iPOS
CANopen
Programming

T E C H N O S O F T
MOTION TECHNOLOGY

User Manual

P091.063.CANopen.iPOS.UM.1224

Table of contents

Read This First

About This Manual

This manual describes how to program Technosoft iPOS family of intelligent drives using **CANopen** protocol. The iPOS drives are conforming to **CiA 301 v4.2** application layer and communication profile, **CiA WD 305 v.2.2.13**[1] Layer Setting Services and to **CiA (DSP) 402 v4.0** device profile for drives and motion control, now included in IEC 61800-7-1 Annex A, IEC 61800-7-201 and IEC 61800-7-301 standards. The manual presents the object dictionary associated with these three profiles. It also explains how to combine the Technosoft Motion Language (**TML**) commands and the CANopen protocol commands in order to distribute the application between the CANopen master and the Technosoft drives.

In order to operate the Technosoft iPOS drives, you need to pass through 3 steps:

- **Step 1 Hardware installation**
- **Step 2 Drive setup** using Technosoft **EasySetup** software for drive commissioning
- **Step 3 Motion programming** using one of the options:
  - A **CANopen master**
  - The drives **built-in motion controller** executing a Technosoft Motion Language (**TML**) program developed using Technosoft **EasyMotion Studio** software
  - A **TML_LIB motion library for PCs** (Windows or Linux)
  - A **TML_LIB motion library for PLCs**
  - A **distributed control** approach which combines the above options, like for example a host calling motion functions programmed on the drives in TML

This manual covers an introductory part of **Step 2** and **Step 3/ Motion programming using the CANopen protocol** in detail.

For Step 1, please consult the drive User Manual, where a detailed hardware installation is described.

Scope of This Manual

This manual applies to the iPOS family of Technosoft intelligent drives.

Notational Conventions

This document uses the following conventions:

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
Axis ID or **CAN ID** or **COB ID** – the unique number allocated to each drive in a network.
RO – read only
RW – read and write
SW – software
H/W or HW - hardware

---

[1] Available only with the firmware F514x.

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**

TML_LIB v2.0 **(part no. P091.040.v20.UM.xxxx)** – explains how to program in **C, C++, C#, Visual Basic or Delphi Pascal** a motion application for the Technosoft intelligent drives using TML_LIB v2.0 motion control library for PCs. The manual includes over 40 ready-to-run examples that can be executed on **Windows** or **Linux** (x86 and x64)

TML_LIB_LabVIEW v2.0 **(part no. P091.040.LABVIEW.v20.UM.xxxx)** – explains how to program in **LabVIEW** a motion application for the Technosoft intelligent drives using TML_LIB_LabVIEW v2.0 motion control library for PCs. The manual includes over 40 ready-to-run examples.

TML_LIB_S7 **(part no. P091.040.S7.UM.xxxx)** – explains how to program a PLC Siemens series S7-300 or S7-400 with a motion application for the Technosoft intelligent drives using TML_LIB_S7 motion control library. The manual includes over 40 ready-to-run examples. The library is PLCOpen compatible.

TML_LIB_CJ1 **(part no. P091.040.CJ1.UM.xxxx)** – explains how to program a PLC Omron series CJ1 with a motion application for the Technosoft intelligent drives using TML_LIB_CJ1 motion control library for PCs. The manual includes over 40 ready-to-run examples. The library is **PLCOpen** compatible.

TML_LIB_X20 **(part no. P091.040.X20.UM.xxxx)** – explains how to program in a PLC **B&R series X20** a motion application for the Technosoft intelligent drives using TML_LIB_X20 motion control library for PLCs. The TML_LIB_X20 library is **IEC61131-3 compatible**

TechnoCAN **(part no. P091.063.TechnoCAN.UM.xxxx)** – presents TechnoCAN protocol – an extension of the CANopen communication profile used for TML commands

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:
  - 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

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:

Using the CD provided by Technosoft. In this case, after installation, use the ***Update via Internet*** tool to check for the latest updates;
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.* ***19.1.1****. 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 or CAN interface. 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.

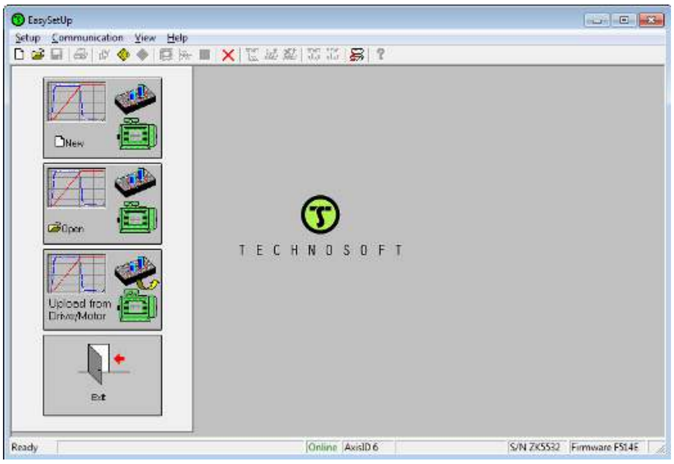


Figure 1.1.1. *EasySetup - Opening window*

All Technosoft drives with CAN interface have a unique AxisID (address) for serial communication. The AxisID value is by default 255 or it is set by the levels of 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 or CAN interface
CAN Protocol: CANopen or TechnoCAN (protocol does not matter if channel type is RS232)
Port: Select the COM port where you have connected the drive
Baud rate: can be any value for RS232 and it is automatically detected. For best performance, we recommend to use the highest value: 115200. For a CAN interface, choose the default baud rate 500 Kbps.

> ***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 **(**autodetected) for RS232 or the CAN Axis ID which is by default 127 in CANopen.

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: 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).

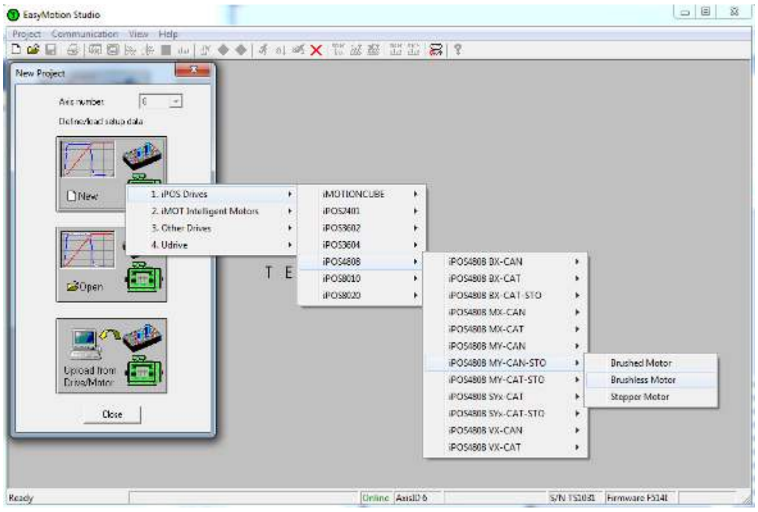


Figure 1.1.2. *EasySetup – Selecting the drive, motor and feedback*

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*

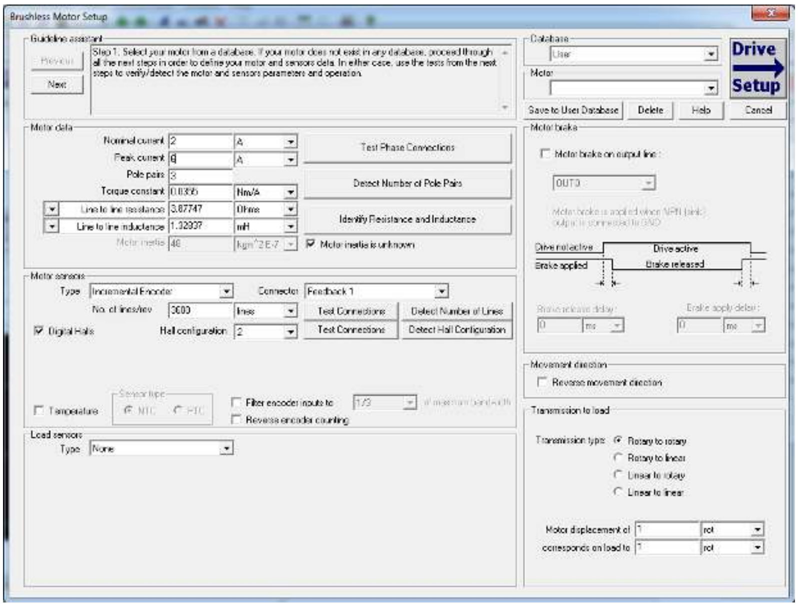


Figure 1.1.3. *EasySetup – Introducing motor data*

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.

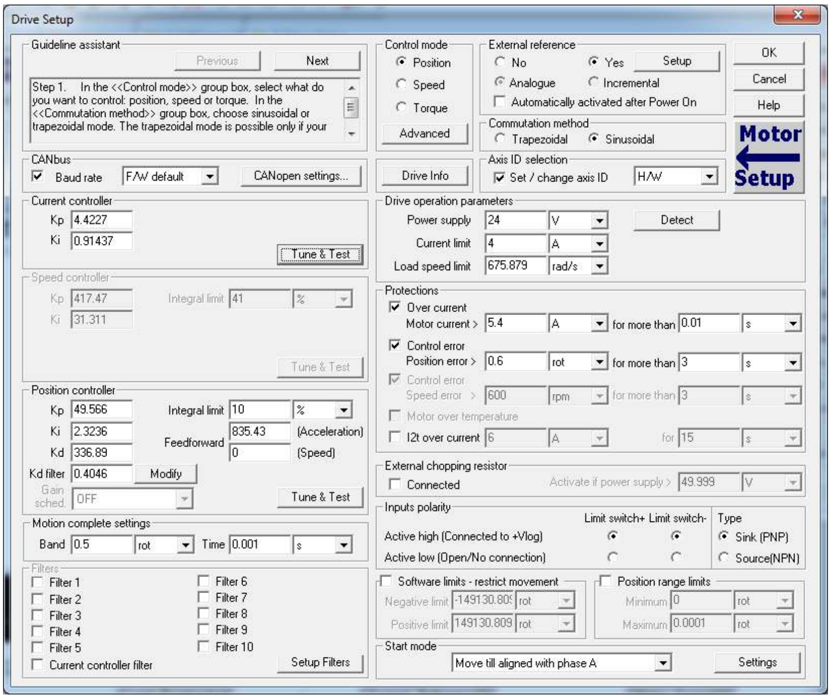


Figure 1.1.4. *EasySetup – Commissioning the drive*

Newer 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.

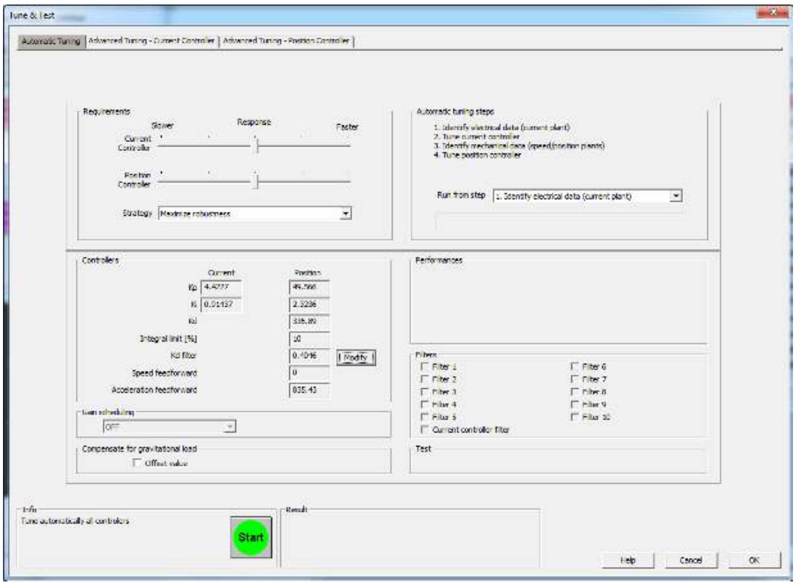
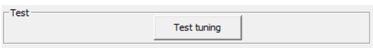


Figure 1.1.5. *EasySetup – Auto tuning interface*

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”.



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

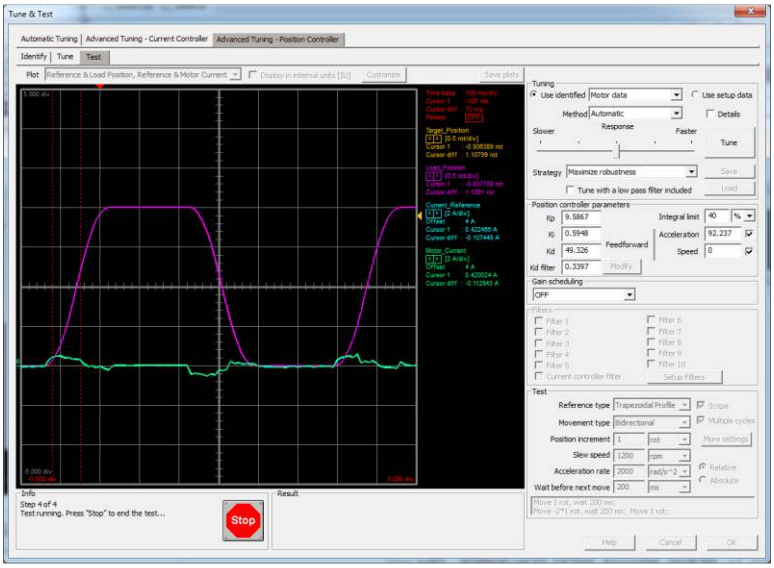


__Figure 1.1.6.__ EasySetup – Testing the motor tuning

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 you have validated your setup, 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 a CANopen master, using the communication objects for writing data into the drive EEPROM (see **Chapter 21** for detailed example)

• 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 a CANopen master

You can program a CANopen master to automatically check after power on if all the Technosoft drives connected to the CAN network have the wright 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 CANopen master. The fastest way to compare a .sw file with the 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 CANopen master. Paragraphs **18.4** and **18.5** present examples on 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

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

1.1.12 TechnoCAN Extension

In order to take full advantage of the powerful Technosoft Motion Language (TML) built into the intelligent drives, Technosoft has developed an extension to CANopen, called TechnoCAN through which TML commands can be exchanged with the drives. Thanks to TechnoCAN, you can inspect or reprogram any of the Technosoft drives from a CANopen network using EasySetup or EasyMotion Studio and an RS-232 link between your PC and any of the drives.

TechnoCAN uses only message identifiers outside of the range used by the CANopen predefined connection set (as defined by CiA DS301 v4.2.0). Thus, TechnoCAN protocol and CANopen protocol can co-exist and communicate simultaneously on the same physical CAN bus, without disturbing each other.

1.2 Changing the drive Axis ID (Node ID)

The axis ID of an iPOS drive can be set in 3 ways:

- Hardware (H/W) – depending on the drive type, it can be via H/W pins or switches.
- Software (via Setup)– any value between 1 and 255, stored in the setup table.
- Software (via CANopen master) – using CiA-305[1] protocol

Remark:

- If the drive is in CANopen mode, a Node ID value above 127 is automatically converted into 255 and the drive is set with CAN communication "non-configured" mode waiting for a CANopen master to configure it using CiA-305 protocol. A "non-configured" drive answers only to CiA-305 commands. All other CANopen commands are ignored and transmission of all other CANopen messages (including boot-up) is disabled. The Ready (green) LED will flash at 1 second time intervals while in this mode.
- In absence of a CANopen master, you can get a drive out from "non-configured" mode, by setting another axis ID between 1 and 127, either by Hardware or by Software (via Setup).

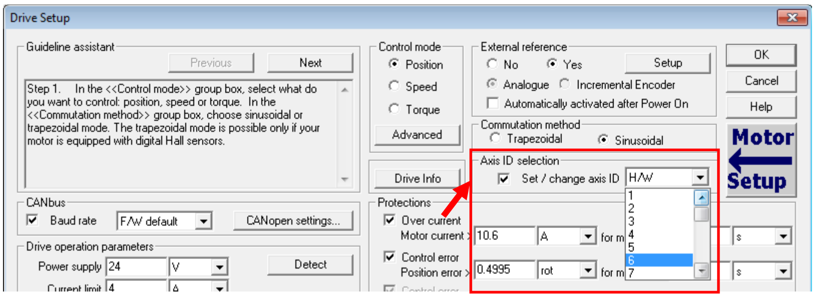


Figure 1.2.1. *EasySetup – Setting the Axis ID*

The axis ID is initialized at power on, using the following algorithm:

a) If a valid setup table exists, and this setup table was created with the *Axis ID Selection* checkbox checked in the Drive Setup dialogue (see above) – with the value read from the setup table. This value can be an axis number 1 to 255 or can indicate that axis ID will be set according with the AxisID inputs levels. If the drive is set in CANopen mode and the Axis ID is over 127 it is converted into 255 and the drive enters in CAN communication "non-configured" mode. The Ready (green) LED will flash at 1 second time intervals while in this mode.

[1] CiA 305 protocol is available only on firmware F514x.

b) If a valid the setup table exists, and this was created with the *Axis ID Selection* checkbox unchecked in the Drive Setup dialogue (see above) – with the last value set either from a valid setup table or by a CANopen master via CiA-305 protocol. This value can be an axis number 1 to 255 for TMLCAN, 1 to 127 for CANopen, or can indicate that axis ID will be set according with the AxisID inputs levels

c) If the setup table is invalid, with the last value set either from a valid setup table or by a CANopen master via CiA-305 protocol. This value can be an axis number 1 to 255 for TMLCAN, 1 to 127 for CANopen, or can indicate that axis ID will be set according with the AxisID inputs levels

d) If the setup table is invalid, there is no previous axis ID set from a valid setup table or by a CANopen master, according with the AxisID inputs levels

***Remark:** If the current drive axis ID is not known, it can be found in the following way:*

a) Connect the drive via a serial RS232 link to a PC where EasySetup or EasyMotion Studio are installed

*b) With the drive powered, open EasySetup or EasyMotion Studio and check the status bar. If communication with the drive is established, the status bar displays **Online** in green and nearby the drive's Axis ID. If the status bar displays **Offline** in red, execute menu command "Communication|Setup…" and in the dialogue opened select at "Channel Type" **RS232** and at "Axis ID of drive/motor connected to PC" the option **Autodetected**. After closing the dialogue with OK, communication with the drive shall be established and the status bar shall display the drive's Axis ID*

*c) If the access to the drive with the unknown Axis ID is difficult, but this drive is connected via CANbus with other Technosoft drives having an easier access, connect your PC serially to one of the other drives. Use EasySetup or EasyMotion Studio menu command **Communication | Scan Network** to find the axis IDs of all the Technosoft drives present in the network.*

1.3 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.

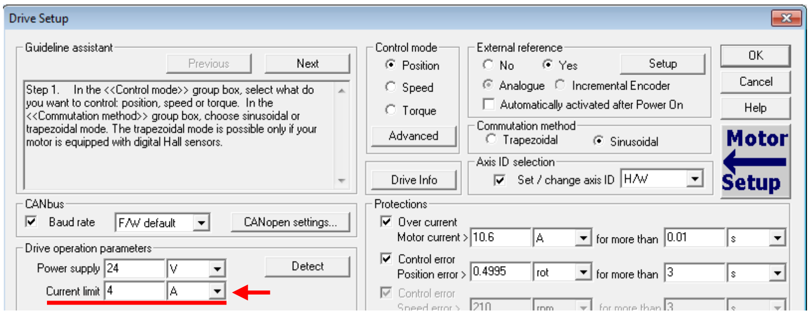


***Figure 1.3.1.** EasySetup – Setting the current limit*

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

1.4 Setting the CAN baud rate

The iPOS drives accept the following CAN baud rates: 125Kbps, 250 Kbps, 500kbps and 1Mbps. Using the Drive Setup dialogue you can choose the initial CAN rate after power on. This information is stored in the setup table The CAN rate is initialized using the following algorithm:

a) If a valid setup table exists, and this setup table was created with the *Set baud rate* checkbox checked in the Drive Setup dialogue (see above) – with the value read from the setup table. This value can be one of the above 4 values or the firmware default (F/W default) which is 500kbs

b) If a valid setup table exists, and this setup table was created with the *Set baud rate* checkbox unchecked in the Drive Setup dialogue (see above) – with the last value set either from a valid setup table or by a CANopen master via CiA-305 protocol

c) If the setup table is invalid, with the last value set either from a valid setup table or by a CANopen master via CiA-305 protocol.

d) If the setup table is invalid, there is no previous CAN rate set from a valid setup table or by a CANopen master, with f/w default value which is 500kbs

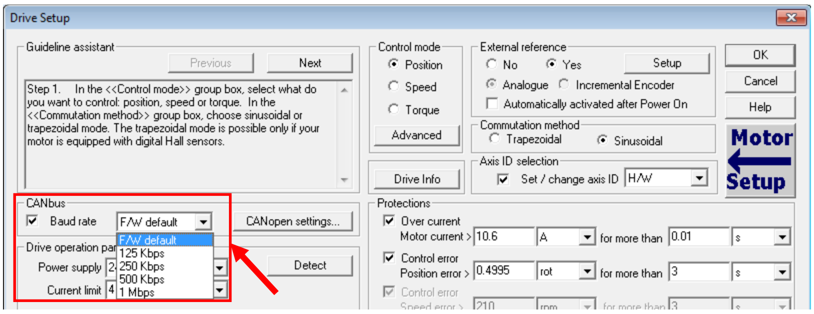


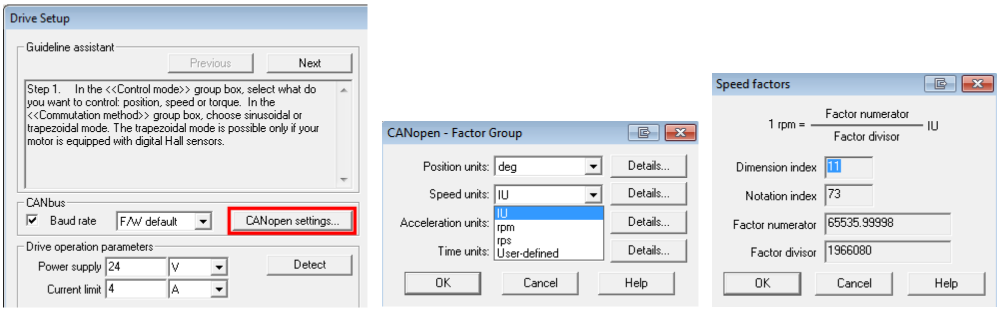
Figure 1.4.1. *EasySetup – Setting the CAN baud rate*

1.5 CANopen factor group setting

The factor group settings currently implemented are complying with:
- CiA-402-2 and later versions – starting with F514K / FA0xx firmware versions
- CiA-402 – for other firmware versions

1.5.1 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 CANopen 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 CANopen 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 CANopen master supports handling the scaling calculations on its side, it is recommended to use them instead of using the "*Factor*" scaling objects.

1.5.2 Factor group setting - CiA-402-2

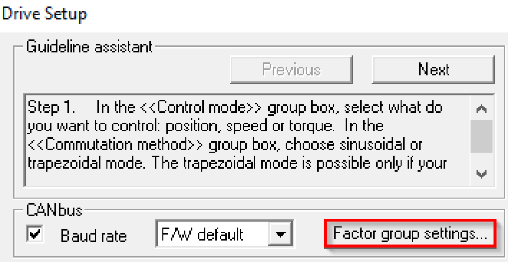
The Factor Group Settings button opens an interface which allows to select the desired physical unit for position, speed, acceleration and jerk values. These settings are linked directly to the objects presented in chapter 6.2.

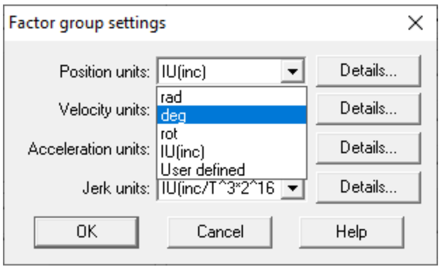
The factor group that complies with CiA-402-2 is available starting with firmware version F514K/FA01A. EasyMotion Studio / EasySetUp must be updated to the latest version. Otherwise, the obsolete version will be displayed.

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

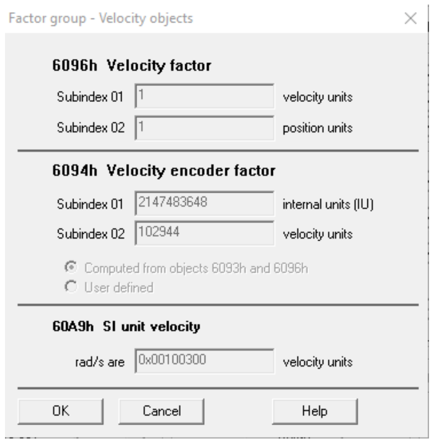
The Factor Group should be adjusted once before any type of movement is realised and not changed during the movement.





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.

1.6 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.

1.6.1 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
- Perform data transfers between axes
- Control motion of an axis from another one via motion commands sent between axes
- Send commands to a group of axes (multicast). This includes the possibility to start simultaneously motion sequences on all the axes from the group
- Synchronize all the axes from a network

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

Chapter 19 describes in detail how the TML features can be combined with the CANopen programming.

2 Layer Setting Services (LSS protocol)[1]

By using layer setting services, the CANopen node-ID and/or the bit timing settings of a LSS slave device may be configured via the CAN network without using any hardware components such as jumpers or DIP-switches. The CANopen device that can configure other devices via CANopen network is called a LSS Master. There must be only one (active) LSS master in a network. The CANopen device that will be configured by the LSS Master via CANopen network is called a LSS Slave.

An LSS Slave can be identified by its unique LSS address. The LSS address consists of the sub objects **Vendor ID**, **Product Code**, **Revision Number** and **Serial Number** of the CANopen "Identity Object" with index 1018_h. In the network, there must not be other LSS Slaves possessing the same LSS address.

With this unique LSS address an individual CANopen device can be allocated within the network. The Node ID is valid if it is in the range of 0x01…0x7F. The value 0xFF indicates not configured CANopen devices.

Communication between LSS Master and LSS Slaves is accomplished by LSS protocols, which use only two COB-IDs:

- LSS master messages from LSS Master to LSS Slaves (COB-ID 0x7E5)
- LSS slave messages from the LSS Slaves to LSS Master (COB-ID 0x7E4).

2.1 Overview

The table below provides an overview on the LSS commands, including details on whether they may be used in states "Waiting" and "Configuration". To change the LSS state, the LSS services **Switch State Global** or **Switch State Selective** may be used.

Table 2.1 *- Drive State Transitions*

Command Specifier (cs)	Services		LSS waiting state	LSS configuration state
0x04	Switch State Global		yes	yes
0x40	Switch state selective procedure	Vendor ID	yes	no
0x41		Product Code	yes	no
0x42		Revision Number	yes	no
0x43		Serial Number	yes	no
0x11	Configure node-ID		no	yes
0x13	Configure bit timing parameters		no	yes
0x15	Activate bit timing parameters		no	yes
0x17	Store configuration		no	yes
0x5A	Inquire LSS address protocol	Identity Vendor ID	no	yes
0x5B		Identity Product Code	no	yes
0x5C		Identity Revision Number	no	yes
0x5D		Identity Serial Number	no	yes
0x5E	Inquire node-ID protocol		no	yes
0x46	Identify remote slave procedure	Vendor ID	yes	yes
0x47		Product Code	yes	yes
0x48		Revision Number Low	yes	yes
0x49		Revision Number High	yes	yes
0x4A		Serial Number Low	yes	yes
0x4B		Serial Number High	yes	yes
0x4C	Identify non-configured Remote Slave		yes	yes

2.2 Configuration services

The LSS configuration services are used to configure the node-ID or bit rate.

2.2.1 Switch State Global

Switches all LSS slave devices in the network into LSS "Waiting" state or LSS "Configuration" state.

The service is unconfirmed.

cs	0x04	Command Specifier for Switch State Global command
mode	0	Switch to LSS state waiting
	1	Switch to LSS state configuration

[1] LSS protocol is available only in the F514x firmware

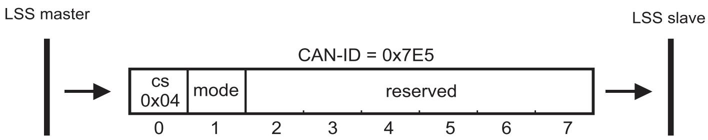


Figure 2.2.1. *LSS – Switch State Global*

2.2.2 Switch State Selective

Changed state of one LSS Slave from “Waiting” to “Configuration”.

LSS command specifier can be:

- 0x40 to submit the Vendor ID,
- 0x41 to submit the Product Code,
- 0x42 to submit the Revision Number,
- 0x43 to submit the Serial Number

To selectively switch a target LSS slave to “Configuration” state, all the Switch State Selective commands must be sent and must contain the same data as found in the “**Identity Object**”, **index 1018$_h$**, of the target drive.

The service is confirmed. The LSS slave sends the command specifier 0x44 meaning it has entered “Configuration” state.

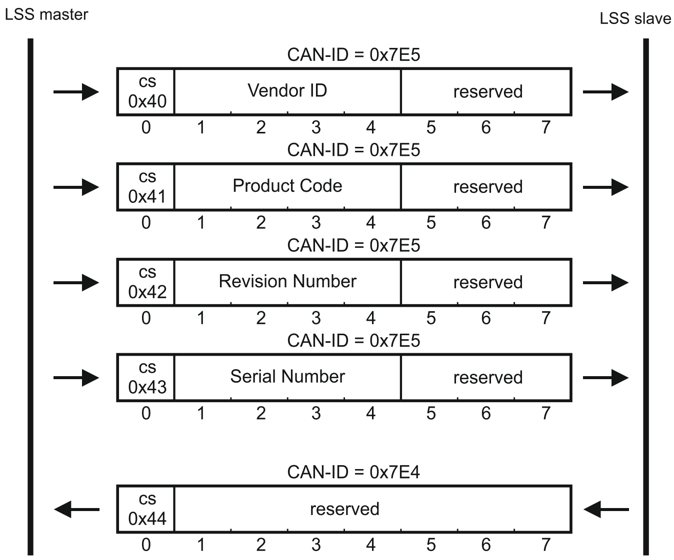


Figure 2.2.2. *LSS – Switch State Selective*

2.2.3 Configure Node ID

Configures the Node ID (of value 1…127 or 255).

The LSS Master can set the LSS Slave’s Node ID only in LSS configuration state. The LSS Master is responsible to switch **a single** LSS Slave into LSS state “Configuration” (with Switch State Selective) before requesting this service. With this service, the LSS Salve’s Node ID can take only values between 1 and 127 (valid Node ID) or 255 (set slave to not-configured).

If the Node ID is set to 255 (0xFF), the LSS slave remains in NMT Initialization sub-state “reset communication” and waits in LSS waiting state for further commands. During this waiting state, the LSS slave is not allowed to send messages, except when LSS replies are needed.

To activate the new node ID, the LSS master has to send the NMT command “Reset communication”. To store the new node ID in the non-volatile memory, the LSS master has to use LSS Store Configuration protocol before resetting the communication or the node.

cs	0x11	Command specifier for configure node-ID protocol
mode	0	Protocol successfully completed
	1	Node ID out of range value
specific error	always 0	

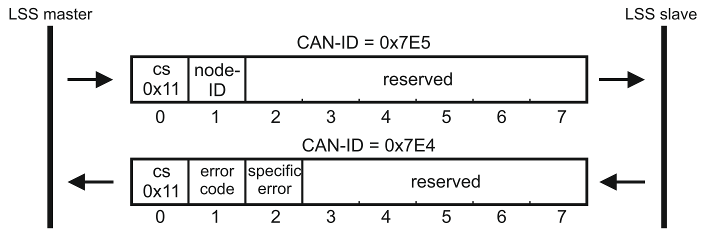


Figure 2.2.3. *LSS – Configure Node ID*

2.2.4 Configure Bit Timing Parameters

By means of the service configure bit timing parameters, the LSS Master can configure new bit timing on a single or multiple LSS Slaves. The new bit timing will be active only after LSS Activate Bit Timing Parameters command or LSS Store Configuration Protocol followed by node reset commands.

cs	0x13	Command specifier for configure bit timing parameters protocol
table selector	always 0	
table index	CAN bit rate codes	
error code	0	Protocol successfully completed
	1	Node ID out of range value
specific error	always 0	

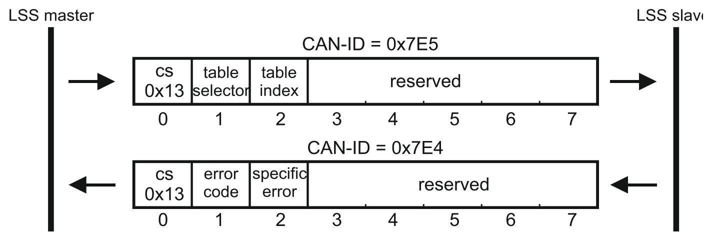


Figure 2.2.4. *LSS – Configure Bit Timing Parameters*

Table 2.2 *– Supported CAN bitrates*

Value	Bit Rate
0	1 Mbit/s
2	500 Kbit/s
3	250 Kbit/s
4	125 Kbit/s

2.2.5 Activate Bit Timing Parameters

Activates bit timing parameters selected with Configure Bit Timing Parameters service.

Switch delay = specifies the duration [in ms] of the two delay periods of equal length. The first period is until the bit timing parameters switch is done. The second period is the time before sending any new CAN message. They are necessary to avoid operating the network with different bit rates.

After receiving an activate bit timing command, the LSS slave stops communication. After the first switch delay, communication is switched to the new bit rate. After the second delay, the LSS slave is allowed to transmit messages with the new bit rate active.

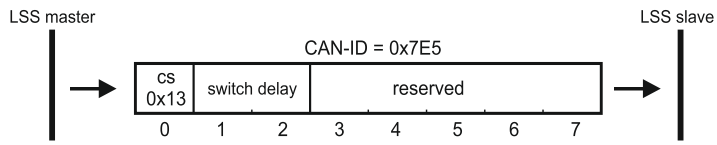


Figure 2.2.5. *LSS – Activate Bit Timing Parameters*

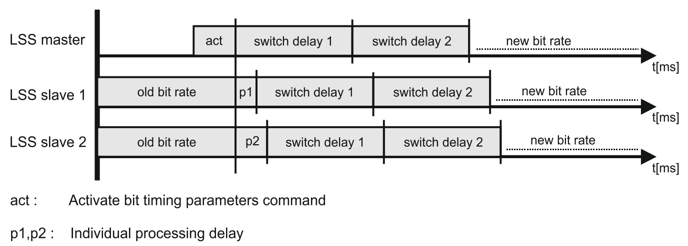


Figure 2.2.6. *LSS – LSS master and LSS slave timings*

2.2.6 Store Configuration Protocol

The pending node-ID and bit rate are copied to the persistent node-ID and bit rate in the non-volatile memory. The result is confirmed by the LSS slave with success or failure message.

cs	0x17	Store Configuration
error code	always 0	
specific error	always 0	

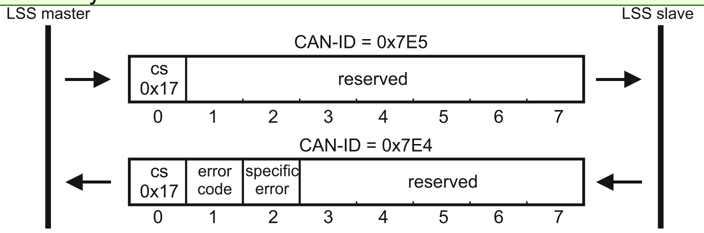


Figure 2.2.7. *LSS – Store Configuration*

2.2.7 Inquire Identity Vendor ID

Reads Vendor ID of LSS slave. The same value can be found in Identity Object, index 1018h, Sub-index 01 of target slave.

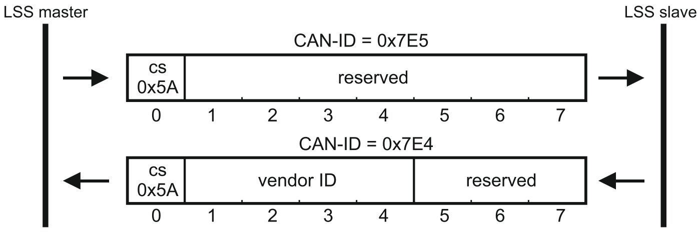


Figure 2.2.8. *LSS – Inquire Identity Vendor ID*

2.2.8 Inquire Identity Product Code

Reads Product Code of LSS slave. The same value can be found in Identity Object, index 1018h, Sub-index 02 of target slave.

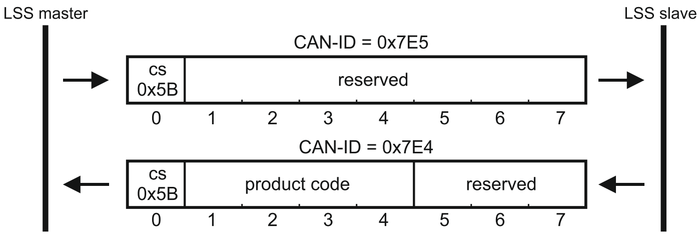


Figure 2.2.9. *LSS – Inquire Identity Product Code*

2.2.9 Inquire Identity Revision Number

Reads Revision Number of LSS slave. The same value can be found in Identity Object, index 1018h Object 1601h: Receive PDO2 Mapping Parameters, Sub-index 03 of target slave.

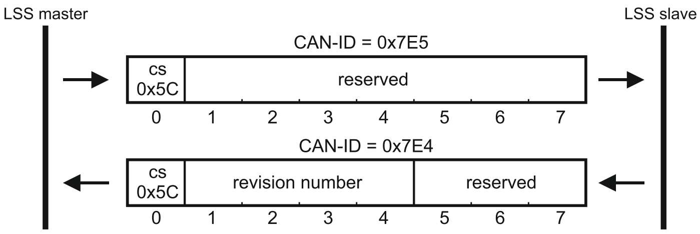


Figure 2.2.10. *LSS – Inquire Identity Revision Number*

2.2.10 Inquire Identity Serial Number

Reads Serial Number of LSS slave. The same value can be found in Identity Object, index 1018h, Sub-index 04 of target slave.

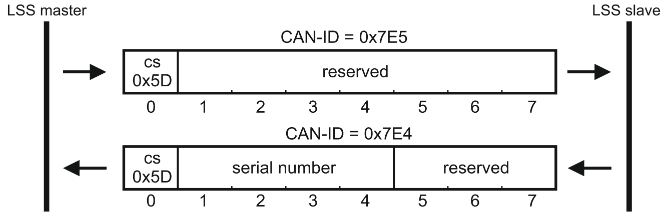


Figure 2.2.11. *LSS – Inquire Identity Serial Number*

2.2.11 Inquire Identity Node ID

Reads active Node ID of LSS slave.

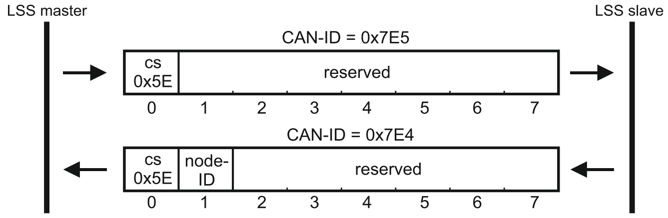


Figure 2.2.12. *LSS – Inquire Identity Node ID*

2.2.12 Identify Remote Slave

Identifies LSS Salves in the CAN network. The LSS master sends identify remote slave commands containing a single Vendor ID, a single Product Code, and a range of Revision Numbers and Serial Numbers. All LSS Slaves that are within these values (including the boundaries) answer with an Identify Remote Slave response (cs=0x4F). An LSS Slave answers, only after all Identify commands are sent and it is within the correct parameters.

With this protocol, a network search can be implemented on the LSS master. With this method, the LSS address range is set to maximum values, and identifies the number of remote slaves in the network. This range will be split in two sub-areas and identify the slaves again. This process will be repeated until all LSS Slaves have been identified.

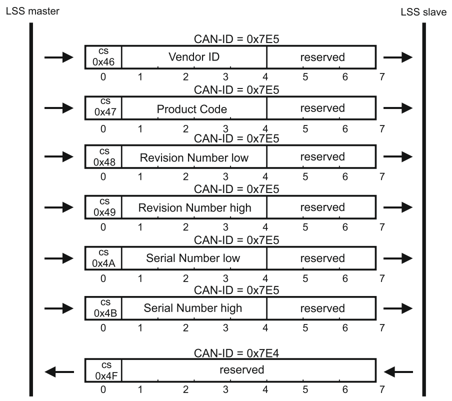


Figure 2.2.13. *LSS – Identify Remote Slave*

2.2.13 Identify non-configured Remote Slave

Allows the LSS master to detect non-configured slave devices in the network. All LSS Slaves without a configured Node ID (0xFF) will answer with a 0x50 command specifier response.

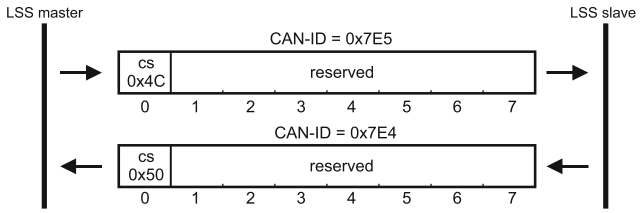


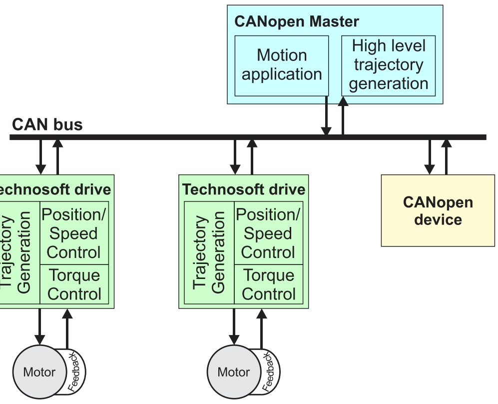
Figure 2.2.14. *LSS – Identify non-configured Remote Slave*

3 CAN and the CANopen protocol

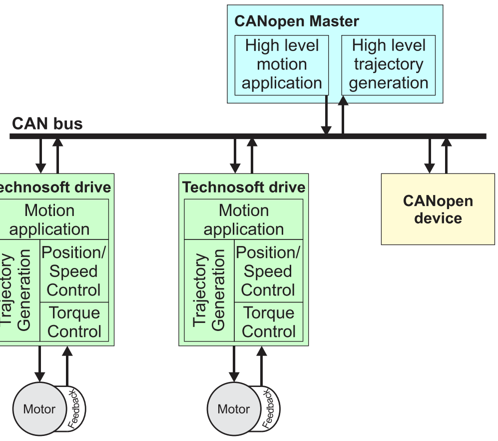
CAN (Controller Area Network) is a serial bus system used in a broad range of automation control systems. The CAN specifies the data link and the physical connection over which lays the CANopen, a high level protocol specifying how various types of devices can use the CAN network.

3.1 CAN Architecture

CAN provides distributed control of the motion application, the control loops are closed locally not on the master controller. The master controller coordinates multiple devices through the commands it sends and receives information about the status of the devices.



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



3.2 Accessing CANopen devices

A CANopen device is controlled through read/write operations to/from objects performed by a CANopen master (PC or PLC).

3.2.1 Object dictionary

The Object Dictionary is a group of objects that describes the complete functionality of a device by way of communication objects and it 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.

3.2.2 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.

3.2.3 Service Data Objects (SDO)

Service Data Objects are used by CANopen master to access any object from the drive's Object Dictionary. Both expedited and segmented SDO transfers are supported (see DS301 v4.2.0 for details). 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 in **Table 3.2.1**.

Table 3.1 *– SDO Abort Codes*

Abort code	Description
0503 0000h	Toggle bit not alternated
0504 0001h	Client/server command specifier not valid or unknown
0601 0000h	Unsupported access to an 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
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
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

3.2.4 Process Data Objects (PDO)

Process Data Objects are used for high priority, real-time data transfers between CANopen 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 have 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.

Two objects define a PDO: the communication object and the mapping object. The communication object defines the COB-ID of the PDO, the transmission type and the event triggering the transmission. The mapping object contains the descriptions of the objects mapped into the PDO, i.e. the index, sub-index and size of the mapped objects.

The following PDO transmission modes are distinguished:

• Synchronous transmission - the PDO is transmitted after the SYNC. In case it is cyclic the sampling is started with the reception of every SYNC, every 2nd SYNC, every 3rd SYNC, and s.o. depending on the given value and the PDO is transmitted afterwards.
• Event-driven transmission (asynchronous) - the PDO will be sent every time anything changes in its data field.
• RTR-only transmission - the PDO is not transmitted normally it shall be requested via RTR. In case it is synchronous the device will start sampling with every SYNC and then will buffer the PDO when a RTR request is received. In case it is event-driven the device will start sampling with the reception of the RTR and will transmit the PDO immediately.

Table 3.2 *– Transmission type*

Value	Description
00h	Reserved.
01h	synchronous (cyclic every SYNC))
02h	synchronous (cyclic every 2nd SYNC)
03h	synchronous (cyclic every 3rd SYNC)
04h	synchronous (cyclic every 4th SYNC)
…	…
F0h	synchronous (cyclic every 240th SYNC)
F1h	Reserved.
…	…
FBh	Reserved.
FCh	RTR-only (synchronous)
FDh	RTR-only (event-driven)
FEh	Event-driven (asynchronous)
FFh	

The inhibit time is the minimum interval for PDO transmission if the transmission type is set to FE_h and FF_h. This means that even though the PDO should be sent faster, it will be sent at minimum inhibit time intervals. The value is defined as multiple of 100 μs, the value 0 disabling the inhibit time.

The event timer is the maximum interval for PDO transmission if the transmission type is set to FE_h and FF_h. This means that even if nothing changes in its data field, the PDO will be sent at event timer intervals. The value is defined as multiple of 1 ms, the value 0 disabling the event timer.

3.3 Objects that define SDOs and PDOs

3.3.1 Object 1200_h: Server SDO Parameter

The object contains the COB-IDs of the messages used for the SDO protocol. The COBID of the SDO packages received by the drive, stored in sub-index 01, is computed as 600_h + drive Node ID. The COB ID of the SDO packages sent by the drive, stored in sub-index 02, is computed as 580_h + drive Node ID.

Object description:

Index	1200_h
Name	Server SDO Parameter
Object code	RECORD
Data type	SDO Parameter

Entry description:

Sub-index	00_h
Description	Number of entries
Access	RO
PDO mapping	No
Value range	2
Default value	2

Sub-index	01_h
Description	SDO receive COB-ID
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	600_h + Node-ID

Sub-index	02_h
Description	SDO transmit COB-ID
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	580_h + Node-ID

3.3.2 Object 1400_h: Receive PDO1 Communication Parameters

The object contains the communication parameters of the receive PDO1. Sub-index 1_h contains the COB ID of the PDO. The transmission type (sub-index 2_h) defines the reception character of the PDO.

Object description:

Index	1400_h
Name	RPDO1 Communication Parameter
Object code	RECORD
Data type	SDO Parameter

Entry description:

Sub-index	00_h
Description	Number of entries
Access	RO
PDO mapping	No
Value range	-
Default value	2

Sub-index	01_h
Description	COB-ID RPDO1
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	200_h + Node-ID

Sub-index	02_h
Description	Transmission type
Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	255

Table 3.3 *– PDO COB-ID entry description*

Bit	Value	Meaning
31	0	PDO exists / is valid / is enabled
	1	PDO does not exist / is not valid / is disabled
30	0	RTR allowed on this PDO
	1	No RTR allowed on this PDO
29	0	11 bit ID
	1	29 bit ID
28…11	0	If bit 29=0
	X	If bit 29=1: Bit 11...28 of 29-bit PDO COB-ID
10...0	X	Bit 0...10 of PDO COB-ID

It is not allowed to change bits 0-29 while the PDO exists (bit 31=0).

3.3.3 Object 1401_h: Receive PDO2 Communication parameters

The object contains the communication parameters of the receive PDO2. Sub-index 1_h contains the COB ID of the PDO. The transmission type (sub-index 2_h) defines the reception character of the PDO. The receive PDO2 COB-ID entry description is identical with the one of the receive PDO1 (see **Table 3.3.1**).

Object description:

Index	1401_h
Name	RPDO2 Communication Parameter
Object code	RECORD
Data type	SDO Parameter

Entry description:

Sub-index	00_h
Description	Number of entries
Access	RO
PDO mapping	No
Value range	-
Default value	2

Sub-index	01_h
Description	COB-ID RPDO2
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	300_h + Node-ID

Sub-index	02_h
Description	Transmission type
Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	255

3.3.4 Object 1402_h: Receive PDO3 Communication parameters

The object contains the communication parameters of the receive PDO3. Sub-index 1_h contains the COB ID of the PDO. The transmission type (sub-index 2_h) defines the reception character of the PDO. The receive PDO3 COB-ID entry description is identical with the one of the receive PDO1 (see **Table 3.3.1**).

Object description:

Index	1402_h
Name	RPDO3 Communication Parameter
Object code	RECORD
Data type	SDO Parameter

Entry description:

Sub-index	00_h
Description	Number of entries
Access	RO
PDO mapping	No
Value range	-
Default value	2

Sub-index	01_h
Description	COB-ID RPDO3
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	400_h + Node-ID

Sub-index	02_h
Description	Transmission type
Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	255

3.3.5 Object 1403_h: Receive PDO4 Communication parameters

The object contains the communication parameters of the receive PDO4. Sub-index 1_h contains the COB ID of the PDO. The transmission type (sub-index 2_h) defines the reception character of the PDO. The receive PDO4 COB-ID entry description is identical with the one of the receive PDO1 (see **Table 3.3.1**).

Object description:

Index	1403_h
Name	RPDO4 Communication Parameter
Object code	RECORD
Data type	SDO Parameter

Entry description:

Sub-index	00_h
Description	Number of entries
Access	RO
PDO mapping	No
Value range	-
Default value	2

Sub-index	01_h
Description	COB-ID RPDO2
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	500_h + Node-ID

Sub-index	02_h
Description	Transmission type
Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	255

3.3.6 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. It is possible to map up to eight objects, each with a size of one byte, resulting in a total of 64 bits.

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	1

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

3.3.7 Object 1601$_h$: Receive PDO2 Mapping Parameters

This object contains the mapping parameters of the receive PDO2. 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. It is possible to map up to eight objects, each with a size of one byte, resulting in a total of 64 bits.

Object description:

Index	1601$_h$
Name	RPDO2 Mapping Parameter
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

Sub-index	02h
Description	2nd mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60600008h – modes of operation

3.3.8 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. It is possible to map up to eight objects, each with a size of one byte, resulting in a total of 64 bits.

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	60400010h – Controlword

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

3.3.9 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. It is possible to map up to eight objects, each with a size of one byte, resulting in a total of 64 bits.

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	60400010h – Controlword

Sub-index	02h
Description	2nd mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60FF0020h – target velocity

3.3.10 Object 1800h: Transmit PDO1 Communication parameters

This object contains the communication parameters of the transmit PDO1. For detailed description see object 1400h (Receive PDO1 communication parameters, COB-ID entry description, described in **Table 3.3.1**). The inhibit time is defined as multiples of 100 µs.

Object description:

Index	1800h
Name	TPDO1 Communication Parameters
Object code	RECORD
Data type	SDO Parameter

Entry description:

Sub-index	00h
Description	Number of entries
Access	RO
PDO mapping	No
Value range	-
Default value	5

Sub-index	01h
Description	COB-ID TPDO1
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	180h + Node-ID

Sub-index	02h
Description	Transmission type
Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	255

Sub-index	03h
Description	Inhibit time
Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	300 (30 ms)

Sub-index	04h
Description	Reserved

Sub-index	05h
Description	Event timer
Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	0

3.3.11 Object 1801$_h$: Transmit PDO2 Communication parameters

This object contains the communication parameters of the transmit PDO2. For detailed description see object 1400$_h$ (Receive PDO1 communication parameters, COB-ID entry description, described in **Table 3.3.1**). The inhibit time is defined as multiples of 100 µs.

Object description:

Index	1801$_h$
Name	TPDO2 Communication Parameters
Object code	RECORD
Data type	SDO Parameter

Entry description:

Sub-index	00$_h$
Description	Number of entries
Access	RO
PDO mapping	No
Value range	-
Default value	5

Sub-index	01$_h$
Description	COB-ID TPDO2
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	280$_h$ + Node-ID

Sub-index	02$_h$
Description	Transmission type
Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	255

Sub-index	03$_h$
Description	Inhibit time
Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	300 (30 ms)

Sub-index	04$_h$
Description	Reserved

Sub-index	05$_h$
Description	Event timer
Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	0

3.3.12 Object 1802$_h$: Transmit PDO3 Communication parameters

This object contains the communication parameters of the transmit PDO3. By default, this TxPDO is disabled by setting Bit31 to 1$_b$ in Sub-index 01$_h$. For detailed description see object 1400$_h$ (Receive PDO1 communication parameters, COB-ID entry description, described in **Table 3.3.1**). The inhibit time is defined as multiples of 100 µs.

Object description:

Index	1802$_h$
Name	TPDO3 Communication Parameters
Object code	RECORD
Data type	SDO Parameter

Entry description:

Sub-index	00$_h$
Description	Number of entries
Access	RO
PDO mapping	No
Value range	-
Default value	5

Sub-index	01_h
Description	COB-ID TPDO3
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	80000380_h + Node-ID

Sub-index	02_h
Description	Transmission type
Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	255

Sub-index	03_h
Description	Inhibit time
Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	300 (30 ms)

Sub-index	04_h
Description	Reserved

Sub-index	05_h
Description	Event timer
Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	0

3.3.13 Object 1803_h: Transmit PDO4 Communication parameters

This object contains the communication parameters of the transmit PDO4. By default, this TxPDO is disabled by setting Bit31 to 1_b in Sub-index 01_h. For detailed description see object 1400_h (Receive PDO1 communication parameters, COB-ID entry description, described in **Table 3.3.1**) . The inhibit time is defined as multiples of 100 µs.

Object description:

Index	1803_h
Name	TPDO4 Communication Parameter
Object code	RECORD
Data type	SDO Parameter

Entry description:

Sub-index	00_h
Description	Number of entries
Access	RO
PDO mapping	No
Value range	-
Default value	5

Sub-index	01_h
Description	COB-ID TPDO4
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	80000480_h + Node-ID

Sub-index	02_h
Description	Transmission type
Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	255

Sub-index	03_h
Description	Inhibit time
Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	300 (30 ms)

Sub-index	04_h
Description	Reserved

Sub-index	05_h
Description	Event timer
Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	0

3.3.14 Object $1A00_h$: Transmit PDO1 Mapping Parameters

This object contains the mapping parameters of the transmit PDO1. For detailed description see object 1600_h (Receive PDO1 mapping parameters). It is possible to map up to eight objects, each with a size of one byte, resulting in a total of 64 bits.

Object description:

Index	$1A00_h$
Name	TPDO1 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	1

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

3.3.15 Object $1A01_h$: Transmit PDO2 Mapping Parameters

This object contains the mapping parameters of the transmit PDO2. For detailed description see object 1600_h (Receive PDO1 mapping parameters). It is possible to map up to eight objects, each with a size of one byte, resulting in a total of 64 bits.

Object description:

Index	$1A01_h$
Name	TPDO2 Mapping Parameter
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	60410010_h – Statusword

Sub-index	02_h
Description	2nd mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60610008_h – modes of operation display

3.3.16 Object 1A02h: Transmit PDO3 Mapping Parameters

This object contains the mapping parameters of the transmit PDO3. For detailed description see object 1600_h (Receive PDO1 mapping parameters). By default, this PDO is disabled with object 1802_h Sub-index 01 by setting Bit31 to 1. It is possible to map up to eight objects, each with a size of one byte, resulting in a total of 64 bits.

Object description:

Index	$1A02_h$
Name	TPDO3 Mapping Parameter
Object code	RECORD
Data type	PDO Mapping

Entry description:

Sub-index	00_h
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	2

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

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

3.3.17 Object 1A03h: Transmit PDO4 Mapping Parameters

This object contains the mapping parameters of the transmit PDO4. For detailed description see Object 1600_h: Receive PDO1 Mapping Parameters. By default, this PDO is disabled with object 1803_h Sub-index 01 by setting Bit31 to 1. It is possible to map up to eight objects, each with a size of one byte, resulting in a total of 64 bits.

Object description:

Index	$1A03_h$
Name	TPDO4 Mapping Parameter
Object code	RECORD
Data type	PDO Mapping

Entry description:

Sub-index	00_h
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	2

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

Sub-index	02_h
Description	2nd mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	$606C0020_h$ – velocity actual value

3.3.18 Object $207D_h$: Dummy

This object may be used to fill a RPDO up to a length matching the CANopen 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

3.4 Dynamic mapping of the PDOs

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

Disable (destroy) the PDO by setting bit *valid* (Bit31) to 1b of sub-index 01_h of the according PDO communication parameter object (index 1400_h-1403_h for RxPDOs and 1800_h-1803_h for TxPDOs). The PDO COB-ID entry description is described in **Table 3.3.1.**

Disable mapping. 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 00_h (the number of mapped objects) to 00_h.

Map the new objects. Write in the PDOs 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.

Maximum PDO size. You can map up to 8 objects, each 1 byte in size, for a total of 64 bits.

Enable mapping. In sub-index 0 of the PDOs associated mapping object (index 1600_h-1603_h for RxPDOs and $1A00_h$-$1A03_h$ for TxPDOs) write the number of mapped objects.

Enable (create) the PDO by setting bit *valid* (Bit31) to 0b of sub-index 01_h of the according PDO communication parameter object (index 1400_h-1403_h for RxPDOs and 1800_h-1803_h for TxPDOs).

3.5 RxPDOs mapping example

Map the Receive PDO3 of axis number 06 with **Controlword** (index 6040_h) and **Modes of Operation** (index 6060_h).

1. **Disable the RxPDO**. Set Bit31 to 1b of sub-index 01_h in object 1402_h, this will disable the RxPDO. The PDO COB-ID entry description is described in **Table 3.3.1.**

Bit31 valid	RxPDO3 COB-ID	Axis Node ID	Resulting data
1_b +	**400_h +**	**06_h =**	**80000406_h**

Send the following message (SDO access to object 1402_h sub-index 1, 32-bit value 80000406_h):

COB-ID	Data
606	23 02 14 01 06 04 00 80

2. **Change the communication parameters.** For example purposes the communication parameters default values are acceptable.
3. **Disable mapping PDO**. Write zero 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):

COB-ID	Data
606	2F 02 16 00 00 00 00 00

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):

COB-ID	Data
606	23 02 16 01 10 00 40 60

   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):

COB-ID	Data
606	23 02 16 02 08 00 60 60

5. **Enable the RxPDO 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):

COB-ID	Data
606	2F 02 16 00 02 00 00 00

6. **Enable the RxPDO**. Set Bit31 to 0b of sub-index 01_h in object 1402_h, this will enable the RxPDO. Set in object 1402_h sub-index 1 Bit31 to 0. The PDO COB-ID entry description is described in **Table 3.3.1.**

Bit31 valid	RxPDO3 COB-ID	Axis Node ID	Resulting data
0_b +	400_h +	06_h =	00000406_h

Send the following message (SDO access to object 1402_h sub-index 1, 32-bit value 0x00000406):

COB-ID	Data
606	23 02 14 01 06 04 00 00

3.6 TxPDOs mapping example

Map the Transmit PDO4 of axis number 06 with **Position actual value** (index 6064_h) and **Digital inputs** (index $60FD_h$).

Disable the TxPDO. Set Bit31 to 1b of sub-index 01_h in object 1803_h, this will disable the TxPDO. The PDO COB-ID entry description is described in **Table 3.3.1.**

Bit31 valid	TxPDO4 COB-ID	Axis Node ID	Resulting data
1_b +	480_h +	06_h =	80000486_h

Send the following message (SDO access to object 1803_h sub-index 1, 32-bit value 80000486_h):

COB-ID	Data
606	23 03 18 03 86 04 00 80

Set the transmission type. Write 255 in object 1803_h sub-index 2. This will set the transmission type as asynchronous, meaning that the PDO will be sent every time anything changes in its data field.

Send the following message (SDO access to object 1803_h sub-index 2, 8-bit value FF_h):

COB-ID	Data
606	2F 03 18 02 FF 00 00 00

Set inhibit time. Write 1000 in object 1803h sub-index 3. This will set an inhibit time of 100ms. This means that even though the PDO should be sent faster, it will be sent at minimum 100ms intervals.

Send the following message (SDO access to object 1803h sub-index 3, 16-bit value 03E8h):

COB-ID	Data
606	2B 03 18 03 E8 03 00 00

Set event timer. Write 1000 in object 1803h sub-index 5. This will set an event timer of 1000 ms. This means that the PDO will be sent at 1000ms intervals, even if nothing changes in its data field.

Send the following message (SDO access to object 1803h sub-index 5, 16-bit value 03E8h):

COB-ID	Data
606	2B 03 18 05 E8 03 00 00

Disable the PDO mapping. Write zero in object 1A03h sub-index 0, this will disable the PDO's mapping.

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

COB-ID	Data
606	2F 03 1A 00 00 00 00 00

Map the new objects.

a. Write in object 1A03h 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):

COB-ID	Data
606	23 03 1A 01 20 00 64 60

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):

COB-ID	Data
606	23 03 1A 02 20 00 FD 60

Enable the TxPDO 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):

COB-ID	Data
606	2F 03 1A 00 02 00 00 00

Enable the TxPDO 4. Set Bit31 to 0b of sub-index 01h in object 1803h, this will enable the TxPDO 4. Set in object 1803h sub-index 1 Bit31 to 0. The PDO COB-ID entry description is described in **Table 3.3.1.**

Bit31 valid	TxPDO4 COB-ID	Axis Node ID	Resulting data
0b +	480h +	06h =	00000486h

Send the following message (SDO access to object 1803h sub-index 1, 32-bit value 0x00000486):

COB-ID	Data
606	23 03 18 01 86 04 00 00

Start remote node 6. Send a NMT message to start the node id 6. This message is to enable the use of the PDOs.

Send the following message:

COB-ID	Data
0	01 06

After the last message, the drive will start emitting at 1s intervals data with COB-ID 0x486 showing the motor actual position and the Digital input status. If the encoder is rotated, the PDO will be sent every time the position changes, but not faster than 100ms.

4 Network Management

4.1 Overview

The Network Management (NMT) services initialize, start, monitor, reset or stop the CANopen nodes. The NMT requires a node in the network (a PC or a PLC) to be designed as a network manager while the Technosoft intelligent drives are the NMT slaves. The NMT services are fulfilled by the NMT objects described later in this chapter.

4.1.1 Network Management (NMT) State Machine

Figure 4.1.1 shows the NMT state diagram of a CANopen device. After finishing the initialization, the iPOS drive enters the NMT state Pre-operational. During this state, both the communication parameters and drive parameters can be changed using SDO messages. In this state, the PDO messages are defined. Once entered in the operational mode, the drive is typically controlled via PDO messages.

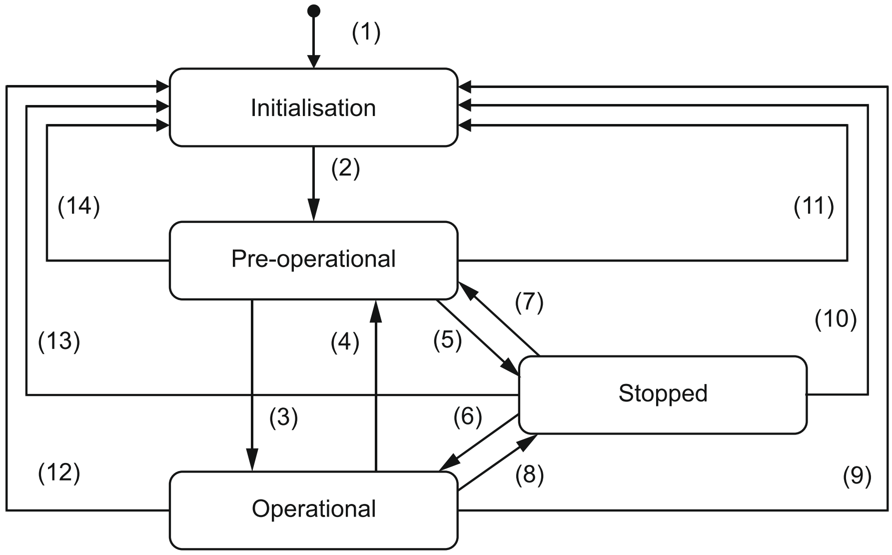


Figure 4.1.1. *NMT state diagram*

Table 4.1 *– NMT state transitions*

(1)	At Power on the NMT state initialization is entered autonomously
(2)	NMT state initialization finished - enter NMT state Pre-operational automatically
(3)	NMT service start remote node indication or by local control
(4),(7)	NMT service enter pre-operational indication
(5),(8)	NMT service stop remote node indication
(6)	NMT service start remote node indication
(9),(10),(11)	NMT service reset node indication
(12),(13),(14)	NMT service reset communication indication

4.1.2 Device control

Through Module Control Services, the NMT master controls the state of the NMT slaves. The following states are implemented on the Technosoft drives:

State	Description
Pre-operational	The drive enters the pre-operational state after finishing its initialization. In this state the communication between the CANopen master and the drive can be done only via SDOs. PDOs are not allowed.
Operational	This is the normal operating state of the drives. The communication through SDO and PDO is allowed
Stopped	In this state, the drive stops the communication except the network management messages.

The network manager can change the state of the drives using one of the following services:

Service	Description
Start Remote Node	The NMT master sets the state of the selected NMT slave to operational
Stop Remote Node	The NMT master sets the state of the selected NMT slave to stopped

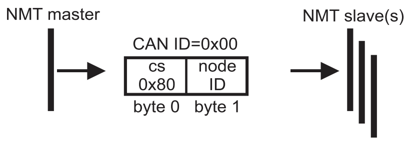
Enter Pre-Operational	The NMT master sets the state of the selected NMT slave to pre-operational
Reset Node	The NMT master sets the state of the selected NMT slave to the “reset application” sub-state. In this state, the drives perform a software reset and enter the pre-operational state.
Reset Communication	The NMT master sets the state of the selected NMT slave to the “reset communication” sub-state. In this state the drives resets their communication and enter the pre-operational state.

All the services are unconfirmed.

4.1.2.1 Enter Pre-Operational

Used to change NMT state of one or all NMT slaves to “Pre-Operational”.

cs	0x80	Command specifier for NMT command Enter Pre-Operational
Node ID	1…127	NMT slave with corresponding Node ID will enter in NMT state Pre-Operational
	0	All NMT Slaves will enter NMT state Pre-Operational



Example for Axis 6. Enter Pre-Operational.

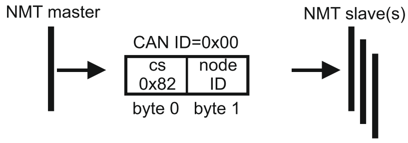
COB-ID	Data
0	80 06

Figure 4.1.2. *NMT Enter Pre-Operational*

4.1.2.2 Reset communication

Used to reset communication of one or all NMT slaves.

cs	0x82	Command specifier for NMT command Reset Communication
Node ID	1…127	NMT slave with corresponding Node ID will reset communication
	0	All NMT Slaves will reset communication



Example for Axis 6. Reset communication.

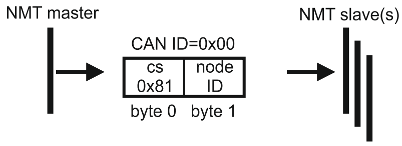
COB-ID	Data
0	82 06

Figure 4.1.3. *NMT Reset Communication*

4.1.2.3 Reset Node

Used to reset one or all NMT slaves.

cs	0x81	Command specifier for NMT command Reset Node
Node ID	1…127	NMT slave with corresponding Node ID will reset
	0	All NMT Slaves will reset



Example for Axis 6. Reset node.

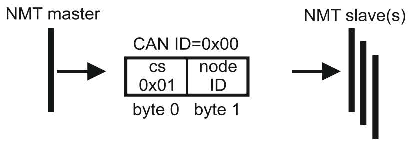
COB-ID	Data
0	81 06

Figure 4.1.4. *NMT Reset Node*

4.1.2.4 Start Remote Node

Used to change NMT state of one or all NMT slaves to “Operational”. PDO communication will be allowed.

cs	0x01	Command specifier for NMT command Start Remote Node
Node ID	1…127	NMT slave with corresponding Node ID will enter “Operational” state
	0	All NMT Slaves will enter “Operational” state



Example for Axis 6. Start Remote Node.

COB-ID	Data
0	01 06

Figure 4.1.5. *NMT Start Remote Node*

4.1.2.5 Stop Remote Node

Used to change NMT state of one or all NMT slaves to “Stopped”.

cs	0x02	Command specifier for NMT command Stop Remote Node
Node ID	1…127	NMT slave with corresponding Node ID will enter “Stopped” state
	0	All NMT Slaves will enter “Stopped” state

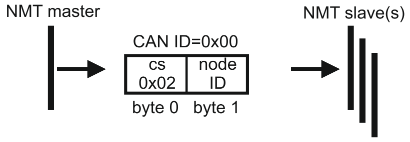


Figure 4.1.6. *NMT Stop Remote Node*

Example for Axis 6. Stop Remote Node.

COB-ID	**Data**
0	**02 06**

4.1.3 Device monitoring

In addition to controlling the drive states, the NMT provides services for monitoring the nodes in the network. The monitoring services are achieved mainly through the periodical transmission of messages by the network manager, with answers from the slaves, or messages sent by the slaves without master intervention. Monitoring services can use the Node Guarding protocol (including Life Guarding) or the Heartbeat protocol.

4.1.3.1 Node guarding protocol

The master polls each NMT slave at regular time intervals. This time interval is called the guard time and may be different for each NMT slave. The slaves answer with a node-guarding message containing their state. This allows both the master and the slave to identify a network error if either the remote request or the guarding messages stop.
The node life time is computed as the product between the guard time (index 100C$_h$) and the life time factor (index 100D$_h$). If the drive is not accessed within the life time then a Life Time event occurs and an emergency telegram is sent.

4.1.3.2 Heartbeat protocol

The Heartbeat protocol defines an error control service without the need of remote frames. It implies independent and cyclical transmission of a telegram by the drive (the Heartbeat producer) indicating the drives current state. The time interval between two heartbeat messages is specified through producer heartbeat time (index 1017$_h$). The master (Heartbeat consumer) guards the reception of the heartbeat messages within the Heartbeat Consumer Time. If the value of this object is 0, the heartbeat transmission is disabled. If the master does not receive the heartbeat message this indicates a problem with the drive or with its network connection.

4.1.3.3 Boot-up protocol

This protocol is used by the drive to signal to the network master that it has entered the state pre-operational. When the drive is powered on for the time or is reset, it will send a boot-up message with the COB-ID (0x700+ Node Id) and Data 00.

4.1.3.4 Synchronization between devices

The synchronization message (SYNC with COB ID 0x80 and no Data) allows synchronizing the devices in the network and triggering the synchronous transmission of PDOs. The SYNC producer broadcasts the synchronization message periodically. This service is unconfirmed. Technosoft intelligent drives can act both as SYNC consumer and producer.
There are two ways to synchronize the drive in a network:
1. Send only the sync message with the COB ID 0x80 and Data null at very precise intervals. This method is the most commonly used and its accuracy is based on how precise the master sends the SYNCS and the CAN bus load
2. For time critical applications, which require more accurate synchronization, the Technosoft drives can use the optional high-resolution synchronization protocol, which employs a special form of time stamp message. The High Resolution Time Stamp can be set with the COB ID 0x100 and 4 bytes of data that represent a time stamp with a resolution of 1µs. When the master sends a time stamp with the COB ID 0x100 it has the same effect as writing the same value to all the slaves in the network in object 1013 $_h$. With this second method, the master sends the sync message (0x80) followed immediately by the time stamp message with the id 0x100.
When one of the Technosoft drives is set as synchronization master, the High resolution time stamp is by default sent using the COB ID defined in COB-ID High Resolution Time Stamp object (index 2004$_h$).

4.1.4 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 4.1.2**. Details regarding the conditions that may generate emergency messages are presented at object Motion Error Register index 2000$_h$.

***Table 4.2** – Emergency Error Codes*

Error code (hex)	Description
0000	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
3210	DC-link over-voltage
3220	DC-link under-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; this emergency message also contains other data; see its description at the end of this table
8110	CAN overrun (message lost)
8130	Life guard error or heartbeat error
8331	I2t protection triggered
8580	Position wraparound
8611	Control error / Following error
9000	Command error
FF01	Generic interpolated position mode error (PVT / PT error); this emergency message also contains other data; see its description at the end of this table
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 circuit hardware error
FF09	Trying to write data to EEPROM while its locked

4.1.4.1 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

0x7500 Communication error:

0 1	2	3 4	5 7
Emergency Error Code	Error Register (Object 1001_h)	Communication Error Register (Object 2003_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 status (Object 2072_h)	Manufacturer specific error field

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

4.2 Network management objects

The section describes the objects related to network management

4.2.1 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	Yes[1]
Value range	UNSIGNED8
Default value	No

***Table 4.3** – 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.

4.2.2 Object 1003h: Pre-defined error field

This object provides the errors that occurred on the iPOS drive and were signaled via the emergency object. If no error was signaled, sub-index 00h reports 0 entries. The object can report up to 5 emergency messages recently transmitted. The last reported error will always be set in sub-index 1.

Object description:

Index	1003h
Name	Pre-defined error field
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	00h
Description	Number of errors in history
Access	RO
PDO mapping	No
Value range	1..5
Default value	0

Sub-index	01h
Description	Standard error field
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	-

Sub-index	02h to 05h
Description	Standard error field
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	-

[1] Object 1001h is PDO mappable starting with F514K firmware version.

4.2.3 Object 1005h: COB-ID of the SYNC Message

This object defines the COB-ID of the Synchronization Object (SYNC) and whether the drive generates the SYNC or not.

Object description:

Index	1005h
Name	COB-ID SYNC Message
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	80h

The structure of the parameter is the following:

***Table 4.4** – Bit description of object 1005h*

Bit	Value	Description
31	X	Reserved
30	0	Drive does not generate synchronization messages
	1	Drive is the synchronization master (SYNC producer)
29	0	Use 11 bit identifier
	1	Use 29 bit identifier
28…11	X	Bit 11...28 of 29-bit SYNC COB-ID
10...0	X	Bit 0...10 of SYNC COB-ID

The first transmission of SYNC object starts within 1 sync cycle after setting bit 30 to 1. It is not allowed to change bit 0...29, while the object exists (bit 30 = 1).

4.2.4 Object 1006h: Communication Cycle Period

The object defines the time interval between SYNC messages expressed in µs. A drive sends SYNC messages if it is configured to send SYNC messages through object 1005h and the object 1006h is set with a non-zero value.

Object description:

Index	1006h
Name	Communication cycle period
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	0

4.2.5 Object 1010h: Store parameters

This object controls the saving of certain object parameters in the non-volatile memory. By writing 65766173h (“save” in /ISO8859/ characters) into sub--index 01h, the drive stores the parameters of the following objects:

- 1400h-1403h ;
- 1600h-1603h ;
- 1800h-1803h;
- 1A00h-1A03h;
- 1005h; 1006h; 100Ch; 100Dh; 1014h; 1017h;
- 207Bh; 207Ch;
- 6007h; 605Ah; 605Bh; 605Ch; 605Dh; 605Eh; 6060h; 6065h; 6066h; 6067h; 6068h; 607Ah; 607Ch; 607Dh; 607Eh; 6081h; 6083h; 6085h; 6098h; 6099h; 609Ah; 60FFh.

By reading sub-index 01h of object 1010h, the reply shall be 0x00000001, meaning the device does not save parameters autonomously and it saves them on command.

On reception of the correct signature in 01h sub-index, the drive will confirm the SDO transmission (SDO download response). Because storing of drive parameters lasts more than an SDO write command, always wait for the SDO confirmation message.

After save command is performed, the iPOS, shall always load the parameters of the previously mentioned objects at startup. To restore the default standard values see *Object 1011h: Restore parameters*.

Object description:

Index	1010h
Name	Store parameters
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	00h
Description	highest sub-index supported
Access	RO
PDO mapping	No
Value range	1
Default value	1

Sub-index	01h
Description	Save parameters
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	-

To save the parameters of the objects previously mentioned, send the following command:

(SDO access to object 1010h sub-index 1, 32-bit value 65766173h)

COB-ID	Data
606	**23 10 10 01 73 61 76 65**

4.2.6 Object 1011h: Restore parameters

This object restores certain object parameters to their default values. By writing 64616F6Ch ("load" in /ISO8859/ characters) into sub--index 01h, the drive restores to their default values the parameters of the following objects :

- 1400h-1403h ;
- 1600h-1603h ;
- 1800h-1803h;
- 1A00h-1A03h;
- 1005h; 1006h; 100Ch; 100Dh; 1014h; 1017h;
- 6065h; 6066h; 6067h; 6068h; 6060h; 607Ch; 6081h; 6083h; 6098h; 6099h; 60FFh

By reading sub-index 01h of object 1011h, the reply shall be 0x00000001, meaning the device can restore CANopen parameters to their default value.

The default values will be set valid after the iPOS drive is reset.

Object description:

Index	1011h
Name	Restore default parameters
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

Sub-index	00h
Description	highest sub-index supported
Access	RO
PDO mapping	No
Value range	1
Default value	1

Sub-index	01h
Description	Restore all default parameters
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	-

To restore the object parameters to their default values, send the following command:

(SDO access to object 1011h sub-index 1, 32-bit value 64616F6C h)

COB-ID	Data
606	**23 11 10 01 6C 6F 61 64**

4.2.7 Object $100C_h$: Guard Time

The Guard Time object multiplied with Lifetime Factor (index $100D_h$) gives the Lifetime of the drive for the Life Guarding Protocol. The Guard Time is expressed in ms. When the Life Guarding Protocol is not used the object must be set to 0. When the Node Guarding is active, i.e. the network manager sends the Node Guarding messages, the Life Guarding Protocol checks if the master has stopped sending messages or not. The decision of Node Guarding failure is taken if no message from the master is received within the period defined as Lifetime.

Object description:

Index	$100C_h$
Name	Guard time
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	0

4.2.8 Object $100D_h$: Life Time Factor

The lifetime factor multiplied with the guard time gives the lifetime for the Life Guarding Protocol. Must be 0 if not used.

Object description:

Index	$100D_h$
Name	Life time factor
Object code	VAR
Data type	UNSIGNED8

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	0

4.2.9 Object 1013_h: High Resolution Time Stamp

This object can receive a time stamp with a resolution of 1µs (1 unit = 1µs). It can be used in order to synchronize the drives in the CANopen network.

When setting up the synchronization mechanism, the master can map the object 1013_h on a receive PDO whose COB-ID should be identical on all the slave drives that need to be synchronized.

This object has to be written immediately after the SYNC message (the one that has the COB-ID 0x80). Upon the time reception in this object, the drive will compensate for the difference between the received value and its internal clock value.

The object also provides the drives internal clock value with a resolution of 1µs when read. It can be mapped to a TxPDO to transmit a precise time over the network.

Remark 1: the drive internal clock will not be read anymore if a value is written into object 1013_h. When object 1013_h is read, it will give either the internal clock or the last value written in it.

Remark 2: If a 4 byte (32bit) High Resolution Time Stamp is sent with the COB ID 0x100 right after the sync message (with ID 0x80), all the drives in the network will receive the time data as if it was received into object 1013_h.

Example**: ID** 0x100 **Data 00 00 E8 03** – absolute time is 1000 (0x03E8) µs = 1ms.

Object description:

Index	1013_h
Name	High resolution time stamp
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0

4.2.10 Object 2004h: COB-ID of the High-resolution time stamp

This object defines the COB-ID used by the high-resolution time stamp message sent by the synchronization master (when the drive is configured as a SYNC producer) in order to achieve synchronization on the network.

When the drive is the SYNC producer, this object defines if the high resolution time stamp is sent or not.

Object description:

Index	2004h
Name	COB-ID High resolution time stamp
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	100h

The structure of the parameter is the following:

Bit	Value	Meaning
31	0	High resolution time stamp exists / is valid
	1	High resolution time stamp does not exist / is not valid
30	0	Reserved (always 0)
29	0	11 bit ID
	1	29 bit ID
28…11	X	Bit 11...28 of 29-bit High resolution time stamp COB-ID
10...0	X	Bit 0...10 of High resolution time stamp COB-ID

It is not allowed to change bits 0-29 while the object exists (bit 31=0).

This object will be used when a Technosoft drive is required to be the master for the synchronization messages. In this case, the CANopen master does not need to map the 1013h into a receive PDO.

4.2.11 Configure the drive as a SYNC master Example

The procedure to activate the synchronization is the following:

- **Set the SYNC interval**. Write the desired SYNC interval into the object 1006h (Communication Cycle Period). For example – 20 ms.

Send the following message (SDO access to object 1006h sub-index 0, 32-bit value 0x4E20 = 20000 µs = 20 ms):

COB-ID	Data
606	**23 06 10 00 20 4E 00 00**

- **Activate the SYNC producer**. Set bit 30 in object 1005h (COB-ID of SYNC Message).

Send the following message (SDO access to object 1005h sub-index 0, 32-bit value 40000080h):

COB-ID	Data
606	**23 05 10 00 80 00 00 40**

The drive will start sending sync messages with COB ID 0x80 Data null. It will also send time stamp messages with COB ID 0x100 Data 0x12 0x34 0x56 0x78 0x00 0x00 where 0x000078563412 is the time stamp data expressed in µs. Also, if in object 2004h the time stamp is disabled, the sync producer will emit only sync messages with COB ID 0x80.

4.2.12 Object 1014h: COB-ID Emergency Object

Index 1014h defines the COB-ID of the Emergency Object (EMCY).

Object description:

Index	1014h
Name	COB-ID Emergency message
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	80h + Node-ID

***Table 4.5** – Structure of the EMCY Identifier*

MSB				LSB
31	30	29	28 - 11	10 - 0
0/1	0	1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	11-bit Identifier
0/1	0	1	29 –bit Identifier	

__Table 4.6__ – Description of the EMCY COD-ID entry

Bit	Value	Description
31 (MSB)	0	EMCY exists / is valid
	1	EMCY does not exist / is not valid
30	0	Reserved
29	0	Use 11 bit identifier
	1	Use 29 bit identifier (not supported)
28…11	0	Reserved
10...0 (LSB)	X	Bit 0...10 of COB-ID

It is not allowed to change Bits 0-29, while the object exists (Bit 31=0).

By setting Bits 0 to 10 to 0, the EMCY messages will be disabled.

4.2.13 Object 1017h: Producer Heartbeat Time

This object defines the cycle time of the heartbeat (if not equal to zero). If the heartbeat is not used, this object must have the default value 0. The time has to be a multiple of 1 ms.

Object description:

Index	1017h
Name	Producer Heartbeat Time
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	0

4.2.14 Object 2089h: Synchronization test config

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

Object description:

Index	2089h
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 4.2.7__ –Bit Assignment in Synchronization test config

Bit	Value	Description
3-15	-	Reserved
2	1	Trigger Control Loop (slow loop) on Ready/OUT3
1	1	View SYNC0 on Error/OUT2

Remarks:

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

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

5 Drive control and status

5.1 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 5.1.1** describes the state machine of the drive along with Controlword and Statusword values for each transition. **Table 5.1.1** describes each transition present in the state machine.

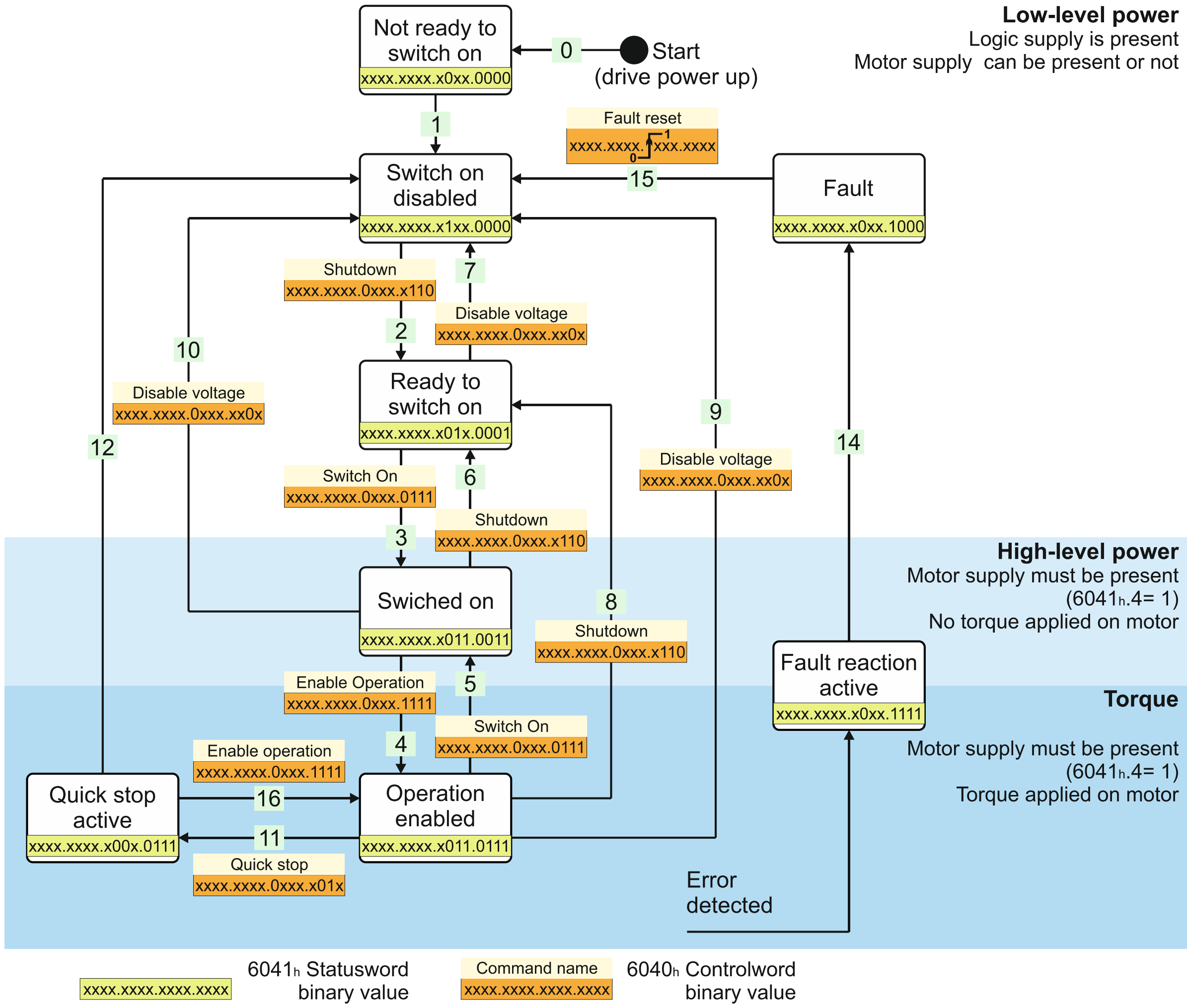


Figure 5.1.1. *Drive's status machine. States and transitions*

Table 5.1 *– 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 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 5.2 *– 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

5.2 Drive control and status objects

5.2.1 Object 6040_h: 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	6040_h
Name	Controlword
Object code	VAR
Data type	UNSIGNED16

Entry description:

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

***Table 5.3** – Bit Assignment in Controlword*

Bit	Value	Meaning
15	0	Registration mode inactive
	1	Activate registration mode[1]
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 2006_h. 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 5.4** – Command coding in Controlword*

Command	Bit in object 6040_h					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 5.1.1. Drive's status machine. States and transitions.*

[1] The Registration mode offers the possibility to super-impose another motion profile on top of an existing electronic gearing profile, at the slave level. For more details, check the "Electronic Gearing Position (EGEAR) Mode" chapter.

5.2.2 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 5.5** – 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 CAN 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 5.6** – 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 5.1.1. 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.

5.2.3 Object 1002h: Manufacturer Status Register

This object is a common status register for manufacturer specific purposes.

Object description:

Index	1002h
Name	Manufacturer status register
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RO
PDO mapping	Optional
Value range	UNSIGNED32
Default value	No

***Table 5.7** – Bit Assignment in Manufacturer Status Register*

Bit	Value	Description
31	1	Drive/motor in fault status
30	1	Reference position in absolute electronic camming mode reached
29	1	Reserved
28	1	Gear ratio in electronic gearing mode reached
27	1	Drive I2t protection warning level reached
26	1	Motor I2t protection warning level reached
25	1	Target command reached
24	1	Capture event/interrupt triggered
23	1	Limit switch negative event / interrupt triggered
22	1	Limit switch positive event / interrupt triggered
21	1	AUTORUN mode enabled
20	1	Position trigger 4 reached
19	1	Position trigger 3 reached
18	1	Position trigger 2 reached
17	1	Position trigger 1 reached
16	1	Drive/motor initialization performed
15…0		Same as Object 6041h, Statusword

5.2.4 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…-6	Reserved
-5	Manufacturer specific – External Reference Torque Mode[1]
-4	Manufacturer specific – External Reference Speed Mode[1]
-3	Manufacturer specific – External Reference Position Mode[1]
-2	Manufacturer specific – Electronic Camming Position Mode
-1	Manufacturer specific – Electronic Gearing Position Mode
0	Reserved
1	Profile Position Mode
2	Reserved
3	Profile Velocity Mode
4	Profile Torque Mode[2]
5	Reserved

[1] The External Reference control mode is not available with firmware version FA01x

[2] This mode is available starting with firmware version F514K or newer and FA01x.

6	Homing Mode
7	Interpolated Position Mode
8	Cyclic Synchronous Position Mode (CSP)
9	Cyclic sync Velocity Mode (CSV)[1]
10	Cyclic sync Torque Mode (CST)[2]
11…127	Reserved

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

5.2.5 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.

5.3 Limit Switch functionality explained

5.3.1 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 enter automatically in quick stop active state (statusword = xxxx xxxx x00x 0111b) where the deceleration value is 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.

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 0111b). 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 enter quick stop.

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

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

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

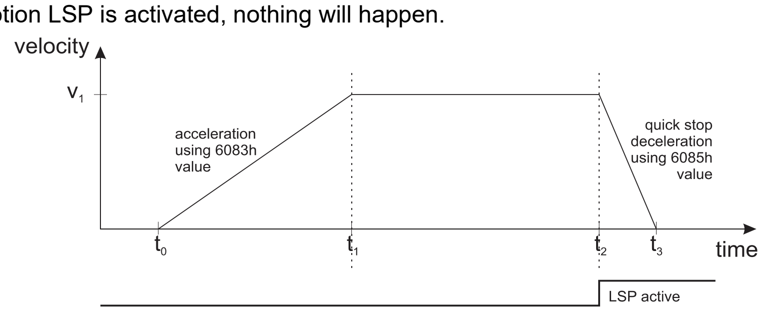


Figure 5.3.1. *Stopping a motion on the positive limit switch*

Figure 5.3.1 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 enters quick stop state where it decelerates using the value defined in Object ***6085h***: Quick stop deceleration.

[1] This mode is available starting with firmware version FA01x.

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:

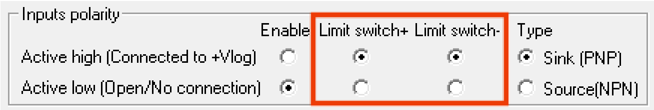


Figure 5.3.2. *Configuring the limit switch active state in Drive setup.*

Statusword 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, Statusword 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 2000h bit 6 will be 1.

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

When a limit switch becomes inactive, the emergency error code 0x0000 will be sent automatically and object 2000h 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.

5.3.2 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 drive will quick stop. A new motion will be accepted only if the motion is opposite the active software or hardware limit switch.

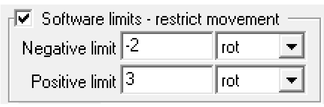


Figure 5.3.3. *Configuring the software limit switches position values in Drive setup.*

The software limit switches can also be configured through Object 607D$_h$: 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.

5.4 Error monitoring

5.4.1 Object 2000$_h$: 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.

Object description:

Index	2000$_h$
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 5.8** – 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. They can be distinguished either by the associated emergency code, or in conjunction with other bits from the DER (2002_h) register.
13	Under-voltage. Set when protection is triggered. Reset by a Reset Fault command
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 F514G and newer: Feedback error. Details found in DER2 (2009_h) bits. Set when protection is triggered. Reset by a Reset Fault command. For F508x/509x; F523x/524x, 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	CAN error. Set when CAN controller is in error mode. Reset by a Reset Fault command

5.4.2 Object 2001_h: Motion Error Register Mask

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

Entry description:

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

5.4.3 Object 2002_h: 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	2002_h
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 5.9 – *Bit Assignment in Detailed Error Register*

Bit	Description
15	EEPROM Locked; an attempt to write in the EEPROM will be ignored.
14	STO or Enable circuit hardware error
13	Self-check error; Internal memory (OTP) checksum error
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.

5.4.4 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 5.10 – *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.
5	Hall sensor missing; can be either Digital or Linear analogue hall error.
4	Absolute Encoder Interface (AEI) interface error; applies only to iPOS80x0 drives.
3	BiSS sensor missing; No BiSS sensor communication detected.
2	BiSS data error bit is set.
1	BiSS data warning bit is set.
0	BiSS data CRC error.

5.4.5 Object 2003$_h$: Communication Error Register (CER)

The Communication Error Register (CER) is a 16-bit status register, containing information about communication errors on CAN, SPI and SCI communication channels. 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	2003$_h$
Name	Communication Error Register
Object code	VAR
Data type	UNSIGNED16

[1] Available only in firmware versions F514x and FA01x.

Entry description:

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

***Table 5.11** – Bit Assignment in Communication Error Register*

Bit	Description
15..8	reserved
7	SPI timeout on write operation
6	CAN bus off error. It is automatically reset if the drive successfully receives a new message over CAN.
5	CAN transmission overrun error
4	CAN reception overrun error
3	CAN reception timeout error
2	RS232 reception timeout error
1	RS232 transmission timeout error
0	RS232 reception error

5.4.6 Object 603Fh: Error code[1]

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 Table 4.2 – Emergency Error Codes .Object description:

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

Entry description:

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

5.4.7 Object $605A_h$: 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 6083_h. The quick stop ramp is a deceleration value set by the Quick stop deceleration object, index 6085_h.

Object description:

Index	$605A_h$
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

[1] Available starting with firmware version FA01x.

5.4.8 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

5.4.9 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

5.4.10 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

5.4.11 Object 605E$_h$: 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 6007$_h$ option 1 is set)

Object description:

Index	605E$_h$
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

5.4.12 Object 6007$_h$: Abort connection option code

The object sets the action performed by the drive when one of the following events occurs: bus-off, heartbeat and life guarding.

Object description:

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

Entry description:

<table>
<tr><td>Access</td><td colspan="2">RW</td></tr>
<tr><td>PDO mapping</td><td colspan="2">Yes</td></tr>
<tr><td>Value range</td><td colspan="2">-32768…32767</td></tr>
<tr><td rowspan="2">Default value</td><td>For F514x firmware</td><td>1 (fault if communication error)</td></tr>
<tr><td>For F508/509/523 and 524x firmware</td><td>0 (no action if communication error)</td></tr>
</table>

[1] Starting with firmware version FA01C, I^2t is no longer a "non-fatal error" that can be configured through object 605E$_h$.

Table 5.12 – 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.

5.4.13 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:

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

5.4.14 Object 2113h: Detailed Option Code[2]

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 2114 is activated; otherwise, the settings will remain inactive.

Object description:

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

[1] Available starting with FA01C firmware version or newer

[2] Available starting with FA01C firmware version or newer

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

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 5.13** – 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.

5.5 Digital I/O control and status objects

5.5.1 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

	Bit	Value	Description
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

5.5.2 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 CAN bus load. This is especially helpful when there are many devices in a network and the data transmission cycle time is low.

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 5.14** – Sub-index 1 bit description*

	Bit	Value	Description
208F$_h$: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 5.15 – 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

5.5.3 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 5.16 – 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

	Bit	Description
Device profile Defined	15…0	Reserved

5.5.3.1 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 **60FE$_h$ 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 **60FE$_h$ 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.

5.5.4 Object 2090$_h$: Digital outputs 8bit

Has the same functionality as object 60FE$_h$ digital outputs, only that its two sub-indexes are 8 bit instead of 32bit. Mapping shorter data to a PDO decreases the total CAN bus load. This is especially helpful when there are many devices in a network and the data transmission cycle time is low.

Object description:

Index	2090$_h$
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 5.17** – 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

5.5.5 Object 2045$_h$: 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).

5.5.6 Object 2102h: Brake status

In Motor Setup, one digital output can be assigned as a brake output. The output will be SET or RESET when the motor PWM power is turned OFF or ON.

This object will show 1 when the brake output is active and 0 when not.

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

5.5.7 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

5.5.8 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

5.5.9 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]. You can read this value in the “Drive Info” dialogue, which can be opened from the “Drive Setup”.

5.5.10 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:

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

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”.

5.5.11 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

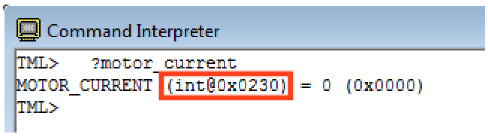
Sub-index	1
Description	16 bit variable address
Access	RW
PDO mapping	Possible
Value range	UNSIGNED16
Default value	0x0230 (address. or motor current)

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	-

5.5.11.1 How object 2108_h works:

Sub-index 1 sets the filtered variable address. To find a variable address, in EasySetup or EasyMotion 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.

5.6 Protections Setting Objects

5.6.1 Object $607D_h$: Software position limit[1]

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 6085_h: 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.

Object description:

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

[1] Object $607D_h$ is available with firmware version F514I or newer and FA01x.

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

5.6.2 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 I_{peak} is the peak current supported by the drive and *current[IU]* is the command value for object 2050$_h$.

5.6.3 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

5.6.4 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:

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

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

5.6.5 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 5.6.1**)

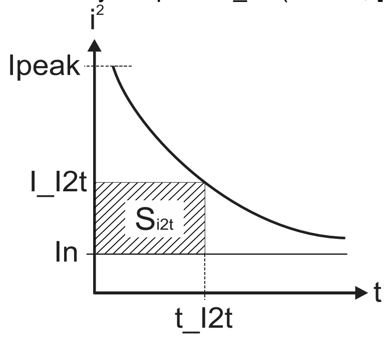


Figure 5.6.1. *I2t motor thermal protection curve*

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 5.6.2**). 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

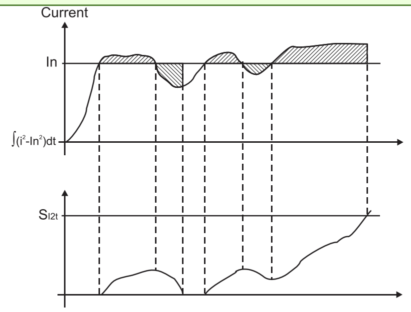


Figure 5.6.2. *I2t protection implementation*

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).

5.6.6 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 5.6.1**.

5.6.7 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 target current in [A] and C*urrent_Limit[IU]* is the target value to be written in object 207Fh.

5.7 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.

5.7.1 Object 2083h: 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	2083h
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.

5.7.2 Object 2084h: 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 5.7.1**) and is controlled with 256 microsteps / step (see **Figure 5.7.2**), 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	2084h
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	-

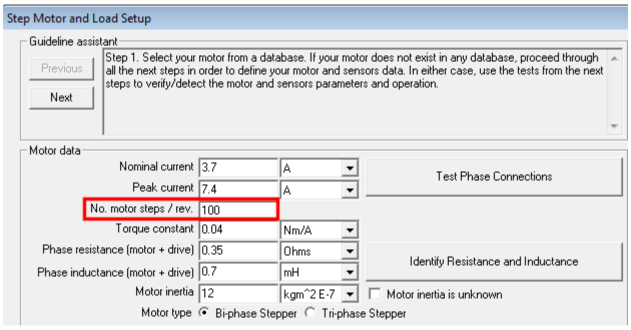


Figure 5.7.1. *Motor steps / revolution*

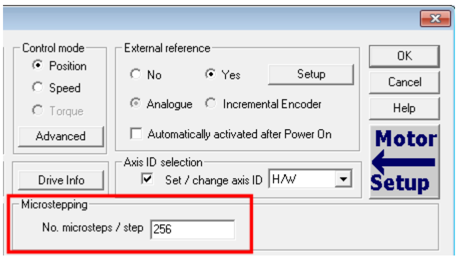


Figure 5.7.2. *Motor microsteps / step*

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.

5.7.3 Enabling step loss detection protection

Before enabling the step loss detection protection, the *Encoder resolution* in object 2083_h and the *Stepper resolution* in object 2084_h 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:

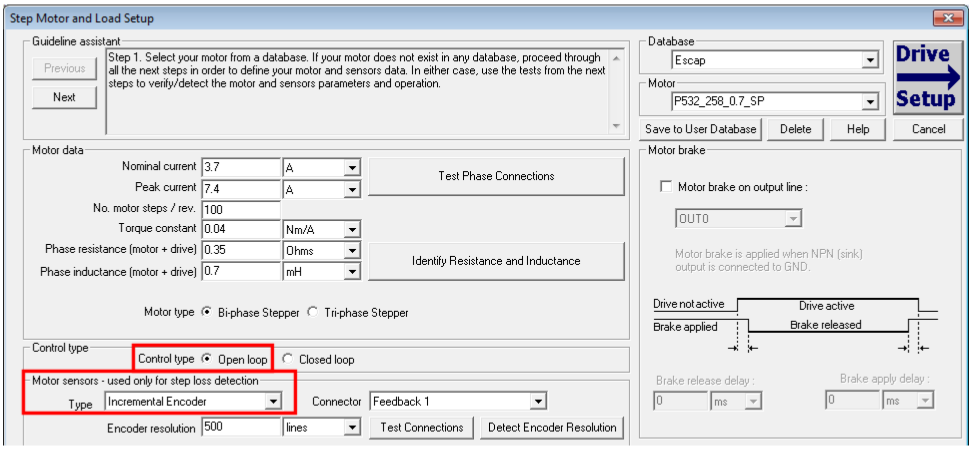


Figure 5.7.3. *Configuring the feedback sensor for step loss detection*

The step loss detection protection parameters are actually the control error parameters: object 6066_h - *Following error time* and object 6065_h - *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 6065_h for a time interval greater than the value set in object 6066_h.

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 6065_h, then set the *Following error time* in object 6066_h to a value different from 65535 (0xFFFF). To disable this protection, set a 65535 value in object 6066_h.

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 $60F4_h$. It is expressed in microsteps.

5.7.4 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 $60F4_h$ - *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 6065_h - *Following error window*.

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

5.7.5 Recovering from step loss detection fault

When the step loss detection protection is triggered, the drive enters in Fault state. The CANopen 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.

5.7.6 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.

5.8 Drive info objects

5.8.1 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

5.8.2 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	$001F0065_h$ for iPOS family

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

5.8.3 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

5.8.4 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:

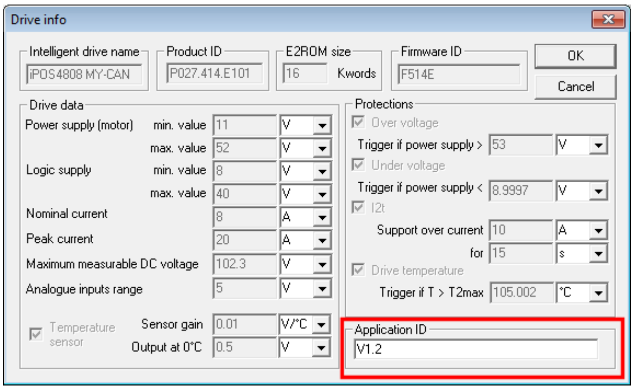
Index	$100A_h$
Name	Manufacturer software version
Object code	VAR
Data type	Visible String

Entry description:

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

5.8.5 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**.

5.8.6 Object 1018h: Identity Object

This object provides general information about the device.

Sub-index 01h shows the unique Vendor ID allocated to Technosoft (1A3h).

Sub-index 02h 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 02h will be read as the number 27214121 in decimal.

Sub-index 03h shows the Revision number.

Sub-index 04h 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	1018h
Name	Identity Object
Object code	RECORD
Data type	Identity

Entry description:

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

Sub-index	01h
Description	Vendor ID
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	000001A3h

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

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

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

5.9 Miscellaneous Objects

5.9.1 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:

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

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

5.9.2 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

5.9.3 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

5.9.4 Object 2075h: Position triggers

This object is used in order to define a set of four position values whose proximity will be signaled through bits 17-20 of object 1002h, *Manufacturer Status Register*. If the *position actual value* is over a certain value set as a position trigger, then the corresponding bit in *Manufacturer Status Register* 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

5.9.5 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 object 1002_h *Manufacturer Status Register*.

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 5.18** – 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.

5.9.6 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	-

5.9.7 Object 208Bh[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	208Bh
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

5.9.8 Object 208Ch[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	208Ch
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

5.9.9 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

[1] Object 208Bh is available only on firmware version F514x and FA01x

[2] Object 208Ch is available only on firmware version F514x and FA01x

Entry description:

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

***Table 5.19** – 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
7	0	Reserved
6	0	Leave position controller history unchanged
	1	Reset position controller history
5	0	Leave speed controller history unchanged
	1	Reset speed controller history
4	0	Leave current controller history unchanged
	1	Reset current controller history
3	0	When 6040_h bit 14 = 1, at the next *update*[1], the Target Speed Starting Value is the Actual Speed
	1	When 6040_h bit 14 = 1, at the next *update*, the Target Speed Starting Value is zero.
0-2	0	Reserved.

5.9.10 Object $210B_h$: 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	$210B_h$
Name	Auxiliary Settings Register2
Object code	VAR
Data type	UNSIGNED16

Entry description:

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

***Table 5.20** – 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 $607C_h$ 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

5.9.11 Object $20A0_h$: 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:

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

[2] Object $20A0_h$ is available starting with firmware version F514K or newer and FA01x.

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

Entry description:

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	-

5.9.12 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	-

5.9.13 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	-

5.9.14 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	-

5.9.15 Object 2091h[1]: Lock EEPROM

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	2091h
Name	Lock EEPROM
Object code	VAR
Data type	UNSIGNED8

Entry description:

Access	RW
PDO mapping	NO
Value range	UNSIGNED8
Default value	0

***Table 5.21** – Bit Assignment in Lock EEPROM*

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

5.9.16 Object 2092h: 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 2076h *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	2092h
Name	User Variables
Object code	ARRAY
Data type	ULONG32

Entry description:

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

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

[1] Object 2091h is available only on firmware versions F514E or newer and FA01x

[2] Object 2092h is available only on firmware versions F514E or newer and FA01x

6 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 F514K / FA01x
- Factor group objects - CiA-402 (obsolete)– for other firmware versions

6.1 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 CANopen master supports handling the scaling calculations on its side, it is recommended to use them instead of using the "Factor" scaling objects.

6.1.1 Object 607E$_h$: Polarity

This object is used to multiply by 1 or -1 position and velocity objects. The object applies only to position profile and velocity profile 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 6.1 *– 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 2076$_h$: Save current configuration*, will set its current values as the a new default.

6.1.2 Object 6089$_h$: Position notation index

The *position notation index* is used to define the position into [SI] units. Its purpose if purely informative for CANopen 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 6093$_h$: Position factor.

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

Object description:

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

Entry description:

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

6.1.3 Object $608A_h$: Position dimension index

The *position dimension index* is used to define the position into [SI] units. Its purpose if purely informative for CANopen 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 6093_h: Position factor.

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

Object description:

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

Entry description:

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

6.1.4 Object $608B_h$: Velocity notation index

The *velocity notation index* is used to define the velocity into [SI] units. Its purpose if purely informative for CANopen 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 6094_h: Velocity encoder factor.

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

Object description:

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

Entry description:

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

6.1.5 Object $608C_h$: Velocity dimension index

The *velocity dimension index* is used to define the velocity into [SI] units. Its purpose if purely informative for CANopen 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 6094_h: Velocity encoder factor.

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

Object description:

Index	$608C_h$
Name	Velocity dimension index
Object code	VAR
Data type	UNSIGNED8

Entry description:

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

6.1.6 Object 608Dh: Acceleration notation index

The *acceleration notation index i*s used to define the acceleration into [SI] units. Its purpose if purely informative for CANopen 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	608Dh
Name	Acceleration notation index
Object code	VAR
Data type	INTEGER8

Entry description:

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

6.1.7 Object 608Eh: Acceleration dimension index

The *acceleration dimension index i*s used to define the acceleration into [SI] units. Its purpose if purely informative for CANopen 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	608Eh
Name	Acceleration dimension index
Object code	VAR
Data type	UNSIGNED8

Entry description:

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

6.1.8 Object 206Fh: Time notation index

The *time dimension index i*s used to define the time into [SI] units. Its purpose if purely informative for CANopen 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	206Fh
Name	Time notation index
Object code	VAR
Data type	INTEGER8

Entry description:

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

6.1.9 Object 2070h: Time dimension index

The *time dimension index i*s used to define the time into [SI] units. Its purpose if purely informative for CANopen 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

6.1.10 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

6.1.10.1 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]

6.1.11 Object 6094h: 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*:*

606Ch Velocity actual value; 606Bh Velocity demand value; 606Fh Velocity threshold; 60FFh Target velocity; 60F8h Max slippage; 6081h Profile velocity

To configure the object with optimal values, see *Setting the numerator and divisor in a factor group object. Example*.

Object description:

Index	6094h
Name	Velocity encoder 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

6.1.12 Object 6097h: 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*:*

6083h Profile acceleration; 6085h 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	6097h
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

6.1.13 Object 2071_h: 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*:*

6066_h Following error time out; 6068_h Position window time; 2023_h Jerk time; 2005_h Max slippage time out; 2051_h 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

6.2 Factor group objects - CiA-402-2

The user-defined units are translated to internal units (IU) by the factor / scaling objects: 6093_h (Position factor), 6094_h (Velocity encoder factor), $210F_h$ (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 F514K or newer and FA01x.*

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: $60A8_h$ (SI unit position), $60A9_h$ (SI unit velocity), $60AA_h$ (SI unit acceleration) and $60AB_h$ (SI unit jerk).

Table 6.2 – 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 6.3 – 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 6.4 – 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 2103_h: Number of encoder counts per revolution.
Step	IU(step)	AC	Available only for step motors. The value can be computed as object 2110h: Number of steps per revolution multiplied by object 2110h: 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 6.5 – 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".

6.2.1 Object 60A8h: SI unit position

This object indicates the user-defined position units. The object structure is defined in table Table 6.2 – *SI Objects Structure*. The profile specific field (bit 0 to bit 7) of this object is reserved (00h).

Object description:

Index	60A8h
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
00h (means 10^0)	41h(means deg)	01h(default)	00h(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
FDh (means 10^{-3})	01h(means m)	01h(default)	00h(default)
31 … 24	23 … 16	15 ... 8	7 ... 0

6.2.2 Object 6093h: 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 - 608Fh, Gear Ratio - 6091h and Feed Constant - 6092h.

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	6093h
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

6.2.3 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)

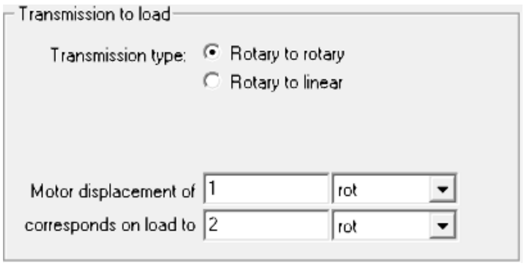
Sub-index	2
Description	Motor rotation
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0x00000001

6.2.4 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

6.2.5 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

6.2.6 Object 60A9$_h$: SI unit velocity

This object indicates the user-defined velocity units. The object structure is defined in Table 6.2 – *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

6.2.7 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

6.2.8 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

6.2.9 Object 60AAh: SI unit acceleration

This object indicates the user-defined acceleration units. The object structure is defined in Table 6.2 – *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

6.2.10 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

6.2.11 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

6.2.12 Object 60AB$_h$: SI unit jerk

This object indicates the user-defined jerk units. The object structure is defined in Table 6.2 – *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

6.2.13 Object 2110h: 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	2110h
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

6.2.14 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

7 Homing Mode

7.1 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.

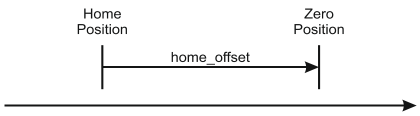


Figure 7.1.1. *Home Offset*

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.

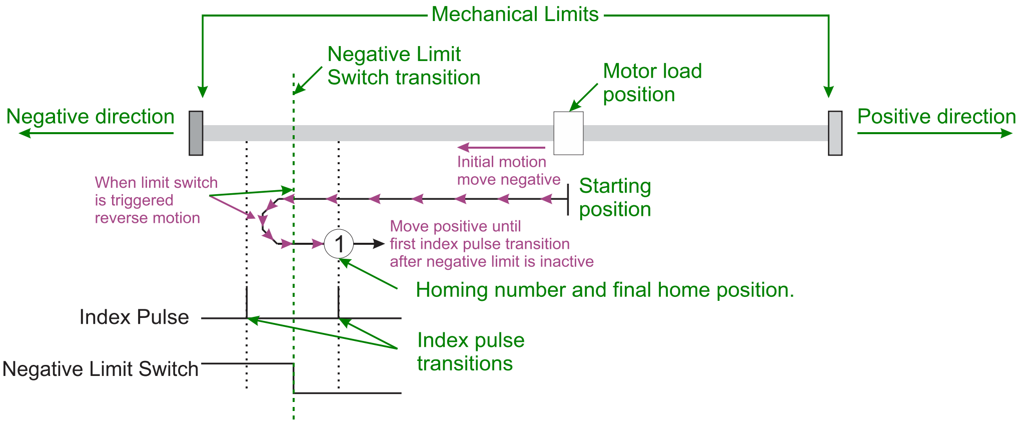


Figure 7.1.2. *Homing method 1 diagram explained*

7.2 Homing methods

7.2.1 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.

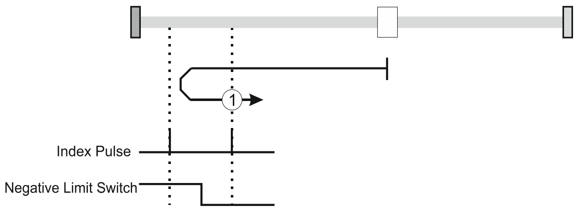


Figure 7.2.1. *Homing on the Negative Limit Switch and Index Pulse*

7.2.2 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 (positive 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.

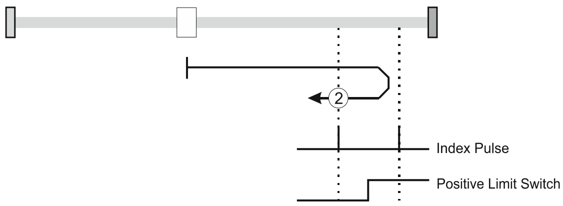


Figure 7.2.2. *Homing on the Positive Limit Switch and Index Pulse*

7.2.3 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).

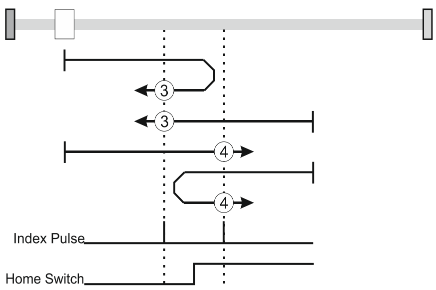


Figure 7.2.3. *Homing on the Positive Home Switch and Index Pulse*

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.

7.2.4 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.

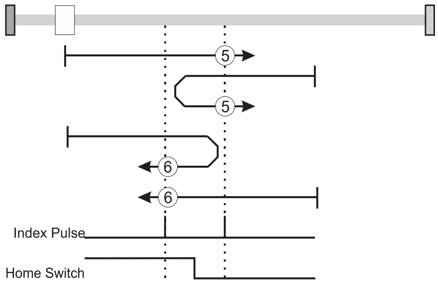


Figure 7.2.4. *Homing on the Negative Home Switch and Index Pulse*

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.

7.2.5 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).

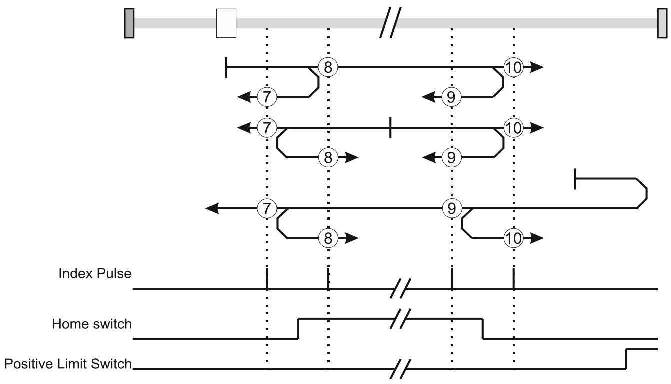


Figure 7.2.5. *Homing on the Home Switch using limit switches and Index Pulse – Positive Initial Move*

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.

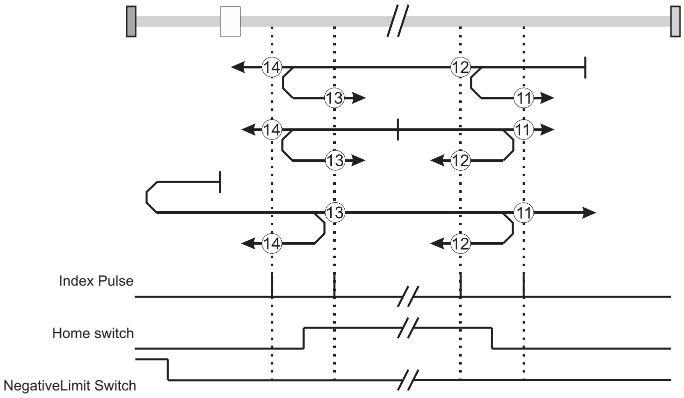


Figure 7.2.6. *Homing on the Home Switch using limit switches and Index Pulse – Negative Initial Move*

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

7.2.6 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.

7.2.7 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.

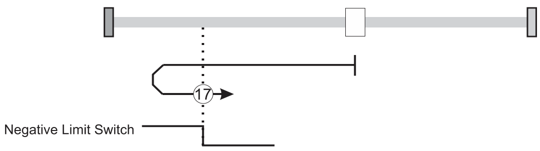


Figure 7.2.7. *Homing on the Negative Limit Switch*

7.2.8 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 (positive 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.

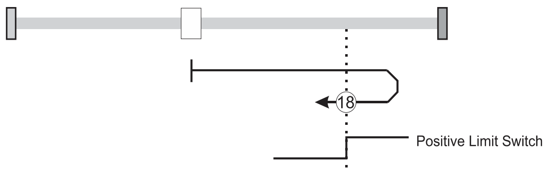


Figure 7.2.8. *Homing on the Positive Limit Switch*

7.2.9 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).

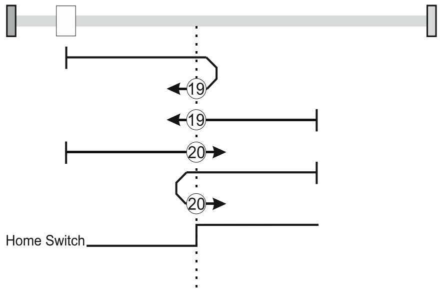


Figure 7.2.9. *Homing on the Positive Home Switch*

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 right after home switch low-high transition.

7.2.10 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.

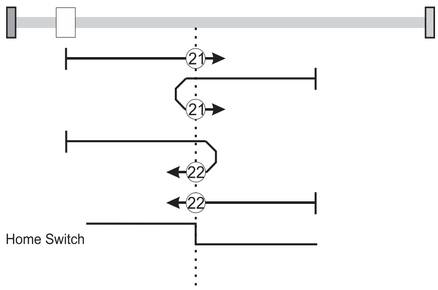


Figure 7.2.10. *Homing on the Negative Home Switch*

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.

7.2.11 Methods 23 to30: Homing on the Home Switch using limit switches

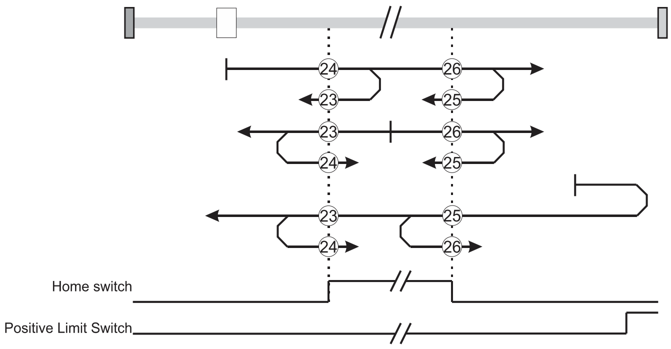


Figure 7.2.11. *Homing on the Home Switch using limit switches – Positive Initial Move*

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).

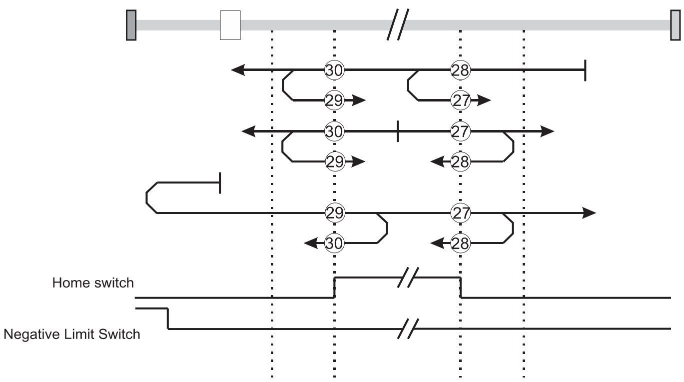


Figure 7.2.12. *Homing on the Home Switch using limit switches – Negative Initial Move*

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

7.2.12 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.

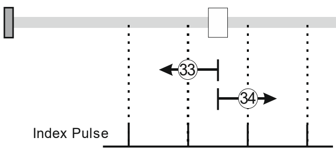


Figure 7.2.13. *Homing on the Index Pulse*

7.2.13 Method 35: Homing on the Current Position

In **method 35** the current position set with the value of home position (object 607Ch).

7.2.14 Method -1: Homing on the Negative Mechanical Limit and Index Pulse

7.2.14.1 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 1.3. Setting the current limit. *Current Threshold* < *current limit*

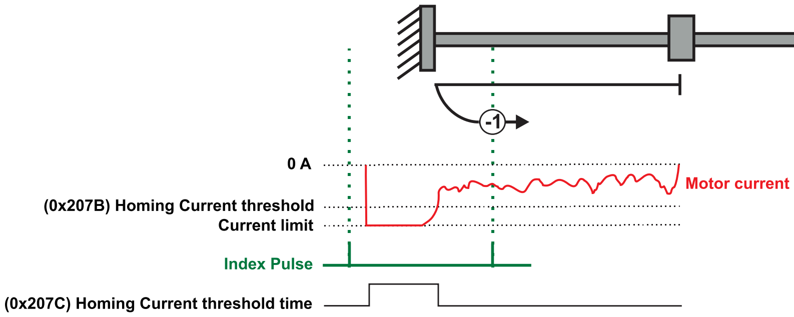


Figure 7.2.14. *Homing on the Negative Mechanical Limit and Index Pulse detecting the motor current increase*

7.2.14.2 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).

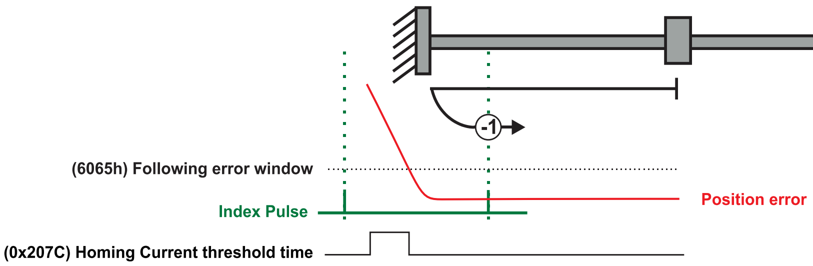


Figure 7.2.15. *Homing on the Negative Mechanical Limit and Index Pulse detecting a control error*

7.2.15 Method -2: Homing on the Positive Mechanical Limit and Index Pulse

7.2.15.1 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 1.3. Setting the current limit. *Current Threshold < current limit*

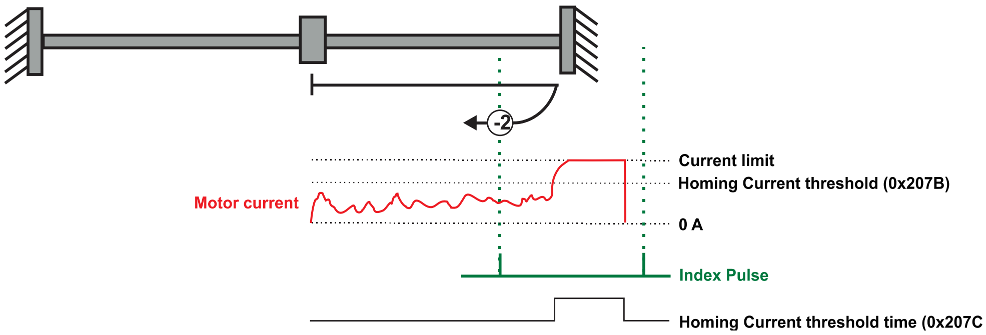


Figure 7.2.16. *Homing on the Positive Mechanical Limit and Index Pulse detecting the motor current increase*

7.2.15.2 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).

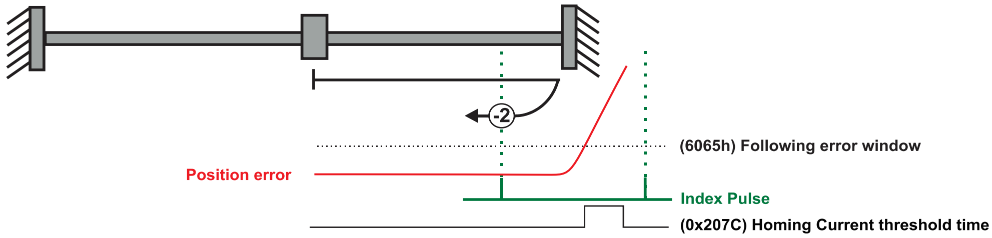


Figure 7.2.17. *Homing on the Positive Mechanical Limit and Index Pulse detecting a control error*

7.2.16 Method -3: Homing on the Negative Mechanical Limit without an Index Pulse.

7.2.16.1 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 1.3. Setting the current limit. *Current Threshold* < *current limit*

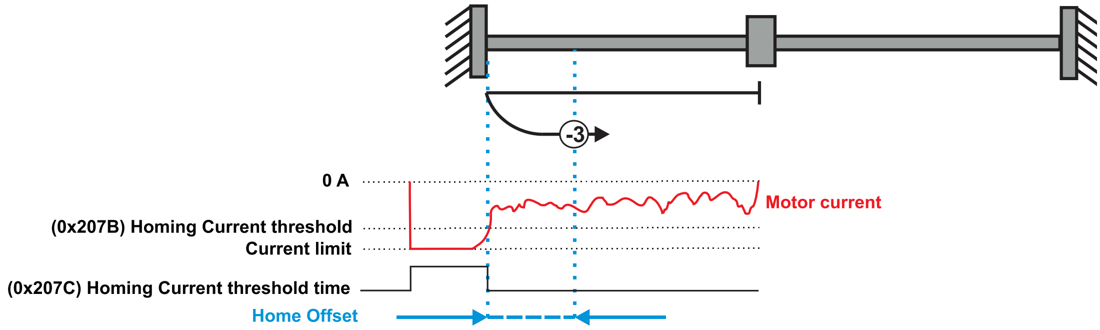


Figure 7.2.18. *Homing on the Positive Mechanical Limit without an Index Pulse detecting the motor current increase*

7.2.16.2 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 6065_h *Following error window* and $207C_h$ *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).

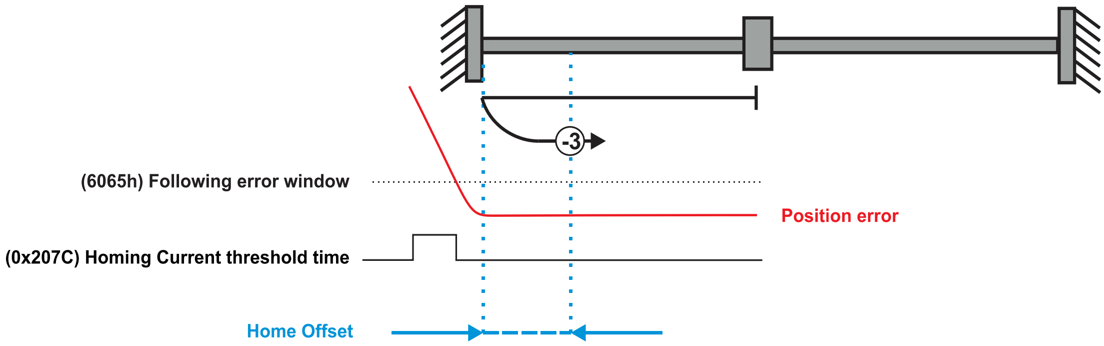


Figure 7.2.19. *Homing on the Positive Mechanical Limit without an Index Pulse detecting a control error*

7.2.17 Method -4: Homing on the Positive Mechanical Limit without an Index Pulse.

7.2.17.1 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 1.3. Setting the current limit. *Current Threshold* < *current limit*

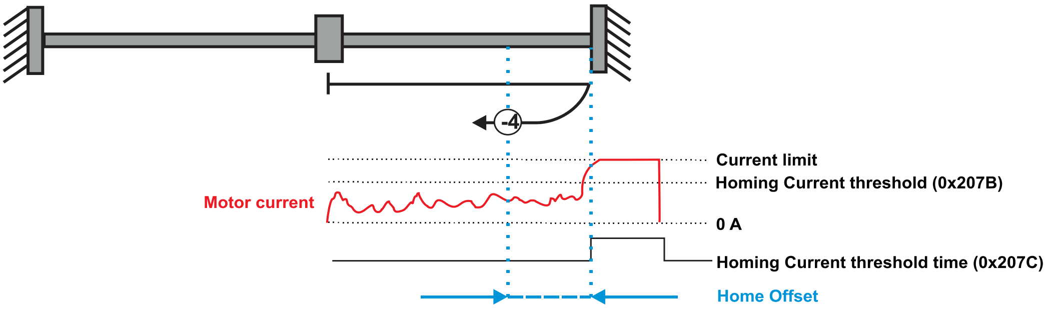


Figure 7.2.20. *Homing on the Positive Mechanical Limit without an Index Pulse detecting the motor current increase*

7.2.17.2 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).

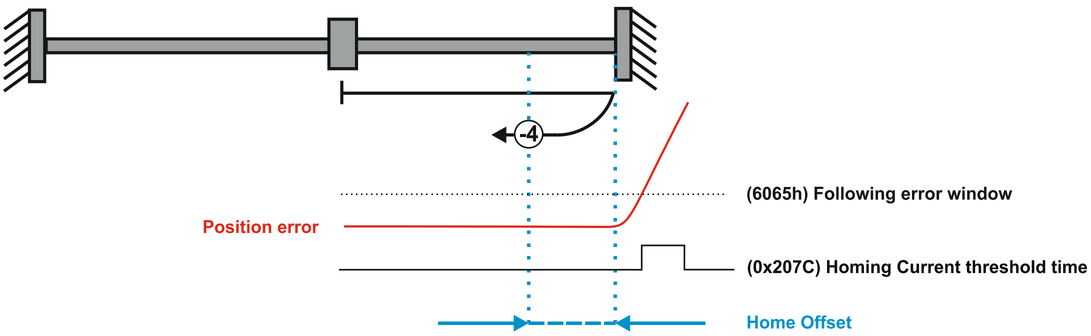


Figure 7.2.21. *Homing on the Positive Mechanical Limit without an Index Pulse detecting a control error*

7.3 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 7.1**. 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:

7.3.1 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 7.1 *– Controlword bits description for Homing Mode*

Name	Value	Description
Homing operation start	0 -> 1	Only a 0 to 1 transition will start homing mode
Halt	0	Execute the instruction of bit 4
	1	Stop drive with *homing acceleration*

7.3.2 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 7.2** – Statusword bits description for Homing Mode*

Name	Value	Description
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
Homing attained	0	Homing mode not yet completed
	1	Homing mode carried out successfully
Homing error	0	No homing error
	1	Homing error occurred; homing mode not carried out successfully.

***Table 7.3** – 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

7.3.3 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.

7.3.4 Object 6098h: Homing method

The *homing method* determines the method that will be used during homing.

Object description:

Index	6098h
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.

7.3.5 Object 6099h: 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	6099h
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.

7.3.6 Object 609Ah: 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	609Ah
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 2076$_h$: Save current configuration*, will set its current values as the a new default.

7.3.7 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 1.3. 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	CU
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 2076$_h$: Save current configuration*, will set its current values as the a new default.

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 207E$_h$.

7.3.8 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.

7.4 Homing example

Execute homing method number 18.

1. **Start remote node**. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	01 06

2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	06 00

3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	07 00

4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	0F 00

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 6099h sub-index 2 expressed in encoder counts per sample is 50000h.

Send the following message (SDO access to object 6099h sub-index 2, 32-bit value 00050000h):

COB-ID	Data
606	23 99 60 02 00 00 05 00

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 6099h sub-index 1 expressed in encoder counts per sample is 140000h.

Send the following message (SDO access to object 6099h sub-index 1, 32-bit value 00140000h):

COB-ID	Data
606	23 99 60 01 00 00 14 00

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 609Ah expressed in encoder counts per square sample is 28Fh.

Send the following message (SDO access to object 609Ah, 32-bit value 0000028Fh):

COB-ID	Data
606	23 9A 60 00 8F 02 00 00

8. **Home offset.** Set the home offset to 1 rotation. By using a 500 lines incremental encoder the corresponding value of object 607Ch expressed in encoder counts is 7D0h.

Send the following message (SDO access to object 607Ch, 32-bit value 000007D0h):

COB-ID	Data
606	23 7C 60 00 D0 07 00 00

9. **Homing method.** Select homing method number 18.

Send the following message (SDO access to object 6098h, 8-bit value 12h):

COB-ID	Data
606	2F 98 60 00 12 00 00 00

10. **Mode of operation.** Select homing mode.

Send the following message (SDO access to object 6060h, 8-bit value 6h):

COB-ID	Data
606	2F 60 60 00 06 00 00 00

11. **Start the homing.**

Send the following message:

COB-ID	Data
206	1F 00

12. **Press for 5s the LSP button on the IO board and release it.**
13. **Wait for homing to end.**
14. **Check the value of motor actual position.**

Send the following message (SDO access to object 6064h):

COB-ID	Data
606	40 64 60 00 00 00 00 00

The node will return the value of motor actual position that should be the same as the value of home offset (plus or minus few encoder counts depending on your position tuning).

8 Position Profile Mode

8.1 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 CANopen 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.

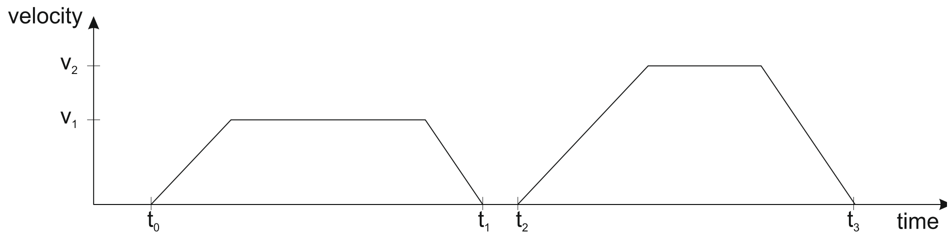
- **S-curve profile** the built-in reference generator computes a position profile with an S-curve shape of the speed. This shape is due to the jerk limitation, leading to a trapezoidal or triangular profile for the acceleration and an S-curve profile for the speed. The CANopen master specifies the absolute 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:

8.1.1 Discrete motion profile (*change set immediately* = 0)

After reaching the *target position* the drive unit signals this status to a CANopen 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 CANopen 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.



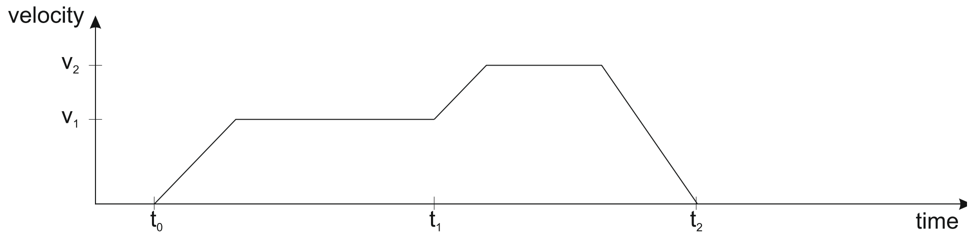
8.1.2 Continuous motion profile (*change set immediately = 1*)

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 slave 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.



8.1.3 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 8.1** – 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*

8.1.4 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 8.2** – 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

8.2 Position Profile Mode Objects

8.2.1 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 the Factor group chapter).

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

8.2.2 Object 6081_h: 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 the Factor group chapter).

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 fractional part.

Example: for a target speed of 50.00 IU, 0x00320000 must be set in 6081_h if no factor group is chosen.

Object description:

Index	6081_h
Name	Profile velocity
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	-

8.2.3 Object 6083_h: 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 the Factor group chapter).

If no factor is applied, the same description as object 6081_h applies. So 65536 IU = 1 encoder increment / sample^2.

Object description:

Index	6083_h
Name	Profile acceleration
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Possible
Value range	$0..(2^{32}-1)$
Default value	-

8.2.4 Object 6085_h: 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 $605A_h$) is set to 2 or 6. It is also used when the *fault reaction option code* object (index $605E_h$) and the *halt option code* object (index $605D_h$) is 2. *The quick stop deceleration* is given in user-defined acceleration units. User-defined means it can be modified by Factor group objects.

Object description:

Index	6085_h
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	-

8.2.5 Object 2023_h: Jerk time

In this object, you can set the time to use for S-curve profile (jerk-limited ramp set in Object 6086_h – 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	-

8.2.6 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

8.2.7 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 that can be modified by the factor group objects.

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	-

8.2.8 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 internal 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	-

8.2.9 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 that can be modified by the factor group objects.

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 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	-

8.2.10 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 firmware versions F514G or newer and FA01x
- 32767 for F508x/509x and F523x/524x 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 firmware version F514G or newer and FA01x
- “ERRMAX” for F508x/509x and F523x/524x firmware

found in parameters.cfg of the project file.

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

8.2.11 Object 6066h: Following error time out

See 6065h, *following error window*. The value is given in control loop time which is by default 1ms.

Object description:

Index	6066h
Name	Following error time out
Object code	VAR
Data type	UNSIGNED16

[1] ASR.11=1 implementation is available only of firmware versions F514x and FA01x.

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 2076$_h$: Save current configuration*, will set its current values as the a new default.

8.2.12 Object 6067$_h$: 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 values are given in user-defined position units that can be modified by the factor group objects. User-defined means it can be modified by Factor group objects. 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	6067$_h$
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 2076$_h$: Save current configuration*, will set its current values as the a new default.

8.2.13 Object 6068$_h$: Position window time

See description of Object 6067$_h$: Position window. The values are given in user-defined time units that can be modified by the factor group objects.

Object description:

Index	6068$_h$
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 2076$_h$: Save current configuration*, will set its current values as the a new default.

8.2.14 Object 607Bh: Position range limit[1]

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 (607Dh). 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 that can be modified by the factor group objects.

Object description:

Index	607Bh
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.

8.2.15 Object 60F2h: Positioning option code[2]

This object configures the positioning behavior as for the profile positioning mode or the interpolated positioning mode.

Object description:

Index	60F2h
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 8.3 *– 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.

[1] Object 607Bh is available only with firmware versions F514x and FA01x.

[2] Object 60F2h is available only with firmware versions F514x and FA01x.

	0	1	Positioning only in negative direction; if target position is higher than actual position, axis moves over the min position limit (607Bh, 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°.

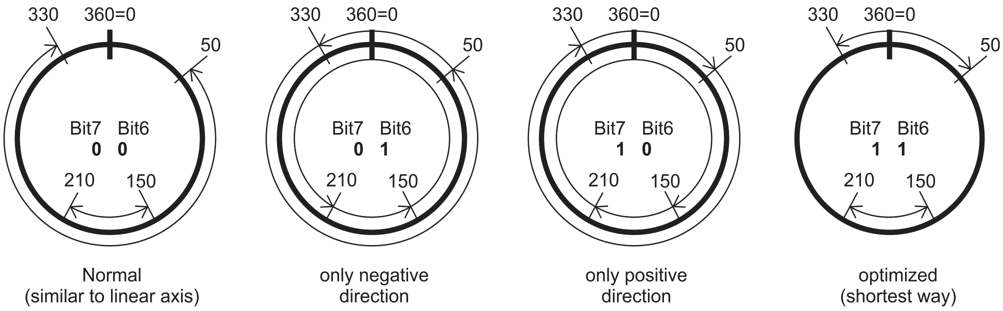


Figure 8.2.1. *Rotary axis positioning example*

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.

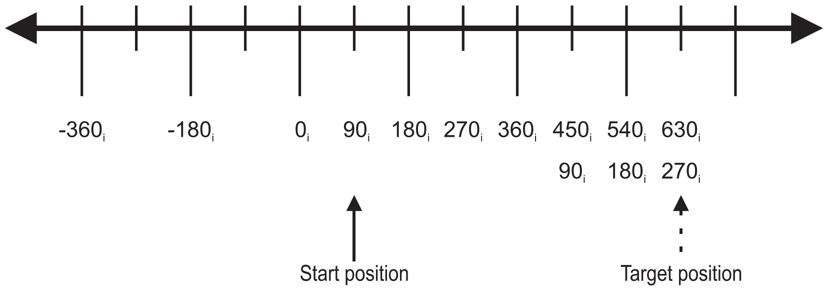


Figure 8.2.2. *Example for absolute movement greater than modulo value*

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.

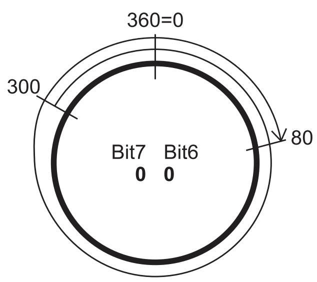


Figure 8.2.3. *Example for relative movement greater than modulo value*

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.

8.2.16 Object 60F4h: Following error actual value

This object represents the actual value of the following error, given in user-defined position units that can be modified by the factor group objects.

Object description:

Index	60F4h
Name	Following error actual value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Value range	INTEGER32
Default value	-

8.2.17 Object 60FCh: 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* (6062h).

Object description:

Index	60FCh
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	-

8.2.18 Object 2022h: Control effort[1]

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	2022h
Name	Control effort
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	Yes
Value range	INTEGER16
Default value	-

8.2.19 Object 2081h: Set/Change the actual motor position

This object sets the motor position to the value written in it. It affects object 6064h, 6063h and 6062h.

The object is not affected by the Factor Group and it receives its value in Internal Units.

Object description:

Index	2081h
Name	Set actual position
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	No
Value range	$-2^{31} \ldots 2^{31}-1$
Default value	-

[1] Available starting with firmware version FA01x

8.2.20 Object 2088$_h$[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	2088$_h$
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	$-2^{31} \ldots 2^{31}-1$
Default value	-

8.2.21 Object 208D$_h$[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	208D$_h$
Name	Auxiliary encoder 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	-

8.3 Position Profile Examples

8.3.1 Relative trapezoidal example

Execute an relative trapezoidal profile. First, perform 20 rotations, wait motion complete and then set the target position of 200 rotations.

1. **Start remote node**. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	**Data**
0	**01 06**

2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	**Data**
206	**06 00**

3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	**Data**
206	**07 00**

[1] Object 2088$_h$ applies only to drives which have a secondary feedback

[2] Object 208D$_h$ applies only to drives which have a secondary feedback

4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	0F 00

5. **Modes of operation.** Select position mode.

Send the following message (SDO access to object 6060_h, 8-bit value 1_h):

COB-ID	Data
606	2F 60 60 00 01 00 00 00

6. **Target position.** Set the target position to 20 rotations. By using a 500 lines incremental encoder the corresponding value of object $607A_h$ expressed in encoder counts is $9C40_h$.

Send the following message (SDO access to object $607A_h$ 32-bit value $00009C40_h$):

COB-ID	Data
606	23 7A 60 00 40 9C 00 00

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 6081_h expressed in encoder counts per sample is 10AAAch(16.667 counts/sample).

Send the following message (SDO access to object 6081_h, 32-bit value 0010AAAch):

COB-ID	Data
606	23 81 60 00 AC AA 10 00

8. **Start the profile.** If Controlword bit 6 is set (Controlword.6 = 1), a relative positioning will start.

Send the following message

COB-ID	Data
206	5F 00

9. **Wait movement to finish.**

10. **Reset the set point.**

Send the following message

COB-ID	Data
206	0F 00

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 $00061A80_h$):

COB-ID	Data
606	23 7A 60 00 80 1A 06 00

12. **Start the profile.**

Send the following message

COB-ID	Data
206	5F 00

13. **Wait movement to finish.**

14. **Check the value of motor actual position.**

Send the following message (SDO access to object 6064_h):

COB-ID	Data
606	40 64 60 00 00 00 00 00

15. **Check the value of position demand value.**

Send the following message (SDO access to object 6062_h):

COB-ID	Data
606	40 62 60 00 00 00 00 00

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.

8.3.2 Absolute trapezoidal example

Execute an absolute trapezoidal profile. First, perform 4 rotations, wait motion complete and then set the target position of 16 rotations.

16. Start remote node. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	01 06

17. Ready to switch on. Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	06 00

18. Switch on. Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	07 00

19. Enable operation. Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	0F 00

20. Modes of operation. Select position mode.

Send the following message (SDO access to object 6060_h, 8-bit value 1_h):

COB-ID	Data
606	2F 60 60 00 01 00 00 00

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 $1F40_h$.

Send the following message (SDO access to object $607A_h$ 32-bit value $00001F40_h$):

COB-ID	Data
606	23 7A 60 00 40 1F 00 00

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 10AAAch(16.667 counts/sample).

Send the following message (SDO access to object 6081_h, 32-bit value 0010AAAch):

COB-ID	Data
606	23 81 60 00 AC AA 10 00

23. Start the profile. If Controlword bit 6 is not set (Controlword.6 = 0), an absolute positioning will start.

Send the following message

COB-ID	Data
206	1F 00

24. Wait movement to finish.

25. Reset the set point.

Send the following message

COB-ID	Data
206	0F 00

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 $00007D00_h$):

COB-ID	Data
606	23 7A 60 00 00 7D 00 00

27. Start the profile.

Send the following message

COB-ID	Data
206	1F 00

28. Wait movement to finish.

29. Check the value of motor actual position.

Send the following message (SDO access to object 6064h):

COB-ID	Data
606	40 64 60 00 00 00 00 00

30. Check the value of position demand value.

Send the following message (SDO access to object 6062h):

COB-ID	Data
606	40 62 60 00 00 00 00 00

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.

8.3.3 Relative Jerk-limited ramp profile example

Execute an absolute Jerk-limited ramp profile.

Start remote node. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	01 06

Ready to switch on. Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	06 00

Switch on. Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	07 00

Enable operation. Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	0F 00

Mode of operation. Select position mode.

Send the following message (SDO access to object 6060h, 8-bit value 1h):

COB-ID	Data
606	2F 60 60 00 01 00 00 00

Motion profile type. Select Jerk-limited ramp.

Send the following message (SDO access to object 6086h, 16-bit value 3h):

COB-ID	Data
606	2B 86 60 00 03 00 00 00

Target position. Set the target position to 10 rotations. By using a 500 lines incremental encoder the corresponding value of object 607Ah expressed in encoder counts is 4E20h.

Send the following message (SDO access to object 607Ah 32-bit value 00004E20h):

COB-ID	Data
606	23 7A 60 00 20 4E 00 00

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 6081h expressed in encoder counts per sample is 000F0000h(15.0 counts/sample).

Send the following message (SDO access to object 6081h, 32-bit value 000F0000h):

COB-ID	Data
606	23 81 60 00 00 00 0F 00

Jerk time. Set the time to use for Jerk-limited ramp. For more information related to this parameter, see the ESM help

Send the following message (SDO access to object 2023h, 16-bit value 01F4h):

COB-ID	Data
606	2B 23 20 00 01 F4 00 00

Start the profile. If Controlword bit 6 is set (Controlword.6 = 1), a relative positioning will start.

Send the following message

COB-ID	Data
206	5F 00

Wait movement to finish.

Check the value of motor actual position.

Send the following message (SDO access to object 6064h):

COB-ID	Data
606	40 64 60 00 00 00 00 00

Check the value of position demand value.

Send the following message (SDO access to object 6062h):

COB-ID	Data
606	40 62 60 00 00 00 00 00

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).

8.3.4 Absolute Jerk-limited ramp profile example

Execute an absolute Jerk-limited ramp profile.

Start remote node. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	01 06

Ready to switch on. Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	06 00

Switch on. Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	07 00

Enable operation. Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	0F 00

Mode of operation. Select position mode.

Send the following message (SDO access to object 6060h, 8-bit value 1h):

COB-ID	Data
606	2F 60 60 00 01 00 00 00

Motion profile type. Select Jerk-limited ramp.

Send the following message (SDO access to object 6086h, 16-bit value 3h):

COB-ID	Data
606	**2B 86 60 00 03 00 00 00**

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.

Send the following message (SDO access to object $607A_h$ 32-bit value 00002710_h):

COB-ID	Data
606	**23 7A 60 00 10 27 00 00**

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(5.0 counts/sample).

Send the following message (SDO access to object 6081_h, 32-bit value 00050000_h):

COB-ID	Data
606	**23 81 60 00 00 00 05 00**

Jerk time. Set the time to use for Jerk-limited ramp. For more information related to this parameter, see the ESM help

Send the following message (SDO access to object 2023_h, 16-bit value $13B_h$):

COB-ID	Data
606	**2B 23 20 00 3B 01 00 00**

Start the profile. If Controlword bit 6 is not set (Controlword.6 = 0), an absolute positioning will start.

Send the following message

COB-ID	Data
206	**1F 00**

Wait movement to finish.

Check the value of motor actual position.

Send the following message (SDO access to object 6064_h):

COB-ID	Data
606	**40 64 60 00 00 00 00 00**

Check the value of position demand value.

Send the following message (SDO access to object 6062_h):

COB-ID	Data
606	**40 62 60 00 00 00 00 00**

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).

9 Torque Profile Mode

9.1 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 F514K and FA01x.*

9.1.1 Controlword in profile torque mode

MSB **LSB**

See 6040h		Halt	See 6040h	Reserved		See 6040h	
15	9	8	7	6	4	3	0

***Table 9.1** – 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)

9.1.2 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 9.2** – 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

9.2 Torque Profile Mode Objects

9.2.1 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 FA01x firmware version, 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:

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

where I_{peak} is the peak current supported by the drive and *current[IU]* is the command value for object 6071_h.

9.2.2 Object 6075_h: Motor rated current

The motor rated current is the motor's nominal current which needs to be expressed in mA. The object contains the nominal motor current declared in EasyMotion Studio / EasySetUp.

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.

9.2.3 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

9.3 Torque Profile Example

Execute a torque profile.

Start remote node. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	**01 06**

Ready to switch on. Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	**06 00**

Switch on. Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	07 00

Enable operation. Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	0F 00

Mode of operation. Select torque mode.

Send the following message (SDO access to object 6060_h, 8-bit value 4_h):

COB-ID	Data
606	2F 60 60 00 04 00 00 00

Motor rated current. Define the motor nominal(rated) current of 1000mA.

Send the following message (SDO access to object 6075_h, 32-bit value $03E8_h$):

COB-ID	Data
606	23 75 60 00 E8 03 00 00

Target slope. Define a target slope of 1000mA.

Send the following message (SDO access to object 6087_h, 32-bit value $03E8_h$):

COB-ID	Data
606	23 87 60 00 E8 03 00 00

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), send the following message (SDO access to object 2115h, 32-bit value 0001h):

COB-ID	Data
606	23 15 21 00 01 00 00 00

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 $01F4_h$):

COB-ID	Data
606	2B 71 60 00 F4 01 00 00

The motor will move positive and reach a current of 50.0% of the motor rated current (500mA).

Now 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 $FB50_h$):

COB-ID	Data
606	2B 71 60 00 50 FB 00 00

The motor will move negative and reach a current of -120.0% of the motor rated current (-1200mA).

[1] Only for FA01x firmware version.

10 Interpolated Position Mode

10.1 Overview

The interpolated Position Mode is used to control multiple coordinated axles or a single axle 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, based on the SYNC and the High Resolution Time Stamp messages (see object 1013 for details).

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.

10.1.1 Internal States

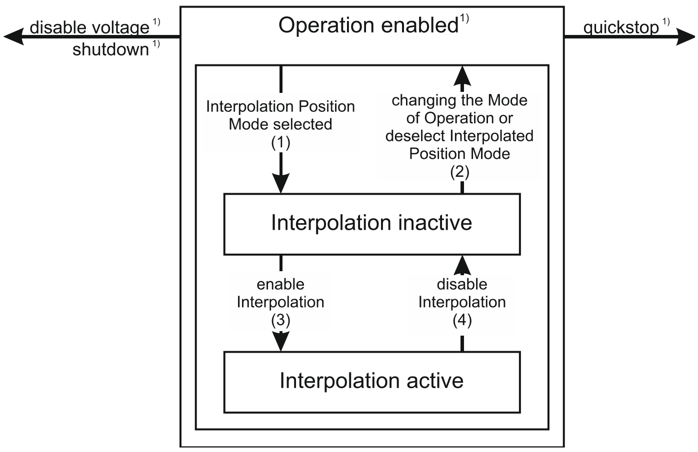


Figure 9.1.1. *Internal States for the Interpolated Position Mode*

[1] See state machine Operation enabled[1]

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

10.1.2 Controlword in interpolated position mode

MSB LSB

See 6040$_h$	Stop option	See 6040$_h$	Halt	See 6040$_h$	Abs / rel	Reserved	Enable ip mode	See 6040$_h$
15 12	11	10 9	8	7	6	5	4	3 0

Table 10.1 – *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.

10.1.3 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 10.2 – *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

10.2 Interpolated Position Objects

10.2.1 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	$-2^{15} \dots 2^{15}-1$
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

10.2.2 Object $60C1_h$: 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	$60C1_h$
Name	Interpolation data record
Object code	ARRAY
Number of elements	2
Data Type	Interpolated Mode dependent

Entry description

Sub-index	01_h
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:

10.2.2.1 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 $60C2_h$.

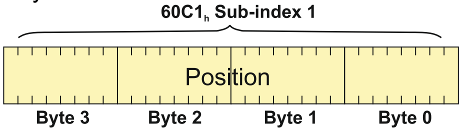


Figure 9.2.1. *Linear interpolation point 32-bit data structure*

10.2.2.2 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

[1] If object $207A_h$ Interpolated position 1st order time is used, these bits will be overwritten with the value defined in it

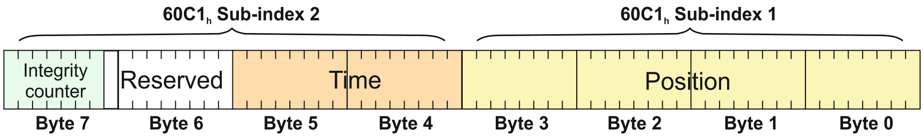


Figure 9.2.2. *PT interpolation point 64-bit data structure*

Remarks:

- The integrity counter is written in byte 3 of 60C1h 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

10.2.2.3 c) For PVT (Position – Velocity – Time) cubic interpolation

To work with this mode, object 208Eh 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

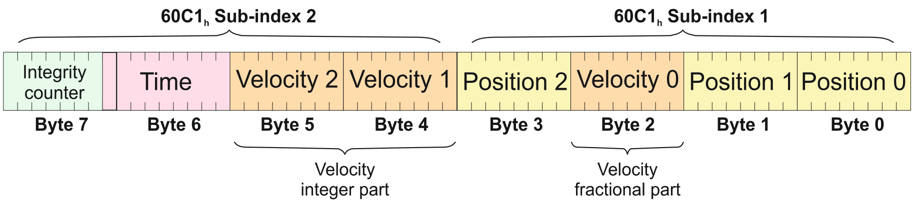


Figure 9.2.3. *PVT interpolation point 64-bit data structure*

Remarks:

- The integrity counter is written in byte 3 of 60C1h 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.

10.2.3 Object 2072h: Interpolated position mode status

The object provides additional status information for the interpolated position mode.

Object description:

Index	2072h
Name	Interpolated position mode status
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Possible
Value range	UNSIGNED16
Default value	-

***Table 10.3** – 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 2074h.
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 2072h)	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.

10.2.4 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

10.2.5 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 10.4** – 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 internal 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 2079_h. 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 2072_h 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

10.2.6 Object 2079_h: 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	2079_h
Name	Interpolated position initial position
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	Possible
Value range	INTEGER32
Default value	0

10.2.7 Object $207A_h$: 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** $60C1_h$**:** Interpolation data record. This object is disabled when it is set with 0. It is given in IU which is by default 0.8ms for steppers and 1ms for the other configurations.

Object description:

Index	$207A_h$
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

10.2.8 Loading the interpolated points

The points can be loaded only in Legacy interpolation mode (object $208E_h$ 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 2079_h the actual position of the drive (read from object 6063_h)
- If the motion is PT, set in object $207A_h$ a fixed time interval if not supplied in 60C1 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

10.3 Linear interpolation example

To work with this mode, object $208E_h$ 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 example* with the following changes:

- the modes of operation 6060_h must be set = 7 instead of 8
- object $60C1_h$ sub-index 1 must be used instead of object $607A_h$.

All the other commands and behavior is the same.

10.4 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 $208E_h$ bit8 must be 0. The default value of this bit is 1.

1. **Start remote node**. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	01 06

2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	06 00

3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	07 00

4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	0F 00

5. **Enable the legacy interpolated mode**. Set bit 8 of object $208E_h$ to 0.

Send the following message (SDO access to object $208E_h$ sub-index 0, 16-bit value 0):

COB-ID	Data
606	2B 8E 20 00 00 00 00 00

6. **Disable the RPDO3**. Write zero in object 1602_h sub-index 0, this will disable the PDO.

Send the following message (SDO access to object 1602_h sub-index 0, 8-bit value 0):

COB-ID	Data
606	2F 02 16 00 00 00 00 00

7. **Map the new objects**.

Write in object 1602_h 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):

COB-ID	Data
606	23 02 16 01 20 01 C1 60

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):

COB-ID	Data
606	23 02 16 02 20 02 C1 60

8. **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):

COB-ID	Data
606	2F 02 16 00 02 00 00 00

9. **Mode of operation**. Select interpolation position mode.

Send the following message (SDO access to object 6060h, 8-bit value 7h):

COB-ID	Data
606	2F 60 60 00 07 00 00 00

10. **Interpolation sub mode select**. Select PT interpolation position mode.

Send the following message (SDO access to object 60C0h, 16-bit value 0000h):

COB-ID	Data
606	2B C0 60 00 00 00 00 00

11. **Interpolated position buffer length**. Set the buffer length to 12. The maximum length is 15.

Send the following message (SDO access to object 2073h, 16-bit value Ch):

COB-ID	Data
606	2B 73 20 00 0C 00 00 00

12. **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):

COB-ID	Data
606	2B 74 20 00 01 A0 00 00

13. **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 03E8h):

COB-ID	Data
606	23 79 20 00 E8 03 00 00

14. **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.

Send the following message:

COB-ID	Data
406	20 4E 00 00 E8 03 00 02

15. **Send the 2nd PT point**.

Position= 30000 IU (0x00007530)

Time = 2000 IU (0x07D0)

IC = 2 (0x02)

Send the following message:

COB-ID	Data
406	30 75 00 00 D0 07 00 04

16. **Send the 3rd PT point**.

Position= 2000 IU (0x000007D0)

Time = 1000 IU (0x03E8)

IC = 3 (0x03)

Send the following message:

COB-ID	Data
406	D0 07 00 00 E8 03 00 06

17. 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)

Send the following message:

COB-ID	Data
406	00 00 00 00 F4 01 00 08

18. Start an absolute motion.

Send the following message:

COB-ID	Data
206	1F 00

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 12.

10.5 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**. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	01 06

2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	06 00

3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	07 00

4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	0F 00

5. **Enable the legacy interpolated mode**. Set bit 8 of object 208Eh to 0.

Send the following message (SDO access to object 208Eh sub-index 0, 16-bit value 0):

COB-ID	Data
606	2B 8E 20 00 00 00 00 00

6. **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):

COB-ID	Data
606	2F 02 16 00 00 00 00 00

7. **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):

COB-ID	Data
606	23 02 16 01 20 01 C1 60

   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):

COB-ID	Data
606	23 02 16 02 20 02 C1 60

8. **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):

COB-ID	Data
606	2F 02 16 00 02 00 00 00

9. **Mode of operation**. Select interpolation position mode.

Send the following message (SDO access to object 6060h, 8-bit value 7h):

COB-ID	Data
606	2F 60 60 00 07 00 00 00

10. **Interpolation sub mode select**. Select PVT interpolation position mode.

Send the following message (SDO access to object 60C0h, 16-bit value FFFFh):

COB-ID	Data
606	2B C0 60 00 FF FF 00 00

11. **Interpolated position buffer length**. Set the buffer length to 15. The maximum length is 15.

Send the following message (SDO access to object 2073h, 16-bit value Fh):

COB-ID	Data
606	2B 73 20 00 0F 00 00 00

12. **Interpolated position buffer configuration**. By setting the value B000h, the buffer is cleared and the integrity counter will be set to 0.

    Send the following message (SDO access to object 2074h, 16-bit value B000h):

COB-ID	Data
606	2B 74 20 00 00 B0 00 00

13. **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 = 0 (0x00) IC=Integrity Counter

    Send the following message:

COB-ID	Data
406	58 00 54 00 03 00 37 00

14. **Send the 2nd PVT point**.

    Position = 370 IU (0x000172)

    Velocity = 6.66 IU (0x0006A8)

    Time = 55 IU (0x37)

    IC = 1 (0x01)

    Send the following message:

COB-ID	Data
406	72 01 A8 00 06 00 37 02

15. **Send the 3rd PVT point**.

    Position = 2982 IU (0x000BA6)

    Velocity = 6.66 IU (0x0006A8)

    Time = 390 IU (0x186)

IC = 2 (0x02)

Send the following message:

COB-ID	Data
406	A6 0B A8 00 06 00 86 05

16. Send the 4th PVT point.

Position = 5631 IU (0x0015FF)

Velocity = 6.66 IU (0x0006A8)

Time = 400 IU (0x190)

IC = 3 (0x03)

Send the following message:

COB-ID	Data
406	FF 15 A8 00 06 00 90 07

17. Send the 5th PVT point.

Position = 5925 IU (0x001725)

Velocity = 3.00 IU (0x000300)

Time = 60 IU (0x3C)

IC = 4 (0x04)

Send the following message:

COB-ID	Data
406	25 17 00 00 03 00 3C 08

18. Send the 6th PVT point.

Position = 6000 IU (0x001770)

Velocity = 0.00 IU (0x000000)

Time = 50 IU (0x32)

IC = 5 (0x05)

Send the following message:

COB-ID	Data
406	70 17 00 00 00 00 32 0A

19. Send the 7th PVT point.

Position = 5127 IU (0x001407)

Velocity = -7.5 IU (0xFFF880)

Time = 240 IU (0xF0)

IC = 6 (0x06)

Send the following message:

COB-ID	Data
406	07 14 80 00 F8 FF F0 0C

20. Send the 8th PVT point.

Position = 3115 IU (0x000C2B)

Velocity = -13.33 IU (0xFFF2AB)

Time = 190 IU (0xBE)

IC = 7 (0x07)

Send the following message:

COB-ID	Data
406	2B 0C AB 00 F2 FF BE 0E

21. Send the 9th PVT point.

Position = -1686 IU (0xFFF96A)

Velocity = -13.33 IU (0xFFF2AB)

Time = 360 IU (0x168)

IC = 8 (0x08)

Send the following message:

COB-ID	Data
406	6A F9 AB FF F2 FF 68 11

22. Send the 10nth PVT point.

Position = -7145 IU (0xFFE417)

Velocity = -13.33 IU (0xFFF2AB)

Time = 410 IU (0x19A)

IC = 9 (0x0A)

Send the following message:

COB-ID	Data
406	17 E4 AB FF F2 FF 9A 13

23. Send the 11th PVT point.

Position = -9135 IU (0xFFDC51)

Velocity = -7.4 IU (0xFFF899)

Time = 190 IU (0xBE)

IC = 10 (0x0A)

Send the following message:

COB-ID	Data
406	51 DC 99 FF F8 FF BE 14

24. Send the 12th PVT point. The last.

Position = -10000 IU (0xFFD8F0)

Velocity = -7.4 IU (0x000000)

Time = 240 IU (0xF0)

IC = 11 (0x0B)

Send the following message:

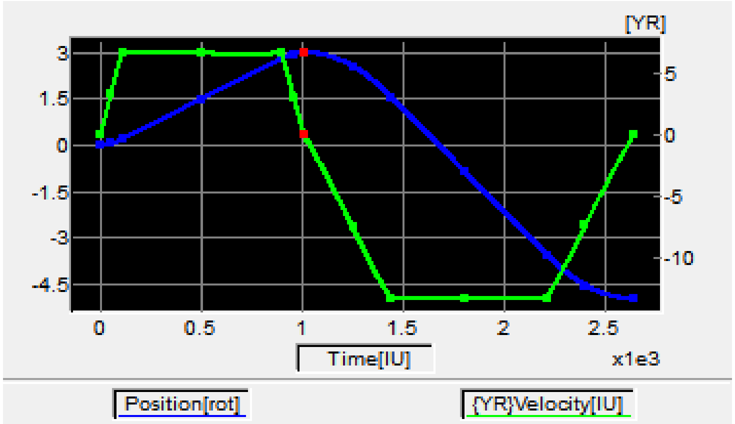
COB-ID	Data
406	F0 D8 00 FF 00 00 F0 16

25. Start an absolute motion.

Send the following message:

COB-ID	Data
206	1F 00

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.

10.6 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**. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	01 06

2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	06 00

3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	07 00

4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	0F 00

5. **Enable the legacy interpolated mode.** Set bit 8 of object 208Eh to 0.

Send the following message (SDO access to object 208Eh sub-index 0, 16-bit value 0):

COB-ID	Data
606	2B 8E 20 00 00 00 00 00

6. **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):

COB-ID	Data
606	2F 02 16 00 00 00 00 00

7. **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):

COB-ID	Data
606	23 02 16 01 20 01 C1 60

   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):

COB-ID	Data
606	23 02 16 02 20 02 C1 60

8. **Enable the RPDO3.** Set the object 1602h sub-index 0 with the value 2.

Send the following message (SDO access to object 1601h sub-index 0, 8-bit value 2):

COB-ID	Data
606	2F 02 16 00 02 00 00 00

9. **Mode of operation.** Select interpolation position mode.

Send the following message (SDO access to object 6060h, 8-bit value 7h):

COB-ID	Data
606	2F 60 60 00 07 00 00 00

10. **Set the relative motion bit.** Set in Controlword mapped in RPDO1 the value 4Fh. For an absolute motion, set 0Fh but the example points will not apply.

Remark: *if the relative motion bit is not set in Controlword before the PVT points are loaded, the trajectory will not be calculated correctly.*

Send the following message:

COB-ID	Data
206	4F 00

11. **Interpolation sub mode select.** Select PVT interpolation position mode.

Send the following message (SDO access to object 60C0h, 16-bit value FFFFh):

COB-ID	Data
606	2B C0 60 00 FF FF 00 00

12. **Interpolated position buffer length.** Set the buffer length to 12. The maximum length is 15.

Send the following message (SDO access to object 2073h, 16-bit value Ch):

COB-ID	Data
606	2B 73 20 00 0C 00 00 00

13. **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):

COB-ID	Data
606	2B 74 20 00 01 A0 00 00

14. **Interpolated position initial position.** Set the initial position to 0 rotations. The object should receive the drives actual position in Internal Units which can be read from object 6063h or 6062h when using steppers in open loop.

Send the following message (SDO access to object 2079h, 32-bit value 0h):

COB-ID	Data
606	23 79 20 00 00 00 00 00

15. **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

Send the following message:

COB-ID	Data
406	90 01 00 00 03 00 FA 02

16. **Send the 2nd PVT point**.

Position = 1240 IU (0x0004D8)

Velocity = 6.00 IU (0x000600)

Time = 250 IU (0xFA)

IC = 2 (0x02)

Send the following message:

COB-ID	Data
406	D8 04 00 00 06 00 FA 04

17. **Send the 3rd PVT point.**

Position = 1674 IU (0x00068A)

Velocity = 6.00 IU (0x000600)

Time = 250 IU (0xFA)

IC = 3 (0x03)

Send the following message:

COB-ID	Data
406	8A 06 00 00 06 00 FA 06

18. **Send the 4th PVT point.**

Position = 1666 IU (0x000682)

Velocity = 6.00 IU (0x000600)

Time = 250 IU (0xFA)

IC = 4 (0x04)

Send the following message:

COB-ID	Data
406	82 06 00 00 06 00 FA 08

19. Send the 5th PVT point.

Position = 1240 IU (0x0004D8)

Velocity = 3.00 IU (0x000300)

Time = 250 IU (0xFA)

IC = 5 (0x05)

Send the following message:

COB-ID	Data
406	**D8 04 00 00 03 00 FA 0A**

20. Send the last PVT point.

Position = 410 IU (0x00019A)

Velocity = 0.00 IU (0x000000)

Time = 250 IU (0xFA)

IC = 6 (0x06)

Send the following message:

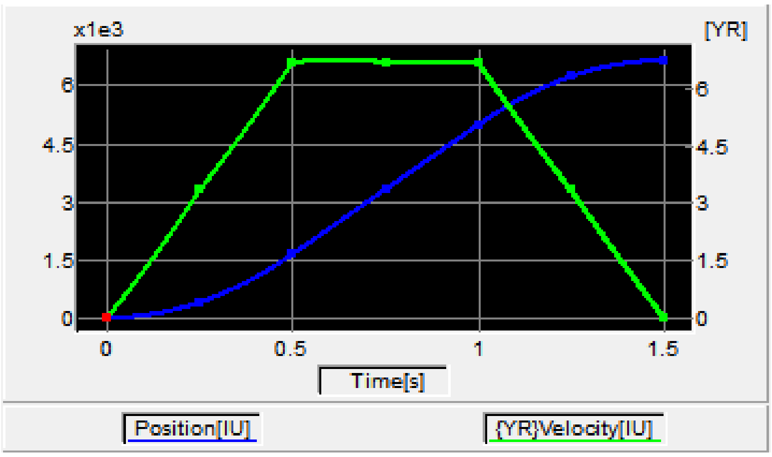
COB-ID	Data
406	**9A 01 00 00 00 00 FA 0C**

21. Start a relative motion.

Send the following message:

COB-ID	Data
206	**5F 00**

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.

11 Cyclic Synchronous Position mode (CSP)

11.1 Overview

The overall structure for this mode is shown in **Figure 10.1.1**. 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.

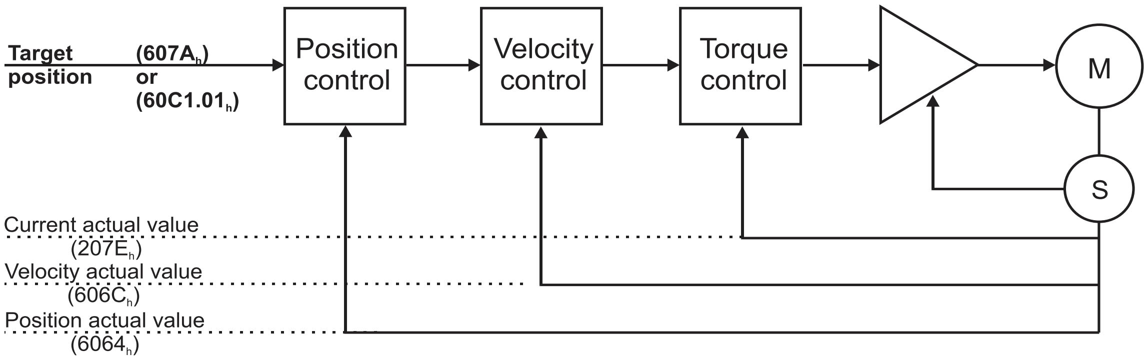


Figure 10.1.1. *Cyclic synchronous position mode overview*

The Target Position for the CSP mode may be received into object 607A$_h$ or into object 60C1$_h$ sub-index 01.

11.1.1 Controlword in Cyclic Synchronous Position mode (CSP)

MSB LSB

See 6040$_h$	Halt	See 6040$_h$	Abs / rel	Reserved	Reserved	See 6040$_h$
15 9	8	7	6	5	4	3 0

Table 11.1 *– 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 607A$_h$. This is the standard mode.

In Relative position mode, the drive will add to its current position the value received in object 607A$_h$. By sending this value periodically and setting the correct interpolation period time in object 60C2$_h$, it will be like working in Cyclic Synchronous Velocity mode (CSV).

11.1.2 Statusword in Cyclic Synchronous Position mode (CSP)

MSB **LSB**

See 6041$_h$	Following error	Target position ignored	See 6041$_h$	Reserved	See 6041$_h$
15 14	13	12	11	10	9 0

Table 11.2 *– 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

11.2 Cyclic Synchronous Position Mode Objects

11.2.1 Object 60C2$_h$: Interpolation time period

The **Interpolation time period** indicates the configured interpolation cycle time. Its value must be set with the time value of the CANopen 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 2ms, 60C2$_h$ sub-index 01$_h$ = 2 and 60C2$_h$ sub-index 02$_h$ = -3. The result is 2ms = $2*10^{-3}$.

Remark: The CSP cycle time is naturally limited by the bandwidth of the CAN bus (and thus by the number of drive controllers connected to the CAN network) and by how precisely the higher-level master controller performs the synchronization. The minimum recommended value is 2 ms. Please contact Technosoft for more information.

Object description:

Index	60C2$_h$
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	2

Sub-index	02$_h$
Description	Interpolation time index
Access	RW
PDO mapping	Possible
Value range	INTEGER8, (-128 to +63)
Default value	-3

11.2.2 Object 2086$_h$: Limit speed for CSP[1]

This object is used to set a maximum velocity during CSP mode of operation.

Object description:

Index	2086$_h$
Name	Limit speed/acceleration for CSP
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	0000$_h$

If 2086$_h$ = 1, the limit is active. During CSP mode, the maximum velocity will be the one defined in object 6081$_h$.

Remark: If 6081$_h$ = 0 and 2086$_h$ =1, during CSP mode, the motor will not move when it receives new position commands because its maximum velocity is limited to 0.

[1] Available only with firmware versions F514x and FA01x.

11.3 Cyclic Synchronous Position Mode example

Short description of the example:

- Start the node
- Remap RPDO1 and set it as synchronous
- Remap TPDO1 and set it as synchronous
- Set CSP mode in Modes of Operation
- Set Operation Enable. The handshake between what is commanded into Controlword and what is read from Statusword will be described in detail
- Send a typical CSP motion command.

Step 1 starts the remote node 6, which means the PDOs will be enabled.

1. **Start remote node**. Send an NMT message to start the node id **06**.

Send the following message:

COB-ID	Data
0	**01 06**

Remark: if **00** is sent instead of 06, all nodes in the network will be enabled.

Steps 2 and 3 set the interpolation time to 2ms.

The interpolation time needs to be set in the object $60C2_h$. Sub-index 1 holds the interpolation time period value (i.e. 2 for 2ms) and sub-index 2 holds the interpolation time index (i.e. -3 for ms = 10^-3 s).

The interpolation time has to be equal to the SYNC period and the period of the synchronous RPDO containing the position command.

2. **Interpolation time period value.** Set the interpolation time value to 2 (0x02).

Send the following message (SDO write access to object $60C2_h$ sub-index 1 the 8-bit value 02_h):

COB-ID	Data
606	**2F C2 60 01 02**

3. **Interpolation time index.** Set the interpolation time index value to -3 (0xFD).

Send the following message (SDO write access to object $60C2_h$ sub-index 2 the 8-bit value FD_h):

COB-ID	Data
606	**2F C2 60 02 FD**

Steps 4 to 7 remap RPDO1 to receive Controlword (6040_h, 16bit) and Target Position ($607A_h$, 32bit).

4. **Disable RPDO1 mapping.** To reconfigure any RPDO mapping, sub-index 0 of the corresponding mapping parameter object must be set to 0 in order to disable the PDO mapping.

Send the following message (SDO write access to object 1600_h sub-index 0 the 8-bit value 00_h):

COB-ID	Data
606	**2F 00 16 00 00**

5. **Map Controlword 6040_h to RPDO1 sub-index 1.**

Send the following message (SDO write access to object 1600_h sub-index 1 the 32-bit value 60400010_h):

COB-ID	Data
606	**23 00 16 01 10 00 40 60**

6. **Map Target Position $607A_h$ to RPDO1 sub-index 2.**

Send the following message (SDO write access to object 1600_h sub-index 2 the 32-bit value $607A0020_h$):

COB-ID	Data
606	**23 00 16 02 20 00 7A 60**

Remark: instead of $607A_h$, object $60C1_h$ sub-index 01 may also be mapped to receive the same position command. In this case, $60C10120_h$ must be written to sub-index 2 of object 1600_h.

7. **Enable RPDO1 mapping.** To enable any RPDO mapping, sub-index 0 of the corresponding mapping parameter object must be set with the number of sub-indexes defined in it. In this case, there are 2.

Send the following message (SDO write access to object 1600_h sub-index 0 the 8-bit value 02_h):

COB-ID	Data
606	**2F 00 16 00 02**

Steps 8 to 10 set RPDO1 as synchronous.

8. **Disable RPDO1.** To change any RPDO Communication parameters, sub-index 1 bit 31 must be set. It is recommended that only bit 31 is set and the number already defined inside should be kept.

   Example: the sub-index 1 value is 0x206 which is the RPDO1 COB ID for axis 6 (0x200 + Axis ID). From this number, only bit 31 should be set. It means that instead of 0x206, 0x80000206 should be written.

Send the following message (SDO write access to object 1400_h sub-index 1 the 32-bit value 80000206_h):

COB-ID	Data
606	23 00 14 01 06 02 00 80

9. **Set RPDO1 as synchronous, with the period of 1 SYNC.** Write 1 into sub-index 2 Transmission type. RPDO1 data will be processed after the reception of each SYNC.

Send the following message (SDO write access to object 1400_h sub-index 2 the 8-bit value 01_h):

COB-ID	Data
606	2F 00 14 02 01

10. **Enable RPDO1.** To enable a RPDO, bit 31 of sub-index 1 must be reset without interfering with the other bits. For the RPDO1 of axis 6, the COB ID should be (0x200 + axis ID). It means 0x206 should be written.

Send the following message (SDO write access to object 1400_h sub-index 1 the 32-bit value 00000206_h):

COB-ID	Data
606	23 00 14 01 06 02 00 00

Steps 11 to 14 remap TPDO1 to send Statusword (6041_h, 16bit) and Position actual value (6064_h, 32bit).

11. **Disable TPDO1 mapping.** To reconfigure any TPDO mapping, sub-index 0 of the corresponding mapping parameter object must be set to 0 in order to disable the PDO mapping.

Send the following message (SDO write access to object $1A00_h$ sub-index 0 the 8-bit value 00_h):

COB-ID	Data
606	2F 00 1A 00 00

12. **Map Statusword 6041_h to TPDO1 sub-index 1.**

Send the following message (SDO write access to object $1A00_h$ sub-index 1 the 32-bit value 60410010_h):

COB-ID	Data
606	23 00 1A 01 10 00 41 60

13. **Map Position actual value 6064_h to TPDO1 sub-index 2.**

Send the following message (SDO write access to object $1A00_h$ sub-index 2 the 32-bit value 60640020_h):

COB-ID	Data
606	23 00 1A 02 20 00 64 60

14. **Enable TPDO1 mapping.** To enable any TPDO mapping, sub-index 0 of the corresponding mapping parameter object must be set with the number of sub-indexes defined in it. In this case, there are 2.

Send the following message (SDO write access to object $1A00_h$ sub-index 0 the 8-bit value 02_h):

COB-ID	Data
606	2F 00 1A 00 02

Steps 15 to 17 set TPDO1 as synchronous.

15. **Disable TPDO1.** To change any TPDO Communication parameters, sub-index 1 bit 31 must be set. It is recommended that only bit 31 is set and the number already defined inside should be kept.

    Example: the sub-index 1 value is 0x186 which is the TPDO1 COB ID for axis 6 (0x180 + Axis ID). From this number, only bit 31 should be set. It means that instead of 0x186, 0x80000186 should be written.

Send the following message (SDO write access to object 1800_h sub-index 1 the 32-bit value 80000186_h):

COB-ID	Data
606	23 00 18 01 86 01 00 80

16. **Set TPDO1 as synchronous, with the period of 1 SYNC.** Write 1 into sub-index 2 Transmission type. TPDO1 data is updated when the SYNC is received, and then TPDO1 is sent as soon as possible.

Send the following message (SDO write access to object 1800_h sub-index 2 the 8-bit value 01_h):

COB-ID	Data
606	2F 00 18 02 01

17. **Enable TPDO1.** To enable a TPDO, bit 31 of sub-index 1 must be reset without interfering with the other bits. For the TPDO1 of axis 6, the COB ID should be (0x180 + axis ID). It means 0x186 should be written.

Send the following message (SDO write access to object 1800_h sub-index 1 the 32-bit value 00000186_h):

COB-ID	Data
606	23 00 18 01 86 01 00 00

Step 18 sets CSP mode into the Modes of operation object.

18. Set modes of operation to CSP. Write 0x08 into object 6060h to set the drive into CSP mode.

Remark: the drive will be in CSP mode only after in reaches the state Operation Enabled. This means that object 6061h (Modes of operation display) will show 8 (drive is in CSP mode), only after Operation Enabled has been reached.

Send the following message (SDO write access to object 6060h sub-index 0 the 8-bit value 08h):

COB-ID	Data
606	2F 60 60 00 08

Steps 19 to 21 bring the drive into *Operation enabled* state and also start the CSP mode motion.

Remark 1: from this point on, the master should send the SYNC messages at precisely 2ms (the same number defined in 60C2h). Transmission of RPDO1 should also be started by the master.

Remark 2: the SYNC message is usually configured at the CANopen master start-up and can be sent from the drive boot-up time. The configuration messages until this point can be sent in parallel with the SYNC messages. Only after all the PDOs are configured as synchronous, the drive will use the SYNC signal for the PDOs.

19. Ready to switch on. Change the node state from *Switch on disabled* to *Ready to switch on* by sending the *shutdown* command via Controlword associated PDO.

Send the following message (SYNC)

COB-ID	Data
80	Null

This was the SYNC signal. It must be sent at precisely 2ms intervals. In this example it can also be sent manually, to understand each command and what it does.

Send the following message (RPDO1)

COB-ID	Data
206	06 00 00 00 00 00

The **0006** is the new value for Controlword, i.e. the command to enter *Ready to switch on* state.

The **00000000** is the position command.

Send the following message (SYNC)

COB-ID	Data
80	Null

This was the SYNC signal. It must be sent at precisely 2ms intervals.

After each SYNC signal, the drive will send its TPDO1. To be able to change the next Controlword command in RPDO1, ensure that the drive reaches *Ready to switch on* state by waiting for the TPDO1 with the following content:

Wait to receive the following message (TPDO1)

COB-ID	Data
186	31 02 00 00 00 00

The **0231** is the Statusword value. The value xx31h shows that the drive reached *Ready to switch on* state. The master may have to wait a few SYNCs and read the TPDOs multiple times until this value is reached (there are also intermediary values)

The **00000000** is the Position actual value and can vary depending on the encoder reported position.

Warning: The master must always wait for the drive to reach the desired state programmed into Controlword by checking the Statusword. No other command must be sent during this time. In this case, because the RPDOs are synchronous, the RPDO1 must be sent continuously without changing the command in Controlword until the drive reaches the desired state as reported into the Statusword.

20. Switch on. Change the node state from *Ready to switch on* to *Switched on* by sending the *switch on* command via Controlword associated PDO.

Send the following message (SYNC)

COB-ID	Data
80	Null

This was the SYNC signal. It must be sent at precisely 2ms intervals.

Send the following message (RPDO1)

COB-ID	Data
206	07 00 00 00 00 00

The 0007 is the new value for Controlword, i.e. the command to enter *Switched on* state.

Send the following message (SYNC)

COB-ID	Data
80	Null

This was the SYNC signal. It must be sent at precisely 2ms intervals.

After each SYNC signal, the drive will send its TPDO1. To be able to change the next Controlword command in RPDO1, ensure that the drive reaches *Switched on* state by waiting for the TPDO1 with the following content:

Wait to receive the following message (TPDO1)

COB-ID	Data
186	33 82 00 00 00 00

The 8233 is the Statusword value. The value xx33h shows that the drive reached *Switched on* state. The master may have to wait a few SYNCs and read the TPDOs multiple times until this value is reached (there are also intermediary values).

At this step, the drive starts applying power to the motor. The time to reach *Switched on* state depends on the motor initialization method and its parameters (the *Start method* as defined in the Drive Setup Dialogue in ESM). Initialization times of up to 2s are not uncommon.

Warning: The master must always wait for the drive to reach the desired state programmed into Controlword by checking the Statusword. No other command must be sent during this time. In this case, because the RPDOs are synchronous, the RPDO1 must be sent continuously without changing the command in Controlword until the drive reaches the desired state as reported into the Statusword.

After the drive reaches *Switched On* state, the master can continue to the next step.

21. Enable operation. Change the node state from *Switched on* to *Operation enabled* by sending the *Enable operation* command via Controlword associated PDO.

Send the following message (SYNC)

COB-ID	Data
80	null

This was the SYNC signal. It must be sent at precisely 2ms intervals.

Send the following message (RPDO1)

COB-ID	Data
206	0F 00 00 00 00 00

The 000F is the command to enter *Operation enable* state in Controlword.

Send the following message (SYNC)

COB-ID	Data
80	null

This was the SYNC signal. It must be sent at precisely 2ms intervals.

After each SYNC signal, the drive will send its TPDO1. Ensure that the drive reaches *Operation enabled* state by waiting for the TPDO1 with the following content:

Wait for the following message (TPDO1)

COB-ID	Data
186	37 96 00 00 00 00

The 9637 is the Statusword value. The value xx37h shows that the drive reached *Operation enable* state. The master may have to wait a few SYNCs and read the TPDOs multiple times until this value is reached (there are also intermediary values).

From this step forward, the motor will execute a motion within 2ms to the absolute position given into RPDO1 as the Target position.

Step 22 describes a CSP motion command:

22. Move to 100 IU. Set the position command to 100 IU.

Send the following message (SYNC)

COB-ID	Data
80	null

This was the SYNC signal. It must be sent at precisely 2ms intervals. The drive will process the previously received RPDO immediately after the reception of the SYNC.

Send the following message (RPDO1)

COB-ID	Data
206	0F 00 64 00 00 00

The **000F** is the command to enter or remain in *Operation enabled* state in Controlword.

The **00000064** is the position command (=100 in decimal).

Send the following message (SYNC)

COB-ID	Data
80	null

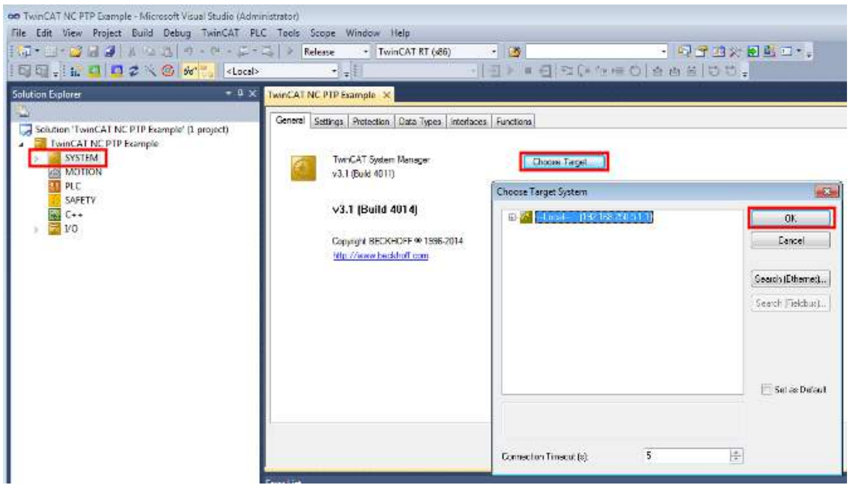
After this SYNC, the motor will start to travel to the absolute position 100 over the following 2ms. The drive also sends the TPDO1 reporting the position of the motor sampled at the SYNC reception.

The master then needs to cyclically send the SYNC and RPDO1 with updated position commands.

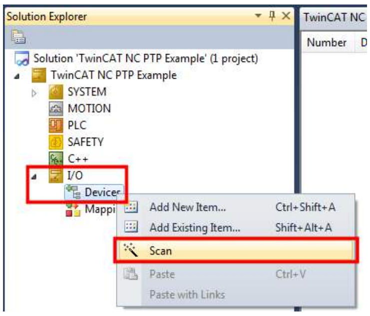
11.4 Configuring Technosoft CANopen Drives for NC-PTP (CSP) operation in TwinCAT 3

11.4.1 Create a new project and scan for the drives

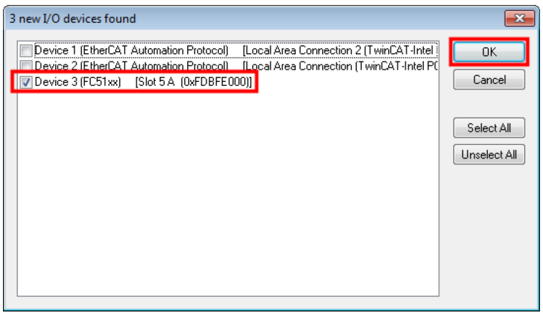
Start the TwinCAT 3 XAE programming environment and create a new project.



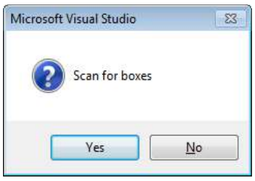
Choose your target system where the CANopen interface is located.



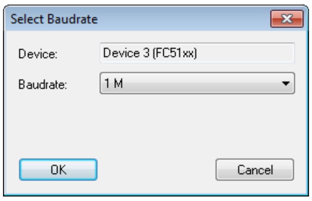
In Solution Explorer, expand the I/O section, right-click on devices and choose Scan.



Depending on the available devices, select only the CAN interface and click OK.

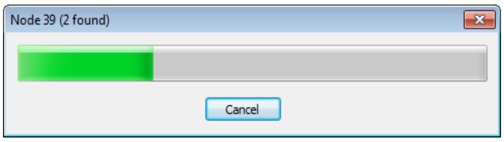


A scan for boxes prompt will appear. Click Yes to find the available CAN drives.



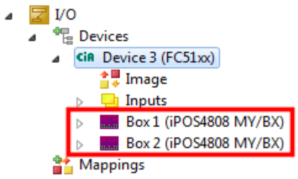
Another prompt appears for baudrate selection. Select the used baudrate and click OK.

Remark: the default baudrate for all Technosoft drives is 500 kbps if not defined otherwise in Drive Setup.



A scan progress bar will show how many nodes are found. Wait for it to finish, or just click cancel if all the nodes are detected.

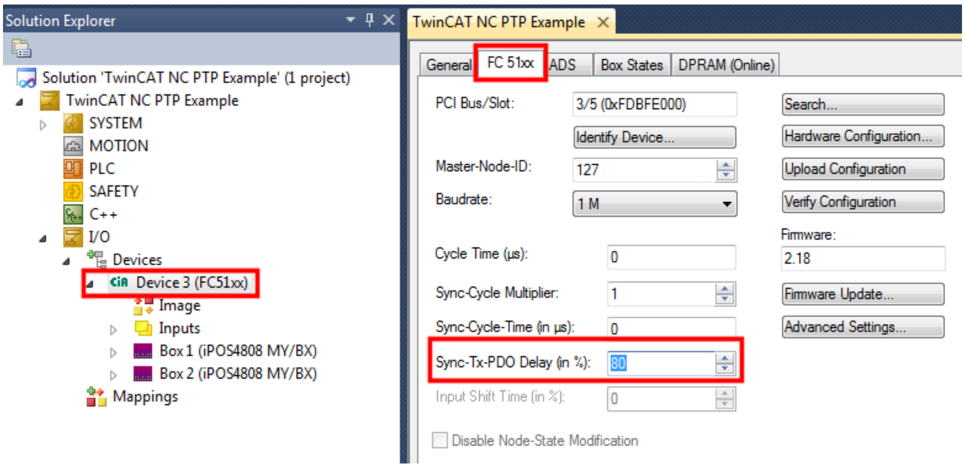
Remark: on the example test system, the first scan does not find any drive. They are found on the second scan.



The new found nodes will be available in the Devices area. The Box number is actually the found CAN ID number.

Remark: The CANopen ID number is the same as the Technosoft AxisID number that can be defined in Drive Setup.

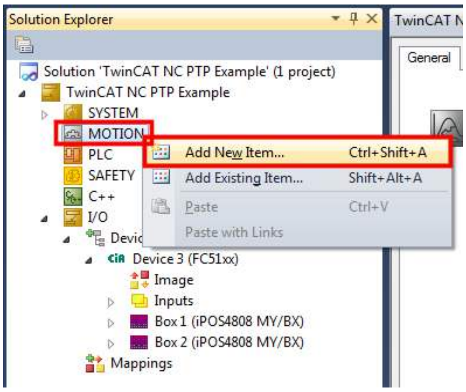
11.4.2 Setting the Sync-TxPDO Delay



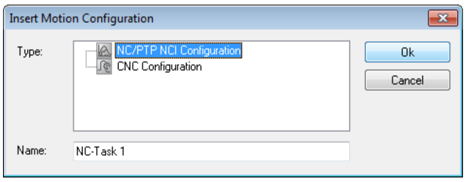
In the Solution Explorer, click on the device with the CAN interface. Select the CAN interface configuration tab and select at Sync-Tx-PDO Delay a higher number than 30%. On some systems, if the time difference between the Sync message (sent by the master) and the Synchronous TxPDO (sent by the drive) exceeds x% of the communication time, TwinCAT considers it as an error, power off the drive and restarts it again. By increasing this time, such sudden power offs will be avoided.

Remark: on the test system, the value of 80% eliminated all issues. On another system, the value of 80% cause the remote device not to communicate anymore. Choose the highest value while being still able to communicate with the CAN nodes.

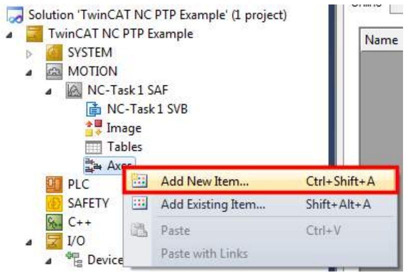
11.4.3 Adding new Nc-PTP axes



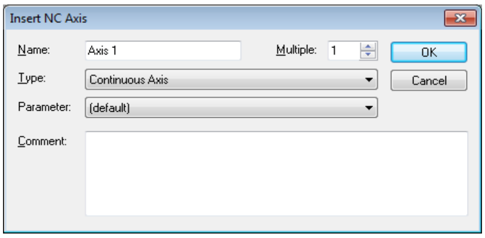
Right Click MOTION and choose add new item... .



A new prompt will come up. Choose the NC/PTP NCI Configuration and click OK.



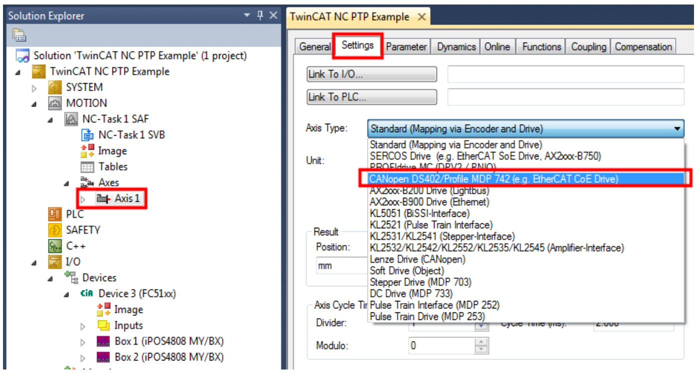
Right Click Axes under the NC-Task and choose the Add New Item...



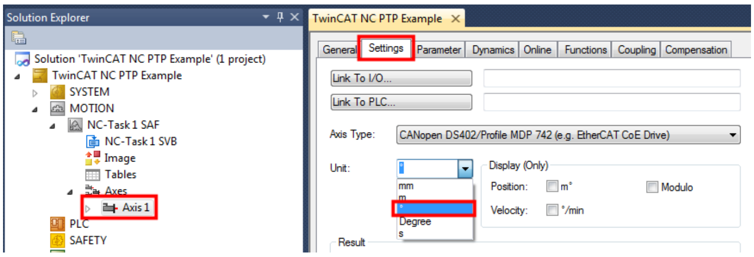
A new prompt will come up. Click OK to add the axis.

If more axis need to be defined, they can be copy-pasted later, after more setting are done.

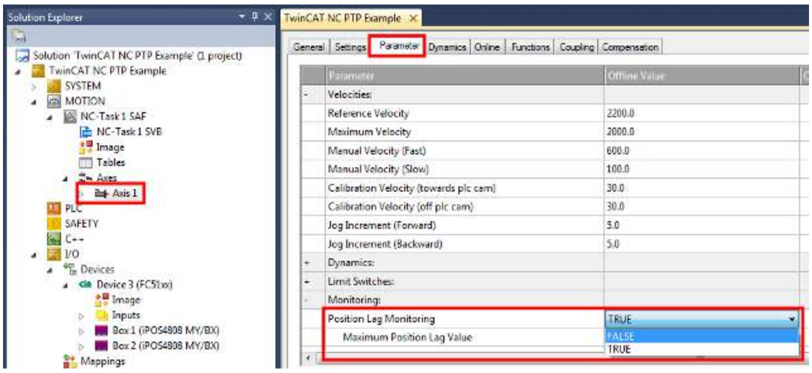
11.4.4 NC-PTP Axis settings



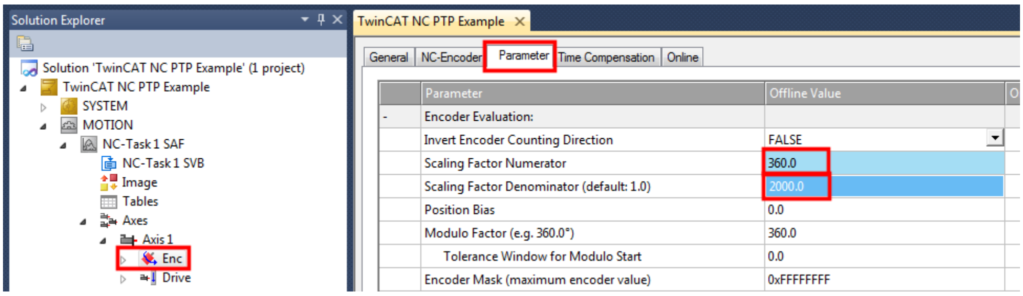
Click the Axis 1, choose the settings tab and select under Axis Type, the CANopen DS402... type.



Under the same settings, you can choose the motor units like mm or geometrical degrees.



Select the parameters tab and set the Position Lag Monitoring to FALSE. This is a TwinCAT protection that monitors the difference between the motor actual position and commanded position. This protection is already present in the Technosoft drive as the Control Error setting, which often is more precise and quicker to react, because the drive internal clock (default 1ms) is usually faster than the CAN communication cycle times (min 2ms). When the drive detects a control error, it will enter Fault state and TwinCAT will stop normal operation.

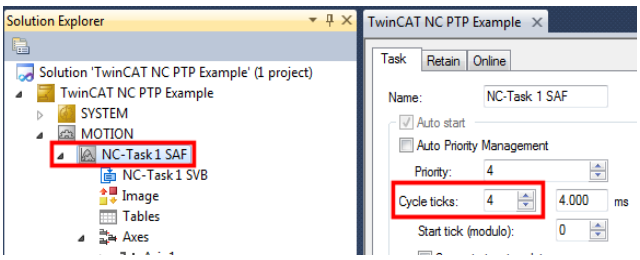


Under NC Task/ Axes / Axis 1/, click the Enc and then choose the Parameter tab on the right hand side.

Because the example uses a rotary motor, write 360 in the Scaling Factor Numerator field. 360 stands for mechanical degrees. In the Scaling Factor Denominator, write 2000 or whatever value it takes in encoder increments for one full motor rotation. In the example case, a 500 lines quadrature encoder was used, resulting in 2000 encoder increments for one motor rotation.

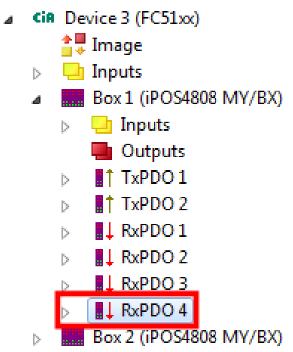
If using a Stepper Open Loop without a feedback, set the denominator to the entire number of microsteps it takes to do one full motor rotation. In addition, in case there is no feedback available, map object 6062_h instead of 6064_h (see later mapping explanations).

11.4.5 Setting the CAN communication cycle time

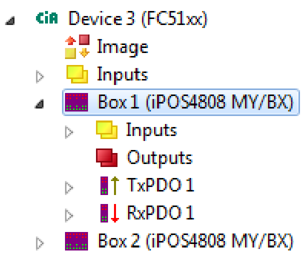


Click the NC-Task and select the CAN communication cycle time. In this example, a 4ms cycle time was chosen. It is not recommended to use more than 4 drives at 1Mbit baud and 2ms cycle time. If more drives are present, more CAN data will fill up the bandwidth and some messages might be lost. More performance tests should be done carefully if the communication settings are tougher.

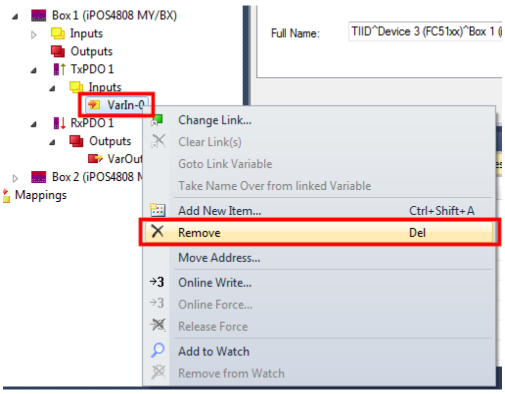
11.4.6 Configuring the TwinCAT PDO layout



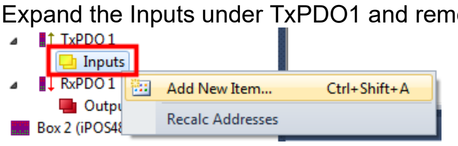
and



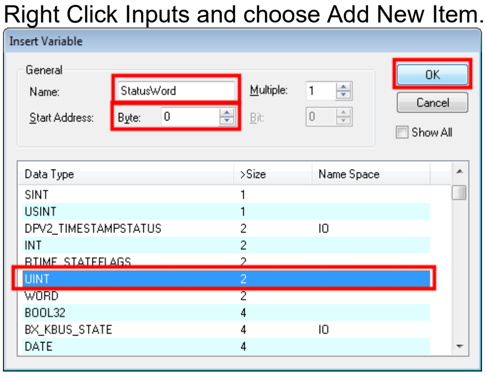
Expand Box 1 and click on RxPDO 4. Press the delete key or choose Edit/ Remove, to delete RxPDO4. Do the same for RxPDO3, 2 and TxPDO2. Leave only TxPDO1 and RxPDO1 active. The less PDO data is active, the less data will be transmitted on CAN and more drives can be added to the network while keeping a low communication cycle time.



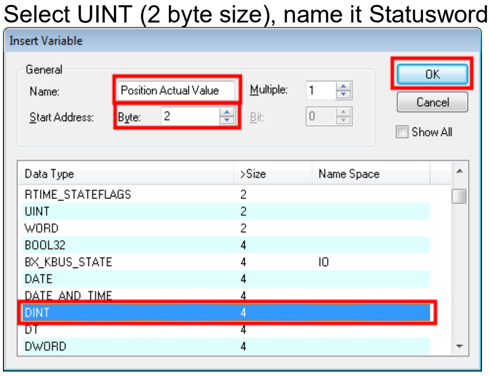
Expand the Inputs under TxPDO1 and remove the VarIn-0. Also, remove the VarOut-0 from the RxPDO1.



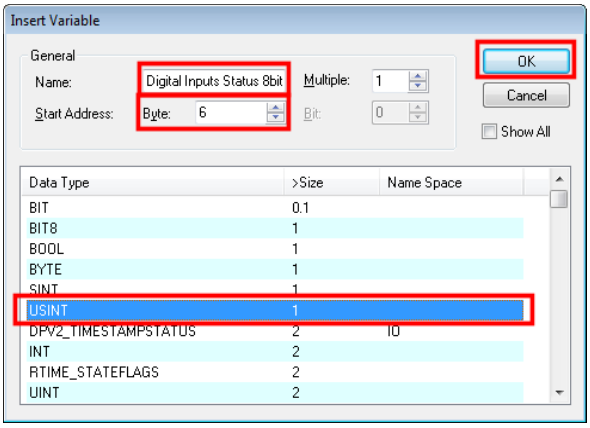
Right Click Inputs and choose Add New Item... .



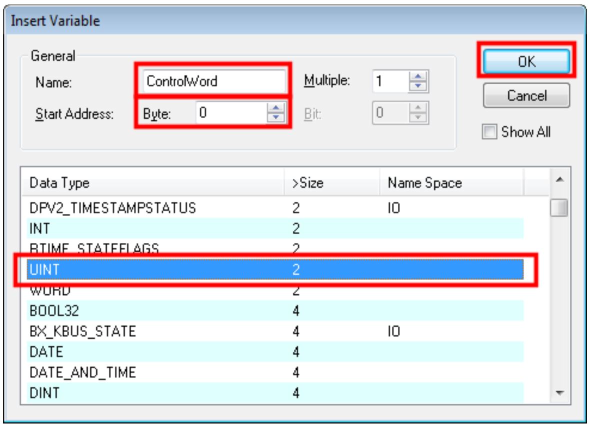
Select UINT (2 byte size), name it Statusword and set the Start Address 0. Click OK.



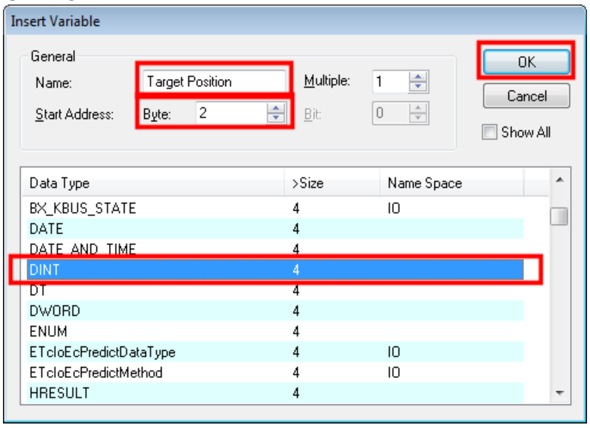
Add another item under Inputs. Select DINT (4 byte size), name it Position actual value and set the Start Address 2. Click OK. This means that the 4 byte position actual value will be found starting with byte 2 of the TxPDO1 data.



(This step is optional) Add another item under Inputs. Select USINT (1 byte size), name it Digital Inputs Status 8bit and set the Start Address 6. Click OK. This means that the 1byte Digital inputs status object will be found starting with byte 6 of the TxPDO1 data.

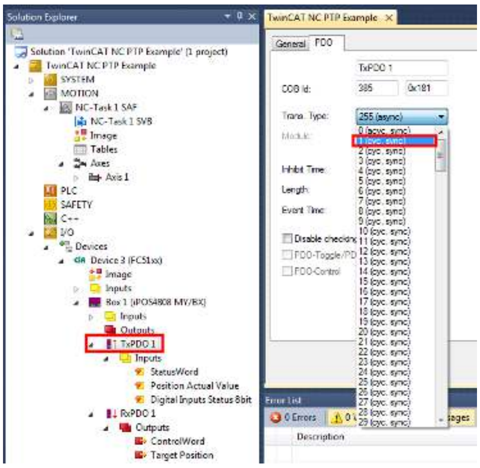


Add an item in RxPDO1 under Outputs. Select UINT (2 byte size), name it Controlword and set the Start Address 0. Click OK.

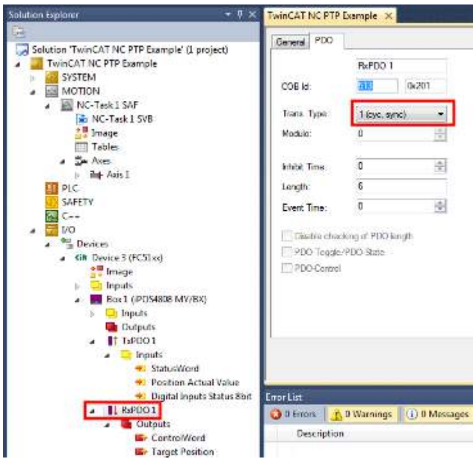


Add another item under Outputs. Select DINT (4 byte size), name it Target Position and set the Start Address 2. Click OK. This means that the 4byte Target position will be found starting with byte 2 of the RxPDO1 data.

11.4.6.1 Setting the PDOs as synchronous



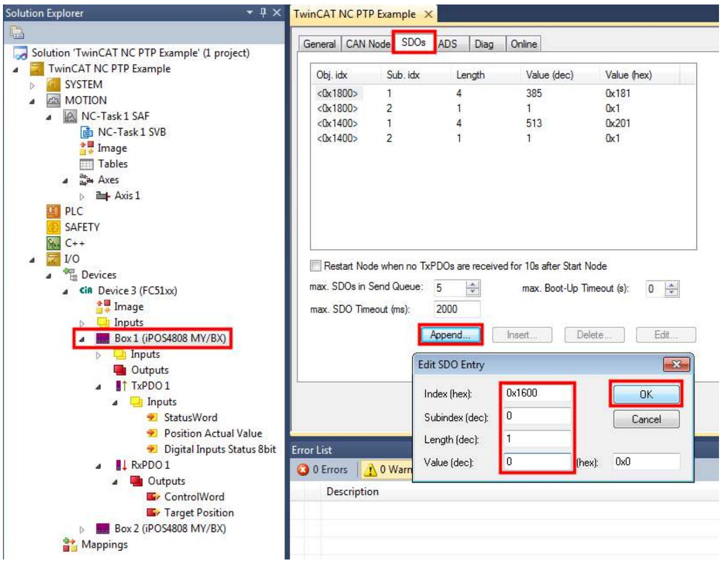
Click TxPDO1 and select the PDO tab on the right side. Set the Trans. type to 1 (cyc, sync). This setting will make TxPDO1 synchronous with every sync message.



Click RxPDO1 and select the PDO tab on the right side. Set the Trans. type to 1 (cyc, sync). This setting will make RxPDO1 synchronous with every sync message.

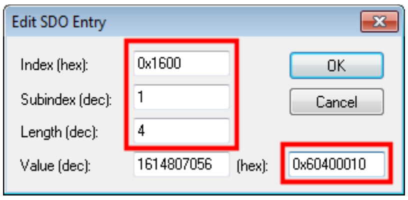
11.4.7 Adding start-up SDO drive configuration messages

11.4.7.1 Mapping objects to RxPDO1



Select Box1; Select the SDOs tab and then click the Append... button to add configuration SDOs. This SDO list will be sent every time the master starts, or it detects that the drive was reset.

First set index 1600h, sub-index 0x0, Length 1, Value 0. Sub-index 0 of object 1600h represents how many objects are mapped in RxPDO1. To be able to define(map) any object, first, sub-index 0 must be set to 0.

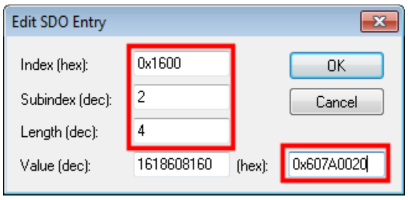


Click the Append.. button to add another SDO.

Set index 1600h, sub-index 0x1, Length 4, Hex Value 0x60400010; This command will map object 6040h (Controlword) to sub-index 1 of object 1600h (RxPDO1) and will represent the first 2 bytes of RxPDO1 data. The length is 4 bytes, because sub-index 1..to 8 of object 1600h is 4 bytes long. The data 0x**60400010** represents the following: **6040** is the mapped object; **00** is the sub-index of the mapped object; **10** is the hex value (16 decimal) of the length in bits of the mapped object sub-index. If it was a 32bit sub-index, it would have been **20.** If it was an 8bit sub-index, it would have been **08.**

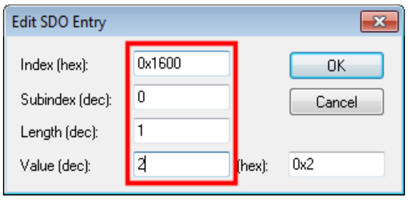
In any Tx or RxPDO, up to 64bit of data can be mapped. This means that all the objects lengths mapped into one PDO must not exceed 64bits (8bytes) of data.

For example, one PDO can support: 1x16 bit object + 1x32bit object + 1x8 bit object + 1x8 bit object.



Click the Append.. button to add another SDO.

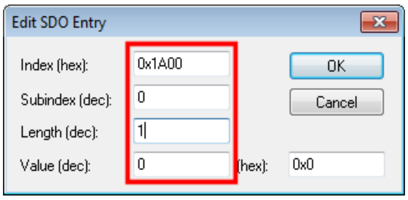
Set index 1600h, sub-index 0x2, Length 4, Hex Value 0x607A0020; This command will map object 607Ah (Target Position) to sub-index 2 of object 1600h (RxPDO1) and will represent the next 4 bytes of RxPDO1 data after the ones occupied by sub-index 1. The data 0x**607A0020** represents the following: **607A** is the mapped object; **00** is the sub-index of the mapped object; **20** is the hex value (32 decimal) of the length in bits of the mapped object sub-index.



Click the Append.. button to add another SDO.

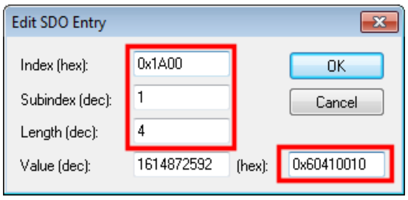
Set index 1600h, sub-index 0x0, Length 1, Value 2. This command will enable the RxPDO1 mapping. Value is set to 2 because two sub-indexes were defined in object 1600h.

11.4.7.2 Mapping objects to TxPDO1



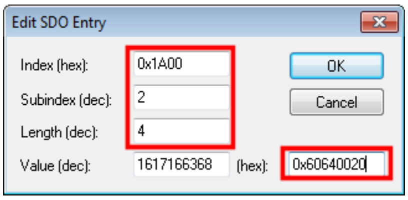
Click the Append.. button to add another SDO.

Set index 1A00h, sub-index 0x0, Length 1, Value 0. Sub-index 0 of object 1A00h represents how many objects are mapped in TxPDO1. To be able to define(map) any object, first, sub-index 0 must be set to 0.



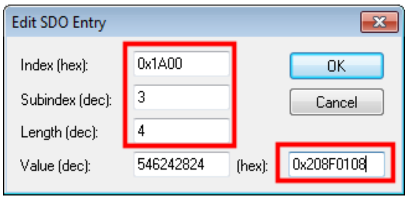
Click the Append.. button to add another SDO.

Set index 1A00h, sub-index 0x1, Length 4, Hex Value 0x60410010; This command will map object 6041h (Statusword) to sub-index 1 of object 1A00h (TxPDO1) and will represent the first 2 bytes of TxPDO1 data. The data 0x**60410010** represents the following: **6041** is the mapped object; **00** is the sub-index of the mapped object; **10** is the hex value (16 decimal) of the length in bits of the mapped object sub-index.



Click the Append.. button to add another SDO.

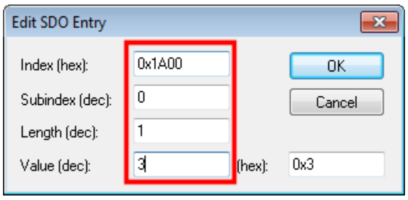
Set index 1A00h, sub-index 0x2, Length 4, Hex Value 0x60640020; This command will map object 6064h (Position Actual Value) to sub-index 2 of object 1A00h (TxPDO1) and will represent the next 4 bytes of TxPDO1 data after the ones occupied by sub-index 1. The data 0x60640020 represents the following: 6064 is the mapped object; 00 is the sub-index of the mapped object; 20 is the hex value (32 decimal) of the length in bits of the mapped object sub-index.



(This step is optional) Click the Append.. button to add another SDO.

Set index 1A00$_h$, sub-index 0x3, Length 4, Hex Value 0x208F0108; This command will map object 208F$_h$ (Digital inputs status 8bit) to sub-index 3 of object 1A00$_h$ (TxPDO1) and will represent the next 1x byte of TxPDO1 data after the ones occupied by sub-index 2. The data 0x**208F0108** represents the following: **208F** is the mapped object; **01** is the sub-index of the mapped object; **08** is the hex value (8 decimal) of the length in bits of the mapped object sub-index.

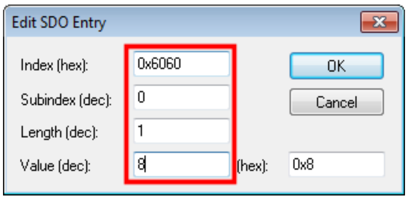
This object is a shorter version of the standard object 60FD$_h$ Digital Inputs Status. Sub-index 1 of 208F$_h$ represents the first 8 bits of 60FD$_h$. The role of using 208F$_h$ instead of 60FD$_h$ is to reduce the number of bits that will be sent over CAN. The drive digital inputs can be later used for the homing procedure.



Click the Append.. button to add another SDO.

Set index 1A00$_h$, sub-index 0x0, Length 1, Value 3. This command will enable the TxPDO1 mapping. Value is set to 3 because three sub-indexes were defined in object 1A00$_h$. If the third sub-index in 1A00$_h$ is not needed, then sub-index 0 should be set with the value 2.

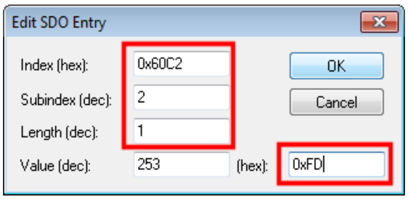
11.4.7.3 Setting Modes of Operation to CSP mode



Click the Append.. button to add another SDO.

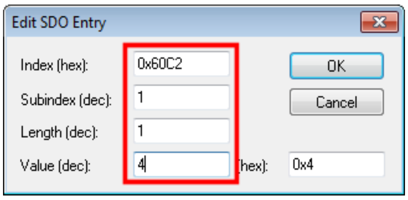
Set index 6060$_h$, sub-index 0x0, Length 1, Value 8. This command will set object 6060$_h$ (Modes of Operation) with the value 8 which is Cyclic Synchronous Position mode.

11.4.7.4 Setting the interpolation object



Click the Append.. button to add another SDO.

Set index 60C2$_h$, sub-index 0x2, Length 1, Hex Value 0xFD. This command will set object 60C2$_h$, sub-index 2 (Interpolation time Period index) with the value 0xFD or decimal -3 because it is a short integer type. A -3 value means milliseconds.

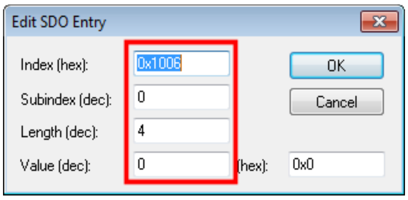


Click the Append.. button to add another SDO.

Set index 60C2$_h$, sub-index 0x1, Length 1, Value 4. This command will set object 60C2$_h$, sub-index 1 (Interpolation time Period value) with the value 4 which will mean 4ms. Because the example is set at 4ms, sub-index 1 is set at 4. If the CAN communication cycle has another value, then sub-index 1 must be set with that value.

The interpolation time must always represent 1x or multiples of the drive slow loop time which is set by default to 1ms.

11.4.7.5 Setting object 1006$_h$ to 0; Synchronization issue workaround



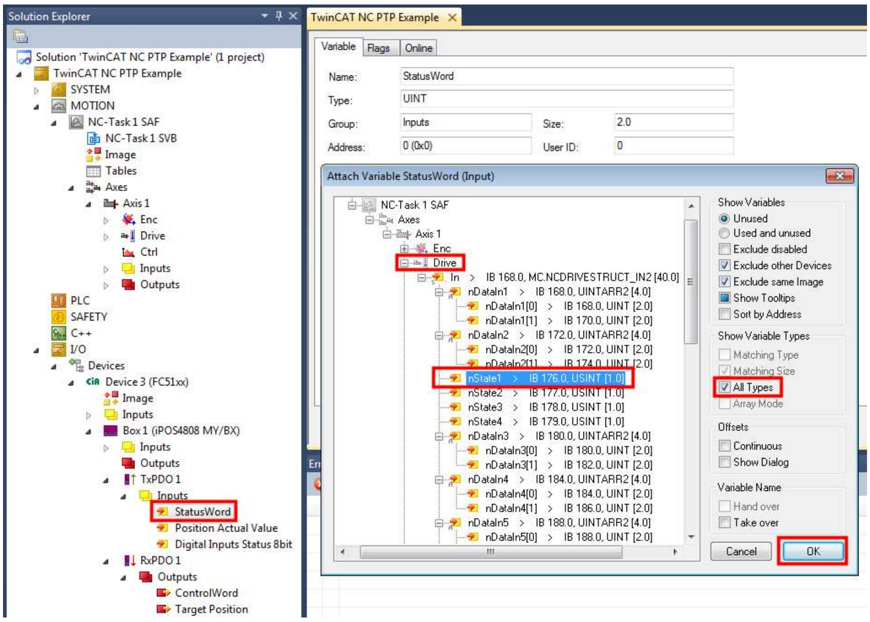
(This step is optional) On F508J/F509J and F514D firmware, if object 1006h receives a non-zero value, the drive will not synchronize when receiving sync messages. TwinCAT automatically sets this object to a non-zero value without being able to stop this behavior. A workaround is to set an SDO to write 0 again in 1006h. If the firmware on the Technosoft drive is newer than the ones mentioned, this step is no longer necessary.

Click the Append.. button to add another SDO.

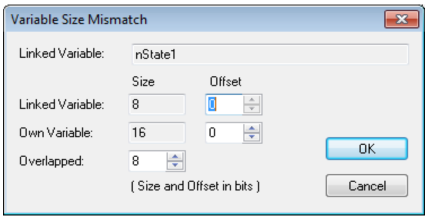
Set index 1006h, sub-index 0x0, Length 4, Value 0.

11.4.8 Linking drive PDO data variables to internal NC-PTP variables

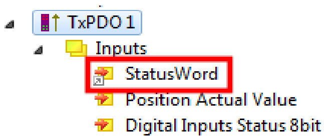
11.4.8.1 Linking standard NC-PTP variables



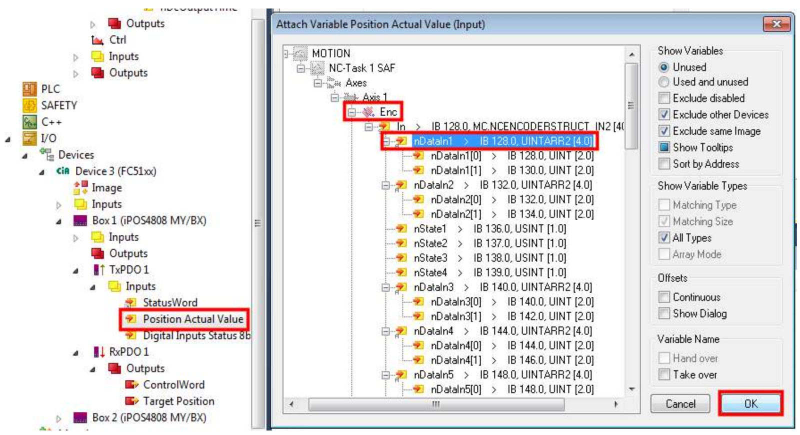
In Box1/ TxPDO1/ Inputs/, double click the Statusword variable, or right click and select change link. A new window called "Attach Variable Statusword" will appear. On the right hand side, select All types. Under NC-Task, Axis 1, Drive, select the nState1 variable and click OK.



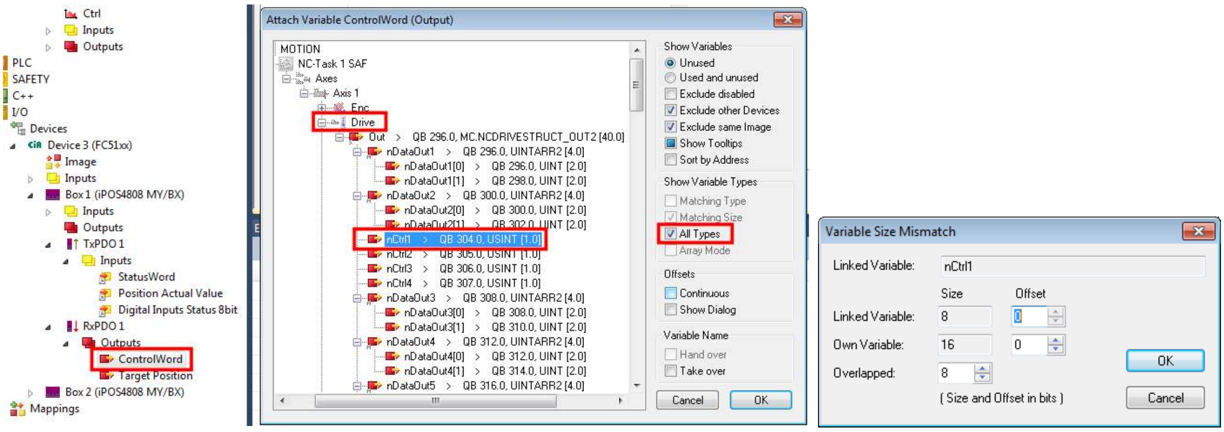
A new window will appear, because a 16 bit variable will be linked over an 8 bit variable. Leave the settings as they are and click OK.



Once a variable is linked, it will have a small arrow icon in front of it. The link can be changed, deleted or view the other linked variable by using the right click mouse menu.

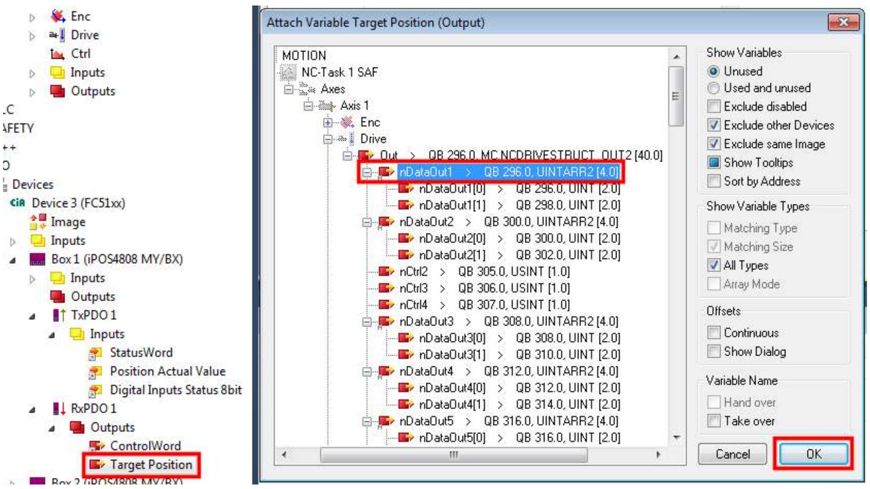


In Box1/ TxPDO1/ Inputs/, double click the Position Actual Value variable. Under NC-Task, Axis 1, <u>Enc</u>, select the nDataIn1 variable and click OK. Because both variables are 32bit, they will link directly.



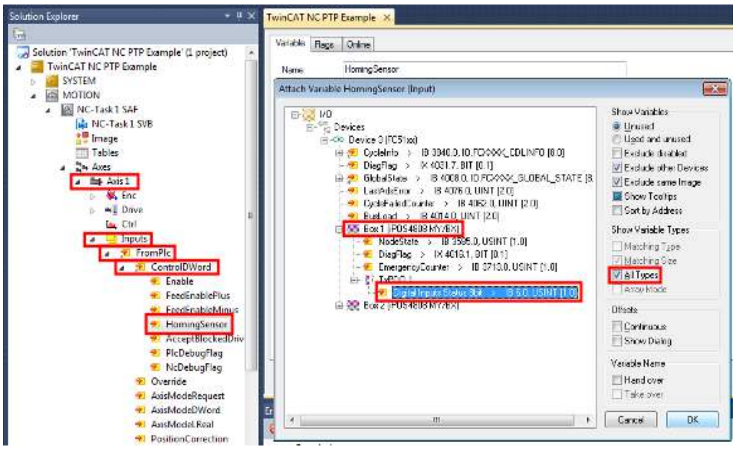
In Box1/ RxPDO1/ Outputs/, double click the Controlword variable. Under NC-Task, Axis 1, <u>Drive</u>, select the nCtrl1 variable and click OK. The Show Variables/ All Types should be checked.

In the new menu that appears, just click OK.

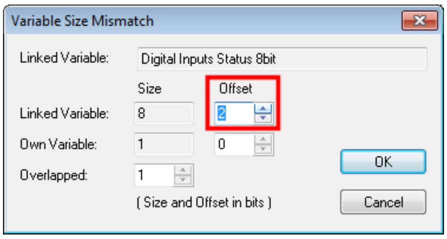


In Box1/ RxPDO1/ Outputs/, double click the Target Position variable. Under NC-Task, Axis 1, <u>Drive</u>, select the nDataOut1 variable and click OK.

11.4.8.2 Linking the home input IN0 to the HomingSensor of the NC-PTP interface

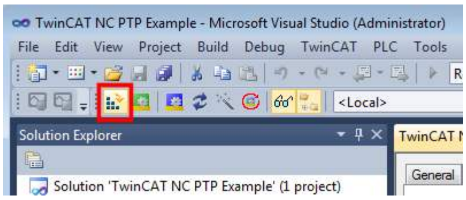


Under NC-Task/ Axis 1/ Inputs/ FromPlc/ ControlDWord/, double click the HomingSensor variable to link it. In the menu that appears, select All types on the right hand side. Link it to Digital Inputs Status 8bit variable.

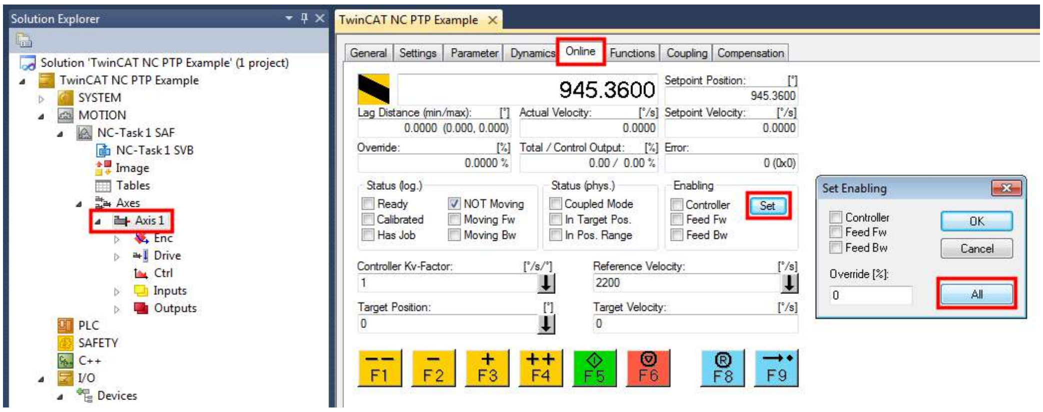


A new menu will appear. The bool (1bit) variable HomingSensor should be linked only with 1 bit from the Digital Inputs Status 8bit. Select Offset=2. Bit 2 is the Home Switch (or IN0) in object 60FDh or 208Fh sub-index 1. Click OK.

11.4.9 Enabling and testing the NC-PTP interface in TwinCAT



To test the NC-PTP interface, click the Activate configuration button, and then click OK twice to the new questions that appear.



To enable PWM power to the motor, click Axis 1 under the NC Task, select the Online tab and click on the Set button. Click on the All button in the Set Enabling menu.

If everything is OK, the motor should apply torque and hold its position.

Press F4++ or F1- - to jog the motor back and forth. Press the F9 -->. button to start a homing procedure. Trigger the digital input on the drive IN0 to finish the homing.

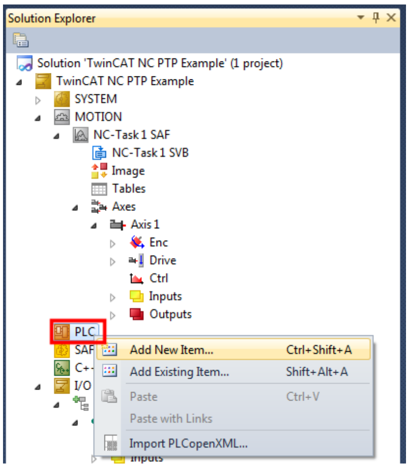
Remark: the homing procedure done with TwinCAT is more imprecise than executing a homing function in the drive. The higher the communication time, the higher the lag between the decisions that the home switch has been reached.

11.4.10 Setting Controlword bit 14 to 1 (Optional)

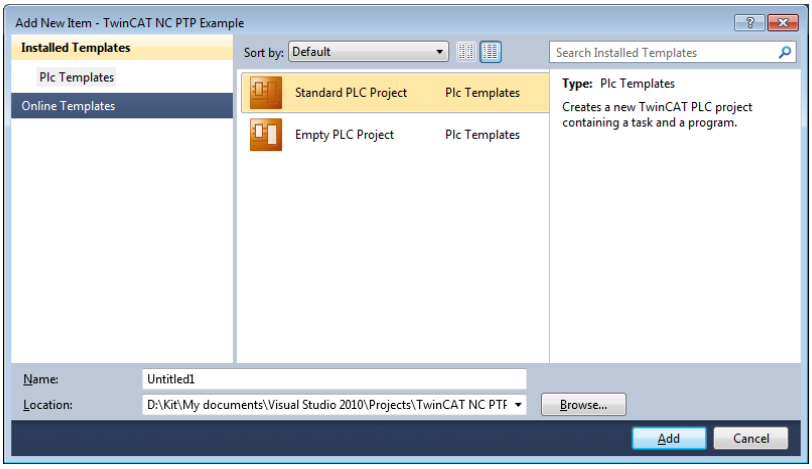
In some cases, in the NC-PTP interface, the motor control is stopped and the motor is moved by external forces. At motor control re-enable, the motor jumps towards the old position. This is because the new motion trajectory starts from the actual position reference (the theoretical position where the motor should be). The position reference is also the old position, when the motor was stopped, before it was moved by external forces.

If Controlword bit 14 is set to 1, then, when re-enabling motor control, the motion trajectory starts from the actual encoder value. The motor will not jump if re-enabled after it was moved by external forces.

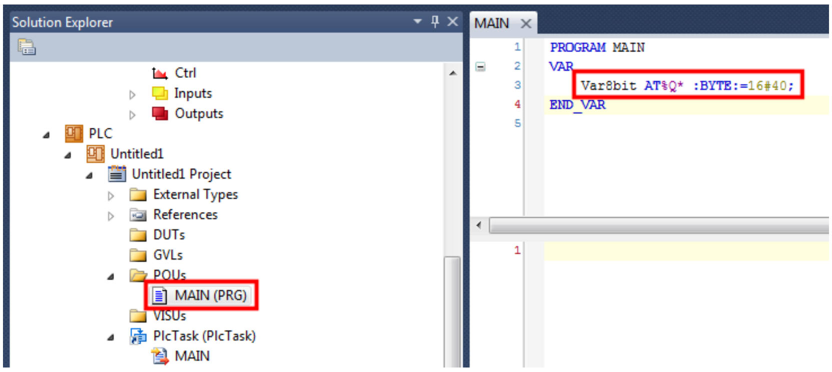
Set TwinCAT in Config Mode.



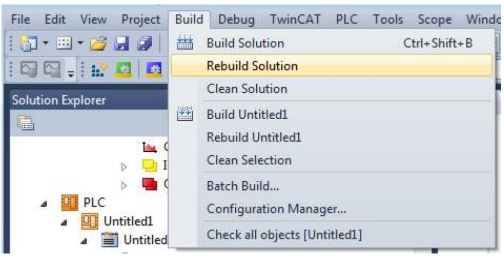
In the Solution Explorer, right click the PLC and choose Add New Item...



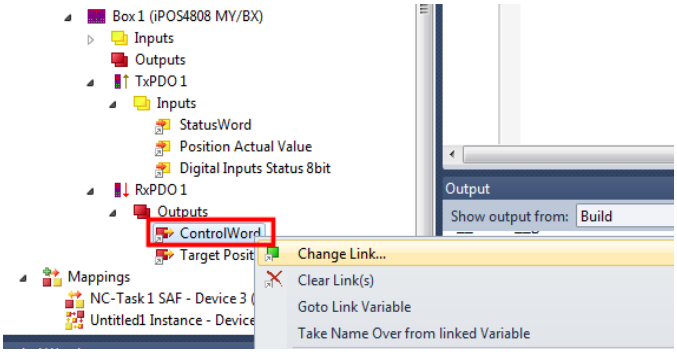
Name your new PLC project file and click the Add button.



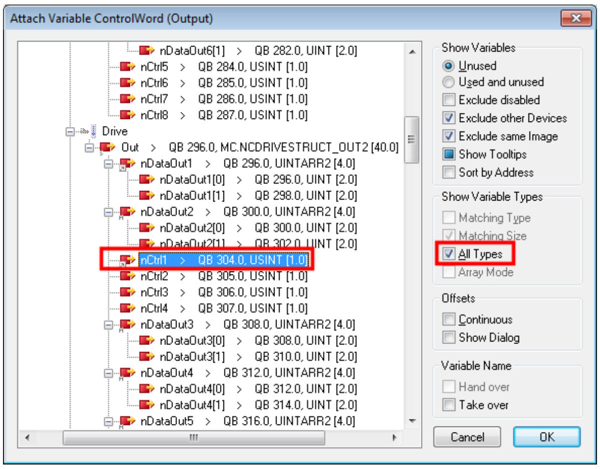
In the Solution Explorer, double click the MAIN(PRG) under the PLC project file. In the newly opened file, under the VAR section, write Var8bit AT%Q* :BYTE:=16#40; .



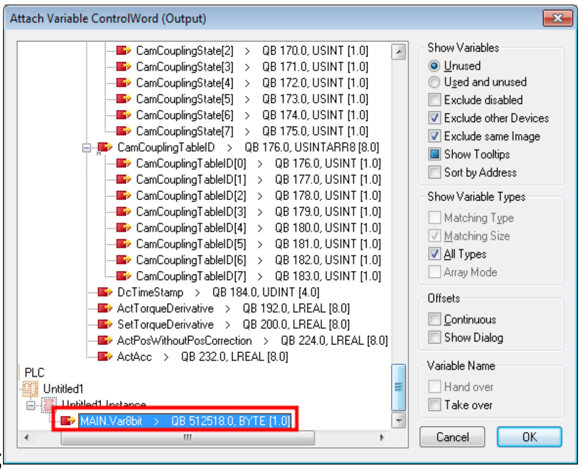
In the toolbar click Build/ Rebuild solution, for the Var8bit to be available for linking.



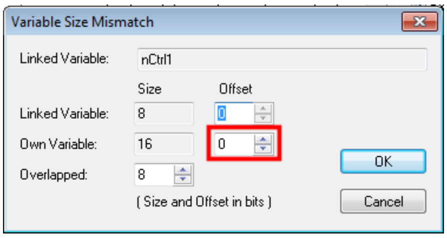
Right click the Controlword variable and choose change link.



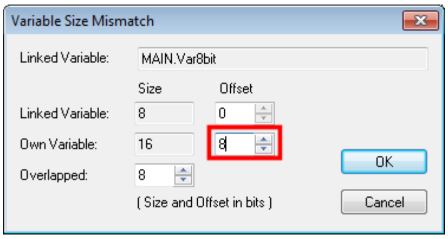
;



While having the All types (for Show variables) checked and nCtrl1 variable selected, hold Ctrl key and select from the end of the list MAIN.Var8bit and click OK. Both nCtrl1 and Var8bit should be selected before clicking OK.



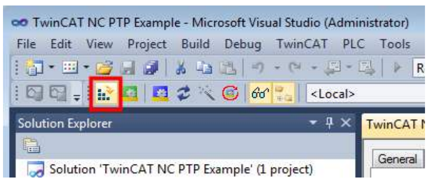
and



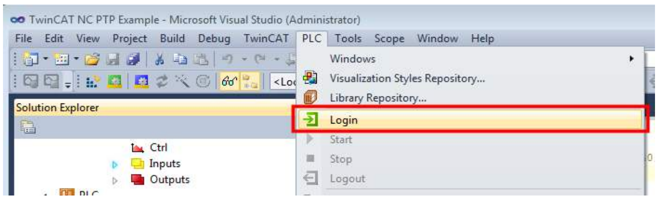
For the next two dialogues that come up next, do the following:

For the nCtrl1, just click OK. The Own Variable offset should be 0.

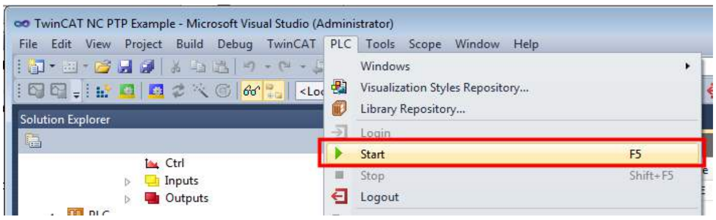
For the Var8bit, select the Own Variable offset = 8 and click OK.



Click the Activate configuration button, and then click OK twice to the new questions that appear.



Click PLC/ Login and click yes to the question that follows.



Click PLC/ Start, to initialize the Var8bit value.

Follow chapter 10.4.9 Enabling and testing the NC-PTP interface in TwinCAT again to test the interface.

12 Cyclic synchronous velocity mode (CSV)

12.1 Overview

The overall structure for this mode is shown in *Figure 12.1.1*. 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.

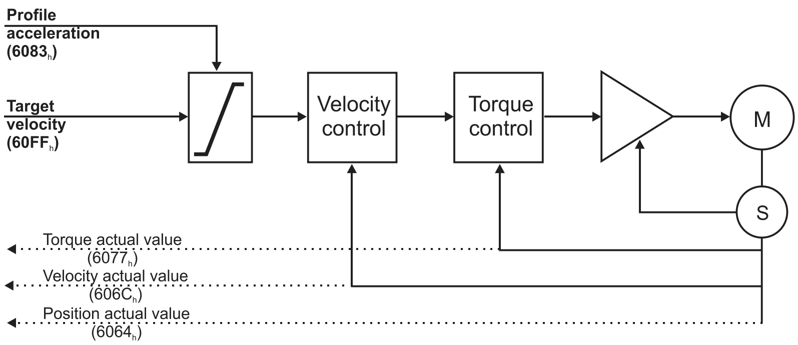


Figure 12.1.1. *Cyclic synchronous velocity mode overview*

Remark: The Cyclic synchronous velocity control mode is available only for firmware versions FA01x.

12.1.1 Controlword in cyclic synchronous velocity mode

The cyclic synchronous velocity mode uses no mode specific bits of the Controlword. See Object 6040_h: Controlword.

12.1.2 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 12.1.1 *– 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

12.2 Cyclic Synchronous Velocity Mode basic example

Short description of the example:

- Start the node
- Remap RPDO1 and set it as synchronous
- Remap TPDO1 and set it as synchronous

- Set CSV mode in Modes of Operation

- Set Operation Enable. The handshake between what is commanded into Controlword and what is read from Statusword will be described in detail

- Send a typical CSV motion command.

Step 1 starts the remote node 6, which means the PDOs will be enabled.

1. **Start remote node**. Send an NMT message to start the node id **06**.

Send the following message:

COB-ID	Data
0	01 06

Remark: if **00** is sent instead of 06, all nodes in the network will be enabled.

Steps 2 and 3 set the interpolation time to 2ms.

The interpolation time needs to be set in the object 60C2h. Sub-index 1 holds the interpolation time period value (i.e. 2 for 2ms) and sub-index 2 holds the interpolation time index (i.e. -3 for ms = 10^-3 s). The interpolation time has to be equal to the SYNC period and the period of the synchronous RPDO containing the position command.

2. **Interpolation time period value.** Set the interpolation time value to 2 (0x02).

Send the following message (SDO write access to object 60C2h sub-index 1 the 8-bit value 02h):

COB-ID	Data
606	2F C2 60 01 02

3. **Interpolation time index.** Set the interpolation time index value to -3 (0xFD).

Send the following message (SDO write access to object 60C2h sub-index 2 the 8-bit value FDh):

COB-ID	Data
606	2F C2 60 02 FD

Steps 4 to 7 remap RPDO1 to receive Controlword (6040h, 16bit) and Target Velocity (60FFh, 32bit).

4. **Disable RPDO1 mapping.** To reconfigure any RPDO mapping, sub-index 0 of the corresponding mapping parameter object must be set to 0 in order to disable the PDO mapping.

Send the following message (SDO write access to object 1600h sub-index 0 the 8-bit value 00h):

COB-ID	Data
606	2F 00 16 00 00

5. **Map Controlword 6040h to RPDO1 sub-index 1.**

Send the following message (SDO write access to object 1600h sub-index 1 the 32-bit value 60400010h):

COB-ID	Data
606	23 00 16 01 10 00 40 60

6. **Map Target Velocity 60FFh to RPDO1 sub-index 2.**

Send the following message (SDO write access to object 1600h sub-index 2 the 32-bit value 60FF0020h):

COB-ID	Data
606	23 00 16 02 20 00 FF 60

7. **Enable RPDO1 mapping.** To enable any RPDO mapping, sub-index 0 of the corresponding mapping parameter object must be set with the number of sub-indexes defined in it. In this case, there are 2.

Send the following message (SDO write access to object 1600h sub-index 0 the 8-bit value 02h):

COB-ID	Data
606	2F 00 16 00 02

Steps 8 to 10 set RPDO1 as synchronous.

8. **Disable RPDO1.** To change any RPDO Communication parameters, sub-index 1 bit 31 must be set. It is recommended that only bit 31 is set and the number already defined inside should be kept.

   Example: the sub-index 1 value is 0x206 which is the RPDO1 COB ID for axis 6 (0x200 + Axis ID). From this number, only bit 31 should be set. It means that instead of 0x206, 0x80000206 should be written.

Send the following message (SDO write access to object 1400h sub-index 1 the 32-bit value 80000206h):

COB-ID	Data
606	23 00 14 01 06 02 00 80

9. **Set RPDO1 as synchronous, with the period of 1 SYNC.** Write 1 into sub-index 2 Transmission type. RPDO1 data will be processed after the reception of each SYNC.

Send the following message (SDO write access to object 1400h sub-index 2 the 8-bit value 01h):

COB-ID	Data
606	2F 00 14 02 01

10. **Enable RPDO1.** To enable a RPDO, bit 31 of sub-index 1 must be reset without interfering with the other bits. For the RPDO1 of axis 6, the COB ID should be (0x200 + axis ID). It means 0x206 should be written.

Send the following message (SDO write access to object 1400h sub-index 1 the 32-bit value 00000206h):

COB-ID	Data
606	23 00 14 01 06 02 00 00

Steps 11 to 14 remap TPDO1 to send Statusword (6041h, 16bit) and Velocity actual value (606Ch, 32bit).

11. **Disable TPDO1 mapping.** To reconfigure any TPDO mapping, sub-index 0 of the corresponding mapping parameter object must be set to 0 in order to disable the PDO mapping.

Send the following message (SDO write access to object 1A00h sub-index 0 the 8-bit value 00h):

COB-ID	Data
606	2F 00 1A 00 00

12. **Map Statusword 6041h to TPDO1 sub-index 1.**

Send the following message (SDO write access to object 1A00h sub-index 1 the 32-bit value 60410010h):

COB-ID	Data
606	23 00 1A 01 10 00 41 60

13. **Map Velocity actual value 606Ch to TPDO1 sub-index 2.**

Send the following message (SDO write access to object 1A00h sub-index 2 the 32-bit value 60640020h):

COB-ID	Data
606	23 00 1A 02 20 00 6C 60

14. **Enable TPDO1 mapping.** To enable any TPDO mapping, sub-index 0 of the corresponding mapping parameter object must be set with the number of sub-indexes defined in it. In this case, there are 2.

Send the following message (SDO write access to object 1A00h sub-index 0 the 8-bit value 02h):

COB-ID	Data
606	2F 00 1A 00 02

Steps 15 to 17 set TPDO1 as synchronous.

15. **Disable TPDO1.** To change any TPDO Communication parameters, sub-index 1 bit 31 must be set. It is recommended that only bit 31 is set and the number already defined inside should be kept.

    Example: the sub-index 1 value is 0x186 which is the TPDO1 COB ID for axis 6 (0x180 + Axis ID). From this number, only bit 31 should be set. It means that instead of 0x186, 0x80000186 should be written.

Send the following message (SDO write access to object 1800h sub-index 1 the 32-bit value 80000186h):

COB-ID	Data
606	23 00 18 01 86 01 00 80

16. **Set TPDO1 as synchronous, with the period of 1 SYNC.** Write 1 into sub-index 2 Transmission type. TPDO1 data is updated when the SYNC is received, and then TPDO1 is sent as soon as possible.

Send the following message (SDO write access to object 1800h sub-index 2 the 8-bit value 01h):

COB-ID	Data
606	2F 00 18 02 01

17. **Enable TPDO1.** To enable a TPDO, bit 31 of sub-index 1 must be reset without interfering with the other bits. For the TPDO1 of axis 6, the COB ID should be (0x180 + axis ID). It means 0x186 should be written.

Send the following message (SDO write access to object 1800h sub-index 1 the 32-bit value 00000186h):

COB-ID	Data
606	23 00 18 01 86 01 00 00

Step 18 sets CSV mode into the Modes of operation object.

18. **Set modes of operation to CSV.** Write 0x09 into object 6060h to set the drive into CSV mode.

Remark: the drive will be in CSV mode only after in reaches the state Operation Enabled. This means that object 6061h (Modes of operation display) will show 9 (drive is in CSV mode), only after Operation Enabled has been reached.

Send the following message (SDO write access to object 6060h sub-index 0 the 8-bit value 09h):

COB-ID	Data
606	2F 60 60 00 09

Steps 19 to 21 bring the drive into *Operation enabled* state and also start the CSV mode motion.

Remark 1: from this point on, the master should send the SYNC messages at precisely 2ms (the same number defined in $60C2_h$). Transmission of RPDO1 should also be started by the master.

Remark 2: the SYNC message is usually configured at the CANopen master start-up and can be sent from the drive boot-up time. The configuration messages until this point can be sent in parallel with the SYNC messages. Only after all the PDOs are configured as synchronous, the drive will use the SYNC signal for the PDOs.

19. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the *shutdown* command via Controlword associated PDO.

Send the following message (SYNC)

COB-ID	Data
80	Null

This was the SYNC signal. It must be sent at precisely 2ms intervals. In this example it can also be sent manually, to understand each command and what it does.

Send the following message (RPDO1)

COB-ID	Data
206	06 00 00 00 00 00

The **0006** is the new value for Controlword, i.e. the command to enter *Ready to switch on* state.

The **00000000** is the velocity command.

Send the following message (SYNC)

COB-ID	Data
80	Null

This was the SYNC signal. It must be sent at precisely 2ms intervals. After each SYNC signal, the drive will send its TPDO1. To be able to change the next Controlword command in RPDO1, ensure that the drive reaches *Ready to switch on* state by waiting for the TPDO1 with the following content:

Wait to receive the following message (TPDO1)

COB-ID	Data
186	31 02 00 00 00 00

The **0231** is the Statusword value. The value $xx31_h$ shows that the drive reached *Ready to switch on* state. The master may have to wait a few SYNCs and read the TPDOs multiple times until this value is reached (there are also intermediary values)

The **00000000** is the Velocity actual value and can vary depending on the encoder reported position.

Warning: The master must always wait for the drive to reach the desired state programmed into Controlword by checking the Statusword. No other command must be sent during this time. In this case, because the RPDOs are synchronous, the RPDO1 must be sent continuously without changing the command in Controlword until the drive reaches the desired state as reported into the Statusword.

20. **Switch on.** Change the node state from *Ready to switch on* to *Switched on* by sending the *switch on* command via Controlword associated PDO.

Send the following message (SYNC)

COB-ID	Data
80	Null

This was the SYNC signal. It must be sent at precisely 2ms intervals.

Send the following message (RPDO1)

COB-ID	Data
206	07 00 00 00 00 00

The **0007** is the new value for Controlword, i.e. the command to enter *Switched on* state.

Send the following message (SYNC)

COB-ID	Data
80	Null

This was the SYNC signal. It must be sent at precisely 2ms intervals. After each SYNC signal, the drive will send its TPDO1. To be able to change the next Controlword command in RPDO1, ensure that the drive reaches *Switched on* state by waiting for the TPDO1 with the following content:

Wait to receive the following message (TPDO1)

COB-ID	Data
186	33 82 00 00 00 00

The 8233 is the Statusword value. The value xx33$_h$ shows that the drive reached *Switched on* state. The master may have to wait a few SYNCs and read the TPDOs multiple times until this value is reached (there are also intermediary values).

At this step, the drive starts applying power to the motor. The time to reach *Switched on* state depends on the motor initialization method and its parameters (the *Start method* as defined in the Drive Setup Dialogue in ESM). Initialization times of up to 2s are not uncommon.

Warning: The master must always wait for the drive to reach the desired state programmed into Controlword by checking the Statusword. No other command must be sent during this time. In this case, because the RPDOs are synchronous, the RPDO1 must be sent continuously without changing the command in Controlword until the drive reaches the desired state as reported into the Statusword.

After the drive reaches *Switched On* state, the master can continue to the next step.

21. Enable operation. Change the node state from *Switched on* to *Operation enabled* by sending the *Enable operation* command via Controlword associated PDO.

Send the following message (SYNC)

COB-ID	Data
80	null

This was the SYNC signal. It must be sent at precisely 2ms intervals.

Send the following message (RPDO1)

COB-ID	Data
206	0F 00 00 00 00 00

The **000F** is the command to enter *Operation enable* state in Controlword.

Send the following message (SYNC)

COB-ID	Data
80	null

This was the SYNC signal. It must be sent at precisely 2ms intervals. After each SYNC signal, the drive will send its TPDO1. Ensure that the drive reaches *Operation enabled* state by waiting for the TPDO1 with the following content:

Wait for the following message (TPDO1)

COB-ID	Data
186	37 96 00 00 00 00

The **9637** is the Statusword value. The value xx37$_h$ shows that the drive reached *Operation enable* state. The master may have to wait a few SYNCs and read the TPDOs multiple times until this value is reached (there are also intermediary values). From this step forward, the motor will execute a motion within 2ms to the absolute position given into RPDO1 as the Target velocity.

Step 22 describes a CSV motion command:

22. Move with 20 IU. Set the velocity command to 20 IU.

Send the following message (SYNC)

COB-ID	Data
80	null

This was the SYNC signal. It must be sent at precisely 2ms intervals. The drive will process the previously received RPDO immediately after the reception of the SYNC.

Send the following message (RPDO1)

COB-ID	Data
206	0F 00 00 00 14 00

The **000F** is the command to enter or remain in *Operation enabled* state in Controlword.

The **00140000** is the speed command (=20 in decimal).

Remark: By default, the speed value is given in IU and it is of a 16.16 bit structure. The integer part is in the MSB and the fractional part is in the LSB.

Send the following message (SYNC)

COB-ID	Data
80	null

After this SYNC, the motor will start to travel with the speed of 20 IU over the following 2ms.

The master then needs to cyclically send the SYNC and RPDO1 with updated position commands.

13 Cyclic synchronous torque mode (CST)

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 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.

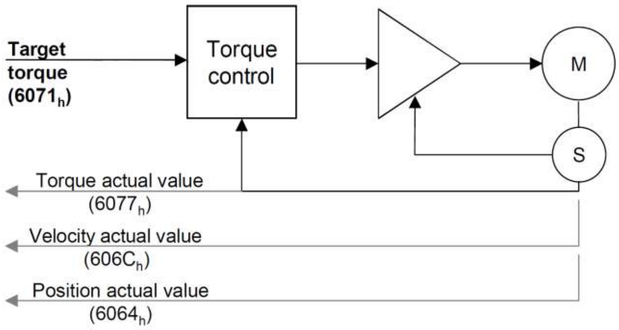


Figure 13.1.1. *Cyclic synchronous torque mode overview*

13.1.1 Controlword in cyclic synchronous torque mode

The cyclic synchronous torque mode uses no mode specific bits of the Controlword.

13.1.2 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 13.1.1 *– 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

13.2 Cyclic synchronous torque mode objects

13.2.1 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 FA01x firmware version, 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:

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

where I_{peak} is the peak current supported by the drive and *current[IU]* is the command value for object 6071_h.

13.2.2 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 FA01x firmware versions. In those versions, the unit is determined by the bit 0 of object 2115h: 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:

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

where I_{peak} is the peak current supported by the drive and *current[IU]* is the read value from object 6077_h.

13.2.3 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 FA01x.

Entry description:

Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0

13.2.4 Object 2115h: ASR4[1]

This object is responsible for configuring the ASR4 register, with Bit 0 of ASR4 specifying the formatting and representation of values in Object 6071h: Target torque and 6077h: Torque actual value. The other 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	1

13.3 Cyclic Synchronous Torque Mode basic example

Short description of the example:

- Start the node
- Remap RPDO1 and set it as synchronous
- Remap TPDO1 and set it as synchronous
- Set CST mode in Modes of Operation
- Set Operation Enable. The handshake between what is commanded into Controlword and what is read from Statusword will be described in detail
- Send a typical CST motion command.

Step 1 starts the remote node 6, which means the PDOs will be enabled.

1. **Start remote node**. Send an NMT message to start the node id **06**.

Send the following message:

COB-ID	Data
0	**01 06**

Remark: if **00** is sent instead of 06, all nodes in the network will be enabled.

Steps 2 and 3 set the interpolation time to 4ms.

The interpolation time needs to be set in the object 60C2h. Sub-index 1 holds the interpolation time period value (i.e. 4 for 4ms) and sub-index 2 holds the interpolation time index (i.e. -3 for ms = 10^-3 s).

The interpolation time has to be equal to the SYNC period and the period of the synchronous RPDO containing the position command.

2. **Interpolation time period value.** Set the interpolation time value to 4 (0x04).

Send the following message (SDO write access to object 60C2h sub-index 1 the 8-bit value 04h):

COB-ID	Data
606	**2F C2 60 01 04**

3. **Interpolation time index.** Set the interpolation time index value to -3 (0xFD).

Send the following message (SDO write access to object 60C2h sub-index 2 the 8-bit value FDh):

COB-ID	Data
606	**2F C2 60 02 FD**

Steps 4 to 7 remap RPDO1 to receive Controlword (6040h, 16bit) and Target Torque (6071h, 16bit).

[1] Available starting with firmware version FA01C or newer.

4. **Disable RPDO1 mapping.** To reconfigure any RPDO mapping, sub-index 0 of the corresponding mapping parameter object must be set to 0 in order to disable the PDO mapping.

Send the following message (SDO write access to object 1600_h sub-index 0 the 8-bit value 00_h):

COB-ID	Data
606	2F 00 16 00 00

5. **Map Controlword 6040_h to RPDO1 sub-index 1.**

Send the following message (SDO write access to object 1600_h sub-index 1 the 32-bit value 60400010_h):

COB-ID	Data
606	23 00 16 01 10 00 40 60

6. **Map Target Torque 6071_h to RPDO1 sub-index 2.**

Send the following message (SDO write access to object 1600_h sub-index 2 the 32-bit value 60710020_h):

COB-ID	Data
606	23 00 16 02 10 00 71 60

7. **Enable RPDO1 mapping.** To enable any RPDO mapping, sub-index 0 of the corresponding mapping parameter object must be set with the number of sub-indexes defined in it. In this case, there are 2.

Send the following message (SDO write access to object 1600_h sub-index 0 the 8-bit value 02_h):

COB-ID	Data
606	2F 00 16 00 02

Steps 8 to 10 set RPDO1 as synchronous.

8. **Disable RPDO1.** To change any RPDO Communication parameters, sub-index 1 bit 31 must be set. It is recommended that only bit 31 is set and the number already defined inside should be kept.

   Example: the sub-index 1 value is 0x206 which is the RPDO1 COB ID for axis 6 (0x200 + Axis ID). From this number, only bit 31 should be set. It means that instead of 0x206, 0x80000206 should be written.

Send the following message (SDO write access to object 1400_h sub-index 1 the 32-bit value 80000206_h):

COB-ID	Data
606	23 00 14 01 06 02 00 80

9. **Set RPDO1 as synchronous, with the period of 1 SYNC.** Write 1 into sub-index 2 Transmission type. RPDO1 data will be processed after the reception of each SYNC.

Send the following message (SDO write access to object 1400_h sub-index 2 the 8-bit value 01_h):

COB-ID	Data
606	2F 00 14 02 01

10. **Enable RPDO1.** To enable a RPDO, bit 31 of sub-index 1 must be reset without interfering with the other bits. For the RPDO1 of axis 6, the COB ID should be (0x200 + axis ID). It means 0x206 should be written.

Send the following message (SDO write access to object 1400_h sub-index 1 the 32-bit value 00000206_h):

COB-ID	Data
606	23 00 14 01 06 02 00 00

Steps 11 to 14 remap TPDO1 to send Statusword (6041_h, 16bit) and Torque actual value (6077_h, 16bit).

11. **Disable TPDO1 mapping.** To reconfigure any TPDO mapping, sub-index 0 of the corresponding mapping parameter object must be set to 0 in order to disable the PDO mapping.

Send the following message (SDO write access to object $1A00_h$ sub-index 0 the 8-bit value 00_h):

COB-ID	Data
606	2F 00 1A 00 00

12. **Map Statusword 6041_h to TPDO1 sub-index 1.**

Send the following message (SDO write access to object $1A00_h$ sub-index 1 the 32-bit value 60410010_h):

COB-ID	Data
606	23 00 1A 01 10 00 41 60

13. **Map Torque actual value 6077_h to TPDO1 sub-index 2.**

Send the following message (SDO write access to object $1A00_h$ sub-index 2 the 32-bit value 60770020_h):

COB-ID	Data
606	23 00 1A 02 10 00 77 60

14. **Enable TPDO1 mapping.** To enable any TPDO mapping, sub-index 0 of the corresponding mapping parameter object must be set with the number of sub-indexes defined in it. In this case, there are 2.

Send the following message (SDO write access to object $1A00_h$ sub-index 0 the 8-bit value 02_h):

COB-ID	Data
606	2F 00 1A 00 02

Steps 15 to 17 set TPDO1 as synchronous.

15. **Disable TPDO1.** To change any TPDO Communication parameters, sub-index 1 bit 31 must be set. It is recommended that only bit 31 is set and the number already defined inside should be kept.

    Example: the sub-index 1 value is 0x186 which is the TPDO1 COB ID for axis 6 (0x180 + Axis ID). From this number, only bit 31 should be set. It means that instead of 0x186, 0x80000186 should be written.

Send the following message (SDO write access to object 1800_h sub-index 1 the 32-bit value 80000186_h):

COB-ID	Data
606	23 00 18 01 86 01 00 80

16. **Set TPDO1 as synchronous, with the period of 1 SYNC.** Write 1 into sub-index 2 Transmission type. TPDO1 data is updated when the SYNC is received, and then TPDO1 is sent as soon as possible.

Send the following message (SDO write access to object 1800_h sub-index 2 the 8-bit value 01_h):

COB-ID	Data
606	2F 00 18 02 01

17. **Enable TPDO1.** To enable a TPDO, bit 31 of sub-index 1 must be reset without interfering with the other bits. For the TPDO1 of axis 6, the COB ID should be (0x180 + axis ID). It means 0x186 should be written.

Send the following message (SDO write access to object 1800_h sub-index 1 the 32-bit value 00000186_h):

COB-ID	Data
606	23 00 18 01 86 01 00 00

Step 18 sets CST mode into the Modes of operation object.

18. **Set modes of operation to CST.** Write 0x0A into object 6060_h to set the drive into CST mode.

Remark: the drive will be in CST mode only after in reaches the state Operation Enabled. This means that object 6061_h (Modes of operation display) will show 10 (drive is in CST mode), only after Operation Enabled has been reached.

Send the following message (SDO write access to object 6060_h sub-index 0 the 8-bit value $0A_h$):

COB-ID	Data
606	2F 60 60 00 0A

Steps 19 to 21 bring the drive into *Operation enabled* state and also start the CST control.

Remark 1: from this point on, the master should send the SYNC messages at precisely 4ms (the same number defined in $60C2_h$). Transmission of RPDO1 should also be started by the master.

Remark 2: the SYNC message is usually configured at the CANopen master start-up and can be sent from the drive boot-up time. The configuration messages until this point can be sent in parallel with the SYNC messages. Only after all the PDOs are configured as synchronous, the drive will use the SYNC signal for the PDOs.

19. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the *shutdown* command via Controlword associated PDO.

Send the following message (SYNC)

COB-ID	Data
80	Null

This was the SYNC signal. It must be sent at precisely 4ms intervals. In this example it can also be sent manually, to understand each command and what it does.

Send the following message (RPDO1)

COB-ID	Data
206	06 00 00 00

The **0006** is the new value for Controlword, i.e. the command to enter *Ready to switch on* state.

The **0000** is the torque command.

Send the following message (SYNC)

COB-ID	Data
80	Null

This was the SYNC signal. It must be sent at precisely 4ms intervals.

After each SYNC signal, the drive will send its TPDO1. To be able to change the next Controlword command in RPDO1, ensure that the drive reaches *Ready to switch on* state by waiting for the TPDO1 with the following content:

Wait to receive the following message (TPDO1)

COB-ID	Data
186	31 02 00 00

The **0231** is the Statusword value. The value xx31h shows that the drive reached *Ready to switch on* state. The master may have to wait a few SYNCs and read the TPDOs multiple times until this value is reached (there are also intermediary values)

The **0000** is the Torque actual value and can vary depending on the encoder reported position.

Warning: The master must always wait for the drive to reach the desired state programmed into Controlword by checking the Statusword. No other command must be sent during this time. In this case, because the RPDOs are synchronous, the RPDO1 must be sent continuously without changing the command in Controlword until the drive reaches the desired state as reported into the Statusword.

20. **Switch on.** Change the node state from *Ready to switch on* to *Switched on* by sending the *switch on* command via Controlword associated PDO.

Send the following message (SYNC)

COB-ID	Data
80	Null

This was the SYNC signal. It must be sent at precisely 4ms intervals.

Send the following message (RPDO1)

COB-ID	Data
206	07 00 00 00

The **0007** is the new value for Controlword, i.e. the command to enter *Switched on* state.

Send the following message (SYNC)

COB-ID	Data
80	Null

This was the SYNC signal. It must be sent at precisely 4ms intervals.

After each SYNC signal, the drive will send its TPDO1. To be able to change the next Controlword command in RPDO1, ensure that the drive reaches *Switched on* state by waiting for the TPDO1 with the following content:

Wait to receive the following message (TPDO1)

COB-ID	Data
186	33 02 00 00

The **0233** is the Statusword value. The value xx33h shows that the drive reached *Switched on* state. The master may have to wait a few SYNCs and read the TPDOs multiple times until this value is reached (there are also intermediary values).

At this step, the drive starts applying power to the motor. The time to reach *Switched on* state depends on the motor initialization method and its parameters (the *Start method* as defined in the Drive Setup Dialogue in ESM). Initialization times of up to 2s are not uncommon.

Warning: The master must always wait for the drive to reach the desired state programmed into Controlword by checking the Statusword. No other command must be sent during this time. In this case, because the RPDOs are synchronous, the RPDO1 must be sent continuously without changing the command in Controlword until the drive reaches the desired state as reported into the Statusword.

After the drive reaches *Switched On* state, the master can continue to the next step.

21. **Enable operation.** Change the node state from *Switched on* to *Operation enabled* by sending the *Enable operation* command via Controlword associated PDO.

Send the following message (SYNC)

COB-ID	Data
80	null

This was the SYNC signal. It must be sent at precisely 4ms intervals.

Send the following message (RPDO1)

COB-ID	Data
206	0F 00 00 00

The **000F** is the command to enter *Operation enable* state in Controlword.

Send the following message (SYNC)

COB-ID	Data
80	null

This was the SYNC signal. It must be sent at precisely 4ms intervals.

After each SYNC signal, the drive will send its TPDO1. Ensure that the drive reaches *Operation enabled* state by waiting for the TPDO1 with the following content:

Wait for the following message (TPDO1)

COB-ID	Data
186	**37 92 00 00**

The **9237** is the Statusword value. The value xx37h shows that the drive reached *Operation enable* state. The master may have to wait a few SYNCs and read the TPDOs multiple times until this value is reached (there are also intermediary values).

From this step forward, the motor will execute a motion within 4ms to the absolute position given into RPDO1 as the Target torque.

Step 22 describes a CST command:

22. Apply a torque of 2 IU. Set the torque command to 2 IU.

Send the following message (SYNC)

COB-ID	Data
80	**null**

This was the SYNC signal. It must be sent at precisely 4ms intervals. The drive will process the previously received RPDO immediately after the reception of the SYNC.

Send the following message (RPDO1)

COB-ID	Data
206	**0F 00 02 00**

The **000F** is the command to enter or remain in *Operation enabled* state in Controlword.

The **0002** is the torque command (=2 in decimal).

Send the following message (SYNC)

COB-ID	Data
80	**null**

After this SYNC, a torque reference of 2 IU over the following 4ms will be applied to the motor.

The master then needs to cyclically send the SYNC and RPDO1 with updated position commands.

14 Velocity Profile Mode

14.1 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 CANopen 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.

14.1.1 Controlword in Profile Velocity mode

MSB **LSB**

See 6040h	Halt	See 6040h	reserved	See 6040h
15 9	8	7	6 4	3 0

Table 14.1 *– Controlword bits for Profile Velocity mode*

Name	Value	Description
Halt	0	Execute the motion
	1	Stop drive with *profile acceleration*

14.1.2 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 14.2 *– 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.*

14.2 Velocity Mode Objects

14.2.1 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 65536 IU = 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	-

14.2.2 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 which can be modified by Factor group objects.

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	-

14.2.3 Object 606Ch: Velocity actual value

The *velocity actual value* is given in user-defined velocity units. User-defined means it can be modified by Factor group objects. It is read from the velocity sensor.

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	-

14.2.4 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	-

14.2.5 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 F514K / FA01x firmware version

[2] Available starting with F514K / FA01x 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	-

14.2.6 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. User-defined means it can be modified by Factor group objects.

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	-

14.2.7 Object 60FFh: Target velocity

The *target velocity* is the input for the trajectory generator and the value is given in user-defined velocity units. User-defined means it can be modified by Factor group objects. By default, the value is given in IU and it is of a 16.16 bit structure. The integer part is in the MSB and the fractional part is in the LSB. To elaborate, see Paragraph 8.2.2 example.

This object is used for the velocity command only when 6060h Modes of Operation is 3 (Speed Mode).

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	-

14.2.8 Object 60F8h: Max slippage

The *max slippage* monitors whether the maximum speed error. The value is given in user-defined velocity units. User-defined means it can be modified by Factor group objects. 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.

By default, the value is given in IU and it is of a 16.16 bit structure. The integer part is in the MSB and the fractional part is in the LSB. To elaborate, see Paragraph 8.2.2 example.

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

14.2.9 Object 2005h: 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 for brushless and brushed motors and 0.8ms for stepper motors. This object is coupled with *Object 60F8h: Max slippage*.

Object description:

Index	2005h
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.

14.2.10 Object 2087h[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.

The read value is of a 16.16 bit structure.

Object description:

Index	2087h
Name	Actual internal velocity sensor on motor
Object code	VAR
Data type	INTEGER32

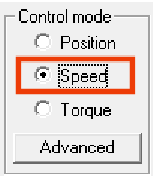
Entry description:

Access	RO
PDO mapping	Possible
Value range	INTEGER32
Default value	-

14.3 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).

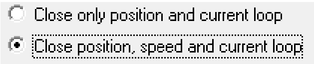
To enable the Current + Speed loop, in Drive setup, select under Control mode the speed radio button:



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.

[1] Object 2087h applies only to drives which have a secondary feedback



After all three loops are selected, the tuning for the speed and position must be done again.

Execute a speed control with 600 rpm target speed.

1. **Start remote node**. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	01 06

2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	06 00

3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	07 00

4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	0F 00

5. **Mode of operation.** Select speed mode.

Send the following message (SDO access to object 6060_h, 8-bit value 3_h):

COB-ID	Data
606	2F 60 60 00 03 00 00 00

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.

Send the following message (SDO access to object $60FF_h$ 32-bit value 00140000_h):

COB-ID	Data
606	23 FF 60 00 00 00 14 00

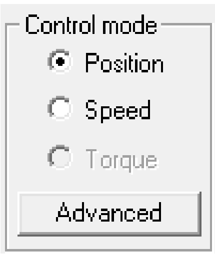
7. **Check the motor actual speed.** It should rotate with 600 rpm.

Send the following message (SDO access to read object $606C_h$ Velocity actual value):

COB-ID	Data
606	40 6C 60 00 00 00 00 00

14.4 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.



1. **Start remote node**. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	**01 06**

2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	**06 00**

3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	**07 00**

4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	**0F 00**

5. **Mode of operation.** Select speed mode.

Send the following message (SDO access to object 6060_h, 8-bit value 3_h):

COB-ID	Data
606	**2F 60 60 00 03 00 00 00**

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 02000000_h):

COB-ID	Data
606	**23 FF 60 00 00 00 00 02**

15 Electronic Gearing Position (EGEAR) Mode

15.1 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 gearing master:

1. Via CANopen master, which writes the master position in object Master position (index 201E$_h$).
2. From another Technosoft CANopen drive set as electronic gearing master with object **Master Settings** (index 2010$_h$). The position is sent using TechnoCAN, an extension of the CANopen protocol, developed by Technosoft.
3. 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.
4. 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 **Registration Mode** offers the possibility to super-impose another profile on top of the existing EGEAR one. By setting **Controlword.15 (Registration mode bit)**, the slave computes the master position increment and multiplies it with its programmed gear ratio, then adds another internally generated trajectory defined by the user (Trapezoidal Profile, S-Curve Profile, etc.). Using Registration mode the movement can be speed up, slow down or a position offset can be added to the EGAER command.
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 6083$_h$: Profile acceleration.

15.1.1 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 15.1 *– 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*

15.1.2 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 15.2 – 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

15.2 Gearing Position Mode Objects

15.2.1 Object 201E[1]h: 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	$201E_h$
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	-

15.2.2 Object 2010h: Master settings

This object contains key settings for the master of EGEAR / ECAM mode. A master in EGEAR / ECAM mode is a drive that controls a motor (irrespective of the control mode) and that will be designated to send the information about its position (demanded position or actual position) via communication to one or more slaves (programmed accordingly).

This object also allows setting the address of the slave to which the master will send its position, or, if there are more slaves to receive simultaneously the position from the master, the Group ID of these slaves.

Object description:

Index	2010_h
Name	Master settings
Object code	VAR
Data type	UNSIGNED16

Entry description:

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

Table 15.3 – Master Settings bits description

Bit	Value	Description
15	0	Master is not active – the master drive does not send any position values
	1	Master is active – the master drive starts sending its position to the slave axis
14 10	0	Reserved
9	0	The master sends its feedback (the position actual value)
	1	The master sends the demand position
8	0	Address is an axis ID
	1	Address is a group ID
7 … 0	x	Address of the slave drive(s)

[1] Available only with FA01x firmware versions.

15.2.3 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)

15.2.4 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

15.2.5 Object 2017h: 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 or from the digital inputs of the drive). The units are increments.

Object description:

Index	2017h
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

15.2.6 Object 2018h: 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	2018h
Name	Master actual speed
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	Possible
Value range	-32768 … 32767
Default value	0

15.2.7 Object 201Dh: 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	201Dh
Name	External Reference Type
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	-

***Table 15.4** – External Reference Type bit description*

Value	Description
0	Reserved
1	On-line. In case of External Reference Position / Speed / Torque Modes, select this option in order to read the reference from the object 201Ch, *External Online Reference* In case of Electronic Gearing and Camming Position Modes, select this option in order to read the master position received from a master drive via communication (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. 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

15.3 Electronic gearing through CAN example

This example is split in two parts:
Part1: Start an electronic gearing master profile on CAN.

1. **Start remote node**. Send a NMT message to start the node id 7.

Send the following message:

COB-ID	Data
0	**01 07**

2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
207	**06 00**

3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
207	07 00

4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
207	0F 00

5. **Modes of operation.** Select speed mode.

Send the following message (SDO access to object 6060h, 8-bit value 3h):

COB-ID	Data
607	2F 60 60 00 03 00 00 00

6. **Target Velocity.** Set speed to 15 IU.

Send the following message (SDO access to object 60FFh, 32-bit value Fh):

COB-ID	Data
607	23 FF 60 00 00 00 0F 00

The master motor should start rotating with 15IU speed.

7. **Master Settings.** Set the drive as master and program it to send its reference to axis 6.

Send the following message (SDO access to object 2010h 32-bit value 00008206h):

COB-ID	Data
607	2B 10 20 00 06 82 00 00

Part2: Start an Electronic Gearing Slave on CAN

1. **Start remote node**. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	01 06

2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	06 00

3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	07 00

4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	0F 00

5. **External reference type.** Slave receives reference through CAN.

Send the following message (SDO access to object 201Dh):

COB-ID	Data
606	2B 1D 20 00 01 00 00 00

6. **Modes of operation.** Select Electronic Gearing mode.

Send the following message (SDO access to object 6060h, 8-bit value -1):

COB-ID	Data
606	2F 60 60 00 FF 00 00 00

7. **Master resolution.** Set the master resolution.

Send the following message (SDO access to object 2012h, 32-bit value 2000):

COB-ID	Data
606	23 12 20 00 D0 07 00 00

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 1):

COB-ID	Data
606	2B 13 20 01 01 00 00 00

Set EG denominator to 1. Send the following message (SDO access to object 2013h,sub-index 2, 16-bit value 1):

COB-ID	Data
606	2B 13 20 02 01 00 00 00

9. **Enable EG slave** in Controlword associated PDO.

Send the following message:

COB-ID	Data
206	1F 00

The slave motor should start rotating with the same speed as the master motor.

16 Electronic Camming Position (ECAM) Mode

16.1 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 via CANbus communication channel, from another Technosoft drive set as electronic camming master with object **Master Settings** (index 2010_h). The position is sent using TechnoCAN, an extension of the CANopen protocol, developed by Technosoft.

The reference type is received online via communication channel and it is set with object **External Reference Type** (index $201D_h$). 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 6081_h).

Typically, the cam tables are first downloaded into the EEPROM memory of the drive by the CANopen master or with EasyMotion Studio. Then using the object **CAM table load address** (index 2019_h) they are copied in the RAM address set in object **CAM table run address** (index $201A_h$). 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.

16.1.1 Controlword in electronic camming position mode

MSB **LSB**

See 6040_h	Halt	See 6040_h	Abs / Rel	Activate Speed Limitation	Enable Electronic Camming Mode	See 6040_h
15 9	8	7	6	5	4	3 0

Table 16.1 *– 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*

16.1.2 Statusword in electronic camming position mode

MSB **LSB**

See 6041h		Following error	Reserved	See 6041h	Target reached	See 6041h	
15	14	13	12	11	10	9	0

***Table 16.2** – 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

16.2 Electronic Camming Position Mode Objects

16.2.1 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

16.2.2 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

16.2.3 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

16.2.4 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

16.2.5 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

16.2.6 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.

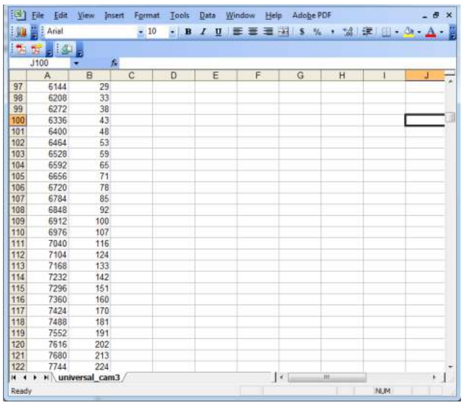


Figure 13.2.1. *MS Excel interface*

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).

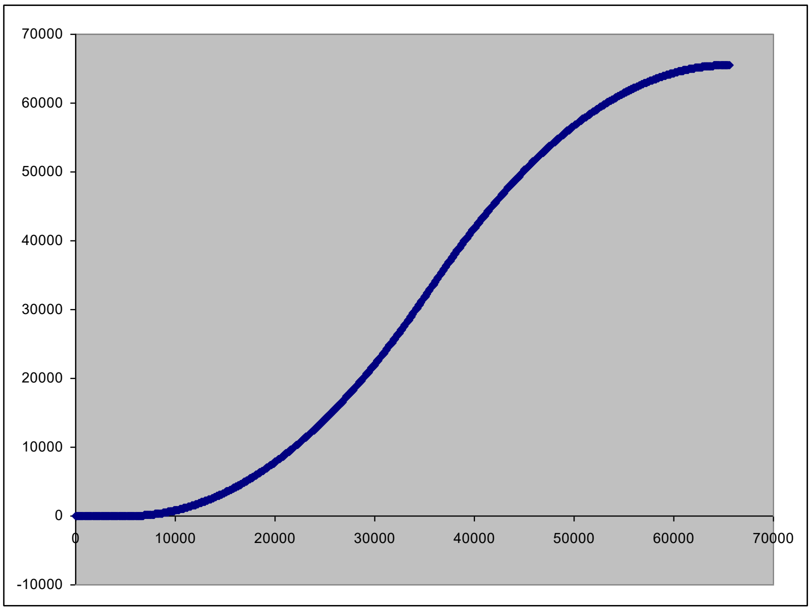


Figure 13.2.2. *Cam example*

After the cam is ready, save it as Text (Tab delimited) (*.txt) file.

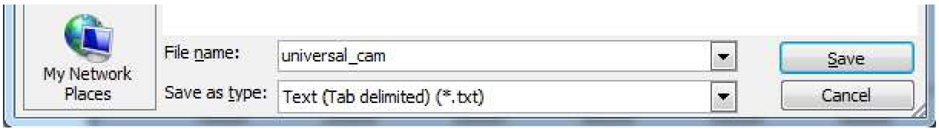


Figure 13.2.3. *Save As example.*

Once you have your cam file saved, start EasyMotion Studio, even the demo[1] version.

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

[1] ESM demo version available in download section here.

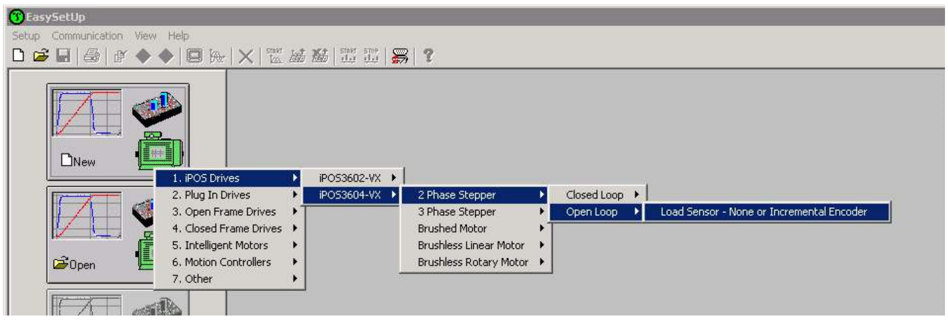


Figure 13.2.4. *Choose drive configuration.*

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 13.2.5**).

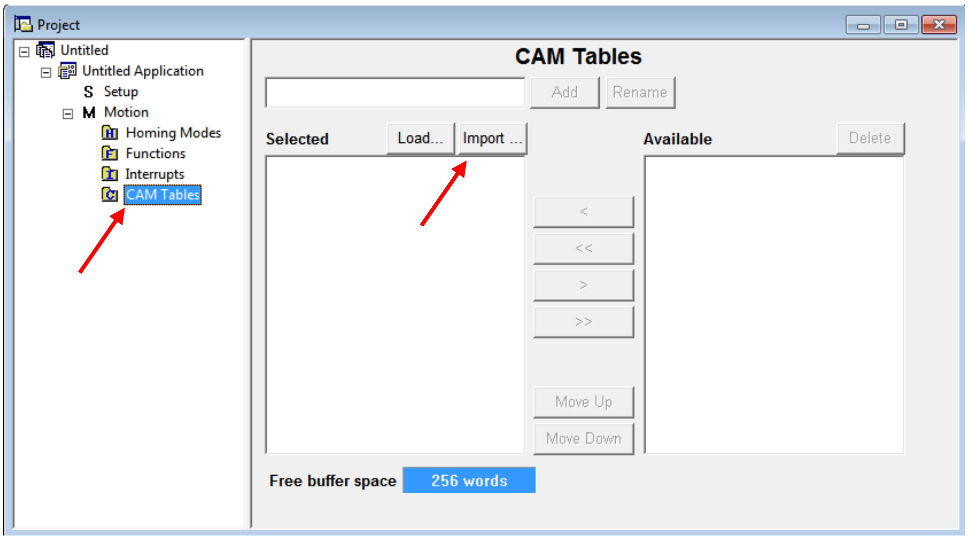


Figure 13.2.5. *CAM tab.*

If the CAM file loaded, it should look like this:

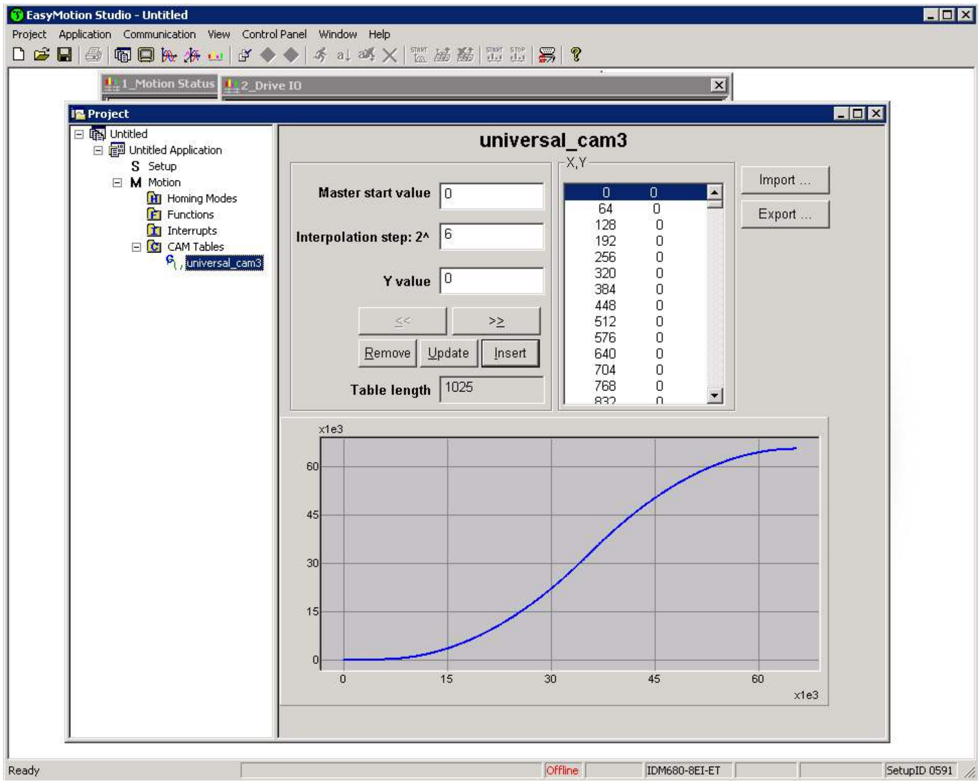
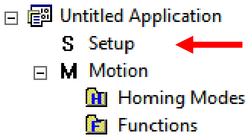
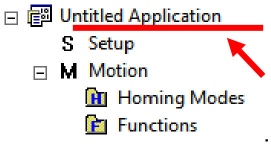


Figure 13.2.6. *CAM file loaded.*

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).

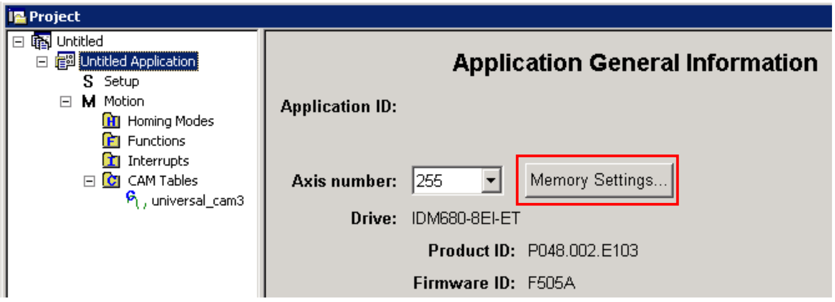


Figure 13.2.7. *Memory Settings location.*

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).

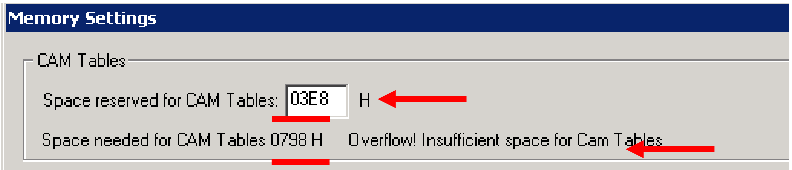


Figure 13.2.8. *Adjusting the necessary CAM space.*

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.

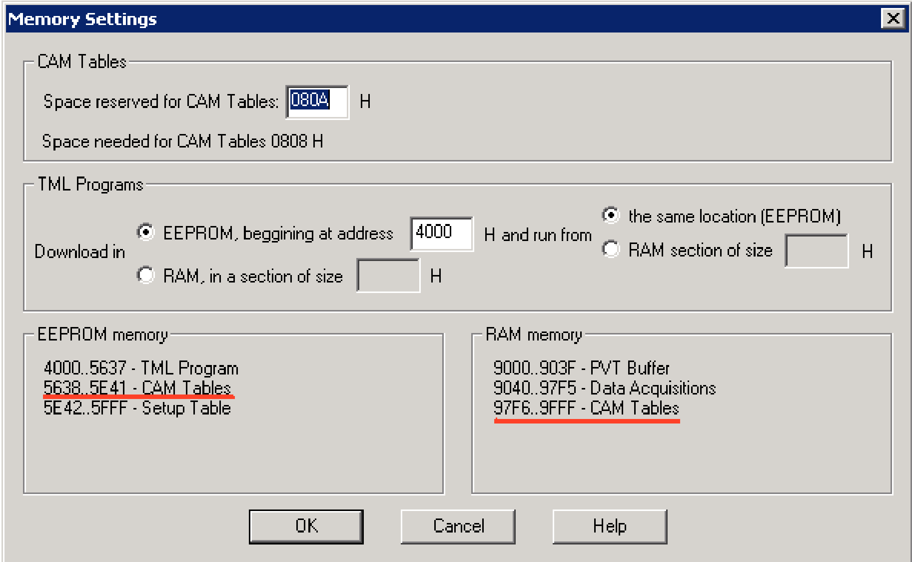


Figure 13.2.9. *Cam table load and run addresses.*

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.

[1] To create a setup file, please check your drive's user manual.

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.

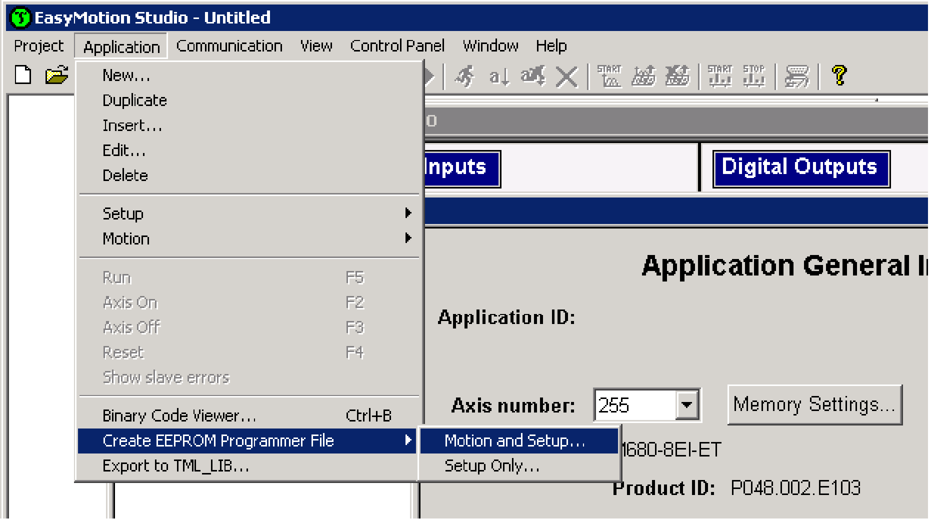


Figure 13.2.10. *Create .sw file.*

16.2.6.1 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 13.2.11**) (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 13.2.12**).

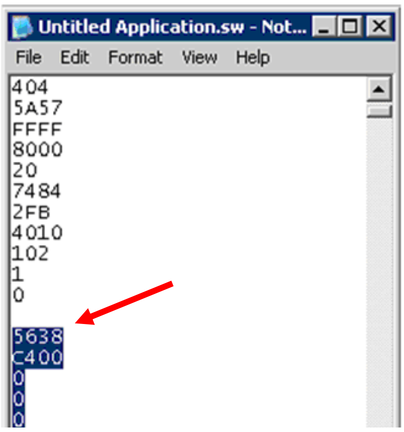


Figure 13.2.11. *.sw file structure example*

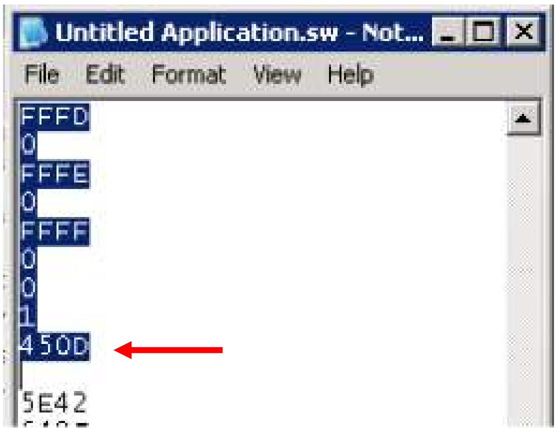


Figure 13.2.12. *.sw file empty line*

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.

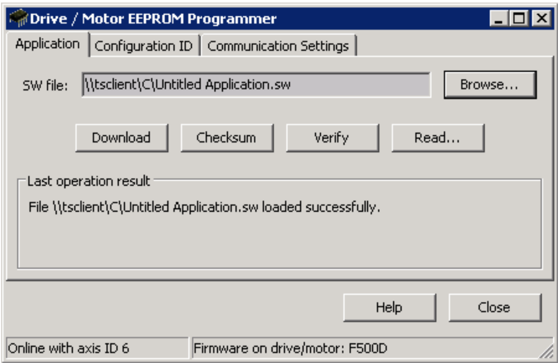


Figure 13.2.13. *THS EEPROM Programmer*

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.

16.2.6.2 Downloading a CAM .sw file with objects 2064h and 2065h example

In order to download the data block pointed by the red arrow found in "Figure 13.2.11. .sw file structure example", first the block start address i.e. **5638h** must be set using an SDO access to object **2064h**:

COB-ID	Data
606	23 64 20 00 08 00 38 56

The above configuration command also indicates that the 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 **5638h** until the following blank line represent data to write in the EEPROM memory at consecutive addresses starting with 5638h. The data writes are done using an SDO access to object **2065h**. First data word **C400h** is written using:

COB-ID	Data
606	23 65 20 00 00 C4 00 00

Next data word 0000h is written with:

COB-ID	Data
606	23 65 20 00 00 00 00 00

do this, until the end the CAM .sw file.

16.3 Electronic camming through CAN example

This example is split in two parts:
Part1: Start an Electronic Camming Slave on CAN
First load a cam table onto the drive as presented in chapter 0 .
Start remote node. Send a NMT message to start the node id 6.
Send the following message:

COB-ID	Data
0	01 06

Ready to switch on. Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.
Send the following message:

COB-ID	Data
206	06 00

Switch on. Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.
Send the following message:

COB-ID	Data
206	07 00

Enable operation. Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.
Send the following message:

COB-ID	Data
206	0F 00

External reference type. Slave receives reference through CAN.
Send the following message (SDO access to object 201Dh):

COB-ID	Data
606	2B 1D 20 00 01 00 00 00

Cam table load address. Set cam table load address as **5638h**.
The cam table load address can be discovered as explained in chapter**0** .
Send the following message (SDO access to object 2019h):

COB-ID	Data
606	2B 19 20 00 1E 5A 00 00

Cam table run address. Set cam table load address as **97F6h**.
The cam table run address can be discovered as explained in chapter "
Building a CAM profile and saving it as an .sw file example**"** .
Send the following message (SDO access to object 201Ah):

COB-ID	Data
606	2B 1A 20 00 F6 97 00 00

Modes of operation. Select Electronic Camming mode.
Send the following message (SDO access to object 6060h, 8-bit value -2):

COB-ID	Data
606	2F 60 60 00 FE 00 00 00

Master resolution. Set the master resolution.
Send the following message (SDO access to object 2012h, 32-bit value 2000):

COB-ID	Data
606	23 12 20 00 D0 07 00 00

Cam offset. Set cam offset to 6000 counts (1770h).
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 201Bh, 32-bit value 1770h):

COB-ID	Data
606	23 1B 20 00 70 17 00 00

Cam input scaling factor. Set it to 1 (10000h corresponds to a scaling factor of 1.0).
Send the following message (SDO access to object 206Bh, 32-bit value 10000h):

COB-ID	Data
606	23 6B 20 00 00 00 01 00

Cam output scaling factor. Set it to 1 (10000h corresponds to a scaling factor of 1.0).
Send the following message (SDO access to object 206Ch, 32-bit value 10000h):

COB-ID	Data
606	23 6C 20 00 00 00 01 00

Enable ECAM slave mode in Controlword associated PDO.
Send the following message:

COB-ID	Data
206	3F 00

The slave shall start moving and applying the cam after the master starts.

Part2: Start an electronic camming master on CAN.

1. **Start remote node**. Send a NMT message to start the node id 7.

Send the following message:

COB-ID	Data
0	01 07

2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
207	06 00

3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
207	07 00

4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
207	0F 00

5. **Modes of operation.** Select speed mode.

Send the following message (SDO access to object 6060h, 8-bit value 3h):

COB-ID	Data
607	2F 60 60 00 03 00 00 00

6. **Target Velocity.** Set speed to 15 IU.

Send the following message (SDO access to object 60FFh, 32-bit value Fh):

COB-ID	Data
607	23 FF 60 00 00 00 0F 00

The master motor should start rotating with 15IU speed.

7. **Master Settings.** Set the drive as master and program it to send it's reference to axis 6.

Send the following message (SDO access to object 2010h 16-bit value 8006h):

COB-ID	Data
607	2B 10 20 00 06 80 00 00

After the master is at position 6000 IU (cam offset), the slave (axis 06) shall rotate depending on the set cam values.

17 External Reference Position Mode

17.1 Overview

In this operating mode, the drive performs position control with the demand position read from the external reference provided by another device.

There are 2 types of external references:

Analogue – read by the drive via a dedicated analogue input (12-bit resolution)

Online – received online via the CAN bus communication channel from the CANopen master in object **External On-line Reference** (index $201C_h$)

The reference type is selected with object **External Reference Type** (index $201D_h$).

In external reference position mode with analogue or online reference, you can limit the maximum speed at sudden changes of the position reference and thus to reduce the mechanical shocks. This feature is activated by setting Controlword.6=1 and the maximum speed value in object **Profile Velocity** (index 6081_h).

Remark: The External Reference Position control mode is available with firmware version 508x/509x; F523x/524x and F514x.

17.1.1 Controlword in external reference position mode

MSB LSB

See 6040_h	Halt	See 6040_h	Reserved	Activate Speed Limitation	Enable External Position Mode	See 6040_h
15 9	8	7	6	5	4	3 0

Table 17.1 *– Controlword bit description for External Reference Position mode*

Name	Value	Description
Enable External Position Mode	0	No action
	0->1	External position mode active
Activate Speed Limitation	0	Do not limit speed on the inactive to active mode transition
	1	Limit speed when enabling the External Position mode with the value set in object 6081_h
Halt	0	Execute the instruction of bit 4
	1	Stop drive with *profile acceleration*

In order to correctly set an external reference position mode, you have to set the way the reference is received (either on-line or analogue), using the object $201D_h$, *External Reference Type*.

17.1.2 Statusword in external reference 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

Table 17.2 *– Statusword bit description for External Reference 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

17.2 External Reference Position Mode Objects

17.2.1 Object $201C_h$: External On-line Reference

This object is used to set the reference in case the *External Reference Type* (Object $201D_h$) is set for *online*. The unit for this object is the internal unit defined for each external reference mode (position / speed / torque).

For the external reference position mode, all 32bits are used.

Object description:

Index	201Ch
Name	External online reference
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	Possible
Units	Internal, operating mode dependent
Value range	INTEGER32
Default value	0

17.3 External reference position profile example

1. **Start remote node**. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	**01 06**

2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	**06 00**

3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	**07 00**

4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	**0F 00**

5. **External reference type.** Slave receives reference through CAN.

Send the following message (SDO access to object 201Dh):

COB-ID	Data
606	**2B 1D 20 00 01 00 00 00**

6. **Mode of operation.** Select External reference position mode.

Send the following message (SDO access to object 6060h, 8-bit value FDh):

COB-ID	Data
606	**2F 60 60 00 FD 00 00 00**

7. **Enable external position mode.** Set bit 4 from 0 to 1 in Controlword associated PDO.

Send the following message:

COB-ID	Data
206	**1F 00**

8. **Move to 500 IU .** Write 500 (0x01F4) into the external online reference object. The motor will jump in 1 control loop (1ms default) from its actual position to the absolute value 500 IU.

Send the following message (SDO access to object 201Ch 32-bit value 000001F4h):

COB-ID	Data
606	**23 1C 20 00 F4 01 00 00**

9. **Move to 1000 IU .** Write 1000 (0x03E8) into the external online reference object. The motor will jump in 1 control loop (1ms default) from its actual position to the absolute value 1000 IU.

Send the following message (SDO access to object 201Ch 32-bit value 000003E8h):

COB-ID	Data
606	**23 1C 20 00 E8 03 00 00**

Remark: if the drive is at position 0 and 500 is written in 201Ch while in external position mode, the motor will jump to position 500 in 1 control loop. This means that the velocity of the motor is 500 IU. To avoid moving with too high velocities, bit5 of Controlword can be set. With bit 5 set, the maximum velocity between external reference points received at 201Ch will be the speed value defined in object 6081h.

18 External Reference Speed Mode

18.1 Overview

In this mode, the drive performs speed control with demand velocity read from the external reference provided by other devices.
There are 2 types of external references:
Analogue – read by the drive via a dedicated analogue input (12-bit resolution)
Online – received online via the CAN bus communication channel from the CANopen master in object **External On-line Reference** (index $201C_h$)
The reference type is selected with object **External Reference Type** (index $201D_h$).
In external reference speed mode, you can limit the maximum acceleration at sudden changes of the speed reference and thus to get a smoother transition. This feature is activated by setting Controlword.5=1 and the maximum acceleration value in object Profile Acceleration (6083_h).
Remark: The External Reference Position control mode is available with firmware version 508x/509x; F523x/524x and F514x.

18.1.1 Controlword in external reference speed mode

MSB **LSB**

See 6040_h		Halt	See 6040_h	Reserved	Activate Acceleration Limitation	Enable External Speed Mode	See 6040_h	
15	9	8	7	6	5	4	3	0

***Table 18.1** – Controlword bit description for External Reference Speed Mode*

Name	Value	Description
Enable External Speed Mode	0	No action
	0->1	External speed mode active
Activate Speed Limitation	0	Do not limit acceleration on the inactive to active mode transition
	1	Limit acceleration when enabling the External Speed mode with the value defined in object 6083_h
Halt	0	Execute the instruction of bit 4
	1	Stop drive with *profile acceleration*

18.1.2 Statusword in external reference speed mode

MSB LSB

See 6041_h		Max slippage error	Speed	See 6041_h	Target reached	See 6041_h	
15	14	13	12	11	10	9	0

***Table 18.2** – Statusword bit description for External Reference Speed 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
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, the object from index $606F_h$, velocity threshold from profile velocity mode will be 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.*

18.2 External reference speed mode objects

18.2.1 Object 201Ch: External On-line Speed Reference

This object is used to set the reference in case the *External Reference Type* (Object $201D_h$) is set for *online*. The unit for this object is the internal unit defined for each external reference mode (position / speed / torque).
For the external reference speed mode, 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 fractional part.
Example: for a target speed of 50.00 IU, 0x00320000 must be set in $201C_h$.

Object description:

Index	$201C_h$
Name	External online reference
Object code	VAR
Data type	INTEGER32

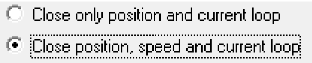
Entry description:

Access	RW
PDO mapping	Possible
Units	Internal, operating mode dependent
Value range	INTEGER32
Default value	0

18.3 External reference 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:

fter 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. **Start remote node**. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	01 06

2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	06 00

3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	07 00

4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	0F 00

5. **External reference type.** Slave receives reference through CAN.

Send the following message (SDO access to object $201D_h$):

COB-ID	Data
606	2B 1D 20 00 01 00 00 00

6. **Mode of operation.** Select External reference speed mode.

Send the following message (SDO access to object 6060_h, 8-bit value FC_h):

COB-ID	Data
606	2F 60 60 00 FC 00 00 00

7. **Enable external speed mode.** Set bit 4 from 0 to 1 in Controlword associated PDO.

Send the following message:

COB-ID	Data
206	1F 00

8. **Set velocity to 2 IU .** Write 2 * 65536 (0x20000) into the external online reference object. The motor will start rotating with the speed 2 IU in 1 control loop (1ms default) from its actual speed to the value 2 IU.

Send the following message (SDO access to object $201C_h$ 32-bit value 00020000_h):

COB-ID	Data
606	23 1C 20 00 00 00 02 00

9. **Set velocity to 8 IU .** Write 8 * 65536 (0x80000) into the external online reference object. The motor will start rotating with the speed 8 IU in 1 control loop (1ms default) from its actual speed to the value 8 IU.

Send the following message (SDO access to object $201C_h$ 32-bit value 00080000_h):

COB-ID	Data
606	23 1C 20 00 00 00 08 00

19 External Reference Torque Mode

19.1 Overview

In this mode, the drive is controlled in torque mode and the external reference is interpreted as torque/current reference.

There are 2 types of external references:

Analogue – read by the drive via a dedicated analogue input (12-bit resolution)

Online – received online via the CAN bus communication channel from the CANopen master in Object 201Ch: External On-line Reference.

The reference type is selected with Object 201Dh: External Reference Type.

Remark: The External Reference Position control mode is available with firmware version 508x/509x; F523x/524x and F514x.

19.1.1 Controlword in external reference torque mode

MSB **LSB**

See 6040h		Halt	See 6040h	Reserved	Reserved	Enable External Torque Mode	See 6040h	
15	9	8	7	6	5	4	3	0

***Table 19.1** – Controlword bit description for External Reference Torque Mode*

Name	Value	Description
Enable External Torque Mode	0	No action
	0->1	External torque mode active
Halt	0	Execute the instruction of bit 4
	1	Stop drive – set torque reference to 0

19.1.2 Statusword in external reference torque mode

MSB **LSB**

See 6041h		Reserved	Reserved	See 6041h	Target reached	See 6041h	
15	14	13	12	11	10	9	0

***Table 19.2** – Statusword bit description for External Reference Torque Mode*

Name	Value	Description
Target reached		Always 0

19.2 External reference torque mode objects

19.2.1 Object 201Ch: External On-line Torque Reference

This object is used to set the reference in case the *External Reference Type* (Object 201Dh) is set for *online*. The unit for this object is the internal unit defined for each external reference mode (position / speed / torque).

For the external reference torque mode, the torque (current) command is given in the MSB of the 32bits of data that must be written in 201Ch.

Object description:

Index	201Ch
Name	External online reference
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	Possible
Units	Internal, operating mode dependent
Value range	INTEGER32
Default value	0

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 $201C_h$ that must be set in the MSB of the 32bit value.

19.2.2 Object 6077_h: Torque actual value

This is used to provide the actual value of the current running through the motor. It corresponds to the instantaneous torque in the motor. The value is given in IU.

Object description:

Index	6077_h
Name	Torque actual value
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	Yes
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 I_{peak} is the peak current supported by the drive and *current[IU]* is the read value from object 6077_h.

19.2.3 Object $207E_h$: Current actual value[1]

The object displays the motor current actual value. This value is given in current internal units.

Object description:

Index	$207E_h$
Name	Current actual value
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	YES
Units	-
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 I_{peak} is the peak current supported by the drive and *current[IU]* is the read value from object $207E_h$.

19.3 External reference torque profile example

1. **Start remote node**. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	**01 06**

2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	**06 00**

[1] Available only with firmware version F508I / F509I and above

3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	07 00

4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	0F 00

5. **External reference type.** Slave receives reference through CAN.

Send the following message (SDO access to object $201D_h$):

COB-ID	Data
606	2B 1D 20 00 01 00 00 00

6. **Mode of operation.** Select External reference torque mode.

Send the following message (SDO access to object 6060_h, 8-bit value FB_h):

COB-ID	Data
606	2F 60 60 00 FB 00 00 00

7. **Enable external torque mode.** Set bit 4 from 0 to 1 in Controlword associated PDO.

Send the following message:

COB-ID	Data
206	1F 00

8. **Set torque to 0.2A for an iPOS4808 (any version) .** Write 328 * 65536 (0x1480000) into the external online reference object. The motor will start applying a positive 0.2A current on the motor in 1 control loop (1ms default) from its actual value.

Send the following message (SDO access to object $201C_h$ 32-bit value 01480000_h):

COB-ID	Data
606	23 1C 20 00 00 00 48 01

9. **Set torque to 0.4A for an iPOS4808 (any version) .** Write 655 * 65536 (0x28F0000) into the external online reference object. The motor will start applying a positive 0.4A current on the motor in 1 control loop (1ms default) from its actual value.

Send the following message (SDO access to object $201C_h$ 32-bit value $028F0000_h$):

COB-ID	Data
606	23 1C 20 00 00 00 8F 02

10. **Read the value of torque (current) actual value .** Read object 6077_h.

Send the following message (SDO access to read object 6077_h):

COB-ID	Data
606	40 77 60 00 00 00 00 00

The read value should be the near the one commanded previously with object $201C_h$.

20 Touch probe functionality

20.1 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.

20.2 Touch probe objects

20.2.1 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 20.1 *– 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.

20.2.2 Object 60B9h: Touch probe status

This object provides the status of the touch probe.

Object description:

Index	60B9h
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 20.2** – 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 60BDh
9	0	Touch probe 2 no positive edge value stored
	1	Touch probe 2 positive edge position stored in object 60BCh
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 60BBh
1	0	Touch probe 1 no positive edge value stored
	1	Touch probe 1 positive edge position stored in object 60BAh
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 60B8h bit 0 is 0). Bit 9 and 10 are set to 0 when touch probe 2 is switched off (object 60B8h bit 8 is 0). Bits 1,2,9 and 10 are set to 0 when object 60B8h bits 4,5,12 and 13 are set to 0.

20.2.3 Object 60BAh: Touch probe 1 positive edge

This object provides the position value of the touch probe 1 at positive edge.

Object description:

Index	60BAh
Name	Touch probe 1 positive edge
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	YES
Value range	$-2^{31}…2^{31}-1$
Default value	-

20.2.4 Object 60BBh: 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	-

20.2.5 Object 60BC$_h$: Touch probe 2 positive edge

This object provides the position value of the touch probe 2 at positive edge.

Object description:

Index	60BC$_h$
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	-

20.2.6 Object 60BD$_h$: Touch probe 2 negative edge

This object provides the position value of the touch probe 2 at negative edge.

Object description:

Index	60BD$_h$
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	-

20.2.7 Object 2104$_h$[1]: Auxiliary encoder function

This object configures the auxiliary feedback position capture on the zero impulse signal.

Object description:

Index	2104$_h$
Name	Auxiliary encoder function
Object code	VAR
Data type	UNSIGNED8

Entry description:

Access	RW
PDO mapping	Yes
Value range	0 … 65535
Default value	0

***Table 20.3** – Bit Assignment of the Auxiliary encoder function*

Bit	Value	Description
7..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.

[1] Object 2104$_h$ applies only to drives which have a secondary feedback input with an index signal

20.2.8 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 … 65535
Default value	0

***Table 20.4** – Bit Assignment of the Auxiliary encoder status*

Bit	Value	Description
7 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.

20.2.9 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:

Access	RO
PDO mapping	YES
Value range	$-2^{31} \ldots 2^{31}-1$
Default value	-

20.2.10 Object 2107h[3]: 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	$-2^{31} \ldots 2^{31}-1$
Default value	-

[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

[3] Object 2107h applies only to drives which have a secondary feedback input with an index signal

20.3 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**. Send a NMT message to start the node id 6.

Send the following message:

COB-ID	Data
0	01 06

2. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	06 00

3. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	07 00

4. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command via Controlword associated PDO.

Send the following message:

COB-ID	Data
206	0F 00

5. **Modes of operation.** Select position mode.

Send the following message (SDO access to object 6060_h, 8-bit value 1_h):

COB-ID	Data
606	2F 60 60 00 01 00 00 00

6. **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.

Send the following message (SDO access to object $607A_h$ 32-bit value 00002710_h):

COB-ID	Data
606	23 7A 60 00 10 27 00 00

7. **Target speed.** Set the target speed normally attained at the end of acceleration ramp to 2 IU (low speed).

Send the following message (SDO access to object 6081_h, 32-bit value 00020000h):

COB-ID	Data
606	23 81 60 00 00 00 02 00

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 $60B8_h$, 16-bit value 0011h):

COB-ID	Data
606	2B B8 60 00 11 00 00 00

9. **Read touch probe status.** Read touch probe status.

Send the following message (SDO read access to object $60B9_h$):

COB-ID	Data
606	40 B9 60 00 00 00 00 00

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. **Start the profile.**

Send the following message

COB-ID	Data
206	1F 00

11. **While the motor is moving, trigger the LSP input. The motor should stop.**
12. **Read touch probe status.** Read touch probe status.

Send the following message (SDO read access to object $60B9_h$):

COB-ID	Data
606	40 B9 60 00 00 00 00 00

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).

13. **Read the touch probe 1 positive edge captured value.**.

Send the following message (SDO read access to object $60BA_h$):

COB-ID	Data
606	40 BA 60 00 00 00 00 00

If the read value should be close to the value of motor actual position (6064_h). When the capture was detected, the motor was moving. The limit switch caused the motor to decelerate and stop after the even occurred.

21 Data Exchange between CANopen master and drives

21.1 Checking Setup Data Consistency

During the configuration phase, a CANopen master can quickly verify using the checksum objects and a reference **.sw** file whether the non-volatile EEPROM memory of the iPOS drive contains the right information. If the checksum reported by the drive does not match the one computed from the **.sw** file, the CANopen 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.

21.2 Image Files Format and Creation

An 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, 42 represents 0x0042.

A software file can contain up to 4 sections:

1. TML program
2. setup table
3. product and application ID
4. setup table start address

The **.sw** software files can be generated either from EasySetup or from EasyMotion Studio.

In EasySetup, you create a **.sw** file with the command **Setup | 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, you create a **.sw** file 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 CANopen 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.

21.3 Data Exchange Objects

21.3.1 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 21.1** – 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

21.3.2 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	**2065_h**
Name	Write data at address set in 2064_h (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 2064_h is 0 (operation on 16 bit variables)
	X	16-bit MSB of data if bit 0 of object 2064_h is 1 (operation on 32 bit variables)
15…0	X	16 bit LSB of data

21.3.3 Object 2066_h: 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 2064_h – Read/Write Configuration Register. After the successful read operation, the memory address in object 2064_h, bits 31…16, will be auto-incremented or not, as defined in the same register.

Object description:

Index	2066_h
Name	Read data from address set in 2064_h (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 2064_h is 0 (operation on 16 bit variables)
	X	16-bit MSB of data if bit 0 of object 2064_h is 1 (operation on 32 bit variables)
15…0	X	16 bit LSB of data

21.3.4 Object 2067h: Write 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

21.3.4.1 Writing 16 bit data to a specific address using object 2067h example

Considering the following variable found in variables.cfg in the /Firmwares/F514I 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:

COB-ID	Data
606	**23 67 20 00 34 12 6A 03**

21.3.5 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

21.3.6 Object $206A_h$: Checksum read register

This object stores the latest computed checksum.

Object description:

Index	$206A_h$
Name	Checksum read register
Object code	VAR
Data type	UNSIGNED16

Entry description:

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

21.4 Downloading an image file (.sw) to the drive using CANopen objects example

The structure of an image file (.sw) is described in paragraph **18.2** and shown in **Figure 18.4.1**.

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 56380008_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 $0000C400_h$.

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**.

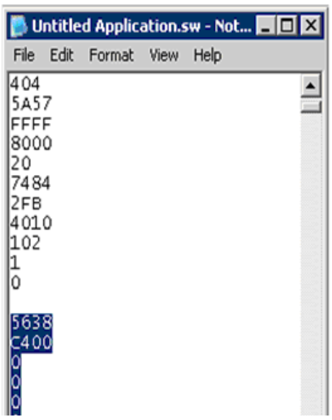


Figure 18.4.1. *.sw file structure example*

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.

21.5 Downloading an image file (.sw) to the drive using CANopen objects 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 on an IXXAT USB to CAN compact interface, together with their software canAnalyzer, which provides a tool for running C# scripts.

The code uses two functions that are not detailed in this example:
SendCANmessage(**MessageId, MessageData**);
The function sends a message over CAN with a configurable COB ID and Data field. The COB ID is declared as a 16bit integer and the MessageData is declared as a Byte array that can have up to 8 bytes.
Most CAN interfaces offer programming examples that contain such a function.
Wait_for_ID_and_Data(Expected COB ID, Expected DATA);
The function reads the received CAN messages and decides if the Expected COB ID and Expected DATA are the same as the ones received. It is a BOOL type function, so it returns TRUE if the expected data matched the one received.
Most CAN interfaces offer programming examples that contain a function that reads a CAN message.
If the implementation of such a function is difficult to implement, just replace it with a 4 or 5ms wait time before sending the next message. This is to make sure that the last SDO write command was read and processed by the drive.
Waiting for an SDO successful write reply from the drive reduces the write time and it is the safest way.

21.5.1 The main script code

It should look like this:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
using System.Threading;
using System.Collections;
using System.Runtime;
using System.Diagnostics;
using CAN.Interface.Services; //this is just an example; replace it with your interface


namespace THS_checksum_calculator
{
    static class Program
    {
        static void Main(string[] args)
        {
            int AxisID = 6;
            String PathToFile = "c:\\setup1.sw";
            Write_SWfile(AxisID, PathToFile);
        }
        private static void Write_SWfile(int AxisId, String Path)
        {//code
        }
        private static void SendCANmessage(int MessageID, Byte[] messageData)
        {//code
        }
        private static bool Wait_for_ID_and_Data(int MsgId, Byte[] ExpectedData)
        {//code
        }
    }
}
```

21.5.2 The function Write_SWfile code

```
private static void Write_SWfile(int AxisId, String Path)
        {
        int MessageId = 0x600 + AxisId;
            console.WriteLine("Writing SW file from path : " + Path);
            console.WriteLine("");
            try
            {
                StreamReader sr = File.OpenText(Path);
                String strLine;
                Byte[] LineData;
                Byte[] MessageData;
                bool setAddress = true; //because the first line in the .sw is an 
address, start with setAddress TRUE.
                while (null != (strLine = sr.ReadLine()))
                {

                    if (strLine == "") //checks for blank spaces with no data
                    {
                        setAddress = true;
                        continue;
                    }
                    if (setAddress) //if setAddress TRUE, set the current file data 
stream as address in 2064h.
                    {
                        LineData = BitConverter.GetBytes(Int16.Parse(strLine, 
System.Globalization.NumberStyles.HexNumber, null));
                        MessageData = new Byte[8] { 0x23, 0x64, 0x20, 0x00, 0x08, 0x00, 
0x00, 0x00 };
                        MessageData[6] = LineData[0];
                        MessageData[7] = LineData[1];   //Bytes 6 & 7 contain the 
section start address

                        SendCANmessage(MessageId, MessageData); //Send the previously 
defined CAN message
                        while (!Wait_for_ID_and_Data((0x580 + AxisId), new byte[3] { 
0x60, 0x64, 0x20 })) { } //wait for SDO confirmation
                        //The function Wait_for_ID_and_Data returns TRUE when it 
receives a successful SDO reply from the drive:
                        //Id 0x580 + AxisNr. and Data 0x60 0xYY 0xXX; where 0xXXYY is 
the object that was written.
                        console.WriteLine("Writing data section starting from address 
0x" + Convert.ToString(BitConverter.ToUInt16(LineData, 0), 16)); //Displays the start 
address of each .sw data segment
                        console.WriteLine("");
                        setAddress = false;
                        continue;
                    }
                    LineData = BitConverter.GetBytes(Int16.Parse(strLine, 
System.Globalization.NumberStyles.HexNumber, null));
                    MessageData = new Byte[8] { 0x23, 0x65, 0x20, 0x00, 0x00, 0x00, 
0x00, 0x00 };
                    MessageData[4] = LineData[0];
                    MessageData[5] = LineData[1];   //Bytes 4 & 5 contain the data from 
the .sw file (to be written in the EEPROM of the drive)
                    SendCANmessage(MessageId, MessageData);
                    while (!Wait_for_ID_and_Data((0x580 + AxisId), new byte[3] { 0x60, 
0x65, 0x20 })) {} //wait SDO confirmation
                }
                console.WriteLine("Writing file " + Path+  " ended");
                sr.Close();
            }
            catch (FileNotFoundException e)
            {
                console.WriteLine(e.Message);
            }
        }
```

21.6 Checking and loading the drive setup via SW file using CANopen commands example.

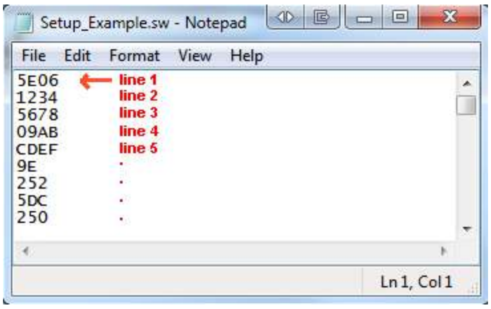
Check the integrity of the setup data on a drive and update it if needed.

Before reading this example, please read **paragraph 18.4.**

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 2069h 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 2069h sub-index 0, 32-bit value 5EFF5E06h.

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 206Ah.

3. **Read the computed checksum from object 206Ah**.

   Read by SDO protocol the value of object 206Ah.

   Let us assume the drive returns the following message (Object 206Ah = 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 2064h (write 32-bit data at EEPROM address 0x5E06 and auto-increment the address after the write operation).

   Send the following message: SDO write to object 2064h sub-index 0, 32-bit value 5E060009h.

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 2065h.

   Send the following message (SDO write to object 2065h sub-index 0, 32-bit value 56781234h):

The number 0x**1234** will be written at address 0x**5E06** and 0x**5678** will be at 0x**5E07**.

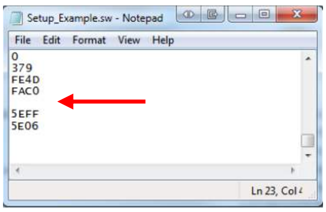
6. Assuming the next data after 0x**5678** will be 0x**09AB** and 0x**CDEF**, write the value 0x**CDEF09AB** into object 2065h.

   Send the following message (SDO write to object 2065h sub-index 0, 32-bit value CDEF09ABh):

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.

21.7 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.
This example is made in the same way as the example from ***18.5 Downloading an image file (.sw) to the drive using CANopen objects C# example code*** and can be easily merged. In this way, a script will download a .sw file and at the same time calculate the checksum for each section in order to verify it later with object 2069_h and $206A_h$.
As described in chapter **18.2**, the 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 2069_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.

21.7.1 The checksum calculation 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 2069h"); //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 2069h"); //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
```

22 Advanced features

Due to its embedded motion controller, a Technosoft intelligent drive offers many programming solutions that may simplify a lot the task of a CANopen master. This paragraph overviews a set of advanced programming features that can be used when combining TML programming at drive level with CANopen master control. All features presented below require usage of EasyMotion Studio as TML programming tool.

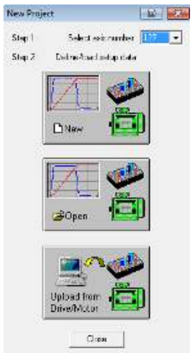
Remark: *If you do not use the advanced features presented below you do not need EasyMotion Studio.*

22.1 Using EasyMotion Studio

22.1.1 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.

22.1.2 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).

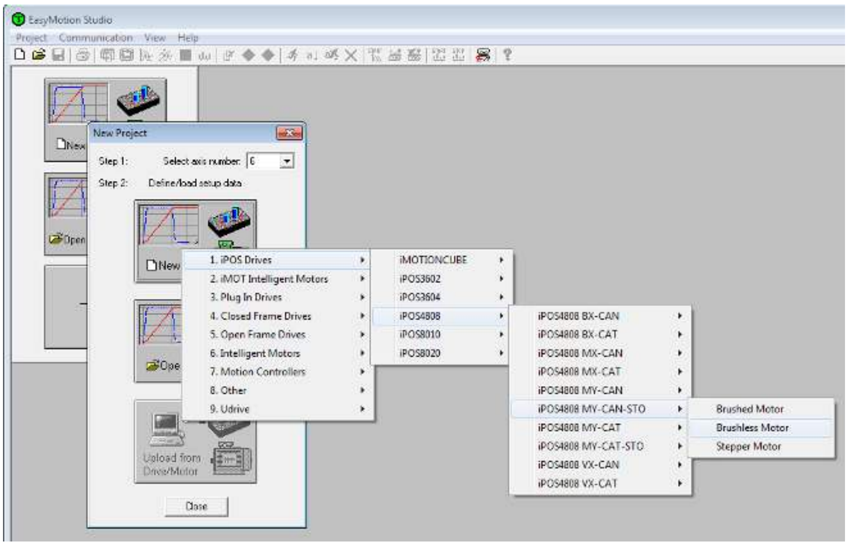


Figure 19.1.1. *EasyMotion Studio – Selecting the drive, motor and feedback*

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.

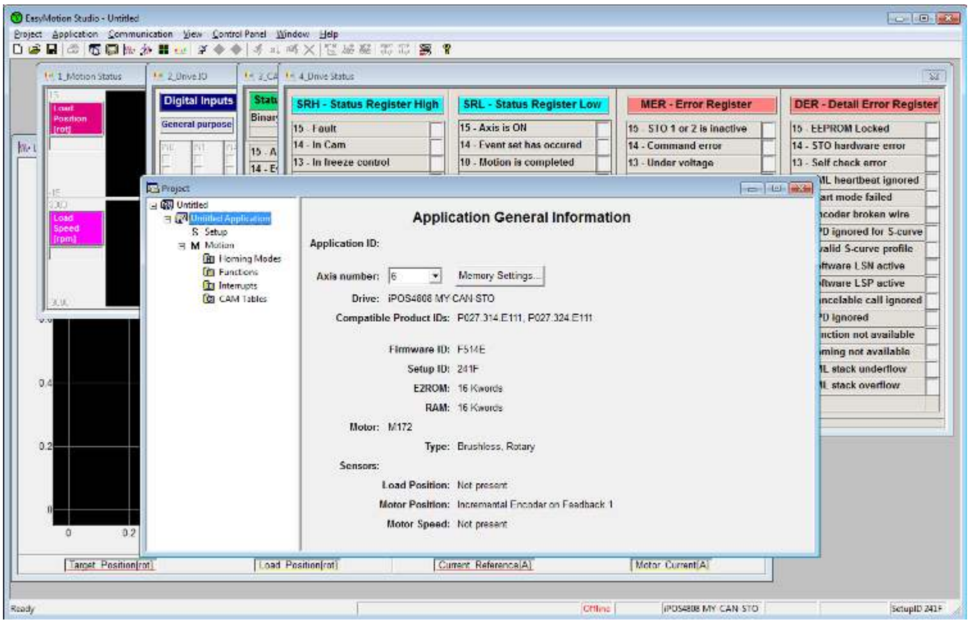


Figure 19.1.2. *EasyMotion Studio – Project information*

To edit the setup, click *View / Modify* button.

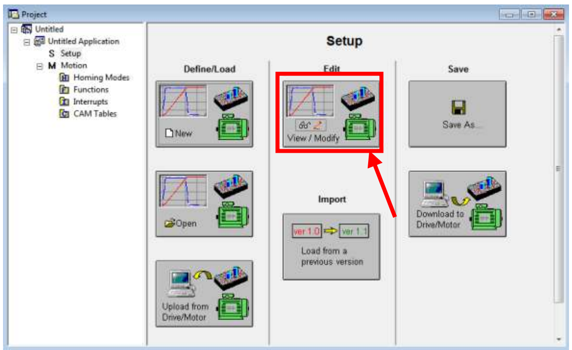


Figure 19.1.3. *EasyMotion Studio – Editing drive setup*

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.**

22.1.3 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 

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.

22.2 Using TML Functions to Split Motion between Master and Drives

With Technosoft intelligent drives you can really distribute the intelligence between a CANopen 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 functions (with possibility to abort their execution) stored in the drives EEPROM and waiting for a message, which confirms the finalization of the TML functions execution.

22.2.1 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.

***Remark:** You can call up to 10 TML functions using the CANopen objects.*

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.
3. **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.

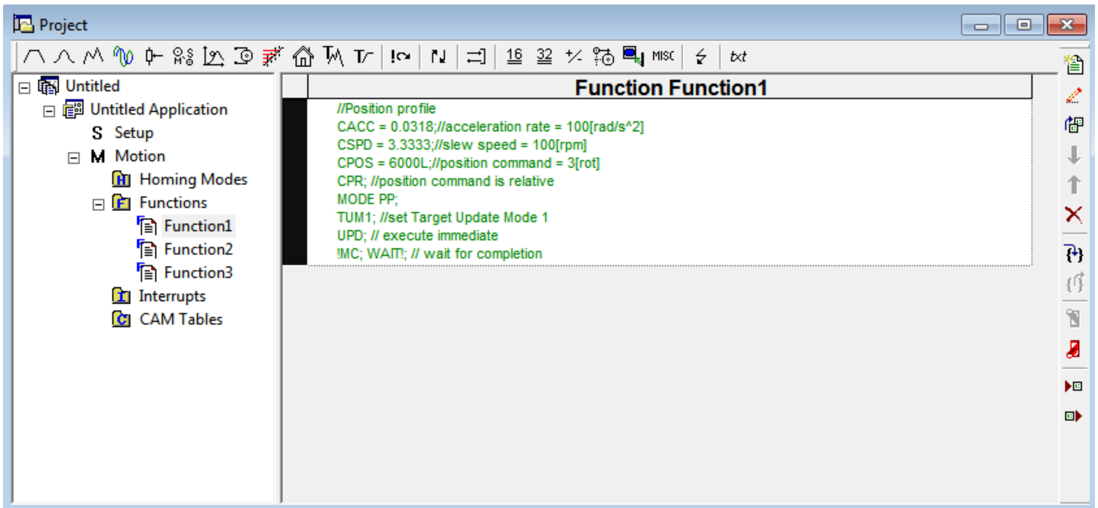


***Figure 19.2.1.** EasyMotion Studio project window – functions edit view*

22.2.2 TML Function Objects

22.2.2.1 Object 2006h: 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 (6041h) shows if a function is running. In case a function was called while another was still running, bits 7 (warning) from the Statusword (6041h) and 14 (command error) from Motion Error Register (2000h) 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 0030h (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 0020h (Data cannot be transferred or stored to the application).

The functions are initialized and available for calling, only after Controlword receives the Shutdown command (6040h = 06).

Object description:

Index	2006h
Name	Call TML function
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	WO
PDO mapping	No
Units	-
Value range	1...10
Default value	-

22.3 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.

22.3.1 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 CANopen 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	-

22.4 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.*

22.4.1 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. 2^{nd} word contains the low part, 3rd word the high part

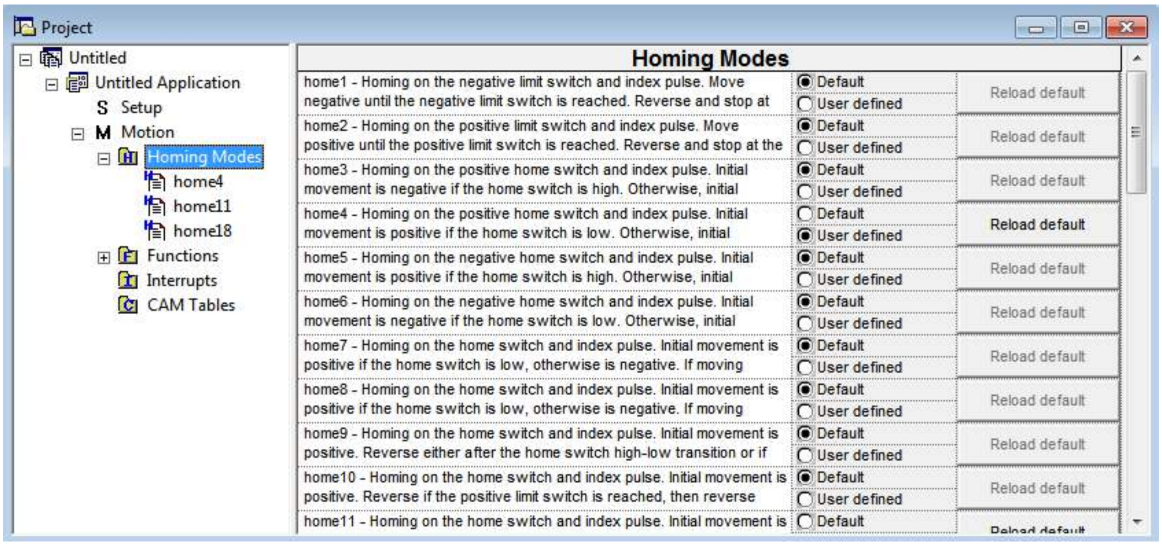
4th and 5th words: Reserved. Must be set to 0

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.

22.5 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.



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.

22.6 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.

Appendix A: Object Dictionary by Index

Index	Sub-index	Description
1000h	00h	Device type
2072h	00h	Error register
1002h	00h	Manufacturer status register
1003h		Predefined error field
	00h	Number of errors in history
	01h	Standard error field (history 1)
	02h	Standard error field (history 2)
	03h	Standard error field (history 3)
	04h	Standard error field (history 4)
	05h	Standard error field (history 5)
1005h	00h	COB-ID of the SYNC message
1006h	00h	Communication cycle period
1008h	00h	Manufacturer device name
100Ah	00h	Manufacturer software version
100Ch	00h	Guard time
100Dh	00h	Lifetime factor
1010h		Store parameters
	00h	Number of entries
	01h	Save all parameters
1011h		Restore default parameters
	00h	Number of entries
	01h	Restore all default parameters
1013h	00h	High resolution time stamp
1014h	00h	COB-ID Emergency object
1017h	00h	Producer heartbeat time
1018h		Identity Object
	00h	Number of entries
	01h	Vendor ID
	02h	Product Code
	03h	Revision Number
	04h	Serial Number
1200h		Server SDO parameter
	00h	Number of entries
	01h	COB-ID Client -> Server (rx)
	02h	COB-ID Client -> Server (tx)
1400h		Receive PDO1 communication parameters
	00h	Number of entries
	01h	COB-ID RPDO1
	02h	Transmission type
1401h		Receive PDO2 communication parameters
	00h	Number of entries
	01h	COB-ID RPDO2
	02h	Transmission type
1402h		Receive PDO3 communication parameters
	00h	Number of entries
	01h	COB-ID RPDO3
	02h	Transmission type
1403h		Receive PDO4 communication parameters
	00h	Number of entries
	01h	COB-ID RPDO4
	02h	Transmission type
1600h		RPDO1 mapping parameters
	00h	Number of entries
	01h	1st mapped object – 6040h – Controlword
1601h		RPDO2 mapping parameters
	00h	Number of entries
	01h	1st mapped object – 6040h – Controlword
	02h	2nd mapped object – 6060h – modes of operation
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

	00_h	Number of entries
	01_h	1st mapped object – 6040_h – Controlword
	02_h	2nd mapped object – $60FF_h$ – target velocity
1800_h		TPDO1 communication parameters
	00_h	Number of entries
	01_h	COB-ID TPDO1
	02_h	Transmission type
	03_h	Inhibit Time
	04_h	Reserved
	05_h	Event timer
1801_h		TPDO2 communication parameters
	00_h	Number of entries
	01_h	COB-ID TPDO2
	02_h	Transmission type
	03_h	Inhibit Time
	04_h	Reserved
	05_h	Event timer
1802_h		TPDO3 communication parameters
	00_h	Number of entries
	01_h	COB-ID TPDO3
	02_h	Transmission type
	03_h	Inhibit Time
	04_h	Reserved
	05_h	Event timer
1803_h		TPDO4 communication parameters
	00_h	Number of entries
	01_h	COB-ID TPDO4
	02_h	Transmission type
	03_h	Inhibit Time
	04_h	Reserved
	05_h	Event timer
$1A00_h$		TPDO1 mapping parameters
	00_h	Number of entries
	01_h	1st mapped object – 6041_h – Statusword
$1A01_h$		TPDO2 mapping parameters
	00_h	Number of entries
	01_h	1st mapped object – 6041_h – Statusword
	02_h	2nd mapped object – 6061_h – modes of operation display
$1A02_h$		TPDO3 mapping parameters
	00_h	Number of entries
	01_h	1st mapped object – 6041_h – Statusword
	02_h	2nd mapped object – 6064_h – position actual value
$1A03_h$		TPDO4 mapping parameters
	00_h	Number of entries
	01_h	1st mapped object – $606B_h$ – velocity demand value
	02_h	2nd mapped object – $606C_h$ – velocity actual value
2000_h	00_h	Motion Error Register
2002_h	00_h	Detailed Error Register
2003_h	00_h	Communication Error Register
2004_h	00_h	COB-ID High resolution time stamp
2005_h	00_h	Max slippage time out
2006_h	00_h	Call TML function
2009_h	00_h	Detailed Error Register 2
2010_h	00_h	Master settings
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
$201C_h$	00_h	External on-line reference
$201D_h$	00_h	External reference type
2022_h	00_h	Control effort
2023_h	00_h	Jerk time

2025h	00h	Stepper current in open loop operation
2026h	00h	Stand-by current for stepper in open loop operation
2027h	00h	Timeout for stepper stand-by current
2045h	00h	Digital outputs status
2046h	00h	Analogue input: Reference
2047h	00h	Analogue input: Feedback
2050h	00h	Over current protection level
2051h	00h	Over current time out
2052h	00h	Motor nominal current
2053h	00h	I2t protection integrator limit
2054h	00h	I2t protection scaling factor
2055h	00h	DC-link voltage
2058h	00h	Drive temperature
2060h	00h	Software version of the TML application
2064h	00h	Read/Write configuration register
2065h	00h	Write data at address set in object 2064h (16/32 bits)
2066h	00h	Read data from address set in object 2064h (16/32 bits)
2067h	00h	Write data at specified address
2069h	00h	Checksum configuration register
206Ah	00h	Checksum read register
206Bh	00h	CAM input scaling factor
206Ch	00h	CAM output scaling factor
206Fh	00h	Time notation index
2070h	00h	Time dimension index
2071h		Time factor
	00h	Number of entries
	01h	Numerator
	02h	Divisor
2072h	00h	Interpolated position mode status
2073h	00h	Interpolated position buffer length
2074h	00h	Interpolated position buffer configuration
2075h		Position triggers
	00h	Number of entries
	01h	Position trigger 1
	02h	Position trigger 2
	03h	Position trigger 3
	04h	Position trigger 4
2076h	00h	Save current configuration
2077h	00h	Execute TML program
2079h	00h	Interpolated position initial position
207Ah	00h	Interpolated position 1st order time
207Bh	00h	Homing current threshold
207Ch	00h	Homing current threshold time
207Dh	00h	Dummy
207Eh	00h	Current actual value
207Fh	00h	Current limit
2081h	00h	Set/Change the actual motor position value
2083h	00h	Encoder resolution for step loss protection
2084h	00h	Stepper resolution for step loss protection
2085h	00h	Position triggered outputs
2086h	00h	Limit speed for CSP
2087h	00h	Actual internal velocity from sensor on motor
2088h	00h	Actual internal position from sensor on motor
208Bh	00h	Sin AD signal from Sin/Cos encoder
208Ch	00h	Cos AD signal from Sin/Cos encoder
208Dh	00h	Auxiliary encoder position
208Eh	00h	Auxiliary Settings Register
208Fh		Digital inputs 8bit
	00h	Number of entries
	01h	Device profile defined inputs
	02h	Manufacturer specific inputs
2090h		Digital outputs 8bit
	00h	Number of entries
	01h	Physical outputs 8bit
	02h	Bit mask 8bit
2091h	00h	Lock EEPROM
2092h		User Variables
	00h	Number of entries

	01h	UserVar1
	02h	UserVar2
	03h	UserVar3
	04h	UserVar4
20A0h		Load Position and Speed monitoring
	00h	Number of entries
	01h	Reserved
	02h	Load Position Monitor
	03h	Load Speed Monitor
2110h	00h	Number of steps per revolution
2110h	00h	Number of microsteps per step
2102h	00h	Brake status
2103h	00h	Number of encoder counts per revolution
2104h	00h	Auxiliary encoder function
2105h	00h	Auxiliary encoder status
2106h	00h	Auxiliary encoder captured position positive edge
2107h	00h	Auxiliary encoder captured position negative edge
2108h		Filter variable 16bit
	00h	Number of entries
	01h	16 bit variable address
	02h	Filter strength
	03h	Filtered variable 16bit
210Bh	00h	Auxiliary Settings Register2
210Fh		Acceleration encoder factor
	00h	Number of entries
	01h	Acceleration internal units (IU)
	02h	Acceleration units (AU)
2110h		Jerk encoder factor
	00h	Number of entries
	01h	Jerk internal units (IU)
	02h	Jerk units (JU)
2113h		Detailed Option Code
	00h	Number of entries
	01h	Short-Circuit option code
	02h	Reserved
	03h	Control error option code
	04h	Communication error option code
	05h	Reserved
	06h	Reserved
	07h	Reserved
	08h	Over current option code
	09h	Reserved
	10h	Over temperature – Motor option code
	11h	Over temperature – Drive option code
	12h	Over voltage option code
	13h	Under voltage option code
	14h	Reserved
	15h	Enable / STO inactive option code
2114h	00h	Fault Override Option Code
2115h	00h	ASR4
6007h	00h	Abort connection option code
603Fh	00h	Error code
6040h	00h	Controlword
6041h	00h	Statusword
605Ah	00h	Quick stop option code
605Bh	00h	Shutdown option code
605Ch	00h	Shutdown option code
605Dh	00h	Disable operation option code
605Eh	00h	Fault reaction option code
6060h	00h	Modes of operation
6061h	00h	Modes of operation display
6062h	00h	Position demand value
6063h	00h	Position actual internal value
6064h	00h	Position actual value
6065h	00h	Following error window
6066h	00h	Following error time out
6067h	00h	Position window
6068h	00h	Position window time

6069$_h$	00$_h$	Velocity sensor actual value	
606B$_h$	00$_h$	Velocity demand value	
606C$_h$	00$_h$	Velocity actual value	
606D$_h$	00$_h$	Velocity window	
606E$_h$	00$_h$	Velocity window time	
606F$_h$	00$_h$	Velocity threshold	
6071$_h$	00$_h$	Target torque	
6075$_h$	00$_h$	Motor rate current	
6077$_h$	00$_h$	Torque actual value	
607A$_h$	00$_h$	Target position	
607B$_h$		Position range limit	
	00$_h$	Number of entries	
	01$_h$	Min position range limit	
	02$_h$	Max position range limit	
607C$_h$	00$_h$	Home offset	
607D$_h$		Software position limit	
	00$_h$	Number of entries	
	01$_h$	Minimum position range limit	
	02$_h$	Maximum position range limit	
607E$_h$	00$_h$	Polarity	
6080$_h$	00$_h$	Max motor speed	
6081$_h$	00$_h$	Profile velocity	
6083$_h$	00$_h$	Profile acceleration	
6085$_h$	00$_h$	Quick stop deceleration	
6086$_h$	00$_h$	Motion profile type	
6087$_h$	00$_h$	Torque slope	
6089$_h$	00$_h$	Position notation index	
608A$_h$	00$_h$	Position dimension index	
608B$_h$	00$_h$	Velocity notation index	
608C$_h$	00$_h$	Velocity dimension index	
608D$_h$	00$_h$	Acceleration notation index	
608E$_h$	00$_h$	Acceleration dimension index	
6091$_h$		Gear Ratio	
	00$_h$	Number of entries	
	01$_h$	Motor rotation	
	02$_h$	Load rotation	
6092$_h$		Feed constant	
	00$_h$	Number of entries	
	01$_h$	Feed	
	02$_h$	Shaft rotation	
6093$_h$		Position factor	
	00$_h$	Number of entries	
		Factor group – CiA 402	Factor group – CiA 402-2
	01$_h$	Numerator	Position internal units (IU)
	02$_h$	Divisor	Position units (PU)
6094$_h$		Velocity encoder factor	
	00$_h$	Number of entries	
		Factor group – CiA 402	Factor group – CiA 402-2
	01$_h$	Numerator	Velocity internal units (IU)
	02$_h$	Divisor	Velocity units (VU)
6096$_h$		Velocity factor	
	00$_h$	Number of entries	
	01$_h$	Velocity units (VU)	
	02$_h$	Position units (PU)	
6097$_h$		Acceleration factor	
	00$_h$	Number of entries	
		Factor group – CiA 402	Factor group – CiA 402-2
	01$_h$	Numerator	Acceleration units (AU)
	02$_h$	Divisor	Velocity units (VU)
6098$_h$	00$_h$	Homing method	
6099$_h$		Homing speeds	
	00$_h$	Number of entries	
	01$_h$	Speed during search for switch	
	02$_h$	Speed during search for zero	
609A$_h$	00$_h$	Homing acceleration	
60A2$_h$		Jerk factor	
	00$_h$	Number of entries	
	01$_h$	Jerk Units (JU)	

	02_h	Acceleration units (AU)
60A8h	00_h	SI unit position
60A9h	00_h	SI unit velocity
60AAh	00_h	SI unit acceleration
60ABh	00_h	SI unit jerk
60B8h	00_h	Touch probe function
60B9h	00_h	Touch probe status
60BAh	00_h	Touch probe 1 positive edge
60BBh	00_h	Touch probe 1 negative edge
60BCh	00_h	Touch probe 2 positive edge
60BDh	00_h	Touch probe 2 negative edge
60C0h	00_h	Interpolation sub mode select
60C1h		Interpolation Data Record
	00_h	Number of entries
	01_h	The first parameter
	02_h	The second parameter
60C2h		Interpolation Time Period
	00_h	Number of entries
	01_h	Interpolation time period value
	02_h	Interpolation time index
60F2h	00_h	Positioning Option Code
60F4h	00_h	Following error actual value
60F8h	00_h	Max slippage
60FCh	00_h	Position demand internal value
60FDh	00_h	Digital inputs
60FEh		Digital outputs
	00_h	Number of entries
	01_h	Physical outputs
	02_h	Bit mask
60FFh	00_h	Target velocity
6502h	00_h	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

TECHNOSOFT
MOTION TECHNOLOGY