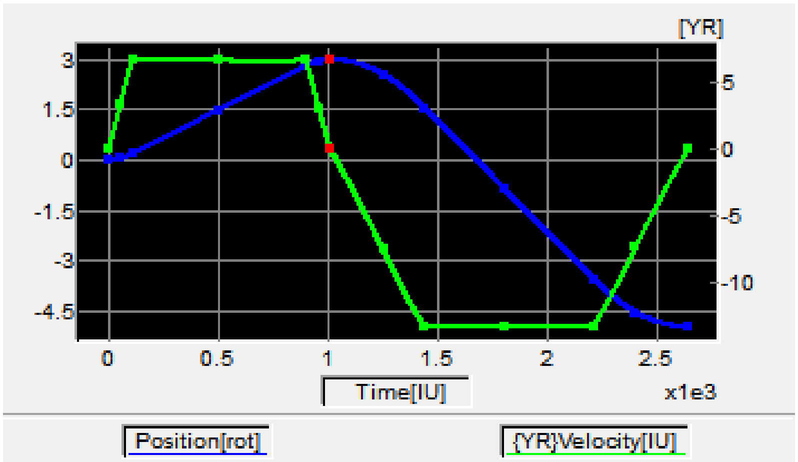
IC = 12 (0x0C)

Set X1=FF00D8F0 h; X2=18F00000h;

28. Start an absolute PVT motion.

Set in **Control Word** mapped in RPDO1 the value 1Fh.

The PVT motion should be like the one below.



The motor should rotate 3 positive rotations and another 8 negatively (for a 500 lines encoder). If the initial position before the motion was 0, the final position should be -10000 IU (-5 rotations). All points should be executed within 2.64s, considering the default time base is 1ms.

9.6 PVT relative movement example

Execute a relative PVT movement. The PVT position points will be given as a difference between next and last position.

Remarks: *Because this is a demo for a single axis the synchronization mechanism is not used here.*

To work with this mode, object 208Eh bit8 must be 0. The default value of this bit is 1.

1. **Start remote node**.

   Enter **Pre-Operational** state.

2. **Disable the RPDO3**. Write zero in object 1602h sub-index 0, this will disable the PDO.

   Send the following message: SDO access to object 1602h sub-index 0, 8-bit value 0.

3. **Map the new objects**:
   - Write in object 1602h sub-index 1 the description of the interpolated data record sub-index 1:
   Send the following message: SDO access to object 1602h sub-index 1, 32-bit value 60C10120h.
   - Write in object 1602h sub-index 2 the description of the interpolated data record sub-index 2:

   Send the following message: SDO access to object 1602h sub-index 2, 32-bit value 60C10220h.
4. **Enable the RPDO3**. Set the object 1602h sub-index 0 with the value 2.

   Send the following message: SDO access to object 1602h sub-index 0, 8-bit value 2.
5. **Add the new TPDO to the Sync Manager**:
   - Write zero in object 1C12h sub-index 0, this will disable the Sync. Manager.
   Send the following message: SDO access to object 1C12h sub-index 0, 8-bit value 00h.
   - Write in object 1C12h sub-index 3 the RPDO3 mapping parameter object number:

   Send the following message: SDO access to object 1C12h sub-index 3, 16-bit value 1602h.
   - Write 03 h in object 1C12h sub-index 0, this will enable the Sync. Manager.
   Send the following message: SDO access to object 1C12h sub-index 0, 8-bit value 03h.

   Note: if using TwinCAT System Manager, enter in Configuration Mode, select the drive, select Process Data tab, uncheck the PDO Assignment and PDO Configuration boxes. Click Load PDO info from device button to load the new PDO configuration. Press F4 to reload the IO devices and enter in Operation state.
6. Enter **Safe-Operational** state.
7. Enter **Operational** state.
8. **Ready to switch on.** Set in **Control Word** mapped in RPDO1 the value 06h.
9. **Switch on.** Set in **Control Word** mapped in RPDO1 the value 07h.
10. **Enable Operation and set a relative motion.** Set in **Control Word** mapped in RPDO1 the value 4Fh. For an absolute motion, set 0Fh but the example points will not apply.
11. Set in **Modes of operation** mapped in RPDO1 the value 7 to enable Interpolated mode.
12. **Interpolation sub mode select**. Select PVT interpolation position mode.

    Send the following message: SDO access to object 60C0h, 16-bit value FFFFh.
13. **Interpolated position buffer length**.

    Send the following message: SDO access to object 2073h, 16-bit value 000Ch. The maximum is 000Fh.
14. **Interpolated position buffer configuration**. By setting the value A001h, the buffer is cleared and the integrity counter will be set to 1.

    Send the following message: SDO access to object 2074h, 16-bit value A001h.
15. **Loading the PVT points.** Assuming X1 and X2 are 60C1 sub index 01 and 02 which were recently mapped, send the following data:
16. **Send the 1st PVT point**.

    Position = 400 IU (0x000190) 1IU = 1 encoder pulse

    Velocity = 3.00 IU (0x000300) 1IU = 1 encoder pulse/ 1 control loop

    Time = 250 IU (0xFA) 1IU = 1 control loop = 1ms by default

    IC = 1 (0x01) IC=Integrity Counter

Set X1=00000190h; X2=02FA0003h;

17. **Send the 2nd PVT point**.

    Position = 1240 IU (0x0004D8)

    Velocity = 6.00 IU (0x000600)

    Time = 250 IU (0xFA)

    IC = 2 (0x02)

Set X1=000004D8h; X2=04FA0006h;

18. **Send the 3rd PVT point**.

    Position = 1674 IU (0x00068A)

    Velocity = 6.00 IU (0x000600)

Time = 250 IU (0xFA)

IC = 3 (0x03)

Set X1=0000068Ah; X2=06FA0006h;

19. Send the 4th PVT point.

Position = 1666 IU (0x000682)

Velocity = 6.00 IU (0x000600)

Time = 250 IU (0xFA)

IC = 4 (0x04)

Set X1=00000682h; X2=08FA0006h;

20. Send the 5th PVT point.

Position = 1240 IU (0x0004D8)

Velocity = 3.00 IU (0x000300)

Time = 250 IU (0xFA)

IC = 5 (0x05)

Set X1=000004D8h; X2=0AFA0003h;

21. Send the last PVT point.

Position = 410 IU (0x00019A)

Velocity = 0.00 IU (0x000000)

Time = 250 IU (0xFA)

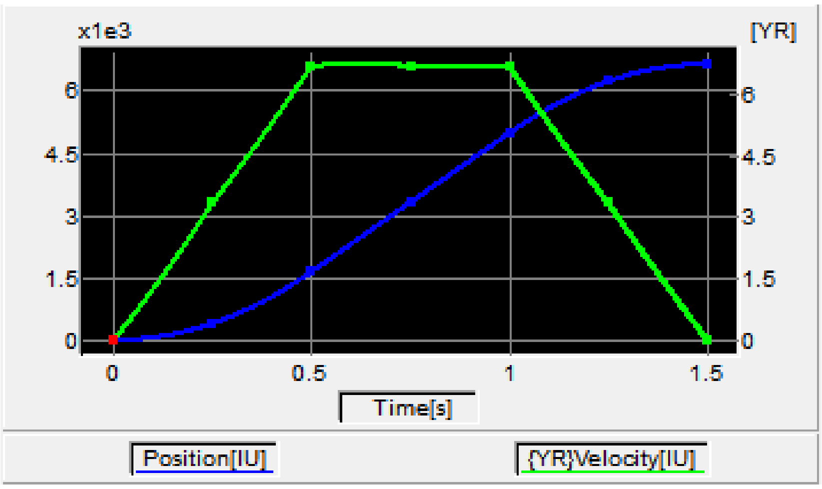
IC = 6 (0x06)

Set X1=0000019A h; X2=0CFA0000h;

22. Start a relative PVT motion.

Set in **Control Word** mapped in RPDO1 the value 5Fh.

The PVT motion should be like the one below.



If the initial position before the motion was 0, the final position should be 6630 IU (3.315 rotation for a 500line encoder). All points should be executed in 1.5s, considering the default time base is 1ms.

10 Velocity Profile Mode

10.1 Overview

In the Velocity Profile Mode the drive performs speed control. The built-in reference generator computes a speed profile with a trapezoidal shape, due to a limited acceleration. The **Target Velocity** object (index 60FFh) specifies the jog speed (speed sign specifies the direction) and the **Profile Acceleration** object (index 6083h) the acceleration/deceleration rate. While the mode is active, any change of the Target Velocity object by the EtherCAT® master will update the drive's demand velocity enabling you to change on the fly the slew speed and/or the acceleration/deceleration rate. The motion will continue until the **Halt** bit from the Controlword is set. An alternate way to stop the motion is to set the jog speed to zero.

While the mode is active (profile velocity mode is selected in *modes of operation*), every time a write access is performed inside the object *target velocity*, the demand velocity of the drive is updated.

Remark1: This mode works only if the speed loop is active in Drive setup/ Advanced button.

Remark2: If the velocity is already set when entering velocity mode, the motion will not start until a value (even if it is the same) will be set again in Target Velocity 60FFh.

10.1.1 Controlword in Profile Velocity mode

MSB **LSB**

See 6040h	Halt	See 6040h	reserved	See 6040h
15 9	8	7	6 4	3 0

***Table 10.1.1** – Controlword bits for Profile Velocity mode*

Name	Value	Description
Halt	0	Execute the motion
	1	Stop drive with *profile acceleration*

10.1.2 Statusword in Profile Velocity mode

MSB **LSB**

See 6041h	Max slippage error	Speed	See 6041h	Target reached	See 6041h
15 14	13	12	11	10	9 0

***Table 10.1.2** – Statusword bits for Profile Velocity mode*

Name	Value	Description
Target reached	0	Halt = 0: *Target velocity* not (yet) reached Halt = 1: Drive decelerates
	1	Halt = 0: *Target velocity* reached Halt = 1: Velocity of drive is 0
Speed	0	Speed is not equal to 0
	1	Speed is equal to 0
Max slippage error	0	Maximum slippage not reached
	1	Maximum slippage reached

***Remark**: In order to set / reset bit 12 (speed), the object 606Fh, velocity threshold is used. If the actual velocity of the drive / motor is below the velocity threshold, then bit 12 will be set, else it will be reset.*

10.2 Velocity Mode Objects

10.2.1 Object 6069h: Velocity sensor actual value

This object describes the value read from the velocity encoder in increments.

The velocity units are user defined speed units. The value can be converted into internal units using the *velocity factor*

If no factor is applied, then the value 65536 = 1 encoder increment / sample.

Object description:

Index	6069h
Name	Velocity sensor actual value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Value range	INTEGER32
Default value	-

10.2.2 Object 606Bh: Velocity demand value

This object provides the output of the trajectory generator and is provided as an input for the velocity controller. It is given in user-defined velocity units.

If no factor is applied, then the value 65536 = 1 encoder increment / sample.

Object description:

Index	606Bh
Name	Velocity demand value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Value range	INTEGER32
Default value	-

10.2.3 Object 606Ch: Velocity actual value

The *velocity actual value* is given in user-defined velocity units and is read from the velocity sensor.

If no factor is applied, then the value 65536 = 1 encoder increment / sample.

Object description:

Index	606Ch
Name	Velocity actual value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Yes
Value range	INTEGER32
Default value	-

10.2.4 Object 606Dh: Velocity window[1]

When the difference between the target velocity (60FFh) and the velocity actual value (606Ch) is in the velocity window for longer than the velocity window time (606Eh), the *Target reached bit* (Statusword) is set. The value is given in user-defined velocity units which means it can be modified by Factor group objects.

Object description:

Index	606Dh
Name	Velocity window
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Yes
Value range	UNSIGNED16
Default value	-

10.2.5 Object 606Eh: Velocity window time[2]

When the difference between the target velocity (60FFh) and the velocity actual value (606Ch) is in the velocity window (606Dh) for longer than the velocity window time, the *Target reached bit* (Statusword) is set. The value is given in milliseconds.

Object description:

[1] Available starting with F515K / FA00x firmware version

[2] Available starting with F515K / FA00x firmware version

Index	606Eh
Name	Velocity window time
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Yes
Value range	UNSIGNED16
Default value	-

10.2.6 Object 606Fh: Velocity threshold

The *velocity threshold* is given in user-defined velocity units and it represents the threshold for velocity at which it is regarded as zero velocity. Based on its value, bit 12 of *Statusword* (speed) will be set or reset.

If no factor is applied, then the value 65536 = 1 encoder increment / sample.

Object description:

Index	606Fh
Name	Velocity threshold
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Possible
Value range	UNSIGNED16
Default value	-

10.2.7 Object 60FFh: Target velocity

This object is used for the velocity command only when 6060h Modes of Operation is 3 (Speed Mode).

The *target velocity* is the input for the trajectory generator and the value is given in user-defined velocity units.

If no factor is applied, then the value 65536 = 1 encoder increment / sample.

Object description:

Index	60FFh
Name	Target velocity
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	possible
Value range	INTEGER32
Default value	-

10.2.8 Object 60F8h: Max slippage

The *max slippage* monitors whether the maximum speed error has been reached. The value is given in user-defined velocity units. When the *max slippage* has been reached, the corresponding bit 13 *max slippage error* in the *Statusword* is set and the drive will fault by signalizing a control error (MER register/object 2000h bit3=1).

The Speed control error is active only if the speed loop is active in setup. By default it is disabled. The speed control error is set when the actual speed error is greater than what is defined in object 60F8h for a time defined in object 2005h.

Object description:

Index	60F8h
Name	Max slippage
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	possible
Value range	INTEGER32
Default value	-

This object is automatically set in Drive Setup by modifying the Speed control error. To modify the speed control error in setup, check the Speed radio button under control in Drive Setup and re-check the position button when done. Even if the GUI does not allow modification, if checked, the protection will still be active.

The value for this object can be changed by editing the parameter "SERRMAX" found in parameters.cfg of the project file.

If no factor is applied, then the value 65536 = 1 encoder increment / sample.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

10.2.9 Object 2005_h: Max slippage time out

Time interval for *max slippage*. The value is given in slow loop (control loop) time which is by default set to 1ms. This object is coupled with *Object 60F8h: Max slippage*.

Object description:

Index	2005_h
Name	Max slippage time out
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	-

The value for this object can be changed by editing the parameter "TSERRMAX" found in parameters.cfg of the project file.

Activating *Object 2076h: Save current configuration*, will set its current values as the a new default.

10.2.10 Object 2087_h[1]: Actual internal velocity from sensor on motor

This object describes the velocity value read from the encoder on the motor in increments, in case a dual loop control method is used. The value is given in increments per sampling loop. The default sampling loop is 1ms.

If no factor is applied, then the value 65536 = 1 encoder increment / sample.

Object description:

Index	2087_h
Name	Actual internal velocity sensor on motor
Object code	VAR
Data type	INTEGER32

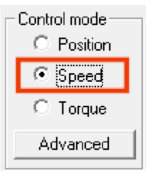
Entry description:

Access	RO
PDO mapping	Possible
Value range	INTEGER32
Default value	-

10.3 Speed profile example

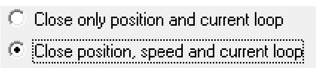
Remark: any speed profile mode can be run only if the speed loop is active in setup (by default it is disabled).

To enable the Current + Speed loop, in Drive setup, select under Control mode the speed radio button:



After the speed is selected, the tuning for the speed loop must be done.

To enable the Current + Speed + Position loop, in Drive setup, select under Control mode the Position radio button and then click the Advanced button. Under control scheme, select the "Close position, speed and current loop" radio button.



After all three loops are selected, the tuning for the speed and position must be done again.

[1] Object 2087_h applies only to drives which have a secondary feedback

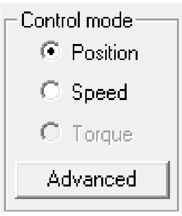
Execute a speed control with 600 rpm target speed.

1. **Start remote node**.
   Enter **Pre-Operational** state.
   Enter **Safe-Operational** state.
   Enter **Operational** state.
2. **Modes of operation.** Select speed mode.
   Set in **Modes of Operation** mapped in RPDO1 the value 03_h.
3. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.
   Set in **Control Word** mapped in RPDO1 the value 06_h.
4. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.
   Set in **Control Word** mapped in RPDO1 the value 07_h.
5. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.
   Set in **Control Word** mapped in RPDO1 the value $0F_h$.
6. **Target velocity.** Set the target velocity to 600 rpm. By using a 500 lines incremental encoder and 1ms sample rate for position/speed control the corresponding value of object $60FF_h$ expressed in encoder counts per sample is 140000_h (20.0 IU).
   Send the following message: SDO access to object $60FF_h$ 32-bit value 00140000_h.
7. **Check the motor actual speed.** It should rotate with 600 rpm.

Read by SDO protocol the value of object $606C_h$.

10.4 Speed profile example for stepper open loop

Remark: In the case of stepper open-loop control, speed control is possible irrespective of the chosen control mode, whether it is speed or position. However, it is crucial to emphasize that for proper operation, the current controller needs to be tuned.



Execute a speed control with 300 rpm target speed.

1. **Start remote node**.
   Enter **Pre-Operational** state.
   Enter **Safe-Operational** state.
   Enter **Operational** state.
2. **Modes of operation.** Select speed mode.
   Set in **Modes of Operation** mapped in RPDO1 the value 03_h.
3. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.
   Set in **Control Word** mapped in RPDO1 the value 06_h.
4. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.
   Set in **Control Word** mapped in RPDO1 the value 07_h.
5. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.
   Set in **Control Word** mapped in RPDO1 the value $0F_h$.
6. **Target velocity.** Set the target velocity to 300 rpm. By using a stepper open-loop control with 200 steps and µstepping set to 512, along with a 1 ms sample rate for position and speed control (slow loop), the 300 rpm expressed in internal units (IU) is 512. This calculation takes into account that 1 rotation is equal to the product of steps number and µstep number, and 1 IU is equivalent to 1 µstep per slow loop. The value to be configured in object $60FF_h$ is 512*65536 = 33554432 = 2000000_h
   Send the following message: SDO access to object $60FF_h$ 32-bit value 00140000_h.

11 Electronic Gearing Position (EGEAR) Mode

11.1 Overview

In Electronic Gearing Position Mode the drive follows the position of an electronic gearing master with a programmable gear ratio.

The electronic gearing slave can get the position information from the electronic camming master in three ways:

1. Via EtherCAT® master, which writes the master position in object **Master position** (index 201E$_h$).
2. Via an external digital reference[1] of type pulse & direction or quadrature encoder. Both options have dedicated inputs. The pulse & direction signals are usually provided by an indexer and must be connected to the pulse & direction inputs of the drive. The quadrature encoder signals are usually provided by an encoder on the master and must be connected to the 2nd encoder inputs.
3. From one of the analogue inputs of the drive.

The reference type, i.e. the selection between the online reference received via communication channel and the digital reference read from dedicated inputs is done with object **External Reference Type** (index 201D$_h$). The source of the digital reference (pulse & direction or second encoder inputs) is set during drive commissioning.

The drive set as slave in electronic gearing mode performs a position control. At each slow loop sampling period, the slave computes the master position increment and multiplies it with its programmed gear ratio. The result is the slave position reference increment, which added to the previous slave position reference gives the new slave position reference.

***Remark:** The slave executes a relative move, which starts from its actual position*

The gear ratio is specified via **EGEAR multiplication factor** object (index 2013$_h$). EGEAR ratio numerator (sub-index 1) is a signed integer, while EGEAR ratio denominator (sub-index 2) is an unsigned integer. The EGEAR ratio numerator sign indicates the direction of movement: positive – same as the master, negative – reversed to the master. The result of the division between EGEAR ratio numerator and EGEAR ratio denominator is used to compute the slave reference increment.

The **Master Resolution** object (index 2012$_h$) provides the master resolution, which is needed to compute correctly the master position and speed (i.e. the position increment). If master position is not cyclic (i.e. the resolution is equal with the whole 32-bit range of position), set master resolution to 0x80000001.

You can smooth the slave coupling with the master, by limiting the maximum acceleration of the slave drive. This is particularly useful when the slave has to couple with a master running at high speed, in order to minimize the shocks in the slave. The feature is activated by setting Controlword.5=1 and the maximum acceleration value in Object 6083h: Profile acceleration.

11.1.1 Controlword in electronic gearing position mode (slave axis)

MSB **LSB**

See 6040$_h$	Halt	See 6040$_h$	Reserved	Activate Acceleration Limitation	Enable Electronic Gearing Mode	See 6040$_h$
15 9	8	7	6	5	4	3 0

***Table 11.1.1** – Controlword bits for Electronic Gearing Position Mode*

Name	Value	Description
Enable Electronic Gearing Mode	0	Do not start operation
	0 -> 1	Start electronic gearing procedure
	1 -> 0	Does nothing (does not stop current procedure)
Activate Acceleration Limitation	0	Do not limit acceleration when entering electronic gear mode
	1	Limit acceleration when entering electronic gear mode to the value set in *profile acceleration* (object 6083$_h$)
Halt	0	Execute the instruction of bit 4
	1	Stop drive with *profile acceleration*

11.1.2 Statusword in electronic gearing position mode

MSB **LSB**

See 6041$_h$	Following error	Reserved	See 6041$_h$	Target reached	See 6041$_h$
15 14	13	12	11	10	9 0

[1] Not all drives have a secondary encoder input.

Table 11.1.2 – Statusword bits for Electronic Gearing Position Mode

Name	Value	Description
Target reached	0	Halt = 0: Always 0 Halt = 1: Drive decelerates
	1	Halt = 0: Always 0 Halt = 1: Velocity of drive is 0
Following error	0	No following error
	1	Following error occurred

11.2 Gearing Position Mode Objects

11.2.1 Object 201Eh: Master position

This object is used in order to receive the position from the master, which is used for Electronic Gearing or Camming calculations. The position units are in increments.

Example: if it takes 4000 increments for the motor to do one revolution, these same increments apply for this object.

Object description:

Index	201Eh
Name	Master position
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	Possible
Units	Increments
Value range	0 … 2^{31}-1
Default value	-

11.2.2 Object 2012h: Master resolution

This object is used in order to set the master resolution in increments per revolution. This object is valid for the slave axis.

Object description:

Index	2012h
Name	Master resolution
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Possible
Units	Increments
Value range	0 … 2^{31}-1
Default value	80000001h (full range)

11.2.3 Object 2013h: EGEAR multiplication factor

In digital external mode, this object sets the gear ratio, or gear multiplication factor for the slaves. The sign indicates the direction of movement: positive – same as the master, negative – reversed to the master. The slave demand position is computed as the master position increment multiplied by the gear multiplication factor.

Example: if the gear ratio is Slave/Master = 1/3, the following values must be set: 1 in EGEAR ratio numerator (sub-index 1) and 3 in EGEAR ratio denominator (sub-index 2) .

Remark: the gear ratio is computed after sub-index 2 is written. So sub-index1 must be written first and then sub-index 2. Even if sub-index 2 has the same value as before, it must be written again for the gear ratio to be computed correctly.

Object description:

Index	2013h
Name	EGEAR multiplication factor
Object code	RECORD
Number of elements	2

Entry description:

Sub-index	1
Description	EGEAR ratio numerator (slave)
Object code	VAR
Data type	INTEGER16
Access	RW
PDO mapping	Possible
Value range	-32768 … 32767
Default value	1

Sub-index	2
Description	EGEAR ratio denominator (master)
Object code	VAR
Data type	UNSIGNED16
Access	RW
PDO mapping	Possible
Value range	0 … 65535
Default value	1

11.2.4 Object 2017$_h$: Master actual position

The actual position of the master can be monitored through this object, regardless of the way the master actual position is delivered to the drive (on-line through a communication channel in object 201E$_h$ or from the digital inputs of the drive). The units are increments.

Object description:

Index	2017$_h$
Name	Master actual position
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Value range	-2^{31} … 2^{31}-1
Default value	0

11.2.5 Object 2018$_h$: Master actual speed

This object is used to inform the user of the actual value of the speed of the master, regardless of the way the master actual position is delivered to the drive (on-line through a communication channel or from the digital inputs of the drive). The units are increments / sampling. 1 IU = 1 encoder increment / sample.

Object description:

Index	2018$_h$
Name	Master actual speed
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	Possible
Value range	-32768 … 32767
Default value	0

11.2.6 Object 201D$_h$: External Reference Type

This object is used to set the type of external reference for use with electronic gearing position, electronic camming position, position external, speed external and torque external modes.

Object description:

Index	201D$_h$
Name	External Reference Type
Object code	VAR
Data type	UNSIGNED16

Entry description:

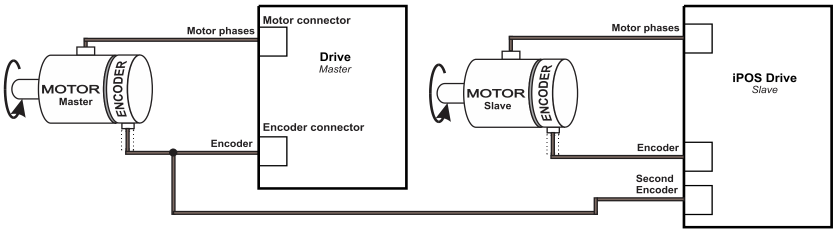
Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	-

Table 11.2.1 – *External Reference Type bit description*

Value	Description
0	Reserved
1	On-line. Received via object 201Eh.
2	Analogue. In case of External Reference Position / Speed / Torque Modes, select this option in order to read the reference from the dedicated analogue input.
3	Digital[1]. In case of External Reference Position Modes, select this option in order to read the reference from the dedicated digital inputs as set in the setup made using EasySetup / EasyMotion Studio (either 2nd encoder or pulse & direction) In case of Electronic Gearing and Camming Position Modes, select this option in order to read master position from the dedicated digital inputs as set in the setup made using EasySetup / EasyMotion Studio (either 2nd encoder or pulse & direction)
4 … 65535	Reserved

11.3 Electronic gearing through second encoder input example

The encoder from the master drive can also be connected in parallel to the second encoder input of an iPOS drive (that has one).



After connecting the master encoder to the second encoder input, the Electronic Gearing Slave can be started.

1. **Start remote node**.

   Enter **Pre-Operational** state.

   Enter **Safe-Operational** state.

   Enter **Operational** state.

2. **Modes of operation.** Select Electronic Gearing mode (-1).

   Set in **Modes of Operation** mapped in RPDO1 the value FFh.

3. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.

   Set in **Control Word** mapped in RPDO1 the value 06h.

4. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.

   Set in **Control Word** mapped in RPDO1 the value 07h.

5. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.

   Set in **Control Word** mapped in RPDO1 the value 0Fh.

6. **External reference type.** Slave receives reference through 2nd encoder input.

   Send the following message: SDO access to object 201Dh 16-bit value 0003h.

7. **Master resolution.** Set the master resolution 2000, assuming a 500 line encoder is used.

   Send the following message: SDO access to object 2012h 32-bit value 000007D0h.

8. **Electronic gearing multiplication factor.**

   Set EG numerator to 1.

   Send the following message: SDO access to object 2013h, sub-index 1, 16-bit value 0001h.

[1] Not all drives and configurations have secondary encoder inputs.

Set EG denominator to 1.

Send the following message SDO access to object 2013h,sub-index 2, 16-bit value 0001h.

9. Set the initial Master position into the associated RPDO where 201Eh is mapped.

10. **Enable/Start EG slave** in control word associated RPDO.

Set in **Control Word** mapped in RPDO1 the value 1Fh.

11. Start moving the master encoder and the slave will follow.

11.4 Electronic gearing through online communication example

Start an Electronic Gearing Slave.

1. Map in a RPDO the object 201Eh Master position to be able to send the drive the position reference every communication cycle. See *2.4 PDOs mapping general example* or paragraph *0* for a TwinCAT PDO mapping example.

The PDOs must be sent every slow loop period which is by default 1ms. It is recommended to set the SYNC 0 time equal to the communication cycle and slow loop.

2. **Start remote node**.

Enter **Pre-Operational** state.

Enter **Safe-Operational** state.

Enter **Operational** state.

3. **Modes of operation.** Select Electronic Gearing mode (-1).

Set in **Modes of Operation** mapped in RPDO1 the value FFh.

4. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.

Set in **Control Word** mapped in RPDO1 the value 06h.

5. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.

Set in **Control Word** mapped in RPDO1 the value 07h.

6. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.

Set in **Control Word** mapped in RPDO1 the value 0Fh.

7. **External reference type.** Slave receives reference through online communication.

Send the following message: SDO access to object 201Dh 16-bit value 0001h.

8. **Master resolution.** Set the master resolution 2000, assuming a 500 line encoder is used.

Send the following message: SDO access to object 2012h 32-bit value 000007D0h.

9. **Electronic gearing multiplication factor.**

Set EG numerator to 1.

Send the following message: SDO access to object 2013h, sub-index 1, 16-bit value 0001h.

Set EG denominator to 1.

Send the following message SDO access to object 2013h,sub-index 2, 16-bit value 0001h.

10. Set the initial Master position into the associated RPDO where 201Eh is mapped.

11. **Enable EG slave** in control word associated RPDO.

Set in **Control Word** mapped in RPDO1 the value 1Fh.

12. Start changing the Master position in the RPDO where 201Eh is mapped every communication cycle.

The slave motor should start rotating with the same speed as the difference between the master position values received every slow loop.

12 Electronic Camming Position (ECAM) Mode

12.1 Overview

In Electronic Camming Position the drive executes a cam profile function of the position of an electronic camming master. The cam profile is defined by a cam table – a set of (X, Y) points, where X is cam table input i.e. the position of the electronic camming master and Y is the cam table output i.e. the corresponding slave position. Between the points the drive performs a linear interpolation.
The electronic camming slave can get the position information from the electronic camming master in three ways:

1. Via EtherCAT® master, who writes the master position in object **Master position** (index 201Eh).
2. Via an external digital reference of type pulse & direction or quadrature encoder. Both options have dedicated inputs. The pulse & direction signals are usually provided by an indexer and must be connected to the pulse & direction inputs of the drive. The quadrature encoder signals are usually provided by an encoder on the master and must be connected to the 2nd encoder inputs.
3. From one of the analogue inputs of the drive.

The reference type, i.e. the selection between the online reference received via communication channel and the digital reference read from dedicated inputs is done with object **External Reference Type** (index 201Dh). The source of the digital reference (pulse & direction or second encoder inputs) is set during drive commissioning.
The electronic camming position mode can be: **relative** (if ControlWord.6 = 0) or **absolute** (if ControlWord.6 = 1).
In the relative mode, the output of the cam table is added to the slave actual position. At each slow loop sampling period the slave computes a position increment **dY = Y – Yold**. This is the difference between the actual cam table output Y and the previous one Yold. The position increment dY is added to the old demand position to get a new demand position. The slave detects when the master position rolls over, from 360 degrees to 0 or vice-versa and automatically compensates in dY the difference between **Ymax** and **Ymin**. Therefore, in relative mode, you can continuously run the master in one direction and the slaves will execute the cam profile once at each 360 degrees with a glitch-free transition when the cam profile is restarted.
When electronic camming is activated in relative mode, the slave initializes **Yold** with the first cam output computed: **Yold = Y = f(X)**. The slave will keep its position until the master starts to move and then it will execute the remaining part of the cam. For example if the master moves from X to Xmax, the slave moves with Ymax – Y.
In the absolute mode, the output of the cam table Y is the demand position to reach.
Remark: *The absolute mode must be used with great care because it may generate abrupt variations on the slave demand position if:*

Slave position is different from Y at entry in the camming mode
Master rolls over and Ymax < Ymin

In the absolute mode, you can introduce a maximum speed limit to protect against accidental sudden changes of the positions to reach. The feature is activated by setting ControlWord.5=1 and the maximum speed value in object **Profile Velocity** (index 6081h).
Typically, the cam tables are first downloaded into the EEPROM memory of the drive by the EtherCAT® master or with EasyMotion Studio. Then using the object **CAM table load address** (index 2019h) they are copied in the RAM address set in object **CAM table run address** (index 201Ah). It is possible to copy more than one cam table in the drive/motor RAM memory. When the ECAM mode is activated it uses the CAM table found at the RAM address contained in **CAM table run address**.
A CAM table can be shifted, stretched or compressed.

12.1.1 Controlword in electronic camming position mode

MSB **LSB**

See 6040h	Halt	See 6040h	Abs / Rel	Activate Speed Limitation	Enable Electronic Camming Mode	See 6040h
15 9	8	7	6	5	4	3 0

Table 12.1.1 *– Controlword bits for electronic camming position mode*

Name	Value	Description
Enable Electronic Camming Mode	0	Do not start operation
	0 -> 1	Start electronic camming procedure
	1 -> 0	Do nothing (does not stop current procedure)
Activate Speed Limitation	0	Do not limit speed when entering absolute electronic camming mode
	1	Limit speed when entering absolute electronic camming mode at the value set in *profile velocity* (ONLY for absolute mode)
Abs / Rel	0	Perform relative camming mode – when entering the camming mode, the slave will compute the cam table relative to the starting moment.
	1	Perform absolute camming mode – when entering the camming mode, the slave will go to the absolute position on the cam table
Halt	0	Execute the instruction of bit 4
	1	Stop drive with *profile acceleration*

12.1.2 Statusword in electronic camming position mode

MSB **LSB**

See 6041h		Following error	Reserved	See 6041h	Target reached	See 6041h	
15	4	13	12	11	10	9	0

***Table 12.1.2** – Statusword bits for electronic camming position mode*

Name	Value	Description
Target reached	0	Halt = 0: Always 0 Halt = 1: Drive decelerates
	1	Halt = 0: Always 0 Halt = 1: Velocity of drive is 0
Following error	0	No following error
	1	Following error occurred

12.2 Electronic Camming Position Mode Objects

12.2.1 Object 2019h: CAM table load address

This is the **load address** of the CAM table. The CAM table is stored in EEPROM memory of the drive starting from the load address. The initialization of the electronic camming mode requires the CAM table to be copied from the EEPROM memory to the RAM memory of the drive, starting from the **run address**, set in object 201Ah, for faster processing. The copy is made every time object 2019h is written by SDO access.

***Remark:** The **CAM table run address** object must be set before writing the object **CAM table load address** to assure a proper copy operation from EEPROM to RAM memory.*

Object description:

Index	2019h
Name	CAM table load address
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Units	-
Value range	UNSIGNED16
Default value	Variable depending on motor + feedback configuration

12.2.2 Object 201Ah: CAM table run address

This is the run address of the CAM table e.g. the RAM address starting from which the CAM table is copied into the RAM during initialization of the electronic camming mode. (See also 2019h).

Object description:

Index	201Ah
Name	CAM table run address
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Units	-
Value range	UNSIGNED16
Default value	9E00h

12.2.3 Object 201Bh: CAM offset

This object may be used to shift the master position in electronic camming mode. The position actually used as X input in the cam table is not the master actual position (2017h) but (master actual position – CAM offset) computed as modulo of master resolution (2012h) The CAM offset must be set before enabling the electronic camming mode. The *CAM offset* is expressed in increments.

Object description:

Index	201Bh
Name	CAM offset
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	No
Value range	0 … 2^{32}-1
Default value	0

12.2.4 Object 206Bh: CAM: input scaling factor

You can use this scaling factor in order to achieve a scaling of the input values of a CAM table. Its default value of 00010000h corresponds to a scaling factor of 1.0.

Object description:

Index	206Bh
Name	CAM input scaling factor
Object code	VAR
Data type	FIXED32

Entry description:

Access	RW
PDO mapping	Possible
Units	-
Value range	FIXED32
Default value	00010000h

12.2.5 Object 206Ch: CAM: output scaling factor

You can use this scaling factor in order to achieve a scaling of the output values of a CAM table. Its default value of 00010000h corresponds to a scaling factor of 1.0.

Object description:

Index	206Ch
Name	CAM output scaling factor
Object code	VAR
Data type	FIXED32

Entry description:

Access	RW
PDO mapping	Possible
Units	-
Value range	FIXED32
Default value	00010000h

12.2.6 Building a CAM profile and saving it as an .sw file example

Build your own cam profile in any program you like.

In this example, we have used MS Excel.

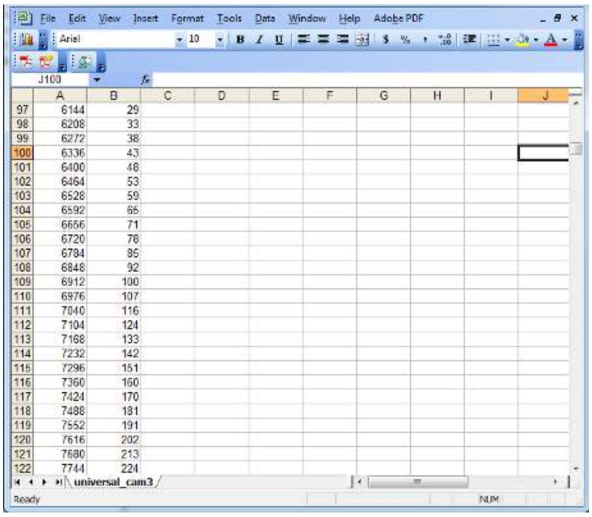


Figure 12.2.1. *MS Excel interface*

The numbers in the columns represent the input and output of the cam file. They are position points represented in the drive's internal units. Let us say that we have a 500 line quadrature encoder on the motor. This means that we will have 2000 counts per motor revolution. So the drive will rotate the rotor once if it receives a position command of 2000 internal units, or it will return 2000 internal units if the rotor turned once.

The first column represents the input position. It is a series of numbers that represent an interpolation step. Meaning that the difference between the values must be a number from the following: 2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6 and 2^7. So let us say that we choose interpolation step of 2^6 (64). The first number in the first column must be 0, the second number must be 64,the third number must be 128 and so on.

The second column represents the Output of the cam file. This number can be anything that fits in an Integer32 bit variable.

For example, let us say we have in the first column the number 640 (which is a multiple of 2^6) and in the second column we have the number 4000. This means that if the master is at position 640 (internal units), the slave must be at the position 4000 (internal units).

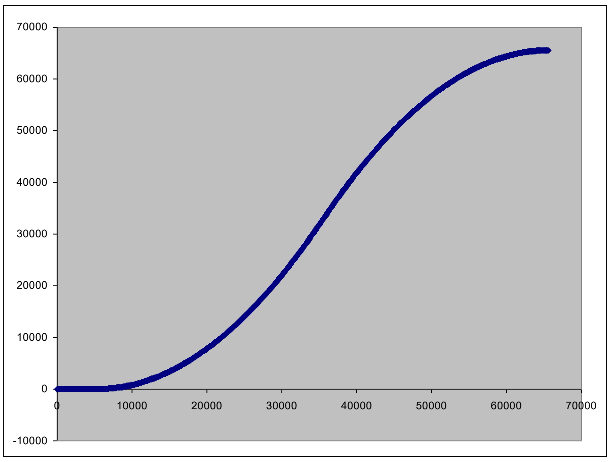


Figure 12.2.2. *Cam example*

After the cam is ready, save it as Text (Tab delimited) (*.txt) file.

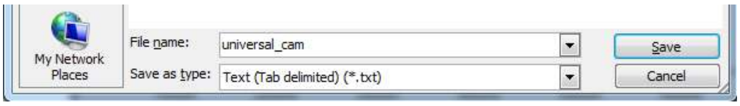


Figure 12.2.3. *Save As example.*

Once you have your cam file saved, start EasyMotion Studio, even the demo[1] version.

Press **New** button  and select your drive type.

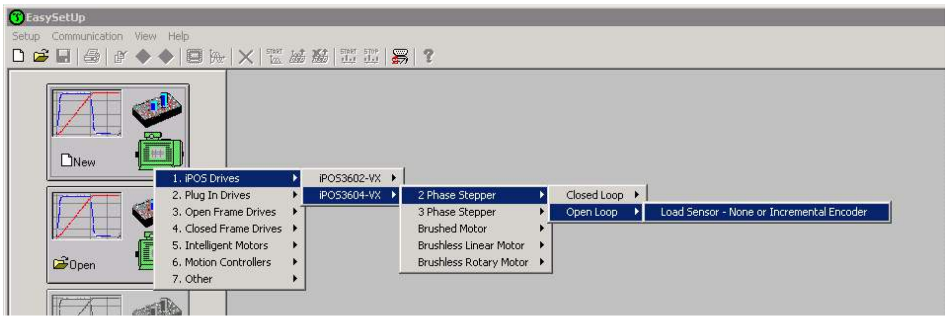


Figure 12.2.4. *Choose drive configuration.*

After the project opens, select CAM Tables tab from the left of the screen. Press the import button and choose your recently saved cam file (see **Figure 12.2.5**).

[1] ESM demo version available in download section here.

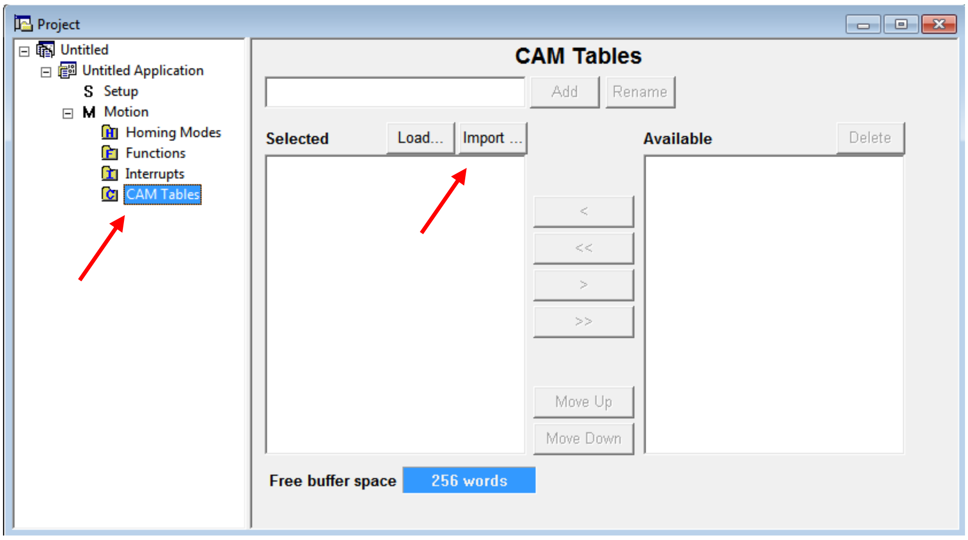


Figure 12.2.5. *CAM tab.*

If the CAM file loaded, it should look like this:

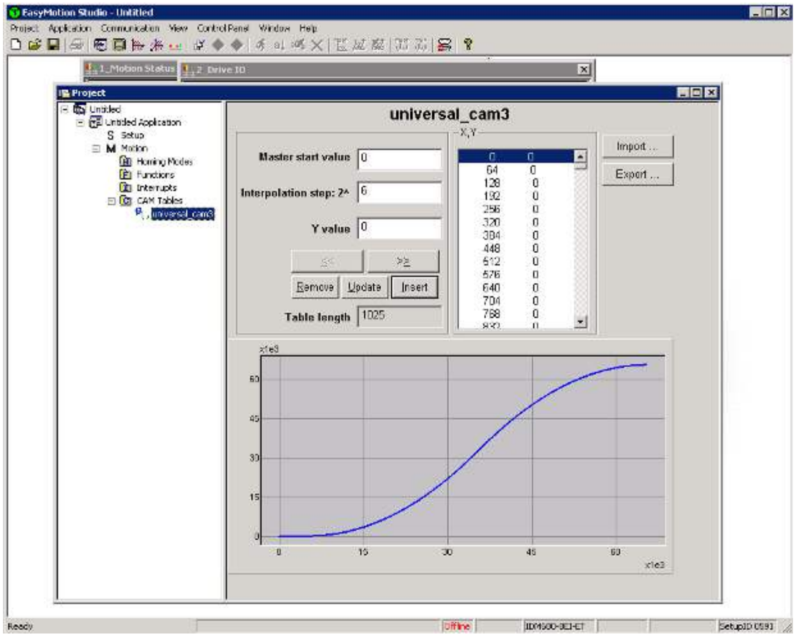
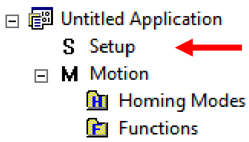
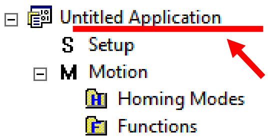


Figure 12.2.6. *CAM file loaded.*

After loading the CAM file successfully, click over the Setup 

tab and load your saved setup[1].

Click the tab with the name of the application 

.

Press the memory settings button (like in the figure below).

[1] To create a setup file, please check your drive's user manual.

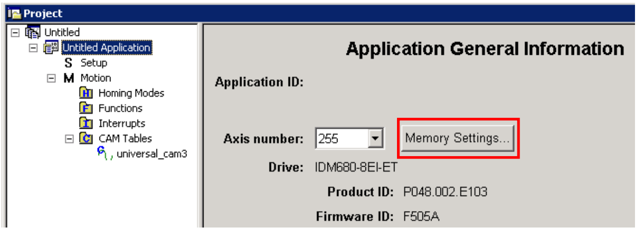


Figure 12.2.7. *Memory Settings location.*

In the window below, see if necessary CAM space is larger than reserved cam space. If it is, write a slightly larger number than the necessary CAM space in the reserved one (Figure below).

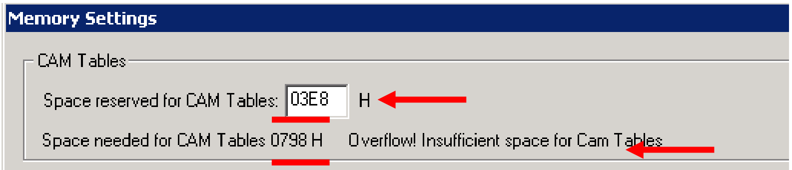


Figure 12.2.8. *Adjusting the necessary CAM space.*

In Memory Settings window look inside EEPROM memory section under CAM Tables. The first number is the **cam table Load Address** that must be set also in object 2019h afterwards.

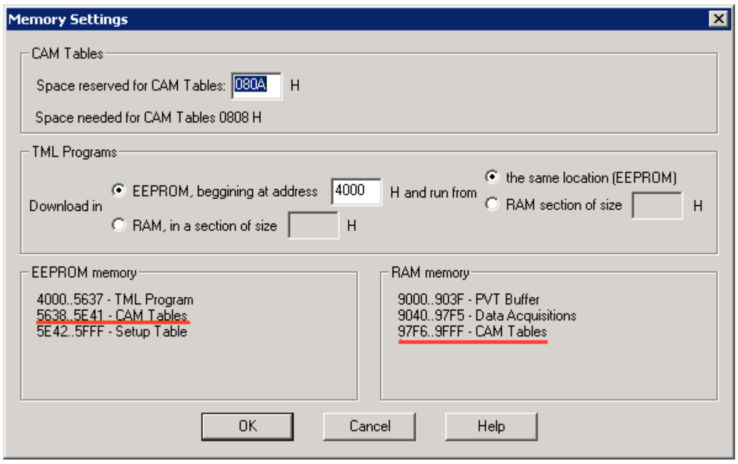


Figure 12.2.9. *Cam table load and run addresses.*

Under the RAM memory section the first number in CAM Tables is the **cam table Run Address** that must also be set in object 201Ah afterwards.

Save the project and select Application -> Create EEPROM programmer file -> Motion and Setup… like in the figure below. Save the EEPROM file that includes your setup and motion (including CAM data) onto your PC.

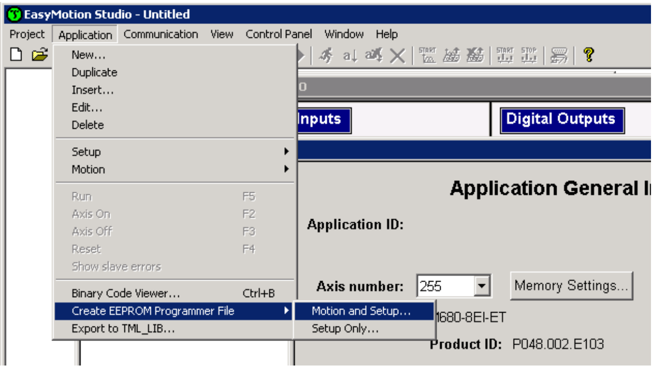


Figure 12.2.10. *Create .sw file.*

12.2.6.1 Extracting the cam data from the motion and setup .sw file

Open the recently saved .sw file with any text editor.

Inside the .sw file search for the number that corresponds to the CAM Table load address.

This number shall be delimited by an empty new line just before it (**Figure 12.2.11**) (the numbers before it represent the setup data).

Select all these numbers that represent the cam file until you find another empty new line (**Figure 12.2.12**).

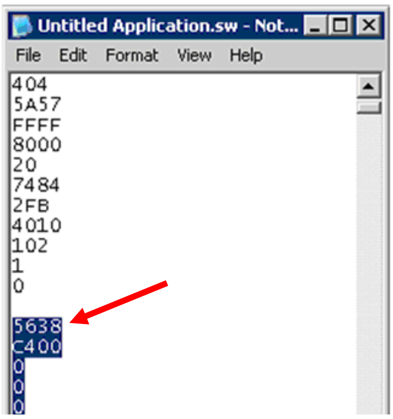


Figure 12.2.11. *.sw file structure example*

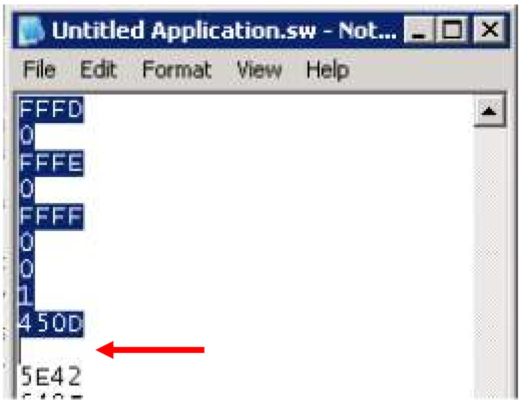


Figure 12.2.12. *.sw file empty line*

Copy all these numbers and save them as a new text file with the extension .sw instead of .txt.

Now you have a file that can be loaded onto the drive either with THS EEPROM Programmer (supplied free with EasySetup or ESM) or load it with the help of **2064h 2065h** objects explained in next sub chapter.

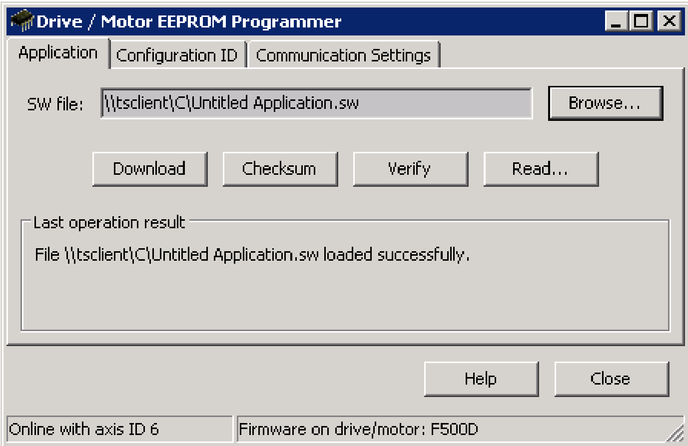


Figure 12.2.13. *THS EEPROM Programmer.*

Note: with the THS EEPROM programmer, you can write the entire setup and motion .sw file, not just the CAM .sw file created in this example.

12.2.6.2 Downloading a CAM .sw file with objects 2064h and 2065h example

- Send the following message: SDO access to object 2064h 32-bit value xxxx0008h.

Where xxxx is the first 16 bit number found in the CAM .sw file and represents the CAM table load address. The 08 activates writing/reading 16 bit data in EEPROM memory in object 2064h (see more in the object description).

All the next numbers until the end of the file must be written with the following type of command.

- Send the following message: SDO access to object 2065h 32-bit value 0000xxxxh.

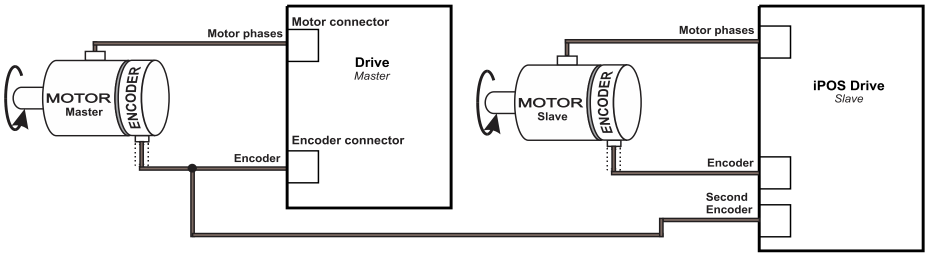
Where xxxx is the 16 bit number taken from the .sw file (the data to write) after the first one (which is the address at which to first write the data and increment it).

Warning!	When object 2064$_h$ bit 7=0 (auto-incrementing is ON), do not read the object list in parallel with a read/write operation using a script. By reading object 2066h in parallel with another application, the target memory address will be incremented and will lead to incorrect data writing or reading.

12.3 Electronic camming through second encoder input example

Start an Electronic Gearing Slave.

The encoder from the master drive can also be connected in parallel to the second encoder input of an iPOS drive (that has one).



After connecting the master encoder to the second encoder input, the Electronic Gearing Slave can be started.

1. **Start remote node**.
   Enter **Pre-Operational** state.
   Enter **Safe-Operational** state.
   Enter **Operational** state.
2. **Modes of operation.** Select Electronic Camming mode (-2).
   Set in **Modes of Operation** mapped in RPDO1 the 8 bit value 0xFE.
3. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.
   Set in **Control Word** mapped in RPDO1 the value 06$_h$.
4. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.
   Set in **Control Word** mapped in RPDO1 the value 07$_h$.
5. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.
   Set in **Control Word** mapped in RPDO1 the value 0F$_h$.
6. **External reference type.** Slave receives reference through 2nd encoder input.
   Send the following message: SDO access to object 201D$_h$ 16-bit value 0003$_h$.
7. **Cam table load address.** Set cam table load address as **5638**$_h$.
   The cam table load address can be discovered as explained in paragraph *12.2.6* .
   Send the following message: SDO access to object 2019$_h$ 16-bit value 5638$_h$.
8. **Cam table run address.** Set cam table load address as **97F6**$_h$.
   The cam table load address can be discovered as explained in paragraph *12.2.6* .
   Send the following message: SDO access to object 201A$_h$ 16-bit value 97F6$_h$.
9. **Master resolution.** Set the master resolution 2000, assuming a 500 line encoder is used.
   Send the following message: SDO access to object 2012$_h$ 32-bit value 000007D0$_h$.
10. **Cam offset.** Set cam offset to 6000 counts (0x1770).
    If the master resolution is 2000 counts/revolution, the slave shall start applying the cam when the master is at position 6000 + CamX value.
    Send the following message: SDO access to object 201B$_h$ 32-bit value 00001770$_h$.
11. **Cam input scaling factor.** Set it to 1.
    Send the following message: SDO access to object 206B$_h$ 32-bit value 00000001$_h$.
12. **Cam output scaling factor.** Set it to 1.
    Send the following message: SDO access to object 206C$_h$ 32-bit value 00000001$_h$.
13. Set the initial Master position into the associated RPDO where 201E$_h$ is mapped.
14. **Enable EG slave** in control word associated RPDO.
    Set in **Control Word** mapped in RPDO1 the value 3F$_h$ to start electronic gearing and activate speed limitation.
15. Start changing the Master position.

The slave motor should start rotating. After the master position of 6000 IU (cam offset), the slave motor will rotate depending on the set cam values.

12.4 Electronic camming through online communication example

Start an Electronic Gearing Slave.

1. Map in a RPDO the object $201E_h$ Master position to be able to send the drive the position reference every communication cycle. See *2.4 PDOs mapping general example* or paragraph *0* for a TwinCAT PDO mapping example.

   The PDOs must be sent every slow loop period which is by default 1ms. It is recommended to set the SYNC 0 time equal to the communication cycle and slow loop.

2. **Start remote node**.

   Enter **Pre-Operational** state.

   Enter **Safe-Operational** state.

   Enter **Operational** state.

3. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.

   Set in **Control Word** mapped in RPDO1 the value 06_h.

4. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.

   Set in **Control Word** mapped in RPDO1 the value 07_h.

5. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.

   Set in **Control Word** mapped in RPDO1 the value $0F_h$.

6. **External reference type.** Slave receives reference through online communication.

   Send the following message: SDO access to object $201D_h$ 16-bit value 0001_h.

7. **Cam table load address.** Set cam table load address as **5638**$_h$.

   The cam table load address can be discovered as explained in paragraph *12.2.6*.

   Send the following message: SDO access to object 2019_h 16-bit value 5638_h.

8. **Cam table run address.** Set cam table load address as **97F6**$_h$.

   The cam table load address can be discovered as explained in paragraph *12.2.6*.

   Send the following message: SDO access to object $201A_h$ 16-bit value $97F6_h$.

9. **Modes of operation.** Select Electronic Gamming mode (-2).

   Set in **Modes of Operation** mapped in RPDO1 the 8 bit value 0xFE.

10. **Master resolution.** Set the master resolution 2000, assuming a 500 line encoder is used.

    Send the following message: SDO access to object 2012_h 32-bit value $000007D0_h$.

11. **Cam offset.** Set cam offset to 6000 counts (0x1770).

    If the master resolution is 2000 counts/revolution, the slave shall start applying the cam when the master is at position 6000 + CamX value.

    Send the following message: SDO access to object $201B_h$ 32-bit value 00001770_h.

12. **Cam input scaling factor.** Set it to 1.

    Send the following message: SDO access to object $206B_h$ 32-bit value 00000001_h.

13. **Cam output scaling factor.** Set it to 1.

    Send the following message: SDO access to object $206C_h$ 32-bit value 00000001_h.

14. Set the initial Master position into the associated RPDO where $201E_h$ is mapped.

15. **Enable EG slave** in control word associated RPDO.

    Set in **Control Word** mapped in RPDO1 the value $3F_h$ to start electronic gearing and activate speed limitation.

16. Start changing the Master position in the RPDO where $201E_h$ is mapped every communication cycle.

The slave motor should start rotating. After the master reports the position 6000 IU (cam offset), the slave motor shall rotate depending on the set cam values..

13 Cyclic Synchronous Position mode (CSP)

13.1 Overview

The overall structure for this mode is shown in **Figure 13.1.1**. With this mode, the trajectory generator is located in the control device, not in the drive device. In cyclic synchronous manner, it provides a target position to the drive device, which performs position control, velocity control and torque control. Measured by sensors, the drive provides actual values for position, velocity and torque to the control device.

The cyclic synchronous position motion can be also limited to a maximum velocity by setting a number in object 6081h Profile velocity when object 2086h is set to 1. By default the object 2086h has the value 0.

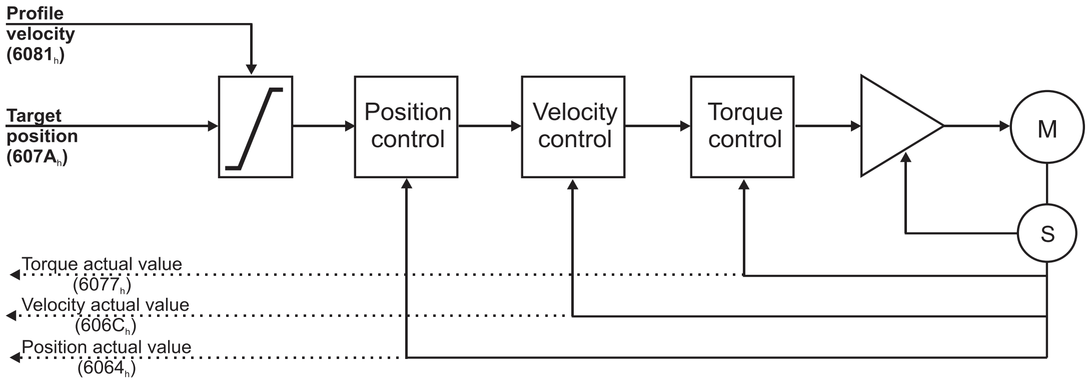


***Figure 13.1.1.** Cyclic synchronous position mode overview*

13.1.1 Controlword in Cyclic Synchronous Position mode (CSP)

MSB | | | | | | LSB

See 6040h	Halt	See 6040h	Abs / rel	Reserved	Reserved	See 6040h
15 9	8	7	6	5	4	3 0

***Table 13.1.1** – Controlword bits description for Cyclic Synchronous Position Mode*

Name	Value	Description
Abs / rel	0	Absolute position mode
	1	Relative position mode

In absolute position mode, the drive will always travel to the absolute position given to object 607Ah . This is the standard mode.

In Relative position mode, the drive will add to its current position the value received in object 607Ah. By sending this value periodically and setting the correct interpolation period time in object 60C2h, it will be like working in Cyclic Synchronous Velocity mode (CSV).

13.1.2 Statusword in Cyclic Synchronous Position mode (CSP)

MSB | | | | | | LSB

See 6041h		Following error	Target position ignored	See 6041h	Reserved	See 6041h	
15	14	13	12	11	10	9	0

***Table 13.1.2** – Statusword bit description for Cyclic Synchronous Position mode*

Name	Value	Description
Bit 10	0	Reserved
	1	Reserved
Target position ignored	0	Target position ignored
	1	Target position shall be used as input to position control loop
Following error	0	No following error
	1	Following error occurred

13.2 Cyclic Synchronous Position Mode Objects

13.2.1 Object 60C2h: Interpolation time period

The **Interpolation time period** indicates the configured interpolation cycle time. Its value must be set with the time value of the EtherCAT master communication cycle time and sync time in order for the Cyclic Synchronous Position mode to work properly. The interpolation time period (sub-index 01_h) value is given in $10^{(\text{interpolation time index})}$ s(second). The interpolation time index (sub-index 02_h) is dimensionless.

Example: to set a communication cycle time of 4ms, 60C2h sub-index 01_h = 4 and 60C2h sub-index 02_h = -3. The result is 4ms = $4 \cdot 10^{-3}$.

Because the drive default control loop is 1ms, it means that every new command (in CSP, CSV or CST) will be divided by 4. In other words, in each 1ms, 1/4 of the command will be executed.

Object description:

Index	60C2h
Name	Interpolation time period
Object code	ARRAY
Number of elements	2
Data Type	Interpolation time period record

Entry description:

Sub-index	00_h
Description	Number of sub-indexes
Access	RO
PDO mapping	No
Default value	2

Sub-index	01_h
Description	Interpolation time period value
Access	RW
PDO mapping	Possible
Value range	Unsigned8
Default value	1

Sub-index	02_h
Description	Interpolation time index
Access	RW
PDO mapping	Possible
Value range	INTEGER8, (-128 to +63)
Default value	-3

13.2.2 Object 2086h: Limit speed/acceleration for CSP/CSV[1]

This object is used to set a maximum velocity during CSP mode of operation.

Object description:

Index	2086h
Name	Limit speed/acceleration for CSP
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	Yes
Value range	UNSIGNED16
Default value	0000h

If 2086 h = 1, the limit is active. During CSP mode, the maximum velocity will be the one defined in object 6081h. During CSV mode, the maximum acceleration will be the one defined in object 6083h.

Remark: If 6081h = 0 and 2086h =1, during CSP mode, the motor will not move when it receives new position commands because its maximum velocity is limited to 0. The same scenario applies to the CSV mode.

[1] Available only with F515x firmware.

13.3 Cyclic Synchronous Position Mode basic example

1. **Start remote node**.

   Enter **Pre-Operational** state.

   Enter **Safe-Operational** state.

   Enter **Operational** state.

2. **Set the interpolation time object to the value of the communication cycle.**

   By default a new project in TwinCAT runs at 4ms.

   Set Object $60C2_h$ sub-index 1 to 4.

   Set object $60C2_h$ sub-index 2 to (-3).

   This means a value of $4^{(-3)}$ which means 4ms.

   Because the drive default control loop is 1ms, it means that when every new position command is received in object $607A_h$, it will be divided by 4. In other words, over the course of 4ms, the position command will be reached in a linear manner.

3. **Modes of operation.** Select cyclic synchronous position mode.

   Set in **Modes of Operation** mapped in RPDO1 the value 08_h.

4. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.

   Set in **Control Word** mapped in RPDO1 the value 06_h.

5. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.

   Set in **Control Word** mapped in RPDO1 the value 07_h.

6. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.

   Set in **Control Word** mapped in RPDO1 the value $0F_h$.

7. **Send position points.** The drive will execute a new motion with every new value it receives in RxPDO2 variable Target position which is object $607A_h$.

Set in **Target position** mapped in RPDO2, the 32bit value $xxxxxxxx_h$. The motor will travel to the new position value within 4ms (as set in $60C2_h$). If actual position =0 and the new target position is 40, then the motor will move 10 increments in 1 ms and translates to a speed of 10IU/ms.

13.4 Cyclic Synchronous Position Mode TwinCAT3 example

In TwinCAT, the NC-PTP interface actually uses by default the CSP mode. Read chapter *1.5.4* to run the TwinCAT 3 example.

14 Cyclic synchronous Velocity mode (CSV)

14.1 Overview

The overall structure for this mode is shown in *Figure 14.1.1*. With this mode, the trajectory generator is located in the control device, not in the drive device. In cyclic synchronous manner, it provides a target velocity to the drive device, which performs velocity control and torque control. Measured by sensors, the drive device provides actual values for position, velocity and torque to the control device.

The cyclic synchronous velocity motion is limited to a maximum acceleration by setting a number in object 6083$_h$ Profile acceleration.

The cyclic synchronous velocity mode covers the following sub-functions:

Demand value input

Velocity capture using position sensor or velocity sensor

Velocity control function with appropriate input and output signals

Limitation of torque demand

Remark: the speed control loop must be active in Easy Setup for this mode to function.

Various sensors may be used for velocity capture. In particular, the aim is that costs are reduced and the drive power system is simplified by evaluating position and velocity using a common sensor, such as is optional using a resolver or an encoder.

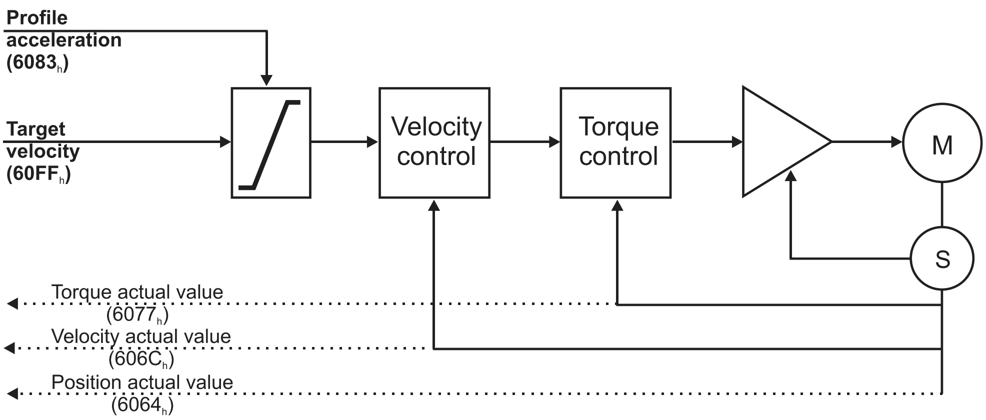


Figure 14.1.1. *Cyclic synchronous velocity mode overview*

14.1.1 Controlword in cyclic synchronous velocity mode

The cyclic synchronous velocity mode uses no mode specific bits of the Controlword. See *Object* 6040h: Controlword.

14.1.2 Statusword in cyclic synchronous velocity mode

MSB **LSB**

See 6041$_h$	Reserved	Target velocity ignored	See 6041$_h$	Reserved	See 6041$_h$
15 14	13	12	11	10	9 0

Table 14.1.1 *– Statusword bit description for cyclic synchronous velocity mode*

Name	Value	Description
Bit10	0	Reserved
	1	Reserved
Target velocity ignored	0	Target velocity ignored. When 6040$_h$.8 Halt is set to 1.
	1	Target velocity shall be used as input to velocity loop control
Bit13	0	Reserved
	1	Reserved

14.2 Cyclic Synchronous Velocity Mode basic example

1. **Start remote node**.

   Enter **Pre-Operational** state.

2. **Disable the Sync Manager 1C12$_h$ by setting Subindex 0 to 0.**

   Send the following message: SDO access to object 1C12$_h$ 8-bit value 00$_h$.

3. **Disable the PRDO4 1603$_h$ by setting Subindex 0 to 0.**

   Send the following message: SDO access to object 1603$_h$, sub-index 0 the 8-bit value 00$_h$.

4. **Map to PRDO4 1603$_h$ the object 60FF$_h$ Target velocity.**

   Send the following message: SDO access to object 1603$_h$, sub-index 1 the 32-bit value 60FF0020$_h$.

5. **Enable the PRDO4 1603$_h$ by setting Subindex 0 to 1.**

   Send the following message: SDO access to object 1603$_h$, sub-index 0 the 8-bit value 01$_h$.

6. **Map RPDO4 to Sync Manager 1C12$_h$ Subindex 3.** The RPDO4 has now mapped the object 60FF$_h$ Target velocity.

   Send the following message: SDO access to object 1C12$_h$, Subindex 3, 16-bit value 1603$_h$.

7. **Enable the Sync Manager 1C12$_h$ by setting Subindex 0 to 3.**

   Send the following message: SDO access to object 1C12$_h$ 8-bit value 03$_h$.

8. **Enter Operational state.**

9. **Set the interpolation time object to the value of the communication cycle.**

   By default a new project in TwinCAT runs at 4ms.

   Set Object 60C2$_h$ sub-index 1 to 4.

   Set object 60C2$_h$ sub-index 2 to (-3).

   This means a value of $4^{(-3)}$ which means 4ms.

   Because the drive default control loop is 1ms, it means that when every new position command is received in object 607A$_h$, it will be divided by 4. In other words, over the course of 4ms, the position command will be reached in a linear manner.

10. **Modes of operation.** Select cyclic synchronous velocity mode.

    Set in **Modes of Operation** mapped in RPDO1 the value 09$_h$.

11. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.

    Set in **Control Word** mapped in RPDO1 the value 06$_h$.

12. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.

    Set in **Control Word** mapped in RPDO1 the value 07$_h$.

13. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.

    Set in **Control Word** mapped in RPDO1 the value 0F$_h$.

14. **Send velocity points.** The drive will change its velocity with every new value it receives in RxPDO4 variable Target velocity which is object 60FF$_h$.

    Set in **Target velocity** mapped in RPDO4, the 32bit value xxxxxxxx$_h$.

Remark: By default, without the Factor Group set, the Target velocity structure is 16.16. Meaning the integer part of the speed in IU is set in the MSB and the fractional is set in the LSB.

15 Cyclic synchronous Torque mode (CST)

15.1 Overview

The overall structure for this mode is shown in Figure 14.1. With this mode, the trajectory generator is located in the control device, not in the drive device. In cyclic synchronous manner, it provides a target torque to the drive device, which performs torque control.

Measured by sensors, the drive device provides actual values for position, velocity and torque to the control device.

The cyclic synchronous torque mode covers the following sub-functions:

- demand value input;
- torque capture;
- torque control function with appropriate input and output signals;

limitation of torque demand.

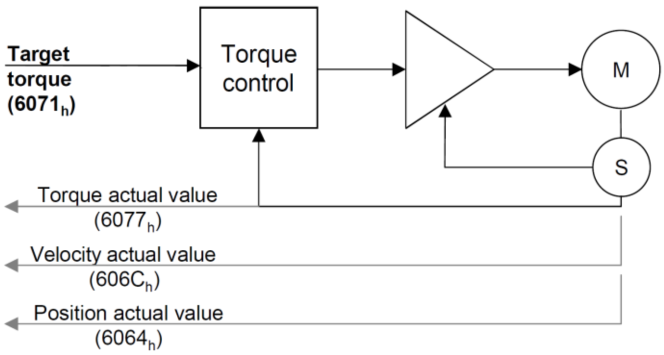


Figure 15.1.1. *Cyclic synchronous torque mode overview*

15.1.1 Controlword in cyclic synchronous torque mode

The cyclic synchronous torque mode uses no mode specific bits of the Controlword. See *Object 6040h: Controlword*.

15.1.2 Statusword in cyclic synchronous torque mode

MSB LSB

See 6041_h		Reserved	Target torque ignored	See 6041_h	Reserved	See 6041_h	
15	14	13	12	11	10	9	0

Table 15.1.1 *– Statusword bit description for Cyclic Synchronous Torque Mode*

Name	Value	Description
Bit10	0	Reserved
	1	Reserved
Target torque ignored	0	Target torque ignored
	1	Target torque shall be used as input to torque control loop
Bit13	0	Reserved
	1	Reserved

15.2 Cyclic synchronous torque mode objects

15.2.1 Object 6071_h: Target torque

This parameter specifies the input value configured for the torque controller when operating in Torque Profile mode. The unit for this object is given in IU, except for FA00x and FA02x firmware versions, where Object 2115_h: ASR4 bit 0 controls the unit in which the object is given:

- If ASR4.0 = 0, the unit for this object is given in IU
- If ASR4.0 = 1, the unit is in thousandths (‰) of the motor's rated current specified in object 6075_h. Example:

- If the target torque is set to 500, it represents 50.0% (500 ‰) of the motor's rated current.
- If the target torque is set to 255, it represents 25.5% (255 ‰) of the motor's rated current.

Remarks:

1. When object 2115_h is set to 1, the target torque can exceed 100% (equivalent to 1000 ‰) of the motor's rated current, as defined by object 6075_h.
2. The current limit is set through Object $207F_h$: Current limit. This value acts as a safety threshold and will restrict the maximum current, regardless of the value specified in object 6071_h.

Object description:

Index	6071_h
Name	Target torque
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	Yes
Value range	INTEGER16
Default value	0000_h

The computation formula for the current [IU] in [A] is:

$$current[IU] = \frac{65520 \cdot current[A]}{2 \cdot Ipeak}$$

where I_{peak} is the peak current supported by the drive and *current[IU]* is the command value for object 6071_h.

15.2.2 Object 6077_h: Torque actual value

This parameter provides the actual value of the torque, reflecting the instantaneous torque in the motor. The unit for this value is in Internal Units (IU), except for FA00x and FA02x firmware versions. In those versions, the unit is determined by the bit 0 of object 2115_h: ASR4:

- If ASR4.0 = 0, the unit is displayed in IU.
- If ASR4.0 = 1, the unit is displayed in thousandths (‰) of the motor's rated current specified in object 6075_h. Example:
  - If the actual torque value is 500, it represents 50.0% (500 ‰) of the motor's rated current.
  - If the actual torque value is 255, it represents 25.5% (255 ‰) of the motor's rated current.

Object description:

Index	6077_h
Name	Torque actual value
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	Yes
Value range	INTEGER16
Default value	No

The computation formula for the current [IU] in [A] is:

$$current[A] = \frac{2 \cdot Ipeak}{65520} \cdot current[IU]$$

where I_{peak} is the peak current supported by the drive and *current[IU]* is the read value from object 6077_h.

15.2.3 Object 6080_h: Max motor speed[1]

This object indicate the configured maximal allowed speed of the motor, taken from the motor specifications, when the mode of operation is CST or External Torque value. The value is given is given in user-defined velocity units. User-defined means it can be modified by Factor group objects. The speed limitation is activated when setting a value different from zero (default).

Object description:

Index	6080_h
Name	Max motor speed
Object code	VAR
Data type	UNSIGNED32

[1] Available starting with firmware version FA00x / FA02x / F515K.

Entry description:

Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0

15.2.4 Object 2115h: ASR4[1]

This object is responsible for setting up the ASR4 register, with Bit 0 of ASR4 determining the formatting and representation of values in Object 6071h: Target torque and 6077h: Torque actual value. The remaining bits in ASR4 are reserved.

Object description:

Index	2115h
Name	ASR4
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	no
Value range	UNSIGNED32
Default value	0

15.3 Cyclic synchronous torque (CST) example

1. **Start remote node**.
   Enter **Pre-Operational** state.
2. **Disable the Sync Manager 1C12h Subindex 0 to 0.**
   Send the following message: SDO access to object 1C12h 8-bit value 00h.
3. **Map RPDO3 to Sync Manager 1C12h Subindex 3.** The RPDO3 has mapped by default, the object 6071h Target torque.
   Send the following message: SDO access to object 1C12h, Subindex 3, 16-bit value 1602h.
4. **Enable the Sync Manager 1C12h by setting Subindex 0 to 3.**
   Send the following message: SDO access to object 1C12h 8-bit value 03h.
5. **Enter Operational state.**
   Enter **Safe-Operational** state.
   Enter **Operational** state.
6. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.
   Set in **Control Word** mapped in RPDO1 the value 06h.
7. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.
   Set in **Control Word** mapped in RPDO1 the value 07h.
8. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.
   Set in **Control Word** mapped in RPDO1 the value 0Fh.
9. **Modes of operation.** Select cyclic synchronous torque mode.
   Set in **Modes of Operation** mapped in RPDO1 the value 0Ah.
10. **Send target torque points.** The drive will apply a new current value with every new command it receives in RxPDO3 variable Target torque which is object 6071h.

Set in **Target torque** mapped in RPDO3, the 16bit value xxxxh.

[1] Available starting with firmware version FA00G / FA02G or newer

16 Touch probe functionality

16.1 Overview

The Touch probe functionality offers the possibility to capture the motor current position when a configurable digital input trigger event happens.

Remark: do not use the touch probe functionality objects during a homing procedure. It may lead to incorrect results.

16.2 Touch probe objects

16.2.1 Object 60B8$_h$: Touch probe function

This object indicates the configuration function of the touch probe.

Object description:

Index	60B8$_h$
Name	Touch probe function
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Yes
Value range	0 … 65535
Default value	0

Table 16.2.1 *– Bit Assignment of the Touch probe function*

Bit	Value	Description
14,15	-	Reserved
13	0	Switch off sampling at negative edge of touch probe 2
	1	Enable sampling at negative edge of touch probe 2*
12	0	Switch off sampling at positive edge of touch probe 2
	1	Enable sampling at positive edge of touch probe 2*
11,10	00$_b$	Trigger with touch probe 2 input (LSN input)
	01$_b$	Trigger with zero impulse signal
	10$_b$	Reserved
	11$_b$	Reserved
9	0	Trigger first event
	1	Reserved
8	0	Switch off touch probe 2
	1	Enable touch probe 2
7	-	Reserved
6	0	Enable limit switch functionality. The motor will stop, using quickstop deceleration, when a limit switch is active.
	1	Disable limit switch functionality. The motor will not stop when a limit switch is active.
5	0	Switch off sampling at negative edge of touch probe 1
	1	Enable sampling at negative edge of touch probe 1*
4	0	Switch off sampling at positive edge of touch probe 1
	1	Enable sampling at positive edge of touch probe 1*
3,2	00$_b$	Trigger with touch probe 1 input (LSP input)
	01$_b$	Trigger with zero impulse signal
	10$_b$	Reserved
	11$_b$	Reserved
1	0	Trigger first event
	1	Reserved
0	0	Switch off touch probe 1
	1	Enable touch probe 1

***Remarks**:

The position cannot be captured on both positive and negative edges simultaneously using the zero impulse signal as a trigger.

The position cannot be captured when touch probe 1 and 2 are active and the trigger is set on the zero impulse signal.

The following bit settings are reserved:

-Bit 3 and Bit2 = 1;

-Bit 13 and Bit12 = 1;

-Bit11 and Bit2 = 1;

The homing procedures also utilize the capture function. Using this object during a homing procedure may lead to unforeseen results.

16.2.2 Object 60B9$_h$: Touch probe status

This object provides the status of the touch probe.

Object description:

Index	60B9$_h$
Name	Touch probe status
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Yes
Value range	0 … 65535
Default value	0

***Table 16.2.2** – Bit Assignment of the Touch probe status*

Bit	Value	Description
11 to 15	-	Reserved
10	0	Touch probe 2 no negative edge value stored
	1	Touch probe 2 negative edge position stored in object 60BD$_h$
9	0	Touch probe 2 no positive edge value stored
	1	Touch probe 2 positive edge position stored in object 60BC$_h$
8	0	Touch probe 2 is switched off
	1	Touch probe 2 is enabled
7	-	Reserved
6	0	Limit switch functionality enabled.
	1	Limit switch functionality disabled.
3 to 5	-	Reserved
2	0	Touch probe 1 no negative edge value stored
	1	Touch probe 1 negative edge position stored in object 60BB$_h$
1	0	Touch probe 1 no positive edge value stored
	1	Touch probe 1 positive edge position stored in object 60BA$_h$
0	0	Touch probe 1 is switched off
	1	Touch probe 1 is enabled

Note: Bit 1 and bit 2 are set to 0 when touch probe 1 is switched off (object 60B8$_h$ bit 0 is 0). Bit 9 and 10 are set to 0 when touch probe 2 is switched off (object 60B8$_h$ bit 8 is 0). Bits 1,2,9 and 10 are set to 0 when object 60B8$_h$ bits 4,5,12 and 13 are set to 0.

16.2.3 Object 60BA$_h$: Touch probe 1 positive edge

This object provides the position value of the touch probe 1 at positive edge.

Object description:

Index	60BA$_h$
Name	Touch probe 1 positive edge
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	YES
Value range	$-2^{31} \ldots 2^{31}-1$
Default value	-

16.2.4 Object 60BB$_h$: Touch probe 1 negative edge

This object provides the position value of the touch probe 1 at negative edge.

Object description:

Index	60BBh
Name	Touch probe 1 negative edge
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	YES
Value range	$-2^{31}...2^{31}-1$
Default value	-

16.2.5 Object 60BCh: Touch probe 2 positive edge

This object provides the position value of the touch probe 2 at positive edge.

Object description:

Index	60BCh
Name	Touch probe 2 positive edge
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	YES
Value range	$-2^{31}...2^{31}-1$
Default value	-

16.2.6 Object 60BDh: Touch probe 2 negative edge

This object provides the position value of the touch probe 2 at negative edge.

Object description:

Index	60BDh
Name	Touch probe 2 negative edge
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	YES
Value range	$-2^{31}...2^{31}-1$
Default value	-

16.2.7 Object 2104h[1]: Auxiliary encoder function

This object configures the auxiliary feedback position capture on the zero impulse signal.

Object description:

Index	2104h
Name	Auxiliary encoder function
Object code	VAR
Data type	UNSIGNED8

Entry description:

Access	RW
PDO mapping	Yes
Value range	0 … 255
Default value	0

[1] Object 2104h applies only to drives which have a secondary feedback input with an index signal

Table 16.2.3 – Bit Assignment of the Auxiliary encoder function

Bit	Value	Description
8..6	-	Reserved
5	0	Switch off sampling at negative edge of touch probe
	1*	Enable sampling at negative edge of touch probe
4	0	Switch off sampling at positive edge of touch probe
	1*	Enable sampling at positive edge of touch probe
3	-	Reserved
2	0	Reserved
	1	Trigger with zero impulse signal
1	-	Reserved
0	0	Switch off touch probe
	1	Enable touch probe

***Remark**

The position cannot be captured on both positive and negative edges simultaneously using the zero impulse signal as a trigger.

16.2.8 Object 2105h[1]: Auxiliary encoder status

This object provides the status of the auxiliary feedback touch probe.

Object description:

Index	2105h
Name	Auxiliary encoder status
Object code	VAR
Data type	UNSIGNED8

Entry description:

Access	RO
PDO mapping	Yes
Value range	0 … 255
Default value	0

Table 16.2.4 – Bit Assignment of the Auxiliary encoder status

Bit	Value	Description
8 to 3	-	Reserved
2	0	Auxiliary feedback touch probe no negative edge value stored
	1	Auxiliary feedback touch probe negative edge position stored in object 2107h
1	0	Auxiliary feedback touch probe no positive edge value stored
	1	Auxiliary feedback touch probe positive edge position stored in object 2106h
0	0	Auxiliary feedback touch probe is switched off
	1	Auxiliary feedback touch probe is enabled

Note: Bit 1 and bit 2 are set to 0 when auxiliary feedback touch probe is switched off (object 2104h bit 0 is 0). Bits 1 and 2 are set to 0 when object 2104h bits 4 and 5 are set to 0.

16.2.9 Object 2106h[2]: Auxiliary encoder captured position positive edge

This object provides the position value of the auxiliary feedback captured at positive edge.

Object description:

Index	2106h
Name	Auxiliary encoder captured positive edge
Object code	VAR
Data type	INTEGER32

Entry description:

[1] Object 2105h applies only to drives which have a secondary feedback input with an index signal

[2] Object 2106h applies only to drives which have a secondary feedback input with an index signal

Access	RO
PDO mapping	YES
Value range	$-2^{31}...2^{31}-1$
Default value	-

16.2.10 Object 2107h[1]: Auxiliary encoder captured position negative edge

This object provides the position value of the auxiliary feedback captured at negative edge.

Object description:

Index	2107h
Name	Auxiliary encoder captured position negative edge
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	YES
Value range	$-2^{31}...2^{31}-1$
Default value	-

16.3 Touch probe example

In this example, the touch probe 1 will be enabled to capture the position when the positive limit switch LSP is triggered on the positive edge while moving the motor in trapezoidal mode.

1. **Start remote node**.
   Enter **Pre-Operational** state.

   Enter **Safe-Operational** state.

   Enter **Operational** state.
2. **Modes of operation.** Select position mode.
   Set in **Modes of Operation** mapped in RPDO1 the value 01h.
3. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.
   Set in **Control Word** mapped in RPDO1 the value 06h.
4. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.
   Set in **Control Word** mapped in RPDO1 the value 07h.
5. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.
   Set in **Control Word** mapped in RPDO1 the value 0Fh.
6. **Target position.** Set the target position to 4 rotations. By using a 500 lines incremental encoder the corresponding value of object 607Ah expressed in encoder counts is 1F40h.
   Set in **Target position** mapped in RPDO2 the value 00001F40h.
7. **Target speed.** Set the target speed normally attained at the end of acceleration ramp to 2IU/ms (low speed).
   Send the following message: SDO access to object 6081h, 32-bit value 00020000h.
8. **Set touch probe function to 0x11.** Set touch probe function to enable touch probe 1, touch probe 1 to be the positive limit switch LSP, capture the position on the positive edge of the signal (when LSP goes low to high).
   Send the following message: SDO access to object 60B8h, 16-bit value 0011h.
9. **Read touch probe status.** Read touch probe status.
   Send the following message: SDO read access to object 60B9h.

   If the read value is 0x0001 it means that touch probe 1 is active (bit0=1) and a capture was detected on the positive edge (bit1=1).
10. **While the motor is moving, trigger the LSP input. The motor should stop.**
11. **Read touch probe status.** Read touch probe status.
    Send the following message: SDO read access to object 60B9h.

    If the read value is 0x0003 it means that touch probe 1 is active (bit0=1) and no capture was detected on the positive edge (bit1=0).
12. **Read the touch probe 1 positive edge captured value.**
    Send the following message: SDO read access to object 60BAh.

    If the read value should be close to the value of motor actual position (6064h). When the capture was detected, the motor was moving. The limit switch caused the motor to decelerate and stop after the even occurred.

[1] Object 2107h applies only to drives which have a secondary feedback input with an index signal

17 Data Exchange between EtherCAT® master and drives

17.1 Checking Setup Data Consistency

During the configuration phase, a EtherCAT® master can quickly verify using the checksum objects and a reference **.sw** file whether the non-volatile EEPROM memory of the IDM680 drive contains the right information. If the checksum reported by the drive doesn't match the one computed from the **.sw** file, the EtherCAT® master can download the entire **.sw** file into the drive EEPROM using the communication objects for writing data into the drive EEPROM.

In order to be able to inspect or to program any memory location of the drive, as well as for downloading of a new TML program (application software), three manufacturer specific objects were defined: Object 2064_h – Read/Write Configuration Register, 2065_h – Write Data at address specified in 2064_h, 2066_h – Read Data from address specified in 2064_h, 2067_h – Write data at specified address.

17.2 Data Exchange Objects

17.2.1 Object 2064_h: Read/Write Configuration Register

Object Read/Write Configuration Register 2064_h is used to control the read from drive memory and write to drive memory functions. This object contains the current memory address that will be used for a read/write operation. It can also be specified through this object the type of memory used (EEPROM, data or program) and the data type the next read/write operation refers to. Additionally, it can be specified whether an increment of the memory address should be performed or not after the read or write operation. The auto-increment of the memory address is particularly important in saving valuable time in case of a program download to the drive as well when a large data block should be read from the device.

Object description:

Index	2064_h
Name	Read/Write configuration register
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Possible
Units	-
Value range	0 … 2^{32}-1
Default value	0x84

Table 17.2.1 *– Read/Write Configuration Register bit description*

Bit	Value	Description
31…16	x	16-bit memory address for the next read/write operation
15…8	0	Reserved (always 0)
7	0	Auto-increment the address after the read/write operation
	1	Do not auto-increment the address after the read/write operation
6…4	0	Reserved (always 0)
3,2	00	Memory type is program memory
	01	Memory type is data memory
	10	Memory type is EEPROM memory
	11	Reserved
1	0	Reserved (always 0)
0	0	Next read/write operation is with a 16-bit data
	1	Next read/write operation is with a 32-bit data

Warning! When object 2064_h bit 7=0 (auto-incrementing is ON), do not read the object list in parallel with a read/write operation using a script. By reading object 2066_h in parallel with another application, the target memory address will be incremented and will lead to incorrect data writing or reading.

17.2.2 Object 2065_h: Write 16/32 bits data at address set in Read/Write Configuration Register

The object is used to write 16 or 32-bit values using the parameters specified in object 2064_h – Read/Write Configuration Register. After the successful write operation, the memory address in object 2064_h, bits 31…16 will be auto-incremented or not, as defined in the same register. The auto-incrementing of the address is particularly useful in downloading a program (software application) in the drives memory.

Object description:

Index	2065h
Name	Write data at address set in 2064h (16/32 bits)
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	WO
PDO mapping	Possible
Units	-
Value range	0 … 2^{32}-1
Default value	No

The structure of the parameter is the following:

Bit	Value	Description
31…16	0	Reserved if bit 0 of object 2064h is 0 (operation on 16 bit variables)
	X	16-bit MSB of data if bit 0 of object 2064h is 1 (operation on 32 bit variables)
15…0	X	16 bit LSB of data

17.2.3 Object 2066h: Read 16/32 bits data from address set in Read/Write Configuration Register

This object is used to read 16 or 32-bit values with parameters that are specified in object 2064h – Read/Write Configuration Register. After the successful read operation, the memory address in object 2064h, bits 31…16, will be auto-incremented or not, as defined in the same register.

Object description:

Index	2066h
Name	Read data from address set in 2064h (16/32 bits)
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RO
PDO mapping	No
Units	-
Value range	UNSIGNED32
Default value	No

The structure of the parameter is the following:

Bit	Value	Description
31…16	0	Reserved if bit 0 of object 2064h is 0 (operation on 16 bit variables)
	X	16-bit MSB of data if bit 0 of object 2064h is 1 (operation on 32 bit variables)
15…0	X	16 bit LSB of data

17.2.4 Object 2067h: Write 16bit data at specified address

This object is used to write a single 16-bit value at a specified address in the memory type defined in object 2064h – Read/Write Configuration Register. The rest of the bits in object 2064h do not count in this case, e.g. the memory address stored in the Read/Write Control Register is disregarded and also the control bits 0 and 7. The object may be used to write only 16-bit data. Once the type of memory in the Read/Write Control Register is set, the object can be used independently. If mapped on a PDO, it offers quick access to any drive internal variable.

Object description:

Index	2067h
Name	Write data at specified address
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	WO
PDO mapping	Possible
Units	-
Value range	UNSIGNED32
Default value	No

Bit	Value	Description
31…16	x	16-bit memory address
15…0	X	16 bit data value to be written

17.2.4.1 Writing 16 bit data to a specific address using object 2067h example

Considering the following variable found in variables.cfg in the /Firmwares/F515I folder:

UINT POSOKLIM @0x036A. It means that it is found at address 0x036A.

Write the data **0x1234** to address **0x036A** using SDO access to object 2067h:

SDO access to object 2067h 32-bit value **036A1234**h.

17.2.5 Object 2069h: Checksum configuration register

This object is used to specify a start address and an end address for the drive to execute a checksum of the E2ROM memory contents. The 16 LSB of this object are used for the start address of the checksum, and the 16 MSB for the end address of the checksum.

Note: The end address of the checksum must be computed as the start address to which you add the length of the section to be checked. The drive will actually compute the checksum for the memory locations between start address and end address.

The checksum is computed as a 16 bit unsigned addition of the values in the memory locations to be checked. When the object is written through SDO access, the checksum will be computed and stored in the read-only object 206Ah.

Object description:

Index	2069h
Name	Checksum configuration register
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	No
Units	-
Value range	UNSIGNED32
Default value	No

The structure of the parameter is the following:

Bit	Value	Description
31…16	X	16-bit end address of the checksum
15…0	X	16 bit start address of the checksum

17.2.6 Object 206Ah: Checksum read register

This object stores the latest computed checksum.

Object description:

Index	206Ah
Name	Checksum read register
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	No
Units	-
Value range	UNSIGNED16
Default value	No

17.2.7 Object 210Ch: enable SW file download

This object allows writing a SW file using FoE protocol while in BOOTSTRAP state.

TwinCAT3 has the function “fbDownload” which can be used for FoE transfer protocol. When this function is called, the drive is reset into BOOTSTRAP state.

This object must be set to 0 to use the function to download a firmware file using FoE protocol.

This object must be set to 1 to use the function to download a SW setup file using FoE protocol.

Object description:

Index	210Ch
Name	Enable SW file download
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	Yes
Units	-
Value range	INTEGER16
Default value	No

17.3 Image Files Format and Creation

An EEPROM image file (with extension .sw) is a text file that can be read with any text editor. It contains blocks of data separated by an empty line. Each block of data starts with the block start address, followed by data values to place in ascending order at consecutive addresses: first data – to write at start address, second data – to write at start address + 1, etc. All the data are hexadecimal 16- bit values (maximum 4 hexadecimal digits). Each line contains a single data value. When less than 4 hexadecimal digits are shown, the value must be right justified. For example 92 represents 0x0092.

The .sw software files can be generated either from EasySetUp or from EasyMotion Studio.

In EasySetUp, a .sw file is created with the command Setup | Create EEPROM Programmer File… The software file generated, includes the setup data and the drive/motor configuration ID with the user programmable application ID.

In EasyMotion Studio, a .sw file is created with one of the commands: Application | EEPROM Programmer File | Motion and Setup or Setup Only. The option Motion and Setup creates a .sw file with complete information including setup data, TML programs, cam tables (if present) and the drive/motor configuration ID. The option Setup Only produces a .sw file identical with that produced by EasySetUp i.e. having only the setup data and the configuration ID.

The .sw file can be programmed into a drive:

from a EtherCAT® master, using the communication objects for writing data into the drive EEPROM

using the EEPROM Programmer tool, which comes with EasySetUp but may also be installed separately. The EEPROM Programmer was specifically designed for repetitive fast and easy programming of .sw files into the Technosoft drives during production.

17.4 Downloading an image file (.sw) to the drive using CoE objects example

The structure of an image file (.sw) is described in paragraph **17.3** and shown in **Figure 17.4.1**.

In order to download the data block pointed by the red arrow, first the block start address i.e. 5638_h must be set using an SDO access to object 2064_h.

- Send the following message: SDO access to object 2064_h, 32-bit value **5638**0008_h.

The above configuration command also indicates that next read or write operation shall be executed with drive's EEPROM memory using 16-bit data and auto increment of address. All the numbers from the lines after 5638_h until the following blank line represents data to write in the EEPROM memory at consecutive addresses starting with 5638_h. The data writes are done using an SDO access to object 2065_h. First data word $C400_h$ is written using:

- Send the following message: SDO access to object 2065_h, 32-bit value 0000**C400**$_h$.

From the whole 32bit number, only **C400**$_h$ will be written and 0000_h will be ignored because the write operation was configured for 16bits in object 2065 $_h$.

Next data word 0000_h is written with:

Send the following message: SDO access to object 2065_h, 32-bit value 0000**0000**$_h$.

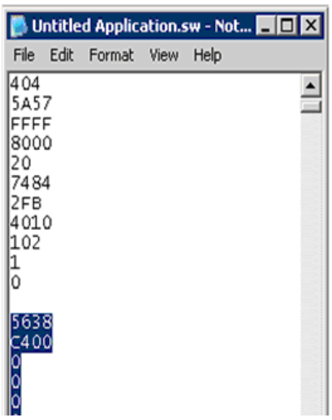


Figure 17.4.1. *.sw file structure example*

Continue sending the 16 bit data, until the next blank line from the .sw file. Because the next data after a blank line is again an address, and the above process repeats. Finally to verify the integrity of the information stored in the drive EEPROM, checksum objects 2069_h and $206A_h$ can be used to compare the checksum computed by the drive with that computed on the master.

Warning!	When object 2064_h bit 7=0 (auto-incrementing is ON), do not read the object list in parallel with a read/write operation using a script. By reading object 2066_h in parallel with another application, the target memory address will be incremented and will lead to incorrect data writing or reading.

17.4.1 Checking and loading the drive setup via .sw file and CoE commands example.

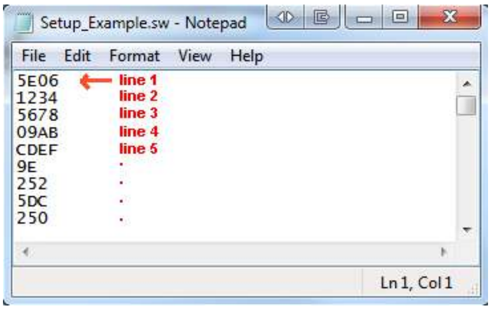
Check the integrity of the setup data on a drive and update it if needed.

Before reading this example, please read **paragraph 17.4.**

To create a .sw file containing only the setup data do the following:

- In Easy Motion Studio, go to Application (in the menu bar at the top)-> Create EEPROM Programmer File -> Setup Only… . Choose where to save the .sw file.
- In EasySetup, Setup (in the menu bar at the top) -> Create EEPROM Programmer File… . Choose where to save the .sw file.

Let's suppose that the setup data of a Technosoft drive is located at EEPROM addresses between 0x**5E06** and 0x**5EFF**. Here are the steps to be taken in order to check the setup data integrity and to re-program the drive if necessary:



1. **Compute the checksum in the .sw file**. Let's suppose that the computed checksum is 0x1234.
2. **Access object 2066_h in order to compute the checksum of the setup table located on the drive.** Write the value 0x**5EFF5E06**

   Send the following message: SDO write to object 2066_h sub-index 0, 32-bit value $5EFF5E06_h$.

   Following the reception of this message, the drive will compute the checksum of the EEPROM locations 0x**5E06** to 0x**5EFF**. The result is stored in the object $206A_h$.
3. **Read the computed checksum from object $206A_h$**.

   Read by SDO protocol the value of object $206A_h$.

   Let us assume the drive returns the following message (Object $206A_h$ = 0x2345):

   As the returned checksum (0x2345) does not match the checksum computed from the .sw file, the setup table has to be configured from the .sw file.
4. **Prepare the Read/Write Configuration Register for EEPROM write**. Let us assume the address 0x**5E06** is the first 16 bit number found in the .sw file where setup data begins**.** Write the value 0x**5E06**0009 into the object 2064_h (write 32-bit data at EEPROM address 0x**5E06** and auto-increment the address after the write operation).

   Send the following message: SDO write to object 2064_h sub-index 0, 32-bit value $5E060009_h$.
5. **Write the sw file data 32 bits at a time**. Supposing that the next 2 entries in the .sw file after the start address 0x**5E06** are 0x**1234** and 0x**5678**, you have to write the value 0x**56781234** into object 2065_h.

   Send the following message (SDO write to object 2065_h sub-index 0, 32-bit value 56781234_h):

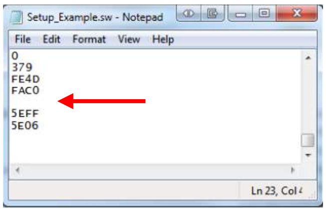
   The number 0x**1234** will be written at address 0x**5E06** and 0x**5678** will be at 0x**5E07**.
6. Assuming the next data after 0x**5678** will be 0x**09AB** and 0x**CDEF**, write the value 0x**CDEF09AB** into object 2065_h.

   Send the following message (SDO write to object 2065_h sub-index 0, 32-bit value $CDEF09AB_h$):

   The number 0x**09AB** will be written at address 0x**5E08** and 0x**CDEF** will be at 0x**5E09**.

7. Repeat step 5 until a blank line is found in the .sw file.

This means that all the setup data is written, even if there is more data after the blank line.



8. Re-check the checksum (repeat steps 2 and 3). If ok, go to step 9

9. Reset the drive in order to activate the new setup.

Send with the Cob ID 0x0 the data 0x81 0x0A. Where 0x0A means Axis ID 10.

Warning!	When object 2064$_h$ bit 7=0 (auto-incrementing is ON), do not read the object list in parallel with a read/write operation using a script. By reading object 2066$_h$ in parallel with another application, the target memory address will be incremented and will lead to incorrect data writing or reading.

17.4.2 SW file Checksum calculation C# example code

The code presented below is written in C# language and its structure can be used as an example for other programming languages.

The program itself works as standalone. Just create a new console script in Visual Studio C# 2005 or newer and copy it directly.

A script cam download a .sw file and at the same time calculate the checksum for each section in order to verify it later with object 2066$_h$ and 206A$_h$.

A SW file has up to 4 data sections. This script will Display the Start, End address and Checksum of each section. These three parameters can later be used with objects 2066$_h$ and 206A$_h$ to verify the checksum on the drive after the SW file is downloaded. Later, to verify the data integrity, at each drive start-up, the checksum can be verified to ensure the correct setup data is present on the drive.

17.4.2.1 SW file Checksum calculation C# example code

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
using System.Threading;
using System.Collections;
using System.Runtime;
using System.Diagnostics;

namespace THS_checksum_calculator
{
    static class Program
    {
        static void Main(string[] args)
        {
            String Path = "c:\\setup1.sw"; //define the SW file path
            CalculateSWfileChecksum(Path);
        }
        private static void CalculateSWfileChecksum(String Path)
        {
            System.Console.WriteLine("");
            System.Console.WriteLine ("Reading SW file from path : " + Path);
            System.Console.WriteLine ("");
```

```
            try
            {
                StreamReader sr = File.OpenText(Path);
                String strLine;
                bool setAddress = true; //because the first line in the SW is an
address, start with setAddress TRUE.
                UInt16 checksumSW = 0;
                UInt16 StartAddress = 0;
                UInt16 EndAddress = 0;
                Byte[] LineData;
                int swFileSection = 1;
                while (null != (strLine = sr.ReadLine()))
                {
                    if (strLine == "") //checks for blank spaces with no data
                    {
                        System.Console.WriteLine ("End address = 0x" +
EndAddress.ToString("X") + "; High 16bit of object 2066h"); //Display in HEX the
current section End address
                        System.Console.WriteLine ("Checksum = 0x" +
(checksumSW).ToString("X") + "; To be compared with object 206Ah value."); //Display in
HEX the current section Checksum value
                        System.Console.WriteLine ("");
                        checksumSW = 0;
                        setAddress = true;
                        continue;
                    }
                    if (setAddress)
                    {
                        LineData = BitConverter.GetBytes(Int16.Parse(strLine,
System.Globalization.NumberStyles.HexNumber, null));
                        StartAddress = BitConverter.ToUInt16(LineData, 0);
                        EndAddress = StartAddress;
                        EndAddress--;
                        System.Console.WriteLine ("SW file Section " + swFileSection +
" parameters:"); //Display the SW file section
                        System.Console.WriteLine ("Start address = 0x" +
StartAddress.ToString("X") + "; Low 16bit of object 2066h"); //Display in HEX the
current section Start address
                        swFileSection++; //increment the file section number
                        setAddress = false;
                        continue;
                    }
                    EndAddress++;
                    LineData = BitConverter.GetBytes(Int16.Parse(strLine,
System.Globalization.NumberStyles.HexNumber, null));
                    checksumSW += BitConverter.ToUInt16(LineData, 0) ;
                }
                System.Console.WriteLine ("Ended reading file " + Path );
                sr.Close();
                Thread.Sleep(5000); //Wait and display results in Debug window before
it closes
            }
            catch (FileNotFoundException e)
            {
                System.Console.WriteLine (e.Message);
            }
        }
    }
```

```
}
```

The output window of the program should look like this:

```
Reading SW file from path : c:\setup1.sw

SW file Section 1 parameters:
Start address = 0x4000; Low 16bit of object 2069h
End address = 0x4173; High 16bit of object 2069h
Checksum = 0xF0BC; To be compared with object 206Ah value.

SW file Section 2 parameters:
Start address = 0x7B7E; Low 16bit of object 2069h
End address = 0x7FAF; High 16bit of object 2069h
Checksum = 0x4168; To be compared with object 206Ah value.

SW file Section 3 parameters:
Start address = 0x7FBF; Low 16bit of object 2069h
End address = 0x7FE0; High 16bit of object 2069h
Checksum = 0xFFFF; To be compared with object 206Ah value.

SW file Section 4 parameters:
Start address = 0x7FFF; Low 16bit of object 2069h
End address = 0x7FFF; High 16bit of object 2069h
Checksum = 0x7B7E; To be compared with object 206Ah value.

Ended reading file c:\setup1.sw
```

17.4.3 FoE software files, creation and use

Only drives with firmware F515F or newer, support FoE (File over EtherCAT) protocol to transfer Setup data or Firmware update.

The **FoE** software files can be generated either from EasySetUp or from EasyMotion Studio.

In EasySetUp, a **FoE** file is created with the command **Setup | Create EtherCAT FoE File…** The software file generated, includes the setup data and the drive/motor configuration ID with the user programmable application ID (editable from Drive Setup/Drive info button).

In EasyMotion Studio, a **.sw** file is created with one of the commands: **Application | Create EtherCAT FoE File | Motion and Setup** or **Setup Only**. The option **Motion and Setup** creates a **FoE** file with complete information including setup data, TML programs, functions, customized homing routines and the drive/motor configuration ID. The option **Setup Only** produces a **.sw** file identical with that produced by EasySetup i.e. having only the setup data and the configuration ID.

A FoESW file can be saved with the extension .bin or .efw.

17.4.3.1 FoE files rules and information

- The FoE file must start with “FOESW_”.
- The entire FoE file name length must not exceed 14 characters. The extension is excluded.
- A Setup data file can be transferred via FoE protocol only in Op, Pre-OP and Safe-Op ECAT states. While in Bootstrap state, the file is rejected.
- The password to program a FoE setup data file is 0.

If any of the rules mentioned above is not fulfilled, the file will be rejected by the drive.

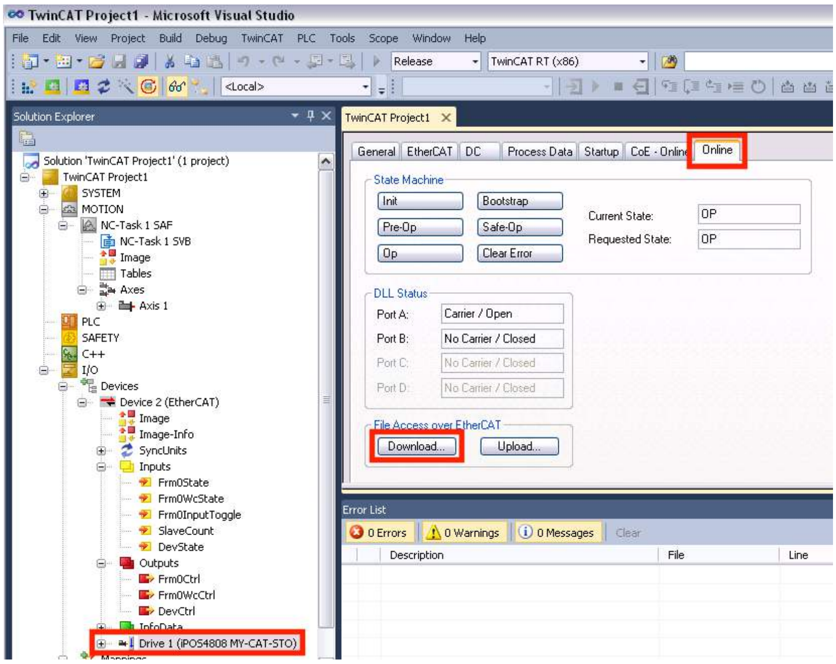
17.4.4 Writing a FoE (File over EtherCAT) Setup data file using TwinCAT 3 example

Only drives with firmware F515F or newer, support FoE (File over EtherCAT) protocol to transfer Setup data or Firmware update.

See Par. *17.3* about creating a FoE file first.

Open a TwinCAT 3 project that is communicating with a Technosoft EtherCAT drive.

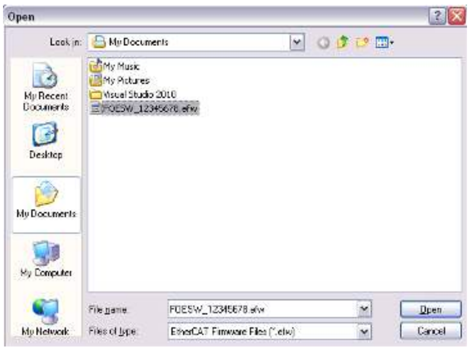
Click the drive name and select the Online tab:



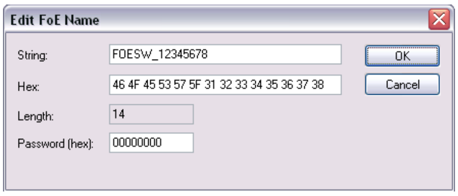
Under the File Access over EtherCAT, press the Download button.

Remark: *the file Access over EtherCAT area becomes available only if the drive is programmed with the latest XML information. The XML revision must be 892417350 (0x35313546 or 515F in ASCII) or later. See Par* 1.5.4.6 Checking and updating the XML file stored in the drive.

Choose the Drive Setup file and click Open.



When prompted, choose the password as 0 and click OK.



After clicking OK, a progress bar on the bottom right corner will confirm that the writing is complete.

Remark: *The TwinCAT function "FB_EcFoeLoad" will not work directly with loading FoE Setup data files because it automatically changes the state into Bootstrap. To transfer Setup data using FoE while in Bootstrap, first set object 210C*$_h$ *to 1.*

17.4.5 Writing a FoE (File over EtherCAT) Setup data file using TwinCAT 3 ST script example

Even if not used, please read first Par *17.4.4*.

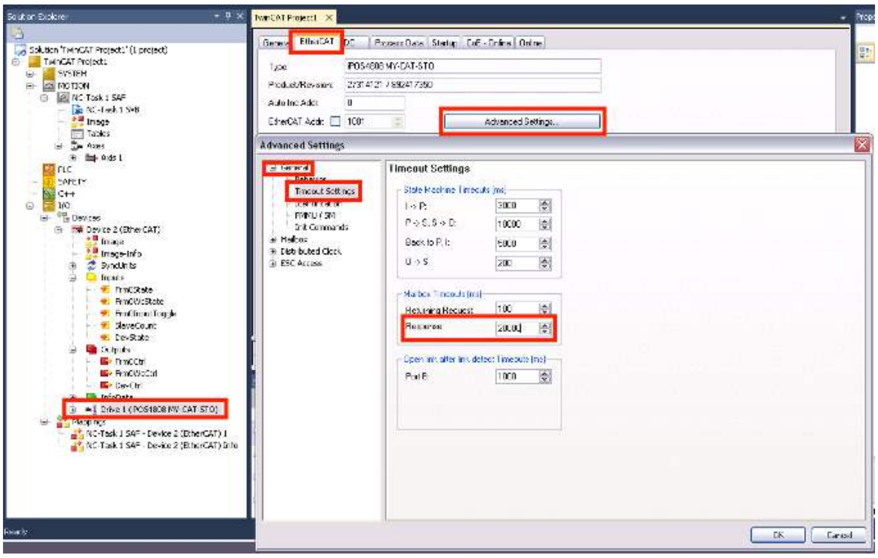
Only drives with firmware F515I or newer, support FoE (File over EtherCAT) protocol to transfer Setup data while in Bootstrap state when object 210Ch = 1.

See Par *0* about creating a FoE file first.

The Setup data can be updated via the TwinCAT “FB_EcFoeLoad” function which sets the drive into Boot mode automatically. Object 210Ch must be 1 while before starting this transfer.

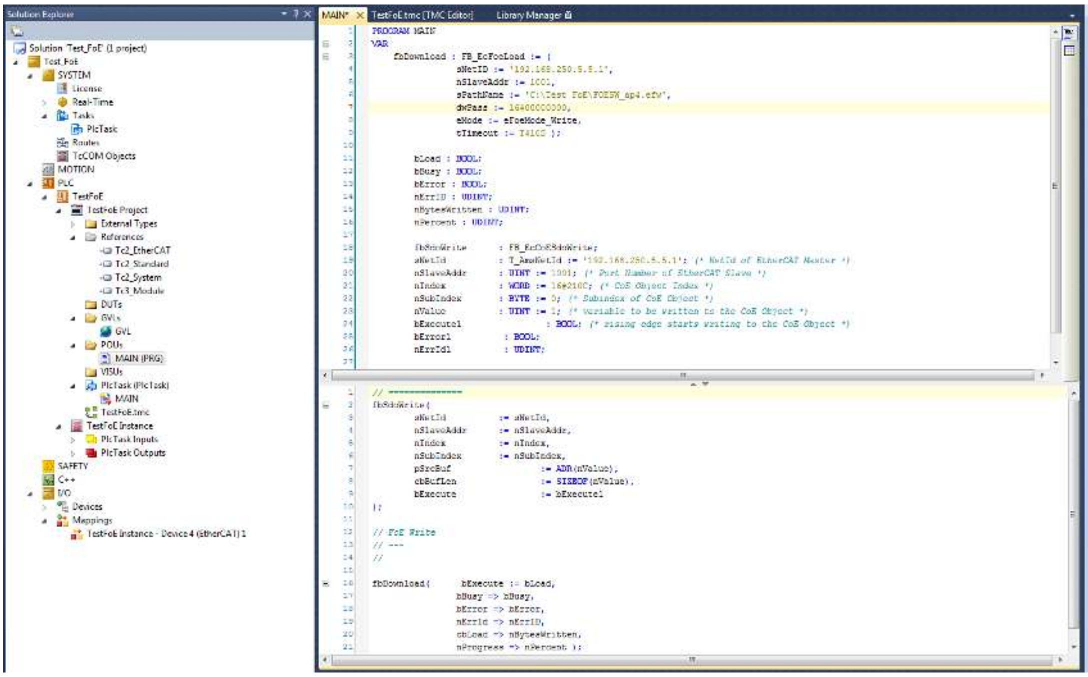
Before running the script file, increase the Mailbox receive timeout to 20000ms.

Click the drive name and select the EtherCAT tab. Click the Advanced settings button and a new window will open.

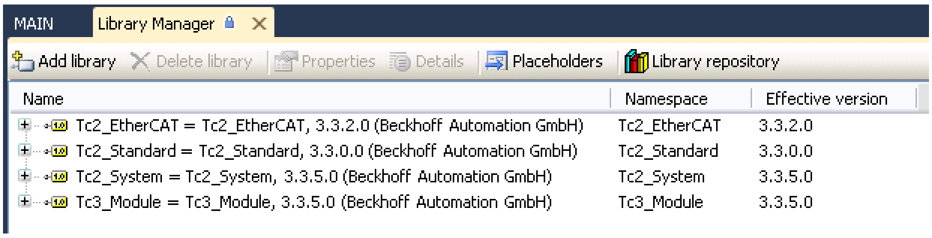


Under General, select the Timeout Settings and write a value of 20000 under response for the Mailbox timeout. Click OK to close the window.

In the PLC area, create a ST program with the lines from the image below:



The library manager must have loaded the libraries from the image below:



17.4.6 Updating the firmware via FoE (File over EtherCAT) TwinCAT 3 GUI example

Only drives with firmware F515F or newer and FA0xx, support FoE (File over EtherCAT) protocol to for Firmware update.

The firmware update file can be supplied only on demand. Please write at support@technosoftmotion.com for more information.

Remark: *the file Access over EtherCAT area in the TwinCAT GUI becomes available only if the drive is programmed with the latest XML information. The XML revision must be 892417350 (0x35313546 or 515F in ASCII) or later. See Par 1.5.4.6 Checking and updating the XML file stored in the drive.*

The firmware file has the following name structure:

“FOEFW_XXXX”. The file must start with and contain the name FOEFW_. XXXX is the firmware name.

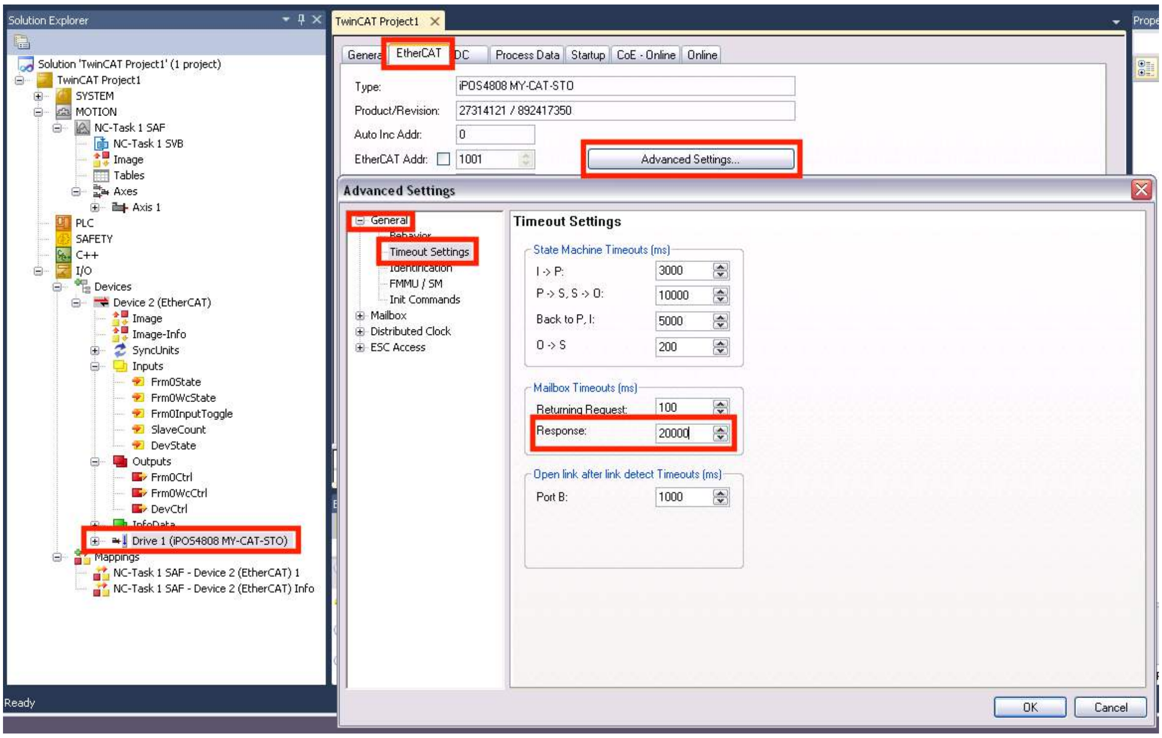
To update a firmware via FoE protocol, the password:

- 0x35313546 must be supplied for F515x firmware versions.
- 0x41303041 must be supplied for FA0xx firmware versions.

Remark: in case the firmware update procedure fails or is interrupted, power cycle the drive and start the procedure again. Starting with F515F, the drives have a non-erasable boot section.

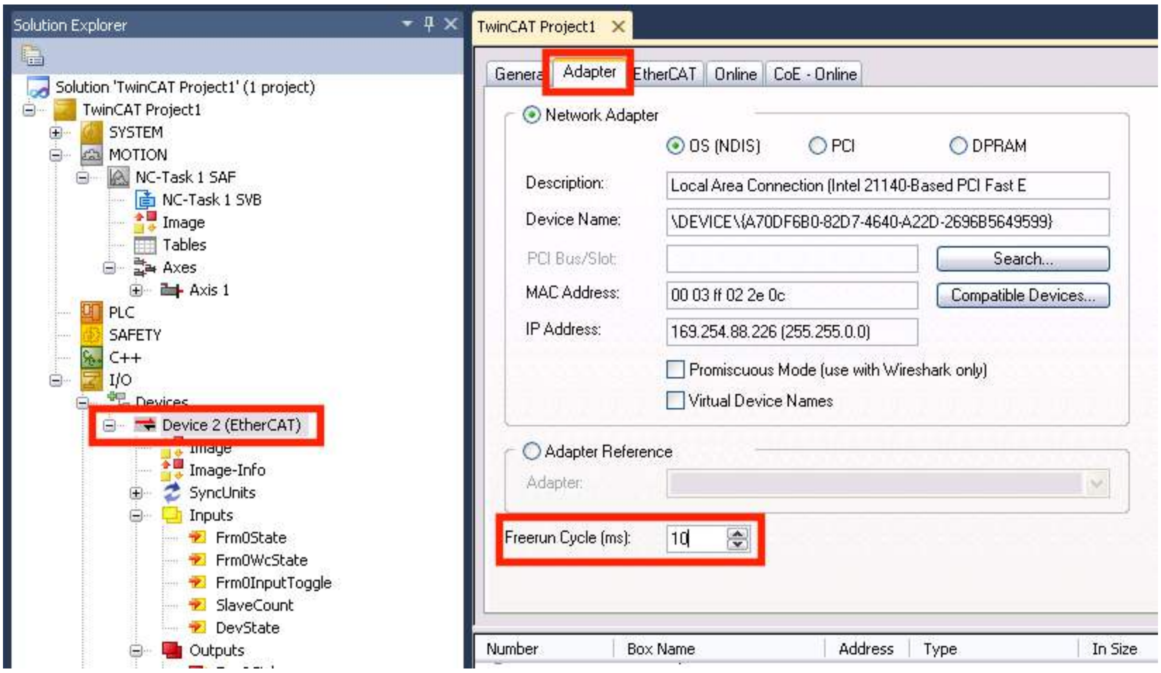
Open a TwinCAT 3 project that is communicating with a Technosoft EtherCAT drive.

Click the drive name and select the EtherCAT tab. Click the Advanced settings button and a new window will open.



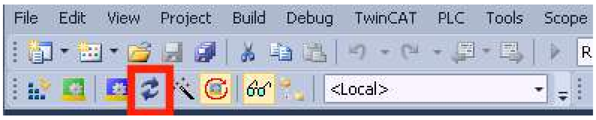
Under General, select the Timeout Settings and write a value of 20000 under response for the Mailbox timeout. Click OK to close the window.

Under Devices, click the EtherCAT device and then select the Adapter tab.

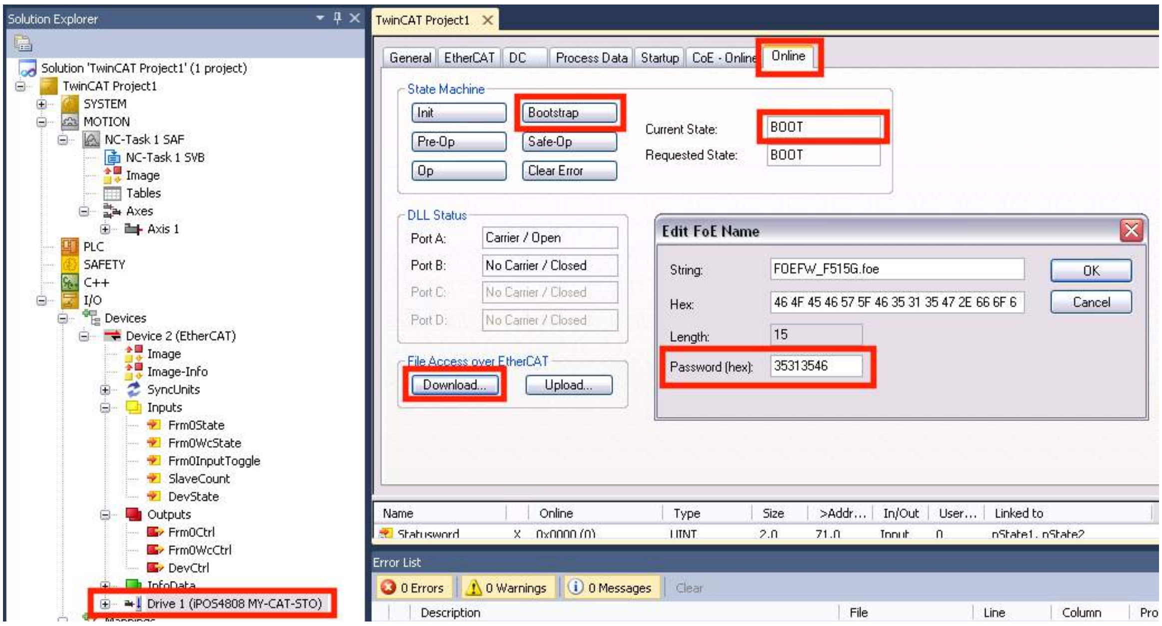


Set for the Freerun cycle a 10ms value.

To apply these new settings, click the Reload Devices in the TwinCAT toolbar to re-start Freerun cycle mode.



Click the drive name again and select the Online tab.



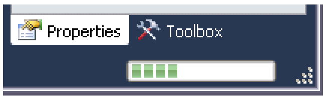
Under state machine, click the Bootstrap button and wait until the drive changes its current state into BOOT.

Under File Access over EtherCAT, click the download button and select the firmware file. In this example, the file name FOEFW_F515G.foe was selected.

Under Password(hex), write 35313546.

Click OK to continue.

A progress bar will be shown in the lower right corner of TwinCAT.



Wait until it finishes and then click the Op button in the state machine section.

The drive will reset internally and start with the new firmware.

Warning: while downloading the firmware, never power off. At first, the flash memory is erased and the drive might be permanently damaged. Only if an error occurs, the power can be cycled to re-establish communication and start the procedure again.

Remarks:

- If the firmware programing fails, most likely the firmware is partially erased. Until a correct firmware is written, the drive will not be fully functional.

In case the TwinCAT update procedure keeps failing, the firmware can still be updated using an RS232 connection and the Technosoft Firmware programmer tool that is included in the EasySetup or Easy Motion Studio software package.

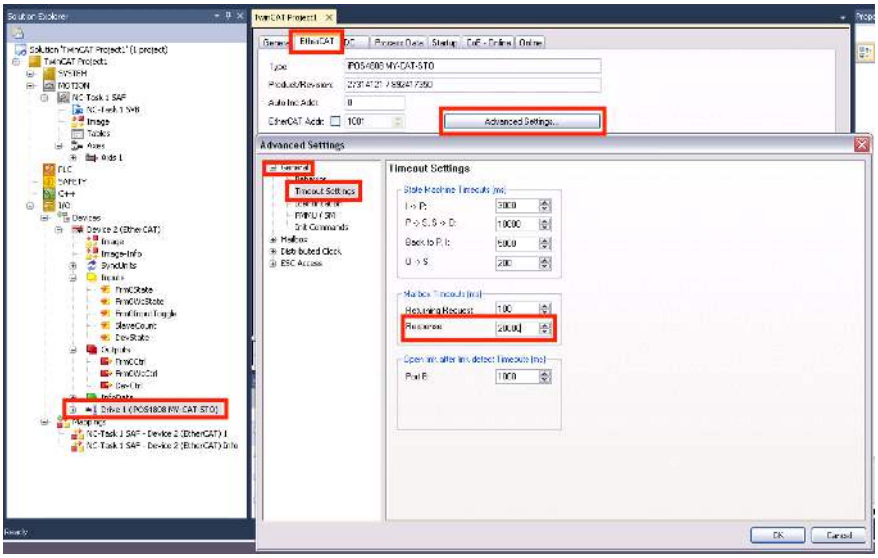
17.4.7 Updating the firmware via FoE with TwinCAT 3 ST script example

Even if not used, please read first Par *17.4.4*.

The firmware can be updated via the TwinCAT “FB_EcFoeLoad” function which sets the drive into Boot mode automatically.

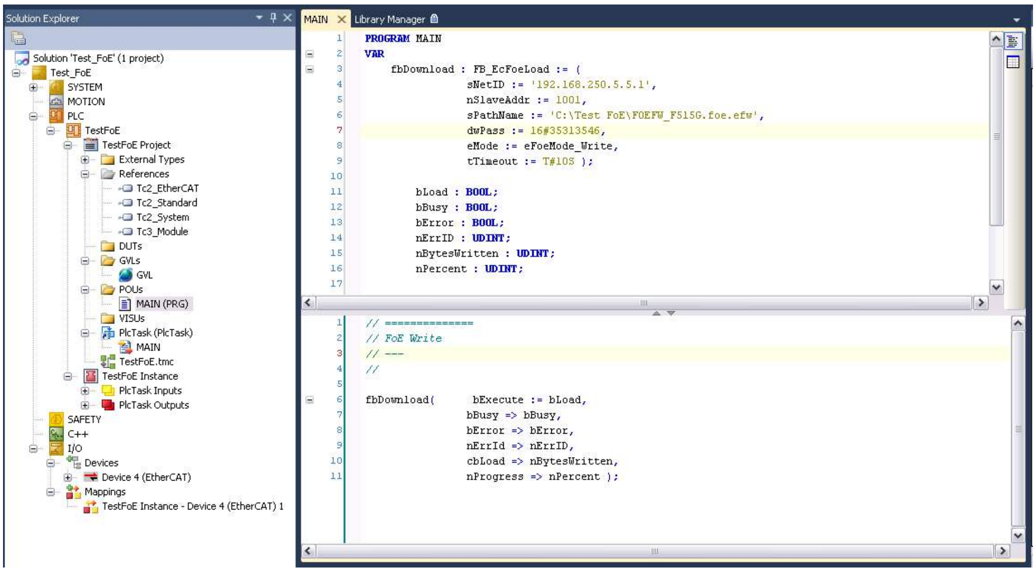
Before running the script file, increase the Mailbox receive timeout to 20000ms.

Click the drive name and select the EtherCAT tab. Click the Advanced settings button and a new window will open.



Under General, select the Timeout Settings and write a value of 20000 under response for the Mailbox timeout. Click OK to close the window.

In the PLC area, create a ST program with the lines from the image below:



The library manager must have loaded the libraries from the image below:

17.5 Ethernet over EtherCAT (EoE) communication[1]

17.5.1 Overview

The Ethernet over EtherCAT (EoE) communication allows the EasyMotion Studio or EasySetup software to communicate with Technosoft EtherCAT slaves over an EtherCAT network without the need of a direct RS232 connection.

Warning: Do not connect the EtherCAT slave directly to a Local Area Network.

The EtherCAT communication will flood the LAN with unsolicited messages. The EtherCAT slave must be connected to an EtherCAT master which can later be connected to the LAN through its dedicated Ethernet port. EoE communication can occur only if the EtherCAT master supports forwarding EoE messages.

For detailed step by step instructions on setting up EoE communication using TwinCAT, read chapter *17.5.3*. The connection diagram below is just an example that uses the settings you can find in chapter *17.5.3*

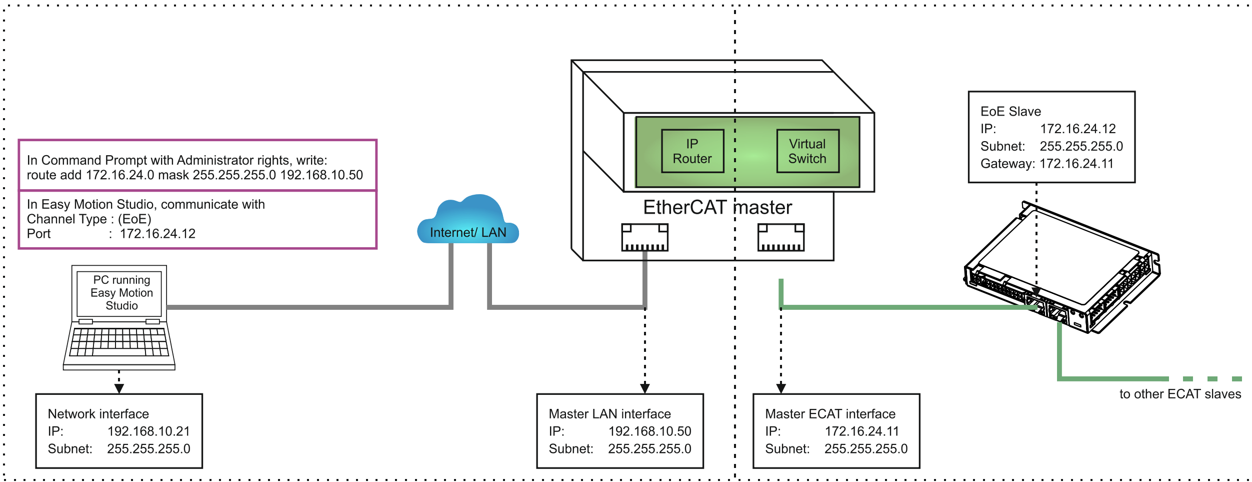


Figure 17.5.1. *EoE example schema*

17.5.2 EoE communication objects

17.5.2.1 Object 210Dh: Virtual MAC address for EoE

This object reveals the virtual MAC address of the drive that has been configured by the EtherCAT master for EoE protocol.

The object is also writable and can be modified with a new value. Keep in mind that the EtherCAT master might overwrite its value every time it re-initializes.

Object description:

Index	210Dh
Name	Virtual MAC address for EoE
Object code	VAR
Data type	UNSIGNED48

Entry description:

[1] EoE communication is available only with F515J firmware or newer.

Access	RW
PDO mapping	No
Units	-
Value range	UNSIGNED48
Default value	No

17.5.2.2 Object 210E$_h$: IP config for EoE

This object reveals the IP settings of the drive that has been configured by the EtherCAT master for EoE protocol.

The object is also writable and can be modified with new values. Keep in mind that the EtherCAT master might overwrite its values every time it re-initializes.

Object description:

Index	210E$_h$
Name	IP config for EoE
Object code	Record
Data type	UNSIGNED32

Entry description:

Sub-index	0
Description	Number of entries
Access	RO
PDO mapping	No
Value range	3
Default value	3

Sub-index	1
Description	IP Address
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	-

Sub-index	2
Description	Subnet Mask
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	-

Sub-index	3
Description	Default Gateway
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	-

Sub-index description and data structure:

Sub-index 1 shows or can set the IP address of the drive. The IP address is written as bytes in little endian format.

Example:

IP address is 192.168.24.12

Converted to bytes in hex it is : **C0**.**A8**.**18**.**0C**

Sub-index 1 value is: 0x**0C18A8C0**

IP address selection:

- the IP address of the drive must be in the same sub-net as the IP address of the Ethernet port of the master that is used for the EtherCAT connection. The EtherCAT slave(s) must be connected to this port.

- the IP address range of the EtherCAT slaves must be on a different subnet than the IP addresses on the LAN.

Example:

Local LAN IP of EtherCAT master: 192.168.**23**.230 – (IP in LAN are between 192.168.23.1 to 255)

IP of the Ethernet port of the master that is connected to the EtherCAT network: 192.168.**24**.11

IP of one EtherCAT slave : 192.168.**24**.12

Sub-index 2 shows or can set the subnet mask of the drive. The data is written as bytes in little endian format.

Example:
Subnet mask is 255.255.255.0
Converted to bytes in hex it is : **FF.FF.FF.00**
Sub-index 1 value is: 0x**00FFFFFF**

Sub-index 3 shows or can set the Gateway address of the drive. The Gateway address is written as bytes in little endian format. The gateway address should be set the same as the IP address of the Ethernet port of the master that is used for the EtherCAT connection.

Example:
Gateway address is 192.168.24.11
Converted to bytes in hex it is : **C0.A8.18.0B**
Sub-index 1 value is: 0x**0B18A8C0**

17.5.3 Setting up EoE communication using EasyMotion Studio and TwinCAT3 example

Prerequisites:
For EoE capability, the Technosoft drives must have:
- the firmware F515J or newer installed
- the newest XML file compatible with F515J or newer should be present in TwinCAT. See *1.5.1 Adding the XML file*.
- the drive XML data should have the revision 892417354 or greater programmed to the ECAT EEPROM. The revision number means 515J when converted to ASCII. See *1.5.4.6 Checking and updating the XML file stored in the drive* to update to the latest version if needed.

17.5.3.1 Step 1 Setting an IP to the EtherCAT network port of the master

On the EtherCAT master, the port that connects to the EtherCAT slaves usually has no fixed IP defined.
Edit Windows Network Adapter settings on the EtherCAT master and manually add an IP to this port and a subnet mask. Make sure no other EtherCAT slave will use this IP.
Important: the IPs for the EtherCAT network must be in a different subnet than the IPs of your local network.

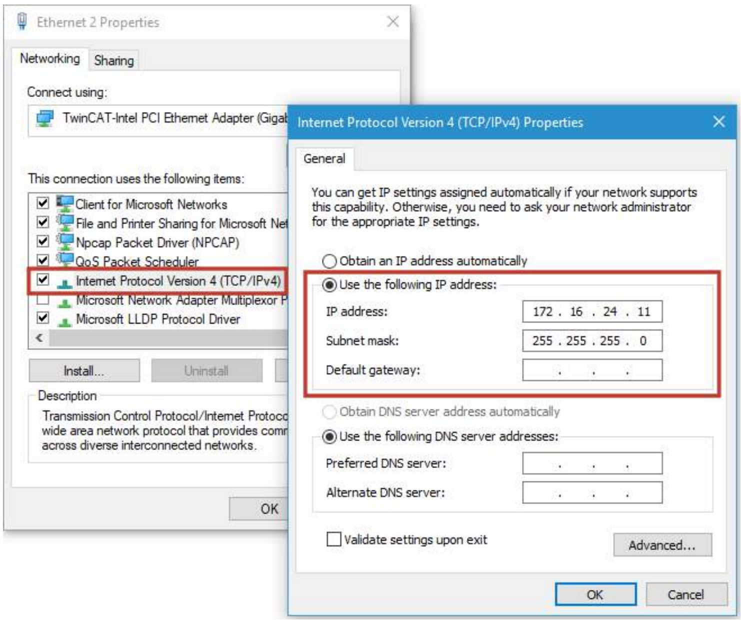
Example of a good configuration:

IPs of local network:	192.168.**23**.1.. to 254	Subnet: 255.255.255.0
IPs of EtherCAT network	192.168.**24**.1 .. to 254	Subnet: 255.255.255.0

Example of a wrong configuration:

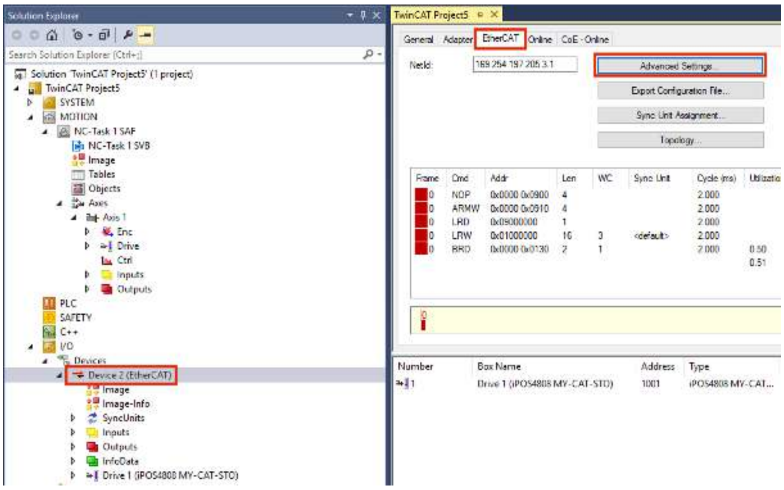
IPs of local network:	192.168.**23**.1.. to 254	Subnet: 255.255.255.0
IPs of EtherCAT network	192.168.**23**.1 .. to 254	Subnet: 255.255.255.0

For this example, set the IP of the EtherCAT interface to 172.16.24.11 and subnet mask to 255.255.255.0. No gateway or DNS are needed.

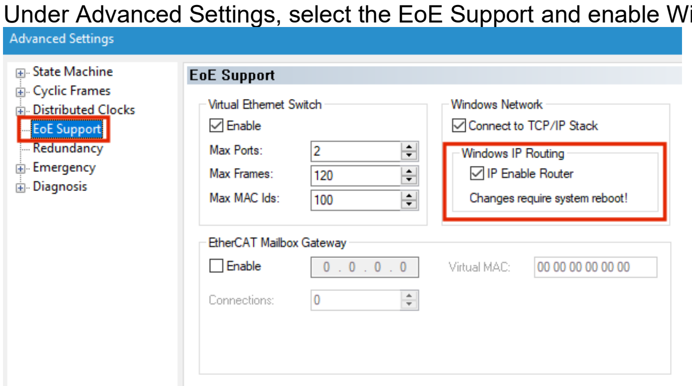


17.5.3.2 Step 2 Configure TwinCAT for EoE by enabling IP routing on the EtherCAT master

In the TwinCAT project, you must first detect all the EtherCAT slaves.
Enable IP Routing on the EtherCAT master to be able to forward EoE packets from the EtherCAT slaves to your LAN.
In TwinCAT, under I/O select the EtherCAT interface, choose the EtherCAT tab and click on the Advanced Settings button as in the image below.



Under Advanced Settings, select the EoE Support and enable Windows IP Routing.



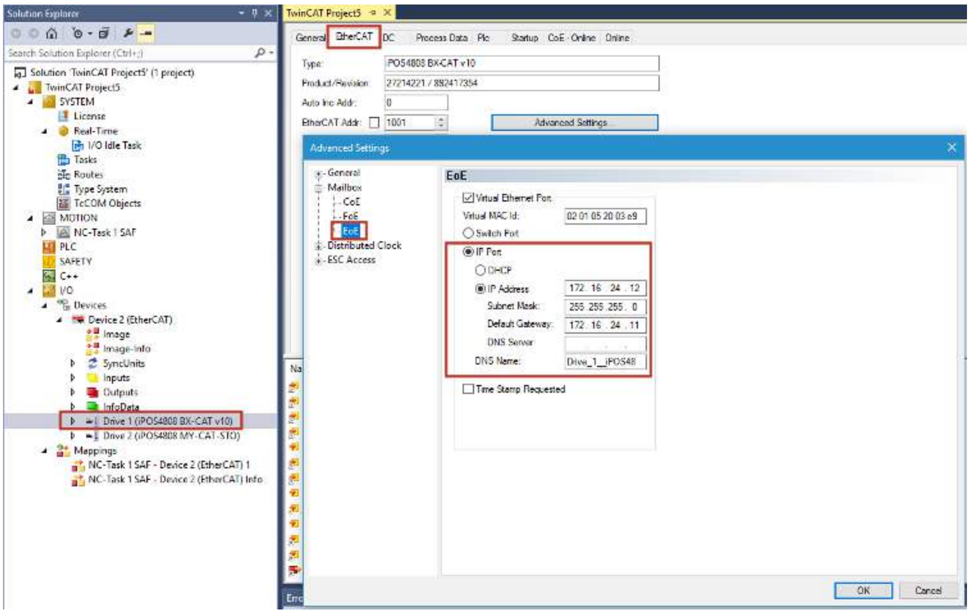
Next, rebuild and save your project, then reboot to apply the new settings.

In case the EtherCAT master has a Windows CE platform, open CX configuration tool and enable "IP Routing".

17.5.3.3 Step 3 Configure TwinCAT to set an IP for the EterCAT slave

In TwinCAT, under I/O select a slave, select the EtherCAT tab and click the Advanced Settings button.
Under Mailbox / EoE, select a manual configuration for the IP Port:

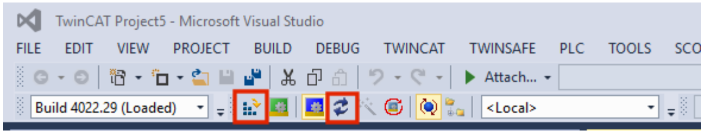
IP Address	- set an IP in the same subnet as the one previously set for the EtherCAT interface - in this example, set 172.16.24.12
Subnet Mask	- in this example, set 255.255.255.0
Default Gateway	- set the IP that was set for the EtherCAT port of the master (set @ step 1) - in this example, set 172.16.24.11



In case other slaves are present and EoE is needed for them too, set an individual IP to each one by repeating this step.

17.5.3.4 Step 4 Enable TwinCAT EoE settings.

Click the “Reload IO devices” button or the “Activate Configuration” button to apply the new settings.



17.5.3.5 Step 5 Configure the PC running EasyMotion to communicate with the EoE slaves.

In Windows, open the Command Prompt using Administrator rights.
Write
route add 172.16.24.0 mask 255.255.255.0 192.168.10.50

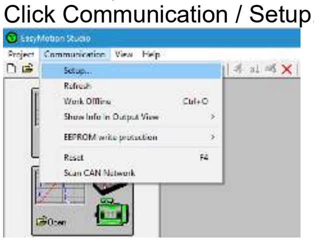
where 172.16.24.0 is the destination IP class of the EtherCAT slaves
255.255.255.0 is the subnet of the destination
192.168.10.50 is the local LAN IP of the EtherCAT master interface.

Remarks:

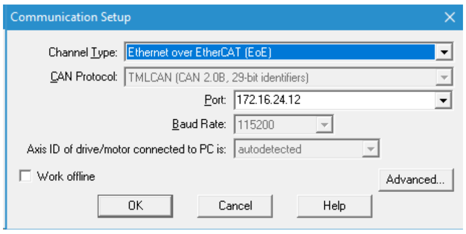
1. the route will be added only until next system reboot. If Windows is restarted, the same command should be given before using EasyMotion Studio with EoE.
2. If the EtherCAT master IP will not change and the route setting is wished to be permanent, write the following in Command Prompt
route add -p 172.16.24.0 mask 255.255.255.0 192.168.10.50
the -p option will set the route as persistent across system reboots

17.5.3.6 Step 6 Configure EasyMotion Studio or EasySetup to communicate with the EoE slave

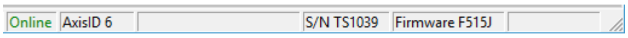
Open EasyMotion Studio or EasySetup.
Click Communication / Setup...



Under Channel Type select Ethernet over EtherCAT (EoE).
Under Port set the IP of the EoE slave you wish to communicate with (the same IP from **Step 3**): 172.16.24.12



Click OK.
If the communication is successful, an "Online" text will be found in the status bar below.



Click New and the communication settings dialogue will come up again. Click OK and choose either:
<u>Upload from Drive/Motor</u> – to read the setup from the drive
<u>Open</u> – in case a project is already present for this drive.
<u>New</u> – to start a new project.

17.5.4 Remarks about EoE limitations

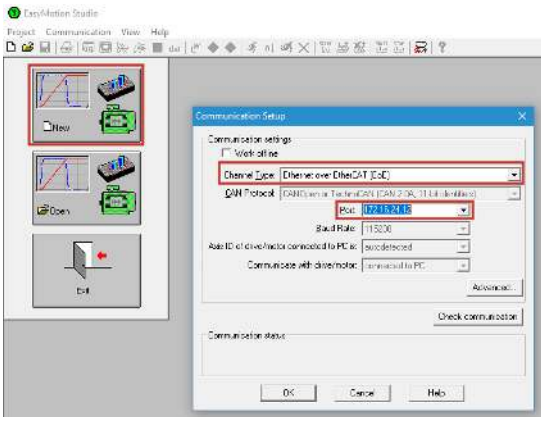
- The response time is slower and the communication is noticeably slower than RS232 communication running at 115200 bps.
- To increase control panels speed, close all of them and choose only the data that is needed. Fewer data means faster updates.
- For better data refresh rate, set the EtherCAT cycle time closest or equal to 1ms.

17.5.5 Example: Starting a new project using EasyMotion Studio with EoE communication

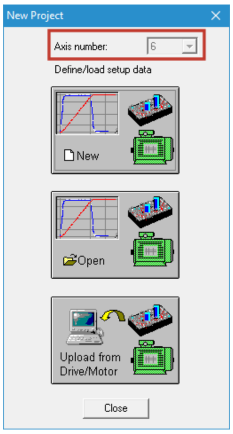
Prerequisites:
- two iPOS drives are already configured in TwinCAT for EoE communication using IPs 172.16.24.12 and 172.16.24.13. See *17.5.3.3*.
- a route was added in Windows to allow communication with the EtherCAT IPs. See *17.5.3.5*.

17.5.5.1 Step 1, establish communication

- Start EasyMotion Studio, click the New button or Project/New and the Communication Setup menu will appear
- For Channel type choose "Ethernet over EtherCAT (EoE)" and for the port number choose an existing EoE IP (172.16.24.12 in this case)

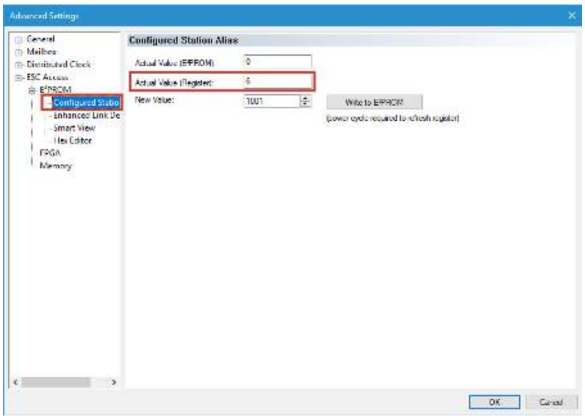


- Click OK
- If the ECAT master is started and the IP route was added, the communication should be successful, and the New Project menu should appear.



- If the drive has HW ID address switches and they were set up, the drive ID number should be visible as a grey number as in the image above.

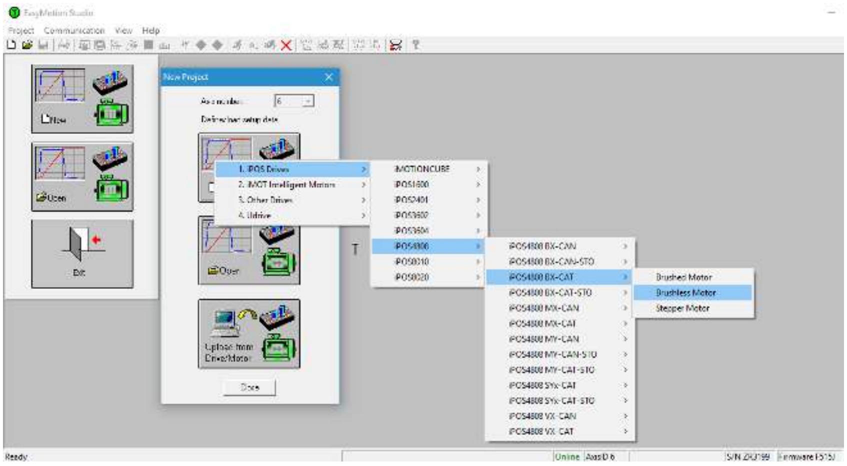
Remark: the drive HW ID address will be visible in TwinCAT as the “Configured Station Alias” value:



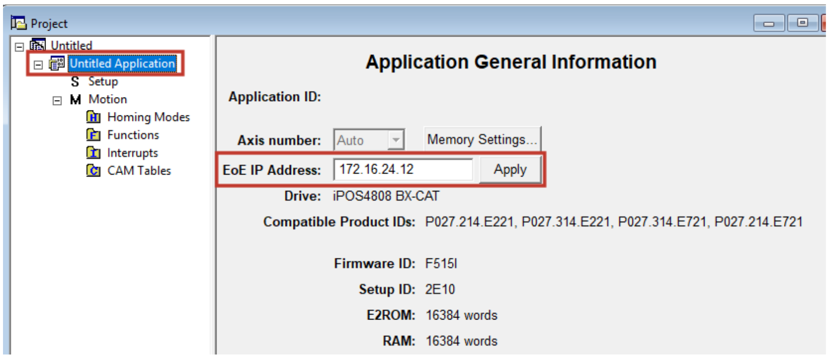
Menu visible from TwinCAT, Advanced Settings button on the EtherCAT tab of the iPOS drive

17.5.5.2 Step 2, create a new project

- Click the New button and choose the drive and motor type.



- The project will load with a new Untitled Application. The IP address of the drive can be found (and modified, if needed) in the “Application General Information” page available when the application name is selected.



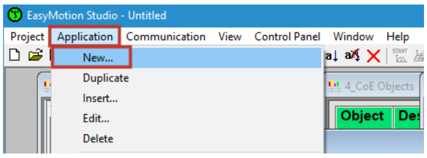
- The Setup and motor tuning can be done via EoE and be later downloaded to the drive.

Remark: because EoE has a higher lag than RS232 communication, setup tests will take longer to execute.

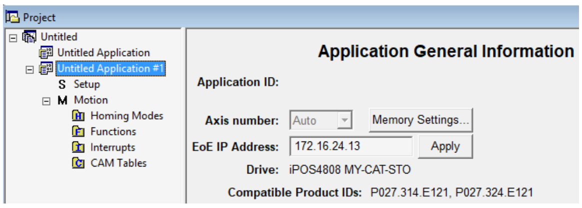
Remark: EasyMotion Studio can have multiple applications within one project file. Each application can correspond to a specific axis within the EtherCAT network. By defining a unique IP for each application, one can switch between applications and communicate directly over EoE with each drive.

17.5.5.3 Step 3, create another application

- Click Application/ New...

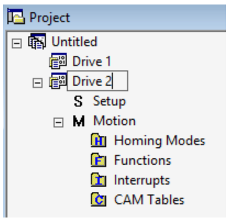


- The Communication Setup menu will appear again as in the previous steps.
- In the “Port” field, define the second EoE IP (172.16.24.13) and click OK to communicate and define the drive and motor type.
- After the new application is defined, the project file should have two applications:



17.5.5.4 Step 4, rename the application names

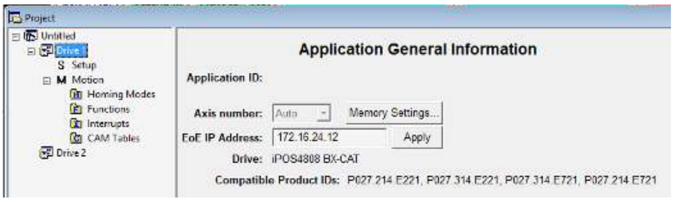
- After selecting an application name, either click once over the name or select Application/Edit to rename the application.



- Now save your project using Project / Save and choose a name.

17.5.5.5 Step 5, switching between applications / drives

- In EasyMotion Studio, just click on the application name of the drive you need and the communication will be established with that drive.
- Control Panels from the selected application will indicate the status of the drive and the setup can be modified / re-tuned.



17.5.6 Example: Converting an existing EasyMotion Studio project made for RS232 to work with EoE communication

Prerequisites:

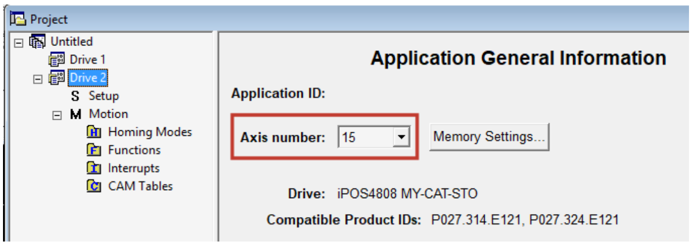
- Two iPOS drives are already configured in TwinCAT for EoE communication using IPs 172.16.24.12 and 172.16.24.13. See *17.5.3.3*.

- A route was added in Windows to allow communication with the EtherCAT IPs. See *17.5.3.5*.

- A project file containing multiple applications that were made while using RS232 communication.

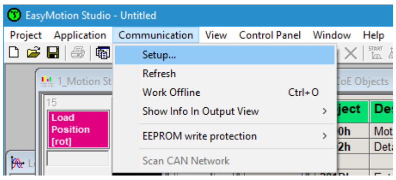
17.5.6.1 Step 1, open the EasyMotion project file and change communication to EtherCAT EoE

- Open an EasyMotion Studio project that was made while using RS232 communication

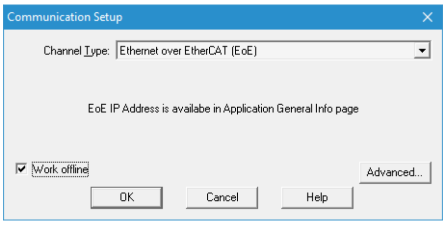


- assuming RS232 communication is still selected in the Communication Settings, an axis number will be assigned and visible in the "Application General Information" page.

- go to Communication / Setup...

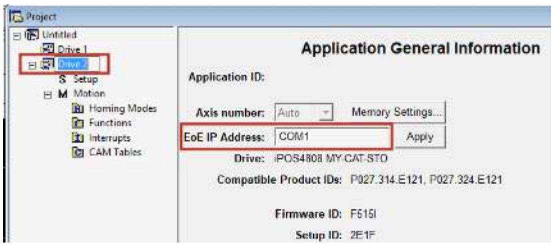


-Choose Ethernet over EtherCAT (EoE) for the channel type, have the Work Offline checkbox enabled and click OK.



17.5.6.2 Step 2, assign the EoE IP of an application

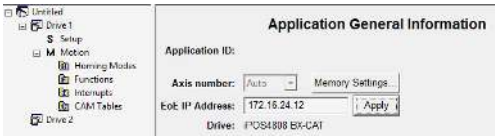
- Now, click on the application name and an EoE IP Address field should be available



- Instead of COM1, write the IP that is assigned to the corresponding drive. In this example the IP 172.16.24.13 will be used. Click Apply when done to assign the IP to the application.

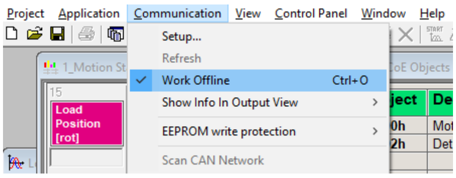
17.5.6.3 Step 3, assign another EoE IP to the second application

- Click on the second application name and edit the IP for the second drive. Click Apply when finished.

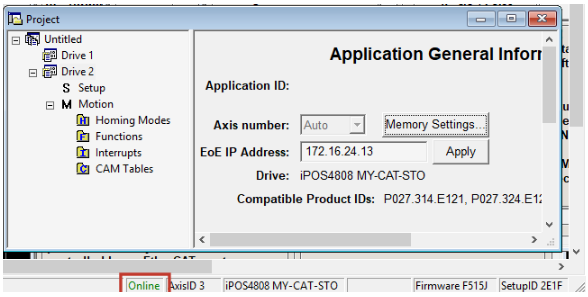


17.5.6.4 Step 4, enable communication and switch between applications / communicate individually with each drive

- go to communication and disable Work Offline mode



- if the prerequisites are met, the communication should be successful. The Online status should be visible in the lower status bar of the EasyMotion window.



- while online with the drive, control panels data can be seen or the setup tuning tests can be done again.

- if communication with the other application / drive is needed, click on the second application name and communication should be established with the defined IP of that application.

18 Advanced features

Due to its embedded motion controller, a Technosoft intelligent drive/motor offers many programming solutions that may simplify a lot the task of a EtherCAT® master. This paragraph overviews a set of advanced programming features which can be used when combining TML programming at drive level with EtherCAT® master control. All features presented below require usage of EasyMotion Studio as TML programming tool.

Remark: *If you do not use the advanced features presented below you do not need EasyMotion Studio.*

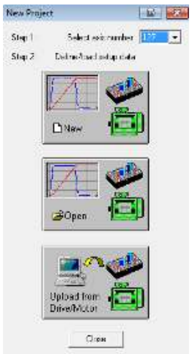
18.1 Using EasyMotion Studio

18.1.1 Starting a new project

Before starting a new project, establish serial communication with the drive. To do this, first read **Paragraph 1.1.3.** The same method for establishing communication applies to EasyMotion Studio as for EasySetup.

Press **New** button 

. A new window will appear.



Step 1, selects the axis number for your drive. By default the drive is delivered with axis number 255.

In Step 2, a setup is defined. The setup data can be opened from a previous save, uploaded from the drive, or select a new one for a new drive.

18.1.2 Choosing the drive, motor and feedback configuration

Press **New** button  and select your drive category: iPOS Drives (all drives from the new iPOS line), Plug In Drives (all plug-in drives, except iPOS line), Open Frame Drives, (all open-frame drives except iPOS line), Closed Frame Drives (all close-frame drives except iPOS line), etc. If you do not know your drive category, you can find it on Technosoft web page.

Continue the selection tree with the motor technology: rotary or linear brushless, brushed, 2 or 3 phase stepper, the control mode in case of steppers (open-loop or closed-loop) and type of feedback device, if any (for example: none or incremental encoder).

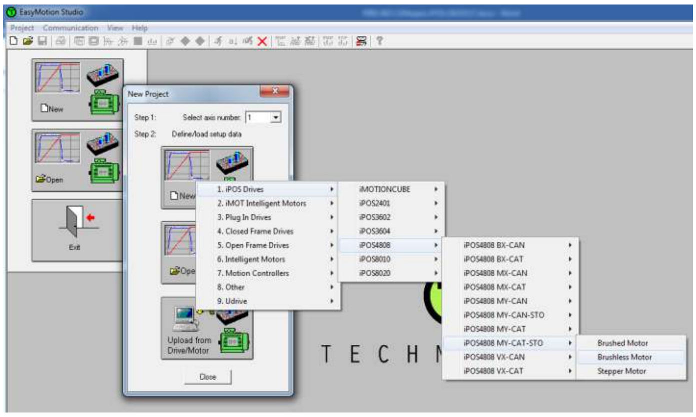


Figure 18.1.1. *EasyMotion Studio – Selecting the drive, motor and feedback*

New windows are loaded which show the project information and current axis number for the selected application. In the background, other customizable windows appear. These are control panels that show and control the drive status through the serial communication interface.

In the left tree, click ***S*** *Setup* item.

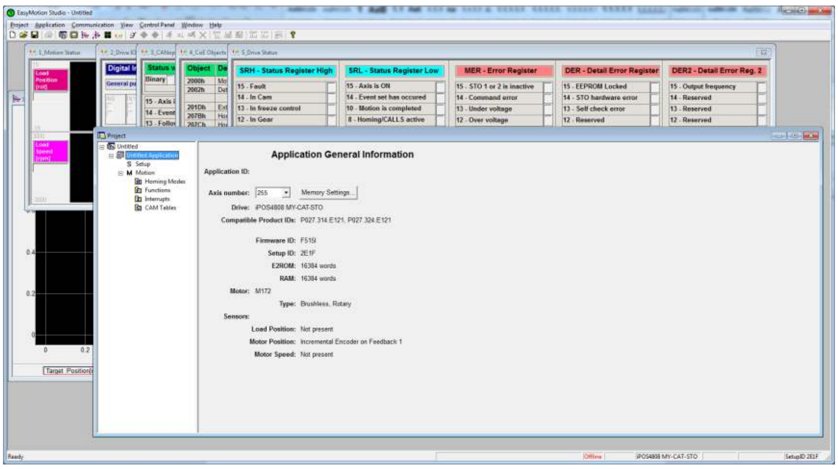


Figure 18.1.2. *EasyMotion Studio – Project information*

To edit the setup, click *View / Modify* button.

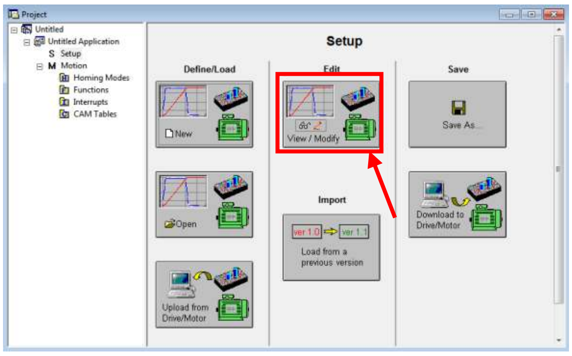


***Figure 18.1.3.** EasyMotion Studio – Editing drive setup*

The selection opens 2 setup dialogues: for **Motor Setup** and for **Drive setup** through which you can introduce your motor data and commission the drive, plus several predefined control panels customized for the drive selected.

For introducing motor data and configuring the drive parameters, please read **Paragraph** *1.1.5* and *1.1.6*.

18.1.3 Downloading setup data to drive/motor

Closing the Drive setup dialogue with **OK**, keeps the new settings only in the EasyMotion Studio project. In order to store the new settings into the drive you need to press the **Download to Drive/Motor** button [Download to Drive/Motor] or the button on the menu toolbar. This downloads the entire setup data in the drive EEPROM memory. The new settings become effective after the next power-on, when the setup data is copied into the active RAM memory used at runtime.

18.2 Using TML Functions to Split Motion between Master and Drives

With Technosoft intelligent drives you can really distribute the intelligence between a EtherCAT® master and the drives in complex multi-axis applications. Instead of trying to command each step of an axis movement, you can program the drives using TML to execute complex tasks and inform the master when these are done. Thus for each axis, the master task may be reduced at: calling TML program / functions (with possibility to abort their execution) stored in the drives EEPROM and waiting for a message, which confirms the finalization of the TML motion / functions execution.

18.2.1 Build TML functions within EasyMotion Studio

The following steps describes how to create TML functions with EasyMotion Studio

1. **Define the TML functions.** Open the EasyMotion Studio project and select the Functions entry from the project tree. On the right side of the project panel add the TML functions executed by the drive. You may also remove, rename and change the functions download order.

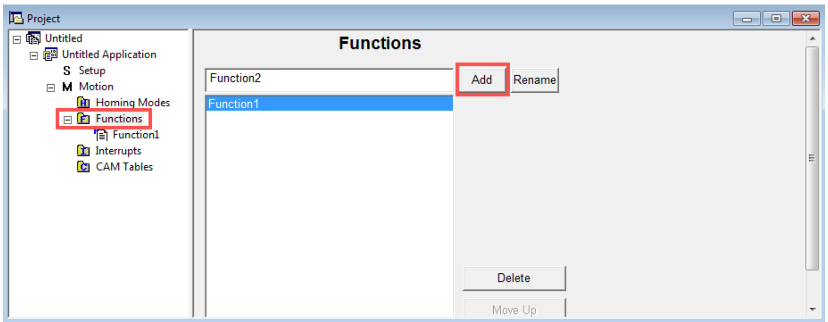


Figure 18.2.1. **EasyMotion Studio project window – Functions add**

2. **Add the TML code.** The added functions are listed in the project tree under the **Functions** entry. Select each function from the list and add the TML code that will be executed by the function.

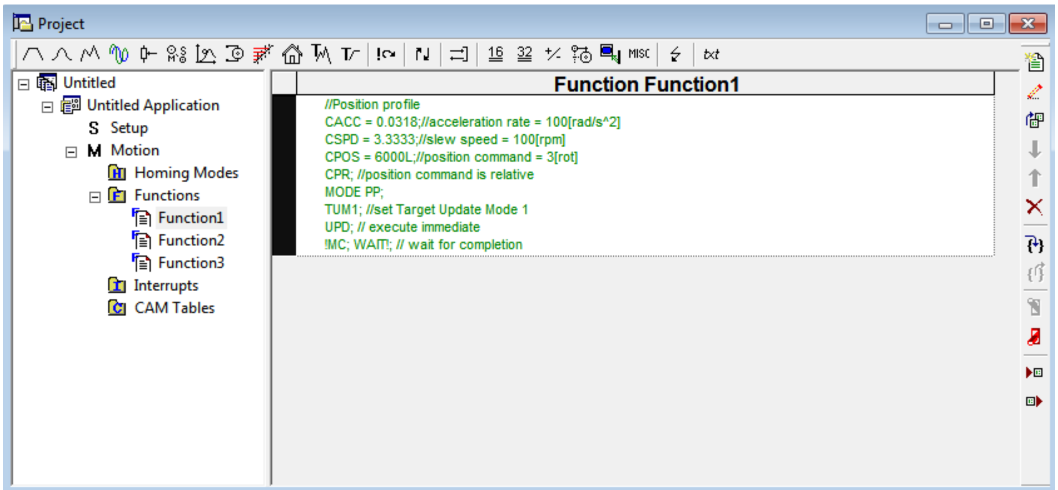


Figure 18.2.2. *EasyMotion Studio project window – functions edit view*

3. **Add the TML code.** The added functions are listed in the project tree under the **Functions** entry. Select each function from the list and add the TML code that will be executed by the function.

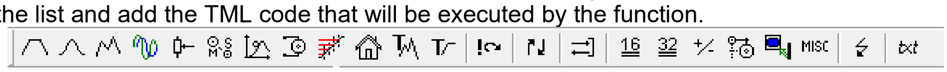


Figure 18.2.3. *EasyMotion Studio project window – Motion wizard bar*

Each button represents a new interactive command.

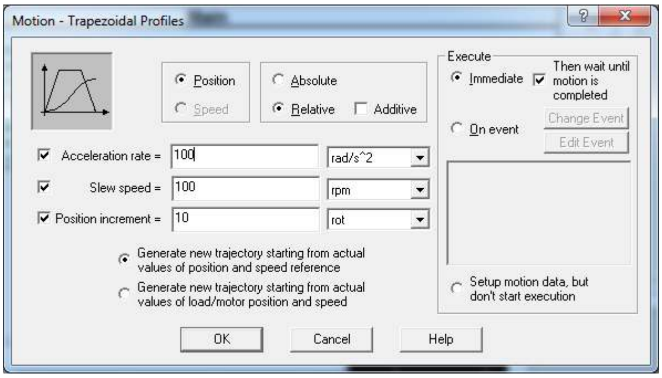


Figure 18.2.4. *EasyMotion Studio project window – Trapezoidal Profile menu*

After clicking **OK** button, the command is converted into code that will be later downloaded to the drive.

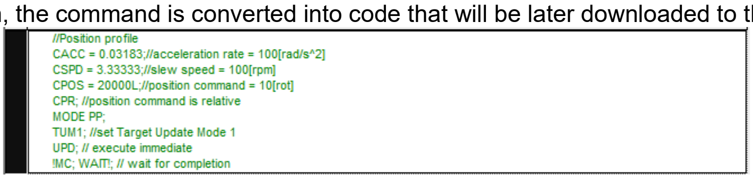


Figure 18.2.5. *EasyMotion Studio project window – Trapezoidal Profile generated motion code*

4. **Download the TML functions into the drive memory**. Use the menu command **Application | Motion | Build** to create the executable code and the menu command **Application | Motion | Download Program** to download the TML code into the drive memory.

18.2.2 TML Function Objects

18.2.2.1 Object 2006_h: Call TML Function

The object allows the execution of a previously downloaded TML function. When a write is performed to this object, the TML function with the index specified in the value provided is called. The TML function body is defined using EasyMotion Studio and saved in the EEPROM memory of the drive. The function index represents an offset in a predefined table of TML callable functions.

It is not possible to call another TML function, while the previous one is still running. Bit 8 of Statusword (6041_h) shows if a function is running. In case a function was called while another was still running, bits 7 (warning) from the Statusword (6041_h) and 14 (command error) from Motion Error Register (2000_h) are set, and the function call is ignored. The execution of any called TML function can be aborted by setting bit 13 in Controlword.

There are 10 TML functions that can be called through this mechanism (the first 10 TML functions defined using the EasyMotion Studio advanced programming environment). Any attempt to call another function (writing a number different from 1...10 in this object) will be signaled with an SDO abort code 0609 0030_h (Value range of parameter exceeded). If a valid value is entered and no TML function is defined in that position, an SDO abort code will be issued: 0800 0020_h (Data cannot be transferred or stored to the application).

The functions are initialized and available for calling, only after Controlword receives the Shutdown command (6040_h = 06).

Object description:

Index	2006_h
Name	Call TML function
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	WO
PDO mapping	No
Units	-
Value range	1...10
Default value	-

18.3 Executing TML programs

The distributed control concept can go on step further. You may prepare and download into a drive a complete TML program including functions, homing procedures, etc. The TML program execution can be started simply by writing a value in the dedicated object.

18.3.1 Object 2077_h: Execute TML program

This object is used in order to execute the TML program from either EEPROM or RAM memory. The TML program is downloaded using the EasyMotion Studio software or by the EtherCAT® master using the .sw file created in EasyMotion Studio.

Writing any value in this object (through the SDO protocol) will trigger the execution of the TML program in the drive. If no TML program is found on the drive, an SDO abort code will be issued: 0800 0020_h (Data cannot be transferred or stored to the application).

If the TML program is downloaded in the EEPROM memory, the beginning address needs to be 4000_h (for F515x firmwares) or 2000_h for (FA00x firmwares).

The TML program can be executed only after Controlword receives the Shutdown command (6040_h = 06).

Object description:

Index	2077_h
Name	Execute TML program
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	WO
PDO mapping	No
Value range	UNSIGNED16
Default value	-

18.4 Loading Automatically Cam Tables Defined in EasyMotion Studio

Apart from CiA402 standard operation modes, Technosoft iPOS drives include others like: electronic gearing, electronic camming, external modes with analogue or digital reference etc. When electronic camming is used, the cam tables can be loaded in the following ways:

The master downloads the cam points into the drive active RAM memory after each power on;

The cam points are stored in the drive EEPROM and the master commands their copy into the active RAM memory

The cam points are stored in the drive EEPROM and during the drive initialization (transition to Ready to switch on status) are automatically copied from EEPROM to the active RAM

For the last 2 options, the cam table(s) are defined in EasyMotion Studio and are included in the information stored in the EEPROM together with the setup data and the TML programs/functions.

Remark: *The cam tables are included in the **.sw** file generated with EasyMotion Studio. Therefore, the master can check the cam presence in the drive EEPROM using the same procedure as for testing of the setup data.*

18.4.1 CAM table structure

The cam tables are arrays of X, Y points, where X is the cam input i.e. the master position and Y is the cam output i.e. the slave position. The X points are expressed in the master internal position units, while the Y points are expressed in the slave internal position units. Both X and Y points 32-bit long integer values. The X points must be positive (including 0) and equally spaced at: 1, 2, 4, 8, 16, 32, 64 or 128 i.e. having the interpolation step a power of 2 between 0 and 7. The maximum number of points for one cam table is 8192.

As cam table X points are equally spaced, they are completely defined by two data: the **Master start value** or the first X point and the **Interpolation step** providing the distance between the X points. This offers the possibility to minimize the cam size, which is saved in the drive/motor in the following format:

1st word (1 word = 16-bit data):

Bits 15-13 – the power of 2 of the interpolation step. For example, if these bits have the binary value 010 (2), the interpolation step is $2^2 = 4$, hence the master X values are spaced from 4 to 4: 0, 4, 8, 12, etc.

Bits 12-0 – the length -1 of the table. The length represents the number of points (one point occupies 2 words)

2nd and 3rd words: the Master start value (long), expressed in master position units. 2nd word contains the low part, 3rd word the high part

4th and 5th words: Reserved. Must be set to 0

Next pairs of 2 words: the slave Y positions (long), expressed in position units. The 1st word from the pair contains the low part and the 2nd word from the pair the high part

Last word: the cam table checksum, representing the sum modulo 65536 of all the cam table data except the checksum word itself.

18.5 Customizing the Homing Procedures

The homing methods defined by the CiA402 are highly modifiable to accommodate your application. If needed, any of these homing modes can be customized. In order to do this you need to select the Homing Modes from your EasyMotion Studio application and in the right side to set as “User defined” one of the Homing procedures. Following this operation the selected procedure will occur under Homing Modes in a sub tree, with the name *HomeX* where X is the number of the selected homing.

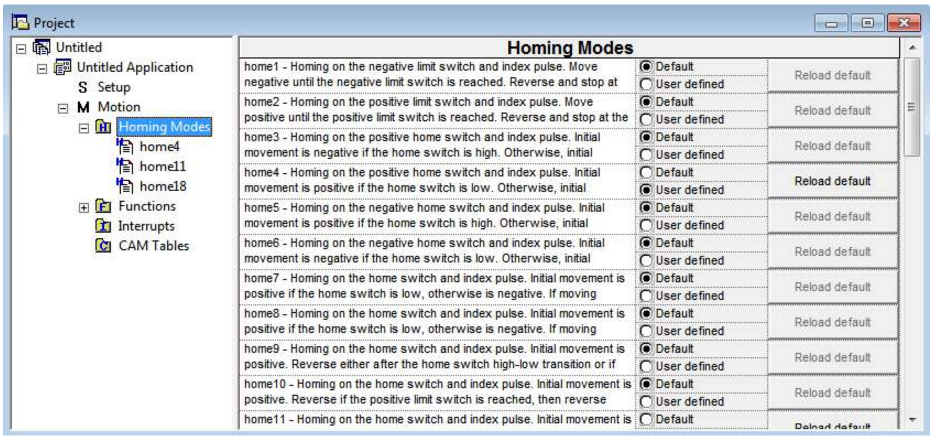


Figure 18.5.1. *EasyMotion Studio project window – Homing procedures customization*

If you click on the *HomeX* procedure, on the right side you’ll see the TML function implementing it. The homing routine can be customized according to your application needs. Its calling name and method remain unchanged.

18.6 Customizing the Drive Reaction to Fault Conditions

Similarly to the homing modes, the default service routines for the TML interrupts can be customized according to your application needs. However, as most of these routines handle the drive reaction to fault conditions, it is mandatory to keep the existent functionality while adding your application needs, in order to preserve the correct protection level of the drive. The procedure for modifying the TML interrupts is similar with that for the homing modes.

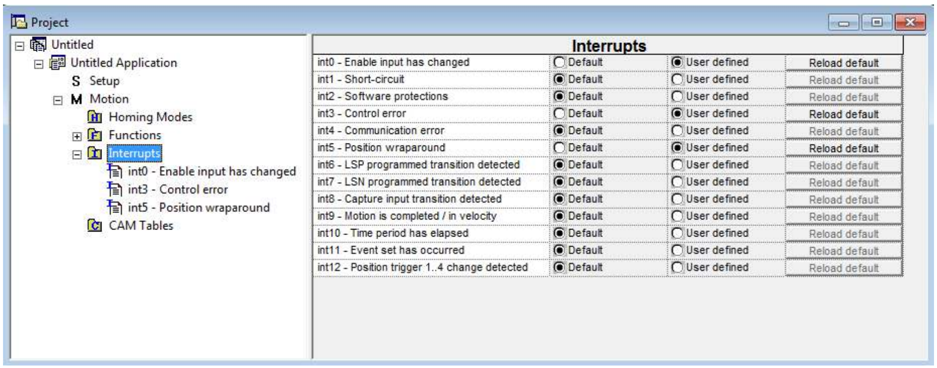


Figure 18.6.1. **EasyMotion Studio project window – TML interrupts customization**

Appendix A: Object Dictionary by Index

Index	Sub-index	Description
1000h	00h	Device type
1001h	00h	Error register
1008h	00h	Manufacturer device name
100Ah	00h	Manufacturer software version
1018h		Identity Object
	00h	Number of entries
	01h	Vendor ID
	02h	Product Code
	03h	Revision Number
	04h	Serial Number
1600h		RPDO1 mapping parameters
	00h	Number of entries
	01h	1st mapped object – 6040h – Controlword
	02h	2nd mapped object – 607Ah – target position
1601h		RPDO2 mapping parameters
	00h	Number of entries
	01h	1st mapped object–60FEh.01 – Digital outputs – Physical outputs
	02h	2nd mapped object–60FEh.02 – Digital outputs – Bit mask
1602h		RPDO3 mapping parameters
	00h	Number of entries
	01h	1st mapped object – 6040h – Controlword
	02h	2nd mapped object – 607Ah – target position
1603h		RPDO4 mapping parameters
	00h	Number of entries
	01h	1st mapped object – 6081h – profile velocity
	02h	2nd mapped object – 6083h – profile acceleration
1A00h		TPDO1 mapping parameters
	00h	Number of entries
	01h	1st mapped object – 6041h – Statusword
	02h	2nd mapped object – 6064h – position actual value
	03h	3rd mapped object – 6077h – Torque (current) actual value
1A01h		TPDO2 mapping parameters
	00h	Number of entries
	01h	1st mapped object – 60F4h – Following error actual value
	02h	2nd mapped object – 60FDh – Digital inputs
1A02h		TPDO3 mapping parameters
	00h	Number of entries
	01h	1st mapped object – 606Ch – Velocity actual value
1A03h		TPDO4 mapping parameters
	00h	Number of entries
	01h	1st mapped object – 6061h – Modes of operation display
1C00h		Sync Manager Communication type
	00h	Number of entries
	01h	Communication Type Sync Manager 0
	02h	Communication Type Sync Manager 1
	03h	Communication Type Sync Manager 2
	04h	Communication Type Sync Manager 3
1C12h		Sync Manager Channel 2 (Process Data Output)
	00h	Number of entries
	01h	PDO Mapping object index of assigned RxPDO : 1st mapped object (1600h)
1C13h		Sync Manager Channel 3 (Process Data Input)
	00h	Number of entries
	01h	PDO Mapping object index of assigned TxPDO : 1st mapped object (1A00h)
	02h	PDO Mapping object index of assigned TxPDO : 2nd mapped object (1A01h)
1C32h		Output Sync Manager Parameter
	00h	Number of entries
	01h	Synchronization Type
	02h	Cycle Time
	04h	Synchronization Types supported
	05h	Minimum Cycle Time

	06$_h$	Calc and Copy Time
	09$_h$	Delay Time
	0B$_h$	SM-Event Missed Counter
	0C$_h$	Cycle Time Too Small Counter
	20$_h$	Sync Error
1C33$_h$		Input Sync Manager Parameter
	00$_h$	Number of entries
	01$_h$	Synchronization Type
	02$_h$	Cycle Time
	04$_h$	Synchronization Types supported
	05$_h$	Minimum Cycle Time
	06$_h$	Calc and Copy Time
	09$_h$	Delay Time
	0B$_h$	SM-Event Missed Counter
	0C$_h$	Cycle Time Too Small Counter
	20$_h$	Sync Error
2000$_h$	00$_h$	Motion Error Register
2001$_h$	00$_h$	Motion Error Register mask
2002$_h$	00$_h$	Detailed Error Register
2005$_h$	00$_h$	Max slippage time out
2006$_h$	00$_h$	Call TML function
2009$_h$	00$_h$	Detailed Error Register 2
2012$_h$	00$_h$	Master resolution
2013$_h$		EGEAR multiplication factor
	00$_h$	Number of entries
	01$_h$	EGEAR ratio numerator (slave)
	02$_h$	EGEAR ratio denominator (master)
2017$_h$	00$_h$	Master actual position
2018$_h$	00$_h$	Master actual speed
2019$_h$	00$_h$	CAM table load address
201A$_h$	00$_h$	CAM table run address
201B$_h$	00$_h$	CAM offset
201D$_h$	00$_h$	External reference type
201E$_h$	00$_h$	Master position
2022$_h$	00$_h$	Control effort
2023$_h$	00$_h$	Jerk time
2025$_h$	00$_h$	Stepper current in open loop operation
2026$_h$	00$_h$	Stand-by current for stepper in open loop operation
2027$_h$	00$_h$	Timeout for stepper stand-by current
2045$_h$	00$_h$	Digital outputs status
2046$_h$	00$_h$	Analogue input: Reference
2047$_h$	00$_h$	Analogue input: Feedback
2050$_h$	00$_h$	Over current protection level
2051$_h$	00$_h$	Over current time out
2052$_h$	00$_h$	Motor nominal current
2053$_h$	00$_h$	I2t protection integrator limit
2054$_h$	00$_h$	I2t protection scaling factor
2055$_h$	00$_h$	DC-link voltage
2058$_h$	00$_h$	Drive temperature
2060$_h$	00$_h$	Software version of the TML application
2064$_h$	00$_h$	Read/Write configuration register
2065$_h$	00$_h$	Write data at address set in object 2064$_h$ (16/32 bits)
2066$_h$	00$_h$	Read data from address set in object 2064$_h$ (16/32 bits)
2067$_h$	00$_h$	Write data at specified address
2066$_h$	00$_h$	Checksum configuration register
206A$_h$	00$_h$	Checksum read register
206B$_h$	00$_h$	CAM input scaling factor
206C$_h$	00$_h$	CAM output scaling factor
206F$_h$	00$_h$	Time notation index
2070$_h$	00$_h$	Time dimension index
2071$_h$		Time factor
	00$_h$	Number of entries
	01$_h$	Numerator
	02$_h$	Divisor
2072$_h$	00$_h$	Interpolated position mode status
2073$_h$	00$_h$	Interpolated position buffer length
2074$_h$	00$_h$	Interpolated position buffer configuration
2075$_h$		Position triggers

	00_h	Number of entries
	01_h	Position trigger 1
	02_h	Position trigger 2
	03_h	Position trigger 3
	04_h	Position trigger 4
2076h	00_h	Save current configuration
2077h	00_h	Execute TML program
2079h	00_h	Interpolated position initial position
207Ah	00_h	Interpolated position 1st order time
207Bh	00_h	Homing current threshold
207Ch	00_h	Homing current threshold time
207Dh	00_h	Dummy
207Fh	00_h	Current limit
2080h	00_h	Reset drive
2081h	00_h	Set/Change the actual motor position value
2082h	00_h	Sync on fast loop
2083h	00_h	Encoder resolution for step loss protection
2084h	00_h	Stepper resolution for step loss protection
2085h	00_h	Position triggered outputs
2086h	00_h	Limit speed for CSP
2087h	00_h	Actual internal velocity from sensor on motor
2088h	00_h	Actual internal position from sensor on motor
2089h	00_h	Synchronization test config
208Ah	00_h	Save setup status
208Bh	00_h	Sin AD signal from Sin/Cos encoder
208Ch	00_h	Cos AD signal from Sin/Cos encoder
208Dh	00_h	Auxiliary encoder position
208Eh	00_h	Auxiliary Settings Register
208Fh		Digital inputs 8bit
	00_h	Number of entries
	01_h	Device profile defined inputs
	02_h	Manufacturer specific inputs
2090h		Digital outputs 8bit
	00_h	Number of entries
	01_h	Physical outputs 8bit
	02_h	Bit mask 8bit
2091h	00_h	Lock EEPROM
2092h		User Variables
	00_h	Number of entries
	01_h	UserVar1
	02_h	UserVar2
	03_h	UserVar3
	04_h	UserVar4
20A0h		Load Position and Speed monitoring
	00_h	Number of entries
	01_h	Reserved
	02_h	Load Position Monitor
	03_h	Load Speed Monitor
2100h	00_h	Number of steps per revolution
2101h	00_h	Number of microsteps per step
2102h	00_h	Brake status
2103h	00_h	Number of encoder counts per revolution
2104h	00_h	Auxiliary encoder function
2105h	00_h	Auxiliary encoder status
2106h	00_h	Auxiliary encoder captured position positive edge
2107h	00_h	Auxiliary encoder captured position negative edge
2108h		Filter variable 16bit
	00_h	Number of entries
	01_h	16 bit variable address
	02_h	Filter strength
	03_h	Filtered variable 16bit
2109h	00_h	Sync offset
210Ah	00_h	Sync rate
210Bh	00_h	Auxiliary Settings Register 2
210Ch	00_h	Enable SW file download
210Dh	00_h	Virtual MAC address for EOE
210Eh		IP config for EoE
	00_h	Number of entries

	01_h	IP address
	02_h	Subnet mask
	03_h	Default gateway
[210Fh](#)		Acceleration encoder factor
	00_h	Number of entries
	01_h	Acceleration internal units (IU)
	02_h	Acceleration units (AU)
2110h		Jerk encoder factor
	00_h	Number of entries
	01_h	Jerk internal units (IU)
	02_h	Jerk units (JU)
2113h		Detailed Option Code
	00_h	Number of entries
	01_h	Short-Circuit option code
	02_h	Reserved
	03_h	Control error option code
	04_h	Communication error option code
	05_h	Reserved
	06_h	Reserved
	07_h	Reserved
	08_h	Over current option code
	09_h	Reserved
	10_h	Over temperature – Motor option code
	11_h	Over temperature – Drive option code
	12_h	Over voltage option code
	13_h	Under voltage option code
	14_h	Reserved
	15_h	Enable / STO inactive option code
2114h	00_h	Fault Override Option Code
2115h	00_h	ASR4
6007h	00_h	Abort connection option code
603Fh	00_h	Error code
6040h	00_h	Controlword
6041h	00_h	Statusword
605Ah	00_h	Quick stop option code
605Bh	00_h	Shutdown option code
605Ch	00_h	Shutdown option code
605Dh	00_h	Disable operation option code
605Eh	00_h	Fault reaction option code
6060h	00_h	Modes of operation
6061h	00_h	Modes of operation display
6062h	00_h	Position demand value
6063h	00_h	Position actual internal value
6064h	00_h	Position actual value
6065h	00_h	Following error window
6066h	00_h	Following error time out
6067h	00_h	Position window
6068h	00_h	Position window time
6069h	00_h	Velocity sensor actual value
606Bh	00_h	Velocity demand value
606Dh	00_h	Velocity window
606Eh	00_h	Velocity window time
606Ch	00_h	Velocity actual value
606Fh	00_h	Velocity threshold
6071h	00_h	Target torque
6075h	00_h	Motor rate current
6077h	00_h	Torque actual value
607Ah	00_h	Target position
607Bh		Position range limit
	00_h	Number of entries
	01_h	Min position range limit
	02_h	Max position range limit
607Ch	00_h	Home offset
607Dh		Software position limit
	00_h	Number of entries
	01_h	Minimal position limit
	02_h	Maximal position limit
607Eh	00_h	Polarity

6080h	00h	Max motor speed	
6081h	00h	Profile velocity	
6083h	00h	Profile acceleration	
6085h	00h	Quick stop deceleration	
6086h	00h	Motion profile type	
6087h	00h	Torque slope	
6089h	00h	Position notation index	
608Ah	00h	Position dimension index	
608Bh	00h	Velocity notation index	
608Ch	00h	Velocity dimension index	
608Dh	00h	Acceleration notation index	
608Eh	00h	Acceleration dimension index	
6091h		Gear Ratio	
	00h	Number of entries	
	01h	Motor rotation	
	02h	Load rotation	
6092h		Feed constant	
	00h	Number of entries	
	01h	Feed	
	02h	Shaft rotation	
6093h		Position factor	
	00h	Number of entries	
		Factor group – CiA 402	Factor group – CiA 402-2
	01h	Numerator	Position internal units (IU)
	02h	Divisor	Position units (PU)
6094h		Velocity encoder factor	
	00h	Number of entries	
		Factor group – CiA 402	Factor group – CiA 402-2
	01h	Numerator	Velocity internal units (IU)
	02h	Divisor	Velocity units (VU)
6096h		Velocity factor	
	00h	Number of entries	
	01h	Velocity units (VU)	
	02h	Position units (PU)	
6097h		Acceleration factor	
	00h	Number of entries	
		Factor group – CiA 402	Factor group – CiA 402-2
	01h	Numerator	Acceleration units (AU)
	02h	Divisor	Velocity units (VU)
6098h	00h	Homing method	
6099h		Homing speeds	
	00h	Number of entries	
	01h	Speed during search for switch	
	02h	Speed during search for zero	
609Ah	00h	Homing acceleration	
60A2h		Jerk factor	
	00h	Number of entries	
	01h	Jerk Units (JU)	
	02h	Acceleration units (AU)	
60A8h	00h	SI unit position	
60A9h	00h	SI unit velocity	
60AAh	00h	SI unit acceleration	
60ABh	00h	SI unit jerk	
60B8h	00h	Touch probe function	
60B9h	00h	Touch probe status	
60BAh	00h	Touch probe 1 positive edge	
60BBh	00h	Touch probe 1 negative edge	
60BCh	00h	Touch probe 2 positive edge	
60BDh	00h	Touch probe 2 negative edge	
60C0h	00h	Interpolation sub mode select	
60C1h		Interpolation Data Record	
	00h	Number of entries	
	01h	The first parameter	
	02h	The second parameter	
60C2h		Interpolation Time Period	
	00h	Number of entries	
	01h	Interpolation time period value	
	02h	Interpolation time index	

60F2h	00h	Positioning Option Code
60F4h	00h	Following error actual value
60F8h	00h	Max slippage
60FCh	00h	Position demand internal value
60FDh	00h	Digital inputs
60FEh		Digital outputs
	00h	Number of entries
	01h	Physical outputs
	02h	Bit mask
60FFh	00h	Target velocity
6502h	00h	Supported drive modes

Appendix B: Definition of Dimension Indices

Dimension/Notation Index Table

physical dimension	dimension index	units exponent	unit type	notation index
non	0	units		0
length	1		metre	0
		milli	metre	-3
		kilo	metre	3
		micro	metre	-6
area	2	square	metre	0
		square milli	metre	-6
		square kilo	metre	6
volume	3	cubic	metre	0
time	4		second	0
			minute	70
			hour	74
			day	77
		milli	second	-3
		micro	second	-6
actual power	9		watt	0
		kilo	watt	3
		mega	watt	6
		milli	watt	-3
apparent power	10		voltampere	0
		kilo	voltampere	3
		mega	voltampere	6
no. of revolutions	11		per second	0
			per minute	73
			per hour	74
angle	12		radian	0
			second	75
			minute	76
			degree	77
			newdegree	78
velocity	13		metre p. second	0
		milli	metre p. second	-3
		milli	metre p. minute	79
			metre p. minute	80
		kilo	metre p. minute	81
		milli	metre p. hour	82
			metre p. hour	83
		kilo	metre p. hour	84
torque	16		newton metre	0
		kilo	newton metre	3
		mega	newton metre	6
temperature	17		kelvin	0
			centigrade	94
			Fahrenheit	95
voltage	21		Volt	0
		kilo	Volt	3
		milli	Volt	-3
		micro	Volt	-6
current	22		Ampere	0
		kilo	Ampere	3
		milli	Ampere	-3
		micro	Ampere	-6
ratio	24		percent	0
frequency	28		Hertz	0
		kilo	Hertz	3
		mega	Hertz	6
		giga	Hertz	9
steps	32		steps	0

encoder resolution	33	revolution steps per	0

Examples for Notation Indices

Examples for notation indices < 64:

For notation index <64 the value is used as an exponent. The unit is defined by the physical dimension and calculated by unit type and exponent, all declared in the dimension/notation index table above.

position unit dimension index = 1: length
notation index = -6: micro meter

position_units = $10^{notation\_index}$ x f(dimension_index) = 10^{-6} m

dimension index = 12: angle notation index
= 0: radian

position_units = $10^{notation\_index}$ x f(dimension_index) = radian

velocity unit

dimension index = 13: velocity notation index = -3: milli metre per second

velocity_units = $10^{notation\_index}$ x f(dimension_index) = 10^{-3} m/s

frequency units dimension index = 28:
frequency notation index = 3: kilo hertz

frequency_units = $10^{notation\_index}$ x f(dimension_index) = 10^{3} Hz

Examples for notation indices > 64:

The unit is defined by the physical dimension and unit type, both declared in the dimension/notation index table.

time units

dimension index = 4: time notation index = 77: day

time_units = f(dimension_index,notation_index) = day

position unit dimension index = 12:
angle notation index = 76:
minute

position_units = f(dimension_index,notation_index)

= minute